**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ АСТРАХАНСКОЙ ОБЛАСТИ**

**Государственное бюджетное профессиональное образовательное учреждение  
Астраханской области «Астраханский колледж вычислительной техники»**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **К защите допустить:** | | | | | | | |
| **Старший методист УМС** | | | | | | | |
|  |  | | | **Л.О.Земцова** | | | |
| **«** |  | **»** |  | | **20** |  | **г.** |

**курсовая работа**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| по МДК | 04.01 | Технология разработки и защиты баз данных | | | | | | |
|  | | | | | | | | |
|  | | | 6 семестр | | | | | |
| Тема курсовой работы | | | Разработка базы данных | | | | | |
|  | | | ***«***Ассоциация фермерских хозяйств» | | | | | |
|  | | |  | | | | | |
| АКВТ.09.02.07.КР00.02ПЗ | | | | | | | | |
| Листов: | | | | 40 |  | | | |
|  | |  | | | | | | |
|  | |  | | | | | | |
| Выполнил | |  | | | | Студент гр. | | ПБ – 31 |
|  | |  | | | | Адаменко Р.Р. | | |
|  | |  | | | |  |  | |
| Проверил | |  | | | | Преподаватель | | |
|  | |  | | | | Мулеев Р.Р. | | |

**2022**

**Государственное бюджетное профессиональное образовательное учреждение**

**Астраханской области «Астраханский колледж вычислительной техники»**

|  |
| --- |
| УТВЕРЖДЕНО |
| Цикловой комиссией |
| специальности 09.02.07 |
| ПЦК \_\_\_\_\_\_\_\_\_ Ю.С. Андрианова |
| «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г. |
| Специальность | *09.02.07 Информационные системы и программирование* | | |
| МДК 04.01 | *Технология разработки и защиты баз данных* | | |
| Группа | *ПБ– 31* | | |
| **ЗАДАНИЕ** | | | |
| **на курсовое проектирование** | | | |
| ***Адаменко Реналю Рамильевичу*** | | | |
| (фамилия, имя, отчество студента) | | | |
| Тема курсового проекта | | ***Разработка БД*** | |
| ***«Ассоциация фермерских хозяйств»*** | | | |

Содержание пояснительной записки курсового проекта

Содержание

Введение

1. Проектирование базы данных
2. Анализ и описание предметной области информационной системы
3. Проектирование концептуальной модели
4. Построение логической модели, схемы базы данных
5. Выбор СУБД
6. Реализация базы данных
7. Физическое проектирование
8. Написание исходного кода БД
9. Написание запросов, функций и хранимых процедур

Заключение

Список использованных источников

Приложения

|  |  |  |  |
| --- | --- | --- | --- |
| Руководитель |  |  | Р.Р. Мулеев |
| Студент |  |  | Р.Р. Адаменко |

**Аннотация**

Данный документ на разработку базы данных имеет 40 страниц, 4 раздела и 7 подразделов, 3 рисунка, 4 источника использованной литературы. Таблицы и приложения отсутствуют.

**Содержание**

[Введение 5](#_Toc105580235)

[1. Проектирование БД 6](#_Toc105580236)

[1.1 Анализ и описание предметной области информационной системы 6](#_Toc105580237)

[1.2 Проектирование концептуальной модели 7](#_Toc105580238)

[1.2.1 Сущности 7](#_Toc105580239)

[1.2.2 Связи 8](#_Toc105580240)

[1.3 Построение логической модели, схемы базы данных 9](#_Toc105580241)

[1.4 Выбор СУБД 14](#_Toc105580242)

[2 Реализация базы данных 16](#_Toc105580243)

[2.1 Физическое проектирование БД 16](#_Toc105580244)

[2.2 Написание исходного кода БД 16](#_Toc105580245)

[2.3 Написание запросов, функций и хранимых процедур 21](#_Toc105580246)

[Заключение 39](#_Toc105580247)

[Список использованных источников 40](#_Toc105580248)

# Введение

Современные условия для работы по обслуживанию клиентов достигло того уровня, когда обслуживающая система предприятия обрабатывает данные с очень большой скоростью в многопользовательском режиме. Весь документооборот осуществляется на компьютерах, и вся информация храниться в специальных базах, потому возрастает значение информационных систем, позволяющих обеспечить информационную поддержку процессов принятия решений. Базы данных являются одним из основных элементов большинства информационных систем.

База данных (далее БД) – это организованная структура, в которой в специальном формате хранится информация.

Система управления базой данных (далее СУБД) – это программа, с помощью которой в компьютер вводится информация, просматривается, сортируется, фильтруется, разыскивается, экспортируется, импортируется.

СУБД позволяют размещать в своих структурах не только данные, но и методы, с помощью которых происходит взаимодействие с потребителем или с другими программно-аппаратными комплексами.

Рассматривая такую предметную область как ассоциация фермерских хозяйств, невозможно обойтись без структурирования информации в БД.

БД «Ассоциация фермерских хозяйств» предназначена для структуризации информации об фермерских хозяйствах и их владельцах, продуктах, производимых хозяйствами, видов их деятельности и специализации, формирования отчетности.

# Проектирование БД

## Анализ и описание предметной области информационной системы

Ассоциация крестьянских хозяйств (далее АКХ) - добровольное объединение на принципах индивидуального членства крестьянских хозяйств, является некоммерческой организацией, учрежденной малыми и средними формами сельскохозяйственных товаропроизводителей.

Цели АКХ:

* защита прав и интересов крестьянства на территории страны;
* координация их предпринимательской деятельности, повышение эффективности крестьянского сектора экономики;
* активное содействие количественному и качественному росту фермерства в стране;
* и другие цели, прописанные в уставе АКХ.

Предметная область представляет собой базу данных, направлена на информационную поддержку АКХ. База данных даёт возможность проведение аналитики и создание отчётности по деятельности всей организации, а также каждого её членов отдельно, автоматизирует некоторые процессы, уменьшает человеческий фактор и связанные с этим возникающие ошибки. Ввиду всех возможностей базы данных возникает необходимость в автоматизации информационных процессов.

Документ, дающий основание разработки базы данных — это устав АКХ.

Входные документы:

* заявка на вступление в АКХ.

Выходные документы:

* отчёт о деятельности АКХ;
* отчёт об участнике АКХ.

База данных создана для сотрудников АКХ (могут просматривать, вносить, изменять, удалять информацию в базе данных в полном объёме), участников АКХ (могут просматривать общедоступную информацию и изменять информацию о себе), сторонних пользователей (могут просматривать общедоступную информацию).

Общедоступная информация определяется уставом АКХ.

База данных должна осуществлять хранение данных о:

* крестьянских фермерских хозяйствах (код, название хозяйства, специализация, личные данные фермера, регион, адрес, телефон);
* видах их деятельности (овощеводство, животноводство, виноградарство и другие);
* предлагаемой хозяйствами продукции (в соответствии со специализацией хозяйства - код хозяйства, произведенный товар, единицы измерения, цена за единицу товара, предлагаемое количество) и ее цене.

Необходимо обеспечить:

* ввод, обновление удаление данных;
* возможности анализа товаров и цен;
* выборку информации из базы данных о деятельности конкретных хозяйств: их продукции и ценах;
* выборку информации о конкретных видах товаров, их производителях и ценах.

## Проектирование концептуальной модели

## Сущности

1. Владелец - содержит информацию о владельце фермерского хозяйства.

Атрибуты: ФИО, номер телефона.

1. Хозяйство - содержит информацию о фермерском хозяйстве, которое входит в ассоциацию.

Атрибуты: адрес, название.

1. Вид деятельности - содержит информацию о виде деятельности фермерского хозяйства.

Атрибуты: наименование.

1. Продукция - содержит информацию о продукции, которая реализуется на фермерском хозяйстве.

Атрибуты: название, количество, цена.

1. Вид продукции - содержит информацию о виде продукции, которая может быть реализована на фермерском хозяйстве.

Атрибуты: наименование.

1. Регион - справочная сущность, которая содержит информацию о всех существующих регионах страны.

