

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ.....	4
1. ПРАКТИЧЕСКИЕ РАБОТЫ.....	5
1.1. Практическая работа №1. Системные базы данных SQL Server	5
1.2. Практическая работа №2. Оптимизация запросов в ПО MS SQL Server	8
1.3. Практическая работа №3. Мониторинг активности и производительности MS SQL Server	15
2. ЛАБОРАТОРНЫЕ РАБОТЫ	24
2.1. Лабораторная работа №1. Перепроектирование физической модели базы данных.....	24
2.2. Лабораторная работа №2. Создание файлов базы данных с помощью языка T-SQL.....	28
2.3. Лабораторная работа №3. Резервное копирование базы данных в среде MS SQL Server	32
2.4. Лабораторная работа №4. Создание ролей и пользователей в базе данных.....	39
Список литературы	44
Сведения об авторах	44

ВВЕДЕНИЕ

Рабочая тетрадь представляет собой описания практических и лабораторных работ, организуемых в ходе изучения дисциплин, связанных с администрированием реляционных СУБД.

В качестве программного обеспечения используется популярная система управления базами данных Microsoft SQL Server 2019 и оболочка управления Microsoft SQL Server Management Studio (SSMS 18). Оба программных продукта имеют “бесплатные без ограничений” версии и просты в установке и первичной настройке перед работой (как в лабораторных классах, так и в домашних условиях), что, в совокупности с их популярностью на индустриальном уровне, обуславливает выбор этого ПО, как основного в процессе обучения. Видеоинструкция по подготовке рабочего места для выполнения практикума и лабораторных работ доступно по следующей ссылке: <https://www.youtube.com/watch?v=h6BGVRy68UY&t=9s>

Тематика предлагаемых практических и лабораторных работ затрагивает такие основные функции администратора, как: резервное копирование и восстановление данных, перепроектирование баз данных, управление системными базами данных сервера, текущий мониторинг активности и состояния сервера баз данных, создание и управление ролевой моделью пользователей базы данных.

Таким образом, пособие раскрывает базовые аспекты администрирования сервера баз данных на различных уровнях и может быть использовано при выполнении практических и лабораторных задач по дисциплине «Администрирование многопользовательских баз данных» студентами бакалавриата, специальности 09.03.02 «Информационные системы и технологии».

1. ПРАКТИЧЕСКИЕ РАБОТЫ

1.1. Практическая работа №1. Системные базы данных SQL Server

Системные базы данных - набор баз, необходимых для обеспечения функционирования ядра сервера, создается *автоматически* при установке экземпляра сервера MS SQL.

К системным базам относят:

- *master* - содержит всю системную информацию о SQL Server.
- *msdb* - используется агентом SQL Server для создания расписания предупреждений и заданий.
- *model* - используется в качестве шаблона для всех баз данных, создаваемых в экземпляре SQL Server.
- *resource* - доступна только для чтения и содержит все системные объекты, составляющие ядро SQL Server.
- *tempdb* - глобальный ресурс, доступный всем пользователям, подключенным к экземпляру сервера.

Задание для самостоятельного выполнения.

Используя базу данных *model*, выполните следующие запросы:

```
select * from sys.database_scoped_configurations; (демонстрация конфигурации базы данных)
```

```
select * from sys.databases where name = 'model'; (вывод свойств выбранной базы данных)
```

// Базу необходимо указывать в апострофах

```
select SERVERPROPERTY('ResourceVersion'); go (вывод версии базы данных resource)
```

```
select SERVERPROPERTY('ResourceLastUpdateDateTime');go (вывод даты последнего обновления базы данных resource)
```

Внесите в лист отчета о практике результаты выполнения запросов в виде текста ответа сервера.

Найдите в проводнике объектов системную базу данных tempdb и сформируйте стандартный отчет “Занято места на диске”. Внесите информацию из полученного документа в лист отчета о практике.

Выполните следующий запрос:

```
USE master;
GO
CREATE TABLE #TempTable (id int IDENTITY (1,1), col1 char (4000));
GO
SET NOCOUNT ON;
DECLARE @i int = 0;
WHILE @i < 10000
BEGIN
INSERT INTO #TempTable (col1) VALUES ('Test data');
SET @i += 1;
END;
GO
```

После выполнения запроса, снова сформируйте стандартный отчет “Занято места на диске” по системной базе данных tempdb, внесите его в лист отчета о практике, сравнив его с предыдущим. Перезапустите сервер, заново сформируйте отчет “Занято места на диске” и внесите его данные в отчет. Проанализируйте изменение размера файла базы данных и зафиксируйте изменения в отчете.

Кратко опишите параметры системной базы данных master. Список параметров необходимо внести в таблицу “Название свойства/Описание свойства” (образец таблицы показан в табл. 1). Всю нужную информацию можно найти по ссылке в разделе “Параметры базы данных”:
<https://docs.microsoft.com/ru-ru/sql/relational-databases/databases/master-database?view=sql-server-ver15>

Таблица 1. Свойства системной базы данных master

Название свойства	Описание свойства
ALLOW_SNAPSHOT_ISOLATION	Изоляция моментальных снимков в транзакции

Содержание итогового отчета по практике:

Отчет должен содержать следующее:

1. Результаты выполнения запросов в системных базах данных.
2. Три стандартных отчета “Занято место на диске”, сформированных в процессе выполнения задания.
3. Таблица “Название свойства/Описание свойства” для системной базы данных master.

Лист отчета о практике 1. Выполнил:

1.2. Практическая работа №2. Оптимизация запросов в ПО MS SQL Server

Оптимизатор запросов - компонент СУБД, который отвечает за обработку данных в запросе с максимальной эффективностью (точность, качество, скорость). Основная задача оптимизатора - перебор всех возможных стратегий выполнения запроса и выбор из них наиболее эффективной стратегии. Она называется *планом выполнения запроса*.

Селективность выборки - отношение количества строк, удовлетворяющих условию выборки, к общему количеству строк в таблице. Главная задача при определении селективности - оценить стоимость использования при реализации запроса индексов (сканировать таблицу или сканировать индекс). Если селективность является 10% и ниже, сканирование таблицы вероятно будет эффективнее сканирования индекса.

Статистика для оптимизации запросов - большие двоичные объекты (BLOB-объекты), содержащие статистические сведения о распределении значения в одном или нескольких столбцах таблицы или индексированного представления. Каждый объект статистики создается для списка из одного или нескольких столбцов таблицы и содержит гистограмму, на которой отображается распределение значения в первом столбце.

Гистограмма измеряет частоту появления каждого различающегося значения в наборе данных. Чтобы создать гистограмму, оптимизатор запросов сортирует значения столбцов и вычисляет количество значений, совпадающих с каждым различающимся значением столбца, а затем осуществляет статистическую обработку значений столбцов с получением непрерывных шагов гистограммы, максимальное количество которых составляет 200.

Задание для самостоятельного выполнения.

Воспользуйтесь учебной базой данных AdventureWorks 2012 и выберите все записи из таблицы Person.Address. Обратите внимание на столбец StateProvinceID. Отсортируйте таблицу по этому столбцу.

В результате сортировки можно увидеть, что максимальное значение в столбце - 181. При этом некоторые значения будут пропущены, например, 2. Из этого можно сделать вывод, что в гистограмме максимальное количество шагов не превысит 181.

Построим гистограмму выбранной таблицы. Для этого найдем таблицу Person.Address в обозревателе объектов. В папке статистика найдем отчет IX_Address_StateProvinceID (рис. 1).



Рисунок 1. Таблица Person.Address в обозревателе объектов.

Нажмем два раза на выбранный отчет и откроем страницу “Подробности”. Перед нами **гистограмма статистики индекса**. Обратим внимание на столбец “EQ_ROWS” — это количество строк, в которых значение StateProvinceID одинаково. Выберем два значения - с высокой и низкой селективностью. В качестве значения с высокой селективностью выберем StateProvinceID = 32, с низкой StateProvinceID = 9.

