МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационные системы и технологии

Специальность 1–40 01 01 Программное обеспечение информационных

технологий

Специализация Программирование интернет-приложений

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

«База данных для программного средства «Ресторан азиатской кухни» с применением технологии шифрования и маскирования данных»

Выполнил студент Буданова Ксения Андреевна

(Ф.И.О.)

Руководитель проекта асс. Копыток Д. В.

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

Заведующий кафедрой к.т.н., доц. Смелов В. В.

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

Консультант: асс. Копыток Д. В.

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

Нормоконтролер: асс. Копыток Д. В.

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

Курсовой проект защищен с оценкой

Оглавление

[**Реферат 3**](#_Toc121386904)

[**Abstract 4**](#_Toc121386905)

[**Введение 5**](#_Toc121386906)

[**1. Постановка задачи 6**](#_Toc121386907)

[**1.1. Описание используемых технологий 6**](#_Toc121386908)

[**2. Проектирование модели базы данных 7**](#_Toc121386909)

[**3. Разработка необходимых объектов 8**](#_Toc121386910)

[**3.1. Таблицы 8**](#_Toc121386911)

[**3.2. Процедуры 10**](#_Toc121386912)

[**3.3. Функции 11**](#_Toc121386913)

[**3.4. Представления 12**](#_Toc121386914)

[**3.5. Индексы 12**](#_Toc121386915)

[**3.6. Логины 12**](#_Toc121386916)

[**3.7. Роли 13**](#_Toc121386917)

[**3.8. Пользователи 13**](#_Toc121386918)

[**4. Описание процедур импорта и экспорта данных 14**](#_Toc121386919)

[**4.1. Процедура импорта данных из XML-файла 14**](#_Toc121386920)

[**4.2. Процедура экспорта данных в формате XML 14**](#_Toc121386921)

[**5. Описание технологии 15**](#_Toc121386922)

[**6. Тестирование 17**](#_Toc121386923)

[**6.1. Тестирование производительности базы данных 17**](#_Toc121386924)

[**Заключение 19**](#_Toc121386925)

[**Список литературных источников 20**](#_Toc121386926)

[**Приложение A 21**](#_Toc121386927)

[**Приложение Б 22**](#_Toc121386928)

# Реферат

Пояснительная записка курсового проекта содержит 35 страниц пояснительной записки, 8 иллюстраций, 3 источника литературы, 2 приложения.

MICROSOFT SQL SERVER, DYNAMIC DATA MASKING, DATA ENCRYPTION.

Цель курсового проекта: разработка базы данных для ресторана с применением технологии шифрования и маскирования данных.

Первая глава описывает задачи, которые нужно выполнить в рамках курсового проекта.

Вторая глава содержит краткое описание модели базы данных.

Третья глава описывает процесс разработки необходимых для взаимодействия с базой данных технологий.

Четвёртая глава описывает процедуры импорта и экспорта данных.

Пятая глава описывает изученные и использованные в процессе разработки технологии.

Шестая глава описывает тестирование производительности.

В заключении приведены результаты проделанной работы.

# Abstract

The course project explanatory note consists of 35 pages of explanatory note, 8 illustrations, 3 literature sources, 2 appendixes.

MICROSOFT SQL SERVER, DYNAMIC DATA MASKING, DATA ENCRYPTION.

The purpose of course project: to design database for restaurant using encryption and data masking technology.

The first chapter describes the tasks that need to be completed within the framework of the course project.

The second chapter contains a brief description of the database model.

The third chapter describes the process of developing the technologies that are necessary for interacting with the database.

The fourth chapter describes the procedures for importing and exporting data.

The fifth chapter describes the technologies studied and used in the development process.

The sixth chapter describes performance testing.

The results of the work are presented in conclusion.

# Введение

На сегодняшний день огромное количество информации хранится в цифровом виде и для хранения такой информации отлично подходят различные базы данных. Использование баз данных упрощает управление и хранение данных. Одним из примеров успешного внедрения баз данных является использование их различными приложениями.

Данный проект представляет собой базу данных ресторана. 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.

В соответствии с заданием курсового проекта для проектирования базы данных ресторана используется система управления базами данных Microsoft SQL Server и технология шифрования и маскирования данных.

# Постановка задачи

Целью данного курсового проекта является проектирование базы данных для программного средства ресторана с применением технологии шифрования и маскирования данных. 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 меню;
* просматривать меню по типу блюда;
* просматривать доступные к брони столы;
* бронировать стол;
* оставлять и просматривать отзывы;
* добавлять новые блюда (функция администратора);
* добавлять новые типы блюд (функция администратора);
* добавлять новые столы (функция администратора);
* просматривать список пользователей (функция администратора);
* подтверждать бронь (функция администратора).

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

* роли;
* логины;
* пользователи;
* таблицы;
* представления;
* индексы;
* хранимые процедуры;
* функции.
  1. **Описание используемых технологий**

СУБД существует огромное множество: Oracle, MS SQL Server, NoSQL, MySql и так далее. Для организации работы с базой данных в данной работе было решено использовать одну из наиболее популярных СУБД «Microsoft SQL Server», так как она предоставляет необходимые возможности оперирования объектами базы данных.

Также в процессе выбора технологии была рассмотрена технология шифрования и маскирования данных.

# Проектирование модели базы данных

Для реализации поставленной задачи была создана база данных Oishi. Диаграмма структуры полученной базы данных, разработанной в СУБД «Microsoft SQL Server», представлена в приложении А.

Для базы данных было разработано 6 таблиц, которые связаны друг с другом внешними ключами.

Таблица USERS хранит данные о пользователях.

Таблица DISHTYPES содержит информацию о типах блюд для более удобного поиска блюд.

Таблица MENU хранит информацию о блюдах.

Таблица RESTABLES хранит информацию о доступных к бронированию столов.

Таблица ORDERS хранит информацию о заказах.

Таблица REVIEWS оставленные пользователями комментарии.

Создание перечисленных таблиц с установлением внешних связей описано в главе 3.

# Разработка необходимых объектов

## Таблицы

Таблицы являются неотъемлемой частью любой реляционной базы данных. Краткая характеристика каждой из таблиц была предоставлена в разделе 2, а код их создания можно увидеть в Приложении Б. Ниже мы рассмотрим каждую таблицу подробнее.

Таблица USERS состоит из четырёх столбцов:

* UserID – хранит идентификатор пользователя;
* UserLogin – логин пользователя;
* UserPassword – пароль пользователя;
* UserAffiliation – роль пользователя;

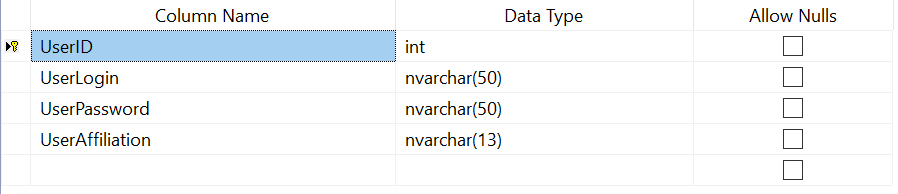


Рисунок 3.1.1 – Описание таблицы USERS

Таблица DISHTYPES состоит из двух столбцов:

* TypeId – хранит идентификатор типа блюда;
* TypeName – хранит название типа блюда;

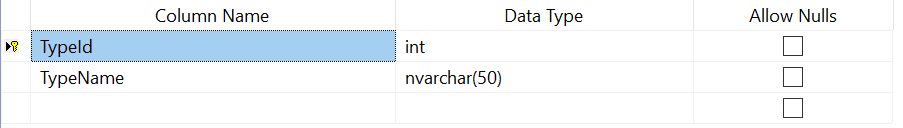


Рисунок 3.1.2 – Описание таблицы DISHTYPES

Таблица MENU состоит из шести столбцов:

* DishId – хранит идентификатор блюда;
* DishName – хранит название блюда;
* DishPrice – хранит стоимость блюда;
* DishWeight – хранит вес блюда;
* DishDescription – хранит описание блюда;
* DishType – хранит тип блюда, внешний ключ к таблице DISHTYPES;

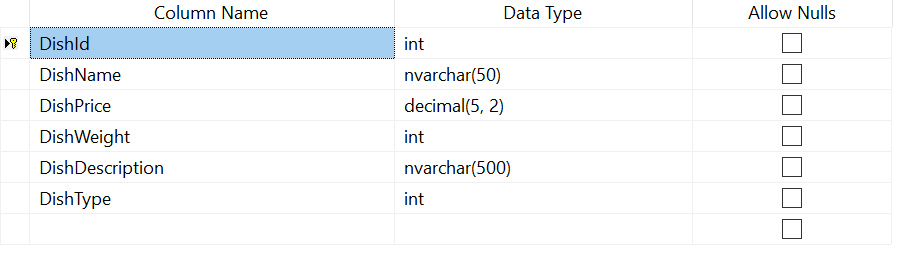


Рисунок 3.1.3 – Описание таблицы MENU

Таблица RESTABLES состоит из трёх столбцов:

