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 моніторингу параметрів відновлення бази даних кадрів підприємства

_____

_____

Виконавець: _____ Рудюк Я.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

ЗАТВЕРЖУЮ
Завідувач кафедри

Литвиненко О. Є.

« ____ »_____2021 р.

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

Рудюка Ярослава Олександровича
(прізвище, ім’я, по батькові випускника в родовому відмінку)

1. Тема дипломної роботи (проекту): Програмний засіб моніторингу параметрів відновлення бази даних кадрів підприємства

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 кадрів, *Microsoft SQL Server*, інтегроване середовище розробки *Visual Studio*, мова *C#*, мова *SQL*.

4. Зміст пояснювальної записки:
1) Резервне копіювання та відновлення баз даних.
2) Моделі відновлення даних.
3) Проектування програмного засобу моніторингу параметрів відновлення бази даних кадрів підприємства.
4) Розробка модулів моніторингу параметрів відновлення бази даних кадрів підприємства.

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

1) Головне вікно програмного засобу “Моніторинг параметрів відновлення бази даних кадрів підприємства”.

2) Вікно перегляду резервних копій бази даних кадрів підприємства.

3) Вікно перегляду моделей відновлення.

4) Вікно конструктора запитів до системних подань.

6. Календарний план-графік

№ пор.	Завдання	Термін виконання	Відмітка про виконання
1	Ознайомитись з постановкою задачі дипломного проектування	18.05.21	
2	Вивчити спеціальну літературу і технічну документацію	14.05.21	
3	Проаналізувати системи управління базами даних та утиліти для адміністрування	16.05.21	
4	Написати розділ 1.	18.05.21	
5	Дослідити параметри системних баз даних, подання, оператори, функції, змінні мови *Transact-SQL*.	19.05.21	
6	Написати розділ 1.	21.05.21	
7	Провести проектування програмного засобу моніторингу параметрів відновлення бази даних кадрів підприємства	25.05.21	
8	Написати розділ 2.	27.05.21	
9	Провести розробку програмного засобу моніторингу параметрів відновлення бази даних кадрів підприємства	07.06.21	
10	Написати розділ 3.	12.06.21	
11	Оформити пояснювальну записку	13.06.21	
12	Підготувати графічний демонстраційний матеріал та доповідь	14.06.21	

7. Дата видачі завдання: “17” травня 2021 р.

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

Завдання прийняв до виконання _____ Рудюк Я.О.
(підпис випускника) (П.І.Б.)

РЕФЕРАТ

Пояснювальна записка до дипломного проекту «Програмний засіб моніторингу параметрів відновлення бази даних кадрів підприємства»: 64 с., 10 рис., 19 літературних джерел.

АДМІНІСТРУВАННЯ БАЗ ДАНИХ, ПАРАМЕТРИ СИСТЕМНИХ ПОДАНЬ, МОДЕЛЬ ВІДНОВЛЕННЯ, ЖУРНАЛ ТРАНЗАКЦІЙ.

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

Предмет проектування – програмний засіб моніторингу параметрів відновлення бази даних кадрів підприємства.

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

Дипломний проект присвячено тематиці моніторингу параметрів баз даних.

ЗМІСТ

ВСТУП..... 58
РОЗДІЛ 1 РЕЗЕРВНЕ КОПІЮВАННЯ ТА ВІДНОВЛЕННЯ БАЗ ДАНИХ..... 11
1.1. Основні завдання адміністрування 11
1.2. Резервне копіювання даних 12
1.3. Відновлення даних..... 16
1.4. Засоби копіювання та відновлення баз даних..... 17
РОЗДІЛ 2 МОДЕЛІ ВІДНОВЛЕННЯ ДАНИХ 21
2.1. Моделі відновлення 21
2.2. Стратегії резервного копіювання та відновлення 23
РОЗДІЛ 3 ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАСОБУ МОНІТОРИНГУ ПАРАМЕТРІВ ВІДНОВЛЕННЯ БАЗИ ДАНИХ КАДРІВ ПІДПРИЄМСТВА..... 32
3.1. Параметри резервного копіювання та відновлення 32
3.2. Системні подання відображення параметрів відновлення та копіювання..... 34
3.3. Збережені процедури та команди мови *SQL* для копіювання та відновлення..... 36
РОЗДІЛ 4 РОЗРОБКА МОДУЛІВ МОНІТОРИНГУ ПАРАМЕТРІВ ВІДНОВЛЕННЯ БАЗИ ДАНИХ КАДРІВ ПІДПРИЄМСТВА..... 43
4.1. Склад файлів проекту 43
4.2. Розробка меню..... 45
4.3. Розробка модулів моніторингу параметрів відновлення 48
ВИСНОВКИ..... 58
СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАННИХ ДЖЕРЕЛ..... 61
ДОДАТОК А. ФРАГМЕНТ ПРОГРАМНОГО КОДУ *FORM1.CS*..... 63

ВСТУП

Менеджери в інформаційному середовищі постійно міркують над тим, як поліпшити управління важливою для компанії інформацією, а також над методами вдосконалення систем управління зберіганням даних. Основна проблема, з якою вони стикаються − це постійне балансування між трьома фундаментальними факторами: зростанням даних, збереженням інформації та інвестиціями в технології, час і персонал. Тільки продумана стратегія збереження бази даних може вирішити це питання.

Багато компаній не бачать приводу для занепокоєння, оскільки вони регулярно резервують свої бази даних і повністю переконані в ефективності резервного копіювання на диск даних, що зберігаються на серверах. Оптимальне рішення полягає не тільки в зберіганні даних, але також і в максимально ефективному відновленні їх в разі необхідності. Завданням дипломної роботи є вивчення методів та засобів резервного копіювання і відновлення баз даних та розробка модулів моніторингу параметрів резервного копіювання та відновлення бази даних. Адміністрування відновлення баз даних пов’язане з процесом резервного копіювання.

Об’єкт проектування – резервне копіювання та відновлення баз даних, предмет проектування – засоби моніторингу параметрів резервного копіювання та відновлення бази даних кадрів підприємства.

Функціонування бази даних неможливе без участі фахівців, які забезпечують створення, функціонування і розвиток бази даних. Така група фахівців називається групою адміністрування бази даних. Тому в першому пункті першого розділу розглянуто завдання адміністратора зі створення та забезпечення функціонування бази даних.

Оскільки предметом проектування являються модулі моніторингу параметрів саме резервного копіювання та відновлення бази даних, то далі в першому розділі висвітлені загальні відомості про резервне копіювання даних, принципи відновлення даних та існуючі засоби копіювання та відновлення баз даних.

Резервне копіювання − це процес створення копії даних з носія, призначений для відновлення цих даних у разі їх пошкодження або видалення. Об’єктами резервного копіювання виступають дані або сукупність даних, з яких можна створити резервну копію. Резервне копіювання бази даних є процесом перенесення даних (із бази даних, протоколу транзакцій, або файлу) в окремий об’єкт або на спеціальний пристрій резервного копіювання, який створюється і підтримуються системою. Існують три методи резервного копіювання: повне *(Full)*, диференційне *(Differential)* та резервне копіювання протоколу транзакцій *(Transaction log)*. Відновлення даних – це процес відновлення доступу до даних, що зберігаються на будь-якому носії запам'ятовування.

Резервні копії можуть зберігатись на різних фізичних носіях у залежності від конкретних потреб. Зазвичай використовується сервер резервного копіювання, який координує процеси резервного копіювання та керує сховищами. Сховища можуть бути такими:

- дискові сховища;
- "хмарний" бекап.

Для виконання відновлення та резервного копіювання баз даних в *SQL Server* використовуються оператори *Transact-SQL* та середовище *SQL Server Management Studio*. При резервному копіюванні за допомогою операторів *Transact-SQL* використовуються оператори *BACKUP DATABASE* та *RESTORE*. *SQL Server Management Studio* є графічним інтерфейсом, за допомогою якого можна вводити команди *SQL*, а також можливо генерувати скрипт *Transact-SQL*. Також засобом резервного копіювання є створення *snapshot*-ів (снепшотів).

У другому розділі розглянуто моделі відновлення даних. Розрізняють три моделі: проста модель *(Simple)*, модель з повним *(Full)* та модель з частковим *(Bulk-logged)* або різницевим протоколюванням. Взагалі моделлю відновлення називають властивість бази даних, яка управляє процесом реєстрації транзакцій. Вона визначає, чи треба для журналу транзакцій резервне копіювання, а також визначає, які типи операцій відновлення доступні. Проста модель *Simple* передбачає резервне копіювання тільки бази даних, відповідно відновити стан бази можна тільки на

момент створення резервної копії, всі зміни після створення останньої резервної копії втрачаються. В моделі повного відновлення *Full* усі операції записуються в протокол транзакцій, тому ця модель надає повний захист від збоїв зовнішніх пристроїв. Ця модель дозволяє відновити базу на будь-який довільний момент часу, але вимагає, крім резервних копій бази, зберігати копії протоколу транзакцій за весь період, для якого може знадобитися відновлення. Відновлення з неповним протоколюванням *Bulk-logged* підтримує протоколи резервних копій при використанні мінімального простору в протоколі транзакцій для деяких масштабних операцій. Модель з неповним протоколюванням рекомендується тільки як доповнення до повної моделі на період великомасштабних масових операцій, коли немає необхідності відновлення бази на певний момент часу.

Важливим для забезпечення безпеки бази даних на підприємствах є існування стратегії резервного копіювання та відновлення, це питання висвітлено в третьому розділі дипломної роботи. Стратегія відновлення повинна займати належне місце, щоб запобігти випадковій втраті даних. Вона повинна включати в себе узгоджену систему резервного копіювання даних. Існує чотири основні стратегії: стратегія повного резервного копіювання бази даних, стратегія резервного копіювання бази даних і журналу транзакцій, стратегія різницевого резервного копіювання та стратегія резервного копіювання файлів і файлових груп.

Стратегія повного резервного копіювання бази даних доцільна, якщо база даних має невеликий розмір і піддається незначним змінам. Стратегію резервного копіювання бази даних і журналу транзакцій рекомендується використовувати, якщо база даних часто змінюється і/або повне резервне копіювання займає надто багато часу. Стратегія різницевого резервного копіювання також доцільна для великої бази даних, щоб скоротити час на резервне копіювання і можлива тоді, коли резервне копіювання журналів транзакцій виконується окремо. Стратегія різницевого резервного копіювання включає в себе створення регулярних повних резервних копій бази даних з проміжними різницевими резервними копіями. Між повними і різницевими резервними копіюваннями можна також додатково виконувати резервні копіювання журналу транзакцій. Стратегія резервного

копіювання файлів і файлових груп використовується, якщо до усіх попередніх умов додаються можливі складності з керуванням. Ця стратегія включає в себе резервне копіювання окремих файлів і файлових груп, що виконується на регулярній основі.

У третьому та четвертому розділів відображено проектування програмного засобу та розробка модулів моніторингу параметрів відновлення баз даних кадрів підприємства. Проектування системного програмного засобу відбувалось з використанням системних подань *sys.sysaltfiles* і *sys.sysdevices*, збережених процедур *sp_addumpdevice* і *sp_dropdevice* та мови *SQL*. Системне подання *sys.sysaltfiles* використано для відображення інформації про усі файли бази даних кадрів підприємства, а системне подання *sys.sysdevices* − для розробки модулів моніторингу. Системна процедура *sp_addumpdevice* додає пристрій резервного копіювання в системне подання *sys.backup_devices*. Після чого пристрій можна вказувати в інструкціях *BACKUP* і *RESTORE* по логічному імені. Процедура *sp_dropdevice* видаляє пристрій бази даних або пристрій резервного копіювання з екземпляра компонента *SQL Server Database Engine*, видаляючи запис з *master.dbo.sysdevices*. Код команд *SQL* для резервного копіювання і відновлення бази даних кадрів підприємства реалізовано з модулів, розроблених в *Visual Studio* 2014 на мові *C#.*

Одне з основних завдань адміністрування баз даних − резервне копіювання та відновлення баз даних. Ефективне адміністрування потребує засобів моніторингу параметрів та виконання копіювання та відновлення. Один з важливих параметрів − модель відновлення, в якій працює база даних, тому тема дипломного проекту “Програмний засіб моніторингу параметрів відновлення бази даних кадрів підприємства” є актуальним завданням.

В залежності від потреб можливе використання декількох засобів для виконання резервування та відновлення. Основне завдання адміністрування баз даних – резервне копіювання та відновлення виконується згідно узгодженої стратегії і будується на виборі моделі та параметрів відновлення.

В результаті створено модулі: модуль резервного копіювання, модуль відновлення бази даних, модуль перегляду резервних копій, модуль перегляду моделі відновлення та додаткові модулі створення бази даних і видалення.

РОЗДІЛ 1
РЕЗЕРВНЕ КОПІЮВАННЯ ТА ВІДНОВЛЕННЯ БАЗ ДАНИХ

1.1. Основні завдання адміністрування

Робота бази даних (БД) забезпечується фахівцями зі створення та супроводу бази даних. Така група фахівців називається адміністраторами бази даних (АБД). Адміністратори бази даних виконують наступні дії зі створення та забезпечення функціонування БД протягом усіх етапів життєвого циклу системи [1]:

1. Проектування структури бази даних: визначення складу і структури інформаційних одиниць, що складають базу даних; завдання зв'язків між ними; вибір методів впорядкування даних і методів доступу до інформації; опис структури БД мовою обробки даних.

2. Забезпечення відновлення БД: розробка програмно-технологічних засобів відновлення БД, організація ведення системних журналів.

3. Захист даних від втрати − резервування. Використовується як при фізичному пошкодженні файлу, так і у випадку, якщо в БД внесені небажані зміни.

4. Завдання обмежень цілісності при описі структури бази даних і процедур обробки БД: завдання обмежень цілісності, властивих предметній області; визначення обмежень цілісності, викликаних структурою бази даних; розробка процедур забезпечення цілісності БД при введенні і коригування даних.

Аналіз ефективності функціонування бази даних і розвиток системи: аналіз показників функціонування системи (час обробки, обсяг пам'яті),

реорганізація та реструктуризація баз даних, зміна складу баз даних, розвиток програмних і технічних засобів.

1.2. Резервне копіювання даних

Резервне копіювання − процес створення копії даних з носія, який призначений для відновлення цих даних у разі їх пошкодження або видалення. Створення резервної копії даних надає можливість виконати відновлення інформації при втраті оригіналу, з якого було створено резервну копію. При цьому під втратою треба розуміти настання події, що призвела до зміни даних, після чого вони втратили цінність або були видалені з носія. Адміністрування відновлення БД пов’язане з процесом резервного копіювання.

Об'єкти резервного копіювання − це дані або сукупність даних, з яких можна створити резервну копію [2]. Приклади об'єктів: файли бази даних, дані прикладних програм, дані операційної системи чи сама операційна система, образи віртуальних машин та дисків віртуальних машин, файлові системи тощо.

Резервне копіювання бази даних є процесом перенесення даних (із бази даних, протоколу транзакцій, або файлу) на пристрій резервного копіювання, який створюється і підтримуються системою. Резервне копіювання може бути статичним та динамічним. Статична резервна копія свідчить про те, що в процесі копіювання є тільки одна активна сесія, що підтримується системою, вона і створює резервну копію. При динамічному типі резервного копіювання бази даних копіювання може виконуватись без зупинки серверу бази даних, видалення користувачів або закриття файлів. Існують наступні варіанти резервного копіювання [9]:

- повне резервне копіювання *(Full)*;
- диференційне резервне копіювання *(Differential)*;
- резервне копіювання журналу (протоколу) транзакцій *(Transaction log)*.

