

ФГБОУ ВО «Воронежский государственный  
технический университет»

Кафедра систем автоматизированного проектирования  
и информационных систем

## МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к лабораторным работам по теме «CASE-средства проектирования баз данных»  
по дисциплине «Базы данных» для студентов направления 09.03.02 «Информаци-  
онные системы и технологии» очной формы обучения



Воронеж 2021

Составители: канд. техн. наук О.Г. Яскевич,  
ст. преп. Д.В. Иванов

УДК 681.38+681.3

Методические указания к лабораторным работам по теме «CASE-средства проектирования баз данных» по дисциплине «Базы данных» для студентов направления 09.03.02 «Информационные системы и технологии» очной формы обучения / ФГБОУ ВО «Воронежский государственный технический университет»; сост. О.Г. Яскевич, Д.В. Иванов. Воронеж, 2021. 65 с.

Методические указания содержат краткие теоретические и практические сведения об основных конструкциях языках SQL и их практического применения.

Методические указания подготовлены в электронном виде в текстовом редакторе MS Word 2003 и содержатся в файле БД.CASE 1.0.pdf.

Табл. 11. Библиогр.: 4 назв.

Рецензент д-р техн. наук, проф. К.А. Разинкин

Ответственный за выпуск зав. кафедрой  
д-р техн. наук, проф. Я.Е. Львович

Издается по решению редакционно-издательского совета Воронежского государственного технического университета

© ФГБОУ ВПО «Воронежский государственный технический университет», 2021

## Лабораторные работы CASE-средства проектирования баз данных

### Краткие теоретические сведения

#### Нормализация

**Нормализация** – процесс проверки и реорганизации сущностей и атрибутов с целью удовлетворения требований к реляционной модели данных. Нормализация позволяет быть уверенным, что каждый атрибут определен для своей сущности, значительно сократить объем памяти для хранения информации и устранить аномалии в организации хранения данных. В результате проведения нормализации должна быть создана структура данных, при которой информация о каждом факте хранится только в одном месте. Процесс нормализации сводится к последовательному приведению структуры данных к **нормальным формам** – формализованным требованиям к организации данных. Известны шесть нормальных форм:

- 1) первая нормальная форма (1NF);
- 2) вторая нормальная форма (2NF);
- 3) третья нормальная форма (3NF);
- 4) нормальная форма Бойса-Кодда (усиленная 3NF);
- 5) четвертая нормальная форма (4NF);
- 6) пятая нормальная форма (5NF).

На практике обычно ограничиваются приведением данных к третьей нормальной форме (полная атрибутивная модель, 3NF).

Нормальные формы основаны на понятии функциональной зависимости (в дальнейшем будет использоваться термин «зависимость»).

**Функциональная зависимость (FD).** Атрибут В сущности Е функционально зависит от атрибута А сущности Е тогда и только тогда, когда каждое значение А и Е связало с ним точно одно значение В и Е, т.е. А однозначно определяет В.

**Полная функциональная зависимость.** Атрибут В сущности Е полностью функционально зависит от ряда атрибутов А сущности Е тогда и только тогда, когда В функционально зависит от А и не зависит ни от какого подряда А. Функциональные зависимости определяются бизнес правилами предметной области.

Нормальная форма — свойство отношения в реляционной модели данных, характеризующее его с точки зрения избыточности, потенциально приводящей к

логически ошибочным результатам выборки или изменения данных. Нормальная форма определяется как совокупность требований, которым должно удовлетворять отношение.

Процесс преобразования отношений базы данных (БД) к виду, отвечающему нормальным формам, называется нормализацией. Нормализация предназначена для приведения структуры БД к виду, обеспечивающему минимальную логическую избыточность, и не имеет целью уменьшение или увеличение производительности работы или же уменьшение или увеличение физического объёма базы данных. Конечной целью нормализации является уменьшение потенциальной противоречивости хранимой в базе данных информации.

Устранение избыточности производится, как правило, за счёт декомпозиции отношений таким образом, чтобы в каждом отношении хранились только первичные факты (то есть факты, не выводимые из других хранимых фактов).

### Первая нормальная форма (1NF)

Сущность находится в первой нормальной форме тогда и только тогда, когда все атрибуты содержат атомарные значения. Среди атрибутов не должно встречаться повторяющихся групп, т.е. несколько значений для каждого экземпляра. Другой ошибкой нормализации является хранение в одном атрибуте разных по смыслу значений.

Для приведения сущности к первой нормальной форме следует:

- 1) разделить сложные атрибуты на атомарные;
- 2) создать новую сущность;
- 3) перенести в нее все «повторяющиеся» атрибуты;
- 4) выбрать возможный ключ для нового РК (или создать новый РК);
- 5) установить идентифицирующую связь от прежней сущности к новой, РК прежней сущности станет внешним ключом (FK) для новой сущности.

### Пример

Исходная не нормализованная (то есть не являющаяся правильным представлением некоторого отношения) таблица:

<b>ФИО</b>	<b>Данные</b>
Иванов Иван Иванович	ПП-119 АСУ Муж. 19.01.1990
Петров Петр Петрович	Э-119 Электронщики Муж. 1991
Васильева Катерина Ильинишна	Прикладная Информатика 1990 Жен ПК-129

Таблица, приведенная к 1NF (являющаяся правильным представлением некоторого отношения):

Фамилия	Имя	Отчество	Пол	Дата рождения	Группа	Специальность
Иванов	Иван	Иванович	Муж.	19.01.1990	ПП-119	АСУ
Петров	Петр	Петрович	Муж.	20.03.1991	Э-119	Электронщики
Васильева	Катерина	Ильинишна	Жен.	17.04.1990	ПК-129	Прикладная Информатика

### Вторая нормальная форма (2NF)

Сущность находится во второй нормальной форме, если она находится в первой нормальной форме, и каждый неключевой атрибут полностью зависит от первичного ключа (не должно быть зависимости от части ключа). Вторая нормальная форма имеет смысл только для сущностей, имеющих сложный первичный ключ.

Для приведения сущности ко второй нормальной форме следует:

- 1) выделить атрибуты, которые зависят только от части первичного ключа, создать новую сущность;
- 2) поместить атрибуты, зависящие от части ключа, в их собственную (новую) сущность;
- 3) установить идентифицирующую связь от прежней сущности к новой.

### Пример

Исходная таблица в 1 нормальной форме представлена выше.

Ниже приведены таблицы во второй нормальной форме.

Таблица Данные о студентах

ID_Stud	Фамилия	Имя	Отчество	Пол	Дата рождения
1	Иванов	Иван	Иванович	Муж.	19.01.1990
2	Петров	Петр	Петрович	Муж.	20.03.1991
3	Васильева	Катерина	Ильинишна	Жен.	17.04.1990
4	Петров	Илья	Петрович	Муж.	20.05.1991

Таблица Список групп

ID_Group	Группа
1	ПП-119
2	Э-119

3	ПК-129
---	--------

Таблица Специальности

ID_Спец	Специальность
1	АСУ
2	Электронщики
3	Прикладная информатика

### Третья нормальная форма (3NF)

Сущность находится в третьей нормальной форме, если она находится во второй нормальной форме и никакой неключевой атрибут не зависит от другого неключевого атрибута (не должно быть взаимозависимости между неключевыми атрибутами).

Для приведения сущности к третьей нормальной форме следует:

- 1) создать новую сущность и перенести в нее атрибуты с одной и той же зависимостью от неключевого атрибута;
- 2) использовать атрибут(ы), определяющий эту зависимость, в качестве первичного ключа новой сущности;
- 3) установить неидентифицирующую связь от новой сущности к старой.

В третьей нормальной форме каждый атрибут сущности зависит от ключа, от всего ключа целиком и ни от чего другого, кроме как от ключа.

### Пример

В результате приведения к 3NF получаются три отношения:

Таблица Данные студентов

ID_Student	Фамилия	Имя	Отчество	Пол	Дата рождения	ID_Gruppa
1	Иванов	Иван	Иванович	Муж.	19.01.1990	1
2	Петров	Петр	Петрович	Муж.	20.03.1991	2
3	Васильева	Катерина	Ильинишна	Жен.	17.04.1990	3
4	Петров	Илья	Петрович	Муж.	20.05.1991	2

Таблица список групп

ID_Grup	Группа	ID_Specialnost
1	ПП-119	1

2	Э-119	2
3	ПК-129	3
4	Э-129	2

Таблица Специальности

ID_Спец	Специальность
1	АСУ
2	Электронщики
3	Прикладная информатика

## Введение в моделирование баз данных

Erwin – современное средство проектирование баз данных (БД), позволяет проводить процессы прямого и обратного проектирования БД. Это означает, что по модели данных можно сгенерировать схему БД или автоматически создать модель данных на основе информации системного каталога. Для создания моделей данных в Erwin используются две методологии: IDEF1X и IE. В данной работе рассматривается методология IDEF1X.

Первым шагом при создании логической модели БД является построение диаграммы ERD (Entity Relationship Diagram). ERD-диаграммы состоят из трех частей: сущностей, атрибутов и взаимосвязей. Сущностями являются существительные, атрибуты - прилагательными или модификаторами, взаимосвязи - глаголами.

ERD-диаграмма позволяет рассмотреть систему целиком и выяснить требования, необходимые для ее разработки, касающиеся хранения информации.

ERD-диаграммы можно подразделить на отдельные куски, соответствующие отдельным задачам, решаемым проектируемой системой. Это позволяет рассматривать систему с точки зрения функциональных возможностей, делая процесс проектирования управляемым.

Как известно основным компонентом реляционных БД является таблица. Таблица используется для структуризации и хранения информации. В реляционных БД каждая ячейка таблицы содержит одно значение. Кроме Того, внутри одной БД существуют взаимосвязи между таблицами, каждая из которых заставит совместное пользование данными таблицы.

ERD-диаграмма графически представляет структуру данных проектируемой информационной системы. Сущности отображаются при помощи прямоугольников, содержащих имя. Имена принято выражать существительными в единственном числе, взаимосвязи - при помощи линий, соединивших отдельные сущности. Взаимосвязь показывает, что данные одной сущности ссылаются или связаны с данными другой.

### Отображение модели данных в ERWin

ERwin имеет два уровня представления модели – логический и физический.

Логический уровень – это абстрактный взгляд на данные, на нем данные представляются так, как выглядят в реальном мире, и могут называться так, как они называются в реальном мире, например, «Фамилия сотрудника», «Отдел». Объекты модели, представляемые на логическом уровне, называются сущностями и атрибутами. Логическая модель может быть построена на основе другой логической модели, например на основе модели процессов. Логическая модель данных является универсальной и никак не связана с конкретной реализацией СУБД.

Физическая модель данных, напротив, зависит от конкретной СУБД, фактически являясь отображением системного каталога. В физической модели содержится информация о всех объектах БД. Поскольку стандартов на объекты БД не существует, физическая модель зависит от конкретной реализации СУБД. Следовательно, одной и той же логической модели могут соответствовать несколько разных физических моделей. Разделение модели данных на логические и физические позволяет решить несколько важных задач.

Документирование модели. На физическом уровне объекты БД могут называться так, как того требуют ограничения СУБД. На логическом уровне можно этим объектам дать синонимы – имена более понятные неспециалистам, в том числе на кириллице и с использованием специальных символов.

### Основные элементы

Для переключения между логической и физической моделью данных служит список выбора в левой части панели инструментов **ERwin**. При переключении, если физической модели еще не существует, то она будет создана автоматически.

Палитра инструментов выглядит различно на разных уровнях отображения модели. На логическом уровне палитра инструментов имеет (рис.1):



Рис.1 – Палитра инструментов на логическом уровне



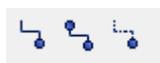
**кнопка указателя (режим мыши)** – в этом режиме можно установить фокус на каком-либо объекте модели;



**кнопка внесения сущности** – для внесения сущности нужно щелкнуть левой кнопкой мыши по кнопке внесения сущности и один раз по свободному пространству на модели. Повторный щелчок приведет к внесению в модель еще одной новой сущности. Для редактирования сущностей или других объектов модели необходимо перейти в режим указателя;



**кнопка категории.** Категория, или категориальная связь, - специальный тип связи между сущностями. Для установления категориальной связи нужно щелкнуть левой кнопкой мыши по кнопке категории, затем один раз щелкнуть по сущности-родовому предку, затем – по сущности-потомку;



**кнопки создания связей**

идентифицирующую, «многие-ко-многим» и неидентифицирующую.



Рис.2 – Элементы для рисования

На этой панели расположены различные элементы для рисования (прямоугольник, круг, линия). Предназначены для создания вспомогательных элементов на схеме.

На физическом уровне палитра инструментов имеет:



Рис.3 – Палитра инструментов на физическом уровне



**кнопка указателя (режим мыши)** – в этом режиме можно установить фокус на каком-либо объекте модели;



**кнопка внесения сущности** – для внесения сущности нужно щелкнуть левой кнопкой мыши по кнопке внесения сущности и один раз по свободному пространству на модели. Повторный щелчок приведет к внесению в модель еще одной новой сущности. Для редактирования сущностей или других объектов модели необходимо перейти в режим указателя;



**кнопка внесения представлений (view);**



**кнопка внесения таблицы (view);**



**кнопки создания связей:**

идентифицирующая, связь представлений, неидентифицирующая.

Для создания моделей данных в ERwin можно использовать две нотации: **IDEF1X** и **IE (Information Engineering)**. Переключение между нотациями можно сделать в закладке **Notation** диалога **Model Properties** (меню **Model/ Model Properties ...**) рис. 4.

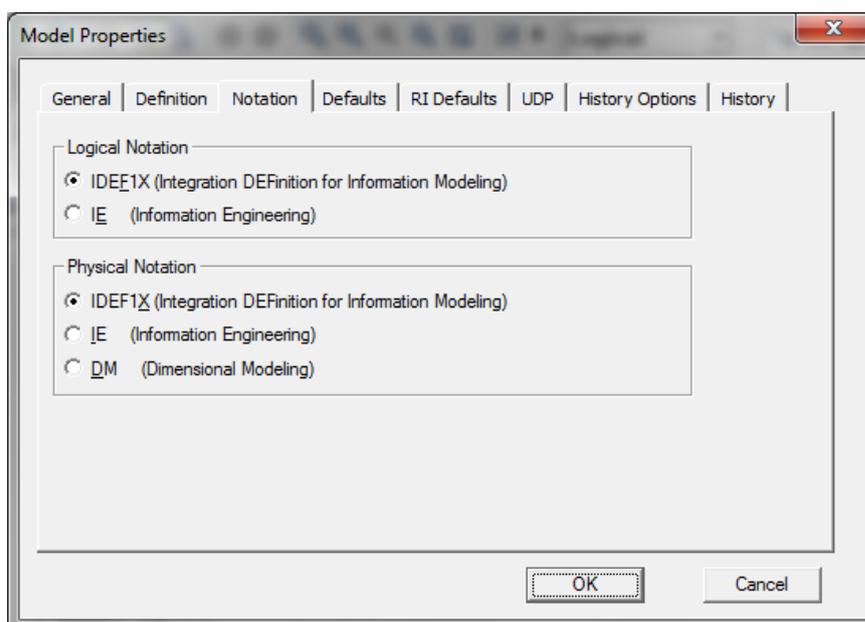


Рис.4. Переключение между нотациями

В данной работе используется нотация **IDEF1X**.

ERwin имеет несколько уровней отображения диаграммы: уровень сущностей, уровень атрибутов, уровень определений, уровень первичных ключей и уровень иконок. Переключиться между первыми тремя уровнями можно с использованием кнопок панели инструментов. Переключиться на другие уровни отображения можно при помощи контекстного меню, которое появляется,

если “кликнуть” по любому месту диаграммы, не занятому объектами модели. В контекстном меню следует выбрать пункт **Display Level** и затем необходимый уровень отображения (рис. 5).

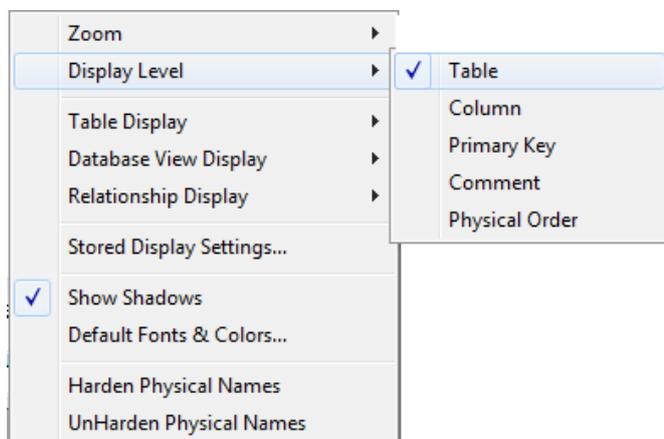


Рис. 5. Выбор уровня отображения

### Уровни логической модели

Различают три уровня логической модели, отличающихся по глубине представления информации о данных:

1. диаграмма сущность-связь (**Entity Relationship Diagram (ERD)**);
2. модель данных, основанная на ключах (**Key Based model (KB)**);
3. полная атрибутивная модель (**Fully Attributed model (FA)**).

**Диаграмма сущность - связь** - представляет собой модель данных верхнего уровня. Она включает сущности и взаимосвязи, отражающие основные бизнес-правила предметной области. Диаграмма сущность- связь может включать связи многие-ко-многим и не включать описание ключей.

**Модель данных, основанная на ключах** - более подробное представление данных. Она включает описание всех сущностей и первичных ключей и предназначена для представления структуры данных и ключей, которые соответствуют предметной области.

**Полная атрибутивная модель** – наиболее детальное представление структуры данных: представляет данные в третьей нормальной форме и включает все сущности, атрибуты и связи.

### Сущность и атрибуты

Основные компоненты диаграммы ERwin – это сущности, атрибуты и связи. Каждая сущность является множеством подобных индивидуальных объектов,

называемых экземплярами. Каждый экземпляр индивидуален и должен отличаться от всех остальных экземпляров. Атрибут выражает определенное свойство объекта. С точки зрения БД (физическая модель) сущности соответствует таблица, экземпляру сущности – строка в таблице, а атрибуту – колонка таблицы.

Построение модели данных предполагает определение сущностей и атрибутов, т.е. необходимо определить, какая информация будет храниться в конкретной сущности или атрибуте. Сущность можно определить как объект, событие или концепцию, информация о которой должна сохраняться. Сущности должны иметь наименование с четким смысловым значением.

Фактически имя сущности дается по имени ее экземпляра. Примером может быть сущность *Студент* (но не *Студенты!*) с атрибутами *Номер Студента*, *Фамилия Студента*. На уровне физической модели ей может соответствовать таблица *Student* с колонками *Student \_number*, *Student \_name*.

Для внесения сущности в модель необходимо убедиться, что вы находитесь на уровне логической модели, и «кликнуть» по кнопке сущности на панели инструментов (**ERwin Toolbox**), затем «кликнуть» по тому месту на диаграмме, где необходимо расположить новую сущность. Щелкнув правой кнопкой мыши по сущности и выбрав из всплывающего меню пункт **Entity Properties**, можно вызвать диалог **Entities**, в котором определяются имя, описание и комментарии сущности.

Каждая сущность должна быть полностью определена с помощью текстового описания в закладке **Definition** (рис.6).

