МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Программной инженерии

Специальность 1-40 01 01 Программное обеспечение информационных технологий

Направление специальности 1-40 01 01 10 Программное обеспечение информационных технологий (программирование интернет-изданий)

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

«База данных для программного средства «Гостиничный комплекс» с применением технологии мониторинга состояния СУБД»

Выполнил студент Гришин Иван Сергеевич

(Ф.И.О.)

Руководитель проекта доцент Блинова Евгения Александровна

(учен. степень, звание, должность, Ф.И.О., подпись)

Заведующий кафедрой к.т.н., доц. Смелов Владимир Владиславович

(учен. степень, звание, должность, Ф.И.О., подпись)

Консультант: доцент Блинова Евгения Александровна

(учен. степень, звание, должность, Ф.И.О., подпись)

Нормоконтролер: доцент Блинова Евгения Александровна

(учен. степень, звание, должность, Ф.И.О., подпись)

Курсовой проект защищен с оценкой

Минск 2021

**СОДЕРЖАНИЕ**

[**ВВЕДЕНИЕ** 3](#_Toc90513957)

[**1.** **Постановка задачи** 4](#_Toc90513958)

[**2.** **Разработка модели** 5](#_Toc90513959)

[**3.** **Разработка необходимых объектов** 7](#_Toc90513960)

[**3.1.** **Логины** 7](#_Toc90513961)

[**3.2.** **Роли** 8](#_Toc90513962)

[**3.3.** **Пользователи** 8](#_Toc90513963)

[**3.4.** **Таблицы** 9](#_Toc90513964)

[**3.5.** **Представления** 11](#_Toc90513965)

[**3.6.** **Индексы** 11](#_Toc90513966)

[**3.7.** **Хранимые процедуры** 12](#_Toc90513967)

[**3.8.** **Функции** 12](#_Toc90513968)

[**3.9.** **Триггеры** 12](#_Toc90513969)

[**4.** **Описание процедур импорт и экспорта данных** 14](#_Toc90513970)

[**4.1.** **Экспорт данных** 14](#_Toc90513971)

[**4.2.** **Импорт данных** 15](#_Toc90513972)

[**5.** **Тестирование производительности** 16](#_Toc90513973)

[**5.1.** **Тесты без индекса** 16](#_Toc90513974)

[**5.2.** **Тесты с индексом** 17](#_Toc90513975)

[**5.3.** **Сравнительный анализ результатов тестирования** 18](#_Toc90513976)

[**6.** **Мониторинг состояния СУБД** 19](#_Toc90513977)

[**7.** **Руководство пользователя** 24](#_Toc90513978)

[**7.1** **Руководство для клиента** 24](#_Toc90513979)

[**7.2** **Руководство для работника** 25](#_Toc90513980)

[**ЗАКЛЮЧЕНИЕ** 27](#_Toc90513981)

[**СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ** 28](#_Toc90513982)

[**Приложение А** 29](#_Toc90513983)

[**Приложение Б** 30](#_Toc90513984)

**ВВЕДЕНИЕ**

Данный проект представляет собой базу данных отеля. Целью проекта является учебно-познавательная деятельность по освоению навыков проектирования базы данных. База данных – это совокупность взаимосвязанных данных. Данные в БД должны соответствовать следующим требованиям:

* полезность - уменьшает информационную энтропию системы;
* полнота информации - информации должно быть достаточно, чтобы осуществить качественное управление;
* точность;
* достоверность - заведомо ошибочные данные не должны храниться в базе данных;
* непротиворечивость;
* актуальность.

Для успешного проектирования базы данных необходимо знать, из каких этапов оно состоит. Этапы проектирования базы данных:

* определение границ исследуемой области – предметной области;
* системный анализ – определение объектов и связей между ними;
* построение логической схемы базы данных в соответствии с определенными правилами – моделью данных;
* реализация базы данных – описание ее в терминах некоторой СУБД.

В проекте также используются технологии мониторинга состояния СУБД с помощью программы «SQL SENRTY» от компании SOLARWINDS.

# **Постановка задачи**

Целью данной работы является проектирование базы данных для программного средства гостиничного комплекса с технологией мониторинга состояния СУБД. В качестве модели данных следует использовать реляционную модель. Проектирование необходимо произвести таким образом, чтобы конечные данные соответствовали общим требованиям к информации в базе данных.

Функционал должен позволять:

* регистрировать нового пользователя;
* просматривать тип номеров;
* просматривать количество свободных номеров по типу номера;
* бронировать номер;
* заселять клиентов в номер (функция работника);
* просматривать доступные сервисы;
* предоставлять сервисы (функция работника);
* просматривать забронированные номера (функция работника);
* просматривать занятые номера (функция работника);
* выселять клиентов из номера (функция работника);
* просматривать заказанные сервисы (функция работника).

Для реализации функционала необходимо разработать ряд объектов базы данных. К таким объектам относятся:

* роли;
* логины;
* пользователи;
* таблицы;
* представления;
* индексы;
* хранимые процедуры;
* функции;
* триггеры.

# **Разработка модели**

Опираясь на тематику, можно выделить основные сущности для хранения и предоставления данных пользователю, которые должны быть реализованы в виде таблиц:

* занятые номера; ОПИСАТЬ ВСЕ ПОЛЯ
* пользователи;
* номера;
* типы номеров;
* забронированные номера;
* сервисы;
* типы сервисов.

Каждая из сущностей и набор их свойств будет рассмотрена далее.

Занятые номера. Является основной сущностью, которая содержит в себе информацию о номерах, связывает сущности «Сервисы», «Номера» и «Пользователи». Имеет следующие свойства:

* комната;
* пользователь;
* дата въезда;
* дата выезда;
* стоимость проживания.

Пользователи. Необходима для изолирования клиентов друг от друга путём создания индивидуальных аккаунтов. Свойства:

* адрес электронной почты;
* пароль;
* имя;
* фамилия;
* номер телефона.

Номера. Необходима для поиска свободных номеров в отеле. Свойства:

* номер комнаты;
* тип номера;
* статус доступности.

Типы номеров. Необходима для выбора типа номера. Свойства:

* тип номера;
* цена за день;
* описание типа.

Забронированные номера. Необходима для хранения информации о забронированных номерах и последующего перевода номера в занятое состояние. Свойства:

* номер комнаты;
* пользователь;
* дата въезда;
* дата выезда;
* стоимость проживания.

Сервисы. Необходима для добавления сервисов к занятым пользователями комнатами. Свойства:

* номер занятой комнаты;
* сервис.

Тип сервиса. Необходима для выбора сервиса. Свойства:

* название сервиса;
* цена.

# **Разработка необходимых объектов**

Для реализации поставленной задачи были разработаны следующие объекты базы данных:

* логины;
* роли;
* пользователи;
* таблицы;
* представления;
* индексы;
* хранимые процедуры;
* функции;
* триггеры.

Далее будет приведено описание каждого созданного объекта.

## **Логины**

Для разграничения возможностей работника и клиента были созданы два логина для подключения к базе данных:

* HOTEL\_CLIENT;
* HOTEL\_EMPLOYEE.

Тексты скриптов приведены ниже. ВСЕ СКРИНЫ ГОВНО

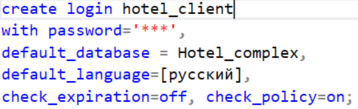


Рисунок 3.1.1 – «Создание логина клиента»

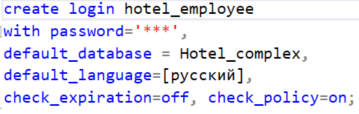


Рисунок 3.1.2 – «Создание логина работника»

Логины клиента и работника отличаются только именами и паролями. Следует отметить, что пароли для логинов были намеренно скрыты.

* 1. **Роли**

Для определения функциональных возможностей клиентов и работников были созданы две роли:

* HOTEL\_CLIENT\_ROLE;
* HOTEL\_EMPLOYEE \_ROLE.

Сразу после чего этим ролям были выделены соответствующим им привилегии для выполнения определенных действий.

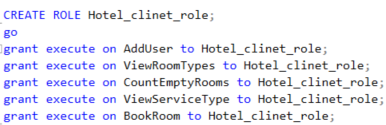
Тексты скриптов приведены ниже:

Рисунок 3.2.1 – «Создание роли клиента»

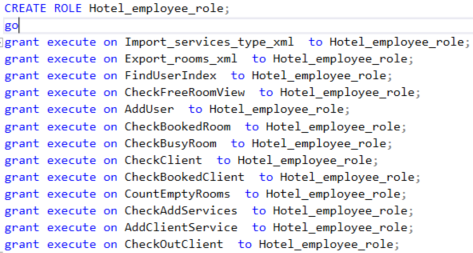
Также следует отметить, что возможности изменять структуру базы данных, а именно: создание новых объектов и изменение имеющихся, отсутствует в обоих ролях.

Рисунок 3.2.2 – «Создание роли работника»

## **Пользователи**

Для реализации вышеупомянутый ролей были созданы два пользователя:

* HOTEL\_COMPLEX\_CLIENT;
* HOTEL\_COMPLEX\_EMPLOYEE.

Скрипт создания приведен ниже.

Рисунок 3.3.1 – «Создание пользователей»

В MS SQL для смены пользователя необходимо создавать новое соединение с сервером, что представляет собой ранее созданный логин (имя для входа). При создании пользователей идёт явная привязка пользователей к именам для входа, для автоматического подключения к пользователям через соответствующие логины.

