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

Специализация 1–40 01 01 10 Программирование интернет-приложений

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

База данных «Поликлиника»

Выполнил студент Болвако Кирилл Сергеевич

(Ф.И.О.)

Руководитель проекта ассистент Нистюк О.А.

(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: ассистент Нистюк О.А

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

Нормоконтролер: ассистент Нистюк О.А

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

Курсовой проект защищен с оценкой

Минск 2020

**Реферат**

Пояснительная записка курсового проекта содержит 46 страниц пояснительной записки, 18 иллюстраций, 7 источников литературы, 2 приложения.

Основная цель курсового проекта: проектирование базы данных поликлиники.

В первой главе осуществляется постановка задачи.

Вторая глава посвящена процессу разработки модели базы данных.

В третьей главе приведено описание разработки объектов базы данных.

В четвертой главе описаны процедуры экспорта и импорта данных.

В пятой главе описано тестирование производительности.

В шестой главе посвящена технологии шифрования и маскирования данных.

В заключении приведены результаты проделанной работы.

**Abstract**

The explanatory note of the course project contains 46 pages of an explanatory note, 18 illustrations, 7 sources of literature, 2 applications.

The main goal of the course project: designing a database of a hospital.

In the first chapter, the problem is formulated.

The second chapter focuses on the process of developing a database model.

The third chapter describes the development of database objects.

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

The fifth chapter describes performance testing.

The sixth chapter is devoted to encryption and data masking.

In conclusion, the results of the work done are presented.

# **Содержание**

[Содержание 4](#_Toc58359011)

[Введение 5](#_Toc58359012)

[1. Постановка задачи 6](#_Toc58359013)

[2. Разработка модели базы данных 7](#_Toc58359014)

[3. Разработка необходимых объектов 9](#_Toc58359015)

[3.1. Таблицы 9](#_Toc58359016)

[3.2. Процедуры 11](#_Toc58359017)

[4. Описание процедур импорта и экспорта 14](#_Toc58359018)

[5. Тестирование производительности 15](#_Toc58359019)

[6. Описание технологии 18](#_Toc58359020)

[6.1 Технология «Шифрование» 18](#_Toc58359021)

[6.2 Применение технологии «Шифрования» 19](#_Toc58359022)

[6.3 Технология «Маскирование» 20](#_Toc58359023)

[6.4 Применение технологии «Маскирование» 21](#_Toc58359024)

[Заключение 23](#_Toc58359025)

[Список используемых источников 24](#_Toc58359026)

[Приложение A 25](#_Toc58359027)

[Приложение Б 26](#_Toc58359028)

# **Введение**

Центральной частью любого бизнес-приложения являются данные. Данные – это представленная в формализованном виде конкретная информация об объектах предметной области, их свойствах и взаимосвязях, отражающая события и ситуации в этой области. Именно на различных манипуляциях с данными строится большая часть пользовательских сценариев современных приложений. Эти сценарии описываются акронимом «CRUD» [1]. Он подразумевает под собой основные возможные пользовательские действия с данными – их создание (create), чтение (read), обновление (update) и удаление (delete).

На сегодняшний день, самой распространённой является реляционная модель структурирования данных. Реляционная база данных – это база данных, основанная на реляционной модели данных [2]. Она подразумевает под собой описание каждой сущности набором атрибутов, которые хранятся в соответствующих, для этих сущностей, таблицах.

Целью данной курсовой работы является разработка реляционной базы данных «Поликлиника». База данных должна содержать необходимые данные о пациентах, поликлиниках, паролях, врачах, заказанных талонов, а также хранить и шифровать данные о паролях.

В качестве СУБД для базы данных была выбрана Microsoft SQL Server 2016, в связи с ее производительностью и надежностью.

Необходимо создать основные объекты, такие как таблицы, процедуры, функции, пользователи, индексы. В базе данных должна быть реализована возможность экспорта и импорта данных в таблицы базы данных. Также после создание объектов и добавления данных в таблицы, нужно протестировать базу данных на производительность, вследствие чего добавить необходимые индексы для таблиц. В качестве технологии для курсового проекта была выбрана технология шифрования и маскирования данных.

1. Постановка задачи

Основной задачей курсового проекта является разработка базы данных «Поликлиника». Главная идея этой базы данных заключается в том, что пользователи могут заказывать талоны к врачам, что даёт возможность заказа талона. Данная база должна упростить и ускорить работу поликлиники, а также должна структурировать работу врачей и уменьшить скопление людей в одном помещении.

Задачу можно сформулировать следующим образом: создать необходимые сущности, которые будут отражать хранимые данные; охарактеризовать эти сущности с помощью соответствующих атрибутов, с корректно подобранными ограничениями целостности; связать таблицы между собой, тем самым получив полноценную реляционную структуру.

Необходимо реализовать шифрования и маскирование и протестировать производительность базы данных на таблицах, содержащих не менее 100 000 строк.

Функционально должны быть выполнены следующие задачи:

* возможность создавать, удалять и редактировать врача;
* возможность создавать, удалять и редактировать пациента;
* возможность добавлять, удалять и редактировать талоны к врачам;
* возможность осуществлять просмотр данных в зашифрованном и расшифрованном виде;
* возможность просмотра талонов, врачей и пациентов;
* возможность просматривать данных в замаскированном виде;
* возможность удалять данные из всех таблиц;
* возможность просматривать данные не в замаскированном виде.

1. Разработка модели базы данных

Одним из ключевых моментов при проектировании и создании базы данных является грамотный анализ предметной области приложения. Как следствие, составление такой модели данных, которая будет правильно отражать то, как с этими с данными в общем, и этой моделью, в частности, подразумевается взаимодействовать.

Результатом корректно проведённого анализа, проектирования, и разработки, является модель, способная предоставить функционально все необходимые возможности для пользователя. Можно выделить набор таких операций, как:

* возможность регистрации, редактирования и удаления врачей;
* возможность добавления, редактирования и удаления департаментов;
* возможность добавления, редактирования и удаления пациентов;
* возможность добавления, редактирования и удаления поликлиник;
* возможность добавлять, редактировать и удалять талоны к врачам;
* возможность вывода информации о врачах, пациентах, отделениях и поликлиниках.

Основой инфраструктуры базы данных является грамотно спроектированная модель, которая отображает связь пользовательских таблиц. Правильное и корректное взаимодействие их друг с другом как раз и заключается в схеме базы данных со связями, верно отображающими их положение.

На рисунке 2.1 отображена модель базы данных, полученная в ходе анализа предметной области и необходимого функционала. Между сущностями, представленными таблицами, также настроены связи, отражающие их взаимосвязь.

|  |
| --- |
|  |

Рисунок 2.1 – Структурная схема базы данных

На рисунке 2.1 проиллюстрированы связи, которые помогли в разработке реляционной базы данных. В основном применялись связи между таблицами один ко многим.

1. Разработка необходимых объектов

При разработке курсового проекта понадобились следующие объекты: таблицы и хранимые процедуры.

* 1. Таблицы

Центральным объектом любой базы данных является связанная совокупность таблиц. Таблицы ограничивают и упорядочивают хранимую информацию, а также обеспечивают связанность за счет внешних ключей.

Для реализации базы данных «Поликлиника» было разработано 8 таблиц.

Таблица DEPARTMENT содержит информацию о отделениях поликлиники и включает 3 столбца:

* столбец ID\_DEPARTMNET является первичным ключом, тип данных int, инкрементируется при добавлении департамента;
* столбец NAME\_DEPARTMENT хранит наименование отделения, тип данных nvarchar(30) и не может быть null.
* столбец ID\_HOSPITAL является внешним ключом, идентификатор поликлиники, тип данных int.

Пример таблицы DEPARTMENT представлен на рисунке 3.1.

|  |
| --- |
|  |

Рисунок 3.1 – Пример таблицы DEPARTMENT

Таблица DOCTOR содержит информацию о врачах и включает 12 столбца:

* столбец ID\_DOCTOR является первичным ключом, тип данных int, инкрементируется при добавлении врача;
* столбец ID\_DEPARTMNET является внешним ключом, идентификатор отделения, тип данных int, может быть нулевым;
* столбец FIRST\_NAME хранит имя врача, тип данных nvarchar(20);
* столбец LAST\_NAME хранит фамилию врача, тип данных nvarchar(20);
* столбец PATRONY хранит отчество врача, тип данных nvarchar(20);
* столбец OFFICE хранит кабинет врача, тип данных nvarchar(10);
* столбец QUALIFICATION хранит квалификацию врача, тип данных nvarchar(30);
* столбец BDAY хранит дату рождения врача, тип данных date;
* столбец DATE\_STARTED хранит дату приёма врача на работу, тип данных date;
* столбец POSITON хранит должность врача, тип данных nvarchar(20);
* столбец PHONE\_NUMBER хранит номер телефона врача, тип данных nvarchar(20);
* столбец ID\_HOSPITAL внешний ключ, хранит идентификатор поликлиники, тип данных int.

Пример таблицы DOCTOR представлен на рисунке 3.2.

|  |
| --- |
|  |

Рисунок 3.2 – Пример таблицы DOCTOR

Таблица HOSPITAL содержит информацию о поликлиниках и включает 6 столбцов:

* столбец ID\_HOSPITAL является первичным ключом, тип данных int, инкрементируется при добавлении поликлиники;
* столбец COUNTRY хранит название страны, тип данных nvarchar(40);
* столбец CITY хранит название города, тип данных nvarchar(40);
* столбец AREA хранит название района, тип данных nvarchar(40);
* столбец NAME\_HOSPITAL хранит название поликлиники, тип данных nvarchar(60);
* столбец REGISTRATION\_PHONE хранит номер регистрации, тип данных nvarchar(10).

Пример таблицы HOSPITAL представлен на рисунке 3.3.

|  |
| --- |
|  |

Рисунок 3.3 – Пример таблицы HOSPITAL

Таблица PASSWORD\_TABLE содержит пароли врачей и пациентов и включает 4 столбца:

* столбец ID\_PASSWORD является первичным ключом, тип данных int, инкрементируется при добавлении команды;
* столбец ID\_DOCTOR внешний ключ, хранит идентификатор доктора, тип данных int, может быть null;
* столбец ID\_PATIENT внешний ключ, хранит идентификатор пациента, тип данных int, может быть null;
* столбец USER\_PASSWORD хранит пароль в зашифрованном виде, тип данных varbinary(128).

Пример таблицы PASSWORD\_TABLE представлен на рисунке 3.4.

|  |
| --- |
|  |

Рисунок 3.4 – Пример таблицы PASSWORD\_TABLE

Таблица PATIENT содержит информацию о пациентах и включает 12 столбцов:

* столбец ID\_PATIENT является первичным ключом, тип данных int, инкрементируется при добавлении пациента;
* столбец GENDER хранит пол человека, тип данных char(1), значения проверяются;
* столбец FIRST\_NAME хранит имя пациента, тип данных nvarchar(20);
* столбец LAST\_NAME хранит фамилию пациента, тип данных nvarchar(20);
* столбец PATRONY хранит отчество пациента, тип данных nvarchar(20);
* столбец BDAY хранит дату рождения пациента, тип данных date;
* столбец HOME\_ADRESS хранит адрес пациента, тип данных nvarchar(100);
* столбец PASSPORT\_ID хранит паспорт пациента, тип данных nvarchar(100), может быть null;
* столбец ID\_HOSPITAL внешний ключ, хранит идентификатор поликлиники, тип данных int;
* столбец PLACE\_OF\_WORK\_OR\_STUDY хранит место учёбы или работы, тип данных nvarchar(40), может быть null;
* столбец DISABILITY хранит информацию о инвалидности, тип данных nvarchar(40), может быть null;
* столбец CHRONIC\_DIASEASES хранит информацию о хроническом заболевании, тип данных nvarchar(40), может быть null.

Пример таблицы PATIENT представлен на рисунке 3.5.

|  |
| --- |
|  |

Рисунок 3.5 – Пример таблицы PATIENT

Таблица RECEPTION содержит информацию о талонах к врачу и включает 9 столбцов:

* столбец ID\_RECEPTION является первичным ключом, тип данных int, инкрементируется при добавлении талона;
* столбец ID\_DOCTOR внешний ключ, содержит идентификатор доктора, тип данных int;
* столбец ID\_PATIENT внешний ключ, содержит идентификатор пациента, тип данных int;
* столбец ID\_HOSPITAL внешний ключ, содержит идентификатор поликлиники, тип данных int;
* столбец ID\_TIME внешний ключ, содержит идентификатор времени приёма, тип данных int;
* столбец SYMPTOMS хранит симптомы, тип данных nvarchar(100), может быть null;
* столбец DIAGNOSIS хранит диагноз, тип данных nvarchar(100);
* столбец MEDICINES хранит лекарства, тип данных nvarchar(100);
* столбец COMMENTS хранит комментарии, тип данных nvarchar(200), может быть null.

Пример таблицы RECEPTION представлен на рисунке 3.6.

|  |
| --- |
|  |

Рисунок 3.6 – Пример таблицы RECEPTION

Таблица SPECIALTY хранит информацию о специальности врачей и включает 3 столбца:

* столбец ID\_SPECIALTY является первичным ключом, тип данных int, инкрементируется при добавлении специальности;
* столбец ID\_DOCTOR является внешним ключом, идентификатор врача, которого добавили, тип данных int;
* столбец SPECIALTY\_NAME хранит название специальности, тип данных nvarchar(50).

Пример таблицы SPECIALTY представлен на рисунке 3.7.

|  |
| --- |
|  |

Рисунок 3.7 – Пример таблицы RECEPTION

Таблица TIME\_TIMETABLE содержит время приёма к врачу и включает 4 столбца:

* столбец ID\_TIME является первичным ключом, тип данных int, инкрементируется при добавлении времени;
* столбец ID\_DOCTOR внешний ключ, хранит идентификатор доктора, тип данных int;
* столбец DAY\_OF\_THE\_WEEK хранит дату приёма врача, тип данных date.
* столбец STARTING\_A\_PATIENT\_RECEPTION хранит время приёма врача в часах, тип данных nvarchar(20).

Пример таблицы TIME\_TIMETABLE представлен на рисунке 3.8.

|  |
| --- |
|  |

Рисунок 3.8 – Пример таблицы TIME\_TIMETABLE

Листинг создания таблиц представлен в приложении А.

* 1. Процедуры

Процедура представляет собой модуль, выполняющий одно или несколько действий. Поскольку вызов процедуры в T-SQL является отдельным исполняемым оператором, блок кода T-SQL может состоять только из вызова процедуры. Процедуры относятся к числу ключевых компонентов модульного кода, обеспечивающих оптимизацию и повторное использование программной логики [3].

Для реализации базы данных «Поликлиника» было разработано 57 процедур, которые реализуют функции CRUD [1].