* TableID – хранит идентификатор стола;
* OrdinalNumber – хранит порядковый номер стола;
* TableDescription – хранит описание стола;

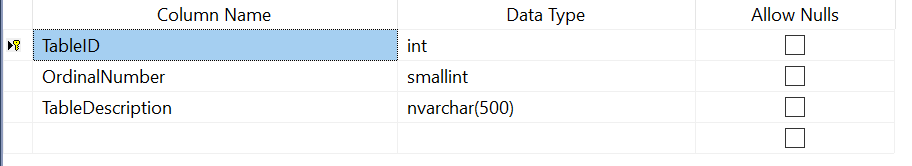


Рисунок 3.1.4 – Описание таблицы RESTABLES

Таблица REVIEWS состоит из трёх столбцов:

* ReviewID – хранит идентификатор отзыва;
* UserRID – хранит идентификатор пользователя, внешний ключ к таблице USERS;
* ReviewText – хранит текст отзыва;

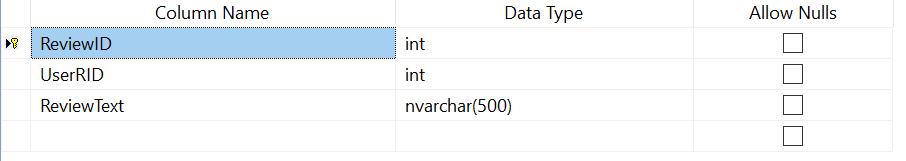


Рисунок 3.1.5 – Описание таблицы REVIEWS

Таблица ORDERS состоит из семи столбцов:

* OrderID – хранит идентификатор заказа;
* UserOID – хранит идентификатор пользователя, внешний ключ к таблице USERS;
* BookingDatetime – хранит дату и время брони;
* TableOID – хранит идентификатор стола, внешний ключ к таблице RESTABLES;
* UserEmail – хранит почту;
* Wishes – хранит текст пожеланий;
* OrderStatus – хранит состояние заказа;

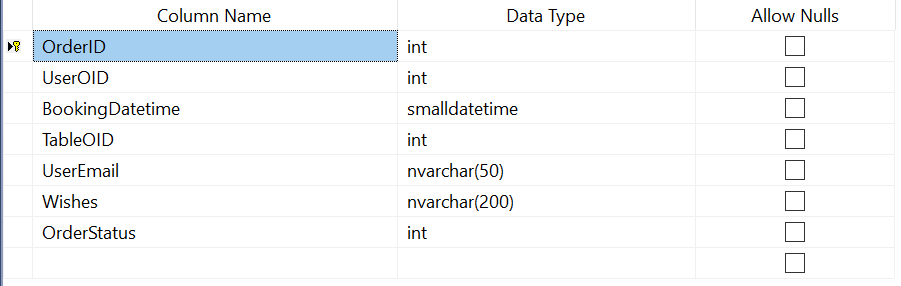


Рисунок 3.1.6 – Описание таблицы ORDERS

## Процедуры

Всего было разработано 40 процедур для работы с данными. Краткое описание процедур описано далее.

Пример создания хранимой процедуры InsertIntoMenu, которая добавляет блюда представлена в приложении Б.

Данная процедура принимает входной параметр name (название блюда), price (цена блюда), weight (вес блюда) и typename (тип блюда), desc (описание блюда).

AddAdmin – процедура для добавления администратора;

AddReview – процедура для добавления отзыва;

Authorize – процедура для авторизации;

ConfirmOrder – процедура для подтверждения заказа;

Decrypt – процедура дешифрования паролей в таблице USERS;

DeleteAllUsers – процедура для удаления всех пользователей;

DeleteDish – процедура для удаления блюда;

DeleteDishType – процедура для удаления типа блюда;

DeleteOrder – процедура для удаления заказа;

DeleteReview – процедура для удаления отзыва;

DeleteTable – процедура для удаления стола;

DeleteUser – процедура для удаления пользователя;

Encrypt – процедура шифрования паролей в таблице USERS;

EncryptByLogin – процедура для шифрования пароля определённого пользователя в таблице USERS;

ExportUsersToXML – процедура для экспорта данных в XML;

GetAllDishes – процедура выборки всех блюд;

GetAllDishTypes – процедура выборки всех типов блюд;

GetAllOrders – процедура выборки всех заказов;

GetAllRestables – процедура выборки всех столов;

GetAllReviews – процедура выборки всех отзывов;

GetAllUnconfirmedOrders– процедура выборки неподтверждённых заказов;

GetAllUsers – процедура выборки всех пользователей;

GetDishesByCategory – процедура выборки блюд по типу;

ImportUsersFromXML– процедура для импорта данных из XML;

InsertDishtypes – процедура для генерации 100000 строк в таблицу DISHTYPES;

InsertIntoDishtypes – процедура для добавления типа блюда;

InsertIntoMenu – процедура для добавления блюда;

InsertIntoRestables – процедура для добавления стола;

InsertMenu – процедура для генерации 100000 строк в таблицу MENU;

InsertOrders – процедура для генерации 100000 строк в таблицу ORDERS;

InsertRestables – процедура для генерации 100000 строк в таблицу RESTABLES;

InsertReviews – процедура для генерации 100000 строк в таблицу REVIEWS;

InsertUsers – процедура для генерации 100000 строк в таблицу USERS;

MakeOrder – процедура для оформления заказа;

Registrate – процедура для регистрации нового пользователя;

UpdateDish – процедура для обновления информации о блюде;

UpdateDishtype – процедура для обновления информации о типе блюда;

UpdateTable – процедура для обновления информации о столе;

UpdateUser – процедура для обновления информации о пользователе;

usp\_GetErrorInfo – процедура для получения информации об ошибках;

Примеры создания процедур представлены в приложении Б.

## Функции

Всего была разработана 1 функция для работы с данными. Пример реализации функции RandomString, которая генерирует рандомные строки, представлен в листинге 3.3.1.

CREATE OR ALTER FUNCTION RandomString(

@min\_length int,

@max\_length int,

@charPool nvarchar(50)

)

RETURNS NVARCHAR(12)

AS

BEGIN

DECLARE @Length int,@PoolLength int, @LoopCount int, @RandomString nvarchar(12)

SET @Length = floor((select vR from vRand) \* (@max\_length - @min\_length) + @min\_length);

SET @PoolLength = LEN(@charPool);

SET @LoopCount = 0;

SET @RandomString = '';

WHILE(@LoopCount < @Length)

BEGIN

SELECT @RandomString = @RandomString + SUBSTRING(@charPool, CONVERT(int, floor((select vR from vRand) \* @PoolLength)),1)

SELECT @LoopCount = @LoopCount +1

END;

RETURN @RandomString;

END;

Листинг 3.3.1 – Функция RandomString

## Представления

Всего было разработано 1 представление для работы с данными. Пример реализации представления VORDERS, которое выводит необработанные заказы, представлен в листинге 3.4.1.

CREATE OR ALTER VIEW VORDERS

AS

SELECT OrderID[Order ID], UserOID[User ID], BookingDatetime[Booking time], TableOID[Table ID], Wishes

FROM ORDERS

WHERE OrderStatus = 0

GROUP BY OrderID, UserOID, BookingDatetime, TableOID, Wishes;

Листинг 3.4.1 – Представление VORDERS

## Индексы

Всего было разработано 3 индекса для оптимизации таблиц. Их реализация представлена в листинге 3.5.1.

CREATE INDEX idx\_dishtypes\_typename ON DISHTYPES(TypeName);

CREATE INDEX idx\_restables\_ordinalnumber ON RESTABLES(OrdinalNumber);

CREATE INDEX idx\_menu\_tpnmprwt ON MENU(DishType) INCLUDE (DishName, DishPrice, DishWeight, DishDescription);

Листинг 3.5.1 – Индексы

## Логины

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

* OishiAdmin;
* OishiClient.

Скрипты представлены в листинге 3.6.1.

CREATE LOGIN OishiAdmin WITH PASSWORD = 'admin';

CREATE LOGIN OishiClient WITH PASSWORD = '1234';

Листинг 3.6.1 – Логины

## Роли

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

* OishiAdminR;
* OishiClientR.