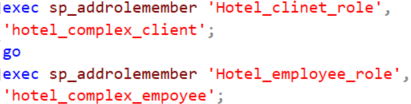
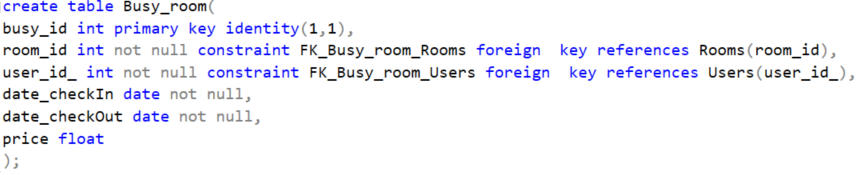
После чего идёт привязка ролей к созданным пользователям. Скрипт привязки приведен ниже:

Рисунок 3.3.2 – «Привязка ролей»

## **Таблицы**

Таблицы являются отражением математических моделей на объекты базы данных. Список сущностей и связанные с ними свойства приведены в главе №2. Поля созданных таблиц содержат в себе все перечисленные свойства, а также поля, необходимые для связи сущностей. Листинги скриптов создания таблиц приведены ниже.

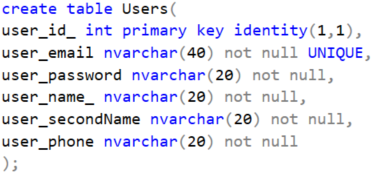
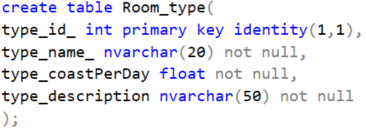
Рисунок 3.4.1 – «Создание таблицы занятых комнат»

Рисунок 3.4.2 – «Создание таблицы пользователей»

Так как подразумевается, что адрес электронной почты пользователя уникален, были добавлены соответствующие ограничения целостности.

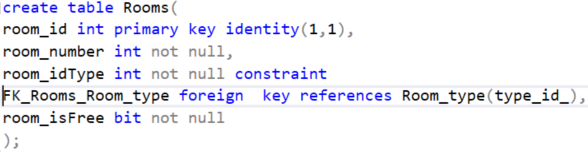
Рисунок 3.4.3 – «Создание таблицы тип комнат»

Рисунок 3.4.4 – «Создание таблицы комнат»

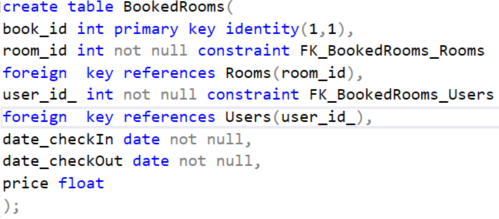
Для реализации связи с комнатой и типом комнаты, см. гл. 2, были добавлены ограничения целостности внешнего ключа. Индикатором того, занята ли комната, служит поле ROOM\_ISFREE, имеющее по умолчанию значение 1.

Рисунок 3.4.5 – «Создание таблицы забронированных комнат»

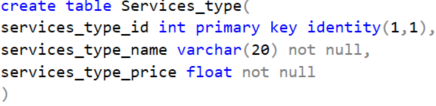
Для реализации связи с комнатой и пользователем, см. гл. 2, были добавлены ограничения целостности внешнего ключа.

Рисунок 3.4.6 – «Создание таблицы типов сервисов»

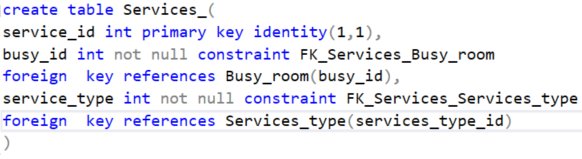
Рисунок 3.4.7 – «Создание таблицы сервисов»

Диаграмма баз данных, полученная в результате создания таблиц, приведена в приложении А.

## **Представления**

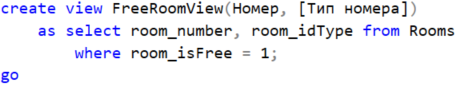
В результате работы было создано представление для быстрого доступа к свободным номерам. Листинг кода приведен ниже.

Рисунок 3.5.1 – «Создание представления»

## **Индексы**

Индексы необходимы для повышения производительности выполнения поиска данных. Таблицы в базе данных могут иметь большое количество строк, которые хранятся в произвольном порядке, и их поиск по заданному критерию путём последовательного просмотра таблицы строка за строкой может занимать много времени. Для оптимальной производительности запросов индексы обычно создаются на тех столбцах таблицы, которые часто используются в запросах. Однако увеличение числа индексов замедляет операции добавления, обновления, удаления строк таблицы, поскольку при этом приходится обновлять сами индексы. Так же следует учесть, что при создании таблицы с первичным ключом, автоматические создает кластеризованный индекс на основе этого ключа, который может существовать только один на каждую таблицу.

Учитывая то, что количество пользователей со временем может стать огромным, а все их данные хранятся в базе данных постоянно, поиск необходимого клиента может занимать большое количество времени. По причине чего был создан некластеризованный индекс, для быстрого поиска пользователя по его электронной почте. Скрипт создания индекса представлен ниже.

Рисунок 3.6.1 – «Создание индекса»

## **Хранимые процедуры**

Доступ к данным осуществляется через хранимые процедуры. Хранимая процедура – это блок SQL-команд, который компилируется и хранится на сервере. Процедуры могу иметь входные и выходные параметры с помощью которых выполняется поиск, обработка и возврат данных.

Хранимые процедуры поделены на блоки, в зависимости о того, к какой сущности они применяются.

Листинги хранимых процедур, а также краткое описание их назначения описаны в приложении Б.

## **Функции**

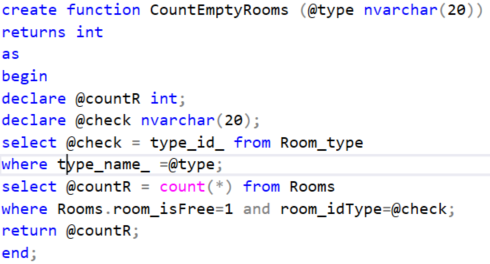
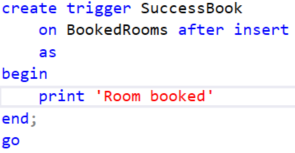
Для получения количества оставшихся свободных номеров в качестве выходного параметра используется функция. Функция SQL — это набор операторов SQL, которые принимают входные данные и выполняют с ними действия SQL, а затем возвращают результаты в виде выходных данных. Листинг функции приведен ниже.

Рисунок 3.8.1 – «Создание функции»

## **Триггеры**

Триггеры – это специальный тип хранимой процедуры, которая запускается автоматические системой SQL SERVER при модификации какой-либо таблицы одним из трех операторов: UPDATE, INSERT, DELETE. Для уведомления клиентов об удачном бронировании комнаты в отеле был создан триггер, представленный на рисунке ниже.

Рисунок 3.9.1 – «Создание триггера»

# **Описание процедур импорт и экспорта данных**

Для импорта и экспорта данных был выбран формат XML. Несмотря на то, что в SQL Server есть встроенное средство импорта и экспорта данных, данные операции будут выполнены в рамках хранимой процедуры.

XML – это расширяемый язык разметки. Он имеет простой формальный синтаксис, понятный и удобный для создания и обработки документов как программами, так и человеком. Язык является расширяемым, поскольку он не фиксирует разметку, используемую в документах: разработчик волен создавать разметку в соответствии с потребностями к конкретной области, будучи ограниченным лишь синтаксическими правилами языка.

## **Экспорт данных**

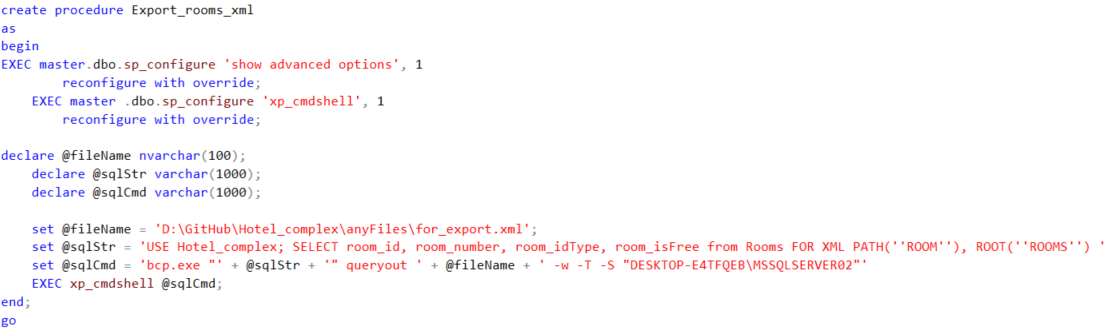
Листинг процедуры, осуществляющей экспорт данных приведен ниже.

Рисунок 4.1.1 – «Экспорт данных»

Для экспорта была взята таблица «Rooms», так как она является ключевой в базе данных гостиницы. А также, перед выполнением процедуры, был создан файл в формате XML для записи данных из базы данных. В блоках «declare» процедуры были объявлены следующие переменные:

* filename: в эту переменную записывается путь к файлу, имя файла и его расширение;
* sqlStr: содержит sql скрипт, включающий в себя присоединение к базе данных, выбор нужных столбцов из необходимой таблицы, а так же имя для корневого и разделительного тегов XML;
* sqlCmd: включает в себя запуск утилиты bcp, переменные, описанные выше, а так же имя устройства и имя сервера;

Для работы с XML были изменены значения параметров «SHOW ADVANCED OPTIONS» для отображения всех параметров, которые возвращает системная процедура «SP\_CONFIGURE» и «XM\_CMDSHELL» для того, чтобы позволить контролировать, можно ли выполнять в системе расширенную хранимую процедуру «XP\_CMDSHELL». Однако, при изменении последнего, важно учесть возможные проблемы безопасности, поскольку данные параметр очень устарел и лучше его всегда оставлять выключенным.

