СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Темой данной курсовой работы является – проектирование реляционной базы данных поисково-спасательного отряда.

В соответствии с предметной областью работы, будущая база данных должна хранить сведения о командах поисковиков, о катастрофах, о пропавших людях, о пользователях базой данных, о месте дислокации команды.

Для обеспечения функционала, а также для удобства пользования информационной системой необходимо разработать приложение, которое позволит добавлять, удалять, редактировать и выводить информацию о катастрофах, командах, пропавших людях. Приложение должно быть простым в использовании, которым могли бы пользоваться даже неквалифицированные сотрудники.

Актуальность данной темы определяется тем, что в настоящее время пропадает большое количество людей, а также происходит большое количество катастроф, с помощью данной базы данных можно понять периодичность катастроф и пропаж людей, и установить между ними взаимосвязь. Это очень облегчит работу координаторам команд.

Аналогов данной программы немало и не только в данной области. Они обычно разрабатываются под конкретную организацию, занимающуюся определенным видом деятельности. Создаваемая информационная система не привязывается к какой-либо существующей организации, и создается на основе некого абстрактной организации.

Для создания информационной базы данных будет использоваться СУДБ SQL. Для создания приложения – среда Visual Studio 2019.

1 ПОСТРОЕНИЕ ИНФОЛОГИЧЕСКОЙ КОНЦЕПТУАЛЬНОЙ МОДЕЛИ

1.1 Анализ предметной области и выявление необходимого набора сущностей

В ходе анализа знаний и разработке базы данных были выявлены следующие основные сущности:

Сущность *Пользователь* описывает пользователей, которые могут пользоваться данной базой данных. Характеризуется номер пользователя, его логин и пароль, его фамилия, имя отчество, номер профессии и номер команды.

Сущность *Команда* описывает название команды и текущие их задания. Характеризуется номером команды, названием команды, номером катастрофы, и номером пропавшего человека, которыми занимается команда.

Сущность *Позиции в команде* описывает все профессии требуемые в команду. Характеризуется номером команды, фамилией именем и отчеством следующих специалистов:

1. главный спасатель;
2. доктор;
3. фельдшер;
4. водитель;
5. пиротехник;
6. сварщик;
7. оператор крана;
8. дайвер;
9. гидравлик;
10. разведчик;
11. электрик;
12. зам главного спасателя.

Сущность *Регион* описывает место, где располагается штаб-квартира команды. Характеризуется порядковым номером региона, адресом и к какому департаменту относится.

Сущность *Департамент* описывает областной штаб команды. Характеризуется порядковым номером департамента, адресом и к какому главному департаменту относится.

Сущность *Главный департамент* главный штаб всех отрядов. Характеризуется порядковым номером главного департамента и адресом.

Сущность *Причины* описывает причины катастрофы. Характеризуется порядковым номером, типом катастрофы, причиной и к какой катастрофе относится.

Сущность *Катастрофа* описывает катастрофу. Характеризуется порядковым номером катастрофы, датой катастрофы, в какой стране произошла катастрофа и в каком городе.

Сущность *Люди* описывает данные пропавших людей. Характеризуется порядковый номер в списке, фамилией, именем, отчеством, датой рождения.

Сущность *Пропавший человек* описывает данные до пропажи человека. Характеризуется порядковым номером, датой пропажи, местом, где в последний раз его видели, особыми приметами.

Сущность *Профессии* описывает профессии, которые необходимы в команду. Характеризуется порядковый номер в списке, названием профессии.

Сущность *Обмундирование* описывает амуницию для каждой профессии. Характеризуется порядковый номер в списке, профессией, названием экипировки, её типом и описанием.

Сущность *Координатор* описывает принадлежность координатора. Характеризуется порядковый номер команды и порядковый номер региона.

При начальном анализе также полезно строить диаграмму потоков данных (*англ.* Data Flow Diagram, DFD-diagram). Такая диаграмма описывает внешние по отношению к системе источники и адресаты данных, логические функции, потоки данных и хранилища данных, к которым осуществляется доступ. Диаграмма потоков данных для предметной области поисково-спасательного отряда представлена на рисунке 1.1.

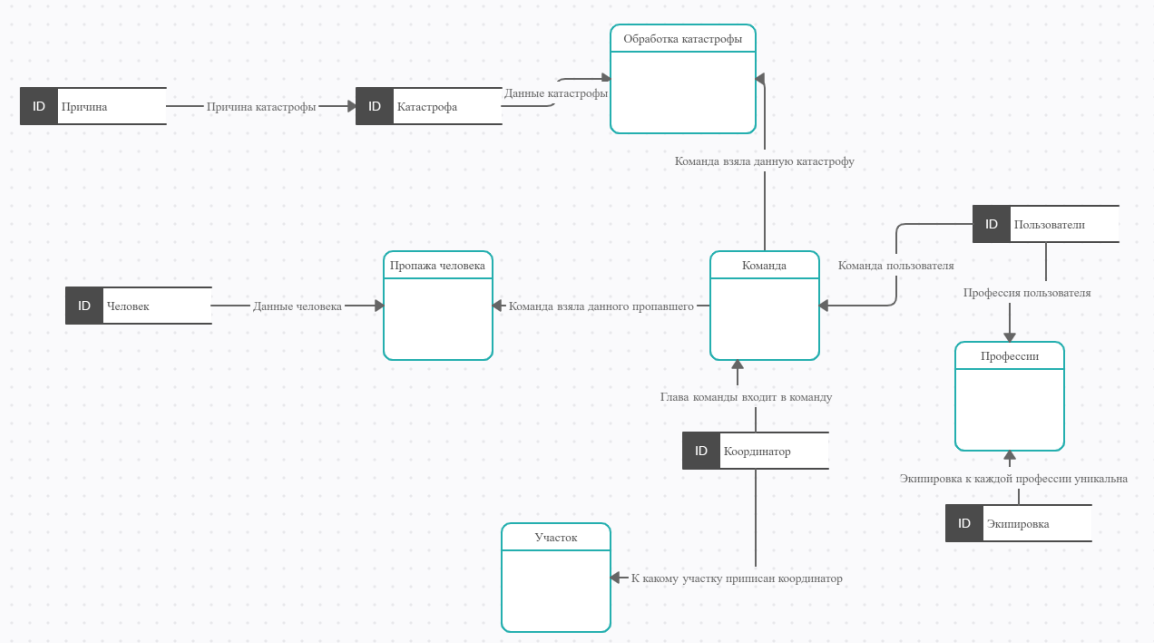

Рисунок 1.1 – Диаграмма потоков данных

По правилам построения DFD диаграмм [1] прямоугольником обозначаются источники и адресаты данных, прямоугольник с круглыми краями – функции, стрелками – потоки данных, на потоках обозначаются перемещаемые объекты.