Предварительно создайте собственную пользовательскую базу данных с любым доступным именем (в скриптах ниже будет использоваться название qtcourse). На основании статистики создадим выборку данных с выделенным индексом:

```
USE qtcourse
SELECT * into new_addresses FROM AdventureWorks2012.Person.Address;
GO
CREATE INDEX stpr on new_addresses (StateProvinceID)
```

Перенесем все данные из таблицы Person.Address в новую таблицу и создадим индекс на столбце StateProvinceID.

Создадим запрос с заранее определенной высокой селективностью:

```
USE qtcourse
SELECT * FROM new_addresses a WHERE a.StateProvinceID = 32;
```

Выведем графический план запроса для заданной инструкции. Для этого выделим запрос - выборку и, вызвав правой кнопкой мыши меню, выберем строку “Показать предполагаемый план выполнения” (рис. 2).

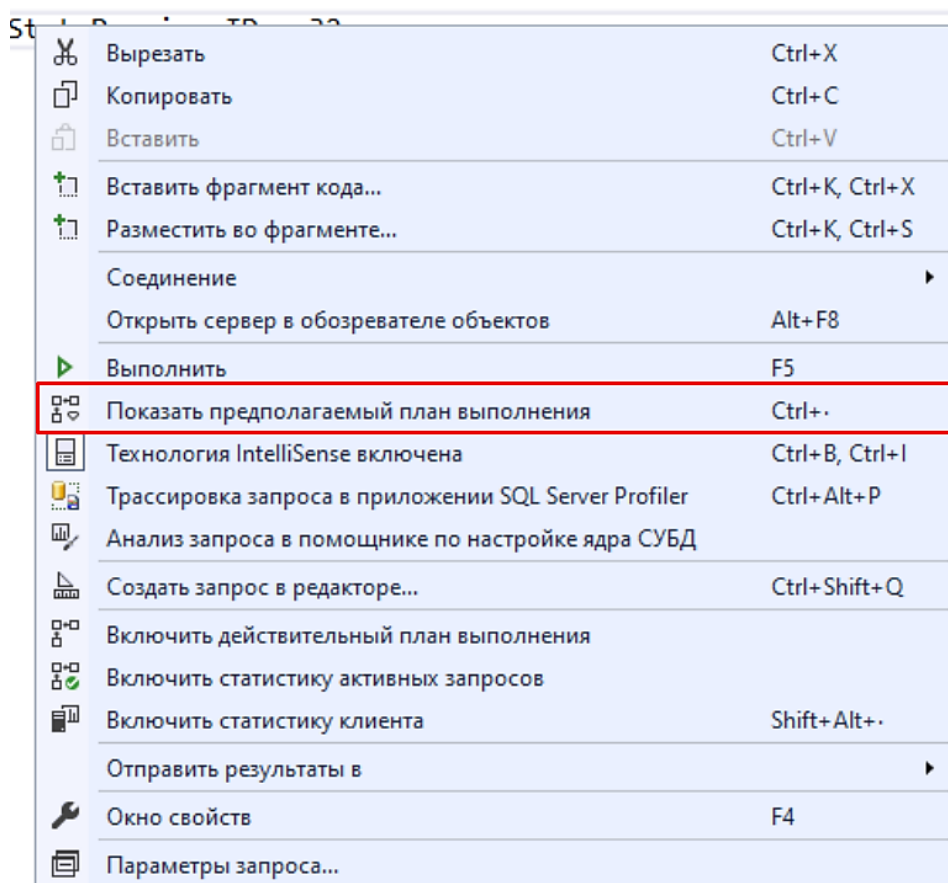


Рисунок 2. Инструкция к показу предполагаемого плана выполнения

В результате получим следующий предполагаемый план выполнения данного запроса (рис. 2.3):

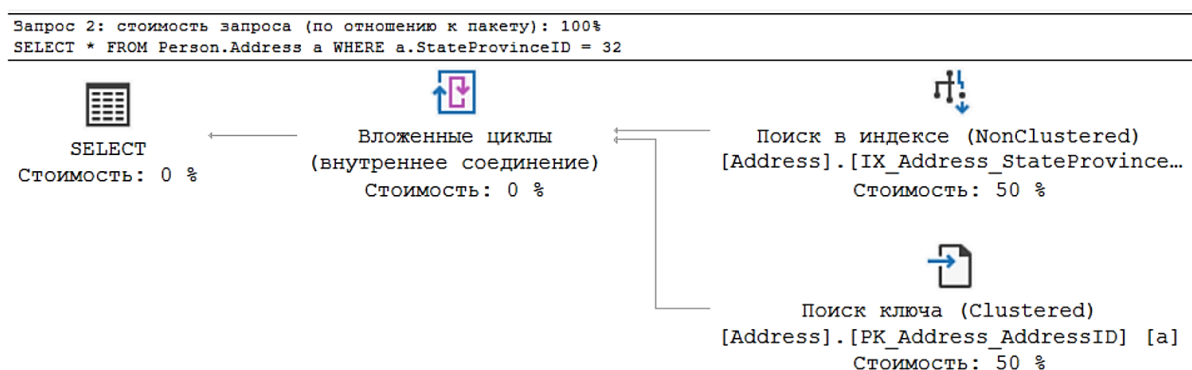


Рисунок 3. Предполагаемый план выполнения

Обратите внимание на то, что СУБД будет использовать индекс для поиска необходимых данных в таблице.

Теперь создадим запрос с заранее определенной низкой селективностью:

```

USE qtcourse
SELECT * FROM new_addresses a WHERE a.StateProvinceID =9;
  
```

Выведем графический план запроса для заданной инструкции (рис. 3):

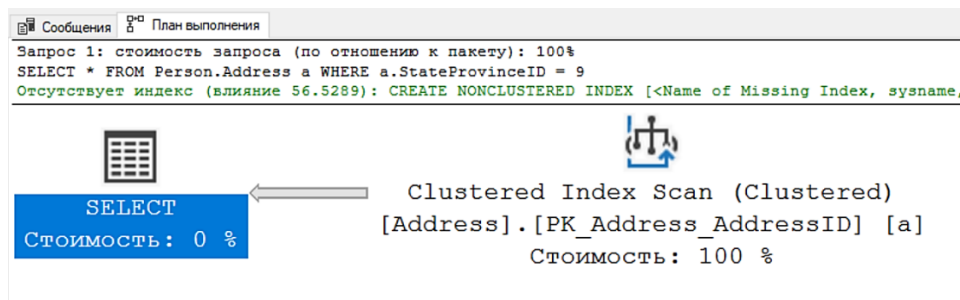


Рисунок 4. Графический план запроса с низкой селективностью

Обратите внимание, СУБД не использует индекс для поиска нужных строк, ведь гораздо дешевле пройти всю таблицу целиком и вывести выбранные строки.

Теперь создадим запрос с неявно определенным кластеризованным индексом:

```
USE AdventureWorks2012
SELECT * FROM HumanResources.Employee
WHERE HumanResources.Employee.BusinessEntityID = 10;
```

Выведем графический план запроса для заданной инструкции (рис. 5).

