**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КОМИ**

**Государственное профессиональное образовательное учреждение**

**«Воркутинский арктический горно-политехнический колледж»**

**КУРСОВОЙ ПРОЕКТ**

По дисциплине МДК.07.01 Управление и автоматизация баз данных

**Разработка базы данных**

**“Планирование и обслуживание станков в цеху”**

Выполнил студент гр. ИСП-20 /\_\_\_\_\_\_\_/Оскретков Данил Васильевич/

(подпись) (Ф.И.О.)

**ОЦЕНКА:\_\_\_\_\_\_\_\_\_\_\_\_\_**

Дата:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ПРОВЕРИЛ**

Научный руководитель\_\_\_\_\_\_\_\_\_\_\_\_\_/ Егоров Данил Павлович/

(подпись) (Ф.И.О.)

Воркута

2023

**СОДЕРЖАНИЕ**

[**ВВЕДЕНИЕ** 3](#_Toc153185881)

[**ГЛАВА 1. ВЫБОР ИНСТРУМЕНТАРИЯ** 4](#_Toc153185882)

[**1.1 СУБД SQL server** 4](#_Toc153185883)

[**1.2 Microsoft SQL Server Management Studio** 6](#_Toc153185884)

[**1.3 Entity Framework** 7](#_Toc153185885)

[**ГЛАВА 2. ПРОЕТИРОВАНИЕ БАЗЫ ДАННЫХ** 10](#_Toc153185886)

[**2.1 Разработка диаграммы ERD** 10](#_Toc153185887)

[**2.2 Разработка базы данных** 11](#_Toc153185888)

[**2.3 Словарь данных** 15](#_Toc153185889)

[**ГЛАВА 3. РАЗРАБОТКА БАЗЫ ДАННЫХ** 18](#_Toc153185890)

[**3.1 Введение в безопасность SQL Server** 18](#_Toc153185891)

[**3.2 Управление безопасностью уровня сервера** 19](#_Toc153185892)

[**3.3 Управление участниками уровня базы данных** 28](#_Toc153185893)

[**3.4 Управление разрешениями уровня базы данных** 30](#_Toc153185894)

[**3.5 Шифрование данных баз данных** 30](#_Toc153185895)

[**зАКЛЮЧЕНИЕ** 33](#_Toc153185896)

[**СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ** 34](#_Toc153185897)

**ВВЕДЕНИЕ**

Планирования технического обслуживания станков в цеху деятельности неразделимо с внедрением передовых технологий и процессов с целью обеспечения наиболее качественного обслуживания станков. Внедрение автоматизированных информационных систем – это не только тенденция времени, но и хорошая база для увеличения производительности и качества труда.

С помощью запросов реализованы функции поиска информации по заданной тематике, группировки информации для различных отчетов, редактирования и удаления записей в таблицах. Поиск станков, результатов тестирования, дате тестирования или соответствию некоторому разделу осуществляется с помощью запросов с параметром. Их удобно использовать в качестве источников записей для форм, отчетов и страниц доступа к данным.

**Объект** – база данных информационной системы «Планирования технического обслуживания станков в цеху».

**Предмет** - автоматизация бизнес-процесса «Планирования технического обслуживания станков в цеху»

**Цель:** разработать базу данных для информационной системы «Планирования технического обслуживания станков в цеху».

**Задачи:**

* выбор инструментария;
* спроектировать базу данных;
* сделать разграничение прав пользователей для базы данных;
* сделать шифрование данных;
* разработать информационную систему.

**ГЛАВА 1. ВЫБОР ИНСТРУМЕНТАРИЯ**

**1.1 СУБД SQL server**

СУБД SQL server используются для создания, размещения, хранения и управления реляционными (табличными) базами данных на специальных серверах или в облаке. Они работают через настольные приложения и web-сайты. К основным преимуществам их функционирования относятся:

* высокоскоростной доступ к данным, обеспечиваемый надежной клиент-серверной архитектурой СУБД;
* простота работы и администрирования, обусловленные понятной структурой языка программирования SQL;
* безопасность хранения информации в БД - благодаря возможности шифрования данных и резервного копирования.

Специфика работы сервера базы данных SQL server заключается в транзакционной обработке данных. Это означает, что по каждому запросу от СУБД обрабатывается и сохраняется небольшое количество информации.

Применение SQL server позволяет автоматизировать решение различных бизнес-задач, поддерживать проведение аналитики данных в режиме онлайн, отслеживать направление ресурсов СУБД, управлять транзакциями (операциями по обработке данных).

Система управления базами данных (СУБД) - это общий набор различных программных компонентов баз данных и собственно баз данных, содержащий следующие составляющие:

* прикладные программы баз данных;
* клиентские компоненты;
* серверы баз данных;
* собственно базы данных.

Прикладная программа баз данных представляет собой программное обеспечение специального назначения, разработанное и реализованное пользователями или сторонними компаниями-разработчиками ПО. В противоположность, клиентские компоненты - это программное обеспечение баз данных общего назначения, разработанное и реализованное компанией-разработчиком базы данных. С помощью клиентских компонентов пользователи могут получить доступ к данным, хранящимся на локальном или удаленном компьютере.

Сервер баз данных выполняет задачу управления данными, хранящимися в базе данных. Клиенты взаимодействуют с сервером баз данных, отправляя ему запросы. Сервер обрабатывает каждый полученный запрос и отправляет результаты соответствующему клиенту.

В общих чертах, базу данных можно рассматривать с двух точек зрения - пользователя и системы базы данных. Пользователи видят базу данных как набор логически связанных данных, а для системы баз данных это просто последовательность байтов, которые обычно хранятся на диске. Хотя это два полностью разных взгляда, между ними есть что-то общее: система баз данных должна предоставлять не только интерфейс, позволяющий пользователям создавать базы данных и извлекать или модифицировать данные, но также системные компоненты для управления хранимыми данными. Поэтому система баз данных должна предоставлять следующие возможности:

* разнообразные пользовательские интерфейсы;
* физическую независимость данных;
* логическую независимость данных;
* оптимизацию запросов;
* целостность данных;
* управление параллелизмом;
* резервное копирование и восстановление;
* безопасность баз данных

**1.2 Microsoft SQL Server Management Studio**

Среда SQL Server Management Studio предназначена для доступа к службам Службы Analysis Services, Integration Services и Reporting Services, а также для их настройки, администрирования и управления ими. Хотя все три технологии бизнес-аналитики полагаются на среду SQL Server Management Studio, административные задачи, связанные с каждой из этих технологий, несколько отличаются.