Атрибуты: регион.

## Связи

1. Владелец управляет фермерским хозяйством. У владельца может быть много хозяйств, а у хозяйства только один владелец (связь 1 к многим).
2. Хозяйство специализируется на определённом виде деятельности. У хозяйства может быть много видов деятельности, так же как много хозяйств могут иметь один и тот же вид деятельности (связь много ко многим).
3. Хозяйство находится в регионе. Хозяйство может находится только в одном регионе, а в регионе может находится множество хозяйств (связь 1 к многим). Хозяйство не может существовать без региона (следовательно, «Хозяйство» слабая сущность) а регион без хозяйства может (следовательно, «Регион» сильная сущность).
4. Хозяйство выпускает продукт. Хозяйство может выпускать множество разных продуктов, так же как продукты могут выпускаться различными хозяйствами (связь много ко многим). Продукт не может выпускаться без хозяйства (следовательно, «Продукт» слабая сущность), а хозяйство может не выпускать продукт (следовательно, «Хозяйство» сильная сущность).
5. Продукт является видом продукции. У продукта может быть только один вид, а у вида продукции может быть много продуктов (связь 1 к многим).

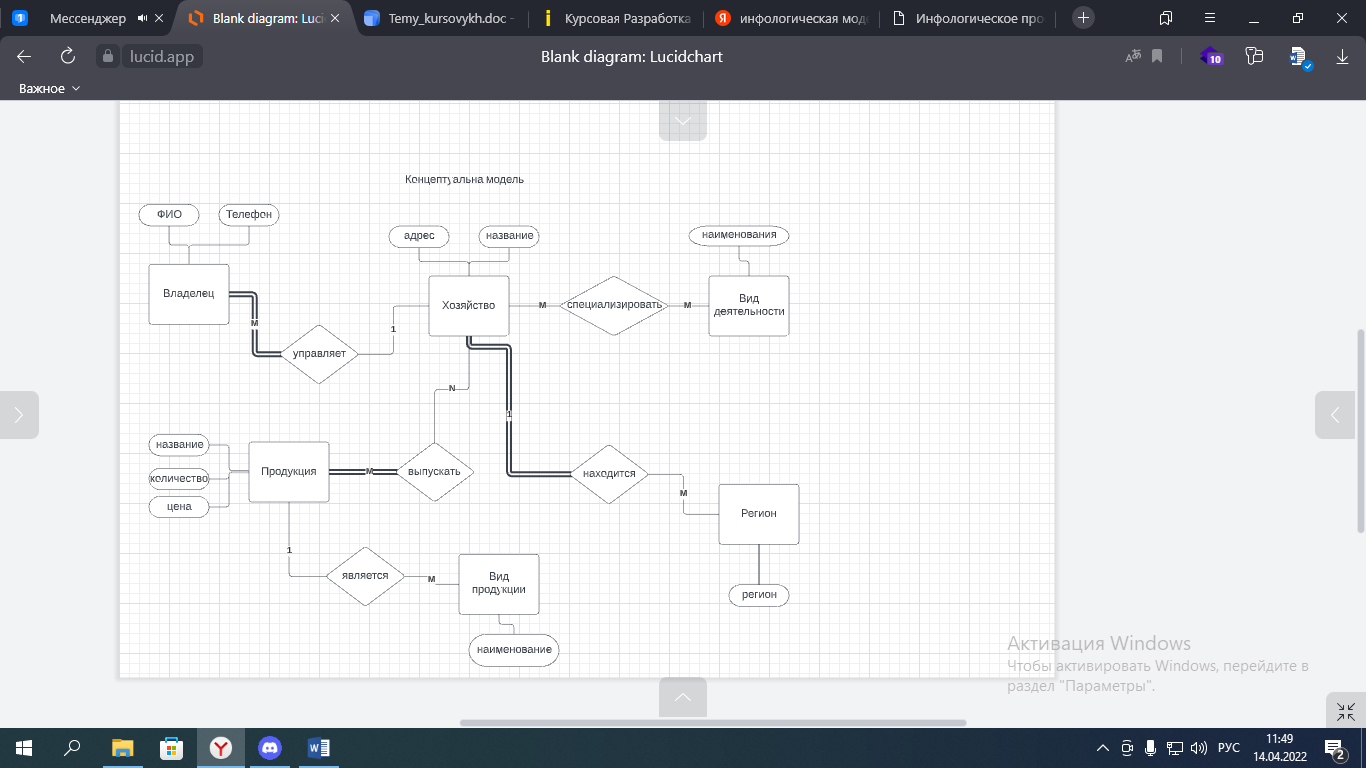


Рисунок 1 – Концептуальная модель

## Построение логической модели, схемы базы данных

Сущности:

1. Владелец (Owner):
   1. Идентификатор владельца (OwnerID):

* нулевое значение: not null;
* тип поля: int;
* первичный ключ, созданный искусственно из-за отсутствия атрибута идентификатора у сущности;
* автоинкримированный.
  1. Фамилия, имя, отчество (FirstName, MiddleName, LastName):
* нулевое значение: FirstName и MiddleName not null, LastName null;
* тип поля: nvarchar;
* маска ввода: только буквы;
* нормализированная форма атрибута ФИО.
  1. Номер телефона (PhoneNumber):
* нулевое значение: null;
* тип поля: nvarchar;
* маска ввода: международный стандарт телефонных номеров.
  1. Электронная почта (Email):
* нулевое значение: null;
* тип поля: nvarchar;
* маска ввода: стандарт электронных почт.
  1. ИНН (INN):
* нулевое значение: not null;
* тип поля: nvarchar;
* маска ввода: 12 цифр.

1. Регион (Region):
   1. Идентификатор региона (RegionID):

* нулевое значение: not null;
* тип поля: int;
* первичный ключ, созданный искусственно из-за отсутствия атрибута идентификатора у сущности;
* автоинкримированный.
  1. Регион (NameRegion):
* нулевое значение: not null;
* тип поля: nvarchar.

1. Хозяйство (Farm):
   1. Идентификатор хозяйства (FarmID):

* нулевое значение: not null;
* тип поля: int;
* первичный ключ, созданный искусственно из-за отсутствия атрибута идентификатора у сущности;
* автоинкримированный.
  1. Код единого государственного реестра недвижимости (codeUSRRE)
* нулевое значение: not null;
* тип поля: nvarchar;
* маска ввода: стандарт кода единого государственного реестра недвижимости.
  1. Название хозяйства (NameFarm):
* нулевое значение: not null;
* тип поля: nvarchar.
  1. Город (City), улица (Street), номер дома (NumberHouse):
* нулевое значение: null;
* тип поля: nvarchar;
* нормализованная форма атрибута Адрес.
  1. Номер телефона (PhoneNumber):
* нулевое значение: null;
* тип поля: nvarchar;
* маска ввода: международный стандарт телефонных номеров.
  1. Электронная почта (Email):
* нулевое значение: null;
* тип поля: nvarchar;
* маска ввода: стандарт электронных почт.
  1. Идентификатор региона (RegionID):
* нулевое значение: not null;
* тип поля: int;
* вторичный ключ, ссылающийся на таблицу Region (поле RegionID).
  1. Идентификатор владельца (OwnerID):
* нулевое значение: not null;
* тип поля: int;
* вторичный ключ, ссылающийся на таблицу Owner (поле OwnerID).

1. Специализация хозяйства (FarmSpecialization), разгрузочная таблица, созданная для нормализации связи между таблицами Специализация и Хозяйство:
   1. Идентификатор специализации хозяйства (FarmSpecializationID):

* нулевое значение: not null;
* тип поля: int;
* первичный ключ, созданный искусственно из-за отсутствия атрибута идентификатора у сущности;
* автоинкримированный.
  1. Идентификатор хозяйства (FarmID):
* нулевое значение: not null;
* тип поля: int;
* вторичный ключ, ссылающийся на таблицу Farm (поле FarmID).
  1. Идентификатор специальности (SpecializationID):
* нулевое значение: not null;
* тип поля: int;
* вторичный ключ, ссылающийся на таблицу Specialization (поле SpecializationID).

