МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
ИНЖЕНЕРНО-ТЕХНОЛОГИЧЕСКАЯ АКАДЕМИЯ

Институт компьютерных технологий и информационной безопасности

Кафедра систем автоматизированного проектирования

Отчет по лабораторной работе № 1
Вариант №9

на тему: «Проектирование логической и физической модели данных в ERWin Data Modeler»
«Библиотека»

по курсу «Базы данных и СУБД»

Выполнили:
студенты гр. КТбо2-5
Козенко Н. В.
Посредников Н.Д.
_____

Проверил:
Ассистент кафедры САПР
Корнилов В. С._____

Таганрог 2024

СОДЕРЖАНИЕ

1 ПОСТАНОВКА ЗАДАЧИ 3
1.1 Цель работы 3
1.2 Задача 3
2 ПРЕДМЕТНАЯ ОБЛАСТЬ 5
2.1 Описание предметной области 5
2.2 Сущности и атрибуты 5
3 ФИЗИЧЕСКАЯ МОДЕЛЬ 8
3.1 Создание физической модели и генерации DDL-скрипта 8
ВЫВОД 19
СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 20

1 ПОСТАНОВКА ЗАДАЧИ

1.1 Цель работы

Изучение методов проектирования модели данных и освоение современных CASE-средств, предназначенных для построения модели данных.

Объектами изучения являются:

− модель структуры данных информационной системы «сущность-связь»;

− методология проектирования модели данных IDEF1x;

− инструментальное CASE-средство ERWin.

При домашней подготовке к работе следует:

− изучить принципы построения моделей типа «сущность-связь»;

− ознакомиться с методологий IDEF1x, используемой для построения моделей данных;

− исследовать предметную область, заданную вариантом задания и построить ее словесное описание.

При выполнении лабораторной работы следует:

− ознакомиться с программным CASE-средством ERWin, реализующим методологию проектирования IDEF1x;

− произвести процесс «прямого» проектирования – построить модель данных по подготовленному описанию предметной области;

− произвести создание базы данных по спроектированной модели.

1.2 Задача

В БД содержатся сведения об абонентах библиотеки, имеющейся литературе и ее хранении. Необходимо вести учет выданной литературы. В каталоге на каждое название книги выписывается каталожная карточка со всеми реквизитами. Для удобства создания БД составляются таблицы наличия книг (по пунктам каталожной карточки), причем обязательно указывается

количество экземпляров. На каждый экземпляр книги выписывается формуляр с основными реквизитами, записями о выдаче читателю и возврате. На каждого читателя заполняется абонементная карточка с перечнем полученных книг и ставится дата их возврата.

2 ПРЕДМЕТНАЯ ОБЛАСТЬ

1.2 Описание предметной области

Мы создаем модель для БД «библиотека». В ней будет 4-е сущности, имеющие разные типы связи. Предметной областью информационной системы является:

Учет книг и экземпляров:

- Система должна отслеживать информацию о каждой книге в библиотеке, включая название, автора, год издания и жанр.
- Для каждой книги должны быть записаны все экземпляры, их статус (доступен, выдан, возвращен) и дата выдачи/возврата.

Регистрация читателей:

- Система должна позволять регистрировать новых читателей, включая их ФИО, дату рождения и адрес проживания.

Выдача книг читателям:

- Читатели могут брать книги на выдачу.
- Система должна регистрировать дату выдачи и дату возврата для каждой выданной книги.

2.2 Сущности и атрибуты

На рисунке 1 мы создали логическую модель, в которой есть такие сущности:

Книги:

ИД_книги: Уникальный идентификатор книги (ключевое поле).

- Название: Название книги.
- Автор: ФИО автора книги.
- Год издания: Год, когда книга была издана.
- Жанр: категория книги.

Экземпляры книг:

- ИД_экземпляра: Уникальный идентификатор экземпляра книги (ключевое поле).
- ИД_книги: Ссылка на соответствующую книгу.
- Статус: Состояние экземпляра (например, доступен, выдан, возвращен).