Управление решениями Analysis Services с помощью SQL Server Management Studio

Среда SQL Server Management Studio позволяет управлять объектами Службы Analysis Services, например выполнять их резервное копирование и обработку.

Среда Management Studio позволяет создавать проекты скриптов служб Службы Analysis Services, в которых выполняются разработка и сохранение скриптов с использованием многомерных выражений (MDX), расширений интеллектуального анализа данных (DMX) и XML для аналитики (XMLA). Проекты скриптов служб Службы Analysis Services используются для выполнения задач управления или повторного создания баз данных, кубов и других объектов в экземплярах служб Службы Analysis Services. Например, можно разработать скрипт XMLA в проекте скрипта служб Службы Analysis Services, который создает объекты непосредственно в существующем экземпляре служб Службы Analysis Services . Проекты скриптов служб Службы Analysis Services могут быть сохранены в составе решения и интегрироваться с контролем исходного кода.

Управление решениями Integration Services с помощью SQL Server Management Studio

Среда SQL Server Management Studio позволяет использовать службу Integration Services для управления пакетами и наблюдения за выполняющимися пакетами. В среде Среда Management Studio можно организовать пакеты в папки, выполнять, импортировать и экспортировать пакеты, переносить пакеты служб DTS и обновлять пакеты служб Integration Services.

Управление проектами Reporting Services с помощью SQL Server Management Studio

Среда SQL Server Management Studio позволяет включать компоненты служб Reporting Services, администрировать серверы и базы данных, управлять ролями и заданиями.

Она реализует функции управления общими расписаниями (в папке «Общие расписания») и базами данных сервера отчетов (ReportServer, ReportServerTempdb). Можно также создать роль RSExecRole в системной базе данных Master, когда база данных сервера отчетов перемещается в новое или другое ядро СУБД SQL Server.

**1.3 Entity Framework**

Entity Framework — это набор технологий в ADO.NET, которые поддерживают разработку программных приложений, ориентированных на данные. Архитекторам и разработчикам приложений, ориентированных на обработку данных, приходится учитывать необходимость достижения двух совершенно различных целей. Они должны моделировать сущности, связи и логику решаемых бизнес-задач, а также работать с ядрами СУБД, используемыми для сохранения и получения данных. Данные могут распределяться по нескольким системам хранения данных, в каждой из которых применяются свои протоколы, но даже в приложениях, работающих с одной системой хранения данных, необходимо поддерживать баланс между требованиями системы хранения данных и требованиями написания эффективного и удобного для обслуживания кода приложения.

Платформа Entity Framework позволяет работать с данными в форме специфических для домена объектов и свойств (например, с клиентами и их адресами) без необходимости учитывать формат базовых таблиц и столбцов базы данных, где хранятся эти данные. Entity Framework дает разработчикам возможность работать с данными на более высоком уровне абстракции, создавать и сопровождать приложения, ориентированные на работу с данными, одновременно с этим сокращая объем кода по сравнению с традиционными приложениями.

Многолетним и общим подходом к разработке является подход, при котором построение приложения или службы представляет собой его разделение на три части: модель домена, логическую модель и физическую модель. Модель домена определяет сущности и связи в моделируемой системе. Логическая модель для реляционной базы данных обеспечивает нормализацию сущностей и связей в целях создания таблиц с ограничениями внешнего ключа. В физической модели учитываются возможности конкретной системы обработки данных путем определения зависящих от ядра базы данных подробных сведений о хранении данных, которые касаются секционирования и индексирование.

Физическая модель совершенствуется администраторами базы данных в целях повышения производительности, но программисты, которые разрабатывают код приложения, в основном вынуждены ограничиваться работой с логической моделью, подготавливая SQL-запросы и вызывая хранимые процедуры. Модели домена в основном используются как инструмент для представления и обмена мнениями о требованиях к приложению, поэтому чаще всего служат в качестве практически не изменяющихся схем, которые рассматриваются и обсуждаются на ранних стадиях проекта, после чего выходят из сферы внимания. Во многих коллективах разработчиков принято пропускать этап создания концептуальной модели и начинать с определения таблиц, столбцов и ключей в реляционной базе данных.

Entity Framework дает жизнь модели, позволяя разработчикам запрашивать сущности и связи в модели предметной области (называемой концептуальной моделью в Entity Framework), в то же время полагаться на Entity Framework преобразования этих операций в команды, относящиеся к источнику данных. Это позволяет отказаться от применения в приложениях жестко заданных зависимостей от конкретного источника данных.

При работе в режиме Code First концептуальная модель сопоставлена с режимом хранения в коде. Entity Framework может вычислять концептуальную модель на основе типов объектов и дополнительных конфигураций, которые вы определяете. Метаданные сопоставления формируются во время выполнения на основе сочетания определений типов домена и дополнительной информации о конфигурации, которая указана в коде. Entity Framework создает базу данных по мере необходимости на основе метаданных.

