**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 | 03.01 | Технология разработки и защиты баз данных | | | | | | |
|  | | | | | | | | |
|  | | | 6 семестр | | | | | |
| Тема курсовой работы | | | Разработка базы данных | | | | | |
|  | | | ***«***Издательство» | | | | | |
|  | | |  | | | | | |
| АКВТ.09.02.07.КР00.00ПЗ | | | | | | | | |
| Листов: | | | | 43 |  | | | |
|  | |  | | | | | | |
|  | |  | | | | | | |
| Студент | |  | | | | Студент гр. | | ПБ – 31 |
|  | |  | | | |  | | |
|  | |  | | | |  |  | |
| Руководитель | |  | | | | 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** | | | |
| ***[ДАННЫЕ УДАЛЕНЫ]*** | | | |
| (фамилия, имя, отчество студента) | | | |
| Тема курсового проекта | | ***Разработка БД*** | |
| ***«Издательство»*** | | | |

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

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Дата выдачи задания | « |  |  | » |  | |  | 20 |  | г. | |
| Руководитель |  |  | | | | Р.Р. Мулеев | | | | |
| Студент |  |  | | | |  | | | | |

**Аннотация**

Данный документ на разработку базы данных имеет 43 страниц, 4 раздела, 3 рисунка, 5 источника использованной литературы. Таблицы и приложения отсутствуют

**Сожержание**

[Введение 5](#_Toc105529452)

[1. Проектирование БД 6](#_Toc105529453)

[1.1 Анализ предметной области 6](#_Toc105529454)

[1.2 Концептуальная модель данных 7](#_Toc105529455)

[1.2.1 Сущности 7](#_Toc105529456)

[1.2.2 Связи 7](#_Toc105529457)

[1.3 Логическая модель данных 8](#_Toc105529458)

[1.3.1 Сущности 8](#_Toc105529459)

[1.4 Выбор СУБД 16](#_Toc105529460)

[2 Реализация базы данных 17](#_Toc105529461)

[2.1 Физическое проектирование БД 17](#_Toc105529462)

[2.2 Написание исходного кода БД 17](#_Toc105529463)

[2.3 Написание запросов, функций и хранимых процедур 24](#_Toc105529464)

[2.3.1 Хранимые процедуры 24](#_Toc105529465)

[2.3.2 Тригеры 41](#_Toc105529466)

[3 Заключение 42](#_Toc105529467)

[4 Список используемой литературы 43](#_Toc105529468)

# 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 и т.д..

БД «Издательство» предназначена для структуризации информации об издательствах, заказчиках, публикации, авторах публикаций, типов публикации и типографий, формирования отчетности.

# Проектирование БД

## Анализ предметной области

Издательство — предприятие (государственное, общественное, кооперативное или частное) — медиакомпания, которая работает в области литературы, искусства, музыки или науки, и продукция которой может воспроизводиться и распространяться.

Целью издательства является:

* зарабатывание денег себе и авторам;
* участвовать в подготовке произведений и сборников к печати;
* продвижение книг на рынок;
* и другие цели, прописанные в уставе издательства.

Предметная область представляет собой базу данных, направлена на информационную поддержку издательства. База данных даёт возможность проведение аналитики и создание отчётности по деятельности всей организации, а также каждого заказа отдельно, автоматизирует некоторые процессы. Ввиду всех возможностей базы данных возникает необходимость в автоматизации информационных процессов.

Документ, дающий основание разработки базы данных — это устав издательства

Входные документы:

* документ на заказ;

Выходные документы:

* Отчёт о деятельности издательства;
* Отчёт об заказах.

База данных создана для сотрудников издательства, которые могут просматривать, вносить, изменять, удалять информацию в базе данных в полном объёме.

База данных должна осуществлять хранение данных о:

* сведения о заказчиках (частное лицо или организация, личные данные контактной персоны, адрес, телефон, факс);
* сведения о заказах (номер заказа, заказчик, вид печатной продукции, издание, типография, дата приема заказа, отметка о выполнении, дата выполнения заказа);
* сведения об изданиях (код издания, автор и название, объем в печатных листах, тираж, номер заказа);
* сведения об авторах (личные данные: Ф.И.О., домашний адрес, телефон; дополнительные сведения);
* сведения о типографиях (название, адрес, телефон).

## Концептуальная модель данных

### **Сущности**

1. Организация - содержит информацию о организации, которая заказала продукцию.

Атрибуты: адрес, название.

1. Представитель - содержит информацию о представителе организации или автора.

Атрибуты: ФИО, Телефон.

1. Заказ - содержит информацию о заказе.

Атрибуты: Дата приёма/выполнения, вид продукции.

1. Типография - содержит информацию о типографии, выполняющую заказ.

Атрибуты: название, адрес.

1. Издание - содержит информацию о изданиях издательства.

Атрибуты: наименование.

1. Автор – содержит информацию о авторах издания.

Атрибуты: ФИО, Телефон.

### **Связи**

1. Представитель представляет автора. Представитель может представлять много авторов, а у автора может быть много представителей (связь много к многим).
2. Представитель представляет организацию. Представитель может представлять много организации, а у организации может быть много представителей (связь много к многим).
3. Представитель оформлять заказ. Представитель может заказать много заказов, а у заказа может быть только один представитель (связь 1 к многим).
4. Типография выполняет заказ. Типография может выполнять много заказов, а у заказа может быть только одна типография (связь 1 к многим).
5. Результат заказа является издание (связь 1 к 1).
6. Автор пишет издание. У автора может быть много изданий и у издания может быть много авторов (связь много ко многим).

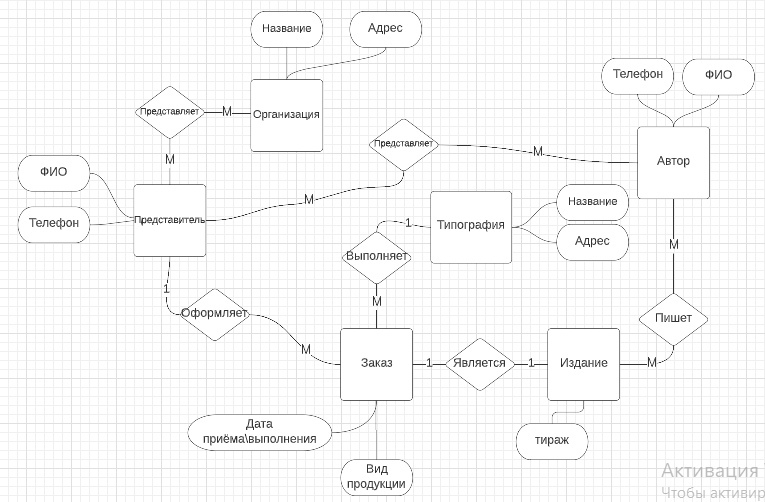


Рисунок 1 – концептуальная модель

## Логическая модель данных

### **Сущности**

1. Город (City):
   1. Идентификатор города (CityId):

* нулевое значение: not null;
* тип поля: int;
* первичный ключ, созданный искусственно из-за отсутствия атрибута идентификатора у сущности;
* автоинкримированный.
  1. Название (Name):
* нулевое значение: not null;
* тип поля: nvarchar;
* маска ввода: только буквы.