После включения необходимых параметров и присваивания значений переменным, вызывается команда «XP\_CMDSHELL», вызывающая «BCP.EXE» для экспорта данных.

BCP(bulk copy program) – строенная консольная утилита, которая применяется для массового перемещения данных между серверов MS SQL и файлом пользователя в необходимом направлении.

## **Импорт данных**

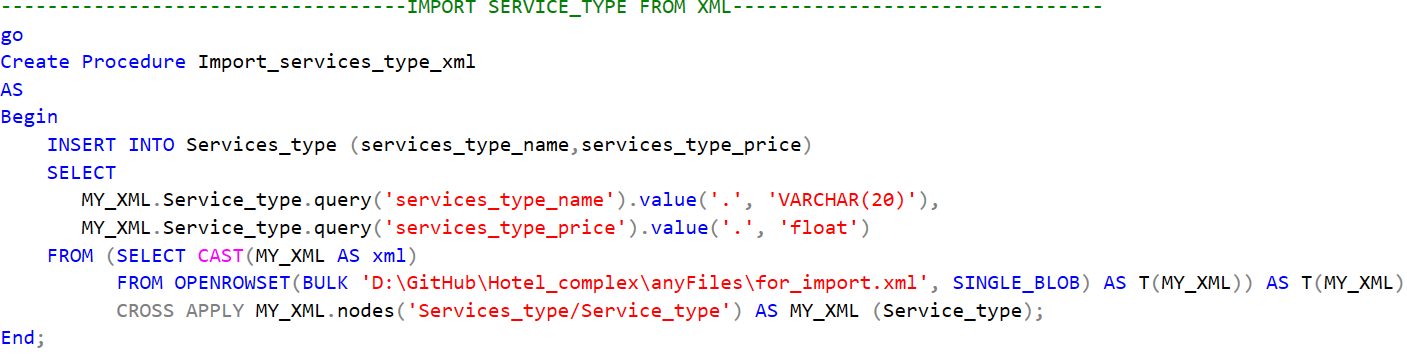
Для импорта была выбрана таблица «SERVICE\_TYPE». Поскольку в результате роста и развития отеля, могут появляться новые виды услуг. Скрипт процедуры для импорта данных представлен ниже.

Рисунок 4.2.1 – «Импорт данных»

Для начала, необходимо указать таблицу и соответствующие ей поля, в которые будут экспортироваться данные. После чего идёт описание типов данных для полей таблицы. При помощи встроенной функции «SQL CAST» идёт преобразование данных к нужному формату. Далее функция «OPENROWSET» позволяет получить доступ к файлам, она считывает содержимое удаленного источника данных, которым является файл, указанный в данной функции. В конце вызывается оператор «CROSS APPLY», которая, в данной процедуре, указывает корневой тег XML и разделительный тег, соответствующий имени таблицы.

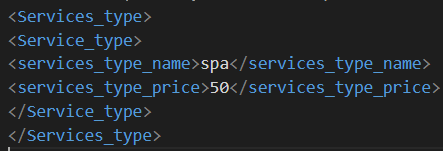
Перед вызовом процедуры импорта, необходимо создавать файл для импорта в папке, указанной в скрипте процедуры. Содержимое файла для импорта представлено ниже.

Рисунок 4.2.2 – «Данные для импорта»

# **Тестирование производительности**

В процессе разработки была создана таблица «USERS», имеющая одну сотню тысяч строк, что является большим числом. В процессе работы с базой данных, работникам будет необходимо искать информацию, связанную с пользователем. А так как идентификатором пользователя является его адрес электронной почты, поиск будет производиться по ней. Однако данное поле не является первичным ключом таблицы, по причине чего был создан некластеризованный индекс. Для определения качества и необходимости созданных индексов будут выполнены запросы к данной таблице с и без индексов, а потом будет проведено сравнение и анализ полученных результатов.

## **Тесты без индекса**

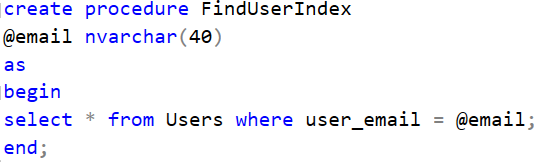
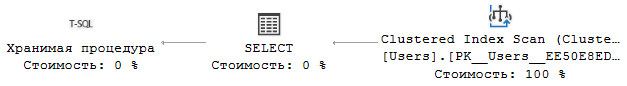
Для тестирования скорости поиска была создана процедура «FindUserIndex», имеющая электронный адрес пользователя, как входной параметр, и выполняющая поиск по нему. Так же была произведена работа по поиску пользователя, как части его электронной почты, однако индексы в таком случае не могу ускорить работу. Скрипт процедуры для поиска пользователя по его электронной почте представлен ниже.

Рисунок 5.1.1 – «Создание процедуры»

Для более точно оценки тестирования, используется фиксирования состояние базы данных и очистка буфера памяти, скрипты чего представлены ниже.

Рисунок 5.1.2 – «Фиксирование состояния»

План выполнения процедуры без индекса представлен ниже.

Рисунок 5.1.3 – «План запроса»

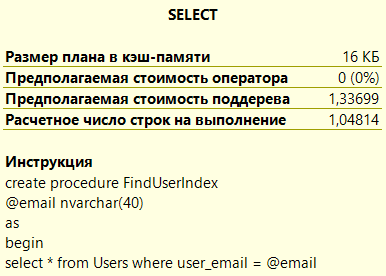
В данном случае поиск информации в таблице осуществляется с помощью кластеризованного индекса. Ключевым показателем для оценки скорости выполнения запроса является предполагаемая стоимость поддерева. Стоимость выполнения представлена ниже.

Рисунок 5.1.4 – «Стоимость запроса без индекса»

## **Тесты с индексом**

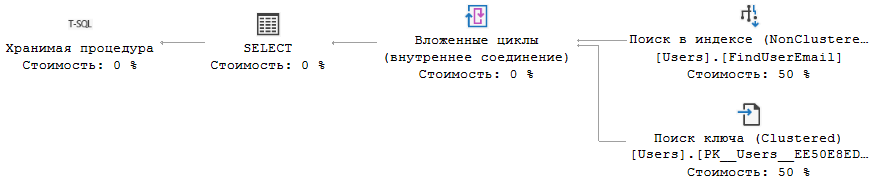
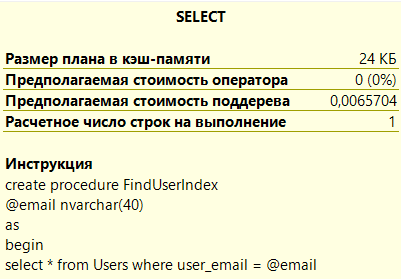
Далее создается индекс (см. главу 3.6). Последующие действия те же самые. План выполения запроса с индексом представлен ниже.

Рисунок 5.2.1 – «План запроса с индексом»

Стоимость выполнения той же процедуры представлен ниже.

Рисунок 5.2.2 – «Стоимость запроса с индексом»

## **Сравнительный анализ результатов тестирования**

Исходя из полученных результатов можно подвести итог, что без дополнительного индекса запросы стоят в 203 раза больше. Данный результат считаю показательным, а задачу по оптимизации выполненной.

# **Мониторинг состояния СУБД**

В качестве дополнительной технологии был выбран мониторинг состояния СУБД. В ходе аналитического анализа информации о всевозможных программах и дополнительных утилитах для мониторинга, была выбрана программа «SQL SENTRY».

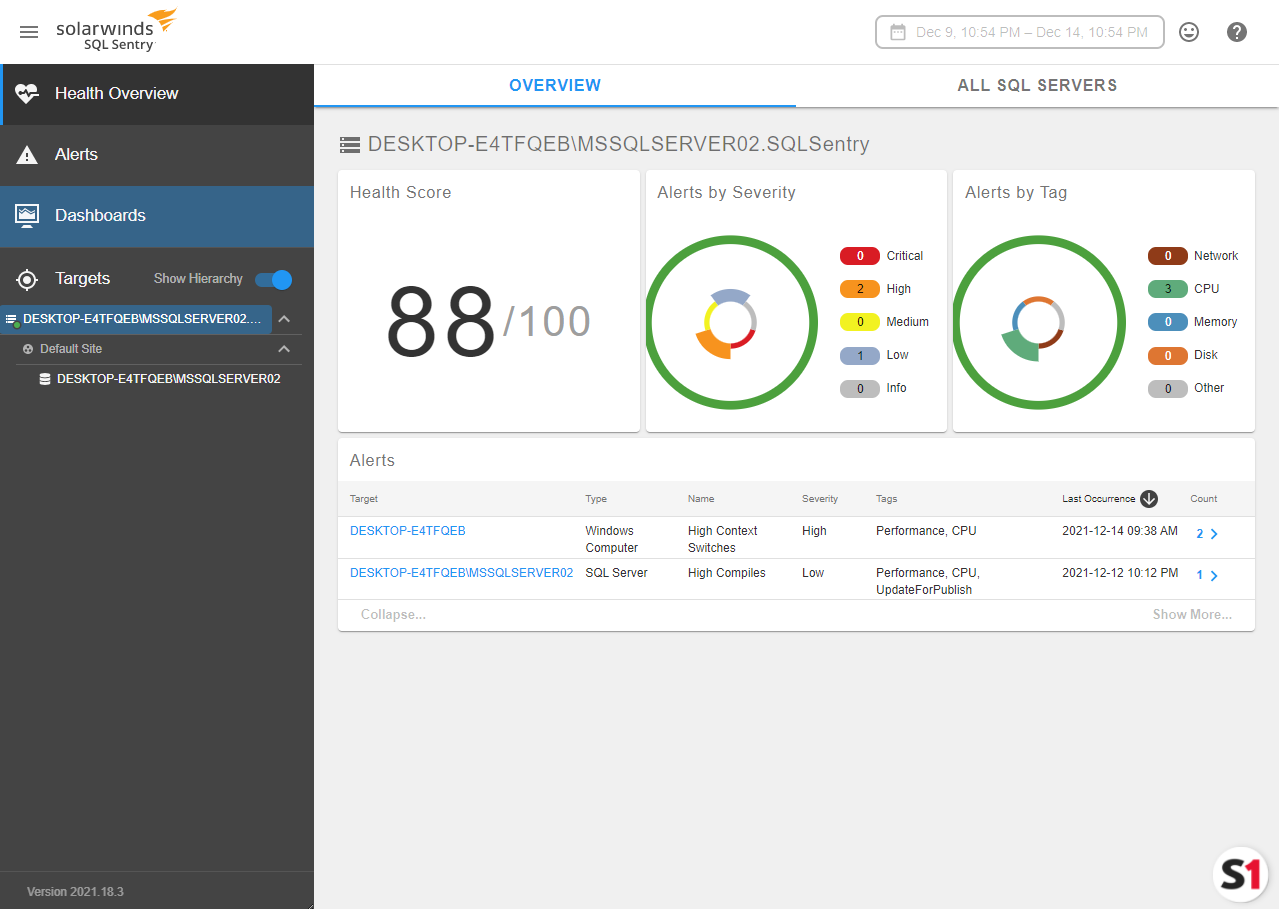
SQL Server – это клиент-серверная система управления базами данных, которая лежит в основе большинства существующих программных продуктов. Наиболее популярной и распространенной СУБД такого типа считается клиент-серверная СУБД Microsoft SQL Server. Однако ее аналоги можно встретить и у других разработчиков софта: Oracle, IBM, Sysbase и т.д.