**ГЛАВА 2. ПРОЕТИРОВАНИЕ БАЗЫ ДАННЫХ**

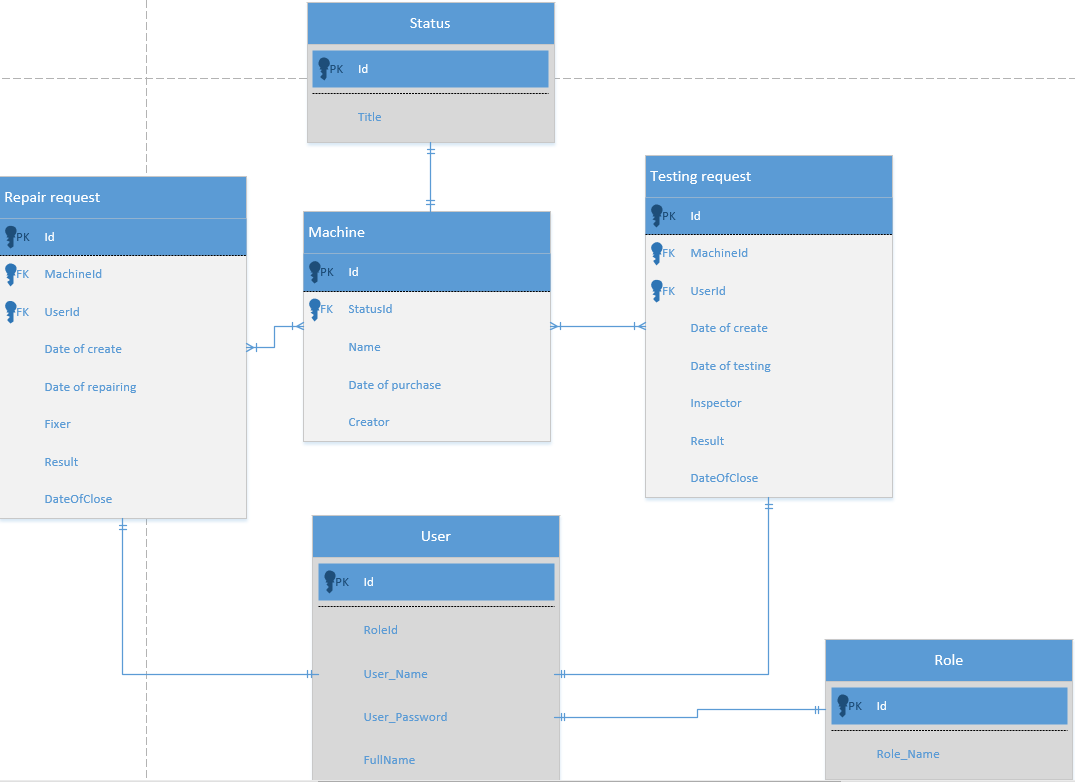
**2.1 Разработка диаграммы ERD**

Схема «сущность-связь» (также ERD или ER-диаграмма) — это разновидность блок-схемы, где показано, как разные «сущности» (люди, объекты, концепции и так далее) связаны между собой внутри системы. ER-диаграммы чаще всего применяются для проектирования и отладки реляционных баз данных в сфере образования, исследования и разработки программного обеспечения и информационных систем для бизнеса.

Базовыми понятиями ER-модели данных (ER — Entity— Relati-onship) являются сущность, атрибут и связь.

Каждая сущность должна иметь:

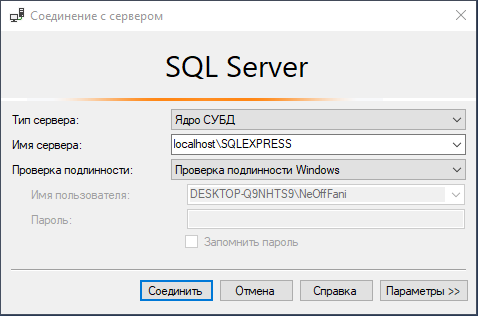
* уникальное имя;
* один или несколько атрибутов, которые либо принадлежат сущности, либо наследуются через связь;
* один или несколько атрибутов, которые однозначно идентифицируют каждый экземпляр сущности.

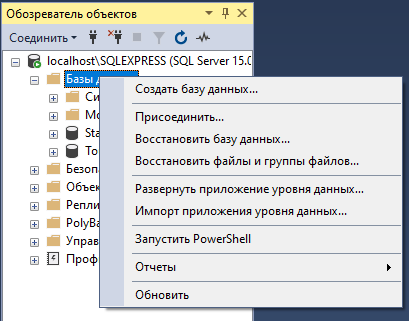
Основными сущностями являются: станок, заявка на тестирование, заявка на починку. Каждая сущность обделена атрибутами. У сущности «Станок» имеется: дата покупки, наименование, создатель, статус работы. У сущностей «Заявка на тестирование» и «Заявка на ремонт» атрибуты: номер станка, дата тестирования, дата создания заявки, создатель заявки, результат, дата закрытия заявки, инспектор у сущности “Заявка на тестирование” и ремонтник у сущности “Заявки на ремонт”. У сущности “Статус” атрибут: наименование. Так же имеется 2 сущности для хранения информации о пользователях информационной системы и хранения ролей для пользователей. Сущность “Пользователь” имеет такие атрибуты: наименование роли, ФИО, логин и пароль. Сущность “Роль” атрибут: наименование роли.

*Рисунок 2.1. Диаграмма ERD*

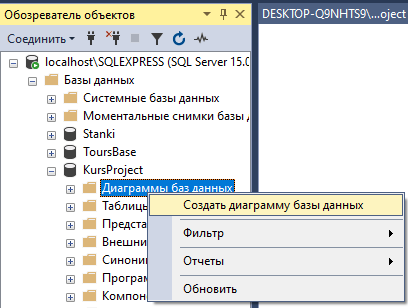
**2.2 Разработка базы данных**

Для того что бы начать создания базы данных. Нужно запустить приложение “Microsoft SQL Server Management Studio” и подключить к серверу с именем сервера “localhost\SQLEXPRESS”.

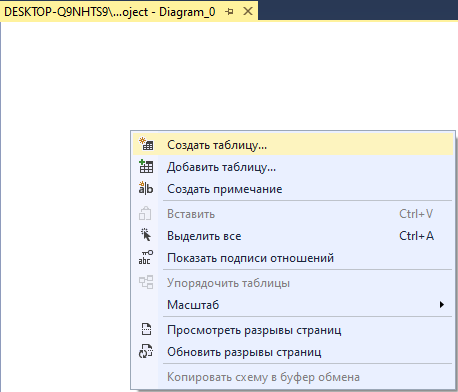
*Рисунок 2.2 Подключение к MSSMS*

Далее в левой части экрана имеется панель “Обозреватель объектов”, нам нужно нажать правой кнопкой мыши по папке с названием “Базы данных” и выбрать пункт “Создать базу данных”.

