МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Забайкальский государственный университет»

(ФГБОУ ВО «ЗабГУ»)

Энергетический факультет

Кафедра прикладной информатики и математики

КУРСОВАЯ РАБОТА

по дисциплине: «Базы данных»

на тему: «Проектирование и разработка базы данных «Страховая компания»»

Семестр 4

Выполнил: ст. гр. ПИ-21

Александров Руслан

Проверил: старший

преподаватель кафедры ПИМ

Демченко К.А.

Чита 2024

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Забайкальский государственный университет»

(ФГБОУ ВО «ЗабГУ»)

Энергетический факультет

Кафедра прикладной информатики и математики

ЗАДАНИЕ

на курсовую работу (проект)

Студенту Александрову Руслану Александровичу

Направление подготовки: 09.03.03 Прикладная информатика

Профиль подготовки Прикладная информатика в экономике

1 Тема курсовой работы (проекта) «Проектирование и разработка базы данных «Страховая компания»

2 Срок сдачи студентом законченной работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3 Перечень подлежащих разработке в курсовой работе (проекте) вопросов:

1) Анализ предметной области: стоматологическая клиника

2) Проектирование инфологической и даталогической модели базы данных

3) Выполнить проектирование базы данных в среде MS SQL для базы данных «Страховая компания»

4. Перечень графического материала (если имеется):

Схема БД

Дата выдачи задания \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель курсовой работы (проекта)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Задание принял к исполнению

«\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_г.

Подпись студента \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оглавление

[Введение 4](#_Toc149614281)

[1 Проектирование базы данных. 5](#_Toc149614282)

[1.1 Описание предметной области. 5](#_Toc149614283)

[1.2 Создание инфологической модели. 6](#_Toc149614284)

[1.3 Даталогическая модель базы данных. 8](#_Toc149614285)

[1.4 Стратегия резервного копирования 11](#_Toc149614286)

[2 Реализация базы данных 13](#_Toc149614287)

[2.1 Характеристика используемой СУБД 13](#_Toc149614288)

[2.2 Создание структуры базы данных 13](#_Toc149614289)

[2.2.1 Таблицы БД 13](#_Toc149614290)

[2.2.2 Индексы 15](#_Toc149614291)

[2.3 Создание представлений 17](#_Toc149614292)

[2.3.1 Однотабличный запрос на выборку по условиям 18](#_Toc149614293)

[2.3.2 Многотабличный запрос 19](#_Toc149614294)

[2.3.3 Запрос с использованием подзапроса 20](#_Toc149614295)

[2.3.4 Итоговый запрос 20](#_Toc149614296)

[2.4 Примеры запросов на модификацию данных 21](#_Toc149614297)

[2.4.1 Простые запросы 21](#_Toc149614298)

[2.4.2 Запросы с использованием подзапросов 21](#_Toc149614299)

[2.5 Описание триггеров 22](#_Toc149614300)

[2.6 Создание пользователей и назначение привилегий 24](#_Toc149614301)

[Заключение 26](#_Toc149614302)

[Список литературы 28](#_Toc149614303)

[Приложения 29](#_Toc149614304)

Введение

Цель данной курсовой работы заключается в проверке умения применять полученные знания по дисциплине «Базы данных» и овладении основами практических навыков, связанных с созданием и проектированием баз данных, с фокусом на страховом агентстве.

База данных, разрабатываемая в рамках курсового проекта, предназначена для оптимизации работы и упрощения процессов, связанных с управлением клиентами и страховыми полисами в страховом агентстве. Это позволит систематизировать информацию, повысить эффективность работы и обеспечить электронное хранение данных.

С использованием данной базы данных станет возможным:

* накапливать информацию о клиентах, их страховых полисах и прочих связанных данных.
* осуществлять поиск и просмотр необходимой информации.
* добавлять, изменять и удалять данные.
* хранить данные в электронном виде.

Для создания базы данных в данной курсовой работе будет использована СУБД MS SQL Server 2017. Эта система обеспечит эффективное хранение и обработку данных для страхового агентства, а также предоставит разнообразные инструменты для работы с базой данных.

Путем разработки данной базы данных и применения полученных знаний студент сможет освоить практические навыки в создании и проектировании баз данных и успешно применить их в конкретной сфере страхового агентства.

1 Проектирование базы данных.

1.1 Описание предметной области.

Страховое агентство - это организация, предоставляющая услуги по страхованию для защиты клиентов от финансовых рисков, связанных с различными событиями и потерями. В настоящее время база данных является неотъемлемой частью работы любой организации, включая страховые агентства. Создание базы данных позволяет структурированно хранить информацию, что значительно оптимизирует работу агентства.

Сотрудникам страхового агентства приходится выполнять ряд повторяющихся операций при работе с клиентами: оформление полисов страхования, расчет страховых премий, ведение информации о страховых случаях, управление данными о клиентах и страховых продуктах. Для эффективной работы страхового агентства также необходимо иметь возможность проводить анализ страховых полисов и клиентской активности. Автоматизация этих процессов становится необходимостью.

Область применения страховых агентств включает:

* страхование имущества;
* страхование автомобилей;
* медицинское страхование;
* страхование жизни;
* страхование ответственности (страхование ответственности предприятий или физических лиц за причинение ущерба третьим лицам, включая материальный и моральный ущерб, и возмещение причиненного ущерба).