Преимуществом SQL Server является то, что управление СУБД осуществляется централизованно. Это позволяет своевременно выявлять и устранять уязвимости, ошибки, попытки несанкционированного доступа. Именно поэтому клиент-серверные СУБД считаются наиболее надежными и безопасными среди существующих аналогов. Однако, чтобы ПО работало правильно, необходимо регулярно проводить мониторинг SQL Server, то есть наблюдать за основными показателями работы СУБД и их влиянием на работоспособность ПО.

Наблюдение за работой сервера проводится в первую очередь для предотвращения возможных неполадок. В процессе мониторинга оценивается производительность ПО и уровень информационной безопасности, выявляются ошибки и уязвимости. Если не проводить эту процедуру регулярно, ошибки будут накапливаться. Это неизбежно отразится на качестве работы приложения. Поэтому лучше периодически проводить диагностику СУБД в стандартных условиях, чем в срочном порядке выполнять восстановление базы данных на сервере.

Sentry1 даёт возможность мониторить состояние СУБД MS SQL используя программу на устройстве или через браузер.

На начальной странице выводится оценка состояния СУБД и предупреждения, связанные с нарушением или замедлением работы СУБД.

Рисунок 6.1 – «Оценка состояния»

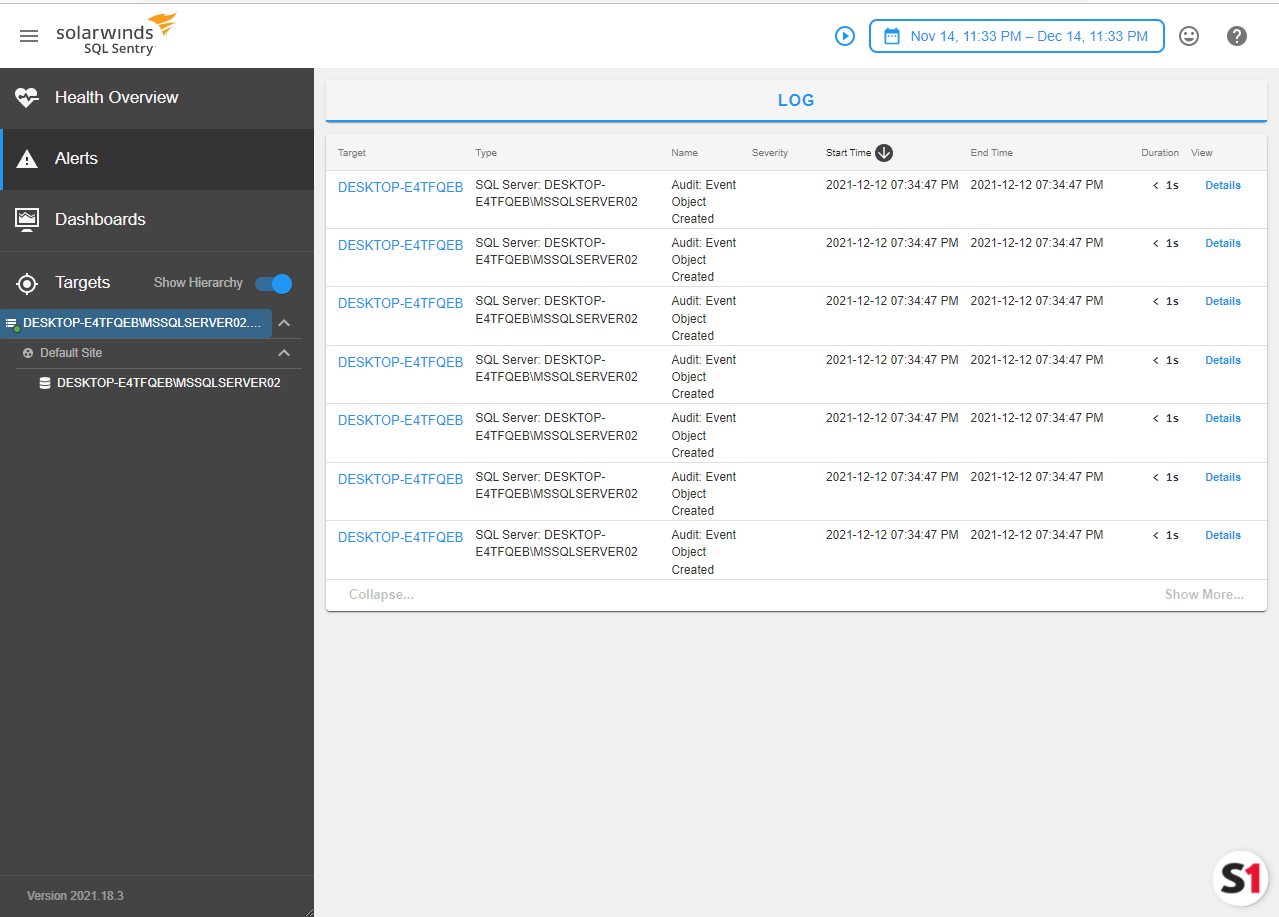
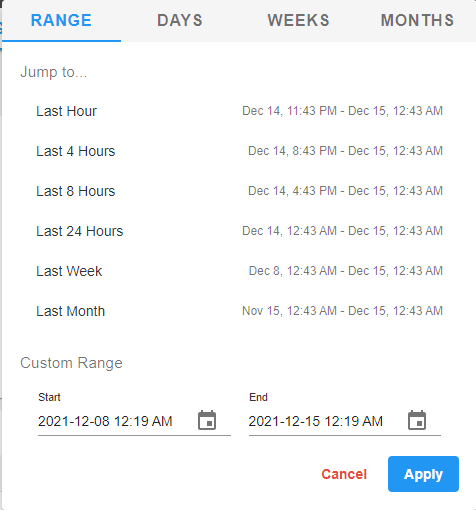
Выбрав пункт Alerts на панели справа, можно смотреть все уведомления, в основном связанные с созданием новых баз данных. В данном случае уведомляется о начале отслеживания их состояния.

Рисунок 6.2 – «Уведомления»

Для каждого раздела можно выбрать диапазон времени, для которого собирается информация, что очень удобно при анализе конкретных событий.

Рисунок 6.3 – «Диапазон времени»

