Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Рязанский государственный радиотехнический университет имени В.Ф.Уткина»  
(ФГБОУ ВО «РГРТУ», РГРТУ)

Кафедра вычислительной и прикладной математики

**Пояснительная записка к курсовой работе**

**по дисциплине «Базы данных»**

**на тему «Пункт выдачи заказов»**

Выполнила:

Ст-т группы 643

Паршина Анна Романовна

Проверил:

доц. кафедры ВПМ

Благодаров Андрей Витальевич

Рязань 2019

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc10758045)

[1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 4](#_Toc10758046)

[1.1. Анализ предметной области, выявление необходимой пользователю функциональности 4](#_Toc10758047)

[2. ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ 5](#_Toc10758048)

[2.1. Разработка концептуальной модели данных 5](#_Toc10758049)

[2.1.1. Выявление сущностей, их атрибутов и ключей 5](#_Toc10758050)

[2.1.2. Выявление связей 5](#_Toc10758051)

[2.1.3. Построение CDM 7](#_Toc10758052)

[2.2. Разработка логической модели данных 8](#_Toc10758053)

[2.2.1. Заполнение сущностей атрибутами 8](#_Toc10758054)

[2.2.2. Проверка сущностей на соответствие нормальным формам 10](#_Toc10758055)

[2.2.3. Построение LDM 16](#_Toc10758056)

[2.3. Разработка физической модели данных 18](#_Toc10758057)

[2.3.1. Задание типов данных для полей таблиц 18](#_Toc10758058)

[2.3.2. Задание частных ограничений целостности данных 19](#_Toc10758059)

[2.3.3. Построение PDM 19](#_Toc10758060)

[3. РАЗРАБОТКА ТРИГГЕРОВ 21](#_Toc10758061)

[4. РАЗРАБОТКА ХРАНИМЫХ ПРОЦЕДУР 22](#_Toc10758063)

[5. РАЗРАБОТКА ЗАПРОСОВ НА ВЫБОРКУ 25](#_Toc10758064)

[ЗАКЛЮЧЕНИЕ 27](#_Toc10758065)

[СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 27](#_Toc10758066)

[ПРИЛОЖЕНИЕ 1 28](#_Toc10758067)

[ПРИЛОЖЕНИЕ 2 38](#_Toc10758068)

# ВВЕДЕНИЕ

Пункты выдачи заказов это удобный способ заказа предметов в интернет-магазинах. Поэтому данная тема становится все более актуальной.

Самым крупным представителем такой системы является «Wildberries». Ты совершаешь заказ через сайт магазина или приложение на телефоне, заказа поступает на формирование и через пару дней он поступает на пункт, которые сами же и выбираете. Таких пунктов по России уже очень много, только на территории Рязани и Рязанской области насчитывается более 30 пунктов. Оповещение о том, что заказ поступил на пункт приходит как на телефон, так и push-уведомлением в приложении. Пункты работают с 10:00 – 21:00. В течение 5 дней вы можете забрать заказ.

Ещё одним популярным представителем данной системы является компания «CDEK», которая является чисто курьерской компанией, т.е. у нее нет своего собственного интернет-магазина, она лишь занимается доставкой посылок с разных онлайн-магазинов. Большим преимуществом данной компании является то, что она сотрудничает с зарубежными компаниями. Вы можете заказать посылку зарубежного партнера и получить посылку очень быстро и также в любое рабочее время.

Тема очень популярная сейчас и интересная, поэтому она была выбрана в качестве курсовой работы.

# АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

## Анализ предметной области, выявление необходимой пользователю функциональности

В данной курсовой работе рассматривается тема «Пункт выдачи заказов». Целевой пользователь данной системы будет работник пункта, которые будет фиксировать данные о заказах.

Для пользователей должна быть предусмотрена система паролей и логинов, которые будут индивидуальны и сохраняться за пользователем навсегда.

У пользователя должна быть возможность фиксировать состояния заказов на каждом их этапе. Заказ пришел на пункт выдачи, пользователь зафиксировал данное состояние, и теперь за данный заказ отвечает он.

В базе данных должна храниться информация о:

* пользователях, их личные данные, их должность;
* пунктах, их адрес и время работы;
* клиентах, их личные данные;
* товарах, их названия, категория и внешние признаки;
* заказах и история состояний заказов.

# ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ

## Разработка концептуальной модели данных

### Выявление сущностей, их атрибутов и ключей

В ходе разработки информационной системы были выявлены следующие сущности:

1. Работник

Первичный ключ: Табельный номер

1. Покупатель

Первичный ключ: Табельный номер

1. Пункт

Первичный ключ: Краткое название

1. Товар

Первичный ключ: Номер товара

1. Категория товара

Первичный ключ: Краткое название

1. Заказ

Первичный ключ: Номер заказа

1. Статус

Первичный ключ: Краткое название

1. История

Первичный ключ: Номер

### Выявление связей

В ходе разработки информационной системы были выявлены следующие связи:



Рисунок 2.1 - Связь «Работает на»

Работник работает только на одном пункте, но на пункте может работать не один работник. Следовательно, связь Работник-Пункт будет 1…1 (Один к одному), Пункт-Работник 0…N (Ноль ко многим). Работник обязательно должен иметь пункт, а пункт не обязательно должен иметь работников.



Рисунок 2.2 - Связь «Доставка на»

На пункте может быть множество заказов, но заказ придет на один конкретный пункт. Следовательно, связь Пункт-Заказ будет 0..N (Ноль ко многим), Заказ-Пункт будет 1…1 (Один к одному). Заказ обязательно должен иметь пункт, но пункт может не иметь ни одного заказа.



Рисунок 2.3 - Связь «Работник обрабатывает заказ»

Работник может обрабатывать множество заказов, но заказ будет обработан одним конкретным работником. Следовательно, связь Работник-Заказ будет 0..N (Ноль ко многим), Заказ работник 0…1 (Ноль к одному). Заказ может не иметь работника, и работник может не иметь заказ.



Рисунок 2.4 - Связь «Заказ содержит товары»

Заказ может содержать множество товар, так и товар может присутствовать во множестве заказов. Следовательно, связь Товар-Заказ будет 1...N (Один ко многим), Заказ-Товар будет 0…N (Ноль ко многим). Товар не обязательно должен быть в каком-то заказе, а вот заказ должен иметь какой-то товар.



Рисунок 2.5 - Связь «Заказ имеет статус»

Статус может быть во множество заказов, но заказу будет присвоен только один статус с определенный момент времени. Следовательно, связь Статус-Заказ будет 0..N (Ноль ко многим), Заказ-Статус будет 1…1(Один к одному). Заказ обязательно должен иметь статус, но статут не должен иметь заказов.



Рисунок 2.6 - Связь «Покупатель делает заказ»

Покупатель может совершать множество заказов, но заказ будет иметь только одного покупателя. Следовательно, связь Покупатель-Заказ будет 0..N (Ноль ко многим), Заказ-Покупатель 1…1 (Один к одному). Заказ должен иметь покупателя, но покупатель может не иметь заказов.



Рисунок 2.7 - Связь «История имеет заказ»

История содержит один заказ, но заказ формирует множество состояний истории. Следовательно, связь Заказ-История будет 1..N (Один ко многим), История-Заказ будет 1…1 (Один к одному). История должна содержать заказ, и заказ должен формировать историю.