Повне резервне копіювання *(Full)* − повна копія даних. Повне резервне копіювання БД охоплює той стан бази даних, який вона має на початку копіювання.

В процесі повного копіювання бази даних система копіює дані, а також схеми усіх таблиць бази даних та відповідні файлові структури. Якщо повне копіювання БД виконується динамічно, то система бази даних записує будь-які дії, які мають місце в процесі виконання резервного копіювання. Тому усі непідтверджені транзакції будуть записані на пристрій резервного копіювання. Цей тип копіювання дозволяє забезпечити максимальну відповідність оригіналу даних його копії. При створенні резервної копії бази даних вперше обов'язковий саме цей метод копіювання.

Диференційне резервне копіювання *(Differential)* − копіювання змін, що були зроблені після створення останньої повної копії. Створює лише копії частин бази даних, які змінились з моменту останнього повного копіювання бази даних. Як і при повному копіюванні даних, будь-які дії, що відбуваються в процесі диференційного резервного копіювання, також копіюються.

Перевагою диференційного резервного копіювання є швидкість. Цей тип резервного копіювання мінімізує час, необхідний для копіювання, оскільки кількість копійованих даних значно менша, ніж у випадку повного резервного копіювання.

Резервне копіювання протоколу транзакцій *(Transaction log)* враховує лише зміни, що внесені до протоколу. Оскільки об'єм даних невеликий, цей процес може бути виконаний значно швидше, аніж повне або диференційне резервне копіювання.

Існує декілька причин, через які варто виконувати резервне копіювання протоколу транзакцій, серед них: перша – збереження даних, які були змінені з моменту останнього копіювання протоколу транзакцій, або бази даних на захищений пристрій; друга − правильне закриття протоколу транзакцій перед початком виконання нових дій з цим протоколом; третя – усічення розмірів журналу.

Використовуючи результати повного резервного копіювання бази даних та усіх протоколів транзакцій, можна переносити копії бази даних на інші комп'ютери. Потім ця копія бази даних може бути використана для заміщення

оригінальної бази у випадку збою. Такий сценарій може бути використаний і при використанні повної копії, і при використанні останньої копії диференційованої.

Цілісність резервної копій даних − це відповідність даних резервної копії та оригіналу на момент створення копії. Якщо було вибрано неправильний варіант створення резервної копії, неможливо буде відновити дані у разі потреби. Для забезпечення цілісності резервних копій використовують різні методи, в залежності від об'єкта резервного копіювання. Вікно адміністрування системних баз даних за допомогою утиліти *SQL Server Management Studio* зображено на рис.1.1.

Резервні копії можуть зберігатись на різних фізичних носіях у залежності від конкретних потреб. Зазвичай використовується сервер резервного копіювання, який координує процеси резервного копіювання та керує сховищами. Сховища можуть бути такими [5]:

- стрічкова бібліотека або стример;
- дискові сховища;
- *Virtual Tape Library* − віртуальна стрічкова бібліотека;
- "хмарний" бекап.

При виконанні резервного копіювання у стрічкову бібліотеку або стример запис резервних копій відбувається на магнітну стрічку стримеру. Перевагами цього методу є: швидкість запису та читання; відносно мала вартість зберігання одиниці інформації; можливість збереження *off-line* (тобто стрічки можна виймати та зберігати в сейфі). Недоліки: необхідність створення штучної надмірності *(redundancy)*; повільний доступ до файлів; обмежена здатність компресії; обмежена кількість одночасних операцій, оскільки обмежена кількість стримерів та роботів (*picker*, що вставляють та виймають стрічки).

Дискові сховища − запис резервних копій відбувається на диски, що можуть бути об'єднані в *RAID* (*Redundant Array of Independent Disks* − надлишковий масив незалежних дисків) або на дискову систему збереження. Переваги цього методу: швидкість доступу до інформації; необмежена кількість одночасних операцій; можливість використання технологій усунення дублювання, що значно зменшує

потреби в просторі (в залежності від типу даних від 1,5 і більше разів) та також прискорює запис даних, оскільки дублікати не записуються а відкидаються. Недоліками є: менша (у порівнянні з стрічками) швидкість запису та читання інформації; відносно більша вартість 1 ГБ.

Virtual Tape Library − запис резервних копій на диски, але саме сховище виглядає для клієнта як стрічкова бібліотека. Перевагами цього методу є: можливість використання технології усунення дублювання, значно збільшена кількість одночасних операцій, велика швидкість доступу до даних.

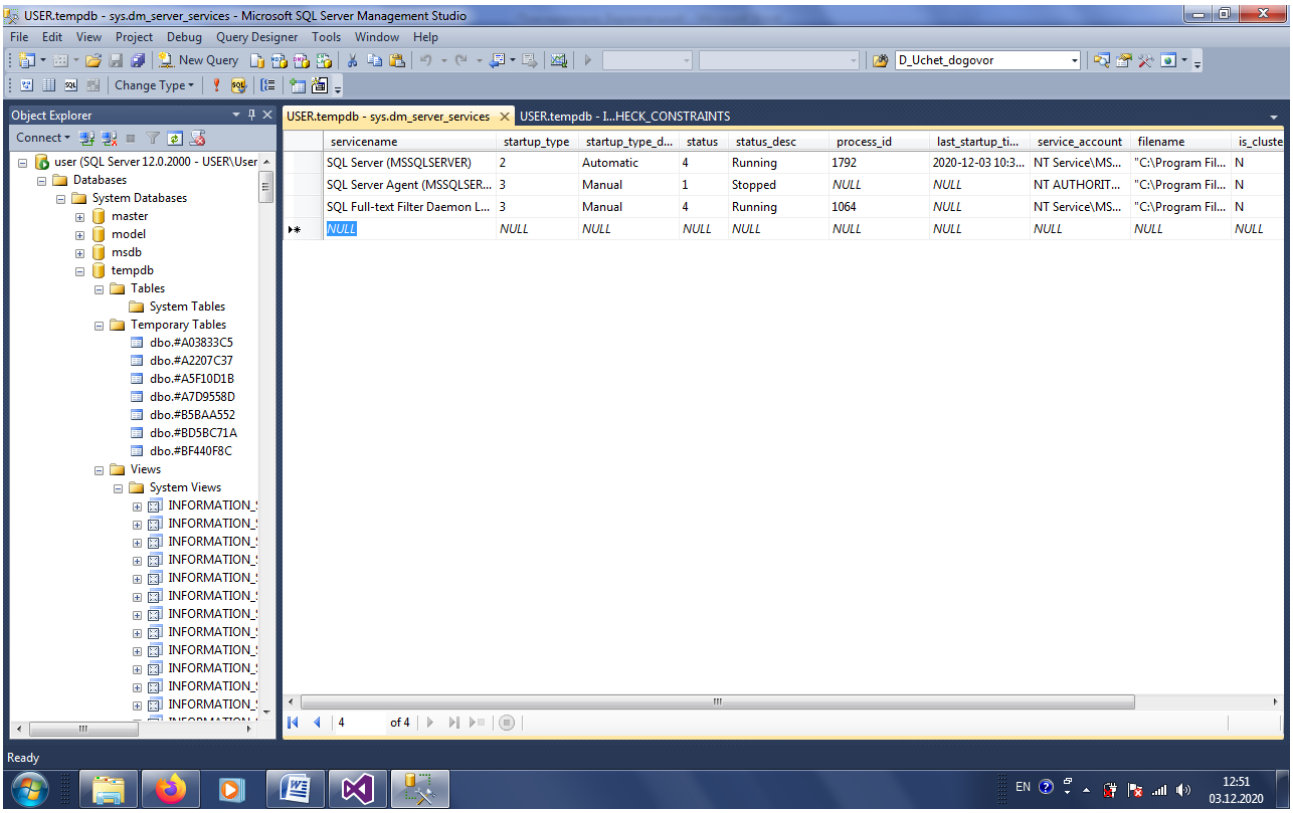

Рис. 1.1. Адміністрування системних баз даних за допомогою утиліти *SQL Server Management Studio*

"Хмарний" бекап − запис резервних даних за «хмарною» технологією через онлайн-служби спеціальних провайдерів. Сервер резервного копіювання також може використовувати як сховище "хмару". Перевагами цього методу є: низька вартість обслуговування, швидке розгортання та масштабування, можливість

використання *Pay-as-you-Go* (оплата по мірі отримання). Недоліки: занадто низька швидкість запису та читання, недостатня безпека доступу до даних.

1.3. Відновлення даних

Відновлення даних − процес відновлення доступу до даних, що зберігаються на будь-якому носії пристрою запам'ятовування [12]. Існує три моделі відновлення: проста модель відновлення *(Simple)*, модель повного відновлення *(Full)* та модель відновлення з неповним протоколюванням *(Bulk-logged).* Відновлення даних включає:

- відновлення пошкоджених чи видалених баз даних;
- реконструкцію пошкоджених файлових систем після пошкодження або форматування.

Система керування базами даних *MS SQL Server* працює в режимі реального часу і це означає, що під час резервного копіювання усі користувачі можуть отримувати доступ до бази даних. Це можливо завдяки тому, що *MS SQL Server* використовує журнали транзакцій. *SQL Server* створює в журналі транзакцій контрольні точки при копіюванні зафіксованих транзакцій з журналу транзакцій в базу даних (копіювання результатів транзакцій виконується автоматично, приблизно через 5 хвилин).

Всі процедури відновлення залежать від того факту, що кожен запис журналу позначений реєстраційним номером транзакції в журналі *(LSN − Log Sequence Numbers)*. Реєстраційний номер транзакції в журналі є зростаючим номером, який однозначно визначає положення запису в журналі транзакцій. Всі записи журналу в транзакції зберігаються в послідовному порядку і містять код транзакції і *LSN* попереднього запису транзакції. Іншими словами, кожна операція, зареєстрована в якості частини транзакції, має зворотне “посилання” на операцію, що безпосередньо їй передує [4].

У простому випадку повернення однієї транзакції механізм відновлення може швидко пройти по ланцюжку записаних в журнал операцій, від самої

останньої до першої операції, і повернути результати всіх операцій в зворотному від їх виконання порядку. Для використання будь-якого типу резервування даних необхідно мати місце для зберігання резервних копій. Носій, який використовується для зберігання резервної копії, називається пристроєм резервного копіювання.

1.4. Засоби копіювання та відновлення баз даних

Для виконання резервного копіювання БД в *SQL Server* можна використати наступні засоби: оператори мови *Transact-SQL* та середовище *SQL Server Management Studio*. Для виконання резервного копіювання та відновлення БД використовують оператори *BACKUP* та *RESTORE* мови *Transact-SQL* відповідно.

При резервному копіюванні усі типи операцій виконуються з використанням операторів *Transact-SQL: BACKUP DATABASE* та *BACKUP LOG* [3]. Оператор *BACKUP DATABASE* використовується для повного копіювання бази даних або диференційованого резервного копіювання. Цей оператор має наступний синтаксис:

BACKUP DATABASE {db_name | @variable}
TO device_list
[MIRROR TO device_list2]
[WITH | option_list]

В цьому записі *db_name* є ім'ям бази даних, для якої повинно бути виконано резервне копіювання. Ім'я бази даних також може бути задано змінною *@variable*. Параметр *device_list* задає одне або більше імен пристроїв, де буде зберігатися копія баз даних. Параметр *device_list* може бути списком імен дискових файлів або магнітних стрічок.

Параметр *option_list* містить кілька опцій, які можуть бути задані для різних типів резервних копій. Найбільш важливими опціями є наступні: *DIFFERENTIAL; NOSKIP/SKIP; NOINIT/INIT; NOFORMAT/FORMAT; UNLOAD/NO UNLOAD; MADIANAME, BLOCKIZE* та *COMPRESSION*. Опція, *DIFFERENTIAL*, задає

диференційоване резервне копіювання. Всі інші пропозиції в цьому списку відносяться до повного копіювання бази даних.

Оператор *BACKUP LOG* застосовується для створення резервної копії протоколу транзакцій. Цей оператор має наступний синтаксис:

BACKUP LOG {db_name | @variable}

TO device_list

[MIRROR TO device_list2]

[WITH | option_list]

В цьому записі *db_name, @variable, device_list* та *device_list2* мають ті ж самі значення, що й параметри з тими ж іменами в операторі *BACKUP DATABASE*. Параметр *option_list* має ті ж опції, що і в операторі *BACKUP DATABASE*, крім того, він підтримує специфічні опції протоколу *NO_TRUNCATE, NORECOVERY, STANDBY*.

Необхідно використовувати опцію *NO_TRUNCATE* для копіювання протоколу без його обрізання, тобто ця опція не очищує підтверджені транзакції в протоколі. Після виконання цієї опції система записує всі останні дії з базою даних в протокол транзакцій. Тому опція *NO_TRUNCATE* дозволяє вам відновлювати дані прямо до тієї точки, коли стався збій бази даних.

Опція *NORECOVERY* копіює залишок протоколу і залишає базу даних в стані відновлення. Значення *NORECOVERY* буде корисним, коли збій відбувається у вторинній базі даних (базі даних-отримувачі) і при збереженні залишку протоколу перед операцією відновлення. Опція *STANDBY* копіює залишок протоколу і залишає базу даних в режимі тільки для читання і в стані *STANDBY*.

Всі операції відновлення БД можуть виконуватись за допомогою двох операторів *Transact-SQL: RESTORE DATABASE* та *RESTORE LOG*. Оператор *RESTORE DATABASE* використовується для виконання процесу відновлення бази даних. Загальний синтаксис цього оператора:

RESTORE DATABASE {db_name | @variable} [FROM device_list]

[WITH option_list]

Де *db_name* − ім'я бази даних, яка буде відновлюватися. Ім'я бази даних може задаватися і з використанням змінної *@variable*. Параметр *device_list* задає одне або більше імен пристроїв, на яких розташовується резервна копія бази даних. Якщо пропозиція *FROM* не задана, то матиме місце тільки процес автоматичного відновлення, а не процес відновлення з резервної копії, і в цьому випадку потрібно задати одну з опцій *RECOVERY, NORECOVERY* або *STANDBY.* Опція *RECOVERY* інструктує *Database Engine* про те, що необхідно пропускати будь-які підтверджені транзакції і скасовувати непідтверджені. Після застосування опції *RECOVERY* база даних буде знаходитися в узгодженому стані та буде готовою для використання.

Оператор *RESTORE LOG* використовується для виконання процесів відновлення протоколу транзакцій. Цей оператор має ту саму синтаксичну форму і ті ж самі опції, як і *RESTORE DATABASE*.

SQL Server Management Studio − це графічний інтерфейс, за допомогою якого можна вводити команди *Transact-SQL*. Також засобом резервного копіювання є план обслуговування бази даних *(database maintenance plan)* [10]. За допомогою плану обслуговування можна не тільки створити завдання на резервне копіювання бази даних, але і запланувати його для виконання за розкладом, налаштувати генерацію звіту про резервне копіювання (навіть його пересилку по електронній пошті). Інструмент командного рядка операційної системи *sqlmaint,* що використовується для цілей забезпечення зворотної сумісності, являється найпростішим способом проведення резервного копіювання баз даних *SQL Server* за розкладом з одночасним створенням звіту.

