

Лабораторная работа № 9

Тема работы: Создание логической модели данных в *ERwin Data Modeler*.

ERwin Data Modeler – средство концептуального моделирования баз данных. В *ERwin* реализованы основные функции, характерные для классических CASE-средств:

- прямое проектирование от создания концептуальной или логической схемы БД до генерации структуры БД на диске или *DDL*-скрипта;
- обратное проектирование (реинжиниринг) – создание физической схемы БД на основе БД на диске или *DDL*-скрипта;
- синхронизация модели БД с самой БД на диске.

К достоинствам *ERwin* относятся:

- поддержка около 10 промышленных СУБД (*ORACLE*, *MySQL*, *DB2*, *MS SQL Server* и др.) и их различных версий, а также возможность разработки БД для любых СУБД поддерживающих интерфейс *ODBC*;
- наличие функции проверки модели БД требованиям полноты, целостности и нормализации.

ERwin имеет два уровня представления модели – логический и физический.

Логический уровень – это абстрактный взгляд на данные, никак не связанный с конкретной реализацией СУБД. Объектами модели, представляемой на логическом уровне, являются сущности и атрибуты.

Физическая модель данных зависит от конкретной СУБД и фактически является отображением системного каталога. В физической модели содержится информация обо всех объектах БД. Поскольку стандартов на объекты БД не существует (например, нет стандарта на типы данных), физическая модель зависит от конкретной реализации СУБД. Следовательно, одной и той же логической модели могут соответствовать несколько разных физических моделей. Если в логической модели не имеет значения, какой конкретно тип данных имеет атрибут, то в физической модели важно описать всю информацию о конкретных физических объектах – таблицах, колонках, индексах, процедурах и т.д.

Создание модели данных, как правило, начинается с разработки логической модели. После описания логической модели проектировщик может выбрать необходимую СУБД, и *ERwin* автоматически создаст соответствующую модель. На основе физической модели *ERwin* может сгенерировать системный каталог СУБД или соответствующий *SQL*-скрипт. Этот процесс называется прямым проектированием (*Forward Engineering*).

Тем самым достигается масштабируемость – создав одну логическую модель данных, можно сгенерировать физические модели под любую поддерживаемую *ERwin* СУБД. С другой стороны, *ERwin* способен по содержимому системного каталога или *SQL*-скрипту воссоздать физическую и логическую модель данных (*Reverse Engineering*). На основе полученной логической модели данных можно сгенерировать физическую модель для другой СУБД и затем создать ее системный каталог.

На логическом уровне *ERwin* поддерживает две нотации (*Information Engineering* и *IDEFIX*), на физическом уровне – три нотации (*Information Engineering*, *IDEFIX* и *Data Warehousing*). По умолчанию используется нотация *IDEFIX* (*Integration DEFINition for information modeling*).

Методология IDEFIX

Методология *IDEF1* позволяет построить модель данных, эквивалентную реляционной модели в третьей нормальной форме. На основе совершенствования метода *IDEF1* создана его новая версия – метод *IDEFIX*, разработанный с учетом таких требований, как простота для изучения и возможность автоматизации. *IDEFIX*-диаграммы используются в ряде распространенных *CASE*-средств, например, *ERwin*, *Design/IDEF*.

Метод *IDEFIX* основан на модели «сущность-связь» (*Entity-Relationship Diagram – ERD*) Питера Чена, компонентами которой являются сущности, атрибуты и связи.

Сущность – это класс однотипных объектов, информация о которых должна быть отражена в модели и накапливаться в ИС. Каждая сущность должна иметь наименование, выраженное существительным в единственном числе. Примерами сущностей могут быть такие классы объектов как Поставщик, Сотрудник, Накладная.

Экземпляр сущности – это конкретный представитель данной сущности. Например, представителем сущности Сотрудник может быть Сотрудник Иванов. Экземпляры сущностей должны быть различимы, т.е. сущности должны иметь некоторые свойства, уникальные для каждого экземпляра этой сущности.

Атрибут сущности – это именованная характеристика сущности, значимая для рассматриваемой предметной области и использованная для квалификации, идентификации, классификации, количественной характеристики или выражения состояния сущности. Наименование атрибута должно быть выражено существительным в единственном числе (возможно, с характеризующими прилагательными).

Экземпляр атрибута – это определенная характеристика отдельного элемента множества. Экземпляр атрибута определяется типом характеристики и ее значением, называемым значением атрибута.

Среди атрибутов особое положение занимают такие, с помощью которых можно идентифицировать экземпляр объекта. Атрибут или несколько атрибутов, значения которых уникальным образом идентифицируют каждый экземпляр сущности, называется *ключом сущности*.

На диаграмме *IDEFIX* сущность представляется графическим объектом в виде прямоугольника, разделенного на две части – в верхней части отображаются ключевые атрибуты, в нижней – неключевые (рис. 1). Верхняя часть называется ключевой областью, а нижняя часть – областью данных. Ключевая область объекта *Сотрудник* содержит атрибуты *Код сотрудника* и *Название отдела (FK)*, в области данных находятся поля *Должность*, *ФИО сотрудника*, *Адрес проживания* и *Оклад*.

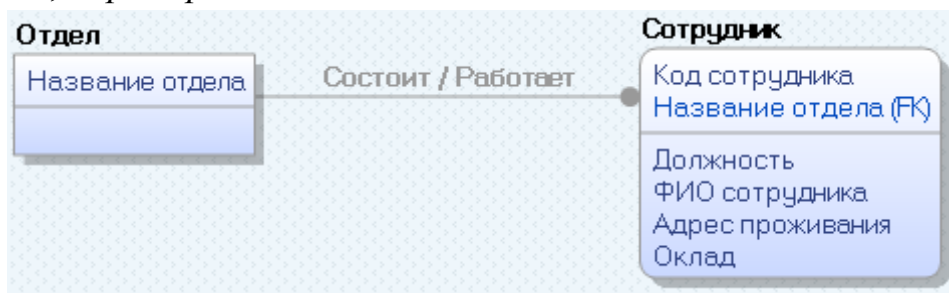


Рис. 1. Пример изображения сущностей и связей на *IDEFIX*-диаграмме

Два объекта могут быть связаны между собой. Подобная связь осуществляется через связь экземпляров одного объекта с экземплярами другого объекта, образуя набор экземпляров связи между двумя объектами, который называется *типом связи*.

Связь – это поименованная ассоциация между двумя сущностями, значимая для рассматриваемой предметной области. Связь – это ассоциация между сущностями, при которой каждый экземпляр одной сущности ассоциирован с произвольным (в том числе нулевым) количеством экземпляров второй сущности, и наоборот. Связи позволяют по одной сущности находить другие сущности, связанные с ней.

Связи в *IDEFIX* представляют собой ссылки, соединения и ассоциации между сущностями. Связи – это суть глаголов, которые показывают, как соотносятся сущности между собой. Например,

- Отдел <состоит из> нескольких Сотрудников;
- Самолет <перевозит> нескольких Пассажиров;
- Сотрудник <пишет> разные Отчеты.

Во всех перечисленных примерах взаимосвязи между сущностями соответствуют схеме *один ко многим*. Это означает, что один экземпляр первой сущности связан с несколькими экземплярами второй сущности. Причем первая сущность называется родительской, а вторая – дочерней. Связи изображаются в виде линии между двумя сущностями с точкой на конце линии у сущности-потомка, имя связи в виде глагола указывается над линией (рис. 1).

Отношения *многие ко многим* обычно используются на начальной стадии разработки диаграммы, например, в диаграмме зависимости сущностей, и отображаются в *IDEFIX* в виде сплошной линии с точками на обоих концах. Так как отношения многие ко многим могут скрыть другие бизнес-правила или ограничения, они должны быть полностью исследованы на одном из этапов моделирования.

Если сущности в *IDEFIX*-диаграмме связаны, связь передает ключ (или набор ключевых атрибутов) дочерней сущности. Эти атрибуты называются *внешними ключами (Foreign Key)*. Внешние ключи определяются как атрибуты первичных ключей родительского объекта, переданные дочернему объекту через их связь. Передаваемые атрибуты называются *мигрирующими*. Для обозначения внешнего ключа внутри блока сущности помещают имена атрибутов, после которых следуют буквы *FK* в скобках. Например, для сущности *Сотрудник* атрибут *Название отдела* является внешним ключом (рис. 1).

При разработке модели зачастую приходится сталкиваться с сущностями, уникальность которых зависит от значений атрибута внешнего ключа. Для этих сущностей (для уникального определения каждой сущности) внешний ключ должен быть частью первичного ключа дочернего объекта.

Дочерняя сущность, уникальность которой зависит от атрибута внешнего ключа, называется *зависимой сущностью*. Например, сущность *Сотрудник* является зависимой сущностью потому, что его идентификация зависит от сущности *Отдел* (рис. 1).

В обозначении *IDEFIX* зависимые сущности представлены в виде закругленных прямоугольников. Зависимые сущности далее классифицируются на сущности, которые не могут существовать без родительской сущности и сущности, которые не могут быть идентифицированы без использования ключа родителя (сущности, зависящие от идентификации).

Сущности, не зависящие при идентификации от других объектов в модели, называются *независимыми сущностями*. Например, *Отдел* можно

считать независимой сущностью. В *IDEF1X* независимые сущности представлены в виде прямоугольников.

В *IDEF1X* концепция зависимых и независимых сущностей усиливается типом взаимосвязей между двумя сущностями. Если необходимо чтобы внешний ключ передавался в дочернюю сущность (и в результате создавал зависимую сущность), то следует создать *идентифицирующую связь* между родительской и дочерней сущностью. Идентифицирующие взаимосвязи обозначаются сплошной линией между сущностями (рис. 1).

Неидентифицирующие связи, являющиеся уникальными для *IDEF1X*, также связывают родительскую сущность с дочерней. Неидентифицирующие связи используются для отображения другого типа передачи атрибутов внешних ключей – передача в область данных дочерней сущности (под линией). Неидентифицирующие связи отображаются пунктирной линией между объектами. Так как переданные ключи в неидентифицирующей связи не являются составной частью первичного ключа дочерней сущности, то этот вид связи не проявляется ни в одной идентифицирующей зависимости. В этом случае и *Отдел*, и *Сотрудник* рассматриваются как независимые сущности (рис. 2).

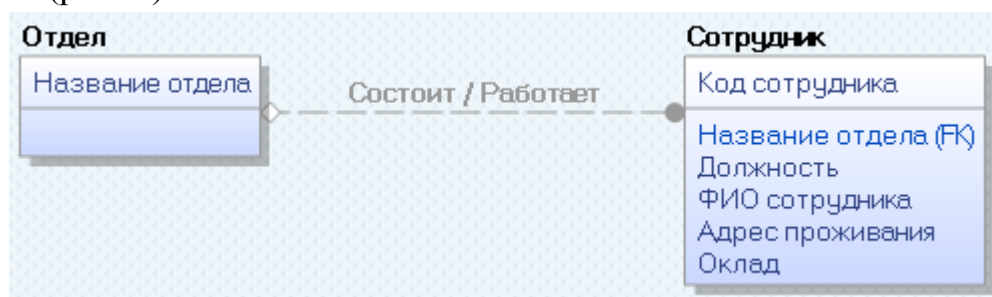


Рис. 2. Пример неидентифицирующей связи

Связь может дополнительно определяться с помощью указания степени или мощности (количества экземпляров сущности-потомка, которое может порождать каждый экземпляр сущности-родителя). В *IDEF1X* могут быть выражены следующие мощности связей:

- каждый экземпляр сущности-родителя может иметь ноль, один или более одного связанного с ним экземпляра сущности-потомка;
- каждый экземпляр сущности-родителя должен иметь не менее одного связанного с ним экземпляра сущности-потомка;
- каждый экземпляр сущности-родителя должен иметь не более одного связанного с ним экземпляра сущности-потомка;
- каждый экземпляр сущности-родителя связан с некоторым фиксированным числом экземпляров сущности-потомка.

Мощность связи может принимать следующие значения: N – ноль, один или более, Z – ноль или один, P – один или более. По умолчанию мощность связи принимается равной N .

Особым типом объединения сущностей является *иерархия наследования* (*иерархия категорий* или *категориальная связь*), согласно которой разделяются их общие характеристики. Обычно иерархию наследования создают, когда несколько сущностей имеют общие по смыслу атрибуты, или когда сущности имеют общие по смыслу связи, или когда это диктуется бизнес-правилами. Для представления информации, общей для всех типов экземпляров сущности, из их свойств формируется обобщенная сущность (родовой предок). При этом специфическая для каждого типа экземпляра сущности информация будет располагаться в категориальных сущностях (потомках).

Например, в отделе работает начальник отдела и менеджеры. Для представления информации, общей для всех типов работников можно сформировать обобщенную сущность *Сотрудник*, а категориальными сущностями будут *Начальник отдела* и *Менеджер* (рис. 3).