Читатели:

- ИД_читателя: Уникальный идентификатор читателя (ключевое поле).
- ФИО: Фамилия, имя и отчество читателя.
- Дата рождения: Дата рождения читателя.

Выдача книг:

- ИД_выдачи: Уникальный идентификатор операции выдачи (ключевое поле).
- ИД_экземпляра: Ссылка на экземпляр книги.
- ИД_читателя: Ссылка на читателя.
- Дата выдачи: Дата, когда книга была выдана читателю.
- Дата возврата: Дата возврата книги.

1. **Книги**:
    - Связь с **Экземплярами книг** (тип: **один ко многим**):
        - Один экземпляр книги может иметь несколько экземпляров (например, разные физические копии) в библиотеке.
        - Экземпляры книг связаны с книгой по **ИД_книги**.
2. **Экземпляры книг**:
    - Связь с **Книгами** (тип: **многие к одному**):
        - Каждый экземпляр книги связан с одной конкретной книгой.
        - Экземпляры книг связаны с книгой по **ИД_книги.**
    - Связь с **Выдачей книг** (тип: **многие ко многим**):
        - Несколько экземпляров книг могут быть выданы разным читателям.
        - Экземпляры книг связаны с выдачей по **ИД_экземпляра**.
        - Выдача книг связана с экземпляром по **ИД_экземпляра**.
3. **Читатели**:
    - Связь с **Выдачей книг** (тип: **многие ко многим**):

- Несколько читателей могут брать одну и ту же книгу.
- Читатели связаны с выдачей по **ИД_читателя**.
- Выдача книг связана с читателем по **ИД_читателя**.

4. **Выдача книг**:
   - Связь с **Экземплярами книг** (тип: **многие ко многим**):
     - Несколько экземпляров книг могут быть выданы разным читателям.
     - Выдача книг связана с экземпляром по **ИД_экземпляра**.
   - Связь с **Читателями** (тип: **многие ко многим**):
     - Несколько читателей могут брать одну и ту же книгу.
     - Выдача книг связана с читателем по **ИД_читателя**.

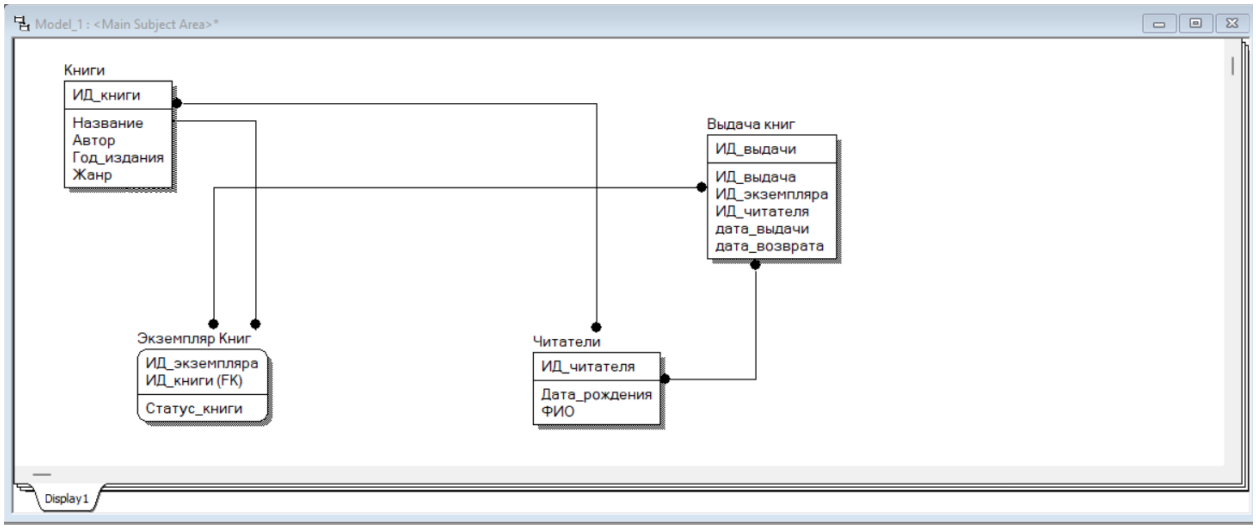

Рисунок 1 – Логическая модель

3 ФИЗИЧЕСКАЯ МОДЕЛЬ

3.1Создание физической модели и генерации DDL-скрипта

Далее мы перешли в раздел физическая модель (рисунок 2).