В залежності від потреб можливе використання декількох засобів для виконання резервування та відновлення. Основне завдання адміністрування баз даних – резервне копіювання та відновлення виконується згідно узгодженої стратегії і будується на виборі моделі та параметрів відновлення, тому тема «Програмний засіб моніторингу параметрів відновлення бази даних кадрів підприємства» є актуальною.

1.5. Висновки до розділу

Функціонування бази даних неможливе без участі фахівців, які забезпечують створення, функціонування і розвиток бази даних − адміністраторів бази даних. Вони виконують роботи зі створення та забезпечення функціонування бази даних. Важливими роботами адміністратора бази даних є резервне копіювання та відновлення бази даних. Існують три варіанти резервного копіювання: повне (*Full*, яке дозволяє забезпечити максимальну відповідність оригіналу даних його копії), диференційне (*Differential*, створює лише копії частин бази даних, які змінились з моменту останнього повного копіювання бази даних) та резервне копіювання протоколу транзакцій (*Transaction log*, що враховує лише зміни, що внесені до протоколу). При створенні резервної копії бази даних вперше обов'язково використовується метод копіювання *Full*.

Відновлення даних – це процес відновлення доступу до даних, що зберігаються на будь-якому носії пристрої запам'ятовування. *SQL Server* (система керування базами даних) створює в журналі транзакцій контрольні точки при копіюванні зафіксованих транзакцій з журналу транзакцій в базу даних. Так механізм відновлення може швидко пройти по ланцюжку записаних в журнал операцій, від самої останньої до першої операції, і повернути результати всіх операцій в зворотному від їх виконання порядку.

Для виконання відновлення та резервного копіювання бази даних в *SQL Server* використовуються оператори *Transact-SQL (BACKUP DATABASE* та *RESTORE). SQL Server Management Studio* є графічним інтерфейсом, за допомогою якого можна вводити команди *SQL*. Також засобом резервного копіювання є план обслуговування бази даних *(database maintenance plan)* та утиліта *SQLmaint*.

РОЗДІЛ 2
МОДЕЛІ ВІДНОВЛЕННЯ ДАНИХ

Операції резервного копіювання і відновлення *SQL Server* виконуються в контексті моделі відновлення бази даних. Моделлю відновлення називають властивість бази даних, яка управляє процесом реєстрації транзакцій. Моделі відновлення призначені для управління обслуговуванням журналів транзакцій. Модель відновлення − це властивість бази даних, яка управляє процесом реєстрації транзакцій, визначає, чи треба для журналу транзакцій резервне копіювання, а також визначає, які типи операцій відновлення доступні. Існує три моделі відновлення [11]: проста модель відновлення *(Simple)*, модель повного відновлення *(Full)* та модель відновлення з неповним протоколюванням *(Bulk-logged)*. Зазвичай в базі даних використовується модель повного відновлення або проста модель відновлення. Модель відновлення задається при створенні бази даних. База даних в будь-який момент може бути переключена на іншу модель відновлення.

2.1. Моделі відновлення

Проста модель *(Simple)* передбачає резервне копіювання тільки бази даних, відповідно відновити стан БД можна тільки на момент створення резервної копії, всі зміни після створення останньої резервної копії будуть втрачені. За цієї моделі відновлення зруйнованої бази даних можливе лише використання повної або диференційної резервної копії бази даних, оскільки вони не потребують резервної копії протоколу. Стратегія резервного копіювання для цієї моделі наступна: виконання відновлення бази даних з

існуючої резервної копії бази даних і, якщо існують диференційні резервні копії, застосування останньої її версії [12].

Перевагою цієї моделі є висока продуктивність усіх об’ємних операцій та невисокі вимоги до об’єму пам’яті протоколу. Журнал автоматично очищується, тобто усікається. В моделі *Simple* можливе резервне копіювання тільки повне та різницеве. Але така модель потребує більше ручної роботи, оскільки усі зміни з моменту останнього резервного копіювання БД мають бути відновлені. В цій моделі відновлення не допускаються відновлення на певний проміжок часу або відновлення конкретних сторінок.

В моделі повного відновлення *(Full)* усі операції записуються в протокол транзакцій, тому ця модель надає повний захист від збоїв зовнішніх пристроїв. Це надає можливість відновлення БД з останньої підтвердженої транзакції, яка була збережена в файлі протоколу [11]. Повна модель дозволяє відновити базу на будь-який довільний момент часу, але вимагає, крім резервних копій бази, зберігати копії протоколу транзакцій за весь період, для якого може знадобитися відновлення. При активній роботі з базою розмір протоколу транзакцій, а, отже, і розмір архівів, можуть досягати великих розмірів. Процес відновлення також набагато більш складний і тривалий за часом. При виборі моделі відновлення варто порівняти витрати на відновлення з витратами на зберігання резервних копій. В моделі *Full* можливе резервне копіювання повне, різницеве та журналу транзакцій.

Недоліком цієї моделі є те, що відповідний протокол транзакцій зростає і може бути великим за обсягом. Для такого об’ємного протоколу витрачається значно більше часу для резервного копіювання. В цьому режимі журнал не усікається.

Для баз з невеликим обсягом додавання інформації може бути вигідніше використовувати просту модель з великою частотою копій, яка дозволить швидко відновитися і продовжити роботу, ввівши втрачені дані вручну. Повна модель в першу чергу повинна використовуватися там, де втрата даних недопустима, а їх можливе відновлення пов'язане зі значними витратами.

Відновлення з неповним протоколюванням *(Bulk-logged)* підтримує протоколи резервних копій при використанні мінімального простору в протоколі транзакцій для деяких масштабних операцій. Протоколювання наступних операцій мінімальне і не може керуватись принципом «операція за операцією». Хоч об'ємні операції повністю не протоколюються, повне резервне копіювання БД після завершення масових операцій виконувати немає необхідності. Під час відновлення з неповним протоколюванням резервні копії протоколу транзакцій містять і протокол, і результат об'ємної операції. Це спрощує перехід між повною моделлю та моделлю відновлення з неповним протоколюванням [11].

Якщо протокол був пошкоджений, або з моменту створення останньої резервної копії виконувалися операції з неповним протоколюванням, усі зміни після цього резервного копіювання потрібно буде вводити повторно. Якщо ні, результати роботи втрачені не будуть. Автоматично журнал не перезаписується, тобто не усікається.

Модель з неповним протоколюванням рекомендується тільки як доповнення до повної моделі на період великомасштабних масових операцій, коли немає необхідності відновлення бази на певний момент часу. Вона дозволяє виконувати високопродуктивні операції масового копіювання, при цьому зменшує місце, займане журналами, за рахунок неповного протоколювання більшості масових операцій. В моделі *Bulk-logged* можливе повне резервне копіювання, різницеве та журналу транзакцій.

2.2. **Стратегії резервного копіювання та відновлення**

Стратегія відновлення повинна займати належне місце, щоб запобігти випадковій втраті даних. Стратегія резервного копіювання та відновлення повинна включати в себе узгоджену систему резервного копіювання даних. Нижче описані основні можливості, які повинні бути реалізовані для забезпечення точного відновлення [12]:

- резервне копіювання даних в обсязі, достатньому для повернення бази даних до узгодженого стану, незалежно від її поточного стану.
- реєстрація транзакцій та змін бази даних.
- відновлення бази даних з достатньою кількістю можливостей для її приведення у вихідний робочий стан, з мінімальними втратами даних і часу.

При розробці стратегії резервного копіювання насамперед треба продумати методи захисту від наступних проблем: відмова жорсткого диска, випадкове видалення файлів, пошкодження вмісту файлів, повне знищення комп'ютера (наприклад, при пожежі), при якому загинуть також резервні копії, які фізично знаходяться поруч.

Найбільш поширеною технологією резервного копіювання є архівація всієї системи з копіюванням на надійний зовнішній носій і розміщення його незалежно від основної системи.

Операції резервного копіювання і відновлення виконуються в контексті моделей відновлення. Модель відновлення є властивістю бази даних, яка регулює управління журналом транзакцій [11]. Також модель відновлення бази даних визначає, які типи резервних копій і сценарії відновлення підтримуються для бази даних. Зазвичай база даних використовує просту модель відновлення або модель повного відновлення. Модель повного відновлення може бути змінена шляхом перемикання на модель з неповним протоколюванням перед виконанням масових операцій вставки.

Кращий вибір моделі відновлення бази даних залежить від бізнес-вимог. Щоб уникнути управління журналом транзакцій і спростити резервне копіювання і відновлення, рекомендується використовувати просту модель відновлення. Щоб знизити ймовірність втрати результатів роботи ціною збільшення адміністративних витрат, потрібно використовувати модель повного відновлення. Після того як обрана модель відновлення, відповідна бізнес-вимогам певної бази даних, необхідно спланувати і виконати відповідну стратегію резервного копіювання. Оптимальна стратегія залежить від багатьох факторів, серед яких найбільш важливі наступні [9]:

1) Скільки годин на день додатки мають доступ до бази даних: якщо існує прогнозований позапіковий період, рекомендується запланувати повне резервне копіювання бази даних саме на цей період.

2) Наскільки часті і вірогідні зміни і оновлення;

Стратегія повного резервного копіювання бази даних передбачає наступне. Повне резервне копіювання виконується, якщо:

− база даних має невеликий розмір;

− база даних піддається незначним змінам або доступна тільки для читання. Слід періодично очищувати журнал транзакцій, якщо використовується повна модель відновлення (рис. 2.1).

Рис. 2.1. Повне резервне копіювання бази даних

Стратегія повного резервного копіювання бази даних − це метод відновлення, що включає в себе створення регулярних повних резервних копій бази даних. Якщо база даних пошкоджена, можна скористатися самою останньою повною резервною копією, щоб відновити базу даних до стану, в якому вона перебувала на момент створення резервної копії. Час і ресурси, необхідні для реалізації стратегії повного резервного копіювання бази даних, визначаються розміром бази даних і частотою зміни даних.

Якщо застосовується тільки стратегія повного резервного копіювання бази даних, і база даних налаштована для використання повної моделі відновлення або моделі відновлення з неповним протоколюванням, журнал транзакцій буде в кінцевому підсумку повністю заповнений. Коли журнал транзакцій повністю

заповниться, дії в базі даних можуть блокуватися сервером *SQL Server* до тих пір, поки журнал транзакцій буде очищений.

Щоб виключити виникнення цієї проблеми, можна виконати наступні дії:

- Встановити просту модель відновлення бази даних.
- Періодично очищати журнал транзакцій за допомогою параметра *NO_LOG* або *TRUNCATE ONLY* інструкції *BACKUP LOG*.

Параметри *NO_LOG* і *TRUNCATE_ONLY* надаються для забезпечення зворотної сумісності і будуть видалені в майбутній версії *SQL Server* [10]. Якщо не передбачається створювати резервні копії журналу транзакцій, слід встановити просту модель відновлення.

Стратегія резервного копіювання бази даних і журналу транзакцій передбачає наступне. Слід об'єднати резервне копіювання бази даних і журналу транзакцій (рис.2.2), якщо:

- база даних часто змінюється;
- повне резервне копіювання займає надто багато часу.

Рис. 2.2. Резервне копіювання бази даних та журналу

Коли для дотримання вимоги відновлюваності даних недоцільно виконувати тільки повні резервні копіювання бази даних, слід створювати проміжні резервні копії журналу транзакцій, щоб вести запис всіх дій в базі даних, які відбуваються між повного резервного копіювання бази даних. Цей підхід відомий як стратегія резервного копіювання бази даних і журналу транзакцій.

При реалізації стратегії резервного копіювання бази даних і журналу транзакцій можна відновити базу даних з самої останньої повної резервної копії бази даних, а потім застосувати всі резервні копії журналу транзакцій, які були створені з моменту останнього повного резервного копіювання.

Використовувати стратегію резервного копіювання бази даних і журналу транзакцій слід для часто змінюваних баз даних. Слід також проаналізувати, чи можна виконати резервне копіювання бази даних і журналів транзакцій за прийнятний час.

Стратегія різницевого резервного копіювання передбачає наступне. Різницеве резервне копіювання слід використовувати, якщо:

- база даних часто змінюється;
- необхідно скоротити час резервного копіювання;
- резервне копіювання журналів транзакцій виконується окремо.

Стратегія різницевого резервного копіювання включає в себе створення регулярних повних резервних копій бази даних з проміжними різницевими резервними копіями. Між повними і різницевими резервного копіювання можна також додатково виконувати резервні копіювання журналу транзакцій(рис.2.3).

Щоб відновити базу даних у випадку аварії, необхідно відновити саму останню повну резервну копію бази даних, після цього саму останню різницеву резервну копію і потім у порядку черговості відновити кожен журнал транзакцій з моменту створення останньої різницевої резервної копії.

Рис.2.3. Стратегія різницевого резервного копіювання

Застосовувати стратегію різницевого резервного копіювання слід для зменшення часу відновлення, якщо база даних пошкоджена. Наприклад, замість застосування декількох великих копій журналів транзакцій можна використовувати різницеву резервну копію, щоб застосувати зміни, які були внесені до бази даних з моменту створення останньої повної резервної копії бази даних. Використовувати стратегію резервного копіювання файлів і файлових груп слід для дуже великої бази даних, яка секціонована на багато файлів. При об'єднанні з регулярним резервним копіювання журналів транзакцій цей метод представляє вражаючу за часом альтернативу повного резервного копіювання бази даних. Наприклад, якщо в розпорядженні є тільки 1:00 для виконання повного резервного копіювання бази даних (яке зазвичай займає 4:00), можна було б виконувати щоночі резервне копіювання окремих файлів і забезпечувати цілісність даних(рис.2.4).

SQL Server підтримує наступні носії: стрічка та диск [13]. Перш ніж виконувати резервне копіювання бази даних в *SQL Server*, необхідно розглянути, який тип носія буде використовуватися для зберігання резервних копій. До кожного типу носія можна отримати доступ за допомогою спеціального шляху, або з фіксованого пристрою резервного копіювання.

Резервне копіювання може виконуватися сервером *SQL Server* у файл на жорсткому диску або на стрічку. Дискові файли (локальні або мережеві) є найбільш поширеними носіями, використовуваними для зберігання резервних копій. Коли виконується резервне копіювання на стрічку, накопичувач на магнітній стрічці повинен бути локально під'єднаний до *SQL Server*.