Таким образом, с помощью базы данных можно структурированно хранить информацию о сотрудниках и клиентах страхового агентства, информацию о страховых полисах, страховых случаях и другие данные, необходимые для анализа деятельности организации. Это позволяет ускорить и упростить работу страхового агентства при выполнении повторяющихся операций и проведении анализа страховых полисов и активности клиентов.

1.2 Создание инфологической модели.

Следующим шагом в разработке базы данных является создание инфологической модели.

Инфологическая модель представляет базу данных в понятном для человека виде.

Для правильного проектирования инфологической модели данных используется метод нормализации отношений.

Нормализация - это процесс организации данных в базе данных, который включает создание таблиц и установление связей между ними.

Цель нормализации состоит в том, чтобы устранить проблемы, такие как:

* избыточность данных;
* аномалии при обновлении данных;
* аномалии при удалении данных;
* аномалии при вводе данных.

Ограничения.

Для применения метода нормализации необходимо определить следующие ограничения:

* в страховом агентстве агенты заключают договоры со страхователями и предлагают им страховые продукты.
* у агента, может быть, один или несколько заключенных договоров, или вовсе не быть заключенных договоров.
* у одного страхователя, может быть, один или несколько заключенных договоров, или вовсе не быть заключенных договоров.
* каждый договор имеет уникальный номер.
* на каждом договоре работает только один агент.
* у каждого страхового продукта есть название и стоимость.
* в рамках одного договора может быть предложено несколько страховых продуктов.
* договор может быть заключен в офисе агентства.

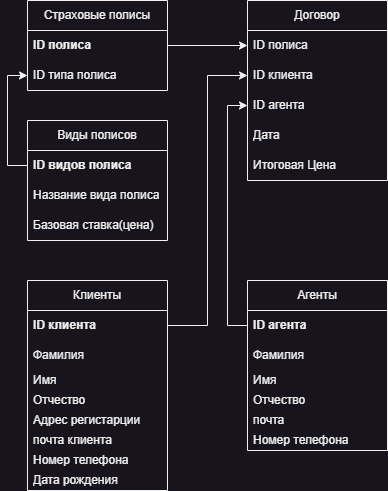


Рисунок *1*. Инфологическая модель базы данных «Страховое агентство».

1.3 Даталогическая модель базы данных.

Проанализировав инфологическую модель, мы можем спроектировать даталогическую модель.

Под даталогической понимается модель, отражающая логические взаимосвязи между элементами данных безотносительно их содержания и физической организации. Конечным результатом даталогического проектирования является описание логической структуры базы данных на языке программирования.

Даталогическая модель базы данных «Страховое агентство» спроектирована следующим образом:

Таблица 1.

Даталогическая модель базы данных «Страховое агентство»



Продолжение таблицы 1



## 1.4 Стратегия резервного копирования

Для обеспечения безопасности данных и предотвращения их потери необходимо разработать и реализовать стратегию резервного копирования и восстановления базы данных. Стратегия резервного копирования баз данных представляет собой план действий, который определяет создание резервных копий данных и их хранение, а также процедуры восстановления данных в случае их потери или повреждения.

Важной составляющей стратегии резервного копирования является регулярное создание копий базы данных. Это позволяет сохранить актуальную информацию и минимизировать потери данных при возникновении непредвиденных сбоев. Для создания резервной копии базы данных «Insurance Agency» воспользуемся следующим запросом:

USE master;

GO BACKUP DATABASE InsuranceAgency

TO DISK = 'C:/BACKUP/InsuranceAgencyBackup.bak'

GO

В случае потери базы данных, ее можно восстановить из резервной копии с помощью следующего запроса:

USE master;

GO

RESTORE DATABASE InsuranceAgency

FROM DISK = 'C:/BACKUP/InsuranceAgencyBackup.bak'

GO

Кроме того, существует возможность создания инкрементных резервных копий, которые копируют только измененные данные с момента последней полной или инкрементной копии. Это помогает сэкономить ресурсы и время при создании копий больших объемов данных. Для создания инкрементной резервной копии базы данных "Insurance Agency" можно использовать следующий запрос:

BACKUP DATABASE InsuranceAgency

TO DISK = 'C:/BACKUP/InsuranceAgencyBackup.bak'

WITH DIFFERENTIAL

Стратегия резервного копирования и восстановления базы данных является важной частью управления базами данных и гарантирует надежность и безопасность хранения данных. Глубокое понимание процессов резервного копирования и восстановления баз данных позволяет избежать потери ценных данных и обеспечить стабильную работу базы данных.

# 2 Реализация базы данных

## 2.1 Характеристика используемой СУБД

Для реализации базы данных страхового агентства в качестве системы управления базами данных (СУБД) можно выбрать реляционную модель, например, MS SQL Server 2017. MS SQL Server 2017 предоставляет надежное и производительное решение для управления данными в различных предметных областях, включая страхование.

MS SQL Server 2017 обладает мощными инструментами, такими как индексы, оптимизация запросов и партиционирование, которые обеспечивают высокую производительность базы данных. Она также поддерживает функции репликации, резервного копирования и восстановления, интеграцию с Microsoft Office и возможности анализа данных.

Для минимальной конфигурации MS SQL Server 2017 требуется двухъядерный процессор, 4 ГБ оперативной памяти и 10 ГБ свободного места на жестком диске. Однако, для оптимальной производительности, рекомендуется использовать более мощные серверы с многопроцессорными системами и высокоскоростным хранилищем данных.