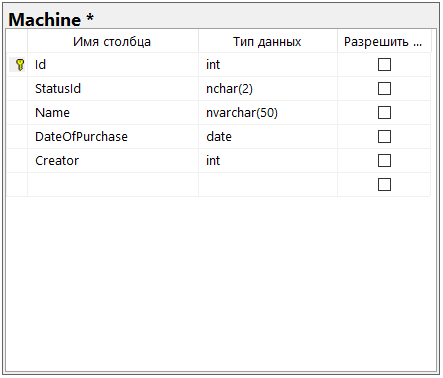
*Рисунок 2.3 Создание базы данных.*

В обозреватели объектов появляется созданная нами база данных раскрываем её нажатием “+” слева от неё, нажимаем правой кнопкой мыши по папке “Диаграммы баз данных” и выбираем “Создать диаграмму базы данных”.

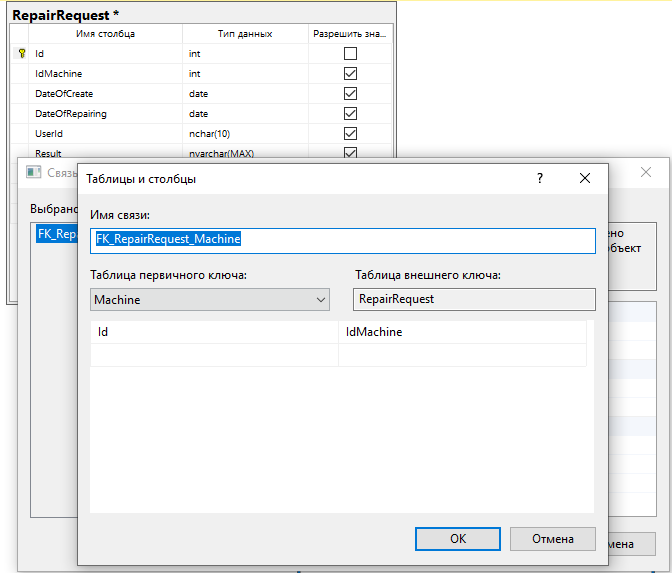
*Рисунок 2.4 Создание диаграммы базы данных*

Для того что бы создавать таблицы нашей базы данных, нужно нажать правой кнопкой мыши по пустому месту нашей диаграммы и выбрать пункт “Создать таблицу” и вводим название нашей таблицы.

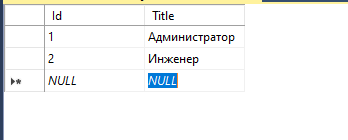
*Рисунок 2.5 Создание таблиц.*

Создаётся первая таблица, в которой можно уже писать атрибуты и настраивать ключи. Вводим первые атрибуты и выбираем типы данных.

*Рисунок 2.6 Атрибуты и типы данных.*

После создания таблиц, настраиваем связи. Зажимая ключ одной таблицы и ведём к полю другой таблицы с нужным атрибутом.

*Рисунок 2.7 Создание свзяей*

Как создали все таблицы, настроили все связи. Сохраняем диаграмму, выбираем таблицу и можно заполнять данные.

*Рисунок 2.8 Заполнение таблиц.*

**2.3 Словарь данных**

Словарь данных представляет собой подсистему банка данных, предназначенную для централизованного хранения информации о структурах данных, взаимосвязях файлов БД друг с другом, типах данных и форматах их представления, принадлежности данных пользователям, когда защиты и разграничения доступа.

Таблица 1. Словарь данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Data Dictionary | | | | |
| **UsersTable** | | | | |
| **KEY** | **FIELD NAME** | **DATA TYPE / FIELD SIZE** | **REQUIRED?** | **NOTES** |
| PRIMARY | **ID** | INT | Y | Уникальный идентификатор |
| FOREIGN | **RoleId** | INT | Y | Уникальный идентификатор роли |
|  | **Login** | varchar(50) | Y | Имя пользователя для входа в систему |
|  | **Password** | varchar(50) | Y | Пароль пользователя для входа в систему |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Role** | | | | |
| **KEY** | **FIELD NAME** | **DATA TYPE / FIELD SIZE** | **REQUIRED?** | **NOTES** |
| PRIMARY | **ID** | INT | Y | Уникальный идентификатор |
|  | **Title** | Nvarchar(50) | Y | Название роли |
| **Machine** | | | | |
| **KEY** | **FIELD NAME** | **DATA TYPE / FIELD SIZE** | **REQUIRED?** | **NOTES** |
| PRIMARY | **ID** | INT | Y | Уникальный идентификатор |
| FOREIGN | **StatusID** | INT | Y | Уникальный идентификатор статуса |
|  | **Name** | Nvarchar(50) | Y | Название станка |
|  | **Date of purhase** | DateTime | Y | Дата покупки станка |
|  | **Creator** | Nvarchar(50) | Y | Кто добавил станок |
| **Repair request** | | | | |
| **KEY** | **FIELD NAME** | **DATA TYPE / FIELD SIZE** | **REQUIRED?** | **NOTES** |
| PRIMARY | **ID** | INT | Y | Уникальный идентификатор |
| FOREIGN | **MachineID** | INT | Y | Уникальный идентификатор станка |
| FOREIGN | **UserId** | INT | Y | Создатель заявки |
|  | **Date of create** | DateTime | Y | Дата создания заявки |
|  | **Date of repairing** | DateTime | Y | Дата проведения работ |
|  | **Fixer** | Nvarchar(50) | Y | Исполнитель заявки |
|  | **Result** | Nvarchar(MAX) | N | Результат проведённых работ |
|  | **Date of close** | DateTime | N | Дата закрытия заявки |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Status** | | | | |
| **KEY** | **FIELD NAME** | **DATA TYPE / FIELD SIZE** | **REQUIRED?** | **NOTES** |
| PRIMARY | **ID** | INT | Y | Уникальный идентификатор |
|  | **Title** | Nvarchar(50) | Y | Наименование статуса |
| **Test request** | | | | |
| **KEY** | **FIELD NAME** | **DATA TYPE / FIELD SIZE** | **REQUIRED?** | **NOTES** |
| PRIMARY | **ID** | INT | Y | Уникальный идентификатор |
| FOREIGN | **MachineID** | INT | Y | Уникальный идентификатор станка |
| FOREIGN | **UserId** | INT | Y | Создатель заявки |
|  | **Date of create** | DateTime | Y | Дата создания заявки |
|  | **Date of testing** | DateTime | Y | Дата проведения работ |
|  | **Fixer** | Nvarchar(50) | Y | Исполнитель заявки |
|  | **Result** | Nvarchar(MAX) | N | Результат проведённых работ |
|  | **Date of close** | DateTime | N | Дата закрытия заявки |

