

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ  
«ЛИПЕЦКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

А.И. МИРОШНИКОВ

**АРХИТЕКТУРА СИСТЕМ УПРАВЛЕНИЯ  
БАЗАМИ ДАННЫХ**

Учебное пособие

Липецк

Липецкий государственный технический университет

2018

УДК 004.658  
М645

**Рецензенты:**

кафедра «Автоматизированные и информационные системы управления»  
Старооскольского технологического института им. А.А. Угарова (филиал)  
федерального государственного автономного образовательного учреждения высшего  
образования «Национальный исследовательский технологический университет  
«МИСиС» (канд. техн. наук, доц., Глущенко А.И.);  
Воробьёв Г.А., канд. техн. наук, доц. кафедры информатики, информационных  
технологий и защиты информации Липецкого государственного педагогического  
университета им. П.П. Семенова-Тян-Шанского

**Мирошников, А.И.**

М645 Архитектура систем управления базами данных [Текст]: учеб. пос. /  
А.И. Мирошников. – Липецк: Изд-во Липецкого государственного технического  
университета, 2018. – 94 с.

**ISBN 978-5-88247-879-6**

В пособии описываются компоненты и функциональные возможности современных систем управления базами данных, рассматриваются устройство индексов, оптимизатора запросов, системы безопасности, вопросы резервного копирования и восстановления.

Учебное пособие предназначено для направлений подготовки бакалавров и магистров, получающих углублённую подготовку по информационным технологиям. Оно также может оказаться полезным специалистам в области администрирования баз данных при решении прикладных задач, связанных с повышением производительности, обеспечением безопасности и поддержанию работоспособности реально функционирующих систем баз данных.

*Рекомендовано УМС ЛГТУ в качестве учебного пособия для бакалавров, обучающихся по направлениям подготовки ВПО 01.03.04 «Прикладная математика», 09.03.01 «Информатика и вычислительная техника», 02.03.03 «Математическое обеспечение и администрирование информационных систем», 09.03.04 «Программная инженерия», магистров по направлениям 01.04.04 «Прикладная математика», 09.04.01 «Информатика и вычислительная техника».*

УДК 004.658

Табл.: 7. Ил.: 64. Библиогр.: 12 наим.

**ISBN 978-5-88247-879-6**

© ФГБОУ ВО «Липецкий государственный  
технический университет», 2018  
© Мирошников А.И., 2018

## Оглавление

Введение .....	4
ГЛАВА 1. ОБЩИЕ СВЕДЕНИЯ .....	6
1.1. Основные понятия .....	6
1.2. Функции СУБД .....	7
1.3. Файловые системы .....	11
1.4. Преимущества и недостатки СУБД .....	13
ГЛАВА 2. АРХИТЕКТУРА СУБД MS SQL SERVER .....	14
ГЛАВА 3. ИНДЕКСЫ .....	20
3.1. Теоретическая часть .....	20
3.2. Лабораторная работа № 1 «Индексы» .....	28
3.3. Пример выполнения лабораторной работы № 1 .....	29
ГЛАВА 4. ОПТИМИЗАТОР ЗАПРОСОВ .....	38
4.1. Теоретическая часть .....	38
4.2. Лабораторная работа № 2 «Оптимизатор запросов» .....	47
4.3. Пример выполнения лабораторной работы № 2 .....	48
ГЛАВА 5. ПОЛЬЗОВАТЕЛИ И РОЛИ .....	58
5.1. Теоретическая часть .....	58
5.2. Лабораторная работа № 3 «Пользователи и роли. Представления. Процедуры» .....	62
5.3. Пример выполнения лабораторной работы № 3 .....	63
ГЛАВА 6. РЕЗЕРВНОЕ КОПИРОВАНИЕ И ВОССТАНОВЛЕНИЕ .....	70
6.1. Теоретическая часть .....	70
6.2. Лабораторная работа № 4 «Резервное копирование и восстановление» ..	79
6.3. Пример выполнения лабораторной работы № 4 .....	80
Заключение .....	92
Библиографический список .....	93

## **Введение**

В настоящее время трудно представить предприятие крупного или среднего бизнеса, которое бы не использовало базы данных при ведении своей деятельности. Объём этих данных может исчисляться десятками терабайт. Их использование позволяет быстро передавать информацию между отделами, защищать важные данные от конкурентов, автоматически создавать консолидированные отчёты.

В то же время большинство десктопных и мобильных приложений, не говоря уже о социальных сетях и интернет-магазинах, используют базы данных в своей работе, например, для хранения аккаунтов пользователей, списка дел в календаре или организации корзины товаров.

В последние годы верхние строчки рейтингов популярности среди систем управления базами данных (СУБД) занимают такие реляционные СУБД, как Oracle Database компании Oracle и Microsoft SQL Server компании Microsoft.

В данном учебном пособии архитектура реляционных систем управления базами данных рассматривается на примере СУБД Microsoft SQL Server. При описании примеров выполнения лабораторных работ используется учебная база данных AdventureWorks2012, представляющая собой результат имитации деятельности предприятия, занимающегося производством и торговлей велосипедами и аксессуарами к ним.

В первой главе приводятся общие аспекты СУБД: определения, функции, преимущества и недостатки, даётся описание файловых систем, являющихся историческими предшественниками СУБД.

Во второй главе представлена архитектура СУБД MS SQL Server, включающая в себя обзор основных компонентов и их функций.

Третья глава посвящена вопросам индексирования в базах данных. Показано значение индексных структур данных при выполнении запросов в базах данных, виды индексов в MS SQL Server, рекомендации по их

проектированию. Содержится задание к лабораторной работе № 1 «Индексы» и приводится пример её выполнения.

В четвёртой главе рассматривается такой компонент MS SQL Server, как оптимизатор запросов. Разбираются этапы обработки запросов, значение планов выполнения запросов, способов их оптимизации. Содержится задание к лабораторной работе № 2 «Оптимизатор запросов» и приводится пример её выполнения.

В пятой главе объясняется механизм безопасности ядра СУБД, включая использование схем и ролей пользователей, выполняющих представления и процедуры. Содержится задание к лабораторной работе № 3 «Пользователи и роли» и приводится пример её выполнения.

В шестой главе представлены методы резервного копирования и восстановления баз данных и журналов транзакций. Дается описание особенностей каждого метода и способов их реализации. Содержится задание к лабораторной работе № 4 «Резервное копирование и восстановление» и приводится пример её выполнения.

# ГЛАВА 1. ОБЩИЕ СВЕДЕНИЯ

## 1.1. Основные понятия

*База данных (БД)* с логической точки зрения – это набор упорядоченных данных, относящихся к определенной предметной области.

*База данных (БД)* с физической точки зрения – это файл или группа файлов на информационном носителе, содержащих упорядоченные данные в установленном формате.

*Система управления базами данных (СУБД)* – это программный комплекс, позволяющий создавать и администрировать базы данных, а также осуществлять к ним контролируемый доступ.

*Архитектура СУБД* – это модель, описывающая устройство отдельных компонентов среды СУБД, а также их взаимодействие друг с другом и с внешним программным обеспечением.

Система управления базами данных представляет собой набор программ, позволяющих разработчикам прикладного программного обеспечения не задумываться над тем, как эффективно и безопасно хранить данные разрабатываемой ими программы. Они просто передают необходимую информацию СУБД, а уже она заботится о том, где и как разместить эти данные на жестком диске таким образом, чтобы быстро их извлечь, когда от внешней программы придет соответствующий запрос. Если бы в этой схеме СУБД отсутствовала, то все реализуемые ею функции пришлось бы программировать заново в каждой разрабатываемой программе.

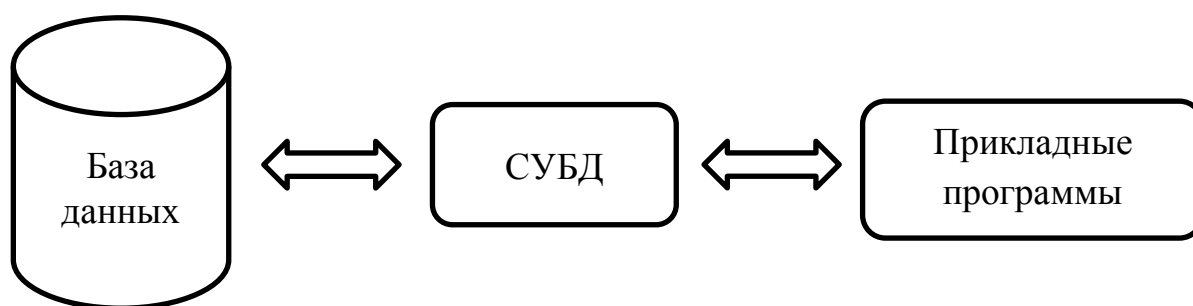


Рис. 1. СУБД в качестве слоя между прикладными программами и базой данных

## 1.2. Функции СУБД

Система управления базами данных выполняет следующие функции [6]:

1. Хранение, извлечение и обновление данных.
2. Предоставление системного каталога, доступного конечным пользователям.
3. Поддержка транзакций.
4. Наличие служб управления параллельной работой.
5. Наличие служб резервного копирования и восстановления.
6. Наличие служб контроля доступа к данным.
7. Осуществление поддержки обмена данными.
8. Наличие служб поддержки целостности данных.
9. Наличие служб поддержки независимости от данных.
10. Предоставление вспомогательных служб.

Рассмотрим эти функции более подробно.

Осуществление эффективного хранения (отсутствие избыточности и противоречивости), быстрого извлечения и обновления данных является основной функцией СУБД. При этом пользователь не должен задумываться о том, в каких таблицах реляционной базы записана нужная ему информация, в каком порядке она хранится на носителе и какие алгоритмы и структуры данных используются для ускорения доступа.

Одной из главных особенностей архитектуры ANSI-SPARC является наличие интегрированного системного каталога, содержащего данные о схемах, пользователях, приложениях. Системный каталог (или словарь данных) является хранилищем информации, описывающей данные в базе данных ("данные о данных" или метаданные). В различных СУБД количество метаданных и способ их применения могут изменяться. Чаще всего в системном каталоге хранятся имена, типы и размеры элементов данных; имена связей; накладываемые на данные ограничения поддержки целостности; имена пользователей, имеющих право доступа к данным; внешняя, концептуальная и

внутренняя схемы и отображения между ними; статистические данные, такие как частота транзакций и счетчики обращений к объектам базы данных.

Наличие системного каталога позволяет централизованно сохранять информацию о данных; назначать владельцев данных и разграничивать права доступа к ним; протоколировать внесённые изменения; упрощать контроль избыточности и поддержки целостности данных. Последствия любых изменений могут быть определены еще до их внесения, поскольку в системном каталоге зафиксированы все существующие элементы данных, установленные между ними связи, а также все их пользователи. Помимо системного каталога выделяется каталог данных (data directory), в котором находится информация о месте и способе хранения данных на носителе.

*Транзакция* – это логически единый набор действий, выполняемых отдельным пользователем или прикладной программой с целью доступа или изменения содержимого базы данных. Любая СУБД должна иметь механизм, гарантирующий выполнение либо всех операций, объявленных в данной транзакции, либо ни одной из них. Примерами простых транзакций могут служить добавление в базу данных сведений о новом сотруднике, обновление сведений о зарплате некоторого сотрудника или удаление сведений о некотором товаре. Примером более сложной транзакции может быть удаление сведений о сотруднике из базы данных и передача ответственности за все курируемые им проекты другому сотруднику. В этом случае потребуется внести изменения сразу в несколько таблиц базы данных. Если во время выполнения транзакции произойдет сбой, например из-за выхода из строя компьютера, база данных попадает в противоречивое состояние, поскольку часть изменений уже будет внесена, а остальная часть – еще нет. Поэтому все уже сделанные изменения должны быть отменены для возвращения базы данных в прежнее, непротиворечивое состояние.

СУБД должна гарантировать корректное обновление базы данных при параллельном выполнении нескольких операций добавления, изменения или удаления многими пользователями. Одна из основных целей создания и



использования СУБД заключается в том, чтобы множество пользователей могли осуществлять параллельный доступ к совместно обрабатываемым данным. Параллельный доступ сравнительно просто организовать, если все пользователи выполняют только чтение данных, поскольку в этом случае они не могут помешать друг другу. Однако когда два или больше пользователей одновременно пытаются провести изменения в базе данных, легко может возникнуть конфликт, приводящий к искажению информации.

СУБД должна предоставлять средства резервного копирования и восстановления базы данных на случай любого её повреждения или даже полной потери всех данных на носителе. Как уже упоминалось, при сбое транзакции, база данных должна быть возвращена в непротиворечивое состояние. Подобный сбой может произойти в результате выхода из строя системы или запоминающего устройства, ошибки программного или аппаратного обеспечения. Кроме того, пользователь может обнаружить ошибку во время выполнения транзакции и потребовать ее отмены. Во всех этих случаях СУБД должна предоставить механизм восстановления базы данных и её возврата к последнему непротиворечивому состоянию.

СУБД должна гарантировать возможность доступа к базе данных только для санкционированных пользователей. Часто приходится скрывать некоторые хранимые в базе данных сведения от других пользователей. Например, сотрудникам бухгалтерии необходимо предоставить всю информацию, связанную с зарплатой сотрудников, но желательно скрыть ее от других пользователей организации. Кроме того, базу данных следует защитить от любого преднамеренного или случайного несанкционированного доступа.

Чаще всего база данных располагается на сервере, а пользователи получают доступ к ней через сеть. СУБД получает запросы в виде сообщений обмена данными и аналогичным образом отвечает на них. Весь обмен сообщениями управляется диспетчером обмена данными (DEM – Data Exchange Manager). Этот диспетчер не является частью собственно СУБД, однако, любая СУБД должна обладать способностью интеграции с разнообразными

существующими диспетчерами обмена данными. Такая организация работы называется распределенной обработкой данных.

СУБД должна обладать инструментами контроля за тем, чтобы данные и их изменения соответствовали заданным правилам. Целостность базы данных означает корректность и непротиворечивость хранимых данных. Она может рассматриваться как еще один тип защиты данных. Помимо того, что данный вопрос связан с обеспечением безопасности, он имеет более широкий смысл, поскольку целостность связана с качеством самих данных. Целостность обычно выражается в виде ограничений или правил сохранения непротиворечивости данных, которые не должны нарушаться в базе. Например, можно указать, что сотрудник не может одновременно участвовать более чем в десяти проектах. В этом случае, при попытке назначить сотрудника на проект, СУБД должна проверить, не превышен ли установленный лимит, и в случае обнаружения подобного превышения запретить выполнение этого действия.

СУБД должна обладать инструментами поддержки независимости программ от фактической структуры базы данных. Обычно она достигается за счет реализации механизма поддержки представлений или подсхем. Физическая независимость от данных достигается довольно просто, так как обычно имеется несколько типов допустимых изменений физических характеристик базы данных, которые никак не влияют на представления. Однако добиться полной логической независимости от данных сложнее. Как правило, система легко адаптируется к добавлению нового объекта, атрибута или связи, но не к их удалению. В некоторых случаях можно запрещать внесение любых изменений в уже существующие компоненты логической структуры.

Вспомогательные утилиты предназначены для оказания помощи администраторам баз данных в эффективном управлении. Одни утилиты функционируют на внешнем уровне, в связи с чем они могут быть разработаны созданы самими администраторами, а другие работают на внутреннем уровне системы и создаются разработчиками СУБД.

### 1.3. Файловые системы

Файловые системы были первой попыткой оцифровать ручные картотеки. Подобная картотека (или подшивка документов) в некоторой организации могла содержать всю внешнюю и внутреннюю документацию, связанную с каким-либо проектом, продуктом, задачей, клиентом или сотрудником.

*Файловые системы* – это набор прикладных программ, которые выполняют для пользователей некоторые операции, например, создание отчетов. При этом каждая программа хранит свои собственные данные и управляет ими [6].

Несмотря на достигнутый прогресс в автоматизации рутинных операций сотрудников компаний, файловые системы имели ряд очевидных ограничений, которые создавали трудности при работе с данными и не позволяли эффективно ими управлять. Естественное желание избавиться от этих проблем, и привело к созданию первых СУБД, какими они являются сегодня.

Рассмотрим ограничения, которые были присущи файловым системам:

- разделение и изоляция данных;
- дублирование данных;
- зависимость от данных;
- несовместимость форматов файлов;
- фиксированные запросы;
- быстрое увеличение количества приложений.

Если логически связанные данные изолированы в отдельных файлах, доступ к ним весьма затруднителен. Выполнять подобную обработку данных в файловых системах достаточно сложно. Для извлечения соответствующей поставленным условиям информации программист должен организовать синхронную обработку двух файлов. Трудности существенно возрастают, когда необходимо извлечь данные более чем из двух файлов.

Из-за децентрализованной работы с данными, проводимой в каждом отделе независимо от других отделов, в файловой системе фактически допускается бесконтрольное дублирование данных, что, в принципе, неизбежно.

Бесконтрольное дублирование данных нежелательно в связи с тем, что оно сопровождается неэкономным расходом ресурсов, поскольку на ввод избыточных данных требуется затрачивать дополнительное время и деньги. Кроме того, для их хранения необходимо дополнительное место во внешней памяти, что связано с дополнительными накладными расходами. Во многих случаях дублирования данных можно избежать за счет совместного использования файлов. Еще более важен тот факт, что дублирование данных может привести к нарушению их целостности. Иначе говоря, данные в разных отделах могут стать противоречивыми.

Физическая структура и способ хранения записей файлов данных жестко фиксировались в коде приложений. Это значит, что изменить существующую структуру данных достаточно сложно. Например, добавление нового атрибута к сущности кажется совершенно незначительным изменением его структуры, но для воплощения этого изменения потребуется, как минимум, создать одноразовую программу специального назначения, преобразующую уже существующие файлы в новый формат. Кроме того, все обращающиеся к таким файлам программы должны быть изменены с целью соответствия новой структуре файла. А таких программ может быть очень много. Следовательно, программист должен, прежде всего, выявить все программы, нуждающиеся в доработке, а затем их перепроверить и изменить.

Поскольку структура файлов определяется кодом приложений, она также зависит от языка программирования, на котором написано это приложение. Прямая несовместимость таких файлов затрудняет процесс их совместной обработки.

С точки зрения пользователя возможности файловых систем намного превосходят возможности ручных картотек. Соответственно возрастают и требования к реализации новых или модифицированных запросов. Однако файловые системы требуют больших затрат труда программиста, поскольку все необходимые запросы и отчеты должны быть созданы именно им. В результате во многих организациях типы применяемых запросов и отчетов имели фиксированную форму, отсутствовали инструменты создания

незапланированных или произвольных запросов как к самим данным, так и к сведениям о том, какие типы данных доступны. В других организациях наблюдалось быстрое увеличение количества файлов и приложений. В конечном счете наступал момент, когда сотрудники отдела обработки данных были просто не в состоянии справиться со всей работой с помощью имеющихся ресурсов.

#### **1.4. Преимущества и недостатки СУБД**

Использование такого программного комплекса, как система управления базами данных, позволяет упростить и обезопасить работу с данными в организации. Однако между прикладными программами и данными появляется дополнительный элемент, что создает ряд проблем [6].

Преимущества СУБД:

- контроль за избыточностью и непротиворечивостью данных;
- больше полезной информации при том же объеме хранимых данных;
- совместное использование данных;
- поддержка целостности данных;
- повышенная безопасность;
- применение стандартов;
- повышение эффективности с ростом масштабов системы;
- возможность нахождения компромисса при противоречивых требованиях;
- повышение доступности данных и их готовности к работе;
- улучшение показателей производительности;
- упрощение сопровождения системы за счет независимости от данных.

Недостатки СУБД:

- сложность;
- высокая стоимость;
- дополнительные затраты на аппаратное обеспечение;
- серьезные последствия при выходе системы из строя.

## ГЛАВА 2. АРХИТЕКТУРА СУБД MS SQL SERVER

Система управления базами данных MS SQL Server представляет собой программный комплекс, состоящий из трёх основных компонентов [2-4]:

- ядро СУБД *Database Engine*;
- службы *Analysis Services*;
- службы *Reporting Services*.

Компания Microsoft предлагает визуализацию архитектуры СУБД MS SQL Server 2012, представленную на рис. 2.

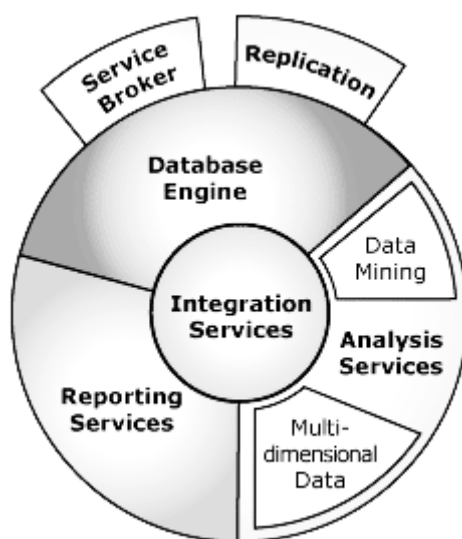


Рис. 2. Архитектура СУБД MS SQL Server 2012

1. *Компонент Database Engine* представляет собой основную службу для хранения, обработки и обеспечения безопасности данных. Этот компонент обеспечивает управляемый доступ к ресурсам и быструю обработку транзакций, что позволяет использовать его даже в самых требовательных корпоративных приложениях обработки данных [10].

Компонент *Database Engine* используется для создания реляционных баз данных для *оперативной обработки транзакций (On-Line Transaction Processing – OLTP)* или *оперативной аналитической обработки данных (On-Line Analytical Processing – OLAP)*.

OLAP-система – технология оперативной аналитической обработки данных, использующая методы и средства для сбора, хранения и анализа многомерных данных в целях поддержки принятия решений.

Возможности компонента Database Engine включают создание таблиц для хранения данных и объектов баз данных, таких как индексы, представления и хранимые процедуры для быстрого просмотра, эффективного хранения, управления данных их защитой. Среда *SQL Server Management Studio* предоставляет пользовательский интерфейс для управления объектами баз данных, а для фиксации событий сервера можно использовать приложение *SQL Server Profiler*.

2.1. Службы Microsoft SQL Server Analysis Services – «многомерные данные» осуществляют процесс OLAP, предоставляют возможность проектировать и создавать многомерные структуры данных, полученные из разных источников данных, и управлять этими многомерными объектами.

*Измерение* – последовательность значений одного из анализируемых параметров. Для параметра «время» измерением может являться календарный список дней, а для параметра «страна» – список областей.

*Многомерный анализ* – одновременный анализ по нескольким измерениям.

Службы Analysis Services позволяют аналитикам работать с многомерными структурами, содержащими детальные статистические данные в единой унифицированной логической модели, реализуя, по сути, модель хранилищ данных.

*Хранилище данных* – предметно-ориентированный, интегрированный, неизменяемый, поддерживающий хронологию набор данных, организованный для целей поддержки принятия решений и выполнении бизнес-аналитики.

Службы Analysis Services обеспечивают возможность быстрого, доступного для понимания пользователя нисходящего анализа большого количества данных, выполняя операции «Срез», «Вращение», «Консолидация» и «Детализация» многомерных данных. Результаты анализа могут предоставляться пользователям в текстовом или графическом формате,

учитывая национальные стандарты (обозначения десятичной части вещественного числа через точку или запятую; формат даты; отображения денежных величин и так далее).

Службы Analysis Services применяются для работы не только с хранилищами данных, но и с витринами данных, производственными базами данных и хранилищами оперативных данных, выполняя анализ накопленной информации и данных, поступающих в реальном времени.

2.2. Благодаря службам Analysis Services можно проектировать, создавать и визуализировать модели *интеллектуального анализа данных* (ИАД). Разнообразие стандартных алгоритмов интеллектуального анализа данных позволяет создавать такие модели на основе других источников данных.

Службы Analysis Services содержат следующие средства для предоставления возможностей интеллектуального анализа данных:

- конструктор интеллектуального анализа данных, предназначенный для создания и просмотра моделей интеллектуального анализа данных, управления моделями и составления прогнозов с помощью построенных моделей;
- язык расширений интеллектуального анализа данных, который можно использовать для управления моделями интеллектуального анализа данных и для создания сложных прогнозирующих запросов.

Для поиска закономерностей в данных используются методы *Data Mining* – исследование и обнаружение машинными алгоритмами в сырых данных скрытых знаний, которые ранее не были известны, нетривиальны, практически полезны и доступны для интерпретации человеком.

3. Службы *SQL Server Reporting Services* представляют серверную платформу, предоставляющую средства создания корпоративных отчетов с поддержкой веб-интерфейса, которые позволяют включать в отчеты данные из различных источников, публиковать отчеты в разнообразных форматах, централизованно управлять безопасностью и подписками.

Службы Reporting Services предлагают широкий выбор готовых к использованию средств и служб для создания, развертывания и управления



отчетами организации, а также функции программирования, позволяющие расширить и настроить функциональность отчетов. Благодаря API-интерфейсам, разработчики могут производить интеграцию и расширять возможности обработки данных и работу с отчетами в пользовательских приложениях. Инструменты служб Reporting Services работают в окружении Microsoft Visual Studio и интегрированы с компонентами SQL Server.

С помощью служб Reporting Services можно создать интерактивные, табличные, графические отчеты и отчеты свободной формы из реляционных, многомерных и XML-источников данных. Имеется возможность публиковать отчеты, планировать их обработку или осуществлять доступ к отчетам по требованию.

При проектировании отчетов можно выбрать мобильный тип отчетов или отчет с разбиением на страницы, заранее подготовленные для отображения на переносимых устройствах, включая расширенную визуализацию данных, графиков, диаграмм, карт и так далее.

4. Службы *Integration Services* это платформа, на которой создаются высокопроизводительные решения по интеграции данных, а также пакеты для хранения данных, которые дают возможность извлекать, преобразовывать и загружать данные.

Службы *Integration Services* созданы для того, чтобы решать сложные бизнес-задачи, выполняя такие функции, как копирование и загрузка файлов, отправка сообщений по электронной почте в ответ на события, обновление хранилищ данных, очистка и интеллектуальный анализ данных, а также при помощи этой службы производится управление объектами и данными SQL Server. Предусматривается работа пакетов как отдельно, так и совместно для решения сложных бизнес-задач. Службы *Integration Services* извлекают и преобразовывают данные из нескольких источников, таких как файлы данных XML, неструктурированные файлы, источники реляционных данных, и затем загружают полученную информацию в реляционные объекты.

Службы Integration Services содержат большое количество встроенных задач и преобразований, в них предоставлено множество средств для построения пакетов, также эта служба занимается управлением пакетами и их выполнением. При помощи графических инструментов служб Integration Services можно создавать готовые решения без единой строки кода, существует способ программирования объектной модели Integration Services для создания пакетов и создания в программном коде пользовательских задач и других объектов пакета.