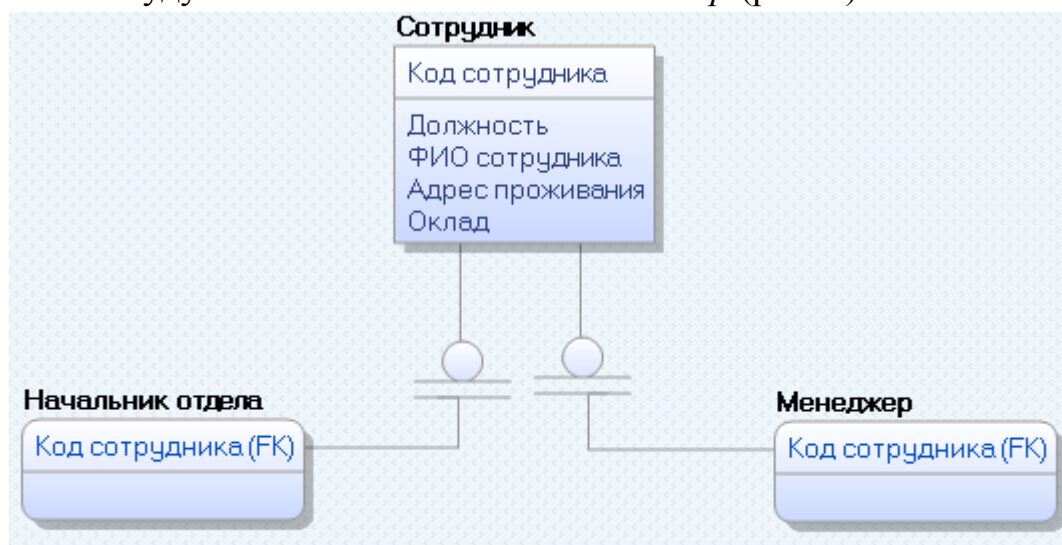


Рис. 3. Пример категориальной связи (иерархии наследования)

Основным преимуществом *IDEFIX*, по сравнению с другими многочисленными методами разработки реляционных баз данных, является жесткая и строгая стандартизация моделирования. Установленные стандарты позволяют избежать различной трактовки построенной модели, которая является значительным недостатком *ER*-диаграмм.

Создание модели данных начинается с разработки логической модели, которая должна представлять состав сущностей предметной области с перечнем атрибутов и отношений между ними.

В зависимости от глубины представления информации о данных различают 3 подуровня логического уровня модели данных:

– *диаграмма сущность-связь (Entity Relationship Diagram, ERD)* – включает сущности и взаимосвязи, отражающие основные бизнес-правила предметной области. Такая диаграмма не слишком детализирована, в нее включаются основные сущности и связи между ними, которые удовлетворяют основным требованиям, предъявляемым к ИС. Диаграмма сущность-связь может включать связи «многие-ко-многим» и не включать описание ключей. Как правило, *ERD* используется для презентаций и обсуждения структуры данных с экспертами предметной области;

– *модель данных, основанная на ключах (Key Based model, KB)* – включает описание всех сущностей и первичных ключей, необходимых для подробного описания предметной области;

– *полная атрибутивная модель данных (Fully Attributed model, FA)* – представляет наиболее детальное представление структуры данных, содержит данные в третьей нормальной форме и включает все сущности, атрибуты и связи.

Описание примера предметной области

Рассмотрим работу отдела сбыта издательства, занимающегося поставкой книг в магазины. В данной предметной области можно выделить сущности СОТРУДНИКИ, МАГАЗИНЫ, КНИГИ, ТРАНСПОРТНЫЕ КОМПАНИИ и ЗАКАЗЫ, связи (рис. 4):

– ДЕЛАЮТ – между объектами МАГАЗИНЫ и ЗАКАЗЫ. Магазин делает несколько заказов в издательство на поставку книг, а один заказ предназначен только для одного магазина, поэтому связь ДЕЛАЮТ относится к типу *один ко многим*;

– ОФОРМЛЯЮТ – между объектами СОТРУДНИКИ и ЗАКАЗЫ. Сотрудник отдела сбыта издательства оформляет несколько заказов, а один заказ может быть оформлен только одним сотрудником, поэтому связь ОФОРМЛЯЮТ относится к типу *один ко многим*;

– ВКЛЮЧАЮТ – между объектами ЗАКАЗЫ и КНИГИ. Заказ может включать несколько книг, а одна книга может входить в несколько различных заказов, поэтому связь ВКЛЮЧАЮТ относится к типу *многие ко многим*;

– ДОСТАВЛЯЮТ – между объектами ТРАНСПОРТНЫЕ КОМПАНИИ и ЗАКАЗЫ. Одна транспортная компания выполняет доставку нескольких заказов, а один заказ доставляет только одна транспортная компания, поэтому связь ДОСТАВЛЯЮТ относится к типу *один ко многим*.

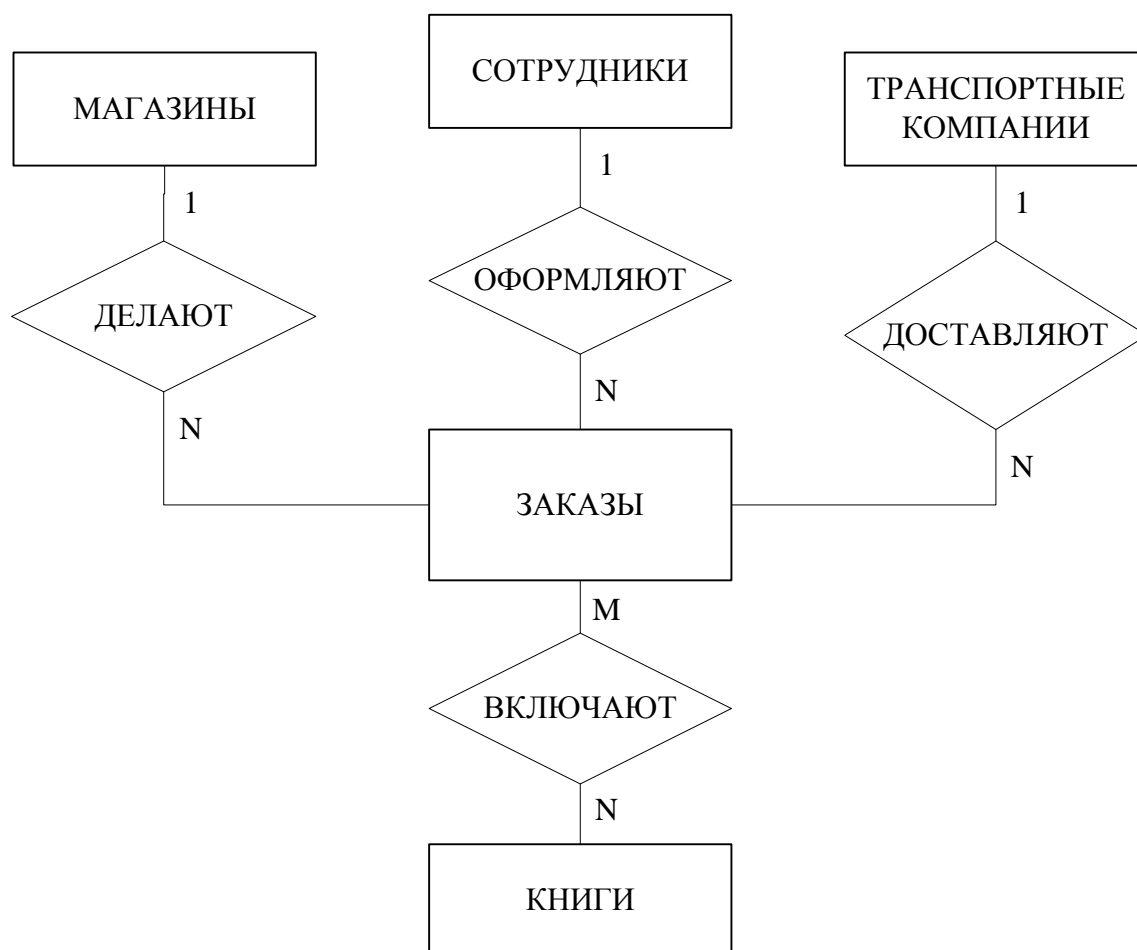


Рис. 4. Диаграмма связей объектов базы данных «Поставки книг в магазины»

Для каждого объекта выделим атрибуты:

- СОТРУДНИКИ (Код сотрудника, Фамилия, Имя, Отчество, Должность, Телефон, Дата рождения, Дата найма);
- МАГАЗИНЫ (Код магазина, Название магазина, ФИО контактного лица, Город, Адрес, Телефон);
- КНИГИ (Код книги, Авторы, Название, Серия, Год издания, Количество страниц, Цена, Диск);
- ТРАНСПОРТНЫЕ КОМПАНИИ (Название компании, ФИО контактного лица, Город, Телефон);
- ЗАКАЗЫ (Код заказа, Требуемая дата выполнения, Фактическая дата выполнения).

Работа с ERwin Data Modeler

Запуск программы *ERwin Data Modeler* осуществляется в главном меню *Windows* выбором команды *Все программы ⇒ ERwin ⇒ ERwin Data Modeler r9.7 ⇒ ERwin Data Modeler*.

Главное окно *ERwin* представлено на рис. 5 и содержит следующие области:

- *ER_Diagram* (Окно диаграммы) – рабочая область диаграммы, которая содержит графическое представление модели данных;
- *Model Explorer* (Проводник модели) – обеспечивает иерархическое представление модели данных, которая отображена в окне диаграммы;
- *Action Log* (Журнал действий) – отображает список действий при работе с моделью;
- *Advisories* (Консультации) – содержит команды для получения справки по работе с программой;
- *Overview* (Обзор) – показывает всю модель целиком, содержит рамку, позволяющую выбрать область для отображения в окне диаграммы.

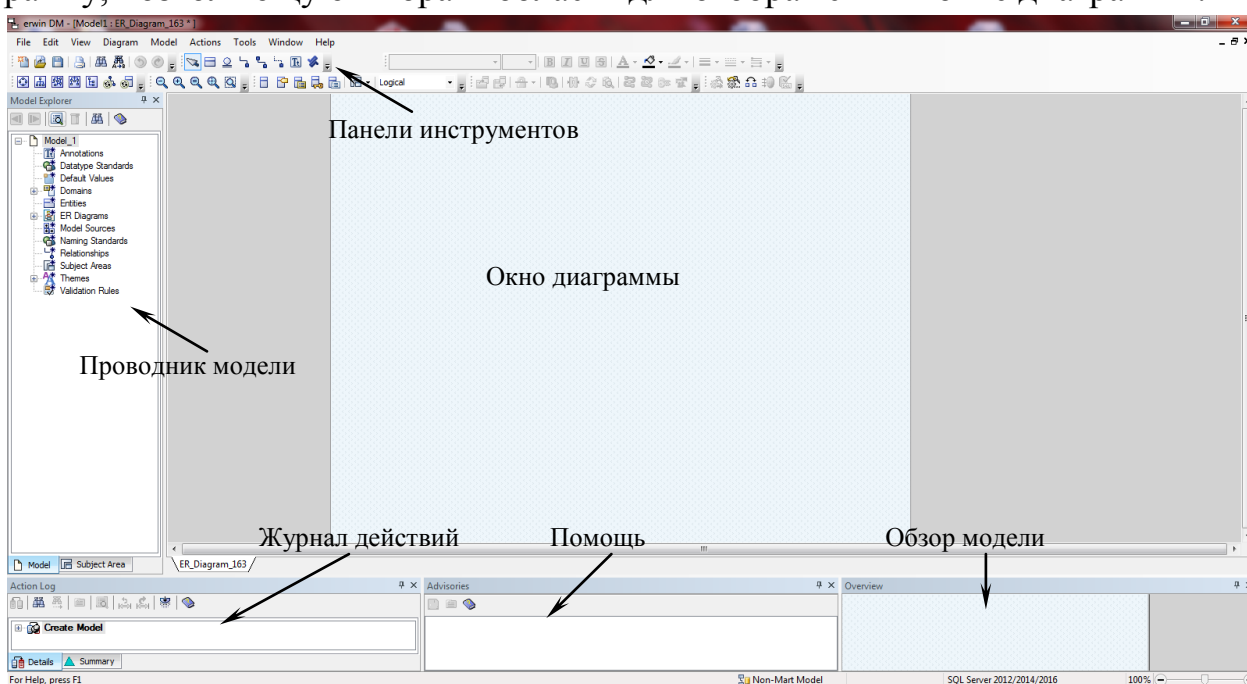


Рис. 5. Окно программы *CA ERwin Data Modeler*

Для создания новой модели необходимо выбрать команду *File* \Rightarrow *New* (Файл \Rightarrow Новый), для открытия существующей модели – *File* \Rightarrow *Open* (Файл \Rightarrow Открыть). Модель сохраняется командой *Save* (Сохранить) или *Save As* (Сохранить как), при сохранении *ERwin* автоматически добавляет выбранное расширение к имени файла. Модель закрывается командой *Close* (Закреть) из меню *File* (Файл). Выход из приложения *ERwin* выполняется командой *File* \Rightarrow *Exit* (Файл \Rightarrow Выход).

Упражнение 1

1. Запустите программу *CA ERwin Data Modeler*.

2. Для создания новой модели в меню программы выберите команду *File* \Rightarrow *New* (Файл \Rightarrow Новый). В появившемся диалоговом окне *New Model* (Новая модель) установите переключатель *New Model Type* (Тип новой модели) (рис.) в положение *Logical / Physical* (Логический / Физический) и нажмите *OK*. В окне программы появится рабочая область для создания логической модели (рис. 6).

