Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии (ИУ-7)

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОМУ ПРОЕКТУ

НА ТЕМУ:

Проектирование базы данных издательской компании

Студент ИУ7-78 _____ А.В. Иванников
(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель курсового проекта _____ А.А. Павлюк
(Подпись, дата) (И.О.Фамилия)

Консультант _____ А.А. Павлюк
(Подпись, дата) (И.О.Фамилия)

2020 г.

Содержание

Введение

Данный проект посвящен проектированию базы данных издательской компании.

В аналитическом разделе курсового проекта будет выполнен анализ предметной области решаемой задачи и концептуальное проектирование базы данных.

В конструкторском разделе курсового проекта будет выполнено логическое проектирование базы данных и составление реляционных отношений.

В технологическом разделе курсовой работы будет выполнен выбор инструментов, технологий и средств разработки.

В заключении приводятся общие выводы о проделанной работе.

В приложении приведены листинги программного кода приложений – генераторов данных и SQL запросов

1. Аналитический раздел

1.1. Анализ предметной области решаемой задачи

База данных создаётся для информационного обслуживания редакторов, менеджеров и других сотрудников издательской компании. База данных должна содержать данные о сотрудниках компании, книгах, авторах, финансовом состоянии компании и предоставлять возможность получать разнообразные отчёты. В соответствии с предметной областью система строится с учётом следующих особенностей:

– каждая книга издаётся в рамках контракта;

– книга может быть написана несколькими авторами;

– контракт подписывается одним менеджером и всеми авторами книги;

– каждый автор может написать несколько книг (по разным контрактам);

– порядок, в котором авторы указаны на обложке, влияет на размер гонорара;

– если сотрудник является редактором, то он может работать одновременно над несколькими книгами;

– у каждой книги может быть несколько редакторов, один из них – ответственный редактор;

– каждый заказ оформляется на одного заказчика;

– в заказе на покупку может быть перечислено несколько книг.

1.2 Концептуальное проектирование базы данных

Были выделены следующие базовые сущности:

– Сотрудники компании (Employees). Атрибуты: ФИО, табельный номер, пол, дата рождения, паспортные данные, ИНН, должность, оклад, телефон. Для

редакторов необходимо хранить сведения о редактируемых книгах; для менеджеров – сведения о подписанных контрактах.

– Авторы (Authors). Атрибуты: ФИО.

– Книги (Books). Атрибуты: авторы, название, тираж, дата выхода, цена одного экземпляра, общие затраты на издание, авторский гонорар.

– Контракты (Orders). Является связью между авторами, книгами и менеджерами. Атрибуты: номер, дата подписания и участники.

Для отражения финансового положения компании в системе нужно учитывать заказы на книги. Для заказа необходимо хранить номер заказа, ФИО заказчика, адрес заказчика, дату поступления заказа, дату его выполнения, список заказанных книг с указанием количества экземпляров. ER–диаграмма издательской компании приведена на рисунке ниже (базовые сущности на рисунках выделены полужирным шрифтом).

ER–диаграмма издательской компании приведена на рисунке 1.

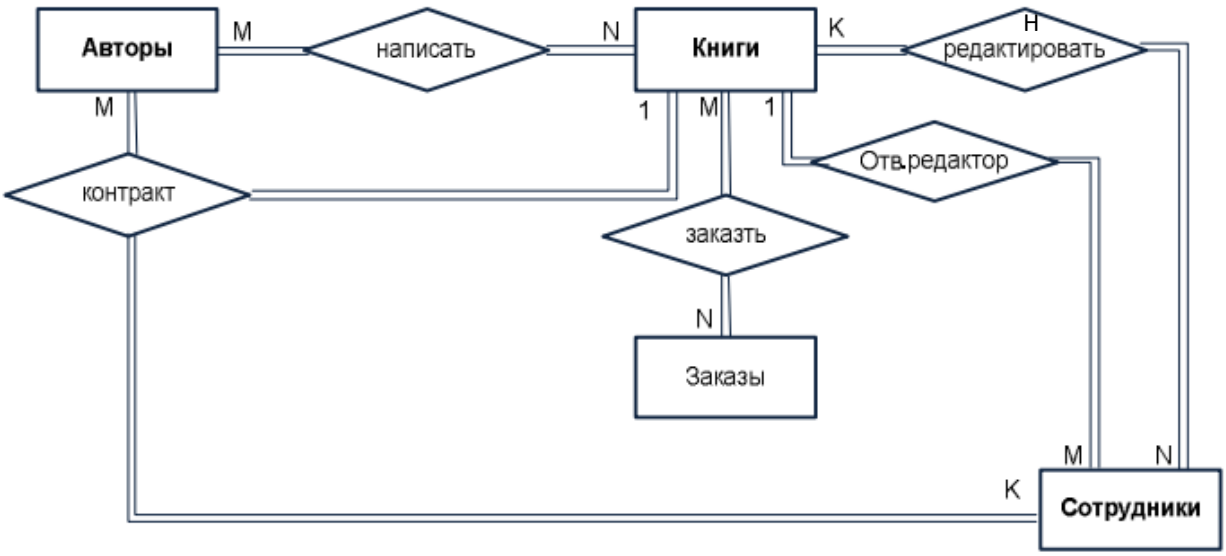

Рисунок 1 – ER-диаграмма издательской компании

2. Конструкторский раздел

2.1 Логическое проектирование базы данных

Для преобразования ER–диаграммы в схему базы данных приведём уточнённую ER– диаграмму издательской компании, содержащая атрибуты сущностей (рисунок 2).