5. *Репликация* – это процесс, представляющий собой копирование и перенос информации между разными базами данных, а также синхронизацию этих баз данных для обеспечения согласованности. Технологии репликации дают возможность размещения данных в различных местах, а также обеспечивают доступ к ним пользователей, которые подключены удаленно. Репликация позволяет подключиться через коммутируемые и беспроводные соединения, а также Интернет, используя локальные или глобальные сети.

Репликация транзакций обычно используется в сценариях «сервер-сервер», для которых необходима высокая пропускная способность, в том числе улучшение масштабируемости и доступности, хранение и протоколирование данных, интеграция данных с нескольких сайтов, объединение разнородных данных, автономная обработка пакетов. Репликация слиянием разработана в основном для мобильных приложений или распределенных серверных приложений, в которых возможно возникновение конфликтов данных. Обычные сценарии включают в себя обмен данными с мобильными пользователями, клиентские приложения точки продажи (POS) и интеграцию данных с нескольких сайтов. Репликация моментальных снимков используется с целью обеспечения начального набора данных для репликации транзакций и репликации слиянием; она также может применяться при необходимости выполнения полного обновления данных. Располагая этими тремя типами репликации, SQL Server представляет собой мощную и гибкую систему для синхронизации данных уровня предприятия.

Помимо репликации, можно синхронизировать базы данных с помощью служб Microsoft Sync Framework и Sync Services for ADO.NET. Службы Sync Services for ADO.NET предоставляют интуитивно понятный, гибкий API, который можно использовать для построения приложений, предназначенных для сценариев вне сети и совместных сценариев.

6. *Компонент Service Broker* создан для помощи разработчикам, которые создают безопасные масштабируемые приложения баз данных. Это новая технология компонента Database Engine, который предоставляет собой платформу взаимодействия пользователей друг с другом на основе обмена сообщениями, в то время как независимые компоненты приложений выступают единым целым. В компонент Service Broker включена инфраструктура асинхронного программирования, которая может использоваться как приложениями в пределах одной базы данных или экземпляра, так и распределенными приложениями.

Компонент SQL Server Service Broker обеспечивает собственную поддержку компонента SQL Server Database Engine для приложений обмена сообщениями и приложений с очередями сообщений. Это облегчает разработчикам создание сложных приложений, использующих компоненты Database Engine для связи между разнородными базами данных. Компонент Service Broker облегчает разработчикам процесс создания распределенных и надежных приложений.

Разработчики приложений, использующие компонент Service Broker, могут распределять рабочую нагрузку между несколькими базами данных без программирования сложного взаимодействия и создания внутреннего обмена сообщениями. Это сокращает разработку и проверочную работу, потому что компонент Service Broker обеспечивает взаимодействие в контексте диалога. Кроме того, это повышает производительность. При помощи компонента Service Broker выполняются все задачи в соответствии с транзакциями, которые гарантируют надежность и техническую согласованность.

## ГЛАВА 3. ИНДЕКСЫ

### 3.1. Теоретическая часть

Для ускорения поиска нужной информации в реляционной базе данных используются индексы.

Индекс представляет собой определенную физическую структуру данных, при помощи которой осуществляется быстрый доступ к одной или нескольким строкам данных. Если правильно настроить индексы, производительность запросов значительно возрастет.

В базах данных индекс обозначает то же самое, что и список терминов в книге. Чтобы найти нужный термин, первым делом следует обратиться к индексу, где указаны страницы, где встречаются различные термины. Среди них нужно лишь найти нужный термин, посмотреть на какой странице он расположен и сразу открыть нужную страницу. Точно так же при поиске конкретной строки таблицы компонент Database Engine в первую очередь воспользуется индексом, который определит, где именно физически она находится.

Существуют два главных различия между индексом книги и индексом базы данных.

Читатель книги имеет возможность самостоятельно решать, использовать индекс или нет. Пользователь базы данных такой возможности не имеет, и за него это решение принимает компонент системы, называемый оптимизатором запросов. Пользователь может использовать подсказки оптимизатору, но их рекомендуется применять только в ограниченном числе особых случаев.

Индекс для конкретной книги создается вместе с книгой, после чего он больше не изменяется. Это означает, что индекс для определенной темы всегда будет указывать на один и тот же номер страницы. Индекс базы данных, наоборот, может меняться при каждом изменении соответствующих данных.

Если для таблицы не создан подходящий индекс, то для выборки строк при выполнении запроса система использует метод *сканирования таблицы* – последовательного извлечения и просмотра системой каждой строки таблицы (начиная с первой, заканчивая последней), и, при удовлетворении условия поиска в предложении WHERE, строка оказывается в результирующем наборе. Таким образом, извлечение строк происходит посредством нахождения их физического расположения в памяти. Этот метод чаще всего оказывается менее эффективен, чем доступ с использованием индексов. Для сохранения индексов в базах данных существуют дополнительные структуры, которые называются страницами индексов.

Для каждой индексируемой строки имеется элемент индекса (index entry), который сохраняется на странице индексов. Элементы индексов включают в себя ключ индекса и указатель, потому получается, что строка таблицы значительно длиннее, чем элемент индекса. Именно поэтому число элементов индекса на соответствующих страницах индексов намного больше по сравнению с количеством строк, которые находятся в странице данных. Это свойство индексов играет очень важную роль, поскольку количество операций ввода/вывода, требуемых для прохода по страницам индексов, значительно меньше, чем количество операций ввода/вывода, требуемых для прохода по соответствующим страницам данных. Другими словами, для сканирования таблицы, вероятнее всего, потребовалось бы намного больше операций ввода/вывода, чем для сканирования индекса этой таблицы. Создание индексов компонентом Database Engine осуществляется с использованием структуры данных сбалансированного дерева B+. B+-дерево имеет древовидную структуру, в которой все самые нижние узлы (листья) находятся на расстоянии одинакового количества уровней от вершины (корневого узла) дерева. При добавлении или удалении данных это свойство сохраняется [7, 9].

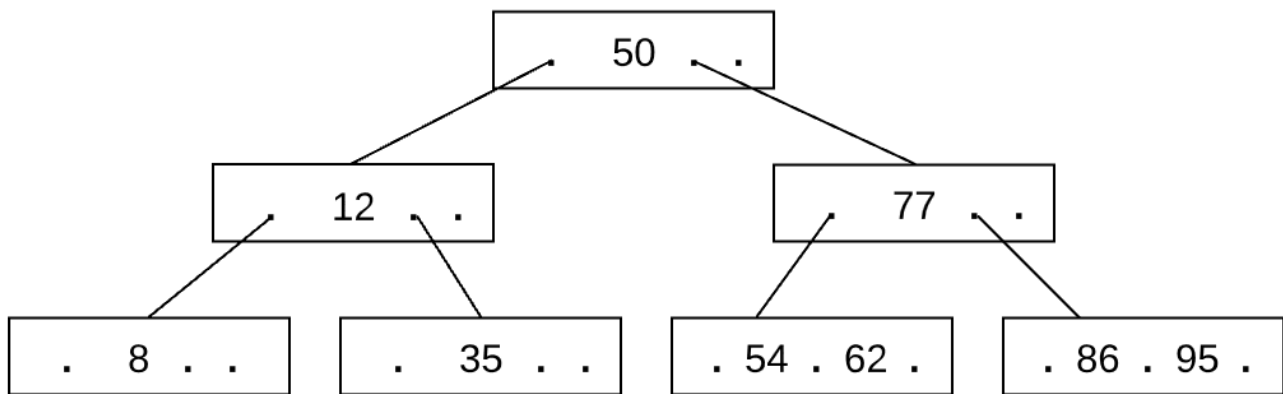


Рис. 3. Сбалансированное индексное B<sup>+</sup>-дерево

В MS SQL Server существует два основных вида индексов:

- кластеризованные;
- некластеризованные.

Кластеризованный индекс определяет физический порядок данных в таблице. Компонент Database Engine позволяет создавать для таблицы лишь один кластеризованный индекс, так как строки таблицы нельзя упорядочить физически более чем одним способом. Поиск с использованием кластеризованного индекса выполняется от корневого узла B<sup>+</sup>-дерева по направлению к листьям дерева, которые связаны между собой и представлены в виде двунаправленного связного списка, называемого цепочкой страниц. Важным свойством кластеризованного индекса является то, что листья его дерева содержат страницы данных. Узлы кластеризованного индекса всех других уровней содержат страницы индекса. Таблица с явно или неявно определенным кластеризованным индексом называется кластеризованной таблицей [8].

Кластеризованный индекс создается по умолчанию для каждой таблицы, для которой с помощью ограничения целостности определен первичный ключ. Кроме этого, каждый кластеризованный индекс однозначен по умолчанию. Это означает, что в столбце, для которого определен кластеризованный индекс, каждое значение данных может встречаться только один раз. Если же создание кластеризованного индекса производится для столбца, в котором значения повторяются, СУБД принудительно обеспечивает однозначность, добавляя

четырёхбайтовый идентификатор к строкам с повторяющимися значениями.

С помощью кластеризованных индексов имеется возможность получения быстрого доступа к данным, когда запрос осуществляет поиск в диапазоне значений.

Некластеризованный индекс имеет такую же структуру, как и кластеризованный, но с двумя важными отличиями:

- некластеризованный индекс не изменяет физическое упорядочивание строк таблицы;
- страницы листьев некластеризованного индекса состоят из ключей индекса и закладок.

При определении одного или нескольких некластеризованных индексов для таблицы, физическое расположение строк этой таблицы остаётся неизменным. Каждому некластеризованному индексу компонентом Database Engine предоставляется дополнительная индексная структура, сохранение которой происходит в индексные страницы.

Закладка в некластеризованном индексе указывает, где находится строка, соответствующая ключу индекса. Составляющая закладки ключа индекса может быть двух видов, в зависимости от того, является ли таблица кластеризованной таблицей или кучей (heap). Согласно терминологии SQL Server, кучей называется таблица без кластеризованного индекса. Если существует кластеризованный индекс, то закладка некластеризованного индекса указывает на узел B+-дерева кластеризованного индекса таблицы. Если для таблицы не создан кластеризованный индекс, то закладка идентична идентификатору строки (RID – Row Identifier), состоящего из трех частей: адреса файла, в котором хранится таблица, адреса страницы, в котором хранится строка, и смещения строки на странице.

Как уже упоминалось ранее, поиск данных с использованием некластеризованного индекса можно осуществлять двумя разными способами, в зависимости от типа таблицы:

– куча – прохождение при поиске по структуре некластеризованного индекса, после чего строка извлекается с использованием идентификатора строки;

– кластеризованная таблица – прохождение при поиске по структуре некластеризованного индекса, после чего следует прохождение по соответствующему кластеризованному индексу.

Оба варианта содержат большое количество операций ввода/вывода, поэтому при проектировании некластеризованного индекса следует быть предельно осторожным и применять его только в том случае, когда есть уверенность в том, что его использование существенно повысит производительность.

В процессе своего жизненного цикла индекс подвергается фрагментации, после чего хранение данных в страницах индекса становится недостаточно эффективным. Фрагментацию индекса разделяют на внутреннюю (тот объем данных, который хранится в каждой странице) и внешнюю (нарушение логического порядка страниц).

Индекс для таблицы создается с помощью инструкции CREATE INDEX. Эта инструкция имеет следующий синтаксис:

```
CREATE [UNIQUE] [CLUSTERED | NONCLUSTERED] INDEX index_name  
ON table_name (column1 [ASC | DESC],...)  
[INCLUDE (column_name [...])]
```

Компонент Database Engine является одной из немногих систем баз данных, которые поддерживают инструкцию ALTER INDEX. Эта инструкция используется для выполнения обслуживания индекса. Синтаксисы инструкций ALTER INDEX и CREATE INDEX очень схожи. Иными словами, эта инструкция позволяет изменять значения параметров ALLOW\_ROW\_LOCKS, ALLOW\_PAGE\_LOCKS, IGNORE\_DUP\_KEY и STATISTICS\_NORECOMPUTE. Также инструкцией ALTER INDEX поддерживаются ряд других параметров:

– параметр REBUILD, который используется, если нужно пересоздать индекс;



- параметр REORGANIZE, который используется, если нужно реорганизовать страницы листьев индекса;
- параметр DISABLE используется, чтобы отключить индекс.

При любом изменении данных, если используются инструкции INSERT, UPDATE или DELETE, возможна фрагментация данных. Если эти данные проиндексированы, то также возможна фрагментация индекса, когда информация индекса оказывается разбросанной по разным физическим страницам. В результате фрагментации данных индекса компонент Database Engine может быть вынужден выполнять дополнительные операции чтения данных, что понижает общую производительность системы. В таком случае требуется пересоздать (rebuild) все фрагментированные индексы.

Параметр REORGANIZE инструкции ALTER INDEX реорганизирует страницы листьев выбранного индекса таким образом, чтобы физический и логический порядок страниц совпадали – слева направо. Это приводит к повышению его производительности посредством удаления определенного объема фрагментации индекса.

Параметр DISABLE отключает указанный индекс. Отключенный индекс недоступен для применения, пока он не будет снова включен. Важно, что отключенный индекс не изменяется при внесении изменений в связанные с ним данные. По этой причине, чтобы снова использовать отключенный индекс, требуется его полное пересоздание. Для включения отключенного индекса применяется параметр REBUILD инструкции ALTER TABLE.

При отключенном кластеризованном индексе таблицы данные этой таблицы будут недоступны, так как все страницы данных таблицы с кластеризованным индексом хранятся в листьях его дерева.

В основном у компонента Database Engine нет никаких практических ограничения на число индексов, но их количество следует ограничивать. Во-первых, каждый индекс занимает определенный объем дискового пространства, следовательно, существует вероятность того, что общее количество страниц

индекса базы данных может превысить количество страниц данных. Во-вторых, в отличие от получения выгоды при использовании индекса для выборки данных, вставка и удаление данных такой выгоды не предоставляют по причине необходимости обслуживания индекса. Чем больше индексов создано для таблицы, тем больший требуется объем работы по их реорганизации при любом изменении связанных данных. Общим правилом будет разумный выбор индексов для частых запросов, а затем оценка их использования.

Создание индекса нужно в том случае, когда в предложении WHERE инструкции SELECT содержится условие поиска с одним столбцом. Это особенно рекомендуется при высокой селективности условия. Под селективностью условия понимается отношение количества строк, которые удовлетворяют условию, к общему числу строк в таблице. Высокая селективность соответствует меньшему значению этого соотношения. Обработка поиска с использованием индексированного столбца будет наиболее успешной при селективности условия, не превышающей 5%.

Столбец не следует индексировать при постоянном уровне селективности условия 80% или более. В этой ситуации для страниц индекса потребуются дополнительные операции ввода/вывода, которые уменьшат любую экономию времени, достигаемую за счет использования индексов. В этом случае быстрее выполнять поиск сканированием таблицы, что и будет обычно выбрано оптимизатором запросов.

Если условие поиска часто используемого запроса содержит операторы AND, лучше всего будет создать составной индекс по всем столбцам таблицы, указанным в предложении WHERE инструкции SELECT.

Когда проводится операция соединения, нужно создавать индекс для каждого соединяемого столбца. Соединяемые столбцы в большинстве случаев являются первичным ключом одной из таблиц и соответствующим внешним ключом другой таблицы. Если указываются ограничения для обеспечения целостности PRIMARY KEY и FOREIGN KEY, для соответствующих соединяемых столбцов следует создать только некластеризованный индекс для столбца

внешнего ключа, так как система неявно создаст кластеризованный индекс для столбца первичного ключа.

Если включить все столбцы запроса в индекс, производительность запроса может значительно увеличиться. Такой индекс называется покрывающим. Покрывающие индексы рекомендуется применять по той причине, что страницы индексов обычно содержат намного больше записей, чем соответствующие страницы данных. Кроме этого, для того чтобы использовать этот метод, фильтруемые столбцы должны быть первыми ключевыми столбцами в индексе.

При помощи компонента Database Engine можно создать следующие специальные типы индексов [8]:

- индексированные представления;
- фильтруемые индексы;
- индексы для вычисляемых столбцов;
- секционированные индексы;
- XML-индексы;
- полнотекстовые индексы.

## 3.2. Лабораторная работа № 1

### «Индексы»

**Цель работы:** изучить основные понятия индексации таблиц.

**Задание кафедры:**

1. Использовать учебную базу данных AdventureWorks.
2. Отобразить диаграмму базы данных.
3. Создать запрос на выборку из трёх таблиц.
4. Создать запрос на модификацию из трёх таблиц.
5. Отключить все индексы для таблиц, участвующих в запросе.
6. Выполнить запрос на выборку 10 раз, записать результаты времени выполнения запроса в таблицу, вычислить среднее время выполнения запроса на выборку.
7. Выполнить запрос на модификацию 10 раз, записать результаты времени выполнения запроса в таблицу, вычислить среднее время выполнения запроса на модификацию.
8. Включить кластеризованные индексы (средствами Management Studio и Transact-SQL кодом).
9. Выполнить пункты 6, 7.
10. Создать или включить некластеризованные индексы (средствами Management Studio и Transact-SQL кодом).
11. Выполнить пункты 6, 7.
12. Создать покрывающий индекс (Transact-SQL кодом).
13. Выполнить пункты 6, 7.
14. Сделать выводы.

### 3.3. Пример выполнения лабораторной работы № 1

#### 1. Получение доступа к базе данных.

При выполнении лабораторной работы используется интегрированная среда SQL Server Management Studio. Она предоставляет визуальный интерфейс для выполнения функций, реализованных в SQL Server. Некоторые задачи удобнее и быстрее выполнить в интегрированной среде, некоторые – с использованием Transact-SQL кода. Для запуска SQL Server Management Studio необходимо открыть файл программы «ssms.exe», ярлык, которой при установке по умолчанию помещается в меню Пуск.

При запуске Management Studio отображается диалоговое окно для соединения с сервером. Поскольку при установке MS SQL Server была выбрана проверка подлинности Windows, а не проверка подлинности SQL Server, то требуется нажать кнопку «Соединить» для соединения с экземпляром сервера по умолчанию.

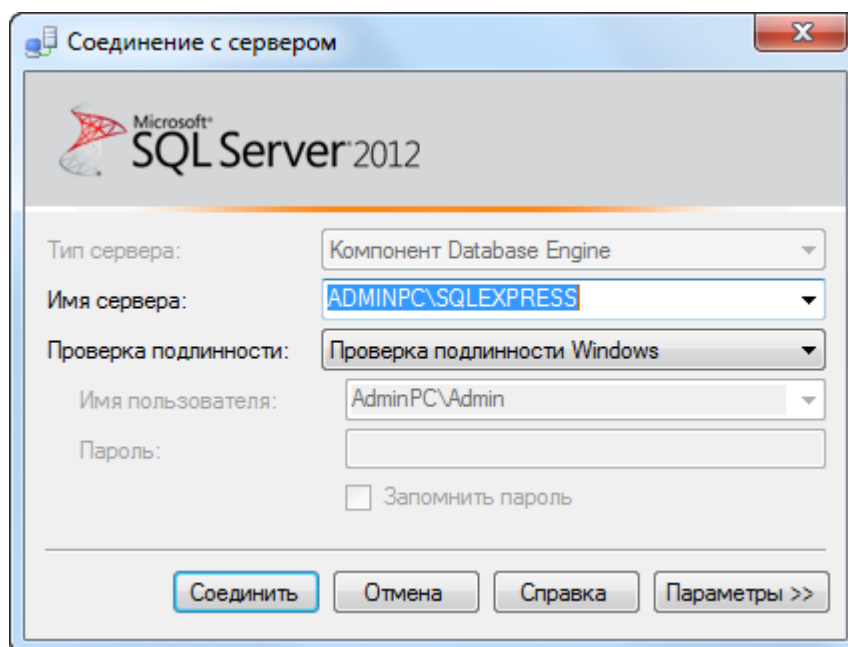


Рис. 4. Соединение с сервером

После нажатия кнопки «Создать запрос» на панели инструментов SQL Server Management Studio, необходимо ввести Transact-SQL код лист. 1, указав путь к файлу базы данных с расширением \*.mdf.

### Присоединение базы данных

```
CREATE DATABASE AdventureWorks2012
ON
(
    FILENAME = 'C:\AdventureWorks2012_Data.mdf'
)
FOR ATTACH_REBUILD_LOG;
GO
ALTER AUTHORIZATION
ON DATABASE::[AdventureWorks2012] TO sa;
```

Чтобы проверить успешность присоединения базы данных, необходимо в обозревателе объектов кликнуть правой кнопкой мыши по элементу «Базы данных» и выбрать пункт «Обновить». После раскрытия каталога баз данных в нем отобразится база данных «AdventureWorks2012» (рис. 5).

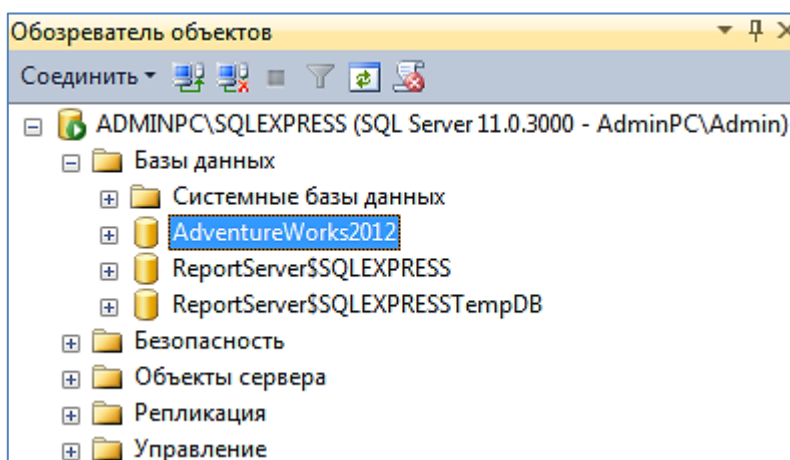


Рис. 5. Присоединение базы данных AdventureWorks2012

## 2. Отображение диаграммы базы данных.

Для отображения диаграммы базы данных в обозревателе объектов необходимо выбрать: *Базы данных* → AdventureWorks2012 → *Диаграммы базы данных* → *Создать диаграмму базы данных* (правой кнопкой мыши) и выделить все таблицы.



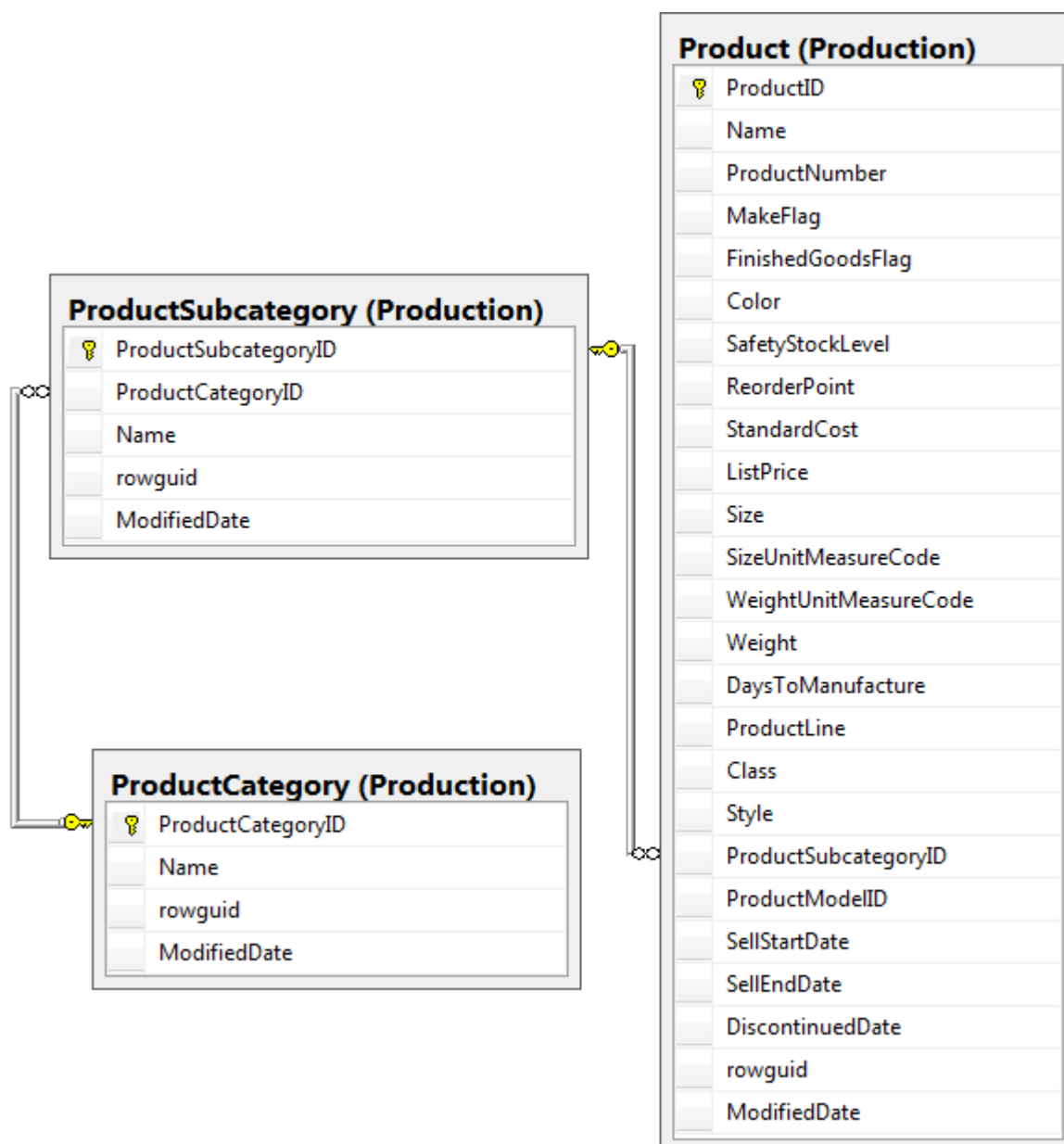


Рис. 7. Диаграмма части базы данных

3. Создание запроса на выборку и модификацию из трех таблиц.

Создадим запрос на выборку данных из таблиц *Production.Product*, *Production.ProductSubcategory* и *Production.ProductCategory* [1, 5, 11].

Необходимо вывести название товаров, категории «*Bikes*».



**Запрос на выборку**

```
USE AdventureWorks2012;
SELECT Product.Name
FROM
Production.Product, Production.ProductSubcategory,
Production.ProductCategory
WHERE Product.ProductSubcategoryID=ProductSubcategory.ProductCategoryID
AND
ProductSubcategory.ProductCategoryID=ProductCategory.ProductCategoryID
AND ProductCategory.Name='Bikes';
```

Составим запрос на модификацию данных, затрагивающий таблицы *Production.Product*, *Production.ProductSubcategory*, *Production.ProductCategory*. Необходимо изменить цвет товара на «*Красный*», если его категория «*Bikes*».