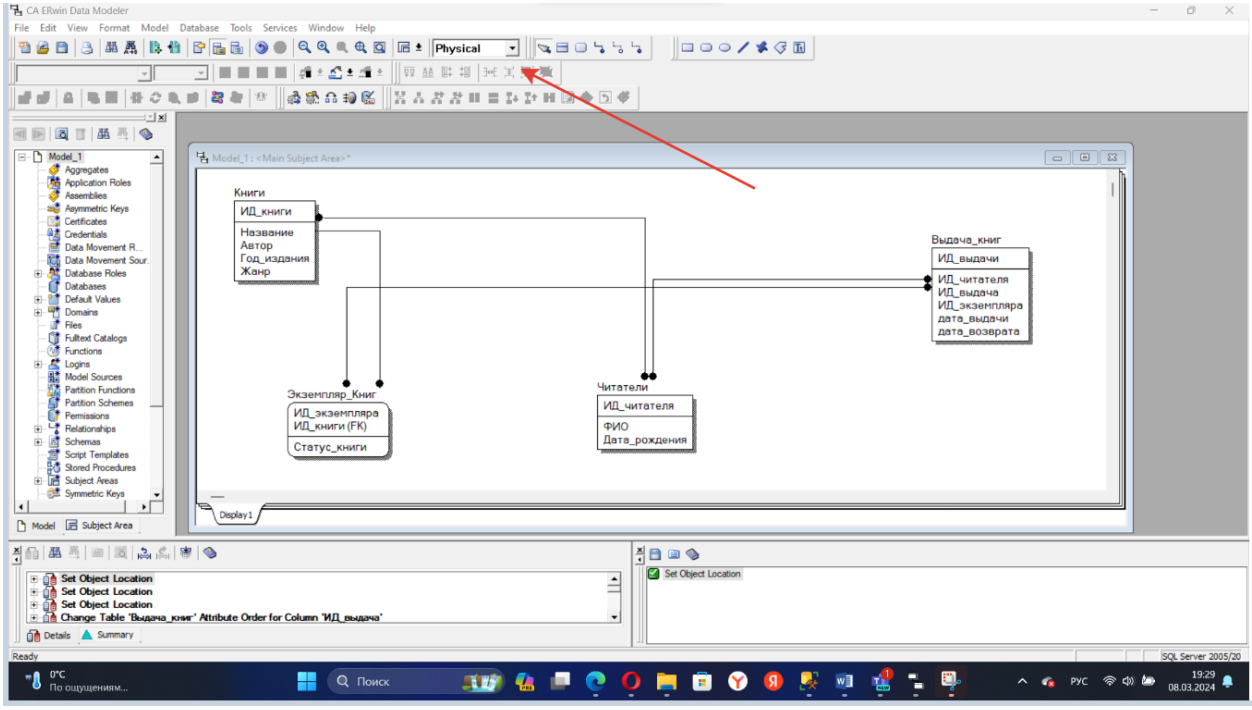

Рисунок 2 – Физическая модель

Где необходимо изменили типы (рисунок 3). Например имя автора расширили с 20 символов до 70 (рисунок 4).