Рисунок 2.8 - Связь «История содержит статус»

История содержит один статус, но статус характеризует множество состояний истории. Следовательно, связь Статус-История будет 1..N (Ноль ко многим), История-Статус будет 1…1(Один к одному). История должна содержать статус, и статус должен характеризовать историю.



Рисунок 2.9 - Связь «Товар имеет категорию»

Товар содержит одну категорию, но категория может иметь множество товаров. Следовательно, связь будет Категория-Товар 0..N (Ноль ко многим), Товар-Категория будет 1…1 (Один к одному). Товар должен иметь категорию, но категория не должна иметь товары.

### Построение CDM

ER-диаграмма, построенная для предметной области «Пункт выдачи заказов», на основе выявленных ранее сущностей и связей между ними представлена на рисунке 2.10.

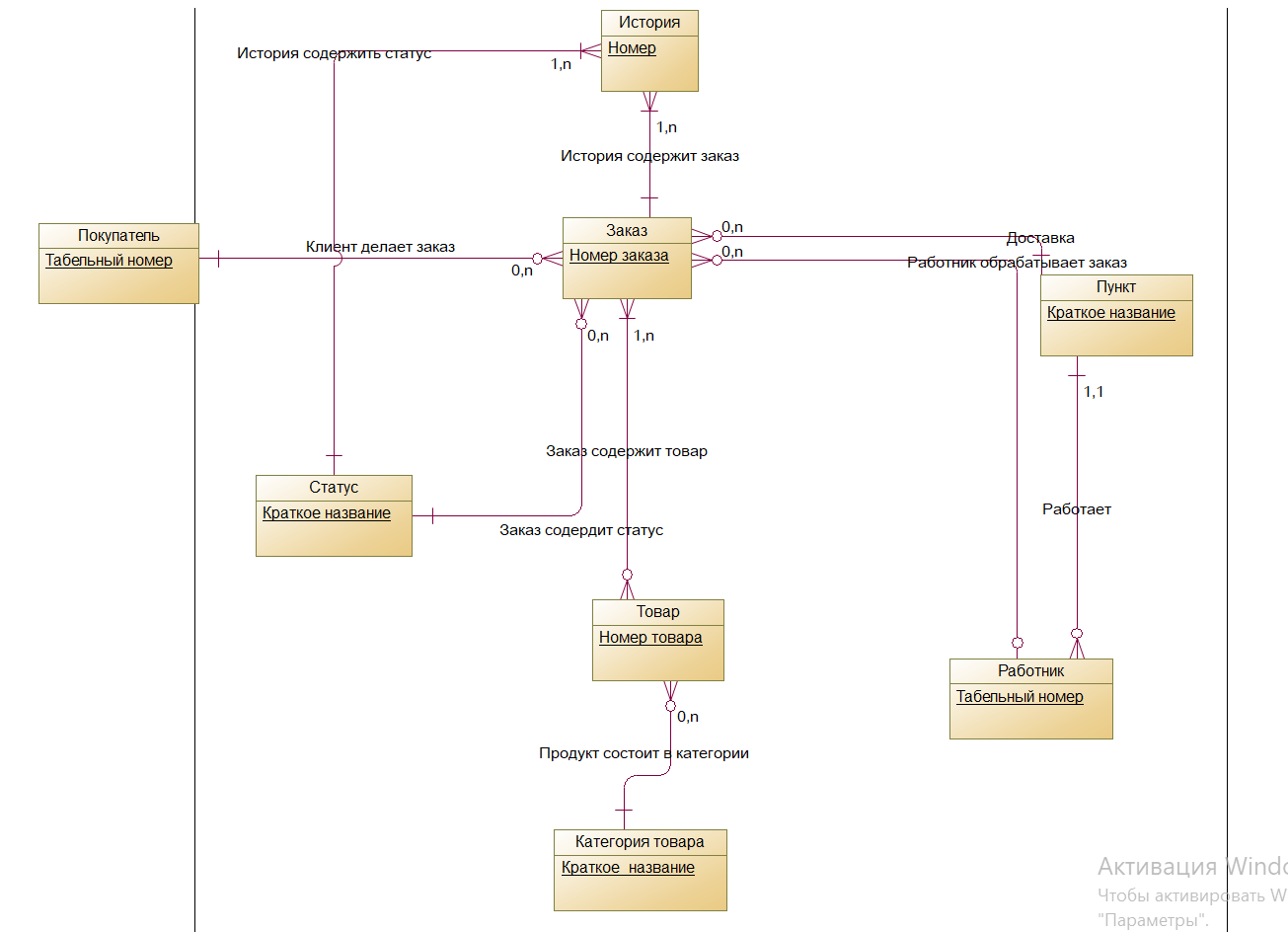


Рисунок 2.10 - Концептуальная модель «Пункт выдачи»

## Разработка логической модели данных

### Заполнение сущностей атрибутами

После перехода к логической модели данных были получены предварительные отношения, которые необходимо заполнить атрибутами:

1. Покупатель

* Табельный номер;
* Гражданство;
* Фамилия;
* Имя;
* Отчество;
* Пол;
* Дата рождения;
* Серия паспорта;
* Номер паспорта;
* Номер телефона.

1. Работник

* Табельный номер;
* Пункт;
* Логин;
* Учебное заведение;
* Образование;
* Должность;
* Фамилия;
* Имя;
* Отчество;
* Дата рождения;
* Стаж работы.

1. Пункт

* Краткое название;
* Название;
* Время работы;
* Индекс;
* Адрес.

1. Товар

* Номер товара;
* Категория товара;
* Наименование;
* Цена;
* Комментарий;
* Вес.

1. Категория товара

* Краткое название;
* Наименование категории.

1. Заказ

* Номер заказа;
* Покупатель;
* Пункт;
* Статус;
* Работник;
* Дата оформления.

1. Статус

* Краткое название;
* Наименование статуса.

1. История

* Номер;
* Статус;
* Заказ;
* Дата.

В ходе разработки логической модели были добавлены несколько таблиц и связь «Многие ко многим» тоже обратилась в таблицу:

1. Гражданство

* Код государства;
* Наименование государства.

1. Образование

* Код образования;
* Наименование образования.

1. Учебное заведение

* Код учебного заведения;
* Наименование учебного заведения.

1. Должность

* Код должности;
* Наименование должности.

1. Логин

* Код;
* Логин;
* Пароль.

1. Заказ содержит товар

* Заказ;
* Товар.

### Проверка сущностей на соответствие нормальным формам

Для каждого отношения нашей БД построим диаграмму функциональных зависимостей и определим, в какой нормальной форме оно находится.

На рисунке 2.11 показана диаграмма функциональных зависимостей для отношения «Position».



Рисунок 2.11 - Диаграмма функциональных зависимостей для отношения «Position»

Отношение находится в 1НФ, 2НФ, 3НФ, НФБК.

На рисунке 2.12 показана диаграмма функциональных зависимостей для отношения «Status».



Рисунок 2.12 - Диаграмма функциональных зависимостей для отношения «Status»

Отношение находится в 1НФ, 2НФ, 3НФ, НФБК.

На рисунке 2.13 показана диаграмма функциональных зависимостей для отношения «Login».

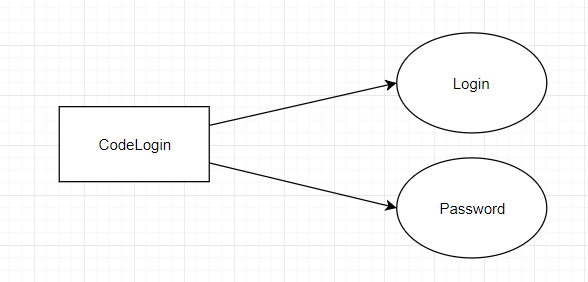


Рисунок 2.13 - Диаграмма функциональных зависимостей для отношения «Login»

Отношение находится в 1НФ, 2НФ, 3НФ, НФБК.