**ГЛАВА 3. РАЗРАБОТКА БАЗЫ ДАННЫХ**

**3.1 Введение в безопасность SQL Server**

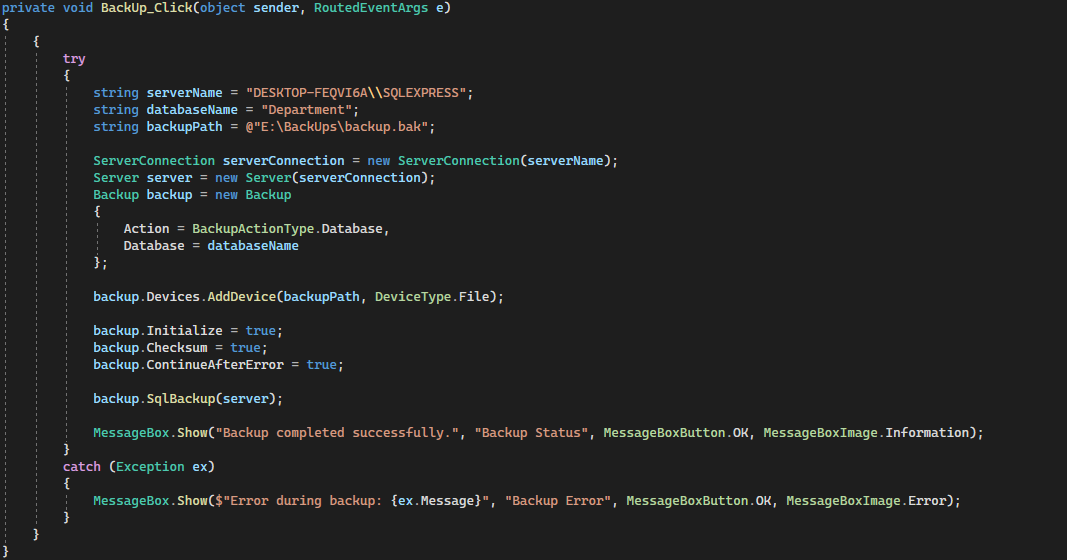
Резервные копии являются важным элементом стратегии защиты базы Резервное копирование баз данных SQL Server, выполнение тестовых процедур восстановления резервных копий в резервных копиях и хранение копий резервных копий в безопасном расположении вне сайта защищает вас от потенциально катастрофической потери данных. Резервное копирование — единственный способ защитить данные.

При правильном создании резервных копий баз данных можно будет восстановить данные после многих видов сбоев, включая следующие:

* Сбой носителя.
* ошибки пользователей (например, удаление таблицы по ошибке);
* сбои оборудования (например, поврежденный дисковый накопитель или безвозвратная потеря данных на сервере);

Кроме того, резервные копии базы данных полезны для обычных административных целей, таких как копирование базы данных с одного сервера на другой.

Влияние, оказываемое осуществлением резервного копирования на выполняемые транзакции, минимально, поэтому операции резервного копирования могут выполняться одновременно с выполнением обычных операций. Резервное копирование SQL Server можно выполнять с минимальным воздействием на рабочие нагрузки.

***Рисунок 3.1. Создание резервной копии через приложение***

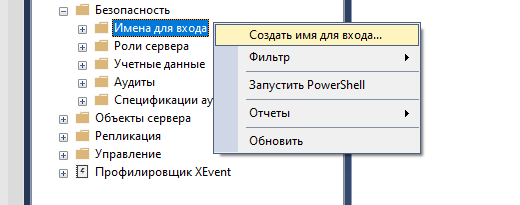
******

*Рисунок 3.2. Код для восстановления базы данных*

**3.2 Управление безопасностью уровня сервера**

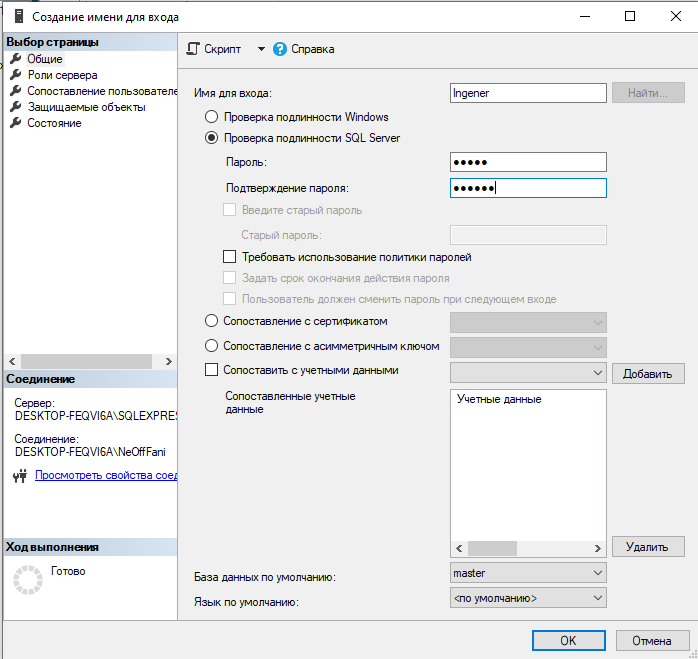
Существует два уровня доступа к экземпляру SQL Server: учетные записи пользователя сервера (или экземпляра) и пользователи базы данных. С помощью учетных записей серверы позволяют внешнему пользователю (далее термин «пользователь» применяется для любого приложения, службы, API и т. д., пытающихся подключиться к SQL Server) выполнить начальное соединение с экземпляром SQL Server. В случае безопасности на основе SQL для этого требуются имя пользователя и пароль. В случае доверенной проверки подлинности это учетная запись домена.