Стоимость MS SQL Server 2017 зависит от версии и лицензионной политики. Также, может потребоваться дополнительные расходы на обучение и настройку системы.

## 2.2 Создание структуры базы данных

### 2.2.1 Таблицы БД

Перед созданием базы данных необходимо описать сценарий создания базы данных и всех её таблиц с указанием полей, ограничений, значений по умолчанию, обозначая первичные и внешние ключи отношений.

Для создания базы данных используется следующий запрос:

USE master

GO

CREATE DATABASE [Insurance agency]

ON

(NAME = [Insurance agency\_dat],

FILENAME =

'C:\mydb\insurance\_agency\_dat.mdf',

SIZE = 10,

MAXSIZE = 50,

FILEGROWTH = 5)

LOG ON

(NAME = [Insurance agency\_log],

FILENAME =

'C:\mydb\insurance\_agency\_log.ldf',

SIZE = 5MB,

MAXSIZE = 25MB,

FILEGROWTH = 5MB);

GO

После того как база данных создана необходимо приступить к созданию таблиц в базе данных с помощью запросов:

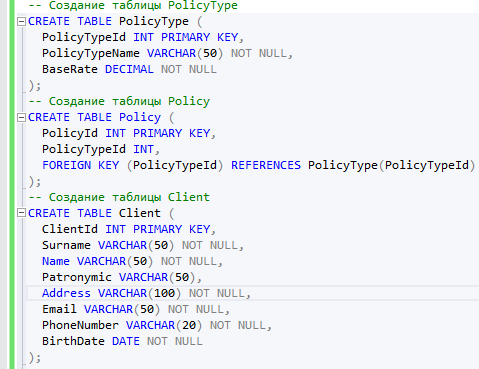


Рисунок *2*. Код создания таблиц «PolicyType», «Policy», «Client»

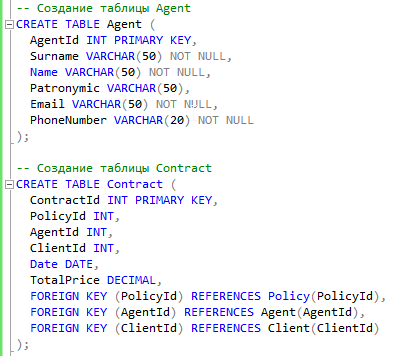


Рисунок *3*. Код создания таблиц «Agent», «Contract»

После выполнения всех вышеуказанных запросов будут созданы все необходимые таблицы, определены поля и ключи, а диаграмма базы данных будет выглядеть согласно рисунку 4:

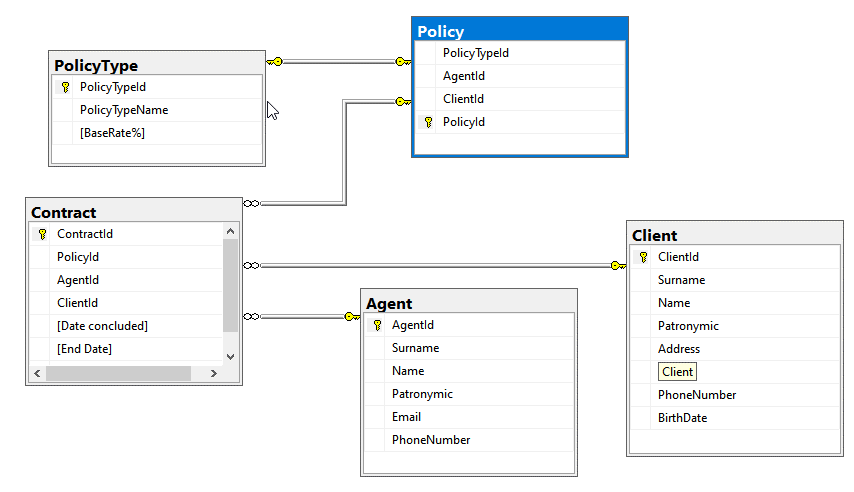


Рисунок 4. Диаграмма базы данных

2.2.2 Индексы

Индекс является структурой на диске или в памяти, которая связана с таблицей или представлением и ускоряет получение строк из таблицы или представления.

Индекс – это упорядоченный список значений, и для каждого значения есть указатели на страницы данных, где находятся эти значения. Сам индекс хранится на страницах, называемых страницами индекса.

В СУБД MS SQL 2017 существуют следующие типы индексов:

* Кластеризованный;
* Некластеризованный;
* Уникальная идентификация.

Кластеризованные индексы сортируют и хранят строки данных в таблице, основываясь на их ключевых значениях. Может быть только один кластеризованный индекс на таблицу, потому что сами строки данных могут быть отсортированы только в одном порядке.

Некластеризованный индекс содержит значения ключей индекса и указатели строк, которые указывают на место хранения табличных данных. Можно создать несколько некластеризованных индексов для таблицы или индексированного представления. Как правило, некластеризованные индексы должны быть разработаны таким образом, чтобы повысить производительность часто используемых запросов, которые не охвачены кластеризованным индексом.

Уникальный индекс гарантирует, что ключ индекса не будет содержать одинаковых значений, а значит, каждая строка в таблице будет уникальна. Создание уникального индекса имеет смысл, только если данные сами по себе могут быть уникальными.

Правильно спроектированные индексы имеют первостепенную важность для достижения высокой производительности баз данных.