1.2 Обоснование требуемого набора атрибутов для каждой сущности и выделение идентифицирующих атрибутов

Для построения инфологической концептуальной модели необходимо для каждой сущности, выявленной в предыдущем пункте, определить требуемый набор атрибутов. Атрибутом является поименованная характеристика сущности. Его наименование должно быть уникальным для конкретного типа сущности, но может быть одинаковым для различного типа сущностей. Атрибуты используются для определения того, какая информация должна быть собрана о сущности.

Ниже представлены сущности и определенные для них атрибуты, а также ключи (подчеркнуты). Имена сущностей и атрибутов указываются, как они будут определены в созданной базе данных, в скобках указывается перевод либо описание:

1. user (пользователь):
   – idUser (номер пользователя);
   – login (логин пользователя);
   – password (пароль пользователя);
   – family (фамилия пользователя);
   – name (имя пользователя);
   – middleName (отчество пользователя);
   – idProfession (номер профессии пользователя);
   – idTeam (номер команды пользователя).
2. team (команда):
   – idTeam (номер команды);
   – teamName (название команды);
   – idDisaster (номер катастрофы);
   – idPeople (номер пропавшего человека).
3. teamPosition (позиции в команде):
   – idTeam (номер команды);
   – seniorLifeguard (номер пользователя с профессией главный спасатель);
   – doctor (номер пользователя с профессией доктор);
   – paramedic (номер пользователя с профессией фельдшер);
   – driver (номер пользователя с профессией водитель);
   – pyrotechnist (номер пользователя с профессией пиротехник);
   – gasWelder (номер пользователя с профессией газосварщик);
   – craneOperator (номер пользователя с профессией оператор крана);
   – scubaDriver (номер пользователя с профессией дайвер);
   – hydraulicTechnician (номер пользователя с профессией гидравлик);
   – scout (номер пользователя с профессией разведчик);
   – electrician (номер пользователя с профессией электрик);

– rescuer (номер пользователя с профессией зам главного спасателя).

4. region (регион):
   – idRegion (номер региона);
   – address (адрес штаба команды);
   – idDepartment (номер областного штаба).
5. department (департамент):
   – idDepartment (номер департамента);
   – address (областной адрес штаба команды);
   – idMainDepartment (номер главного штаба).
6. mainDepartment (главный департамент):
   – idMainDepartment (номер главного штаба);
   – address (адрес главного штаба команды).
7. reason (причина катастрофы):
   – idReason (номер причины);
   – typeReason (тип причины);
   – reason (описание причины);
   – idDisaster (номер катастрофы).
8. disaster (катастрофа):
   – idDisaster (номер катастрофы);
   – date (дата катастрофы);
   – country (страна, где произошла катастрофа);
   – city (город, где произошла катастрофа).
9. people (человек):
   – idPeople (номер человека);
   – family (фамилия человека);
   – name (имя человека);
   – middleName (отчество человека);
   – dateOfBirth (дата рождения человека).
10. missingPeople (пропавший человеек):
   – idPeople (номер человека);
   – dateOfLoss (дата пропажи человека);
   – lastLocation (последнее место, где видели человека);
   – specialSign (особые приметы).
11. profession (профессия):
   – idProfession (номер профессии);
   – position (название профессии).
12. equipment (обмундирование):
   – idEquipment (номер экипировки);
   – idProfession (профессия, к которой прилагается данная экипировка);
   – equipmentName (название экипировки);
   – type (тип экипировки).
   – description (описание экипировки).

13. coordinator (координатор):
– idTeam (номер команды);
– idRegion (номер региона).

1.3 Определение связей между объектами

Кроме атрибутов каждой сущности модель данных должна определять связи между сущностями. На концептуальном уровне связи представляют собой простые ассоциации между сущностями.

Связь – это ассоциирование двух или более сущностей. Если бы назначением базы данных было только хранение отдельных, не связанных между собой данных, то ее структура могла бы быть очень простой. Однако, одно из основных требований к организации базы данных – это обеспечение возможности отыскания одних сущностей по значениям других, для чего необходимо установить между ними определенные связи. А так как в реальных базах данных нередко содержатся десятки или даже сотни сущностей, то между ними может быть установлено великое множество связей. Наличие такого множества связей и определяет сложность инфологических моделей.

Для реализации информационной системы поисково-спасательного отряда необходимо установить все связи между объектами. А именно, нужно рассмотреть всю информационную систему поисково-спасательного отряда в совокупности и определить взаимное влияние объектов, составляющих систему.