Способа создать эти учетные записи пользователя через графический интерфейс. Чтобы запустить диалоговое окно для создания учетных записей пользователей, подключитесь к экземпляру SQL Server в среде SQL Server Management Studio (SSMS) в обозревателе объектов, а затем разверните узел Security\Logins («Безопасность\Имена пользователя»). Щелкните правой кнопкой мыши на пункте Logins и выберите в контекстном меню пункт New Login («Создать учетную запись»).



***Рисунок 3.3. Создание учетной записи пользователя SQL Server***

Вы увидите диалоговое окно для настройки параметров учетной записи, показанное на рисунке 3.4. Изменить имя пользователя можно в том же окне.

***Рисунок 3.4 Настройка учетной записи пользователя SQL Server***

Это вкладка General («Общие») для создания (и изменения) параметров учетной записи. На вкладке General можно задать:

**Login name («Имя пользователя»).** Используется при проверке подлинности. В случае Windows, или доверенной проверки подлинности, необходимо задать имя в формате DOMAIN\LOGIN, где LOGIN — имя пользователя внутри домена, из которого пользователь выполняет проверку подлинности. Если экземпляр SQL Server расположен в другом домене, то необходимы отношения доверия между этим доменом и доменом SQL Server.

Password («Пароль»). При проверке подлинности SQL Server текстовое поле пароля включено, и вы вводите как имя пользователя, так и связанный с ним пароль.

Password Policy («Настройки политики паролей») и Expiration («Срок действия»)***.*** Флажки для политики пароля и срока действия также установлены в режиме проверки подлинности SQL Server, и применяются те политики, которые действуют в Active Directory в домене, где размещается SQL Server. Назначая имя пользователя SQL Server, вы можете разрешить пользователям менять свои пароли после регистрации. В результате администратор базы данных лишается доступа к имени учетной записи конечного пользователя.

Default Database («База данных по умолчанию»).Когда подключение к SQL Server установлено, выполняются два шага: проверка подлинности (должно существовать имя пользователя для учетных данных домена пользователя, если используется Windows или доверенная проверка подлинности либо необходимо передать комбинацию имени пользователя и пароля в экземпляр SQL Server). Это первый барьер. Второй заключается в том, что у проверенного имени пользователя имеется связанный объект пользователя в базе данных по умолчанию — базе данных, первоначально настроенной как контекст имени пользователя после проверки удостоверения. Даже если первое препятствие преодолено, при отсутствии соответствующего пользователя базы данных в базе данных по умолчанию подключение не будет установлено, и соответствующая запись будет внесена в журнал ошибок SQL. Но есть исключения: если серверная роль пользователя настолько важна, что нужно установить для него по умолчанию неявные права в каждой базе данных, то необязательно наличие соответствующего пользователя в базе данных по умолчанию. Однако я забегаю вперед, так как мы еще не рассматривали пользователей базы данных или роли сервера. Достаточно отметить, что, когда вы выбираете базу данных по умолчанию в графическом интерфейсе, связанный пользователь базы данных не создается. Вы просто указываете, какой должна быть база данных по умолчанию. При этом вы используете вкладку User Mapping («Сопоставление пользователей») диалогового окна Create Login («Создание учетной записи»), чтобы создать связанного пользователя базы данных.

Перейдем к следующей вкладке Server Roles («Роли сервера»), показанной на рисунке 3.5. На этой странице можно выбрать любые роли на уровне SQL Server (экземпляра) для нового пользователя. Роли сервера представляют собой коллекции прав, также известные как защищаемые объекты, которые упаковываются в коллекцию, чтобы вам не приходилось назначать права каждому защищаемому объекту отдельно. По умолчанию каждая учетная запись является членом общедоступной роли, что позволяет установить основное подключение к экземпляру SQL Server.

|  |
| --- |
|  |
| ***Рисунок 3.5. Вкладка Server Roles*** |

Следующая страница диалогового окна Create Login в среде SQL Server Management Studio предназначена для сопоставления учетных записей пользователей. Каждая учетная запись может иметь пользователя в одной или нескольких базах данных. На этой странице можно создать пользователей базы данных, связанных с новой учетной записью. Для этого нужно предоставить следующую информацию.

Database («База данных»).Установите флажок рядом с базой данных, в которой нужно создать связанного пользователя для учетной записи.

User Name («Имя пользователя»)***.*** Имя объекта пользователя не обязательно соответствует имени учетной записи, и далее будет показано, как это можно изменить.

Default Schema («Схема по умолчанию»). Каждый пользователь базы данных должен быть назначен схеме по умолчанию. Схема представляет собой коллекцию объектов базы данных, отделенных логически (но не обязательно физически) от других объектов в базе данных. Можно предоставить пользователю или группе пользователей права для всех объектов данной схемы, например предоставить всем пользователям из бухгалтерии (или учетной записи службы бухгалтерского приложения) определенные права для всех объектов в схеме Billing, но не давать доступ к этим объектам другим пользователям. При назначении схемы по умолчанию для пользователя базы данных нет необходимости включать имя схемы в вызовы T-SQL к базе данных при адресации объектов в этой схеме. Это также означает, что если пользователю предоставлены права на создание объектов, то по умолчанию они будут созданы в этой схеме, если только не указать имя схемы при создании объектов. Далее в статье мы еще коснемся концепции схем.