Индексы для базы данных "Стоматологическая клиника" создаются с помощью следующих запросов:

* сценарий создания индекса для таблицы Policy. Этот индекс позволит ускорить поиск по id полиса:

USE [Insurance agency]

CREATE NONCLUSTERED INDEX idx\_Policy\_Policyid

ON [dbo].[Policy](PolicyId);

* сценарий создания индекса для таблицы Contract. Этот индекс ускорит поиск договоров по идентификатору полиса:

USE [Insurance agency]

CREATE NONCLUSTERED INDEX idx\_Contract\_PolicyId

ON [dbo].[Contract] ([PolicyId]);

* сценарий создания индекса для таблицы Client. Этот индекс ускорит поиск клиентов по фамилии:

USE [Insurance agency]

CREATE NONCLUSTERED INDEX idx\_Client\_Surname

ON [dbo].[Client] ([Surname]);

* сценарий создания индекса для таблицы Agent. Этот индекс ускорит поиск агентов по электронной почте:

USE [Insurance agency]

CREATE NONCLUSTERED INDEX idx\_Agent\_Email

ON [dbo].[Agent] ([Email]);

2.3 Создание представлений

Представление — это виртуальная таблица, содержимое которой определяется запросом. Как и таблица, представление состоит из ряда именованных столбцов и строк данных. Пока представление не будет проиндексировано, оно не существует в базе данных как хранимая совокупность значений. Строки и столбцы данных извлекаются из таблиц, указанных в определяющем представление запросе и динамически создаваемых при обращениях к представлению.

Представление выполняет функцию фильтра базовых таблиц, на которые оно ссылается. Определяющий представление запрос может быть инициирован в одной или нескольких таблицах или в других представлениях текущей или других баз данных. Кроме того, для определения представлений с данными из нескольких разнородных источников можно использовать распределенные запросы. Это полезно, например, если нужно объединить структурированные подобным образом данные, относящиеся к разным серверам, каждый из которых хранит данные конкретного отдела организации.

Представления обычно используются для направления, упрощения и настройки восприятия каждым пользователем информации базы данных. Представления могут использоваться как механизмы безопасности, давая возможность пользователям обращаться к данным через представления, но не предоставляя им разрешений на непосредственный доступ к базовым таблицам, лежащим в основе представлений. Представления могут использоваться для обеспечения интерфейса обратной совместимости, моделирующего таблицу, которая существует, но схема которой изменилась. Представления могут также использоваться при прямом и обратном копировании данных в SQL Server для повышения производительности и секционирования данных.

**2.3.1 Однотабличный запрос на выборку по условиям**

Оператор SELECT в SQL является основным инструментом для извлечения данных из базы данных. Он позволяет выбрать определенные строки из таблицы, удовлетворяющие определенным условиям. Оператор WHERE используется в сочетании с SELECT для указания этих условий, которые могут включать в себя операторы сравнения, равенства и ключевое слово IS. Если вам нужно задать несколько условий, вы можете объединить их с помощью ключевого слова AND. Таким образом, SELECT и WHERE работают в паре, чтобы извлечь именно те данные, которые вас интересуют из базы данных.

Ниже представлен запрос представления, отображающего полисы, которые заканчиваться в 2024 году.

CREATE VIEW PoliciesExpiring

AS

SELECT ContractId AS 'Номер контракта', PolicyId As 'Номер полиса', AgentId As 'Номер агента', ClientId As 'Номер клиента'

FROM Contract

WHERE YEAR([End Date]) = 2024;



Рисунок 5. Результат выполнения представления «PoliciesExpiring»

### **2.3.2 Многотабличный запрос**

Многотабличный запрос SQL позволяет оперировать данными из нескольких таблиц одновременно. Он позволяет выбрать нужную информацию по заданным критериям из разных таблиц базы данных.

Для Бд был создан многотабличный запрос, который выводит информацию о высокоценных полисов в нашем случае это цена более 50000.

CREATE VIEW HighValue

AS

SELECT P.PolicyId, PT.PolicyTypeName, C.TotalPrice

FROM Policy AS P

INNER JOIN PolicyType AS PT ON P.PolicyTypeId = PT.PolicyTypeId

INNER JOIN Contract AS C ON P.PolicyId = C.PolicyId

WHERE C.TotalPrice > 50000;

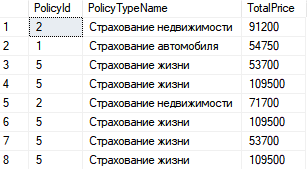


Рисунок 6. Результат выполнения представления «HighValue»

### **2.3.3 Запрос с использованием подзапроса**

Запрос, выполняющий выборку информации из одной или нескольких таблиц, используя вложенный запрос

Следующий запрос выведет информацию о суммарной стоимости заключенных полисов по клиентам.

Запрос записан в представление:

CREATE VIEW ClientTotalPrice

AS

SELECT C.ClientId, C.Surname, C.Name, (SELECT SUM(CT.TotalPrice) FROM Contract CT WHERE CT.ClientId = C.ClientId) AS TotalPrice

FROM Client C

WHERE EXISTS (SELECT 1 FROM Contract CT WHERE CT.ClientId = C.ClientId);

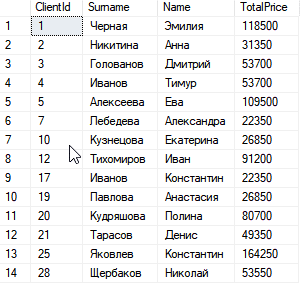


Рисунок 7. Результат выполнения представления «ClientTotalPrice»

### **2.3.4 Итоговый запрос**

Запрос, выполняющий выборку информации из одной или нескольких таблиц, выполняя группировку записей по одному из полей и вычисляя итоговые значения с использованием агрегатных функций по другим полям.