Закладки **Note**, **Note 2**, **Note 3** служат для внесения дополнительных комментариев и определений к сущности:

**Definition** - используется для ввода определения сущности.

**Note** - можно ввести полезное замечание.

**Note 2** - можно задокументировать некоторые возможные запросы.

**Note 3** - позволяет вводить примеры данных для сущности.

**Icon** - каждой сущности можно поставить в соответствие изображение.

Для определения **UDP** служит диалог **User-Defined Properties** ( меню **Model/UDP dictionary...**)

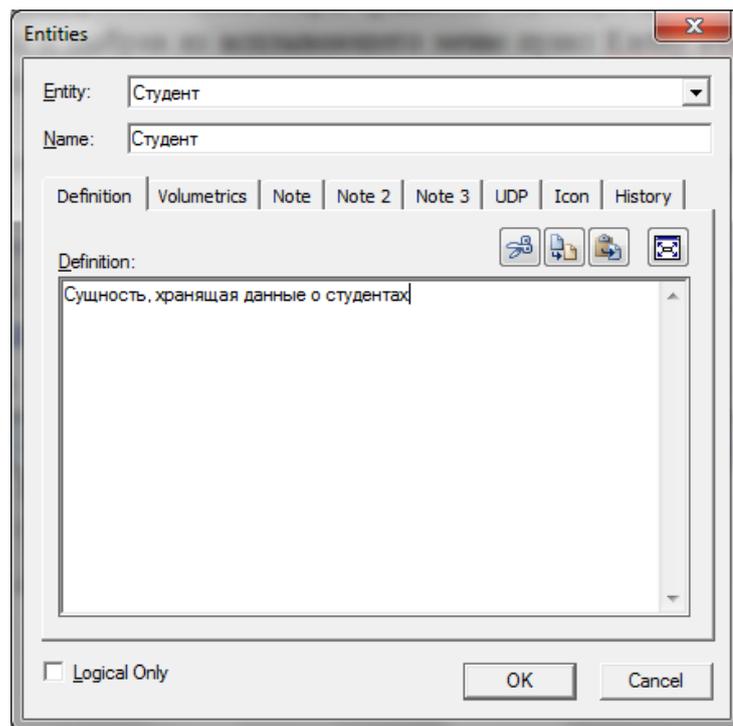


Рис.6 – Описание сущности

В нем необходимо указать вид объекта, для которого заводится **UDP** (диаграмма в целом, сущность, атрибут и т. д.) и тип данных. ERwin поддерживает для UDP шесть типов данных, в том числе:

**List** – список - при задании списка значения следует разделять запятой, значение по умолчанию выделяется символом «~»;

**Command** – команда – выполняемая строка.

Значение свойств, определяемых пользователем, задается в закладке **UDP** диалога **Entity Properties**.

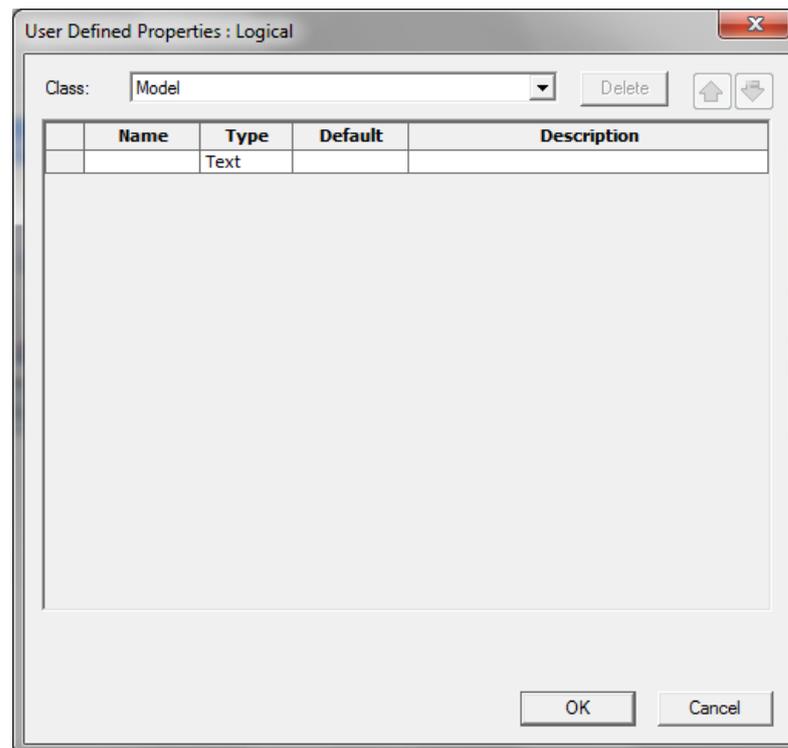


Рис.7 – Определение UDP

Атрибут или группа атрибутов, которые идентифицируют сущность, называется **первичным ключом**. Для описания атрибутов следует, «кликнув» правой кнопкой по сущности, выбрать в появившемся меню пункт **Attributes**. Появляется диалог **Attributes** (рис.8).

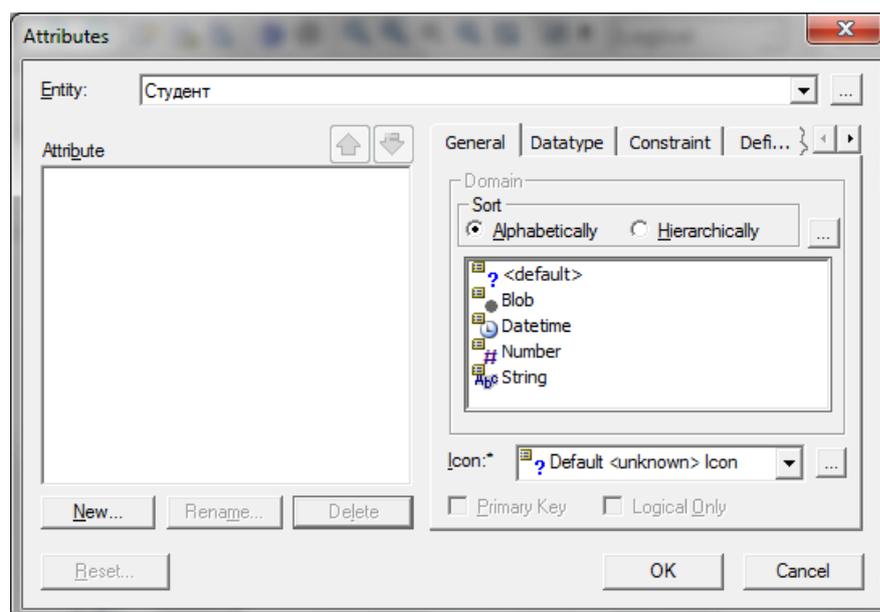


Рис.8 – Окно атрибутов

Если щелкнуть по кнопке **New...**, то в появившемся диалоге **New Attribute** можно указать имя атрибута, имя соответствующей ему в физической модели колонки и домен. Домен атрибута будет использоваться при определении типа колонки на уровне физической модели. Для атрибутов первичного ключа в закладке **General** диалога **Attributes** необходимо сделать пометку в окне выбора **Primary Key**.

Закладка **Datatype** позволяет выбрать тип атрибута.

Закладка **Definition** позволяет записывать определения отдельных атрибутов.

Очень важно дать атрибуту правильное имя. Атрибуты должны именоваться в единственном числе и иметь четкое смысловое значение. Согласно синтаксису **IDEF1X** имя атрибута должно быть уникально в рамках модели (а не только в рамках сущности!).

Каждый атрибут должен быть определен (закладка **Definition**).

Часто приходится создавать производные атрибуты, то есть атрибуты, значение которых можно вычислять из других атрибутов. Примером производного атрибута может служить *Возраст студента*, который может быть вычислен из атрибута *Дата рождения студента*. Производные атрибуты – ошибка нормализации, однако, их вводят для повышения производительности системы.

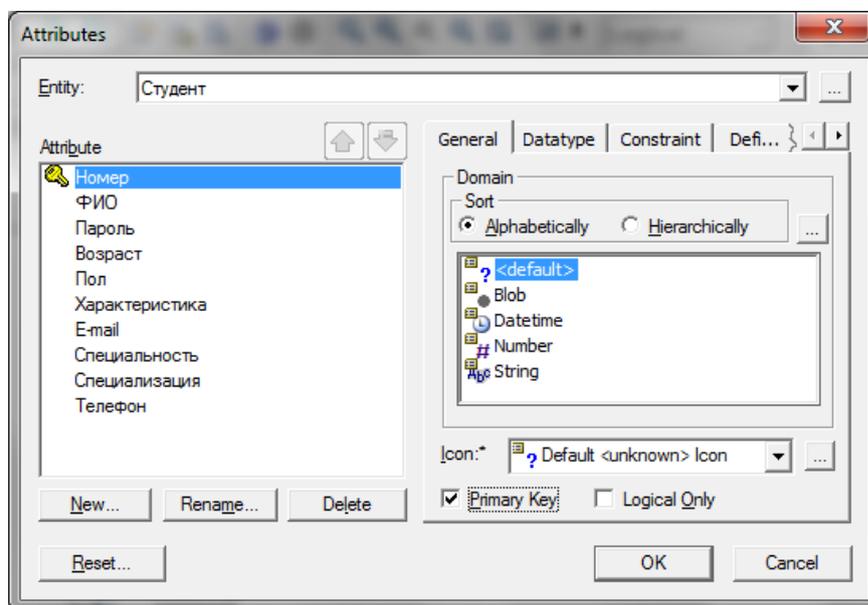


Рис.9 - Ввод атрибутов студента

## Связи

**Связь** является логическим соотношением между сущностями. Каждая связь должна именоваться глаголом или глагольной фразой. Имя связи выражает некоторое ограничение или бизнес-правило и облегчает чтение диаграммы.

Связь показывает, какие именно действия делает студент. По умолчанию имя связи на диаграмме не показывается. Для отображения имени следует в контекстном меню, которое появляется, если щелкнуть левой кнопкой мыши по любому месту диаграммы, не занятому объектами модели, выбрать пункт **Relationship Display** и затем включить опцию **verb Phrase**.

На логическом уровне можно установить идентифицирующую связь один-ко-многим, связь многие-ко-многим и неидентифицирующую связь один-ко-многим (соответственно это кнопки слева направо в палитре инструментов).

В **IDEF1X** различают **зависимые** и **независимые сущности**. Тип сущности определяется ее связью с другими сущностями. **Идентифицирующая** связь устанавливается между независимой (родительский конец связи) и зависимой (дочерний конец связи) сущностями. Когда рисуется идентифицирующая связь, ERwin автоматически преобразует дочернюю сущность в зависимую.

Зависимая сущность изображается прямоугольником со скругленными углами (рис.10).

Экземпляр зависимой сущности определяется только через отношение к родительской сущности.

При установлении идентифицирующей связи атрибуты первичного ключа родительской сущности автоматически переносятся в состав первичного ключа дочерней сущности. Эта операция дополнения атрибутов дочерней сущности при создании связи называется миграцией атрибутов. В дочерней сущности новые атрибуты помечаются как внешний ключ – **(FK)**.

В дальнейшем, при генерации схемы БД, атрибуты первичного ключа получают признак **NOT NULL**, что означает невозможность внесения записи в таблицу заказов без информации о номере студента.

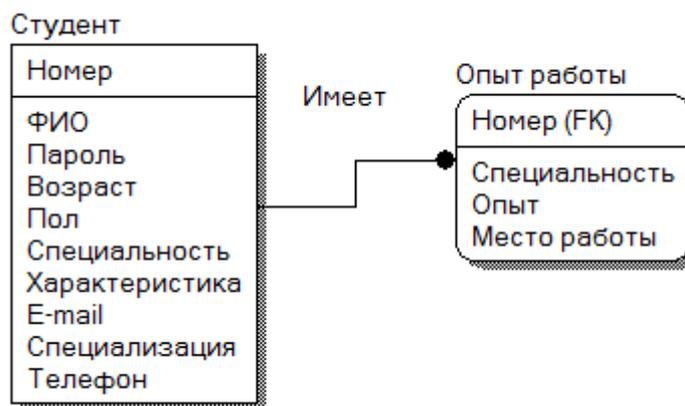


Рис.10 – Идентифицирующая связь (пример ERD - диаграммы)

В дальнейшем, при генерации схемы БД, атрибуты первичного ключа получают признак **NOT NULL**, что означает невозможность внесения записи в таблицу заказов без информации о номере студента.

При установлении **неидентифицирующей** связи дочерняя сущность остается независимой, а атрибуты первичного ключа родительской сущности мигрируют в состав неключевых компонентов дочерней сущности. Неидентифицирующая связь служит для связывания независимых сущностей.

Идентифицирующая связь показывается на диаграмме сплошной линией жирной точкой на дочернем конце связи, неидентифицирующая – пунктирной.

Для создания новой связи следует:

- 1) установить курсор на нужной кнопке в палитре инструментов (идентифицирующая или неидентифицирующая связь) и нажать левую кнопку мыши;
- 2) щелкнуть сначала по родительской, а затем по дочерней сущности.

Для редактирования свойств связи следует “кликнуть” правой кнопкой мыши по связи и выбрать на контекстном меню пункт **Relationship Properties** (рис.11).

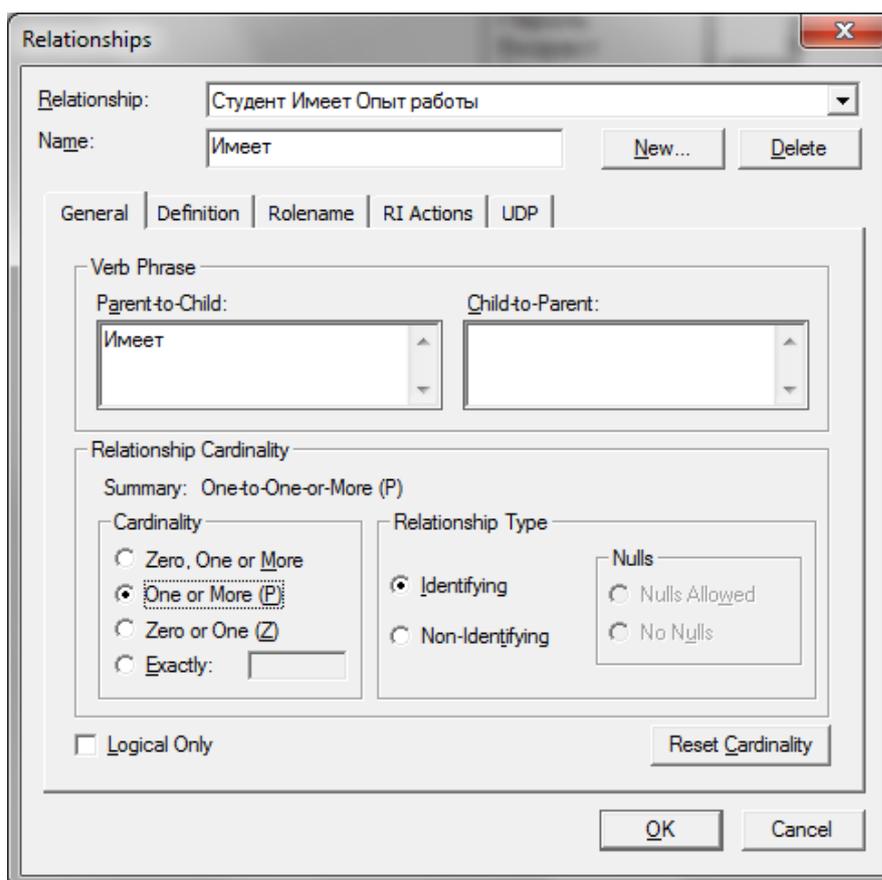


Рис.11 – Окно задания отношений

В закладке **General** появившегося диалога можно задать мощность, имя и тип связи.

**Мощность связи (Cardinality)** – служит для обозначения отношения числа экземпляров родительской сущности к числу экземпляров дочерней.

Различают четыре типа мощности:

Общий случай, когда одному экземпляру родительской сущности соответствуют 0, 1 или много экземпляров дочерней сущности, не помечается каким-либо символом («Ноль, один или много»);

Символом P помечается случай, когда одному экземпляру родительской сущности соответствуют 1 или много экземпляров дочерней сущности (исключено нулевое значение) («Один или много»);

Символом Z помечается случай, когда одному экземпляру родительской сущности соответствуют 0 или 1 экземпляр дочерней сущности (исключены множественные значения) («Ноль или один»);

Цифрой помечается случай точного соответствия, когда одному экземпляру родительской сущности соответствует заранее заданное число экземпляров дочерней сущности («Точно указанное количество»).

По умолчанию символ, обозначающий мощность связи, не показывается на диаграмме. Для отображения имени следует в контекстном меню, которое появляется, если щелкнуть левой кнопкой мыши по любому месту диаграммы, не занятому объектами модели, выбрать пункт **Relationship Display** и затем включить опцию **Cardinality**.

**Имя связи (Verb Phrase)** – фраза, характеризующая отношение между родительской и дочерней сущностями.

**Тип связи (идентифицирующая /неидентифицирующая).**

Для неидентифицирующей связи можно указать обязательность (**Nulls**). В случае обязательной связи (**No Nulls**) при генерации схемы БД атрибут внешнего ключа получит признак **NOT NULL**, несмотря на то, что внешний ключ не войдет в состав первичного ключа дочерней сущности. В случае необязательной связи (**Nulls Allowed**) внешний ключ может принимать значение **NULL**. Необязательная неидентифицирующая связь помечается прозрачным ромбом со стороны родительской сущности.

В закладке **Definition** можно дать более полное определение связи для того, чтобы в дальнейшем иметь возможность на него ссылаться.

В закладке **Rolename** можно задать имя роли.

В закладке **RI Actions** правила ссылочной целостности.

**Имя роли (функциональное имя)** – это синоним атрибута внешнего ключа, который показывает, какую роль играет атрибут в дочерней сущности. По умолчанию в списке атрибутов показывается только имя роли. Для отображения полного имени атрибута (как функционального имени, так и имени роли) следует в контекстном меню, которое появляется, если щелкнуть левой кнопкой мыши по любому месту диаграммы, не занятому объектами модели, выбрать пункт **Entities Display** и затем включить опцию **Attribute**. Полное имя показывается как функциональное имя и базовое имя, разделенные точкой.

Обязательным является применение имен ролей в том случае, когда два или более атрибутов одной сущности определены по одной и той же области, т. е. они имеют одинаковую область значений, но разный смысл.

**Связь типа один-к-одному** означает, что один экземпляр первой сущности (левой) связан с одним экземпляром второй сущности (правой).

Связь один-к-одному чаще всего свидетельствует о том, что на самом деле мы имеем всего одну сущность, неправильно разделенную на две.

**Связь типа один-ко-многим** означает, что один экземпляр первой сущности (левой) связан с несколькими экземплярами второй сущности (правой). Это

наиболее часто используемый тип связи. Левая сущность (со стороны "один") называется родительской, правая (со стороны "много") - дочерней.

**Связь типа многие-ко-многим** означает, что каждый экземпляр первой сущности может быть связан с несколькими экземплярами второй сущности, и каждый экземпляр второй сущности может быть связан с несколькими экземплярами первой сущности. Тип связи много-ко-многим является временным типом связи, допустимым на ранних этапах разработки модели. В дальнейшем этот тип связи должен быть заменен двумя связями типа один-ко-многим путем создания промежуточной сущности.

**Связь многие-ко-многим** возможно только на уровне логической модели данных. Такая связь обозначается сплошной линией с двумя точками на концах. Для внесения связи следует установить курсор на кнопке с изображением сплошной линии с двумя точками на концах в палитре инструментов, щелкнуть сначала по одной, а затем по другой сущности.

При переходе к физическому уровню ERwin автоматически преобразует связь многие-ко-многим, добавляя новую таблицу и устанавливая две новые связи один-ко-многим от старых к новой таблице. При этом имя новой таблице автоматически присваивается как «Имя1\_Имя2».

## Типы сущностей и иерархия наследования

Как было указано выше, связи определяют, является ли сущность независимой или зависимой.

Различают несколько типов зависимых сущностей:

**Характеристическая** – зависимая дочерняя сущность, которая связана только с одной родительской и по смыслу хранит информацию о характеристиках родительской сущности.

**Ассоциативная** – сущность, связанная с несколькими родительскими сущностями. Такая сущность содержит информацию о связях сущностей.

**Именующая** – частный случай ассоциативной сущности, не имеющей собственных атрибутов (только атрибуты родительских сущностей, мигрировавших в качестве внешнего ключа).

**Категориальная** – дочерняя сущность в иерархии наследования.

**Иерархия наследования** (или **иерархия категорий**) представляет собой особый тип объединения сущностей, которые разделяют общие характеристики. Обычно иерархию наследования создают, когда несколько сущностей имеют об-

щие по смыслу атрибуты, либо когда сущности имеют общие по смыслу связи, либо когда это диктуется бизнес- правилами.

## Ключи

Каждый экземпляр должен быть уникален и отличаться от других атрибутов.