1. Специализация (Specialization):
   1. Идентификатор специализации (SpecializationID):

* нулевое значение: not null;
* тип поля: int;
* первичный ключ, созданный искусственно из-за отсутствия атрибута идентификатора у сущности;
* автоинкримированный.
  1. Наименование специализации (NameSpecialization)
* нулевое значение: not null;
* тип поля: nvarchar.

1. Продукция хозяйства (FarmProduct) разгрузочная таблица, созданная для нормализации связи между таблицами Продукция и Хозяйство:
   1. Идентификатор продукции хозяйства (FarmProductID):

* нулевое значение: not null;
* тип поля: int;
* первичный ключ, созданный искусственно из-за отсутствия атрибута идентификатора у сущности;
* автоинкримированный.
  1. Идентификатор фермы (FarmID):
* нулевое значение: not null;
* тип поля: int;
* вторичный ключ, ссылающийся на таблицу Farm (поле FarmID).
  1. Идентификатор продукции (ProductID):
* нулевое значение: not null;
* тип поля: int;
* вторичный ключ, ссылающийся на таблицу Product (поле ProductID).
  1. Количество (Quantity):
* нулевое значение: not null;
* тип поля: float;
* маска ввода: числа больше 0.
  1. Цена за единицу (UnitPrice):
* нулевое значение: not null;
* тип поля: float;
* маска ввода: числа больше 0.

1. Продукт (Product):
   1. Идентификатор продукта (ProductID):

* нулевое значение: not null;
* тип поля: int;
* первичный ключ, созданный искусственно из-за отсутствия атрибута идентификатора у сущности;
* автоинкримированный.
  1. Наименование продукта (NameProduct):
* нулевое значение: not null;
* тип поля: nvarchar.
  1. Единица измерения (Measure):
* нулевое значение: not null;
* тип поля: nvarchar;
* маска ввода: краткое название единиц измерения.
  1. Идентификатор вида продукта (TypeProductID):
* нулевое значение: not null;
* тип поля: int;
* вторичный ключ, ссылающийся на таблицу TypeProduct (поле TypeProductID).

1. Вид продукции (TypeProduct):
   1. Идентификатор продукта (TypeProductID):

* нулевое значение: not null;
* тип поля: int;
* первичный ключ, созданный искусственно из-за отсутствия атрибута идентификатора у сущности;
* автоинкримированный.
  1. Наименование вида продукта (NameTypeProduct):
* нулевое значение: not null;
* тип поля: nvarchar.

1. Вид деятельности (TypeActivity):
   1. Идентификатор вида деятельности (TypeProductID):

* нулевое значение: not null;
* тип поля: int;
* первичный ключ, созданный искусственно из-за отсутствия атрибута идентификатора у сущности;
* автоинкримированный.
  1. Наименование вида деятельности (NameTypeActivity):
* нулевое значение: not null;
* тип поля: nvarchar.

1. Вид деятельности хозяйства (FarmTypeActivity), разгрузочная таблица, созданная для нормализации связи между таблицами Вид деятельности и Хозяйство:
   1. Идентификатор вида деятельности хозяйства (FarmTypeActivityID):

* нулевое значение: not null;
* тип поля: int;
* первичный ключ, созданный искусственно из-за отсутствия атрибута идентификатора у сущности;
* автоинкримированный.
  1. Идентификатор фермы (FarmID):
* нулевое значение: null;
* тип поля: int;
* вторичный ключ, ссылающийся на таблицу Farm (поле FarmID).
  1. Идентификатор вида деятельности (TypeActivityID):
* нулевое значение: null;
* тип поля: int;
* вторичный ключ, ссылающийся на таблицу TypeActivity (поле TypeActivityID).

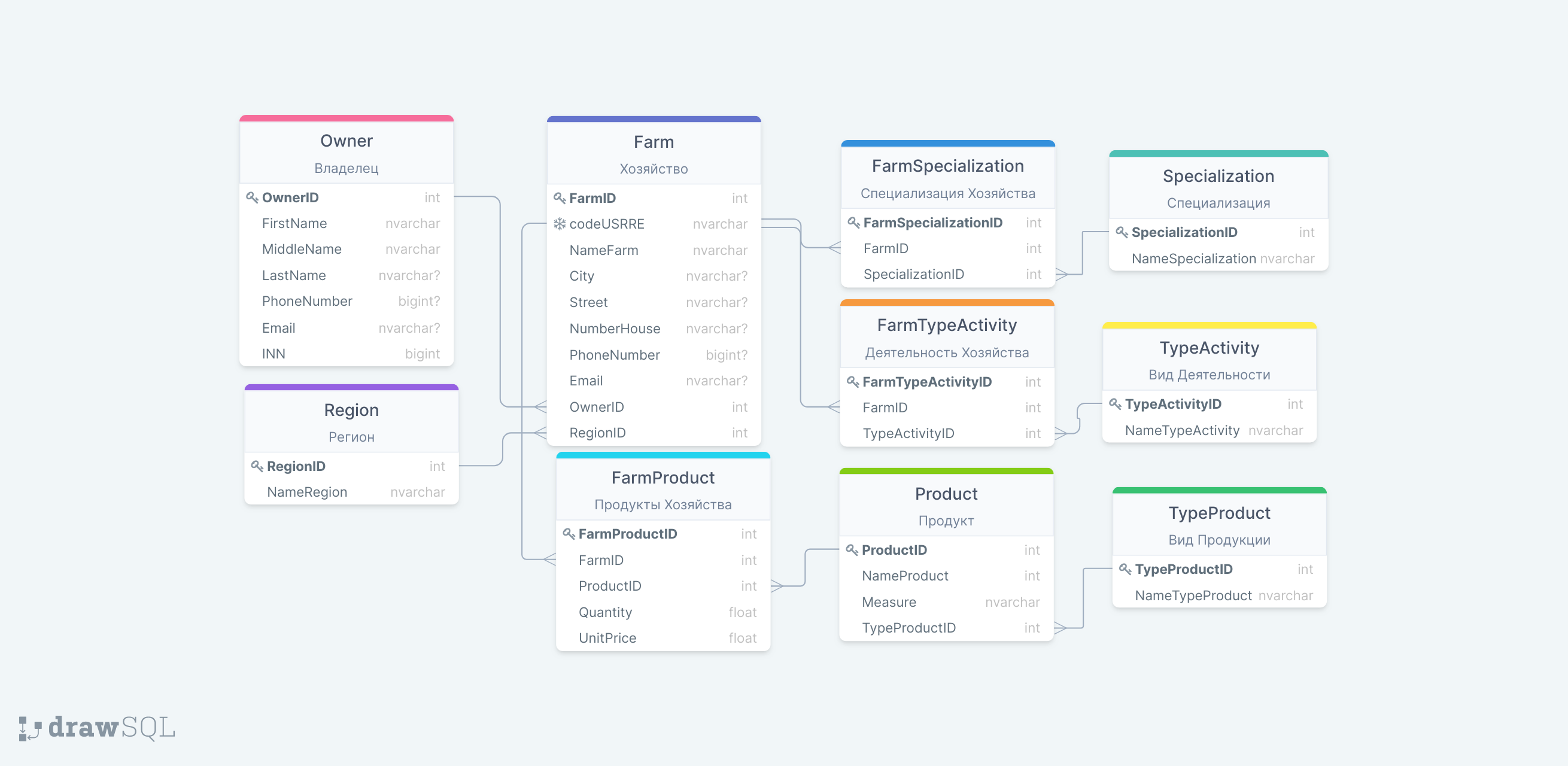


Рисунок 2 – Логическая модель

## Выбор СУБД

СУБД предназначена для централизованного управления базой данных в интересах, всех работающих в этой системе. Используемые в настоящее время СУБД, обладающих средствами обеспечения целостности данных и надёжной безопасности, что даёт возможность разработчикам гарантировать большую безопасность данных при меньших затратах сил на низкоуровневое программирование.