Процедуры, связанные c манипулированием данных в таблицах, находятся в файле SQLQuery2.sql:

* ADD\_DEPARTMENT – добавление отделения;
* ADD\_DOCTOR – добавление доктора;
* ADD\_HOSPITAL – добавление поликлиники;
* ADD\_PATIENT – добавление пациента;
* ADD\_RECEPTION – добавление талона к врачу;
* ADD\_TIME\_TIMETABLE – добавление времени приёма к врачу;
* CREATE\_STRING\_60 – генерация строки nvarchar(60);
* DEL\_ALL\_TIME – удаление всех данных из таблицы TIME\_TIMETABLE и каскадное удаление из таблицы RECEPTION;
* DEL\_BY\_NAME\_HOSPITAL – удаление поликлиники по названию;
* DEL\_DATA\_ALL\_TABLE – удаление всех данных из всех таблиц;
* DEL\_DEPARTMENT\_BY\_ID – удаление отделения по идентификатору;
* DEL\_DOCTOR\_BY\_ID – удаление доктора по идентификатору;
* DEL\_HOSPITAL\_BY\_ID – удаление поликлиники по идентификатору;
* DEL\_PATIENT\_BY\_ID – удаление пациента по идентификатору;
* DEL\_RECEPTION\_BY\_ID – удаление талона к врачу по идентификатору;
* DEL\_TIME – удаление времени;
* DROP\_ALL\_TABLE – удаление всех таблиц;
* EXPORT\_TO\_JSON\_DEPARTMENT – экспорт отделений;
* EXPORT\_TO\_JSON\_DOCTOR – экспорт врачей;
* EXPORT\_TO\_JSON\_HOSPITAL – экспорт поликлиник;
* EXPORT\_TO\_JSON\_PATIENT – экспорт пациентов;
* EXPORT\_TO\_JSON\_RECEPTION – экспорт талонов;
* EXPORT\_TO\_JSON\_SPECIALTY – экспорт специальностей;
* EXPORT\_TO\_JSON\_TIME – экспорт времени приёма;
* EXPORT\_TO\_JSON\_PASSWORD – экспорт паролей;
* IMPORT\_HOSPITAL – импорт всех таблиц;
* INSERT\_HOSPITAL – генерация и вставка строк;
* SELECT\_DEPARTMENT\_ON\_HOSPITAL\_ID –вывод отделений, по идентификатору поликлиники;
* SELECT\_DEPARTMENT\_ON\_HOSPITAL\_NAME – вывод отделений, по названию поликлиники;
* SELECT\_ALL\_DEPARTMENT – вывод всех отделений;
* SELECT\_ALL\_DOCTOR – вывод всех врачей;
* SELECT\_ALL\_DOCTOR\_INCLUDE\_IN\_DEPARTMENT – вывод всех врачей входящих в отделение;
* SELECT\_ALL\_HEAD\_PLHYSICIAN – вывод всех главных врачей;
* SELECT\_ALL\_HOSPITAL – вывод всех поликлиник;
* SELECT\_ALL\_PATIENT\_MASK – вывод всех пациентов с применением маскировка;
* SELECT\_ALL\_PATIENT\_UNMASK – вывод всех пациентов без применения маскировки;
* SELECT\_ALL\_RECEPTION – вывод всех талонов;
* SELECT\_DOCTOR\_FIRST\_AND\_LAST\_NAME – вывод доктора по его имени и фамилии;
* SELECT\_DOCTOR\_ID – вывод доктора по его идентификатору;
* SELECT\_DOCTOR\_SPECIALTY – вывод информации о специальности и о докторе;
* SELECT\_HOSPITAL\_FOR\_COUNTRY – вывод поликлиник в заданной стране;
* SELECT\_HOSPITAL\_FOR\_ID – вывод поликлиники по её идентификатору;
* SELECT\_HOSPITAL\_FOR\_NAME – вывод поликлиники по её названию;
* SELECT\_PASSWORD\_DOCTOR\_DECRYPTED – вывод паролей врачей в зашифрованном виде;
* SELECT\_PASSWORD\_DOCTOR\_ENCRYPTED – вывод паролей врачей в расшифрованном виде;
* SELECT\_PATIENT\_AND\_PASSWORD – вывод пациентов и их паролей;
* SELECT\_PATIENT\_FOR\_FIRST\_AND\_LAST\_NAME – вывод пациента по его имени и фамилии;
* SELECT\_RECEPTION\_FOR\_DATE – вывод талонов по определённой дате;
* SELECT\_RECEPTION\_FOR\_DATE\_AND\_FIRST\_AND\_LAST\_NAME\_DOCTOR – вывод талонов по определённой дате и по имени и фамилии врача;
* SELECT\_RECEPTION\_FOR\_DATE\_AND\_TIME – вывод всех талонов по дате и времени приёма;
* SELECT\_RECEPTION\_FOR\_DOCTOR\_FIRST\_AND\_LAST\_NAME – вывод всех талонов по имени и фамилии доктора;
* SELECT\_RECEPTION\_FOR\_PATIENT\_AND\_LAST\_NAME – вывод всех талонов по имени и фамилии пациента;
* SELECT\_RECEPTION\_FOR\_TIME – вывод талонов по определённому времени;
* UPDATE\_DEPARTMENT\_INPUT\_NAME – изменение отделения по его имени;
* UPDATE\_DOCTOR – изменение доктора по его идентификатору;
* UPDATE\_HOSPITAL\_INPUT\_NAME – изменение поликлиники по названию;
* UPDATE\_PATIENT – изменение пациента по идентификатору;
* UPDATE\_RECEPTION – изменение талону по идентификатору.

Пример вызова процедуры ADD\_DOCTOR представлен на рисунке 3.9.

|  |
| --- |
|  |

Рисунок 3.9 – Пример вызова процедуры ADD\_DOCTOR

Листинг создания процедур представлены в приложениях Б соответственно.

1. Описание процедур импорта и экспорта

Одной из задач по разработке инфраструктуры базы данных являлась реализация процедур, которые будут осуществлять экспорт и импорт данных в формате JSON.

Для импорта данных из JSON-файла необходимо выполнить запрос, который изображен на рисунке 4.1.

|  |
| --- |
|  |

Рисунок 4.1 – Импорт данных из JSON файл

Эта процедура импортирует данные для всех таблиц и вызывает соответствующие процедуры по добавлению данных в таблицы. Если JSON не будет соответствовать требованиям, то будет выведено соответствующие сообщение.

Для экспорта данных в JSON-файл реализовано 7 процедур, которые экспортируют данные из таблиц в файлы с соответствующим расширением и вывода статуса выполненной операции. Список процедур, которые экспортируют данные в JSON:

* EXPORT\_TO\_JSON\_DEPARTMENT – экспорт департаментов;
* EXPORT\_TO\_JSON\_DOCTOR – экспорт врачей;
* EXPORT\_TO\_JSON\_HOSPITAL – экспорт поликлиник;
* EXPORT\_TO\_JSON\_PATIENT – экспорт пациентов;
* EXPORT\_TO\_JSON\_RECEPTION – экспорт талонов к врачу;
* EXPORT\_TO\_JSON\_SPECIALTY – экспорт специальностей врачей;
* EXPORT\_TO\_JSON\_PASSWORD – экспорт паролей;
* EXPORT\_TO\_JSON\_TIME – экспорт времен приёма врачей пациентов.

1. Тестирование производительности

Оптимизация запросов — процесс изменения запроса и/или структуры БД с целью уменьшения использования вычислительных ресурсов при выполнении запроса. Один и тот же результат может быть получен СУБД различными способами (планами выполнения запросов), которые могут существенно отличаться как по затратам ресурсов, так и по времени выполнения.

В MS SQL Server оптимизация запросом в основном заключается в построение индексов над таблицами, и изменением плана запроса. Индекс – это объект базы данных, предназначенный для ускорения запросов к данным в таблице базы данных. Индекс состоит из набора страниц, узлов индекса, которые организованы в виде древовидной структуры — сбалансированного дерева. Эта структура является иерархической по своей природе и начинается с корневого узла на вершине иерархии и конечных узлов, листьев. Есть два основных типа индексов: кластеризованные и некластеризованные индексы [4].

При создании кластеризованного индекса данные индексируемой таблицы располагаются в физическом порядке, соответствующем индексу, и становятся частью кластеризованного индекса.

Некластеризованный индекс — это отдельный объект, имеющий указатели на строки таблицы. Максимальное количество некластерированных индексов для одной таблицы не должно превышать 1000.

Рассмотрим следующие рекомендации при планировании стратегии индексирования:

* для таблиц которые часто обновляются используйте как можно меньше индексов;
* если таблица содержит большое количество данных, но их изменения незначительны, тогда используйте столько индексов, сколько необходимо для улучшения производительности ваших запросов;
* уникальность значений в столбце влияет на производительность индекса, в общем случае, чем больше у вас дубликатов в столбце, тем хуже работает индекс;
* для кластеризованных индексов старайтесь использовать настолько короткие поля насколько это возможно. Наилучшим образом будет применение кластеризованного индекса на столбцах с уникальными значениями и не позволяющими использовать NULL. Вот почему первичный ключ часто используется как кластеризованный индекс.

Важной характеристикой кластеризованного индекса является то, что все значения отсортированы в определенном порядке либо возрастания, либо убывания. Таким образом, таблица или представление может иметь только один кластеризованный индекс.

Для тестирования базы данных проекта была выбрана таблица HOSPITAL. В базе данных данного курсового проекта в каждой таблице находится поле с уникальным значением поля — id типа int, соответственно каждая таблица содержит кластеризованный индекс.

База данных содержит множество процедур с выборкой содержащий оператор WHERE сравнивающий строки. Даже при незначительном заполнении базы данных (около 100000 записей в таблице) время выборки начинает занимать существенное время.

Ниже на рисунке 5.1 представлена план запроса при выборке в таблице HOSPITAL.

|  |
| --- |
|  |

Рисунок 5.1 – План запроса с кластеризованным индексом

Как можно увидеть, стоимость запроса достаточно велика по сравнению с таблицей, содержащей некластеризованный индекс. Был создан для этой таблицы некластеризованный индекс и стоимость запроса уменьшилась. План запроса с применением некластеризованного индекса представлен на рисунке 5.2.

|  |
| --- |
|  |

Рисунок 5.2 – План запроса с некластеризованным индексом

Как видно по планам запроса стоимость запроса с некластеризованным меньше, чем с индексом. В следствии этого можно сделать вывод, что мы ускорили выборку данных из таблицы.

1. Описание технологии

6.1 Технология «Шифрование»

Шифрование представляет собой способ скрытия данных с помощью ключа или пароля. Это делает данные бесполезными без соответствующего ключа или пароля для дешифрования. Шифрование не решает проблемы управления доступом. Однако оно повышает защиту за счет ограничения потери данных даже при обходе системы управления доступом. Например, если компьютер, на котором установлена база данных, был настроен неправильно и злоумышленник смог получить конфиденциальные данные, то украденная информация будет бесполезна, если она была предварительно зашифрована.

SQL Server шифрует данные, используя иерархическую структуру средств шифрования и управления ключами. На каждом уровне данные низшего уровня шифруются на основе комбинации сертификатов, асимметричных ключей и симметричных ключей. Асимметричные и симметричные ключи можно хранить вне модуля расширенного управления ключами SQL Server.

SQL Server поддерживает следующие механизмы шифрования:

– transact-SQL – отдельные элементы можно шифровать по мере того, как они вставляются или обновляются, с помощью функций Transact-SQL;

– асимметричные ключи – асимметричный ключ состоит из закрытого ключа и соответствующего открытого ключа. Каждый из этих ключей позволяет дешифровать данные, зашифрованные другим ключом. На выполнение асимметричных операций шифрования и дешифрования требуется сравнительно много ресурсов, но они обеспечивают более надежную защиту, чем симметричное шифрование. Асимметричный ключ можно использовать для шифрования симметричного ключа перед его сохранением в базе данных;

– симметричные ключи – это ключ, используемый и для шифрования, и для дешифрования данных. Данные при использовании симметричного ключа шифруются и дешифруются быстро, и он вполне подходит для повседневной защиты конфиденциальных данных, хранящихся в базе данных;

– сертификаты – сертификат открытого ключа, или просто сертификат, представляет собой подписанную цифровой подписью инструкцию, которая связывает значение открытого ключа с идентификатором пользователя, устройства или службы, имеющей соответствующий закрытый ключ. Сертификаты поставляются и подписываются центром сертификации (certification authority, CA). Сущность, получающая сертификат от центра сертификации, является субъектом этого сертификата. Главное преимущество сертификатов в том, что они позволяют не хранить на узлах совокупность паролей отдельных субъектов. Вместо этого узел просто устанавливает доверительные отношения с поставщиком сертификата; после этого поставщик может подписать неограниченное количество сертификатов;

– прозрачное шифрование данных – прозрачное шифрование данных (TDE) является особым случаем шифрования с использованием симметричного ключа. TDE шифрует всю базу данных, используя симметричный ключ, который называется ключом шифрования базы данных. Ключ шифрования базы данных защищен другими ключами или сертификатами, которые, в свою очередь, защищаются главным ключом базы данных или асимметричным ключом, хранящимся в модуле расширенного управления ключами.

6.2 Применение технологии «Шифрования»

Для реализации технологии применяется функция EncryptByKey. Функция получает два аргумента:

* key\_GUID – Идентификатор GUID ключа, используемого для шифрования cleartext;
* cleartext – переменная типа **nvarchar**, **char**, **varchar**, **binary**, **varbinary** или **nchar**, содержащая данные, которые будут зашифрованы с помощью ключа.

В курсовой работе данные будут шифроваться при помощи симметричного шифрования и будет использован алгоритм AES\_256.

На рисунке 6.1 продемонстрирована процедура, которая совершает шифрование паролей пользователей.

|  |
| --- |
|  |

Рисунок 6.1 – Процедура шифрования пароля

Данная процедура принимает пароль в качестве параметра и добавляет пароль в зашифрованном виде в таблицу PASSWORD\_TABLE. Для того, чтобы вывести данные в расшифрованном виде требуется вызвать функцию DecryptByKey, которая принимает зашифрованное поле и затем конвертировать в nvarchar. Пример применения вывода данных в расшифрованном виде представлен на рисунке 6.2.

|  |
| --- |
|  |

Рисунок 6.2 – Процедура вывода пароля в расшифрованном виде

6.3 Технология «Маскирование»

Маскирование данных – это один из методов защиты конфиденциальной информации от несанкционированного доступа, при котором используются различные методы модификации данных, требующих защиты. Метод чаще всего используется при работе с базами данных (в том числе с помощью веб-приложений). Маскирование позволяет скрыть часть данных либо при создании копии базы данных, либо в ответе на запрос какой-либо информации, сохраняя общую информационную структуру и оставляя доступной ту часть данных, которая необходима для работы.

Динамическое маскирование данных ограничивает возможность раскрытия конфиденциальных данных за счёт маскирования этих данных для непривилегированных пользователей. Оно позволяет значительно упростить проектирование и написание кода для системы безопасности в приложении.

Назначение динамического маскирования данных – ограничение раскрытия конфиденциальных данных, при котором пользователи, у которых нет доступа к данным, не смогут их просматривать.

Правило маскирования можно задать для столбца в таблице, чтобы замаскировать данные в этом столбце. Доступны четыре типа маскирования:

* по умолчанию – полное маскирование в соответствии с типами данных назначенных полей;
* email – метод маскирования, который раскрывает первую букву адреса электронной почты и постоянный суффикс, в формате адреса электронной почты;
* случайные – функция случайного маскирования для использования с любым числовым типом, которая маскирует исходное значение случайным значением в указанном диапазоне;
* пользовательская строка – метод маскирования, который раскрывает первую и последнюю буквы и добавляет пользовательскую строку заполнения в середине.