**Запрос на модификацию**

```
UPDATE Production.Product
SET Product.Color = 'Red'
FROM
Production.Product, Production.ProductSubcategory,
Production.ProductCategory
WHERE Product.ProductSubcategoryID=ProductSubcategory.ProductCategoryID
AND
ProductSubcategory.ProductCategoryID=ProductCategory.ProductCategoryID
AND ProductCategory.Name='Bikes';
```

**4. Использование кластеризованных индексов.**

Проведем измерение времени выполнения запросов при использовании только кластеризованных индексов. Для этого необходимо отключить все остальные индексы для таблиц запросов. Это можно сделать следующим образом: AdventureWorks2012 → *Диаграммы базы данных* → *Production.Product* → *Индексы* → *Отключить*.

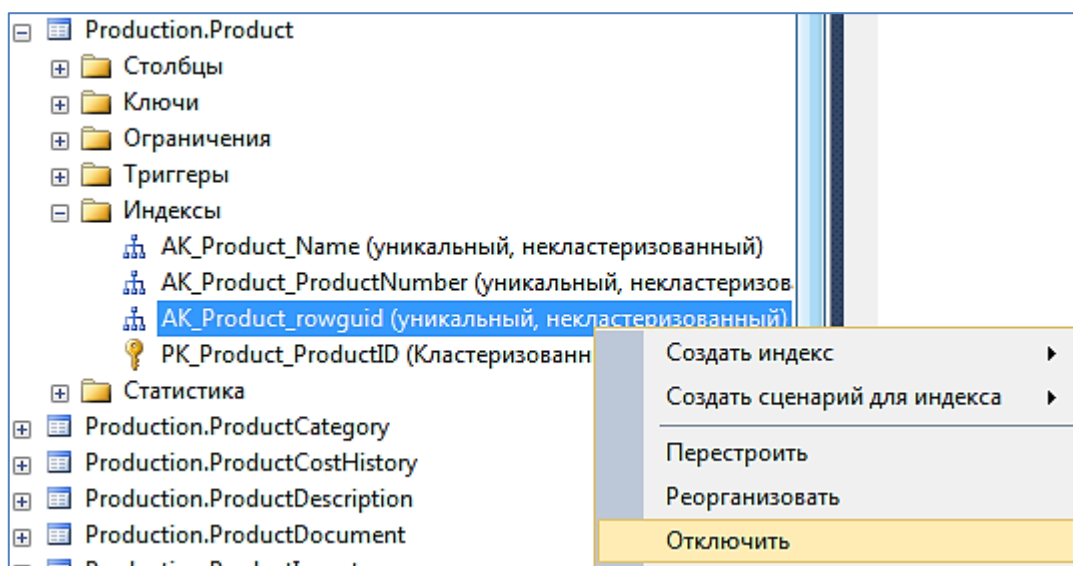


Рис. 8. Отключение некластеризованных индексов

После отключения производим 10 измерений времени выполнения запросов.

Таблица 1

**Время выполнения запросов с кластеризованными индексами, мс**

№ п/п	SELECT	UPDATE
1	0,1400905	0,103277
2	0,0590383	0,093150
3	0,0790484	0,034016
4	0,0790484	0,027554
5	0,0850566	0,003102
6	0,0899458	0,033022
7	0,0924755	0,032025
8	0,0882375	0,033025
9	0,1046006	0,033016
10	0,0088728	0,035025
Среднее	0,0906270	0,045513

Из табл. 1 видно, что среднее время обработки запроса на модификацию больше, чем запроса на выборку.

**5. Использование некластеризованных индексов.**

Включим некластеризованные индексы с помощью SQL-Server Management Studio. Нажмем правой кнопкой мыши по индексу и выберем пункт «Перестроить».

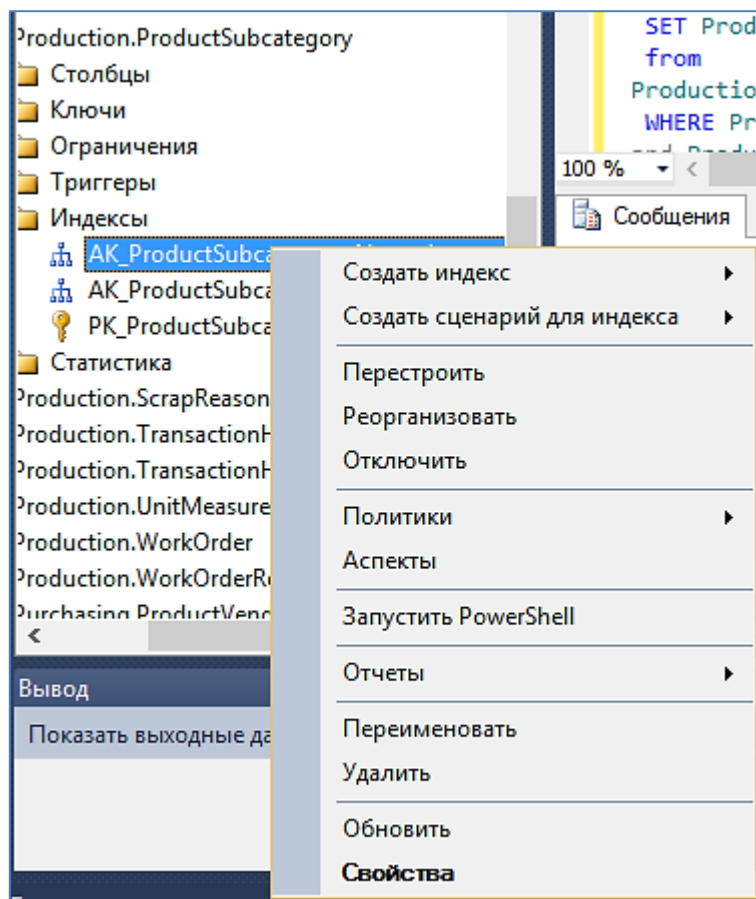


Рис. 9. Включение некластеризованных индексов

После включения некластеризованных индексов произведем 10 замеров времени обработки запросов.

**Время выполнения запросов с некластеризованными индексами, мс**

№ п/п	SELECT		UPDATE	
	Кластеризованные индексы	Некластеризованные индексы	Кластеризованные индексы	Некластеризованные индексы
1	0,1400905	0,0657048	0,103277	0,061045
2	0,0590383	0,0585623	0,093150	0,052034
3	0,0790484	0,0620362	0,034016	0,035031
4	0,0790484	0,0790930	0,027554	0,071047
5	0,0850566	0,0870580	0,031020	0,059040
6	0,0899458	0,0880612	0,033022	0,026015
7	0,0924755	0,0833218	0,032025	0,049035
8	0,0882375	0,0936937	0,033025	0,052506
9	0,1046006	0,0630424	0,033016	0,068045
10	0,0887280	0,0604532	0,035025	0,054241
Среднее	0,0906270	0,0741027	0,045513	0,052804

Из табл. 2 видно, что включение некластеризованных индексов привело к уменьшению времени обработки запроса SELECT и увеличению времени выполнения запроса UPDATE.

#### 6. Использование покрывающих индексов.

Создадим покрывающие индексы для таблиц Production.Product, Production.ProductSubcategory, Production.ProductCategory, которые будут содержать все столбцы запросов.

Листинг 4

#### Создание покрывающих индексов

```
CREATE INDEX i_category
ON Production.ProductCategory (Name)
INCLUDE (ProductCategoryID);
CREATE INDEX i_subcategory
ON Production.ProductSubcategory (ProductCategoryID)
INCLUDE (ProductSubcategoryID);
CREATE INDEX i_product
ON Production.Product (Name)
INCLUDE (Color, ProductCategoryID);
```

После создания покрывающих индексов произведем 10 измерений времени обработки запросов.

Таблица 3

**Время выполнения запросов с покрывающими индексами, мс**

№ п/п	SELECT			UPDATE		
	Кластеризованные индексы	Некластеризованные индексы	Покрывающие индексы	Кластеризованные индексы	Некластеризованные индексы	Покрывающие индексы
1	0,1400905	0,0657048	0,069067	0,103277	0,061045	0,141985
2	0,0590383	0,0585623	0,077936	0,093150	0,052034	0,064320
3	0,0790484	0,0620362	0,065049	0,034016	0,035031	0,052039
4	0,0790484	0,0790930	0,074053	0,027554	0,071047	0,055185
5	0,0850566	0,0870580	0,063042	0,031020	0,059040	0,048033
6	0,0899458	0,0880612	0,068049	0,033022	0,026015	0,022007
7	0,0924755	0,0833218	0,079066	0,032025	0,049035	0,063034
8	0,0882375	0,0936937	0,064038	0,033025	0,052506	0,065044
9	0,1046006	0,0630424	0,062347	0,033016	0,068045	0,038443
10	0,0887280	0,0604532	0,071054	0,035025	0,054241	0,030019
Среднее	0,0906270	0,0741027	0,693701	0,045513	0,052804	0,058011

Из табл. 3 видно, что включение покрывающих индексов привело к увеличению времени обработки запроса SELECT и увеличению UPDATE.

**Вывод**

Рассмотрены индексы различных видов и их влияние на время выполнения запросов. Создание некластеризованных и покрывающих индексов приводит к увеличению скорости выполнения запросов на выборку по сравнению с наличием только кластеризованных индексов, однако выполнение запросов на модификацию при этом увеличивается в связи с необходимостью перестроения индексных структур.

## ГЛАВА 4. ОПТИМИЗАТОР ЗАПРОСОВ

### 4.1. Теоретическая часть

При выполнении запроса компонентом Database Engine или любой другой реляционной СУБД возникает задача получения доступа и обработки требуемых для запроса данных с максимальной эффективностью. Компонент системы баз данных, занимающийся решением этой задачи, называется оптимизатором запросов [12].

Оптимизатор запросов (или просто оптимизатор) рассматривает различные возможные стратегии выборки данных для конкретного запроса и осуществляет выбор наиболее эффективной стратегии. Выбранная стратегия – это план выполнения запроса. Оптимизатор принимает решения, принимая во внимание такие факторы, как размер таблиц, к которым относится запрос, наличие существующих индексов и логических операторов (AND, OR или NOT), используемых в предложении WHERE. Эти факторы называются статистическими данными.

Задача оптимизатора заключается в выработке наиболее эффективного плана выполнения каждого конкретного запроса. Эта задача выполняется в следующие четыре этапа.

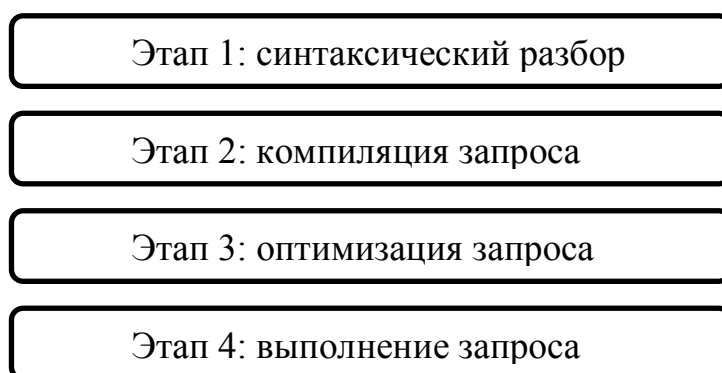


Рис. 10. Этапы обработки запроса

1. Синтаксический разбор. Проверяется синтаксис запроса и запрос преобразовывается в дерево. После этого выполняется проверка всех объектов базы данных, на которые ссылается запрос. Например, проверяется

существование всех столбцов, на которые ссылается запрос, и определяются их идентификаторы. После завершения процесса проверки создается окончательное дерево запроса.

2. Компиляция запроса. Оптимизатор запросов компилирует дерево запроса.

3. Оптимизация запроса. В качестве вводных данных принимается скомпилированное дерево запроса из предыдущего этапа, и оптимизатор запроса исследует разные варианты стратегий обращения к данным перед тем, как будет принято решение об обработке запроса. Для нахождения оптимального плана выполнения, оптимизатор запросов выполняет анализ запроса, в процессе которого выявляются аргументы поиска и операции соединения. Позже оптимизатор решает какие индексы использовать. В случае, если запрос содержит операции соединения, выбирается порядок выполнения соединений и метод их обработки.

4. Выполнение запроса. Созданный план выполнения сохраняется и затем выполняется.

В процессе анализа запроса оптимизатор исследует его на наличие аргументов поиска, использование оператора OR и существование критериев соединения, в приведенном в предложении WHERE порядке. Аргументом поиска является часть запроса, ограничивающая промежуточный результирующий набор запроса. Главное предназначением аргументов поиска – возможность использовать существующие индексы для конкретного выражения:

```
company_name = 'Star';  
budget >= 10000;  
project_name = 'Education 2.0' AND budget >= 10000.
```

Существует несколько типов выражений, которые оптимизатор не может использовать в качестве аргументов поиска. Первую группу таких выражений составляют выражения с оператором отрицания NOT (<>). Кроме этого, если с левой стороны оператора используется выражение, то все выражение не может быть аргументом поиска.

В качестве примеров выражений, не являющихся аргументами поиска, можно привести следующие:

```
NOT IN('project_1', 'project_2');
```

```
age <> 18;
```

```
salary * 0.87 > 25000.
```

Основным недостатком выражений, которые нельзя использовать в качестве аргументов поиска, является то, что оптимизатор не может использовать существующие индексы применительно к выражению, чтобы ускорить выполнение соответствующего запроса, то есть единственный способ доступа оптимизатора к данным – сканирование таблицы.

Принимать решение о том, можно ли использовать один или более существующих индексов оптимизатору позволяет идентификация аргументов поиска. На этом этапе оптимизации оптимизатор исследует каждый аргумент поиска на наличие индексов, имеющих отношение к соответствующему выражению. Если имеется подходящий индекс, оптимизатор решает, использовать его или нет. Это решение в основном зависит от селективности соответствующего выражения. Оптимизатор проверяет селективность выражения с индексированным столбцом, используя статистические данные, которые создаются для распределения значений в столбце. Оптимизатор запросов использует эту информацию, чтобы определить оптимальный план выполнения запроса, оценивая стоимость использования индекса для выполнения запроса.

Ранее отмечалось, что такой компонент СУБД, как оптимизатор запросов, использует индексы с целью эффективного и быстрого выполнения запроса. В случае если в таблице, по которой осуществляется запрос, не содержится индексов или оптимизатор не использует индексы при выполнении данного запроса, то система выбирает данные, осуществляя сканирование всей таблицы. Компонент Database Engine в данном случае последовательно считывает страницы данных и определяет строки, относящиеся к результирующему набору.



Доступ по индексу – это метод доступа к данным, в соответствии с которым СУБД читает, а впоследствии записывает данные на страницах, используя при этом существующий индекс. Эффективность данного метода состоит в том, что он позволяет значительно сократить количество выполняемых операций ввода-вывода при чтении в сравнении с последовательным сканированием всей таблицы данных.

Компонент Database Engine для поиска данных использует некластеризованный индекс двумя способами в зависимости от вида таблицы:

- таблица без кластеризованного индекса (система осуществляет обход некластеризованного индекса, далее извлекается строка на основе идентификатора строки);

- кластеризованная таблица (система осуществляет обход некластеризованного индекса, а затем обход структуры кластеризованного индекса таблицы).

Кластеризованный индекс, используемый при поиске данных, всегда универсален в отличие от некластеризованного. Компонент Database Engine осуществляет поиск данных с корня соответствующего B<sup>+</sup>-дерева и уже после трех-четырех операций достигает уровня узлов листьев, где и находятся данные. Это самое главное преимущество данного вида индекса, так как он позволяет осуществлять обход в разы быстрее, чем обход структуры некластеризованных индексов.

Но, сравнивая методы сканирования индекса или сканирования таблицы, сложно сказать, какой из них является наиболее эффективным и быстрым. Здесь уже все зависит от селективности и типа индекса.

Как правило, доступ сканированием таблицы начинает выполняться быстрее, чем доступ с использованием некластеризованного индекса, когда выбирается, по меньшей мере, 10% строк. Именно поэтому здесь не стоит исправлять решение оптимизатора, в какой момент переключаться от доступа по некластеризованному индексу к сканированию таблицы. Если все-таки возникла необходимость изменить решение оптимизатора, считая, что он

переключается на сканирование таблицы преждевременно, то можно использовать в запросе подсказку INDEX.

Сравнивая производительность поиска с помощью кластеризованного индекса и некластеризованного индекса, можно отметить преимущества первого.

Во-первых, осуществляя сканирование с помощью кластеризованного индекса, система способна не покидать структуру  $B^+$ -дерева, чтобы выполнять сканирование страниц данных. Связано это с тем, что страницы данных находятся в том же индексе на уровне листьев дерева.

Во-вторых, для некластеризованного индекса необходимо выполнение большего количества операций ввода-вывода по сравнению с кластеризованным индексом, основная причина чего заключается в том, что после обхода структуры  $B^+$ -дерева для некластеризованного индекса необходимо выполнить чтение либо страниц данных, либо  $B^+$ -дерева кластеризованного индекса, если у другой таблицы есть такой индекс.

Именно по этим причинам доступ с использованием кластеризованного индекса будет выполняться быстрее, чем доступ с использованием некластеризованного индекса, даже при плохой селективности.

Статистические данные индекса и статистические данные столбца создаются с целью оптимизации плана выполнения запроса.

Статистические данные индекса обычно создаются, когда создается индекс для конкретного столбца (столбцов). Создание статистических данных для индекса предполагает, что Database Engine создает гистограмму на основе до 200 значений столбца, соответственно создается до 199 интервалов. На данной гистограмме отображается:

- количество точных совпадений строк для каждого интервала;
- среднее количество строк с разными значениями в каждом интервале;
- плотность значений.

Необходимо отметить, что статистические данные всегда создаются для одного столбца. Если же используется составной индекс (по нескольким столбцам), то статистические данные создаются только для первого столбца.

Создать статистические данные можно, используя одно из двух следующих средств.

#### 1. Системная процедура `sp_createstats`.

Данная процедура позволяет создать данные по одному столбцу для всех столбцов всех пользовательских таблиц в текущей базе данных. В таком случае статистическим данным присваивается имя столбца, на котором они созданы.

#### 2. Среда SQL Server Management Studio.

Если возникла потребность создать статистические данные, то в среде необходимо развернуть узел сервера, соответствующие папки и таблицы и на их основе выполнить процедуру формирования данных.

После создания статистических данных и по мере изменения данных в столбцах статистические данные индекса устаревают, что отрицательно сказывается на производительности запроса. В подобном случае можно использовать компонент Database Engine, который позволяет автоматически обновлять данные индекса, однако для этого необходимо активировать процедуру `AUTO_UPDATE_STATISTICS`, то есть заменить значение `OFF`, на `ON`. Также в компоненте Database Engine существует опция `AUTO_CREATE_STATISTICS`, которая позволяет создать недостающие статистические данные, требуемые для оптимизации запроса [8].

Активировать (деактивировать) данные опции можно двумя способами:

- с помощью инструкции `ALTER DATABASE`;
- с помощью среды SQL Server Management Studio.

Помимо того, что компонент Database Engine позволяет создать наборы статистических данных для всех индексов, он способен создавать данные и для неиндексированных столбцов. В таком случае данные называются статистическими данными столбцов.

Часто с целью оптимизации выполнения запросов статистические данные индексов и столбцов используются вместе. Компонент Database Engine создает статистические данные для неиндексированного столбца, который входит в условие предложения `WHERE`.

Преимущество статистических данных столбцов состоит в том, что они позволяют оптимизатору принять рациональное решение в той или иной ситуации. К примеру, наличие составного индекса по двум или более столбцам. Так как система создает индекс только для первого столбца индекса, то наличие статистических данных второго и последующего индексов может помочь принять правильное решение и выбрать оптимальный план выполнения.

Компонент Database Engine имеет два представления каталога для работы со статистическими данными столбца.

#### 1. Представление sys.stats.

Данное представление располагает строкой с целью набора статистических данных таблицы, а также тремя столбцами:

- столбец name (указывается имя статистики);
- столбец auto\_created (содержит статистические данные созданные оптимизатором запросов);
- столбец user\_created (содержит статистические данные, созданные пользователем).

#### 2. Представление sys.stats\_columns.

Данное представление содержит дополнительную информацию о столбцах представления sys.stats.

Представления sys.stats и sys.stats\_columns можно использовать также для редактирования информации, связанной со статистическими данными индекса.

Помимо прочих перечисленных функций и возможностей компонент Database Engine использует набор кэшей для хранения данных и выполнения плана запросов. Изначально при выполнении первого запроса в памяти сохраняется его скомпилированная версия (такую память называют кэшем планов), затем перед выполнением следующих запросов осуществляется поиск в памяти планов, и, если окажется, что такой план имеется, повторная компиляция запроса не выполняется.

Иногда возникает необходимость редактирования планов выполнения, для чего разработано несколько методов, позволяющих осуществлять редактирование.

#### 1. Опция optimize for ad hoc workloads.

Данная опция позволяет при первом выполнении плана не размещать версию запроса в кэше планов. Туда помещается только заглушка плана, которая содержит минимальную информацию, необходимую системе для нахождения совпадений. Такой способ позволяет минимизировать неуправляемый рост кэша планов.

#### 2. Инструкция DBCC FREEPROCCACHE.

Данная опция позволяет удалять все планы из кэша. Такой способ часто используют для оценки производительности.

Как правило, оптимизатор всегда выбирает самый быстрый план выполнения запросов. Но бывают ситуации, когда оптимизатор не может самостоятельно определить наиболее эффективное и рациональное решение. В таком случае используются подсказки, которые воздействуют на оптимизатор, вынуждая его выбрать наиболее производительный план выполнения запроса. Данные подсказки являются факультативной частью инструкции SELECT и направляют оптимизатора осуществлять запрос определенным образом.

Компонент Database Engine содержит следующие типы подсказок:

- табличные подсказки;
- подсказки соединения;
- подсказки запросов;
- структуры планов.

Ядро SQL Server Database Engine может показывать, каким образом оно переходит к таблицам и использует индексы для доступа к данным или их обработки для запроса. Это называется выводом плана выполнения. Для проведения анализа медленно выполняемого запроса полезно изучить сам план выполнения запроса.

Так в процессе выполнения запроса выполняется ряд основных операций, которые представлены на рис. 11-14.

1. Данная операция предполагает, что каждая строка верхнего входа используется для создания хэш-таблицы, каждая строка нижнего входа – для проверки хэш-таблицы. В итоге такой операции выводятся все совпадающие строки. Данная операция представлена на рис. 11.



Рис. 11. Использование хэш-таблицы

2. Операция позволяет осуществлять просмотр всего кластеризованного индекса или его части. На рис. 12 изображена соответствующая операция.



Рис. 12. Использование кластеризованного индекса

3. Операция позволяет осуществлять из двух отсортированных соответствующим образом входных таблиц сопоставление строк с использованием их порядка сортировки. Данная процедура представлена на рис. 13.



Рис. 13. Использование техники слияния соединения

4. Данная процедура позволяет осуществлять просмотр всего некластеризованного индекса или его части. На рис. 14 приведена данная операция.

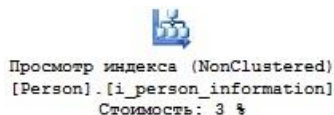


Рис. 14. Использование некластеризованного индекса

## 4.2. Лабораторная работа № 2

### «Оптимизатор запросов»

**Цель работы:** исследовать работу оптимизатора запросов и изучить составляющие элементы планов выполнения запросов.

**Задание кафедры:**

1. Использовать учебную базу данных AdventureWorks.
2. Создать запрос на выборку из трёх таблиц с использованием операторов соединения (AND) и пересечения (OR).
3. Отключить все индексы.
4. Отобразить предполагаемый план выполнения запроса.
5. Отобразить действительный план выполнения запроса.
6. Выполнить запрос на выборку 10 раз, записать результаты времени выполнения запроса в таблицу, вычислить среднее время выполнения запроса на выборку.
7. Изменить порядок следования условий в запросе, поставив, по возможности, на первое место все операторы соединения.
8. Выполнить пункты 4, 5, 6.
9. Создать или включить некластеризованные индексы.
10. Выполнить пункты 4, 5, 6.
11. Создать покрывающий индекс.
12. Выполнить пункты 4, 5, 6.
13. Сделать выводы.

### 4.3. Пример выполнения лабораторной работы № 2

#### 1. Присоединение базы данных

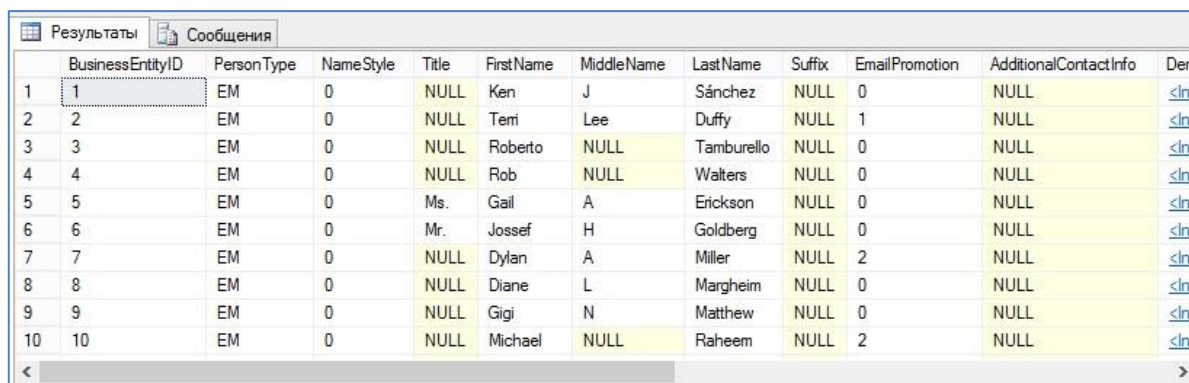
Для выполнения работы необходимо скачать учебную базу данных AdventureWorks2012 и присоединить ее к Microsoft SQL Server 2012. Для присоединения базы данных можно воспользоваться Transact-SQL кодом, указанным в лист. 5.

Листинг 5

#### Присоединение базы данных

```
CREATE DATABASE AdventureWorks2012
ON
(
    FILENAME = 'C:\AdventureWorks2012_Data.mdf'
)
FOR ATTACH_REBUILD_LOG;
GO
ALTER AUTHORIZATION
ON DATABASE::[AdventureWorks2012] TO sa;
```

2. Создадим запрос на выборку из трёх таблиц с использованием операторов соединения (AND) и пересечения (OR). В запросе будем использовать три таблицы: *Person*, *PersonPhone* и *EmailAddress* (рис. 15-17).