В качестве средства реализации БД была выбрана система Microsoft SQL Server Management Studio 2019 (далее MSSQL).

MSSQL– это мощная и надежная система управления данными, обеспечивающая множество функций, защиту данных и высокую производительность для внедренных приложений-клиентов и локальных хранилищ данных. MSSQL предназначен для упрощенного развертывания и быстрого создания прототипов; его можно получить бесплатно и свободно распространять вместе с приложениями. Также можно сказать, что он удовлетворяет требованиям, предъявляемым к системам распределенной обработки информации. Эта СУБД поддерживает: тиражирование данных, параллельную обработку, создание и обработку больших баз данных на недорогих аппаратных платформах.

MSSQL упрощает развёртывание, управление и оптимизацию данных предприятия и аналитических приложений. Как платформа управления данными предприятия, он предоставляет единую консоль управления, которая позволяет администраторам данных, находящимся в любом месте вашей организации, отслеживать, управлять и настраивать все базы данных и связанные службы по всему предприятию. Он предоставляет расширяемую инфраструктуру управления, которая может быть легко запрограммирована при помощи SQL Management Objects, позволяя пользователям переделывать и расширять их среду управления и независимым поставщикам программных продуктов создавать дополнительные инструменты и функциональность для дальнейшего расширения возможностей, поставляемых по умолчанию.

Средством, связывающим клиента с сервером, является язык SQL - язык структурированных запросов. В данном курсовом проекте были использованы основные операторы этого языка, в частности для создания запросов/процедур/триггеров. Этот язык является очень распространенным на данный момент и имеет ряд особенностей.

Он позволяет:

* создавать базы данных и таблицы с полным описанием их структуры;
* выполнять основные операции манипулирования данными, такие как вставка, модификация и удаление данных из таблиц;
* выполнять простые и сложные запросы.

# Реализация базы данных

После анализа предметной области базы данных «Ассоциация фермерских хозяйств» приступаем к этапу проектированию базы данных, которое заключается в определении состава полей её таблиц и связей между ними. От того, насколько тщательно проведен анализ и насколько грамотно спроектирована база данных, в существенной мере зависит эффективность будущего приложения и его полезность для пользователя.

## Физическое проектирование БД

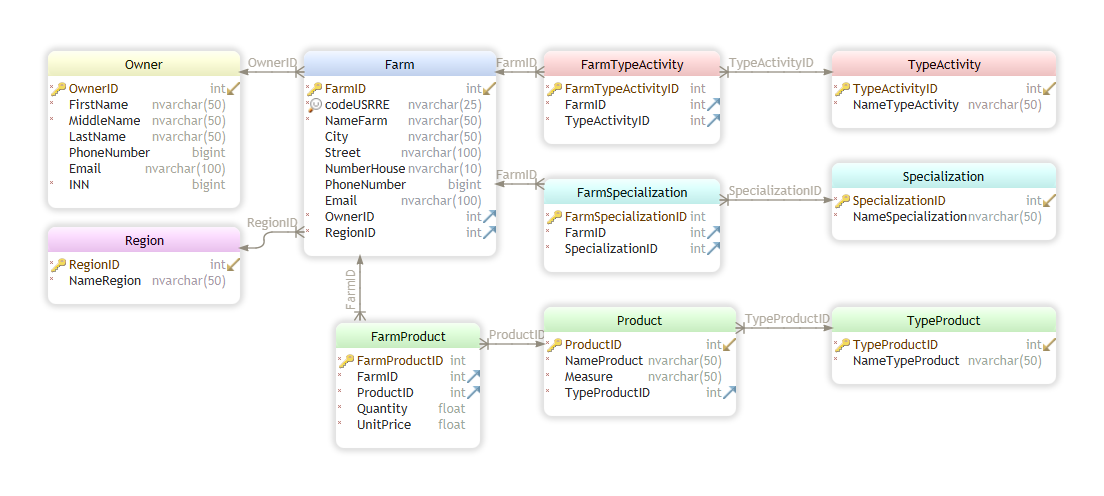


Рисунок 3 – Диаграмма базы данных

## Написание исходного кода БД

* 1. Owner

Данная таблица создана с целью хранения информации о владельце.

Листинг:

CREATE TABLE Owner(

OwnerID INT identity(1,1) NOT NULL,

FirstName NVARCHAR(50) NOT NULL,

MiddleName NVARCHAR(50) NOT NULL,

LastName NVARCHAR(50) NULL,

PhoneNumber BIGINT NULL,

Email NVARCHAR(100) NULL,

INN NVARCHAR(50) NOT NULL

);

ALTER TABLE

Owner ADD CONSTRAINT owner\_ownerid\_primary PRIMARY KEY(OwnerID);

alter table

Owner ADD CONSTRAINT Owner\_check\_phoheNumber CHECK (DATALENGTH(CAST(PhoneNumber AS varchar(10))) = 10))

alter table

Owner ADD CONSTRAINT Owner\_check\_Email CHECK (Email like '[^=-)(\*&^%$#@!]%[^=-)(\*&^%$#@!]@[^=-)(\*&^%$#@!]%[^=-)(\*&^%$#@!].[^=-)(\*&^%$#@!]%[^=-)(\*&^%$#@!]')

alter table

Owner ADD CONSTRAINT Owner\_check\_INN CHECK (DATALENGTH(CAST(INN AS varchar(12))) = 12)

* 1. Region

Данная справочная таблица создана с целью хранения информации о регионах.

Листинг:

CREATE TABLE Region(

RegionID INT identity(1,1) NOT NULL,

NameRegion NVARCHAR(50) NOT NULL

);

ALTER TABLE

Region ADD CONSTRAINT region\_regionid\_primary PRIMARY KEY(RegionID);

* 1. Farm

Данная таблица создана с целью хранения информации о хозяйстве.

Листинг:

CREATE TABLE Farm(

FarmID int IDENTITY(1,1) NOT NULL,

codeUSRRE NVARCHAR(50) NOT NULL,

NameFarm NVARCHAR(50) NULL,

City NVARCHAR(50) NULL,

Street NVARCHAR(100) NULL,

NumberHouse NVARCHAR(50) NULL,

PhoneNumber BIGINT NULL,

Email NVARCHAR(100) NULL,

OwnerID INT NOT NULL,

RegionID INT NOT NULL

);

ALTER TABLE

Farm ADD CONSTRAINT farm\_FarmID\_primary PRIMARY KEY(FarmID);

ALTER TABLE

Farm ADD CONSTRAINT farm\_codeUSRRE\_UNIQUE UNIQUE(codeUSRRE);

ALTER TABLE

Farm ADD CONSTRAINT farm\_regionid\_foreign FOREIGN KEY(RegionID) REFERENCES Region(RegionID) on delete cascade on update cascade;

ALTER TABLE

Farm ADD CONSTRAINT farm\_ownerid\_foreign FOREIGN KEY(OwnerID) REFERENCES Owner(OwnerID) on delete cascade on update cascade;

alter table

Farm ADD CONSTRAINT farm\_check\_phoheNumber CHECK (DATALENGTH(CAST(PhoneNumber AS varchar(10))) = 10))

alter table

Farm ADD CONSTRAINT farm\_check\_Email CHECK (Email like '[^=-)(\*&^%$#@!]%[^=-)(\*&^%$#@!]@[^=-)(\*&^%$#@!]%[^=-)(\*&^%$#@!]%.%[^=-)(\*&^%$#@!]%[^=-)(\*&^%$#@!]%')

alter table

Farm ADD CONSTRAINT farm\_check\_codeUSRRE CHECK (codeUSRRE like '[0-9][0-9]-[0-9][0-9]-[0-9][0-9]/[0-9][0-9][0-9]/[0-9][0-9][0-9][0-9]-[0-9][0-9][0-9]')

* 1. FarmSpecialization

Данная таблица создана с целью хранения информации о специализация фермы связи таблиц Farm и Specialization.