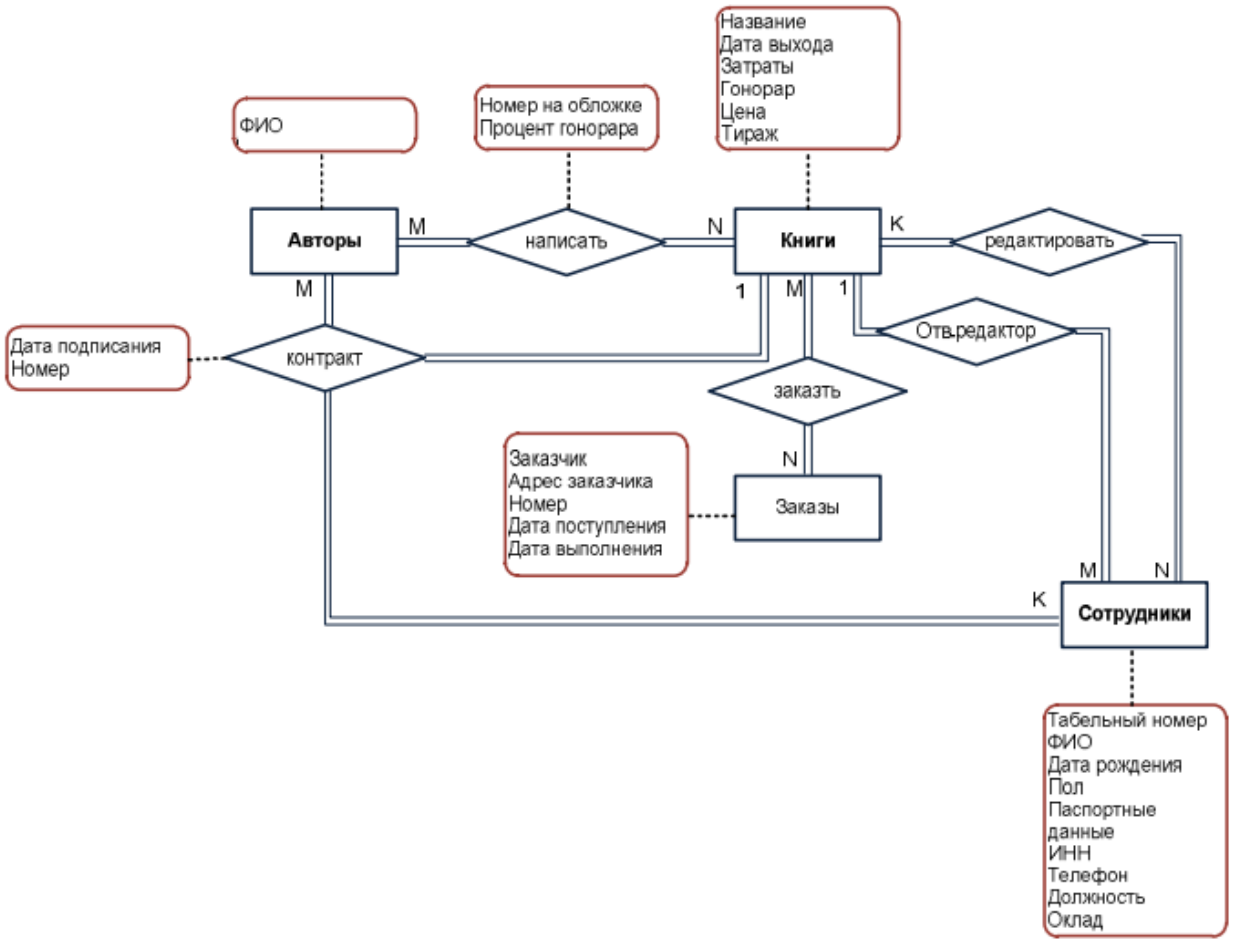

Рисунок 2 – Уточненная ER-диаграмма издательской компании

Преобразование ER–диаграммы в схему базы данных выполняется путем сопоставления каждой сущности и каждой связи, имеющей атрибуты,

отношения (таблицы базы данных) с помощью обозначений, представленных на рисунке 3.

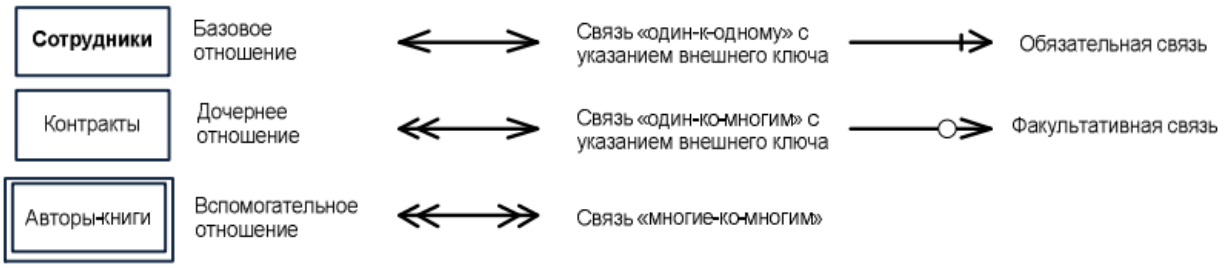

Рисунок 3 – Обозначения, используемые для преобразования ER–диаграммы в схему базы данных

Полученная схема реляционной базы данных приведена на рисунке 4.

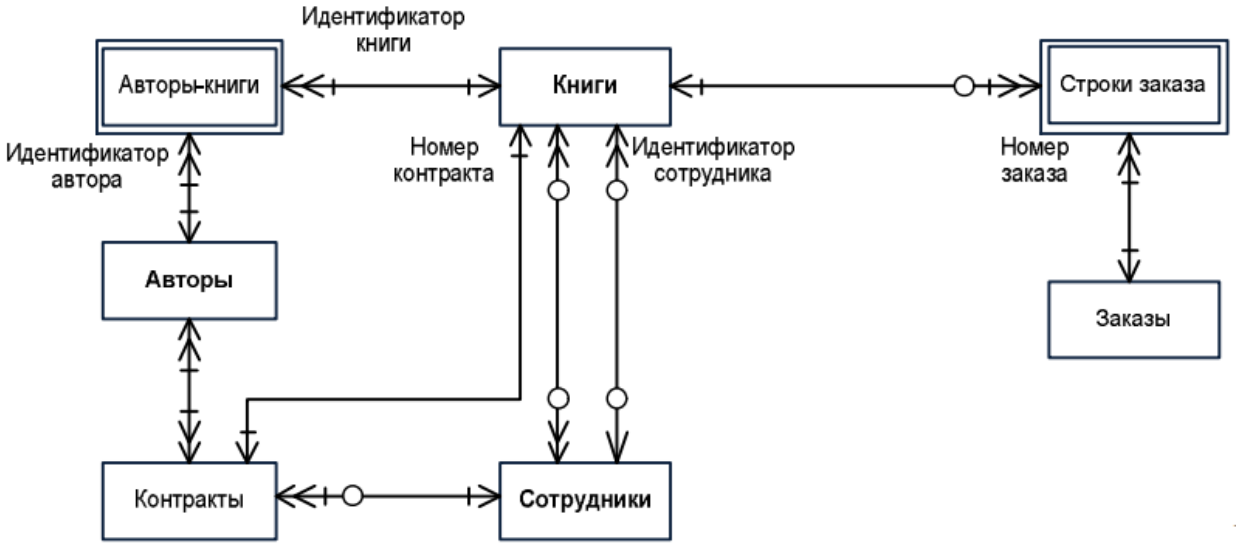

Рисунок 4 – Схема реляционной базы данных, издательской компании полученная из ER–диаграммы

На схеме (рисунок 4) есть связь типа 1:1 – обязательная связь между Книгами и Контрактами. Такие отношения следует объединять в одно. Дополнительный эффект от объединения этих отношений – слияние связей Авторы – Контракты и Авторы – Книги, так как контракт заключается именно для написания книги.

Связь типа 1:n (один-ко-многим) между отношениями реализуется через внешний ключ. Ключ вводится для того отношения, к которому осуществляется множественная связь (Книги).