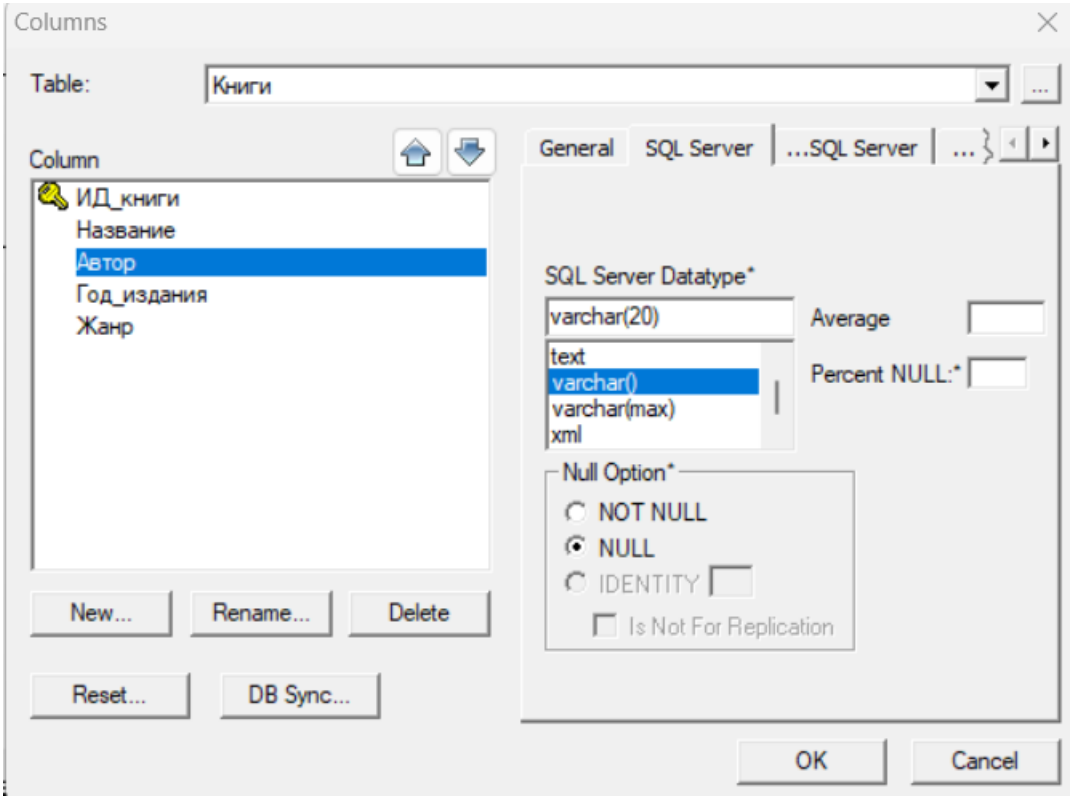

Рисунок 3 – Изменение типов

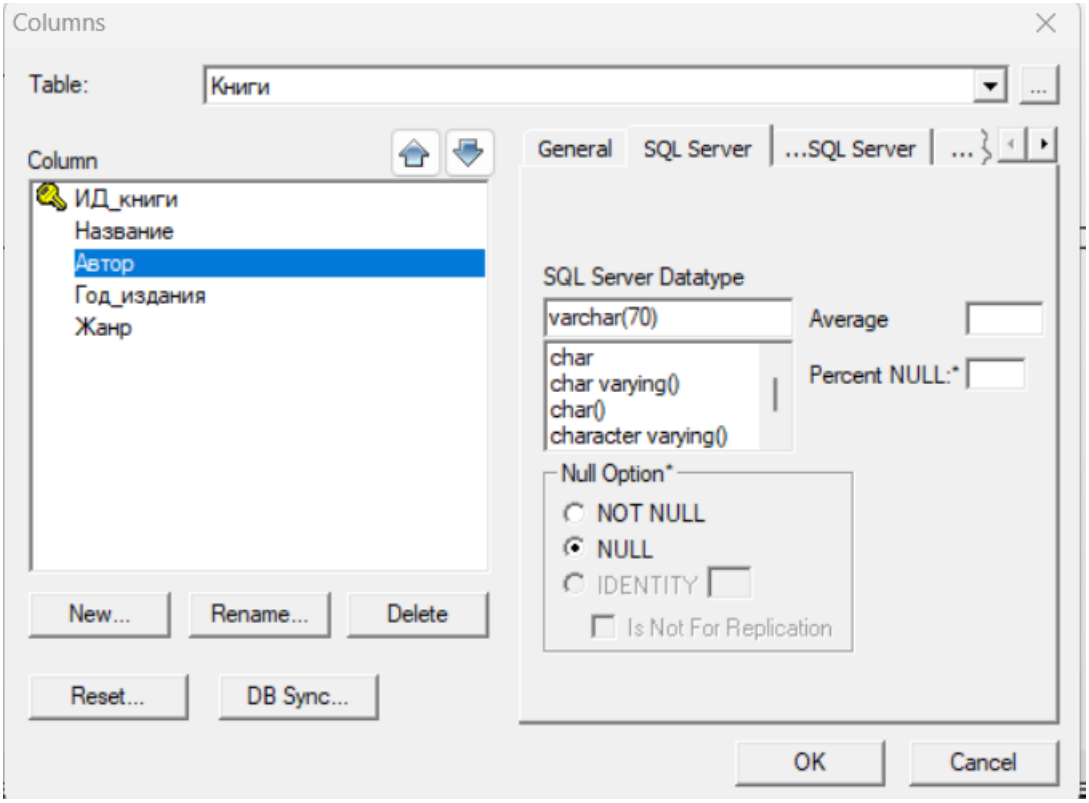

Рисунок 4 – Увеличение varchar

А затем мы сгенерировали DDL-скрипт (рисунки 5-6). И на этом наша работа завершилась.