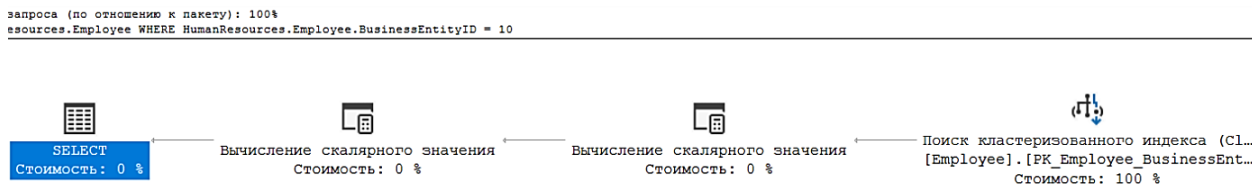


Рисунок 5. Графический план запроса с неявно определенным кластеризованным индексом

СУБД будет использовать кластеризованный индекс для поиска необходимых строк. Помимо оптимизации запросов баз данных существует также возможность оптимизации соединений баз данных. В зависимости от статистических данных используются три техники обработки соединений:

Вложенные циклы. Для каждой строки внешней таблицы отыскивается и сравнивается каждая строка из внутренней таблицы.

Слияние соединения. Строки соединяемых таблиц должны быть соединены общим предикатом соединения. Сканирование происходит как раз по предикату.

Хеширование соединения. Соединение без каких-либо индексов для соединяемых столбцов.

Создадим запрос, оптимальное выполнение которого осуществляется через вложенные циклы:

```
USE AdventureWorks2012
SELECT * FROM HumanResources.Employee e
JOIN HumanResources.EmployeeDepartmentHistory d
```

```
ON e.BusinessEntityID = d.BusinessEntityID  
AND e.BusinessEntityID = 10;
```

Выведем графический план запроса для заданной инструкции. Обратите внимание на то, что запрос действительно выполняется с использованием вложенного цикла.

Создадим запрос, оптимальное выполнение которого осуществляется через слияние соединения:

```
USE AdventureWorks2012  
SELECT * FROM HumanResources.Employee e  
JOIN HumanResources.EmployeeDepartmentHistory d  
ON e.BusinessEntityID = d.BusinessEntityID;
```

Выведем графический план запроса для заданной инструкции.

Создадим запрос, оптимальное выполнение которого осуществляется через хэширование соединения:

```
USE AdventureWorks2012  
SELECT * FROM Person.Address a  
JOIN Person.StateProvince s  
ON a.StateProvinceID = s.StateProvinceID;
```

Выведем графический план запроса для заданной инструкции.

В тестовой базе данных AdventureWorks найдите таблицу с большим количеством строк (от 10000) и подходящим индексом (если его нет, создайте). Постройте диаграмму статистики по индексу и покажите случаи с высокой и низкой селективностью выборки. Все получаемые диаграммы следует сохранить в виде скриншотов, после чего вставить в лист отчета о практике, также необходимо подписать используемые листинги запросов.

Покажите план выполнения запроса для выборки с высокой селективностью и объясните, почему выбран именно такой план реализации запроса. Повторите тоже самое с выборкой с низкой селективностью

Содержание итогового отчета по практике.

Отчет должен содержать следующее:

1. Скриншот гистограммы статистики индексов с указаниями запросов с высокой и низкой селективностью.
2. План выполнения запроса для выборки с высокой селективностью с текстовыми пояснениями.
3. План выполнения запроса с низкой селективностью с текстовыми пояснениями.

Лист отчета о практике 2. Выполнил:

1.3. Практическая работа №3. Мониторинг активности и производительности MS SQL Server

Факторы, влияющие на производительность базы данных, должны находиться под постоянным контролем со стороны администратора.

Для отслеживания текущего состояния производительности системы применяются *графический инструмент мониторинга (Монитор ресурсов)* и *специальные представления*, формируемые автоматически в системных базах данных.

Вызов утилиты *Монитор ресурсов*: Панель управления -> Система и безопасность -> Администрирование -> Монитор ресурсов.

С помощью данной утилиты можно мониторить четыре вида ресурсов системы: нагрузку на центральный процессор, использование памяти, использование диска и нагрузку на сеть (рис. 6).

Также для отслеживания и записи деятельности базы данных и сервера БД, такие как соединение, пользование, информация из приложений используется графический инструмент MS SQL Server - SQL Server Profiler (рис. 7) (SQL Server Profiler доступен начиная с 17-ой версии MS SQL Server).

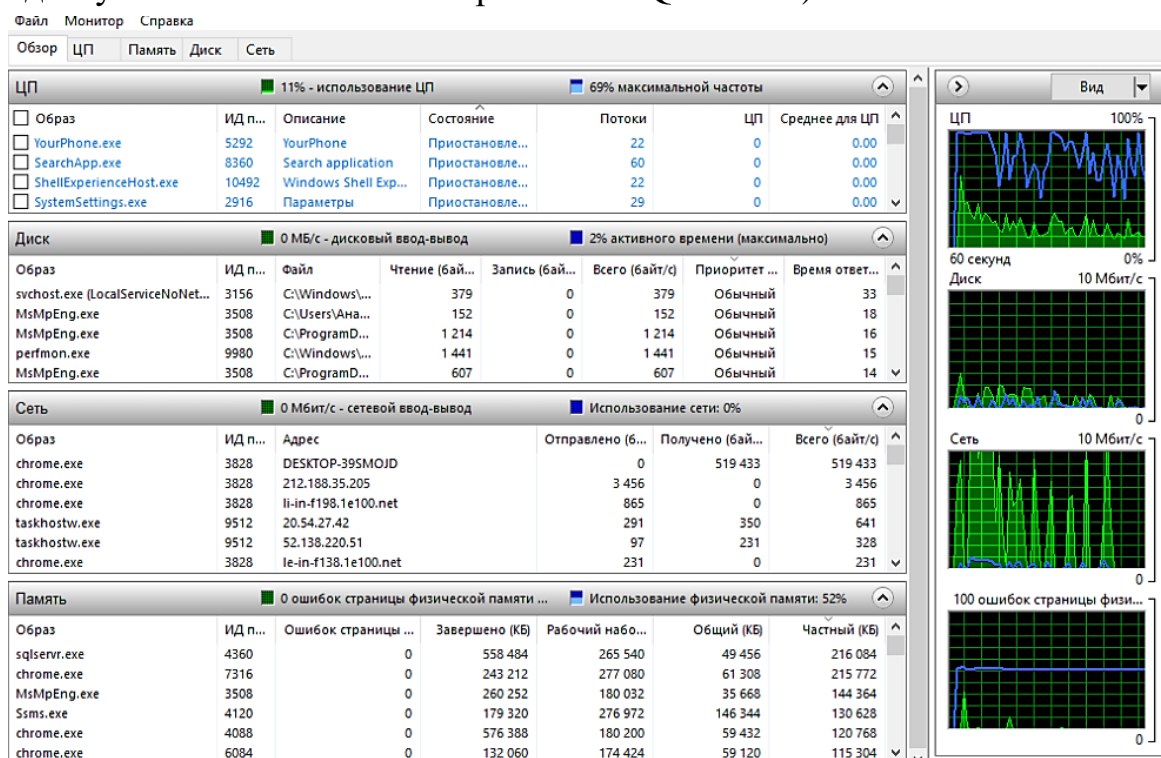


Рисунок 6. Общий вид утилиты Монитор ресурсов

Свойства трассировки

Общие | Выбор событий

Имя трассировки:

Имя поставщика трассировки:

Тип поставщика трассировки: версия:

Использовать шаблон:

☐ Сохранить в файл:

Установить максимальный размер файла (МБ):

☒ Включить операцию переключения на файл продолжения

☐ Сервер обрабатывает данные трассировки

☐ Сохранить в таблицу:

☐ Максимальное число строк (тыс.):

☐ Включить время остановки трассировки:

☒ Задать продолжительность трассировки (в минутах):

Рисунок 7. Создание трассировки в SQL Server Profiler

Свойства трассировки

Общие | Выбор событий

Обзор выбранных для трассировки событий и столбцов событий. Чтобы увидеть полный список событий, выберите параметр "Показать все события", а затем параметр "Показать все столбцы".