	BusinessEntityID	PersonType	NameStyle	Title	FirstName	MiddleName	LastName	Suffix	EmailPromotion	AdditionalContactInfo	<img alt="SQL Server icon" data-bbox="855 625 875 645"/>
1	1	EM	0	NULL	Ken	J	Sánchez	NULL	0	NULL	<img alt="SQL Server icon" data-bbox="855 625 875 645"/>
2	2	EM	0	NULL	Terri	Lee	Duffy	NULL	1	NULL	<img alt="SQL Server icon" data-bbox="855 625 875 645"/>
3	3	EM	0	NULL	Roberto	NULL	Tamburello	NULL	0	NULL	<img alt="SQL Server icon" data-bbox="855 625 875 645"/>
4	4	EM	0	NULL	Rob	NULL	Walters	NULL	0	NULL	<img alt="SQL Server icon" data-bbox="855 625 875 645"/>
5	5	EM	0	Ms.	Gail	A	Erickson	NULL	0	NULL	<img alt="SQL Server icon" data-bbox="855 625 875 645"/>
6	6	EM	0	Mr.	Josief	H	Goldberg	NULL	0	NULL	<img alt="SQL Server icon" data-bbox="855 625 875 645"/>
7	7	EM	0	NULL	Dylan	A	Miller	NULL	2	NULL	<img alt="SQL Server icon" data-bbox="855 625 875 645"/>
8	8	EM	0	NULL	Diane	L	Margheim	NULL	0	NULL	<img alt="SQL Server icon" data-bbox="855 625 875 645"/>
9	9	EM	0	NULL	Gigi	N	Matthew	NULL	0	NULL	<img alt="SQL Server icon" data-bbox="855 625 875 645"/>
10	10	EM	0	NULL	Michael	NULL	Raheem	NULL	2	NULL	<img alt="SQL Server icon" data-bbox="855 625 875 645"/>

Рис. 15. Таблица *Person.Person*



	BusinessEntityID	PhoneNumber	PhoneNumberTypeID	ModifiedDate
1	1	697-555-0142	1	2003-02-08 00:00:00.000
2	2	819-555-0175	3	2002-02-24 00:00:00.000
3	3	212-555-0187	1	2001-12-05 00:00:00.000
4	4	612-555-0100	1	2001-12-29 00:00:00.000
5	5	849-555-0139	1	2002-01-30 00:00:00.000
6	6	122-555-0189	3	2002-02-17 00:00:00.000
7	7	181-555-0156	3	2003-03-05 00:00:00.000
8	8	815-555-0138	1	2003-01-23 00:00:00.000
9	9	185-555-0186	1	2003-02-10 00:00:00.000
10	10	330-555-2568	3	2003-05-28 00:00:00.000
11	11	719-555-0181	1	2004-12-29 00:00:00.000

Рис. 16. Таблица *Person.PersonPhone*

	BusinessEntityID	EmailAddressID	EmailAddress	rowguid	ModifiedDate
1	1	1	ken0@adventure-works.com	8A1901E4-671B-431A-871C-EADB2942E9EE	2003-02-08 00:00:00.000
2	2	2	tem0@adventure-works.com	B5FF9EFD-72A2-4F87-830B-F338FDD4D162	2002-02-24 00:00:00.000
3	3	3	roberto0@adventure-works.com	C8A51084-1C03-4C58-A8B3-55854AE7C499	2001-12-05 00:00:00.000
4	4	4	rob0@adventure-works.com	17703ED1-0031-4B4A-AFD2-77487A556B3B	2001-12-29 00:00:00.000
5	5	5	gail0@adventure-works.com	E76D2EA3-08E5-409C-BBE2-5DD1CDF89A3B	2002-01-30 00:00:00.000
6	6	6	josef0@adventure-works.com	A9C4093A-4F4A-4CAD-BBB4-2C4E920BACCB	2002-02-17 00:00:00.000
7	7	7	dylan0@adventure-works.com	70429DE4-C3BF-4F19-A00A-E976C8017FB3	2003-03-05 00:00:00.000
8	8	8	diane1@adventure-works.com	37F02A87-058D-49F8-A20D-965738B0A71F	2003-01-23 00:00:00.000
9	9	9	gigi0@adventure-works.com	F888A16D-0C33-459E-9D72-D16AE0BB1F43	2003-02-10 00:00:00.000
10	10	10	michael6@adventure-works.com	E0DD366D-433D-4F5A-9347-1A5FE7FBE0A3	2003-05-28 00:00:00.000
11	11	11	ovidiu0@adventure-works.com	0FF9523D-F398-4237-85F8-2834DE441692	2004-12-29 00:00:00.000

Рис. 17. Таблица *Person.EmailAddress*

На рис. 18 представлена диаграмма связей этих таблиц.

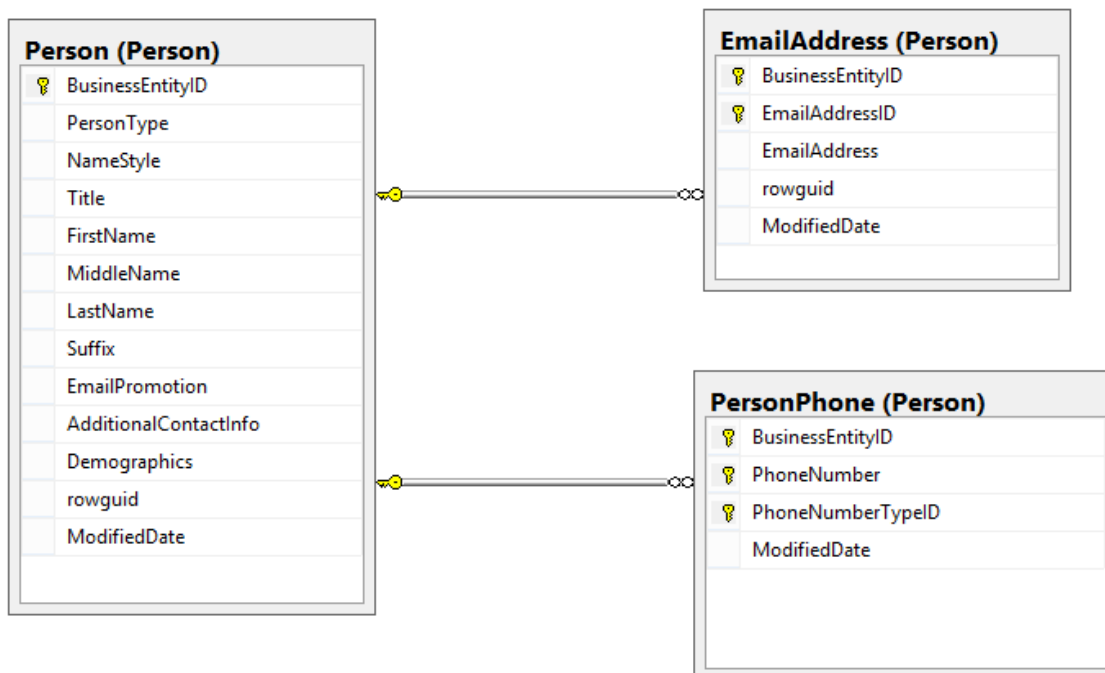


Рис. 18. Диаграмма связей

### Запрос на выборку из трех таблиц

```

SET STATISTICS TIME ON GO
SELECT Person.Person.BusinessEntityID, Person.Person.FirstName,
Person.Person.MiddleName,
Person.Person.LastName, Person.PersonPhone.PhoneNumber,
Person.EmailAddress
FROM Person.Person, Person.PersonPhone, Person.EmailAddress
WHERE Person.PersonPhone.BusinessEntityID=Person.Person.BusinessEntityID
AND Person.EmailAddress.BusinessEntityID=Person.Person.BusinessEntityID
AND Person.Person.BusinessEntityID IN(
SELECT BusinessEntityID FROM Person.Person
WHERE Person.Person.FirstName='Roberto'
OR Person.Person.MiddleName='NULL '
AND Person.PersonPhone.ModifiedDate='2003-02-08'
OR Person.PersonPhone.ModifiedDate='2002-02-08'
AND Person.EmailAddress.ModifiedDate='2003-02-08'
OR Person.EmailAddress.ModifiedDate='2002-02-08'
GO SET STATISTICS TIME OFF

```

### 3. Выполним отключение всех индексов.

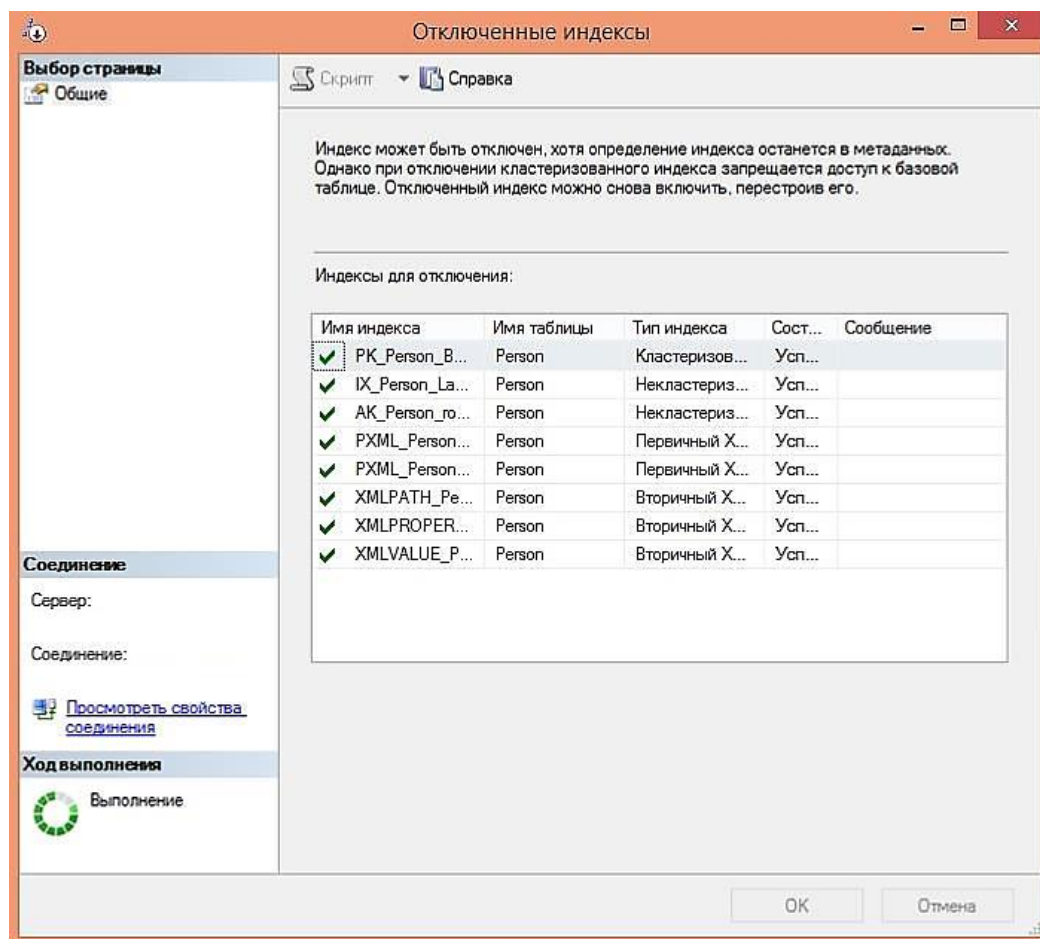


Рис. 19. Отключение индексов

#### 4. Отообразим предполагаемый план выполнения.

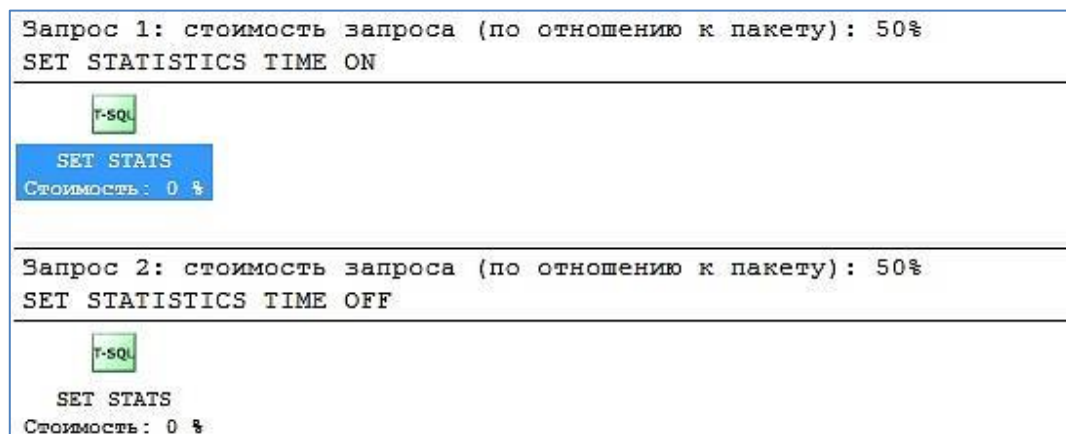


Рис. 20. Предполагаемый план выполнения

#### 5. Отообразим действительный план выполнения запроса.

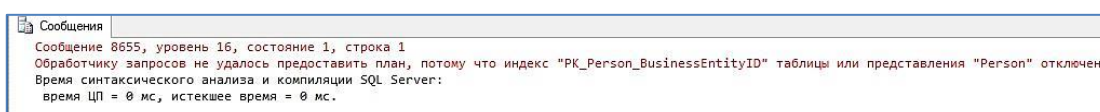


Рис. 21. Ошибка отображения действительного плана  
выполнения запроса

Обработчику запросов не удалось предоставить план, потому что индекс "PK\_Person\_BusinessEntityID" таблицы или представления "Person" отключен. Выполнить запрос на выборку невозможно, так как все индексы отключены.

6. Изменим порядок следования условий в запросе, поставив на первое место все операторы соединения.

Листинг 7

#### Запрос на выборку с изменённым порядком следования условий

```
SET STATISTICS TIME ON
GO
SELECT Person.Person.BusinessEntityID, Person.Person.FirstName,
Person.Person.MiddleName,
Person.Person.LastName, Person.PersonPhone.PhoneNumber,
Person.EmailAddress
FROM Person.Person, Person.PersonPhone, Person.EmailAddress
WHERE Person.PersonPhone.BusinessEntityID=Person.Person.BusinessEntityID
AND Person.EmailAddress.BusinessEntityID=Person.Person.BusinessEntityID
AND Person.Person.BusinessEntityID IN(
SELECT BusinessEntityID FROM Person.Person
WHERE Person.Person.MiddleName='NULL ')
```

```

AND Person.PersonPhone.ModifiedDate='2003-02-08'
AND Person.EmailAddress.ModifiedDate='2003-02-08'
OR Person.EmailAddress.ModifiedDate='2002-02-08'
OR Person.Person.FirstName='Roberto'
OR Person.PersonPhone.ModifiedDate='2002-02-08'
GO
SET STATISTICS TIME OFF

```

7. Включим кластеризованные индексы и отобразим предполагаемый план выполнения запроса.

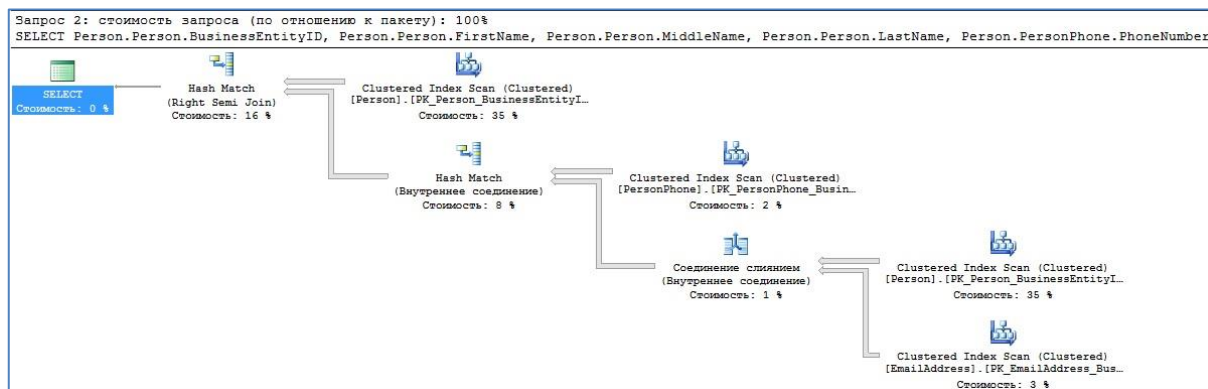


Рис. 22. Предполагаемый план выполнения запроса

Видно, что оптимизатор запросов будет использовать кластеризованные индексы при выборке из всех трех таблиц и применять техники соединения слиянием и хешированием.

8. Отобразим действительный план выполнения запроса.

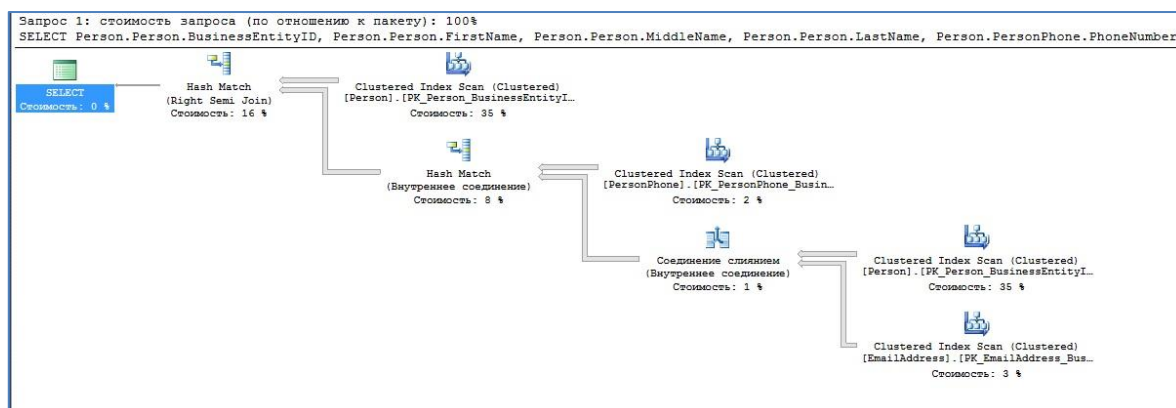


Рис. 23. Действительный план выполнения

Выполним запрос на выборку 10 раз, запишем результаты времени выполнения запроса в табл. 4 и вычислим среднее время выполнения запроса на выборку.

**Время выполнения запроса с кластеризованными индексами**

№ п/п	Время выполнения запроса, мс
1	0,109
2	0,078
3	0,156
4	0,108
5	0,124
6	0,078
7	0,120
8	0,109
9	0,140
10	0,093
Среднее	0,112

## 9. Перестроим некластеризованные индексы.

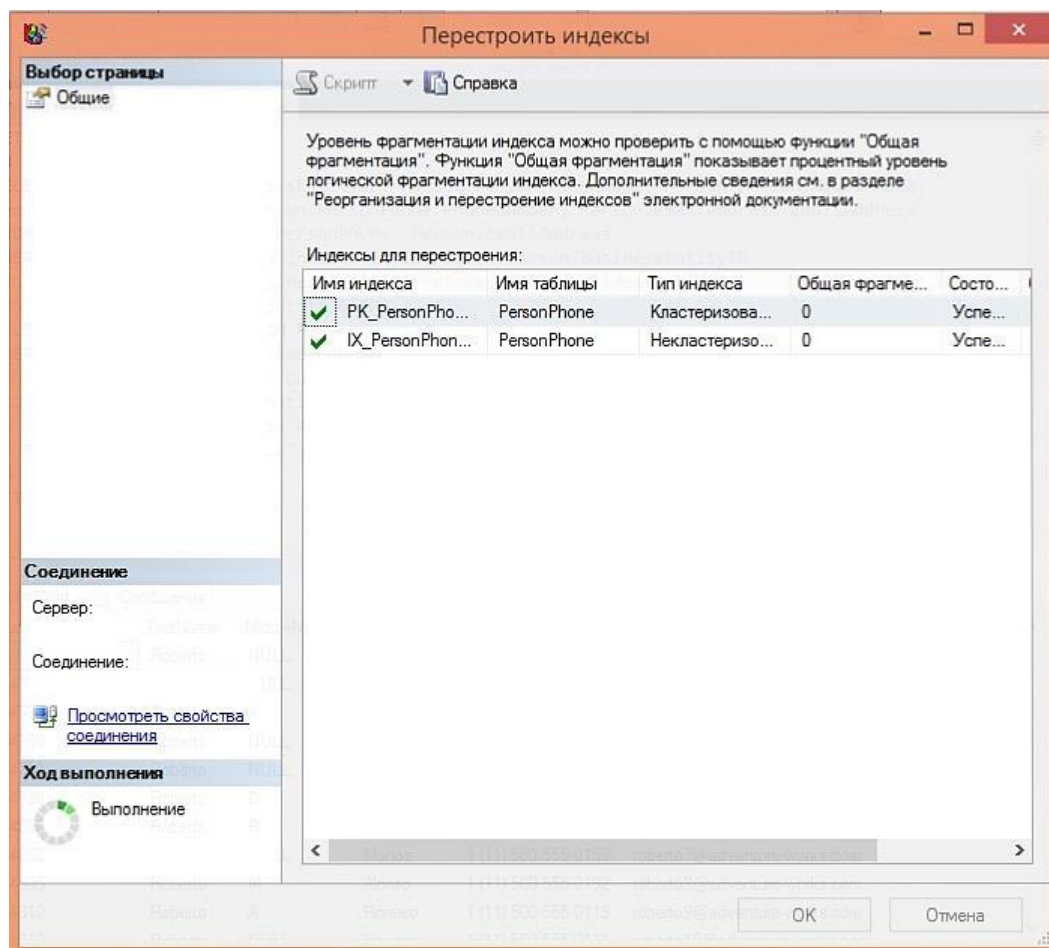


Рис. 24. Включение некластеризованных индексов

## 10. Отообразим предполагаемый план выполнения запроса.

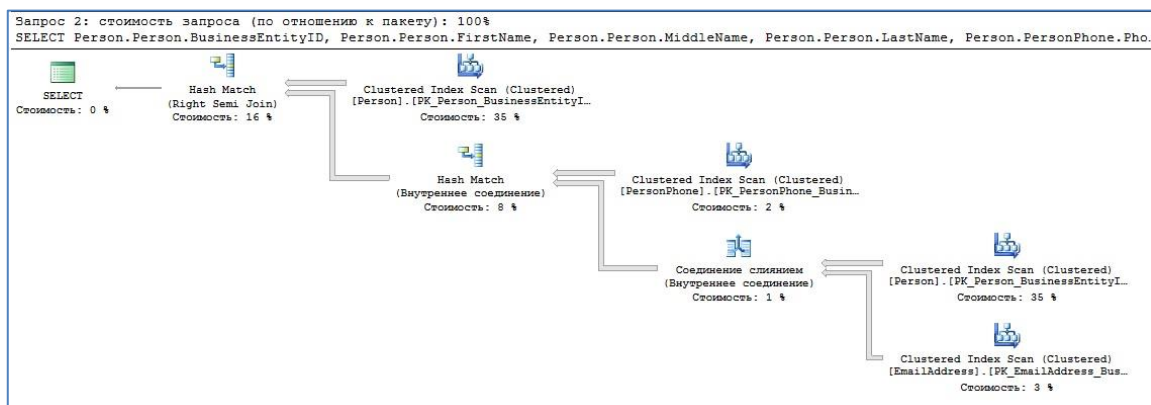


Рис. 25. Предполагаемый план выполнения запроса

## 11. Отообразим действительный план выполнения.

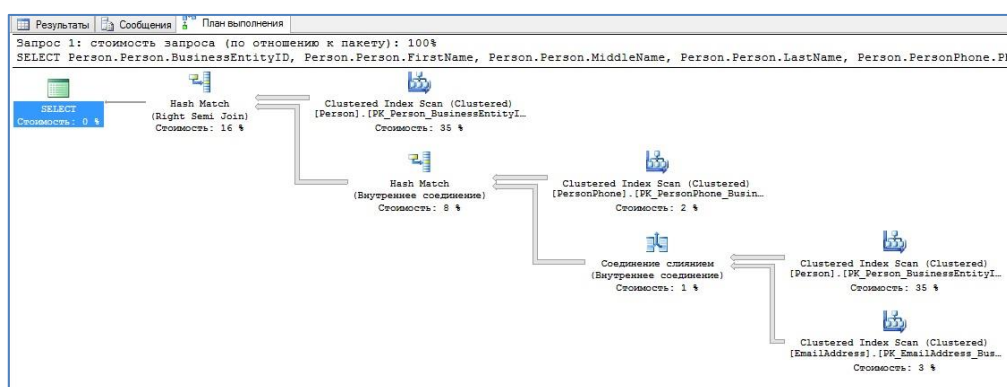


Рис. 26. Действительный план выполнения запроса

Видно, что, несмотря на наличие некластеризованных индексов, оптимизатор запросов выбирает такой план выполнения, который использует только кластеризованные индексы. Возможно, это связано с высоким значением селективности.

12. Выполним запрос на выборку 10 раз, запишем результаты времени выполнения запроса в табл. 5 и вычислим среднее время выполнения запроса на выборку.



**Время выполнения запроса с некластеризованными индексами**

№ п/п	Время выполнения запроса, мс
1	0,109
2	0,143
3	0,067
4	0,140
5	0,070
6	0,130
7	0,067
8	0,128
9	0,109
10	0,065
Среднее	0,1028

13. Создадим покрывающий индекс.

Листинг 8

**Создание покрывающего индекса**

```
CREATE INDEX i_person_information
ON Person.Person (FirstName)
INCLUDE (BusinessEntityID,MiddleName,LastName);
```

Новый индекс *i\_person\_information* для таблицы *Person.Person* можно просмотреть в *Обозревателе объектов* для данной таблицы.