6.4 Применение технологии «Маскирование»

Для реализации технологии в курсовой работе используется метод маскирования, который представлен на рисунке 6.3.

|  |
| --- |
|  |

Рисунок 6.3 – Используем метод маскирования

Затем я создал пользователя, у которого нету прав на вывод данных в незамаскированном виде рисунок 6.4.

|  |
| --- |
|  |

Рисунок 6.4 – Вывод данных в замаскированном виде

Чтобы вывести данные в незамаскированном виде требуется использовать процедуру SELECT\_ALL\_PATIENT\_UNMASK пример применения, которой представлен на рисунке 6.5.

|  |
| --- |
|  |

Рисунок 6.5 – Вывод данных в незамаскированном виде

Таким образом, можно скрывать данные от пользователей, которые не имеют нужные права на просмотр.

Заключение

В ходе выполнения данной работы подробно были закреплены знания для проектирования баз данных, с учётом большого количества хранимых параметров, учётом множества связей и использование процедур, курсоров и функций для осуществления взаимодействий с таблицами. Было проведено тестирование производительности в котором анализ времени выполнения показал, что база данных работает быстро, даже не в реалистичных условиях.

Была изучена технология шифрования и маскирования данных, которая позволяет скрывать/шифровать данные, что помогает скрыть данные от посторонних пользователей.

Благодаря проведенной работе, была создана база данных, которая в дальнейшем может быть использована на проект и также её можно легко дорабатывать и расширять её возможности.

В ходе курсового проекта было разработано 57 процедур, которые предоставляют основной функционал к моей базе данных. Также было разработано 8 таблиц, которые связаны между собой и представляют реляционную базу данных.

В записке представлено 18 иллюстраций, 7 литературных источников, которые помогли в разработке базы данных и разработке технологий.

Основным плюсом данной бд является простота и интуитивность для других разработчиков. Я считаю, что данный проект позволил понять, как устроена одна из сфер медицины.

# **Список используемых источников**

1. Свободная энциклопедия Википедия [Электронный ресурс] /CRUD – Режим доступа – URL: https://ru.wikipedia.org/wiki/CRUD – Дата доступа: 23.11.2020
2. Базы Данных. Учебное пособие [Электронный ресурс] /Базы данных – Режим доступа – URL: <https://siblec.ru/informatika-i-vychislitelnaya-tekhnika/bazy-dannykh> – Дата доступа: 23.11.2020
3. Об процедурах в MS SQL Server [Электронный ресурс] / Режим доступа – URL: https://metanit.com/sql/sqlserver/11.1.php
4. Об индексах в MS SQL Server [Электронный ресурс] / Режим доступа – URL: <https://www.sql.ru/articles/mssql/03013101indexes.shtml>.
5. Блинова Е.А. Курс лекций по базам данных/ Е.А. Блинова
6. Об используемом маскировании данных в MS SQL Server [Электронный ресурс] / Режим доступа – URL: <https://docs.microsoft.com/en-us/sql/relational-databases/security/dynamic-data-masking?view=sql-server-ver15>
7. Об шифровании данных в MS SQL Server [Электронный ресурс] / Режим доступа – URL: https://docs.microsoft.com/ru-ru/sql/t-sql/statements/create-symmetric-key-transact-sql?view=sql-server-ver15

Приложение A

|  |
| --- |
| create table HOSPITAL(  ID\_HOSPITAL INT PRIMARY KEY IDENTITY,  COUNTRY nvarchar(40) NOT NULL,  CITY nvarchar(40) NOT NULL,  AREA nvarchar(40) NOT NULL,  NAME\_HOSPITAL nvarchar(60) NOT NULL,  REGISTRATION\_PHONE nvarchar(10) NOT NULL);  create table DEPARTMENT (  ID\_DEPARTMENT INT PRIMARY KEY IDENTITY,  NAME\_DEPARTMENT NVARCHAR(30) NOT NULL,  ID\_HOSPITAL INT REFERENCES HOSPITAL(ID\_HOSPITAL) ON DELETE CASCADE NOT NULL);  CREATE TABLE PATIENT(  ID\_PATIENT INT PRIMARY KEY IDENTITY,  GENDER CHAR(1) DEFAULT 'M' CHECK (GENDER in('M','W')) not null,  FIRST\_NAME NVARCHAR(20) NOT NULL,  LAST\_NAME NVARCHAR(20) NOT NULL,  PATRONY NVARCHAR(20) NOT NULL,  BDAY DATE NOT NULL,  HOME\_ADRESS varchar(100) NOT NULL,  PASSPORT\_ID varchar(100) NULL,  ID\_HOSPITAL INT REFERENCES HOSPITAL(ID\_HOSPITAL) ON DELETE CASCADE NOT NULL,  PLACE\_OF\_WORK\_OR\_STUDY NVARCHAR(40),  DISABILITY NVARCHAR(40),  CHRONIC\_DISEASES NVARCHAR(40);  CREATE TABLE SPECIALTY(  ID\_SPECIALTY INT PRIMARY KEY IDENTITY,  ID\_DOCTOR INT REFERENCES DOCTOR(ID\_DOCTOR) ON DELETE CASCADE NOT NULL,  SPECIALTY\_NAME NVARCHAR(50) NOT NULL)  CREATE TABLE DOCTOR(  ID\_DOCTOR INT PRIMARY KEY IDENTITY,  ID\_DEPARTMENT INT REFERENCES DEPARTMENT(ID\_DEPARTMENT) ,  FIRST\_NAME NVARCHAR(20) NOT NULL,  LAST\_NAME NVARCHAR(20) NOT NULL,  PATRONY NVARCHAR(20) NOT NULL,  OFFICE NVARCHAR(10)NOT NULL,  QUALIFICATION NVARCHAR(30) NOT NULL,  BDAY DATE NOT NULL,  DATE\_STARTED DATE NOT NULL,  POSITION NVARCHAR(20) not null,  PHONE\_NUMBER NVARCHAR(20) null,  ID\_HOSPITAL INT REFERENCES HOSPITAL(ID\_HOSPITAL) NOT NULL  );  CREATE TABLE TIME\_TIMETABLE(  ID\_TIME INT PRIMARY KEY IDENTITY,  ID\_DOCTOR INT REFERENCES DOCTOR(ID\_DOCTOR) NOT NULL,  DAY\_OF\_THE\_WEEK DATE not null,  STARTING\_A\_PATIENT\_RECEPTION NVARCHAR(20) not null  );  CREATE TABLE RECEPTION(  ID\_RECEPTION INT PRIMARY KEY IDENTITY,  ID\_DOCTOR INT REFERENCES DOCTOR(ID\_DOCTOR) NOT NULL,  ID\_PATIENT INT REFERENCES PATIENT(ID\_PATIENT) NOT NULL,  ID\_HOSPITAL INT REFERENCES HOSPITAL(ID\_HOSPITAL) ON DELETE CASCADE NOT NULL,  ID\_TIME INT REFERENCES TIME\_TIMETABLE(ID\_TIME) ON DELETE CASCADE NOT NULL,  SYMPTOMS NVARCHAR(100),  DIAGNOSIS NVARCHAR(100) NOT NULL,  MEDICINES NVARCHAR(100) NOT NULL,  COMMENTS NVARCHAR(200));  CREATE TABLE PASSWORD\_TABLE(  ID\_PASSWORD INT PRIMARY KEY IDENTITY,  ID\_DOCTOR INT REFERENCES DOCTOR(ID\_DOCTOR) ON DELETE CASCADE,  ID\_PATIENT INT REFERENCES PATIENT(ID\_PATIENT) ON DELETE CASCADE,  USER\_PASSWORD varbinary(128) null); |