Листинг:

CREATE TABLE FarmSpecialization(

FarmSpecializationID INT identity(1,1) NOT NULL,

FarmID INT NOT NULL,

SpecializationID INT NOT NULL

);

ALTER TABLE

FarmSpecialization ADD CONSTRAINT farmspecialization\_farmspecializationid\_primary PRIMARY KEY(FarmSpecializationID);

ALTER TABLE

FarmSpecialization ADD CONSTRAINT farmspecialization\_farmid\_foreign FOREIGN KEY(FarmID) REFERENCES Farm(FarmID) on delete cascade on update cascade;

ALTER TABLE

FarmSpecialization ADD CONSTRAINT farmspecialization\_specializationid\_foreign FOREIGN KEY(SpecializationID) REFERENCES Specialization(SpecializationID) on delete cascade on update cascade;

* 1. Specialization

Данная таблица создана с целью хранения информации о специализация фермы связи таблиц Farm и Specialization.

Листинг:

CREATE TABLE Specialization(

SpecializationID INT identity(1,1) NOT NULL,

NameSpecialization NVARCHAR(50) NOT NULL

);

ALTER TABLE

Specialization ADD CONSTRAINT specialization\_specializationid\_primary PRIMARY KEY(SpecializationID);

* 1. FarmProduct

Данная таблица создана с целью хранения информации о продуктах фермы и связи таблиц Farm и Product.

Листинг:

CREATE TABLE FarmProduct(

FarmProductID INT identity(1,1) NOT NULL,

FarmID INT NOT NULL,

ProductID INT NOT NULL,

Quantity FLOAT NOT NULL,

UnitPrice FLOAT NOT NULL

);

ALTER TABLE

FarmProduct ADD CONSTRAINT farmproduct\_farmproductid\_primary PRIMARY KEY(FarmProductID);

ALTER TABLE

FarmProduct ADD CONSTRAINT farmproduct\_productid\_foreign FOREIGN KEY(ProductID) REFERENCES Product(ProductID);

ALTER TABLE

FarmProduct ADD CONSTRAINT farmproduct\_farmid\_foreign FOREIGN KEY(FarmID) REFERENCES Farm(FarmID) on delete cascade on update cascade;

alter table

FarmProduct ADD CONSTRAINT FarmProduct\_check\_Quantity CHECK (Quantity >= 0)

alter table

FarmProduct ADD CONSTRAINT FarmProduct\_check\_UnitPrice CHECK (UnitPrice >= 0)

* 1. Product

Данная таблица создана с целью хранения информации о продуктах

Листинг:

CREATE TABLE Product(

ProductID INT identity(1,1) NOT NULL,

NameProduct NVARCHAR(50) NOT NULL,

Measure NVARCHAR(50) NOT NULL,

TypeProductID INT NOT NULL

);

ALTER TABLE

Product ADD CONSTRAINT product\_productid\_primary PRIMARY KEY(ProductID);

ALTER TABLE

Product ADD CONSTRAINT product\_typeproductid\_foreign FOREIGN KEY(TypeProductID) REFERENCES TypeProduct(TypeProductID);

* 1. TypeProduct

Данная таблица создана с целью хранения информации о видах продуктах.

Листинг:

CREATE TABLE TypeProduct(

TypeProductID INT identity(1,1) NOT NULL,

NameTypeProduct NVARCHAR(50) NOT NULL

);

ALTER TABLE

TypeProduct ADD CONSTRAINT typeproduct\_typeproductid\_primary PRIMARY KEY(TypeProductID);

* 1. TypeActivity

Данная таблица создана с целью хранения информации о видах деятельности

Листинг:

CREATE TABLE TypeActivity(

TypeActivityID INT identity(1,1) NOT NULL,

NameTypeActivity NVARCHAR(50) NOT NULL

);

ALTER TABLE

TypeActivity ADD CONSTRAINT typeactivity\_typeactivityid\_primary PRIMARY KEY(TypeActivityID);

* 1. FarmTypeActivity

Данная таблица создана с целью хранения информации о видах деятельности фермы и связи таблиц Farm и TypeActivity.

Листинг:

CREATE TABLE FarmTypeActivity(

FarmTypeActivityID INT identity(1,1) NOT NULL,

FarmID INT NOT NULL,

TypeActivityID INT NOT NULL

);

ALTER TABLE

FarmTypeActivity ADD CONSTRAINT farmtypeactivity\_farmtypeactivityid\_primary PRIMARY KEY(FarmTypeActivityID);

ALTER TABLE

FarmTypeActivity ADD CONSTRAINT farmtypeactivity\_farmid\_foreign FOREIGN KEY(FarmID) REFERENCES Farm(FarmID) on delete cascade on update cascade;

ALTER TABLE

FarmTypeActivity ADD CONSTRAINT farmtypeactivity\_typeactivityid\_foreign FOREIGN KEY(TypeActivityID) REFERENCES TypeActivity(TypeActivityID);

## Написание запросов, функций и хранимых процедур

Хранимые процедуры

1. OwnerInsertUpdateDelete

Описание: процедура создана для вставки, удаления, обновления строк в таблицу Owner и включает следующие переменные: @Delete (0 – не удаление, 1 - удаление), @OwnerID, @FirstName, @MiddleName, @LastName, @PhoneNumber, @Email, @INN соответствующие поля в таблице Owner.

Листинг процедуры:

Create PROCEDURE OwnerInsertUpdateDelete (

@Delete bit = 0,

@OwnerID int = -1,

@FirstName nvarchar(50),

@MiddleName nvarchar(50),

@LastName nvarchar(50) = NULL,

@PhoneNumber BIGINT = NULL,

@Email nvarchar(100) = NULL,

@INN BIGINT

)

As

Begin

if(@Delete = 1 and @OwnerID in (select OwnerID from Owner))

delete Owner where OwnerID = @OwnerID

else

if(@Delete = 0 and @OwnerID not in (select OwnerID from Owner))

insert Owner (FirstName, MiddleName, LastName, PhoneNumber, Email, INN)

values (@FirstName, @MiddleName, @LastName, @PhoneNumber, @Email, @INN)

else

if(@Delete = 0 and @OwnerID in (select OwnerID from Owner))

UPDATE [dbo].[Owner]

SET [FirstName] = @FirstName

,[MiddleName] = @MiddleName

,[LastName] = @LastName

,[PhoneNumber] = @PhoneNumber

,[Email] = @Email

,[INN] = @INN

where OwnerID = @OwnerID

end

Пример использования:

* Удаление: execute OwnerInsertUpdateDelete 1,2,'','','','','',''

При удалении важно указать первым (@Delete) 1 и вторым (@OwnerID) существующий OwnerID, остальные значения не важны, их можно заменить кавычками.

* Добавление: execute OwnerInsertUpdateDelete 0,-1,'Иванов','Иван','Иванович', 89314010975, null,123456789123

При добавлении важно указать первым (@Delete) 0 и вторым (@OwnerID) -1, а остальное заполнить нужными значениями.

* Обновление: execute OwnerInsertUpdateDelete 0,1,'Иванов','Иван','Иванович', 89314010975, null,123456789123

При обновлении важно указать первым (@Delete) 0 и вторым (@OwnerID) существующий OwnerID, а остальное заполнить нужными значениями.

1. FarmInsertUpdateDelete

Описание: процедура создана для вставки, удаления, обновления строк в таблицу Farm и включает следующие переменные: @Delete (0 – не удаление, 1 - удаление), @FarmID, @codeUSRRE, @NameFarm, @City, @Street, @NumberHouse, @PhoneNumber, @Email, @OwnerID, @RegionID соответствующие поля в таблице Farm.