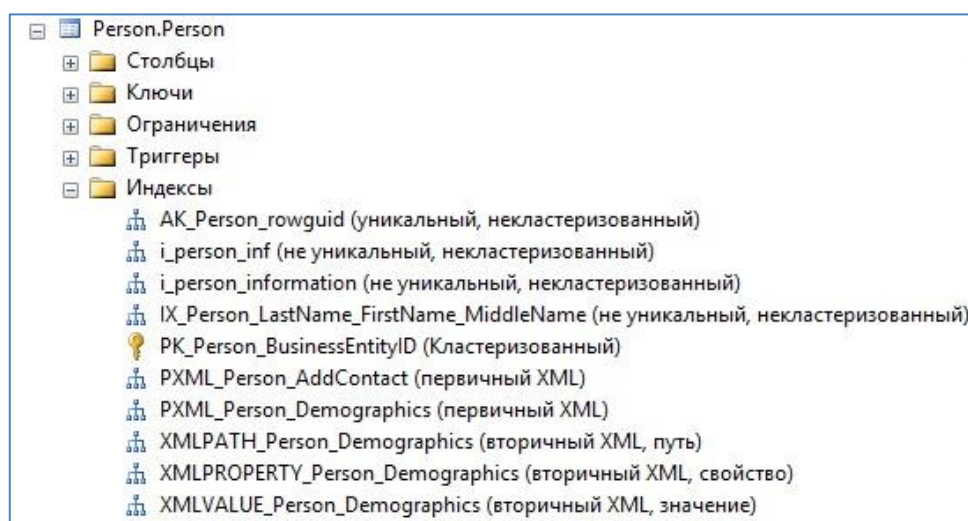


Рис. 27. Покрывающий индекс

#### 14. Отобразим предполагаемый план выполнения запроса.

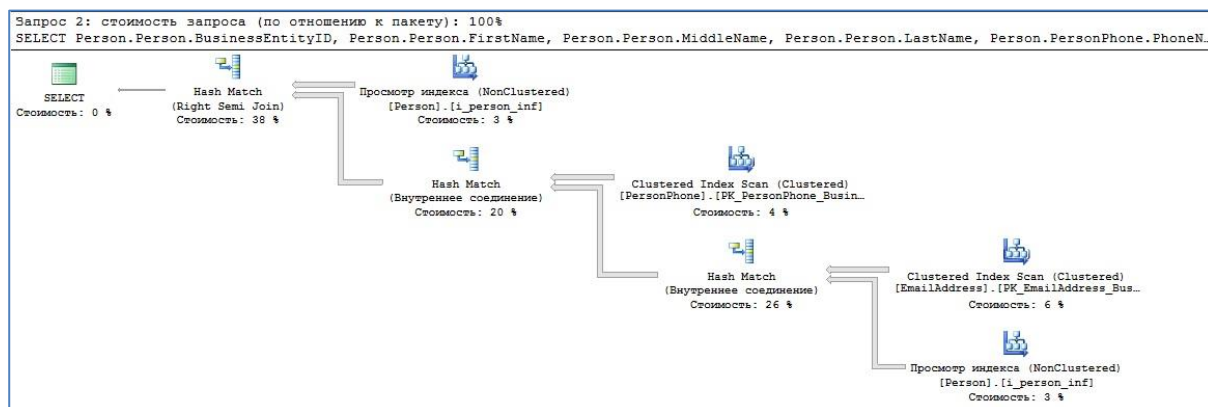


Рис. 28. Предполагаемый план выполнения запроса

#### 15. Отобразим действительный план выполнения запроса.

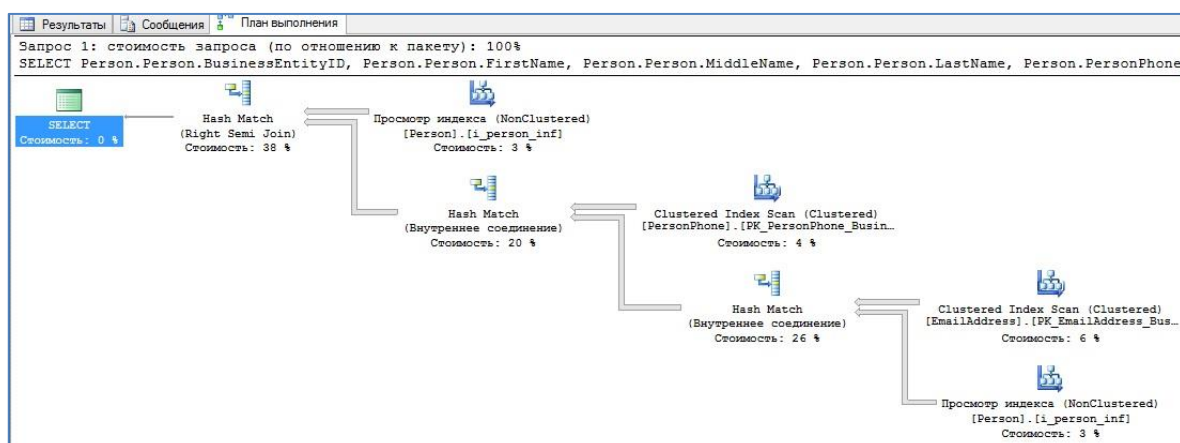


Рис. 29. Действительный план выполнения

#### 16. Измерим время выполнения запроса на выборку.

Таблица 6

#### Время выполнения запроса с покрывающими индексами

№ п/п	Время выполнения запроса, мс
1	0,109
2	0,093
3	0,125
4	0,093
5	0,109
6	0,087
7	0,097
8	0,108
9	0,109
10	0,079
Среднее	0,1009



**Сводная таблица**

Вид используемого индекса	Среднее время выполнения запроса, мс
Все индексы отключены	-
Кластеризованный индекс	0,1115
Некластеризованный индекс	0,1028
Покрывающий индекс	0,1009

**Вывод**

При отключении кластеризованного индекса также отключаются ограничения данной таблицы и всех связанных с ней таблиц, поэтому при попытке выполнения запроса SQL Server выводит сообщение о том, что кластеризованный индекс отключен.

Из сводной табл. 6 видно, что использование индексов снижает среднее время выполнения запросов. При создании покрывающего индекса увеличивается скорость выполнения запроса, так как оптимизатор запросов может определить местонахождение всех значений столбцов на диске по страницам индекса без обращения к данным в таблице.

## ГЛАВА 5. ПОЛЬЗОВАТЕЛИ И РОЛИ

### 5.1. Теоретическая часть

Система безопасности SQL Server ориентируется на четыре важных компонента, обеспечивающих защиту данных.

#### 1. Аутентификация

Принцип действия аутентификации основан на получении ответа на вопрос: «Имеет ли данный пользователь законное право на доступ к системе?» В этом случае происходит проверка подлинности учетных данных пользователя с целью не допустить вход в систему несанкционированных пользователей. Проверка осуществляется на основе запросов, задаваемых пользователю, входящему в систему. Данные запросы могут потребовать дать информацию:

- о том, что известно пользователю (к примеру, пароль);
- о том, что принадлежит пользователю (пропускная карта);
- физические характеристики пользователя (подпись).

Как правило, для проверки подлинности личности используют имя пользователя и пароль, но для более надежной защиты от взлома данную проверку можно усложнить с помощью шифрования.

#### 2. Шифрование

Данный процесс основан на кодировании информации таким образом, чтобы она стала непонятной внешнему пользователю. А для ее декодирования необходим ключ, который есть только у законных пользователей системы.

#### 3. Авторизация

Данный процесс применяется к пользователям, прошедшим аутентификацию. Смысл данного компонента состоит в определении прав пользователя на использование определенных ресурсов.

#### 4. Отслеживание изменений

Данный процесс основан на отслеживании и определении действий, совершаемых несанкционированными пользователями. Иначе говоря, все

действия, совершаемые над базой данных (удаление, изменение, вставка, обновление), документируются. В дальнейшем проведенные операции могут быть проверены авторизованными пользователями, и в случае осуществления подозрительных операций, несанкционированный вход в систему будет выявлен.

Модель безопасности, применяемая в SQL Server, состоит из трех разных категорий, которые взаимодействуют друг с другом [8].

### 1. Принципы

Данная категория предполагает наличие субъектов, у которых есть разрешение на доступ к определенным объектам базы данных. К типичным принципам относят учетные записи Windows и SQL Server. Также существуют группы Windows и роли SQL Server. Группа Windows – это коллекция учетных записей и групп Windows. Пользователь, получая учетную запись, получает все разрешения, предоставленные данной группе. Таким образом, коллекция учетных записей формирует роль.

### 2. Защищаемые объекты

Защищаемые объекты – это все ресурсы, доступ к которым регулируется через авторизацию к БД.

Существует три основных класса защищаемых объектов: сервер, база данных и схема, которые содержат другие защищаемые объекты, такие как регистрационные имена, пользователи базы данных, таблицы и хранимые процедуры.

Как правило, большинство защищаемых объектов создают иерархию (некоторые из объектов могут быть заключены в другие объекты). Также защищаемые объекты могут иметь определенное число разрешений, применимых к ним.

### 3. Разрешения

Каждый защищаемый объект имеет связанные с ним разрешения, которые могут быть предоставлены принципалу.

Рассматривая компонент Database Engine и его систему безопасности, можно выделить две разные подсистемы безопасности.

### 1. Система безопасности Windows

Данная система предполагает безопасность на уровне операционной системы. Иначе говоря, пользователи могут войти в систему баз данных, используя свою учетную запись Windows. Аутентификация посредством данного метода называется аутентификацией Windows.

### 2. Система безопасности SQL Server

Данная система позволяет обеспечивать дополнительную безопасность, требуемую на уровне системы баз данных. Система обеспечивает способ подключения пользователя, уже вошедшего в операционную систему, к серверу баз данных. Система безопасности SQL Server определяет регистрационное имя входа (login) в SQL Server, которое создается в системе и ассоциируется с определенным паролем. Часто бывает, что регистрационные имена в SQL Server совпадают с учетными записями Windows. Аутентификация посредством этой подсистемы также называется аутентификацией SQL Server.

Схема – это коллекция объектов базы данных, имеющая одного владельца и формирующая одно пространство имен. В одной схеме два объекта не могут иметь одинаковое имя. При этом Database Engine поддерживает именованные схемы с использованием понятия принципала. Принципалом может быть любой индивидуальный или групповой принципал.

1. Индивидуальный принципал – один пользователь (к примеру, учетная запись записи пользователя Windows).

2. Групповой принципал – группа пользователей (к примеру, роль или группа Windows).

Необходимо отметить, что хотя принципалы и владеют схемой, в любой момент ее владение может быть передано другому принципалу без изменения схемы.

Отделение пользователей базы данных от схем дает значительные преимущества, среди которых:

- один принципал может быть владельцем нескольких схем;
- несколько индивидуальных принципалов могут владеть одной схемой (добиться этого можно с помощью членства в группах или ролях Windows);
- при удалении пользователя базы данных не возникает необходимости переименовывать объекты, содержащиеся в схеме.

В случае, если пользователям необходимо выполнять действия в определенной базе данных, но они не являются членами соответствующей группы, есть возможность воспользоваться ролью базы данных, задающей группу пользователей базы данных, которые могут иметь доступ к одним и тем же объектам базы данных.

Членами роли базы данных могут быть:

- группы и учетные записи Windows;
- регистрационные имена входа в SQL Server;
- другие роли.

Рассматривая структуру безопасности компонента Database Engine, можно отметить, что у нее есть несколько «системных ролей», имеющих специальные разрешения. Кроме того существует два типа предопределенных ролей:

- фиксированные серверные роли;
- фиксированные роли базы данных.

Для выполнения задач, связанных с авторизацией применяются последующие три инструкции языка Transact-SQL.

1. GRANT. Данная инструкция языка Transact-SQL позволяет принципалам получить разрешения на защищаемые объекты.

2. DENY. Данная инструкция запрещает участнику выполнять определенные действия над объектами, удаляя существующие разрешения для учетной записи пользователя, а также предотвращает наследование разрешения участником через его членство в группе или роли.

3. REVOKE. Данная инструкция удаляет определенное разрешение для объекта безопасности.

## 5.2. Лабораторная работа № 3

### «Пользователи и роли. Представления. Процедуры»

**Цель работы:** изучить механизмы создания пользователей и управления их правами; освоить создание и использование представлений и процедур.

**Задание кафедры:**

1. Использовать учебную базу данных AdventureWorks.
2. Переключиться в режим проверки подлинности SQL Server.
3. Создать пользователя с произвольным именем '*user\_name*'.
4. Перезапустить сервер.
5. Переключиться на пользователя *user\_name*.
6. Убедиться, что пользователь *user\_name* не может выполнить ни один запрос.
7. Переключиться на пользователя *master* и выдать пользователю *user\_name* право на выполнение запросов.
8. Переключиться на пользователя *user\_name* и убедиться, что пользователь *user\_name* может выполнить запрос, разрешенный в пункте 7.
9. Переключиться на пользователя *master* и забрать права у пользователя *user\_name*.
10. Создать представление.
11. Переключиться на пользователя *user\_name* и убедиться, что пользователь *user\_name* не может выполнить представление.
12. Переключиться на пользователя *master* и выдать пользователю *user\_name* право на выполнение представления.
13. Переключиться на пользователя *user\_name* и убедиться, что пользователь *user\_name* может выполнить представление.
14. Создать процедуру на добавление 5000 случайных записей в одну из таблиц.
15. Выполнить процедуру пользователем *user\_name*.
16. Сделать выводы.

### 5.3. Пример выполнения лабораторной работы № 3

1. Для выполнения лабораторной работы необходимо скачать учебную базу данных AdventureWorks2008. Далее требуется присоединить базу к Microsoft SQL Server 2012. Для этого необходимо в Обозревателе объектов выбрать: *Базы данных* → *Присоединить* (правой кнопкой мыши).

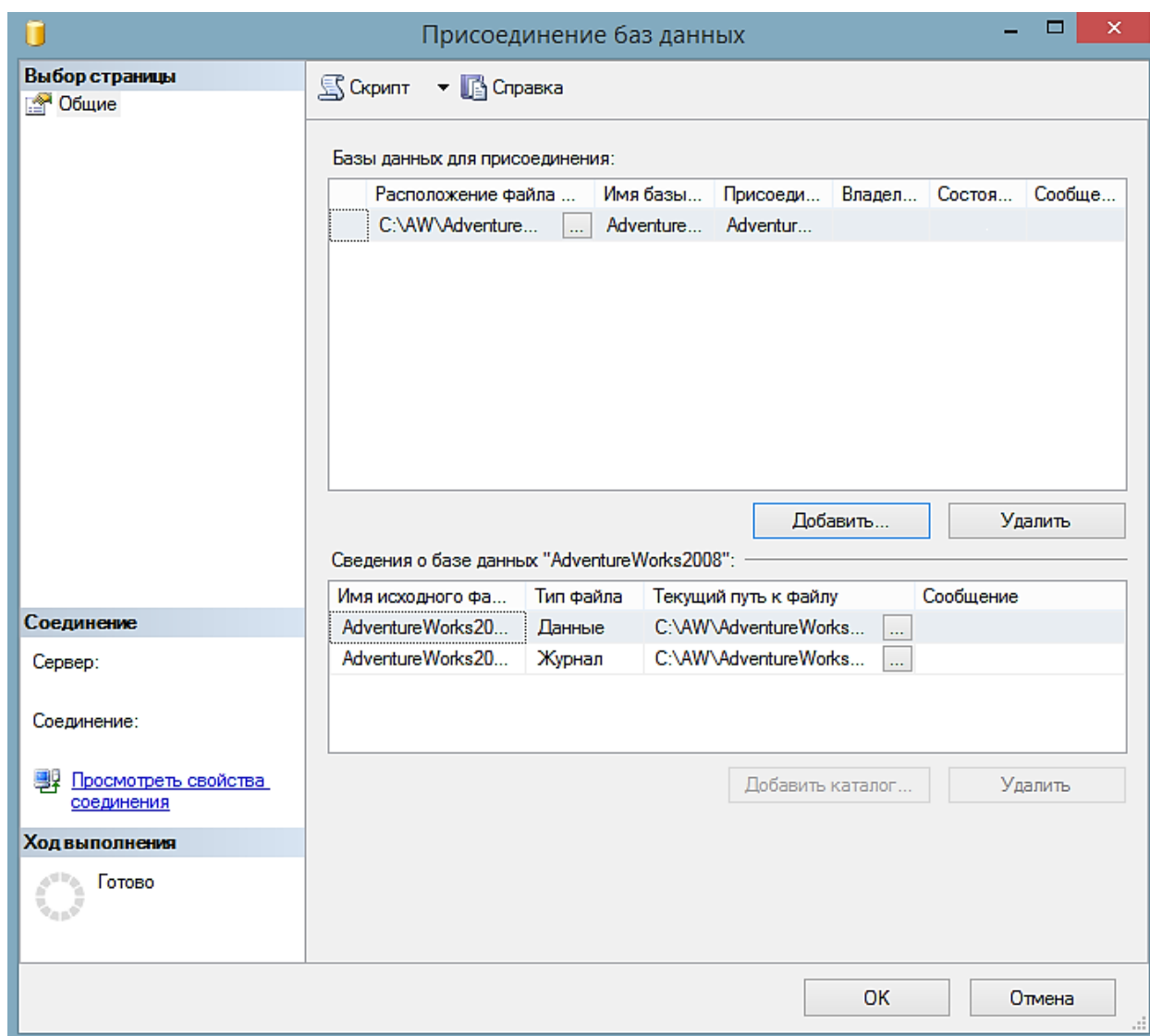


Рис. 30. Присоединение базы данных

Присоединённую базу данных можно увидеть в Обозревателе объектов (рис 31).

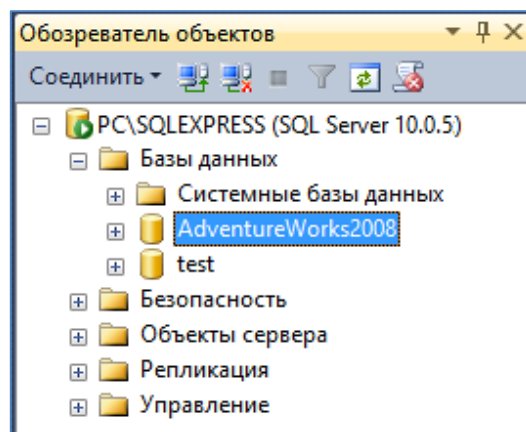


Рис. 31. Присоединение базы данных

2. Для переключения в режим проверки подлинности SQL Server в Обозревателе объектов необходимо проделать следующие действия: *Сервер* → *Свойства* (правой кнопкой мыши) → *Безопасность*.

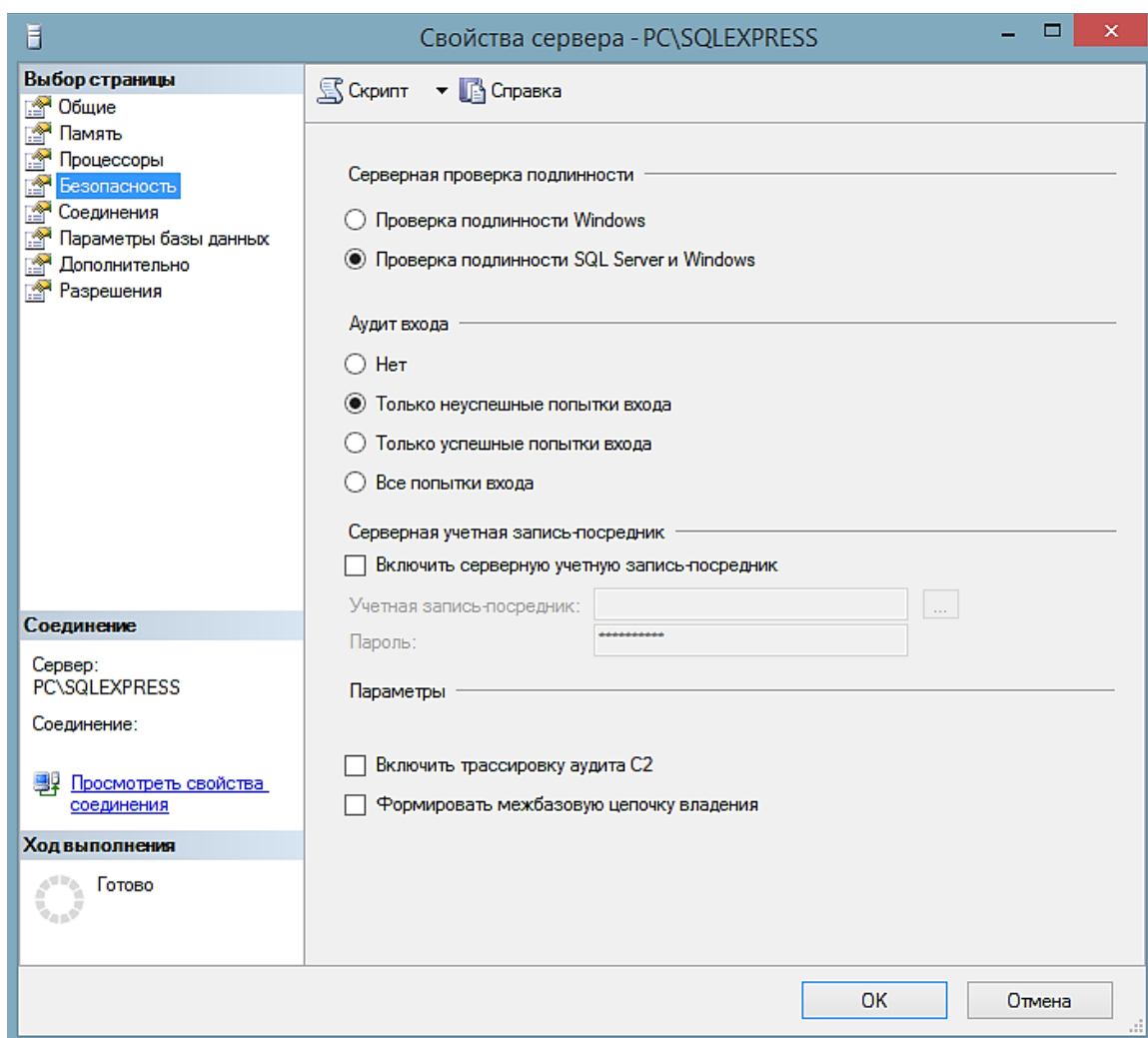


Рис. 32. Выбор режима проверки подлинности



3. Создадим пользователя *user\_2* с соответствующими полями: логин (*user\_1*) и пароль (*123456*).

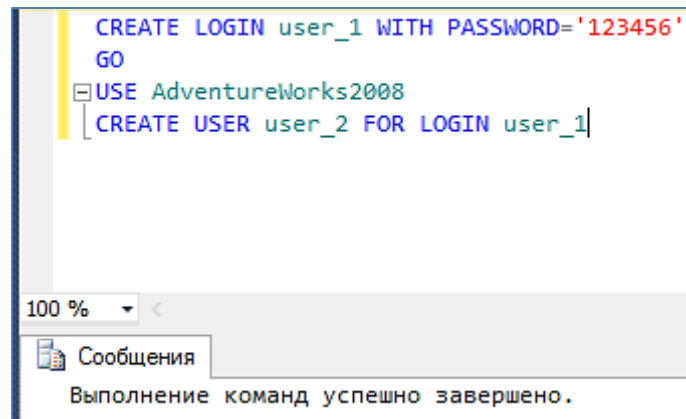


Рис. 33. Создание нового пользователя

4. Для перезапуска сервера необходимо в Обозревателе объектов нажать правой кнопкой мыши на активном соединении и выбрать пункт контекстного меню «Перезапустить». В Обозревателе объектов можно увидеть, что пользователь создан.

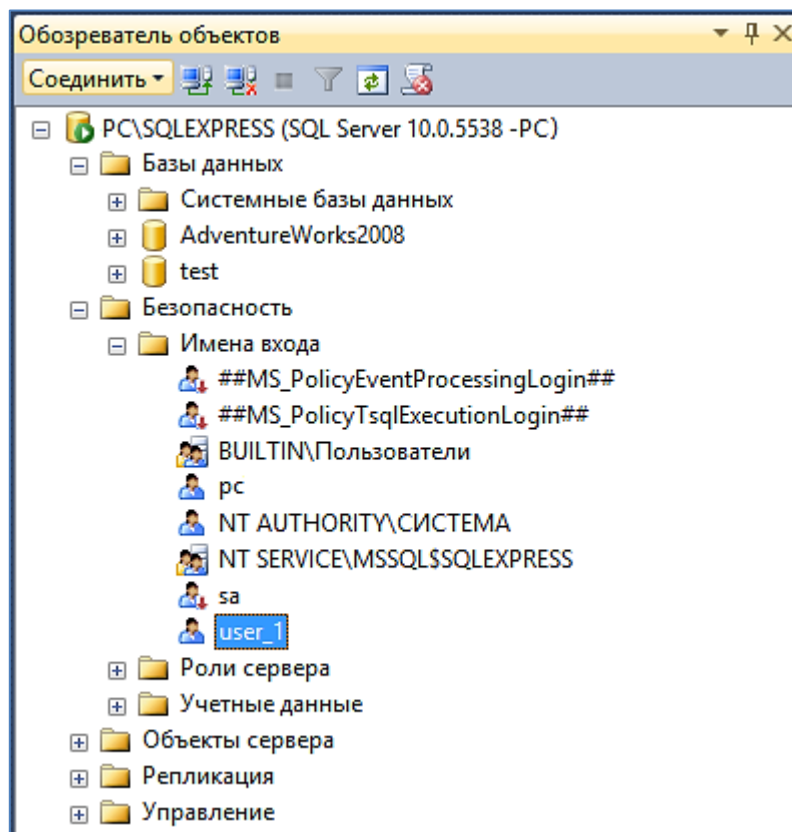



Рис. 34. Результат создания нового пользователя

5. Далее необходимо переключиться из пользователя *master* на созданного пользователя *user\_2*. Для этого нужно нажать кнопку  в обозревателе объектов, в появившемся соединении с сервером выбрать проверку подлинности SQL Server и ввести соответствующие логин и пароль.

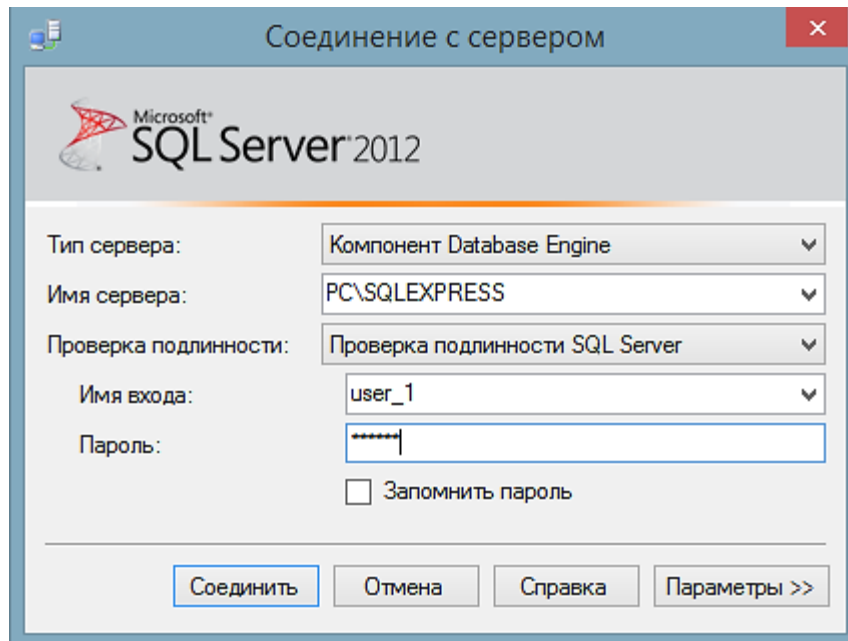


Рис. 35. Переключение пользователей

6. Необходимо выполнить запрос к базе данных и убедиться, что у созданного пользователя нет прав просматривать произвольную таблицу.