Следующий запрос выведет информацию о среднем возрасте клиента по каждому виду полиса.

Запрос представлен в представлении:

CREATE VIEW AverageClientAge

AS

SELECT PT.PolicyTypeName, AVG(DATEDIFF(YEAR, C.BirthDate, GETDATE())) AS AverageAge

FROM Policy AS P

INNER JOIN PolicyType AS PT ON P.PolicyTypeId = PT.PolicyTypeId

INNER JOIN Client AS C ON P.ClientId = C.ClientId

GROUP BY PT.PolicyTypeName;

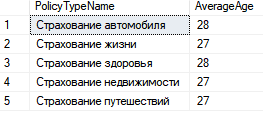


Рисунок 7. Результат выполнения представления «AverageClientAge»

## 2.4 Примеры запросов на модификацию данных

### **2.4.1 Простые запросы**

Простые запросы – запросы для выполнения операций вставки, обновления удаления.

Запрос на добавление нового клиента:

INSERT INTO Client (Surname, Name, Patronymic, Address, Email, PhoneNumber, BirthDate)

VALUES ('Иванов', 'Петр', 'Сергеевич', 'ул. Примерная, 123', 'ivanov@example.com', '123-456-7890', '1990-01-15');

Запрос на обновление базовой ставки:

UPDATE PolicyType

SET BaseRate = 10

WHERE PolicyTypeId = 5;

### **2.4.2 Запросы с использованием подзапросов**

Запрос найдем средний возраст клиентов, у которых есть контракты на страхование, и выведем клиентов старше среднего возраста.

SELECT C.ClientId, C.Surname, C.Name, C.BirthDate

FROM Client C

WHERE YEAR(C.BirthDate) < (

SELECT AVG(YEAR(BirthDate))

FROM Client

WHERE ClientId IN (SELECT DISTINCT ClientId FROM Contract)

);

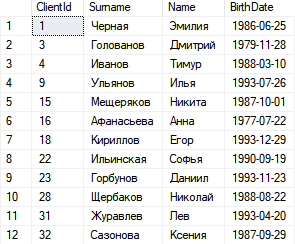


Рисунок 8. Результат выполнения запроса

## 2.5 Описание триггеров

Триггеры в MS SQL Server - это специальный тип хранимых процедур, которые автоматически выполняются при определенных действиях с таблицами или представлениями, таких как добавление, обновление или удаление данных. Для создания триггера используется выражение CREATE TRIGGER, за которым следует имя триггера и имя таблицы или представления, с которыми он связан. Существуют два типа триггеров: AFTER и INSTEAD OF, и они могут быть созданы для действий INSERT, UPDATE или DELETE. SQL-выражения, которые составляют тело триггера, следуют за ключевым словом AS. Триггеры можно отключить с помощью команды DISABLE TRIGGER и включить с помощью команды ENABLE TRIGGER.

Для базы данных были написаны следующие триггеры:

Триггер для проверки изменения/добавления корректных данных:

CREATE TRIGGER CheckPolicyDates

ON

Policy

AFTER INSERT, UPDATE

AS

BEGIN

IF EXISTS (

SELECT 1

FROM inserted

WHERE [Date concluded] > [End Date]

)

BEGIN

PRINT 'Дата начала полиса не может быть позже даты окончания'

ROLLBACK TRANSACTION

END

END;

Данный триггер полезен тем что не позволяет вводить околёсицу и сохранять целостность и правильную структуру данных необходимую для правильной работы бд.

Триггер для автоматического определения статуса полиса:

CREATE TRIGGER CheckPolicyAmount

ON

Policy

AFTER INSERT, UPDATE

AS

BEGIN

IF EXISTS (

SELECT 1

FROM inserted

WHERE [PolicyPrice] < 0

)

BEGIN

PRINT 'Сумма страховой выплаты не может быть отрицательной'

ROLLBACK TRANSACTION

END

END;

Этот запрос выполнит обновление статусов для всех записей в таблице «Policy» в соответствии с условиями триггера. После этого, триггер будет срабатывать и на новые записи, и на существующие записи при вставке или обновлении данных в таблице «Contract».

## 2.6 Создание пользователей и назначение привилегий

База данных содержит много информации, которая должна предоставляться одной группе лиц и должна быть недоступна для другой группы лиц.

В соответствии с созданной системой можно выделить 2 группы: ADMIN (Администратор) и USER (Пользователь).