1. Тип продукции (ProductType):
   1. Идентификатор типа продукции (ProductTypeId):

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

1. Организация (Organization):
   1. Идентификатор организации (OrganizationId):

* нулевое значение: not null;
* тип поля: int;
* первичный ключ, созданный искусственно из-за отсутствия атрибута идентификатора у сущности;
* автоинкримированный.
  1. Идентификатор города (CityId)
* нулевое значение: not null;
* тип поля: int;
* вторичный ключ, ссылающийся на таблицу City (поле CytiId);
* нормализованная форма атрибута Адрес.
  1. Улица (Street), номер дома (NumberHouse):
* нулевое значение: not null;
* тип поля: nvarchar;
* нормализованная форма атрибута Адрес.

1. Представители организации (OrganizationRepresentative), разгрузочная таблица, созданная для нормализации связи между таблицами, Представитель и Организация:
   1. Идентификатор представителя организации (OrganizationRepresentativeId):

* нулевое значение: not null;
* тип поля: int;
* первичный ключ, созданный искусственно из-за отсутствия атрибута идентификатора у сущности;
* автоинкримированный.
  1. Идентификатор организации (OrganizationId):
* нулевое значение: not null;
* тип поля: int;
* вторичный ключ, ссылающийся на таблицу Organization (поле OrganizationId).
  1. Идентификатор представителя (RepresentativeId):
* нулевое значение: not null;
* тип поля: int;
* вторичный ключ, ссылающийся на таблицу Representative (поле RepresentativeId).

1. Автор (Author):
   1. Идентификатор автора (AuthorId):

* 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: not null;
* тип поля: bigint;
* маска ввода: 10 цифр.
  1. Идентификатор города (CityId)
* нулевое значение: not null;
* тип поля: int;
* вторичный ключ, ссылающийся на таблицу City (поле CytiId);
* нормализованная форма атрибута Адрес.
  1. Улица (Street), номер дома (NumberHouse):
* нулевое значение: not null;
* тип поля: nvarchar;
* нормализованная форма атрибута Адрес.

1. Представители автора (AuthorRepresentative), разгрузочная таблица, созданная для нормализации связи между таблицами, Представитель и Автор:
   1. Идентификатор представителя организации (AuthorRepresentativeId):

* нулевое значение: not null;
* тип поля: int;
* первичный ключ, созданный искусственно из-за отсутствия атрибута идентификатора у сущности;
* автоинкримированный.
  1. Идентификатор организации (AuthorId):
* нулевое значение: not null;
* тип поля: int;
* вторичный ключ, ссылающийся на таблицу Author (поле AuthorId).
  1. Идентификатор представителя (RepresentativeId):
* нулевое значение: not null;
* тип поля: int;
* вторичный ключ, ссылающийся на таблицу Representative (поле RepresentativeId).

1. Статус заказа (StatusOrder):
   1. Идентификатор статуса заказа (StatusOrderId):

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

1. Издание (Publication):
   1. Идентификатор издания (PublicationId):

* нулевое значение: not null;
* тип поля: int;
* первичный ключ, созданный искусственно из-за отсутствия атрибута идентификатора у сущности;
* автоинкримированный.
  1. Наименование (Name):
* нулевое значение: not null;
* тип поля: nvarchar.
  1. Количество страниц (VolumeList)
* нулевое значение: not null;
* тип поля: int;
* маска ввода: больше 0.
  1. Тип продукции (ProductTypeId)
* нулевое значение: not null;
* тип поля: int;
* вторичный ключ, ссылающийся на таблицу ProductType (поле ProductTypeId).

1. Издания автора (AuthorPublication), разгрузочная таблица, созданная для нормализации связи между таблицами, Издание и Автор:
   1. Идентификатор представителя организации (AuthorPublicationId):

* нулевое значение: not null;
* тип поля: int;
* первичный ключ, созданный искусственно из-за отсутствия атрибута идентификатора у сущности;
* автоинкримированный.
  1. Идентификатор организации (AuthorId):
* нулевое значение: not null;
* тип поля: int;
* вторичный ключ, ссылающийся на таблицу Author (поле AuthorId).
  1. Идентификатор издания (PublicationId):
* нулевое значение: not null;
* тип поля: int;
* вторичный ключ, ссылающийся на таблицу Publication (поле PublicationId).
  1. Соавтор (CoAuthor):
* нулевое значение: not null;
* тип поля: bit.

1. Типография (PrintingHouse):
   1. Идентификатор типографии (PrintingHouseId):

* нулевое значение: not null;
* тип поля: int;
* первичный ключ, созданный искусственно из-за отсутствия атрибута идентификатора у сущности;
* автоинкримированный.
  1. Идентификатор города (CityId)
* нулевое значение: not null;
* тип поля: int;
* вторичный ключ, ссылающийся на таблицу City (поле CytiId);
* нормализованная форма атрибута Адрес.
  1. Улица (Street), номер дома (NumberHouse):
* нулевое значение: not null;
* тип поля: nvarchar;
* нормализованная форма атрибута Адрес.

1. Представитель (Representative):
   1. Идентификатор автора (RepresentativeId):

* 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. Идентификатор города (CityId)
* нулевое значение: not null;
* тип поля: int;
* вторичный ключ, ссылающийся на таблицу City (поле CytiId);
* нормализованная форма атрибута Адрес.
  1. Улица (Street), номер дома (NumberHouse):
* нулевое значение: not null;
* тип поля: nvarchar;
* нормализованная форма атрибута Адрес.
  1. Номер телефона (PhoneNumber):
* нулевое значение: not null;
* тип поля: bigint;
* маска ввода: 10 цифр.
  1. Факс (fax):
* нулевое значение: not null;
* тип поля: bigint;
* маска ввода: 10 цифр.
  1. Организация (IsOrganization):
* нулевое значение: not null;
* тип поля: bit;

1. Заказ (Order):
   1. Идентификатор заказа (OrderId):

* нулевое значение: not null;
* тип поля: int;
* первичный ключ, созданный искусственно из-за отсутствия атрибута идентификатора у сущности;
* автоинкримированный.
  1. Идентификатор представителя (RepresentativeId):
* нулевое значение: not null;
* тип поля: int;
* вторичный ключ, ссылающийся на таблицу Representative (поле RepresentativeId).
  1. Дата выполнения (DateCompiction):
* нулевое значение: not null;
* тип поля: date.
  1. Дата приёма (DateAdmission):
* нулевое значение: not null;
* тип поля: date.
  1. Дата статуса (StatusDate):
* нулевое значение: not null;
* тип поля: date.
  1. Идентификатор статуса заказа (StatusOrderId):
* нулевое значение: not null;
* тип поля: int;
* вторичный ключ, ссылающийся на таблицу StatusOrder (поле StatusOrderId).