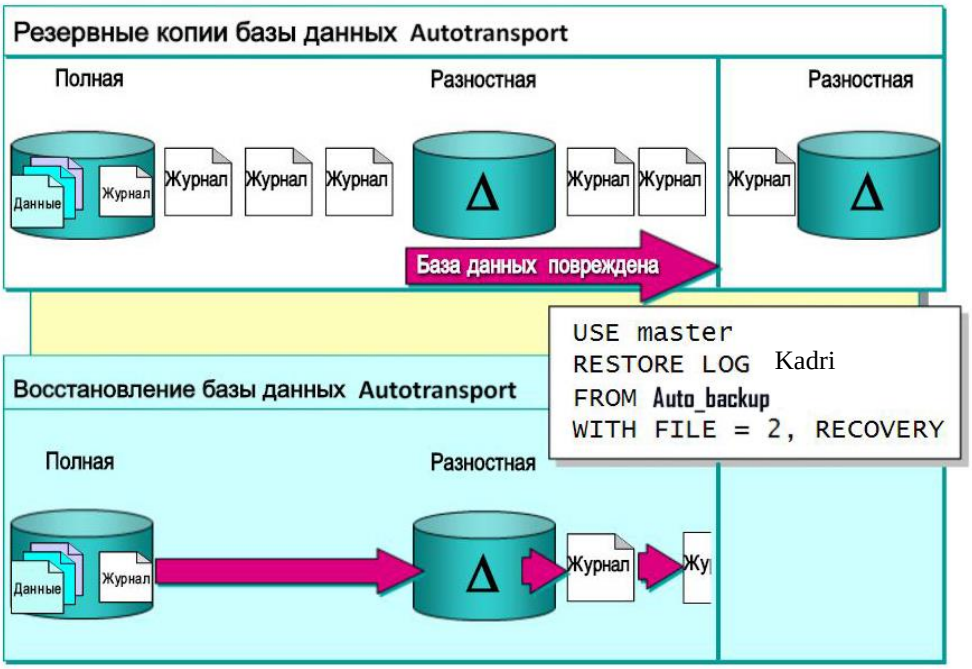

Рис. 2.4. Відновлення бази даних кадрів підприємства

Перший крок резервного копіювання полягає у створенні файлів резервних копій, які будуть містити архів. Файл резервної копії, створюваний до того, як він буде використовуватися для операції резервного копіювання, називається пристроєм резервного копіювання. Пристрої резервного копіювання можна створювати за допомогою *SQL Server Management Studio* або шляхом виконання системної збереженої процедури *sp_addumpdevice*.

2.3. Висновки до розділу

Моделлю відновлення називають властивість бази даних, яка управляє процесом реєстрації транзакцій. Вона визначає, чи треба для журналу транзакцій резервне копіювання, а також визначає, які типи операцій відновлення доступні. Розрізняють три моделі: проста модель *(Simple)*, модель з повним *(Full)* та модель з частковим *(Bulk-logged)* протоколюванням. Проста модель передбачає резервне копіювання тільки бази даних, відповідно відновити стан бази можна тільки на момент створення резервної копії, всі зміни після створення останньої

резервної копії втрачаються, журнал автоматично усікається. В моделі повного відновлення усі операції записуються в протокол транзакцій, тому ця модель надає повний захист від збоїв зовнішніх пристроїв. Ця модель дозволяє відновити базу на будь-який довільний момент часу, але вимагає, крім резервних копій бази, зберігати копії протоколу транзакцій за весь період, для якого може знадобитися відновлення, журнал не усікається. Відновлення з неповним протоколюванням підтримує протоколи резервних копій при використанні мінімального простору в протоколі транзакцій для деяких масштабних операцій. Модель з неповним протоколюванням рекомендується тільки як доповнення до повної моделі на період великомасштабних масових операцій, коли немає необхідності відновлення бази на певний момент часу, журнал також не перезаписується. Операції резервного копіювання і відновлення *SQL Server* виконуються в контексті моделі відновлення бази даних.

Стратегія відновлення повинна займати належне місце, щоб запобігти випадковій втраті даних. Вона повинна включати в себе узгоджену систему резервного копіювання даних. Існує чотири основні стратегії: стратегія повного резервного копіювання бази даних, стратегія резервного копіювання бази даних і журналу транзакцій, стратегія різницевого резервного копіювання та стратегія резервного копіювання файлів і файлових груп.

Стратегія повного резервного копіювання бази даних доцільна, якщо база даних має невеликий розмір і піддається незначним змінам. Стратегію резервного копіювання бази даних і журналу транзакцій рекомендується використовувати, якщо база даних часто змінюється і/або повне резервне копіювання займає надто багато часу. Стратегія різницевого резервного копіювання також доцільна для великої бази даних, щоб скоротити час на резервне копіювання і можлива тоді, коли резервне копіювання журналів транзакцій виконується окремо. Стратегія різницевого резервного копіювання включає в себе створення регулярних повних резервних копій бази даних з проміжними різницевими резервними копіями. Між повними і різницевими резервними копіюваннями можна також додатково виконувати резервне

копіювання журналу транзакцій. Стратегію резервного копіювання файлів і файлових груп слід застосовувати для дуже великої бази даних, яка секціонована на декілька файлів.

РОЗДІЛ 3
ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАСОБУ МОНІТОРИНГУ ПАРАМЕТРІВ ВІДНОВЛЕННЯ БАЗИ ДАНИХ КАДРІВ ПІДПРИЄМСТВА

3.1. Параметри резервного копіювання та відновлення

Для проектування системного програмного засобу моніторингу параметрів резервного копіювання та відновлення баз даних кадрів підприємства було використано середовище *Management Studio* та команди мови *SQL*. Вікно резервного копіювання в *Management Studio* можна відкрити з контекстного меню *Tasks, Backup* для бази даних кадрів підприємства. Також можна скористатися контекстним меню для контейнера *Server Objects*, *Backup Devices*. Основні параметри резервного копіювання наступні: *Database*, *Recovery mode*, *Backup type*, *Backup component*, *Backup set name* та ін. [15].

Database − ім'я бази даних, резервне копіювання якої буде проводитись. У команді *BACKUP* в нашому випадку вказується як *BACKUP DATABASE Kadri;*

Recovery mode (режим відновлення) − довідкова інформація про поточний режим відновлення бази даних. Режим відновлення змінюється відповідно до властивостей бази даних;

Backup type − тип резервного копіювання. Для повного або часткового резервного копіювання використовується команда *BACKUP DATABASE* (для часткового ще вказується параметр *DIFFERENTIAL*), для резервного копіювання журналу транзакцій − команда *BACKUP LOG*;

Backup component (компонент для резервного копіювання) – компонент, який дозволяє вибрати резервне копіювання всієї бази даних або окремих файлових груп (окремих файлів). У команді *BACKUP* для вказівки файлів або файлових груп використовуються ключові слова *FILE* і *FILEGROUP*.

Destination (призначення) – місце призначення резервної копії у вигляді файлу на диску, стримерів або логічного пристрою резервного копіювання. Можна вказати декілька призначень одночасно (але тільки одного типу). У цьому випадку запис резервної копії буде виконуватись паралельно в декілька файлів або на кілька стримерів для підвищення продуктивності. У команді *BACKUP* для вказівки файлу на диску використовується ключове слово *DISK*, а для вказівки стримерів − *TAPE.*

Overwrite media (перезаписати носій) − це параметри (на графічному екрані вони розташовані на вкладці *Options*), що дозволяють визначити режим перезапису носія (файлу на диску або магнітної стрічки). Можливі наступні варіанти:

- *Append to the existing media set* (Додати до існуючого набору носія). Відповідає параметрам *NOFORMAT* (не перезаписувати заголовок носія) і *NOINIT* (не перезаписувати стару резервну копію);
- *Overwrite all existing backup sets* (Перезаписати всі існуючі набори носія). Відповідає параметрам *NOFORMAT* і *INIT* (заголовок носія збережеться, але всі старі резервні копії будуть перезаписані);
- *Check media set name and backup set expiration* (Перевірити ім'я набору носіїв і старіння резервної копії). Відповідає параметрам *NOFORMAT, INIT, NOSKIP* (перезапис буде проведений тільки в тому випадку, якщо ім'я резервної копії збігається з існуючою на носії, але існуюча резервна копія застаріла).

У середовищі *Management Studio* існують наступні основні параметри відновлення баз даних: *Restore with recovery, Restore with norecovery* та *Restore with standby*.

Restore with recovery − відновлює базу даних після відновлення останньої резервної копії, зазначеної в групі параметрів «Резервні набори даних для відновлення». Цей параметр застосовується за замовчуванням і рівнозначний пропозиції *WITH RECOVERY* в інструкції *RESTORE*.

Restore with norecovery – залишає базу даних в стані відновлення, що дозволяє встановити додаткові резервні копії в поточному шляху відновлення. Щоб відновити базу даних, необхідно виконати операцію відновлення з параметром *RESTORE WITH RECOVERY*.

Restore with standby − залишає базу даних в стані, в якому база даних доступна в обмеженому режимі тільки для читання.

3.2. Системні подання відображення параметрів відновлення та копіювання

Для проектування системного програмного засобу моніторингу параметрів відновлення баз даних кадрів підприємства було використано системні подання *sys.sysaltfiles, sys.sysdevices, sys.databases, sys.restorehistory* та *sys.restorefile* [13]. Інформація про відновлення та резервне копіювання міститься в системних поданнях та таблицях баз даних *Master* та *MSdb*.

Системне подання *sys.sysaltfiles* використовується для відображення інформації про файли бази даних кадрів підприємства. Подання містить наступі елементи:

file id − ідентифікаційний номер файлу, він унікальний для кожної бази даних;

groupid − ідентифікаційний номер файлової групи;

size − розмір файлу, в сторінках по 8 кілобайт (КБ);

maxsize − максимальний розмір файлу, в сторінках по 8 КБ, де елемент 0 означає, що файл не збільшується; елемент -1 означає, що розмір файлу може збільшуватися до повного заповнення диска; елемент 268435456 означає, що файл журналу може збільшуватися до 2 ТБ.

growth − граничний розмір бази даних; в залежності від значення стану може являти собою або відсоток від розміру файлу, або число сторінок.

dbid − ідентифікаційний номер бази даних, якій належить даний файл;

name − логічне ім'я файлу (у дипломній роботі *Kadri*);

filename − ім'я фізичного пристрою, включаючи повний шлях до файлу (у дипломній роботі *C:\ProgramFiles\MicrosoftSQLServer\MSSQL12.MSSQL SERVER\MSSQL\DATA\Kadri.mdf).*

Для розробки модулів моніторингу параметрів відновлення кадрів підприємства використовувалось системне подання *sys.sysdevices.* У цьому поданні використовуються основні параметри:

*name (*тип *sysname)* − логічне ім'я файлу резервної копії або бази даних;

size (int) − розмір файлу в сторінках по 2 КБ;

status (smallint) − бітові карти, які мають відповідні значення:

тип 1 − диск за умовчанням; тип 2 − фізичний диск; тип 4 − логічний диск;

phyname (nvarchar) − ім'я фізичної файлу.

Системне подання *sys.databases* містить один рядок для кожної бази даних в екземплярі SQL Server. У цьому поданні використовуються такі основні параметри:

*Name (*тип *sysname)* − ім'я бази даних, унікальне всередині примірника SQL Server або на сервері баз даних SQL Azure.

database_id (int) − ідентифікатор бази даних, унікальний всередині примірника SQL Server або на сервері баз даних *SQL Azure*.

*collation_name (*тип *sysname)* − схема сортування.

is_read_only (тип *bit*) – якщо дорівнює 1, то база даних знаходиться в режимі тільки читання *READ_ONLY;* 0 – база даних знаходиться в режимі читання/запису *READ_WRITE*.

is_cleanly_shutdown (тип *bit*) **–** якщо дорівнює 1, то база даних закрита вірно; відновлення при запуску не потрібно; 0 – база даних закрита невірно; потрібне відновлення при запуску.

Для перегляду параметрів спроектовано наступний запит з подання *sys.databases:*

select name,database_id,recovery_model, recovery_model_desc, collation_name, is_read_only, is_cleanly_shutdown from sys.databases

Системне подання *sys.restorehistory* містить один рядок для кожного відновленого файлу, включаючи файли, які відновлені по імені файлової групи. Ця таблиця знаходиться в *MSdb* базі даних і містить наступі елементи:

restore_history_id − унікальний ідентифікаційний номер, який ідентифікує кожну операцію відновлення;

restore_date − дата і час завершення операції відновлення;

destination_database_name − ім'я бази даних призначення для операції відновлення;

backup_set_id − унікальний ідентифікаційний номер, що ідентифікує резервний набір відновлення.

Системне подання *sys.restorefile* містить один рядок для кожного відновленого файлу. Ця таблиця знаходиться в базі даних *MSdb* і містить наступі елементи:

restore_history_id − унікальний ідентифікаційний номер, який відповідає операції відновлення;

file_number − ідентифікаційний номер відновлюваного файлу, цей номер унікальний в межах кожної бази даних;

destination_phys_drive − диск або розділ, до якого файл був відновлений.

3.3. Збережені процедури та команди мови *SQL* для копіювання та відновлення

Для проектування системного програмного засобу моніторингу параметрів відновлення баз даних кадрів підприємства було використано збережені процедури *sp_addumpdevice* та *sp_dropdevice*. Процедура *sp_addumpdevice* додає пристрій резервного копіювання в подання *sys.backup_devices*. Після цього

пристрій можна вказувати в інструкціях *BACKUP і RESTORE* по логічному імені. Збережена процедура *sp_addumpdevice* не дає доступ до фізичного пристрою. Звернення до нього проводиться тільки при виконанні інструкцій *BACKUP* і *RESTORE* [14]. Створення логічного пристрою резервного копіювання спрощує інструкції *BACKUP* і *RESTORE*, дозволяючи замість шляху пристрою в параметрах *TAPE* і *DISK* вказувати імена пристроїв.

Для видалення пристрою використано процедуру *sp_dropdevice*. Процедура *sp_dropdevice* видаляє пристрій бази даних або пристрій резервного копіювання з екземпляра компонента *SQL Server Database Engine*, видаляючи запис з подання *sys.sysdevices*.

sp_dropdevice [@logicalname =] 'device'

[, [@delfile =] 'delfile']

[@ Logicalname =] 'device' − логічне ім'я пристрою бази даних і пристрою резервного копіювання, наведене в стовпці *master.dbo.sysdevices.name*.

[@Delfile =] 'delfile' − вказує, чи потрібно видаляти файл з фізичного пристрою резервного копіювання. Аргумент *delfile* має тип *varchar*. Якщо задано значення *DELFILE,* фізичний пристрій резервного копіювання видаляється. Код звернення має два значення: 0 (успішне завершення) та 1 (не успішне завершення).

В розробці програмного модулю для дипломної роботи використовувались команди мови *SQL: BACKUP* та *RESTORE*. Команда *BACKUP* створює резервну копію всієї бази даних *SQL Server* або файлів або файлових груп бази даних. Крім того, при використанні моделі повного відновлення або моделі відновлення з неповним протоколюванням створюється резервна копія журналу транзакцій за допомогою команди *BACKUP LOG*.

BACKUP DATABASE { database_name | @database_name_var }

TO <backup_device> [,...n]

[<MIRROR TO clause>] [next-mirror-to]

[WITH { DIFFERENTIAL | <general_WITH_options> [,...n] }]

[;]

BACKUP DATABASE { database_name | @database_name_var }

READ_WRITE_FILEGROUPS [, <read_only_filegroup> [,...n]]

TO <backup_device> [,...n]

[<MIRROR TO clause>] [next-mirror-to]

[WITH { DIFFERENTIAL | <general_WITH_options> [,...n] }]

[;]

BACKUP LOG { database_name | @database_name_var }

TO <backup_device> [,...n]

[<MIRROR TO clause>] [next-mirror-to]

[WITH { <general_WITH_options> | <log-specific_optionspec> } [,...n]]

Параметр *DATABASE* вказує, що повинна бути створена резервна копія всієї бази даних. Якщо вказаний список файлів і файлових груп, то тільки вони включаються в резервну копію. Під час відновлення резервної копії, створеної за допомогою інструкції *BACKUP DATABASE* (резервної копії даних), відновлюється вся резервна копія. Поновлення на певний момент часу або до певної транзакції можливо тільки для резервної копії журналів.