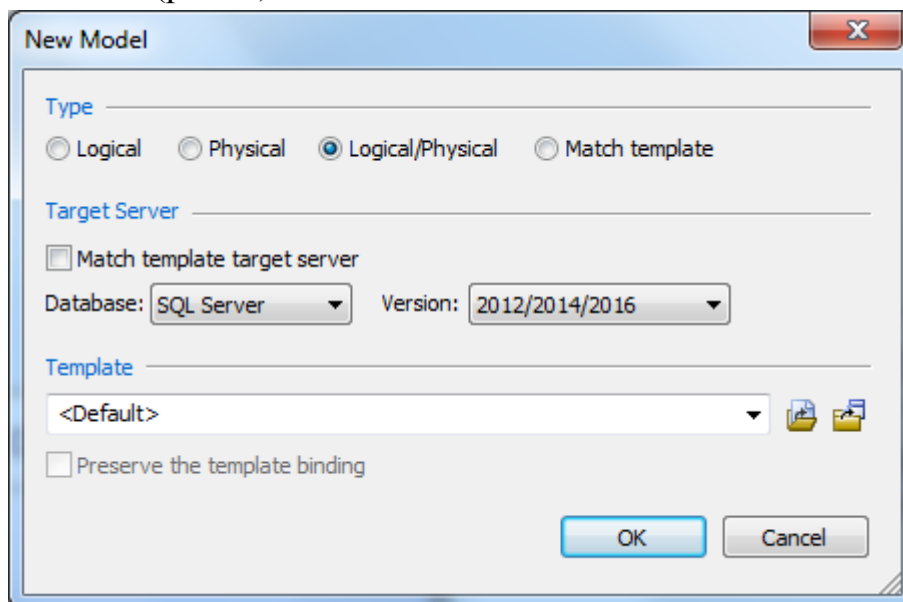


Рис. 6. Окно *New Model* (Новая модель)

Для перехода между логической и физической модели данных на панели инструментов существует соответствующий список (рис. 7).



Рис. 7. Список переключения между логической и физической моделями

Построение логической модели данных предполагает определение сущностей, их атрибутов и связей между ними. Для построения логической модели существует панель инструментов *ToolBox* (рис. 8), назначение кнопок которой представлено в табл. 1.



Рис. 8. Панель *ToolBox*


Таблица 1

Назначение кнопок панели инструментов *ToolBox*

Пиктограмма	Назначение
	Выбор элемента диаграммы
	Сущность
	Категория
	Идентифицирующая связь
	Идентифицирующая связь многие-ко-многим
	Неидентифицирующая связь
	Аннотация
	Ломаная линия

Упражнение 2

Добавьте сущности на диаграмму. Для этого:

1. Щелкните мышью по кнопке *Entity* (Сущность) , а затем – в поле проектирования в месте расположения сущности. В рабочем поле модели появится сущность с именем *E/1*, задаваемое по умолчанию (рис. 9). В момент вставки имя сущности выделено. Удалите его и введите *Магазины*. Обратите внимание, что справа в *Model Explorer* (Проводнике модели) в группе *Entities* (Сущности) появится сущность *Магазины*.

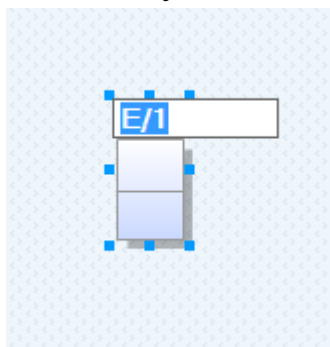


Рис. 9. Добавленная сущность *E/1*

2. Добавьте еще 4 сущности. Они будут названы: *E/2*, *E/3*, *E/4* и *E/5*.

3. Переименуйте добавленные сущности, назвав их *Сотрудники*, *Транспортные компании*, *Заказы*, *Книги*. Для изменения имени сущности или других ее свойств используется окно *Entity Editor* (Редактор сущности) (рис. 10), которое открывается вызовом в меню команды *Model* \Rightarrow *Entities* (Модель \Rightarrow Сущности) или выбором команды *Entity Properties* (Свойства сущности) из контекстного меню при нажатии правой кнопки мыши на объекте, или в *Model Explorer* (Проводнике модели) щелкнув правой кнопкой мыши по необходимому объекту и выбрав команду *Properties* (Свойства).

4. Сохраните модель.

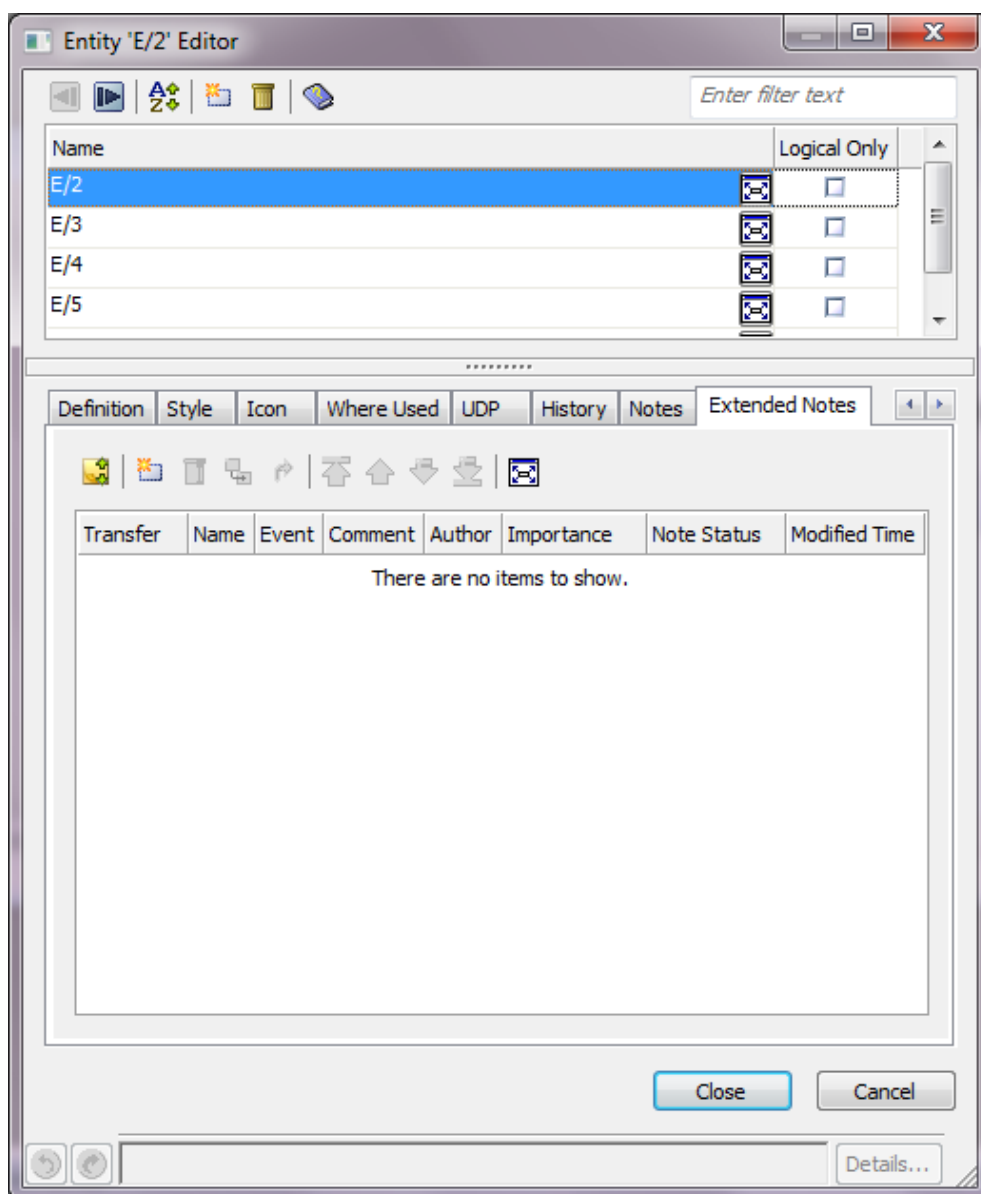



Рис. 10. Окно изменения свойств сущностей

Добавлять атрибуты сущностям можно двумя способами:

1. В контекстном меню сущности выбрать команду *Attribute Properties* (Свойства атрибутов) и в окне *Entity Attribute Editor* (Редактор атрибутов сущности) (рис. 11), следует нажать на кнопку *New* (Новый) . В появившейся строке, соответствующей добавленному атрибуту, изменить имя *<default>* на необходимое.

2. В *Model Explorer* (Проводнике модели), раскрыв объектный список необходимой сущности, следует щелкнуть правой кнопкой мыши по строке *Attributes* (Атрибуты) и из контекстного меню выбрать команду *New* (Новый). Затем следует переименовать появившейся атрибут *<default>*.

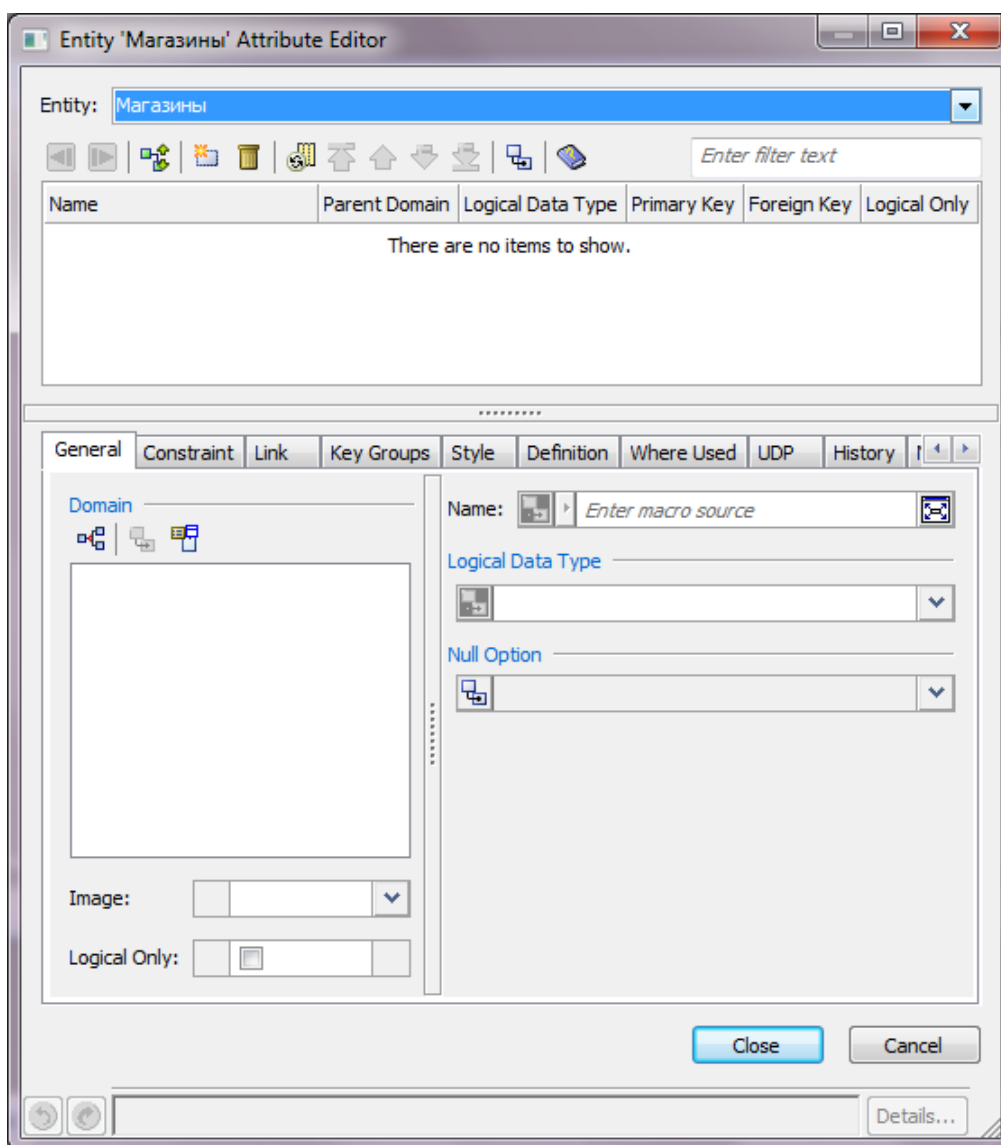


Рис. 11. Окно работы с атрибутами сущности

Упражнение 3

Добавьте атрибуты сущностям:

- *Магазины*: Код магазина, Название магазина, ФИО контактного лица, Город, Адрес, Телефон;
- *Сотрудники*: Код сотрудника, Фамилия, Имя, Отчество, Должность, Телефон, Дата рождения, Дата найма;
- *Транспортные компании*: Название компании, ФИО контактного лица, Город, Телефон;
- *Заказы*: Код заказа, Дата заказа, Требуемая дата выполнения, Фактическая дата выполнения;
- *Книги*: Код книги, Авторы, Название, Серия, Год издания, Количество страниц, Цена, Диск.

Результат показан на рис. 12.