События	TextData	ApplicationName	NTUserName	LoginName	CPU	Reads	Writes	Duration	ClientProcess
Security Audit									
<input checked="" type="checkbox"/> Audit Login	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>					<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/> Audit Logout		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Sessions									
<input checked="" type="checkbox"/> ExistingConnection	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>				<input type="checkbox"/>	<input checked="" type="checkbox"/>
Stored Procedures									
<input checked="" type="checkbox"/> RPC:Completed	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
TSQL									
<input checked="" type="checkbox"/> SQL:BatchCompleted	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/> SQL:BatchStarting	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>					<input checked="" type="checkbox"/>

Security Audit
Содержит классы событий, используемые для аудита активности сервера.

☐ Показать все события

☐ Показать все столбцы

Столбец данных не выбран.

Рисунок 8. Создание трассировки в SQL Server Profiler

Данное ПО позволяет отображать информацию о различных активностях сервера или создавать фильтры, собирающие информацию об отдельных событиях пользователя, типах команд или инструкций SQL (рис. 8).

При настройке трассировки можно назначить определенные фильтры. Для этого необходимо в окне “Свойства трассировки” выбрать “Фильтры столбцов” и указать необходимую группу фильтров (табл. 2).

Таблица 2. Фильтры столбцов Profiler

Наименование	Описание
ApplicationName	Имя клиентского приложения, установившего соединение с экземпляром SQL Server. Этот столбец заполняется не отображаемым именем программы, а значениями, которые передаются приложением.
BinaryData	Значение типа Binary, зависящие от класса событий, фиксируемых при трассировке
ClientProcessID	Идентификатор процесса приложения, вызывающего SQL Server.
CPU	Время ЦП (в миллисекундах), использованное событием
Duration	Количество времени (в миллисекундах), использованного событием.
EndTime	Время окончания события, если оно известно. Этот столбец не заполняется для начальных классов событий, таких как SQL:BatchStarting или SP:Starting.
LoginName	Имя для входа пользователя (либо имя для входа безопасности SQL Server, либо учетные данные для входа Windows в формате ДОМЕН\Имя_пользователя).
NTUserName	Имя пользователя Windows.
Reads	Число логических чтений диска, выполненных сервером от имени события.
SPID	Идентификатор серверного процесса, который SQL Server назначил процессу, связанному с клиентской программой.
StartTime	Время начала события, если известно.
TextData	Текстовое значение, зависящее от класса события, в котором было зафиксирован при трассировке.
Writes	Число операций записи на физический диск, выполненных сервером от имени события.

Также функционал SQL Server Profiler позволяет отслеживать следующие события:

- соединения, попытки соединения, ошибки соединения, отключение

- использование пакетов центрального процессора
- проблемы взаимных блокировок
- все операторы DML
- запуск и завершение работы хранимой процедуры.

Также возможен перехват деятельности, связанных с запросами с передачей в дальнейшую оптимизацию (Database Engine Tuning Advisor) (рис. 9).

EventClass	TextData	ApplicationName	NTUserName	LoginName	CPU	Reads	Writes	Duration	ClientProcessID	SPID	StartTime	EndTime
Trace Start											2021-05-13 18:25:41...	
ExistingConnection	-- network protocol: LPC set quoted...	SQLServerCEP	SQLTELE...	NT SER...					4832	51	2021-05-13 18:21:06...	
ExistingConnection	-- network protocol: LPC set quoted...	Microsoft SQL...	Анастасия	DESKTO...					10060	53	2021-05-13 18:10:04...	
ExistingConnection	-- network protocol: LPC set quoted...	Среда Micros...	Анастасия	DESKTO...					10060	58	2021-05-13 18:15:28...	
SQL:BatchStarting	SELECT @@SPID;	Среда Micros...	Анастасия	DESKTO...					10060	58	2021-05-13 18:25:46...	
SQL:BatchCompleted	USE Adventureworks2014; SELECT La...	Среда Micros...	Анастасия	DESKTO...	0	0	0	0	10060	58	2021-05-13 18:25:46...	2021-05-13 18:25:46...
SQL:BatchStarting	USE Adventureworks2014; SELECT La...	Среда Micros...	Анастасия	DESKTO...					10060	58	2021-05-13 18:25:46...	
SQL:BatchCompleted	USE Adventureworks2014; SELECT La...	Среда Micros...	Анастасия	DESKTO...	78	2728	0	80	10060	58	2021-05-13 18:25:46...	2021-05-13 18:25:46...
Audit Logout		SQLServerCEP	SQLTELE...	NT SER...	0	250	0	295660	4832	51	2021-05-13 18:21:06...	2021-05-13 18:26:02...
Audit Login	-- network protocol: LPC set quoted...	SQLServerCEP	SQLTELE...	NT SER...					4832	51	2021-05-13 18:26:06...	
SQL:BatchStarting	SET DEADLOCK_PRIORITY -10	SQLServerCEP	SQLTELE...	NT SER...					4832	51	2021-05-13 18:26:06...	
SQL:BatchCompleted	SET DEADLOCK_PRIORITY -10	SQLServerCEP	SQLTELE...	NT SER...	0	0	0	0	4832	51	2021-05-13 18:26:06...	2021-05-13 18:26:06...
SQL:BatchStarting	SELECT target_data FROM sy...	SQLServerCEP	SQLTELE...	NT SER...					4832	51	2021-05-13 18:26:06...	
SQL:BatchCompleted	SELECT target_data FROM sy...	SQLServerCEP	SQLTELE...	NT SER...	110	0	0	119	4832	51	2021-05-13 18:26:06...	2021-05-13 18:26:07...
Audit Logout		SQLServerCEP	SQLTELE...	NT SER...	110	0	0	247	4832	51	2021-05-13 18:26:06...	2021-05-13 18:26:07...
RPC:Completed	exec sp_reset_connection	SQLServerCEP	SQLTELE...	NT SER...	0	0	0	0	4832	51	2021-05-13 18:26:07...	2021-05-13 18:26:07... OK
Audit Login	-- network protocol: LPC set quoted...	SQLServerCEP	SQLTELE...	NT SER...					4832	51	2021-05-13 18:26:07...	
SQL:BatchStarting	SET DEADLOCK_PRIORITY -10	SQLServerCEP	SQLTELE...	NT SER...					4832	51	2021-05-13 18:26:07...	
SQL:BatchCompleted	SET DEADLOCK_PRIORITY -10	SQLServerCEP	SQLTELE...	NT SER...	0	0	0	0	4832	51	2021-05-13 18:26:07...	2021-05-13 18:26:07...
SQL:BatchStarting	if not exists (select * from sys.dm...	SQLServerCEP	SQLTELE...	NT SER...					4832	51	2021-05-13 18:26:07...	
SQL:BatchCompleted	if not exists (select * from sys.dm...	SQLServerCEP	SQLTELE...	NT SER...	0	20	0	1	4832	51	2021-05-13 18:26:07...	2021-05-13 18:26:07...


```

USE Adventureworks2014;
SELECT LastName, FirstName
FROM Person.Person c
INNER JOIN HumanResources.Employee e
ON c.BusinessEntityID = e.BusinessEntityID
JOIN Sales.Salesperson s
ON e.BusinessEntityID = s.BusinessEntityID;

SELECT Name, ProductNumber
FROM Production.Product
WHERE ProductNumber Like 'FW%' and ProductSubcategoryID IN
(SELECT ProductSubcategoryID
FROM Production.ProductSubcategory
WHERE Name = 'Wheels');

SELECT SUM(UnitPrice) AS TotalPrice, ProductID
FROM Sales.SalesOrderDetail WHERE CarrierTrackingNumber Like '4E0AN'
GROUP BY ProductID ORDER BY ProductID ASC;

SELECT MAX(UnitPrice) AS MaxPrice, ProductID
FROM Sales.SalesOrderDetail GROUP BY ProductID ORDER BY MaxPrice DESC;

```

Рисунок 9. Пример работы MS SQL Profiler.