1. Стока заказа (OrderLines):
   1. Идентификатор строки заказа (OrderLinesId):

* нулевое значение: not null;
* тип поля: int;
* первичный ключ, созданный искусственно из-за отсутствия атрибута идентификатора у сущности;
* автоинкримированный.
  1. Идентификатор заказа (OrderId):
* нулевое значение: not null;
* тип поля: int;
* вторичный ключ, ссылающийся на таблицу Order (поле OrderId).
  1. Идентификатор издания (PublicationId):
* нулевое значение: not null;
* тип поля: int;
* вторичный ключ, ссылающийся на таблицу Publication (поле PublicationId).
  1. Идентификатор типографии (PrintingHouseId):
* нулевое значение: not null;
* тип поля: int;
* вторичный ключ, ссылающийся на таблицу PrintingHouse (поле PrintingHouseId).
  1. Количество (Quantity):
* нулевое значение: not null;
* тип поля: int;
* маска ввода: больше 0.
  1. Цена (Price):
* нулевое значение: not null;
* тип поля: float;
* маска ввода: больше 0.

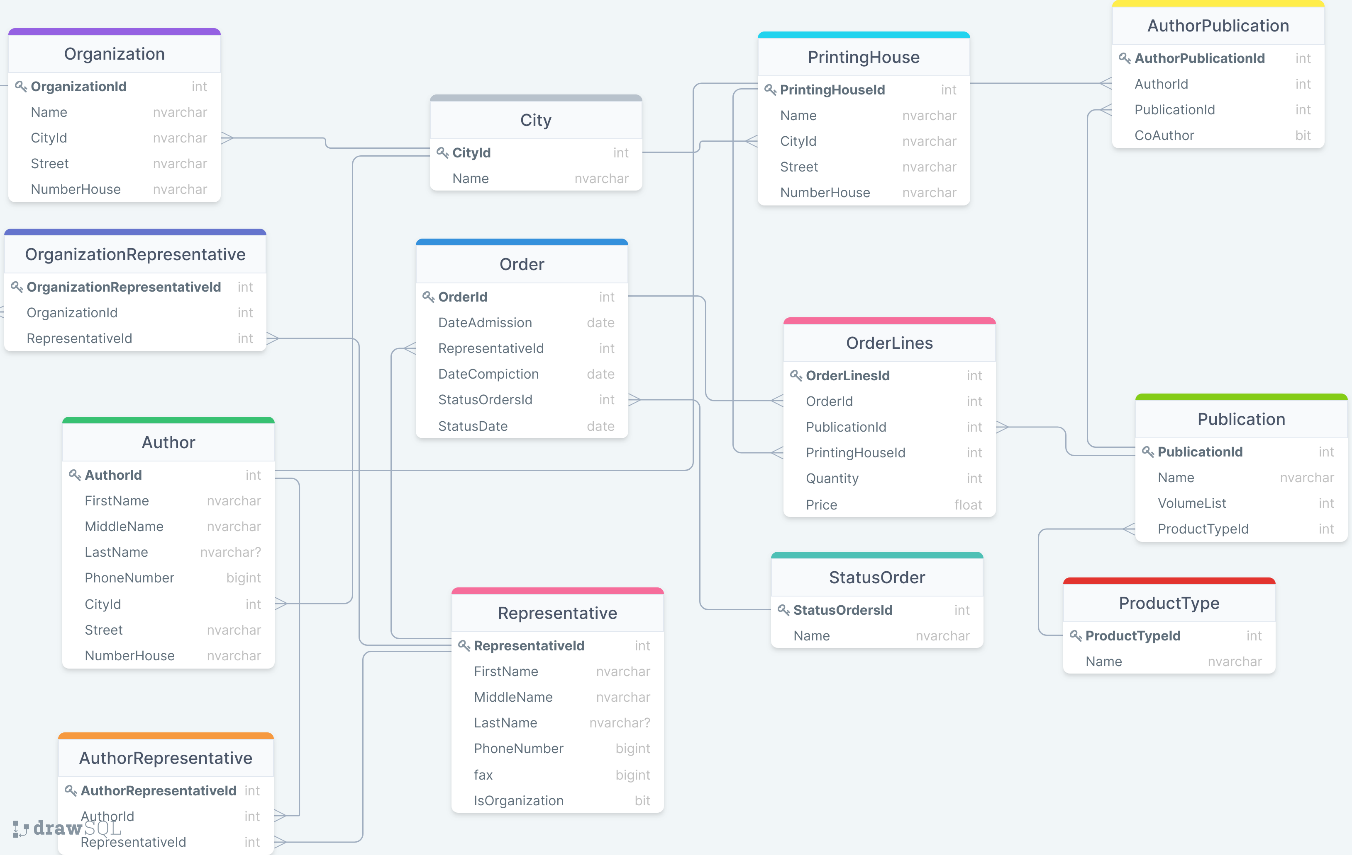


Рисунок 2 – логическая модель

## 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.

# Реализация базы данных

## Физическое проектирование БД

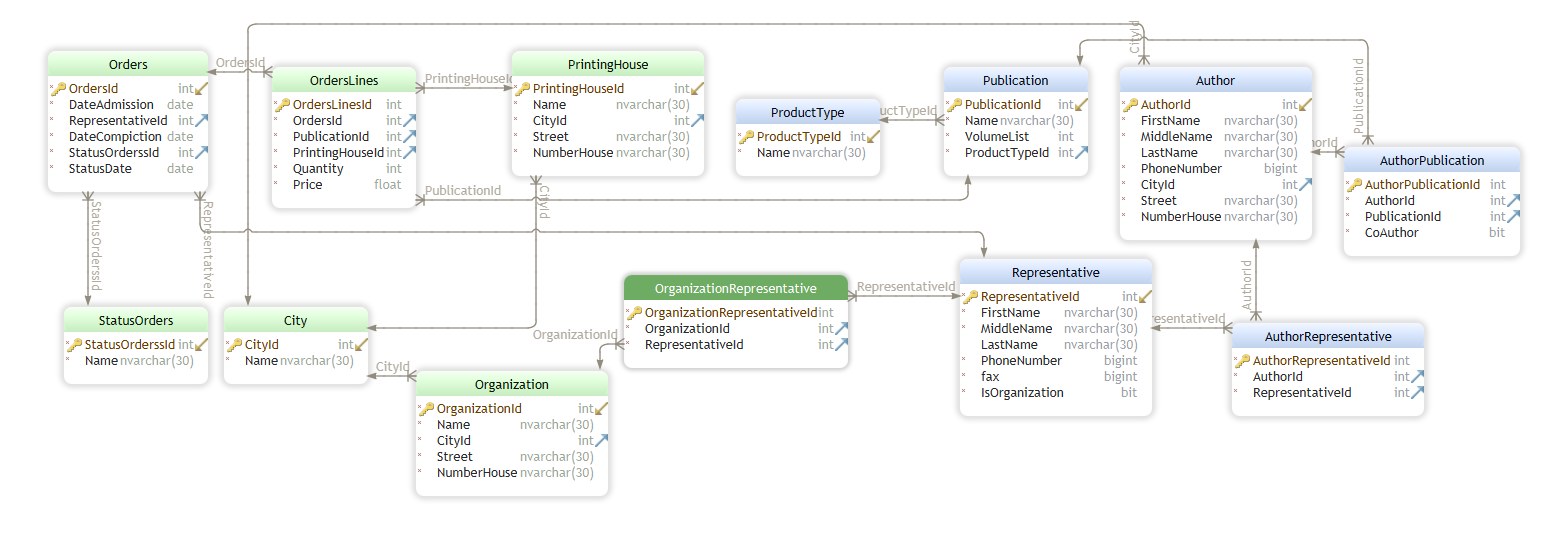


Рисунок 3 – диаграмма базы данных

## Написание исходного кода БД

* 1. AuthorPublication

Данная таблица создана с целью хранения информации о публикациях автора и связи таблиц Author и Publication.

Листинг:

CREATE TABLE AuthorPublication(

AuthorPublicationId INT identity(1,1) NOT NULL,

AuthorId INT NOT NULL,

PublicationId INT NOT NULL,

СoАuthor BIT NOT NULL

);

ALTER TABLE

AuthorPublication ADD CONSTRAINT authorpublication\_authorpublicationid\_primary PRIMARY KEY(AuthorPublicationId);

ALTER TABLE

AuthorPublication ADD CONSTRAINT authorpublication\_publicationid\_foreign FOREIGN KEY(PublicationId) REFERENCES Publication(PublicationId);

ALTER TABLE

AuthorPublication ADD CONSTRAINT authorpublication\_authorid\_foreign FOREIGN KEY(AuthorId) REFERENCES Author(AuthorId);

* 1. ОrganizationRepresentative

Данная таблица создана с целью хранения информации о представителей организации и связи таблиц Оrganization и Representative.

Листинг:

CREATE TABLE ОrganizationRepresentative(

ОrganizationRepresentativeId INT identity(1,1) NOT NULL,

ОrganizationId INT NOT NULL,

RepresentativeId INT NOT NULL

);

ALTER TABLE

ОrganizationRepresentative ADD CONSTRAINT Оrganizationrepresentative\_Оrganizationrepresentativeid\_primary PRIMARY KEY (ОrganizationRepresentativeId);

ALTER TABLE

ОrganizationRepresentative ADD CONSTRAINT Оrganizationrepresentative\_representativeid\_foreign FOREIGN KEY (RepresentativeId) REFERENCES Representative (RepresentativeId);

ALTER TABLE

ОrganizationRepresentative ADD CONSTRAINT Оrganizationrepresentative\_Оrganizationid\_foreign FOREIGN KEY (ОrganizationId) REFERENCES Оrganization (ОrganizationId);

* 1. AuthorRepresentative

Данная таблица создана с целью хранения информации о представителей авторов и связи таблиц Author и Representative.