На рисунке 2.14 показана диаграмма функциональных зависимостей для отношения «Nationality».

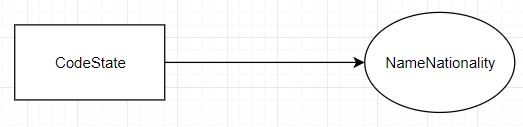


Рисунок 2.14 - Диаграмма функциональных зависимостей для отношения «Nationality»

Отношение находится в 1НФ, 2НФ, 3НФ, БКНФ.

На рисунке 2.15 показана диаграмма функциональных зависимостей для отношения «Education».

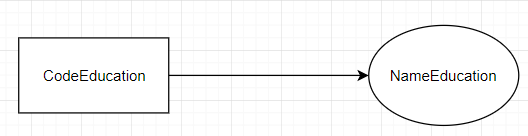


Рисунок 2.15 - Диаграмма функциональных зависимостей для отношения «Education».

Отношение находится в 1НФ, 2НФ, 3НФ, НФБК.

На рисунке 2.16 показана диаграмма функциональных зависимостей для отношения «EducationInstirurion».

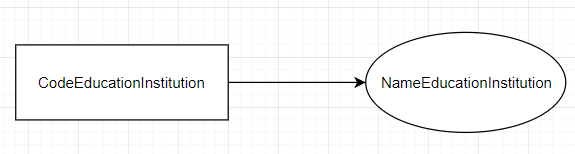


Рисунок 2.16 - Диаграмма функциональных зависимостей для отношения «EducationInstirurion»

Отношение находится в 1НФ, 2НФ, 3НФ, НФБК.

Диаграмма функциональных зависимостей для отношения «Order» примет вид, показанный на рисунке 2.17.

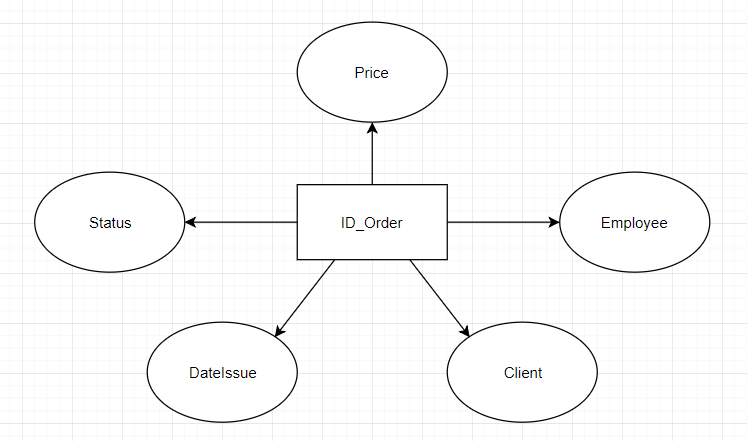


Рисунок 2.17 - Новая диаграмма функциональных зависимостей для отношения «Order»

Отношение находится в 1НФ, 2НФ, 3НФ, НФБК.

На рисунке 2.18 показана диаграмма функциональных зависимостей для отношения «Product».

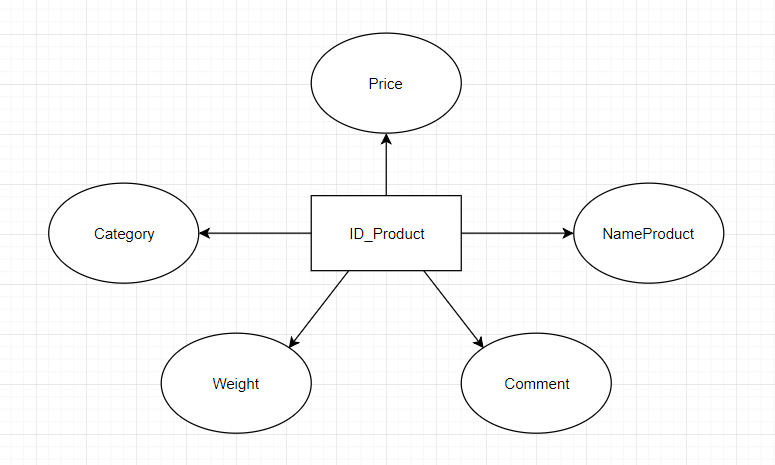


Рисунок 2.18 - Диаграмма функциональных зависимостей для отношения «Product»

Отношение находится в 1НФ, 2НФ, 3НФ, НФБК.

На рисунке 2.19 показана диаграмма функциональных зависимостей для отношения «Category».

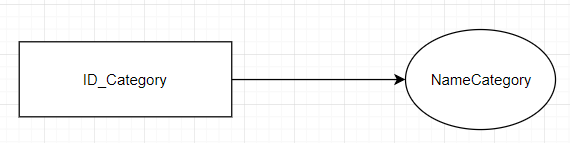


Рисунок 2.19 - Диаграмма функциональных зависимостей для отношения «Category»

Отношение находится в 1НФ, 2НФ, 3НФ, НФБК.

На рисунке 2.20 показана диаграмма функциональных зависимостей для отношения «Client».

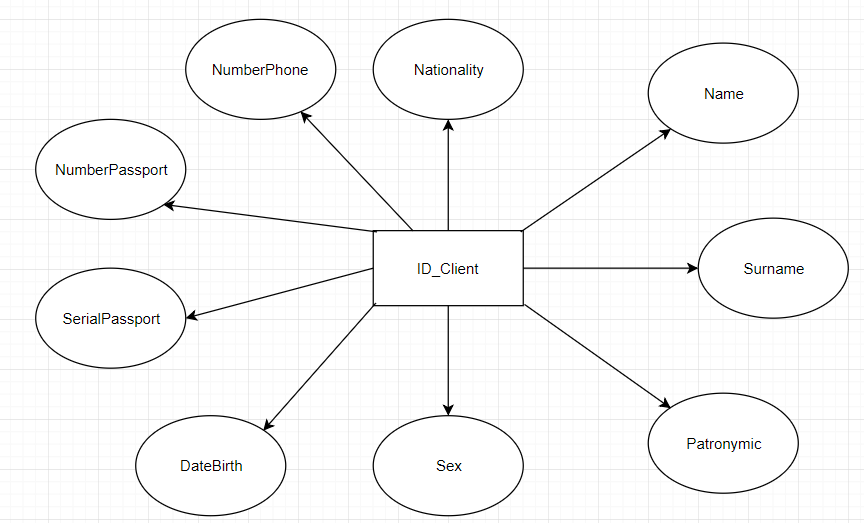


Рисунок 2.20 - Диаграмма функциональных зависимостей для отношения «Clietn»

Отношение находится в 1НФ, 2НФ, 3НФ, НФБК.

На рисунке 2.21 показана диаграмма функциональных зависимостей для отношения «Employee».