Листинг процедуры:

Create PROCEDURE FarmInsertUpdateDelete (

@Delete bit = 0,

@FarmID int = -1,

@codeUSRRE nvarchar(50),

@NameFarm nvarchar(50),

@City nvarchar(50) = null,

@Street NVARCHAR(100) = null,

@NumberHouse nvarchar(50) = null,

@PhoneNumber BIGINT = null,

@Email nvarchar(100) = null,

@OwnerID int,

@RegionID int

)

As

Begin

if(@Delete = 1 and @FarmID in (select FarmID from Farm))

delete Farm where FarmID = @FarmID

else

if(@Delete = 0 and @FarmID not in (select FarmID from Farm))

insert Farm (codeUSRRE, NameFarm, City, Street, NumberHouse, PhoneNumber, Email, OwnerID, RegionID)

values (@codeUSRRE, @NameFarm, @City, @Street, @NumberHouse, @PhoneNumber, @Email, @OwnerID, @RegionID)

else

if(@Delete = 0 and @FarmID in (select FarmID from Farm))

UPDATE [dbo].[Farm]

SET [codeUSRRE] = @codeUSRRE

,[NameFarm] = @NameFarm

,[City] = @City

,[Street] = @Street

,[NumberHouse] = @NumberHouse

,[PhoneNumber] = @PhoneNumber

,[Email] = @Email

,[OwnerID] = @OwnerID

,[RegionID] = @RegionID

where FarmID = @FarmID

end

Пример использования:

* Удаление: execute FarmInsertUpdateDelete 1,2, '','','','','','','',-1,-1

При удалении важно указать первым (@Delete) 1 и вторым (@FarmID) существующий FarmID, остальные значения не важны, их можно заменить кавычками.

* Добавление: execute FarmInsertUpdateDelete 0,-1,'65-56-65/101/2022-423','aaaaa’,null,null,null, 9324010987,null,1,4

При добавлении важно указать первым (@Delete) 0 и вторым (@FarmID) -1, а остальное заполнить нужными значениями.

* Обновление: execute FarmInsertUpdateDelete 0,1,'55-66-65/101/2022-423','aaaaa',null,null,null, 9324010987,null,1,4

При добавлении важно указать первым (@Delete) 0 и вторым (@FarmID) существующий FarmID, а остальное заполнить нужными значениями.

1. ProductInsertUpdateDelete

Описание: процедура создана для вставки, удаления, обновления строк в таблицу Product и включает следующие переменные: @Delete (0 – не удаление, 1 - удаление), @ProductID, @NameProduct,@Measure,@TypeProductID соответствующие поля в таблице Product.

Листинг процедуры:

Create PROCEDURE ProductInsertUpdateDelete (

@Delete bit = 0,

@ProductID int = -1,

@NameProduct nvarchar(50),

@Measure nvarchar(50),

@TypeProductID int

)

As

Begin

if(@Delete = 1 and @ProductID in (select ProductID from Product))

delete Product where ProductID = @ProductID

else

if(@Delete = 0 and @ProductID not in (select ProductID from Product))

insert Product (NameProduct,Measure,TypeProductID)

values (@NameProduct,@Measure,@TypeProductID)

else

if(@Delete = 0 and @ProductID in (select ProductID from Product))

UPDATE [dbo].[Product]

SET [NameProduct] = @NameProduct,

[Measure] = @Measure,

[TypeProductID] = @TypeProductID

where ProductID = @ProductID

end

Пример использования:

* Удаление: execute ProductInsertUpdateDelete 1,1, '','',-1

При удалении важно указать первым (@Delete) 1 и вторым (@ProductID) существующий ProductID, остальные значения не важны, их можно заменить случайными значениями.

* Добавление: execute ProductInsertUpdateDelete 0,-1, 'Пшеница','киллограмм',2

При добавлении важно указать первым (@Delete) 0 и вторым (@ProductID) -1, а остальное заполнить нужными значениями.

* Обновление: execute ProductInsertUpdateDelete 0,1, 'Пшеница','киллограмм',2

При обновлении важно указать первым (@Delete) 0 и вторым (@ProductID) существующий ProductID, а остальное заполнить нужными значениями.

1. RegionInsertUpdateDelete

Описание: процедура создана для вставки, удаления, обновления строк в таблицу Region и включает следующие переменные: @Delete (0 – не удаление, 1 - удаление), @RegionID, @RegionID соответствующие поля в таблице Region.

Листинг процедуры:

Create PROCEDURE RegionInsertUpdateDelete (

@Delete bit = 0,

@RegionID int = -1,

@NameRegion nvarchar(50)

)

As

Begin

if(@Delete = 1 and @RegionID in (select RegionID from Region))

delete Region where RegionID = @RegionID

else

if(@Delete = 0 and @RegionID not in (select RegionID from Region))

insert Region (NameRegion)

values (@NameRegion)

else

if(@Delete = 0 and @RegionID in (select RegionID from Region))

UPDATE [dbo].[Region]

SET [NameRegion] = @NameRegion

where RegionID = @RegionID

end

Пример использования:

* Удаление: execute RegionInsertUpdateDelete 1,2, ''

При удалении важно указать первым (@Delete) 1 и вторым (@RegionID) существующий RegionID, остальные значения не важны, их можно заменить кавычками.

* Добавление: execute RegionInsertUpdateDelete 0,-1,’Дагестан’

При добавлении важно указать первым (@Delete) 0 и вторым (@RegionID) -1, а остальное заполнить нужными значениями.

* Обновление: execute RegionInsertUpdateDelete 0,1, ’Дагестан’

При обновлении важно указать первым (@Delete) 0 и вторым (@RegionID) существующий RegionID, а остальное заполнить нужными значениями.

1. SpecializationInsertUpdateDelete

Описание: процедура создана для вставки, удаления, обновления строк в таблицу Specialization и включает следующие переменные: @Delete (0 – не удаление, 1 - удаление), @codeUSRRE, @NameSpecialization соответствующие поля в таблице Specialization.

Листинг процедуры:

Create PROCEDURE SpecializationInsertUpdateDelete (

@Delete bit = 0,

@SpecializationID int = -1,

@NameSpecialization nvarchar(50)

)

As

Begin

if(@Delete = 1 and @SpecializationID in (select SpecializationID from Specialization))

delete Specialization where SpecializationID = @SpecializationID

else

if(@Delete = 0 and @SpecializationID not in (select SpecializationID from Specialization))

insert Specialization (NameSpecialization)

values (@NameSpecialization)

else

if(@Delete = 0 and @SpecializationID in (select SpecializationID from Specialization))

UPDATE [dbo].[Specialization]

SET [NameSpecialization] = @NameSpecialization

where SpecializationID = @SpecializationID

end

Пример использования:

* Удаление: execute SpecializationInsertUpdateDelete 1,2, ''

При удалении важно указать первым (@Delete) 1 и вторым (@SpecializationID) существующий SpecializationID, остальные значения не важны, их можно заменить кавычками.

* Добавление: execute SpecializationInsertUpdateDelete 0,-1, 'Мясное'

При добавлении важно указать первым (@Delete) 0 и вторым (@SpecializationID) -1, а остальное заполнить нужными значениями.

* Обновление: execute SpecializationInsertUpdateDelete 0,1, 'Молочномясное'

При обновлении важно указать первым (@Delete) 0 и вторым (@SpecializationID) существующий SpecializationID, а остальное заполнить нужными значениями.

1. TypeActivityInsertUpdateDelete

Описание: процедура создана для вставки, удаления, обновления строк в таблицу TypeActivity и включает следующие переменные: @Delete (0 – не удаление, 1 - удаление), @TypeActivityID, @NameTypeActivity соответствующие поля в таблице TypeActivity.

Листинг процедуры:

Create PROCEDURE TypeActivityInsertUpdateDelete (

@Delete bit = 0,

@TypeActivityID int = -1,

@NameTypeActivity nvarchar(50)

)

As

Begin

if(@Delete = 1 and @TypeActivityID in (select TypeActivityID from TypeActivity))

delete TypeActivity where TypeActivityID = @TypeActivityID

else

if(@Delete = 0 and @TypeActivityID not in (select TypeActivityID from TypeActivity))

insert TypeActivity (NameTypeActivity)

values (@NameTypeActivity)

else

if(@Delete = 0 and @TypeActivityID in (select TypeActivityID from TypeActivity))

UPDATE [dbo].[TypeActivity]

SET [NameTypeActivity] = @NameTypeActivity

where TypeActivityID = @TypeActivityID

end

Пример использования:

* Удаление: execute TypeActivityInsertUpdateDelete 1,2, ''

При удалении важно указать первым (@Delete) 1 и вторым (@TypeActivityID) существующий TypeActivityID, остальные значения не важны, их можно заменить кавычками.