Листинг:

CREATE TABLE AuthorRepresentative(

AuthorRepresentativeId INT identity(1,1) NOT NULL,

AuthorId INT NOT NULL,

RepresentativeId INT NOT NULL

);

ALTER TABLE

AuthorRepresentative ADD CONSTRAINT authorrepresentative\_authorrepresentativeid\_primary PRIMARY KEY (AuthorRepresentativeId);

ALTER TABLE

AuthorRepresentative ADD CONSTRAINT authorrepresentative\_representativeid\_foreign FOREIGN KEY (RepresentativeId) REFERENCES Representative (RepresentativeId);

ALTER TABLE

AuthorRepresentative ADD CONSTRAINT authorrepresentative\_authorid\_foreign FOREIGN KEY (AuthorId) REFERENCES Author (AuthorId);

* 1. City

Данная справочная таблица создана с целью хранения информации о городах

Листинг:

CREATE TABLE City(

CityId INT identity(1,1) NOT NULL,

Name NVARCHAR(30) NOT NULL

);

ALTER TABLE

City ADD CONSTRAINT city\_cityid\_primary PRIMARY KEY(CityId);

* 1. StatusOrders

Данная справочная таблица создана с целью хранения информации о статусе заказа

Листинг:

CREATE TABLE StatusOrders(

StatusOrderssId INT identity(1,1) NOT NULL,

Name NVARCHAR(30) NOT NULL

);

ALTER TABLE

StatusOrders ADD CONSTRAINT statusOrders\_statusOrderssid\_primary PRIMARY KEY(StatusOrderssId);

* 1. OrdersLines

Данная таблица создана с целью хранения информации о строке заказа

Листинг:

CREATE TABLE OrdersLines(

OrdersLinesId INT identity(1,1) NOT NULL,

OrdersId INT NOT NULL,

PublicationId INT NOT NULL,

PrintingHouseId INT NOT NULL,

Quantity INT NOT NULL,

Price FLOAT NOT NULL

);

ALTER TABLE

OrdersLines ADD CONSTRAINT Orderslines\_Orderslinesid\_primary PRIMARY KEY(OrdersLinesId);

ALTER TABLE

OrdersLines ADD CONSTRAINT Orderslines\_publicationid\_foreign FOREIGN KEY(PublicationId) REFERENCES Publication(PublicationId);

ALTER TABLE

OrdersLines ADD CONSTRAINT Orderslines\_printinghouseid\_foreign FOREIGN KEY(PrintingHouseId) REFERENCES PrintingHouse(PrintingHouseId);

ALTER TABLE

OrdersLines ADD CONSTRAINT Orderslines\_Ordersid\_foreign FOREIGN KEY(OrdersId) REFERENCES Orders(OrdersId);

alter table

OrdersLines ADD CONSTRAINT OrdersLines\_check\_Quantity CHECK (Quantity > 0)

alter table

OrdersLines ADD CONSTRAINT OrdersLines\_check\_Price CHECK (Price > 0)

* 1. Orders

Данная таблица создана с целью хранения информации о закзах

Листинг:

CREATE TABLE Orders(

OrdersId INT identity(1,1) NOT NULL,

DateAdmission DATE NOT NULL,

RepresentativeId INT NOT NULL,

DateCompiction DATE NOT NULL,

StatusOrderssId INT NOT NULL,

StatusDate DATE NOT NULL

);

ALTER TABLE

Orders ADD CONSTRAINT Orders\_Ordersid\_primary PRIMARY KEY(OrdersId);

ALTER TABLE

Orders ADD CONSTRAINT Orders\_representativeid\_foreign FOREIGN KEY(RepresentativeId) REFERENCES Representative(RepresentativeId);

ALTER TABLE

Orders ADD CONSTRAINT Orders\_statusOrderssid\_foreign FOREIGN KEY(StatusOrderssId) REFERENCES StatusOrders(StatusOrderssId);

* 1. PrintingHouse

Данная таблица создана с целью хранения информации о типографиях.