Database Role Membership («Членство в роли базы данных»).Точно так же, как на уровне экземпляра или сервера, каждая база данных располагает заранее определенной коллекцией прав, упакованных в ролях. Чуть позже мы рассмотрим роли базы данных, поставляемые с Microsoft SQL Server.

|  |
| --- |
|  |
| ***Рисунок 3.6. Пример настроек учетной записи пользователя*** |

На следующей странице, Securables, представлены защищаемые объекты на уровне сервера или экземпляра. Как отмечалось выше, защищаемые объекты — это разрешения, предоставленные объектам. Защищаемые объекты обычно предоставляются в следующих случаях:

* предопределенная роль слишком широка (много других прав для учетной записи);
* назначенная роль или набор ролей не охватывает полностью все права, необходимые для учетной записи.

|  |
| --- |
|  |
| ***Рисунок 3.7. Назначение дополнительных прав*** |

Наконец, переходим к странице Status («Состояние»). На этой странице можно разрешить или отменить доступ для пользователя (по умолчанию выбирается Grant — разрешить). Поэтому можно создать учетную запись, предоставить права, создать связанных пользователей, а затем отменить доступ. Вы можете вернуться в это окно для существующего пользователя и отменить доступ к экземпляру SQL Server. Аналогично происходит включение и отключение учетной записи (см. рисунок 3.7). Наконец, мы можем просмотреть состояние учетной записи пользователя и узнать, была ли учетная запись заблокирована из-за слишком большого числа неудачных попыток регистрации с неверным паролем.

|  |
| --- |
|  |
| ***Рисунок 3.8 Вкладка Status*** |

На данном этапе важно отметить, как организована связь пользователей базы данных с учетной записью пользователя сервера. Как я уже указывал, соответствие имен между двумя объектами необязательно. Это объясняется тем, что объекты объединены в системных таблицах не по имени, а по идентификатору, именуемому sid (идентификатор безопасности). Это позволяет избавиться от привязки к учетной записи, соответствующей имени пользователя, или избежать возникновения ситуации, в которой вы восстанавливаете базу данных с именем пользователя.

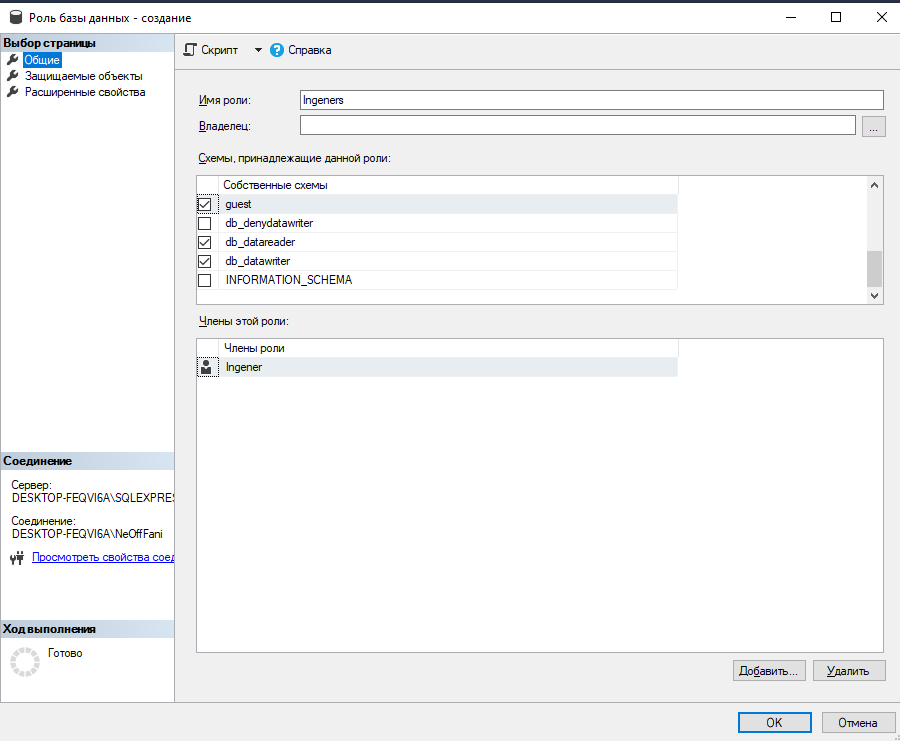
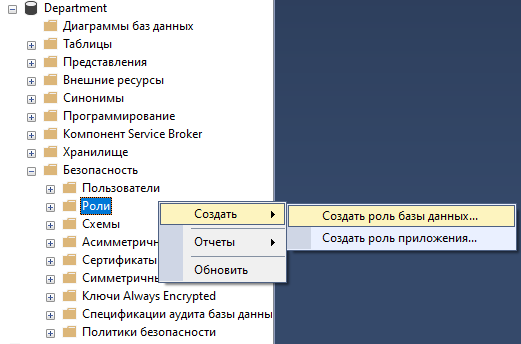
**3.3 Управление участниками уровня базы данных**

Чтобы легко управлять разрешениями в базах данных, SQL Server предоставляет несколько ролей, которые являются субъектами безопасности, которые группируют другие субъекты. Они похожи на группы в операционной системе Microsoft Windows. Разрешения ролей уровня базы данных распространяются на всю базу данных.

Существует два типа ролей уровня базы данных: предопределенные роли базы данных, являющиеся стандартными для базы данных, и пользовательские роли базы данных, которые можно создавать.

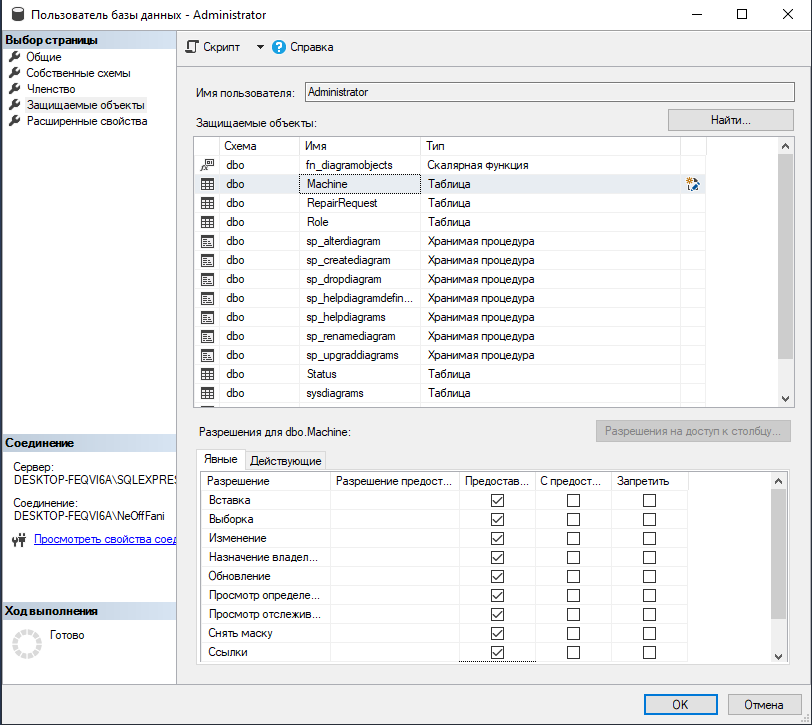
Предопределенные роли базы данных задаются на уровне базы данных и предусмотрены в каждой базе данных.