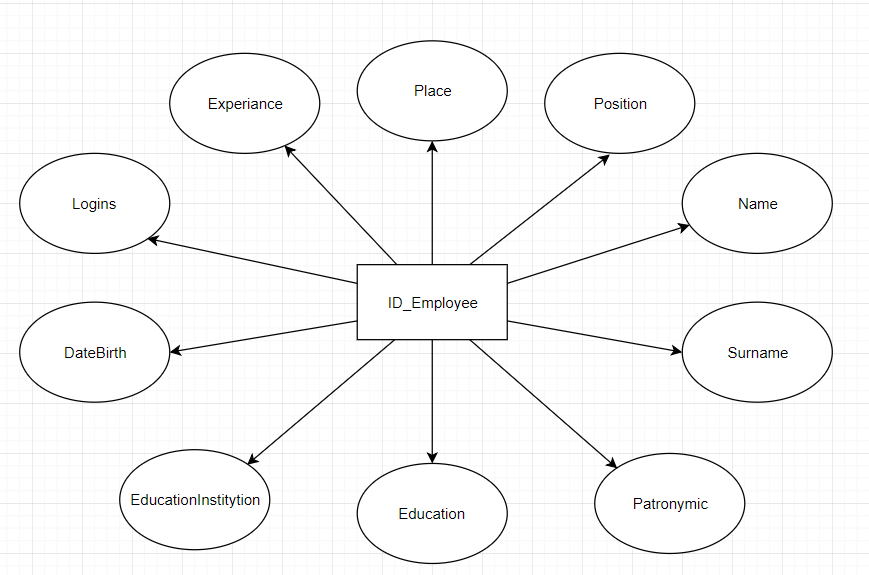


Рисунок 2.21 - Диаграмма функциональных зависимостей для отношения «Employee»

Отношение находится в 1НФ, 2НФ, 3НФ, НФБК.

На рисунке 2.22 показана диаграмма функциональных зависимостей для отношения «Place».

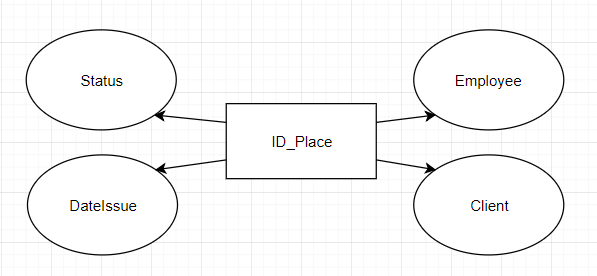


Рисунок 2.22 - Диаграмма функциональных зависимостей для отношения «Place»

Отношение находится в 1НФ, 2НФ, 3НФ, НФБК.

### Построение LDM

После заполнения атрибутами и нормализации отношений построена следующая логическая модель данных рисунок 2.23.

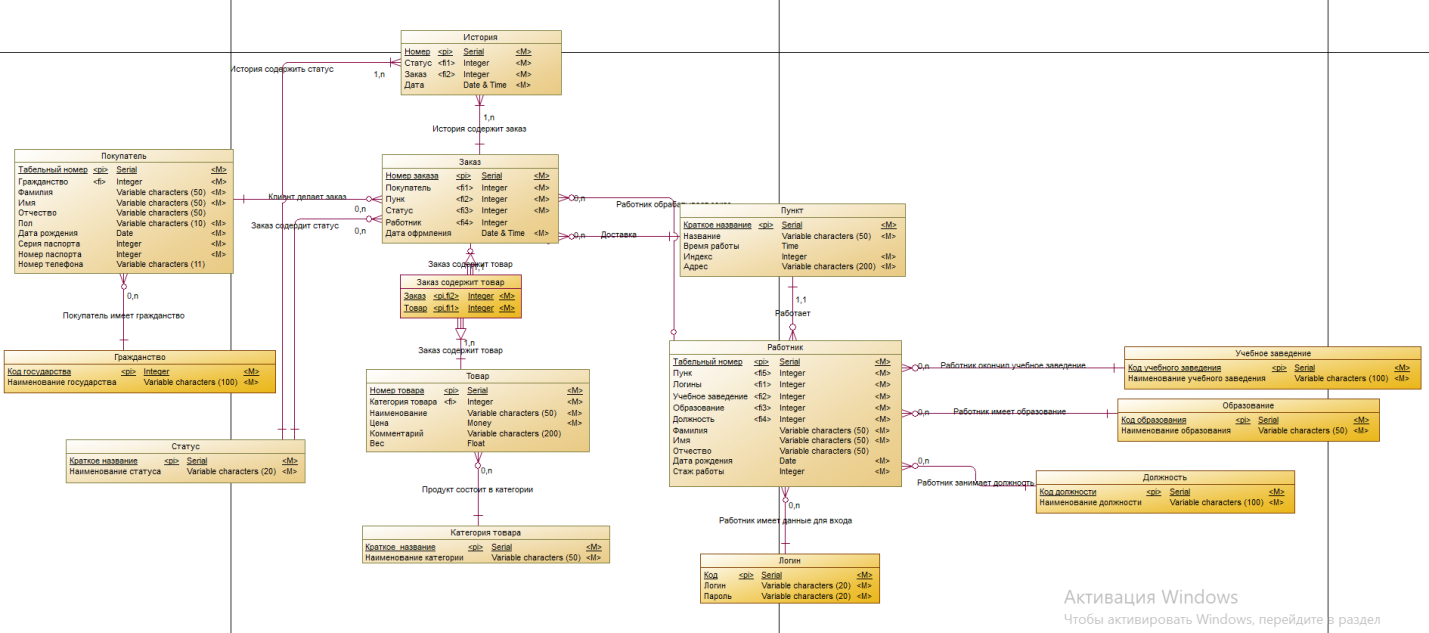


Рисунок 2.23 - Логическая модель «Пункт выдачи заказов»

## Разработка физической модели данных

### Задание типов данных для полей таблиц

Для полей таблиц были заданы типы, указанные в таблице 1.

Таблица 1

| Таблица | Поле | тип |
| --- | --- | --- |
| Order | ID\_Order | int |
| Client | int |
| Place | int |
| Status | int |
| Employee | int |
| DateIssue | datetime |
| Client | ID\_Client | int |
| Nationality | int |
| Surname | varchar(50) |
| Name | varchar(50) |
| Patronymic | varchar(50) |
| Sex | varchar(10) |
| DateOfBirth | datetime |
| SeriesPassport | int |
| NumberPassport | int |
| NumberPhone | varchar(11) |
| Employee | ID\_Employee | int |
| Place | int |
| Logins | int |
| EducationInstitution | int |
| Education | int |
| Position | int |
| Surname | varchar(50) |
| Name | varchar(50) |
| Patronymic | varchar(50) |
| DateBirth | datetime |
| Experience | int |
| Place | ID\_Place | int |
| NamePlace | varchar(50) |
| WorkTime | datetime |
| Postcode | int |
| Address | varchar(200) |
| Product | ID\_Product | int |
| Category | int |
| NameProduct | varchar(50) |
| Price | money |
| Comment | varchar(200) |
| Weight | float |
| History | ID\_History | int |
| Status | int |
| Order | int |
| Date | datetime |
| Order\_contains\_Product | Order | int |
| Product | int |
| Category | ID\_Category | int |
| NameCategory | varchar(50) |
| Login | CodeLogin | int |
| Login | varchar(20) |
| Password | varchar(20) |
| Status | ID\_Status | int |
| NameStatus | varchar(20) |
| Nationality | CodeState | int |
| NameNationality | varchar(100) |
| Education | CodeEducation | int |
|  | Name Education | varchar(50) |
| EducationInstitution | ID\_ EducationInstitution | int |
|  | NameEducationInstitution | varchar(100) |
| Position | CodePosition | int |
|  | Name Position | varchar(100) |

### Задание частных ограничений целостности данных

Частные ограничения целостности задаются первичными и внешними ключами, разрешением каскадного обновления. Ограничения целостности для взаимоисключающего законченного наследования будут реализованы на уровне хранимых процедур и путем запрета обычным пользователям БД выполнения запросов INSERT, UPDATE и DELETE напрямую. Это позволит не реализовывать данные ограничения с помощью триггеров, замедляющих работу БД при большой активности пользователей.