LOG вказує тільки на резервне копіювання журналу транзакцій. Створюється резервна копія частини журналу, що починається з кінця останньої успішно створеної копії і закінчується поточним кінцем журналу. До створення першої резервної копії журналу необхідно створити повну резервну копію бази даних. Резервну копію журналів можна відновити на певний момент часу або до певної транзакції, вказавши пропозицію *WITH STOPAT, STOPATMARK* або *STOPBEFOREMARK* в інструкції *RESTORE LOG*.

{Databasename | database_name_var} − база даних, журнал транзакцій і частина даних або всі дані, які піддаються резервному копіюванню. Якщо це ім'я надається в якості змінної *(database_name_var)*, воно може бути зазначено у вигляді строкової константи *(@database_name_var = database name)* або змінної з типом даних символьного рядка, за винятком типів даних *ntext* і *text*.

FILE {logical_file_name | logical_file_name_var} − логічне ім'я файлу або змінна, значення якої дорівнює логічному імені файлу, який слід включити в резервну копію.

FILEGROUP {logical_filegroup_name | logical_filegroup_name_var} − логічне ім'я файлової групи або змінна зі значенням, рівним логічному імені файлової групи, яку слід включити в резервну копію. У простій моделі відновлення створення резервної копії файлової групи дозволено лише для файлових груп, доступних тільки для читання.

Команда *RESTORE* відновлює резервні копії, виконані за допомогою команди *BACKUP*. Ця команда дозволяє виконати наступні сценарії відновлення: відновити базу даних повної резервної копії (повне відновлення); відновити частину бази даних (часткове відновлення); відновити в базі даних певні файли або файлові групи (відновлення файлів); відновити в базі даних певні сторінки (відновлення сторінок); відновити в базі даних журнал транзакцій (відновлення журналу транзакцій); повернути базу даних до моменту часу, на який був виконаний моментальний знімок бази даних.

Схема модулю моніторингу параметрів відновлення бази даних кадрів підприємства відображено на рис. 3.1.

SQL Server підтримує різні сценарії відновлення: повне відновлення бази даних, відновлення файлів, відновлення сторінки, поетапне відновлення та відновлення журналу транзакцій. Повне відновлення бази даних, відновлює всю базу даних, починаючи з повної резервної копії, за якою може слідувати відновлення з проміжної резервної копії бази даних. При відновленні файлів відновлюється файл або файлова група в базі даних. Після повного відновлення файлів може бути відновлена різницева резервна копія файлів. При відновленні сторінки проводиться відновлення окремих сторінок. Відновлення сторінки можливо тільки в моделях повного відновлення і відновлення з неповним протоколюванням. Поетапне відновлення передбачає відновлення бази даних по етапах, починаючи з первинної файлової групи і однієї або декількох

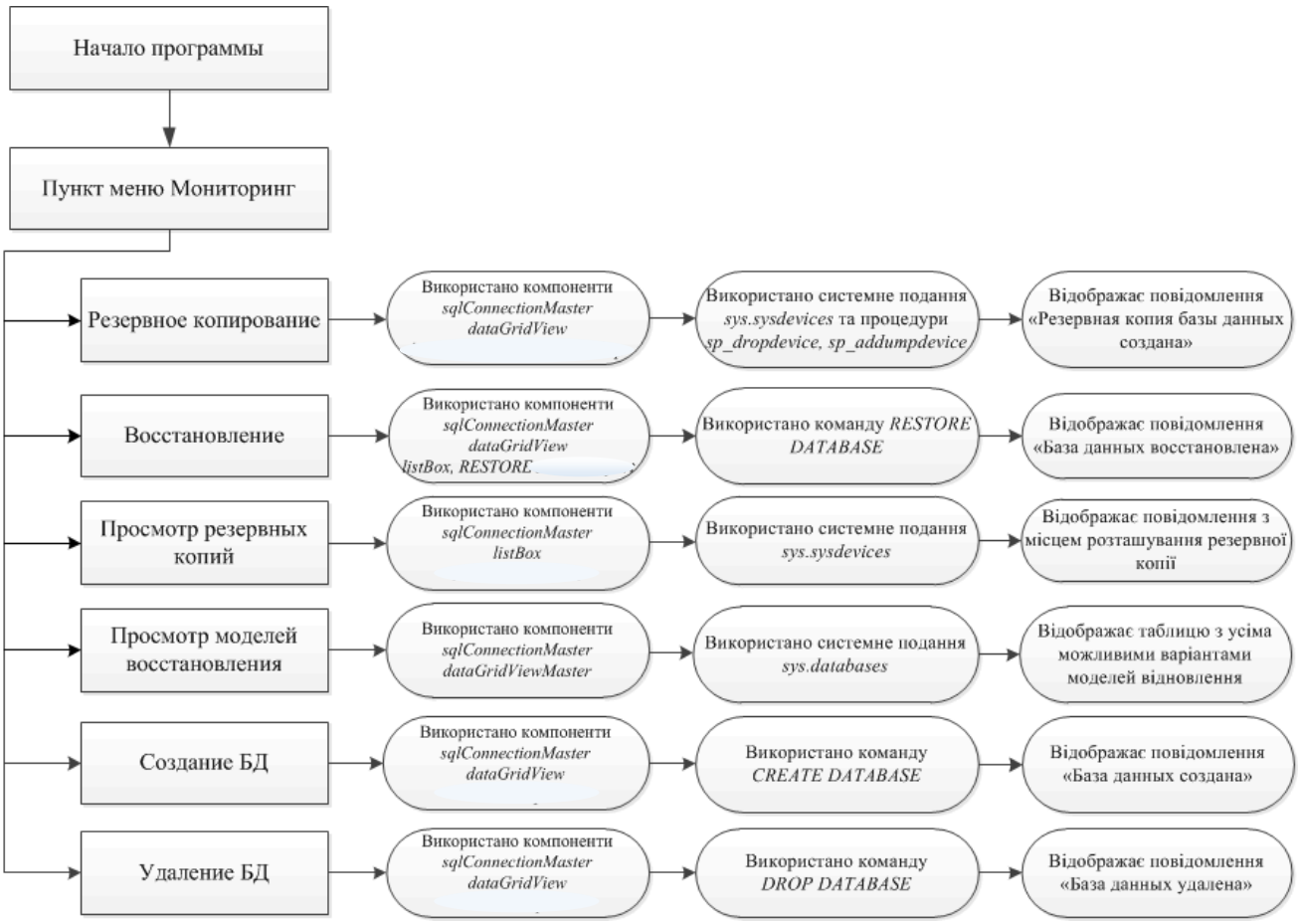

Рис. 3.1. Схема функціональна модулю моніторингу параметрів відновлення бази даних кадрів підприємства

вторинних файлових груп. Поетапне відновлення починається з інструкції *RESTORE DATABASE* з параметром *PARTIAL* і вказівки однієї або декількох відновлюваних вторинних файлових груп. Відновлення журналу транзакцій у моделях повного відновлення і відновлення з неповним протоколюванням відновлення резервних копій журналів необхідно для досягнення необхідної точки відновлення. Якщо підтримується оперативне відновлення і база даних доступна в мережі, автоматично виконується оперативне відновлення файлів і сторінок. Крім того, після початкового етапу поетапного відновлення виконується відновлення вторинної файлової групи.

Код команд *SQL* для резервного копіювання і відновлення бази даних кадрів підприємства реалізовано з модулів, розроблених в *Visual Studio* на мові *C#*. Вікно конструктора запитів до системних подань в середовищі розробки *Visual Studio* відображено на рис. 3.2.

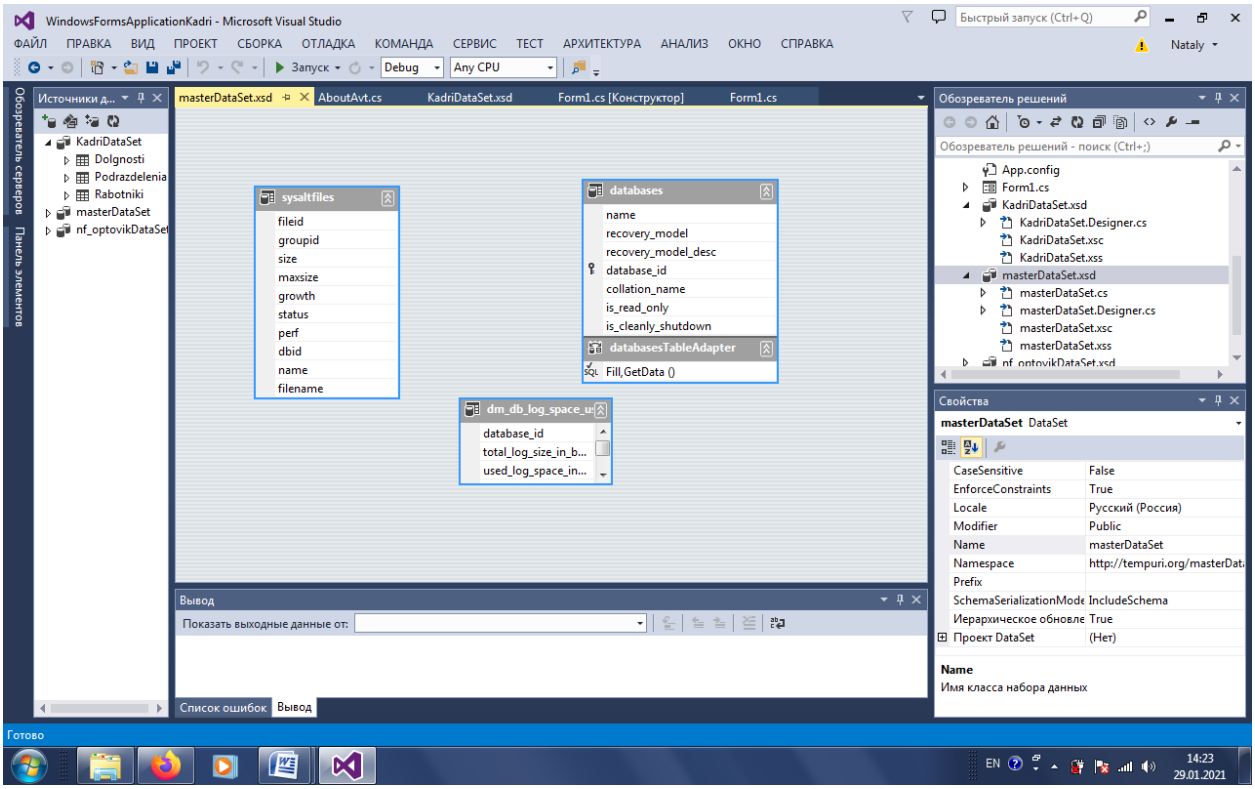

Рис. 3.2. Вікно конструктора запитів до системних подань

3.4. Висновки до розділу

Для проектування системного програмного засобу моніторингу параметрів резервного копіювання та відновлення баз даних кадрів підприємства було використано основні параметри резервного копіювання наступні: *Database*, *Recovery mode*, *Backup type*, *Backup component*, *Backup set name*. Для резервного копіювання використовується команда *BACKUP DATABASE*, для відновлення – *RESTORE DATABASE*.

Проектування системного програмного засобу відбувалось у середовищі *Management Studio* з використанням системних подань *sys.sysaltfiles* і *sys.sysdevices*, збережених процедур *sp_addumpdevice* і *sp_dropdevice* та мови

SQL. Системне подання *sys.sysaltfiles* використано для відображення інформації про усі файли бази даних автотранспортного підприємства, а системне подання *sys.sysdevices* − для розробки модулів моніторингу. Системна процедура *sp_addumpdevice* додає пристрій резервного копіювання в системне подання *sys.backup_devices*. Після чого пристрій можна вказувати в інструкціях *BACKUP* і *RESTORE* по логічному імені. Процедура *sp_dropdevice* видаляє пристрій бази даних або пристрій резервного копіювання з екземпляра компонента *SQL Server Database Engine*, видаляючи запис з *master.dbo.sysdevices*. Код команд *SQL* для резервного копіювання і відновлення бази даних кадрів підприємства реалізовано з модулів, розроблених в *Visual Studio* 2014 на мові *C#.*

РОЗДІЛ 4
РОЗРОБКА МОДУЛІВ МОНІТОРИНГУ ПАРАМЕТРІВ ВІДНОВЛЕННЯ БАЗИ ДАНИХ КАДРІВ ПІДПРИЄМСТВА

4.1. Склад файлів проекту

При розробці модулів моніторингу параметрів відновлення баз даних кадрів підприємства у середовищі *Visual Studio* використовуються два типи файлів *(SLN і SUO)* для зберігання параметрів рішень [5]. Файл *Kadri_Solution.sln* організовує проекти, елементи проектів та елементи рішень в рішення. Файл *SUO* записує всі параметри, пов'язані з рішенням так, що кожен раз при відкриванні рішення включаються налаштування користувача.

При використанні шаблонів *Visual Studio* для створення проектів створюється ряд файлів, склад яких залежить від шаблону, який використовується. Типи файлів, які згенеровано в проект *Kadri*, залежать від типу проекту та параметрів, вибраних при роботі:

- файли проекту і рішення *(Project and Solution Files*);
- файли, які створюються для проектів *CLR (Files Created for CLR Projects);*
- файли прекомпільованих заголовків *(Precompiled Header Files)*;
- файли ресурсів *(Resource Files)*;
- файли довідки *(WinHelp);*
- різні файли проекту *(Miscellaneous Project Files).*

До складу проекту відновлення бази даних кадрів підприємства входять наступні файли:

Файл *Properties-AssemblyInfo.cs* надає інформацію про додаток, таку як

номер версії, описи та ін.

Файл *Resources.resx-Resources.Designer.cs* поміщає елементи, додані в проект за допомогою конструктора ресурсів, в папку ресурсів проекту. Відомості конструктора зберігаються у файлі з ім'ям *Resources.resx*, а код ресурсу зберігається у файлі *Resources.Designer.cs*.

Файл *Settings.Designer.cs* містить вихідний код, який записує конструктор форм при вмкористанні елементів управління на формі, налаштуванні властивостей у вікні Властивості і так далі.

Файл *App.config* містить параметри, які стосуються окремих додатків. В файлах конфігурації присутні загальні елементи, хоча ім'я і розташування будь-якого файлу конфігурації в значній мірі залежать від того, де розміщується додаток.

Файл *KadriDataSet.xsd* має набір даних *DataSet* знаходиться в пам'яті і є представленням даних, що надає узгоджену реляційну модель, незалежну від джерела даних. Набір даних *DataSet* являє собою повну сукупність даних, яка включає таблиці, обмеження і зв'язки між таблицями. Набір даних *DataSet* є незалежним від джерела даних, тому *DataSet* може включати дані, локальні по відношенню до додатка, а також дані з декількох джерел даних. Управління взаємодією з існуючими джерелами даних здійснюється за допомогою *DataAdapter*.

Файли *Form1.cs, Form1.Designer.cs* та *Form1.resx* генеруються при створенні проекту *Windows Forms Visual C#* і створюють форму *Form1*. Представляють форму два файли, що мають назву *Form1.cs і Form1.designer.cs*. Код модулів моніторингу написано у файлі *Form1.cs*; у файл *designer.cs file* конструктор *Windows Forms* записує код, що реалізовує всі дії, виконані шляхом перетягування елементів управління з Панелі елементів.