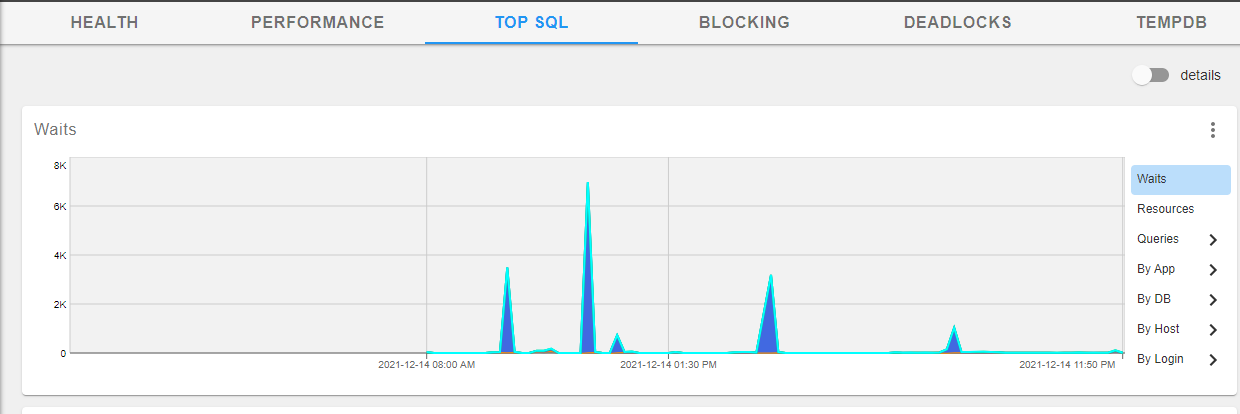
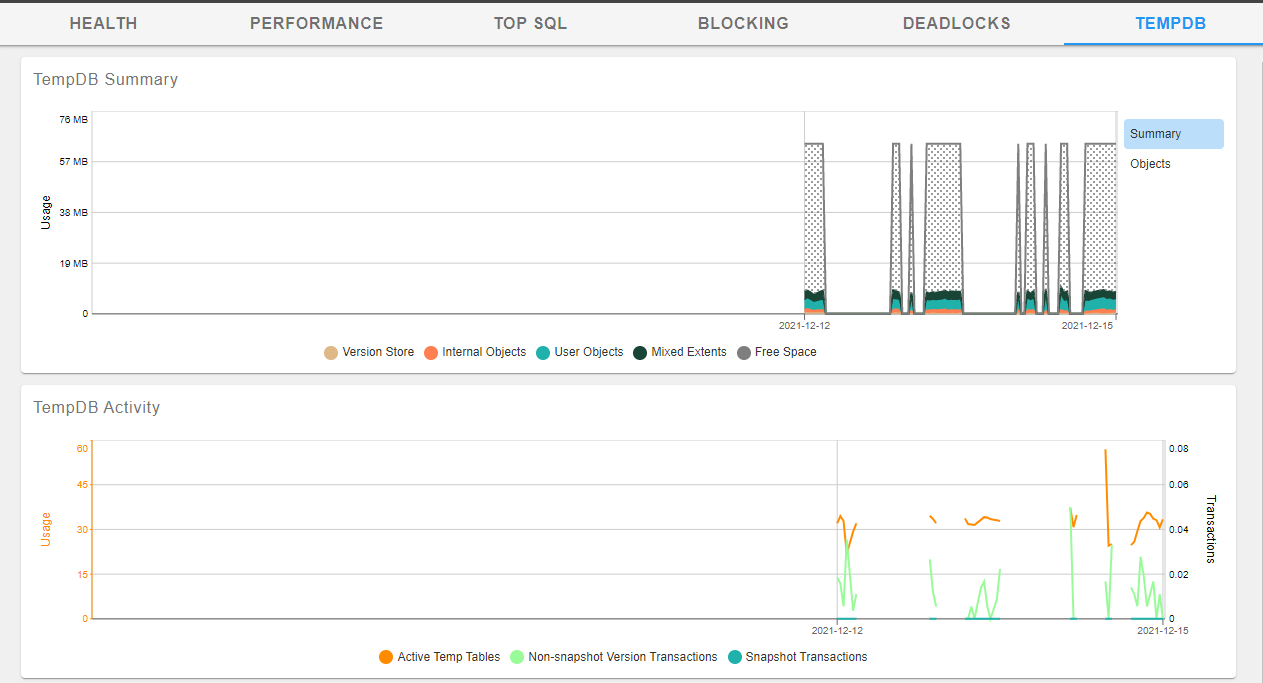
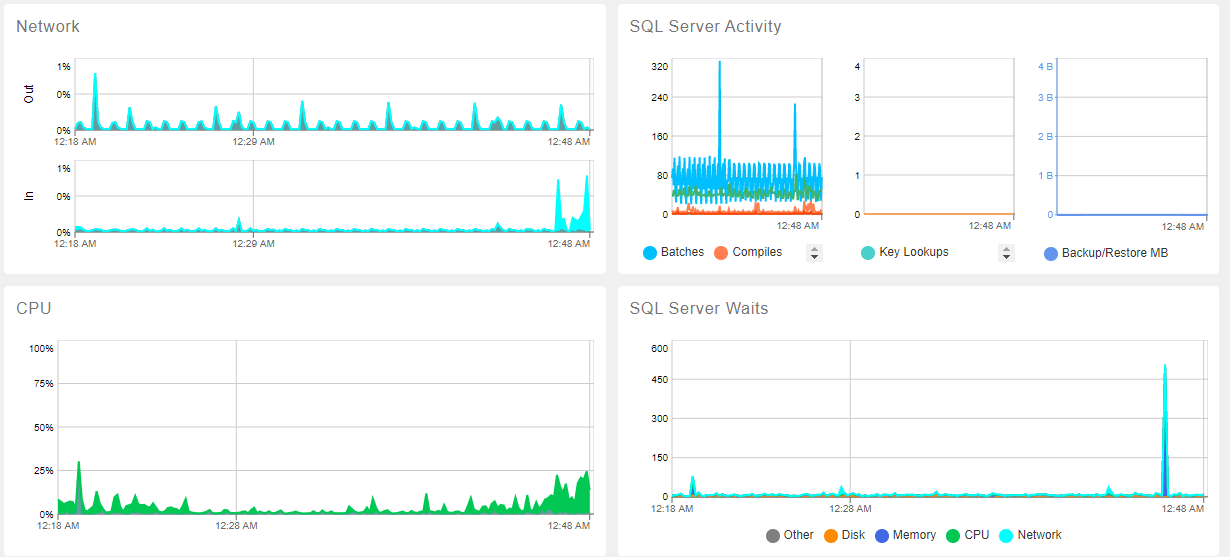
Панель TOP SQL предназначена для того, чтобы быстро находить запросы, которые имеют наибольшее время ожидания, используют наибольшее количество ресурсов и, соответственно, создают наибольшую нагрузку на сервер.

Рисунок 6.4 – «TOP SQL»

По данному графику стало понятно, что наибольшую нагрузку на сервер вызвало добавление одной сотни тысяч строк в таблицу.

Следующим значимым разделом идёт TEMPDB, который отображает информацию о базе данных и насколько эффективно она используется. TempDB является системной базой данных, которая используется для хранения временных данных.

Рисунок 6.5 – «TEMPDB»

Ключевым для анализа разделом является Performance. В левой части которого представлена нагрузка на систему устройства, а в правой нагрузку на сервер.

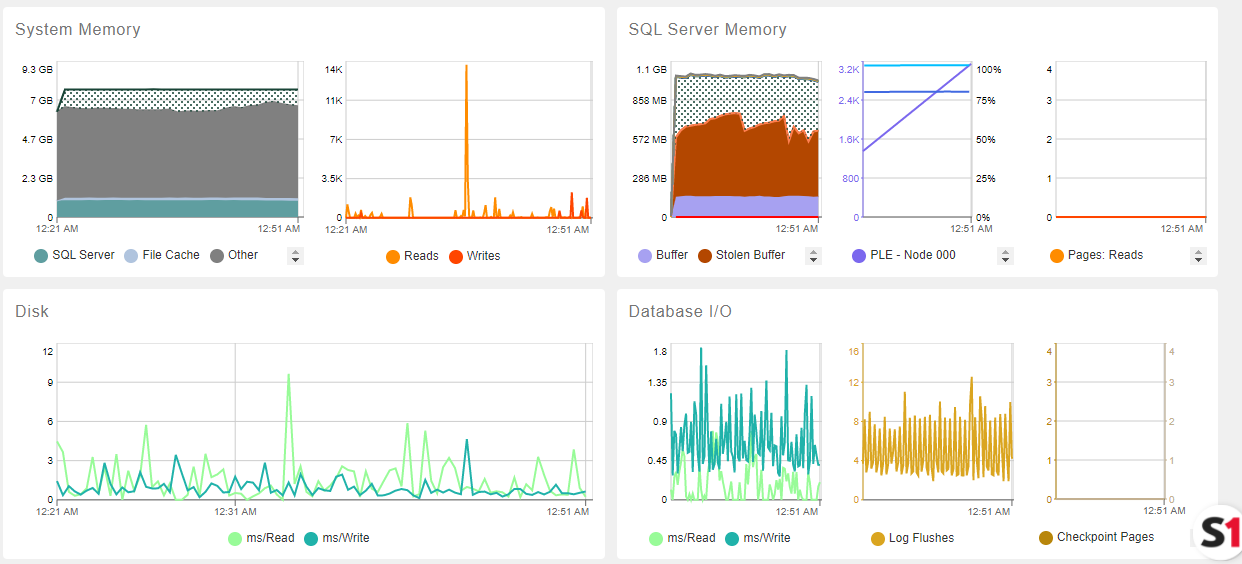
Рисунок 6.6 – «PERFORMANCE»

Рисунок 6.7 – «PERFORMANCE»

Здесь отслеживаются активность сервера, время ожидания сервера, память и дисковая активность, возникающая при чтении и записи блоков данных.

Огромным плюсом данной программы является выбор промежутков времени для анализа, возможность обновления с интервалом в полторы минуты, а так же получением расширенной информации о конкретных временных периодах.

Учитывая, что оценивается состояния всего сервера, создать серьезную нагрузку на который достаточно сложно, имея всего пару баз данных, не считая системных, состояния сервера оценивается более чем удовлетворительным. Все существенные скачки графиков вызваны периодическим выключением сервера при выключении компьютера. Все остальные показатели находятся в пределах нормы и не оказывают серьезного влияния на работоспособность сервера.