**Первичный ключ (primary key)** – это атрибут или группа атрибутов, однозначно идентифицирующая экземпляр сущности. Атрибуты первичного ключа на диаграмме не требуют специального обозначения – это те атрибуты, которые находятся в списке атрибутов выше горизонтальной линии.

При внесении нового атрибута в диалоге **Attribute Editor** для того, чтобы сделать его атрибутом первичного ключа, нужно включить флажок Primary Key в нижней части закладки **General** (рис.9). На диаграмме неключевой атрибут можно внести в состав первичного ключа, воспользовавшись режимом переноса атрибутов (кнопка в палитре инструментов).

Выбор первичного ключа может оказаться непростой задачей, решение которой может повлиять на эффективность будущей ИС. В одной сущности могут оказаться несколько атрибутов или набор атрибутов, претендующих на роль первичного ключа. Такие претенденты называются **потенциальными ключами (candidate key)**.

Ключи могут быть **сложными**, т.е. содержащими несколько атрибутов. Сложные первичные ключи не требуют специального обозначения – это список атрибутов выше горизонтальной линии. Для того, чтобы стать первичным, потенциальный ключ должен удовлетворять ряду требований:

**Уникальность.** Два экземпляра не должны иметь одинаковых значений возможного ключа.

**Компактность.** Сложный возможный ключ не должен содержать ни одного атрибута, удаление которого не приводило бы к утрате уникальности.

Атрибуты ключа не должны содержать нулевых значений. Значение атрибутов ключа не должно меняться в течение всего времени существования экземпляра сущности. Каждая сущность должна иметь, по крайней мере, один потенциальный ключ. Многие сущности имеют только один потенциальный ключ. Такой ключ становится первичным. Некоторые сущности могут иметь более одного возможного ключа. Тогда один из них становится первичным, а остальные аль-

тернативными ключами. **Альтернативный ключ (Alternate Key)** – это потенциальный ключ, не ставший первичным.

Атрибуты, участвующие в неуникальных индексах, называются инверсионными входами (Inversion Entries). Инверсионные входы - это атрибут или группа атрибутов, которые не определяют экземпляр уникальным образом, но часто используются для обращения к экземплярам сущности. ERWin генерирует неуникальный индекс для каждого инверсионного входа.

**Внешние ключи (Foreign Key)** создаются автоматически, когда связь соединяет сущности: связь образует ссылку на атрибуты первичного ключа в дочерней сущности, и эти атрибуты образуют внешний ключ в дочерней сущности (миграция ключа). Атрибуты внешнего ключа обозначаются символом **(FK)** после своего имени.

Зависимая сущность может иметь один и тот же внешний ключ из нескольких родительских сущностей. Сущность может также получить один и тот же внешний ключ несколько раз от одного и того же родителя через несколько разных связей. Когда ERwin обнаруживает одно из этих событий, он распознает, что два атрибута одинаковы, и помещает атрибут внешнего ключа в зависимой сущности только один раз. Хотя в закладке **Key Group** диалога **Attribute** этот атрибут будет входить в два внешних ключа, на диаграмме он показывается только один раз. Это комбинирование или объединение идентичных атрибутов называется **унификацией**.

Унификация производится, поскольку правила нормализации запрещают существование в одной сущности двух атрибутов с одинаковыми именами.

Когда унификация нежелательна (например, когда два атрибута имеют одинаковые имена, но на самом деле они отличаются по смыслу и необходимо, чтобы это отличие отражалось в диаграмме), нужно использовать имена ролей атрибутов внешнего ключа.

## Денормализация

После нормализации все взаимосвязи данных становятся определены, исключая ошибки при оперировании данными. Но нормализация данных снижает быстродействие БД. Для более эффективной работы с данными, используя возможности конкретного сервера БД, приходится производить процесс, обратный нормализации, - денормализацию. Для процесса денормализации не существует

стандартного алгоритма, поэтому в каждом конкретном случае приходится искать свое решение.

Денормализация обычно проводится на физическом уровне модели. ERWin имеет следующие возможности по поддержке процесса денормализации:

- Сущности, атрибуты, группы ключей и домены можно создавать только на логическом уровне модели. В ERWin существует возможность выделения элементов логической модели таким образом, чтобы они не появлялись на физическом уровне.

- Таблицы, столбцы, индексы и домены можно создавать только на физическом уровне. В ERWin существует возможность выделения элементов модели таким образом, чтобы они не появлялись на логическом уровне. Эта возможность напрямую поддерживает денормализацию физической модели, так как позволяет проектировщику включать таблицы, столбцы и индексы в физическую модель, ориентированную на конкретную СУБД.

- Разрешение связей «многие-ко-многим». При разрешении этих связей в логической модели ERWin добавляет ассоциированные сущности и позволяет добавить в них атрибуты. При разрешении связей в логической модели автоматически разрешаются связи и в физической модели.

## Лабораторная работа № 1 «Нормализация баз данных»

### Цель работы

Целью работы является ознакомление с нормализацией баз данных на основе первой, второй и третьей нормальных форм.

### Технология выполнения лабораторной работы

1. Для выполнения лабораторной работы необходимо ознакомиться с предметной областью темы варианта, выданного преподавателем.
2. Определить информацию, которую требуется зафиксировать в будущей базе данных.
3. Представить выделенную информацию в виде данных.
4. Привести имеющиеся данные к атомарному виду.
5. Привести полученные данные к первой нормальной форме.
6. Далее следует разделить имеющийся массив данных на логически связанные данные.
7. Привести данные ко второй нормальной форме.
8. Связать таблицы по смыслу, добавив атрибуты, являющиеся внешними ключами.
9. Привести имеющиеся данные к третьей нормальной форме.
10. Составить отчет в виде имен заголовков полученных таблиц.

### Задание на лабораторную работу

1. Получить вариант задания у преподавателя.
2. Привести данные к первой нормальной форме.
3. Привести данные к второй нормальной форме.
4. Привести данные к третьей нормальной форме.
5. Подготовить отчет о проделанной работе.

## Лабораторная работа № 2

### «Основы логического моделирования реляционных баз данных в ERWin»

#### Цель работы

Целью работы является ознакомление с технологией построения логической модели в ERWin, изучение методов определения ключевых атрибутов сущностей, освоение методов проверки адекватности логической модели, изучение типов связей между сущностями.

#### Технология выполнения лабораторной работы

Первым шагом при создании логической модели БД является построение диаграммы ERD (Entity Relationship Diagram). ERD-диаграмма, включает в себя атрибуты сущностей.

Рассмотрим процесс построения логической модели на примере БД студентов системы «Служба занятости в рамках вуза». Первым этапом является определение сущностей и атрибутов. В БД будут храниться записи о студентах, следовательно, сущностью будет студент.

Составим ERD-диаграмму, определяя типы атрибутов и проставляя связи между сущностями.

- 1) На панели инструментов нажали на кнопку , вставили в любое место рабочего окна;
- 2) Далее написали имя сущности, например, «Студент»;
- 3) Щёлкнули ПКМ по сущности и выбрали в контекстном меню Attributes, далее добавляем новые атрибуты и устанавливаем ключи;
- 4) Далее аналогичным образом создаются все остальные сущности, результат на рис. 12;

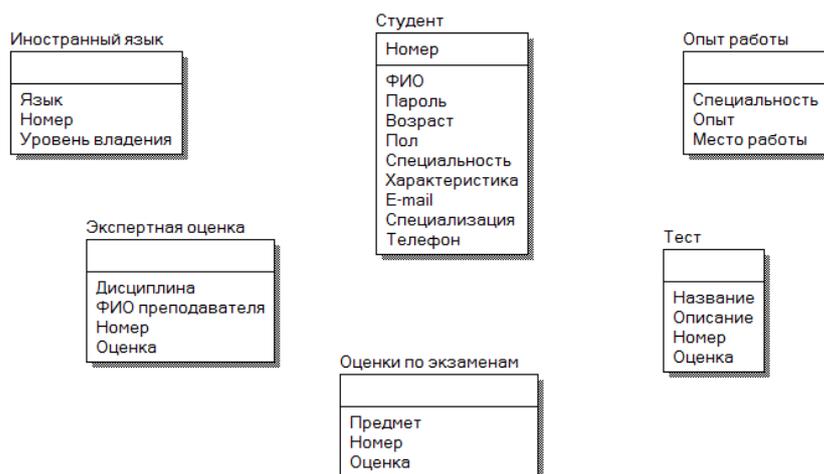


Рис. 12. Созданные сущности

5) Теперь расставляем связи. Все сущности будут зависимыми от сущности «Студент». Связи будут типа «один-ко-многим». Для этого мы из панели инструментов выбираем нужную связь из . В нашем случае это будет , и связываем необходимые сущности (сначала нажимаем на родительскую сущность, потом на дочернюю). При связывании родительской сущности «Студент» с другими-дочерними, появляется окно (рис.13). В нём выбираем первый вариант, т.е. соединить с внешним ключом (FK).

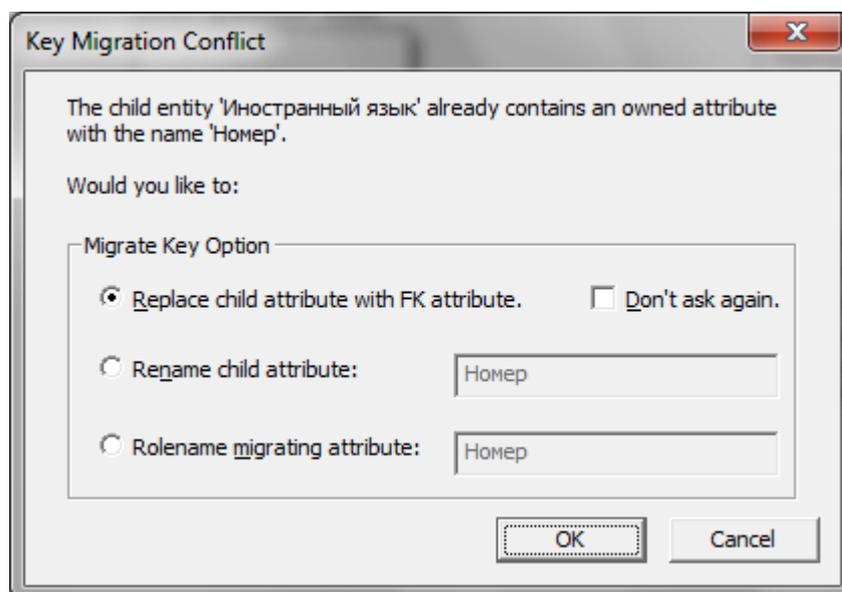


Рис.13. Настройки связи между сущностями

Для связи один-ко-многим идентифицирующей или неидентифицирующей достаточно указать имя, характеризующее отношение от родительской к дочерней сущности (**Parent-to-Child**).

Для связи многие-ко-многим следует указывать имена как **Parent-to-Child** так и **Child-to-Parent**.

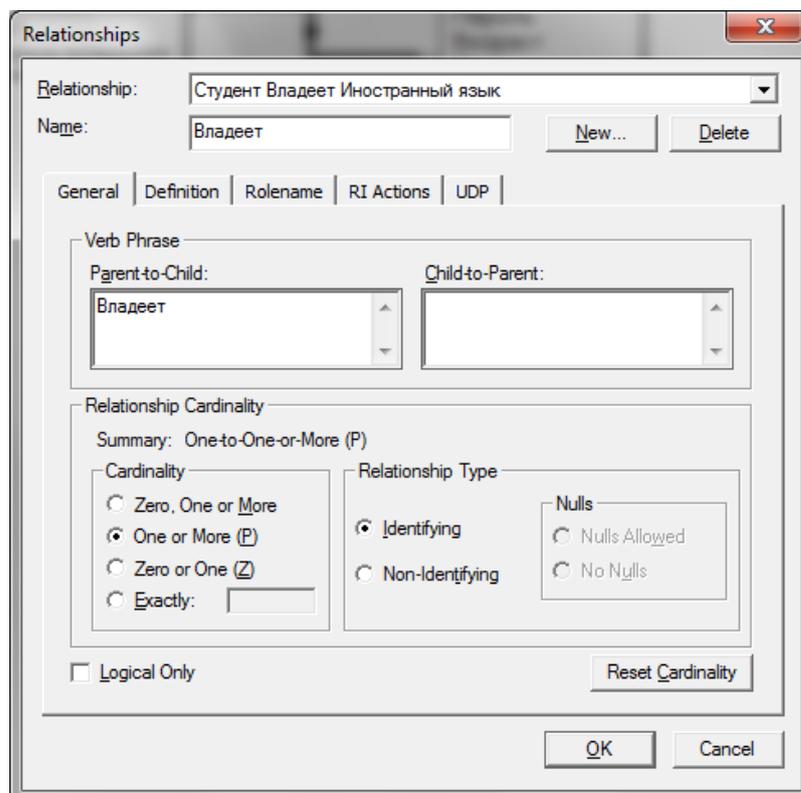


Рис.14 – Настройка отношений между сущностями

Для отображения имени следует в контекстном меню, которое появляется, если щелкнуть левой кнопкой мыши по любому месту диаграммы, не занятому объектами модели, выбрать пункт **Relationship Display** и затем включить опцию **verb Phrase**.

На полученной диаграмме рядом со связью отражается ее имя, показывающее соотношение между сущностями. При проведении связи между сущностями первичный ключ мигрирует в дочернюю сущность (рис.15).

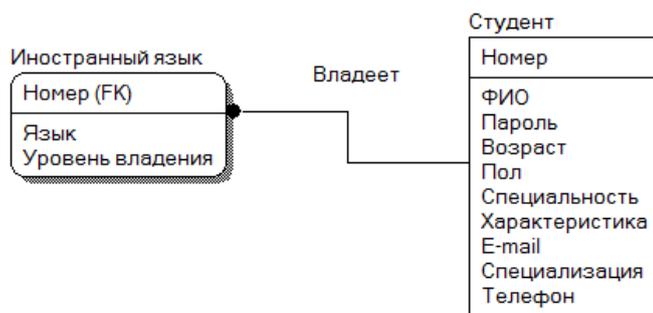


Рис. 15 – Связь между сущностями

6) Аналогичным образом расставляем оставшиеся связи. Полученный результат представлен на рис.16:

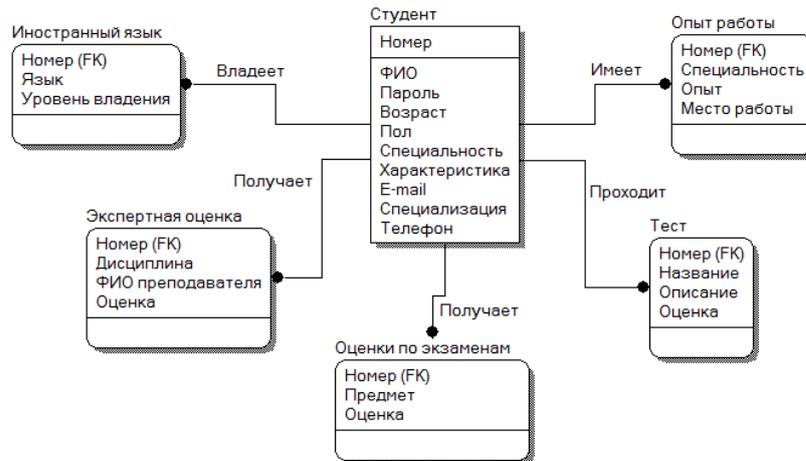


Рис.16 - ERD-диаграмма БД студентов

7) Следующим этапом при построении логической модели является определение ключевых атрибутов и типов атрибутов.

Чтобы установить тип атрибута, надо выбрать в Attributes вкладку Datatype:

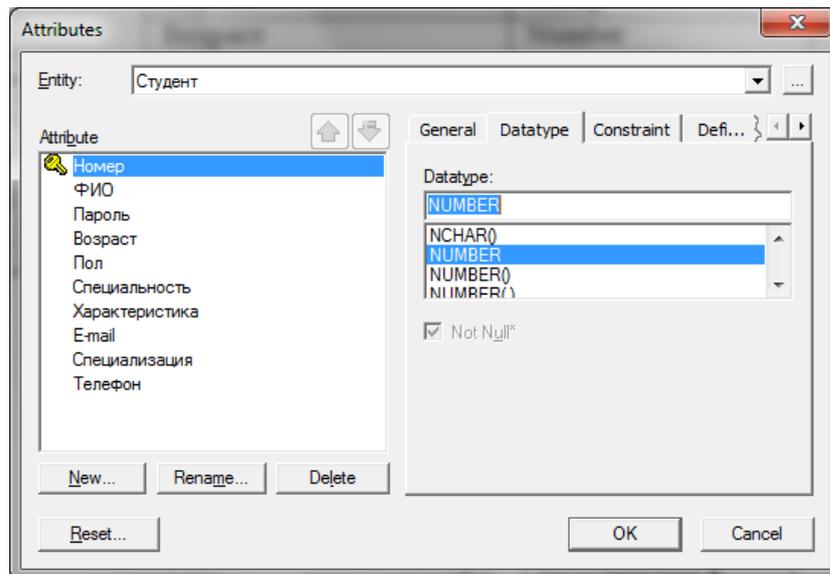


Рис.17. Выбор типа атрибутов

8) Выберем для каждой сущности ключевые атрибуты, однозначно определяющие сущность. Для сущности «Студент» это будет уникальный номер, для сущности «Опыт работы» все поля являются ключевыми, так как по разным специальностям студент может иметь разный опыт работы в разных фирмах. Сущность «Тест» определяется названием, так как студент по одному тесту может иметь только одну оценку. Оценка по экзамену определяется только названием предмета, экспертная оценка зависит от преподавателя, который ее составил.

Поэтому в качестве ключевых атрибутов выберем “Дисциплину” и “Ф.И.О. преподавателя”. У сущности “Иностранный язык” уровень владения зависит только от наименования языка, следовательно, это и будет являться ключевым атрибутом.

9) Выберем для каждой сущности ключевые атрибуты, однозначно определяющие сущность. Для сущности “Студент” это будет уникальный номер, для сущности “Опыт работы” все поля являются ключевыми, так как по разным специальностям студент может иметь разный опыт работы в разных фирмах. Сущность “Тест” определяется названием, так как студент по одному тесту может иметь только одну оценку. Оценка по экзамену определяется только названием предмета, экспертная оценка зависит от преподавателя, который ее составил, поэтому в качестве ключевых атрибутов выберем “Дисциплину” и “Ф.И.О. преподавателя”. У сущности “Иностранный язык” уровень владения зависит только от наименования языка, следовательно, это и будет являться ключевым атрибутом.

Создать ключевые атрибуты для сущностей можно 2 способами:

- перетащить необходимые атрибуты с помощью мыши;
- щёлкнув ПКМ по сущности и выбрав Key Groups, в появившемся окне переместить нужные атрибуты в нужное место:

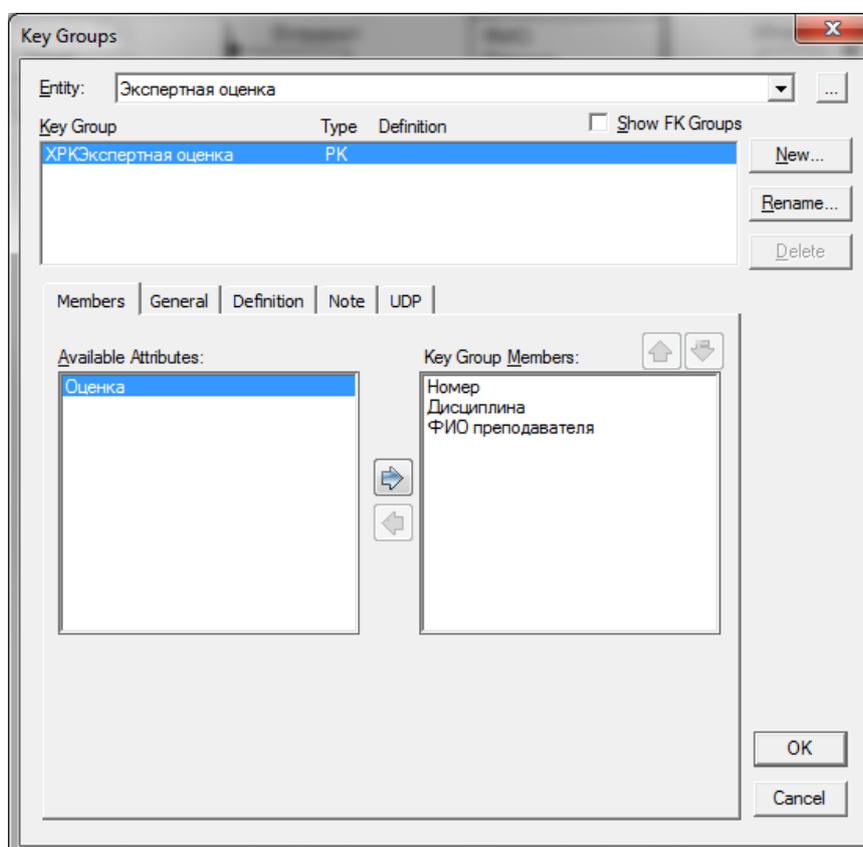


Рис.18 – Окно перемещения ключей

Получим новую диаграмму, изображенную на рис. 19, где все ключевые атрибуты будут находиться над горизонтальной чертой внутри рамки, изображающей сущность.