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

CREATE ROLE OishiAdminR;

GO

GRANT ALTER, REFERENCES, CONTROL, VIEW DEFINITION ON SYMMETRIC KEY::MySymKey TO OishiAdminR WITH GRANT OPTION;

GRANT ALTER, REFERENCES, CONTROL, VIEW DEFINITION ON CERTIFICATE::MyCert TO OishiAdminR WITH GRANT OPTION;

GRANT UNMASK to OishiAdminR;

GRANT EXECUTE ON Encrypt TO OishiAdminR WITH GRANT OPTION;

GRANT EXECUTE ON EncryptByLogin TO OishiAdminR WITH GRANT OPTION;

GRANT EXECUTE ON Decrypt TO OishiAdminR WITH GRANT OPTION;

Листинг 3.7.1 – Создание роли администратора

CREATE ROLE OishiClientR;

GO

GRANT EXECUTE ON Registrate TO OishiClientR WITH GRANT OPTION;

GRANT EXECUTE ON Authorize TO OishiClientR WITH GRANT OPTION;

GRANT EXECUTE ON AddReview TO OishiClientR WITH GRANT OPTION;

GRANT EXECUTE ON MakeOrder TO OishiClientR WITH GRANT OPTION;

Листинг 3.7.2 – Создание роли клиента

## Пользователи

Для реализации вышеупомянутый ролей были созданы два пользователя:

* OishiAdminU;
* OishiClientU.

Скрипты представлены в листинге 3.8.1.

CREATE USER OishiAdminU FOR LOGIN OishiAdmin;

CREATE USER OishiClientU FOR LOGIN OishiClient;

Листинг 3.8.1 – Создание пользователей

После чего идёт привязка ролей к созданным пользователям. Скрипты представлены в листинге 3.8.2.

exec sp\_addrolemember 'OishiAdminR', 'OishiAdminU';

exec sp\_addrolemember 'OishiClientR', 'OishiClientU';

Листинг 3.8.2 – Привязка ролей к пользователям

# Описание процедур импорта и экспорта данных

При всей своей отлаженности и очевидности, классический способ хранения и представления объектов развитой структуры имеет и вполне определенные недостатки и может вызывать проблемы, с которыми сталкивался любой разработчик, пытавшийся реализовать таким способом достаточно сложную систему. В некоторых ситуациях, решить эти проблемы позволяет хранение объекта в виде XML.

## Процедура импорта данных из XML-файла

Для импорта используется стандартная функция, входящая в набор OLE DB – OPENROWSET, в которую передаются параметры о типе импортируемых данных и пути файла, где они находятся. С помощью процедуры ImportUsersFromXML администратор может добавить новых пользователей в базу данных из XML-файла. Листинг данной процедуры можно увидеть в приложении Б

## Процедура экспорта данных в формате XML

Процедура ExportUsersToXML по выходному параметру возвращает данные о всех пользователях в формате XML, которые потом отображаются в приложении. Её реализацию можно увидеть в приложении Б.

# Описание технологии

Чтобы замаскировать данные столбца таблицы, необходимо указать правило, с помощью которого данные будут замаскированы. В своей базе данных я применила методы маскирования (partial(prefix, padding, suffix)), email(), default(). Первый метод может раскрывать несколько первых (prefix) и последних (suffix) букв и добавляет пользовательскую строку заполнения (padding) в середине. Второй же метод необходим для маскирования почтовых адресов(xxxx@xxxx). default() предоставляет полное маскирование данных.

Пример реализации технологии маскирования данных в базе данных представлен в листинге 5.1.

CREATE TABLE USERS(

UserID int MASKED WITH (FUNCTION = 'default()') identity(1,1) primary key,

UserLogin nvarchar(50) MASKED WITH (FUNCTION = 'partial(1,"XXXXXXX", 1)') not null unique,

UserPassword nvarchar(50) not null,

UserAffiliation nvarchar(13) MASKED WITH (FUNCTION = 'default()') default 'client'

)

Листинг 5.1 – Маскирование столбцов UserID, UserLogin, UserAffiliation таблицы USERS

Шифрование данных в моей базе данных реализовано с помощью симметричного ключа на уровне ячеек. Метод шифрования с помощью симметричного ключа достаточно надежен, удовлетворяет большинству требований к безопасности данных и обеспечивает достаточное быстродействие. Для шифрования и дешифрования данных используется один ключ.

В первую очередь необходимо создать главный ключ DMK (Database master key) базы данных, шифруемый с помощью парольной фразы Ksusha1234 (листинг 5.2.).

CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'Ksusha1234';

Листинг 5.2 – Создание главного ключа DMK

Как было сказано ранее, данные шифруются с использованием симметричного ключа, который будет зашифрован с помощью сертификата. Для этого необходимо создать сертификат, создать симметричный ключ и зашифровать симметричный ключ с помощью только что созданного сертификата. Программный код представлен в листинге 5.3.

CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'Ksusha1234';

CREATE CERTIFICATE MyCert WITH SUBJECT = 'Cert'

CREATE SYMMETRIC KEY MySymKey WITH ALGORITHM = AES\_256 ENCRYPTION BY CERTIFICATE MyCert;

Листинг 5.3 – Создание ассиметричного и симметричного ключа

Теперь можно приступить к шифрованию данных. Для этого необходимо сначала открыть симметричный ключ, только что созданный с помощью команды OPEN SYMMETRIC KEY, за которой следует имя симметричного ключа. Затем указать, что нужно расшифровать его с использованием заданного асимметричного ключа. Программный код представлен в листинге 5.4.

OPEN SYMMETRIC KEY MySymKey

DECRYPTION BY CERTIFICATE MyCert;

Листинг 5.4 – Открытие симметричного ключа

SQL Server располагает встроенными функциями для шифрования и дешифрования на уровне ячеек. Функция шифрования, которая использует симметричный ключ: ENCRYPTBYKEY.

С помощью функции DECRYPTBYKEY можно просмотреть зашифрованные данные.

Шифрование и дешифрование данных симметричным ключом показано в листинге 5.5.

CREATE OR ALTER PROCEDURE Encrypt

AS BEGIN

OPEN SYMMETRIC KEY MySymKey

DECRYPTION BY CERTIFICATE MyCert;

UPDATE dbo.USERS

SET UserPassword = EncryptByKey (Key\_GUID('MySymKey'), UserPassword)

FROM dbo.USERS;

CLOSE SYMMETRIC KEY MySymKey;

END;

CREATE OR ALTER PROCEDURE Decrypt

AS BEGIN

OPEN SYMMETRIC KEY MySymKey

DECRYPTION BY CERTIFICATE MyCert;

SELECT UserLogin, UserAffiliation, UserPassword AS 'Encrypted password',

CONVERT(nvarchar, DecryptByKey(UserPassword)) AS 'Decrypted password'

FROM dbo.USERS;

END;

Листинг 5.5 – Шифрование и дешифрование данных

# Тестирование

## Тестирование производительности базы данных

Для тестирования производительности была взята за основу таблица DISHTYPES, которую мы заполнили 1922264 строками. После этого был применён SELECT-запрос к таблице DISHTYPES на столбец DishType и при помощи стандартных средств IDE MS SQL Server Management Studio оценена цена выборки к таблице. Результат данной оценки запроса приведён на рисунке 6.1.

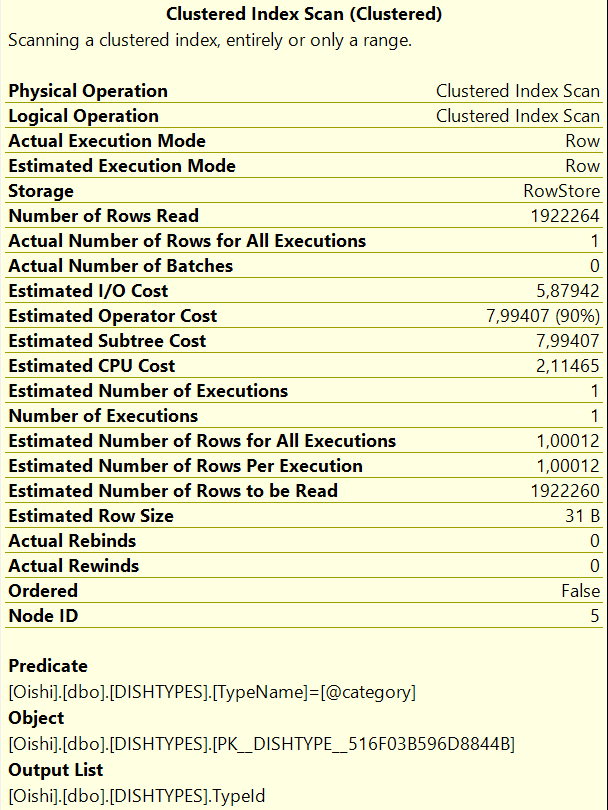


Рисунок 6.1 – Оценка запроса к таблице DISHTYPES с кластеризованным индексом

После проведения первоначальной оценки был построен некластеризованный индекс к таблице DISHTYPES, создание которого можно увидеть в приложении Б, и проведена оценка такого же SELECT-запроса к таблице LOCATIONS. Результаты, полученные во время оценки, представлены на рисунке 6.2.