Листинг:

CREATE TABLE PrintingHouse(

PrintingHouseId INT identity(1,1) NOT NULL,

Name NVARCHAR(30) NOT NULL,

CityId int NOT NULL,

Street NVARCHAR(30) NOT NULL,

NumberHouse NVARCHAR(30) NOT NULL

);

ALTER TABLE

PrintingHouse ADD CONSTRAINT printinghouse\_printinghouseid\_primary PRIMARY KEY(PrintingHouseId);

ALTER TABLE

PrintingHouse ADD CONSTRAINT printinghouse\_cityid\_foreign FOREIGN KEY(CityId) REFERENCES City(CityId);

* 1. Representative

Данная таблица создана с целью хранения информации о представителях

Листинг:

CREATE TABLE Representative(

RepresentativeId INT identity(1,1) NOT NULL,

FirstName NVARCHAR(30) NOT NULL,

MiddleName NVARCHAR(30) NOT NULL,

LastName NVARCHAR(30) NULL,

PhoneNumber BIGINT NOT NULL,

fax BIGINT NOT NULL,

IsОrganization BIT NOT NULL

);

ALTER TABLE

Representative ADD CONSTRAINT representative\_representativeid\_primary PRIMARY KEY(RepresentativeId);

alter table

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

alter table

Representative ADD CONSTRAINT Representative\_check\_fax CHECK (DATALENGTH(CAST(fax AS varchar(10))) = 10)

* 1. Оrganization

Данная таблица создана с целью хранения информации о организации.

Листинг:

CREATE TABLE Оrganization(

ОrganizationId INT identity(1,1) NOT NULL,

Name NVARCHAR(30) NOT NULL,

CityId int NOT NULL,

Street NVARCHAR(30) NOT NULL,

NumberHouse NVARCHAR(30) NOT NULL

);

ALTER TABLE

Оrganization ADD CONSTRAINT Оrganization\_Оrganizationid\_primary PRIMARY KEY(ОrganizationId);

ALTER TABLE

Оrganization ADD CONSTRAINT Оrganization\_cityid\_foreign FOREIGN KEY(CityId) REFERENCES City(CityId);

* 1. Author

Данная таблица создана с целью хранения информации об авторах.

Листинг:

CREATE TABLE Author(

AuthorId INT identity(1,1) NOT NULL,

FirstName NVARCHAR(30) NOT NULL,

MiddleName NVARCHAR(30) NOT NULL,

LastName NVARCHAR(30) NULL,

PhoneNumber BIGINT NOT NULL,

CityId INT NOT NULL,

Street NVARCHAR(30) NOT NULL,

NumberHouse NVARCHAR(30) NOT NULL

);

ALTER TABLE

Author ADD CONSTRAINT author\_authorid\_primary PRIMARY KEY(AuthorId);

ALTER TABLE

Author ADD CONSTRAINT author\_cityid\_foreign FOREIGN KEY(CityId) REFERENCES City(CityId);

alter table

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

* 1. ProductType

Данная справочная таблица создана с целью хранения информации о видах продукции.

Листинг:

CREATE TABLE ProductType(

ProductTypeId INT identity(1,1) NOT NULL,

Name NVARCHAR(30) NOT NULL

);

ALTER TABLE

ProductType ADD CONSTRAINT producttype\_producttypeid\_primary PRIMARY KEY(ProductTypeId);

* 1. Publication

Данная таблица создана с целью хранения информации о изданиях.

Листинг:

CREATE TABLE Publication(

PublicationId INT identity(1,1) NOT NULL,

Name NVARCHAR(30) NOT NULL,

VolumeList INT NOT NULL,

ProductTypeId INT NOT NULL

);

ALTER TABLE

Publication ADD CONSTRAINT publication\_publicationid\_primary PRIMARY KEY(PublicationId);

ALTER TABLE

Publication ADD CONSTRAINT publication\_producttypeid\_foreign FOREIGN KEY(ProductTypeId) REFERENCES ProductType(ProductTypeId);

alter table

Publication ADD CONSTRAINT Publication\_check\_Price CHECK (VolumeList > 0)

## Написание запросов, функций и хранимых процедур

### **Хранимые процедуры**

1. AuthorPublicationAction

Описание: процедура создана для вставки, удаления, обновления строк в таблицу AuthorPublication и включает следующие переменные: @ Action (1 – удаление, 2 – добавление, 3 – обновление) и переменные, соответствующие полям в таблице AuthorPublication по названию и типу.

Листинг процедуры:

Create PROCEDURE AuthorPublicationAction (

@Action tinyint,

@AuthorPublicationId INT,

@AuthorId INT,

@PublicationId INT,

@СoАuthor BIT = 0

)

As

Begin

if(@Action = 1)

delete AuthorPublication where AuthorPublicationId = @AuthorPublicationId

else

if(@Action = 2)

insert AuthorPublication (AuthorId,PublicationId ,СoАuthor)

values (@AuthorId,@PublicationId ,@СoАuthor)

else

if(@Action = 3)

UPDATE [dbo].[AuthorPublication]

SET AuthorId = @AuthorId,

PublicationId = @PublicationId,

СoАuthor = @СoАuthor

where AuthorPublicationId = @AuthorPublicationId

end

Пример использования:

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

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

* Добавление: execute AuthorPublicationAction 2,1,1,1,1

При добавлении важно указать первым (@Action) 2, а остальное заполнить нужными значениями.

* Обновление: execute AuthorPublicationAction 3,1,1,1,1

При обновление важно указать первым (@Action) 3 и вторым (@AuthorPublicationId) существующий AuthorPublicationId, а остальное заполнить нужными значениями.

1. ОrganizationRepresentativeAction

Описание: процедура создана для вставки, удаления, обновления строк в таблицу ОrganizationRepresentative и включает следующие переменные: @ Action (1 – удаление, 2 – добавление, 3 – обновление) и переменные, соответствующие полям в таблице ОrganizationRepresentative по названию и типу.

Листинг процедуры:

Create PROCEDURE ОrganizationRepresentativeAction (

@Action tinyint,

@ОrganizationRepresentativeId INT,

@ОrganizationId INT,

@RepresentativeId INT

)

As

Begin

if(@Action = 1)

delete ОrganizationRepresentative where ОrganizationRepresentativeId = @ОrganizationRepresentativeId

else

if(@Action = 2)

insert ОrganizationRepresentative (ОrganizationId,RepresentativeId)

values (@ОrganizationId,@RepresentativeId)

else

if(@Action = 3)

UPDATE [dbo].[ОrganizationRepresentative]

SET ОrganizationId = @ОrganizationId,

RepresentativeId = @RepresentativeId

where ОrganizationRepresentativeId = @ОrganizationRepresentativeId

end

Пример использования:

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

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

* Добавление: execute ОrganizationRepresentativeAction 2,1,1,1

При добавлении важно указать первым (@Action) 2, а остальное заполнить нужными значениями.