Файл *Program.cs*. містить точку входу для програми. Окремий файл *CS* може містити будь-яке число визначень класів і структур. Щоб додати в проект нові або існуючі файли або класи, в меню Проект потрібно вибрати команду Додати новий елемент або Додати існуючий елемент.

4.2. Розробка меню

При створенні програми в середовищі *Visual Studio* першим кроком був вибір типу (шаблону) проекту. Для кожного типу проекту *Visual Studio* задає параметри компілятора і автоматично створює стартовий код [16]. У проекті *Kadri* створюється база даних кадрів підприємства.

У меню *File* послідовно обрано пункти *New i Project*. У діалоговому вікні *New Project* на лівій панелі розкрито вузол *Installed*, вузол *Templates* вузол *Visual C#*, у списку встановлених шаблонів обрано далі *Windows Forms Application*. В полі *Name* введено ім'я проекту *WindowsFormsApplicationKadri*. В полі *Location* указано шлях до файлів рішення *D:\RUDUK\Kadri*. За замовчуванням ім'я рішення збігається з ім'ям проекту *WindowsFormsApplicationKadri*. Для того, щоб розрізнити імена файлів рішення і проекту (в доповнення розширення імені файлів за замовчуванням) в полі *Solution Name* введено ім'я файлів рішення *Auto_Solution* та *Auto_Project* підтверджено дії *OK*.

Для встановлення властивостей компонента *Form* було обрано конструктор *Windows Forms*, що відображає *Form1* створеного проекту. Властивості цього компонента з'явилися на панелі *Properties*. В групі *Appearance* встановлено властивість *Text* «Мониторинг параметров восстановления базы данных кадров предприятия». Наступним етапом додано компонент *Panel*. Знайдено компонент *Panel* в списку *All Windows Forms* панелі елементів *Toolbox* та перетягнуто компонент за допомогою миші на форму приблизно в те місце, де потрібно розмістити компонент. В групі *Layout* компонента *Panel* встановлено властивість *Dock* такою, щоб дорівнювала *Bottom*. В групі *Layout* розкрито властивість *Size* та встановлено властивість *Height* компонента *Panel* такою, щоб дорівнювала 28.

В групі *Design* встановлено властивість *Name* компонента *Panel* такою, щоб дорівнювала *Panel*1. В групі *Appearance* встановлено властивість *BorderStyle* компонента *Panel* такою, щоб дорівнювала *FixedSingle*.

Для створення меню до форми додано компонент *menuStrip* із списку *All Windows* панелі елементів *Toolbox*, обрано компонент *menuStrip*, в групі *Data* розкрито властивість *Items*. У вікні *Items Collection Editor* в полі вводу *Select items and add to list below* обрано *MenuItems* обрано *Add*. У списку властивостей в групі *Appearance* встановлено значення властивості *Text* для першого елемента горизонтального меню: Работники. В групі *Design* встановлено властивість *Name* компонента *toolStripMenuItem*1 такою, щоб дорівнювала *toolStripMenuItemRabotniki*. Зазначені вище дії повторені для додавання усіх необхідних елементів горизонтального та вертикального меню.

Значення властивості *Name* встановлено відповідно наступні:

- «Работники»;
- «Подразделения»;
- «Должности»;
- «Мониторинг»;
- «Об авторе»;
- «Выход».

Пункт меню «Мониторинг» призначений для адміністратора кадрів підприємства, який матиме змогу робити резервне копіювання та відновлення бази даних з метою захисту бази даних від втрати.

Для додавання вертикальних пунктів меню після вибору горизонтального пункту *toolStripMenuItemMonit* у вікні *Properties* обрано *DropDownItems*. У вікні *Items Collection Editor* в полі вводу *Select items and add to list below* обрано *MenuItems*, обрано *Add*. У списку властивостей в групі *Appearance* встановлено значення властивості *Text* для першого елемента вертикального меню: «Резервное копирование». Встановлено властивість *Name* компонента *toolStripMenuItem1* такою, щоб дорівнювала *toolStripMenuItemRezCopy*. Аналогічно додано усі вертикальні пункти меню.

Пункт меню «Мониторинг» складається з вертикальних пунктів меню (рис. 4.1):

- «Резервное копирование»;

– «Восстановление БД»;
– «Просмотр резервных копий»;
– «Просмотр моделей восстановления»;
– «Создание БД»;
– «Удаление БД».

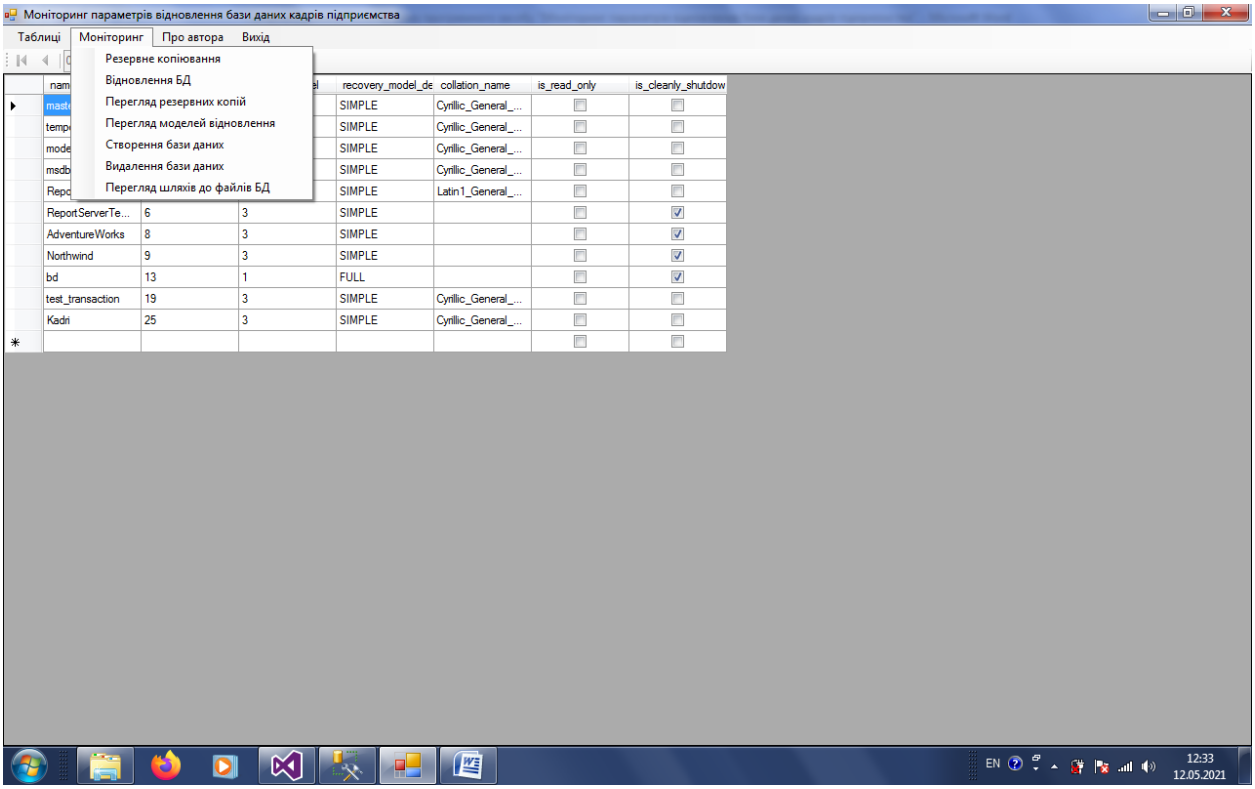

Рис. 4.1. Головне вікно програмного засобу “Моніторинг параметрів відновлення бази даних кадрів підприємства”

Підпункт «Резервное копирование» розроблено для регулярного копіювання бази даних адміністратором, підпункт «Восстановление БД» використовується лише в разі втрати бази (в такому разі буде відновлена остання збережена резервна копія).

Усі існуючі резервні копії можна знайти за допомогою підпункту «Просмотр резервных копий». Для створення та видалення бази даних призначені додаткові підпункти меню «Создание БД» та відповідно «Удаление БД».

Для створення інформаційної компоненти *LabelInf* було виконано наступне. Для додання компонента *Label* в списку *All Windows* панелі елементів *Toolbox* знайдено компонент *Label* та перетягнуто компонент за допомогою миші

на компоненті *Panel*1. В групі *Design* встановлено властивість *Name* компонента *Label* такою, щоб дорівнювала *LabelInf*. В групі *Layout* розкрито властивість *Location* та встановлено властивість *X* компонента *Label* такою, щоб дорівнювала 304, а властивість *Y* 0.

Далі для створення меню додано до форми компонент *menuStrip* із списку *All Windows* панелі елементів *Toolbox*. Обрано компонент *menuStrip*. В групі *Data* розкрито властивість *Items*, у вікні *Items Collection Editor* в полі вводу *Select items and add to list below* обрано *MenuItems*, обрано *Add.* У списку властивостей в групі *Appearance* встановлено значення властивості *Text* для першого елемента горизонтального меню: &Работники. В групі *Design* встановлено властивість *Name* компонента *toolStripMenuItem1* такою, щоб дорівнювала *toolStripMenuItemRabotniki*. Зазначені вище дії були повторені для додавання усіх необхідних елементів горизонтального та вертикального меню.

Було створено оброблювачі подій на всі пункти меню. Для того, щоб додати оброблювач події *Click* на пункт меню "Выход" на головній формі , виконано такі дії:

1) обрано головну форму у вікні конструктора, для чого у вікні *Solution Explore* правою кнопкою обрано вузол *Form1.h*, пункт *View Design*;

2) вибрано пункт меню "Выход" на формі одинарним натисканням миші;

3) вибрано кнопку *Events* у вікні *Properties*;

4) двічі натиснуто "мишею" на колонку праворуч від події *Click*;

5) ім'я функції *private void toolStripMenuItemExit_Click (object sender, EventArgse)* з'явилося в колонці. *Visual Studio* згенерувало прототип оброблювача події в модулі «Мониторинг параметров восстановления» і показало його в редакторі коду. Створено код оброблювача події *Click*, для чого використано метод *Close()* об'єкта *Form*, введено код у тіло функції.

4.3. Розробка модулів моніторингу параметрів відновлення

4.3.1. Модуль резервного копіювання

В проекті для розробки модулів моніторингу параметрів відновлення бази даних кадрів підприємства було організовано з'єднання з базою даних *SQL Server*. 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 нового підключення обрано *New Connection*, у вікні *Add Connection* у випадаючому списку *Server name* обрано ім'я сервера *User*. В групі *Connect to a database* в списку *Select enter a dabase name* обрано ім'я бази даних *Kadri*, в якій розташовуються таблиці *Rabotniki, Podrazdelenia, Dolgnosti*. Рядок підключення відображено в поточному вікні *Data Source=user;Initial Catalog=Kadri;Integrated Security=True.* Далі у вікні «Выбор объектов базы данных» розкрито об’єкт «Таблицы» та обрано таблиці *Rabotniki, Podrazdelenia, Dolgnosti*. Таким чином сформовано конструктор схем наборів даних *KadriDataSet.*

Для організації зв’язку таблиці з даними в середовищі розробки обрано панель *Data Source*. Для цього обрано пункт меню «Вид», «Другие окна», «Источники данных». Далі для організації зв’язку таблиці з даними на панелі *Data Sources* обрано таблицю *Rabotniki* та переміщено її на форму.

На формі з'явилися компоненти *DataGridView* для візуального відображення даних таблиці, компонента *BindingNavigator* для переміщення по записах таблиці. Компоненти *RabotnikiBindingSource, RabotnikiTableAdapter, tableAdapterManager* та конструктор схем наборів даних *kadriDataSet* з'явились на панелі для невидимих компонентів. Властивість *Name* компонент *RabotnikiBindingSource, RabotnikiTableAdapter* для зручності встановлено в *BindingSourceRabotniki, TableAdapterRabotniki* відповідно.

Для відображення даних таблиці *Rabotniki* на формі візуальний компонент *DataGridView* налаштовано в групі *Design*. Властивість *Name* вказано *DataGridViewRabotniki*, в групі *Data* властивість *DataSource* вказано *BindingSourceRabotniki*, властивість *Visible* вказано *false*, в групі *Layout* обрано

властивість *Anchor*, вказано *Top, Left*. властивість *Dock* вказано *Fill*. Для відображення даних таблиці *Podrazdelenia* і *Dolgnosti* виконано дії аналогічно попереднім пунктам для таблиці *Rabotniki*. Елементи *DataGridView* відображають на формі таблиці бази даних *Kadri*. Цей елемент використовується для відображення переліку працівників, підрозділів та посад.

Для текстового повідомлення про виконання копіювання за допомогою панелі елементів *Toolbox* до форми було додано *ListBox*. Для відображення даних на формі візуальний компонент *ListBox* налаштовано в групі *Design* властивість *Name* вказано *ListBox*1, властивість *Visible* вказано *false* і властивість *Dock* вказано *Fill*.

У вікні *Form.cs* було введено код для пункту «Резервное копирование»:

System.Data.SqlClient.SqlCommand cmd = new System.Data.SqlClient.SqlCommand();

Елемент *SqlCommand* − ініціює новий екземпляр класу. *cmd.CommandType = System.Data.CommandType.Text;*

Елемент *CommandType* − задає значення, яке вказує, як буде інтерпретуватися властивість *CommandText.*

*cmd.CommandText = "USE master; " + "IF EXISTS (SELECT * FROM sys.sysdevices WHERE name='Kadri_backup') " +*

"begin print 'Dropping device' exec sp_dropdevice Kadri _backup end " + "exec sp_addumpdevice 'disk', ' Kadri _backup', 'D:\\Ruduk\\Kadri.bak' " +

"*BACKUP DATABASE Kadri to Kadri_backup*";

Елемент *CommandText* − задає інструкцію *Transact-SQL*, ім'я таблиці або збережену процедуру, виконувану для джерела даних.

Твердження *USE master* активує системну базу даних *master*. Оператор *IF EXISTS* має вкладений запит, цей запит перевіряє наявність логічного пристрою з ім’ям *Auto,* для цього використано системне подання *sys.sysdevices* [14]. Якщо логічний пристрій *Kadri_backup* існує, то він видаляється за допомогою збереженої процедури *sp_dropdevice*. Якщо логічний пристрій *Auto_backup* не існує, то він створюється за допомогою збереженої процедури *sp_addumpdevice*

з логічним ім’ям *Kadri_backup* і розташуванням по шляху *'D:\\Ruduk\\Kadri.bak*' з фізичним ім’ям файлу *Kadri.bak*. Наступний оператор виконує копіювання бази даних *Kadri* на логічний пристрій з ім’ям *Kadri_backup*. Для текстового повідомлення про виконання копіювання за допомогою панелі елементів *Toolbox* до форми було додано *ListBox*.

Для відображення даних на формі візуальний компонент *ListBox* налаштовано в групі *Design* властивість *Name* вказано *ListBox*1, властивість *Visible* вказано *false* і властивість *Dock* вказано *Fill*. Для властивості *Items* використано функцію *Add* з аргументом «Резервная копия базы данных создана» (Рис.4.).