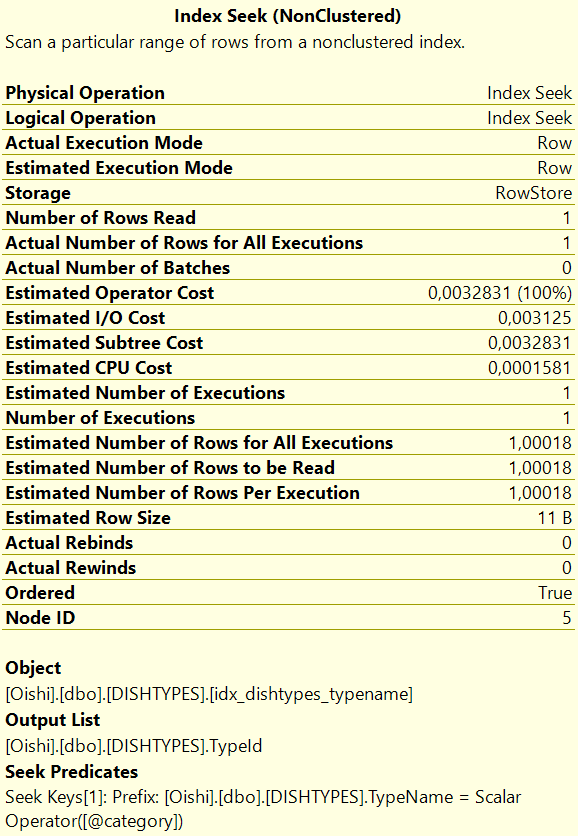


Рисунок 6.2 – Оценка запроса к таблице с построенным некластеризованным индексом

По результатам проведённых оценок, можно сделать вывод, что после создания некластеризованного индекса:

– стоимость по параметру “Предполагаемая стоимость операций ввода-вывода” стала в 1881 раз меньше;

– стоимость по параметру “Предполагаемая стоимость оператора” снизилась почти в 2435 раз;

– стоимость по параметру “Предполагаемая стоимость поддерева” уменьшилась почти в 2435 раз.

Таким образом, постройка индекса к таблице была более чем оправдана, так как мы получили прирост производительности в 1800 и более раза.

# Заключение

Перед началом разработки базы данных было произведено определение функциональных возможностей моей базы данных.

В процессе решения поставленной задачи была достигнута поставленная цель по созданию базы данных «Ресторан».

В процессе выполнения курсовой работы была спроектирована база данных для аукционной площадки. База данных была разработана с помощью системы управления базами данных «Microsoft SQL Server 2018». Помимо этого, была изучена и применена при разработке приложения технология шифрования и маскирования данных.

При разработке базы данных было созданы следующие объекты:

– 6 таблиц с внешними связями;

– 40 хранимых процедур;

– 3 индекса;

При разработке выполнены следующие пункты:

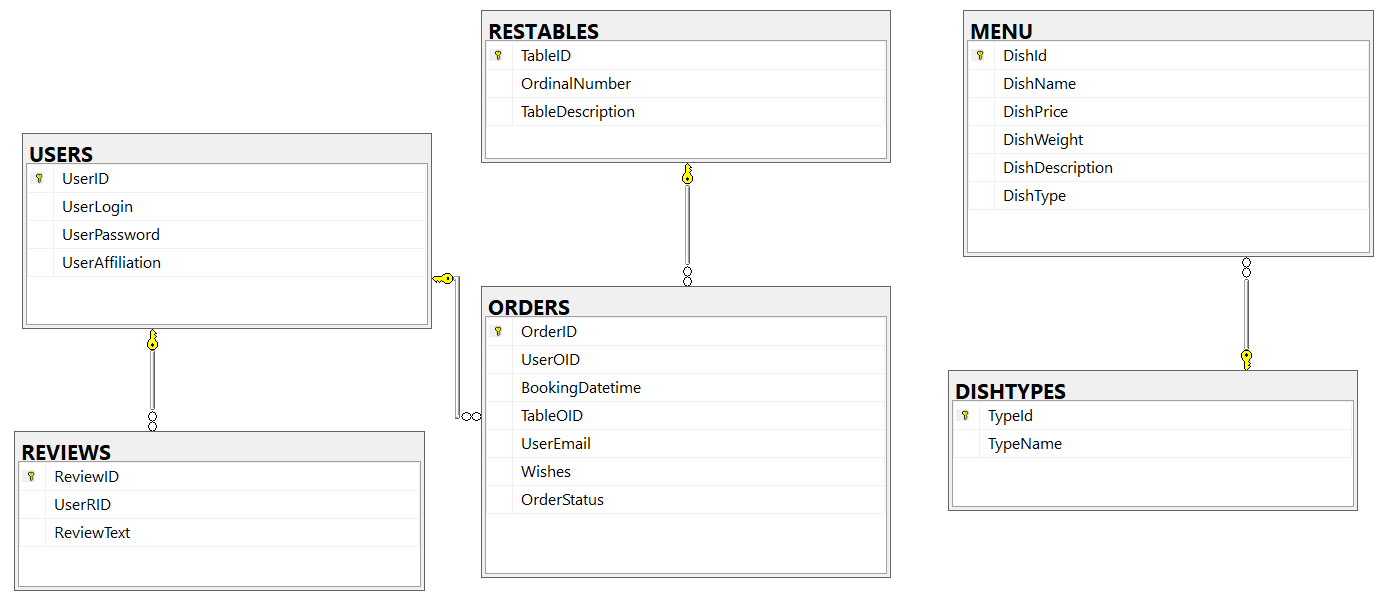
* Получение списка всех блюд.
* Получение списка блюд по категориям.
* Добавление отзывов.
* Добавление нового типа блюда.
* Добавление нового блюда.
* Добавление нового стола.
* Возможность оформлять заказы (бронь).
* Шифрование и маскирование данных.
* Просмотр неподтверждённых заказов.
* Подтверждение заказа администратором.
* Импорт и экспорт XML.

1. В соответствии с полученным результатом, можно сказать, что разработанная база данных функционирует верно, требования технического задания реализованы в полном объеме, поэтому цель курсового проекта можно считать достигнутой

**Список литературных источников**

1. Документация СУБД «Microsoft SQL Server» [Электронный ресурс] – Режим доступа: https://learn.microsoft.com/ru-ru/sql/?view=sql-server-ver16
2. Шифрование столбца данных – SQL Server [Электронный ресурс] – Режим доступа: https://learn.microsoft.com/ru-ru/sql/relational-databases/security/encryption/encrypt-a-column-of-data?view=sql-server-ver16
3. Динамическое маскирование данных – SQL Server [Электронный ресурс] – Режим доступа: https://learn.microsoft.com/ru-ru/sql/relational-databases/security/dynamic-data-masking?view=sql-server-ver16

# Приложение A



# Приложение Б

Листинг создания таблиц

CREATE TABLE USERS(

UserID int MASKED WITH (FUNCTION = 'default()') identity(1,1) primary key,

UserLogin nvarchar(50) MASKED WITH (FUNCTION = 'partial(1,"XXXXXXX", 1)') not null unique,

UserPassword nvarchar(50) not null,

UserAffiliation nvarchar(13) MASKED WITH (FUNCTION = 'default()') default 'client'

)

CREATE TABLE DISHTYPES(

TypeId int MASKED WITH (FUNCTION = 'default()') identity(1,1) primary key,

TypeName nvarchar(50) not null

)

CREATE TABLE MENU(

DishId int MASKED WITH (FUNCTION = 'default()') identity(1,1) primary key,

DishName nvarchar(50) not null,

DishPrice decimal(5,2) not null,

DishWeight int not null,

DishDescription nvarchar(500) not null,

DishType int foreign key references DISHTYPES(TypeId) ON DELETE CASCADE

)

CREATE TABLE RESTABLES(

TableID int MASKED WITH (FUNCTION = 'default()') identity(1,1) primary key,

OrdinalNumber smallint not null,

TableDescription nvarchar(500) not null

)