# **Руководство пользователя**

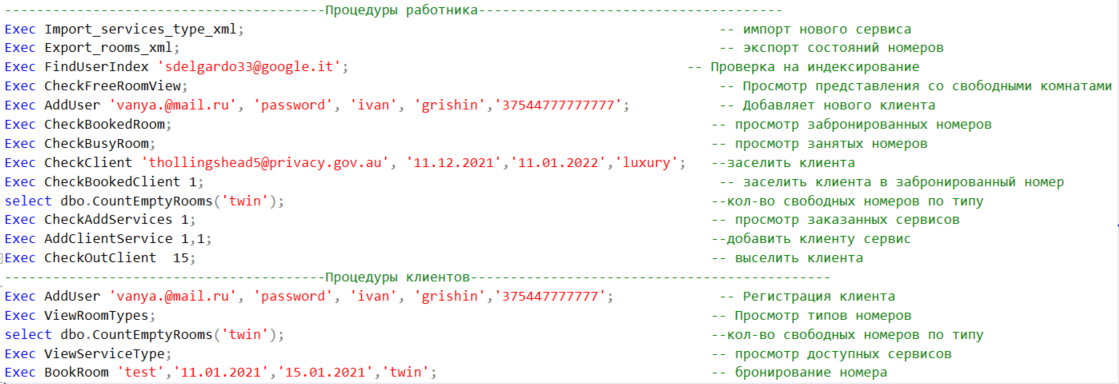
Построенный проект позволяет выполнять различные операции, в зависимости от типа пользователя. Функционал проект демонстрируется через вызов хранимых процедур и функций через утилиту SQL Server Management Studio (SSMS). Вызов команд происходит из файла exec\_proc.sql, скрипт которого представлен ниже.

Рисунок 7.1 – «Вызов процедур и функций»

* 1. **Руководство для клиента**

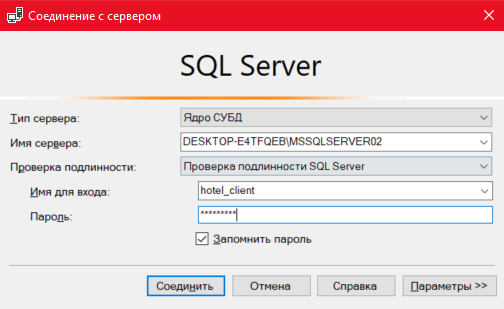
Для подключения к базе данных от лица клиента нужно поменять проверку подлинности, ввести логин и пароль.

Рисунок 7.1.1 – «Вход клиента»

Для регистрации пользователя используется вызов процедуры AddUser, куда в качестве входных параметров передается адрес электронной почты, пароль, имя, фамилия и номер телефон нового клиента.

После регистрации, клиент может узнать о типах номеров отеля и выбрать подходящий при помощи вызова процедуры ViewRoomTypes. После выбора типа номера, ему предлагается узнать кол-во свободных номеров данного типа на данный момент. Это реализуется при помощи вызова функции CountEmptyRooms с параметром в виде типа комнаты. В случае наличия свободных номеров, клиент может перейти к оформлению брони на этот номер при помощи процедуры BookRoom, указав тип номера, дату заезда и выезда, а также адрес электронной почты, которая является идентификатором клиента. Помимо этого, клиент может сразу ознакомиться с наличием дополнительных услуг, вызвав функцию ViewServiceType, однако заказать себе её он сможет только через сотрудников.

* 1. **Руководство для работника**

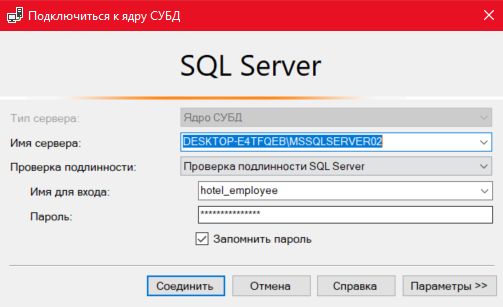
Процесс подключения работников в базе данных аналогичен с клиентом, используя только другой логин и пароль.

Рисунок 7.2.1 – «Вход работника»

К возможностям работника относятся импорт и экспорт XML.

Для импорта данных используется процедура Import\_services\_type\_xml, которая добавляет новый сервис в таблицу с типами сервисов. Данные, для внесения в таблицу, хранятся в файле for\_import.xml.

В случае с экспортом используется процедура Export\_rooms\_xml, которая экспортирует информацию о комнатах в файл for\_export.xml.

Для быстрого поиска информации о клиенте по его адресу электронной почты используется процедура FindUserIndex.

Работник имеет доступ к процедурам аналогичным процедурам клиента, а именно: добавления нового пользователя, просмотру количества свободных номеров по типу комнаты.

Работник так же имеет доступ к представлению со свободными комнатами при помощи вызова процедуры CheckFreeRoomView.

Работнику также необходимо иметь возможность просматривать информацию о забронированных и занятых номерах, что реализуется при помощи вызова процедур CheckBookedRooms и CheckBusyRooms соответственно.

Основной обязанностью работника является заселение клиентов в номера. Для этого используется вызов процедуры CheckClient, куда передаётся адрес электронной почты клиента, дату заезда, дату выезда и тип комнаты.

Также работник может заселить клиентов, забронировавших номер в отеле при помощи вызова процедуры CheckBookedClient, указав номер брони.

После заселения клиента, ему предоставляется возможность выбора дополнительных услуг из таблицы сервисов. Сотрудники могу назначать выбранные клиентом сервисы при помощи вызова процедуры AddClientServices. В качестве параметров указывается номер занятой комнаты и номер сервиса, который выбрал клиент. После чего идёт пересчёт итоговой стоимости за проживания. Далее сотрудники могут мониторить заказанные сервисы при помощи вызова процедуры CheckAddServices, указав номер занятой комнаты.

При выезде клиента из отеля, сотрудники удаляют его данные из таблицы с занятыми комнатами и, соответственно, предоставляют доступ к этой комнате другим клиентам при помощи процедуры CheckOutClient.

**ЗАКЛЮЧЕНИЕ**

В данном курсовом проекте были освоены навыки проектирования базы данных, а также спроектирована база данных для программного средства «Гостиничный комплекс» с использованием технологии мониторинга состояния СУБД.

В ходе работы выполнены следующие задачи:

* разработана модель данных;
* разработаны необходимые объекты базы данных;
* разработаны процедуры экспорта и импорта данных;
* произведено тестирование приложения;
* применена технология мониторинга состояния СУБД.

Модель данных состоит из 7 сущностей:

* занятые номера;
* пользователи;
* номера;
* типы номеров;
* забронированные номера;
* сервисы;
* типы сервисов.

Для реализации поставленной задачи были разработаны следующие объекты базы данных:

* логины;
* роли;
* пользователи;
* таблицы;
* представления;
* индексы;
* хранимые процедуры;
* функции;
* триггеры.

Для экспорта и импорта используется формат данных XML. Был произведен экспорт данных таблицы с комнатами и импорт данных в таблицу с сервисами.

Тестирование приложение на производительность показало эффективность предпринятых мер.

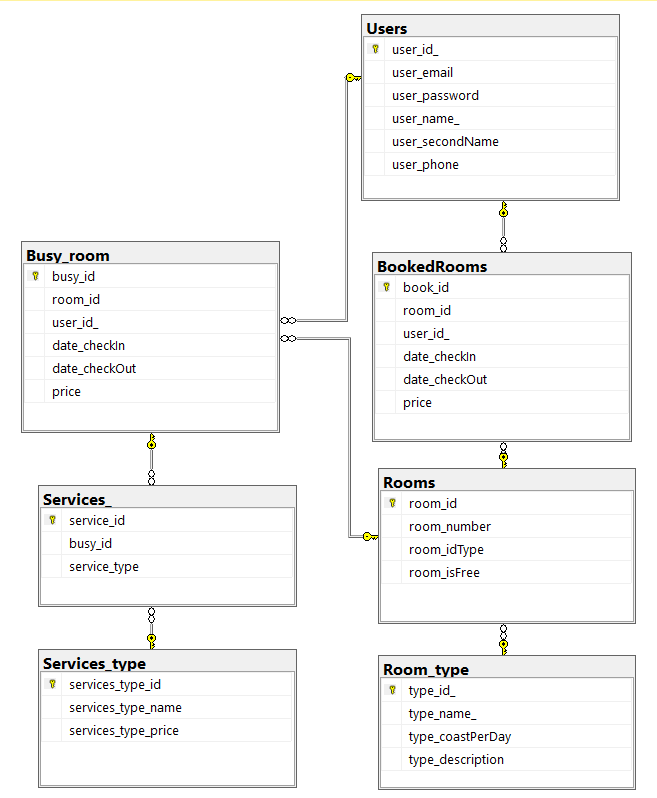
В рамках технологии мониторинга состояния СУБД было оценено конечное состояние, а также нагрузки за весь период манипуляций с данным в рамках разрабатываемой базы данных. В конечном итоге СУБД показывало практически эталонные показатели состояния.