* Обновление: execute ОrganizationRepresentativeAction 3,1,1,1

При обновление важно указать первым (@Action) 3 и вторым (@ОrganizationRepresentativeId) существующий ОrganizationRepresentativeId, а остальное заполнить нужными значениями.

1. AuthorRepresentativeAction

Описание: процедура создана для вставки, удаления, обновления строк в таблицу AuthorRepresentative и включает следующие переменные: @ Action (1 – удаление, 2 – добавление, 3 – обновление) и переменные, соответствующие полям в таблице AuthorRepresentative по названию и типу.

Листинг процедуры:

Create PROCEDURE AuthorRepresentativeAction (

@Action tinyint,

@AuthorRepresentativeId INT,

@AuthorId INT,

@RepresentativeId INT

)

As

Begin

if(@Action = 1)

delete AuthorRepresentative where AuthorRepresentativeId = @AuthorRepresentativeId

else

if(@Action = 2)

insert AuthorRepresentative (AuthorId,RepresentativeId)

values (@AuthorId,@RepresentativeId)

else

if(@Action = 3)

UPDATE [dbo].[AuthorRepresentative]

SET AuthorId = @AuthorId,

RepresentativeId = @RepresentativeId

where AuthorRepresentativeId = @AuthorRepresentativeId

end

Пример использования:

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

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

* Добавление: execute AuthorRepresentativeAction 2,1,1,1

При добавлении важно указать первым (@Action) 2, а остальное заполнить нужными значениями.

* Обновление: execute AuthorRepresentativeAction 3,1,1,1

При обновление важно указать первым (@Action) 3 и вторым (@AuthorRepresentativeId) существующий AuthorRepresentativeId, а остальное заполнить нужными значениями.

1. CityAction

Описание: процедура создана для вставки, удаления, обновления строк в таблицу City и включает следующие переменные: @ Action (1 – удаление, 2 – добавление, 3 – обновление) и переменные, соответствующие полям в таблице City по названию и типу.

Листинг процедуры:

Create PROCEDURE CityAction (

@Action tinyint,

@CityId INT,

@Name NVARCHAR(30)

)

As

Begin

if(@Action = 1)

delete City where CityId = @CityId

else

if(@Action = 2)

insert City (Name)

values (@Name)

else

if(@Action = 3)

UPDATE [dbo].[City]

SET Name = @Name

where CityId = @CityId

end

Пример использования:

* Удаление: execute CityAction 1,2,’’

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

* Добавление: execute CityAction 2,1,’’

При добавлении важно указать первым (@Action) 2, а остальное заполнить нужными значениями.

* Обновление: execute CityAction 3,1,’’

При обновление важно указать первым (@Action) 3 и вторым (@CityId) существующий CityId, а остальное заполнить нужными значениями.

1. StatusOrdersAction

Описание: процедура создана для вставки, удаления, обновления строк в таблицу StatusOrders и включает следующие переменные: @ Action (1 – удаление, 2 – добавление, 3 – обновление) и переменные, соответствующие полям в таблице StatusOrders по названию и типу.

Листинг процедуры:

Create PROCEDURE StatusOrdersAction (

@Action tinyint,

@StatusOrderssId INT,

@Name NVARCHAR(30)

)

As

Begin

if(@Action = 1)

delete StatusOrders where StatusOrderssId = @StatusOrderssId

else

if(@Action = 2)

insert StatusOrders (Name)

values (@Name)

else

if(@Action = 3)

UPDATE [dbo].[StatusOrders]

SET Name = @Name

where StatusOrderssId = @StatusOrderssId

end

Пример использования:

* Удаление: execute StatusOrdersAction 1,2,’’

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

* Добавление: execute StatusOrdersAction 2,1,’’

При добавлении важно указать первым (@Action) 2, а остальное заполнить нужными значениями.

* Обновление: execute StatusOrdersAction 3,1,’’

При обновление важно указать первым (@Action) 3 и вторым (@StatusOrdersId) существующий StatusOrdersId, а остальное заполнить нужными значениями.

1. OrdersLinesAction

Описание: процедура создана для вставки, удаления, обновления строк в таблицу StatusOrders и включает следующие переменные: @ Action (1 – удаление, 2 – добавление, 3 – обновление) и переменные, соответствующие полям в таблице OrdersLines по названию и типу.

Листинг процедуры:

Create PROCEDURE OrdersLinesAction (

@Action tinyint,

@OrdersLinesId INT ,

@OrdersId INT ,

@PublicationId INT ,

@PrintingHouseId INT ,

@Quantity INT ,

@Price FLOAT

)

As

Begin

if(@Action = 1)

delete OrdersLines where OrdersLinesId = @OrdersLinesId

else

if(@Action = 2)

insert OrdersLines (OrdersId, PublicationId, PrintingHouseId, Quantity, Price)

values (@OrdersId, @PublicationId, @PrintingHouseId, @Quantity, @Price)

else

if(@Action = 3)

UPDATE [dbo].[OrdersLines]

SET OrdersId = @OrdersId

,PublicationId = @PublicationId

,PrintingHouseId = @PrintingHouseId

,Quantity = @Quantity

,Price = @Price

where OrdersLinesId = @OrdersLinesId

end

Пример использования:

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

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

* Добавление: execute OrdersLinesAction 2,1,1,1,1,1,1

При добавлении важно указать первым (@Action) 2, а остальное заполнить нужными значениями.

* Обновление: execute OrdersLinesAction 3,1,1,1,1,1,1

При обновление важно указать первым (@Action) 3 и вторым (@OrdersLinesId) существующий OrdersLinesId, а остальное заполнить нужными значениями.

1. OrdersAction

Описание: процедура создана для вставки, удаления, обновления строк в таблицу Orders и включает следующие переменные: @ Action (1 – удаление, 2 – добавление, 3 – обновление) и переменные, соответствующие полям в таблице Orders по названию и типу.

Листинг процедуры:

Create PROCEDURE OrdersAction (

@Action tinyint,

@OrdersId INT ,

@DateAdmission DATE ,

@RepresentativeId INT ,

@DateCompiction DATE ,

@ StatusOrderssId INT ,

@StatusDate DATE

)

As

Begin

if(@Action = 1)

delete Orders where OrdersId = @OrdersId

else

if(@Action = 2)

insert Orders (DateAdmission, RepresentativeId, DateCompiction, StatusOrderssId, StatusDate)

values (@DateAdmission, @RepresentativeId, @DateCompiction, @StatusOrderssId, @StatusDate)

else

if(@Action = 3)

UPDATE [dbo].[Orders]

SET DateAdmission = @DateAdmission

,RepresentativeId = @RepresentativeId

,DateCompiction = @DateCompiction

,StatusOrderssId = @StatusOrderssId

,StatusDate = @StatusDate

where OrdersId = @OrdersId

end

Пример использования:

* Удаление: execute OrdersAction 1,2,null,1,null,0,null

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

* Добавление: execute OrdersAction 2,1,’12.12.2012’,1,’24.12.2012’,’13.12.2012’

При добавлении важно указать первым (@Action) 2, а остальное заполнить нужными значениями.