Задание для самостоятельного выполнения.

Все дальнейшие действия с вызовом представлений с использованием запросов должны быть зафиксированы в листе отчета о практике.

Для отображения результатов мониторинга текущих процессоров БД используется системные представления sys.sysprocesses и sys.dm_exec_requests.

Представление sys.dm_exec_requests (рис. 10) хранит информацию о сессиях конкретных пользователей и данные об использовании этими пользователями центрального процессора. Служит для анализа использования центрального процессора.

```

USE Master;

SELECT session_id, database_id, user_id, cpu_time, sql_handle
FROM sys.dm_exec_requests ORDER BY cpu_time DESC;

```


Результаты		Сообщения			
	session_id	database_id	user_id	cpu_time	sql_handle
1	12	0	1	3953	NULL
2	25	1	1	422	NULL
3	7	0	1	140	NULL
4	8	0	1	109	NULL
5	1	0	1	31	NULL
6	10	0	1	31	NULL
7	30	1	1	15	NULL
8	31	1	1	15	NULL
9	32	1	1	15	NULL
10	5	0	1	15	NULL
11	68	1	1	5	0x0300FF7FFF7AAEE1F148EC00D2AA000001000000000000...
12	52	1	1	2	0x020000001D8BB72C10E8ADFFA817E344246120CB840277...
13	53	1	1	0	NULL
14	54	1	1	0	NULL
15	55	1	1	0	NULL
16	11	0	1	0	NULL
17	33	1	1	0	NULL
18	34	1	1	0	NULL
19	45	1	1	0	NULL
20	49	1	1	0	NULL
21	26	1	1	0	NULL
22	27	1	1	0	NULL

Рисунок 10. Результат запроса к представлению sys.dm_exec_requests

Для отображения результатов анализа использования памяти БД используется системное представление sys.dm_os_memory_objects (рис. 11). Это представление хранит информацию об объектах памяти, распределенных по разным типам. Используется для анализа использования памяти и для анализа возможного недостатка памяти.

```
USE Master;
SELECT type, SUM (pages_in_bytes) AS total_memory
FROM sys.dm_os_memory_objects GROUP BY type
ORDER BY total_memory DESC;
```

	type	total_memory
1	MEMOBJ_SOSNODE	34430976
2	MEMOBJ_PARSE	32620544
3	MEMOBJ_COMPILE_ADHOC	18374656
4	MEMOBJ_SQLCLR_CLR_EE	17252352
5	MEMOBJ_METADATADB	10739712
6	MEMOBJ_GLOBALPMO	10452992
7	MEMOBJ_EXECUTE	7643136
8	MEMOBJ_PERDATABASE	6553600
9	MEMOBJ_RESOURCE	4333568
10	MEMOBJ_STATEMENT	3211264
11	MEMOBJ_QUERYEXECNTXTFORSE	3031040
12	MEMOBJ_LOGPOOL	2736128
13	MEMOBJ_SYSTEMROWSET	2605056
14	MEMOBJ_SOSDEADLOCKMONITORRINGBUFFER	2138112
15	MEMOBJ_SQLMGR	1826816
16	MEMOBJ_XTPBLOCKALLOC	1802240
17	MEMOBJ_SOSSCHEDULER	1785856

Рисунок 11. Результат запроса к представлению sys.dm_os_memory_objects

Для отображения результатов мониторинга дисковой системы БД используется системное представление sys.dm_os_wait_stats (рис. 12). Это представление возвращает информацию об ожиданиях в выполняемых потоках инструкций и пакетов. Ключевыми столбцами представления являются wait_type (имена типов ожидания) и waiting_tasks_count (количество ожиданий соответствующего типа).

```
USE Master;
SELECT * FROM sys.dm_os_wait_stats
ORDER BY waiting_tasks_count DESC;
```

	wait_type	waiting_tasks_count	wait_time_ms	max_wait_time_ms	signal_wait_time_ms
1	MEMORY_ALLOCATION_EXT	260354	586	4	0
2	SOS_WORK_DISPATCHER	51075	126830363	1442453	5282
3	PREEMPTIVE_XE_CALLBACKEXECUTE	42866	62	1	0
4	ONDEMAND_TASK_QUEUE	40686	1256637	990879	796
5	LOGMGR_QUEUE	32556	6706634	804222	304
6	SLEEP_TASK	11498	8084129	805125	581
7	PAGEIOLATCH_SH	5155	17208	608	19
8	HADR_FILESTREAM_IOMGR_IOCOMPLETION	4980	3343802	804558	79
9	RESERVED_MEMORY_ALLOCATION_EXT	3682	3	0	0
10	LAZYWRITER_SLEEP	2581	3353182	804475	90
11	ASYNC_NETWORK_IO	2406	891	138	26
12	SOS_SCHEDULER_YIELD	2201	41	11	39
13	PREEMPTIVE_OS_QUERYREGISTRY	1234	128	30	0
14	BROKER_TO_FLUSH	1232	2073535	804348	32
15	PARALLEL_REDO_WORKER_WAIT_WORK	1036	14681	40	31
16	PREEMPTIVE_OS_FILEOPS	676	1315	21	0
17	PREEMPTIVE_OS_AUTHENTICATIONOPS	557	771	644	0
18	REQUEST_FOR_DEADLOCK_SEARCH	510	3352907	809059	3352907

Рисунок 12. Результат запроса к представлению sys.dm_os_wait_stats

Для отображения результатов мониторинга сетевого интерфейса БД

используется системное представление `sys.dm_exec_connections` (рис. 13). Это представление возвращает информацию соединениях, установленных с ядром БД, и детали каждого соединения. Ключевыми столбцами представления являются `num_reads` (количество прочитанных пакетов в данном соединении) и `num_writes` (количество записанных пакетов в данном соединении).

```
USE Master;
SELECT session_id, num_reads, num_writes, last_read, last_write
FROM sys.dm_exec_connections;
```

	session_id	num_reads	num_writes	last_read	last_write
1	51	9	475	2021-06-01 13:37:45.070	2021-06-01 13:37:45.073
2	68	67	125	2021-06-01 13:35:43.737	2021-06-01 13:39:20.403
3	64	65	67	2021-06-01 13:22:24.497	2021-06-01 13:22:24.497
4	52	48	107	2021-06-01 13:39:20.407	2021-06-01 13:39:20.403

Рисунок 13. Результат запроса к представлению `sys.dm_exec_connections`

Содержание итогового отчета по практике.

Отчет должен содержать все результаты выполнения запросов к системным представлениям, представленные в виде скриншотов с краткими текстовыми пояснениями.