Приложение Б

|  |
| --- |
| CREATE PROCEDURE ADD\_DEPARTMENT @NAME\_DEPARTMENT NVARCHAR(30),@NAME\_HOSPITAL NVARCHAR(60)  AS BEGIN  DECLARE @ID\_HOSPITAL INT  BEGIN TRY  IF NOT EXISTS(SELECT ID\_HOSPITAL FROM HOSPITAL WHERE NAME\_HOSPITAL=@NAME\_HOSPITAL)  BEGIN PRINT 'ТАКОЙ ПОЛИКЛИНИ НЕТУ'; END;  ELSE  BEGIN  SET @ID\_HOSPITAL=(SELECT ID\_HOSPITAL FROM HOSPITAL WHERE NAME\_HOSPITAL=@NAME\_HOSPITAL);  IF NOT EXISTS(SELECT ID\_DEPARTMENT FROM DEPARTMENT WHERE NAME\_DEPARTMENT=@NAME\_DEPARTMENT AND ID\_HOSPITAL=@ID\_HOSPITAL)  BEGIN  INSERT INTO DEPARTMENT(NAME\_DEPARTMENT,ID\_HOSPITAL)  VALUES(@NAME\_DEPARTMENT,@ID\_HOSPITAL);  PRINT 'ОТДЕЛЕНИЕ ДОБАВЛЕНО'  END;  ELSE  BEGIN PRINT 'ТАКОЕ ОТДЕЛЕНИЕ УЖЕ ЕСТЬ' END;  END;END TRY BEGIN CATCH  PRINT 'Error ' + CONVERT(VARCHAR, ERROR\_NUMBER()) + ':' + ERROR\_MESSAGE()  END CATCH END  CREATE PROCEDURE ADD\_DOCTOR @NAME\_DEPARTMENT NVARCHAR(30)=NULL,@FIRST\_NAME NVARCHAR(20),@LAST\_NAME NVARCHAR(20),@PATRONY NVARCHAR(20),@OFFICE NVARCHAR(10),@QUALIFICATION NVARCHAR(30),@BDAY DATE,@DATE\_STARTED DATE,@POSITION NVARCHAR(20),@PHONE\_NUMBER NVARCHAR(20)=NULL,@NAME\_HOSPITAL NVARCHAR(60),@PASSWORD\_USER NVARCHAR(60)='123',  @SPECIALTY\_NAME NVARCHAR(50)  AS  DECLARE @ID\_HOSPITAL INT  DECLARE @ID\_DEPARTMENT1 INT  DECLARE @ID\_DOCTOR INT  DECLARE @FLAG INT=0  BEGIN BEGIN TRY  IF NOT EXISTS(SELECT ID\_HOSPITAL FROM HOSPITAL WHERE NAME\_HOSPITAL=@NAME\_HOSPITAL)  BEGIN SET @FLAG=1; PRINT 'ТАКОЙ ПОЛИКЛИНИ НЕТУ'; END;  ELSE BEGIN  SET @ID\_HOSPITAL=(SELECT ID\_HOSPITAL FROM HOSPITAL WHERE NAME\_HOSPITAL=@NAME\_HOSPITAL);  END;  IF NOT EXISTS(SELECT ID\_DEPARTMENT FROM DEPARTMENT WHERE NAME\_DEPARTMENT=@NAME\_DEPARTMENT and ID\_HOSPITAL=@ID\_HOSPITAL) AND @FLAG=0  BEGIN  IF @POSITION='гл. врач'  BEGIN SET @FLAG=0; END;  ELSE  BEGIN SET @FLAG=1; PRINT 'ТАКОГО ОТДЕЛЕНИЯ НЕТУ'; END;  END;  print @flag  IF @FLAG=0  BEGIN  SET @ID\_DEPARTMENT1=(SELECT ID\_DEPARTMENT FROM DEPARTMENT WHERE NAME\_DEPARTMENT=@NAME\_DEPARTMENT AND ID\_HOSPITAL=@ID\_HOSPITAL)  IF NOT EXISTS(SELECT ID\_DOCTOR FROM DOCTOR WHERE  FIRST\_NAME=@FIRST\_NAME AND LAST\_NAME=@LAST\_NAME AND PATRONY=@PATRONY AND OFFICE=@OFFICE AND QUALIFICATION=@QUALIFICATION AND BDAY=@BDAY AND DATE\_STARTED=@DATE\_STARTED  BEGIN  IF @POSITION='гл. врач'  BEGIN  IF NOT EXISTS(SELECT ID\_DOCTOR FROM DOCTOR WHERE ID\_HOSPITAL=@ID\_HOSPITAL AND POSITION='гл. врач')  BEGIN  INSERT INTO DOCTOR(ID\_DEPARTMENT,FIRST\_NAME,LAST\_NAME,PATRONY,OFFICE,QUALIFICATION,BDAY,DATE\_STARTED,POSITION,PHONE\_NUMBER,ID\_HOSPITAL) VALUES(NULL,@FIRST\_NAME,@LAST\_NAME,@PATRONY,@OFFICE,@QUALIFICATION,@BDAY,@DATE\_STARTED,@POSITION,@PHONE\_NUMBER,@ID\_HOSPITAL);  SET @ID\_DOCTOR=(SELECT MAX(ID\_DOCTOR) FROM DOCTOR);  INSERT INTO PASSWORD\_TABLE(ID\_DOCTOR,ID\_PATIENT,USER\_PASSWORD)VALUES(@ID\_DOCTOR,NULL,NULL);  OPEN SYMMETRIC KEY SSN\_Key\_1  DECRYPTION BY CERTIFICATE HumanResources;  UPDATE PASSWORD\_TABLE  SET USER\_PASSWORD = EncryptByKey(Key\_GUID('SSN\_Key\_1'), @PASSWORD\_USER) where ID\_DOCTOR=@ID\_DOCTOR ;  INSERT INTO SPECIALTY(ID\_DOCTOR,SPECIALTY\_NAME)VALUES(@ID\_DOCTOR,@SPECIALTY\_NAME);  PRINT 'ГЛ ВРАЧ ДОБАВЛЕН';  SET @FLAG=2;  END;  ELSE BEGIN PRINT 'ТАКОЙ ГЛАВ ВРАЧ УЖЕ ЕСТЬ' SET @FLAG=2; END;  END;  IF @POSITION='зам. отделения'  BEGIN  IF NOT EXISTS(SELECT ID\_DOCTOR FROM DOCTOR WHERE ID\_HOSPITAL=@ID\_HOSPITAL AND ID\_DEPARTMENT=@ID\_DEPARTMENT1 AND POSITION=@POSITION) AND @FLAG!=2  BEGIN  INSERT INTO DOCTOR(ID\_DEPARTMENT,FIRST\_NAME,LAST\_NAME,PATRONY,OFFICE,QUALIFICATION,BDAY,DATE\_STARTED,POSITION,PHONE\_NUMBER,ID\_HOSPITAL) VALUES(@ID\_DEPARTMENT1,@FIRST\_NAME,@LAST\_NAME,@PATRONY,@OFFICE,@QUALIFICATION,@BDAY,@DATE\_STARTED,@POSITION,@PHONE\_NUMBER,@ID\_HOSPITAL);  SET @ID\_DOCTOR=(SELECT MAX(ID\_DOCTOR) FROM DOCTOR);  INSERT INTO PASSWORD\_TABLE(ID\_DOCTOR,ID\_PATIENT,USER\_PASSWORD)VALUES(@ID\_DOCTOR,NULL,NULL);  OPEN SYMMETRIC KEY SSN\_Key\_1  DECRYPTION BY CERTIFICATE HumanResources;  UPDATE PASSWORD\_TABLE  SET USER\_PASSWORD = EncryptByKey(Key\_GUID('SSN\_Key\_1'), @PASSWORD\_USER) where ID\_DOCTOR=@ID\_DOCTOR ;  INSERT INTO SPECIALTY(ID\_DOCTOR,SPECIALTY\_NAME)VALUES(@ID\_DOCTOR,@SPECIALTY\_NAME);  PRINT 'ЗАМ. ОТЕДЕЛЕНИЯ ДОБАВЛЕН'  SET @FLAG=2;  END  ELSE BEGIN PRINT 'ТАКОЙ ЗАМ. ОТЕДЕЛЕНИЯ УЖЕ ЕСТЬ' SET @FLAG=2; END;  END;  IF @FLAG=0  BEGIN  INSERT INTO DOCTOR(ID\_DEPARTMENT,FIRST\_NAME,LAST\_NAME,PATRONY,OFFICE,QUALIFICATION,BDAY,DATE\_STARTED,POSITION,PHONE\_NUMBER,ID\_HOSPITAL) VALUES(@ID\_DEPARTMENT1,@FIRST\_NAME,@LAST\_NAME,@PATRONY,@OFFICE,@QUALIFICATION,@BDAY,@DATE\_STARTED,@POSITION,@PHONE\_NUMBER,@ID\_HOSPITAL);  SET @ID\_DOCTOR=(SELECT MAX(ID\_DOCTOR) FROM DOCTOR);  INSERT INTO PASSWORD\_TABLE(ID\_DOCTOR,ID\_PATIENT,USER\_PASSWORD)VALUES(@ID\_DOCTOR,NULL,NULL);  OPEN SYMMETRIC KEY SSN\_Key\_1  DECRYPTION BY CERTIFICATE HumanResources;  UPDATE PASSWORD\_TABLE  SET USER\_PASSWORD = EncryptByKey(Key\_GUID('SSN\_Key\_1'), @PASSWORD\_USER) where ID\_DOCTOR=@ID\_DOCTOR ;  INSERT INTO SPECIALTY(ID\_DOCTOR,SPECIALTY\_NAME)VALUES(@ID\_DOCTOR,@SPECIALTY\_NAME);  PRINT 'ВРАЧ ДОБАВЛЕН'  END;  END  ELSE BEGIN PRINT 'ТАКОЙ ВРАЧ УЖЕ ЕСТЬ' END;  END;  END TRY BEGIN CATCH  PRINT 'Error ' + CONVERT(VARCHAR, ERROR\_NUMBER()) + ':' + ERROR\_MESSAGE()  END CATCH END;  CREATE PROCEDURE ADD\_HOSPITAL @COUNTRY NVARCHAR(40)='NO NAME',@CITY NVARCHAR(40)='NO NAME',@AREA NVARCHAR(40)='NO NAME',@NAME\_HOSPITAL NVARCHAR(60),@REGISTRATION\_PHONE NVARCHAR(10)='NO NAME'  AS BEGIN BEGIN TRY  IF NOT EXISTS(SELECT ID\_HOSPITAL FROM HOSPITAL WHERE NAME\_HOSPITAL=@NAME\_HOSPITAL)  BEGIN  INSERT INTO HOSPITAL(COUNTRY,CITY,AREA,NAME\_HOSPITAL,REGISTRATION\_PHONE)  VALUES(@COUNTRY,@CITY,@AREA,@NAME\_HOSPITAL,@REGISTRATION\_PHONE);  PRINT 'ПОЛИКЛИНИКА ДОБАВЛЕНА'  END  ELSE BEGIN PRINT 'ОШИБКА В ПАРАМЕТРАХ' END;  END TRY BEGIN CATCH  PRINT 'Error ' + CONVERT(VARCHAR, ERROR\_NUMBER()) + ':' + ERROR\_MESSAGE()  END CATCH END;  CREATE PROCEDURE ADD\_PATIENT @GENDER CHAR(1),@FIRST\_NAME NVARCHAR(20),@LAST\_NAME NVARCHAR(20),@PATRONY NVARCHAR(20),@BDAY DATE,@HOME\_ADRESS NVARCHAR(40),@PASSPORT\_ID NVARCHAR(30)=NULL,@NAME\_HOSPITAL NVARCHAR(60),@PLACE\_OF\_WORK\_OR\_STUDY NVARCHAR(40)=NULL,@DISABILITY NVARCHAR(40)=NULL,@CHRONIC\_DISEASES NVARCHAR(40)=NULL,@PASSWORD\_USER NVARCHAR(60)='123'  AS DECLARE @ID\_HOSPITAL INT DECLARE @ID\_PATIENT INT BEGIN BEGIN TRY  IF NOT EXISTS(SELECT ID\_HOSPITAL FROM HOSPITAL WHERE NAME\_HOSPITAL=@NAME\_HOSPITAL) OR (@GENDER!='M' AND @GENDER!='W')  BEGIN PRINT 'ТАКОЙ ПОЛИКЛИНИКИ НЕТ ИЛИ НЕВЕРНО УКАЗАН ПОЛ' END;  ELSE  BEGIN  IF NOT EXISTS(SELECT ID\_PATIENT FROM PATIENT WHERE GENDER=@GENDER AND FIRST\_NAME=@FIRST\_NAME AND LAST\_NAME=@LAST\_NAME AND PATRONY=@PATRONY AND BDAY=@BDAY AND HOME\_ADRESS=@HOME\_ADRESS  AND PASSPORT\_ID=@PASSPORT\_ID )  BEGIN  SET @ID\_HOSPITAL=(SELECT ID\_HOSPITAL FROM HOSPITAL WHERE NAME\_HOSPITAL=@NAME\_HOSPITAL);  INSERT INTO PATIENT(GENDER,FIRST\_NAME,LAST\_NAME,PATRONY,BDAY,HOME\_ADRESS,PASSPORT\_ID,ID\_HOSPITAL,PLACE\_OF\_WORK\_OR\_STUDY,DISABILITY,CHRONIC\_DISEASES) VALUES(@GENDER,@FIRST\_NAME,@LAST\_NAME,@PATRONY,@BDAY,@HOME\_ADRESS,@PASSPORT\_ID,@ID\_HOSPITAL,@PLACE\_OF\_WORK\_OR\_STUDY,@DISABILITY,@CHRONIC\_DISEASES);  SET @ID\_PATIENT=(SELECT MAX(ID\_PATIENT) FROM PATIENT);  INSERT INTO PASSWORD\_TABLE(ID\_DOCTOR,ID\_PATIENT,USER\_PASSWORD)VALUES(NULL,@ID\_PATIENT,NULL);  OPEN SYMMETRIC KEY SSN\_Key\_1  DECRYPTION BY CERTIFICATE HumanResources;  UPDATE PASSWORD\_TABLE  SET USER\_PASSWORD = EncryptByKey(Key\_GUID('SSN\_Key\_1'), @PASSWORD\_USER) where ID\_PATIENT=@ID\_PATIENT ;  PRINT 'ДАННЫЕ ДОБАВЛЕНЫ'  END;  ELSE BEGIN PRINT 'ЧЕЛОВЕК ТАКОЙ УЖЕ ЕСТЬ' END;  END  END TRY BEGIN CATCH  PRINT 'Error ' + CONVERT(VARCHAR, ERROR\_NUMBER()) + ':' + ERROR\_MESSAGE()  END CATCH END;  CREATE PROCEDURE ADD\_RECEPTION @FIRST\_NAME NVARCHAR(20),@LAST\_NAME NVARCHAR(20),@SPECIALTY\_NAME NVARCHAR(50),@FIRST\_NAME\_PATIENT NVARCHAR(20),@LAST\_NAME\_\_PATIENT NVARCHAR(20),@NAME\_HOSPITAL NVARCHAR(60),@TIME NVARCHAR(20),@DATE DATE,@SYMPTOMS NVARCHAR(100)=NULL,@DIAGNOSIS NVARCHAR(100),@MEDICINES NVARCHAR(100),@COMMENTS NVARCHAR(200)=NULL  AS  DECLARE @ID\_HOSPITAL INT  DECLARE @ID\_PATIENT INT  DECLARE @ID\_DOCTOR INT  DECLARE @FLAG INT  DECLARE @ID\_TIME INT  BEGIN BEGIN TRY  SET @FLAG=0  IF NOT EXISTS(SELECT ID\_HOSPITAL FROM HOSPITAL WHERE NAME\_HOSPITAL=@NAME\_HOSPITAL)  BEGIN SET @FLAG=1 PRINT 'ТАКОЙ ПОЛИКЛИНИКИ НЕТ ' END;  ELSE  BEGIN  SET @ID\_HOSPITAL=(SELECT ID\_HOSPITAL FROM HOSPITAL WHERE NAME\_HOSPITAL=@NAME\_HOSPITAL);  END  IF NOT EXISTS(SELECT ID\_DOCTOR FROM DOCTOR WHERE ID\_HOSPITAL=@ID\_HOSPITAL AND FIRST\_NAME=@FIRST\_NAME AND LAST\_NAME=@LAST\_NAME) AND @FLAG!=0  BEGIN SET @FLAG=1 PRINT 'ТАКОГО ВРАЧА НЕТ' END;  ELSE  BEGIN  SET @ID\_DOCTOR=(SELECT TOP(1) ID\_DOCTOR FROM DOCTOR WHERE ID\_HOSPITAL=@ID\_HOSPITAL AND FIRST\_NAME=@FIRST\_NAME AND LAST\_NAME=@LAST\_NAME)  IF NOT EXISTS(SELECT ID\_SPECIALTY FROM SPECIALTY WHERE ID\_DOCTOR=@ID\_DOCTOR AND SPECIALTY\_NAME=@SPECIALTY\_NAME)  BEGIN  PRINT 'ВРАЧ НЕ ЛЕЧИТ ТАКИХ ДУРАКОВ ИЛИ ВЫБРАНА НЕ ТА ДУРКА ИЛИ НЕ ВЫБРАНА ПРАВИЛЬНО СПЕЦИАЛЬНОСТЬ ВРАЧА'  SET @FLAG=1  END;  END;  IF NOT EXISTS(SELECT ID\_PATIENT FROM PATIENT WHERE ID\_HOSPITAL=@ID\_HOSPITAL AND FIRST\_NAME=@FIRST\_NAME\_PATIENT AND LAST\_NAME=@LAST\_NAME\_\_PATIENT) AND @FLAG!