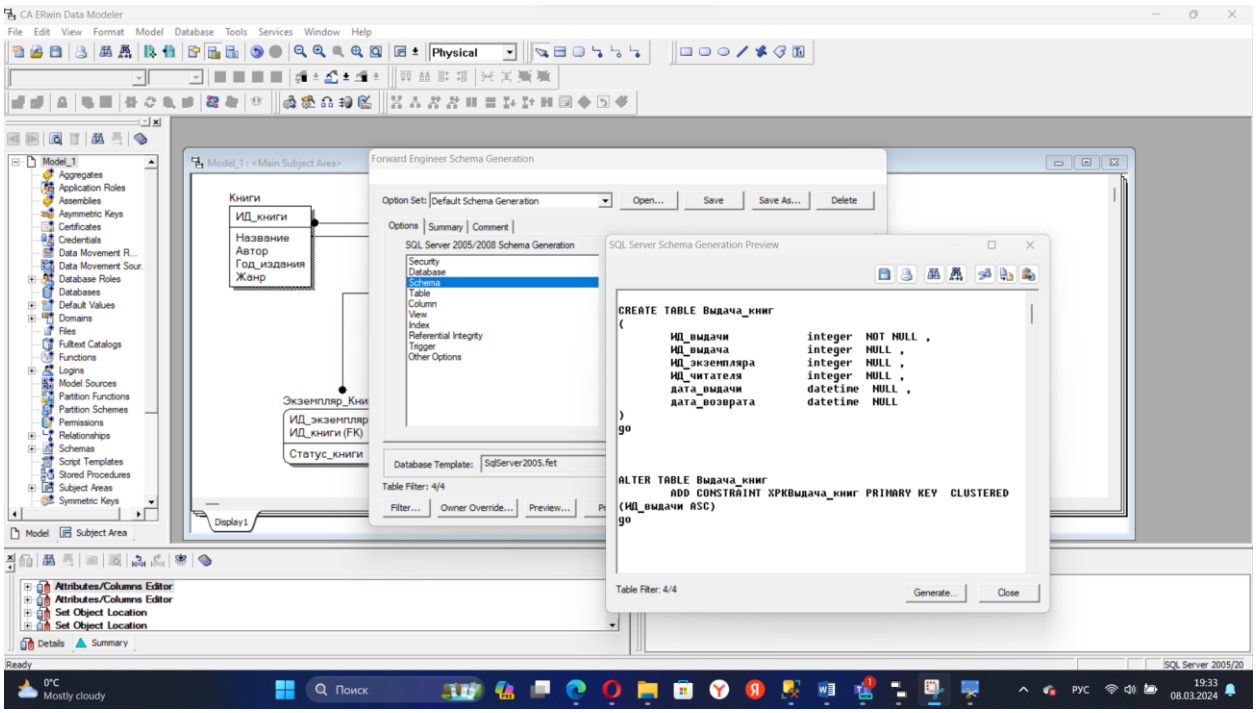

Рисунок 5 - Скрипт

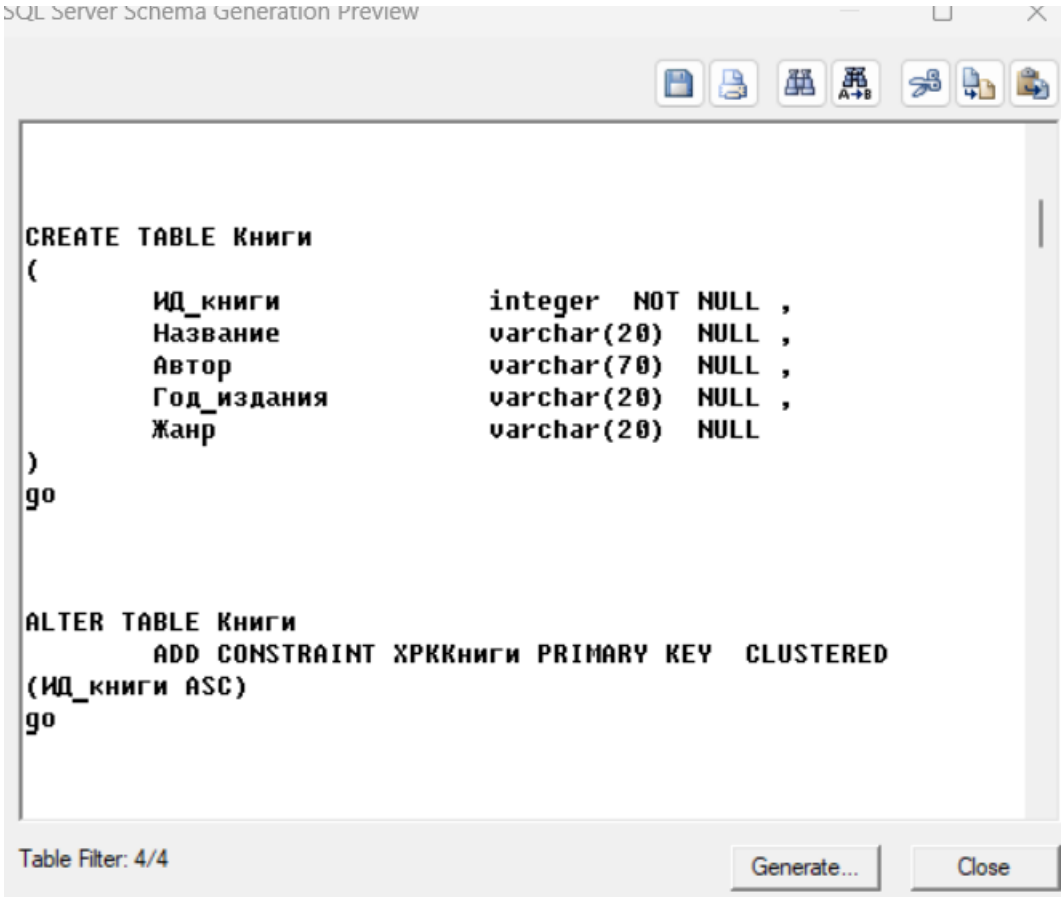

Рисунок 6 - Скрипт

Полный пример скрипта:

```
CREATE TABLE Выдача_книг
(
    ИД_выдачи          integer  NOT NULL ,
    ИД_выдача          integer  NULL ,
    ИД_экземпляра      integer  NULL ,
    ИД_читателя        integer  NULL ,
    дата_выдачи        datetime  NULL ,
    дата_возврата      datetime  NULL
)
go
ALTER TABLE Выдача_книг
    ADD CONSTRAINT XPKВыдача_книг PRIMARY KEY  CLUSTERED (ИД_выдачи ASC)
go
CREATE TABLE Книги
(
    ИД_книги           integer  NOT NULL ,
    Название           varchar(20)  NULL ,
    Автор              varchar(70)  NULL ,
    Год_издания        varchar(20)  NULL ,
    Жанр               varchar(20)  NULL
)
go
ALTER TABLE Книги
    ADD CONSTRAINT XPKКниги PRIMARY KEY  CLUSTERED (ИД_книги ASC)
go
```

```
CREATE TABLE Читатели
(
	ИД_читателя		integer  NOT NULL ,
	ФИО			varchar(20)  NULL ,
	Дата_рождения		datetime  NULL
)
go
ALTER TABLE Читатели
	ADD CONSTRAINT XPKЧитатели PRIMARY KEY  CLUSTERED (ИД_читателя ASC)
go
CREATE TABLE Экземпляр_Книг
(
	ИД_экземпляра		integer  NOT NULL ,
	ИД_книги		integer  NOT NULL ,
	Статус_книги		varchar(20)  NULL
)
go
ALTER TABLE Экземпляр_Книг
	ADD CONSTRAINT XPKЭкземпляр_Книг PRIMARY KEY CLUSTERED (ИД_экземпляра ASC,ИД_книги ASC)
go
ALTER TABLE Экземпляр_Книг
	ADD CONSTRAINT R_3 FOREIGN KEY (ИД_книги) REFERENCES Книги(ИД_книги)
		ON DELETE NO ACTION
		ON UPDATE NO ACTION
go
CREATE TRIGGER tD_Книги ON Книги FOR DELETE AS
/* ERwin Builtin Trigger */
```

```
/* DELETE trigger on Книги */
BEGIN
  DECLARE  @errno   int,
           @errmsg  varchar(255)
    /* ERwin Builtin Trigger */