### Построение PDM

Физическая модель приведена на рисунке 2.24.

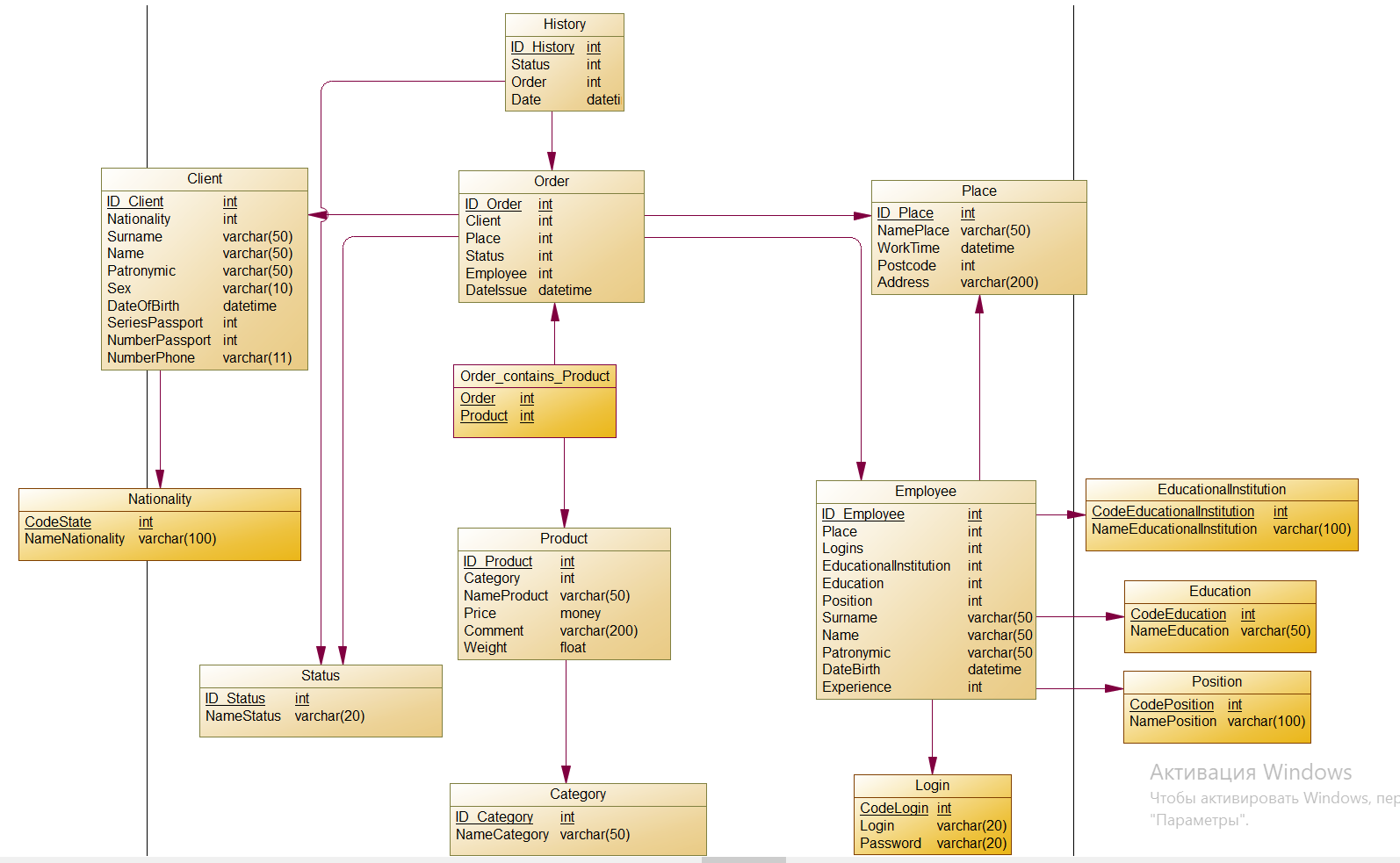


Рисунок 2.24 - «Физическая модель «Пункт выдачи заказов»

# РАЗРАБОТКА ТРИГГЕРОВ

В ходе разработки информационной системы были разработаны следующие триггеры:

1. ALTER-триггер, который срабатывает при добавлении или изменении записей в таблице Заказ:

CREATE TRIGGER trigger\_AddOrder ON "Order"

AFTER INSERT, UPDATE

1. INSTEADOF-триггер, который срабатывает при добавлении, удалении или изменении записей в таблице Статус:

CREATE TRIGGER trigger\_AddNewStatus ON Status

INSTEAD OF INSERT, DELETE, UPDATE

1. INSTEADOF -триггер, который срабатывает при добавлении, удалении и изменении записей в таблице Образование:

CREATE TRIGGER trigger\_AddNewEducation ON Education

INSTEAD OF INSERT, DELETE, UPDATE



# РАЗРАБОТКА ХРАНИМЫХ ПРОЦЕДУР

В данном курсовом проекте в ходе разработки информационной системы были созданы следующие хранимые процедуры:

1. Процедура, добавляющая нового клиента.

CREATE PROC AddClient(@SurnameClientchar(50),

@NameClient char(50),

@PatronymicClient char(50) = null,

@SexClient char(10),

@DateBirthClient datetime,

@SeriesPassportClient int,

@NumberPassportClient int,

@NameNationalityClient char(100),

@NumberPhoneClient char(11))

Входные параметры:

* @SurnameClient – Фамилия добавляемого клиента,
* @NameClient – Имя добавляемого клиента,
* @PatronymicClient – Отчество добавляемого клиента (необязательный параметр),
* @SexClient – Пол добавляемого клиента,
* @DateBirthClient – Дата Рождения добавляемого клиента,
* @SeriesPassportClient – Серия паспорта добавляемого клиента,
* @NumberPassportClient – Номер паспорта добавляемого клиента,
* @NameNationalityClient – Гражданство добавляемого клиента,
* @NumberPhoneClient – Номер телефона.

1. Процедура, добавляющая нового работника.

CREATE PROC AddEmployer(@Place int,

@Login char(20),

@Password char(20),

@NameEducationalInstitution char(100),

@Education int,

@NamePosition char(100),

@Surname char(50),

@Name char(50),

@Patronymic char(50) = null,

@DateBirth datetime,

@Experience int)

Входные параметры:

* @Place – Пункт, на котором будет работать работник,
* @Login – Логин для входа в систему,
* @Password – Пароль для входа в систему,
* @Education – Образование добавляемого работника,
* @NameEducationalInstitution– Наименование учебного заведения добавляемого работника,
* @Position – Должность добавляемого работника,
* @Surname – Фамилия добавляемого работника,
* @Name – Имя добавляемого работника,
* @Patronymic – Отчество добавляемого работника (необязательный параметр),
* @DateBirth – Дата Рождения добавляемого работника,
* @Experience – Стаж работы добавляемого клиента.

1. Процедура, добавляющая новый пункт выдачи заказов.