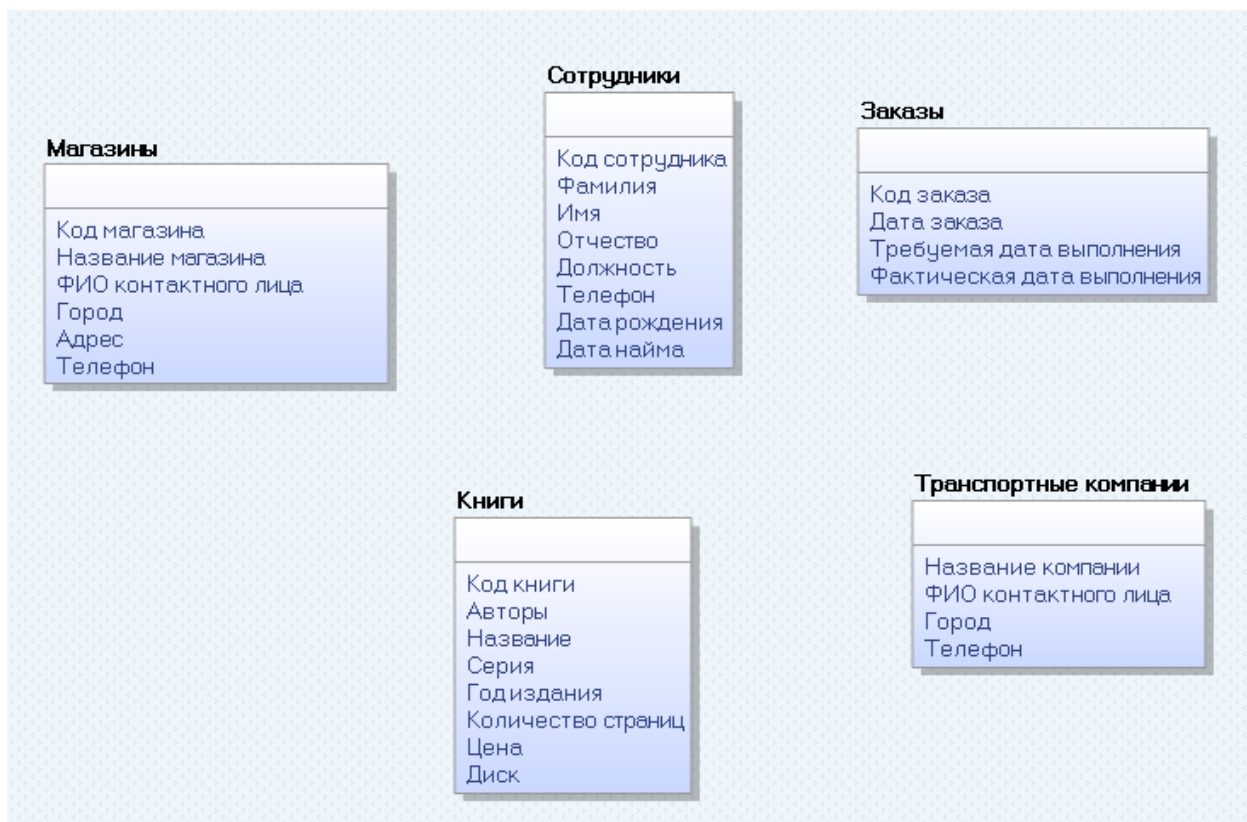
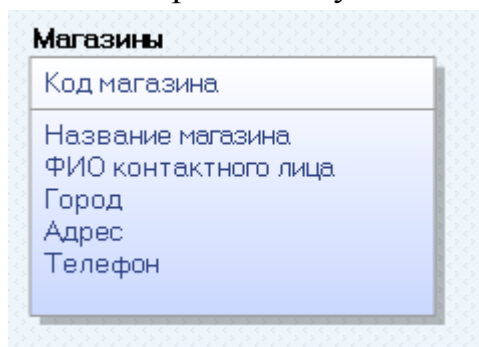


Рис. 12. Сущности и их атрибуты

Среди атрибутов сущности всегда существуют ключевые атрибуты, которые однозначно определяют значения других атрибутов. Для определения ключевых атрибутов сущности в окне *Entity Attribute Editor* (*Редактор атрибутов сущности*) надо установить галочку в поле *Primary Key* (*Первичный ключ*) соответствующему атрибуту. Ключевые атрибуты указываются в верхней части изображения сущности (рис. 13).

Рис. 13. Изображение сущности с ключевым атрибутом *Код магазина*

Упражнение 4

Установите свойства атрибутов сущностей. Для этого:

1. Щелкните правой кнопкой мыши по сущности *Магазины* и в контекстном меню выберите команду *Attributes Properties* (*Свойства атрибутов*).

2. В появившемся окне *Entity 'Магазины' Attribute Editor (Редактор атрибутов сущности Магазины)* выделите строку с атрибутом *Код магазина*, установите в поле *Primary Key (Первичный ключ)* галочку. Рядом с названием появится пиктограмма с изображением ключа. Нажмите в окне *Close (Заккрыть)*. На диаграмме атрибут *Код магазина* мигрирует в верхнюю часть столбца сущности *Магазины* (рис. 13). Это означает, что атрибут *Код магазина* стал ключевым.

3. Вновь откройте окно *Entity 'Магазины' Attribute Editor*. В списке атрибутов выделите строку *Код магазина*. Для данного атрибута на вкладке *General (Общие)* в поле *Domain (Домен)* установите галочку в поле *Number*, а затем в списке *Logical Date Type (Тип данных)* – *Integer (Целый)*.

4. Установите свойства другим атрибутам сущности *Магазины* в соответствии с табл. 2.

Таблица 2

Описание таблицы *Магазины*

Имя поля	Домен	Тип данных	Свойства
Код магазина	Number	Integer	Ключевое поле
Название магазина	String	VarChar(30)	Not Null
ФИО контактного лица	String	VarChar(50)	Not Null
Город	String	VarChar(25)	Not Null
Адрес	String	VarChar(70)	Not Null
Телефон	String	Char(10)	

5. Аналогично задайте свойства остальным сущностям в соответствии с табл. 3-6.

Таблица 3

Описание таблицы *Сотрудники*

Имя поля	Домен	Тип данных	Свойства
Код сотрудника	Number	Integer	Ключевое поле
Фамилия	String	VarChar(20)	Not Null
Имя	String	VarChar(20)	Not Null
Отчество	String	VarChar(20)	Not Null
Должность	String	VarChar(25)	Not Null
Телефон	String	Char(10)	
Дата рождения	Datetime	Date	Not Null
Дата найма	Datetime	Date	Not Null

Таблица 4

Описание таблицы *Транспортные компании*

Имя поля	Домен	Тип данных	Свойства
Название компании	String	VarChar(5)	Ключевое поле
ФИО контактного лица	String	VarChar(50)	Not Null
Город	String	VarChar(25)	Not Null
Телефон	String	Char(10)	

Таблица 5

Описание таблицы *Заказы*


Имя поля	Домен	Тип данных	Свойства
Код заказа	Number	Integer	Ключевое поле
Дата заказа	Datetime	Date	Not Null
Требуемая дата выполнения	Datetime	Date	Not Null
Фактическая дата выполнения	Datetime	Date	

Таблица 6

Описание таблицы *Книги*

Имя поля	Домен	Тип данных	Свойства
Код книги	Number	Integer	Ключевое поле
Авторы	String	VarChar(70)	Not Null
Название	String	VarChar(100)	Not Null
Серия	String	VarChar(30)	
Год издания	Number	Integer	
Количество страниц	Number	Integer	
Цена	Number	Money	Not Null
Диск	Number	Boolean	

Создание связей между сущностями**Упражнение 5**

1. Для создания связи между сущностями *Транспортные компании* и *Заказы* на панели инструментов нажмите на кнопку *Non-identifying relationship* (неидентифицирующая связь) . Затем щелкните по родительской сущности *Транспортные компании*, а потом – по дочерней сущности *Заказы*. Ключевой атрибут *Название компании* родительской

сущности *Транспортные компании* мигрирует в дочернюю сущность *Заказы* и станет внешним ключом (рис. 14).

2. Для настройки свойств связи щелкните правой кнопкой мыши по ней и в контекстном меню выберите команду *Properties* (*Свойства*).

3. В появившемся окне *Relationship Editor* (*Редактор связи*) (рис. 15) установите *Name* (*Имя*) *Доставляют*. На вкладке *General* (*Общие*) в секции *Cardinality Properties* (*Свойства кардинальности*) установите тип *Cardinality* (*Мощность*) – *One or More* (1 или более).

4. Для неидентифицирующей связи необходимо указать обязательность связи (*Nulls Allowed* или *Nulls Not Allowed*). В случае обязательной связи (*Nulls Not Allowed*), несмотря на то, что внешний ключ не войдет в состав первичного ключа дочерней сущности, при генерации схемы базы данных атрибут внешнего ключа получит признак (*Not Null*). В случае необязательной связи (*Nulls Allowed*) внешний ключ может принимать значение (*Null*). Это означает, что экземпляр дочерней сущности не будет связан ни с одним экземпляром родительской сущности. Для связи *Доставляют* в окне *Relationship Editor* (*Редактор связи*) (рис.) установите значение свойства *Null Option* – *Nulls Not Allowed*, т.е. связь является обязательной.

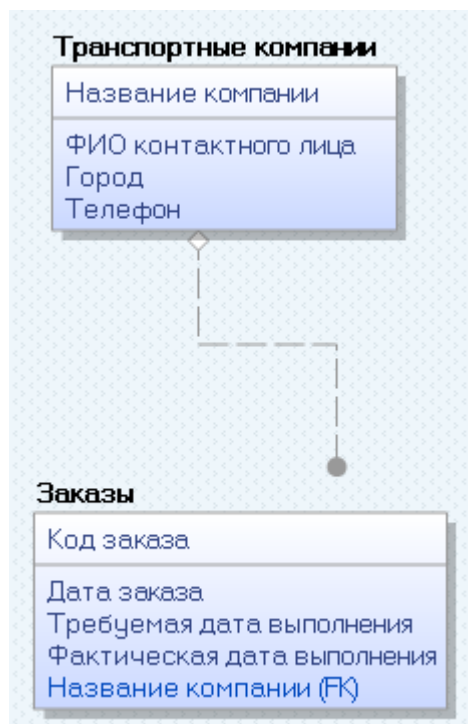


Рис. 14. Неидентифицирующая связь между сущностями *Транспортные компании* и *Заказы*

5. По умолчанию имя связи на диаграмме не показывается. Для отображения имени щелкните дважды левой кнопкой мыши по любому свободному месту диаграммы и в появившемся окне *ER Diagram Editor*

(Редактор ER-диаграммы) на вкладке *Relationship* (Связь) установите галочки в поля *Display Logical Relationship Name* (Отображать имя логической связи) и *Display Physical Relationship Name* (Отображать имя физической связи). Закройте окно с помощью кнопки *Close* (Заккрыть).

Результат показан на рис. 16.

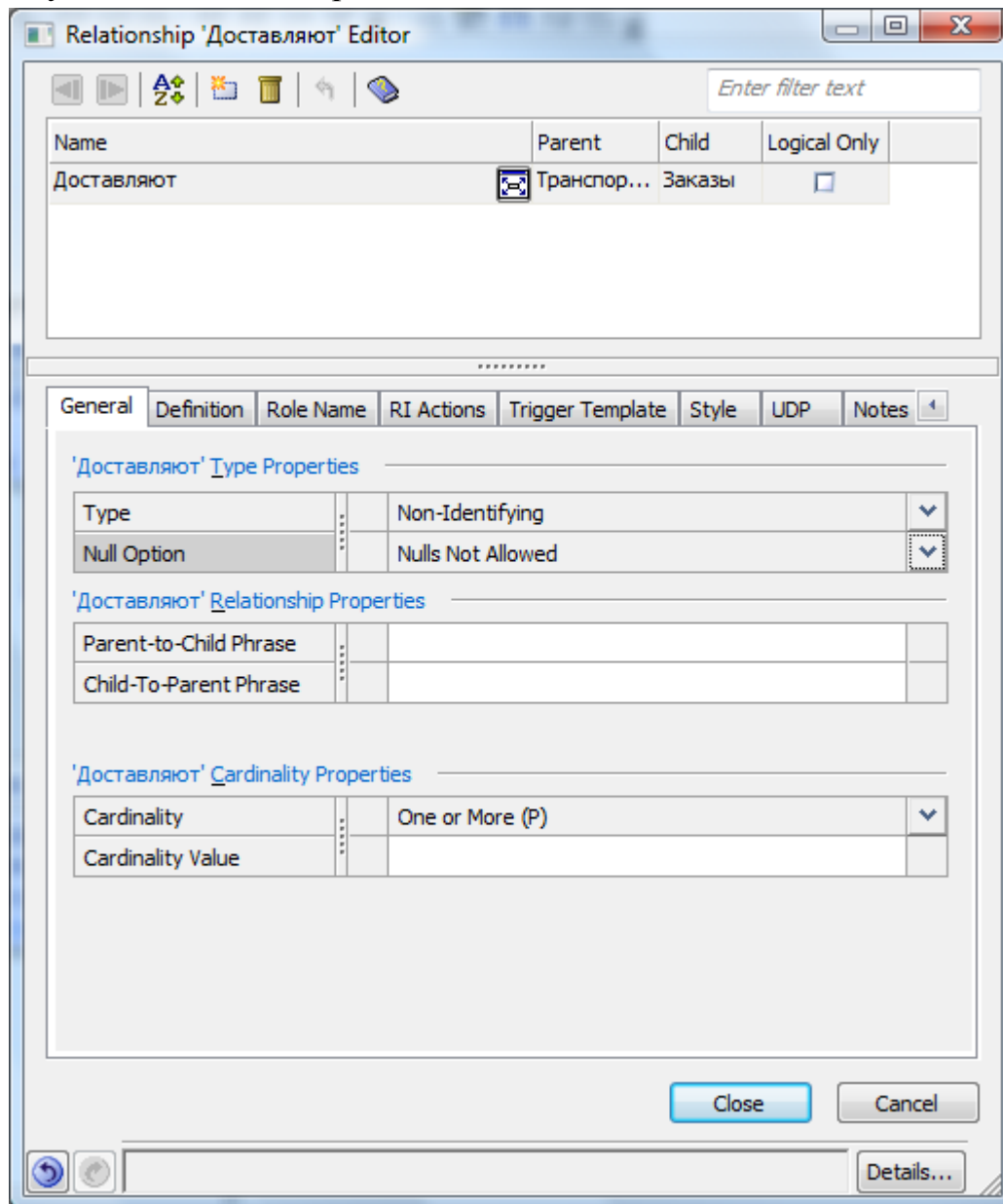


Рис. 15. Окно *Relationship Editor* (Редактор связи)

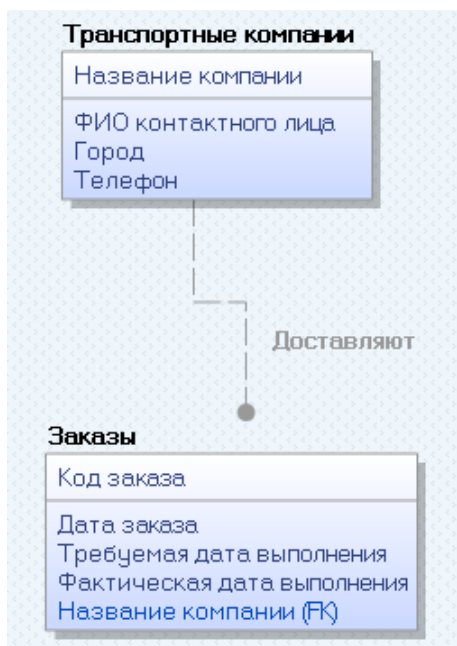


Рис. 16. Неидентифицирующая обязательная связь между сущностями *Транспортные компании* и *Заказы*

Упражнение 6

1. Установите неидентифицирующие обязательные связи между сущностями:

- *Делают* – между *Магазины* и *Заказы*;
- *Оформляют* – между *Сотрудники* и *Заказы*.