Лист отчета о практике 3. Выполнил:

2. ЛАБОРАТОРНЫЕ РАБОТЫ

2.1. Лабораторная работа №1. Перепроектирование физической модели базы данных

Ниже приведен скрипт перепроектирования физической модели базы данных, содержащей три таблицы, триггер и представление. Подлежит переименовать одну из таблиц, на основании данных которой строится представление, в которой присутствует триггер и которая является родителем для внешних ключей двух других таблиц. Скрипт с процедурой создания физической модели данных:

```
CREATE TABLE work (WorkID int NOT NULL, Title char (35), Description  
varchar (1000), ArtistID int);
```

```
CREATE TABLE artist (ArtistID int NOT NULL, LastName char (25), FirstName  
char (25), Nationality char (30), DateofBirth decimal (4,0), DateDeceased  
decimal (4,0));
```

```
ALTER TABLE artist  
ADD CONSTRAINT ArtistPK PRIMARY KEY (ArtistID);  
ALTER TABLE WORK  
ADD CONSTRAINT WorkPK PRIMARY KEY (WorkID),
```

```
CONSTRAINT ArtistFK FOREIGN KEY (ArtistID) REFERENCES artist (ArtistID);  
INSERT INTO artist VALUES  
(001, 'Айвазовский', 'Иван', 'русский', 1817, 1900);  
BEGIN TRANSACTION  
INSERT INTO work VALUES  
(001, 'Севастополь', 'На картине изображена бухта города в утренней дымке', 001),  
(002, 'Шхуна под парусами', 'На картине изображена шхуна с флагом', 001),  
(003, 'Ялта. Горы ночью', 'Группа людей прогуливается вдоль берега Крыма', 001);
```

```
IF (@@error <> 0)  
ROLLBACK  
ELSE  
COMMIT;
```

```
ALTER TABLE WORK  
ADD Sold char (4) NULL;
```

```
CREATE TABLE Sell (SellID int NOT NULL, WorkID int,  
CONSTRAINT WorkFK FOREIGN KEY (WorkID)  
REFERENCES work (WorkID));
```

```
CREATE TRIGGER SellWork  
ON Sell  
AFTER INSERT  
AS  
BEGIN  
DECLARE @SellWorkID int  
SELECT @SellWorkID = (SELECT WorkID FROM inserted)  
UPDATE WORK  
SET Sold = 'Sold'  
WHERE WorkID = @SellWorkID
```

END

```
INSERT INTO Sell VALUES  
(001, 001);
```

```
CREATE VIEW auction AS SELECT Title, Description, ArtistID  
FROM WORK  
WHERE Sold IS NULL;
```

Задание для самостоятельного выполнения.

Создайте файл lab_1_<фамилия_группа>. Используя приведенный скрипт T-SQL, создайте физическую модель базы данных. Постройте график зависимости для исследуемой модели. Затем перепроектируйте базу данных, заменив в физической модели таблицу **work** на таблицу **work2**. В ходе выполнения процедур перепроектирования необходимо связать между таблицами **Artist** и **Work2** сделать идентификационно – зависимой; созданный триггер и представление привязать к новой таблице **Work2**, а таблицу **Work** удалить. Полученный скрипт проверьте в СУБД, в случае его работоспособности, перенесите скрипт с комментариями в лист отчета по лабораторной работе 1.

Содержание итогового отчета по лабораторной работе.

1. График зависимости для приведенной в скрипте работы физической модели данных.
2. Полный скрипт перепроектирования модели с заданными условиями.
3. Текстовые пояснения к приведенному скрипту с перепроектированием.

Лист отчета о лабораторной работе 1. Выполнил:

2.2. Лабораторная работа №2. Создание файлов базы данных с помощью языка T-SQL

Ниже приведены ключевые скрипты T-SQL, используемые для создания и изменения файлов и файловых групп баз данных. Внимательно изучите приведенные образцы.

```
/** T-SQL скрипт создания базы данных и журнала транзакций **/

USE master;
CREATE DATABASE // Имя базы данных
ON
( NAME = // Логическое наименование БД,
FILENAME = // 'Путь до файла, где будет храниться основная информация. Обратите
внимание, что файл имеет расширение mdf. ',
SIZE = // Начальный размер БД,
MAXSIZE = // Максимальный размер БД,
FILEGROWTH = // Прирост базы данных. Может указываться
конкретным размером данных или процентом. Например, 10MB или 10%)

/** T-SQL скрипт создания журнала транзакций **/

LOG ON
( NAME = // Логическое имя журнала. Обычно создается в формате Имя базы данных_log,
FILENAME = // 'Путь до журнала транзакций (Журнал содержится в той же папке, что и
основной файл базы данных). Файл имеет расширение ldf.',
SIZE = // Размер журнала транзакций,
MAXSIZE = UNLIMITED // (Максимальный размер журнала традиционно устанавливается
UNLIMITED),
FILEGROWTH = // Значение прироста);

/** T-SQL скрипт создания файловой группы **/

USE // Имя БД, где будет создаваться файловая группа;
ALTER DATABASE // Имя БД
ADD FILEGROUP // Название файловой группы;

/** T-SQL скрипт создания пользовательских файлов **/

USE master;
ALTER DATABASE // Имя БД
ADD FILE ( NAME = // Логическое имя файла,
FILENAME = // 'Путь до файла с расширением .ndf',
SIZE = // Начальный размер,
MAXSIZE = // Максимальный размер,
FILEGROWTH = // Прирост файла
)
TO FILEGROUP // Название файловой группы, куда будет помещен файл;
/** T-SQL скрипт установления файловой группы по умолчанию **/

ALTER DATABASE // Имя БД
```

MODIFY FILEGROUP // Имя файловой группы
DEFAULT

В начале работы по умолчанию установлена файловая группа PRIMARY. Основной файл базы данных создается именно в группе PRIMARY, файл журнала транзакций не имеет файловой группы.

Задание для самостоятельного выполнения.

Ориентируясь на приложенные в таблице 3 параметры, напишите скрипт на языке T-SQL, создающий базу данных с заданными параметрами файлов.

Таблица 3. Требования к файлам

Файл	Требование
Имя базы данных	RateTracking
Первичный файл базы данных	Логическое имя - RateTracking_dat Имя файла - RateTracking.mdf Путь к папке - на усмотрение студента Начальный размер - 10MB Максимальный размер - 100MB Прирост - 10MB Файловая группа - PRIMARY
Файл журнала транзакций	Логическое имя - RateTracking_log Имя файла - RateTracking.ldf Путь к папке - на усмотрение студента Начальный размер - 20MB Максимальный размер - UNLIMITED Прирост - 20MB Файловая группа - Not Applicable
Пользовательский файл #1	Логическое имя - RateTracking_dat_1 Имя файла - RateTracking.ndf Путь к папке - на усмотрение студента Начальный размер - 20MB Максимальный размер - 100MB Прирост - 10MB Файловая группа - USERDATA
Пользовательский файл#2	Логическое имя - RateTracking_dat_2 Имя файла - RateTracking.ndf Путь к папке - на усмотрение студента Начальный размер - 20MB Максимальный размер - 100MB Прирост - 10MB Файловая группа - USERDATA

Пользовательский файл#3	Логическое имя - RateTracking_dat_3 Имя файла - RateTracking.ndf Путь к папке - на усмотрение студента Начальный размер - 200MB Максимальный размер - 500MB Прирост - 50MB Файловая группа - ARCHIVE
Пользовательский файл#4	Логическое имя - RateTracking_dat_4 Имя файла - RateTracking.ndf Путь к папке - на усмотрение студента Начальный размер - 200MB Максимальный размер - 500MB Прирост - 50MB Файловая группа - ARCHIVE
Файловая группа по - умолчанию	USERDATA

Содержание итогового отчета по практике.

Полный скрипт выполненного задания в языке t-sql.

Лист отчета о лабораторной работе 2. Выполнил:

2.3. Лабораторная работа №3. Резервное копирование базы данных в среде MS SQL Server

Резервное копирование базы данных - процесс создания файла копии данных на носителях, предназначенных для восстановления данных в случае их повреждения или разрушения. Необходимо обратить внимание на то, что резервному копированию в среде MS SQL Server подлежит только элемент “база данных”.

В зависимости от того, какую информацию из базы данных необходимо скопировать, можно использовать три основных пути:

- полная резервная копия;
- дифференцированная резервная копия;
- резервная копия журнала транзакций.

Полная резервная копия - создание полной копии всех данных БД. Время создание такой копии напрямую зависит от объема копируемой базы данных, чем больше база, тем большее время займет снятие ее копии. Это порождает следующую проблему: такую копию необходимо снимать в периоды, когда нагрузка на базу данных отсутствуем или минимальна. Обычно такая копия снимается раз в две недели, при необходимости это делается раз в неделю.