CREATE PROC AddPlace (@NamePlace char(50),

@WorkTime char(20),

@Postcode int,

@Address char(200))

Входные параметры:

* @NamePlace – Наименование добавляемого пункта,
* @WorkTime – Время работы добавляемого пункта,
* @Postcode – Индекс добавляемого пункта,
* @Address – Адрес добавляемого пункта.

1. Процедура, добавляющая новый пункт товар.

CREATE PROC AddProduct (@NameCategory char(50),

@NameProduct char(50),

@Price money,

@Comment char(200),

@Weight float)

Входные параметры:

* @NameCategory – Наименование категории добавляемого товара,
* @NameProduct – Наименование добавляемого товара,
* @Price – Цена добавляемого товара,
* @Comment – Комментарий добавляемого товара,
* @Weight – Вес добавляемого товара.

1. Процедура, добавляющая новую категорию товаров.

CREATE PROC AddCategory (@NameCategory char(50))

Входные параметры:

* @NameCategory – Наименование добавляемой категории.

1. Процедура, добавляющая новое гражданство.

CREATE PROC AddNationality (@NameNationality char(100))

Входные параметры:

* @NameNationality – Наименование добавляемого гражданства.

1. Процедура, добавляющая новое гражданство.

CREATE PROC AddPosition(@NamePosition char(100))

Входные параметры:

* @NameNationality – Наименование добавляемого гражданства.

1. Процедура, добавляющая новую должность.

CREATE PROC AddEducationalInstitution(@NameEducationalInstitution char(100))

Входные параметры:

* @NamePosition – Наименование добавляемой должности.

1. Процедура, добавляющая новое учебное заведение.

CREATE PROC AddEducationalInstitution(@NameEducationalInstitution char(100))

Входные параметры:

* @NamePosition – Наименование добавляемого учебного заведения.

1. Процедура, добавляющая новый заказ.

CREATE PROC AddOrder(@Client int,

@Place int,

@Products ProductsIds readonly)

Входные параметры:

* @Client – Клиент добавляемого заказа,
* @Place – Пункт, на которые поступит заказ,
* @Products – Таблица товаров добавляемого заказа.

1. Процедура, меняющая статус заказа, когда он приходит на пункт выдачи заказов.

CREATE PROC AddOrderPlace(@ID\_Order int,

@Client int,

@Place int,

@Employer int)

Входные параметры:

* @ID\_Order – Номер заказа, у которого меняется статус,
* @Client – Клиент, кому принадлежит заказ,
* @Place – Пункт, на который пришел заказ,
* @Employee – Работник, который его обрабатывает.

1. Процедура, меняющая статус заказа, когда он выдается клиенту.

CREATE PROC IssueOrder(@ID\_Order int,

@Client int,

@Place int,

@Employer int)

Входные параметры:

* @ID\_Order – Номер заказа, у которого меняется статус,
* @Client – Клиент, кому принадлежит заказ,
* @Place – Пункт, на который пришел заказ,
* @Employee – Работник, который его обрабатывает.

1. Процедура, меняющая статус заказа, когда он отменяется.

CREATE PROC OrderСancellation(@ID\_Order int)

Входные параметры:

* @ID\_Order – Номер заказа, у которого меняется статус.

# РАЗРАБОТКА ЗАПРОСОВ НА ВЫБОРКУ

1. Вывод полной информации о товарах

SELECT NameProduct, Category.ID\_Category AS Category, Price, Weight, Comment

FROM Product INNER JOIN Category ON Product.Category = Category.ID\_Category

1. Вывод полной информации о клиентах мужского пола

SELECT Surname, Name, Patronymic, Sex, DateOfBirth, Nationality.NameNationality AS NameNationality

FROM Client INNER JOIN Nationality ON Client.Nationality = Nationality.CodeState

WHERE Sex = 'Мужской'

1. Вывод информации о заказе и содержащихся в нем товарах

SELECT "Order", Product.NameProduct As NameProduct

FROM Order\_contains\_Product INNER JOIN Product ON Order\_contains\_Product.Product = Product.ID\_Product

1. Вывод полной информации о работнике, которые работает на должности «Кассир»

SELECT Surname, Name, Patronymic, Position.NamePosition AS NamePosition, Experence

FROM Employee INNER JOIN Position ON Employee.Position = Position.CodePosition

WHERE Position.NamePosition = 'Кассир'

1. Вывод информации о товарах, вес которых превышает 10

SELECT NameProduct, Price, Weight, Comment

FROM Product

WHERE Weigth > 10.00

1. Вывод информации о истории, статус которой «Оформлен»

SELECT History."Order", Status.NameStatus AS NameStatus

FROM History INNER JOIN Status ON History.Status = Status.ID\_Status

WHERE Status = 1

1. Вывод информации о работнике, стаж работы которого больше 5, но меньше 10

SELECT Surname, Name, Patronymic, Position.NamePosition AS NamePosition, Experence

FROM Employee INNER JOIN Position ON Employee.Position = Position.CodePosition

WHERE Experence > 5 AND Experence < 10

1. Вывод информации об общей сумме купленных товаров в 2019 году

SELECT SUM(Product.Price) AS SumPrice

FROM Order\_contains\_Product INNER JOIN Product ON Order\_contains\_Product.Product = Product.ID\_Product

INNER JOIN "Order" ON "Order".ID\_Order = Order\_contains\_Product."Order"

WHERE "Order".DateIssue.YEAR() = 2019

1. Вывод информации о заказах и количестве товаров в нем

SELECT "Order".ID\_Order, COUNT(Order\_contains\_Product.Product) AS CountProduct

FROM "Order" INNER JOIN Order\_contains\_Product ON "Order".ID\_Order = Order\_contains\_Product."Order"

GROUP BY "Order".ID\_Order

# ЗАКЛЮЧЕНИЕ

В итоге разработки была спроектирована база данных «Пункт выдачи заказов».

# СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Автоматизация проектирования баз данных в среде Sybase PowerDesigner. Методические указания к лабораторным работам. Рязань 2010. Благодаров А.В., Тишкин Р.В.

# ПРИЛОЖЕНИЕ 1

use [Place\_of\_issue]

/\*==============================================================\*/

/\* DBMS name: Microsoft SQL Server 2016 \*/

/\* Created on: 29.05.2019 20:35:22 \*/

/\*==============================================================\*/

/\*==============================================================\*/

/\* Table: Category \*/

/\*==============================================================\*/

create table Category (

ID\_Category int identity,

NameCategory varchar(50) not null unique

)

go

alter table Category

add constraint PK\_CATEGORY primary key (ID\_Category)

go

/\*==============================================================\*/

/\* Table: Client \*/

/\*==============================================================\*/

create table Client (

ID\_Client int identity,

Nationality int not null,

Surname varchar(50) not null,

Name varchar(50) not null,

Patronymic varchar(50) null,

Sex varchar(10) not null,

DateOfBirth datetime not null default GETDATE(),

SeriesPassport int not null,

NumberPassport int not null,

NumberPhone varchar(11) null unique

)

go

alter table Client

add constraint PK\_CLIENT primary key (ID\_Client)

go

alter table Client

add constraint CHCK\_NUMBERPASSPORT CHECK (NumberPassport > 0)

go

alter table Client

add constraint CHCK\_SERIESPASSPORT CHECK (SeriesPassport > 0)

go

/\*==============================================================\*/

/\* Table: Education \*/

/\*==============================================================\*/

create table Education (

CodeEducation int identity,

NameEducation varchar(50) not null unique

)