Задайте им необходимую мощность.

2. Установите связь *Входят / Включают* типа *многие-ко-многим* между объектами *Книги* и *Заказы*.

Результат представлен на рис. 17.

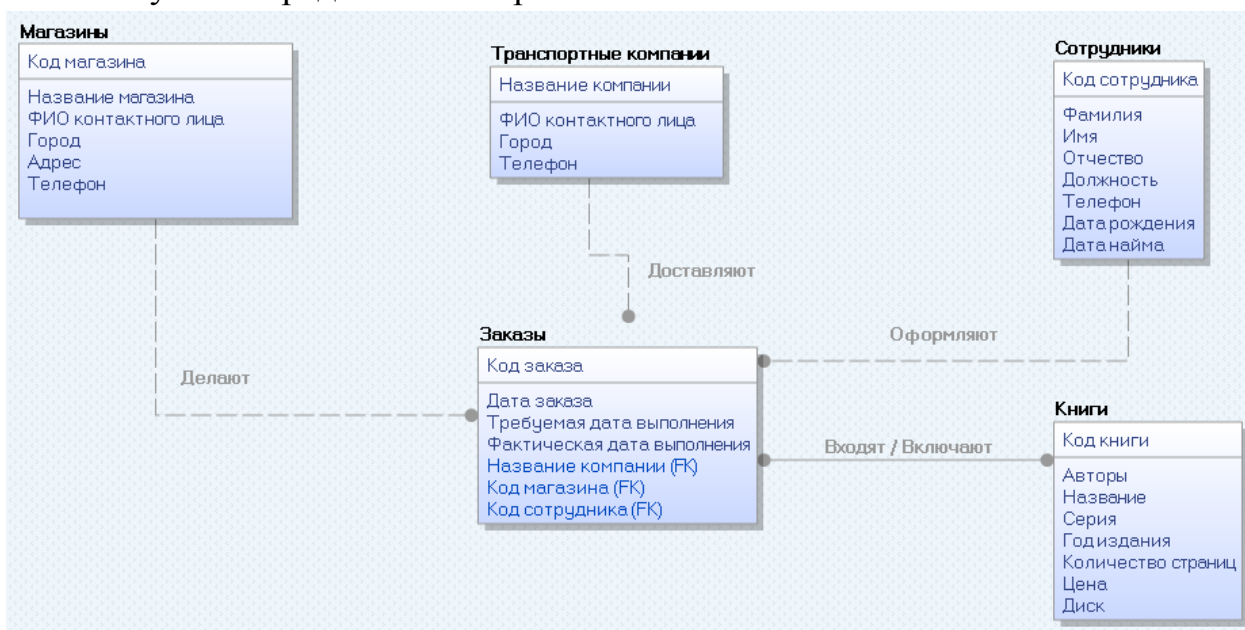


Рис. 17. Результирующая диаграмма

Обеспечение целостности данных

Правила ссылочной целостности (*referential integrity (RI)*) – логические конструкции, которые выражают бизнес-правила использования данных и представляют собой правила вставки, замены и удаления. Определение правил ссылочной целостности осуществляется с помощью вкладки *RI Actions* в окне *Relationship Editor (Редактор связей)* (рис. 18). Заданные на вкладке *RI Actions* опции логической модели используются при генерации схемы базы данных для определения правил ссылочной целостности, которые предписываются для каждой связи, и триггеров, обеспечивающих ссылочную целостность. Триггеры представляют собой программы, выполняемые всякий раз при реализации команд вставки, замены или удаления (*INSERT*, *UPDATE* или *DELETE*).

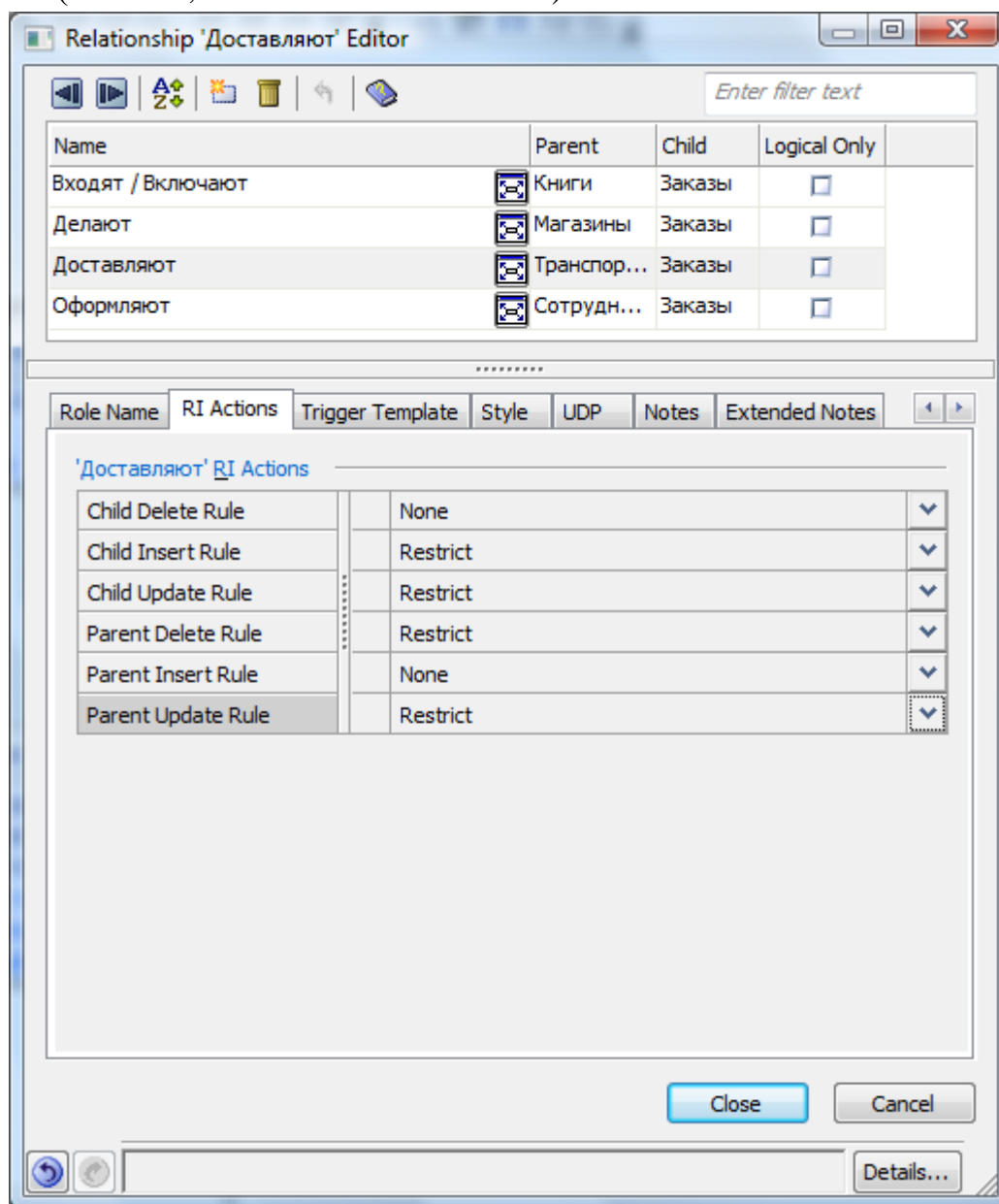


Рис. 18. Вкладка *RI Actions* окна *Relationship Editor (Редактор связей)*

Возможными значениями правил для удаления, вставки, изменения родительской и дочерней сущности могут быть:

- *None* (Нет) – целостность не нарушается, проверка не выполняется;
- *No Action* (Нет опции);
- *Restrict* (Ограничить) – связь нарушается, изменение запрещено;
- *Cascade* (Каскадом) – например, изменение внешних ключей дочерних сущностей приводит к изменению ключа родительской сущности;
- *Set Null* (значение *Null*) – атрибуту внешнего или потенциального ключа присваивается значение *Null*. Такая ситуация возможна, если связь является неидентифицирующей необязательной;
- *Set Default* (значение по умолчанию) – атрибуту внешнего или потенциального ключа присваивается значение по умолчанию.

Упражнение 7

Для неидентифицирующих связей установите правила ссылочной целостности:

- правило *Child Delete Rule* – значение *None*, т.е. при удалении экземпляра дочерней сущности ссылочная целостность не нарушается;
- правило *Child Insert Rule* – значение *Restrict*, т.е. при вставке дочерней сущности необходимо проверить совпадение ее внешнего ключа ключу родительской сущности иначе нарушиться ссылочная целостность;
- правило *Child Update Rule* – значение *Restrict*, которое не допускает присвоения внешнему ключу дочерней сущности значения, отличного от значения родительского ключа;
- правило *Parent Delete Rule* – значение *Restrict*, т.е. экземпляр родительской сущности нельзя удалить, если с ней связан экземпляр дочерней сущности;
- правило *Parent Insert Rule* – значение *None*, т.е. при вставке родительской сущности ссылочная целостность не нарушается;
- правило *Parent Update Rule* – значение *Restrict*, т.е. при изменении ключевых атрибутов родительской сущности нарушится связь с дочерними сущностями, так как внешний ключ дочерних сущностей не равен ключу родительской сущности. Поэтому такое изменение запрещено.

Проанализируйте заданные правила.

Связь *многие-ко-многим* может быть создана только на уровне логической модели. Нотация *IDEFIX* требует, чтобы на физическом уровне связь *многие-ко-многим* была преобразована, так как реляционные СУБД не поддерживают данный тип связи.

ERwin позволяет реализовать принудительное преобразование связи *многие-ко-многим*, которое включает создание новой (связывающей) таблицы и двух новых связей от старых таблиц к новой таблице.

Упражнение 8

1. С помощью ниспадающего списка переключитесь на физическую модель. В результате связь *Входят / Включают* заменится на сущность *Книги_Заказы* с атрибутами *Код_книги*, *Код_заказа* с двумя идентифицирующими связями между сущностями *Заказы* и *Книги_Заказы* и *Книги* и *Книги_Заказы* (рис. 19).

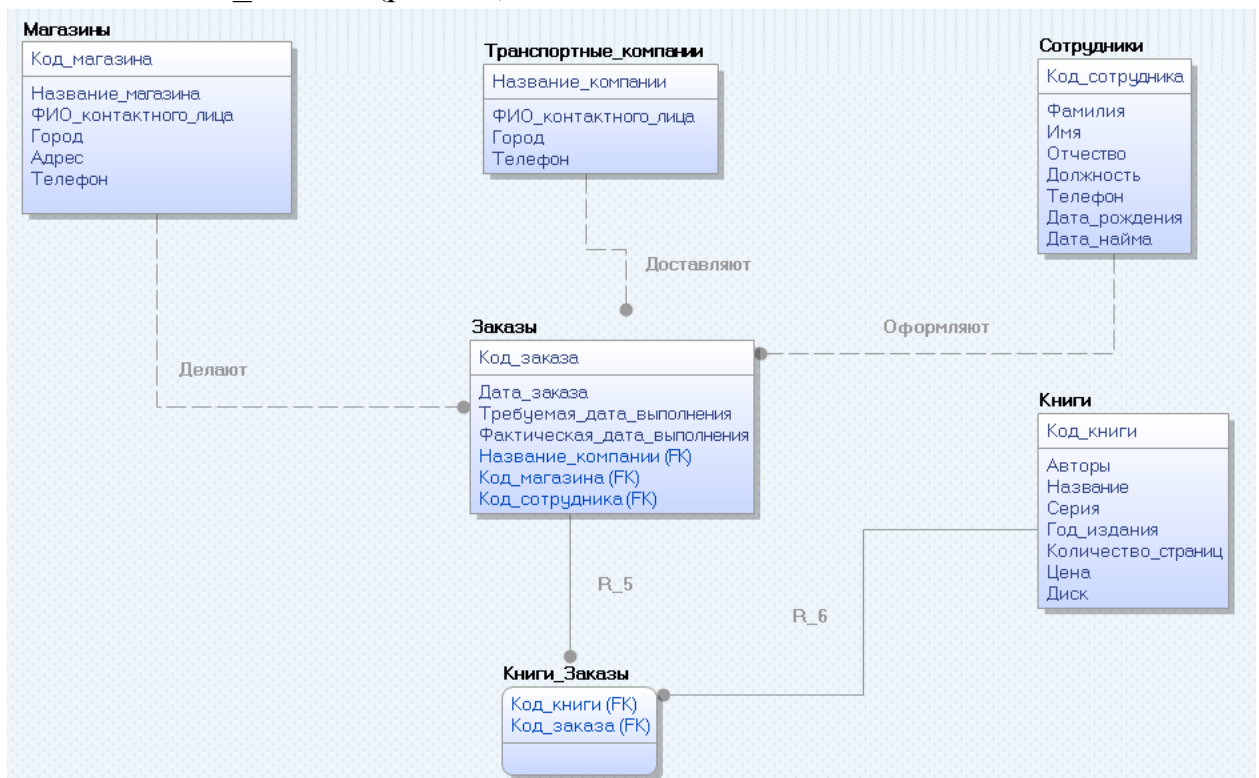


Рис. 19. Диаграмма физической модели данных

2. Вновь переключитесь на логическую модель, добавьте сущность *Описание заказов*, удалите связь *Входят / Включают* и установите обязательные идентифицирующие связи:

- *Входят* – между сущностями *Книги* и *Описание заказов*;
- *Включают* – между сущностями *Заказы* и *Описание заказов*.