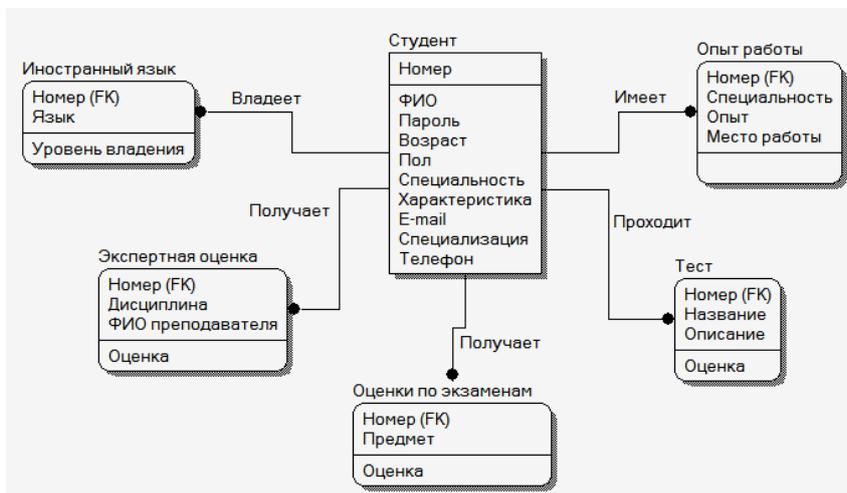


Рис.19. ERD-диаграмма БД студентов с ключевыми атрибутами

На этом создание логической модели закончено. Теперь можно переходить к созданию физической модели.

### Задание на лабораторную работу

6. Получить вариант задания у преподавателя.
7. Создать сущности.
8. Задать атрибуты сущностей.
9. Определить первичные ключи в сущностях.
10. Определить состав альтернативных ключей.
11. Связать сущности между собой, используя описанные типы связей.
12. После проведения связей определить состав внешних ключей.
13. Сохранить полученную диаграмму.

### Лабораторная работа № 3

#### «Основы физического моделирования реляционных баз данных в Toad Data Modeler»

#### Цель работы

- освоить проектирование баз данных с помощью CASE-средства Toad Data Modeler;
- построить логическую модель;
- построить физическую модель;
- сгенерировать код для переноса физической модели на СУБД.

#### Технология выполнения лабораторной работы

Целью создания физической модели является обеспечение администратора соответствующей информацией для переноса логической модели данных в СУБД.

Toad Data Modeler поддерживает генерацию код для переноса физической модели на конкретную СУБД. При этом логическая модель трансформируется в физическую по следующему принципу: сущности становятся таблицами, атрибуты становятся столбцами, а ключи становятся индексами.

Таблица - Сопоставление компонентов логической и физической модели

<b>Логическая модель</b>	<b>Физическая модель</b>
Сущность	Таблица
Атрибут	Столбец
Логический тип (текст, число, дата,	Физический тип (корректный тип,
Первичный ключ	Первичный ключ, индекс РК
Внешний ключ	Внешний ключ, индекс FK
Альтернативный ключ	AK-индекс - уникальный, не- первичный
Правило бизнес- логики	Триггер или сохраненная про- цедура
Взаимосвязи	Взаимосвязи, определяемые использованием FK-атрибутов

Рассмотрим пример нормализации полученной в предыдущей лабораторной работе БД до третьей нормальной формы. Для приведения БД в первую нормальную форму необходимо выполнить условие, при котором все атрибуты содержат атомарные значения. Рассмотрим атрибуты сущности «Студент». Студент может иметь несколько адресов электронной почты и несколько телефонных номеров, что является нарушением первой нормальной формы.

Необходимо создать отдельные сущности «E-mail» и «Телефон» и связать их с сущностью «Студент» (рис. 20).

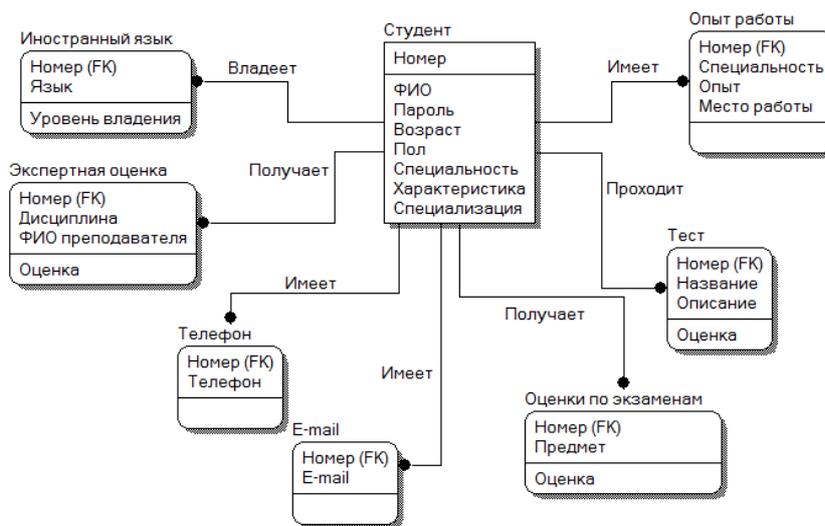


Рис. 20 - ERD-диаграмма БД студентов в первой нормальной форме

Проверим соответствие БД второй нормальной форме. Все неключевые атрибуты полностью должны зависеть от первичного ключа. Нетрудно заметить, что это условие выполняется для всех сущностей БД; следовательно, можно сделать вывод о том, что она находится во второй нормальной форме.

Для приведения БД к третьей нормальной форме необходимо обеспечить отсутствие транзитивных зависимостей неключевых атрибутов. Такая зависимость наблюдается у атрибутов «Специальность» и «Специализация» у сущности «Студент»: специализация зависит от специальности и от группы, в которой обучается студент. Создадим новую независимую сущность «Специальность», перенеся в нее атрибут «Специализация» и создав новый атрибут «Группа», являющийся ключевым и определяющий атрибуты «Специальность» и «Специализация». Проведем неидентифицирующую связь от сущности «Специальность» к сущности «Студент», при этом ключевой атрибут «Группа» мигрирует в сущность «Студент». Получим БД в третьей нормальной форме, так как других транзитивных зависимостей неключевых атрибутов нет (рис. 21).

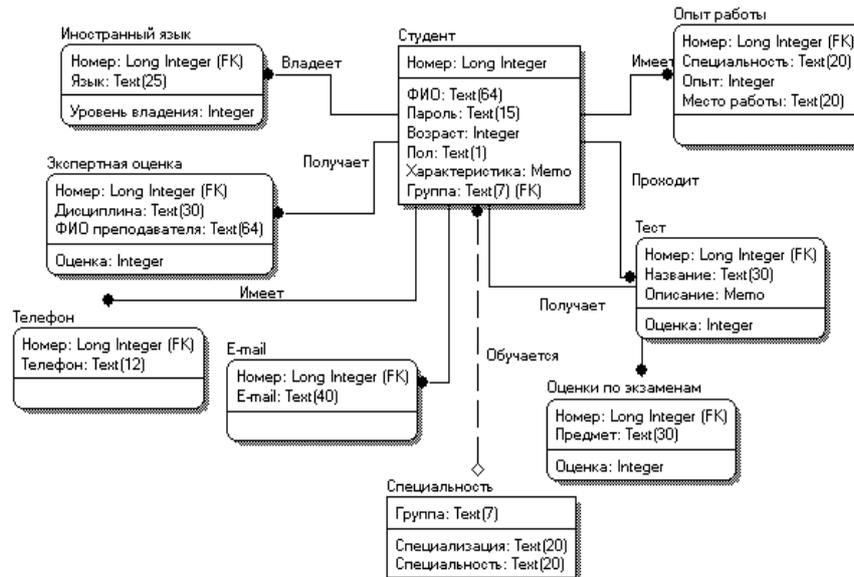


Рис. 21 - Физическая модель БД студентов

### Порядок выполнения лабораторной работы

1. Запустить ПО Toad Data Modeller.
2. Средствами ПО разработать логическую модель для хранения указанных сведений и указать отношения между представленными объектами.
  - 2.1. Создайте новый проект. Меню File\New\Project.

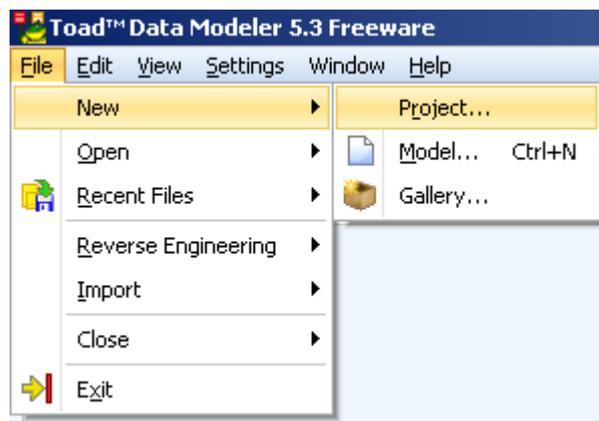


Рис. 22. Создание нового проекта

- 2.2. После создания проекта добавьте логическую модель

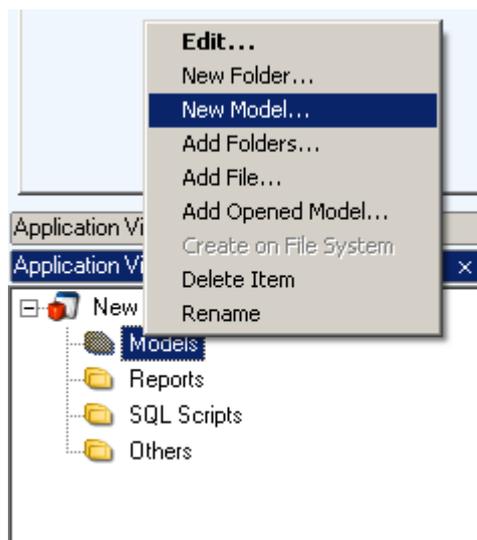


Рис. 23. Создание новой модели

2.3. Выберите пункт New -> Model. В следующем диалоге задайте имя для логической модели.

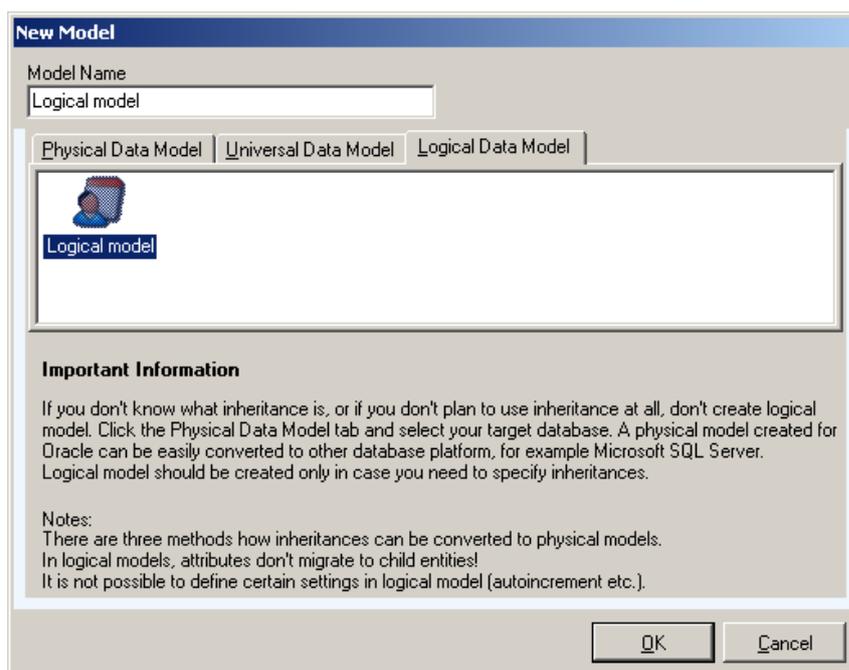


Рис. 24 - Создание логической модели

2.4. Разместите в рабочей области необходимые объекты.

Добавление объектов к модели выполняется через панель инструментов

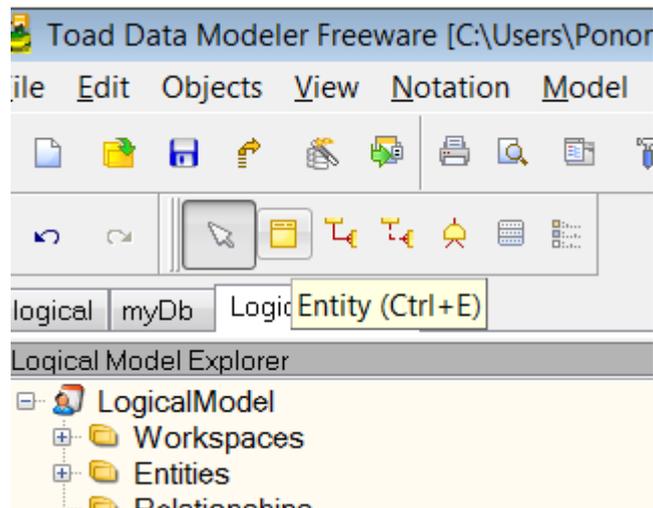


Рис. 25. Добавление объектов к модели

2.5. Задайте необходимые свойства для объектов и уникальные ключи в разделе Unique Identifiers.

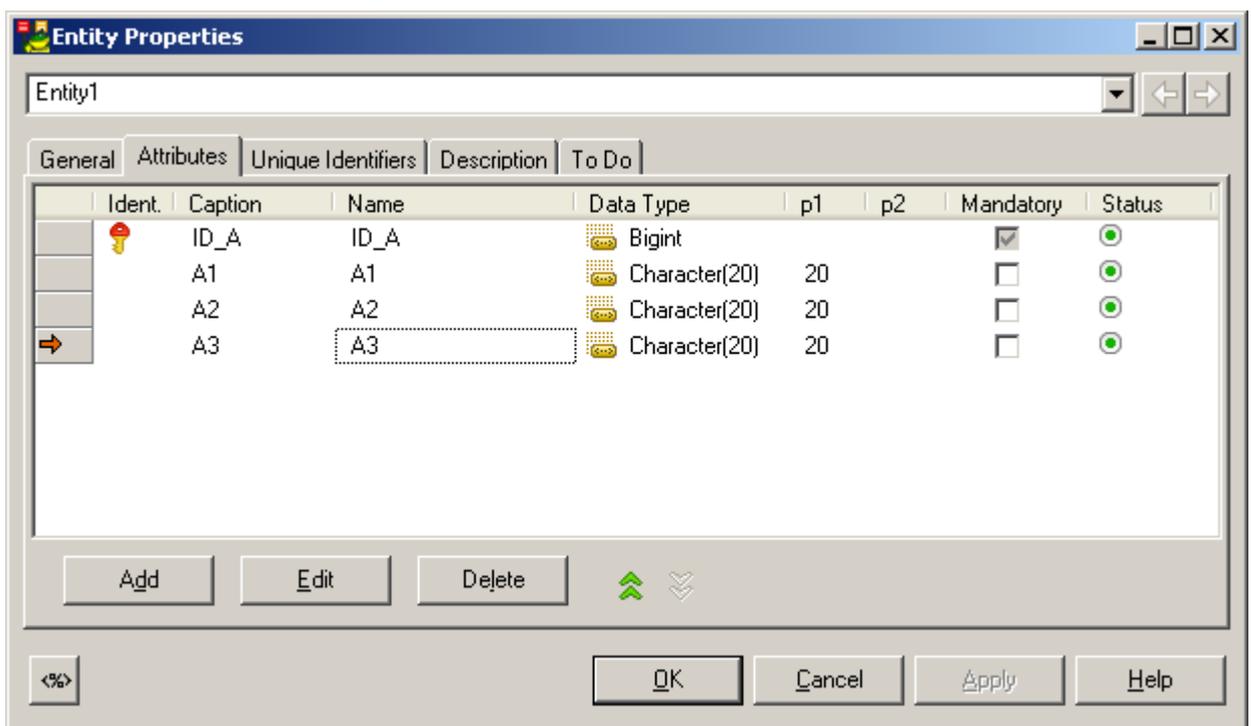


Рис. 26. Добавление атрибутов объекту

Результатом выполнения работы должна быть концептуальная схема.

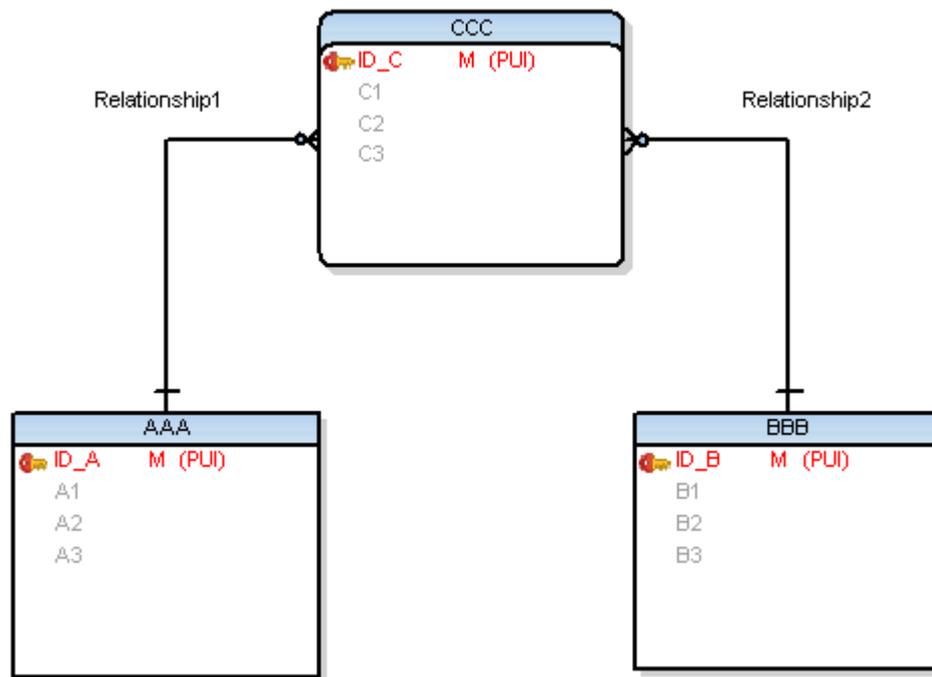


Рис. 27. Логическая модель

### 3. Разработать физическую модель БД.

3.1. Выполните преобразование полученной в предыдущей работе модели в физическую. Выполните команду Simple Conversion в меню Model -> Convert Model.