Этот процесс изображен на рисунок 1.2

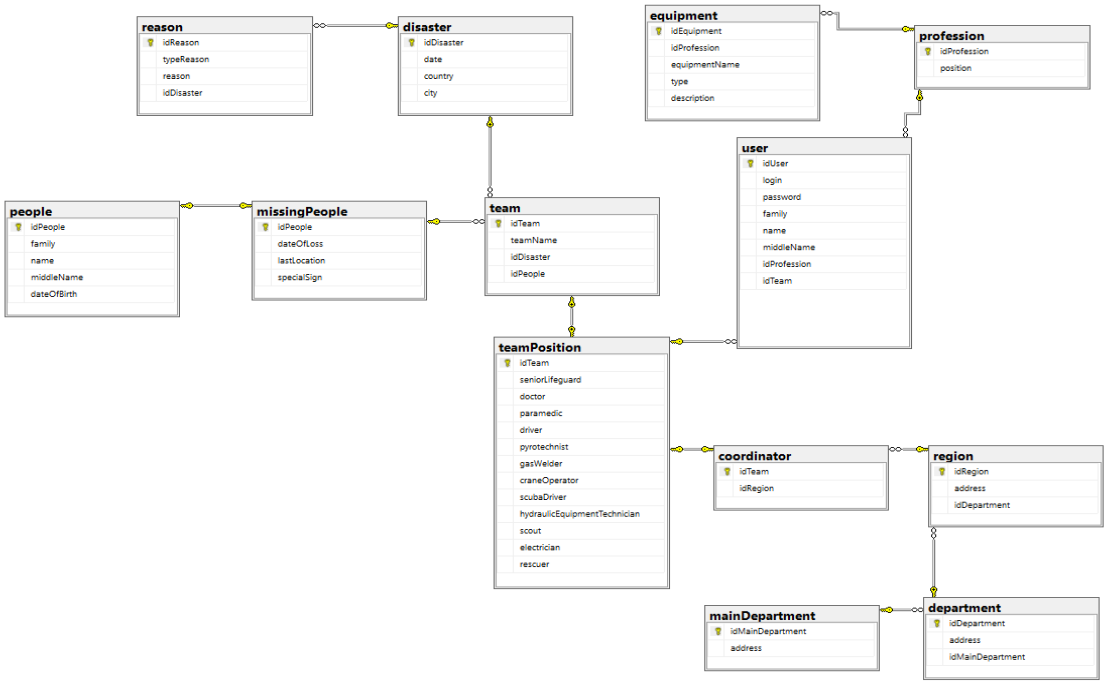

Рисунок 1.2 – Концептуальная схема базы данных

Проследить отношения, в которых состоят таблицы базы данных можно по схеме, изображенной на рис. А.1 в приложении А.

1.4 Описание полученной модели на языке инфологического проектирования

Проектирование инфологической модели предметной области – частично формализованное описание объектов предметной области в терминах некоторой семантической модели, например, в терминах ER-модели (*англ.* entity-relationship model). По правилам построения ER-диаграмм в нотации Питера Чена, сущности изображаются прямоугольниками, их атрибуты – овалами, отношения – ромбами. Связи между объектами изображаются линиями [2].

На основе проведенного проектирования, в частности на основе инфологической схемы, получим ER-диаграмму базы данных поисково-спасательного отряда, представленную в Приложении А.

2 ПОСТРОЕНИЕ СХЕМЫ РЕЛЯЦИОННОЙ БАЗЫ ДАННЫХ

2.1 Построение набора необходимых отношений базы данных

Чтобы построить схему реляционной базы данных необходимо определить совокупность отношений, которые составляют базу данных. Эта совокупность отношений будет содержать всю информацию, которая должна храниться в базе данных.

В предыдущем пункте была создана инфологическая концептуальная модель базы данных магазина, построенной с помощью языка «Таблицы-связи». На основе полученной концептуальной модели можно определить набор необходимых отношений базы данных. На рисунке 2.1 представлены отношения для базы данных поисково-спасательного отряда.

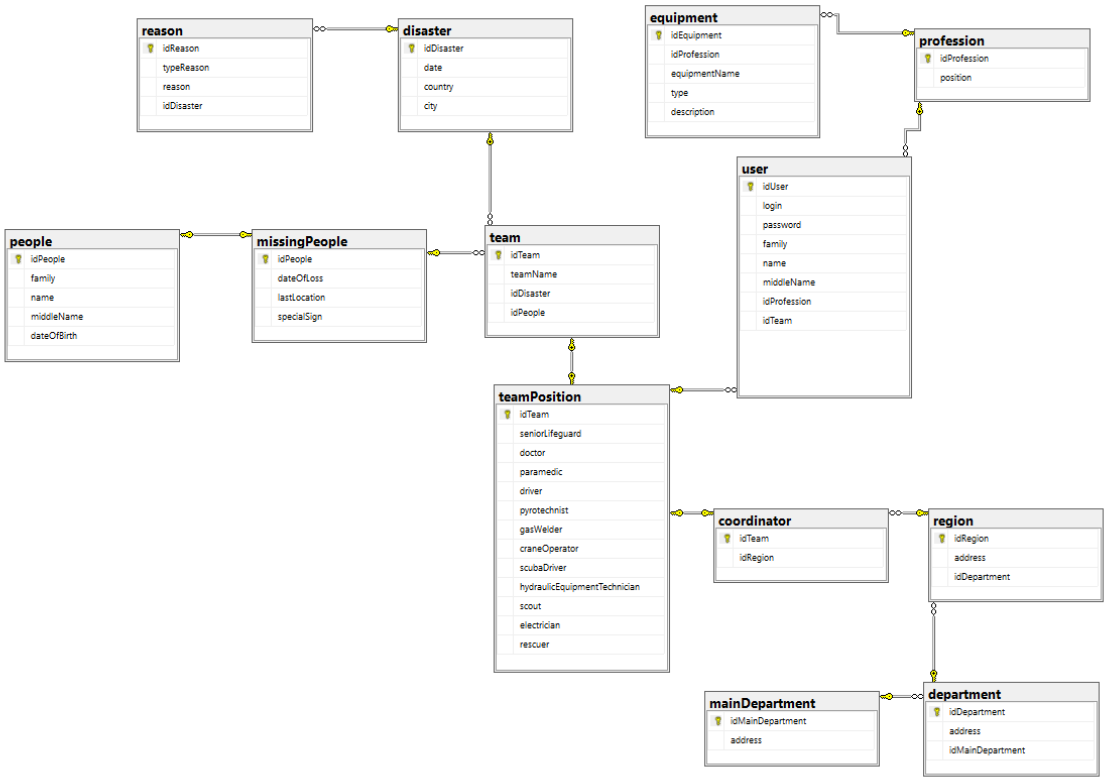

Рисунок 2.1 – Набор необходимых отношений базы данных

2.2 Задание первичных и внешних ключей определенных отношений

В реляционной базе данных каждому объекту и сущности реального мира соответствуют кортежи отношений. И любое отношение должно обладать первичным ключом. Ключ – это минимальный набор атрибутов, по значениям которых можно однозначно найти требуемый экземпляр сущности. Минимальность означает, что исключение из набора любого атрибута не позволяет идентифицировать сущность по оставшимся. Каждое отношение должно обладать хотя бы одним ключом. В таблице 2.1 определены первичные и внешние ключи для отношений.