Таблица 2

Распределение привилегий

|  |  |  |
| --- | --- | --- |
| Таблица | Группа пользователей | |
| Администратор | Оператор |
| Policy | ALL | SELECT, UPDATE |
| PolicyType | ALL | SELECT |
| Client | ALL | SELECT, UPDATE, INSERT |
| Agent | ALL | SELECT |
| Contract | ALL | SELECT, INSERT |

Создание учетной записи администратора «Администратор»:

USE [Insurance agency]

CREATE LOGIN Admin1

WITH PASSWORD = 'Admin1Password';

CREATE USER Admin1 FOR LOGIN Admin1;

GRANT SELECT, UPDATE, INSERT, DELETE ON [dbo].[PolicyType] TO Admin1;

GRANT SELECT, UPDATE, INSERT, DELETE ON [dbo].[Agent] TO Admin1;

GRANT SELECT, UPDATE, INSERT, DELETE ON [dbo].[Client] TO Admin1;

GRANT SELECT, UPDATE, INSERT, DELETE ON [dbo].[Policy] TO Admin1;

Создание учетной записи пользователя «USER»:

CREATE LOGIN USER1

WITH PASSWORD = 'USER1';

GO

CREATE USER USER1 FOR LOGIN USER1;

GO

GRANT SELECT ON [dbo].[PolicyType] TO USER1;

GRANT SELECT ON [dbo].[Agent] TO USER1;

GRANT SELECT, UPDATE, INSERT ON [dbo].[Client] TO USER1;

GRANT SELECT, UPDATE ON [dbo].[Policy] TO USER1;

# Заключение

В рамках данной курсовой работы была проведена разработка базы данных для страховой компании, что представляет собой важный этап в создании информационной системы, способной эффективно управлять страховыми операциями и клиентской информацией. Проектирование и разработка базы данных выполнялись с учетом основных бизнес-процессов страховой компании, таких как управление полисами, агентами, клиентами и финансовыми операциями.

В результате выполнения данной работы были достигнуты следующие основные цели и результаты:

1. Создание структуры базы данных: была разработана структура базы данных, включающая таблицы для хранения информации о полисах, типах полисов, агентах, клиентах и финансовых операциях. Эта структура была спроектирована с учетом нормализации данных, что способствует эффективному хранению и обработке информации.
2. Определение связей между таблицами: были определены связи между таблицами базы данных, что позволяет эффективно связывать данные и обеспечивать целостность информации. Например, таблица полисов связана с таблицей агентов и клиентов через уникальные идентификаторы.
3. Разработка схемы безопасности: для обеспечения безопасности данных были созданы роли пользователей и предоставлены соответствующие разрешения на выполнение операций в базе данных. Это позволяет ограничить доступ к чувствительной информации.
4. Создание триггеров и хранимых процедур: для автоматизации определенных бизнес-процессов и поддержания целостности данных были разработаны триггеры и хранимые процедуры.
5. Разработка представлений: были созданы представления, которые облегчают выполнение запросов к данным и предоставляют удобный доступ к информации.

Анализ работы объектов показал, что они в полной мере выполнили возложенные на них функции.

Таким образом, все поставленные в курсовой работе задачи выполнены, цель работы можно считать достигнутой.

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Реляционные БД. Типы нормального вида данных. Нормализация баз данных. Режим доступа: <https://intellect.icu/relyatsionnye-bd-tipy-normalnogo-vida-dannykh-normalizatsiya-baz-dannykh-154>, дата обращения 3.04.2022
2. Даталогическая модель базы данных. Режим доступа: <https://lektsia.com/9x68c4.html>, дата обращения: 3.04.2022
3. Резервное копирование и восстановление баз данных SQLServer. Режим доступа: <https://docs.microsoft.com/ru-ru/sql/relational-databases/backup-restore/back-up-and-restore-of-sql-server-databases?view=sql-server-ver15>, дата обращения: 8.05.2022
4. Выбор СУБД. Режим доступа: <https://www.olly.ru/katalogresh/arkhiv/vybor-subd/>, дата обращения: 8.05.2022
5. Оконные функции. Режим доступа: <https://learn.microsoft.com/ru-RU/sql/t-sql/queries/select-over-clause-transact-sql?view=sql-server-linux-ver15/>, дата обращения: 20.05.2022
6. Руководство по архитектуре и разработке индексов SQLServer и AzureSQL. Режим доступа: <https://docs.microsoft.com/ru-ru/sql/relational-databases/sql-server-index-design-guide?view=sql-server-ver15>, дата обращения: 20.05.2022
7. Триггеры. Режим доступа: <https://metanit.com/sql/sqlserver/12.1.php>, дата обращения 20.05.2022
8. Представления. Режим доступа: <https://docs.microsoft.com/ru-ru/sql/relational-databases/views/views?view=sql-server-ver15>, дата обращения: 20.05.2022
9. Хранимые процедуры. Режим доступа: <https://docs.microsoft.com/ru-ru/sql/relational-databases/stored-procedures/stored-procedures-database-engine>, дата обращения: 20.05.2022

# Приложения

Приложение 1

Содержимое таблицы «Policy»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PolicyTypeId | AgentId | ClientId | PolicyId | PolicyPrice |
| 1 | 6 | 1 | 1 | 30 |
| 2 | 4 | 2 | 2 | 60 |
| 3 | 9 | 3 | 3 | 90 |
| 4 | 12 | 4 | 4 | 120 |
| 5 | 8 | 5 | 5 | 150 |
| 1 | 8 | 20 | 6 | 30 |
| 1 | 4 | 12 | 7 | 30 |
| 1 | 6 | 25 | 8 | 30 |
| 1 | 9 | 17 | 9 | 30 |
| 1 | 12 | 1 | 10 | 30 |
| 1 | 8 | 4 | 11 | 30 |
| 1 | 12 | 1 | 12 | 30 |
| 1 | 5 | 30 | 13 | 30 |
| 1 | 7 | 20 | 14 | 30 |
| 1 | 4 | 28 | 15 | 30 |
| 1 | 10 | 2 | 16 | 30 |
| 1 | 4 | 5 | 17 | 30 |
| 1 | 8 | 21 | 18 | 30 |
| 1 | 1 | 2 | 19 | 30 |
| 1 | 11 | 3 | 20 | 30 |
| 1 | 3 | 28 | 21 | 30 |
| 1 | 8 | 25 | 22 | 30 |
| 1 | 15 | 19 | 23 | 30 |
| 1 | 14 | 10 | 24 | 30 |
| 1 | 14 | 7 | 25 | 30 |
| 2 | 8 | 20 | 26 | 60 |
| 2 | 4 | 12 | 27 | 60 |
| 2 | 6 | 25 | 28 | 60 |
| 2 | 9 | 17 | 29 | 60 |