CREATE TABLE ORDERS(

OrderID int MASKED WITH (FUNCTION = 'default()') identity(1,1) primary key,

UserOID int MASKED WITH (FUNCTION ='default()') foreign key references USERS(UserID) ON DELETE CASCADE,

BookingDatetime smalldatetime not null,

TableOID int foreign key references RESTABLES(TableID) ON DELETE CASCADE,

UserEmail nvarchar(50) MASKED WITH (FUNCTION = 'email()') not null,

Wishes nvarchar(200),

OrderStatus int default 0

)

CREATE TABLE REVIEWS(

ReviewID int MASKED WITH (FUNCTION = 'default()') identity(1,1) primary key,

UserRID int foreign key references USERS(UserID) ON DELETE CASCADE,

ReviewText nvarchar(500) not null

)

Листинг процедуры InsertIntoMenu

CREATE OR ALTER PROCEDURE InsertIntoMenu

@name nvarchar(50),

@price decimal(5,2),

@weight int,

@typename nvarchar(50),

@desc nvarchar(500)

AS

BEGIN TRY

DECLARE @type int;

SET @type = (SELECT TypeId from DISHTYPES where DISHTYPES.TypeName = @typename);

IF (@type != 0)

BEGIN

IF EXISTS(SELECT TOP 1 \* FROM MENU WHERE MENU.DishName = @name)

print 'This unit exists in menu!';

ELSE IF (@name != '' AND @price != 0 AND @weight != 0 AND @desc != '' AND @type != 0)

BEGIN

INSERT INTO MENU(DishName, DishPrice, DishWeight, DishType, DishDescription)

VALUES(@name, @price, @weight, @type, @desc)

END;

ELSE print 'Fields cant be null.'

END;

ELSE print 'Invalid dish type'

END TRY

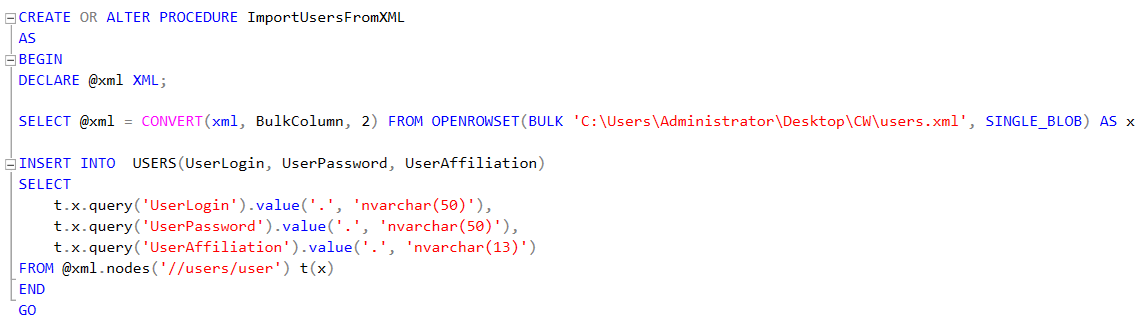
BEGIN CATCH

EXECUTE usp\_GetErrorInfo;

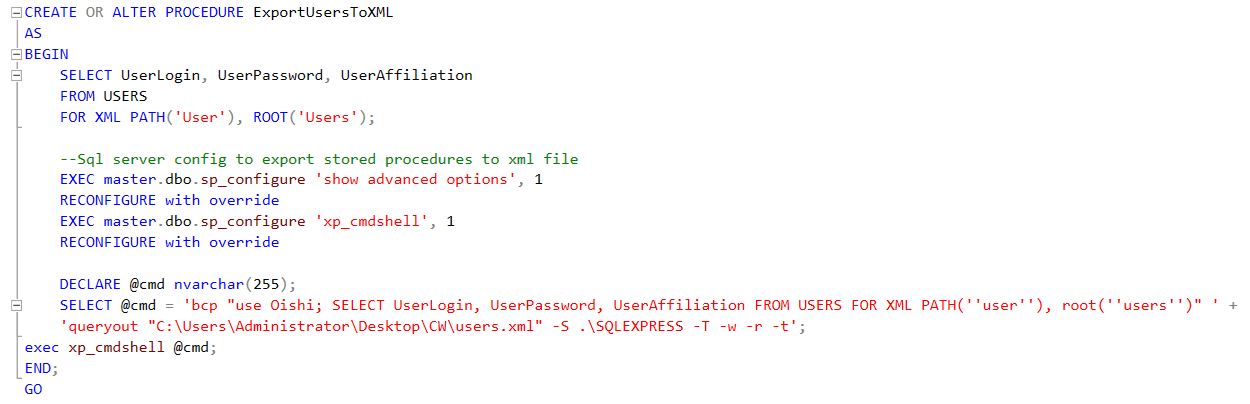
END CATCH;

GO

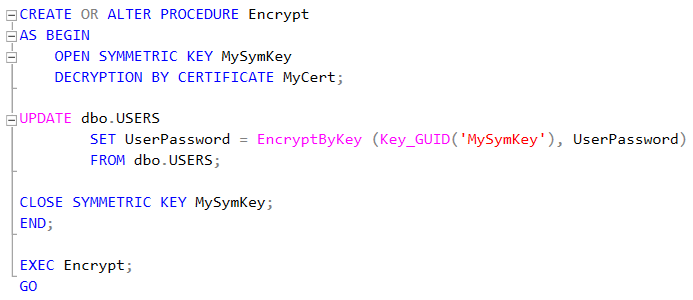
Процедура импорта данных из XML-файла



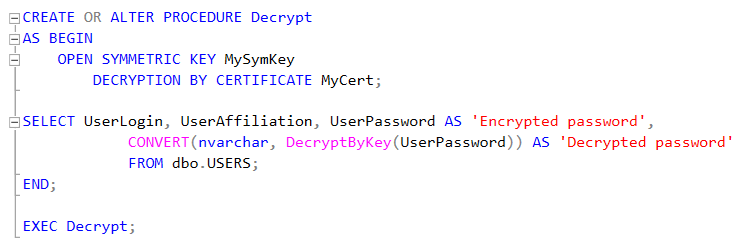
Процедура экспорта данных в формат XML



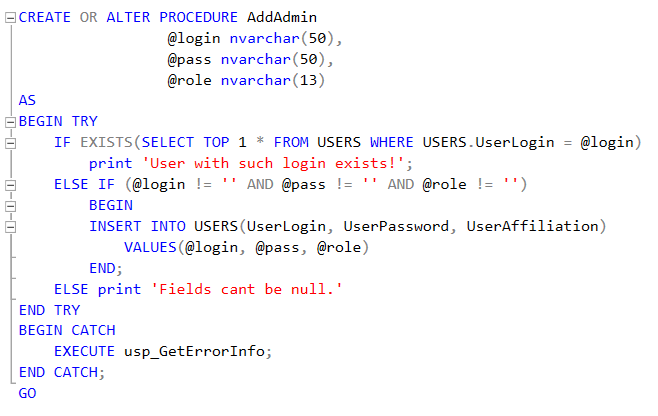
Процедура Encrypt, процедура применяющая технологию шифрования:



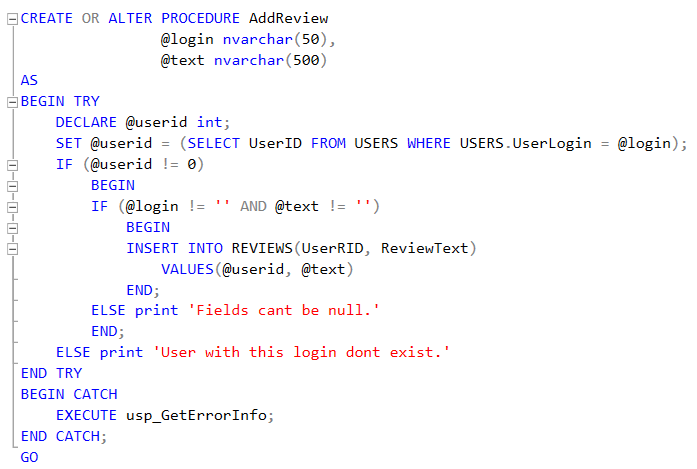
Процедура Decrypt для дешифрования данных:



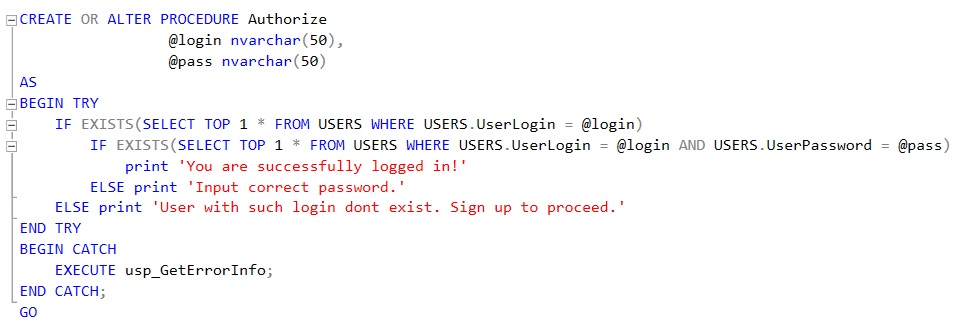
Процедура для добавления администратора



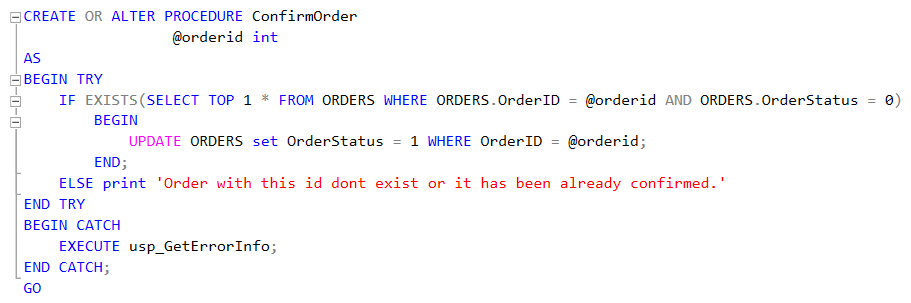
Процедура для добавления отзыва



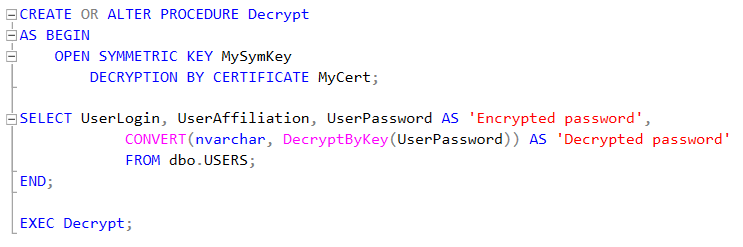
Процедура для авторизации



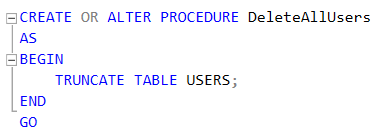
Процедура для подтверждения заказа



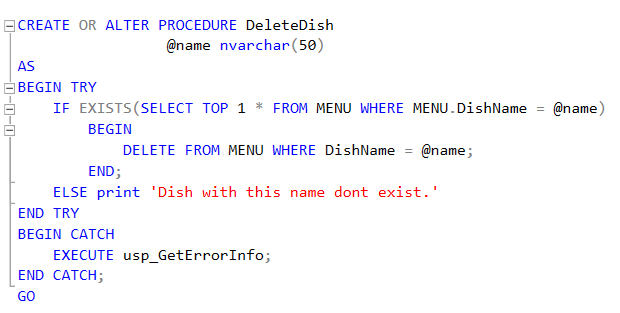
Процедура дешифрования паролей в таблице USERS



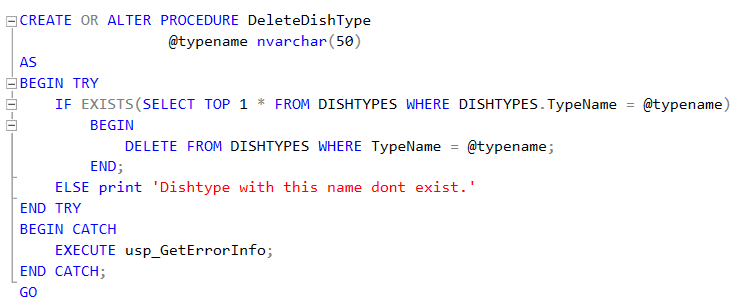
Процедура для удаления всех пользователей



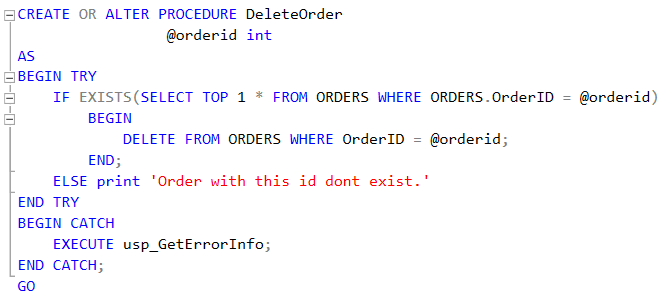
Процедура для удаления блюда



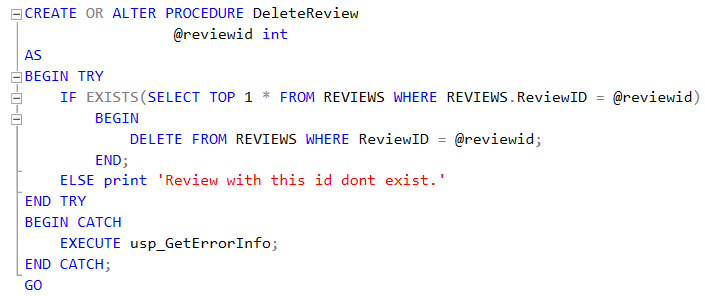
Процедура для удаления типа блюда



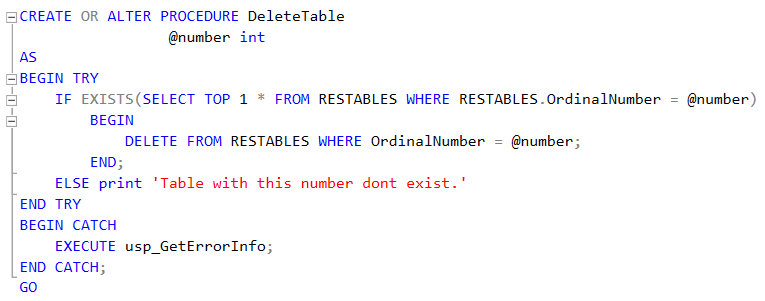
Процедура для удаления заказа



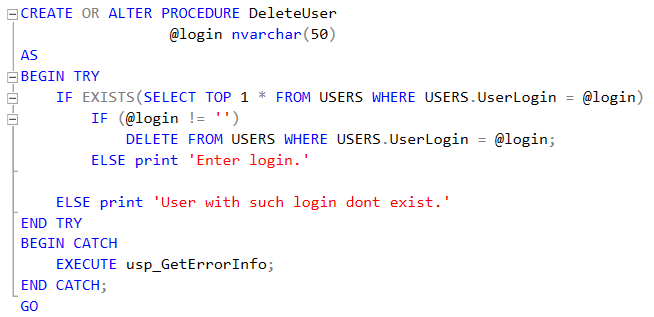
Процедура для удаления отзыва



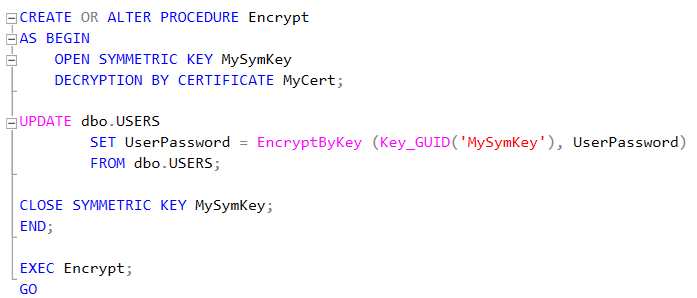
Процедура для удаления стола



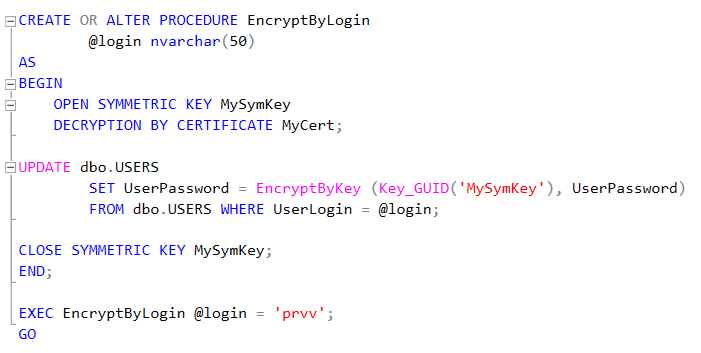
Процедура для удаления пользователя



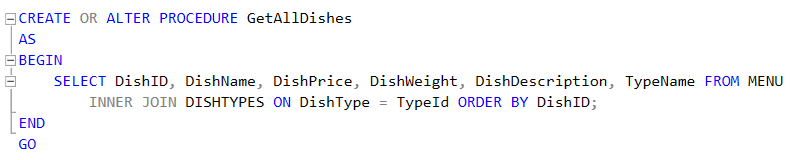
Процедура шифрования паролей в таблице USERS