Связь между отношениями Книги и Сотрудники принадлежит к типу n:m (многие-ко-многим). Этот тип связи реализуется через вспомогательное отношение, которое является соединением первичных ключей соответствующих отношений.

Бинарная связь между отношениями не может быть обязательной для обоих отношений. После объединения сущностей Книги и Контракты остаётся три связи, обязательные для всех участников: между Авторами и Книгами и между Заказами и Строками. Такой тип связи означает, что, например, прежде чем добавить новый заказ в отношение Заказы, необходимо добавить новую строку в отношение Строки заказа и наоборот. Поэтому для такой связи необходимо снять с одной стороны условие обязательности. Так как все эти связи будут реализованы с помощью внешнего ключа, условие обязательности связей для отношений, содержащих первичные ключи снимается. Уточнённая схема реляционной БД издательской компании приведена на рисунке 5.

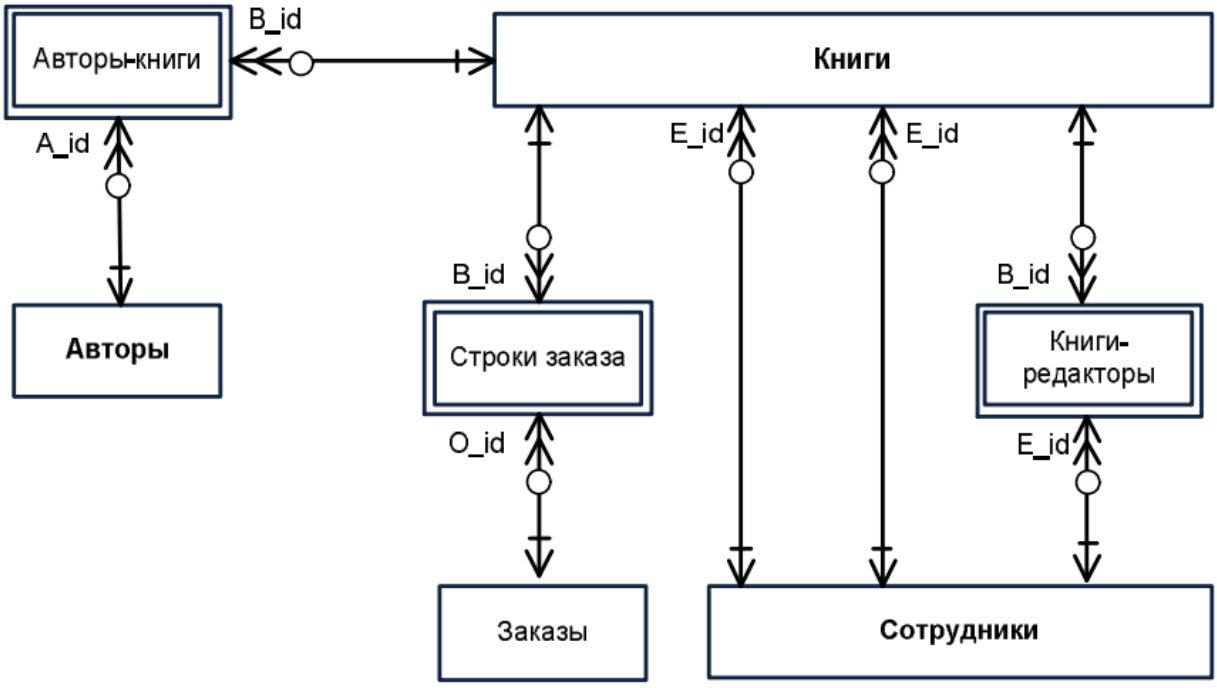

Рисунок 5 – Уточнённая схема реляционной БД издательской компании

2.2 . Составление реляционных отношений

Каждое реляционное отношение соответствует одной сущности (объекту предметной области), и в него вносятся все атрибуты сущности. Для каждого отношения необходимо определить первичный ключ и внешние ключи (если они есть). В том случае, если базовое отношение не имеет потенциальных ключей, вводится суррогатный первичный ключ, который не несёт смысловой нагрузки и служит только для идентификации записей.

Потенциальным ключем отношения Авторы является ФИО. Но так как ФИО хранится как длинная строка, то для Авторов необходимо ввести суррогатный ключ A_id.

Книги можно идентифицировать по атрибуту контракт: его номер обязателен и уникален. Потенциальные ключи отношения Сотрудники: ИНН, Паспортные данные, Табельный номер, причём все они

обязательные. Табельный номер занимает меньше памяти, чем ИНН, поэтому он и будет первичным ключом.

Кортежи отношения Заказы можно идентифицировать ключом Номер заказа. Потенциальными ключами вспомогательных отношений являются комбинации первичных ключей соответствующих базовых отношений. Отношения приведены в таблицах 1 - 7. Для каждого отношения указаны атрибуты с их внутренним названием, типом и длиной.

Таблица 1 – Схема отношения Сотрудники (Employees)

Содержание поля	Имя поля	Тип, длина	Примечания
Табельный номер	E_ID	INT	первичный ключ
Фамилия	E_FAMILY	VARCHAR(50)	обязательное поле
Имя	E_NAME	VARCHAR(50)	обязательное поле
Отчество	E_SURNAME	VARCHAR(50)	обязательное поле
Дата рождения	E_BORN	DATE	обязательное поле
Паспортные данные	E_PASSP	VARCHAR(50)	обязательное поле
ИНН	E_INN	N(12)	обязательное поле
Должность	E_POST	VARCHAR(50)	обязательное поле
Оклад	E_SALARY	DECIMAL(8,2)	обязательное поле
Телефон	E_TEL	VARCHAR(50)	многозначное поле

Таблица 2 – Схема отношения Книги (Books)

Содержание поля	Имя поля	Тип, длина	Примечания
Номер контракта	B_CONTRACT	INT	первичный ключ
Дата подписания контракта	B_DATE	DATE	обязательное поле
Менеджер	B_MAN	INT	внешний ключ (к Employees)
Название книги	B_TITLE	VARCHAR(50)	обязательное поле
Цена	B_PRICE	DECIMAL(6, 2)	цена экземпляра книги
Затраты	B_ADVANCE	DECIMAL(10, 2)	общая сумма затрат на книгу
Авторский гонорар	B_FEE	DECIMAL(8, 2)	общая сумма гонорара
Дата выхода	B_PUBL	DATE	обязательное поле
Тираж	B_CIRCUL	INT	обязательное поле
Ответственный редактор	B_EDIT	INT	внешний ключ (к Employees)

Таблица 3 – Схема отношения Авторы (Authors)

Содержание поля	Имя поля	Тип, длина	Примечания
Код автора	A_ID	INT	суррогатный первичный ключ
Фамилия, имя, отчество	A_NAME	VARCHAR(50)	обязательное поле

Таблица 4 – Схема отношения Заказы (Orders)