Таблица 2.1 – Первичные и внешние ключи отношений.

№ п/п	Название таблицы	Первичный ключ	Внешние ключи
1	user	idUser	idProfession (номер профессии пользователя); idTeam (номер команды пользователя).
2	team	idTeam	idDisaster (номер катастрофы); idPeople (номер пропавшего человека).
3	teamPosition	idTeam	–
4	region	idRegion	idDepartment (номер областного штаба)
5	department	idDepartment	idMainDepartment (номер главного штаба)
6	mainDepartment	idMainDepartment	–
7	reason	idReason	idDisaster (номер катастрофы)
8	disaster	idDisaster	–
9	people	idPeople	–
10	missingPeople	idPeople	–
11	profession	idProfession	–
12	equipment	idEquipment	idProfession (профессия, к которой прилагается данная экипировка);
13	coordinator	idTeam	idRegion (номер региона)

В дальнейшем построении схемы реляционной базы данных ключи будут служить для организации связей между отношениями.

Таким образом, ненормализованная схема базы данных представлена в приложении В на рисунке В.1.

2.3 Третья нормальная форма

Процесс преобразования базы данных к виду, отвечающему нормальным формам, называется нормализацией. Нормализация предназначена для приведения структуры базы данных к виду, обеспечивающему минимальную избыточность, то есть нормализация не имеет целью уменьшение или увеличение производительности работы или же уменьшение или увеличение объёма БД. Конечной целью нормализации является уменьшение потенциальной противоречивости хранимой в БД.

Для реляционных баз данных необходимо, чтобы все отношения базы данных обязательно находились в 1НФ. Нормальные формы более высокого порядка могут использоваться разработчиками по своему усмотрению. Однако грамотный специалист стремится к тому, чтобы довести уровень нормализации базы данных хотя бы до 3НФ, тем самым, исключив из базы данных избыточность и аномалии обновления.

Определение 3НФ – неключевые атрибуты не должны определять другие неключевые атрибуты.

Например, изначально Disaster была с полями typeReason, reason. Может быть несколько причин катастрофы и в данном варианте невозможно построить множество причин к одной катастрофе.

Поэтому необходимо привести эту таблицу к 3НФ. Результатом приведения будут две таблицы, приведенных на рисунке 2.3.

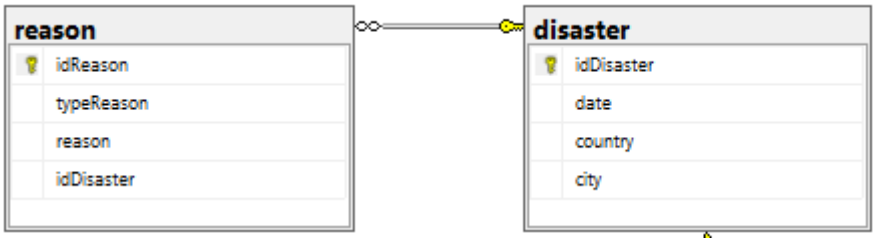

Рисунок 2.3 – Результат приведения таблицы Disaster к 3НФ

2.4 Определение ограничений целостности для внешних ключей отношений и для отношений в целом

Ограничение целостности отношений заключается в том, что в любом отношении должны отсутствовать записи с одним и тем же значением первичного ключа. Конкретно требование состоит в том, что любая запись любого отношения должна быть отличной от любой другой записи этого отношения. Это требование автоматически удовлетворяется, если в системе не нарушаются базовые свойства отношений.

У проектируемой базы следующие таблицы будут иметь первичные ключи: *team, teamPosition, people, missingPeople coordinator, user, profession, equipment, reason, disaster, region, department, mainDepartment.*

2.5 Графическое представление связей между внешними и первичными ключами

По результатам нормализации, определении первичных и внешних ключей, связей между сущностями, была получена схема реляционной базы данных, представленная в приложении В на рисунке В.2. На ней изображаются все отношения базы данных, а также связей между внешними и первичными ключами.

3 СОЗДАНИЕ СПРОЕКТИРОВАННОЙ БАЗЫ ДАННЫХ

Для реализации спроектированной базы данных была выбрана система управления базами данных SQL. Это обусловлено следующими причинами:

1. SQL довольно широко распространена;
2. является свободно распространяемой;
3. SQL поддерживается большинством применяемых средств доступа к данным.

В создаваемой базе данных будут использоваться следующие типы данных [3]:

1. int – целочисленный тип. Размер – 4 байта;
2. nvarchar – строковый тип;
3. date – тип, определяющий дату.

Опишем все таблицы, которые будут созданы в базе данных.

Таблица *user* содержит данные зарегистрированного пользователя. Ее структура приведена в таблице 3.1.

Таблица 3.1 – Характеристика атрибутов таблицы user

Имя атрибута	**Тип**	**Описание**
idUser	int	Идентификатор пользователя. Ключевой атрибут
login	nvarchar(50)	Логин пользователя
password	nvarchar(50)	Пароль пользователя
family	nvarchar(50)	Фамилия пользователя
name	nvarchar(50)	Имя пользователя
middleName	nvarchar(50)	Отчество пользователя
idProfession	int	Профессия пользователя
idTeam	int	Номер команды пользователя. Необязательный атрибут

Таблица *team* содержит данные команды: её название, текущие задания. Ее структура приведена в таблице 3.2.

Таблица 3.2 – Характеристика атрибутов таблицы team

Имя атрибута	**Тип**	**Описание**
idTeam	int	Идентификатор команды. Ключевой атрибут
teamName	nvarchar(50)	Название команды
idDisaster	int	Номер катастрофы. Необязательный атрибут

Окончание таблицы 3.2

idPeople	int	Номер пропавшего. Необязательный атрибут

Таблица *teamPosition* содержит данные позиций команды. Ее структура приведена в таблице 3.3.