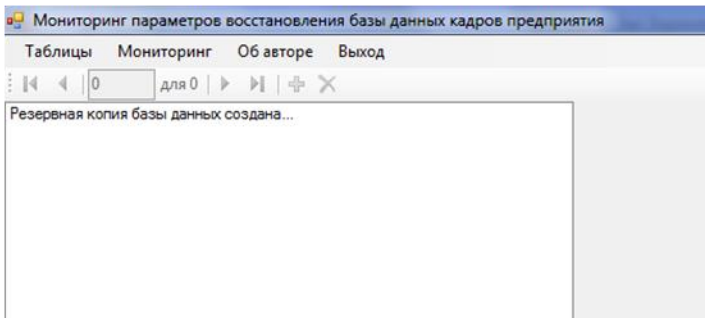

Рис. 4.2. Вікно резервного копіювання

4.3.2. Модуль відновлення бази даних

Для модулю відновлення бази даних у оброблювач події *private void toolStripMenuItemRestore_Click (object sender, EventArgs e)* в модулі *Form1.cs* було введено код для пункту «Восстановление БД»:

```
System.Data.SqlClient.SqlCommand cmd = new
System.Data.SqlClient.SqlCommand();
cmd.CommandType = System.Data.CommandType.Text;
cmd.CommandText = "RESTORE DATABASE kadri FROM DISK =
'D:\\Ruduk\\Kadri.bak '";
cmd.Connection = sqlConnectionMaster;
```

```
sqlConnectionMaster.Open();
System.Data.SqlClient.SqlDataReader reader = cmd.ExecuteReader();
.....................
            dataGridViewModel.Visible = false;
            DataGridViewRabotniki.Visible = false;
            DataGridViewPodrazdelenia.Visible = false;
           DataGridViewDolgnosti.Visible = false;
            listBox1.Visible = true;
            listBox1.Items.Add("База данных восстановлена...");
            reader.Close();
sqlConnectionMaster.Close();
        }
```

CommandText − задає інструкцію *Transact-SQL*, яка відновлює резервну копію бази даних *Kadri* за місцем розташування *D:/Ruduk/Kadri.bak*.

Команда *RESTORE DATABASE* відновлює резервні копії, виконані за допомогою команди *BACKUP* [14]. Ця команда дозволяє виконати наступні сценарії відновлення: відновити базу даних повної резервної копії (повне відновлення); відновити частину бази даних (часткове відновлення); відновити в базі даних певні файли або файлові групи (відновлення файлів); відновити в базі даних певні сторінки (відновлення сторінок); відновити в базі даних журнал транзакцій (відновлення журналу транзакцій); повернути базу даних до моменту часу, на який був виконаний моментальний знімок бази даних. *SQL Server* підтримує різні сценарії відновлення.

Для компонентів, *DataGridViewRabotniki*, *DataGridViewPodrazdelenia*, *DataGridViewDolgnosti* властивість *Visible* встановлено *false*, для того, щоб зробити їх невидимими. Для компонент *dataGridViewModel*, *dataGridViewSysaltfiles* властивість *Visible* встановлено *false*, для того, щоб також зробити їх невидимими. Повідомлення про виконання відновлення відображено в компоненті *ListBox*1. Для властивості *Items* використано функцію *Add* з

аргументом "База данных восстановлена..." (Рис.4.3). Для компоненти *reader sqlConnectionMaster* використано функцію *Close()* для закриття наборів даних.

4.3.3. Модуль перегляду резервних копій

Для модулю відновлення бази даних у оброблювач події *private void ToolStripMenuItemProsmRezCop_Click(object sender, EventArgs e)* в модулі *Form1.cs* було введено код для пункту «Просмотр резервных копий»:

```
System.Data.SqlClient.SqlCommand cmd = new
System.Data.SqlClient.SqlCommand();
cmd.CommandType = System.Data.CommandType.Text;
cmd.CommandText = "SELECT * FROM sys.sysdevices";
cmd.Connection = sqlConnectionMaster;
sqlConnectionMaster.Open();
System.Data.SqlClient.SqlDataReader reader = cmd.ExecuteReader();
……
listBox1.Items.Clear();
listBox1.Visible = true;
listBox1.Items.Add("Просмотр резервных копий...");
while (reader.Read())
{
listBox1.Items.Add(reader.GetName(0));
listBox1.Items.Add(reader.GetValue(0));
listBox1.Items.Add(reader.GetName(6));
listBox1.Items.Add(reader.GetValue(6));
reader.NextResult();
}
reader.Close();
sqlConnectionMaster.Close();
}
```

CommandText − задає інструкцію *Transact-SQL,* яка використовує оператор вибору *SELECT* для отримання даних про існуючі резервні копії з системного подання *sys.sysdevices*. Подання *sys.sysdevices* використане для відображення резервних копій.

Для з'єднання з базою даних *Master* використано компоненту *sqlConnectionMaster*. Компонента *sqlConnectionMaster* використовується для з'єднання бази даних з додатком [14]. Компонента *SqlDataReader* використовується для отримання рядків з результатів запиту. Доступ до окремих стовпців відновленого рядка здійснюється по імені або порядковому номеру стовпця через об'єкт *DataReader*. Для компоненти *SqlDataReader* використано функцію *ExecuteReader*. Функція *ExecuteReader* використовується для відновлення об'єкту *SqlDataReader*, забезпечуючи послідовне зчитування записів. Для компонентів *DataGridViewRabotniki, DataGridViewPodrazdelenia, DataGridViewDolgnosti* властивість *Visible* встановлено *false,* для того, щоб зробити їх невидимими. Результат виконання перегляду резервних копій відображено в компоненті *ListBox*1. Для властивості *Items* використано функцію *Add* з аргументом "Просмотр резервных копий..."(рис. 4.2). Для компоненти *reader sqlConnectionMaster* використано функцію *Close()* для закриття наборів даних.

Для відновлення об'єкту *SqlDataReader*, що використовується для читання даних застосовано метод *ExecuteReader*. Так, отримано всі дані з таблиці користувачів і виведено їх на консоль. Для вибірки даних з БД використано *SQL*-вираз *SELECT*. У даному випадку обрано всі стовпці всіх рядків таблиці. Отримавши при виконанні запиту об'єкт *SqlDataReader*, тепер можливо переглянути всі отримані дані. Але спочатку перевіривши наявність даних взагалі за допомогою властивості *HasRows*. Якщо дані є, то заголовки таблиці виводяться за допомогою методів *reader*.*GetName()*. Стовпці отримано

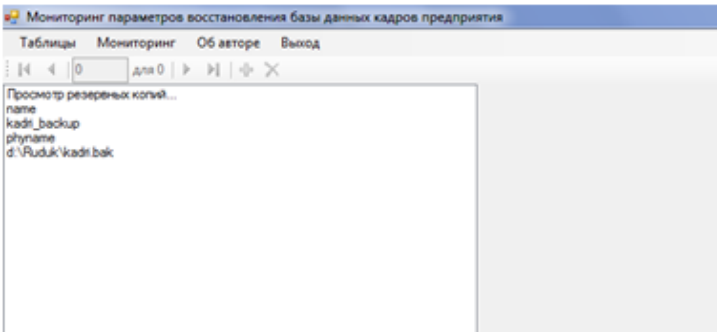

Рис. 4.3. Вікно перегляду резервних копій

у вибірці саме в тому порядку, в якому вони були визначені в таблиці. Тобто якщо першим у таблиці стоятиме стовпець «*Name*», то щоб отримати його дані застосовується метод *GetName(1)* (так як нумерація стовпців йде з нуля). Далі зчитано самі дані. За допомогою методу *reader.Read(), reader* перейшов до наступного рядка і повернув логічне значення, яке вказує, чи є дані для зчитування. У циклі, *while* (*reader.Read()*) в порядку проходження стовпців отримано дані за допомогою методу *GetValue()*, який відновив дані у вигляді об'єкта типу *object*. Стовпець *Name* (*Kadri_backup*) вказано першим, тоді для його отримання застосовано метод *reader.GetValue(0),* а стовпець «*Phyname*» (фізичний шлях до бази даних *'D:/Ruduk/Kadri.bak'*), вказано другим і його значення отримано за допомогою *reader.GetValue(1).*

4.3.4. Модуль перегляду моделі відновлення бази даних

Для модулю відновлення бази даних у оброблювач події *private private void ToolStripMenuItemModel_Click(object sender, EventArgs e)* в модулі *Form1.cs* було введено код для пункту «Просмотр моделей восстановления»:

```
masterDataSet.databases.Clear();
sqlDataAdapterModel.Fill(masterDataSet.databases);
    ……
    listBox1.Visible = false;
    dataGridViewModel.ReadOnly = true;
    dataGridViewModel.Visible = true;
```

KadriDataSet є резидентним поданням даних, що забезпечує послідовну реляційну модель програмування незалежно від джерела даних. *DataSet* представляє повний набір даних, який включає в себе таблиці, обмеження і зв'язки між таблицями. Оскільки *DataSet* не залежить від джерела даних, то набір даних може включати в себе дані, локальні для застосування, і дані з декількох джерел даних. Взаємодія з існуючими джерелами даних контролюється за допомогою *DataAdapter*. Елемент *DataAdapter* надає набір виконуваних над даними команд і підключень бази даних, які використовується для заповнення *DataSet* і оновлення бази даних *SQL Server*. Для елементу *DataAdapter* застосовано функцію *Fill*. Функція *Fill* додає або оновлює рядки в об'єкті *KadriDataSet*. Виконано запит: *select name,database_id,recovery_model, recovery_model_desc, collation_name, is_read_only, is_cleanly_shutdown from sys.databases.*

Для компонентів *DataGridViewRabotniki, DataGridViewPodrazdelenia, DataGridViewDolgnosti* властивість *Visible* встановлено *false*, для того, щоб зробити їх невидимими. Результат виконання відновлення відображено в компоненті *DataGridView*. Для компонентів *dataGridViewModel* властивості *ReadOnly* та *Visible* встановлено *true*. Вікно перегляду моделей відновлення відображено на рис. 4.3.

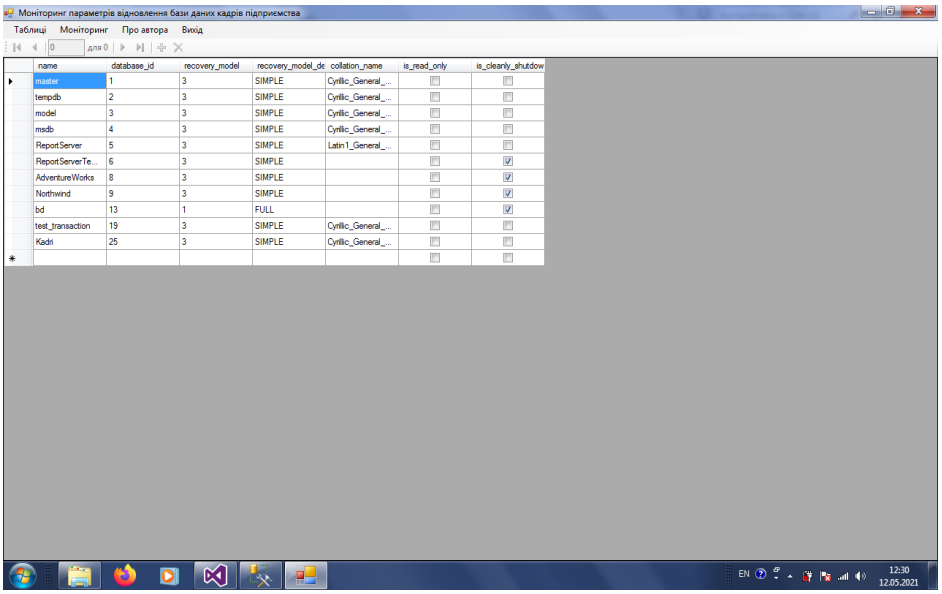

Рис. 4.3. Вікно перегляду моделей відновлення

4.4. Висновки до розділу

У середовищі *Visual Studio* використовуються два типи файлів *SLN* і *SUO* для зберігання параметрів рішень. Файл *WindowsFormsApplicationKadri.SLN* організовує проекти, елементи проектів та елементи рішень в рішення. Файл *SUO* записує всі параметри, пов'язані з рішенням так, що кожен раз при відкриванні рішення включаються налаштування користувача.

В результаті розробки було створено модулі для пунктів меню. Пункт «Мониторинг» призначений для адміністратора кадрів підприємства, який матиме змогу робити резервне копіювання та відновлення бази даних з метою захисту бази даних від втрати. В результаті дипломної роботи створено модулі резервного копіювання, модуль відновлення бази даних, модуль перегляду резервних копій та модуль перегляду моделі відновлення.

Основним завданням створених модулів являється запобігання втрати бази даних завдяки проведенню резервного копіювання. Для цього адміністратору підприємства рекомендується розробити стратегію резервного копіювання за якою проводити періодичні резервні копіювання. Це дасть змогу без проблем відновити базу даних кадрів підприємства в разі її втрати. Захищення бази даних в такий спосіб являється простим та надійним, що буде зручно у використанні системному адміністратору підприємства.

Проект було виконано з дотриманням діючих стандартів та положень [18], [19].

ВИСНОВКИ

Важливими роботами адміністратора бази даних є резервне копіювання та відновлення бази даних. Існують три методи резервного копіювання: повне (*Full*, яке дозволяє забезпечити максимальну відповідність оригіналу даних його копії), диференційне (*Differential*, створює лише копії частин бази даних, які змінились з моменту останнього повного копіювання бази даних) та резервне копіювання протоколу транзакцій (*Transaction log*, що враховує лише зміни, що внесені до протоколу). При створенні резервної копії бази даних вперше обов'язково використовується метод копіювання *Full*.

Відновлення даних – це процес відновлення доступу до даних, що зберігаються на будь-якому носії пристрої запам'ятовування. *SQL Server* (система керування базами даних) створює в журналі транзакцій контрольні точки при копіюванні зафіксованих транзакцій з журналу транзакцій в базу даних. Так механізм відновлення може швидко пройти по ланцюжку записаних в журнал операцій, від самої останньої до першої операції, і повернути результати всіх операцій в зворотному від їх виконання порядку.

Для виконання відновлення та резервного копіювання бази даних в *SQL Server* використовуються оператори *Transact-SQL (BACKUP DATABASE* та *RESTORE)* та середовище *SQL Server Management Studio*. Також засобом резервного копіювання є план обслуговування бази даних *(database maintenance plan)* та утиліта *SQLmaint*.

Розрізняють три моделі відновлення: проста модель *(Simple)*, модель з повним *(Full)* та модель з частковим *(Bulk-logged)* протоколюванням. Проста модель передбачає резервне копіювання тільки бази даних, відповідно відновити стан базу можна тільки на момент створення резервної копії, всі зміни після створення останньої резервної копії втрачаються, журнал автоматично усікається. В моделі повного відновлення усі операції записуються в протокол транзакцій, тому ця модель надає повний захист від збоїв зовнішніх пристроїв. Ця модель дозволяє відновити базу на будь-який довільний момент часу, але

вимагає, крім резервних копій бази, зберігати копії протоколу транзакцій за весь період, для якого може знадобитися відновлення, журнал не усікається. Відновлення з неповним протоколюванням підтримує протоколи резервних копій при використанні мінімального простору в протоколі транзакцій для деяких масштабних операцій (вставки, видалення, зміни). Модель з неповним протоколюванням рекомендується тільки як доповнення до повної моделі на період великомасштабних масових операцій, коли немає необхідності відновлення бази на певний момент часу, журнал також не перезаписується. Операції резервного копіювання і відновлення *SQL Server* виконуються в контексті моделі відновлення бази даних.