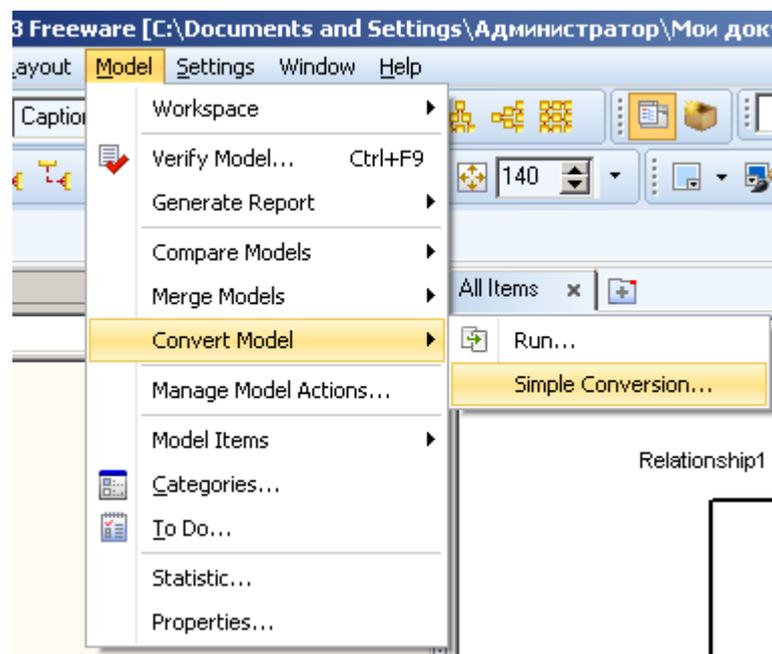


Рис. 28. Преобразование модели

3.2. В качестве целевой СУБД выберите MS SQL Server 2012.

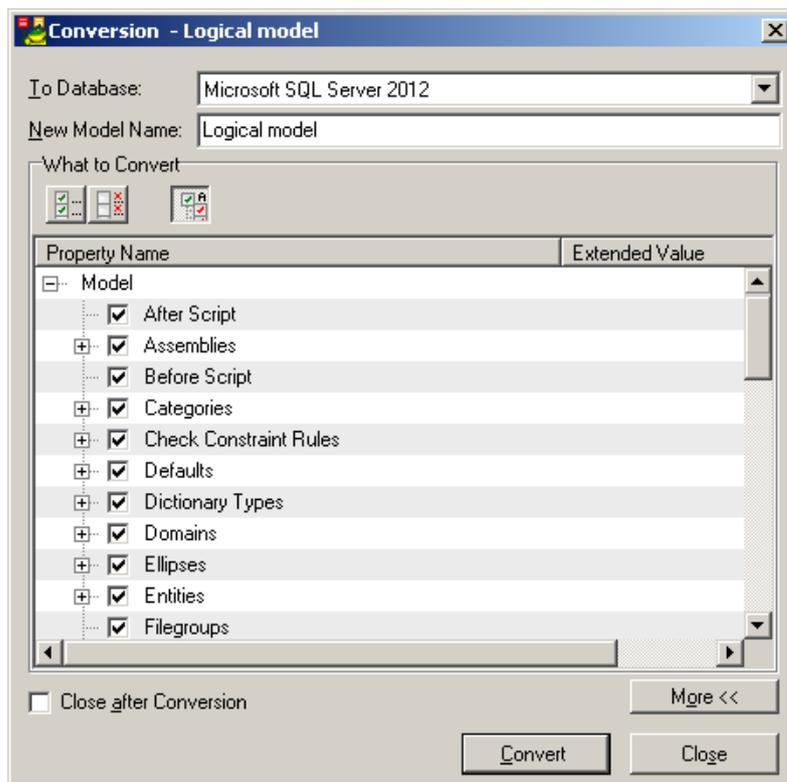


Рис. 29. Выбор целевой СУБД

Далее будет получена физическая модель базы данных.

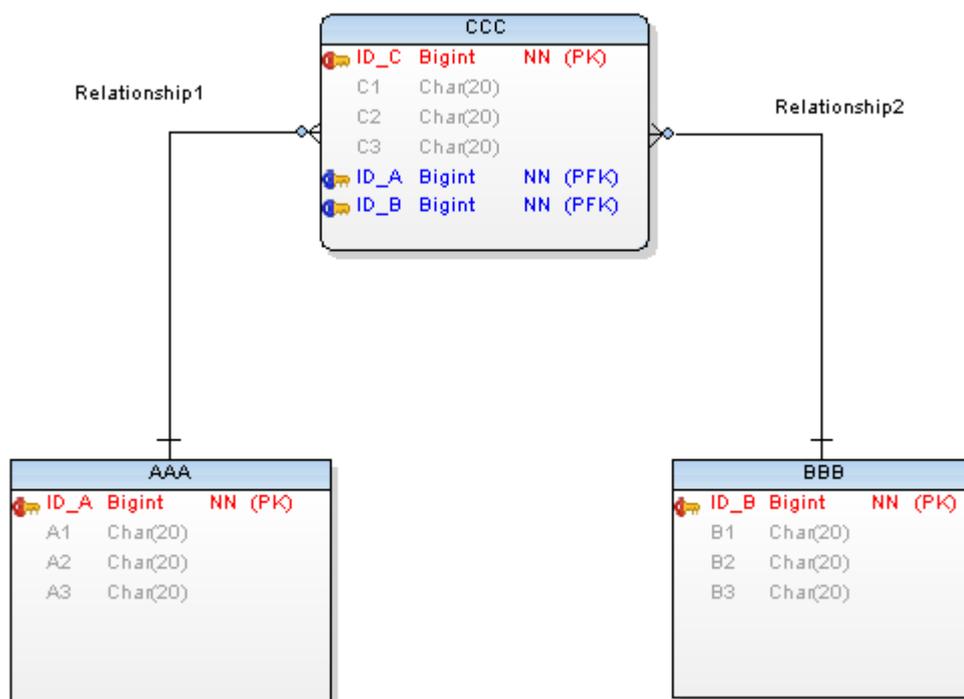


Рис. 30. Физическая модель базы данных

3.3. Далее необходимо получить код для переноса физической модели базы данных в выбранную СУБД. Выберите команду **Generate DDL Script...** в меню **Model**.

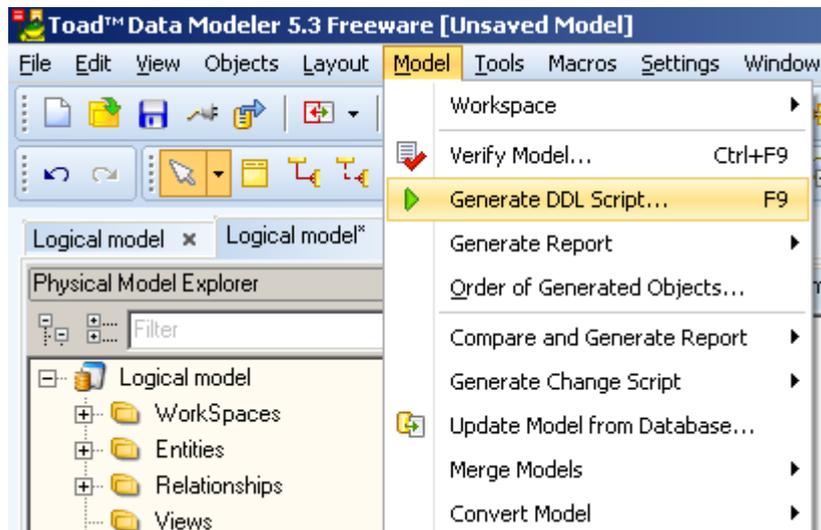


Рис. 31. Выбор меню генерации кода

3.4. В представленном меню нажмите кнопки **Verify** для проверки целостности базы данных. Далее кнопку **Generate** для генерации кода, который будет сохранен в файл. И затем кнопку **Show Code** для просмотра.

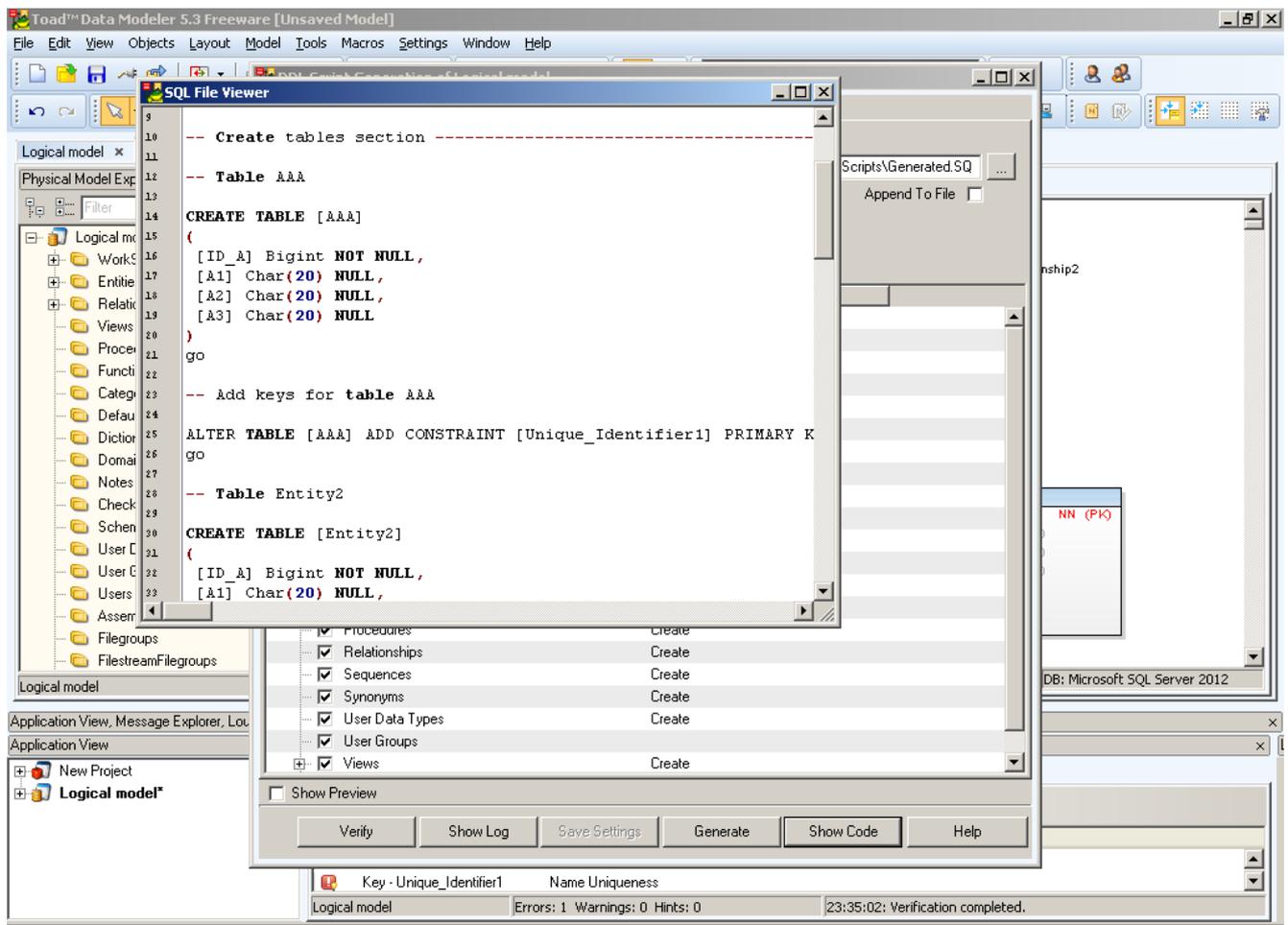


Рис. 32. Генерация кода

4. Необходимо сгенерированный код выполнить в целевой СУБД для переноса физической модели.

4.1. Запустите Microsoft SQL Server 2012.

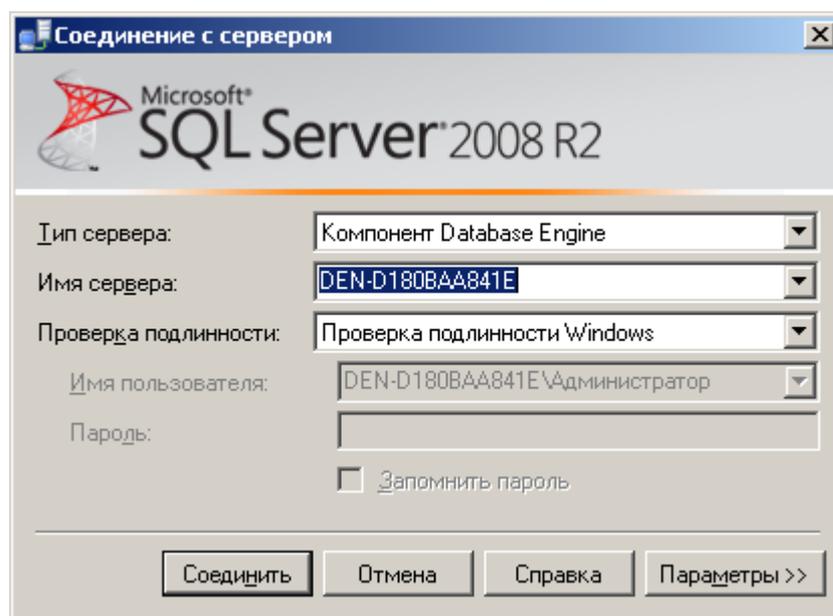


Рис. 33. Запуск MS SQL Server

4.2. Правой кнопкой мыши добавьте новую базу данных и затем так же правой кнопкой мыши создайте новый SQL запрос для созданной БД.

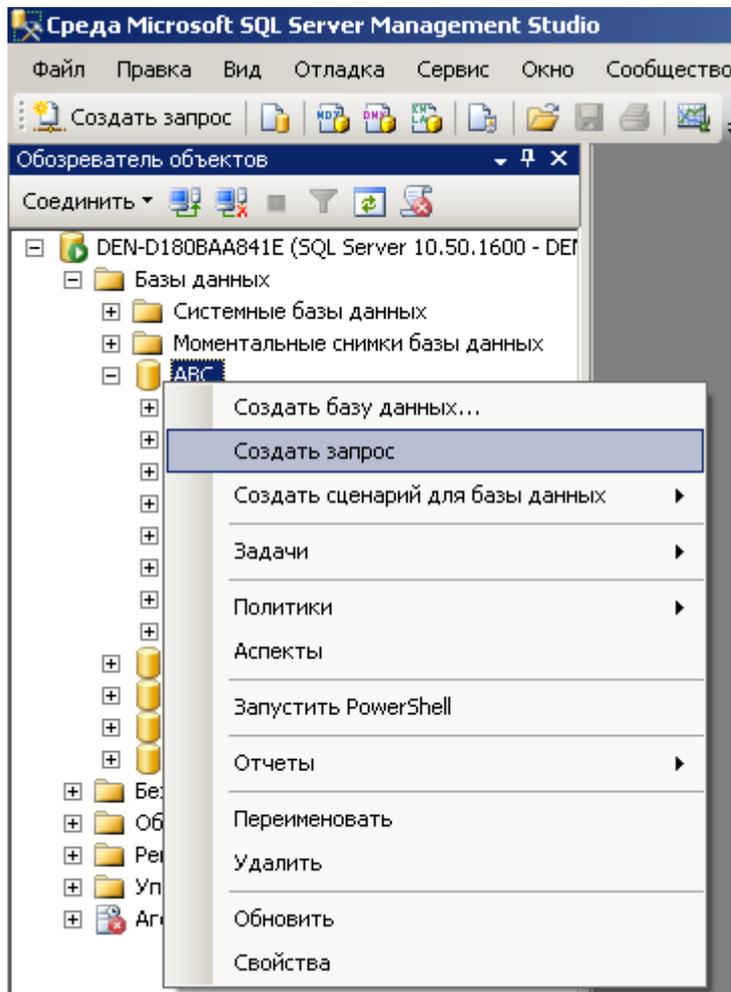


Рис. 34. Создание запроса БД

4.3. Выполните сгенерированный код, нажав кнопку с восклицательным знаком на панели инструментов.

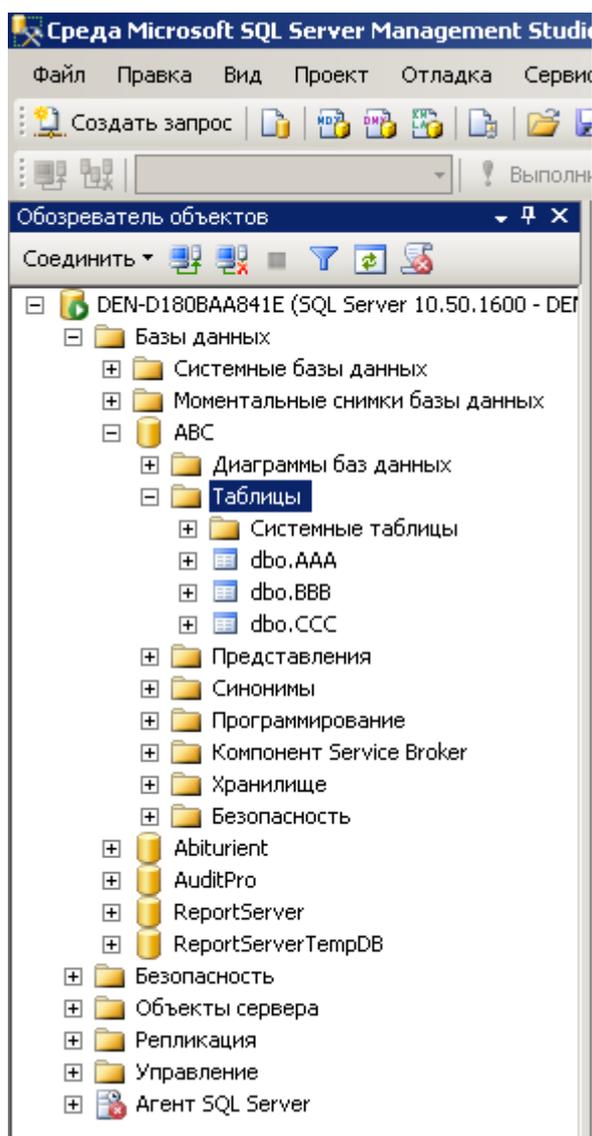


Рис. 35. Таблицы на сервере

Таким образом с помощью CASE-средства Toad Data Modeler была создана логическая модель заданной предметной области, затем получено преобразование в физическую модель, сгенерирован код для переноса на целевую СУБД, а также разобран пример выполнения кода на сервере MS SQL Server.

### Задание на лабораторную работу

1. На основе утвержденной преподавателем предметной области с помощью Toad Data Modeler создать логическую модель базы данных.
2. Построить физическую модель БД.
3. Сгенерировать код для переноса на целевую СУБД MS SQL Server 2012.
4. Создать БД и выполнить сгенерированный код на сервере MS SQL Server 2012.

## Лабораторная работа № 4

### «Проектирование базы данных на сервере MySQL»

#### Цель работы

Спроектировать базу данных в соответствии с вариантом на сервере MySQL.  
Изучить основы реинжиниринга моделей баз данных.

#### Краткие теоретические сведения

##### Использование контекстных меню

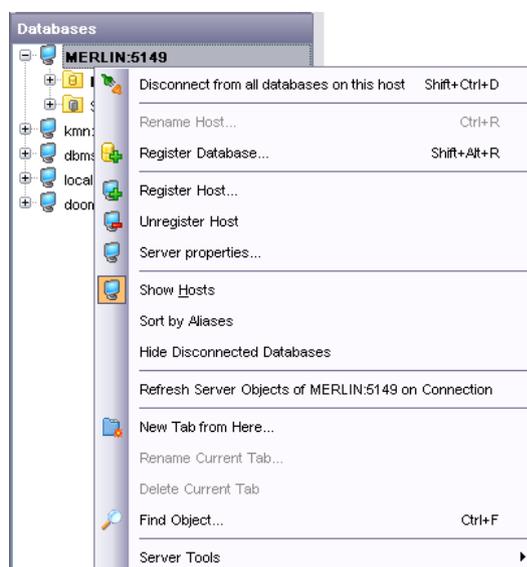
Контекстное меню - это меню, отрывающееся при нажатии правой кнопкой мыши на объекте. Эти меню позволяют упростить работу с базами данных и с их объектами, обеспечивая быстрый доступ к наиболее часто используемым операциям.

Контекстные меню, открывающиеся в проводнике баз данных:

- контекстное меню сервера;
- контекстное меню базы данных;
- контекстное меню объекта.