* Обновление: execute OrdersAction 3,1, ’12.12.2012’,1,’24.12.2012’,’13.12.2012’

При обновление важно указать первым (@Action) 3 и вторым (@OrdersId) существующий OrdersId, а остальное заполнить нужными значениями.

1. PrintingHouseAction

Описание: процедура создана для вставки, удаления, обновления строк в таблицу PrintingHouse и включает следующие переменные: @ Action (1 – удаление, 2 – добавление, 3 – обновление) и переменные, соответствующие полям в таблице PrintingHouse по названию и типу.

Листинг процедуры:

Create PROCEDURE PrintingHouseAction (

@Action tinyint,

@PrintingHouseId INT ,

@Name NVARCHAR(30) ,

@CityId int ,

@Street NVARCHAR(30),

@NumberHouse NVARCHAR(30)

)

As

Begin

if(@Action = 1)

delete PrintingHouse where PrintingHouseId = @PrintingHouseId

else

if(@Action = 2)

insert PrintingHouse (Name,CityId,Street,NumberHouse)

values (@Name,@CityId,@Street,@NumberHouse)

else

if(@Action = 3)

UPDATE [dbo].[PrintingHouse]

SET Name = @Name

,CityId = @CityId

,Street = @Street

,NumberHouse = @NumberHouse

where PrintingHouseId = @PrintingHouseId

end

Пример использования:

* Удаление: execute PrintingHouseAction 1,2,null,1,null,0,null

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

* Добавление: execute PrintingHouseAction 2,1, ’12.12.2012’, 1, ’24.12.2012’, ’13.12.2012’

При добавлении важно указать первым (@Action) 2, а остальное заполнить нужными значениями.

* Обновление: execute PrintingHouseAction 3,1, ’12.12.2012’, 1, ’24.12.2012’, ’13.12.2012’

При обновление важно указать первым (@Action) 3 и вторым (@PrintingHouseId) существующий PrintingHouseId, а остальное заполнить нужными значениями.

1. RepresentativeAction

Описание: процедура создана для вставки, удаления, обновления строк в таблицу Representative и включает следующие переменные: @ Action (1 – удаление, 2 – добавление, 3 – обновление) и переменные, соответствующие полям в таблице Representative по названию и типу.

Листинг процедуры:

Create PROCEDURE RepresentativeAction (

@Action tinyint,

@RepresentativeId INT ,

@FirstName NVARCHAR(30),

@MiddleName NVARCHAR(30),

@LastName NVARCHAR(30),

@PhoneNumber BIGINT,

@fax BIGINT,

@IsОrganization BIT

)

As

Begin

if(@Action = 1)

delete Representative where RepresentativeId = @RepresentativeId

else

if(@Action = 2)

insert Representative (FirstName, MiddleName, LastName, PhoneNumber, fax, IsОrganization)

values (@FirstName, @MiddleName, @LastName, @PhoneNumber, @fax, @IsОrganization)

else

if(@Action = 3)

UPDATE [dbo].[Representative]

SET FirstName = @FirstName

,MiddleName = @MiddleName

,LastName = @LastName

,PhoneNumber = @PhoneNumber

,fax = @fax

,IsОrganization = @IsОrganization

where RepresentativeId = @RepresentativeId

end

Пример использования:

* Удаление: execute RepresentativeAction 1,2,null,null,null,null,null,0

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

* Добавление: execute RepresentativeAction 2,1, ‘ds’,’cc’,’vsd’,9324024023,null,0

При добавлении важно указать первым (@Action) 2, а остальное заполнить нужными значениями.

* Обновление: execute RepresentativeAction 3,1, ‘ds’,’cc’,’vsd’,9324024023,null,0

При обновление важно указать первым (@Action) 3 и вторым (@RepresentativeId) существующий RepresentativeId, а остальное заполнить нужными значениями.

1. ОrganizationAction

Описание: процедура создана для вставки, удаления, обновления строк в таблицу Оrganization и включает следующие переменные: @ Action (1 – удаление, 2 – добавление, 3 – обновление) и переменные, соответствующие полям в таблице Оrganization по названию и типу.

Листинг процедуры:

Create PROCEDURE ОrganizationAction (

@Action tinyint,

@ОrganizationId INT ,

@Name NVARCHAR(30),

@CityId int,

@Street NVARCHAR(30),

@NumberHouse NVARCHAR(30)

)

As

Begin

if(@Action = 1)

delete Оrganization where ОrganizationId = @ОrganizationId

else

if(@Action = 2)

insert Оrganization (Name,CityId,Street,NumberHouse)

values (@Name,@CityId,@Street,@NumberHouse)

else

if(@Action = 3)

UPDATE [dbo].[Оrganization]

SET Name = @Name

,CityId = @CityId

,Street = @Street

,NumberHouse = @NumberHouse

where ОrganizationId = @ОrganizationId

end

Пример использования:

* Удаление: execute ОrganizationAction 1,2,null,null,null,null

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

* Добавление: execute ОrganizationAction 2,1, ‘ds’,1,’dsd’,’sda’

При добавлении важно указать первым (@Action) 2, а остальное заполнить нужными значениями.

* Обновление: execute ОrganizationAction 3,1,‘ds’,1,’dsd’,’sda’

При обновление важно указать первым (@Action) 3 и вторым (@ОrganizationId) существующий ОrganizationId, а остальное заполнить нужными значениями.

1. AuthorAction

Описание: процедура создана для вставки, удаления, обновления строк в таблицу Author и включает следующие переменные: @ Action (1 – удаление, 2 – добавление, 3 – обновление) и переменные, соответствующие полям в таблице Author по названию и типу.

Листинг процедуры:

Create PROCEDURE AuthorAction (

@Action tinyint,

@AuthorId INT ,

@FirstName NVARCHAR(30),

@MiddleName NVARCHAR(30),

@LastName NVARCHAR(30),

@PhoneNumber BIGINT,

@CityId INT,

@Street NVARCHAR(30),

@NumberHouse NVARCHAR(30)

)

As

Begin

if(@Action = 1)

delete Author where AuthorId = @AuthorId

else

if(@Action = 2)

insert Author (FirstName, MiddleName, LastName, PhoneNumber, CityId, Street, NumberHouse)

values (@FirstName, @MiddleName, @LastName, @PhoneNumber, @CityId, @Street, @NumberHouse)

else

if(@Action = 3)

UPDATE [dbo].[Author]

SET FirstName = @FirstName

,MiddleName = @MiddleName

,LastName = @LastName

,PhoneNumber = @PhoneNumber

,CityId = @CityId

,Street = @Street

,NumberHouse = @NumberHouse

where AuthorId = @AuthorId

end

Пример использования:

* Удаление: execute AuthorAction 1,2,null,null,null,null,null,null,null

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

* Добавление: execute AuthorAction 2,1, ‘ds’,’dsd’,’sda’,9233323211,1,’ew’,’sf’

При добавлении важно указать первым (@Action) 2, а остальное заполнить нужными значениями.