Содержание поля	Имя поля	Тип, длина	Примечания
Номер заказа	O_ID	INT	первичный ключ
Фамилия Заказчика	O_FAMILY	VARCHAR(50)	обязательное поле
Имя Заказчика	O_NAME	VARCHAR(50)	обязательное поле
Отчество Заказчика	O_SURNAME	VARCHAR(50)	обязательное поле
Дата поступления заказа	O_DATE	DATE	обязательное поле
Адрес Заказчика	O_ADDR	VARCHAR(50)	обязательное поле
Дата выполнения заказа	O_READY	DATE	обязательное поле

Таблица 5 – Схема отношения Книги - Редакторы (Editors)

Содержание поля	Имя поля	Тип, длина	Примечания
Код книги (№ контракта)	B_ID	INT	внешний ключ (к Books)
Код редактора	E_ID	INT	внешний ключ (к Employees)

Таблица 6 – Схема отношения СТРОКИ ЗАКАЗА (Items)

Содержание поля	*Имя поля*	*Тип, длина*	*Примечания*
Номер заказа	O_ID	INT	внешний ключ (к Orders)
Код книги (№ контракта)	B_ID	INT	внешний ключ (к Books)
Количество	B_COUNT	INT	обязательное поле

2. Технологический раздел

В качестве языка программирования базы данных был выбран язык запросов SQL, обладающий рядом достоинств:

- стандартность – использование языка SQL в программах стандартизировано международными организациями;

-независимость от конкретных СУБД – все распространенные СУБД используют SQL;

- возможность переноса с одной вычислительной системы на другую – СУБД может быть ориентирована на различные вычислительные системы;

- реляционная основа языка – SQL является языком реляционных баз данных;

- простота изучения;

- возможность создания интерактивных запросов – язык SQL обеспечивает пользователям немедленный доступ к данным, при этом в интерактивном режиме можно получить результат запроса за очень короткое время без написания сложной программы;

- возможность программного доступа к БД – язык SQL легко использовать в приложениях, которым необходимо обращаться к базам данных;

- обеспечение различного представления данных – с помощью языка SQL можно представить такую структуру данных, что тот или иной пользователь будет видеть различные их представления;

- возможность динамического изменения и расширения структуры базы данных – язык SQL позволяет манипулировать структурой базы данных;

- поддержка архитектуры клиент-сервер.

Так как база данных была спроектирована с помощью средств СУБД Microsoft SQL Server, то в качестве процедурного расширения для работы в среде языка SQL был выбран язык Transact-SQL.

Язык Transact-SQL является ключом к использованию SQL Server. Все приложения, взаимодействующие с экземпляром SQL Server, независимо от их реализации и пользовательского интерфейса, отправляют серверу инструкции Transact-SQL.

База данных была разработана в среде Microsoft SQL Server Management Studio 18.

В качестве языка программирования для написания приложений – генераторов данных был выбран язык C++.

Язык C++ является компилируемым статически типизированным языком программирования общего назначения. Поддерживает разные парадигмы программирования, но, в сравнении с его предшественником — языком C, — наибольшее внимание уделено поддержке объектно-ориентированного и обобщённого программирования. Язык C++ имеет синтаксис, основанный на синтаксисе C.

Язык C++ обладает достоинств:

— высокая совместимость с языком C, позволяющая использовать весь существующий C-код;

—поддержка различных стилей и технологий программирования, включая традиционное директивное программирование, ООП, обобщенное программирование, метапрограммирование (шаблоны, макросы);

— наличие возможности работы на низком уровне с памятью, адресами, портами;

— возможность создания обобщённых контейнеров и алгоритмов для разных типов данных, их специализация и вычисления на этапе компиляции, при использовании шаблонов;

— кроссплатформенность;

— эффективность.

В то же самое время язык C++ не лишен недостатков, ряд из которых унаследованы от языка C:

— синтаксис, провоцирующий ошибки;

— примитивный препроцессор;

— плохая поддержка модульности.

В качестве среды разработки приложения была выбрана Visual Studio 2017.

Данная среда обладает рядом достоинств:

—поддержка множества языков при разработке;

—интуитивный стиль кодирования;

— более высокая скорость разработки;

— возможности отладки.

В качестве недостатка можно отметить невозможность отладчика (Microsoft Visual Studio Debugger) отслеживать в коде режима ядра. Отладка в Windows в режиме ядра в общем случае выполняется при использовании Win Dbg, KD или Soft ICE.

Диаграмма спроектированной базы данных издательской компании приведена на рисунке 6.

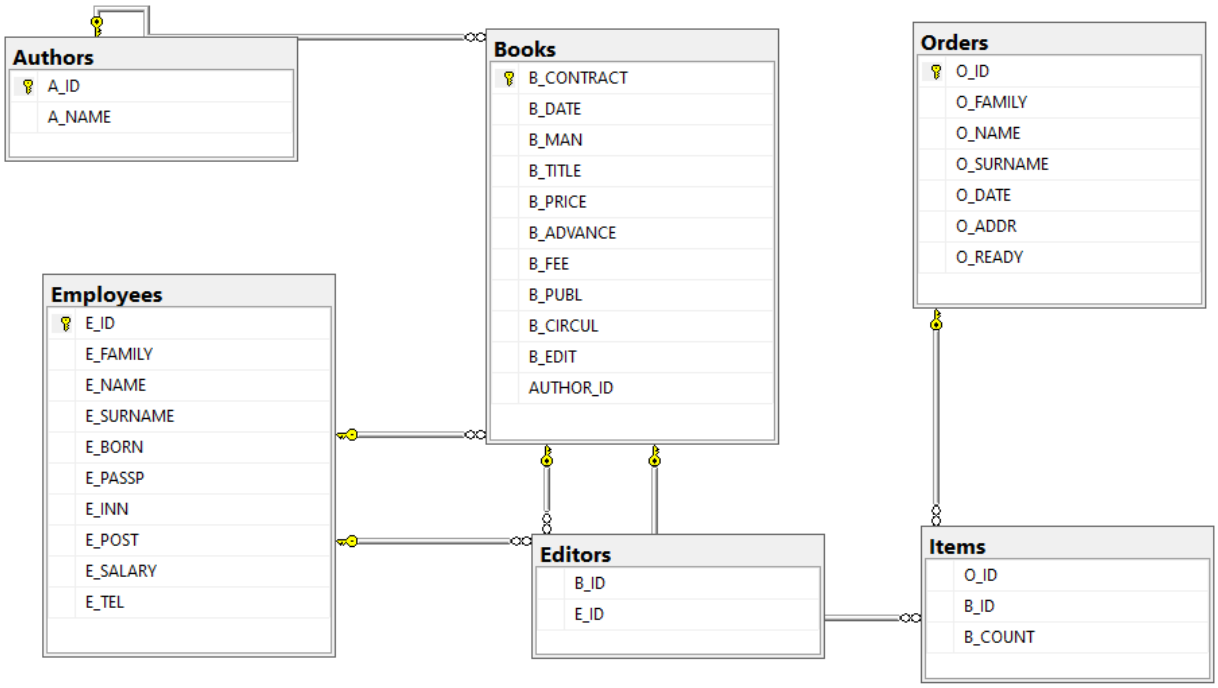

Рисунок 6 – Диаграмма базы данных издательской компании

Заключение

В результате работы, выполненной в рамках курсового проекта, была спроектирована база данных издательской компании.