Продолжение приложение 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 | 9 | 17 | 29 | 60 |
| 2 | 12 | 1 | 30 | 60 |
| 2 | 8 | 4 | 31 | 60 |
| 2 | 12 | 1 | 32 | 60 |
| 2 | 5 | 30 | 33 | 60 |
| 2 | 7 | 20 | 34 | 60 |
| 2 | 4 | 28 | 35 | 60 |
| 2 | 10 | 2 | 36 | 60 |
| 2 | 4 | 5 | 37 | 60 |
| 2 | 8 | 21 | 38 | 60 |
| 2 | 1 | 2 | 39 | 60 |
| 2 | 11 | 3 | 40 | 60 |
| 2 | 3 | 28 | 41 | 60 |
| 2 | 8 | 25 | 42 | 60 |
| 2 | 15 | 19 | 43 | 60 |
| 2 | 14 | 10 | 44 | 60 |
| 2 | 14 | 7 | 45 | 60 |
| 3 | 8 | 20 | 46 | 90 |
| 3 | 4 | 12 | 47 | 90 |
| 3 | 6 | 25 | 48 | 90 |
| 3 | 9 | 17 | 49 | 90 |
| 3 | 12 | 1 | 50 | 90 |
| 3 | 8 | 4 | 51 | 90 |
| 3 | 12 | 1 | 52 | 90 |
| 3 | 5 | 30 | 53 | 90 |
| 3 | 7 | 20 | 54 | 90 |
| 3 | 4 | 28 | 55 | 90 |
| 3 | 10 | 2 | 56 | 90 |
| 3 | 4 | 5 | 57 | 90 |

Продолжение приложение 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | 1 | 2 | 59 | 90 |
| 3 | 11 | 3 | 60 | 90 |
| 3 | 3 | 28 | 61 | 90 |
| 3 | 8 | 25 | 62 | 90 |
| 3 | 15 | 19 | 63 | 90 |
| 3 | 14 | 10 | 64 | 90 |
| 3 | 14 | 7 | 65 | 90 |
| 4 | 8 | 20 | 66 | 120 |
| 4 | 4 | 12 | 67 | 120 |
| 4 | 6 | 25 | 68 | 120 |
| 4 | 9 | 17 | 69 | 120 |
| 4 | 12 | 1 | 70 | 120 |
| 4 | 8 | 4 | 71 | 120 |
| 4 | 12 | 1 | 72 | 120 |
| 4 | 5 | 30 | 73 | 120 |
| 4 | 7 | 20 | 74 | 120 |
| 4 | 4 | 28 | 75 | 120 |
| 4 | 10 | 2 | 76 | 120 |
| 4 | 4 | 5 | 77 | 120 |
| 4 | 8 | 21 | 78 | 120 |
| 4 | 1 | 2 | 79 | 120 |
| 3 | 1 | 2 | 59 | 90 |
| 3 | 11 | 3 | 60 | 90 |
| 3 | 3 | 28 | 61 | 90 |

Приложение 2

Содержимое таблицы «PolicyType»

|  |  |  |
| --- | --- | --- |
| PolicyTypeId | PolicyTypeName | BaseRate% |
| 1 | Страхование автомобиля | 5 |
| 2 | Страхование недвижимости | 10 |
| 3 | Страхование здоровья | 5 |
| 4 | Страхование путешествий | 5 |
| 5 | Страхование жизни | 10 |

Приложение 3

Содержимое таблицы «Client»