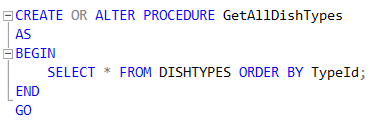
Процедура для шифрования пароля определённого пользователя в таблице USERS



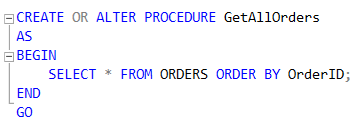
Процедура выборки всех блюд



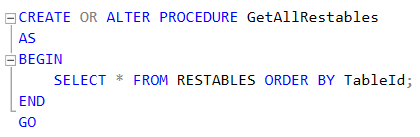
Процедура выборки всех типов блюд



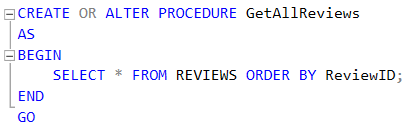
Процедура выборки всех заказов



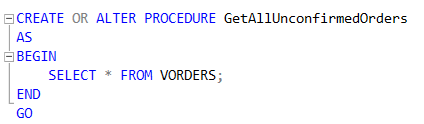
Процедура выборки всех столов



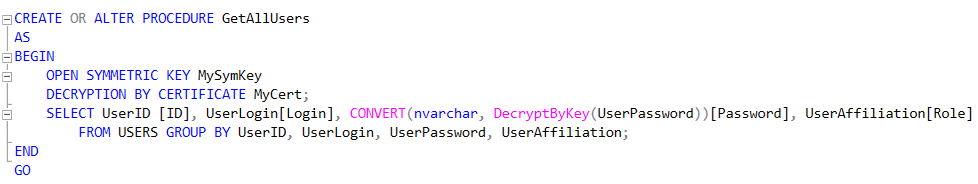
Процедура выборки всех отзывов



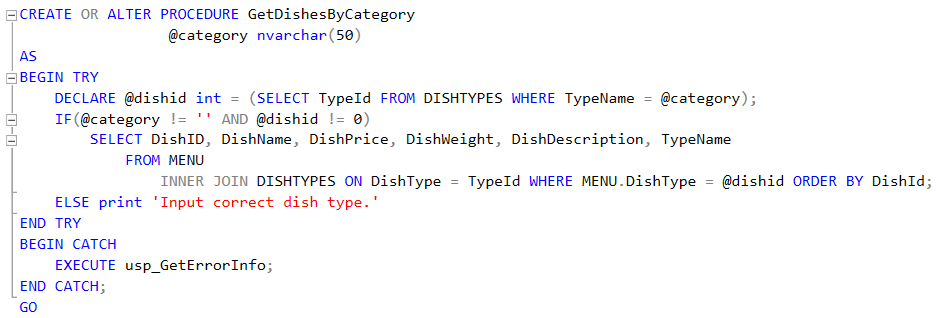
Процедура выборки неподтверждённых заказов



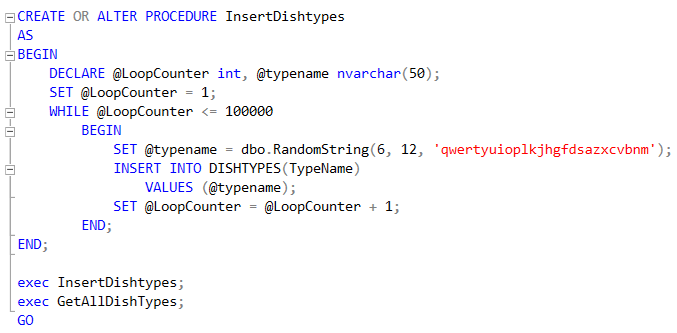
Процедура выборки всех пользователей



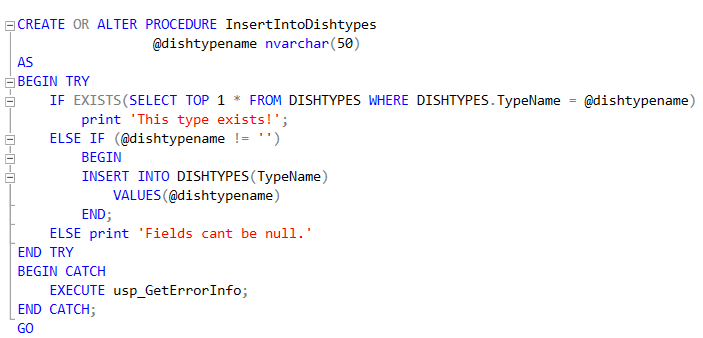
Процедура выборки блюд по типу



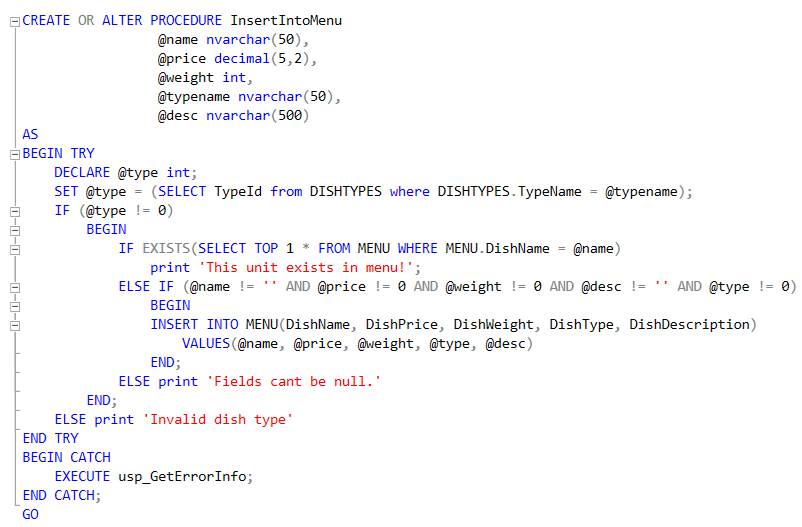
Процедура для генерации 100000 строк в таблицу DISHTYPES



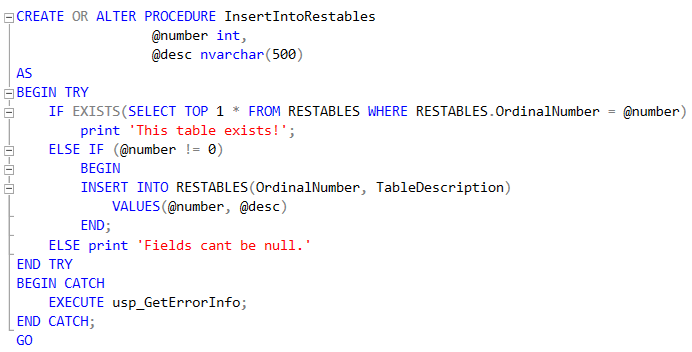
Процедура для добавления типа блюда



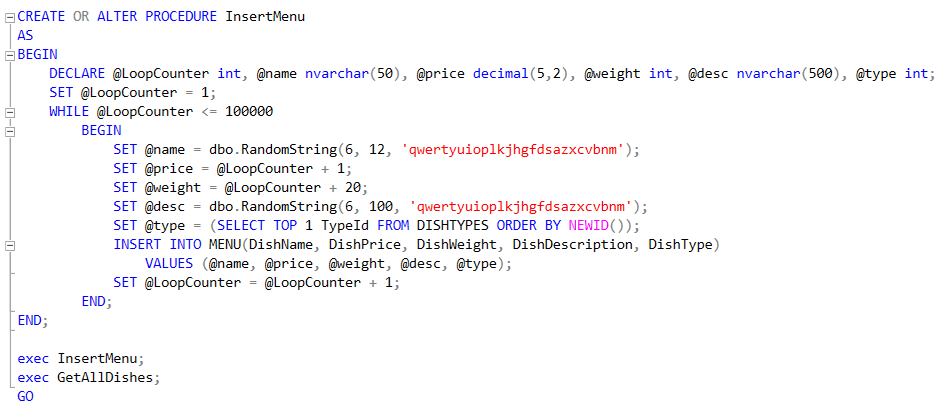
Процедура для добавления блюда



Процедура для добавления стола



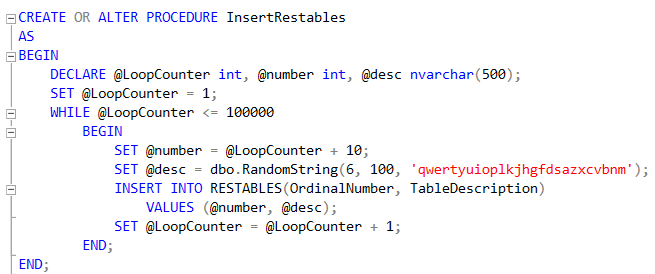
Процедура для генерации 100000 строк в таблицу MENU