##### Контекстное меню сервера

Контекстное меню сервера - это список действий, который можно открыть, нажав правой кнопкой мыши на сервере в проводнике баз данных.



Connect to Host/Disconnect from Host - подключиться к серверу/отключиться от сервера.

Rename Host - переименовать сервер.

Register Database - зарегистрировать новую базу с помощью Register Database Wizard.

Register Host - зарегистрировать хост.

Unregister Host - удалить регистрацию хоста.

Server properties - открыть Свойства сервера.

включить опцию Show Hosts.

Sort by Aliases - отсортировать базы данных по именам (По умолчанию, они расположены в порядке регистрации).

Hide Disconnected Databases - эта функция применяется для того, чтобы скрыть неподключенные базы данных.

Refresh Server Objects of server on Connection - обновлять серверные объекты при подключении к серверу.

New Tab from Here - создать новую вкладку.

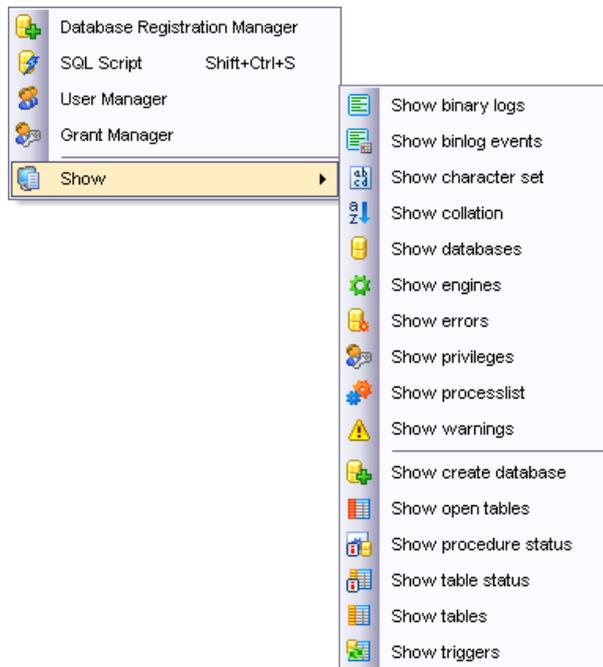
Rename Current Tab - переименовать активную вкладку.

Delete Current Tab - удалить активную вкладку.

Find Object - поиск задаваемого пользователем сочетания букв и цифр в названиях баз данных и объектов.

открыть подменю Server Tools.

Server Tools:



Database Registration Manager - открыть Администратор регистрации баз данных.

SQL Script - открыть Редактор скриптов.

User Manager - открыть Редактор пользователей.

Grant Manager - управлять правами.

Show - просмотреть информацию по Серверу и базам данных.

Контекстное меню базы данных

Контекстное меню баз данных открывается при нажатии в проводнике на базе данных правой кнопкой мыши.

Connect to Database/Disconnect from Database - подключиться к выбранной базе данных/отключиться от выбранной базы данных.

Database Registration Info - редактировать регистрационную информацию.

Database Properties - открыть Свойства базы данных.

Rename Database - переименовать базу данных.

Register Database - зарегистрировать новую базу с помощью Мастера регистрации сервера/баз данных.

Unregister Database - удалить регистрацию базы данных.

Register Host - зарегистрировать новый хост с помощью мастера Мастера регистрации сервера/баз данных.

Unregister Host - удалить регистрацию хоста и всех зарегистрированных на нем баз данных.

Server Properties - открыть Свойства сервера.

включить опцию Show Hosts.

Sort by Aliases - отсортировать базы данных по именам.

Hide Disconnected Databases - скрыть неподключенные базы данных.

Refresh Database - обновить базу данных.

Refresh Server Objects of server on Connection - обновлять серверные объекты при подключении к серверу.

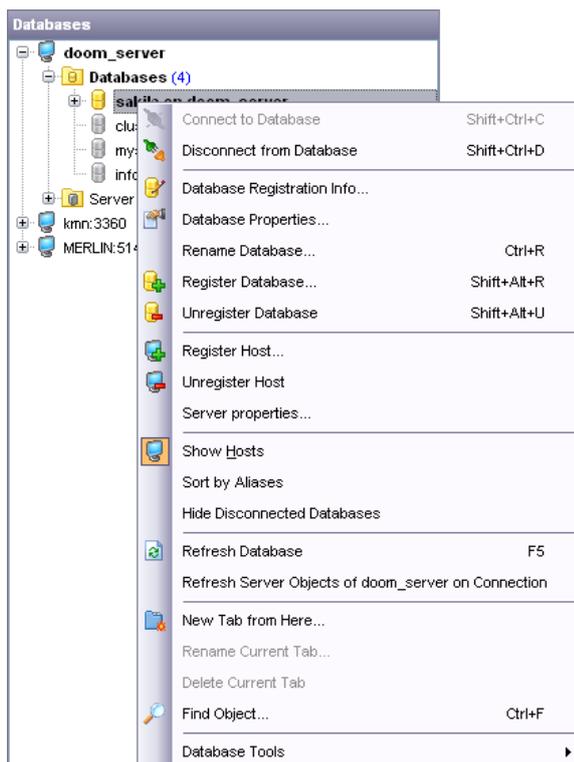
New Tab from Here - создать новую вкладку.

Rename Current Tab - переименовать активную вкладку.

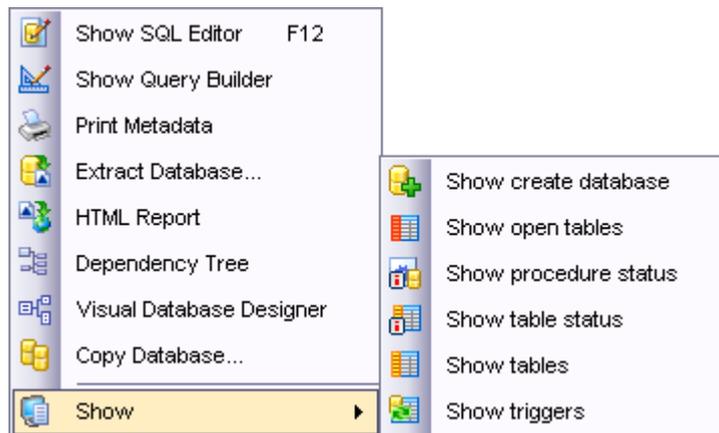
Delete Current Tab - удалить активную вкладку.

Find Object - поиск задаваемого пользователем сочетания букв и цифр в названиях баз данных и объектов,

Database Tools - открыть инструменты базы данных



## Database Tools



SQL Script - Открыть Редактор скриптов.

Show Query Builder - Открыть Визуальный конструктор запросов.

Print Metadata - Открыть Печать метаданных.

Extract Database - Извлечь базу данных.

HTML Report - Создать HTML отчет.

Dependency Tree - Открыть Дерево зависимостей.

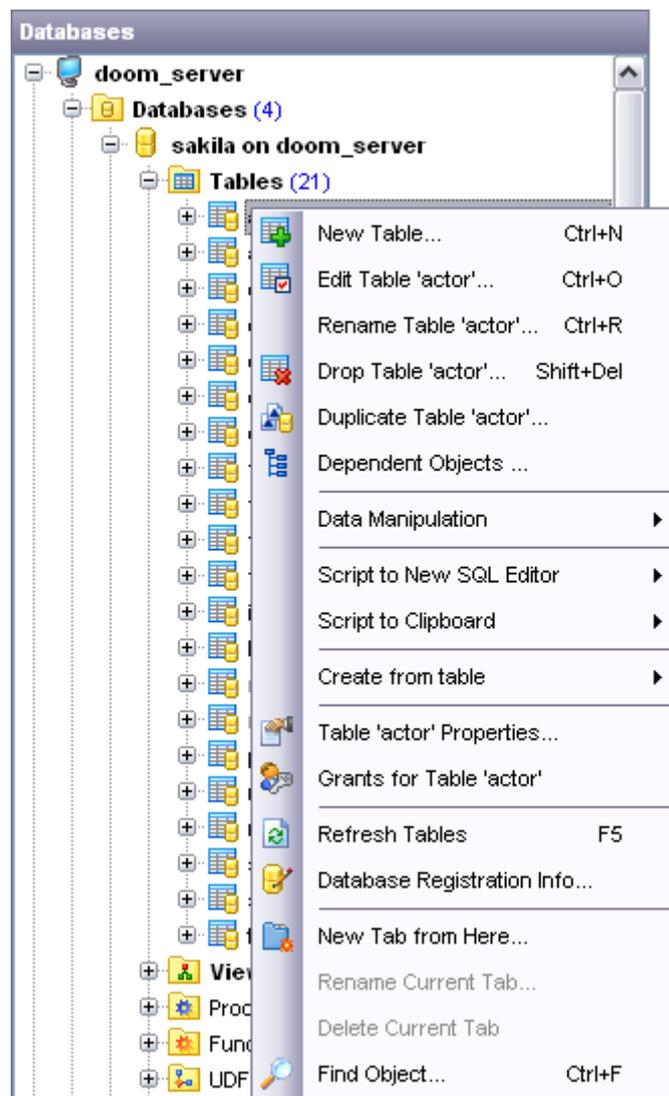
Visual Database Designer - Открыть Визуальный конструктор баз данных.

Copy Database - Копировать базу данных.

Show - Просмотреть информацию по базе данных.

### Контекстное меню объекта

Это меню открывается при нажатии на любом объекте базы данных правой кнопкой мыши.



New <объект> - создать новый объект базы данных

Edit <имя объекта> - открыть выбранный объект в редакторе

Rename <имя объекта> - переименовать выбранный объект

Drop <имя объекта> - удалить выбранный объект

Duplicate <"имя объекта"> - копировать объект с помощью Мастера копирования объектов

Dependent Objects - открыть дерево зависимостей

Data Manipulation - выбрать одну из операций импорта/экспорта

Script to New SQL Editor - копировать скрипт объекта в редактор SQL

Script to Clipboard - копировать скрипт объекта в буфер обмена

Create from table - создать другой объект на основании этой таблицы

Table properties - свойства таблицы (только для таблиц)

Grants for Table - задать права на таблицу (только для таблиц)

Refresh <имя объекта> - обновить информацию об объекте

Database Registration Info - редактировать регистрационную информацию  
базы данных

New Tab from Here - создать новую вкладку

Rename Current Tab - переименовать активную вкладку

Delete Current Tab - удалить активную вкладку

Find Object - найти элемент базы данных.

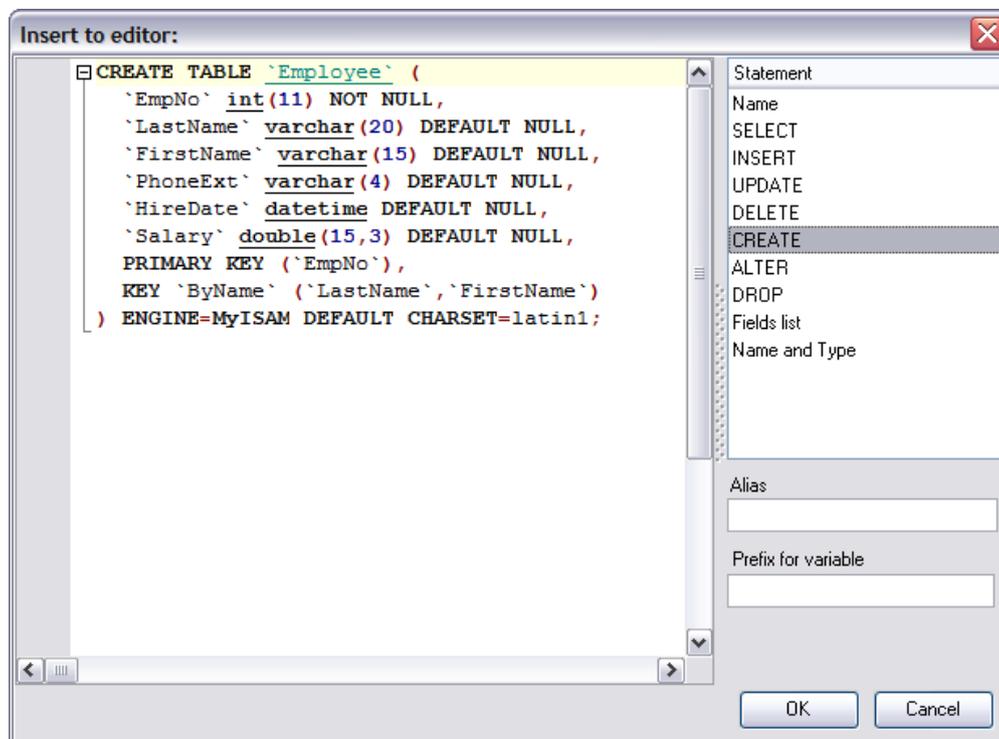
### Основные операции над объектами

Проводник баз данных позволяет получить доступ к основным действиям над объектом базы данных с помощью контекстного меню объекта.

Важно: Для каждого типа объектов свое меню.

Чтобы открыть объект в соответствующем редакторе, достаточно двойного щелчка мыши на нем в проводнике баз данных.

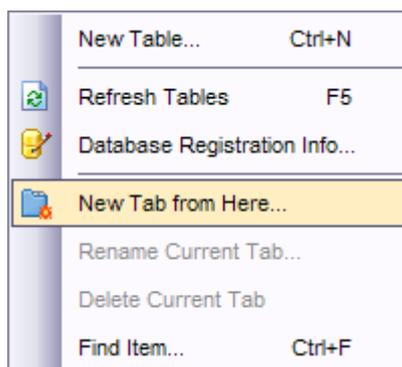
С помощью операции drag-and-drop Вы можете добавить объекты в Редактор запросов, Визуальный конструктор запросов или в Редактор SQL скриптов.



## Использование вкладок

Чтобы сделать Вашу работу как можно более удобной, в SQL Manager 2011 for MySQL существует специальный сервис вкладок (Tab). Используйте вкладки, если хотите обеспечить быстрый доступ к определенной группе объектов проводника. Это может быть определенный сервер или база данных, или группа объектов базы данных. Например, Вы можете вынести папку с представлениями на отдельную вкладку или создать несколько вкладок на каждой из которых будет только одна база данных.

Поместить объект на отдельную вкладку можно с помощью пункта контекстного меню New Tab from Here, который присутствует в контекстном меню каждого объекта.



Важно: Если отключить опцию отображения подобъектов таблицы на панели инструментов проводника, то New Tab from Here будет неактивен, так как таблица не является узлом дерева.

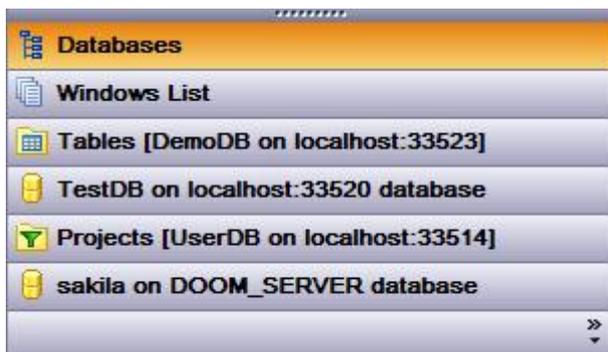
Если же эта опция включена, (кнопка View Mode | Show Table Subobjects), то пункт меню New Tab from Here становится активным.

В окне проводника баз данных можно отображать вкладки двух видов:

в виде иконок на нижней панели проводника,



в виде закладок.



Для перехода от одного вида к другому достаточно потянуть разделитель вверх или вниз.

Для этой же цели используются пункты Show more Buttons и Show Fewer Buttons всплывающего меню, открывающегося при нажатии на стрелку Configure buttons.

Для переключения между вкладками достаточно выделить мышью иконку или закладку.

Для этой же цели используются сочетания клавиш:

Ctrl+Shift+N - переход на следующую вкладку,

Ctrl+Shift+P - переход на предыдущую вкладку.

### **Переименование вкладок**

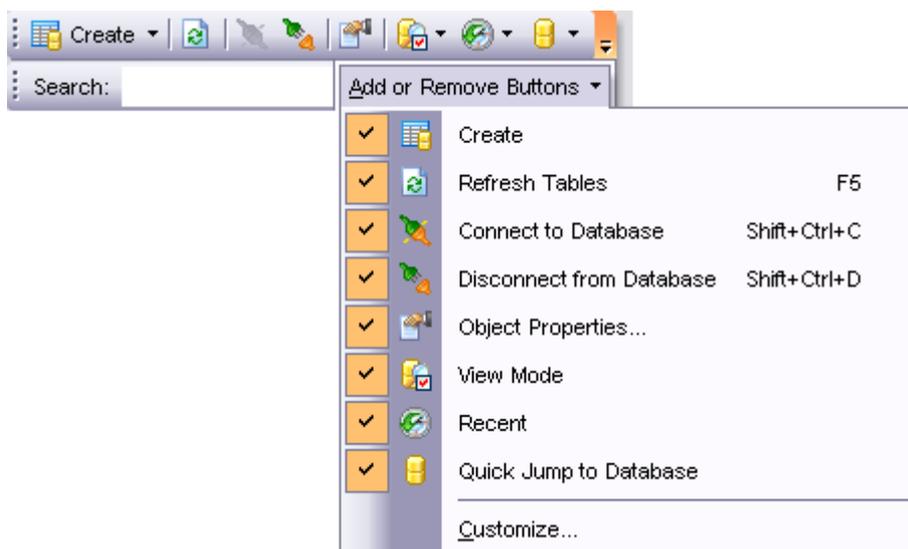
Если хотите переименовать вкладку, то нужно перейти на нее и выбрать пункт контекстного меню Rename Current Tab.

### **Удаление вкладок**

Для удаления вкладки необходимо перейти на нее и выбрать пункт контекстного меню Delete Current Tab.

### **Панели инструментов проводника баз данных**

Это набор инструментов над окном проводника баз данных. На эту панель вынесены кнопки управления базами данных и настройки проводника.



Панель инструментов можно настраивать, используя функцию Add or Remove Buttons, открывающуюся при нажатии на кнопку More Buttons в правом углу панели.

Create - создать объект.

Refresh - обновить выбранную группу объектов.

Connect to Database - подключиться к базе данных.

Disconnect from Database - отсоединить базу данных.

Object Properties - открыть свойства выделенного объекта.

View Mode - настройка проводника.

Resent - список недавно использовавшихся объектов.

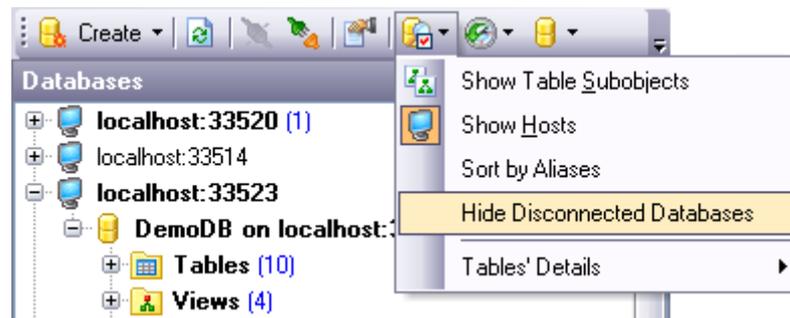
Quick Jump To Database - быстрый переход к базе данных.

Все команды, кроме View Mode, предназначены для работы с базами данных. View Mode отвечает за настройку самого проводника и SQL Assistant.

View Mode позволяет:

- показать подобъекты таблиц в проводнике баз данных - Show Table Subobjects,
- показать хосты Show Hosts,
- сортировать базы данных по именам (по умолчанию базы отсортированы внутри сервера по времени подключения) - Sort by Aliases,
- скрывать неподключенные базы данных Hide Disconnected Databases,

– настраивать Tables' Details в SQL Assistant,



Используйте Options | Environment Options... | DB Explorer для доступа к дополнительным настройкам проводника баз данных.