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ClientId | Surname | Name | Patronymic | Address | Email | PhoneNumber | BirthDate |
| 1 | Черная | Эмилия | Григорьевна | г Чита ул курнатовского 33 кв 12 | emilia\_black86@gmail.com | 89144969006 | 25.06.1986 |
| 2 | Никитина | Анна | Михайловна | г Чита ул Анохина 94 кв 21 | anna\_nik98@gmail.com | 89244954404 | 12.01.1998 |
| 3 | Голованов | Дмитрий | Матвеевич | г Чита ул Ленинрадская 58 кв 41 | dmitry\_golovanov@mail.ru | 89144959505 | 28.11.1979 |
| 4 | Иванов | Тимур | Дмитриевич | г Чита ул Кочеткова 21 кв 24 | ivanov\_tumyr88@gmail.com | 89247444404 | 10.03.1988 |
| 5 | Алексеева | Ева | Петровна | г Чита ул Ленинградская 54 кв 25 | eva\_alex2001@gmail.com | 89999090999 | 20.12.2001 |
| 6 | Зайцев | Марк | Алексеевич | г Чита ул Бутина 73 кв 45 | mark\_zaev@mail.ru | 89244373703 | 09.04.1999 |
| 7 | Лебедева | Александра | Михайловна | г Чита ул Бутина 78 кв 24 | aleksandra\_leb@gmail.com | 89147077007 | 12.03.1997 |
| 8 | Муравьев | Арсений | Андреевич | г Чита Бутина 107 кв 14 | arkenii\_mur@mail.ru | 89243101003 | 21.05.1999 |
| 9 | Ульянов | Илья | Максимович | г Чита ул Нечаева 36 кв 26 | ilya\_maks@mail.ru | 89142102120 | 26.07.1993 |
| 10 | Кузнецова | Екатерина | Николаевна | г Чита ул Бутина 76 кв 35 | katya\_kuz@gmail.com | 89249980909 | 24.04.1998 |
| 11 | Морозова | Ева | Максимовна | г Чита ул Полины Осипенко 38 кв 10 | morozeva\_eva@mail.ru | 89144204420 | 15.02.2000 |
| 12 | Тихомиров | Иван | Сергеевич | г Чита ул Ангарская 72 кв 35 | ivan\_tur@mail.ru | 89241909001 | 25.02.1998 |
| 13 | Капустина | Дарья | Кирилловна | г Чита ул Ленина 52 кв 22 | darya\_kap@mail.ru | 89248707007 | 15.05.2002 |
| 14 | Матвеев | Дмитрий | Глебович | г Чита ул Ленина 42 кв 10 | dm\_gleb@gmail.com | 89142202002 | 20.04.1998 |

Приложение 4

Содержимое таблицы «Agent»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| AgentId | Surname | Name | Patronymic | Email | PhoneNumber |
| 1 | Алексеев | Николай | Даниилович | alex97@gmail.com | 89991343404 |
| 2 | Кузнецов | Дмитрий | Никитич | kuz\_dm@mail.ru | 89992595825 |
| 3 | Юдина | Анастасия | Александровна | anastasiaUA@gmail.com | 89245985202 |
| 4 | Казаков | Андрей | Денисович | kazAndrey001@gmail.com | 89244343333 |
| 5 | Кузнецова | София | Михайловна | sofia\_bus@mail.ru | 89149686608 |
| 6 | Муратов | Тимур | Васильевич | tumur94@inbox.ru | 89985111111 |
| 7 | Корчагина | Ксения | Данииловна | ksenyaKD@mail.ru | 89992882808 |
| 8 | Комаров | Ярослав | Маркович | komar\_mark@gmail.com | 89144774707 |
| 9 | Баранова | Наталья | Григорьевна | nata\_b\_95@mail.ru | 89247017070 |
| 10 | Аксенов | Сергей | Максимович | sergey\_aks@mail.ru | 89990101002 |
| 11 | Петров | Глеб | Маркович | petrov\_gleb98@gmail.com | 89144314031 |
| 12 | Максимов | Никита | Михайлович | nikita\_max96@inbox.ru | 89244980909 |
| 13 | Фролова | Эвелина | Алексеевна | frolova\_top98@gmail.com | 89996552505 |
| 14 | Кожевников | Даниил | Константинович | danilkoj@mail.ru | 89983900909 |
| 15 | Кузьмин | Лев | Викторович | leb\_kuz94@mail.ru | 89247687470 |

Приложение 5

Содержимое таблицы «Contract»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ContractId | PolicyId | AgentId | ClientId | Date concluded | End Date | TotalPrice |
| 1 | 5 | 8 | 20 | 12.01.2023 | 12.02.2023 | 9000 |
| 2 | 2 | 4 | 12 | 14.01.2023 | 14.10.2023 | 91200 |
| 3 | 1 | 6 | 25 | 20.01.2023 | 20.01.2024 | 54750 |
| 4 | 3 | 9 | 17 | 20.01.2023 | 20.06.2023 | 22350 |
| 5 | 4 | 12 | 1 | 01.02.2023 | 01.04.2023 | 9000 |
| 6 | 5 | 8 | 4 | 03.02.2023 | 03.08.2023 | 53700 |
| 7 | 5 | 12 | 1 | 10.02.2023 | 10.02.2024 | 109500 |
| 8 | 1 | 5 | 30 | 12.02.2023 | 12.06.2023 | 17850 |
| 9 | 2 | 7 | 20 | 14.02.2023 | 14.10.2023 | 71700 |
| 10 | 1 | 4 | 28 | 14.02.2023 | 14.06.2023 | 17850 |
| 11 | 4 | 10 | 2 | 14.02.2023 | 14.04.2023 | 9000 |
| 12 | 5 | 4 | 5 | 16.02.2023 | 16.02.2024 | 109500 |
| 13 | 4 | 8 | 21 | 18.02.2023 | 18.01.2024 | 49350 |
| 14 | 3 | 1 | 2 | 19.02.2023 | 19.07.2023 | 22350 |
| 15 | 5 | 11 | 3 | 20.02.2023 | 20.08.2023 | 53700 |
| 16 | 2 | 3 | 28 | 23.02.2023 | 23.06.2023 | 35700 |
| 17 | 5 | 8 | 25 | 26.02.2023 | 26.02.2024 | 109500 |
| 18 | 1 | 15 | 19 | 02.03.2023 | 02.09.2023 | 26850 |
| 19 | 1 | 14 | 10 | 05.03.2023 | 05.09.2023 | 26850 |
| 20 | 4 | 14 | 7 | 09.03.2023 | 09.08.2023 | 22350 |