Проектирование базы данных осуществлялось с помощью языка запросов SQL в среде Microsoft SQL Server Management Studio 18 с помощью средств СУБД SQL Server.

Спроектированная база данных содержит данные о сотрудниках компании, книгах, авторах, финансовом состоянии компании и предоставляет возможность получать разнообразные отчёты, что существенно облегчает работу сотрудников компании, повышая результативность их работы.

Список использованных источников

1. Томас Коннолли, Каронлин Бегг, Анна Страчан. Базы Данных "Проектирование, реализация и сопровождение. Теория и практика. / Второе издание: исправленное и дополненное: Пер. с англ. - М.: Издательский дом "Вильямс", 2001. - 1120 с.: ил. - Парал. Тит. Англ.
2. Бен Форта. SQL за 10 минут / третье издание: пер. с. Англ. - М.: Издательский дом "Вильямс", 2005. - 288 с.: ил. - Парал. Тит. Англ.

Приложение А

Листинг программного кода приложений – генераторов данных

```
/*Генератор данных для таблицы Authors*/
#include "pch.h"

#define _CRT_SECURE_NO_WARNINGS

#include <iostream>
#include <fstream>
#include <cstdio>
#include <ctime>
#include <cstdlib>
#include <memory.h>

using namespace std;

void GenerateData(int MaxRecs);

int main()
{
	setlocale(LC_ALL, "Russian");

	GenerateData(11);
}

void GenerateData(int MaxRecs)
{
	FILE *Authors = fopen("C:\\Users\\Alex\\Desktop\\Программная инженерия\\Базы данных\\Курсовой проект\\Данные\\Authors.txt", "w");

	srand(time(NULL));

	const char *A_AUTHOR[] =
	{
		"Стивен Кинг", "Маргарет Митчелл", "Александр Дюма",
		"Артур Конан Дойль", "Марио Пьюзо", "Агата Кристи",
		"Михаил Булгаков", "Эрих Мария Ремарк", "Виктор Гюго",
		"Александр Волков", "Джейн Остин"
	};

	for (int i = 0; i < MaxRecs; i++) {
		fprintf(Authors, "%d\t", i+1);
		fprintf(Authors, "%s\n", A_AUTHOR[i]);
	}

	fclose(Authors);
}



/*Генератор данных для таблицы Books*/
#include "pch.h"

#define _CRT_SECURE_NO_WARNINGS

#include <iostream>
#include <fstream>
#include <cstdio>
#include <ctime>
#include <cstdlib>
#include <memory.h>
```

```
using namespace std;

double RandDbl(void);

int RandInt(int n);

void GenerateData(int MaxRecs);

int main()
{
	setlocale(LC_ALL, "Russian");

	GenerateData(15);
}

double RandDbl(void)
{
	return rand() / ((double)RAND_MAX + 1.0);
}

int RandInt(int n)
{
	return (int)(n * RandDbl());
}

void GenerateData(int MaxRecs)
{
	FILE *Books = fopen("C:\\Users\\Alex\\Desktop\\Программная инженерия\\Базы данных\\Курсовой проект\\Данные\\Books.txt", "w");

	srand(time(NULL));

	const char *B_BOOK[] =
	{
		"Зеленая миля", "Унесенные ветром", "Граф Монте-Кристо",
		"Приключения Шерлока Холмса", "Крестный отец", "Темная башня",
		"Десять негритят", "Побег из Шоушенка", "Мастер и Маргарита",
		"Записки юного врача", "Искра жизни", "На Западном фронте без перемен",
		"Отверженные", "Волшебник Изумрудного города", "Гордость и предубеждение"
	};

	int MANAGER_ID[6] = { 8565, 2844, 6235, 2614,  3840, 8578 };
	int EDITOR_ID[4] = { 6517, 2419, 1056, 5290 };

	size_t NumMANAGER_ID = sizeof MANAGER_ID / sizeof MANAGER_ID[0];
	size_t NumEDITOR_ID = sizeof EDITOR_ID / sizeof EDITOR_ID[0];

	for (int i = 0; i < MaxRecs; i++) {
		fprintf(Books, "%d\t", 100000 + (int)rand() % (999999 - 100000));
		fprintf(Books, "%d%s%d%d%s%d%d\t", 2015 + (int)rand() % (2019 - 2015), "-", 0, (int)rand() % 8 + 1, "-", 0, (int)rand() % 8 + 1);
		fprintf(Books, "%d\t", MANAGER_ID[RandInt(NumMANAGER_ID)]);
		fprintf(Books, "%s\t", B_BOOK[i]);
		fprintf(Books, "%d%s%d\t", 2000 + rand() % (9999 - 2000), ".", 10 + rand() % (99 - 10));
		fprintf(Books, "%d%s%d\t", 20000000 + rand() % (99999999 - 20000000), ".", 10 + rand() % (99 - 10));
		fprintf(Books, "%d%s%d\t", 200000 + rand() % (999999 - 200000), ".", 10 + rand() % (99 - 10));
		fprintf(Books, "%d%s%d%d%s%d%d\t", 2000 + (int)rand() % (2019 - 2000), "-", 0, (int)rand() % 8 + 1, "-", 0, (int)rand() % 8 + 1);
		fprintf(Books, "%d\t", 10000 + (int)rand() % (99999 - 10000));
		fprintf(Books, "%d\n", EDITOR_ID[RandInt(NumEDITOR_ID)]);
	}


	fclose(Books);
}
```

```
/*Генератор данных для таблицы Editors*/
#include "pch.h"

#define _CRT_SECURE_NO_WARNINGS

#include <iostream>
#include <fstream>
#include <cstdio>
#include <ctime>
#include <cstdlib>
#include <memory.h>

using namespace std;

double RandDbl(void);

int RandInt(int n);

void GenerateData(int MaxRecs);

int main()
{
	setlocale(LC_ALL, "Russian");

	GenerateData(15);
}

double RandDbl(void)
{
	return rand() / ((double)RAND_MAX + 1.0);
}

int RandInt(int n)
{
	return (int)(n * RandDbl());
}

void GenerateData(int MaxRecs)
{
	FILE *Books = fopen("C:\\Users\\Alex\\Desktop\\Программная инженерия\\Базы данных\\Курсовой
проект\\Данные\\Books.txt", "w");

	srand(time(NULL));

	const char *B_BOOK[] =
	{
		"Зеленая миля", "Унесенные ветром", "Граф Монте-Кристо",
		"Приключения Шерлока Холмса", "Крестный отец", "Темная башня",
		"Десять негритят", "Побег из Шоушенка", "Мастер и Маргарита",
		"Записки юного врача", "Искра жизни", "На Западном фронте без перемен",
		"Отверженные", "Волшебник Изумрудного города", "Гордость и предубеждение"
	};

	int MANAGER_ID[6] = { 8565, 2844, 6235, 2614,  3840, 8578 };
	int EDITOR_ID[4] = { 6517, 2419, 1056, 5290 };

	size_t NumMANAGER_ID = sizeof MANAGER_ID / sizeof MANAGER_ID[0];
	size_t NumEDITOR_ID = sizeof EDITOR_ID / sizeof EDITOR_ID[0];

	for (int i = 0; i < MaxRecs; i++) {
		fprintf(Books, "%d\t", 100000 + (int)rand() % (999999 - 100000));
		fprintf(Books, "%d%s%d%d%s%d%d\t", 2015 + (int)rand() % (2019 - 2015), "-", 0,
(int)rand() % 8 + 1, "-", 0, (int)rand() % 8 + 1);
		fprintf(Books, "%d\t", MANAGER_ID[RandInt(NumMANAGER_ID)]);
		fprintf(Books, "%s\t", B_BOOK[i]);