* Добавление: execute TypeActivityInsertUpdateDelete 0,-1, 'Овощеводство'

При добавлении важно указать первым (@Delete) 0 и вторым (@TypeActivityID) -1, а остальное заполнить нужными значениями.

* Обновление: execute TypeActivityInsertUpdateDelete 0,1, 'Животноводство'

При обновлении важно указать первым (@Delete) 0 и вторым (@TypeActivityID) существующий TypeActivityID, а остальное заполнить нужными значениями.

1. TypeProductInsertUpdateDelete

Описание: процедура создана для вставки, удаления, обновления строк в таблицу TypeProduct и включает следующие переменные: @Delete (0 – не удаление, 1 - удаление), @TypeProductID, @NameTypeProduct соответствующие поля в таблице TypeProduct.

Листинг процедуры:

Create PROCEDURE TypeProductInsertUpdateDelete (

@Delete bit = 0,

@TypeProductID int = -1,

@NameTypeProduct nvarchar(50)

)

As

Begin

if(@Delete = 1 and @TypeProductID in (select TypeProductID from TypeProduct))

delete TypeProduct where TypeProductID = @TypeProductID

else

if(@Delete = 0 and @TypeProductID not in (select TypeProductID from TypeProduct))

insert TypeProduct (NameTypeProduct)

values (@NameTypeProduct)

else

if(@Delete = 0 and @TypeProductID in (select TypeProductID from TypeProduct))

UPDATE [dbo].[TypeProduct]

SET [NameTypeProduct] = @NameTypeProduct

where TypeProductID = @TypeProductID

end

Пример использования:

* Удаление: execute TypeProductInsertUpdateDelete 1,2, ''

При удалении важно указать первым (@Delete) 1 и вторым (@TypeProductID) существующий TypeProductID, остальные значения не важны, их можно заменить кавычками.

* Добавление: execute TypeProductInsertUpdateDelete 0,-1, 'Овощеводство'

При добавлении важно указать первым (@Delete) 0 и вторым (@TypeProductID) -1, а остальное заполнить нужными значениями.

* Обновление: execute TypeProductInsertUpdateDelete 0,1, 'Животноводство'

При обновлении важно указать первым (@Delete) 0 и вторым (@TypeProductID) существующий TypeProductID, а остальное заполнить нужными значениями.

1. FarmProductInsertUpdateDelete

Описание: процедура создана для вставки, удаления, обновления строк в таблицу FarmProduct и включает следующие переменные: @Delete (0 – не удаление, 1 - удаление), @FarmID,@ProductID,@Quantity,@UnitPrice соответствующие поля в таблице FarmProduct.

Листинг процедуры:

Create PROCEDURE FarmProductInsertUpdateDelete (

@Delete bit = 0,

@FarmProductID int = -1,

@FarmID int,

@ProductID int,

@Quantity float,

@UnitPrice float

)

As

Begin

if(@Delete = 1 and @FarmProductID in (select FarmProductID from FarmProduct))

delete FarmProduct where FarmProductID = @FarmProductID

else

if(@Delete = 0 and @FarmProductID not in (select FarmProductID from FarmProduct))

insert FarmProduct (FarmID,ProductID,Quantity,UnitPrice)

values (@FarmID,@ProductID,@Quantity,@UnitPrice)

else

if(@Delete = 0 and @FarmProductID in (select FarmProductID from FarmProduct))

UPDATE [dbo].[FarmProduct]

SET [FarmID] = FarmID,

[ProductID] = ProductID,

[Quantity] = Quantity,

[UnitPrice] = UnitPrice

where FarmProductID = @FarmProductID

end

Пример использования:

* Удаление: execute FarmProductInsertUpdateDelete 1,2,0,0,0,0

При удалении важно указать первым (@Delete) 1 и вторым (@FarmProductID) существующий FarmProductID, остальные значения не важны, их можно заменить кавычками.

* Добавление: execute FarmProductInsertUpdateDelete 0,-1, 1,1,10,110

При добавлении важно указать первым (@Delete) 0 и вторым (@FarmProductID) -1, а остальное заполнить нужными значениями.

* Обновление: execute FarmProductInsertUpdateDelete 0,1, 1,1,10,1100

При обновлении важно указать первым (@Delete) 0 и вторым (@FarmProductID) существующий FarmProductID, а остальное заполнить нужными значениями.

1. FarmSpecializationInsertUpdateDelete

Описание: процедура создана для вставки, удаления, обновления строк в таблицу FarmSpecialization и включает следующие переменные: @Delete (0 – не удаление, 1 - удаление), @FarmSpecializationID,@FarmID,@SpecializationID соответствующие поля в таблице FarmSpecialization.

Листинг процедуры:

Create PROCEDURE FarmSpecializationInsertUpdateDelete (

@Delete bit = 0,

@FarmSpecializationID int = -1,

@FarmID int,

@SpecializationID int

)

As

Begin

if(@Delete = 1 and @FarmSpecializationID in (select FarmSpecializationID from FarmSpecialization))

delete FarmSpecialization where FarmSpecializationID = @FarmSpecializationID

else

if(@Delete = 0 and @FarmSpecializationID not in (select FarmSpecializationID from FarmSpecialization))

insert FarmSpecialization (FarmID,SpecializationID)

values (@FarmID,@SpecializationID)

else

if(@Delete = 0 and @FarmSpecializationID in (select FarmSpecializationID from FarmSpecialization))

UPDATE [dbo].[FarmSpecialization]

SET [FarmID] = FarmID,

[SpecializationID] = SpecializationID

where FarmSpecializationID = @FarmSpecializationID

end

Пример использования:

* Удаление: execute FarmSpecializationInsertUpdateDelete 1,2,0,0

При удалении важно указать первым (@Delete) 1 и вторым (@FarmSpecializationID) существующий FarmSpecializationID, остальные значения не важны, их можно заменить кавычками.

* Добавление: execute FarmSpecializationInsertUpdateDelete 0,-1,1,1

При добавлении важно указать первым (@Delete) 0 и вторым (@FarmSpecializationID) -1, а остальное заполнить нужными значениями.

* Обновление: execute FarmSpecializationInsertUpdateDelete 0,1,1,2

При обновлении важно указать первым (@Delete) 0 и вторым (@FarmSpecializationID) существующий FarmSpecializationID, а остальное заполнить нужными значениями.

1. FarmTypeActivityInsertUpdateDelete

Описание: процедура создана для вставки, удаления, обновления строк в таблицу FarmTypeActivity и включает следующие переменные: @Delete (0 – не удаление, 1 - удаление), @FarmTypeActivityID,@FarmID,@SpecializationID соответствующие поля в таблице FarmTypeActivity.

Листинг процедуры:

Create PROCEDURE FarmTypeActivityInsertUpdateDelete (

@Delete bit = 0,

@FarmTypeActivityID int = -1,

@FarmID int,

@TypeActivityID int

)

As

Begin

if(@Delete = 1 and @FarmTypeActivityID in (select FarmTypeActivityID from FarmTypeActivity))

delete FarmTypeActivity where FarmTypeActivityID = @FarmTypeActivityID

else

if(@Delete = 0 and @FarmTypeActivityID not in (select FarmTypeActivityID from FarmTypeActivity))

insert FarmTypeActivity (FarmID,TypeActivityID)

values (@FarmID,@TypeActivityID)

else

if(@Delete = 0 and @FarmTypeActivityID in (select FarmTypeActivityID from FarmTypeActivity))

UPDATE [dbo].[FarmTypeActivity]

SET [FarmID] = FarmID,

[TypeActivityID] = TypeActivityID

where FarmTypeActivityID = @FarmTypeActivityID

end

Пример использования:

* Удаление: execute FarmTypeActivityInsertUpdateDelete 1,2,0,0

При удалении важно указать первым (@Delete) 1 и вторым (@FarmTypeActivityID) существующий FarmTypeActivityID, остальные значения не важны, их можно заменить кавычками.