В моделі *Simple* можливе резервне копіювання тільки повне та різницеве. В моделі *Full* можливе резервне копіювання повне, різницеве та журналу транзакцій. В моделі Bulk-logged можливе повне резервне копіювання, різницеве та журналу транзакцій.

Стратегія відновлення повинна займати належне місце, щоб запобігти випадковій втраті даних. Вона повинна включати в себе узгоджену систему резервного копіювання даних. Існує чотири основні стратегії: стратегія повного резервного копіювання бази даних, стратегія резервного копіювання бази даних і журналу транзакцій, стратегія різницевого резервного копіювання та стратегія резервного копіювання файлів і файлових груп.

Стратегія повного резервного копіювання бази даних доцільна, якщо база даних має невеликий розмір і піддається незначним змінам. Стратегію резервного копіювання бази даних і журналу транзакцій рекомендується використовувати, якщо база даних часто змінюється і/або повне резервне копіювання займає надто багато часу. Стратегія різницевого резервного копіювання також доцільна для великої бази даних, щоб скоротити час на резервне копіювання і можлива тоді, коли резервне копіювання журналів транзакцій виконується окремо. Стратегія різницевого резервного копіювання включає в себе створення регулярних повних резервних копій бази даних з проміжними різницевими резервними копіями. Між повними і різницевими

резервними копіюваннями можна також додатково виконувати резервні копіювання журналу транзакцій.

Проектування системного програмного засобу відбувалось у середовищі *Management Studio* з використанням системних подань *sys.sysaltfiles* і *sys.sysdevices* та збережених процедур *sp_addumpdevice* і *sp_dropdevice*. Системне подання *sys.sysaltfiles* використано для відображення інформації про усі файли бази даних кадрів підприємства, а системне подання *sys.sysdevices* − використано при розробці модулів моніторингу. Системна процедура *sp_addumpdevice* додає пристрій резервного копіювання в системне подання *sys.backup_devices*, після чого пристрій можна вказувати в інструкціях *BACKUP* і *RESTORE* по логічному імені. Процедура *sp_dropdevice* видаляє пристрій бази даних або пристрій резервного копіювання з екземпляра компонента *SQL Server Database Engine*, видаляючи запис з *master.dbo.sysdevices*.

В результаті розробки було створено модулі для пунктів меню. Пункт «Мониторинг» призначений для адміністратора кадрів підприємства, який матиме змогу робити резервне копіювання та відновлення бази даних з метою захисту бази даних від втрати. В дипломному проекті створено модулі резервного копіювання, модуль відновлення бази даних, модуль перегляду резервних копій та модуль перегляду моделі відновлення.

Основним завданням створених модулів являється запобігання втрати бази даних завдяки проведенню резервного копіювання. Для цього адміністратору підприємства рекомендується розробити стратегію резервного копіювання за якою проводити періодичні резервні копіювання. Це дасть змогу без проблем відновити базу даних кадрів підприємства в разі її втрати. Захищення бази даних в такий спосіб являється простим та надійним, що буде зручно у використанні системному адміністратору підприємства.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАННИХ ДЖЕРЕЛ

1. Дж. Буркс *Microsoft SQL Server* 2008.Руководство для начинающих – С.: «БХВ-Петербург», 2009. – 752с.
2. Системные базы данных [Електронний ресурс] – Режим доступу: *https://msdn.microsoft.com/ru-ru/library/ms178028(v=sql.90).aspx* (дата звернення 25.12.2020)
3. Параметры базы данных [Електронний ресурс] – Режим доступу: *https://technet.microsoft.com/ru-ru/library/ms179472(v=sql.105).aspx* (дата звернення 25.12.2020)
4. Рикарди Г., Системы баз данных. СПб.: «Вильямс», 2001. – 201с.
5. Бондарь А. *Microsoft SQL Server* 2014. СПб.: «БХВ-Петербург », 2015. – 592с.
6. Джеймс Р. Грофф, Пол Н. Вайнберг. *SQL*: Полное руководство. – 3-е изд.: Пер.с англ. – М.: Издательский дом " Диалектика-Вильямс", 2012. – 960с.
7. Роберт Виейра. Програмування баз даних *Microsoft SQL Server 2005*. Базовий курс = Программирование баз данных *Microsoft SQL Server 2005*. Базовый курс. – М. : «Діалектика», 2007. – 832с.
8. Майк Гандерлой, Джозеф Джорден, Дейвид Чанц. Освоєння *Microsoft SQL Server 2005* = Освоение *Microsoft SQL Server 2005*. – М. : «Діалектика», 2007. – 2204с.
9. Рудикова Л. Базы даных, Разработка приложений СПб.: «БХВ-Петербург », 2006. – 487с.
10. Джоунс Е., Функции *SQL* Справочник программиста, М.: «Диалектика» 2007. – 760с.
11. Конноли, Т. Базы данных. Проектирование, реализация и сопровождение. – 3-е изд. – М.: Изд. Дом «Вильяме» 2003. . – 1440с.
12. *Microsoft® SQL Server™ 2005*.Реализация и обслуживание. Учебный курс *Microsoft* (Экзамен 70-431). – М.: «Питер», 2007. – 767с.
13. Герберт Шилдт. *C#* 4.0: полное руководство = *C# 4.0 The Complete Reference*. – М.: «Вильямс», 2010. – 1056с.

14. Майо Д. Самоучитель *Microsoft Visual Studio 2010 – Microsoft Visual Studio 2010: A Beginner's Guide (A Beginners Guide)*. – С.: «БХВ-Петербург», 2010. – 46с.
15. Тернстрем Т. *Microsoft SQL SERVER 2008* разработка баз данных – М. «Русская редакция», 2010. – 494с.
16. Повна документація по можливостям *.NET Framework* [Електронний ресурс] – Режим доступу: *https://docs.microsoft.com/ru-ru/dotnet/framework/* (дата звернення 20.05.2021)
17. Повна документація по можливостям модуля *Windows Forms* [Електронний ресурс] – Режим доступу: *https://docs.microsoft.com/ru-ru/dotnet/framework/winforms/* (дата звернення 20.05.2021)
18. 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.
19. Слободян О. Положення про дипломні роботи (проекти) випускників Національного авіаційного університету. – К.: Видавництво НАУ, 2017. – 63с.

ДОДАТОК А. ФРАГМЕНТ ПРОГРАМНОГО КОДУ *FORM1.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
"kadriDataSet.Rabotniki". При необходимости она может быть перемещена или удалена.
            this.TableAdapterRabotniki.Fill(this.KadriDataSet.Rabotniki);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"kadriDataSet.Rabotniki". При необходимости она может быть перемещена или удалена.
            this.TableAdapterRabotniki.Fill(this.KadriDataSet.Rabotniki);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"nf_optovikDataSet.postav". При необходимости она может быть перемещена или удалена.
            this.postavTableAdapter.Fill(this.nf_optovikDataSet.postav);
        }


        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 toolStripMenuItemCreateBD_Click(object sender, EventArgs e)
        {
            // создание базы данных
            dataGridViewSysaltfiles.Visible = false;
            System.Data.SqlClient.SqlCommand cmd = new
System.Data.SqlClient.SqlCommand();
            cmd.CommandType = System.Data.CommandType.Text;
            cmd.CommandText = "IF DB_ID(N'ruduk_primer') IS NOT NULL begin DROP DATABASE
ruduk_primer end " +
                              "CREATE DATABASE ruduk_primer";
```

```
            cmd.Connection = sqlConnectionMaster;
            sqlConnectionMaster.Open();
            System.Data.SqlClient.SqlDataReader reader = cmd.ExecuteReader();
            listBox1.Visible = true;
            listBox1.Items.Add("База данных создана...");
            reader.Close();
            sqlConnectionMaster.Close();
        }

        private void ToolStripMenuItemDeleteDB_Click(object sender, EventArgs e)
        {
            System.Data.SqlClient.SqlCommand cmd = new
System.Data.SqlClient.SqlCommand();
            cmd.CommandType = System.Data.CommandType.Text;
            cmd.CommandText = "IF DB_ID(N'ruduk_primer') IS NOT NULL begin DROP DATABASE
ruduk_primer end";
            cmd.Connection = sqlConnectionMaster;
            sqlConnectionMaster.Open();
            System.Data.SqlClient.SqlDataReader reader = cmd.ExecuteReader();
            listBox1.Visible = true;
            listBox1.Items.Add("База данных удалена...");
            reader.Close();
            sqlConnectionMaster.Close();
        }

        private void toolStripMenuItemRezCopy_Click(object sender, EventArgs e)
        {
            System.Data.SqlClient.SqlCommand cmd = new
System.Data.SqlClient.SqlCommand();
            cmd.CommandType = System.Data.CommandType.Text;
            cmd.CommandText = "USE master; "+"IF EXISTS (SELECT * FROM sys.sysdevices
WHERE name='kadri_backup') "+
            "begin print 'Dropping device' exec sp_dropdevice kadri_backup end " +
             "exec sp_addumpdevice 'disk', 'kadri_backup', 'd:\\Ruduk\\kadri.bak' " +
            "BACKUP DATABASE kadri to kadri_backup";
            cmd.Connection = sqlConnectionMaster;
            sqlConnectionMaster.Open();
            System.Data.SqlClient.SqlDataReader reader = cmd.ExecuteReader();
            DataGridViewPostav.Visible = false;
            dataGridViewModel.Visible = false;
            dataGridViewLog_space_usage.Visible = false;
            DataGridViewRabotniki.Visible = false;
            listBox1.Items.Clear();
            listBox1.Visible = true;
            listBox1.Items.Add("Резервная копия базы данных создана...");
            reader.Close();
            sqlConnectionMaster.Close();
        }

        private void ToolStripMenuItemProsmRezCop_Click(object sender, EventArgs e)
        {
            System.Data.SqlClient.SqlCommand cmd = new
System.Data.SqlClient.SqlCommand();
            cmd.CommandType = System.Data.CommandType.Text;
            cmd.CommandText = "SELECT * FROM sys.sysdevices where name='kadri_backup'";
            cmd.Connection = sqlConnectionMaster;
            sqlConnectionMaster.Open();
            System.Data.SqlClient.SqlDataReader reader = cmd.ExecuteReader();
            DataGridViewPostav.Visible = false;
            dataGridViewModel.Visible = false;
            dataGridViewSysaltfiles.Visible = false;
            dataGridViewLog_space_usage.Visible = false;
            DataGridViewRabotniki.Visible = false;
            listBox1.Items.Clear();
            listBox1.Visible = true;
```

```
            listBox1.Items.Add("Просмотр резервных копий...");
            while (reader.Read())
            {
                listBox1.Items.Add(reader.GetName(0));
                listBox1.Items.Add(reader.GetValue(0));
                listBox1.Items.Add(reader.GetName(6)); // шестая колонка в sys.sysdevices
                listBox1.Items.Add(reader.GetValue(6));
                reader.NextResult();
            }
            reader.Close();
            sqlConnectionMaster.Close();
        }

        private void toolStripMenuItemProsmPuti_Click(object sender, EventArgs e)
        {
            //выполняется в dataGridViewSysaltfiles
            //select  fileid, groupid, size, maxsize, growth, dbid, name, filename from
sys.sysaltfiles
            //select database_id, file_id, type_desc, name,physical_name from
sys.master_files
            masterDataSet1.sysaltfiles.Clear();
            this.sqlDataAdapter_sysalytfiles.Fill(this.masterDataSet1.sysaltfiles);
            DataGridViewPostav.Visible = false;
            dataGridViewModel.Visible = false;
            dataGridViewLog_space_usage.Visible = false;
            DataGridViewRabotniki.Visible = false;
            listBox1.Visible = false;
            dataGridViewSysaltfiles.ReadOnly = true;
            dataGridViewSysaltfiles.Visible = true;
        }

        private void ToolStripMenuItemModel_Click(object sender, EventArgs e)
        {
            // просмотр моделей восстановления
            //select name,recovery_model_desc from sys.databases
            masterDataSet.databases.Clear(); /* очистка от данных в кэше */
            sqlDataAdapterModel.Fill(masterDataSet.databases);
            DataGridViewPostav.Visible = false;
            dataGridViewSysaltfiles.Visible = false;
            dataGridViewLog_space_usage.Visible = false;
            DataGridViewRabotniki.Visible = false;
            listBox1.Visible = false;
            dataGridViewModel.ReadOnly = true;
            dataGridViewModel.Visible = true;
        }

        private void toolStripMenuItemRestore_Click(object sender, EventArgs e)
        {
            System.Data.SqlClient.SqlCommand cmd = new
System.Data.SqlClient.SqlCommand();
            cmd.CommandType = System.Data.CommandType.Text;
            //cmd.CommandText = "RESTORE DATABASE nf_primer FROM DISK =
'd:\\406_HNF\\nf_primer.bak'";
            cmd.CommandText = "RESTORE DATABASE nf_primer FROM DISK =
'd:\\Ruduk\\ruduk_primer'";
            cmd.Connection = sqlConnectionMaster;
            sqlConnectionMaster.Open();
            System.Data.SqlClient.SqlDataReader reader = cmd.ExecuteReader();
            DataGridViewPostav.Visible = false;
            dataGridViewModel.Visible = false;
            dataGridViewLog_space_usage.Visible = false;
            DataGridViewRabotniki.Visible = false;
            listBox1.Visible = true;
            listBox1.Items.Add("База данных восстановлена...");
            reader.Close();
```

```
            sqlConnectionMaster.Close();
        }

        private void sphelpToolStripMenuItem_Click(object sender, EventArgs e)
        {
            System.Data.SqlClient.SqlCommand cmd = new
System.Data.SqlClient.SqlCommand();
            cmd.CommandType = System.Data.CommandType.Text;
            //cmd.CommandText = "exec sp_help";
            cmd.CommandText = "use nf_optovik exec sp_help tovar";
            cmd.Connection = sqlConnectionMaster;
            sqlConnectionMaster.Open();
            System.Data.SqlClient.SqlDataReader reader = cmd.ExecuteReader();
            DataGridViewPostav.Visible = false;
            dataGridViewModel.Visible = false;
            dataGridViewLog_space_usage.Visible = false;
            DataGridViewRabotniki.Visible = false;
            listBox1.Items.Clear();
            listBox1.Visible = true;
            listBox1.Items.Add("Хранимая процедура sp_help выполнена...");
            int nstr; nstr = 1; /* номер строки для табл */
            while (reader.Read())
            {
                listBox1.Items.Add(nstr++);
                listBox1.Items.Add(reader.GetName(0));
                listBox1.Items.Add(reader.GetValue(0));
                listBox1.Items.Add(reader.GetName(1));
                listBox1.Items.Add(reader.GetValue(1));
                listBox1.Items.Add(reader.GetName(2));
                listBox1.Items.Add(reader.GetValue(2));
                listBox1.Items.Add('_');
                //reader.NextResult();
            }
            reader.Close();
            sqlConnectionMaster.Close();
        }

        private void ToolStripMenuItemRabotniki_Click(object sender, EventArgs e)
        {
            BindingNavigator.BindingSource = BindingSourceRabotniki;
            TableAdapterRabotniki.Update(KadriDataSet.Rabotniki);
            dataGridViewSysaltfiles.Visible = false;
            dataGridViewModel.Visible = false;
            dataGridViewLog_space_usage.Visible = false;
            DataGridViewRabotniki.Visible = false;
            listBox1.Visible = false;
            TableAdapterRabotniki.Fill(KadriDataSet.Rabotniki);
            DataGridViewRabotniki.Visible = true;
        }
    }
}
```