		fprintf(Books, "%d%s%d\t", 2000 + rand() % (9999 - 2000), ".", 10 + rand() % (99 - 10));
```

```
			fprintf(Books, "%d%s%d\t", 20000000 + rand() % (99999999 - 20000000), ".", 10 + rand() %
(99 - 10));
			fprintf(Books, "%d%s%d\t", 200000 + rand() % (999999 - 200000), ".", 10 + rand() % (99 -
10));
			fprintf(Books, "%d%s%d%d%s%d%d\t", 2000 + (int)rand() % (2019 - 2000), "-", 0,
(int)rand() % 8 + 1, "-", 0, (int)rand() % 8 + 1);
			fprintf(Books, "%d\t", 10000 + (int)rand() % (99999 - 10000));
			fprintf(Books, "%d\n", EDITOR_ID[RandInt(NumEDITOR_ID)]);
		}


		fclose(Books);
}



/*Генератор данных для таблицы Employees*/
#include "pch.h"

#define _CRT_SECURE_NO_WARNINGS

#include <iostream>
#include <fstream>
#include <cstdio>
#include <ctime>
#include <cstdlib>
#include <memory.h>

using namespace std;

double RandDbl(void);

int RandInt(int n);

void GenerateData(int MaxRecs);

int main()
{
		setlocale(LC_ALL, "Russian");

		GenerateData(10);
}

double RandDbl(void)
{
		return rand() / ((double)RAND_MAX + 1.0);
}

int RandInt(int n)
{
		return (int)(n * RandDbl());
}

void GenerateData(int MaxRecs)
{
		FILE *Employees = fopen("C:\\Users\\Alex\\Desktop\\Программная инженерия\\Базы данных\\Курсовой
проект\\Данные\\Employees.txt", "w");

		srand(time(NULL));

		const char *E_FAMILY[] =
		{
			"Петров", "Алексеев", "Сергеев",
			"Константинов", "Андреев", "Александров",
			"Денисов", "Романов", "Павлов",
			"Викторов", "Тимофеев", "Михайлов",
			"Владимиров", "Борисов", "Иванов"
		};
```

```
	const char *E_NAME[] =
	{
		"Петр", "Алексей", "Сергей",
		"Константин", "Евгений", "Александр",
		"Денис", "Роман", "Павел",
		"Игорь", "Тимофей", "Даниил",
		"Владимир", "Борис", "Иван"
	};

	const char *E_SURNAME[] =
	{
		"Петрович", "Алексеевич", "Сергеевич",
		"Константинович", "Евгеньевич", "Александрович",
		"Денисович", "Романович", "Павлович",
		"Игоревич", "Тимофеевич", "Данилович",
		"Владимирович", "Борисович", "Иванович"
	};

	const char *E_POST[] =
	{
		"Менеджер", "Редактор"
	};

	size_t NumE_FAMILY = sizeof E_FAMILY / sizeof E_FAMILY[0];
	size_t NumE_NAME = sizeof E_NAME / sizeof E_NAME[0];
	size_t NumE_SURNAME = sizeof E_SURNAME / sizeof E_SURNAME[0];
	size_t NumE_POST = sizeof E_POST / sizeof E_POST[0];

	for (int i = 0; i < MaxRecs; i++) {
		fprintf(Employees, "%d\t", 100 + (int)rand() % (9999 - 100));
		fprintf(Employees, "%s\t", E_FAMILY[RandInt(NumE_FAMILY)]);
		fprintf(Employees, "%s\t", E_NAME[RandInt(NumE_NAME)]);
		fprintf(Employees, "%s\t", E_SURNAME[RandInt(NumE_SURNAME)]);
		fprintf(Employees, "%d%s%d%d%s%d%d\t", 1970 + (int)rand() % (1990 - 1970), "-", 0,
(int)rand() % 8 + 1, "-",0, (int)rand() % 8+1);
		fprintf(Employees, "%d%s%d\t", 1000 + (int)rand() % (9999 - 1000), " ", 100000 +
(int)rand() % (999999 - 100000));
		fprintf(Employees, "%d\t", 100000 + (int)rand() % (999999 - 100000));
		fprintf(Employees, "%s\t", E_POST[RandInt(NumE_POST)]);
		fprintf(Employees, "%d%s%d\t", 20000 + rand() % (50000 - 20000), ".", 10 + rand() % (99 -
10));
		fprintf(Employees, "%d%s%d%s%d%s%d%s%d\n", 8, "(", 900 + (int)rand()%(999-900), ")", 100
+ (int)rand() % (999 - 100), "-", 10 + (int)rand() % (99 - 10), "-", 10 + (int)rand()%(99-10));
	}

	fclose(Employees);
}



/*Генератор данных для таблицы Orders*/
#include "pch.h"

#define _CRT_SECURE_NO_WARNINGS

#include <iostream>
#include <fstream>
#include <cstdio>
#include <ctime>
#include <cstdlib>
#include <memory.h>

using namespace std;

double RandDbl(void);

int RandInt(int n);
```

```
void GenerateData(int MaxRecs);

int main()
{
	setlocale(LC_ALL, "Russian");

	GenerateData(20);
}

double RandDbl(void)
{
	return rand() / ((double)RAND_MAX + 1.0);
}

int RandInt(int n)
{
	return (int)(n * RandDbl());
}

void GenerateData(int MaxRecs)
{
	FILE *Orders = fopen("C:\\Users\\Alex\\Desktop\\Программная инженерия\\Базы данных\\Курсовой проект\\Данные\\Orders.txt", "w");

	srand(time(NULL));

	const char *O_FAMILY[] =
	{
		"Петров", "Алексеев", "Сергеев",
		"Константинов", "Андреев", "Александров",
		"Денисов", "Романов", "Павлов",
		"Викторов", "Тимофеев", "Михайлов",
		"Владимиров", "Борисов", "Иванов"
	};

	const char *O_NAME[] =
	{
		"Петр", "Алексей", "Сергей",
		"Константин", "Евгений", "Александр",
		"Денис", "Роман", "Павел",
		"Игорь", "Тимофей", "Даниил",
		"Владимир", "Борис", "Иван"
	};

	const char *O_SURNAME[] =
	{
		"Петрович", "Алексеевич", "Сергеевич",
		"Константинович", "Евгеньевич", "Александрович",
		"Денисович", "Романович", "Павлович",
		"Игоревич", "Тимофеевич", "Данилович",
		"Владимирович", "Борисович", "Иванович"
	};

	const char *O_STREET[] =
	{
		"ул. Достоевского", "ул. Тургенева", "ул. Ломоносова",
		"ул. Ленина", "ул. Гагарина", "ул. Гайдара",
		"пр. Просвещения", "Ленинский пр.", "ул. Пушкина"
	};

	size_t NumO_FAMILY = sizeof O_FAMILY / sizeof O_FAMILY[0];
	size_t NumO_NAME = sizeof O_NAME / sizeof O_NAME[0];
	size_t NumO_SURNAME = sizeof O_SURNAME / sizeof O_SURNAME[0];
	size_t NumO_STREET = sizeof O_STREET / sizeof O_STREET[0];

	for (int i = 0; i < MaxRecs; i++) {
```

```
            fprintf(Orders, "%d\t", 100 + (int)rand() % (9999 - 100));
            fprintf(Orders, "%s%\t", O_FAMILY[RandInt(NumO_FAMILY)]);
            fprintf(Orders, "%s\t",O_NAME[RandInt(NumO_NAME)]);
            fprintf(Orders, "%s\t", O_SURNAME[RandInt(NumO_SURNAME)]);
            fprintf(Orders, "%d%s%d%d%s%d%d\t", 2019, "-", 0, 1 + (int)rand() % (5 - 1), "-", 0, 1 +
(int)rand() % (10 - 1));
            fprintf(Orders, "%s%s%d%s%d\t", O_STREET[RandInt(NumO_STREET)], " д. ", (int)rand() % 50,
" кв. ", (int)rand() % 120);
            fprintf(Orders, "%d%s%d%d%s%d%d\n", 2019, "-", 0, 6 + (int)rand() % 4, "-", 0, 6 +
(int)rand() % 4);
      }

      fclose(Orders);
}
```