Таблица 3.3 – Характеристика атрибутов таблицы teamPosition

Имя атрибута	Тип	Описание
idTeam	int	Идентификатор команды. Ключевой атрибут.
seniorLifeguard	int	Номер пользователя с профессией главный спасатель. Необязательный атрибут
doctor	int	Номер пользователя с профессией доктор. Необязательный атрибут
paramedic	int	Номер пользователя с профессией фельдшер. Необязательный атрибут
driver	int	Номер пользователя с профессией водитель. Необязательный атрибут
pyrotechnist	int	Номер пользователя с профессией пиротехник. Необязательный атрибут
gasWelder	int	Номер пользователя с профессией газосварщик. Необязательный атрибут
craneOperator	int	Номер пользователя с профессией оператор крана. Необязательный атрибут
scubaDriver	int	Номер пользователя с профессией дайвер. Необязательный атрибут
hydraulicTechnician	int	Номер пользователя с профессией гидравлик. Необязательный атрибут
scout	int	Номер пользователя с профессией разведчик. Необязательный атрибут
electrician	int	Номер пользователя с профессией электрик. Необязательный атрибут
rescuer	int	Номер пользователя с профессией зам главного спасателя. Необязательный атрибут

Таблица *region* содержит данные региона. Ее структура приведена в таблице 3.4.

Таблица 3.4 – Характеристика атрибутов таблицы region

Имя атрибута	Тип	Описание
idRegion	int	Идентификатор региона. Ключевой атрибут.
address	nvarchar(150)	Адрес региона
idDepartment	int	Идентификатор департамента

Таблица *department* содержит данные департамента. Ее структура приведена в таблице 3.5.

Таблица 3.5 – Характеристика атрибутов таблицы department

Имя атрибута	Тип	Описание
idDepartment	int	Идентификатор департамента. Ключевой атрибут.
address	nvarchar(150)	Адрес департамента
idMainDepartment	int	Идентификатор главного департамента

Таблица *mainDepartment* содержит данные главного департамента. Ее структура приведена в таблице 3.6.

Таблица 3.6 – Характеристика атрибутов таблицы mainDepartment

Имя атрибута	Тип	Описание
idMainDepartment	int	Идентификатор главного департамента. Ключевой атрибут.
address	nvarchar(150)	Адрес главного департамента

Таблица *reason* содержит данные причины катастрофы. Ее структура приведена в таблице 3.7.

Таблица 3.7 – Характеристика атрибутов таблицы reason

Имя атрибута	Тип	Описание
idReason	int	Идентификатор причины катастрофы Ключевой атрибут.
typeReason	nvarchar(50)	Тип катастрофы
reason	nvarchar(50)	Описание катастрофы
idDisaster	int	Идентификатор катастрофы

Таблица *disaster* содержит данные катастрофы. Ее структура приведена в таблице 3.8.

Таблица 3.8 – Характеристика атрибутов таблицы disaster

Имя атрибута	Тип	Описание
idDisaster	int	Идентификатор катастрофы Ключевой атрибут.
date	date	Дата катастрофы
country	nvarchar(50)	Страна, где произошла катастрофа
city	nvarchar(50)	Город, где произошла катастрофа

Таблица *people* содержит данные человека. Ее структура приведена в таблице 3.9.

Таблица 3.9 – Характеристика атрибутов таблицы people

Имя атрибута	Тип	Описание
idPeople	int	Идентификатор человека. Ключевой атрибут.
family	nvarchar(50)	Фамилия человека
name	nvarchar(50)	Имя человека
middleName	nvarchar(50)	Отчество человека
dateOfBirth	date	Дата рождения человека

Таблица *missingPeople* содержит данные пропавшего человека. Ее структура приведена в таблице 3.10.

Таблица 3.10 – Характеристика атрибутов таблицы missingPeople

Имя атрибута	Тип	Описание
idPeople	int	Идентификатор человека. Ключевой атрибут.
dateOfLoss	date	Дата пропажи
lastLocation	nvarchar(200)	Последнее место, где видели человека
specialSign	nvarchar(200)	Особые приметы человека

Таблица *professin* содержит данные о профессии. Ее структура приведена в таблице 3.11.

Таблица 3.11 – Характеристика атрибутов таблицы profession

Имя атрибута	Тип	Описание
idProfession	int	Идентификатор профессии. Ключевой атрибут.
position	nvarchar(50)	Название профессии

Таблица *equipment* содержит данные о оборудовании. Ее структура приведена в таблице 3.12.

Таблица 3.12 – Характеристика атрибутов таблицы equipment

Имя атрибута	Тип	Описание
idEquipment	int	Идентификатор оборудования. Ключевой атрибут.
idProfession	int	Идентификатор профессии, к которой относиться данная экипировка
equipmentName	nvarchar(100)	Название экипировки
type	nvarchar(50)	Тип экипировки
description	nvarchar(200)	Описание экипировки

Таблица *coordinator* содержит данные о координаторе. Ее структура приведена в таблице 3.13.

Таблица 3.13 – Характеристика атрибутов таблицы coordinator

Имя атрибута	Тип	Описание
idTeam	int	Идентификатор номера команды. Ключевой атрибут.
idRegion	int	Идентификатор номера региона. Необязательный атрибут.

4 ЗАПИСЬ ВЫРАЖЕНИЙ УКАЗАННЫХ В ВАРИАНТЕ ЗАДАНИЯ ТИПОВ ЗАПРОСОВ НА ЯЗЫКЕ SQL

Триггер на обновление номера команды у пользователей представлен в листинге 4.1.