=1  BEGIN SET @FLAG=1 PRINT 'ТАКОГО БОЛЬНОГО НЕ СУЩЕСТВУЕТ ИЛИ НЕ ТОЙ ПОЛИКЛИНИКИ ОН ПРИНАДЛЕЖИТ' END;  ELSE  BEGIN  SET @ID\_PATIENT=(SELECT TOP(1) ID\_PATIENT FROM PATIENT WHERE ID\_HOSPITAL=@ID\_HOSPITAL AND FIRST\_NAME=@FIRST\_NAME\_PATIENT AND LAST\_NAME=@LAST\_NAME\_\_PATIENT)  END;  IF(GETDATE()>@DATE) OR DATENAME(weekday, @DATE)='воскресенье' OR DATENAME(weekday, @DATE)='суббота' OR  (DATENAME(hour, @TIME)=13 OR DATENAME(hour, @TIME)=14 OR (DATENAME(hour, @TIME)<10 OR DATENAME(hour, @TIME)>17))  BEGIN SET @FLAG=1 PRINT 'НЕВЕРНАЯ ДАТА ИЛИ ВРЕМЯ' END  IF @FLAG!=1  BEGIN PRINT 'ВСЁ ГУД'  INSERT INTO TIME\_TIMETABLE(ID\_DOCTOR,DAY\_OF\_THE\_WEEK,STARTING\_A\_PATIENT\_RECEPTION)values  (@ID\_DOCTOR,@DATE,@TIME);  SET @ID\_TIME=(SELECT MAX(ID\_TIME) FROM TIME\_TIMETABLE);  INSERT INTO RECEPTION(ID\_DOCTOR,ID\_PATIENT,ID\_HOSPITAL,ID\_TIME,SYMPTOMS,DIAGNOSIS,MEDICINES,COMMENTS)VALUES(@ID\_DOCTOR,@ID\_PATIENT,@ID\_HOSPITAL,@ID\_TIME,@SYMPTOMS,@DIAGNOSIS,@MEDICINES,@COMMENTS)  END;  END TRY BEGIN CATCH  PRINT 'Error ' + CONVERT(VARCHAR, ERROR\_NUMBER()) + ':' + ERROR\_MESSAGE()  END CATCH END;  CREATE PROCEDURE ADD\_TIME\_TIMETABLE @ID\_DOCTOR INT,@DAY\_OF\_THE\_WEEK NVARCHAR(20),@STARTING\_A\_PATIENT\_RECEPTION NVARCHAR(20)  AS  INSERT INTO TIME\_TIMETABLE(ID\_DOCTOR,DAY\_OF\_THE\_WEEK,STARTING\_A\_PATIENT\_RECEPTION)  VALUES(@ID\_DOCTOR,@DAY\_OF\_THE\_WEEK,@STARTING\_A\_PATIENT\_RECEPTION);  CREATE PROCEDURE DEL\_DATA\_ALL\_TABLE  AS BEGIN  DELETE PASSWORD\_TABLE;  DELETE RECEPTION;  DELETE TIME\_TIMETABLE;  DELETE SPECIALTY;  DELETE PATIENT;  DELETE DOCTOR;  DELETE DEPARTMENT;  DELETE HOSPITAL;  END;  CREATE PROCEDURE DROP\_ALL\_TABLE  AS BEGIN  DROP TABLE PASSWORD\_TABLE;  DROP TABLE RECEPTION;  DROP TABLE TIME\_TIMETABLE;  DROP TABLE SPECIALTY;  DROP TABLE PATIENT;  DROP TABLE DOCTOR;  DROP TABLE DEPARTMENT;  DROP TABLE HOSPITAL;  END;  CREATE PROCEDURE SELECT\_ALL\_DEPARTMENT  AS BEGIN SELECT \* FROM DEPARTMENT; END;  CREATE PROCEDURE SELECT\_\_DEPARTMENT\_ON\_HOSPITAL\_NAME @NAME\_HOSPITAL NVARCHAR(60)  AS BEGIN  SELECT NAME\_DEPARTMENT,COUNTRY,REGISTRATION\_PHONE,NAME\_HOSPITAL FROM DEPARTMENT INNER JOIN HOSPITAL ON DEPARTMENT.ID\_HOSPITAL=HOSPITAL.ID\_HOSPITAL  WHERE HOSPITAL.NAME\_HOSPITAL=@NAME\_HOSPITAL  END;  CREATE PROCEDURE SELECT\_\_DEPARTMENT\_ON\_HOSPITAL\_ID @ID\_HOSPITAL INT  AS BEGIN  SELECT NAME\_DEPARTMENT,COUNTRY,REGISTRATION\_PHONE,NAME\_HOSPITAL FROM DEPARTMENT INNER JOIN HOSPITAL ON DEPARTMENT.ID\_HOSPITAL=HOSPITAL.ID\_HOSPITAL  WHERE HOSPITAL.ID\_HOSPITAL=@ID\_HOSPITAL;  END;  CREATE PROCEDURE SELECT\_ALL\_DOCTOR\_INCLUDE\_IN\_DEPARTMENT AS BEGIN  SELECT ID\_DOCTOR,DEPARTMENT.NAME\_DEPARTMENT,HOSPITAL.NAME\_HOSPITAL,FIRST\_NAME,LAST\_NAME,PATRONY,OFFICE,QUALIFICATION,  BDAY,DATE\_STARTED,POSITION,PHONE\_NUMBER FROM DOCTOR INNER JOIN DEPARTMENT ON DOCTOR.ID\_DEPARTMENT=DEPARTMENT.ID\_DEPARTMENT  INNER JOIN HOSPITAL ON DOCTOR.ID\_HOSPITAL=HOSPITAL.ID\_HOSPITAL  END;  CREATE PROCEDURE SELECT\_ALL\_HEAD\_PLHYSICIAN AS BEGIN  SELECT ID\_DOCTOR,HOSPITAL.NAME\_HOSPITAL,FIRST\_NAME,LAST\_NAME,PATRONY,OFFICE,QUALIFICATION,  BDAY,DATE\_STARTED,POSITION,PHONE\_NUMBER FROM DOCTOR  INNER JOIN HOSPITAL ON DOCTOR.ID\_HOSPITAL=HOSPITAL.ID\_HOSPITAL WHERE DOCTOR.POSITION='гл. врач'  END;  CREATE PROCEDURE SELECT\_ALL\_DOCTOR AS BEGIN SELECT \* FROM DOCTOR; END;  CREATE PROCEDURE SELECT\_DOCTOR\_SPECIALTY AS BEGIN  SELECT DOCTOR.ID\_DOCTOR,SPECIALTY\_NAME,FIRST\_NAME,LAST\_NAME,PATRONY,OFFICE,QUALIFICATION,  BDAY,DATE\_STARTED,POSITION,PHONE\_NUMBER FROM DOCTOR INNER JOIN SPECIALTY  ON DOCTOR.ID\_DOCTOR=SPECIALTY.ID\_DOCTOR  END;  CREATE PROCEDURE SELECT\_PASSWORD\_DOCTOR\_ENCRYPTED AS BEGIN  SELECT DOCTOR.ID\_DOCTOR,PASSWORD\_TABLE.USER\_PASSWORD,FIRST\_NAME,LAST\_NAME,PATRONY,OFFICE,QUALIFICATION,  BDAY,DATE\_STARTED,POSITION,PHONE\_NUMBER FROM DOCTOR INNER JOIN PASSWORD\_TABLE  ON DOCTOR.ID\_DOCTOR= PASSWORD\_TABLE.ID\_DOCTOR  END;  CREATE PROCEDURE SELECT\_PASSWORD\_DOCTOR\_DECRYPTED AS BEGIN  OPEN SYMMETRIC KEY SSN\_Key\_1  DECRYPTION BY CERTIFICATE HumanResources;  SELECT DOCTOR.ID\_DOCTOR,CONVERT(nvarchar, DecryptByKey(USER\_PASSWORD)) AS 'Decrypted USER PASSWORD',FIRST\_NAME,LAST\_NAME,PATRONY,OFFICE,QUALIFICATION,  BDAY,DATE\_STARTED,POSITION,PHONE\_NUMBER FROM DOCTOR INNER JOIN PASSWORD\_TABLE  ON DOCTOR.ID\_DOCTOR= PASSWORD\_TABLE.ID\_DOCTOR  END;  CREATE PROCEDURE SELECT\_DOCTOR\_FIRST\_AND\_LAST\_NAME @FIRST\_NAME NVARCHAR(20),  @LAST\_NAME NVARCHAR(20) AS BEGIN  SELECT ID\_DOCTOR,HOSPITAL.NAME\_HOSPITAL,FIRST\_NAME,LAST\_NAME,PATRONY,OFFICE,QUALIFICATION,  BDAY,DATE\_STARTED,POSITION,PHONE\_NUMBER FROM DOCTOR  INNER JOIN HOSPITAL ON DOCTOR.ID\_HOSPITAL=HOSPITAL.ID\_HOSPITAL WHERE FIRST\_NAME=@FIRST\_NAME AND LAST\_NAME=@LAST\_NAME  END;  CREATE PROCEDURE SELECT\_DOCTOR\_ID @ID\_DOCTOR INT AS BEGIN  SELECT ID\_DOCTOR,HOSPITAL.NAME\_HOSPITAL,FIRST\_NAME,LAST\_NAME,PATRONY,OFFICE,QUALIFICATION,  BDAY,DATE\_STARTED,POSITION,PHONE\_NUMBER FROM DOCTOR  INNER JOIN HOSPITAL ON DOCTOR.ID\_HOSPITAL=HOSPITAL.ID\_HOSPITAL WHERE ID\_DOCTOR=@ID\_DOCTOR  END;  CREATE PROCEDURE SELECT\_ALL\_PATIENT\_UNMASK AS BEGIN  SELECT GENDER,FIRST\_NAME,LAST\_NAME,PATRONY,BDAY,HOME\_ADRESS,PASSPORT\_ID,HOSPITAL.NAME\_HOSPITAL  ,PLACE\_OF\_WORK\_OR\_STUDY,DISABILITY,CHRONIC\_DISEASES FROM PATIENT INNER JOIN HOSPITAL ON PATIENT.ID\_HOSPITAL=HOSPITAL.ID\_HOSPITAL  END;  CREATE PROCEDURE SELECT\_ALL\_PATIENT\_MASK AS BEGIN  EXECUTE AS USER = 'TestUser';  SELECT GENDER,FIRST\_NAME,LAST\_NAME,PATRONY,BDAY,HOME\_ADRESS,PASSPORT\_ID  ,PLACE\_OF\_WORK\_OR\_STUDY,DISABILITY,CHRONIC\_DISEASES FROM PATIENT  REVERT;  END;  CREATE PROCEDURE SELECT\_PATIENT\_AND\_PASSWORD AS BEGIN  OPEN SYMMETRIC KEY SSN\_Key\_1  DECRYPTION BY CERTIFICATE HumanResources;  SELECT GENDER,FIRST\_NAME,LAST\_NAME,PATRONY,BDAY,HOME\_ADRESS,PASSPORT\_ID,CONVERT(nvarchar, DecryptByKey(USER\_PASSWORD)) AS 'Decrypted USER PASSWORD'  ,PLACE\_OF\_WORK\_OR\_STUDY,DISABILITY,CHRONIC\_DISEASES FROM PATIENT INNER JOIN PASSWORD\_TABLE ON PATIENT.ID\_PATIENT=PASSWORD\_TABLE.ID\_PATIENT  END;  CREATE PROCEDURE SELECT\_PATIENT\_FOR\_FIRST\_AND\_LAST\_NAME @FIRST\_NAME NVARCHAR(20),@LAST\_NAME NVARCHAR(20) AS BEGIN  SELECT GENDER,FIRST\_NAME,LAST\_NAME,PATRONY,BDAY,HOME\_ADRESS,PASSPORT\_ID,HOSPITAL.NAME\_HOSPITAL  ,PLACE\_OF\_WORK\_OR\_STUDY,DISABILITY,CHRONIC\_DISEASES FROM PATIENT INNER JOIN HOSPITAL ON PATIENT.ID\_HOSPITAL=HOSPITAL.ID\_HOSPITAL  WHERE PATIENT.FIRST\_NAME=@FIRST\_NAME AND PATIENT.LAST\_NAME=@LAST\_NAME  END;  CREATE PROCEDURE SELECT\_ALL\_HOSPITAL AS BEGIN  SELECT ID\_HOSPITAL,COUNTRY,CITY,AREA,REGISTRATION\_PHONE,NAME\_HOSPITAL FROM HOSPITAL  END;  CREATE PROCEDURE SELECT\_HOSPITAL\_FOR\_NAME @NAME\_HOSPITAL NVARCHAR(60) AS BEGIN  SELECT ID\_HOSPITAL,COUNTRY,CITY,AREA,REGISTRATION\_PHONE,NAME\_HOSPITAL FROM HOSPITAL WHERE NAME\_HOSPITAL=@NAME\_HOSPITAL  END;  CREATE PROCEDURE SELECT\_HOSPITAL\_FOR\_ID @ID\_HOSPITAL INT AS BEGIN  SELECT ID\_HOSPITAL,COUNTRY,CITY,AREA,REGISTRATION\_PHONE,NAME\_HOSPITAL FROM HOSPITAL WHERE ID\_HOSPITAL=@ID\_HOSPITAL  END;  CREATE PROCEDURE SELECT\_HOSPITAL\_FOR\_COUNTRY @COUNTRY NVARCHAR(40) AS BEGIN  SELECT ID\_HOSPITAL,COUNTRY,CITY,AREA,REGISTRATION\_PHONE,NAME\_HOSPITAL FROM HOSPITAL WHERE COUNTRY=@COUNTRY  END;  CREATE PROCEDURE SELECT\_ALL\_RECEPTION AS BEGIN  SELECT SYMPTOMS,DIAGNOSIS,MEDICINES,COMMENTS,DOCTOR.FIRST\_NAME AS 'DOCTOR FIRST NAME',DOCTOR.LAST\_NAME AS 'DOCTOR LAST NAME',DOCTOR.POSITION  ,PATIENT.FIRST\_NAME AS' PATIENT FIRST NAME',PATIENT.LAST\_NAME AS 'PATIENT LAST NAME',HOSPITAL.NAME\_HOSPITAL,  TIME\_TIMETABLE.DAY\_OF\_THE\_WEEK AS 'VISIT DAY',TIME\_TIMETABLE.STARTING\_A\_PATIENT\_RECEPTION AS 'TIME VISIT' FROM RECEPTION INNER JOIN DOCTOR ON RECEPTION.ID\_DOCTOR=DOCTOR.ID\_DOCTOR  INNER JOIN PATIENT ON RECEPTION.ID\_PATIENT=PATIENT.ID\_PATIENT INNER JOIN HOSPITAL ON RECEPTION.ID\_HOSPITAL=HOSPITAL.ID\_HOSPITAL  INNER JOIN TIME\_TIMETABLE ON RECEPTION.ID\_TIME=TIME\_TIMETABLE.ID\_TIME  END;  CREATE PROCEDURE SELECT\_RECEPTION\_FOR\_PATIENT\_FIRST\_AND\_LAST\_NAME  @FIRST\_NAME NVARCHAR(20),@LAST\_NAME NVARCHAR(20) AS BEGIN  SELECT SYMPTOMS,DIAGNOSIS,MEDICINES,COMMENTS,DOCTOR.FIRST\_NAME AS 'DOCTOR FIRST NAME',DOCTOR.LAST\_NAME AS 'DOCTOR LAST NAME',DOCTOR.POSITION  ,PATIENT.FIRST\_NAME AS' PATIENT FIRST NAME',PATIENT.LAST\_NAME AS 'PATIENT LAST NAME',HOSPITAL.NAME\_HOSPITAL,  TIME\_TIMETABLE.DAY\_OF\_THE\_WEEK AS 'VISIT DAY',TIME\_TIMETABLE.STARTING\_A\_PATIENT\_RECEPTION AS 'TIME VISIT' FROM RECEPTION INNER JOIN DOCTOR ON RECEPTION.ID\_DOCTOR=DOCTOR.ID\_DOCTOR  INNER JOIN PATIENT ON RECEPTION.ID\_PATIENT=PATIENT.ID\_PATIENT INNER JOIN HOSPITAL ON RECEPTION.ID\_HOSPITAL=HOSPITAL.ID\_HOSPITAL  INNER JOIN TIME\_TIMETABLE ON RECEPTION.ID\_TIME=TIME\_TIMETABLE.ID\_TIME WHERE PATIENT.FIRST\_NAME=@FIRST\_NAME AND PATIENT.LAST\_NAME=@LAST\_NAME  END;  CREATE PROCEDURE SELECT\_RECEPTION\_FOR\_DOCTOR\_FIRST\_AND\_LAST\_NAME @FIRST\_NAME NVARCHAR(20),@LAST\_NAME NVARCHAR(20) AS BEGIN  SELECT SYMPTOMS,DIAGNOSIS,MEDICINES,COMMENTS,DOCTOR.FIRST\_NAME AS 'DOCTOR FIRST NAME',DOCTOR.LAST\_NAME AS 'DOCTOR LAST NAME',DOCTOR.POSITION  ,PATIENT.FIRST\_NAME AS' PATIENT FIRST NAME',PATIENT.LAST\_NAME AS 'PATIENT LAST NAME',HOSPITAL.NAME\_HOSPITAL,  