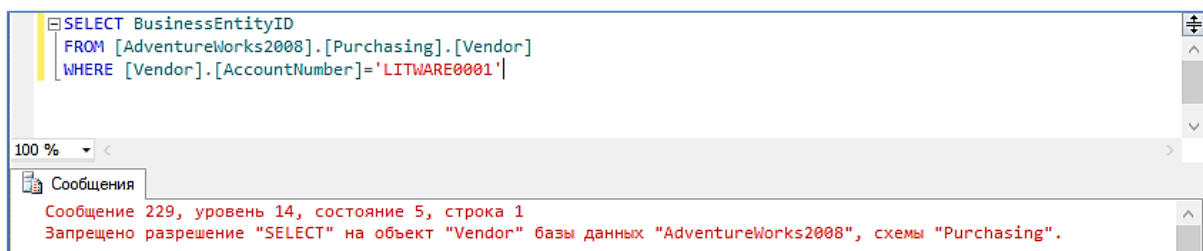


Рис. 36. Запрос на выборку из произвольной таблицы

На рис. 36 видно, что у пользователя *user\_2* нет прав просматривать данную таблицу.

7. В режиме *master* наделим пользователя *user\_2* правом доступа.

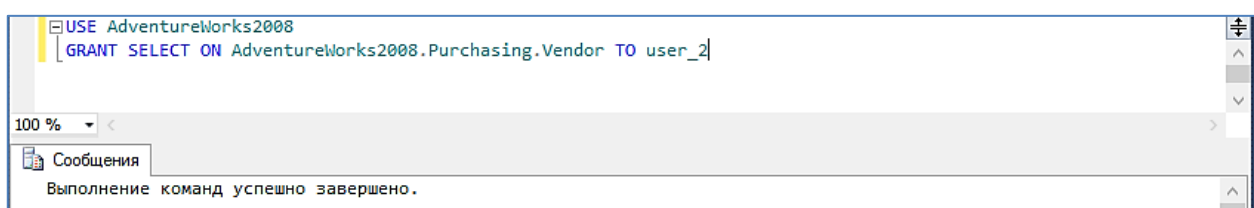


Рис. 37. Предоставление права доступа к таблице

8. При переключении в режим пользователя *user\_2* видно, что теперь SELECT-запрос успешно выполнен (рис. 38).

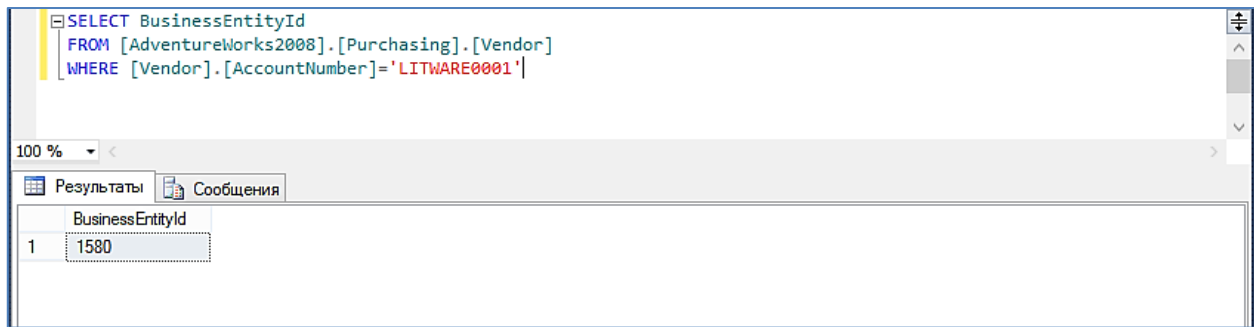


Рис. 38. Успешное выполнение запроса

9. Переключимся в режим пользователя *master* и заберем права у пользователя *user\_2*.

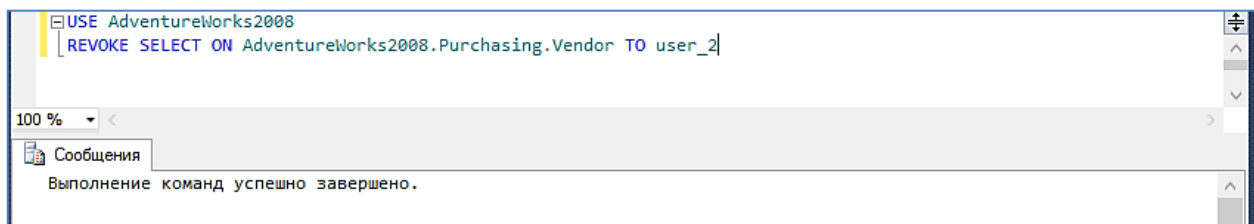


Рис. 39. Изъятие прав пользователя

В результате этого при попытке выполнения запроса к данной таблице пользователем *user\_2* возникнет ошибка, представленная на рис. 40.

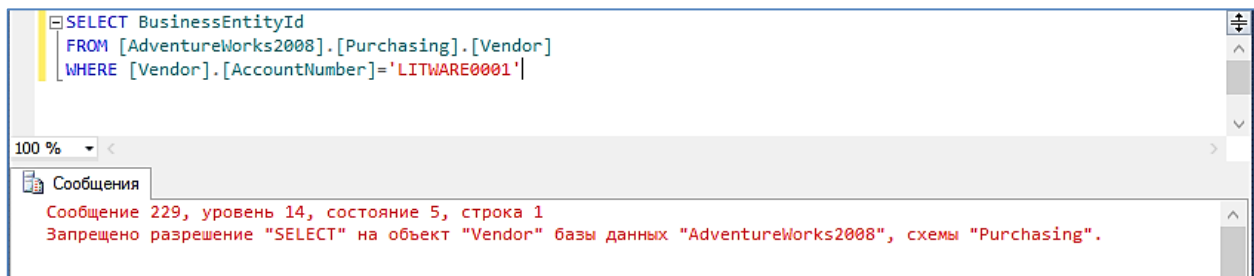


Рис. 40. Отсутствие прав доступа к таблице

10. Переключимся в режим пользователя *master* и создадим представление *v\_Purchasing\_Vendor*.

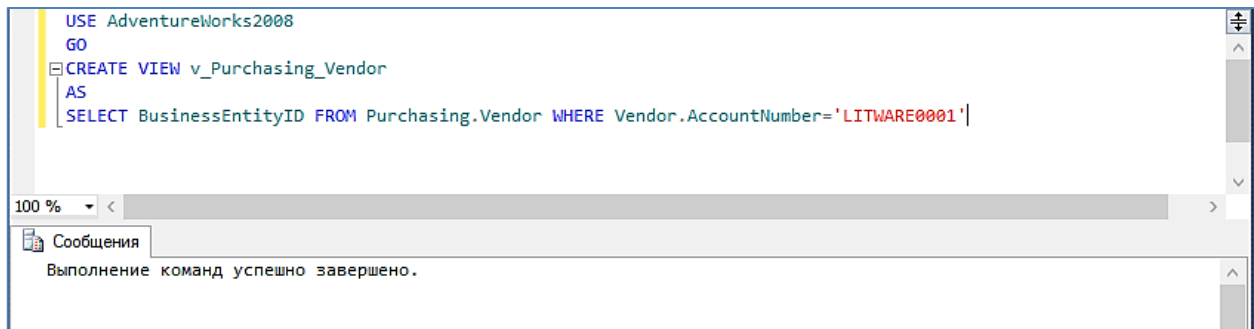


Рис. 41. Создание представления

11. Теперь необходимо наделить пользователя *user\_2* правами доступа к представлению.

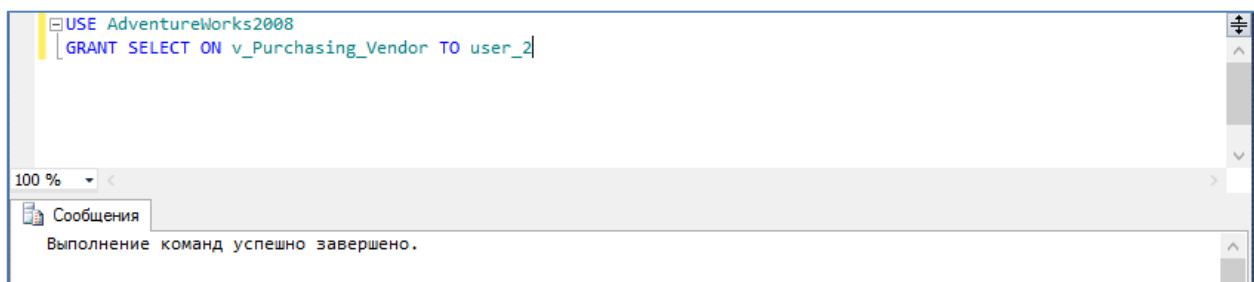


Рис. 42. Наделение пользователя правом доступа к представлению

12. Переключимся в режим пользователя *user\_2* и убедимся, что он имеет возможность вызвать представление.

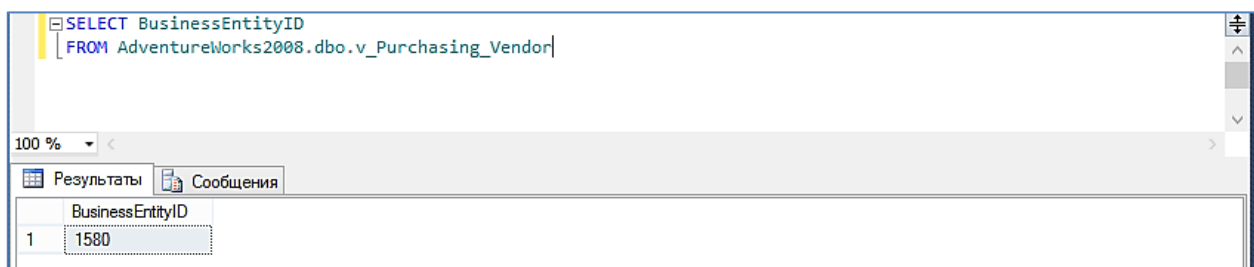


Рис. 43. Вызов представления

13. Переключимся в режим пользователя *master* и создадим процедуру, которая добавляет в таблицу 5000 строк.

```
USE AdventureWorks2008;
GO
CREATE PROCEDURE y
AS
DECLARE @i INT
SET @i=0
BEGIN
WHILE (@i<=5000) BEGIN INSERT INTO HumanResources.Shift(ShiftID, Name, StartTime,EndTime, ModifiedDate)
VALUES (3+@i, 'Night', '23:00:00.0000000', '07:00:00.0000000', '2002-06-01 00:00:00.000') SET @i=@i+1 END
END
```

Сообщения  
Выполнение команд успешно завершено.

Рис. 44. Процедура для вставки 5000 строк в таблицу

	ShiftID	Name	StartTime	EndTime	ModifiedDate
1	1	Day	07:00:00.0000000	15:00:00.0000000	2002-06-01 00:00:00.000
2	2	Evening	15:00:00.0000000	23:00:00.0000000	2002-06-01 00:00:00.000
3	3	Night	23:00:00.0000000	07:00:00.0000000	2002-06-01 00:00:00.000
4	4	Night	23:00:00.0000000	07:00:00.0000000	2002-06-01 00:00:00.000
5	5	Night	23:00:00.0000000	07:00:00.0000000	2002-06-01 00:00:00.000

Запрос успешно выполнен.

Рис. 45. Результат выполнения процедуры

## Вывод

В ходе выполнения лабораторной работы был создан новый пользователь с индивидуальными правами на пользование базой данных. В зависимости от того, как изменялись эти права, можно было видеть, как менялся доступ к данным у данного пользователя. Рассмотрен механизм создания и выполнения процедур и представлений.

## ГЛАВА 6. РЕЗЕРВНОЕ КОПИРОВАНИЕ И ВОССТАНОВЛЕНИЕ

### 6.1. Теоретическая часть

*Резервное копирование (backup)* – процесс создания копии базы данных и журналов транзакций на отдельных носителях с целью восстановления исходных данных в случае их повреждения или утраты.

*Восстановление (recovery)* – процесс замены неподтвержденных, несогласованных или потерянных данных данными с устройства, содержащего резервную копию.

*Доступность системы (system availability)* – свойство, направленное на минимизацию простоев СУБД.

Создание резервных копий является мерой предосторожности, которая осуществляется с целью предотвращения потери данных.

Выделяют пять основных причин потери или повреждения данных:

- программные ошибки;
- аппаратные ошибки (неисправность дискового накопителя);
- человеческий фактор (ошибки администратора);
- стихийное бедствие (пожар, наводнение, землетрясение);
- кража.

В процессе выполнения программы возможно возникновение условий, вызывающих её аварийное завершение. Такие программные ошибки связаны только с приложениями баз данных и обычно не распространяются на всю систему базы данных. Поскольку причиной таких ошибок является неисправность логики программы, то система базы данных в таких случаях не может выполнить восстановление. Поэтому восстановление должен выполнять программист, используя для обработки таких исключений инструкции COMMIT и ROLLBACK.

Другой причиной потери данных является «человеческий фактор». Пользователи, имеющие достаточно большие полномочия (например, администраторы базы данных), случайно могут удалить или исказить данные

(удалить «не ту» базу данных, обновить или удалить «не те» данные). Конечно, в идеальном случае такое никогда не должно происходить, и необходимо стараться настроить такой режим работы, при котором вероятность потери производственных данных будет минимизироваться. Однако следует учитывать тот факт, что людям свойственно ошибаться, и данные будут теряться вследствие их ошибок. Самое лучшее, что можно сделать в этом отношении, – это предпринять все возможные меры для предотвращения потери данных вследствие человеческого фактора и иметь возможность восстановить данные в случае возникновения такой ситуации.

Сбой в работе компьютера может произойти вследствие различных ошибок в программном или аппаратном обеспечении. В качестве одного примера системного сбоя можно назвать выход из строя оперативной памяти компьютера, вследствие чего ее содержимое будет потеряно. Другим примером можно назвать неисправность жесткого диска, вызванную выходом из строя головки чтения/записи, или когда система ввода/вывода в процессе записи либо чтения данных обнаруживает поврежденные сектора на диске.

В случае стихийного бедствия или кражи необходимо иметь в наличии достаточное количество зарезервированной информации для восстановления потерянных данных. Это обычно достигается хранением носителей с резервной информацией в удалённом от другого оборудования месте, таким образом предохраняя их от потери данных в случае кражи и стихийного бедствия.

Выполнение резервного копирования может обеспечить восстановление утраченных данных в большинстве случаев описанных сбоев.

Создание резервной копии базы данных представляет собой копирование данных (из базы данных, журнала транзакций или файла) на устройства резервного копирования. Компонент Database Engine поддерживает как статическое, так и динамическое резервное копирование. Статическое резервное копирование означает, что в процессе создания резервной копии единственным активным сеансом, поддерживаемым системой, является сеанс, который создает данную резервную копию. Таким образом, выполнение

пользовательских процессов в течение операции резервного копирования не разрешается. Динамическое резервное копирование означает, что резервная копия может создаваться без прекращения работы сервера базы данных, отключения пользователей или даже закрытия файлов. Более того, пользователи даже не будут подозревать, что в момент их работы с данными выполняется резервное копирование.

Компонент Database Engine поддерживает четыре метода создания резервных копий [8]:

- полное резервное копирование базы данных;
- разностное (дифференциальное) резервное копирование;
- резервное копирование журнала транзакций;
- резервное копирование файлов или файловых групп.

1. *Полное резервное копирование базы данных (full database backup)* фиксирует то состояние базы данных, которое она имела на момент начала выполнения резервного копирования. В процессе создания такой резервной копии система копирует как сами данные, так и схему всех таблиц базы данных и соответствующие файловые структуры. Если резервное копирование выполняется динамически, то копируются все действия, происходящие в процессе создания резервной копии. Таким образом, в резервную копию попадают даже неподтвержденные незафиксированные транзакции.

2. *Разностное (дифференциальное) резервное копирование (differential backup)* создает копию только тех частей базы данных, которые были добавлены или изменены после выполнения последнего полного резервного копирования базы данных. Как и в случае полного резервного копирования, любая деятельность во время динамического разностного резервного копирования также заносится в создаваемую копию. Достоинством разностного резервного копирования является скорость его выполнения. Для создания такой резервной копии базы данных требуется существенно меньше времени, чем для полной копии, так как объем копируемых данных значительно меньше, поскольку полное резервное копирование базы данных включает копии всех ее страниц.



### 3. Резервное копирование журнала транзакций (*transaction log backup*)

учитывает только те данные, которые связаны с изменениями, записанными в журнал транзакций. Этот способ резервного копирования основан не на физических составляющих базы данных (страницах данных), а на логических операциях, то есть на изменениях, выполненных посредством DML-инструкций: INSERT, UPDATE и DELETE. Так как при этом копируется меньший объем данных, то процесс создания резервной копии может быть выполнен значительно быстрее, чем полное или разностное резервное копирование базы данных.

Не имеет смысла выполнять разностное или резервное копирование журнала транзакций, если не было проведено по крайней мере одно полное резервное копирование базы данных.

Существует две основные причины, по которым следует выполнять резервное копирование журнала транзакций. Первая, чтобы сохранить на защищенном носителе данные, которые изменились со времени последнего резервного копирования журнала транзакций, а вторая, более важная, чтобы должным образом закрыть часть журнала транзакций перед началом новой порции действий его активной части. Активная часть журнала транзакций содержит записи обо всех неподтвержденных незафиксированных транзакциях.

Используя полное резервное копирование базы данных и цепочки всех резервных копий журналов транзакций, можно распространить копию базы данных и на другой компьютер. Эту копию базы данных можно использовать для восстановления исходной базы данных в случае ее повреждения или полной потери. Подобным образом базу данных можно восстановить, используя полную резервную копию и последнюю разностную резервную копию.

Компонент Database Engine не позволяет сохранять журнал транзакций в том же самом файле, в котором сохраняется база данных, так как, в случае повреждения этого файла, невозможно будет использовать журнал транзакций для восстановления всех изменений, выполненных после последнего резервного копирования.

Использование журнала транзакций для записи изменений в базе данных является достаточно широко распространенной функциональной возможностью, которая применяется почти во всех реляционных СУБД. Тем не менее, иногда могут возникнуть ситуации, в которых будет полезным эту возможность отключить. Например, обработка большого объема загружаемых данных может длиться несколько часов. В таком случае скорость выполнения программы можно увеличить, отключив протоколирование транзакций. Но с другой стороны, это чревато опасными последствиями, поскольку при этом разрушаются существующие цепочки журналов транзакций. Чтобы обеспечить успешное восстановление базы данных, после окончания загрузки данных настоятельно рекомендуется выполнить полное резервное копирование базы данных.

Одной из самых распространенных причин системных сбоев является переполнение журнала транзакций. Следует иметь в виду, что такая проблема может полностью остановить систему. Если дисковое пространство, выделенное для хранения журнала транзакций, полностью заполнится, то система будет вынуждена прекратить выполнение всех текущих транзакций до тех пор, пока для журнала не будет предоставлено дополнительное пространство. Этой проблемы можно избежать, только выполняя частое резервное копирование журнала транзакций: при каждом закрытии части журнала транзакций и сохранении его на другой носитель эта часть журнала становится повторно используемой, тем самым освобождая дисковое пространство.

Как разностное резервное копирование, так и резервное копирование журнала транзакций сокращают время, требуемое для создания резервной копии базы данных. Но между ними есть одно важное различие: резервная копия журнала транзакций содержит все множественные изменения строк таблиц данных, выполненные после последнего резервного копирования, тогда как разностная резервная копия содержит только последнее изменение этой строки.

Преимущество разностного резервного копирования заключается в том, что оно позволяет сэкономить время при восстановлении, поскольку для полного восстановления базы данных требуется ее полная резервная копия и

только последняя разностная резервная копия. Для полного же восстановления с использованием резервных копий журнала транзакций необходимо использовать полную резервную копию базы данных и все резервные копии журнала транзакций.

Недостатком разностного резервного копирования является то, что такая копия не позволяет восстановить данные на определенный момент времени, так как в ней не сохраняются промежуточные изменения базы данных.

4. *Резервное копирование файлов или файловых групп* позволяет вместо полной резервной копии базы данных создать резервную копию только определенных файлов (или файловых групп) базы данных. В таком случае компонент Database Engine создает резервную копию только указанных файлов. Из резервной копии базы данных можно восстанавливать отдельные файлы (или файловые группы), что позволяет выполнять восстановление данных после сбоя, который повлиял только лишь на часть файлов базы данных. Восстановление отдельных файлов или файловых групп можно выполнять из полной резервной копии базы данных или из резервной копии файловой группы. Это означает, что в качестве процедуры резервного копирования можно применять полное резервное копирование базы данных и резервное копирование журнала транзакций и в то же время иметь возможность восстанавливать отдельные файлы (или файловые группы) из этих резервных копий.

Резервное копирование файлов также называется резервным копированием на уровне файлов. Данный тип резервного копирования рекомендуется применять только в случае очень большой базы данных, когда нет времени достаточного для выполнения полного резервного копирования.

Резервное копирование производственных баз данных следует выполнять на регулярной основе. Кроме этого, следует выполнять резервное копирование любой рабочей базы данных после выполнения следующих действий:

- после создания базы данных;
- после создания индексов;
- после очистки журнала транзакций;

– после выполнения незапротоколированных операций.

Всегда следует выполнять полное резервное копирование сразу же после создания базы данных на случай сбоя в период времени между ее созданием и ее первым запланированным резервным копированием. Резервное копирование базы данных после создания одного или большего количества индексов позволяет сэкономить время при восстановлении, поскольку вместе с данными копируются и индексные структуры. Но резервное копирование журнала транзакций после создания индексов такой экономии не предоставляет, так как в журнале транзакций записывается только факт создания индекса, а сама измененная структура индекса не записывается. Резервное копирование базы данных после очистки журнала транзакций необходимо выполнять по той причине, что журнал больше не содержит записей операций с базой данных, которые используются для восстановления базы данных. Все операции, которые не записаны в журнале транзакций, называются *незапротоколированными операциями*. Поэтому все изменения, выполненные этими операциями, не могут быть воссозданы в процессе восстановления.

*Массив дисков RAID* (англ. «Redundant Array of Independent Disk» – «избыточный массив независимых жестких дисков»). Массив дисков RAID представляет собой набор из двух или более жестких дисков, составляющих единую логическую единицу. Данный способ позволяет располагать части файлов не на одном жестком диске, а на нескольких. При этом система видит файл как единое целое. Достоинство такой технологии состоит в том, что она позволяет повысить надежность, несмотря на понижение производительности.

Система RAID может быть реализована двумя способами:

- аппаратным образом (аппаратная система RAID имеет относительно высокую стоимость, так как необходимо приобретать дополнительные носители, но она имеет лучшую производительность);
- программным образом (обычно поддерживается операционной системой).

Данная технология оказывает влияние на следующие свойства:

- устойчивость к ошибкам (отказоустойчивость);
- производительность.

Массив дисков RAID имеет шесть основных уровней (от 0 до 5). Для системы базы данных важными являются только три уровня (0, 1 и 5). Эти три уровня представляют способы защиты от сбоев жестких дисков и, соответственно, потери данных.

#### 1. Уровень 0 – расслоение дисков

RAID 0 определяет чередование (расслоение) дисков без контроля по четности. Данный уровень предполагает, что данные разбиваются на несколько блоков, затем они одновременно записываются на несколько дисков. В результате такой способ позволяет повысить скорость записи и чтения данных. Главным минусом является низкая надежность хранения данных (в случае отказа одного из дисков данные на всех дисках массива становятся недоступными).

#### 2. Уровень 1 – зеркалирование

RAID 1 представляет массив, состоящий из двух дисков – копий друг друга. То есть – это особая форма расслоения дисков, когда на каждый из дисков записываются одинаковые данные. Такая технология позволяет сохранить и получать данные на одном диске в случае сбоя в работе другого. Аппаратная реализация зеркалирования дисков имеет более высокую стоимость, но и более высокую скорость доступа к данным. Помимо этого, при реализации системы RAID 1 аппаратным образом предоставляются возможности кэширования, что повышает пропускную способность массива. В данной технологии, по сравнению с RAID 0, скорость доступа к данным ниже, но надежность – выше, так как данные дублируются. Если говорить об отказоустойчивости, то данная система представляет лучшую ее конфигурацию. Связано это с тем, что даже при выходе из строя половины зеркальных дисков, система может продолжать работать, не требуя выключения сервера и восстановления данных. Помимо всего прочего, зеркалирование может понижать или повышать уровень производительности

операций (чтение и запись). В случае записи данных уровень производительности понижается, потому что выполнение операции удваивается, так как одна осуществляется на основном диске, а вторая – на зеркальном. С другой стороны, уровень производительности операции по чтению информации повышается, так как система может считывать данные с любого диска, в зависимости от того, который из них наименее занят в данный момент.

### 3. Уровень 5 – контроль по четности (англ. «parity check»)

RAID 5 представляет собой массив, в котором реализуется процесс вычисления контрольной суммы записываемых данных, а затем блоки данных распределяются на разные диски на основе данной контрольной суммы. Преимущество данной технологии заключается в том, что если происходит сбой на одном из дисков, то его можно заменить новым, а данные на нем восстановить по контрольным суммам, записанным на другие диски массива. Данная система предполагает наличие только одного добавочного диска вне зависимости от количества основных дисков, используемых в обычной конфигурации. Недостатком данной системы можно назвать то, что данные можно восстановить только в случае сбоя в работе одного диска, если из строя выйдут два диска и более, данные восстановить уже не получится, так как весь массив выйдет из строя. В результате это снижает отказоустойчивость такой конфигурации. Помимо всего прочего, для вычисления и записи информации восстановления требуются дополнительные операции дискового ввода/вывода. Для RAID 5 требуется четыре дисковых операций ввода/вывода, тогда как для RAID 0 только одна, а для RAID 1 две таких операции, что в результате оказывает влияние на уровень производительности.

## **6.2. Лабораторная работа № 4**

### **«Резервное копирование и восстановление»**

**Цель работы:** изучить основные методы создания резервных копий баз данных и освоить методы их восстановления.

#### **Задание кафедры:**

1. Подключить учебную базу данных AdventureWorks.
2. Осуществить полное резервное копирование базы данных (средствами Management Studio и с использованием Transact-SQL кода).
3. Определить размер файла резервной копии и сравнить его с размером исходной базы данных.
4. Внести изменения в базу данных.
5. Выполнить восстановление базы данных (средствами Management Studio и с использованием Transact-SQL кода).
6. Убедиться, что изменения из пункта 4 были отменены.
7. Выполнить разностное копирование базы данных (средствами Management Studio и с использованием Transact-SQL кода).
8. Выполнить пункты 4, 5, 6.
9. Выполнить резервное копирование журнала транзакций.
10. Произвести отказ системы (открыть файл базы данных текстовым редактором, удалить часть строк, сохранить изменения).
11. Убедиться в невозможности открыть базу данных.
12. Выполнить восстановление базы данных.
13. Убедиться, что база данных восстановлена.
14. Сделать выводы.