    /* Книги  Экземпляр_Книг on parent delete no action */
    /*    ERWIN_RELATION:CHECKSUM="0000f168",    PARENT_OWNER="",
PARENT_TABLE="Книги"
    CHILD_OWNER="", CHILD_TABLE="Экземпляр_Книг"
    P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
    FK_CONSTRAINT="R_3", FK_COLUMNS="ИД_книги" */
    IF EXISTS (
      SELECT * FROM deleted,Экземпляр_Книг
      WHERE
        /*  %JoinFKPK(Экземпляр_Книг,deleted," = "," AND") */
        Экземпляр_Книг.ИД_книги = deleted.ИД_книги
    )
    BEGIN
      SELECT @errno  = 30001,
             @errmsg = 'Cannot delete Книги because Экземпляр_Книг exists.'
      GOTO ERROR
    END
    /* ERwin Builtin Trigger */
    RETURN
ERROR:
    raiserror @errno @errmsg
    rollback transaction
END
go
```

```
CREATE TRIGGER tU_Книги ON Книги FOR UPDATE AS
/* ERwin Builtin Trigger */
/* UPDATE trigger on Книги */
BEGIN
  DECLARE  @NUMROWS int,
           @nullcnt int,
           @validcnt int,
           @insИД_книги integer,
           @errno   int,
           @errmsg  varchar(255)

  SELECT @NUMROWS = @@rowcount
  /* ERwin Builtin Trigger */
  /* Книги  Экземпляр_Книг on parent update no action */
  /*    ERWIN_RELATION:CHECKSUM="000117c2",    PARENT_OWNER="",
PARENT_TABLE="Книги"
    CHILD_OWNER="", CHILD_TABLE="Экземпляр_Книг"
    P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
    FK_CONSTRAINT="R_3", FK_COLUMNS="ИД_книги" */
  IF
    /* %ParentPK(" OR",UPDATE) */
    UPDATE(ИД_книги)
  BEGIN
    IF EXISTS (
      SELECT * FROM deleted,Экземпляр_Книг
      WHERE
        /*  %JoinFKPK(Экземпляр_Книг,deleted," = "," AND") */
        Экземпляр_Книг.ИД_книги = deleted.ИД_книги
    )
```

```
    BEGIN
      SELECT @errno  = 30005,