Дифференцированная резервная копия - создание копий данных, которые были внесены в базу между снятиями полной копии. Это делается для того, чтобы, при поломке базы данных в период между снятиями полной резервной копии, внесенные данные не были утрачены. Дифференцированная резервная копия требует меньше времени для снятия, поэтому обычно ее создают раз в день в конце рабочего дня для того, чтобы максимально исключить потерю данных.

Резервная копия журнала транзакций - создание копии данных, которые были внесены в базу между снятиями дифференцированной резервной копии. Такие копии предназначены для того, чтобы при поломке базы в течение дня, внесенные данные не были утеряны. Как правило, копию журнала транзакций снимается один раз в несколько часов или несколько раз за час, все зависит от объема вносимых данных. Маленький объем журнала транзакций позволяет создавать копию максимально быстро.

Обратите внимание: файлы полной и дифференцированной копии имеют разрешение “bak”, а файл копии журнала транзакций “trn”.

Типовой код создания полной и резервной копии баз данных показаны на скриптах ниже.

```
/** T-SQL скрипт создания полной резервной копии базы данных */
```

```
DECLARE @backupLocationFull varchar(200)
SELECT @backupLocationFull= // 'Путь до папки, где будет храниться файл полной резервной копии'
+ REPLACE(convert(nvarchar(20),GetDate(),120),':','-') + '.bak'
// Для удобства восстановления базы на файл ставится метка с датой и временем
BACKUP DATABASE // Имя базы данных
TO DISK = @backupLocationFull
WITH NOFORMAT, NOINIT, NAME = // 'Наименование файла полной копии базы данных ',
SKIP, NOREWIND,
NOUNLOAD, STATS = 5;
```

```
/** T-SQL скрипт создания дифференцированной резервной копии базы данных */
```

```
DECLARE @backupLocationDiff varchar(200)
SELECT @backupLocationDiff= // 'Путь до папки, где будет храниться дифференцированная копия базы
данных'
+ REPLACE(convert(nvarchar(20),GetDate(),120),':','-') + '.bak'
BACKUP DATABASE // Имя базы данных
TO DISK = @backupLocationDiff
WITH DIFFERENTIAL, NOFORMAT, NOINIT, NAME = // 'Наименование файла дифференцированной
копии',
SKIP, NOREWIND, NOUNLOAD, STATS = 5;
```

Копию журнала транзакций **невозможно** будет снять, если у базы данных установлена **модель восстановления “Простая”**. Для того, чтобы создать копию необходимо перевести базу в режим **“Полная”** или **“С неполным протоколированием”**.

Изменить модель можно по следующему пути: *Свойства базы данных-> Параметры-> Модель восстановления* (рис. 14).

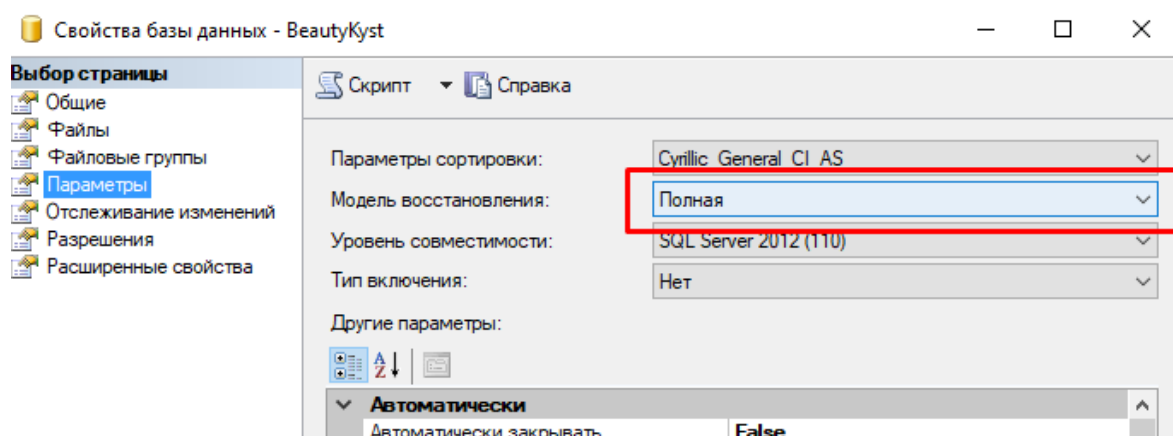


Рисунок 14. Изменение модели восстановления БД

Ниже будет приведен типовой скрипт создания журнала транзакции в языке T-SQL.

```

DECLARE @backupLocationLog varchar (200)
SELECT @backupLocationLog= // 'Путь до папки, где будет храниться копия журнала транзакций'
+ REPLACE(convert(nvarchar(20),GetDate(),120),':','-') + '.trn'
BACKUP LOG Имя базы данных TO DISK = @backupLocationLog WITH NOFORMAT,
NOINIT, NAME = // 'Наименование файла копии журнала транзакций', SKIP, NOREWIND, NOUNLOAD, STATS
= 5;

```

Отдельно рассмотрим команды, используемые при создании резервной копии.

WITH NOFORMAT - определяет, что при текущей операции резервного копирования существующие заголовки носителей и резервные наборы данных сохраняются на томах носителей, используемых для текущей операции резервного копирования. Это поведение по умолчанию.

NOINIT - указывает, что резервный набор данных дозаписывается на заданный набор носителей с сохранением уже существующих резервных наборов данных. Если для набора носителей задан пароль носителя, то этот пароль должен быть предоставлен. NOINIT является значением по умолчанию.

SKIP - отключает проверку сроков действия и имен резервных наборов данных, приводящуюся обычно с помощью инструкции BACKUP для предотвращения перезаписи.

NOREWIND - указывает, что SQL Server сохранит ленту открытой после операции резервного копирования. С помощью этого параметра можно улучшить производительность при выполнении нескольких операций резервного копирования на ленту.

NOUNLOAD - Указывает, что после завершения операции BACKUP лента остается в ленточном накопителе.

STATS - делит процесс снятия копии на фрагменты, для того чтобы можно было отследить весь процесс. В приведенном выше скрипте выводит сообщение об обработке каждых 5% информации.

Восстановление базы данных происходит по сценарию, похожему на резервное копирование, но в обратном порядке. Ниже приведен типовой листинг процедуры восстановления базы данных из ее резервной копии.

```

RESTORE DATABASE // [Имя базы данных]
FROM DISK = // 'Путь до файла резервной копии базы данных'
WITH FILE = 1,
NORECOVERY, NOUNLOAD, STATS = 5

```

Отдельно рассмотрим команды, используемые при восстановлении базы

данных.

NORECOVERY - указывает на то, что откат не выполняется. Это позволяет продолжать накат при помощи следующей инструкции в последовательности. В этом случае последовательность восстановления может восстановить другие резервные копии и выполнить их накат.

WITH FILE - номер резервного набора данных на устройстве.

Особое внимание при резервном копировании следует уделить планированию характеристик объекта копирования, на основе которых затем можно будет составить план резервного копирования (табл. 4).

Таблица 4. Планируемые характеристики объекта резервного копирования

Характеристики	Ожидаемое значение
Размер базы данных	1,5 - 2 ГБ
Совокупная скорость резервного копирования базы данных	40 МБ/с
Совокупная скорость восстановления базы данных	55 МБ/с
Средний уровень изменения базы данных за рабочее время (под нагрузкой)	1,5-2 МБ/д
Средняя доля изменения базы данных за рабочее время по отношению к общему размеру (под нагрузкой)	1%
Рабочее время (пн-пт)	9:00-18:00

Исходя из планируемых характеристик базы данных, формируется план резервного копирования (табл. 5).