* Обновление: execute AuthorAction 3,1, ‘ds’,’dsd’,’sda’,9233323211,1,’ew’,’sf’

При обновление важно указать первым (@Action) 3 и вторым (@AuthorId) существующий AuthorId, а остальное заполнить нужными значениями.

1. ProductTypeAction

Описание: процедура создана для вставки, удаления, обновления строк в таблицу ProductType и включает следующие переменные: @ Action (1 – удаление, 2 – добавление, 3 – обновление) и переменные, соответствующие полям в таблице ProductType по названию и типу.

Листинг процедуры:

Create PROCEDURE ProductTypeAction (

@Action tinyint,

@ProductTypeId INT,

@Name NVARCHAR(30)

)

As

Begin

if(@Action = 1)

delete ProductType where ProductTypeId = @ProductTypeId

else

if(@Action = 2)

insert ProductType (Name)

values (@Name)

else

if(@Action = 3)

UPDATE [dbo].[ProductType]

SET Name = @Name

where ProductTypeId = @ProductTypeId

end

Пример использования:

* Удаление: execute ProductTypeAction 1,2,null

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

* Добавление: execute ProductTypeAction 2,1, ‘ds’

При добавлении важно указать первым (@Action) 2, а остальное заполнить нужными значениями.

* Обновление: execute ProductTypeAction 3,1,’dsd’

При обновление важно указать первым (@Action) 3 и вторым (@ProductTypeId) существующий ProductTypeId, а остальное заполнить нужными значениями.

1. PublicationAction

Описание: процедура создана для вставки, удаления, обновления строк в таблицу Publication и включает следующие переменные: @ Action (1 – удаление, 2 – добавление, 3 – обновление) и переменные, соответствующие полям в таблице Publication по названию и типу.

Листинг процедуры:

Create PROCEDURE PublicationAction (

@Action tinyint,

@PublicationId INT,

@Name NVARCHAR(30),

@VolumeList INT ,

@ProductTypeId INT

)

As

Begin

if(@Action = 1)

delete Publication where PublicationId = @PublicationId

else

if(@Action = 2)

insert Publication (Name,VolumeList,ProductTypeId)

values (@Name,@VolumeList,@ProductTypeId)

else

if(@Action = 3)

UPDATE [dbo].[Publication]

SET Name = @Name

,VolumeList= @VolumeList

,ProductTypeId= @ProductTypeId

where PublicationId = @PublicationId

end

Пример использования:

* Удаление: execute PublicationAction 1,2,null,null,null

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

* Добавление: execute PublicationAction 2,1, ‘ds’,123,1

При добавлении важно указать первым (@Action) 2, а остальное заполнить нужными значениями.

* Обновление: execute PublicationAction 3,1,’dsd’,12,1

При обновление важно указать первым (@Action) 3 и вторым (@PublicationId) существующий PublicationId, а остальное заполнить нужными значениями.

1. RepresentativeInfo

Описание: процедура создана для выводы информации по представителю. Включает следующую переменную: @Representative соответствующую полю RepresentativeId в таблице Representative

Листинг:

create PROCEDURE RepresentativeInfo (

@Representative int

)

as

begin

select r.\*, CountOrders from Representative r

inner join (select RepresentativeId, count(OrdersId) as CountOrders

from Orders

where RepresentativeId = @Representative group by RepresentativeId) o on r.RepresentativeId = o.RepresentativeId

end

Выходные данные: все поля таблицы Representative, CountOrders

1. OrdersInfo

Описание: процедура создана для выводы информации по заказу. Включает следующую переменную: @Orders соответствующую полю OrdersId в таблице Orders

Листинг:

create PROCEDURE OrdersInfo (

@Orders int

)

as

begin

select o.\*, CountOrdersLines, SumPrice, tp.Name from Orders o

inner join (select OrdersId, count(OrdersLinesId) as CountOrdersLines, sum(Price) as SumPrice

from OrdersLines

where OrdersId = @Orders group by OrdersId) oq on o.RepresentativeId = o.RepresentativeId

inner join OrdersLines ol on ol.OrdersId=o.OrdersId

inner join Publication p on p.PublicationId = ol.PublicationId

inner join ProductType tp on tp.ProductTypeId = p.ProductTypeId

end

execute OrdersInfo 2

Выходные данные: все поля таблицы Orders, CountOrdersLines, SumPrice, Name

1. PublicationInfo

Описание: процедура создана для выводы информации по заказу. Включает следующую переменную: @Publication соответствующую полю PublicationId в таблице Publication

Листинг:

create PROCEDURE PublicationInfo (

@Publication int

)

as

begin

select p.\*, MiddleName ,OrdersId, Quantity from Publication p

inner join OrdersLines ol on ol.PublicationId=p.PublicationId

inner join AuthorPublication ap on ap.PublicationId = p.PublicationId

inner join Author a on a.AuthorId = ap.AuthorId

where p.PublicationId = @Publication

end

execute OrdersInfo 2

Выходные данные: все поля таблицы Publication, MiddleName ,OrdersId

### **Тригеры**

1. checkInsertOreders

Тригер создан как ограничение таблицы Oreders на вставку и обновление полей DateAdmission и DateCompiction с условием: дата принятия заказа не может больше даты выполнения заказа.

Листинг:

create trigger checkInsertOreders on Orders

for insert, update

as

declare @DateAdmission date, @DateCompiction date

Select

@DateAdmission=DateAdmission,

@DateCompiction=DateCompiction

from inserted

if (@DateCompiction >= @DateAdmission)

Begin

print 'Необходимо правильно указать дату!'

ROLLBACK TRANSACTION

End

# Заключение

В данной курсовой работе был проведен детальный анализ предметной области издательства. 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 издательства.

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

Использование БД в реальной ассоциации позволит автоматизировать деятельность сотрудников издательства. Создаст благоприятные условия труда для сотрудников, позволит всегда отражать актуальную информацию.

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

# Список используемой литературы

1. <https://ru.wikipedia.org/wiki/%D0%98%D0%B7%D0%B4%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D1%82%D0%B2%D0%BE>
2. <https://docs.microsoft.com/ru-ru/sql/sql-server/?view=sql-server-ver15>
3. <https://metanit.com/sql/sqlserver/>
4. <https://www.cossa.ru/152/192645/>