3. Добавьте собственные атрибуты *Количество* и *Скидка* сущности *Описание заказов*. Установите им необходимые свойства.

Результат представлен на рис. 20.

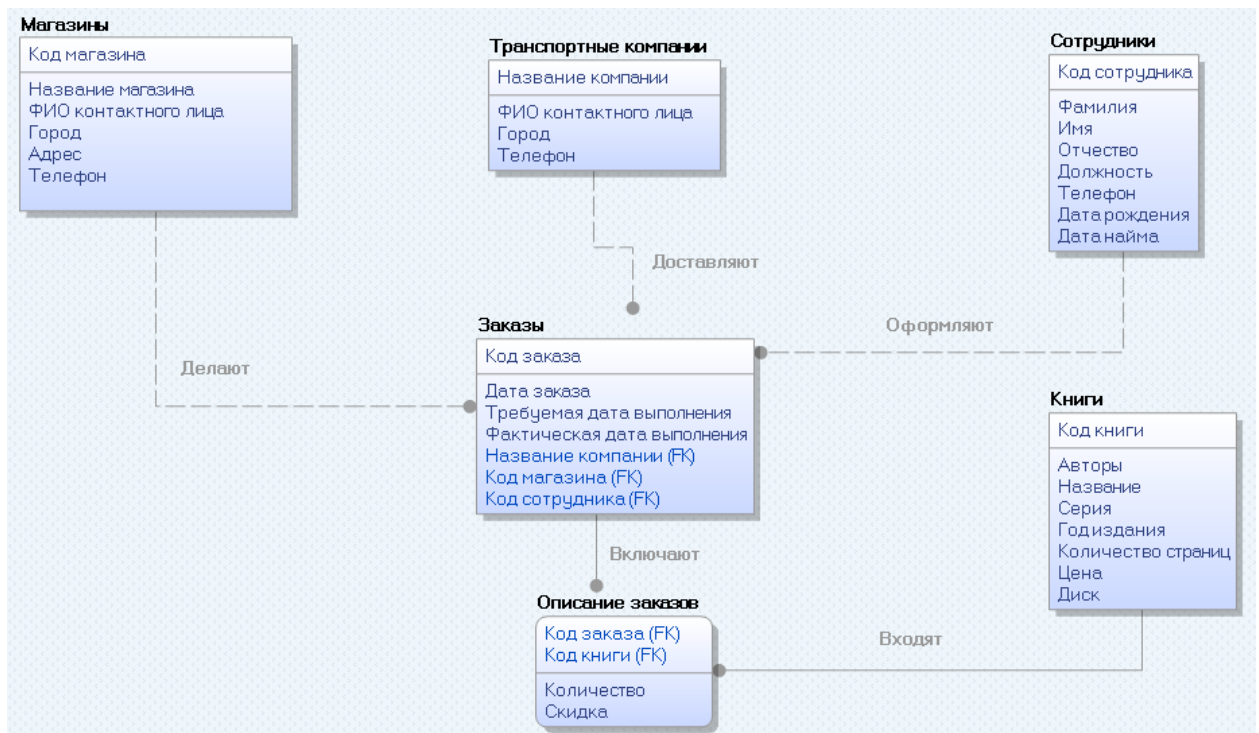


Рис. 20. Результирующая логическая модель

Контрольные вопросы

1. Для чего предназначена программа *ERwin Data Modeler*?
2. Какие уровни представления моделей поддерживает *ERwin*?
3. Модель «сущность-связь» и ее компоненты.
4. Назначение методологии *IDEFIX*.
5. Типы связей в методологии *IDEFIX*. Их свойства: обязательная/необязательная; мощность связи.
6. Характеристика категориальной связи в методологии *IDEFIX*.
7. Типы подуровней логического уровня модели данных в *ERwin*?
8. Как создать сущности в *ERwin*?
9. Как добавить атрибуты сущностям? Какие характеристики можно им задать.
10. Расскажите правила ссылочной целостности данных.

Лабораторная работа № 10

Тема работы: Создание физической модели данных в *ERwin Data Modeler*.

Логический уровень модели данных является универсальным и никак не связан с конкретной реализацией СУБД. Одному и тому же логическому уровню модели могут соответствовать несколько разных физических уровней различных моделей, где описывается вся информация о конкретных физических объектах: таблицах, колонках, индексах, процедурах и т.д.

В случае автоматического перехода от логической к физической модели *ERwin* генерирует имена таблиц и колонок по умолчанию на основе имен соответствующих сущностей и атрибутов логической модели, учитывая максимальную длину имени и другие синтаксические ограничения, накладываемые выбранной СУБД. При этом правила ссылочной целостности, принятые на логическом уровне модели данных системы сохраняются.

Таблицы физической модели данных определяются на основе семантического анализа предметной области. При этом каждой сущности предметной области ставится в соответствие таблица, атрибутам соответствуют поля таблицы.

Упражнение 1

Запустите *ERwin Data Modeler* и откройте файл с логической моделью данных «Поставки книг в магазины», созданный в предыдущей лабораторной работе.

На панели инструментов в раскрывающемся списке, соответствующем типу модели, выберите *Physical* (Физическая). В *Model Explorer* (Проводнике модели) раскройте список *Tables* (Таблицы), физическая модель должна содержать 6 таблиц: *Заказы*, *Книги*, *Магазины*, *Описание_заказов*, *Сотрудники*, *Транспортные_компании*, соответствующих одноименным сущностям логической модели данных.

Обратите внимание, что в именах таблиц и полей, состоящих из нескольких слов, вместо пробела в именах сущностей стоит знак подчеркивания (_).

На диаграмме модели щелкните правой кнопкой мыши на изображении таблицы *Магазины* и в контекстном меню выберите команду *Column Properties* (Свойства полей). Откроется окно *SQL Server Table 'Магазины' Column 'Код магазина' Editor* (рис. 21), в котором приведен список полей таблицы *Магазины*. Свойства полей таблицы соответствуют установленным свойствам атрибутов сущностей логической модели.

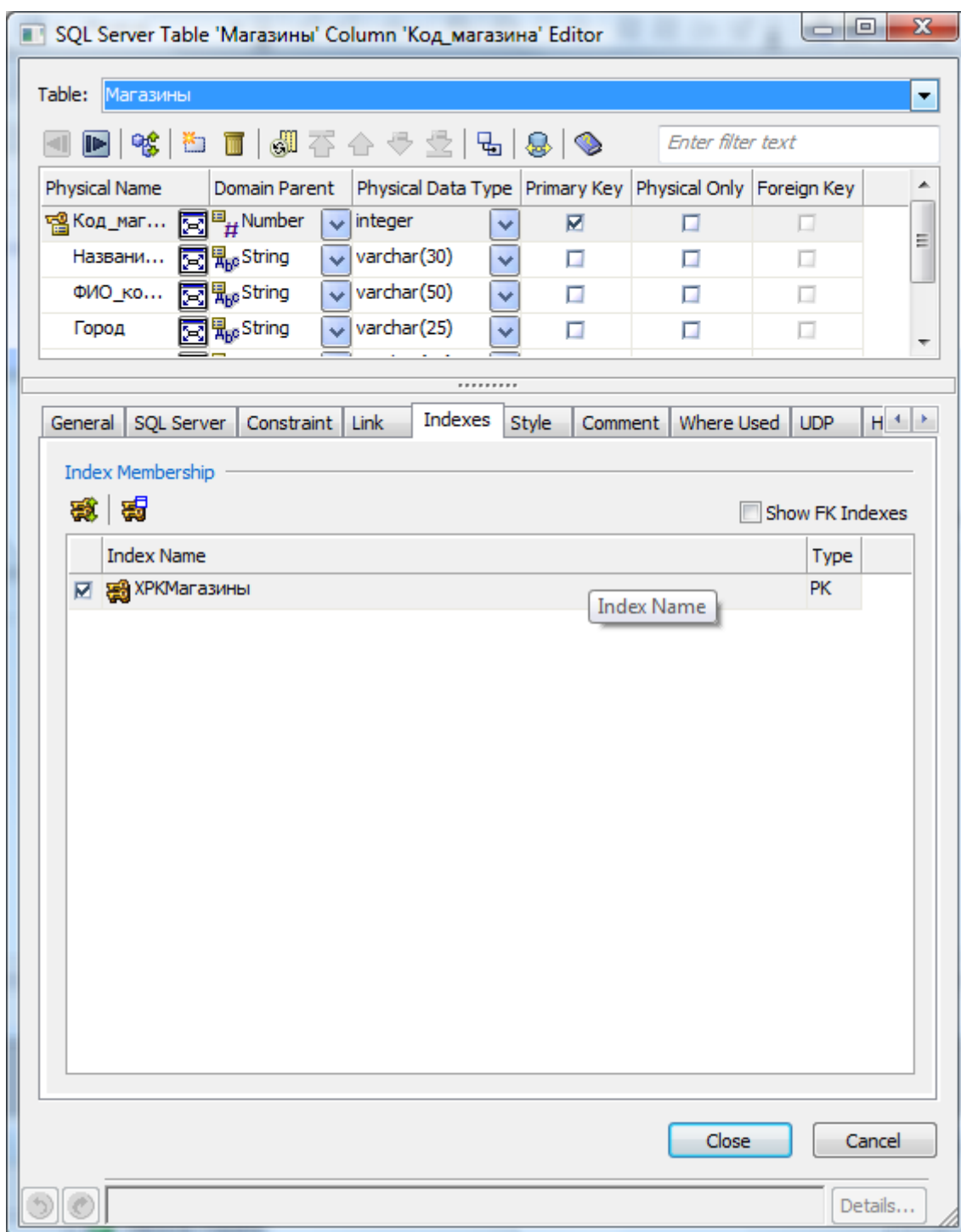


Рис. 21. Окно *SQL Server Table 'Магазины' Column 'Код магазина' Editor*

Нормализация данных

Нормализация – это процесс приведения структуры базы данных к форме, обладающей лучшими свойствами при включении, изменении и удалении данных. Цель нормализации заключается в получении проекта базы данных, в котором каждый атрибут хранится только в одном месте, т.е. исключена избыточность информации. Кроме задачи более эффективного использования памяти, нормализация позволяет снизить угрозу нарушения целостности базы данных из-за появления в ней внутренних противоречий.

Нормальная форма – совокупность требований, которым должно удовлетворять отношение реляционной базы данных. Отношение находится в данной нормальной форме, если оно удовлетворяет указанным ограничениям.

Различают:

- первая нормальная форма (1НФ);
- вторая нормальная форма (2НФ);
- третья нормальная форма (3НФ);
- нормальная форма Бойса-Кодда (НФБК);
- четвёртая нормальная форма (4НФ);
- пятая нормальная форма (5НФ).

Отношение находится в *первой нормальной форме* (1НФ) если все его атрибуты имеют простые (атомарные) значения.

Отношение находится во *второй нормальной форме* (2 НФ), если оно находится в первой нормальной форме, и каждый неключевой атрибут функционально полно зависит от первичного ключа.

Вторая нормальная форма применяется к отношениям с составными ключами. Отношение, у которого первичный ключ включает только один атрибут, всегда находится во 2НФ.

Отношение находится в *третьей нормальной форме* (3НФ) только в том случае, если оно находится во второй нормальной форме, и каждый неключевой атрибут не транзитивно зависит от первичного ключа.

Транзитивной зависимостью неключевых атрибутов от ключевых называется зависимость следующего типа: если $A \rightarrow B$ и $B \rightarrow C$, где A – набор ключевых атрибутов (ключ), B и C – различные множества неключевых атрибутов.

Отношение находится в *нормальной форме Бойса-Кодда* в том случае, если любая функциональная зависимость между ее полями сводится к полной функциональной зависимости от возможного ключа.

Ситуация, когда отношение будет находиться в 3НФ, но не в НФБК, возникает при условии, что отношение имеет два (или более) возможных ключа, которые являются составными и имеют общий атрибут. На практике такая ситуация встречается достаточно редко.

Отношение находится в *четвертой нормальной форме* (4НФ), если оно находится в нормальной форме Бойса-Кодда, и в нем отсутствуют многозначные зависимости, не являющиеся функциональными зависимостями.

Отношение находится в *пятой нормальной форме* (5НФ) только тогда, когда в каждой ее полной декомпозиции все проекции содержат возможный

ключ. Таблица, не имеющая ни одной полной декомпозиции, также находится в пятой нормальной форме.

При решении практических задач в большинстве случаев третья нормальная форма является достаточной. Процесс проектирования реляционной базы данных, как правило, заканчивается приведением к 3НФ.

Упражнение 2

Проанализируйте таблицы разработанной модели данных «Поставки книг в магазины». В каких нормальных формах они находятся?

Индексирование

Для повышения скорости извлечения данных в базах данных используется механизм *индексирования*. Индекс содержит отсортированную по полю или нескольким полям информацию и указывает на строки, в которых хранится конкретное значение поля.