### 6.3. Пример выполнения лабораторной работы № 4

#### 1. Присоединение базы данных.

Для выполнения работы необходимо скачать учебную базу данных AdventureWorks2012 и присоединить ее к Microsoft SQL Server 2012. Для этого выполняется следующая последовательность действий: *Обозреватель объектов* → *Базы данных* → *Присоединить*.

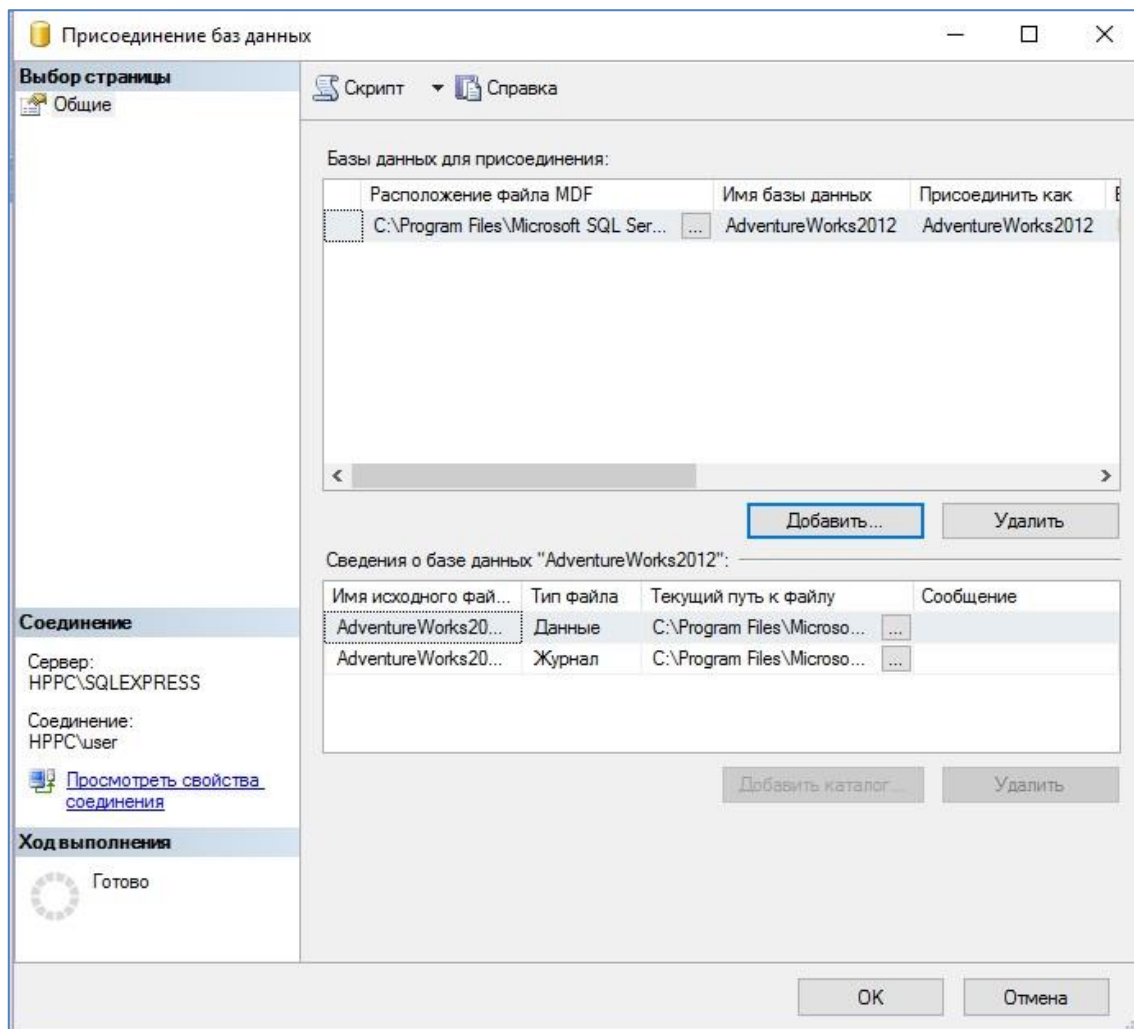


Рис. 46. Присоединение базы данных

В полученном окне, показанном на рис. 46, нажать «Добавить» и указать путь к файлу базы данных. После этого база данных AdventureWorks2012 будет присоединена, ее можно найти в каталоге «Базы данных» окна «Обозреватель объектов» (рис. 47).



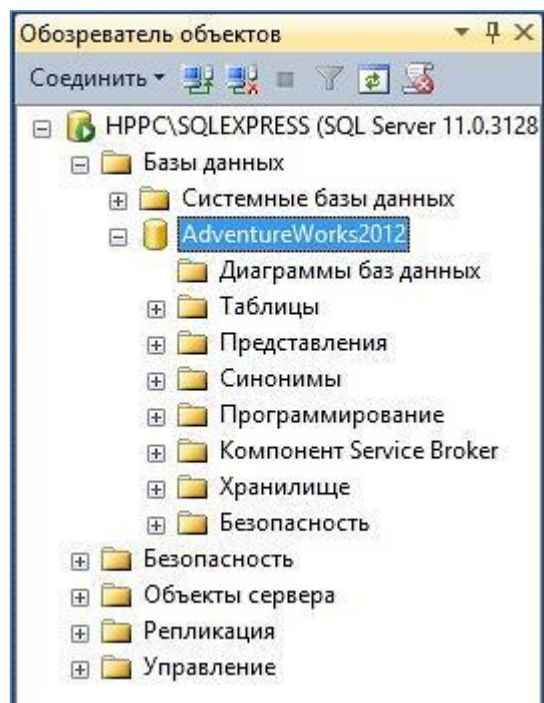


Рис. 47. Присоединенная база данных

2. Полное резервное копирование базы данных выполняется при помощи команды `BACKUP DATABASE` с указанием пути сохранения резервной копии, а также с использованием опции `INIT`, которая указывает, что нужно перезаписать все существующие данные на носителе, за исключением заголовка носителя, если таковой имеется.

Листинг 9

### Полное резервное копирование базы данных

```
BACKUP DATABASE [AdventureWorks2012]
TO DISK = 'D:\AW\AdventureWorks2012.bak'
WITH INIT;
```

Для того чтобы произвести полное резервное копирование базы данных средствами Management Studio, необходимо в *Обозревателе объектов* выбрать узел *Объекты сервера* → *Создать* → *Устройство резервного копирования* и в следующем окне ввести название и путь к папке, куда будет производиться резервное копирование (рис. 48).

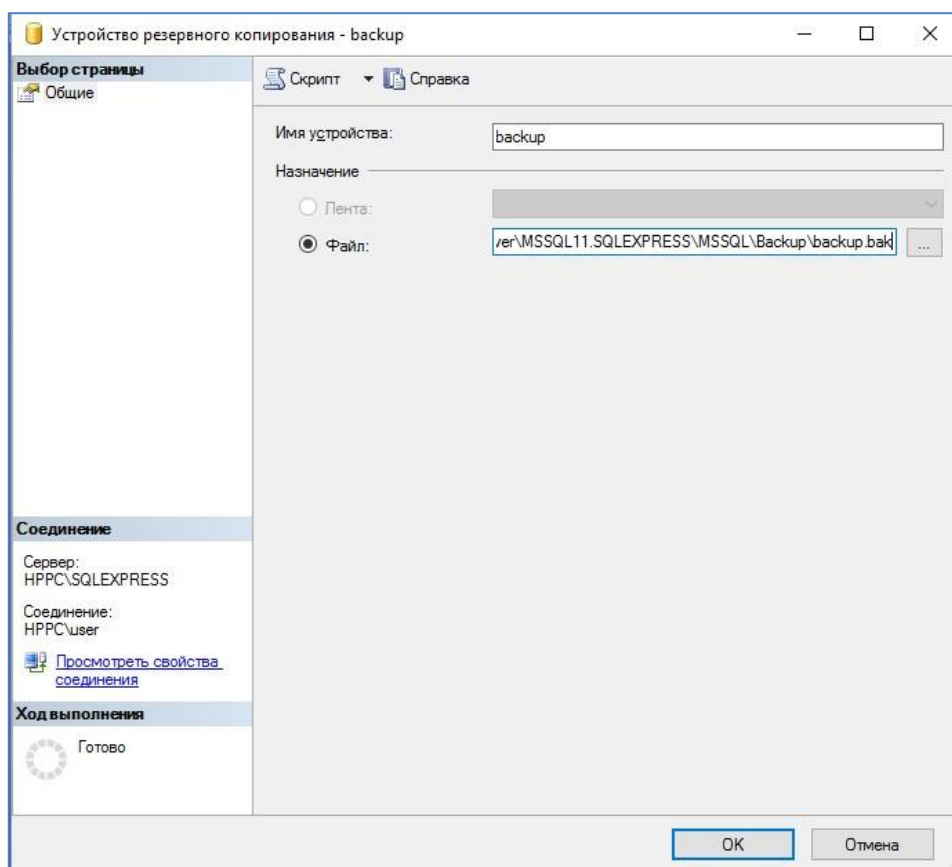


Рис. 48. Создание пути для хранения резервной копии

После создания пути можно выполнять копирование. Для этого необходимо в *Обозревателе объектов* нажать правой кнопкой на *База данных* → *Задачи* → *Создать резервную копию*. В появившемся окне (рис. 49) необходимо указать тип резервного копирования (в данном случае – полное), название копии, описание, а также срок действия резервной копии.

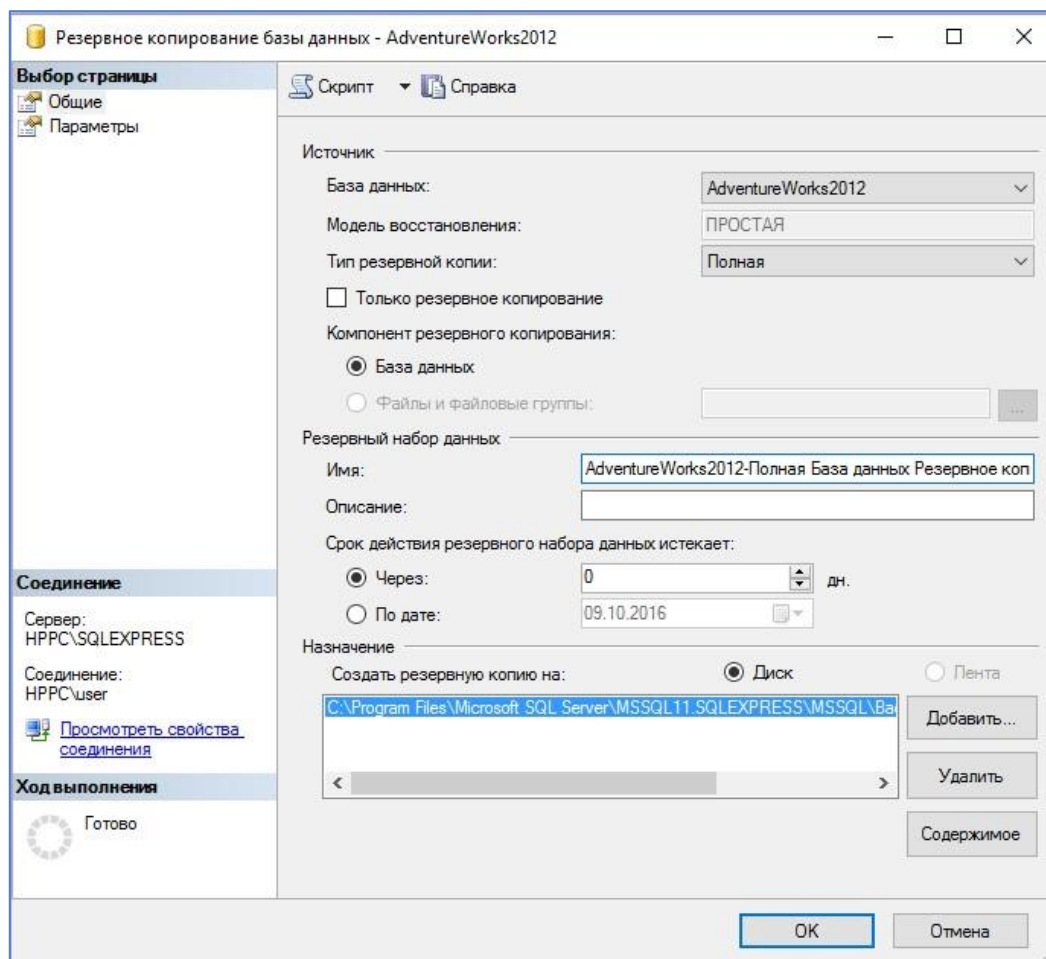


Рис. 49. Полное резервное копирование средствами Management Studio

### 3. Сравнение размеров исходной базы данных и её резервной копии.

Program Files > Microsoft SQL Server > MSSQL11.SQLEXPRESS	
Имя	Размер
AdventureWorks2012_Data.mdf	193 536 КБ

Рис. 50. Исходная база данных

Этот компьютер > RECOVERY (D:) > AW	
Имя	Размер
AdventureWorks2012.bak	193 780 КБ

Рис. 51. Резервная копия базы данных

Изменений в базе данных не происходило, но размер резервной копии больше, чем размер исходной базы данных. Это связано с тем, что в резервной копии хранятся не только данные, но и служебная информация.

#### 4. Внесение изменений в базу данных.

Внесём изменения в базу путем удаления таблицы Person.Password: *База данных* → *Таблицы* → *Удалить*.

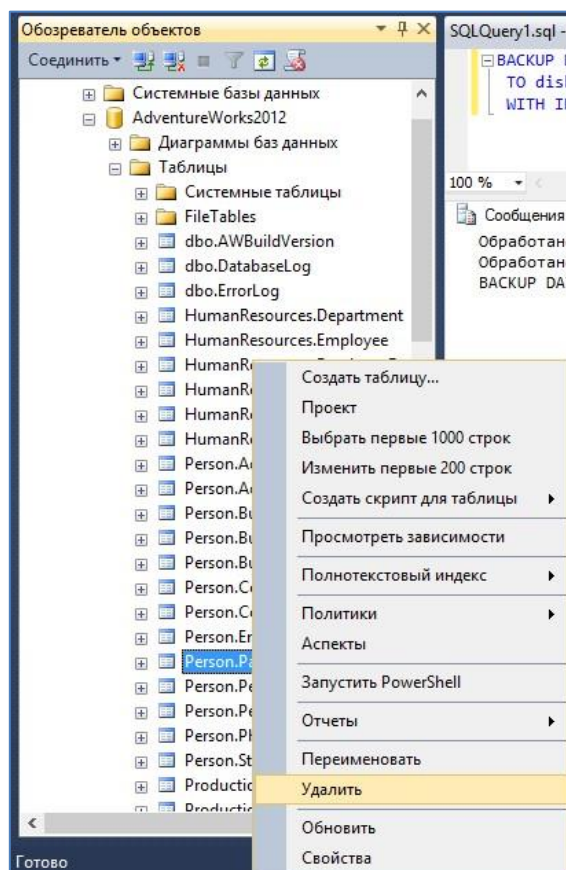


Рис. 52. Удаление произвольной таблицы

#### 5. Восстановление базы данных

Рассмотрим возможность восстановления базы данных двумя способами. Первый способ: восстановление при помощи T-SQL-кода. Восстановление происходит при использовании команды RESTORE с указанием места, откуда будет браться резервная копия повреждённой базы данных, а также при помощи опции REPLACE, которая указывает на замену существующей базы данных данными из резервной копии.

Листинг 10

##### Восстановление базы данных

```
RESTORE DATABASE [AdventureWorks2012]
FROM DISK = 'C:\Program Files\Microsoft SQL Server\MSSQL11.SQLEXPRESS\
MSSQL\Backup\AdventureWorks2012.bak'
WITH REPLACE;
```

Второй способ: восстановление базы данных с помощью среды Management Studio. Для этого в *Обозревателе объектов* необходимо нажать правой кнопкой на *Базы данных* → *Задачи* → *Восстановить* → *Базу данных*. В появившемся окне, представленном на рис. 53, необходимо выбрать базу для восстановления и резервную копию, откуда будет происходить восстановление.

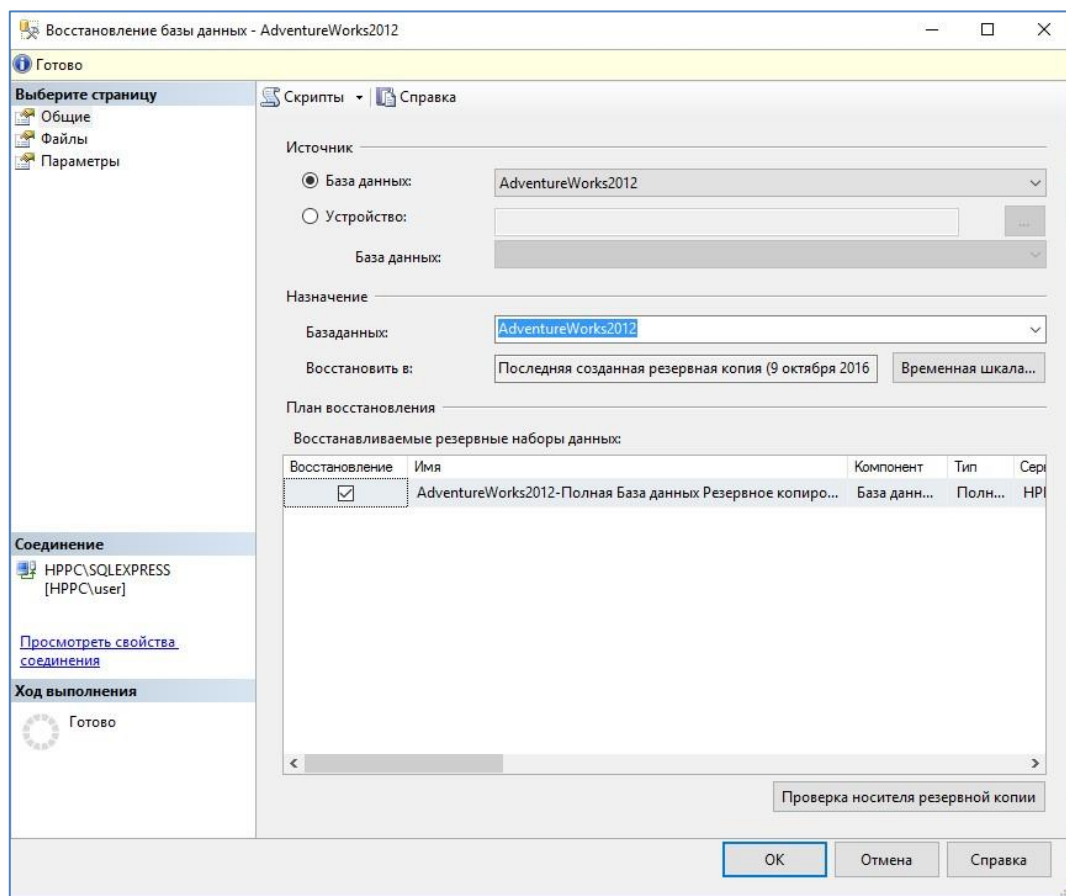


Рис. 53. Восстановление базы данных средствами Management Studio

При необходимости во вкладке *Параметры* можно указать необходимые параметры восстановления. Пример выбранных параметров показан на рис. 54.

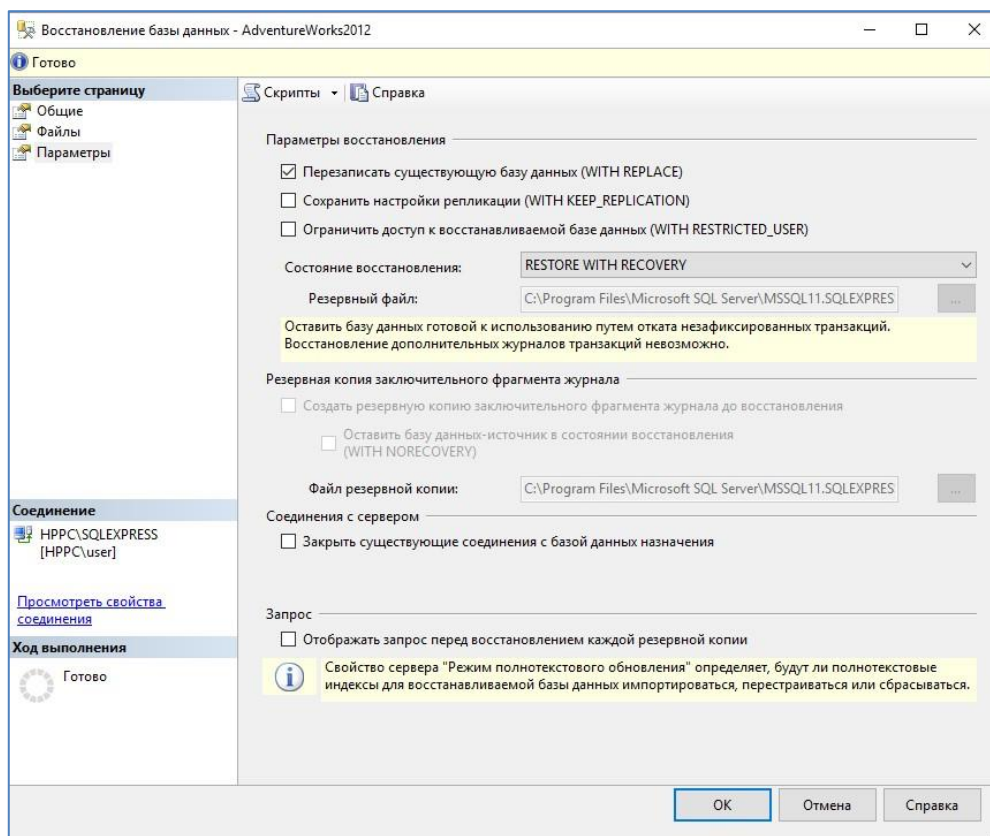


Рис. 54. Выбор необходимых параметров восстановления

## 6. Проверка успешности восстановления.

Теперь для того чтобы проверить, что восстановление произведено и что внесенные изменения отменены, можно выполнить запрос к ранее удаленной таблице Person.Password.

SQLQuery1.sql - HP...2 (HPPC\user (52))

```
USE AdventureWorks2012
SELECT * FROM Person.Password;
```

100 %

	BusinessEntityID	PasswordHash	PasswordSalt	rowguid	ModifiedDate
1	1	pbFwXWE99vobT6g+vwPWfy93NtUU/om/Waf01hcdM=	bE3XWw=	329EACBE-C883-4F48-B8B6-17AA4627EFFF	2003-02-08 00:00:00.000
2	2	bawRVNzZQYQ05qF05Gz6VLnviZmrqBRreTTAGAudm0=	EjJaC3U=	A4C82398-7466-4FE6-B9EE-CEC34D116F68	2002-02-24 00:00:00.000
3	3	8BUxZfDQ01lyHCWOYzYmqN11hTUn3CJMpdX/UCQ3Y=	wbPZqMw=	AC3F4536-BB2E-41C5-B70D-4548E460C1BD	2001-12-05 00:00:00.000
4	4	SjLxpiaHSlz+6AG+H+4QpB/IPRzras/+9q/5Wr7f8=	PwSunQU=	B3FA4C24-2E96-477C-A923-0CB0F6FA5C80	2001-12-29 00:00:00.000
5	5	8FYdAiY6gWuBsgjCFdg0UibtsqOcWHf9TyaHIP7+paA=	qYhZRII=	C4D13BCF-0209-44C7-AC67-6F817FDD7F16	2002-01-30 00:00:00.000
6	6	u5kbN5n84NRE1h/a+kdRrXucjgmF6wZC4g82jHM=	a9GiLUA=	BBE788B9-8D6D-4799-87A7-7B85B6BD67DC	2002-02-17 00:00:00.000
7	7	zSqrIn8T8eq3nYHC4Lx4vMuxZaxkDylVwWnP2ZT6QA=	13mu8BA=	AFD3A20A-787B-4069-92DB-AEE666C02847	2003-03-05 00:00:00.000
8	8	s+FUWADIZzXBKpobxe40wL2uImjLogJNYXXHvc1X/k=	FCpzTU=	4FE51B60-130E-4209-9E67-28DC4D91446C	2003-01-23 00:00:00.000
9	9	fCvCTy3RwzA2LNhhHhYUbT7erkb9Au5wyM2q7ReHroV0=	FTcZMvQ=	99F4F320-F05D-4FA1-BB0D-81C9425422F3	2003-02-10 00:00:00.000
10	10	/8biMxuAtETGeluloSrMQHBraZiZeUz50J1Rn6M=	K7dMpTY=	82F25F0C-5D75-4246-958A-B6DF67559D09	2003-05-28 00:00:00.000
11	11	iaZ6ky76dbOG+0Y069v4bm78UhfGXSeYbpx4Vgd15o=	wTGoQ8=	3AF51C6F-B835-4483-94B3-8F6FD30EFA2	2004-12-29 00:00:00.000
12	12	I9HGCr3bwF3LYBIVaM/cOC2IHfg7ns5t2krejnWZ9Ko=	S0g9tDo=	FC239924-2C45-4BAD-BBE7-26C74CAE7526	2002-01-04 00:00:00.000
13	13	3ZuojojoqvBKntr+HeqWoiZNCap6N1abPoymj+O+4=	S7XWexC=	80862E73-CD76-4E2E-AA80-26BD8476F736	2005-01-16 00:00:00.000
14	14	mKAlVVzgDo12qro7ZMDm3oEmCkdlWc7+dYDFC58Vv4=	BpZw68c=	28B6CB19-3E58-4DA6-BB63-5EB981D22D4A	2005-01-23 00:00:00.000

Рис. 55. Выполнение запроса к ранее удаленной таблице



Так как запрос выполнен, можно сделать вывод, что восстановление произведено успешно.

## 7. Разностное резервное копирование.

Разностное резервное копирование выполняется с помощью добавления в Transact-SQL код опции DIFFERENTIAL, которая является признаком разностного резервного копирования.

Листинг 11

### Разностное резервное копирование

```
BACKUP DATABASE [AdventureWorks2012]
TO DISK = 'D:\AW\AdventureWorks2012.bak'
WITH INIT,DIFFERENTIAL;
```

Разностное копирование можно выполнить при помощи среды Managment Studio. Для этого в *Обозревателе объектов* нажать правой кнопкой мыши на *Базы данных* → *Задачи* → *Создать резервную копию* и в окне *Тип резервной копии* выбрать «Разностная» (рис. 56).