Листинг 4.1 – Триггер на обновление номера команды у пользователей

```
1: create trigger [dbo].[TeamPosistion]
2: on [PSO].[dbo].[teamPosition]
3: after insert, update
4: as
5: if exists(select * from deleted)
6: begin
7: update [PSO].[dbo].[user]
8: Set idTeam = null
9: from deleted  as d
10: where [PSO].[dbo].[user].idUser = d.seniorLifeguard or
11: [PSO].[dbo].[user].idUser = d.doctor or
12: [PSO].[dbo].[user].idUser = d.paramedic or
13: [PSO].[dbo].[user].idUser = d.driver or
14: [PSO].[dbo].[user].idUser = d.pyrotechnist or
15: [PSO].[dbo].[user].idUser = d.gasWelder or
16: [PSO].[dbo].[user].idUser = d.craneOperator or
17: [PSO].[dbo].[user].idUser = d.scubaDriver or
18: [PSO].[dbo].[user].idUser =
19: d.hydraulicEquipmentTechnician or
20: [PSO].[dbo].[user].idUser =d.scout or
21: [PSO].[dbo].[user].idUser = d.electrician or
22: [PSO].[dbo].[user].idUser = d.rescuer
23:       DECLARE @id2 AS INT
24:       SELECT @id2 = idTeam
25:       FROM INSERTED
26: update [PSO].[dbo].[user]
27: Set idTeam = @id2
28: from inserted  as i
29: where [PSO].[dbo].[user].idUser = i.seniorLifeguard or
30: [PSO].[dbo].[user].idUser = i.doctor or
31: [PSO].[dbo].[user].idUser = i.paramedic or
32: [PSO].[dbo].[user].idUser = i.driver or
33: [PSO].[dbo].[user].idUser = i.pyrotechnist or
34: [PSO].[dbo].[user].idUser = i.gasWelder or
35: [PSO].[dbo].[user].idUser = i.craneOperator or
36: [PSO].[dbo].[user].idUser = i.scubaDriver or
37: [PSO].[dbo].[user].idUser =
38: i.hydraulicEquipmentTechnician or
39: [PSO].[dbo].[user].idUser = i.scout or
40: [PSO].[dbo].[user].idUser = i.electrician or
41: [PSO].[dbo].[user].idUser = i.rescuer
42: end
43: else
44: begin
```

```
45:       DECLARE @id AS INT
46:       SELECT @id = idTeam
47:       FROM INSERTED
48: update [PSO].[dbo].[user]
49: Set idTeam = @id
50: from inserted  as i
51: where [PSO].[dbo].[user].idUser = i.seniorLifeguard or
52: [PSO].[dbo].[user].idUser = i.doctor or
53: [PSO].[dbo].[user].idUser = i.paramedic or
54: [PSO].[dbo].[user].idUser = i.driver or
55: [PSO].[dbo].[user].idUser = i.pyrotechnist or
56: [PSO].[dbo].[user].idUser = i.gasWelder or
57: [PSO].[dbo].[user].idUser = i.craneOperator or
58: [PSO].[dbo].[user].idUser = i.scubaDriver or
59: [PSO].[dbo].[user].idUser =
60: i.hydraulicEquipmentTechnician or
61: [PSO].[dbo].[user].idUser = i.scout or
62: [PSO].[dbo].[user].idUser = i.electrician or
63: [PSO].[dbo].[user].idUser = i.rescuer
64: End
```

Далее используются пользовательские функции, которые выполняют следующие задания:

1. получают фамилию имя отчество пользователя по его id (листинг 4.2);
2. получают новую таблицу с данными пропавших людей (листинг 4.3);
3. получают данные экипировки для каждой профессии (листинг 4.4);
4. получают данные о катастрофе(листинг 4.5);

Листинг 4.2 – Функция фамилии, имени, отчества пользователя

```
1: create function [dbo].[GetFIO](@id int)
2: returns varchar(150)
3: begin
4: declare @fio varchar(150)
5: select @fio=us.family + ' ' + us.name + ' ' +
6: us.middleName
7: from [PSO].[dbo].[user] as us
8: where us.idUser =@id;
9: return @fio
10: end
```

Листинг 4.3 – Функция получения данных пропавшего человека

```
1: create FUNCTION [dbo].[GetPeopleData]()
2: RETURNS TABLE
3: AS
4: RETURN
5: (
6: (Select people.idPeople, people.family, people.name,
```

```
7: people.middleName, people.dateOfBirth ,
8: missingPeople.dateOfLoss, missingPeople.lastLocation,
9: missingPeople.specialSign
10: from [PSO].[dbo].[missingPeople] as missingPeople,
11: [PSO].[dbo].[people] as people
12: where people.idPeople = missingPeople.idPeople)
13: )
```

Листинг 4.4 – Функция данных экипировки для каждой профессии

```
1: create FUNCTION [dbo].[GetEquipmentData]()
2: RETURNS TABLE
3: AS
4: RETURN
5: (
6: (Select equipment.idEquipment, equipment.description,
7: equipment.type,
8: equipment.equipmentName,profession.position
9: from [PSO].[dbo].[profession] as profession,
10: [PSO].[dbo].[equipment] as equipment
11: where profession.idProfession =
12: equipment.idProfession)
13: )
```

Листинг 4.5 – Функция получения данных о катастрофе

```
1: create FUNCTION [dbo].[GetDisasterData]()
2: RETURNS TABLE
3: AS
4: RETURN
5: (
6: (Select reason.idReason, reason.reason,
7: reason.typeReason, disaster.city, disaster.country,
8: disaster.date
9: from [PSO].[dbo].[disaster] as disaster,
10: [PSO].[dbo].[reason] as reason
11: where disaster.idDisaster = reason.idDisaster)
12: )
```

5 ВЫБОР И ОСНОВАНИЕ СРЕДСТВ РАЗРАБОТКИ ПРИЛОЖЕНИЯ

Выбор СУБД является сложной задачей и должен основываться, в первую очередь, на потребностях с точки зрения информационной системы и пользователей. Определяющими здесь являются вид программного продукта и категория пользователей (или профессиональные программисты, или конечные пользователи, или и то, и другое). Другими показателями, влияющими на выбор СУБД, являются [2]:

1. удобство и простота использования;
2. качество средств разработки, защиты и контроля базы данных;
3. уровень коммуникационных средств в случае применения ее в сетях;
4. фирма-разработчик;
5. стоимость.

СистемаSQL Server позволяет обращаться к данным из любого приложения, разработанного с применением технологий Microsoft .NET и Visual Studio. SQL Server обеспечивает высочайший уровень безопасности, надежности и масштабируемости для критически важных приложений. Чтобы использовать новые возможности, постоянно возникающие в быстро меняющемся деловом мире, предприятиям нужно быть способными быстро создавать и развертывать решения, управляемые данными. MsSQL Server позволяет сократить затраты времени и средств, требуемые на управление и развертывание таких приложений. Также следует учесть, что фирма-разработчик данной СУБД является также разработчиком самой распространенной ОС. В финансовом плане важным фактором является то, что существуют бесплатные сборки данной СУБД (Express).