go

alter table Education

add constraint PK\_EDUCATION primary key (CodeEducation)

go

/\*==============================================================\*/

/\* Table: EducationalInstitution \*/

/\*==============================================================\*/

create table EducationalInstitution (

CodeEducationalInstitution int identity,

NameEducationalInstitution varchar(100) not null unique

)

go

alter table EducationalInstitution

add constraint PK\_EDUCATIONALINSTITUTION primary key (CodeEducationalInstitution)

go

/\*==============================================================\*/

/\* Table: Employee \*/

/\*==============================================================\*/

create table Employee (

ID\_Employee int identity,

Place int not null,

Logins int not null,

EducationalInstitution int not null,

Education int not null,

Position int not null,

Surname varchar(50) not null,

Name varchar(50) not null,

Patronymic varchar(50) null,

DateBirth datetime not null,

Experience int null

)

go

alter table Employee

add constraint PK\_EMPLOYEE primary key (ID\_Employee)

go

/\*==============================================================\*/

/\* Table: History \*/

/\*==============================================================\*/

create table History (

ID\_History int identity,

Status int not null,

"Order" int not null,

Date datetime not null default GETDATE()

)

go

alter table History

add constraint PK\_HISTORY primary key (ID\_History)

go

/\*==============================================================\*/

/\* Table: Login \*/

/\*==============================================================\*/

create table Login (

CodeLogin int identity,

Login varchar(20) not null unique,

Password varchar(20) not null

)

go

alter table Login

add constraint PK\_LOGIN primary key (CodeLogin)

go

/\*==============================================================\*/

/\* Table: Nationality \*/

/\*==============================================================\*/

create table Nationality (

CodeState int identity,

NameNationality varchar(100) not null unique

)

go

alter table Nationality

add constraint PK\_NATIONALITY primary key (CodeState)

go

/\*==============================================================\*/

/\* Table: "Order" \*/

/\*==============================================================\*/

create table "Order" (

ID\_Order int identity,

Client int not null,

Place int null,

Status int not null,

Employee int null,

DateIssue datetime not null default GETDATE()

)

go

alter table "Order"

add constraint PK\_ORDER primary key (ID\_Order)

go

/\*==============================================================\*/

/\* Table: Order\_contains\_Product \*/

/\*==============================================================\*/

create table Order\_contains\_Product (

"Order" int not null,

Product int not null

)

go

alter table Order\_contains\_Product

add constraint PK\_ORDER\_CONTAINS\_PRODUCT primary key ("Order", Product)

go

/\*==============================================================\*/

/\* Table: Place \*/

/\*==============================================================\*/

create table Place (

ID\_Place int identity,

NamePlace varchar(50) not null unique,

WorkTime varchar(20) null,

Postcode int not null,

Address varchar(200) not null unique

)

go

alter table Place

add constraint PK\_PLACE primary key (ID\_Place)

go

/\*==============================================================\*/

/\* Table: Position \*/

/\*==============================================================\*/

create table Position (

CodePosition int identity,

NamePosition varchar(100) not null unique

)

go

alter table Position

add constraint PK\_POSITION primary key (CodePosition)

go

/\*==============================================================\*/

/\* Table: Product \*/

/\*==============================================================\*/

create table Product (

ID\_Product int identity,

Category int not null,

NameProduct varchar(50) not null,

Price money not null,

Comment varchar(200) null,

Weight float null

)

go

alter table Product

add constraint PK\_PRODUCT primary key (ID\_Product)

go

/\*==============================================================\*/

/\* Table: Status \*/

/\*==============================================================\*/

create table Status (

ID\_Status int identity,

NameStatus varchar(20) not null unique

)

go

alter table Status

add constraint PK\_STATUS primary key (ID\_Status)

go

alter table Client

add constraint FK\_CLIENT\_CLIENT\_HA\_NATIONAL foreign key (Nationality)

references Nationality (CodeState)

go

alter table Employee

add constraint FK\_EMPLOYEE\_EMPLOYEE\_\_EDUCATIO foreign key (EducationalInstitution)

references EducationalInstitution (CodeEducationalInstitution)

go

alter table Employee

add constraint FK\_EMPLOYEE\_EMPLOYEE\_\_EDUCATION foreign key (Education)

references Education (CodeEducation)

go

alter table Employee

add constraint FK\_EMPLOYEE\_EMPLOYEE\_\_POSITION foreign key (Position)

references Position (CodePosition)

go

alter table Employee

add constraint FK\_EMPLOYEE\_EMPOYEE\_H\_LOGIN foreign key (Logins)

references Login (CodeLogin)

go

alter table Employee

add constraint FK\_EMPLOYEE\_WORK\_PLACE foreign key (Place)

references Place (ID\_Place)

go

alter table History

add constraint FK\_HISTORY\_HISTORY\_C\_STATUS foreign key (Status)

references Status (ID\_Status)

go

alter table History

add constraint FK\_HISTORY\_HISTORY\_C\_ORDER foreign key ("Order")

references "Order" (ID\_Order)

go

alter table "Order"

add constraint FK\_ORDER\_CLIENT\_MA\_CLIENT foreign key (Client)

references Client (ID\_Client)

go

alter table "Order"

add constraint FK\_ORDER\_DELIVERY\_PLACE foreign key (Place)

references Place (ID\_Place)

go

alter table "Order"

add constraint FK\_ORDER\_EMPLOYEE\_\_EMPLOYEE foreign key (Employee)

references Employee (ID\_Employee)

go

alter table "Order"

add constraint FK\_ORDER\_ORDER\_CON\_STATUS foreign key (Status)

references Status (ID\_Status)

go

alter table Order\_contains\_Product

add constraint FK\_ORDER\_CO\_ORDER\_CON\_ORDER foreign key ("Order")

references "Order" (ID\_Order)

go

alter table Order\_contains\_Product

add constraint FK\_ORDER\_CO\_ORDER\_CON\_PRODUCT foreign key (Product)

references Product (ID\_Product)

go

alter table Product

add constraint FK\_PRODUCT\_PRODUCT\_C\_CATEGORY foreign key (Category)

references Category (ID\_Category)

go

use [Place\_of\_issue]

CREATE PROC AddClient (@SurnameClient char(50),

@NameClient char(50),

@PatronymicClient char(50) = null,

@SexClient char(10),

@DateBirthClient datetime,

@SeriesPassportClient int,

@NumberPassportClient int,

@NameNationalityClient char(100),

@NumberPhoneClient char(11))

AS BEGIN

DECLARE @CodeState int

SELECT @CodeState = 0

SELECT @CodeState = CodeState

FROM Nationality

WHERE NameNationality = @NameNationalityClient

IF @CodeState > 0

INSERT INTO Client

VALUES(@CodeState, @SurnameClient, @NameClient, @PatronymicClient, @SexClient, @DateBirthClient, @SeriesPassportClient, @NumberPassportClient, @NumberPhoneClient)

ELSE

BEGIN

INSERT INTO Nationality

VALUES(@NameNationalityClient)

SELECT @CodeState = @@IDENTITY

INSERT INTO Client

VALUES(@CodeState, @SurnameClient, @NameClient, @PatronymicClient, @SexClient, @DateBirthClient, @SeriesPassportClient, @NumberPassportClient, @NumberPhoneClient)