### Технология выполнения лабораторной работы

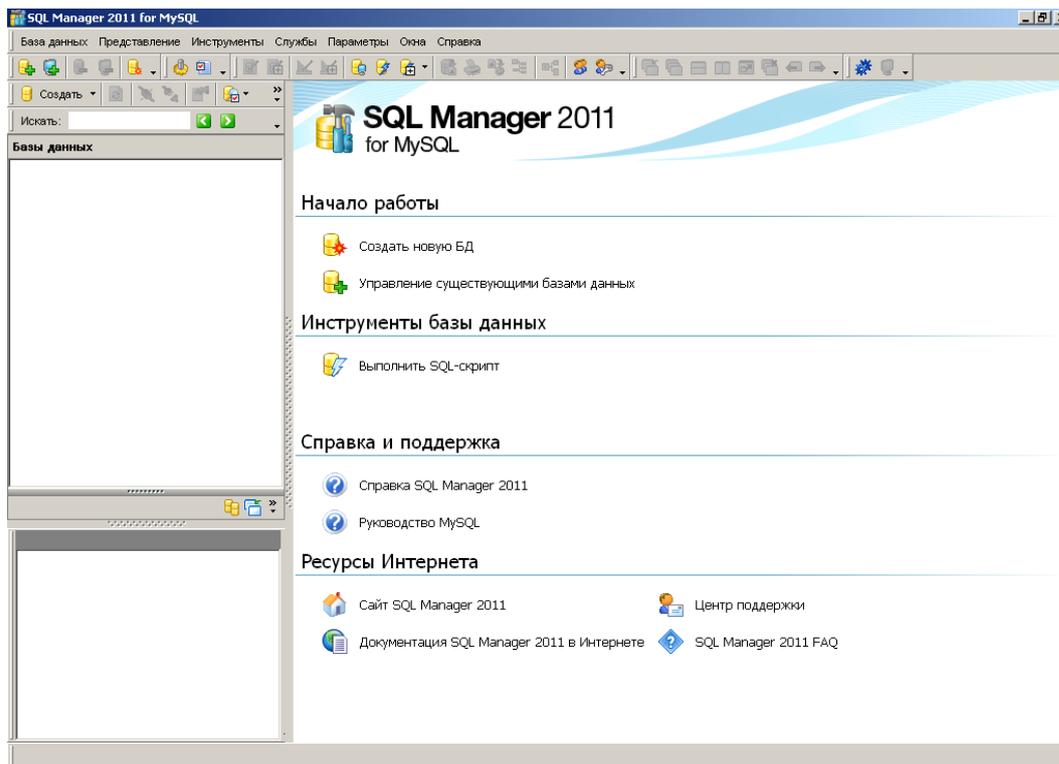
1. Запустите MySQL сервер.

1.1. Запустите приложение MySQL Notifier (это приложение легко найти в поиске меню Пуск).

1.2. В меню приложения MySQL Notifier выберите команду Start для запуска MySQL сервера.

2. Создайте базу данных в студии SQL Manager Lite for MySQL.

2.1. Запустите приложение SQL Manager Lite for MySQL (установите галку на режиме 30-дневной версии).



2.2. Создайте базу данных на сервере MySQL с помощью открытой студии:

Базы данных -> Создать базу данных...

2.3. В мастере создания баз данных введите имя базы данных. Оставляем галку «Зарегистрировать после создания».

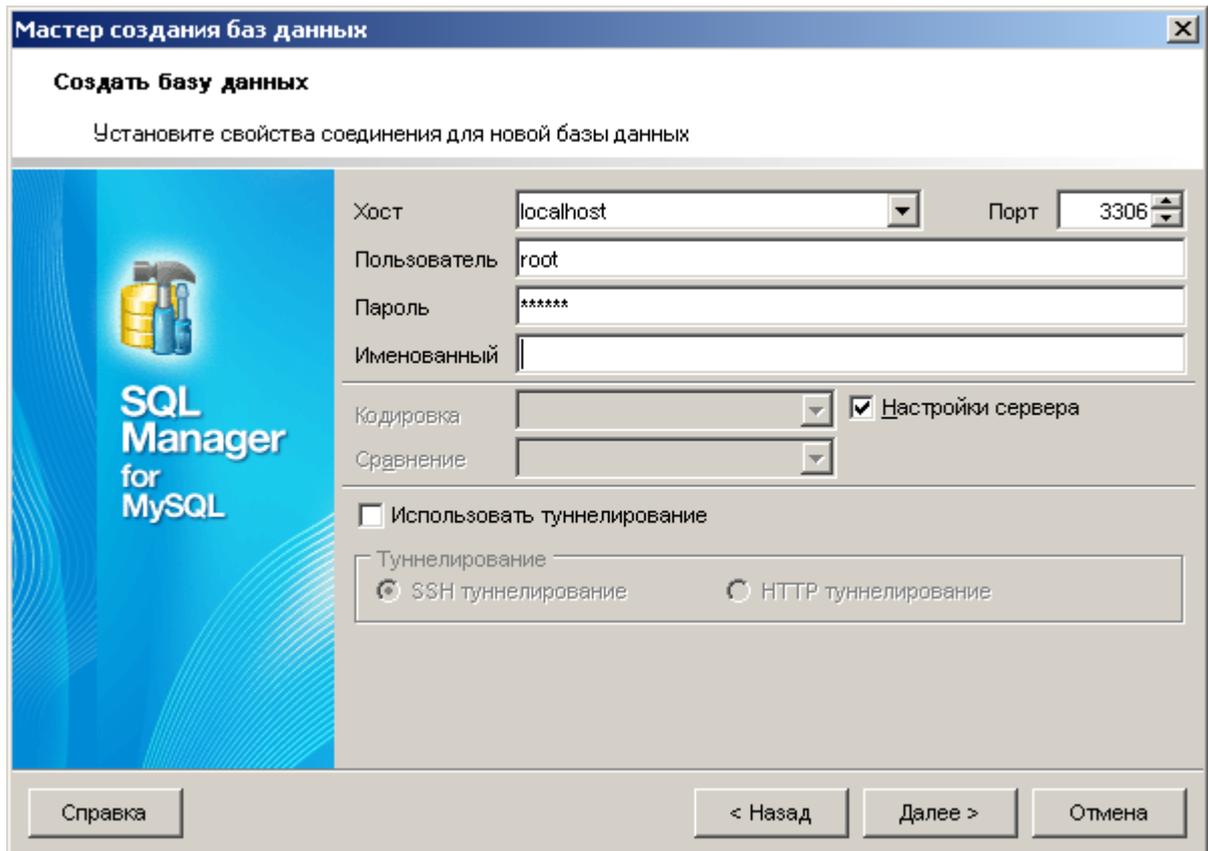
2.4. В следующем окне вводим идентификационные данные:

Хост: localhost

Порт: 3306

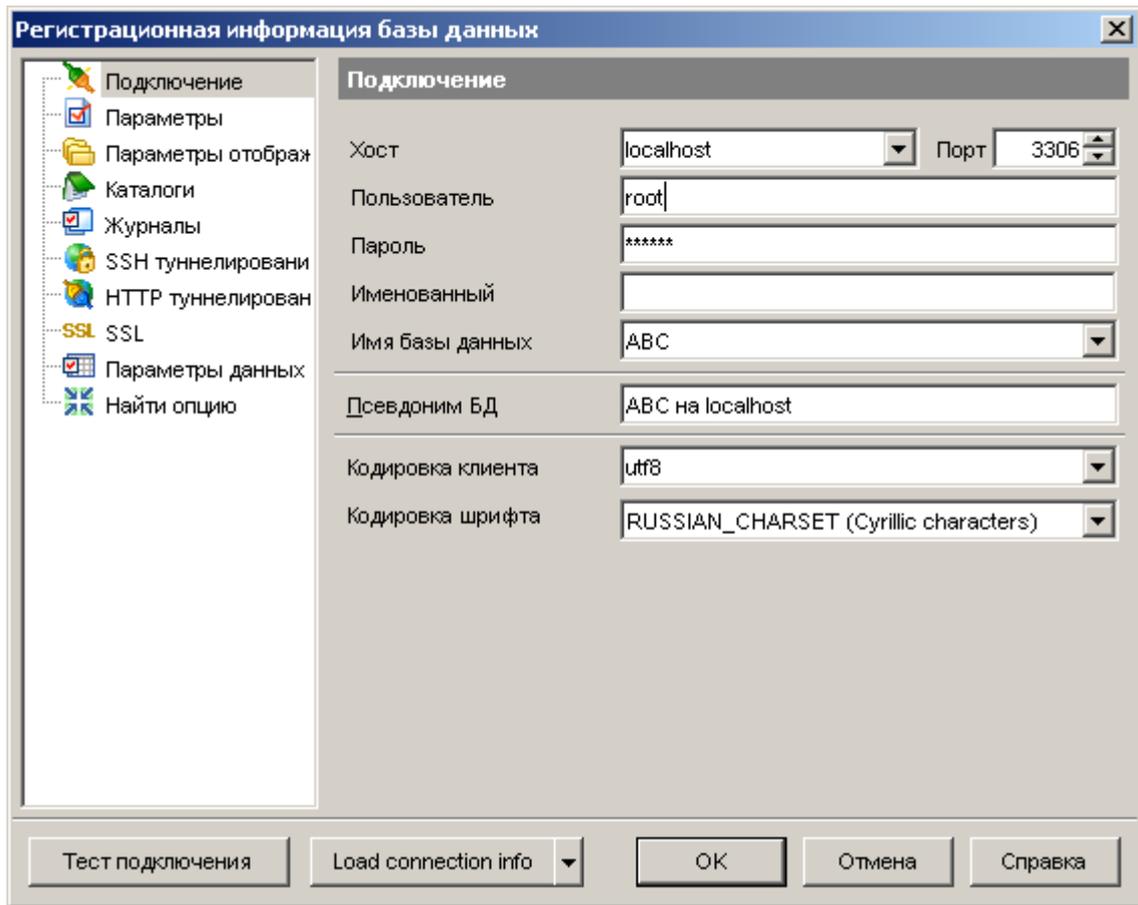
Пользователь: root

Пароль: 0000

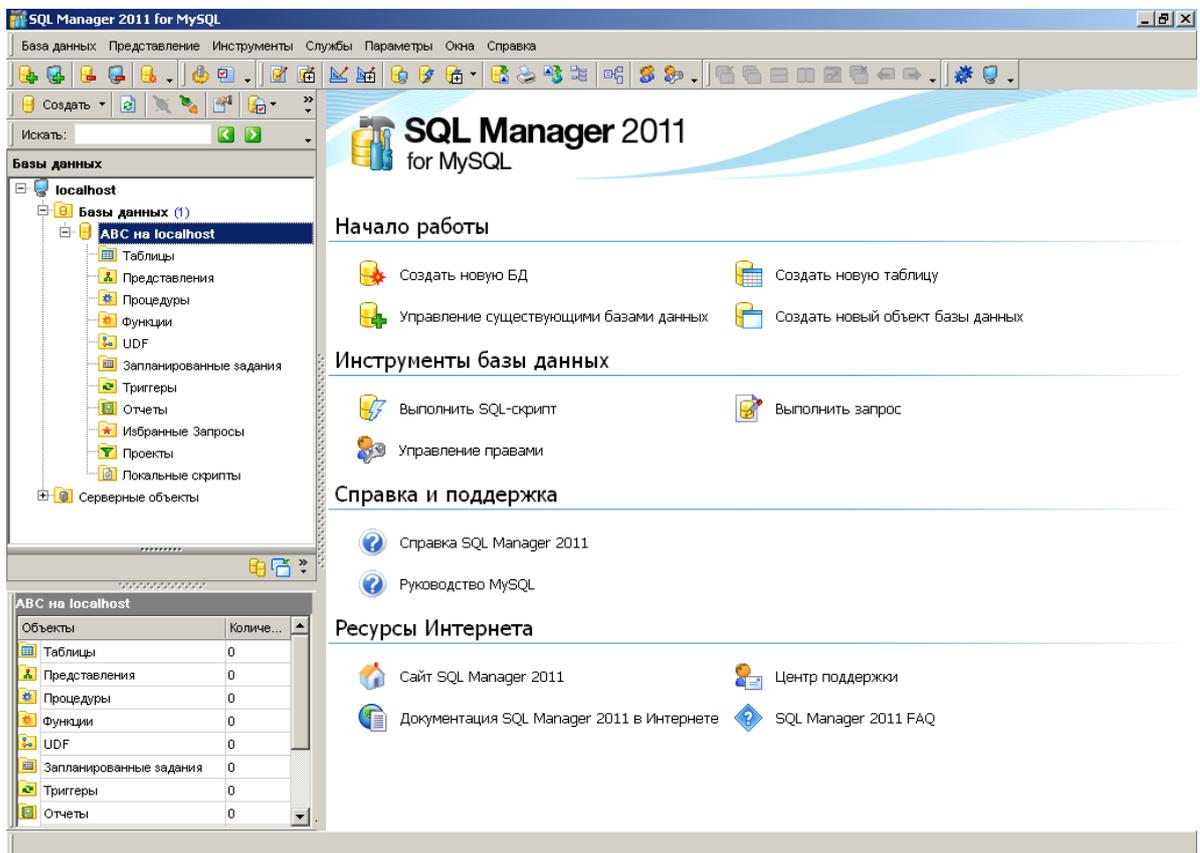


2.5 Далее будет сгенерирован SQL-код для создания базы данных. Нажимаем кнопку «Запустить».

2.6. В окне «Регистрационная информация базы данных» можно настроить параметры будущей базы данных, либо оставить их по умолчанию.



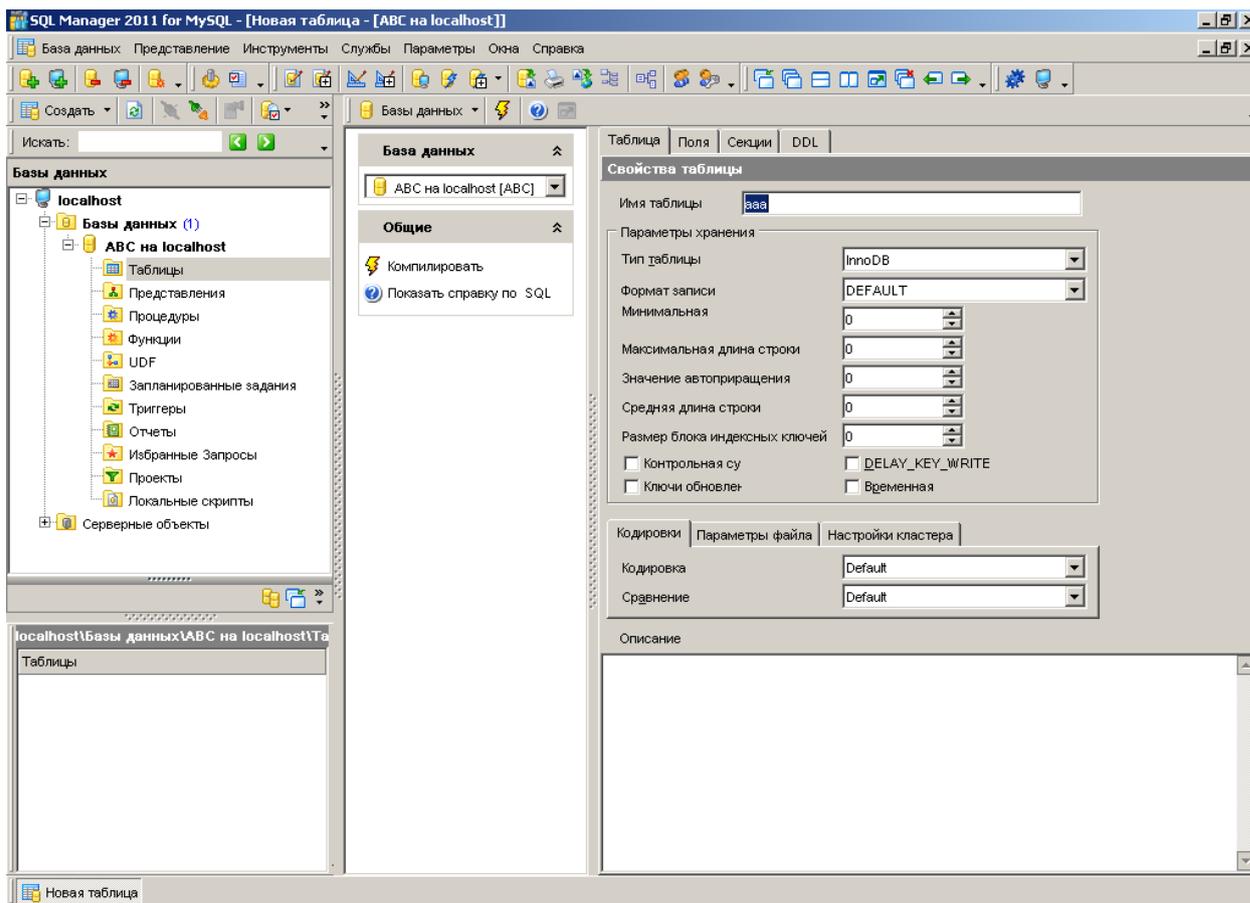
3. Открываем созданную базу данных. Изучаем интерфейс и возможности приложения (в соответствии с теорией).



4. Далее необходимо вручную создать таблицы своего варианта.

4.1 Выбираем команду «Создать новую таблицу» (или в выпадающем меню при нажатии правой кнопкой мыши по компоненту «Таблицы» выбираем команду «Новый объект: таблица»).

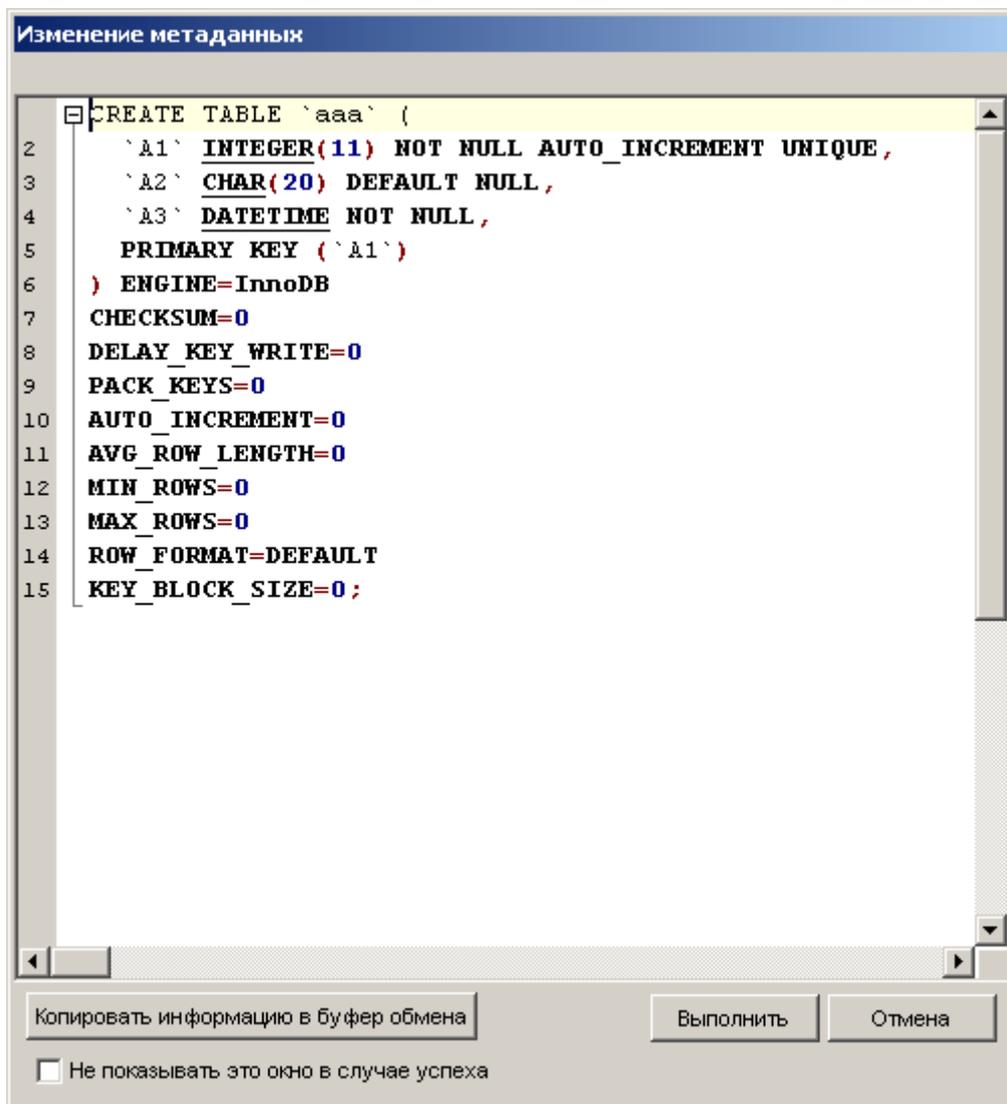
4.2. В открывшемся диалоговом окне необходимо указать параметры будущей таблицы.