В SQL Server Management Studio разверните базу данных табличной модели, для которой требуется создать новую роль, щелкните правой кнопкой мыши роли и выберите команду “Создать роль”.

 *Рисунок 3.9. Создание новой роли*

*Рисунок 3.10. Настройка роли.*

**3.4 Управление разрешениями уровня базы данных**

Предоставление разрешений позволяет определенным пользователям или ролям выполнять определенные операции, такие как чтение, запись, изменение или удаление данных. Разрешения также могут быть назначены на уровне таблиц, представлений, хранимых процедур и других объектов базы данных. Это помогает обеспечить безопасность данных и предотвращать несанкционированный доступ к информации.

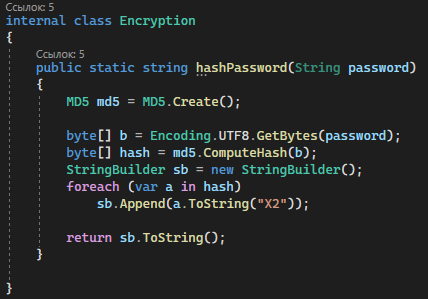
*Рисунок 3.11 Выдача прав пользователю*

**3.5 Шифрование данных баз данных**

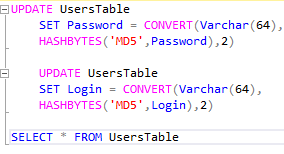
Шифрование данных баз данных - это процесс преобразования данных в зашифрованный формат с использованием алгоритмов шифрования, чтобы обеспечить их конфиденциальность и защитить от несанкционированного доступа. Шифрование может быть применено ко всей базе данных целиком или к отдельным колонкам с чувствительной информацией (например, пароли, личные данные и финансовые сведения).

При шифровании данных баз данных, информация хранится в зашифрованном виде и может быть дешифрована только с использованием соответствующего ключа. Это обеспечивает защиту данных, даже если злоумышленники получат физический доступ к базе данных или копию ее файлов.

Шифрование данных баз данных является важной мерой безопасности, помогающей предотвратить утечку информации и защитить чувствительные данные от несанкционированного доступа.

 В базе данных хранятся зашифрованные логины и пароли. При авторизации пользователя через приложение данные введённые в соответствующие поля, шифруются по тому-же ключу, которым зашифровали столбцы в базе данных и сравниваются в зашифрованном виде.

*Рисунок 3.12 Шифрование поля логин и пароль в приложении*

**

*Рисунок 3.13 Шифрование поля логин и пароль в базе данных*

**зАКЛЮЧЕНИЕ**

Информационные системы очень важны в наше время. Они позволяют исключить ошибки при работе и сократить затраты времени. А также предоставляет своевременную и надёжную информацию, позволяет улучшить процесс принятия решения.

Базы данных остаются критически важными в современном информационном обществе, обеспечивая эффективное хранение, организацию и анализ данных, поддерживая безопасность информации и улучшая бизнес-процессы.

В ходе выполнения курсового проекта были выполнены следующие задачи:

* выбран инструментарий;
* спроектирована база данных;
* сделано разграничение прав пользователей для базы данных;
* сделано шифрование данных;
* разработана информационная система.

Экономический эффект от внедрения автоматизированной информационной системы «Планирование обслуживания технического обслуживания станков в цеху» ожидается за счет сокращения затрат времени на выполняемые администратором и главным инженером, исключения ошибок при формировании отчетов, увеличения времени на анализ. При ручном создании отчетов могут быть допущены ошибки, правильно составленный алгоритм разрабатываемой программы экономит время и исключает ошибки.

Исходя из общего положения и опираясь на совокупность всех ранее вышеперечисленных и упомянутых фактов можно сделать вывод, что поставленные цель и задачи при разработки информационной системы были выполнены в полной мере.

Ссылка на репозиторий: https://github.com/NeOffFani/Kursach4Kyrs

**СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ**

1. Грин Дженнифер. Head First: изучаем С#/ Грин Дженнифер, Стиллмен Эндрю – Санкт-Питербург: 2022.
2. Мэтью Мак-Дональд WPF Windows Presentation Foundation в .NET 4.5 с примерами на C#: для профессионалов/ Мэтью Мак-Дональд: Санкт-Питербург, 2016.

Интернет-ресурсы

1. Документация по Windows Presentation Foundadion // Microsoft Learn. Режим доступа: <https://learn.microsoft.com/ru-ru/dotnet/desktop/wpf/?view=netdesktop-6.0>
2. Документация по C#. Руководства, справочные материал // Microsoft Learn. Режим доступа: <https://learn.microsoft.com/ru-ru/dotnet/csharp/>
3. Документация по SQL Server // Microsoft Learn. Режим доступа: <https://learn.microsoft.com/ru-ru/sql/sql-server/?view=sql-server-ver16>
4. Руководство по MS SQL Server // Metanit. Режим доступа: <https://metanit.com/sql/sqlserver/>
5. Обзор функционала SQL Server Management Studio (SSMS) – Режим доступа: <https://info-comp.ru/ssms-feature-overview?ysclid=lenvbogeq8366784796>
6. WPF и C# | Полное руководство // Metanit. Режим доступа: <https://metanit.com/sharp/wpf/>