TIME\_TIMETABLE.DAY\_OF\_THE\_WEEK AS 'VISIT DAY',TIME\_TIMETABLE.STARTING\_A\_PATIENT\_RECEPTION AS 'TIME VISIT' FROM RECEPTION INNER JOIN DOCTOR ON RECEPTION.ID\_DOCTOR=DOCTOR.ID\_DOCTOR  INNER JOIN PATIENT ON RECEPTION.ID\_PATIENT=PATIENT.ID\_PATIENT INNER JOIN HOSPITAL ON RECEPTION.ID\_HOSPITAL=HOSPITAL.ID\_HOSPITAL  INNER JOIN TIME\_TIMETABLE ON RECEPTION.ID\_TIME=TIME\_TIMETABLE.ID\_TIME WHERE DOCTOR.FIRST\_NAME=@FIRST\_NAME AND DOCTOR.LAST\_NAME=@LAST\_NAME  END;  CREATE PROCEDURE SELECT\_RECEPTION\_FOR\_TIME @TIME NVARCHAR(20) AS BEGIN  SELECT SYMPTOMS,DIAGNOSIS,MEDICINES,COMMENTS,DOCTOR.FIRST\_NAME AS 'DOCTOR FIRST NAME',DOCTOR.LAST\_NAME AS 'DOCTOR LAST NAME',DOCTOR.POSITION  ,PATIENT.FIRST\_NAME AS' PATIENT FIRST NAME',PATIENT.LAST\_NAME AS 'PATIENT LAST NAME',HOSPITAL.NAME\_HOSPITAL,  TIME\_TIMETABLE.DAY\_OF\_THE\_WEEK AS 'VISIT DAY',TIME\_TIMETABLE.STARTING\_A\_PATIENT\_RECEPTION AS 'TIME VISIT' FROM RECEPTION INNER JOIN DOCTOR ON RECEPTION.ID\_DOCTOR=DOCTOR.ID\_DOCTOR  INNER JOIN PATIENT ON RECEPTION.ID\_PATIENT=PATIENT.ID\_PATIENT INNER JOIN HOSPITAL ON RECEPTION.ID\_HOSPITAL=HOSPITAL.ID\_HOSPITAL  INNER JOIN TIME\_TIMETABLE ON RECEPTION.ID\_TIME=TIME\_TIMETABLE.ID\_TIME WHERE TIME\_TIMETABLE.STARTING\_A\_PATIENT\_RECEPTION=@TIME  END;  CREATE PROCEDURE SELECT\_RECEPTION\_FOR\_DATE @TIME DATE AS BEGIN  SELECT SYMPTOMS,DIAGNOSIS,MEDICINES,COMMENTS,DOCTOR.FIRST\_NAME AS 'DOCTOR FIRST NAME',DOCTOR.LAST\_NAME AS 'DOCTOR LAST NAME',DOCTOR.POSITION  ,PATIENT.FIRST\_NAME AS' PATIENT FIRST NAME',PATIENT.LAST\_NAME AS 'PATIENT LAST NAME',HOSPITAL.NAME\_HOSPITAL,  TIME\_TIMETABLE.DAY\_OF\_THE\_WEEK AS 'VISIT DAY',TIME\_TIMETABLE.STARTING\_A\_PATIENT\_RECEPTION AS 'TIME VISIT' FROM RECEPTION INNER JOIN DOCTOR ON RECEPTION.ID\_DOCTOR=DOCTOR.ID\_DOCTOR  INNER JOIN PATIENT ON RECEPTION.ID\_PATIENT=PATIENT.ID\_PATIENT INNER JOIN HOSPITAL ON RECEPTION.ID\_HOSPITAL=HOSPITAL.ID\_HOSPITAL  INNER JOIN TIME\_TIMETABLE ON RECEPTION.ID\_TIME=TIME\_TIMETABLE.ID\_TIME WHERE TIME\_TIMETABLE.DAY\_OF\_THE\_WEEK=@TIME  END;  CREATE PROCEDURE SELECT\_RECEPTION\_FOR\_DATE\_AND\_TIME @TIME DATE,@TIME1 NVARCHAR(20) AS BEGIN  SELECT SYMPTOMS,DIAGNOSIS,MEDICINES,COMMENTS,DOCTOR.FIRST\_NAME AS 'DOCTOR FIRST NAME',DOCTOR.LAST\_NAME AS 'DOCTOR LAST NAME',DOCTOR.POSITION  ,PATIENT.FIRST\_NAME AS' PATIENT FIRST NAME',PATIENT.LAST\_NAME AS 'PATIENT LAST NAME',HOSPITAL.NAME\_HOSPITAL,  TIME\_TIMETABLE.DAY\_OF\_THE\_WEEK AS 'VISIT DAY',TIME\_TIMETABLE.STARTING\_A\_PATIENT\_RECEPTION AS 'TIME VISIT' FROM RECEPTION INNER JOIN DOCTOR ON RECEPTION.ID\_DOCTOR=DOCTOR.ID\_DOCTOR  INNER JOIN PATIENT ON RECEPTION.ID\_PATIENT=PATIENT.ID\_PATIENT INNER JOIN HOSPITAL ON RECEPTION.ID\_HOSPITAL=HOSPITAL.ID\_HOSPITAL  INNER JOIN TIME\_TIMETABLE ON RECEPTION.ID\_TIME=TIME\_TIMETABLE.ID\_TIME WHERE TIME\_TIMETABLE.DAY\_OF\_THE\_WEEK=@TIME AND  TIME\_TIMETABLE.STARTING\_A\_PATIENT\_RECEPTION=@TIME1  END;  CREATE PROCEDURE SELECT\_RECEPTION\_FOR\_DATE\_AND\_FIRST\_AND\_LAST\_NAME\_DOCTOR  @TIME DATE,@FIRST\_NAME NVARCHAR(20),@LAST\_NAME NVARCHAR(20) AS BEGIN  SELECT SYMPTOMS,DIAGNOSIS,MEDICINES,COMMENTS,DOCTOR.FIRST\_NAME AS 'DOCTOR FIRST NAME',DOCTOR.LAST\_NAME AS 'DOCTOR LAST NAME',DOCTOR.POSITION  ,PATIENT.FIRST\_NAME AS' PATIENT FIRST NAME',PATIENT.LAST\_NAME AS 'PATIENT LAST NAME',HOSPITAL.NAME\_HOSPITAL,  TIME\_TIMETABLE.DAY\_OF\_THE\_WEEK AS 'VISIT DAY',TIME\_TIMETABLE.STARTING\_A\_PATIENT\_RECEPTION AS 'TIME VISIT' FROM RECEPTION INNER JOIN DOCTOR ON RECEPTION.ID\_DOCTOR=DOCTOR.ID\_DOCTOR  INNER JOIN PATIENT ON RECEPTION.ID\_PATIENT=PATIENT.ID\_PATIENT INNER JOIN HOSPITAL ON RECEPTION.ID\_HOSPITAL=HOSPITAL.ID\_HOSPITAL  INNER JOIN TIME\_TIMETABLE ON RECEPTION.ID\_TIME=TIME\_TIMETABLE.ID\_TIME WHERE TIME\_TIMETABLE.DAY\_OF\_THE\_WEEK=@TIME AND  DOCTOR.FIRST\_NAME=@FIRST\_NAME AND DOCTOR.LAST\_NAME=@LAST\_NAME  END;  CREATE PROCEDURE DEL\_RECEPTION\_BY\_ID @ID\_RECEPTION INT AS DECLARE @ID\_TIME INT BEGIN  BEGIN TRY  IF NOT EXISTS(SELECT ID\_RECEPTION FROM RECEPTION WHERE ID\_RECEPTION=@ID\_RECEPTION)  BEGIN PRINT 'ТАКОГО ТАЛОНА НЕТУ' END;  ELSE  BEGIN  SET @ID\_TIME=(SELECT ID\_TIME FROM RECEPTION WHERE ID\_RECEPTION=@ID\_RECEPTION)  DELETE RECEPTION WHERE ID\_RECEPTION=@ID\_RECEPTION  DELETE TIME\_TIMETABLE WHERE ID\_TIME=@ID\_TIME  PRINT 'УДАЛЕНИЕ ПРОШЛО УСПЕШНО'  END;  END TRY  BEGIN CATCH  PRINT 'Error ' + CONVERT(VARCHAR, ERROR\_NUMBER()) + ':' + ERROR\_MESSAGE()  END CATCH  END;  CREATE PROCEDURE DEL\_PATIENT\_BY\_ID  @ID\_PATIENT INT  AS BEGIN  BEGIN TRY  IF NOT EXISTS(SELECT ID\_PATIENT FROM PATIENT WHERE ID\_PATIENT=@ID\_PATIENT)  BEGIN  PRINT 'ТАКОГО ЧЕЛОВЕКА НЕТУ'  END;  ELSE  BEGIN  DELETE PASSWORD\_TABLE WHERE ID\_PATIENT=@ID\_PATIENT  DELETE RECEPTION WHERE ID\_PATIENT=@ID\_PATIENT  DELETE PATIENT WHERE ID\_PATIENT=@ID\_PATIENT  EXEC DEL\_TIME  PRINT 'УДАЛЕНИЕ ПРОШЛО УСПЕШНО'  END;  END TRY  BEGIN CATCH  PRINT 'Error ' + CONVERT(VARCHAR, ERROR\_NUMBER()) + ':' + ERROR\_MESSAGE()  END CATCH  END;  CREATE PROCEDURE DEL\_TIME AS BEGIN BEGIN TRY  IF NOT EXISTS(SELECT ID\_TIME FROM TIME\_TIMETABLE WHERE DAY\_OF\_THE\_WEEK<GETDATE())  BEGIN  PRINT 'C ВРЕМЕНЕМ ВСЁ ОКЕЙ НИЧЕГО НЕ НАДО УДАЛЯТЬ'  END;  ELSE  BEGIN  DELETE TIME\_TIMETABLE WHERE DAY\_OF\_THE\_WEEK<GETDATE()  PRINT 'УДАЛЕНИЕ ПРОШЛО УСПЕШНО'  END;  END TRY  BEGIN CATCH  PRINT 'Error ' + CONVERT(VARCHAR, ERROR\_NUMBER()) + ':' + ERROR\_MESSAGE()  END CATCH  END;  CREATE PROCEDURE DEL\_ALL\_TIME AS BEGIN BEGIN TRY  IF NOT EXISTS(SELECT ID\_TIME FROM TIME\_TIMETABLE )  BEGIN  PRINT 'НЕЧЕГО УДАЛЯТЬ'  END;  ELSE  BEGIN  DELETE TIME\_TIMETABLE  PRINT 'УДАЛЕНИЕ ПРОШЛО УСПЕШНО'  END;  END TRY  BEGIN CATCH  PRINT 'Error ' + CONVERT(VARCHAR, ERROR\_NUMBER()) + ':' + ERROR\_MESSAGE()  END CATCH  END;  CREATE PROCEDURE DEL\_DOCTOR\_BY\_ID @ID\_DOCTOR INT AS BEGIN BEGIN TRY  IF NOT EXISTS(SELECT ID\_DOCTOR FROM DOCTOR WHERE ID\_DOCTOR=@ID\_DOCTOR)  BEGIN  PRINT 'ТАКОГО ВРАЧА НЕТУ'  END;  ELSE  BEGIN  DELETE PASSWORD\_TABLE WHERE ID\_DOCTOR=@ID\_DOCTOR  DELETE RECEPTION WHERE ID\_DOCTOR=@ID\_DOCTOR  DELETE TIME\_TIMETABLE WHERE ID\_DOCTOR=@ID\_DOCTOR  DELETE SPECIALTY WHERE ID\_DOCTOR=@ID\_DOCTOR  DELETE DOCTOR WHERE ID\_DOCTOR=@ID\_DOCTOR  PRINT 'УДАЛЕНИЕ ПРОШЛО УСПЕШНО'  END;  END TRY  BEGIN CATCH  PRINT 'Error ' + CONVERT(VARCHAR, ERROR\_NUMBER()) + ':' + ERROR\_MESSAGE()  END CATCH  END;  CREATE PROCEDURE DEL\_DEPARTMENT\_BY\_ID @ID\_DEPARTMENT INT AS BEGIN BEGIN TRY  IF NOT EXISTS(SELECT ID\_DEPARTMENT FROM DEPARTMENT WHERE ID\_DEPARTMENT=@ID\_DEPARTMENT)  BEGIN  PRINT 'ТАКОГО ОТДЕЛЕНИЯ НЕТУ'  END;  ELSE  BEGIN  DECLARE @ID\_DOCTOR INT  DECLARE CUR\_DEL\_DEPARTMENT CURSOR FOR SELECT ID\_DOCTOR FROM DOCTOR WHERE ID\_DEPARTMENT=@ID\_DEPARTMENT  OPEN CUR\_DEL\_DEPARTMENT  FETCH NEXT FROM CUR\_DEL\_DEPARTMENT INTO @ID\_DOCTOR  WHILE @@FETCH\_STATUS = 0  BEGIN  exec DEL\_DOCTOR\_BY\_ID @ID\_DOCTOR  FETCH NEXT FROM CUR\_DEL\_DEPARTMENT INTO @ID\_DOCTOR  END  CLOSE CUR\_DEL\_DEPARTMENT  DEALLOCATE CUR\_DEL\_DEPARTMENT  DELETE DEPARTMENT WHERE ID\_DEPARTMENT=@ID\_DEPARTMENT  PRINT 'УДАЛЕНИЕ ПРОШЛО УСПЕШНО'  END;  END TRY  BEGIN CATCH  PRINT 'Error ' + CONVERT(VARCHAR, ERROR\_NUMBER()) + ':' + ERROR\_MESSAGE()  END CATCH  END;  CREATE PROCEDURE DEL\_HOSPITAL\_BY\_ID @ID\_HOSPITAL INT AS BEGIN BEGIN TRY  IF NOT EXISTS(SELECT ID\_HOSPITAL FROM HOSPITAL WHERE ID\_HOSPITAL=@ID\_HOSPITAL)  BEGIN PRINT 'ТАКОЙ ПОЛИКЛИНИКИ НЕТУ' END;  ELSE  BEGIN  DECLARE @ID\_DEPARTMENT INT  DECLARE CUR\_DEL\_HOSPITAL CURSOR FOR SELECT ID\_DEPARTMENT FROM DEPARTMENT WHERE ID\_HOSPITAL=@ID\_HOSPITAL  OPEN CUR\_DEL\_HOSPITAL  FETCH NEXT FROM CUR\_DEL\_HOSPITAL INTO @ID\_DEPARTMENT  WHILE @@FETCH\_STATUS = 0  BEGIN  exec DEL\_DEPARTMENT\_BY\_ID @ID\_HOSPITAL  FETCH NEXT FROM CUR\_DEL\_HOSPITAL INTO @ID\_DEPARTMENT  END  CLOSE CUR\_DEL\_HOSPITAL  DEALLOCATE CUR\_DEL\_HOSPITAL  DELETE HOSPITAL WHERE ID\_HOSPITAL=@ID\_HOSPITAL  PRINT 'УДАЛЕНИЕ ПРОШЛО УСПЕШНО'  END;  END TRY  BEGIN CATCH  PRINT 'Error ' + CONVERT(VARCHAR, ERROR\_NUMBER()) + ':' + ERROR\_MESSAGE()  END CATCH  END;  CREATE PROCEDURE UPDATE\_HOSPITAL\_INPUT\_NAME @COUNTRY NVARCHAR(40),@CITY NVARCHAR(40),@AREA NVARCHAR(40),@NAME\_HOSPITAL NVARCHAR(60),@REGISTRATION\_PHONE NVARCHAR(10)  AS BEGIN DECLARE @ID\_HOSPITAL INT BEGIN TRY  IF NOT EXISTS(SELECT ID\_HOSPITAL FROM HOSPITAL WHERE NAME\_HOSPITAL=@NAME\_HOSPITAL)  BEGIN PRINT 'ТАКОЙ ПОЛИКЛИНИКИ НЕТ ' END;  ELSE  BEGIN  SET @ID\_HOSPITAL=(SELECT ID\_HOSPITAL FROM HOSPITAL WHERE NAME\_HOSPITAL=@NAME\_HOSPITAL)  UPDATE HOSPITAL SET CITY=@CITY, COUNTRY=@COUNTRY,AREA=@AREA,REGISTRATION\_PHONE=@REGISTRATION\_PHONE WHERE ID\_HOSPITAL =@ID\_HOSPITAL  PRINT 'ОПЕРАЦИЯ ПРОВЕДЕНА УСПЕШНО'  END;  END TRY  BEGIN CATCH  PRINT 'Error ' + CONVERT(VARCHAR, ERROR\_NUMBER()) + ':' + ERROR\_MESSAGE()  END CATCH  END;  CREATE PROCEDURE UPDATE\_DEPARTMENT\_INPUT\_NAME@NAME\_DEPARTMENT NVARCHAR(30),  @NAME\_DEPARTMENT1 NVARCHAR(30) AS BEGIN BEGIN TRY  DECLARE @ID\_HOSPITAL INT  IF NOT EXISTS(SELECT ID\_DEPARTMENT FROM DEPARTMENT WHERE NAME\_DEPARTMENT=@NAME\_DEPARTMENT)  BEGIN PRINT 'ТАКОГО ДЕПАРТАМЕНТА НЕТ' END;  ELSE  BEGIN  IF NOT EXISTS(SELECT ID\_DEPARTMENT FROM DEPARTMENT WHERE NAME\_DEPARTMENT=@NAME\_DEPARTMENT1)  BEGIN  UPDATE DEPARTMENT SET NAME\_DEPARTMENT=@NAME\_DEPARTMENT1 