Использование индексов позволяет существенно повысить эффективность информационных систем по обработке хранимых данных. Индекс подобен содержанию книги, которое указывает на все номера страниц, посвященных конкретной теме. Например, если необходимо найти сотрудника по фамилии, то можно создать индекс по полю *Фамилия* таблицы *Сотрудник*. В индексном поле фамилии сотрудников будут отсортированы в алфавитном порядке. Для фамилии индекс будет содержать ссылку, указывающую, в каком месте таблицы хранится эта строка. Индекс можно создать для всех полей таблицы, по которым часто производится поиск. Поскольку значения в индексе хранятся в определенном порядке, то просматривать нужно гораздо меньший объем данных, что значительно уменьшает время выполнения запроса.

При генерации физической базы данных *ERwin* автоматически, по умолчанию создает отдельный индекс на основе первичного ключа каждой таблицы, а также на основе всех альтернативных ключей, внешних ключей и инверсных входов, поскольку эти поля наиболее часто используются для поиска данных.

Первичный ключ (Primary Key) – однозначно идентифицирует экземпляр сущности.

Альтернативный ключ (Alternate Key) – это потенциальный ключ, не выбранный в качестве первичного.

Внешний ключ (Foreign Key) – создается в дочерней сущности при внесении в диаграмму связи и включает в себя мигрирующие атрибуты родительской сущности.

Инверсные входы (Inversion Entry) – атрибуты или группы атрибутов, не определяющие экземпляр сущности уникальным образом, но часто используемые для обращения к экземплярам сущности.


Для повышения производительности системы *ERwin* позволяет редактировать ключевые группы и создавать собственные индексы. При этом целесообразно выбирать индексы с различными полями и порядком сортировки, предварительно проанализировав наиболее часто выполняемые запросы.

ERwin автоматически генерирует имя индекса, созданного на основе ключа по принципу: «X» + имя ключа + имя таблицы, где имя ключа – «PK» для первичного ключа, «IFn» – для внешнего, «AKn» – альтернативного, «IEn» – для инверсного входа. Например, по умолчанию при создании таблицы *Магазины* будут созданы индекс *ХРКМагазины*, в состав которого войдет поле *Код магазина*.

Упражнение 3

В таблице *Магазины* сделайте поле *Название магазина* индексом. Для этого:

1. В области диаграммы щелкните правой кнопкой по таблице *Магазины* и выберите команду *Index Properties (Свойства индексов)*. Откроется окно *SQL Server Table 'Магазины' Index 'ХРКМагазины' Editor* (рис. 22), в котором можно изменить характеристики существующего индекса или создать новый индекс.

2. Для создания нового индекса щелкните по кнопке *New (Новый)* , появится всплывающее меню для выбора типа индекса: *New Unique Index (Новый индекс с уникальными значениями)* или *New Non-Unique Index (Новый индекс с повторяющимися значениями)*. Выберите создание нового индекса с повторяющимися значениями. Добавиться строка с названием индекса *XIE1Магазины*. Переименуйте его на *XIE1НазваниеМагазина*.

3. В нижней части окна в области *Index Members (Составляющие индекса)* щелкните по полю *Название магазина* и закройте окно свойств индекса.

Обратите внимание, что в изображении таблицы *Магазины* на диаграмме данных имя поля *Название_магазина* выделено черным цветом.

4. Создайте еще один индекс в таблице *Магазины* для поля *Город*.

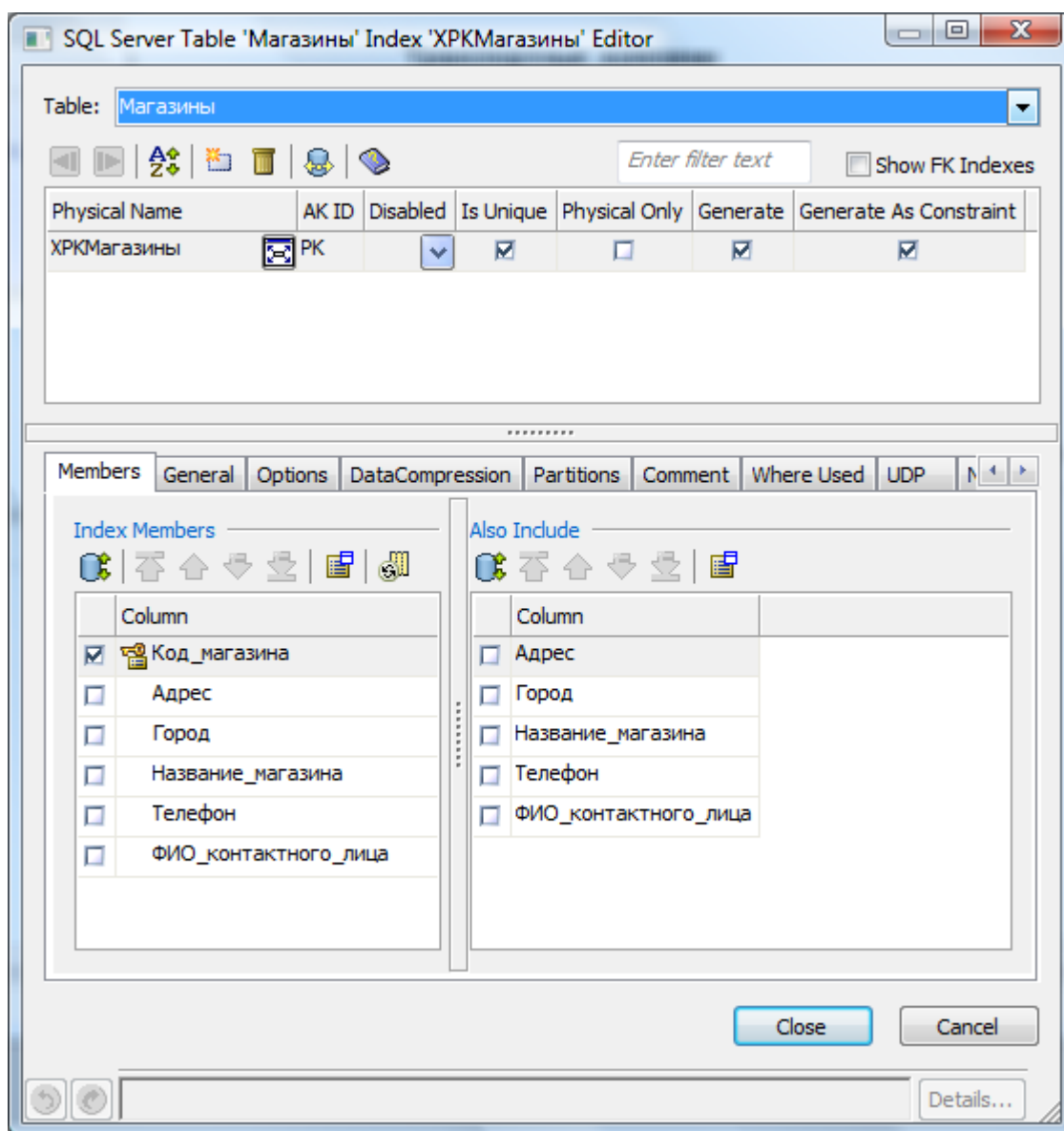


Рис. 22. SQL Server Table 'Магазины' Index 'ХРКМагазины' Editor

Упражнение 4


Для всех таблиц создайте индексные поля, продумав возможные запросы к данным.

Правила валидации полей

Правило валидации задает список допустимых значений для конкретного поля таблицы и/или правила допустимых значений. Значение по умолчанию – значение, которое нужно ввести в поле, если никакое другое значение не задано.

Упражнение 5

Задайте возможные значения поля *Город* таблицы *Магазины*. Для этого:


1. В меню *ERwin Data Modeler* выберите команду *Model ⇒ Validation Rules* (*Модель ⇒ Правила валидации*). Откроется окно *SQL Server Validation Rule Editor* (*Редактор правил валидации SQL Server*) (рис. 23), в котором щелкните по кнопке *New* (*Новый*) .

2. Появится строка, соответствующая новому правилу, в которую в ячейку *Physical name* (*Физическое имя*) введите *Город*. Свойство *Type* (*Тип*) позволяет выбрать тип правила:

- *User-Defined* – позволяет задать вручную фрагмент *SQL*-выражения, который соответствует правилу валидации и будет использоваться при генерации схемы базы данных;

- *Min/Max* – позволяет задать минимальное и максимальное значение поля, которое будет проверяться в базе данных на вхождение в заданный диапазон;

- *Valid Values List* – позволяет задать список допустимых значений.

3. Для свойства *Type* (*Тип*) выберите *Valid Values List*. В нижней части окна появится секция '*Город*' *Valid Values* (рис. 24), в которую добавьте возможные значения: Воронеж, Санкт-Петербург, Москва, Саратов, Самара. Отсортируйте их по алфавиту с помощью кнопки *Sort Items* (*Сортировать элементы*) .

4. Откройте окно редактора полей таблицы *Магазины* и выделите поле *Город*. На вкладке *Constraint* (*Ограничения*) снимите галочку с поля *Use Inherited Constraint* (*Использовать наследуемые ограничения*) и в списке *Validation* выберите *Город*.

5. Аналогично для поля *Город* таблицы *Транспортные_компании* установите правило валидации *Город*.