Приложение Б

Листинг программного кода SQL запросов

```
/*Создание базы данных*/

USE [master]
GO

IF EXISTS (SELECT name FROM sys.databases WHERE name = N'Orders')
DROP DATABASE [Orders]
GO

CREATE DATABASE [Orders]
GO


/*Создание таблицы "Авторы"*/
USE [Orders]
GO

DROP TABLE IF EXISTS dbo.Authors
GO

CREATE TABLE Authors(
A_ID INT NOT NULL PRIMARY KEY,
A_NAME VARCHAR(50) NOT NULL
);
GO



/*Заполнение таблицы "Авторы" данными из текстового файла"*/
USE [Orders]
GO

BULK INSERT [Orders].[dbo].[Authors]
FROM 'C:\Users\Alex\Desktop\Программная инженерия\Базы данных\Курсовой проект\Данные\Authors.txt'

WITH (CODEPAGE = 'ACP', DATAFILETYPE = 'char', FIELDTERMINATOR = '\t', ROWTERMINATOR =
'\n');
GO
```

```
/*Создание таблицы "Книги"*/
USE [Orders]
GO

DROP TABLE IF EXISTS dbo.Books
GO

CREATE TABLE Books(
B_CONTRACT INT NOT NULL PRIMARY KEY,
B_DATE DATE NOT NULL,
B_MAN INT NOT NULL FOREIGN KEY REFERENCES Employees(E_ID),
B_TITLE VARCHAR(50) NOT NULL,
B_PRICE DECIMAL(6, 2) NOT NULL,
B_ADVANCE DECIMAL(10, 2) NOT NULL,
B_FEE DECIMAL(8, 2) NOT NULL,
B_PUBL DATE NOT NULL,
B_CIRCUL INT NOT NULL,
B_EDIT INT NOT NULL FOREIGN KEY REFERENCES Employees(E_ID)
);
GO



/*Заполнение таблицы "Книги" данными из текстового файла"*/
USE [Orders]
GO

BULK INSERT [Orders].[dbo].[Books]
FROM 'C:\Users\Alex\Desktop\Программная инженерия\Базы данных\Курсовой проект\Данные\Books.txt'

WITH (CODEPAGE = 'ACP', DATAFILETYPE = 'char', FIELDTERMINATOR = '\t', ROWTERMINATOR =
'\n');
GO



/*Создание таблицы "Сотрудники магазина"*/
USE [Orders]
GO

DROP TABLE IF EXISTS dbo.Employees
GO

CREATE TABLE Employees(
E_ID INT NOT NULL PRIMARY KEY,
E_FAMILY VARCHAR (50) NOT NULL,
E_NAME VARCHAR (50) NOT NULL,
E_SURNAME VARCHAR (50) NOT NULL,
E_BORN DATE NOT NULL,
E_PASSP VARCHAR(50) NOT NULL,
E_INN INT NOT NULL,
E_POST VARCHAR(50) NOT NULL,
E_SALARY DECIMAL(8, 2) NOT NULL,
E_TEL VARCHAR(50) NOT NULL,
);
GO


/*Заполнение таблицы "Сотрудники магазина" данными из текстового файла"*/
USE [Orders]
GO

BULK INSERT [Orders].[dbo].[Employees]
FROM 'C:\Users\Alex\Desktop\Программная инженерия\Базы данных\Курсовой проект\Данные\Employees.txt'

WITH (CODEPAGE = 'ACP', DATAFILETYPE = 'char', FIELDTERMINATOR = '\t', ROWTERMINATOR =
'\n');
GO
```

```
/*Создание таблицы "Книги-Редакторы"*/
USE [Orders]
GO

DROP TABLE IF EXISTS dbo.Editors
GO

CREATE TABLE Editors(
B_ID INT NOT NULL FOREIGN KEY REFERENCES Books(B_CONTRACT),
E_ID INT NOT NULL FOREIGN KEY REFERENCES Employees(E_ID)
);
GO



/*Создание таблицы "Заказы"*/
USE [Orders]
GO

DROP TABLE IF EXISTS dbo.Orders
GO

CREATE TABLE Orders(
O_ID INT NOT NULL PRIMARY KEY,
O_FAMILY VARCHAR (50) NOT NULL,
O_NAME VARCHAR (50) NOT NULL,
O_SURNAME VARCHAR (50) NOT NULL,
O_DATE DATE NOT NULL,
O_ADDR VARCHAR(50) NOT NULL,
O_READY DATE NOT NULL,
);
GO



/*Заполнение таблицы "Заказы" данными из текстового файла"*/
USE [Orders]
GO

BULK INSERT [Orders].[dbo].[Orders]
FROM 'C:\Users\Alex\Desktop\Программная инженерия\Базы данных\Курсовой проект\Данные\Orders.txt'

WITH (CODEPAGE = 'ACP', DATAFILETYPE = 'char', FIELDTERMINATOR = '\t', ROWTERMINATOR =
'\n');
GO



/*Создание таблицы "Количество книг в заказе"*/
USE [Orders]
GO

DROP TABLE IF EXISTS dbo.Items
GO

CREATE TABLE Items(
O_ID INT NOT NULL FOREIGN KEY REFERENCES Orders(O_ID),
B_ID INT NOT NULL FOREIGN KEY REFERENCES Books(B_CONTRACT),
B_COUNT INT NOT NULL
);
GO
```

```sql
/*Создание представления "Авторы - Книги"*/
USE [Orders]
GO

CREATE VIEW AuthorsBooks AS
SELECT Books.B_TITLE AS Book,
          Authors.A_NAME AS Author,
          Books.B_PRICE AS Price
FROM Books INNER JOIN Authors ON Authors.A_ID = Books.AUTHOR_ID
GO



/*Создание представления "Книги - Редакторы"*/