WHERE NAME\_DEPARTMENT=@NAME\_DEPARTMENT  PRINT 'ОПЕРАЦИЯ ПРОВЕДЕНА УСПЕШНО'  END  ELSE  BEGIN  PRINT 'ДЕПАРТАМЕНТ С ТАКИМ ИМЕНЕМ СУЩЕСТВУЕТ'  END;  END;  END TRY BEGIN CATCH  PRINT 'Error ' + CONVERT(VARCHAR, ERROR\_NUMBER()) + ':' + ERROR\_MESSAGE()  END CATCH END;  CREATE PROCEDURE UPDATE\_DOCTOR @ID\_DOCTOR INT,@FIRST\_NAME NVARCHAR(20),@LAST\_NAME NVARCHAR(20),@PATRONY NVARCHAR(20),  @OFFICE NVARCHAR(10),@QUALIFICATION NVARCHAR(30),@BDAY DATE,@DATE\_STARTED DATE,@POSITION NVARCHAR(20),@PHONE\_NUMBER NVARCHAR(20)=NULL,@PASSWORD\_USER NVARCHAR(60)='123',@SPECIALTY\_NAME NVARCHAR(50) AS DECLARE @FLAG INT=0 BEGIN BEGIN TRY  IF NOT EXISTS(SELECT ID\_DOCTOR FROM DOCTOR WHERE ID\_DOCTOR=@ID\_DOCTOR)  BEGIN PRINT 'ТАКОГО ВРАЧА НЕТ' END;  ELSE  BEGIN  IF @POSITION='гл. врач' OR @POSITION='зам. отделения'  BEGIN PRINT 'ТАКИЕ ДОЛЖНОСТИ НЕЛЬЗЯ ИЗМЕНИТЬ!!!' END  ELSE  BEGIN  IF NOT EXISTS(SELECT ID\_DOCTOR FROM DOCTOR WHERE  FIRST\_NAME=@FIRST\_NAME AND LAST\_NAME=@LAST\_NAME AND PATRONY=@PATRONY AND OFFICE=@OFFICE AND QUALIFICATION=@QUALIFICATION AND  BDAY=@BDAY AND DATE\_STARTED=@DATE\_STARTED)  BEGIN  UPDATE DOCTOR SET POSITION=@POSITION,FIRST\_NAME=@FIRST\_NAME,LAST\_NAME=@LAST\_NAME,PATRONY=@PATRONY,OFFICE=@OFFICE,QUALIFICATION=@QUALIFICATION,BDAY=@BDAY,DATE\_STARTED=@DATE\_STARTED,PHONE\_NUMBER=@PHONE\_NUMBER  OPEN SYMMETRIC KEY SSN\_Key\_1  DECRYPTION BY CERTIFICATE HumanResources;  UPDATE PASSWORD\_TABLE  SET USER\_PASSWORD = EncryptByKey(Key\_GUID('SSN\_Key\_1'), @PASSWORD\_USER) where ID\_DOCTOR=@ID\_DOCTOR ;  UPDATE SPECIALTY SET SPECIALTY\_NAME=@SPECIALTY\_NAME WHERE SPECIALTY.ID\_DOCTOR=@ID\_DOCTOR  PRINT 'ОПЕРАЦИЯ ПРОВЕДЕНА УСПЕШНО'  END  ELSE  BEGIN PRINT 'НЕВЕРНЫЕ ДАННЫЕ' END;  END  END;  END TRY BEGIN CATCH  PRINT 'Error ' + CONVERT(VARCHAR, ERROR\_NUMBER()) + ':' + ERROR\_MESSAGE()  END CATCH END;  CREATE PROCEDURE UPDATE\_PATIENT @ID\_PATIENT INT,@GENDER CHAR(1),@FIRST\_NAME NVARCHAR(20),@LAST\_NAME NVARCHAR(20),@PATRONY NVARCHAR(20),@BDAY DATE,@HOME\_ADRESS NVARCHAR(40),@PASSPORT\_ID NVARCHAR(30)=NULL,@NAME\_HOSPITAL NVARCHAR(60),@PLACE\_OF\_WORK\_OR\_STUDY NVARCHAR(40)=NULL,@DISABILITY NVARCHAR(40)=NULL,@CHRONIC\_DISEASES NVARCHAR(40)=NULL,@PASSWORD\_USER NVARCHAR(60) AS BEGIN DECLARE @ID\_HOSPITAL INT BEGIN TRY  IF NOT EXISTS(SELECT ID\_PATIENT FROM PATIENT WHERE ID\_PATIENT=@ID\_PATIENT) OR (@GENDER!='M' AND @GENDER!='W') OR  NOT EXISTS(SELECT ID\_HOSPITAL FROM HOSPITAL WHERE NAME\_HOSPITAL =@NAME\_HOSPITAL )  BEGIN  PRINT 'ТАКОГО ПАЦИЕНТА НЕТ ИЛИ НЕВЕРНО УКАЗАН ПОЛ ИЛИ НЕВЕРНО УКАЗАНА ПОЛИКЛИНИКА'  END;  ELSE  BEGIN  SET @ID\_HOSPITAL=(SELECT ID\_HOSPITAL FROM HOSPITAL WHERE NAME\_HOSPITAL=@NAME\_HOSPITAL);  IF NOT EXISTS(SELECT ID\_PATIENT FROM PATIENT WHERE GENDER=@GENDER AND FIRST\_NAME=@FIRST\_NAME AND LAST\_NAME=@LAST\_NAME AND PATRONY=@PATRONY AND BDAY=@BDAY AND HOME\_ADRESS=@HOME\_ADRESS  AND PASSPORT\_ID=@PASSPORT\_ID )  BEGIN  UPDATE PATIENT SET GENDER=@GENDER,FIRST\_NAME=@FIRST\_NAME,LAST\_NAME=@LAST\_NAME,PATRONY=@PATRONY,BDAY=@BDAY, HOME\_ADRESS=@HOME\_ADRESS,PASSPORT\_ID=@PASSPORT\_ID,ID\_HOSPITAL=@ID\_HOSPITAL,PLACE\_OF\_WORK\_OR\_STUDY=@PLACE\_OF\_WORK\_OR\_STUDY,  DISABILITY=@DISABILITY,CHRONIC\_DISEASES=@CHRONIC\_DISEASES WHERE ID\_PATIENT=@ID\_PATIENT  OPEN SYMMETRIC KEY SSN\_Key\_1  DECRYPTION BY CERTIFICATE HumanResources;  UPDATE PASSWORD\_TABLE  SET USER\_PASSWORD = EncryptByKey(Key\_GUID('SSN\_Key\_1'), @PASSWORD\_USER) where ID\_PATIENT=@ID\_PATIENT ;  PRINT 'ОПЕРАЦИЯ ПРОВЕДЕНА УСПЕШНО'  END;  ELSE  BEGIN  PRINT 'ТАКОЙ ЧЕЛОВЕК УЖЕ ЕСТЬ'  END;  END;  END TRY  BEGIN CATCH  PRINT 'Error ' + CONVERT(VARCHAR, ERROR\_NUMBER()) + ':' + ERROR\_MESSAGE()  END CATCH  END;  CREATE PROCEDURE UPDATE\_RECEPTION @ID\_RECEPTION INT,@ID\_DOCTOR INT,@ID\_PATIENT INT,@ID\_HOSPITAL INT,@ID\_TIME INT,@TIME NVARCHAR(20),@DATE DATE,@SYMPTOMS NVARCHAR(100)=NULL,@DIAGNOSIS NVARCHAR(100),@MEDICINES NVARCHAR(100),@COMMENTS NVARCHAR(200)=NULL AS DECLARE @FLAG INT =0; BEGIN BEGIN TRY  IF NOT EXISTS(SELECT ID\_RECEPTION FROM RECEPTION WHERE ID\_RECEPTION=@ID\_RECEPTION) OR  NOT EXISTS(SELECT ID\_DOCTOR FROM DOCTOR WHERE ID\_DOCTOR=@ID\_DOCTOR) OR  NOT EXISTS(SELECT ID\_PATIENT FROM PATIENT WHERE ID\_PATIENT=@ID\_PATIENT) OR  NOT EXISTS(SELECT ID\_HOSPITAL FROM HOSPITAL WHERE ID\_HOSPITAL=@ID\_HOSPITAL) OR  NOT EXISTS(SELECT ID\_TIME FROM TIME\_TIMETABLE WHERE ID\_TIME=@ID\_TIME)  BEGIN  PRINT 'ТАКОГО ТАЛОНА|ВРАЧА|ПАЦИЕНТА|ПОЛИКЛИНИКИ|ВРЕМЕНИ НЕТ'  END;  ELSE  BEGIN  IF(GETDATE()>@DATE) OR DATENAME(weekday, @DATE)='воскресенье' OR DATENAME(weekday, @DATE)='суббота' OR  (DATENAME(hour, @TIME)=13 OR DATENAME(hour, @TIME)=14 OR (DATENAME(hour, @TIME)<10 OR DATENAME(hour, @TIME)>17))  BEGIN  SET @FLAG=1  PRINT 'НЕВЕРНАЯ ДАТА ИЛИ ВРЕМЯ'  END  IF @FLAG!=1  BEGIN  PRINT 'ВСЁ ГУД'  UPDATE TIME\_TIMETABLE SET ID\_DOCTOR=@ID\_DOCTOR,DAY\_OF\_THE\_WEEK=@DATE,STARTING\_A\_PATIENT\_RECEPTION=@TIME  WHERE ID\_TIME=@ID\_TIME  UPDATE RECEPTION SET ID\_DOCTOR=@ID\_DOCTOR,ID\_PATIENT=@ID\_PATIENT,ID\_HOSPITAL=@ID\_HOSPITAL,ID\_TIME=@ID\_TIME, SYMPTOMS=@SYMPTOMS,DIAGNOSIS=@DIAGNOSIS,MEDICINES=@MEDICINES,COMMENTS=@COMMENTS WHERE ID\_RECEPTION=@ID\_RECEPTION  END;  END  END TRY  BEGIN CATCH  PRINT 'Error ' + CONVERT(VARCHAR, ERROR\_NUMBER()) + ':' + ERROR\_MESSAGE()  END CATCH  END;  CREATE PROCEDURE CREATE\_STRING\_60@SIZE integer, @NAME char(60) OUTPUT AS BEGIN  SET @NAME = (  SELECT  c1 AS [text()]  FROM  (  SELECT TOP (60) c1  FROM  (VALUES ('A'), ('B'), ('C'), ('D'), ('E'), ('F'), ('G'), ('H'), ('I'), ('J'), ('K'), ('L'), ('M'), ('N'), ('O'), ('P'), ('Q'), ('R'), ('S'), ('T'), ('U'), ('V'), ('W'), ('X'), ('Y'), ('Z')) AS T1(c1)ORDER BY ABS(CHECKSUM(NEWID()))) AS T2  FOR XML PATH(''));  END  CREATE PROCEDURE INSERT\_HOSPITAL @SIZE INT AS Begin  WHILE @SIZE > 0  BEGIN  DECLARE @NAME\_HOSPITAL NVARCHAR(60)  EXEC CREATE\_STRING\_10 60,@NAME\_HOSPITAL OUTPUT  EXEC ADD\_HOSPITAL 'Belarus','f','f',@NAME\_HOSPITAL,'--'  SET @SIZE = @SIZE - 1  END;  END;  CREATE PROCEDURE IMPORT\_HOSPITAL AS BEGIN DECLARE @HOSPITAL VARCHAR(max);  SELECT @HOSPITAL =  BulkColumn  FROM OPENROWSET(BULK 'C:\ycheba\3course1sem\курсач\_бд\Hospital.json', Single\_Blob) JSON;  PRINT @HOSPITAL  IF (ISJSON(@HOSPITAL) = 1)  BEGIN  DECLARE @COUNTRY NVARCHAR(40)  DECLARE @CITY NVARCHAR(40)  DECLARE @AREA NVARCHAR(40)  DECLARE @NAME\_HOSPITAL NVARCHAR(60)  DECLARE @REGISTRATION\_PHONE NVARCHAR(10)  DECLARE HOSPITAL\_CURSOR CURSOR FOR  SELECT \*  FROM OPENJSON(@HOSPITAL, '$.HOSPITAL')  WITH(COUNTRY NVARCHAR(40) '$.COUNTRY',CITY NVARCHAR(40) '$.CITY',AREA NVARCHAR(40) '$.AREA',NAME\_HOSPITAL NVARCHAR(60) '$.NAME\_HOSPITAL',REGISTRATION\_PHONE NVARCHAR(10) '$.REGISTRATION\_PHONE');OPEN HOSPITAL\_CURSOR;  FETCH NEXT FROM HOSPITAL\_CURSOR INTO @COUNTRY, @CITY, @AREA, @NAME\_HOSPITAL,@REGISTRATION\_PHONE;  WHILE @@FETCH\_STATUS = 0  BEGIN  EXEC ADD\_HOSPITAL @COUNTRY,@CITY,@AREA,@NAME\_HOSPITAL,@REGISTRATION\_PHONE  FETCH NEXT FROM HOSPITAL\_CURSOR INTO @COUNTRY, @CITY, @AREA, @NAME\_HOSPITAL,@REGISTRATION\_PHONE;  END;  CLOSE HOSPITAL\_CURSOR;  DEALLOCATE HOSPITAL\_CURSOR;  ------------------------------------------------------------------  DECLARE @NAME\_HOSPITAL1 NVARCHAR(60)  DECLARE @NAME\_DEPARTMENT NVARCHAR(30)  DECLARE DEPARTMENT\_CURSOR CURSOR FOR  SELECT \*  FROM OPENJSON(@HOSPITAL, '$.DEPARTMENT')  WITH(NAME\_HOSPITAL1 NVARCHAR(60) '$.NAME\_HOSPITAL',NAME\_DEPARTMENT NVARCHAR(30) '$.NAME\_DEPARTMENT');OPEN DEPARTMENT\_CURSOR;  FETCH NEXT FROM DEPARTMENT\_CURSOR INTO @NAME\_HOSPITAL1,@NAME\_DEPARTMENT  WHILE @@FETCH\_STATUS = 0  BEGIN  EXEC ADD\_DEPARTMENT @NAME\_DEPARTMENT,@NAME\_HOSPITAL1  FETCH NEXT FROM DEPARTMENT\_CURSOR INTO @NAME\_HOSPITAL1,@NAME\_DEPARTMENT  END;  CLOSE DEPARTMENT\_CURSOR;  DEALLOCATE DEPARTMENT\_CURSOR;  ------------------------------------------------------------------  DECLARE @NAME\_DEPARTMENT1 NVARCHAR(30)=NULL, @FIRST\_NAME NVARCHAR(20),@LAST\_NAME NVARCHAR(20),@PATRONY NVARCHAR(20),@OFFICE NVARCHAR(10),@QUALIFICATION NVARCHAR(30),@BDAY DATE,@DATE\_STARTED DATE,@POSITION NVARCHAR(20),@PHONE\_NUMBER NVARCHAR(20)=NULL,@NAME\_HOSPITAL2 NVARCHAR(60),@PASSWORD\_USER NVARCHAR(60)='123',@SPECIALTY\_NAME NVARCHAR(50)  DECLARE DOCTOR\_CURSOR CURSOR FOR  SELECT \* FROM OPENJSON(@HOSPITAL, '$.DOCTOR')WITH(NAME\_DEPARTMENT1 NVARCHAR(30) '$.NAME\_DEPARTMENT',FIRST\_NAME NVARCHAR(20) '$.FIRST\_NAME',LAST\_NAME NVARCHAR(20) '$.LAST\_NAME',PATRONY NVARCHAR(20) '$.PATRONY',OFFICE NVARCHAR(10) '$.OFFICE',QUALIFICATION NVARCHAR(30) '$.QUALIFICATION',BDAY DATE '$.BDAY',DATE\_STARTED DATE '$.DATE\_STARTED',POSITION NVARCHAR(20) '$.POSITION',PHONE\_NUMBER NVARCHAR(20) '$.PHONE\_NUMBER',NAME\_HOSPITAL2 NVARCHAR(60) '$.NAME\_HOSPITAL',PASSWORD\_USER NVARCHAR(60) '$.PASSWORD\_USER',SPECIALTY\_NAME NVARCHAR(50) '$.SPECIALTY\_NAME');  OPEN DOCTOR\_CURSOR;  FETCH NEXT FROM DOCTOR\_CURSOR INTO @NAME\_DEPARTMENT1,@FIRST\_NAME,@LAST\_NAME,@PATRONY,@OFFICE,@QUALIFICATION, @BDAY,@DATE\_STARTED,@POSITION,@PHONE\_NUMBER,@NAME\_HOSPITAL2,@PASSWORD\_USER,@SPECIALTY\_NAME  WHILE @@FETCH\_STATUS = 0  BEGIN  IF @NAME\_DEPARTMENT1 IS NULL  BEGIN  EXEC ADD\_DOCTOR NULL,@FIRST\_NAME,@LAST\_NAME,@PATRONY,@OFFICE,@QUALIFICATION ,@BDAY,@DATE\_STARTED,'гл.