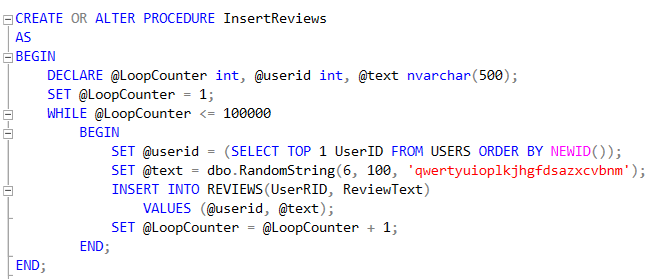
Процедура для генерации 100000 строк в таблицу ORDERS



Процедура для генерации 100000 строк в таблицу RESTABLES



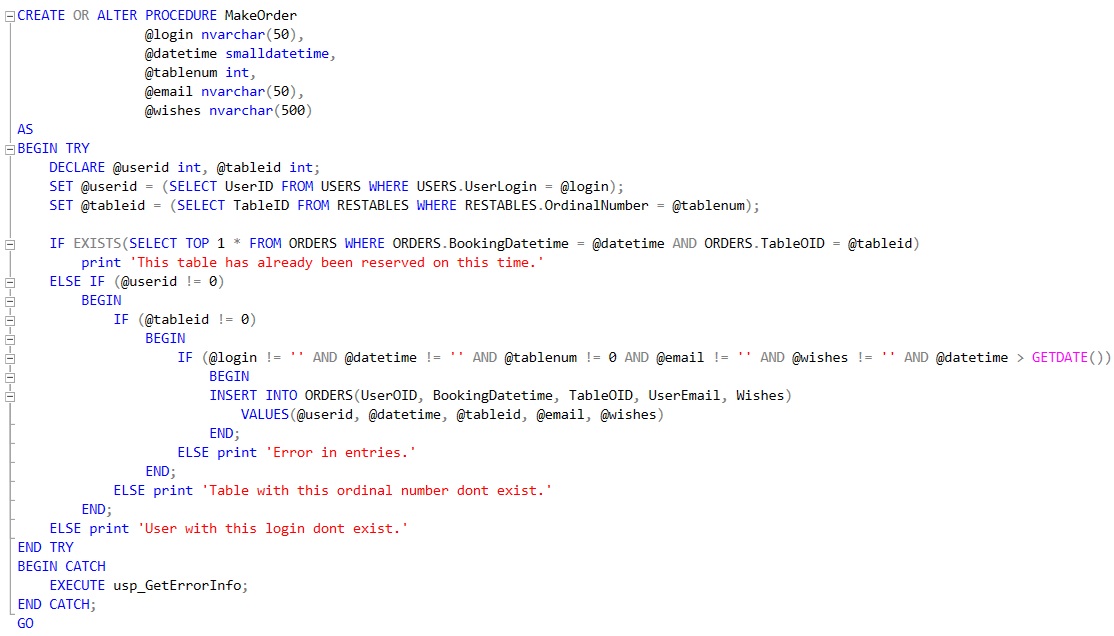
Процедура для генерации 100000 строк в таблицу REVIEWS



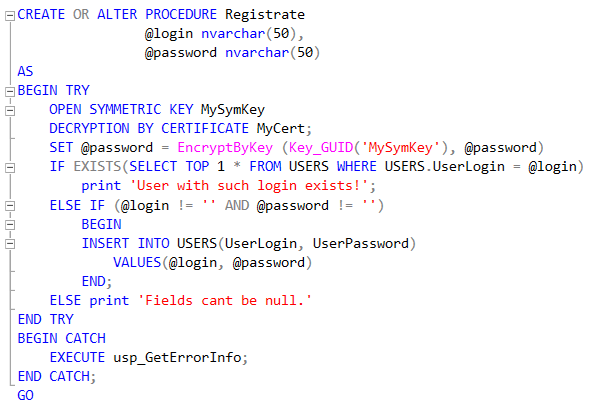
Процедура для генерации 100000 строк в таблицу USERS



Процедура для оформления заказа



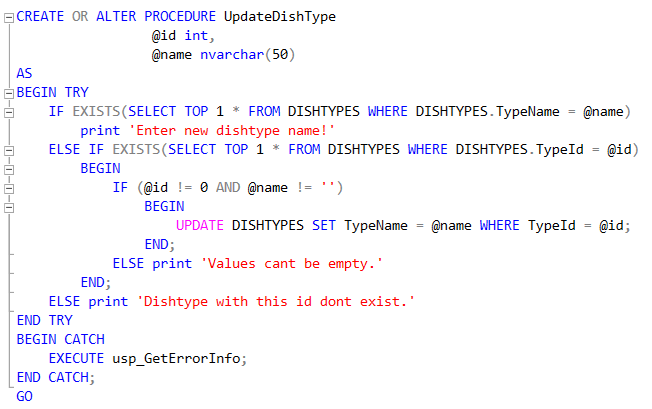
Процедура для регистрации нового пользователя



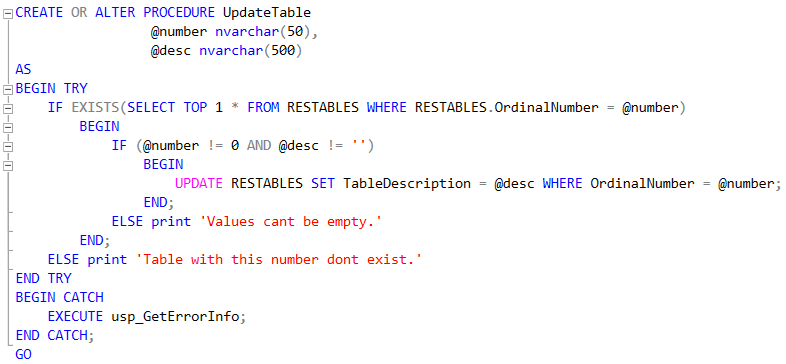
Процедура для обновления информации о блюде



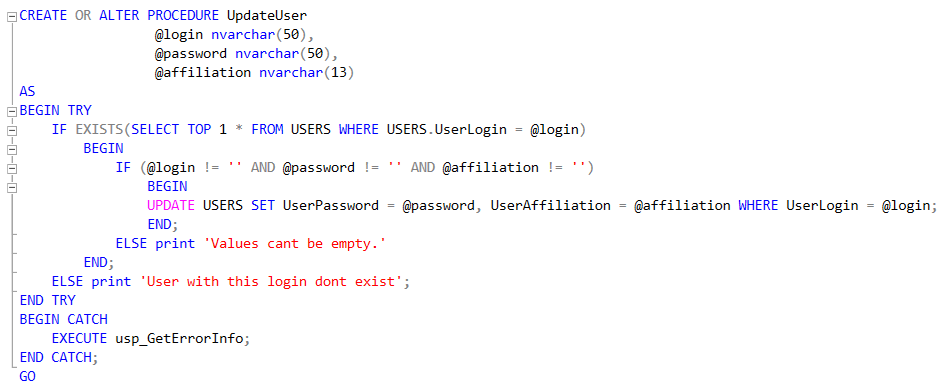
Процедура для обновления информации о типе блюда



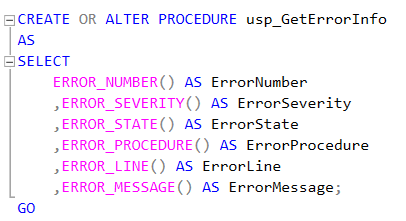
Процедура для обновления информации о столе



Процедура для обновления информации о пользователе



Процедура для получения информации об ошибках



Создание индексов:

CREATE INDEX idx\_dishtypes\_typename ON DISHTYPES(TypeName);

CREATE INDEX idx\_restables\_ordinalnumber ON RESTABLES(OrdinalNumber);

CREATE INDEX idx\_menu\_tpnmprwt ON MENU(DishType) INCLUDE (DishName, DishPrice, DishWeight, DishDescription);