# **СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ**

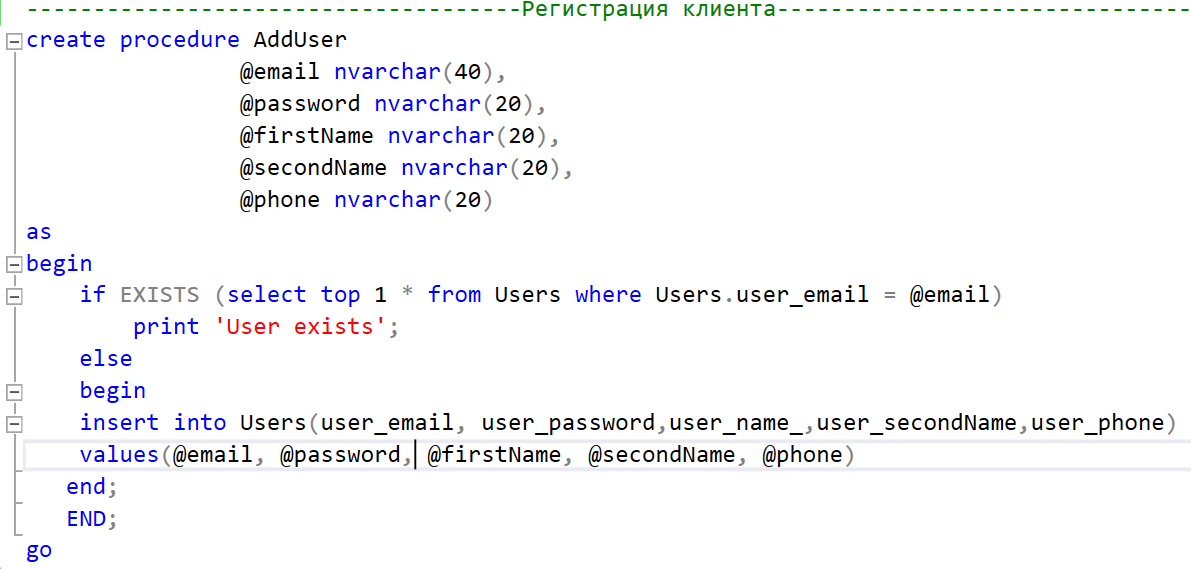
СДЕЛАТЬ ССЫЛКИ В ТЕКСТЕ НА СПИСОК ЛИТЕРАТУРЫ ЧЕРЕЗ [1,2,3]. ДОБАВИТЬ ДАТУ ДОСТУПА ДЛЯ САЙТОВ. ССЫЛКИ ДОЛЖНЫ БЫТЬ ТОЧНЫЕ,А НЕ ОБОБЩЕННЫЕ

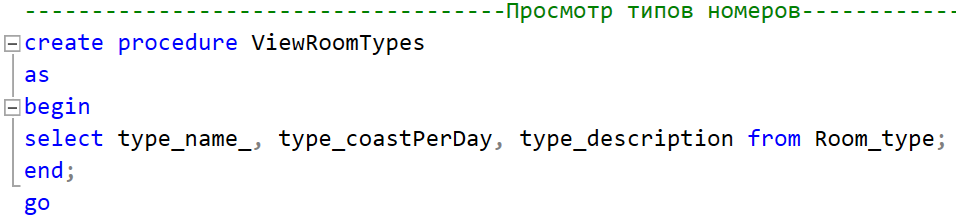
1. Д. Петкович. Microsoft SQL Server. Руководство для начинающих.
2. Е. А. Блинова, Н. Н. Пустовалова. Практикум базы данных. MICROSOFT SQL SERVER 2019.
3. Руководство по MS SQL SERVER 2019 [Электронный ресурс] / Режим доступа: https://metanit.com/sql/sqlserver/.
4. Документация MICROSOFT SQL [Электронный ресурс] / Режим доступа: https://www.microsoft.com/ru-ru/sql-server/sql-server-2019.
5. SolarWinds Worldwibe [Электронный ресурс] / Режим доступа: <https://www.sentryone.com>.
6. Leran2002. Учебник по языку SQL. Часть первая. [Электронный ресурс] / Режим доступа: https://habr.com/ru/post/255361/