врач',@PHONE\_NUMBER,@NAME\_HOSPITAL2,@PASSWORD\_USER,@SPECIALTY\_NAME  END  ELSE  BEGIN  EXEC ADD\_DOCTOR @NAME\_DEPARTMENT=@NAME\_DEPARTMENT1,@FIRST\_NAME=@FIRST\_NAME,@LAST\_NAME=@LAST\_NAME,@PATRONY=@PATRONY,@OFFICE=@OFFICE,@QUALIFICATION=@QUALIFICATION  ,@BDAY=@BDAY,@DATE\_STARTED=@DATE\_STARTED, @POSITION=@POSITION,@PHONE\_NUMBER=@PHONE\_NUMBER,@NAME\_HOSPITAL=@NAME\_HOSPITAL2,@PASSWORD\_USER=@PASSWORD\_USER,@SPECIALTY\_NAME=@SPECIALTY\_NAME  END  SET @NAME\_DEPARTMENT1=NULL  FETCH NEXT FROM DOCTOR\_CURSOR INTO @NAME\_DEPARTMENT1,@FIRST\_NAME,@LAST\_NAME,@PATRONY,@OFFICE,@QUALIFICATION,  @BDAY,@DATE\_STARTED,@POSITION,@PHONE\_NUMBER,@NAME\_HOSPITAL2,@PASSWORD\_USER,@SPECIALTY\_NAME  END;  CLOSE DOCTOR\_CURSOR;  DEALLOCATE DOCTOR\_CURSOR;  ------------------------------------------------------------------  DECLARE @GENDER CHAR(1),@FIRST\_NAME1 NVARCHAR(20),@LAST\_NAME1 NVARCHAR(20),@PATRONY1 NVARCHAR(20),@BDAY1 DATE,@HOME\_ADRESS NVARCHAR(40),@PASSPORT\_ID NVARCHAR(30)=NULL,@NAME\_HOSPITAL3 NVARCHAR(60),@PLACE\_OF\_WORK\_OR\_STUDY NVARCHAR(40)=NULL,@DISABILITY NVARCHAR(40)=NULL,@CHRONIC\_DISEASES NVARCHAR(40)=NULL,@PASSWORD\_USER1 NVARCHAR(60)='123'  DECLARE PATIENT\_CURSOR CURSOR FOR SELECT \* FROM OPENJSON(@HOSPITAL, '$.PATIENT')  WITH(GENDER CHAR(1) '$.GENDER',FIRST\_NAME1 NVARCHAR(20) '$.FIRST\_NAME',LAST\_NAME1 NVARCHAR(20) '$.LAST\_NAME',PATRONY1 NVARCHAR(20) '$.PATRONY',BDAY1 DATE '$.BDAY',HOME\_ADRESS NVARCHAR(40) '$.HOME\_ADRESS',PASSPORT\_ID NVARCHAR(30) '$.PASSPORT\_ID',NAME\_HOSPITAL3 NVARCHAR(60) '$.NAME\_HOSPITAL',PLACE\_OF\_WORK\_OR\_STUDY NVARCHAR(40) '$.PLACE\_OF\_WORK\_OR\_STUDY',DISABILITY NVARCHAR(40) '$.DISABILITY',CHRONIC\_DISEASES NVARCHAR(40) '$.CHRONIC\_DISEASES',PASSWORD\_USER1 NVARCHAR(60) '$.PASSWORD\_USER');  OPEN PATIENT\_CURSOR;  FETCH NEXT FROM PATIENT\_CURSOR INTO @GENDER,@FIRST\_NAME1,@LAST\_NAME1,@PATRONY1  ,@BDAY1,@HOME\_ADRESS,@PASSPORT\_ID,@NAME\_HOSPITAL3,@PLACE\_OF\_WORK\_OR\_STUDY,@DISABILITY,@CHRONIC\_DISEASES,@PASSWORD\_USER1  WHILE @@FETCH\_STATUS = 0  BEGIN  EXEC ADD\_PATIENT @GENDER ,@FIRST\_NAME1 ,@LAST\_NAME1 ,@PATRONY1 ,@BDAY1 ,@HOME\_ADRESS,@PASSPORT\_ID ,@NAME\_HOSPITAL3 ,@PLACE\_OF\_WORK\_OR\_STUDY ,@DISABILITY ,@CHRONIC\_DISEASES ,@PASSWORD\_USER1  SET @PASSPORT\_ID=NULL  SET @PLACE\_OF\_WORK\_OR\_STUDY=NULL  SET @DISABILITY=NULL  SET @CHRONIC\_DISEASES=NULL  FETCH NEXT FROM PATIENT\_CURSOR INTO @GENDER,@FIRST\_NAME1,@LAST\_NAME1,@PATRONY1  ,@BDAY1,@HOME\_ADRESS,  @PASSPORT\_ID,@NAME\_HOSPITAL3,@PLACE\_OF\_WORK\_OR\_STUDY,@DISABILITY,@CHRONIC\_DISEASES,@PASSWORD\_USER1  END;  CLOSE PATIENT\_CURSOR;  DEALLOCATE PATIENT\_CURSOR;  ------------------------------------------------------------------  DECLARE @FIRST\_NAME5 NVARCHAR(20)  DECLARE @LAST\_NAME5 NVARCHAR(20)  DECLARE @SPECIALTY\_NAME5 NVARCHAR(50)  DECLARE @FIRST\_NAME\_PATIENT NVARCHAR(20)  DECLARE @LAST\_NAME\_\_PATIENT NVARCHAR(20)  DECLARE @NAME\_HOSPITAL5 NVARCHAR(60)  DECLARE @TIME1 NVARCHAR(20)  DECLARE @DATE1 DATE  DECLARE @SYMPTOMS NVARCHAR(100)=NULL  DECLARE @DIAGNOSIS NVARCHAR(100)  DECLARE @MEDICINES NVARCHAR(100)  DECLARE @COMMENTS NVARCHAR(200)=NULL  DECLARE RECEPTION\_CURSOR CURSOR FOR SELECT \* FROM OPENJSON(@HOSPITAL, '$.RECEPTION')WITH(FIRST\_NAME5 NVARCHAR(20) '$.FIRST\_NAME',LAST\_NAME5 NVARCHAR(20) '$.LAST\_NAME',SPECIALTY\_NAME5 NVARCHAR(50) '$.SPECIALTY\_NAME',FIRST\_NAME\_PATIENT NVARCHAR(20) '$.FIRST\_NAME\_PATIENT',LAST\_NAME\_\_PATIENT NVARCHAR(20) '$.LAST\_NAME\_\_PATIENT',NAME\_HOSPITAL5 NVARCHAR(60) '$.NAME\_HOSPITAL',TIME1 NVARCHAR(20) '$.TIME',DATE1 DATE '$.DATE',SYMPTOMS NVARCHAR(100) '$.SYMPTOMS',DIAGNOSIS NVARCHAR(100) '$.DIAGNOSIS',MEDICINES NVARCHAR(100) '$.MEDICINES',COMMENTS NVARCHAR(200) '$.COMMENTS');  OPEN RECEPTION\_CURSOR;  FETCH NEXT FROM RECEPTION\_CURSOR INTO @FIRST\_NAME5 ,@LAST\_NAME5,@SPECIALTY\_NAME5 ,@FIRST\_NAME\_PATIENT ,@LAST\_NAME\_\_PATIENT ,@NAME\_HOSPITAL5 ,@TIME1 ,@DATE1 ,@SYMPTOMS ,@DIAGNOSIS ,@MEDICINES,@COMMENTS  WHILE @@FETCH\_STATUS = 0  BEGIN  EXEC ADD\_RECEPTION @FIRST\_NAME5 ,@LAST\_NAME5,@SPECIALTY\_NAME5 ,@FIRST\_NAME\_PATIENT ,@LAST\_NAME\_\_PATIENT ,@NAME\_HOSPITAL5 ,@TIME1 ,@DATE1 ,@SYMPTOMS ,@DIAGNOSIS ,@MEDICINES,@COMMENTS  SET @SYMPTOMS=NULL  SET @COMMENTS=NULL  FETCH NEXT FROM RECEPTION\_CURSOR INTO @FIRST\_NAME5 ,@LAST\_NAME5,@SPECIALTY\_NAME5 ,@FIRST\_NAME\_PATIENT ,@LAST\_NAME\_\_PATIENT ,@NAME\_HOSPITAL5 ,@TIME1 ,@DATE1 ,@SYMPTOMS ,@DIAGNOSIS ,@MEDICINES,@COMMENTS  END;  CLOSE RECEPTION\_CURSOR;  DEALLOCATE RECEPTION\_CURSOR;  ------------------------------------------------------------------  PRINT 'ОПЕРАЦИЯ ПРОВЕДЕНА УСПЕШНО'  END  ELSE  BEGIN  PRINT 'ФАЙЛ СОСТАВЛЕН НЕ ВЕРНО';  END  END;  CREATE PROCEDURE EXPORT\_TO\_JSON\_HOSPITAL AS BEGIN  SELECT \* FROM HOSPITAL FOR json path, root('courseDataBase');  exec master.dbo.sp\_configure 'show advanced options', 1  reconfigure with override  exec master.dbo.sp\_configure 'xp\_cmdshell', 1  reconfigure with override;  DECLARE @cmd NVARCHAR(255);  select @cmd = '  bcp "use courseDataBase; SELECT NAME\_HOSPITAL,COUNTRY,CITY,AREA,REGISTRATION\_PHONE FROM HOSPITAL for json path, root(''courseDataBase'');" ' +  'queryout "C:\ycheba\3course1sem\курсач\_бд\EXPORT\_HOSPITAL.json" -S .\SQLEXPRESS01 -T -w -r -t';  exec xp\_cmdshell @cmd;  END;  CREATE PROCEDURE EXPORT\_TO\_JSON\_DEPARTMENT AS BEGIN  SELECT \* FROM DEPARTMENT FOR json path, root('courseDataBase');  exec master.dbo.sp\_configure 'show advanced options', 1  reconfigure with override  exec master.dbo.sp\_configure 'xp\_cmdshell', 1  reconfigure with override;  DECLARE @cmd NVARCHAR(255);  select @cmd = '  bcp "use courseDataBase; select \* from DEPARTMENT for json path, root(''courseDataBase'');" ' +  'queryout "C:\ycheba\3course1sem\курсач\_бд\EXPORT\_DEPARTMENT.json" -S .\SQLEXPRESS01 -T -w -r -t';  exec xp\_cmdshell @cmd;  END;  CREATE PROCEDURE EXPORT\_TO\_JSON\_DOCTOR AS BEGIN  SELECT \* FROM DOCTOR FOR json path, root('courseDataBase');  exec master.dbo.sp\_configure 'show advanced options', 1  reconfigure with override  exec master.dbo.sp\_configure 'xp\_cmdshell', 1  reconfigure with override;  DECLARE @cmd NVARCHAR(255);  select @cmd = '  bcp "use courseDataBase; SELECT \* FROM DOCTOR FOR json path, root(''courseDataBase'');" ' +  'queryout "C:\ycheba\3course1sem\курсач\_бд\EXPORT\_DOCTOR.json" -S .\SQLEXPRESS01 -T -w -r -t';  exec xp\_cmdshell @cmd;  END;  CREATE PROCEDURE EXPORT\_TO\_JSON\_PATIENT AS BEGIN  SELECT \* FROM PATIENT FOR json path, root('courseDataBase');  exec master.dbo.sp\_configure 'show advanced options', 1  reconfigure with override  exec master.dbo.sp\_configure 'xp\_cmdshell', 1  reconfigure with override;  DECLARE @cmd NVARCHAR(255);  select @cmd = '  bcp "use courseDataBase; SELECT \* FROM PATIENT FOR json path, root(''courseDataBase'');" ' +  'queryout "C:\ycheba\3course1sem\курсач\_бд\EXPORT\_PATIENT.json" -S .\SQLEXPRESS01 -T -w -r -t';  exec xp\_cmdshell @cmd;  END;  CREATE PROCEDURE EXPORT\_TO\_JSON\_RECEPTION AS BEGIN  SELECT \* FROM RECEPTION FOR json path, root('courseDataBase');  exec master.dbo.sp\_configure 'show advanced options', 1  reconfigure with override  exec master.dbo.sp\_configure 'xp\_cmdshell', 1  reconfigure with override;  DECLARE @cmd NVARCHAR(255);  select @cmd = '  bcp "use courseDataBase; SELECT \* FROM RECEPTION FOR json path, root(''courseDataBase'');" ' +  'queryout "C:\ycheba\3course1sem\курсач\_бд\EXPORT\_RECEPTION.json" -S .\SQLEXPRESS01 -T -w -r -t';  exec xp\_cmdshell @cmd;  END;  CREATE PROCEDURE EXPORT\_TO\_JSON\_SPECIALTY AS BEGIN  SELECT \* FROM SPECIALTY FOR json path, root('courseDataBase');  exec master.dbo.sp\_configure 'show advanced options', 1  reconfigure with override  exec master.dbo.sp\_configure 'xp\_cmdshell', 1  reconfigure with override;  DECLARE @cmd NVARCHAR(255);  select @cmd = '  bcp "use courseDataBase; SELECT \* FROM SPECIALTY FOR json path, root(''courseDataBase'');" ' +  'queryout "C:\ycheba\3course1sem\курсач\_бд\EXPORT\_SPECIALTY.json" -S .\SQLEXPRESS01 -T -w -r -t';  exec xp\_cmdshell @cmd;  END;  CREATE PROCEDURE EXPORT\_TO\_JSON\_TIME AS BEGIN  SELECT \* FROM TIME\_TIMETABLE FOR json path, root('courseDataBase');  exec master.dbo.sp\_configure 'show advanced options', 1  reconfigure with override  exec master.dbo.sp\_configure 'xp\_cmdshell', 1  reconfigure with override;  DECLARE @cmd NVARCHAR(255);  select @cmd = '  bcp "use courseDataBase; SELECT \* FROM TIME\_TIMETABLE FOR json path, root(''courseDataBase'');" ' +  'queryout "C:\ycheba\3course1sem\курсач\_бд\EXPORT\_TIME.json" -S .\SQLEXPRESS01 -T -w -r -t';  exec xp\_cmdshell @cmd;  END;  CREATE PROCEDURE EXPORT\_TO\_JSON\_PASSWORD  AS  BEGIN  OPEN SYMMETRIC KEY SSN\_Key\_1  DECRYPTION BY CERTIFICATE HumanResources;  SELECT ID\_DOCTOR,USER\_PASSWORD,ID\_PATIENT FROM PASSWORD\_TABLE FOR json path, root('courseDataBase');  exec master.dbo.sp\_configure 'show advanced options', 1  reconfigure with override  exec master.dbo.sp\_configure 'xp\_cmdshell', 1  reconfigure with override;  DECLARE @cmd NVARCHAR(255);  select @cmd = '  bcp "use courseDataBase; SELECT ID\_DOCTOR,USER\_PASSWORD,ID\_PATIENT FROM PASSWORD\_TABLE FOR json path, root(''courseDataBase'');" ' +  'queryout "C:\ycheba\3course1sem\курсач\_бд\EXPORT\_PASSWORD.json" -S .\SQLEXPRESS01 -T -w -r -t';  exec xp\_cmdshell @cmd;  END |