* Добавление: execute FarmTypeActivityInsertUpdateDelete 0,-1, 1,1 При добавлении важно указать первым (@Delete) 0 и вторым (@FarmTypeActivityID) -1, а остальное заполнить нужными значениями.
* Обновление: execute FarmTypeActivityInsertUpdateDelete 0,1,1,2

При обновлении важно указать первым (@Delete) 0 и вторым (@FarmTypeActivityID) существующий FarmTypeActivityID, а остальное заполнить нужными значениями.

1. FarmInfo

Процедура создана для получения информации о конкретном хозяйстве и его продукции, ранжирование продукции. Включает переменную @FarmID соответствующее поле в таблице Farm.

Листинг:

Create PROCEDURE FarmInfo (

@FarmID int

)

As

Begin

select

NameFarm,

SumQuantity as TotalNumberProduct,

SumQuantityUnitPrice as TotalPriceProduct,

NameProduct,

DENSE\_RANK() over (order by SumQuantityProduct desc) as TotalRangNumberSalses,

SumQuantityProduct,

DENSE\_RANK() over (order by SumQuantityUnitPriceProduct desc) as TotalRangPriceSalses,

SumQuantityUnitPriceProduct,

AVGUnitPrice

from Farm f

Left join FarmProduct fp on f.FarmID = fp.FarmID

Left join (select

FarmID,

sum(Quantity) as SumQuantity,

sum(Quantity \* UnitPrice) as SumQuantityUnitPrice

from FarmProduct fp

group by FarmID)

fap on fap.FarmID = f.FarmID

Left join (select

ProductID,

sum(Quantity) as SumQuantityProduct,

sum(Quantity \* UnitPrice) as SumQuantityUnitPriceProduct,

AVG(UnitPrice) AS AVGUnitPrice

from FarmProduct fp

where FarmID = @FarmID

group by ProductID) pap on pap.ProductID = fp.ProductID

Left join Product p on p.ProductID = fp.ProductID

where f.FarmID = @FarmID

group by f.NameFarm,SumQuantity,SumQuantityUnitPrice,NameProduct,SumQuantityProduct,SumQuantityUnitPriceProduct,AVGUnitPrice

end

Пример использования: вам нужно указать существующий FarmID, execute FarmInfo 1.

Выходные данные: NameFarm, TotalNumberProduct, TotalPriceProduct, NameProduct, TotalRangNumberSalses, SumQuantityProduct, TotalRangPriceSalses, SumQuantityUnitPriceProduct, AVGUnitPrice.

1. ProductInfo

Процедура создана для получения информации о продукции, ранжирование продукции.

Листинг

create procedure ProductInfo

as

begin

select

NameProduct,

DENSE\_RANK() over (order by SumQuantity desc) as RangNumberProduct,

SumQuantity,

DENSE\_RANK() over (order by SumQuantityUnitPrice desc) as RangPriceProduct,

SumQuantityUnitPrice ,

AVG(UnitPrice) as AVGUnitPrice,

Min(UnitPrice) as MinUnitPrice,

Max(UnitPrice) as MaxUnitPrice

from Product p

Left join FarmProduct fp on p.ProductID = fp.ProductID

Left join (select

ProductID,

sum(Quantity) as SumQuantity,

sum(Quantity \* UnitPrice) as SumQuantityUnitPrice

from FarmProduct fp

group by ProductID)

pp on pp.ProductID = p.ProductID

group by NameProduct,SumQuantity,SumQuantityUnitPrice

end

Пример использования: execute ProductInfo.

Выходные данные: NameProduct, RangNumberProduct, SumQuantity, RangPriceProduct, SumQuantityUnitPrice,AVGUnitPrice,MinUnitPrice,MaxUnitPrice

1. TypeProductInfo

Процедура создана для получение информации о видах продукции, ранжирование продукции.

Листинг:

create procedure TypeProductInfo

as

begin

select

NameTypeProduct,

Count(p.ProductID) as NumberProduct,

DENSE\_RANK() over (order by SumQuantity desc) as RangSumQuantity,

SumQuantity as SumPriceProduct,

DENSE\_RANK() over (order by SumQuantityUnitPrice desc) as RangSumPrice,

SumQuantityUnitPrice as SumPrice,

AVG(UnitPrice) as AVGUnitPrice,

Min(UnitPrice) as MinUnitPrice,

Max(UnitPrice) as MaxUnitPrice

from TypeProduct tp

Left join Product p on tp.TypeProductID = p.TypeProductID

Left join FarmProduct fp on fp.ProductID = p.ProductID

Left join (select

TypeProductID,

sum(Quantity) as SumQuantity,

sum(Quantity \* UnitPrice) as SumQuantityUnitPrice

from FarmProduct fp Left join Product p on p.ProductID = fp.ProductID

group by TypeProductID) pp on pp.TypeProductID = p.TypeProductID

group by NameTypeProduct,SumQuantity,SumQuantityUnitPrice

end

Пример использования: execute TypeProductInfo.

Выходные данные: NameTypeProduct, NumberProduct, RangSumQuantity, SumPriceProduct, RangSumPrice, SumPrice, AVGUnitPrice, MinUnitPrice, MaxUnitPrice

Тригеры:

1. checkInsertFarm

Тригер создан как ограничение таблицы Farm на вставку и обновление полей Email и PhoneNumber с условием: хотя бы одно из двух полей должно быть заполнено.

Листинг:

create trigger checkInsertFarm on Farm

for insert, update

as

declare @PhoneNumber BIGINT, @Email nvarchar(100)

Select

@PhoneNumber=PhoneNumber,

@Email=Email

from inserted

if (DATALENGTH(@Email) = 0 and DATALENGTH(@PhoneNumber) = 0)

Begin

print 'Необходимо указать либо номер телефона, либо email!'

ROLLBACK TRANSACTION

End

1. checkInsertOwner

Тригер создан как ограничение таблицы Owner на вставку и обновление полей Email и PhoneNumber с условием: хотя бы одно из двух полей должно быть заполнено.

Листинг:

create trigger checkInsertOwner on Owner

for insert, update

as

declare @PhoneNumber BIGINT, @Email nvarchar(100)

Select

@PhoneNumber=PhoneNumber,

@Email=Email

from inserted

if (DATALENGTH(@Email) = 0 and DATALENGTH(@PhoneNumber) = 0)

Begin

print 'Необходимо указать либо номер телефона, либо email!'

ROLLBACK TRANSACTION

End

# Заключение

В данной курсовой работе был проведен детальный анализ предметной области ассоциации фермерских хозяйств. Раскрыто само понятие БД и способы его построения в соответствии с основными этапами проектирования, осуществляемыми в рамках современных информационных технологий.

В конечном итоге, на основе MSSQL была разработана база данных для ассоциации фермерских хозяйств.

Разрабатываемая БД поможет отслеживать информацию о каждом участнике ассоциации, его хозяйстве и производимых хозяйством продуктах. Полученная БД позволяет сформировать отчет о всех товарах, конкретных хозяйствах.

Использование БД в реальной ассоциации позволит автоматизировать деятельность сотрудников ассоциации, увеличив производительность труда персонала. Создаст благоприятные условия труда для сотрудников, обеспечит доступность и позволит всегда отражать актуальную информацию для сторонних лиц и участников ассоциации и позволит всегда отражать актуальную информацию.

Поставленная цель курсовой работы достигнута, задачи решены в полном объеме.

# Список использованных источников

1. <https://www.akkor.ru/>
2. <https://docs.microsoft.com/ru-ru/sql/sql-server/?view=sql-server-ver15>
3. <https://metanit.com/sql/sqlserver/>
4. Занимательное программирование. Базы данных. Манга / Мана Такахаси (автор), Сёко Адзума (худож.); пер. Сенниковой Т. И. — М.: ДМК Пресс, 2014. — 238 с.: ил. — (Серия «Образовательная манга»). — Доп. тит.л. яп. —ISBN 978-5-97060-044-3.