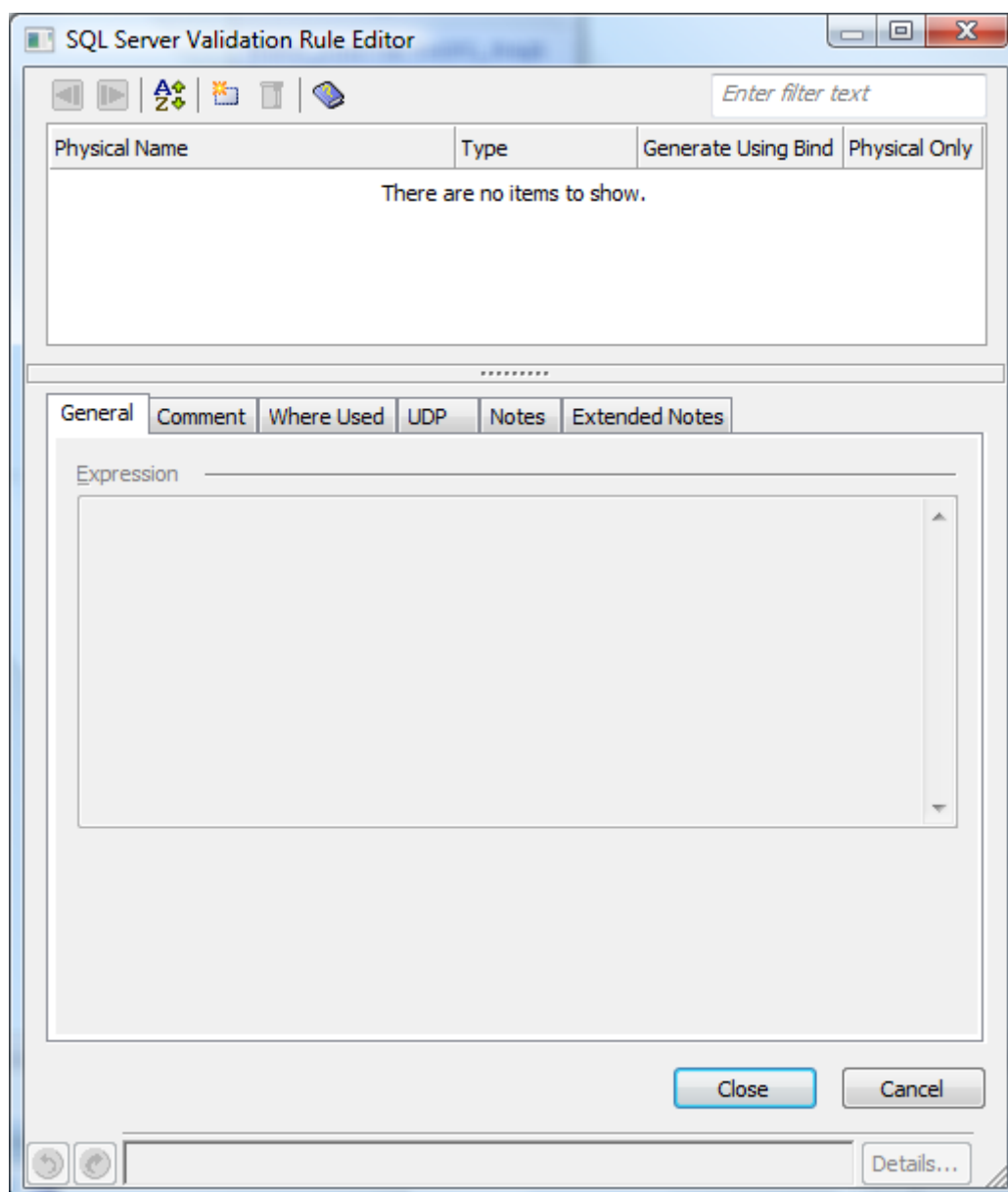


Рис. 23. Окно *SQL Server Validation Rule Editor*

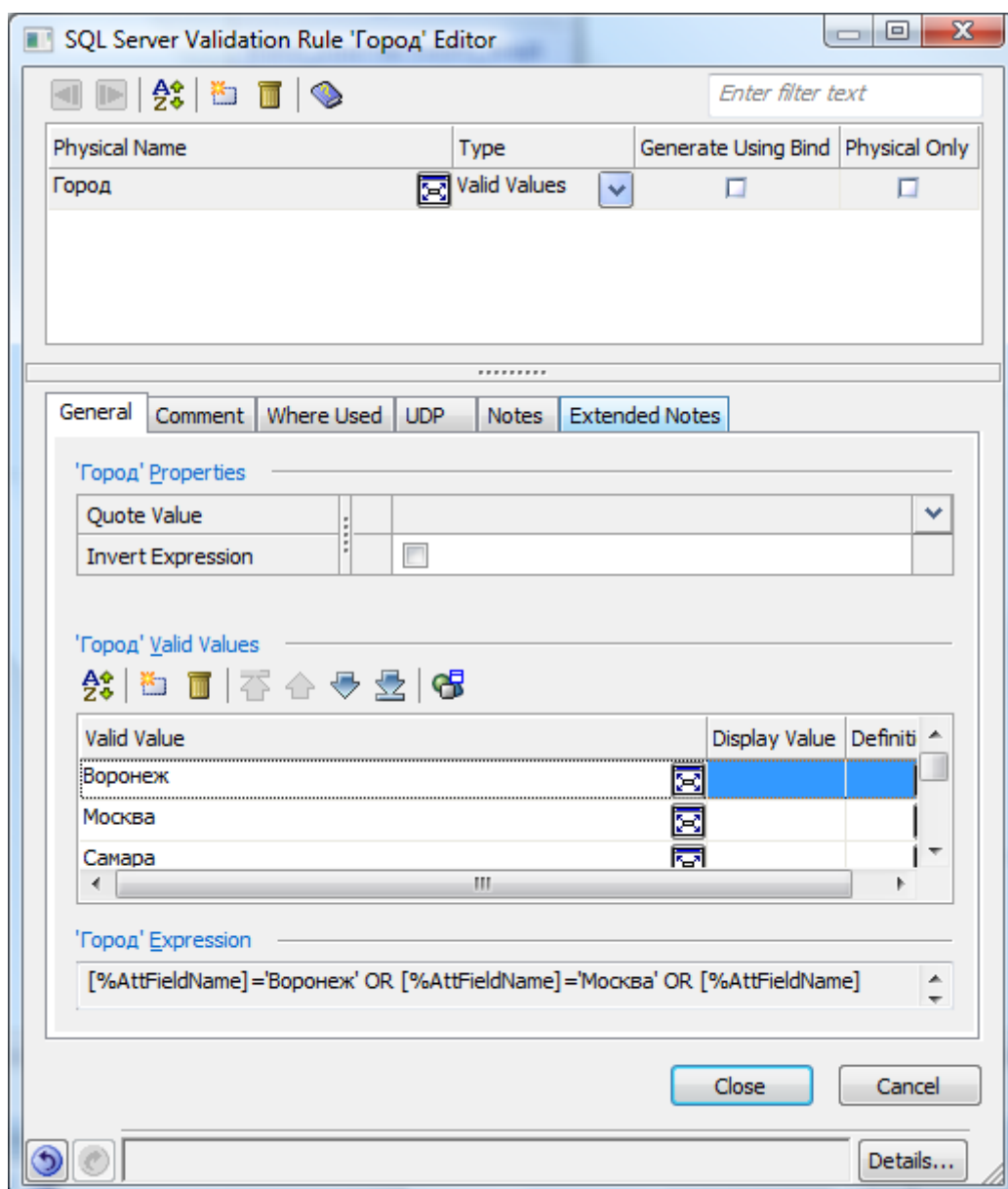



Рис. 24. Окно *SQL Server Validation Rule 'Город' Editor*


Прямое проектирование

Процесс генерации физической схемы БД из логической модели данных называется *прямым проектированием*. При генерации физической схемы *ERwin* включает триггеры ссылочной целостности, хранимые процедуры, индексы ограничения и другие возможности, доступные при определении таблиц в выбранной СУБД.

Процесс генерации логической модели из физической БД называется *обратным проектированием*. *ERwin* позволяет создать модель данных путем обратного проектирования имеющейся БД. После того как модель создана, можно переключиться на другой сервер (модель будет конвертирована) и произвести прямое проектирование структуры БД для другой СУБД.

Для генерации кода создания физической базы данных в меню следует выбрать команду *Actions* \Rightarrow *Forward Engineer* \Rightarrow *Schema* (Действия \Rightarrow Прямая разработка \Rightarrow Схема) или нажать на панели инструментов кнопку *Forward Engineer Schema Generation* (Прямая разработка схемы) .

Упражнение 6

1. На панели инструментов нажмите на кнопку *Forward Engineer Schema Generation* (Прямая разработка схемы) , откроется одноименное окно (рис. 25).

2. Для предварительного просмотра *SQL*-скрипта нажмите на кнопку *Preview* (Просмотр). Откроется окно (рис. 26), содержащее *SQL*-скрипт для создания спроектированной базы данных «Поставки книг в магазины».

3. Для генерации схемы в окне *SQL Server Schema Generation Preview* нажмите кнопку *Generate*. В процессе генерации *ERwin* связывается с выбранной базой данных, выполняя *SQL*-скрипт.

Если в процессе генерации возникнут какие-либо ошибки, то она прекращается, и откроется окно сообщений об ошибках.

4. При успешном выполнении *SQL*-скрипта будет создана база данных. Запустите *SQL*-сервер и убедитесь в этом.

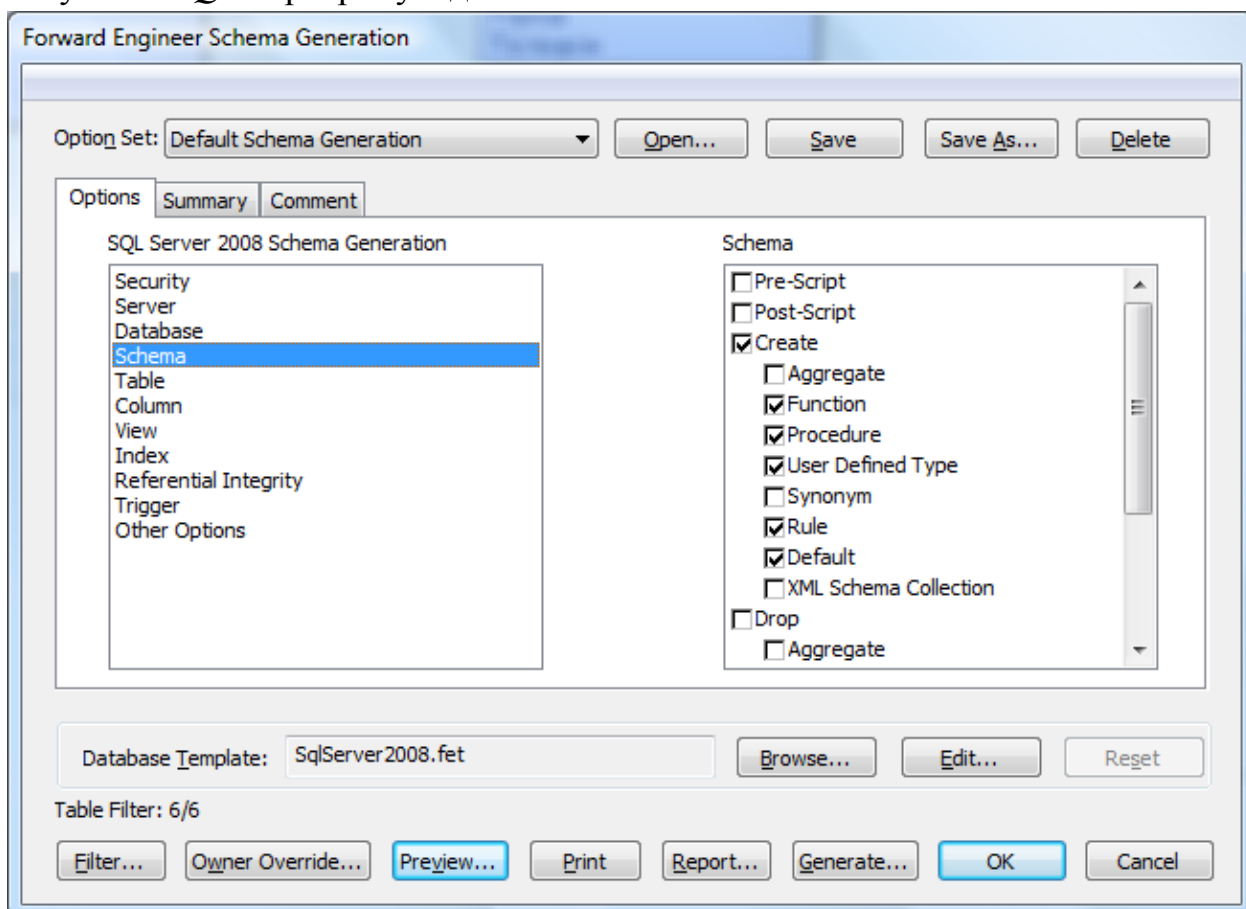


Рис. 25. Окно *Forward Engineer Schema Generation*

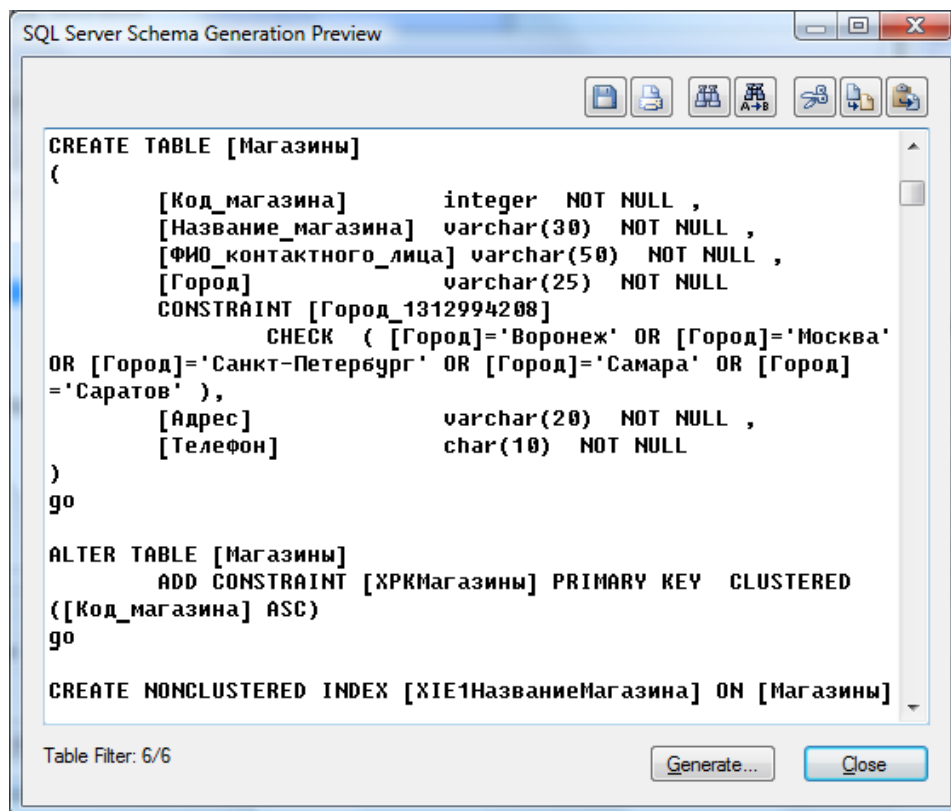


Рис. 26. Окно предварительного просмотра схемы генерации базы данных

Упражнение 7

Разработать логическую и физическую модель данных для автоматизации учета входящих и исходящих документов организации», имеющих следующее описание.

Входящие в организацию и исходящие из организации документы (письма) регистрируются. При этом входящее письмо решением начальника может быть направлено для исполнения одному или нескольким сотрудникам, а исходящее письмо готовят один или несколько сотрудников. Регистрация документов в настоящее время ведется в журнале, содержащем для исходящих писем следующие графы:

- Дата
- № письма (ставится отправляющей организацией)
- В ответ на входящее письмо № (номер был поставлен сторонней организацией)
- Кому (должность, Фамилия И.О.)
- Аннотация
- Кто подготовил письмо
- Его координаты (телефон, адрес электронной почты)

Для входящей документации графы следующие:

- Дата

- № письма (ставится сторонней организацией)
- В ответ на входящее письмо № (номер был поставлен своей организацией)
- От кого (должность, Фамилия И.О.)
- Аннотация
- Кому отписано для исполнения
- Его координаты (телефон, адрес электронной почты)
- Уровень срочности, важности
- Дата контроля

Упражнение 8

Разработать логическую и физическую модель данных для автоматизации товарного учета торговой организации. Первичные документы, используемые в элементарном складском учете, и их поля приведены в табл. 7.

Таблица 7

Первичные документы товарного учета и их атрибуты

Документ	Атрибуты
Карточка учета товара (содержит также графы, перечисленные ниже)	Название Артикул Модель Марка Единицы измерения Расфасовка Срок годности Цена поступления
Приход товара	Товар Дата прихода Дата изготовления Поставщик Количество
Расход товара	Товар Дата расхода Кому Количество Цена расхода
Переоценка	Товар Дата переоценки Количество Старая цена расхода Новая цена расхода

Контрольные вопросы

1. Что такое нормализация данных? Дайте характеристику нормальным формам.
2. Что такое индексирование базы данных? Какие типы индексов можно создать в *ERwin Data Modeler*?
3. Для чего используются правила валидации полей? Как их задать в *ERwin Data Modeler*?
4. Дайте определения прямому и обратному проектированию базы данных.

Индивидуальное задание № 3

Тема работы: Разработка логической и физической моделей данных для заданной предметной области в *ERwin Data Modeler*.

Для заданной предметной области необходимо разработать логическую модель данных, содержащую не менее 4 сущностей. Для атрибутов сущностей задать тип данных, связям задать ограничения целостности данных.

На основе спроектированной логической модели данных разработать физическую модель данных для SQL Server. Необходимо проверить ее соответствие как минимум третьей нормальной форме. В таблицах задать индексные поля и правила валидации различного типа.

Отчет должен содержать модели и их описание.