            @errmsg = 'Cannot update Книги because Экземпляр_Книг exists.'
      GOTO ERROR
    END
  END


  /* ERwin Builtin Trigger */
  RETURN
ERROR:
    raiserror @errno @errmsg
    rollback transaction
END
go
CREATE TRIGGER tD_Экземпляр_Книг ON Экземпляр_Книг FOR DELETE
AS
/* ERwin Builtin Trigger */
/* DELETE trigger on Экземпляр_Книг */
BEGIN
  DECLARE  @errno   int,
           @errmsg  varchar(255)
    /* ERwin Builtin Trigger */
    /* Книги  Экземпляр_Книг on child delete no action */
    /*   ERWIN_RELATION:CHECKSUM="00013516",   PARENT_OWNER="",
PARENT_TABLE="Книги"
    CHILD_OWNER="", CHILD_TABLE="Экземпляр_Книг"
    P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
    FK_CONSTRAINT="R_3", FK_COLUMNS="ИД_книги" */
```

```
    IF EXISTS (SELECT * FROM deleted,Книги
      WHERE
        /* %JoinFKPK(deleted,Книги," = "," AND") */
        deleted.ИД_книги = Книги.ИД_книги AND
        NOT EXISTS (
          SELECT * FROM Экземпляр_Книг
          WHERE
            /* %JoinFKPK(Экземпляр_Книг,Книги," = "," AND") */
            Экземпляр_Книг.ИД_книги = Книги.ИД_книги
        )
    )
    BEGIN
      SELECT @errno  = 30010,
             @errmsg = 'Cannot delete last Экземпляр_Книг because Книги exists.'
      GOTO ERROR
    END
    /* ERwin Builtin Trigger */
    RETURN
ERROR:
    raiserror @errno @errmsg
    rollback transaction
END
go
CREATE TRIGGER tU_Экземпляр_Книг ON Экземпляр_Книг FOR UPDATE
AS
/* ERwin Builtin Trigger */
/* UPDATE trigger on Экземпляр_Книг */
BEGIN
  DECLARE  @NUMROWS int,
```

```
        @nullcnt int,
        @validcnt int,
        @insИД_экземпляра integer,
        @insИД_книги integer,
        @errno   int,
        @errmsg  varchar(255)