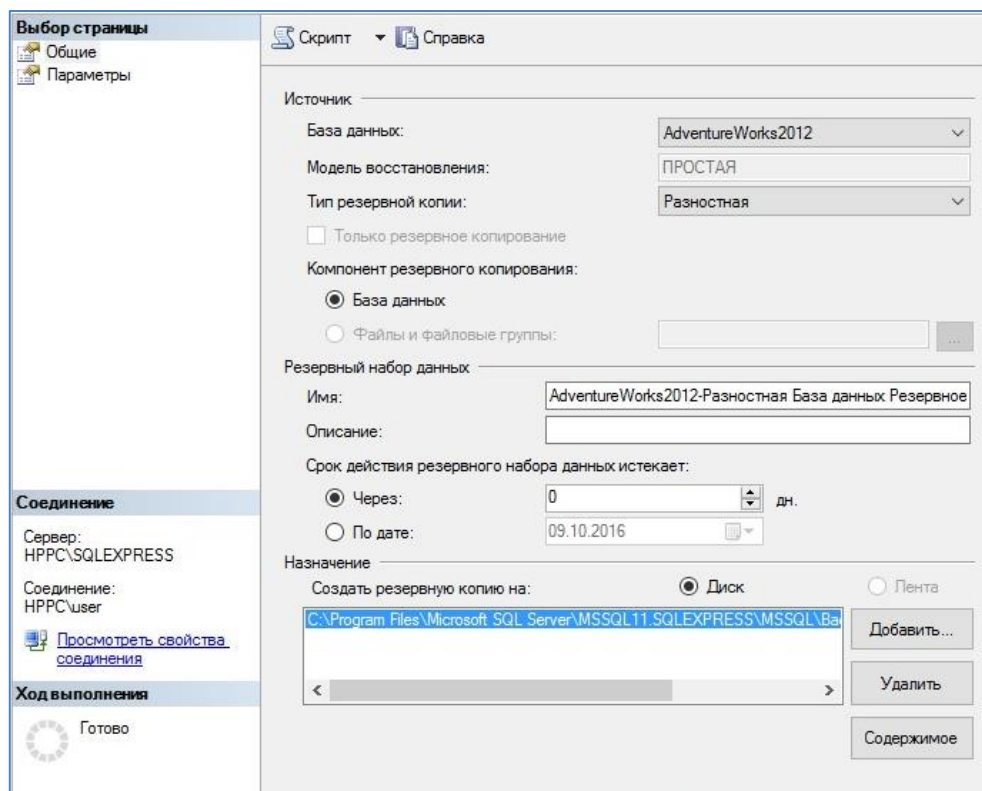


Рис. 56. Разностное резервное копирование с использованием Management Studio

8. Теперь внесём изменение в базу данных путем повторного удаления таблицы Person.Password (п. 4). Далее произведём восстановление базы данных аналогичными способами, описанными в п. 5. После этого для проверки успешного восстановления можно выполнить запрос к ранее удаленной таблице.

	BusinessEntityID	PasswordHash	PasswordSalt	rowguid	ModifiedDate
1	1	pbFwXWE99vobT6g+vpWFy93NtUU/orrIWaf01hccfM=	bE3XWw=	329EAC8E-C883-4F48-88B6-17AA4627EFFF	2003-02-08 00:00:00.000
2	2	bawRVNzQYQ05qF05Gz6VUlnviZmrqBReTTAGAudm0=	EjJaC3U=	A4C82398-7466-4FE6-B9EE-CEC34D116F68	2002-02-24 00:00:00.000
3	3	8BUxZFDQ01yHCWOYzYmqN11hTUn3CJMpdz/UCQ3iY=	wbPZqMw=	AC3F4536-BB2E-41C5-B70D-454BE460C1BD	2001-12-05 00:00:00.000
4	4	SjLxpiaHSlz+6AG+H+4QpB/PRzras/+9q/5W7r7f8=	PwSunQU=	B3FA4C24-2E96-477C-A923-0CB0F6FA5C80	2001-12-29 00:00:00.000
5	5	8FYd4Y6gWuBsgCFdg0UlbtsqCqWH9TYaHIP7+paA=	qYhZrIM=	C4D13BCF-0209-44C7-AC67-6F817FDD7F16	2002-01-30 00:00:00.000
6	6	uSkbN5n84NRE1h/a+kdRrXucjgmF6wZC4g82yJHM=	a9GILUA=	B8E788B9-8D6D-4799-87A7-7B85B6BD67DC	2002-02-17 00:00:00.000
7	7	zSqrth8T8eq3nYHC4Lx4vMuxZaxkDylVwWnP2ZT6QA=	13mu8BA=	AFD3A20A-787B-4069-92DB-AEE666C02847	2003-03-05 00:00:00.000
8	8	s+FUWADIZzXBKpbcxe4OwL2uJmjLogJNYXXHvc1X/k=	FCpzTU=	4FE51B60-130E-4209-9E67-28DC4D91446C	2003-01-23 00:00:00.000
9	9	fCvCTy3RwzA2LNhnhYUbt7erkb9Au5wyM2q7ReHroV0=	FTcZMvQ=	99F4F320-F05D-4FA1-BB0D-81C9425422F3	2003-02-10 00:00:00.000
10	10	/8biMoxuA4ETGeluloSrMQHBrZiZ+eU2z50U1Rhn6M=	K7dMpTY=	82F25F0C-5D75-4246-958A-B6DF67559D09	2003-05-28 00:00:00.000
11	11	iaZ9ky76dbOG+DY069v4bm78UrhfGXSeYbpx4Vgd15o=	wTGciQ8=	3AF51C6F-B835-4483-94B3-8F6FD30EFD42	2004-12-29 00:00:00.000
12	12	I9HGC3jwF3LYBIVsM/cOC2IHfg7ns9t2xejnWZ9Ko=	S0g9Do=	FC239924-2C45-48AD-BBE7-26C74CAE7526	2002-01-04 00:00:00.000
13	13	3ZuoogjvBkmtr+HeqWoiZNCzp6N1abPoympj+O+4=	S7XWeXc=	80862E73-CD76-4E2E-AA80-26BD8476F736	2005-01-16 00:00:00.000
14	14	mKAIVzsgDo12gro7ZDmr3oEmCcklWc7+dYDFIC58Vv4=	BpZw68c=	28B6CB19-3E5B-4DA6-BB63-5EB981D22D4A	2005-01-23 00:00:00.000
15	15	m5ESf7Vndqqa/s8QX/r3UK95GTrMPM4YhGwQzS9c=	dhGWm88=	AE7E33BE-25CD-419C-9965-A3080864D3F8	2005-02-11 00:00:00.000
16	16	oaeJoTn5hbyNfemp2qzlpGTP5uNle8NRPKi9Ur3ZnI8=	CTdtN+Q=	AE8D247F-C6D3-4842-B34C-46061CE85303	2002-01-13 00:00:00.000
17	17	70dV1zJ/Q0TtaAgSo1KHZAFAUhNDXqzL3gXmpl5BE=	nqdPuls=	FF714E7E-B072-4F44-A63F-27380090A8E8	2001-02-19 00:00:00.000
18	18	+Ssx1+1UOOR+oosTOz1L7fA79CUdo05d5uv+scXE=	5T5pzE=	0106C76D-A026-48EA-A6F3-56EE3E837182	2005-03-03 00:00:00.000
19	19	+NeFAMBZDuBhJmJVF5RQmWChbKw506H7LR8AboLXc=	9S8bKmU=	2A41CB38-D36B-4A71-934D-44CBDD67DA83	2005-03-10 00:00:00.000

Рис. 57. Запрос на выборку из таблицы

Видно, что восстановление прошло успешно.

## 9. Резервное копирование журнала транзакций.

Резервное копирование журнала транзакций выполняется при помощи команды BACKUP LOG и опции NO\_TRUNCATE, которая позволяет восстановить данные вплоть до точки сбоя в базе данных.

Листинг 12

### Резервное копирование журнала транзакций

```
BACKUP LOG [AdventureWorks2012]
TO DISK = 'D:\AW\AdventureWorks2012_log.bak'
WITH NO_TRUNCATE;
```



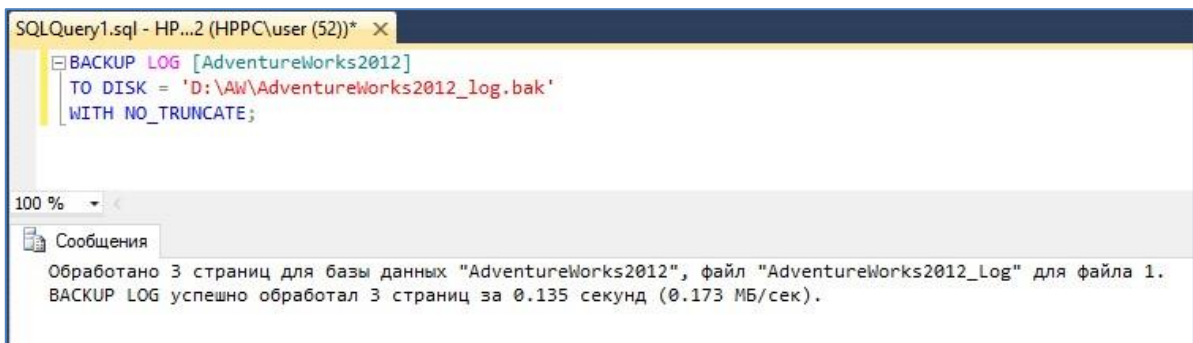


Рис. 58. Резервное копирование журнала транзакций

## 10. Восстановление базы данных при отказе системы.

Имитируется отказ системы. Для этого открывается файл базы данных через текстовый редактор и удаляются произвольные строки.

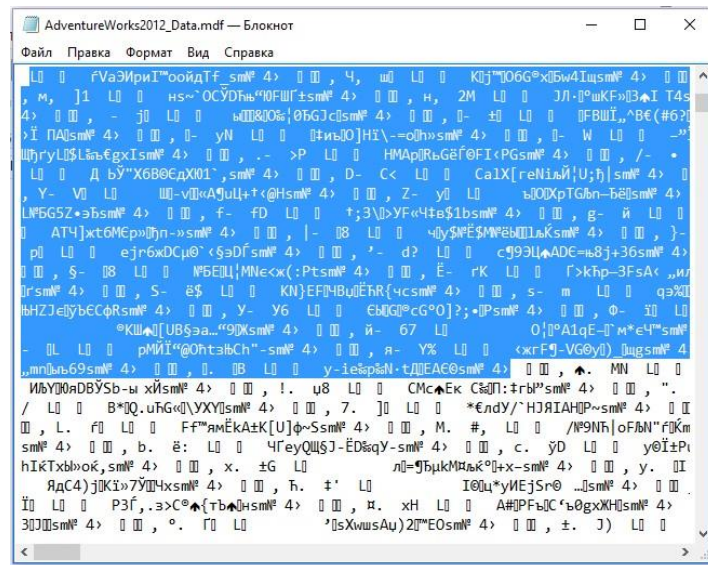


Рис. 59. Удаление части базы данных

11. При попытке открытия повреждённой базы данных появляется ошибка, показанная на рис. 60.

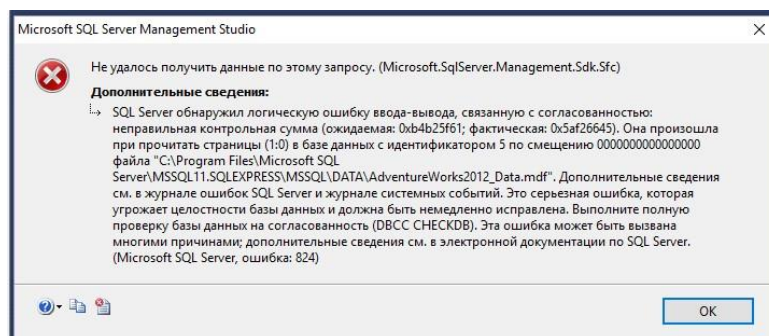


Рис. 60. Ошибка при попытке доступа к базе данных

12. Рассмотрим возможность восстановления базы данных двумя способами.

Первый способ: восстановление при помощи Transact-SQL кода.

Листинг 13

### Резервное копирование журнала транзакций

```
RESTORE DATABASE [AdventureWorks2012]
FROM DISK = 'D:\AW\AdventureWorks2012.bak'
WITH REPLACE;
```

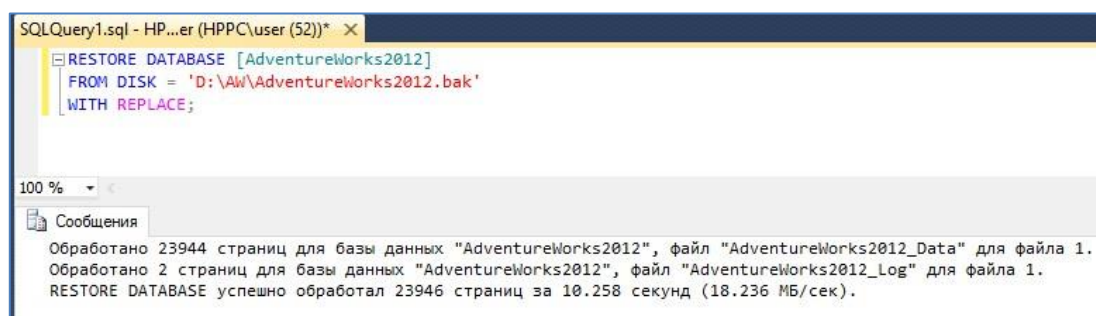


Рис. 61. Восстановление базы данных T-SQL-кодом

Второй способ: восстановление базы данных с помощью среды Management Studio.

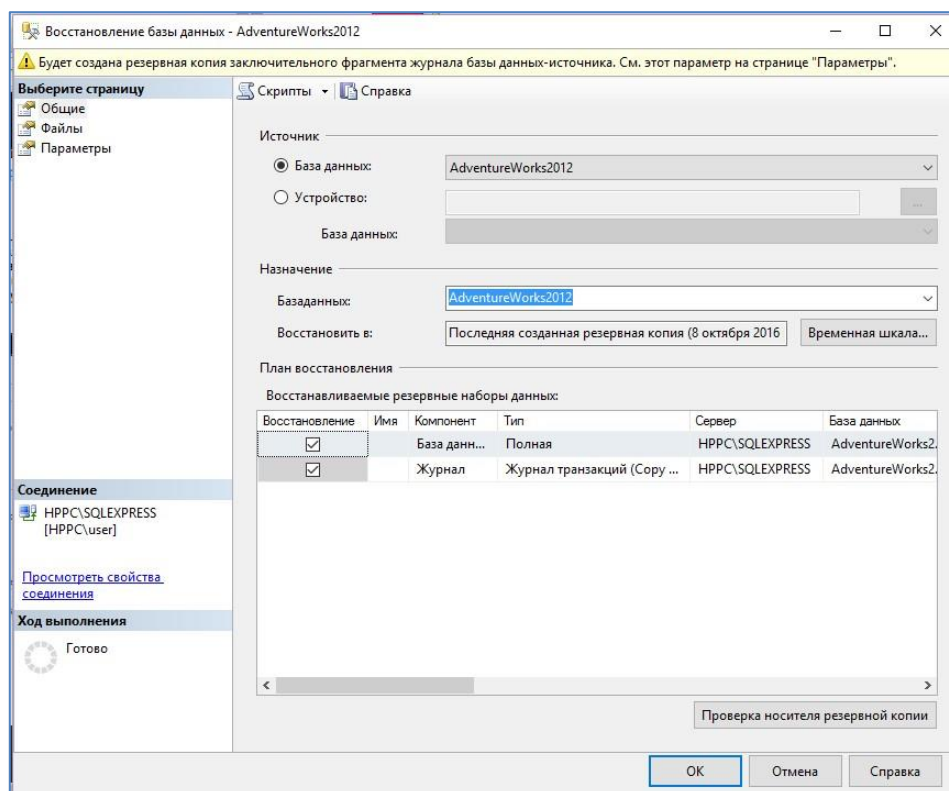


Рис. 62. Восстановление базы в среде Management Studio

При необходимости во вкладке Параметры можно указать необходимые параметры восстановления. При нажатии на *OK* выполнится восстановление базы данных.

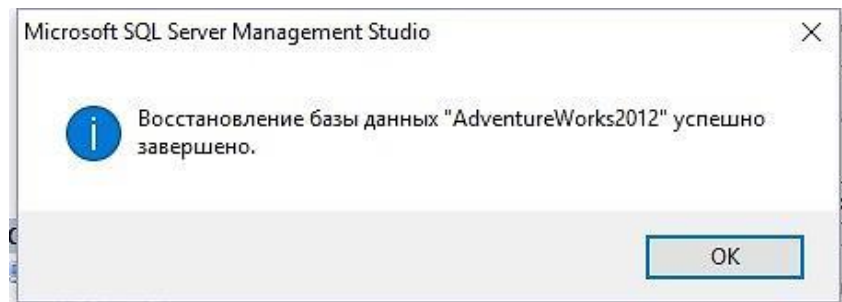
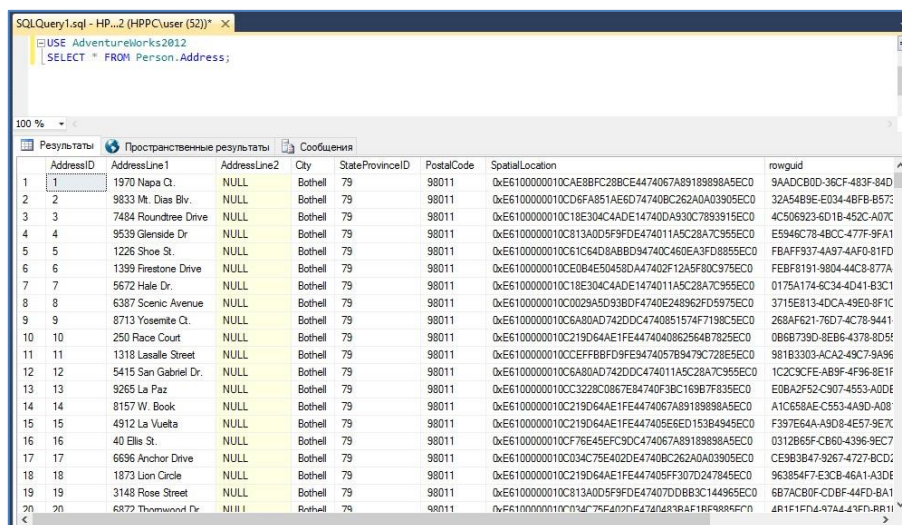


Рис. 63. Успешное восстановление базы данных

13. Чтобы проверить, доступна ли база данных, можно выполнить запрос к произвольной таблице, продемонстрированный на рис. 64.



AddressID	AddressLine1	AddressLine2	City	StateProvinceID	PostalCode	SpatialLocation	rowguid
1	1970 Napa Ct.	NULL	Bothell	79	98011	0xE6100000010CAE8BFC28BCE4474067A89189898A5EC0	9AADC8B0D-36CF-483F-84D
2	9833 Mt. Dias Blv.	NULL	Bothell	79	98011	0xE6100000010CD6FA851AE6D74740BC262A0A03905EC0	32A54B9E-E034-48FB-B57C
3	7484 Roundtree Drive	NULL	Bothell	79	98011	0xE6100000010C18E304C4ADE14740DA930C7893915EC0	4C506923-6D1B-452C-A07C
4	9539 Glenside Dr	NULL	Bothell	79	98011	0xE6100000010C813A0D5F9FDE474011A5C28A7C955EC0	E5946C78-4BCC-477F-9FA1
5	1226 Shoe St.	NULL	Bothell	79	98011	0xE6100000010C61C6AD8ABD94740C460EA3FD8855EC0	FB4FF937-4A97-4AF0-81FD
6	1399 Firestone Drive	NULL	Bothell	79	98011	0xE6100000010CE0B4E50458DA47402F12A5F80C975EC0	FE8F8191-9804-44C8-877A
7	5672 Hale Dr.	NULL	Bothell	79	98011	0xE6100000010C18E304C4ADE1474011A5C28A7C955EC0	0175A174-6C34-4D41-B3C1
8	6387 Scenic Avenue	NULL	Bothell	79	98011	0xE6100000010CD028A5D93BDF4740E249962FD5975EC0	3715E813-4DCA-49E0-8F1C
9	8713 Yosemite Ct.	NULL	Bothell	79	98011	0xE6100000010C6A80AD742DDC4740851574F7198C5EC0	268AF621-76D7-4C78-9441
10	250 Race Court	NULL	Bothell	79	98011	0xE6100000010C219D64AE1FE4474040862564B7825EC0	0B6B739D-8EB6-4378-8D54
11	1318 Lassalle Street	NULL	Bothell	79	98011	0xE6100000010CCEFFB8FD9FE947405789479C728E5EC0	981B3303-ACA2-49C7-9A96
12	5415 San Gabriel Dr.	NULL	Bothell	79	98011	0xE6100000010C6A80AD742DDC474011A5C28A7C955EC0	1C2C9C9E-AB9F-4F96-8E1F
13	9265 La Paz	NULL	Bothell	79	98011	0xE6100000010CC3228C08E7E84740F3BC169B7F835EC0	E0BA2F52-C907-4553-A0DE
14	8157 W. Book	NULL	Bothell	79	98011	0xE6100000010C219D64AE1FE4474067A89189898A5EC0	A1C658AE-C553-4A9D-A08
15	4912 La Vuelta	NULL	Bothell	79	98011	0xE6100000010C219D64AE1FE447405E6ED153B4945EC0	F397E64A-A9D8-4E57-9E7C
16	40 Ellis St.	NULL	Bothell	79	98011	0xE6100000010CF76E45EFC9DC474067A89189898A5EC0	0312B65F-CB6D-4396-9EC7
17	6696 Anchor Drive	NULL	Bothell	79	98011	0xE6100000010C034C75E402DE4740BC262A0A03905EC0	CE983B47-9267-4727-8CD4
18	1873 Lion Circle	NULL	Bothell	79	98011	0xE6100000010C219D64AE1FE447405FF307D247845EC0	963854F7-E3CB-46A1-A3DE
19	3148 Rose Street	NULL	Bothell	79	98011	0xE6100000010C813A0D5F9FDE47407DDB83C144965EC0	6B7ACB0F-CDBF-44FD-BA1
20	6872 Thomwood Dr.	NULL	Bothell	79	98011	0xE6100000010C18E304C4ADE1474011A5C28A7C955EC0	4B1F1FD4-9748-43FD-RR11

Рис. 64. Вывод данных по запросу

Можно сделать вывод, что база данных корректно восстановлена.

## Вывод

В данной лабораторной работе было выполнено полное и разностное резервное копирование, проделана процедура восстановления базы данных двумя способами: при помощи SQL-кода и при помощи среды Management Studio, выполнено резервное копирование журнала транзакций и проверена стабильная работа базы данных при помощи выполнения запросов после каждой операции.

## **Заключение**

Современные системы управления реляционными базами данных представляют собой сложную совокупность программных средств, позволяющих эффективно управлять хранением и извлечением данных. Для грамотного и продуманного администрирования СУБД необходимо разбираться в строении компонентов этой системы и знать как можно более полный перечень функциональных возможностей, предоставляемых используемой системой. Именно изучению этих вопросов посвящено данное учебное пособие.

В пособии приводятся теоретические понятия и определения, имеющиеся в авторитетной научной литературе, дается историческая справка по файловым системам, позволяющая проследить путь, который привел к появлению современных СУБД, разбирается архитектура системы SQL Server, рассматриваются устройство индексов, оптимизатора запросов, системы безопасности, вопросы резервного копирования и восстановления.

Практическую значимость представляют задания к лабораторным работам и пошаговые примеры их выполнения. Предполагается, что читатель имеет некоторый опыт работы с базами данных, владеет знаниями о реляционной модели, навыками написания SQL-запросов. Для каждой лабораторной работы приводится теоретический материал по соответствующей теме, позволяющий контролировать результат усвоения материала при его защите.

## Библиографический список

1. Бен-Ган, И. Microsoft SQL Server 2012. Высокопроизводительный код T-SQL. Оконные функции [Текст] / И. Бен-Ган. – Санкт-Петербург: БХВ-Петербург, 2013. – 256 с.
2. Бен-Ган, И. Microsoft SQL Server 2012. Основы T-SQL [Текст] / И. Бен-Ган. – Москва: Эксмо, 2015. – 400 с.
3. Бондарь, А. Microsoft SQL Server 2014 [Текст] / А. Бондарь. – Санкт-Петербург: БХВ-Петербург, 2015. – 589 с.
4. Вьейра, Р. Программирование баз данных Microsoft SQL Server 2008. Базовый курс [Текст] / Р. Вьейра. – Москва: Вильямс, 2010. – 816 с.
5. Грофф, Д.Р. SQL. Полное руководство [Текст] / Д. Р. Грофф, П. Н. Вайнберг, Э. Дж. Оппель. – Москва: Вильямс, 2014. – 960 с.
6. Коннолли, Т. Базы данных. Проектирование, реализация и сопровождение. Теория и практика [Текст] / Т. Коннолли, К. Бегг. – Москва: Вильямс, 2017. – 1440 с.
7. Алгоритмы. Построение и анализ [Текст] / Т. Кормен [и др.]. – Москва: Вильямс, 2016. – 1328 с.
8. Петкович, Д. Microsoft SQL Server 2012. Руководство для начинающих [Текст] / Д. Петкович. – Санкт-Петербург: БХВ-Петербург, 2013. – 816 с.
9. Седжвик, Р. Алгоритмы на C++ [Текст] / Р. Седжвик. – Москва: Вильямс, 2017. – 1056 с.
10. Станек, У.Р. Microsoft SQL Server 2012. Справочник администратора [Текст] / У.Р. Станек. – Санкт-Петербург: БХВ-Петербург, 2013. – 674 с.
11. Форта, Б. Освой самостоятельно SQL за 10 минут [Текст] / Б. Форта. – Москва: Вильямс, 2016. – 288 с.
12. MySQL. Оптимизация производительности [Текст] / Б. Шварц [и др.]. – Санкт-Петербург: Символ-Плюс, 2010. – 832 с.

*Учебное издание*

**Мирошников Артём Игоревич**

# **АРХИТЕКТУРА СИСТЕМ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ**

**Учебное пособие**

Редактор Черникова Е.Н.

Подписано в печать . Формат 60 × 84 1/16. Бумага офсетная.

Ризография. Объем 5,86 п.л. Тираж 100 экз. Заказ №

Издательство Липецкого государственного технического университета.

Полиграфическое подразделение Издательства ЛГТУ.

398055 Липецк, ул. Московская. 30.