USE [Orders]
GO

CREATE VIEW BooksEditors AS
SELECT Books.B_TITLE AS Book,
          Employees.E_FAMILY AS EditorFamily,
          Employees.E_NAME AS EditorName,
          Employees.E_SURNAME AS EditorSurname

FROM Books INNER JOIN Employees ON Books.B_EDIT = Employees.E_ID
GO



/*Создание представления "Заказчики*/
CREATE VIEW Customers AS
SELECT Orders.O_FAMILY AS CustomerFamily,
          Orders.O_NAME AS CustomerName,
          Orders.O_SURNAME AS CustomerSurname,
          Orders.O_ID AS OrderID,
          Books.B_TITLE AS Book,
          Books.B_CONTRACT AS BookID,
          Items.B_COUNT AS Quantity,
          Orders.O_READY AS OrderReady

FROM Items INNER JOIN Orders ON Items.O_ID = Orders.O_ID
INNER JOIN Books ON Items.B_ID = Books.B_CONTRACT
GO



/*Создание хранимой процедуры "Книги - Цена - Менеджер*/
USE [Orders]
GO

DROP PROCEDURE IF EXISTS dbo.BookSummary
GO

CREATE PROCEDURE BookSummary AS
SELECT B_TITLE AS Book, B_PRICE AS Price, B_MAN AS Manager
FROM Books



/*Создание хранимой процедуры "Реквизиты сотрудников*/
DROP PROCEDURE IF EXISTS dbo.EmployeeSummary
GO

CREATE PROCEDURE EmployeeSummary AS
SELECT E_FAMILY AS Family, E_NAME AS Name, E_SURNAME AS Surname, E_POST AS Position, E_TEL AS Telephone
FROM Employees
```

```
/*Создание хранимой процедуры "Информация о заказах*/
USE [Orders]
GO

DROP PROCEDURE IF EXISTS dbo.OrderSummary
GO

CREATE PROCEDURE OrderSummary AS
SELECT O_ID AS Id, O_FAMILY AS Family, O_NAME AS Name, O_SURNAME AS Surname, O_ADDR AS Address
FROM Orders


/*Создание таблицы "События базы данных"*/
USE [Orders]
GO

DROP TABLE IF EXISTS dbo.History
GO

CREATE TABLE History
(
       B_CONTRACT INT NOT NULL,
    Operation VARCHAR(200) NOT NULL,
    CreateAt DATETIME NOT NULL DEFAULT GETDATE(),
);
GO



/*Добавление новой записи в таблицу "Книги*/
USE [Orders]
GO

INSERT INTO Books (B_CONTRACT, B_DATE, B_MAN, B_TITLE, B_PRICE, B_ADVANCE, B_FEE, B_PUBL, B_CIRCUL,
B_EDIT, AUTHOR_ID)

VALUES(1354837, '2015-07-08', 6235, 'Собачье сердце', 5453.5, 25622841.51, 265846.82, '2001-05-06',
27576, 5290, 7)

SELECT * FROM History



/*Создание триггера "Добавлена книга*/
CREATE TRIGGER BooksInsert
ON Books
AFTER INSERT
AS
INSERT INTO History (B_CONTRACT, OPERATION)
SELECT B_CONTRACT, 'Добавлена книга ' + B_TITLE
FROM INSERTED



/*Удаление записи в таблице "Книги*/
USE [Orders]
GO

DELETE FROM Books
WHERE B_CONTRACT = 1354837

SELECT * FROM History
```

```
/*Создание триггера "Удалена книга*/
USE [Orders]
GO

CREATE TRIGGER BooksDelete
ON Books
AFTER DELETE
AS
INSERT INTO History (B_CONTRACT, OPERATION)
SELECT B_CONTRACT, 'Удалена книга ' + B_TITLE
FROM DELETED



/*Изменение записи в таблице "Книги*/
USE [Orders]
GO

UPDATE Books SET B_PRICE = 9000.00
WHERE B_PRICE = 8800.00;

SELECT *FROM HISTORY



/*Создание триггера "Обновлена книга*/
USE [Orders]
GO

CREATE TRIGGER BookUpdate
ON Books
AFTER UPDATE
AS
INSERT INTO History (B_CONTRACT, OPERATION)
SELECT B_CONTRACT, 'Обновлена книга ' + B_TITLE
FROM INSERTED



/*Функция вычисления максимальной стоимости книги, помноженной на 100"*/
USE [Orders]
GO

DROP FUNCTION IF EXISTS MaxPrice
GO

CREATE FUNCTION MaxPrice(@a INT)
RETURNS INT
BEGIN
DECLARE @maxPr INT
SET  @maxPr = (SELECT MAX(B_PRICE) FROM BOOKS)*@a;
RETURN @maxPr
END
GO

PRINT Orders.dbo.MaxPrice(100)
```

```sql
/*Функция выводящая авторов, у которых нескольких книг*/
USE [Orders]
GO

DROP FUNCTION IF EXISTS MultiAuthors
GO

CREATE FUNCTION MultiAuthors()
RETURNS TABLE
AS
RETURN (SELECT Authors.A_NAME AS Author
FROM Authors INNER JOIN Books ON Authors.A_ID = Books.AUTHOR_ID
GROUP BY Authors.A_NAME
HAVING COUNT(*)  > 1 );
GO

SELECT * FROM MultiAuthors()
```