Для реализации приложения была выбрана среда разработки Microsoft Visual Studio 2019, в качестве языка программирования – C#.

Достоинства платформы .NET [1]:

1. Вся платформа .NET основана на единой объектно-ориентированной модели. Все сервисы, интерфейсы и объекты, которые платформа предоставляет разработчику объединены в единую иерархию классов. Другими словами, все, что может вам потребоваться при создании приложений под платформу .NET будет всегда у вас под рукой. Причем, все это сгруппировано очень удобно и интуитивно понятно.
2. Приложение, написанное на любом .NET-совместимом языке, является межплатформенным (в идеале). Почему в идеале? Дело в том, что приложение, написанное, скажем, на том же C#, не зависит от платформы, на которой будет выполняться, но зато зависит от наличия платформы .NET.
3. В состав платформы .NET входит «сборщик мусора», который освобождает ресурсы. Таким образом, приложения защищены от утечки памяти и от необходимости освобождать ресурсы. Это делает программирование более легким и более безопасным.

4. Приложения .NET используют безопасные типы, что повышает их надежность и совместимость.

5. .NET приложения могут быть сертифицированы на безопасность. Это является особенность промежуточного кода, в который преобразуются все .NET приложения.

6. Абсолютно все ошибки обрабатываются механизмом исключительных ситуаций. Это позволяет избежать разногласий, которые иногда возникают при программировании под Win32.

7. Повторное использование кода стало еще удобнее. Это связано с тем, что промежуточный язык MSIL не зависит от языка программирования. Например, вы можете написать программу на C#, а патч к ней писать уже, скажем, на J#.

Для связи программного кода с базой данных используется Entity Framework Core.

Entity Framework Core (EF Core) представляет собой объектно-ориентированную, легковесную и расширяемую технологию от компании Microsoft для доступа к данным. EF Core является ORM-инструментом (object-relational mapping - отображения данных на реальные объекты). То есть EF Core позволяет работать базами данных, но представляет собой более высокий уровень абстракции: EF Core позволяет абстрагироваться от самой базы данных и ее таблиц и работать с данными независимо от типа хранилища. Если на физическом уровне мы оперируем таблицами, индексами, первичными и внешними ключами, но на концептуальном уровне, который нам предлагает Entity Framework, мы уже работаем с объектами.

Entity Framework Core поддерживает множество различных систем баз данных. Таким образом, мы можем через EF Core работать с любой СУБД, если для нее имеется нужный провайдер.

По умолчанию на данный момент Microsoft предоставляет ряд встроенных провайдеров: для работы с MS SQL Server, для SQLite, для PostgreSQL. Также имеются провайдеры от сторонних поставщиков, например, для MySQL.

Также стоит отметить, что EF Core предоставляет универсальный API для работы с данными. И если, к примеру, мы решим сменить целевую СУБД, то основные изменения в проекте будут касаться прежде всего конфигурации и настройки подключения к соответствующим провайдерам. А код, который непосредственно работает с данными, получает данные, добавляет их в БД и т.д., останется прежним.

Entity Framework Core многое унаследовал от своих предшественников, в частности, Entity Framework 6. В тоже время надо понимать, что EF Core - это не новая версия по отношению к EF 6, а совершенно иная технология, хотя в целом принципы работы у них будут совпадать. Поэтому в рамках EF Core используется своя система версий. Текущая версия – 5.0 была выпущена в ноябре 2020 года.

Как технология доступа к данным Entity Framework Core может использоваться на различных платформах стека .NET. Это и стандартные платформы типа Windows Forms, консольные приложения, WPF, UWP и ASP.NET Core. При этом кроссплатформенная природа EF Core позволяет задействовать ее не только на ОС Windows, но и на Linux и Mac OS X.

Центральной концепцией Entity Framework является понятие сущности или entity. Сущность определяет набор данных, которые связаны с определенным объектом. Поэтому данная технология предполагает работу не с таблицами, а с объектами и их коллекциями.

Любая сущность, как и любой объект из реального мира, обладает рядом свойств. Например, если сущность описывает человека, то мы можем выделить такие свойства, как имя, фамилия, рост, возраст. Свойства необязательно представляют простые данные типа int или string, но могут также представлять и более комплексные типы данных. И у каждой сущности может быть одно или несколько свойств, которые будут отличать эту сущность от других и будут уникально определять эту сущность. Подобные свойства называют ключами.

При этом сущности могут быть связаны ассоциативной связью один-ко-многим, один-ко-одному и многие-ко-многим, подобно тому, как в реальной базе данных происходит связь через внешние ключи.

Отличительной чертой Entity Framework Core, как технологии ORM, является использование запросов LINQ для выборки данных из БД. С помощью LINQ мы можем создавать различные запросы на выборку объектов, в том числе связанных различными ассоциативными связями. А Entity Framework при выполнение запроса транслирует выражения LINQ в выражения, понятные для конкретной СУБД (как правило, в выражения SQL).

6 РЕАЛИЗАЦИЯ ЗАКОНЧЕННОГО ПРИЛОЖЕНИЯ, РАБОТАЮЩЕГО С СОЗДАННОЙ БАЗОЙ ДАННЫХ

6.1 Разработка и построение интерфейса главной и рабочих форм

Прежде всего, в виду необходимости защиты информации, находящейся в базе, только после того, как пройдет удачная Windows-аутентификации пользователь может увидеть одну из 3 главных форм, в зависимости от его профессии:

1. админ – оперирует данными катастроф, регионов, пропавших людей, профессии (добавляет, редактирует, удаляет), для его регистрации требуется дополнительный пароль админа;
2. координатор – составляет свою команду, выбирает текущие задания команды, выбирает регион, в котором работает. Для его регистрации требуется дополнительный пароль координатора;
3. все остальные профессии – владеют информацией своей экипировки, информацией об участниках команды (если такова имеется), и текущих заданий команды (если таковы имеются).