END

END

CREATE PROC AddEmployer(@Place int,

@Login char(20),

@Password char(20),

@NameEducationalInstitution char(100),

@Education int, @NamePosition char(100),

@Surname char(50),

@Name char(50),

@Patronymic char(50) = null,

@DateBirth datetime,

@Experience int)

AS

BEGIN TRAN

IF NOT EXISTS(SELECT \*

FROM Place

WHERE ID\_Place = @Place)

ROLLBACK TRAN

IF NOT EXISTS(SELECT \*

FROM Education

WHERE CodeEducation = @Education)

ROLLBACK TRAN

IF NOT EXISTS(SELECT \*

FROM EducationalInstitution

WHERE NameEducationalInstitution = @NameEducationalInstitution)

BEGIN

INSERT INTO EducationalInstitution

VALUES(@NameEducationalInstitution)

END

IF NOT EXISTS(SELECT \*

FROM Position

WHERE NamePosition = @NamePosition)

BEGIN

INSERT INTO Position

VALUES(@NamePosition)

END

DECLARE @CodeEducationalInstitution int

DECLARE @CodePosition int

DECLARE @CodeLogin int

SELECT @CodeEducationalInstitution = CodeEducationalInstitution

FROM EducationalInstitution

WHERE NameEducationalInstitution = @NameEducationalInstitution

SELECT @CodePosition = CodePosition

FROM Position

WHERE NamePosition = @NamePosition

INSERT INTO Login

VALUES(@Login, @Password)

SELECT @CodeLogin = @@IDENTITY

INSERT INTO Employee

VALUES(@Place, @CodeLogin, @CodeEducationalInstitution, @Education, @CodePosition, @Surname, @Name, @Password, @DateBirth, @Experience)

COMMIT TRAN

GO

CREATE PROC AddPlace (@NamePlace char(50),

@WorkTime char(20),

@Postcode int,

@Address char(200))

AS

INSERT INTO Place

VALUES(@NamePlace, @WorkTime, @Postcode, @Address)

GO

CREATE PROC AddProduct (@NameCategory char(50),

@NameProduct char(50),

@Price money,

@Comment char(200),

@Weight float)

AS

DECLARE @ID\_Category int

SELECT @ID\_Category = 0

SELECT @ID\_Category = ID\_Category

FROM Category

WHERE NameCategory = @NameCategory

IF @ID\_Category > 0

INSERT INTO Product

VALUES(@ID\_Category, @NameProduct, @Price, @Comment, @Weight)

ELSE

BEGIN

INSERT INTO Category

VALUES(@NameCategory)

SELECT @ID\_Category = @@IDENTITY

INSERT INTO Product

VALUES(@ID\_Category, @NameProduct, @Price, @Comment, @Weight)

END

GO

CREATE PROC AddCategory (@NameCategory char(50))

AS

INSERT INTO Category

VALUES(@NameCategory)

GO

CREATE PROC AddNationality (@NameNationality char(100))

AS

INSERT INTO Nationality

VALUES(@NameNationality)

GO

CREATE PROC AddPosition(@NamePosition char(100))

AS

INSERT INTO Position

VALUES(@NamePosition)

GO

CREATE PROC AddEducationalInstitution(@NameEducationalInstitution char(100))

AS

INSERT INTO EducationalInstitution

VALUES(@NameEducationalInstitution)

GO

CREATE TYPE ProductsIds

AS TABLE(id int)

GO

ALTER PROC AddOrder(@Client int,

@Place int,

@Products ProductsIds readonly)

AS

BEGIN TRAN

IF NOT EXISTS(SELECT \*

FROM Place

WHERE ID\_Place = @Place)

IF NOT EXISTS(SELECT \*

FROM Client

WHERE ID\_Client = @Client)

ROLLBACK TRAN

DECLARE @ID\_Order int

INSERT INTO "Order"

VALUES(@Client, @Place, 1, NULL,GETDATE())

SELECT @ID\_Order = @@IDENTITY

INSERT INTO Order\_contains\_Product("Order", Product)

SELECT @ID\_Order, id

FROM @Products

COMMIT TRAN

GO

CREATE PROC AddOrderPlace(@ID\_Order int,

@Client int,

@Place int,

@Employer int)

AS

BEGIN TRAN

IF NOT EXISTS(SELECT \*

FROM Place

WHERE ID\_Place = @Place)

ROLLBACK TRAN

IF NOT EXISTS(SELECT \*

FROM Client

WHERE ID\_Client = @Client)

ROLLBACK TRAN

IF NOT EXISTS(SELECT \*

FROM Employee

WHERE ID\_Employee = @Employer)

ROLLBACK TRAN

IF NOT EXISTS(SELECT \*

FROM Employee

WHERE ID\_Employee = @Employer AND Place = @Place)

ROLLBACK TRAN

UPDATE "Order"

SET Client = @Client, Place = @Place, Employee = @Employer, Status = 2

WHERE ID\_Order = @ID\_Order

COMMIT TRAN

GO

CREATE PROC IssueOrder(@ID\_Order int,

@Client int,

@Place int,

@Employer int)

AS

BEGIN TRAN

IF NOT EXISTS(SELECT \*

FROM Place

WHERE ID\_Place = @Place)

ROLLBACK TRAN

IF NOT EXISTS(SELECT \*

FROM Client

WHERE ID\_Client = @Client)

ROLLBACK TRAN

IF NOT EXISTS(SELECT \*

FROM Employee

WHERE ID\_Employee = @Employer)

ROLLBACK TRAN

IF NOT EXISTS(SELECT \*

FROM Employee

WHERE ID\_Employee = @Employer AND Place = @Place)

ROLLBACK TRAN

UPDATE "Order"

SET Status = 3

WHERE ID\_Order = @ID\_Order

COMMIT TRAN

GO

CREATE PROC OrderСancellation(@ID\_Order int)

AS

BEGIN TRAN

UPDATE "Order"

SET Status = 4

WHERE ID\_Order = @ID\_Order

COMMIT TRAN

GO

CREATE TRIGGER trigger\_AddOrder ON "Order"

AFTER INSERT, UPDATE

AS

IF EXISTS(SELECT \*

FROM INSERTED)

BEGIN

INSERT INTO History(Status, "Order") SELECT Status, ID\_Order FROM INSERTED;

END

GO

CREATE TRIGGER trigger\_AddNewStatus ON Status

INSTEAD OF INSERT, DELETE, UPDATE

AS

IF NOT EXISTS(SELECT \*

FROM INSERTED)

BEGIN

INSERT INTO Status

VALUES('Заказ добавлен')

INSERT INTO Status

VALUES('Заказ поступил на пункт')

INSERT INTO Status

VALUES('Заказ выдан получателю')

INSERT INTO Status

VALUES('Заказ отменен')

END

ELSE

BEGIN

PRINT 'Эту таблицу изменять нельзя'

ROLLBACK TRAN

END

GO

CREATE TRIGGER trigger\_AddNewEducation ON Education

INSTEAD OF INSERT, DELETE, UPDATE

AS

IF NOT EXISTS(SELECT \*

FROM Education)

BEGIN

INSERT INTO Education

VALUES('Высшее')

INSERT INTO Education

VALUES('Неоконченное высшее')

INSERT INTO Education

VALUES('Среднее')

INSERT INTO Education

VALUES('Неоконченное среднее')

INSERT INTO Education

VALUES('Среднее профессиональное')

END

ELSE

BEGIN

PRINT 'Эту таблицу изменять нельзя'

ROLLBACK TRAN

END

GO

# ПРИЛОЖЕНИЕ 2