  SELECT @NUMROWS = @@rowcount
  /* ERwin Builtin Trigger */
  /* Книги  Экземпляр_Книг on child update no action */
  /*    ERWIN_RELATION:CHECKSUM="00014311",    PARENT_OWNER="",
PARENT_TABLE="Книги"
    CHILD_OWNER="", CHILD_TABLE="Экземпляр_Книг"
    P2C_VERB_PHRASE="", C2P_VERB_PHRASE="",
    FK_CONSTRAINT="R_3", FK_COLUMNS="ИД_книги" */
  IF
    /* %ChildFK(" OR",UPDATE) */
    UPDATE(ИД_книги)
  BEGIN
    SELECT @nullcnt = 0
    SELECT @validcnt = count(*)
      FROM inserted,Книги
        WHERE
          /* %JoinFKPK(inserted,Книги) */
          inserted.ИД_книги = Книги.ИД_книги
    /* %NotnullFK(inserted," IS NULL","select @nullcnt = count(*) from inserted
where"," AND") */
    IF @validcnt + @nullcnt != @NUMROWS
    BEGIN
```

```
    SELECT @errno  = 30007,
        @errmsg = 'Cannot update Экземпляр_Книг because Книги does not exist.'
    GOTO ERROR
  END
 END
 /* ERwin Builtin Trigger */
 RETURN
ERROR:
  raiserror @errno @errmsg
  rollback transaction
END
go
```

ВЫВОД

При выполнении данной лабораторной работы были изучены методы проектирования модели данных и освоение современных CASE-средств, предназначенных для построения модели данных.

Также на основе проведенного анализа было осуществлено проектирование информационной модели с использованием инструментальных средств CASE-технологий по методологии IDEF1x. Итогом данной работы стала модель данных, то есть предметная область.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Проектирование информационных систем / http://www.plam.ru/compinet/bpwin_i_erwin_case_sredstva_dlja_razrabotki_informacionnyh_sistem/p5.php / [Электронный ресурс] / [Режим доступа: свободный] / (Дата обращения: 08.03.2024).