Главные формы построены по принципу меню (более подробно в пункте 6.2). Все остальные формы являются всплывающими, они закрывают форму меню, что позволяет не вызывать путаницы в окнах.

Внешний вид главных форм можно увидеть в приложении В (рисунок В.1-В.3)

Обращение к базе данных происходит благодаря модели ADO.NET EDM. Которая в себе содержит все таблицы и функции, а также связь между таблицами и диаграмму таблиц.

При разработке интерфейса форм главным был принцип предотвращения ошибок, а не их констатации, потому были использованы (по возможности), к примеру, интерактивные обращения к базе еще на этапе заполнения форм. Таким образом, частично проверяется уникальность имен, номеров; также для всех полей, в которые должны быть занесены сугубо числовые данные, сделан соответствующий фильтр, что также позволяет предотвратить некорректные данные в базе и пугающие неопытного пользователя сообщения об ошибках.

6.2 Построение главного меню и кнопок панели инструментов

Главные окна представлены в виде меню, с содержанием определенных количеств кнопок, каждая кнопка открывает определенное окно, где можно совершить определенные действия для каждого пользователя.

6.3 Выполнение программного кода на языке C#

Опишем работу приложения с базой данных. Для подключения к базе будем использовать Entity Framework (листинг 6.1). Для подключения к базе данных использую данную строку:

Листинг 6.1 – Подключение базы данных к коду

```
1: var context = new PSOConnect();
```

Для заполнения пропавшего человека используется следующий метод (листинг 6.2):

Листинг 6.2 – Метод заполнения данными о пропавшем человеке

```
1: private void AddMissingPeople()
2: {
3: var context = new PSOConnect();
4: var people = context.people.FirstOrDefault(peoples =>
5: peoples.family.Equals(FamilyField.Text) &&
6: peoples.name.Equals(NameField.Text) &&
7: peoples.middleName.Equals(MiddleNameField.Text) &&
8: EntityFunctions.TruncateTime(peoples.dateOfBirth.Value)
9: == EntityFunctions.TruncateTime(DateOfBirthField.Value));
10: var missingPeople =
11: context.missingPeople.FirstOrDefault(missingPeoples =>
12: missingPeoples.specialSign.Equals(SpecialSignField.Text)
13: &&
14: missingPeoples.lastLocation.Equals(LastLocationField.Tex
15: t) &&
16: EntityFunctions.TruncateTime(missingPeoples.dateOfLoss.V
17: alue) ==
18: EntityFunctions.TruncateTime(DateOfLossField.Value));
19: if (people == null || missingPeople == null) {
20: var newPeople = new people
21: {
22: idPeople = context.people.Count() > 0 ?
23: context.people.Max(idPeople => idPeople.idPeople) + 1 :
24: 1,
25: family = FamilyField.Text,
26: name = NameField.Text,
27: middleName = MiddleNameField.Text,
28: dateOfBirth = DateOfBirthField.Value};
29: var newMissingPeople = new missingPeople{
30: idPeople = context.missingPeople.Count() > 0 ?
31: context.missingPeople.Max(idPeople => idPeople.idPeople)
32: + 1 : 1,
33: specialSign = SpecialSignField.Text,
34: lastLocation = LastLocationField.Text,
35: dateOfLoss = DateOfLossField.Value};
36: context.people.Add(newPeople);
37: context.missingPeople.Add(newMissingPeople);
38: context.SaveChanges();}}
```

ЗАКЛЮЧЕНИЕ

В результате выполненной курсового проекта была создана база данных поисково-спасательного отряда, а также программный продукт, позволяющий пользователю взаимодействовать с базой данных. Базы данных была разработана в среде MySQL, приложение – Visual Studio 2019.

Приложение позволяет:

1. получать требуемую информацию;
2. редактировать требуемую информацию;
3. удалять требуемую информацию;
4. заполнять требуемые поля.

Запросы, указанные в варианте задания, были «растворены» в приложении.

Использовался Entity Framework Core, для подключения к базе данный PSO. Данный фрэймворк позволил значительно сократить количество кода, для составления запросов, а также исключить возможные ошибки и баги в ходе выполнения курсового проекта.

Также была усвоена методика составление динамической, гибкой, легко расширяемой базы данных, что позволяет изменять её без какого либо вреда данным.

В процессе выполнения данной курсового проекта были закреплены навыки в программировании на Visual Studio 2019, проектировании баз данных и реализации их в СУБД MsSQL.

В ходе выполнения данного проекта, не было замечено критических ошибок в работе программы и базе данных, процесс программы не прерывался аварийно.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Data flow diagram - Wikipedia, the free encyclopedia / Многоязычная общедоступная свободно распространяемая энциклопедия, Википедия [Электронный ресурс] / «Нупедия». – Вашингтон, 2000. – Режим доступа: http://en.wikipedia.org/wiki/Data_flow_diagram. – Дата доступа: 29.11.2020.

2. Entity-relationship model – Wikipedia, the free encyclopedia / Многоязычная общедоступная свободно распространяемая энциклопедия, Википедия [Электронный ресурс] / «Нупедия». – Вашингтон, 2000. – Режим доступа: http://en.wikipedia.org/wiki/Entity-relationship_model. – Дата доступа: 29.11.2020.

3. MySQL 5.0 Reference Manual [Электронный ресурс] – Лондон 2014. – электрон. опт. Диск (DVD-RW).

4. Документация по языку C# [Электронный ресурс]. – Лос-Анджелес, 2007. – Режим доступа https://docs.microsoft.com/en-us/dotnet/csharp/. – Дата доступа: 15.11.2020.

5. Документация по SQL Connector [Электронный ресурс]. – Бостон, 2000. – Режим доступа: https://dev.mysql.com/doc/connector-net/en/– Дата доступа: 18.11.2020.

6. Unified Modeling Language – Wikipedia, the free encyclopedia / Многоязычная общедоступная свободно распространяемая энциклопедия, Википедия [Электронный ресурс] / «Нупедия». – Вашингтон, 2000. – Режим доступа: http://en.wikipedia.org/wiki/Unified_Modeling_Language. – Дата доступа: 29.11.2020.

ПРИЛОЖЕНИЯ