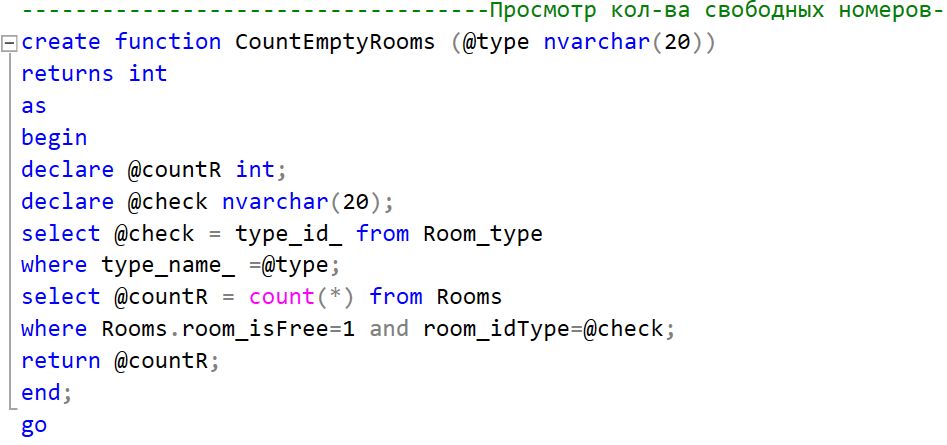
# **Приложение А**

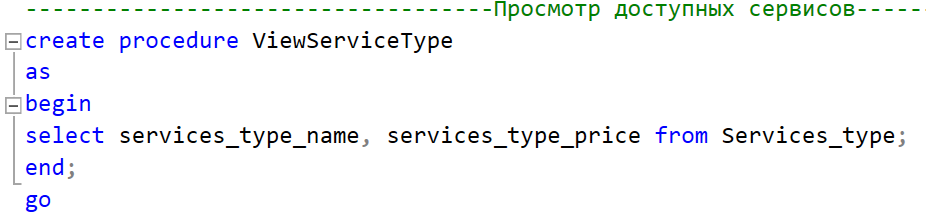
****

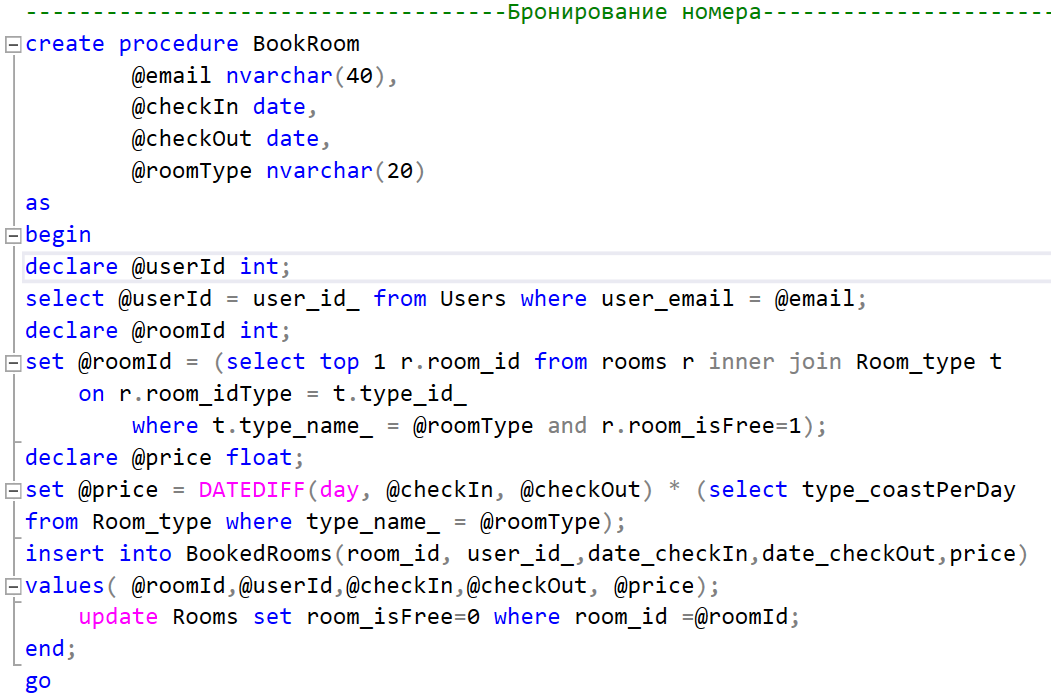
# **Приложение Б**

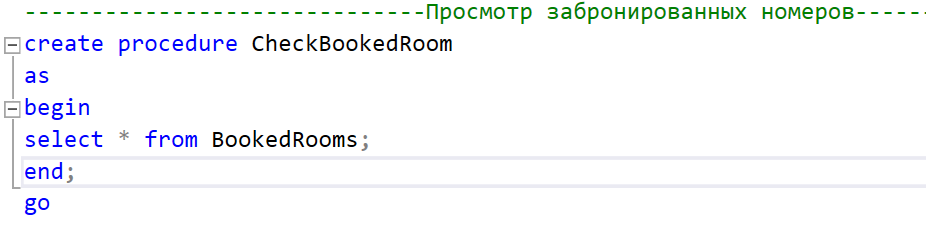
****

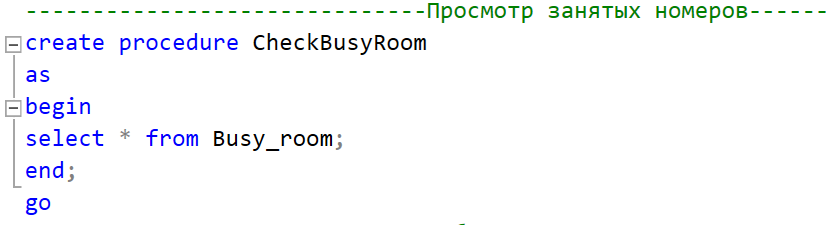
****

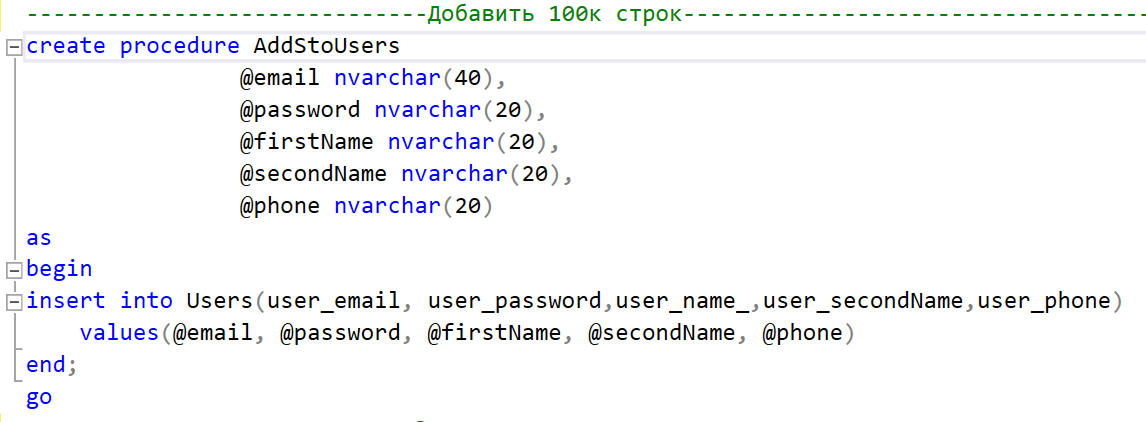
****

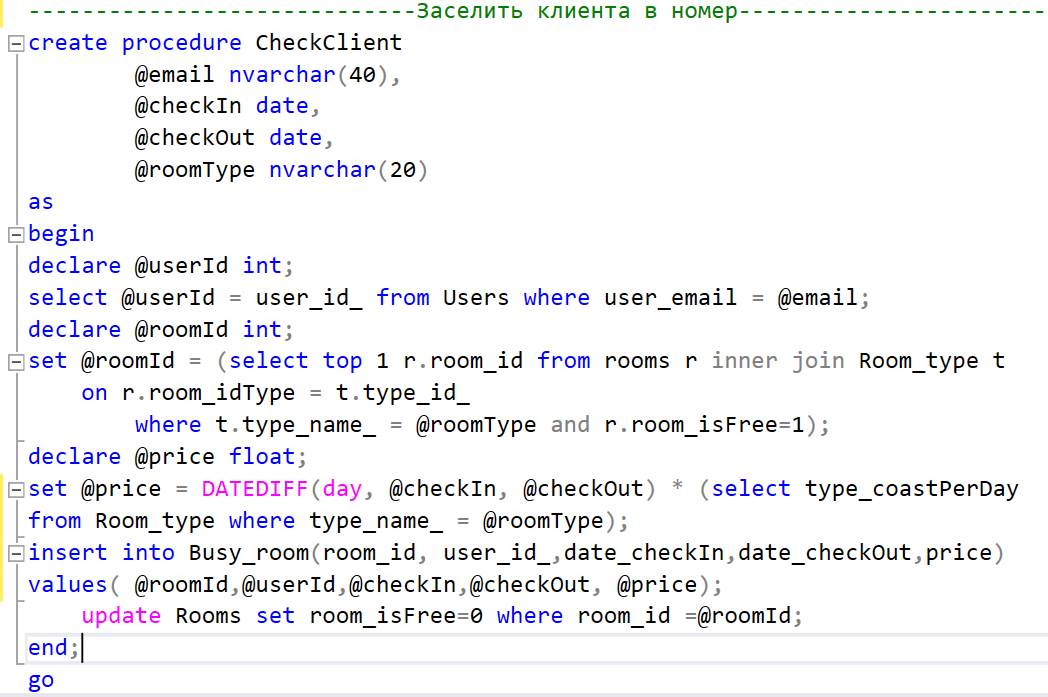


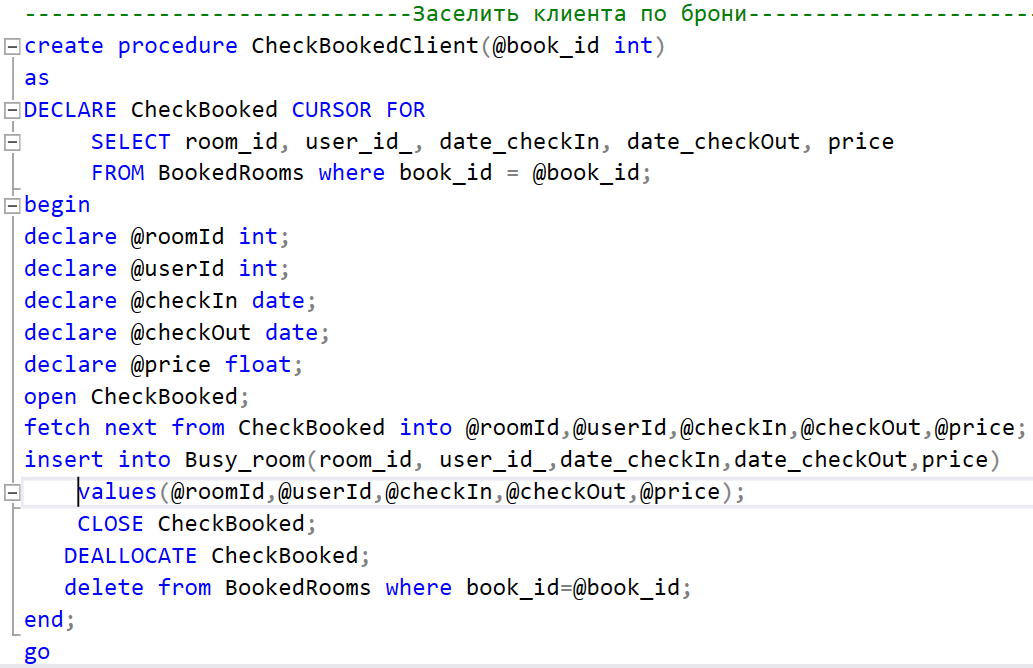
****

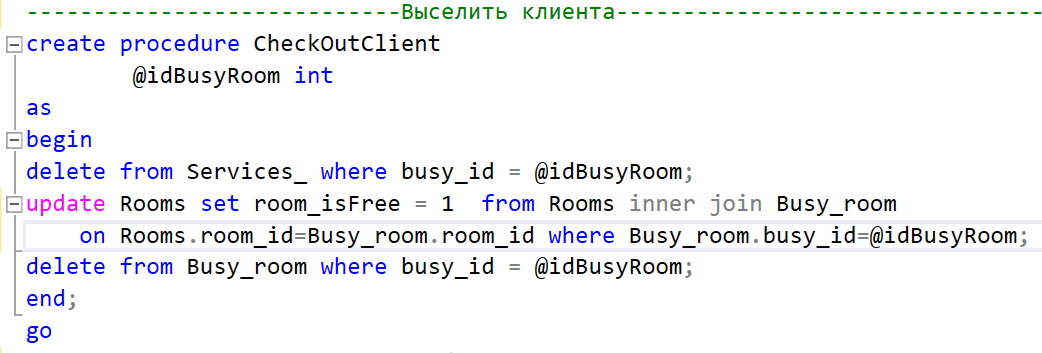
****

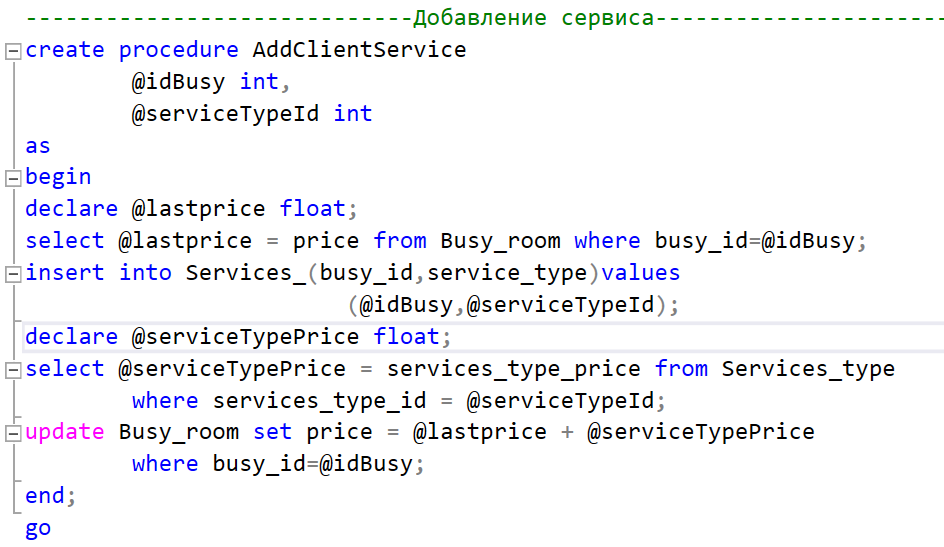
****

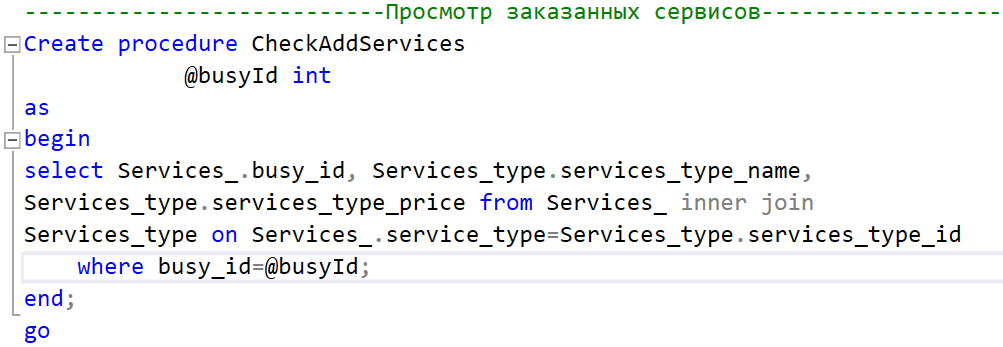
****

****

****

****

****

****