Таблица 5. План резервного копирования

День недели	Время	Действия	Частота	Описание
Пн-Пт	22:00	Дифференциальная копия	1 раз в день	plumb
Сб	22:00	Полная копия	1 раз в неделю	plumb
Пн-Пт	10:00-18:00	Копия ЖТ	Каждый час	plumb
Сб	20:00	Резервная копия	1 раз в неделю	master
Сб	20:00	Резервная копия	1 раз в неделю	msdb

Задание для самостоятельного выполнения.

Используя скрипты T-SQL, создайте все три вида резервной копии базы данных AdventureWorks, удалите ее, а затем выполните восстановление, используя созданные копии

Содержание итогового отчета по практике.

1. Таблица планируемых характеристик базы данных и план резервного копирования для базы данных AdventureWorks 2019.
2. Скрипт выполненного задания в языке t-sql.
3. Скриншоты, подтверждающие выполнение задания (результаты выполнения запросов).

Лист отчета о лабораторной работе 3. Выполнил:

2.4. Лабораторная работа №4. Создание ролей и пользователей в базе данных

Роль – субъект безопасности базы данных, группирующий пользователей и предоставляющий им какие-либо права. В отличие от серверных ролей, которые могут быть только встроенными, роль в базе данных может быть как встроенная, так и пользовательская.

User (пользователь) – это идентификатор имени входа при подключении к базе данных. Имя пользователя базы данных может совпадать с именем входа, но это не является обязательным условием.

Для удобства управления разрешениями в базе данных MS SQL Server предоставляет несколько ролей.

Разрешения пользовательских ролей базы данных можно настроить с помощью инструкций GRANT, DENY и REVOKE. Разрешения можно предоставлять как на базу данных, так и на отдельные объекты базы: таблицы, хранимые процедуры, представления и т.д.

Далее будет приведен типовый скрипт создания роли в базе данных MS SQL Server.

```
/** T-SQL скрипт создания роли в базе данных **/
```

```
CREATE ROLE // Название роли;
```

```
/** T-SQL скрипт выдачи разрешения на действия ролям **/
```

```
GRANT // Выданные разрешения  
ON // Объект, на который выдается решение  
TO //Название роли;
```

```
/** T-SQL скрипт выдачи запретов на действия ролям **/
```

```
DENY // Запрещенные для роли разрешения  
ON // Название роли  
TO // Объект, на который устанавливается запрет
```

```
/** T-SQL скрипт отзыва разрешения **/
```

```
REVOKE // Выданные разрешения, которые необходимо отозвать  
ON // Название роли  
FROM // Объект, на который были выданы разрешения
```

Имя входа — это субъект безопасности, с помощью которого система безопасности может проверить подлинность лица. Логин необходим пользователю для соединения с SQL Server. Для подключения к определенной базе данных на экземпляре SQL Server имя входа должно быть сопоставлено с пользователем базы данных.

Ниже приведен листинг скрипта создания логина и пароля для пользователя базы данных.

```

/** T-SQL скрипт создания логина и пароля для пользователя */

CREATE LOGIN // Логин, который будет использован для подключения к базе данных
WITH PASSWORD = // 'Пароль для входа';

/** T-SQL скрипт создания пользователя для логина */

CREATE USER // Имя пользователя
FOR LOGIN // Логин, который будет использован для подключения к базе данных;

/** T-SQL скрипт прикрепления пользователя к роли */

ALTER ROLE // Наименование роли, к которой будет прикреплен пользователь
ADD MEMBER // Имя пользователя;

```

Закрепление пользователей за определенными ролями позволит предоставить разрешения один раз вместо того, чтобы выдавать разрешения каждому отдельному пользователю, подключенному к базе данных.

Перед созданием ролей и пользователей базы данных, необходимо сформировать к ним требования. Они формируются на основе сценариев использования базы данных. Для удобства требования можно оформить в таблицу (табл. 5).

Таблица 5. Таблица требований к ролям и пользователям базы данных “*plumb*”

	Менеджеры (роль)	Рабочие склада (роль)	Системный администратор (роль)
Supply (T)	Запрос	Вставка, изменение, запрос	Вставка, изменение, запрос, удаление, предоставление прав, модификация структуры элемента
Supplier (T)	Вставка, изменение, запрос	Запрос	Вставка, изменение, запрос, удаление, предоставление прав, модификация структуры элемента
Product (T)	Вставка, изменение, удаление, запрос	Вставка, изменение, удаление, запрос	Вставка, изменение, запрос, удаление, предоставление прав, модификация структуры элемента
Client (T)	Вставка, изменение, запрос	Запрос	Вставка, изменение, запрос, удаление, предоставление прав, модификация структуры элемента
Employee (T)	Запрос	Запрос	Вставка, изменение, запрос, удаление, предоставление прав, модификация структуры элемента

Order (V)	Вставка, изменение, запрос	Запрос	Вставка, изменение, запрос, удаление, предоставление прав, модификация структуры элемента
Delete_Order (SP)	Выполнение	Нет	Выполнение, изменение, удаление, предоставление прав, модификация структуры элемента

*/** Образец оформления T-SQL скриптов создания ролей **/*

```
CREATE ROLE managers;
GRANT SELECT ON Supply TO managers;
GRANT INSERT, UPDATE, SELECT ON Supplier TO managers;
GRANT DELETE, INSERT, UPDATE, SELECT ON Product TO managers;
GRANT INSERT, UPDATE, SELECT ON Client TO managers;
GRANT SELECT ON Employee TO managers;
GRANT INSERT, UPDATE, SELECT ON Order TO managers;
GRANT EXECUTE ON SP_Delete_Order TO managers;
```

*/** Образец оформления T-SQL скриптов создания логинов и пользователей **/*

```
CREATE LOGIN plumb WITH PASSWORD = 'plumb';
CREATE USER plumb_manager FOR LOGIN plumb;
ALTER ROLE managers ADD MEMBER plumb_manager;
```

Задание для самостоятельного выполнения.

Оформить в соответствии с приведенным образцом таблицу требований к ролям и пользователям, а также сформировать скрипты создания ролей и пользователей с определенными в рамках таблицы требований доступами для базы данных по выбранной для курсовой работы теме.

Содержание итогового отчета по практике.

1. Таблица требований к ролям и пользователям базы данных
2. Скрипт на языке t-sql с выполненным заданием

Лист отчета о лабораторной работе 4. Выполнил:

Список литературы

1. Бегг К., Коннолли Т. Базы данных. Проектирование, реализация и сопровождение. Теория и практика. Пер. с англ. - М.: Вильямс, 2017. - 1440 с.: ил..
2. Kroenke D.M., Auer D.J. Database Processing: Fundamentals, Design and Implementation. PEARSON, 2019. - 691 с.: ил..
3. Кренке Д. Теория и практика построения баз данных - Спб.: Питер, 2005. - 859 с.: ил..
4. Кузнецов С. Базы данных: Модели и языки. - Мск.: Бином, 2008. - 720 с.: ил..
5. Kroenke D.M., Auer D.J. Database Concepts. англ. - PEARSON, 2018. - 579 с.: ил..
6. Моргунов Е.П. PostgreSQL. Основы языка SQL. - Спб.: БХВ-Петербург, 2021. - 335 с.:ил..
7. Петкович Д. Microsoft SQL Server 2012. Руководство для начинающих. - Спб.: БХВ-Петербург, 2013. - 817 с.: ил..

Сведения об авторах

Смирнов Михаил Вячеславович, кандидат экономических наук, доцент кафедры “Предметно-ориентированные информационные системы” Института комплексной безопасности и специального приборостроения РТУ-МИРЭА.

Горбатюк Ангелина Павловна, магистр кафедры “Предметно-ориентированные информационные системы” Института комплексной безопасности и специального приборостроения РТУ-МИРЭА.

Коровкина Анастасия Борисовна, магистр кафедры “Предметно-ориентированные информационные системы” Института комплексной безопасности и специального приборостроения РТУ-МИРЭА.