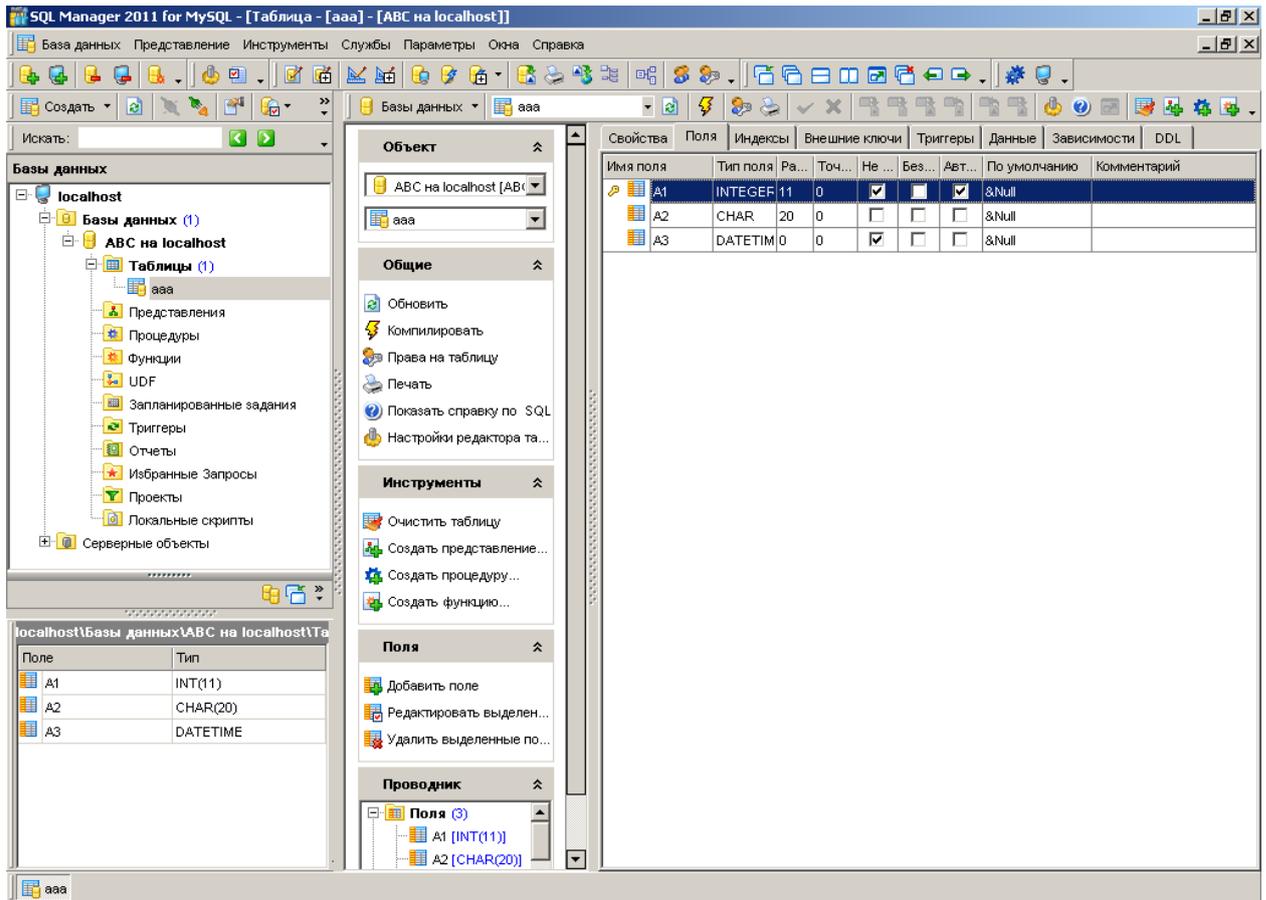
4.3. На вкладке «Поля» необходимо указать поля, входящие в таблицу, в том числе и будущие внешние ключи (FK).



4.5. Нажимаем кнопку «Компилировать» (Ctrl + F9). В появившемся окне нажимаем кнопку «Выполнить».



4.6. В открывшем окне после выполнения SQL-запроса появляется интерфейс для редактирования данных таблицы, в том числе для создания отношений между таблицами посредством внешних ключей (вкладка «Внешние ключи»).

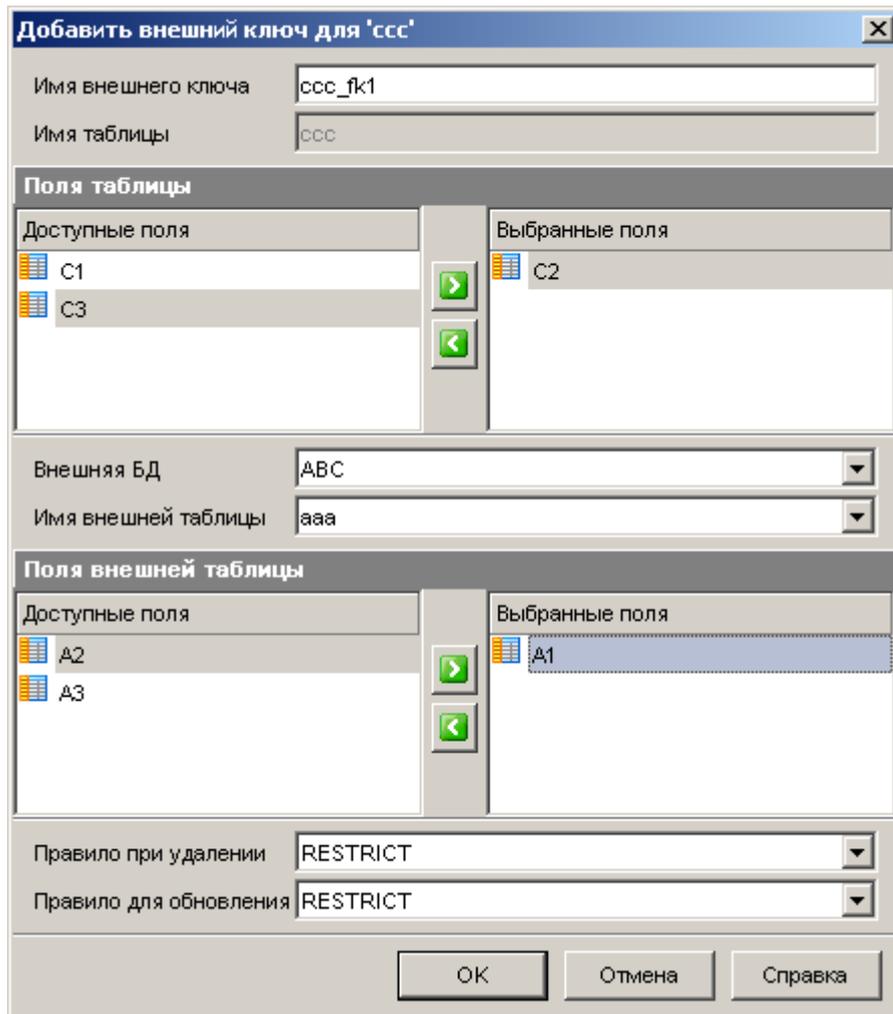


5. Добавляем все таблицы в соответствии с вариантом.

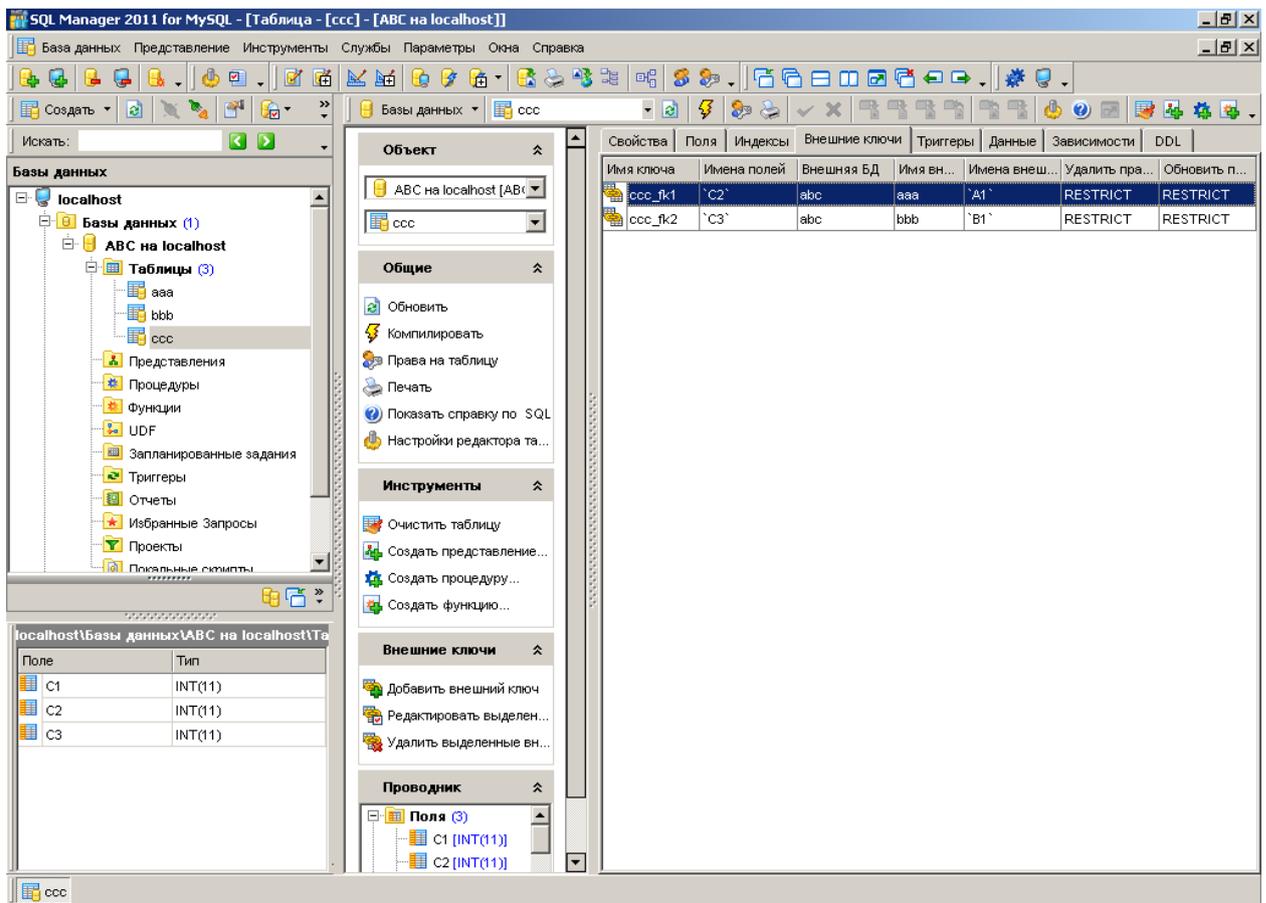
6. Создайте внешние ключи.

6.1. Открываем таблицу, содержащую внешние ключи. Переходим на вкладку Внешние ключи.

6.2. Связываем поля между собой.



6.3. Связываем все необходимые поля.

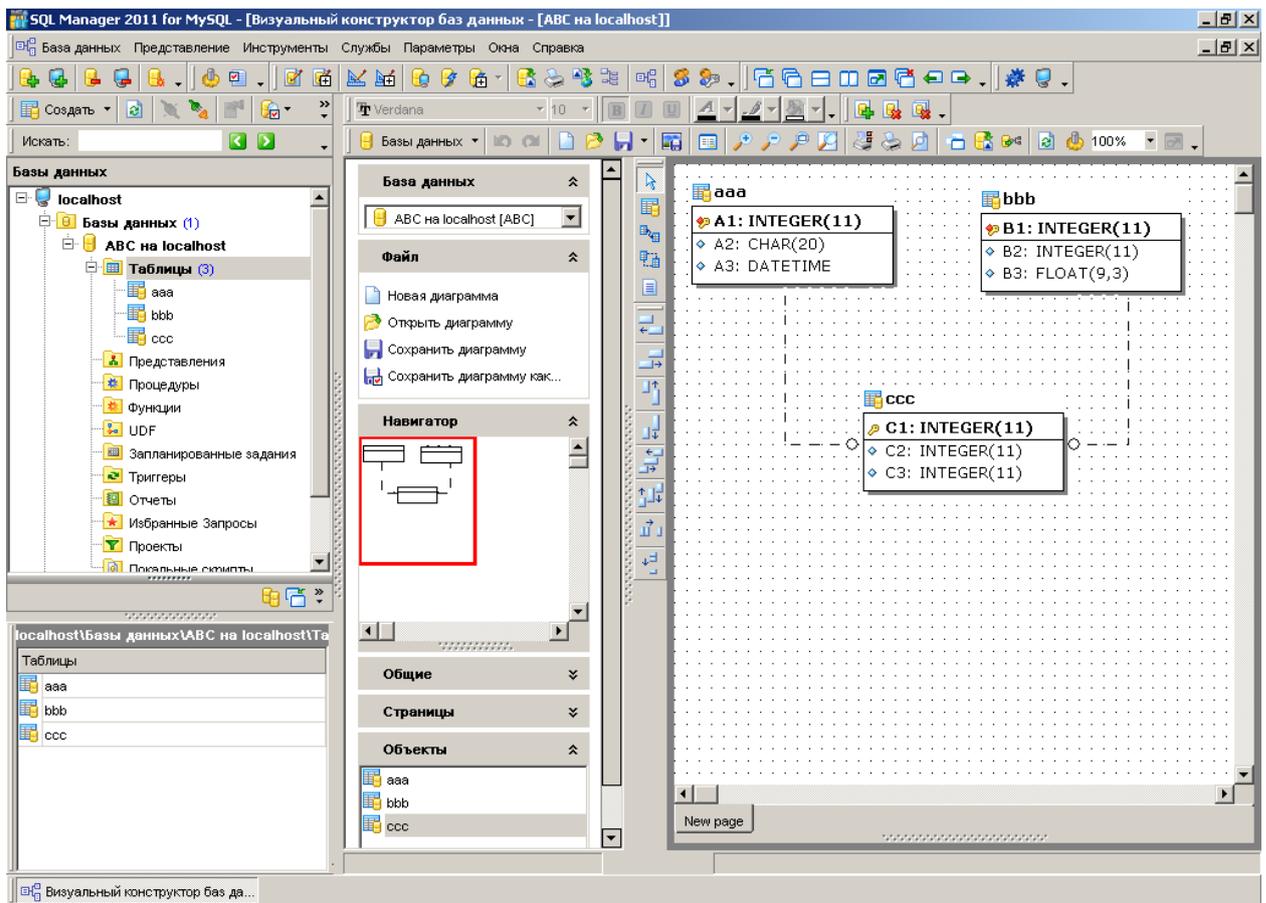


7. Создайте диаграмму базы данных.

7.1. Открываем Визуальный конструктор баз данных:

Инструменты -> Визуальный конструктор баз данных.

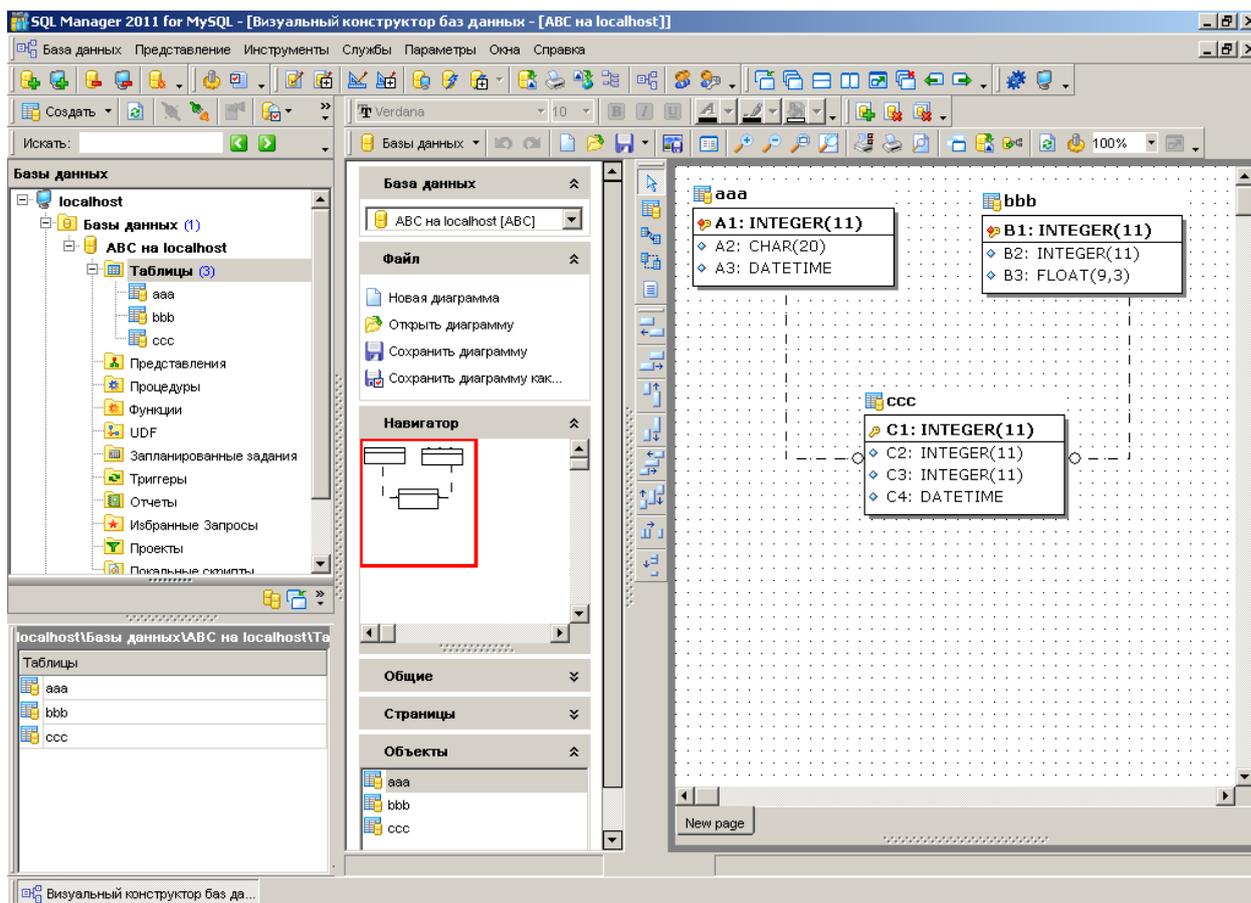
7.2. Добавить все имеющиеся объекты.



7.3. Сохранить диаграмму.

8. Внесите изменения в одну из таблиц с помощью диаграммы.

8.1. Выбираем таблицу для внесения изменений. Нажимаем правой кнопкой мыши и выбираем желаемое изменение, например, команда «Новое поле».



8.2. Далее выбираем команду на панели инструментов «Реконструирование» и вносим изменения в существующую БД.

9. Свяжите студию моделирования БД с открытой студией управления СУБД.

9.1. Открываем приложение Toad Data Modeler.

9.2. Вызываем мастер подключений:

File -> Reverse Engineering -> Connections

9.3 Далее вводим:

- имя подключения;
- выбираем БД для подключения (MySQL 5.5);
- Connection via TCP/IP
- Указываем имя пользователя (root) и пароль (0000).
  - Выбираем объекты реинжиниринга.
  - Настраиваем дополнительные параметры реинжиниринга (по умолчанию не требуется).

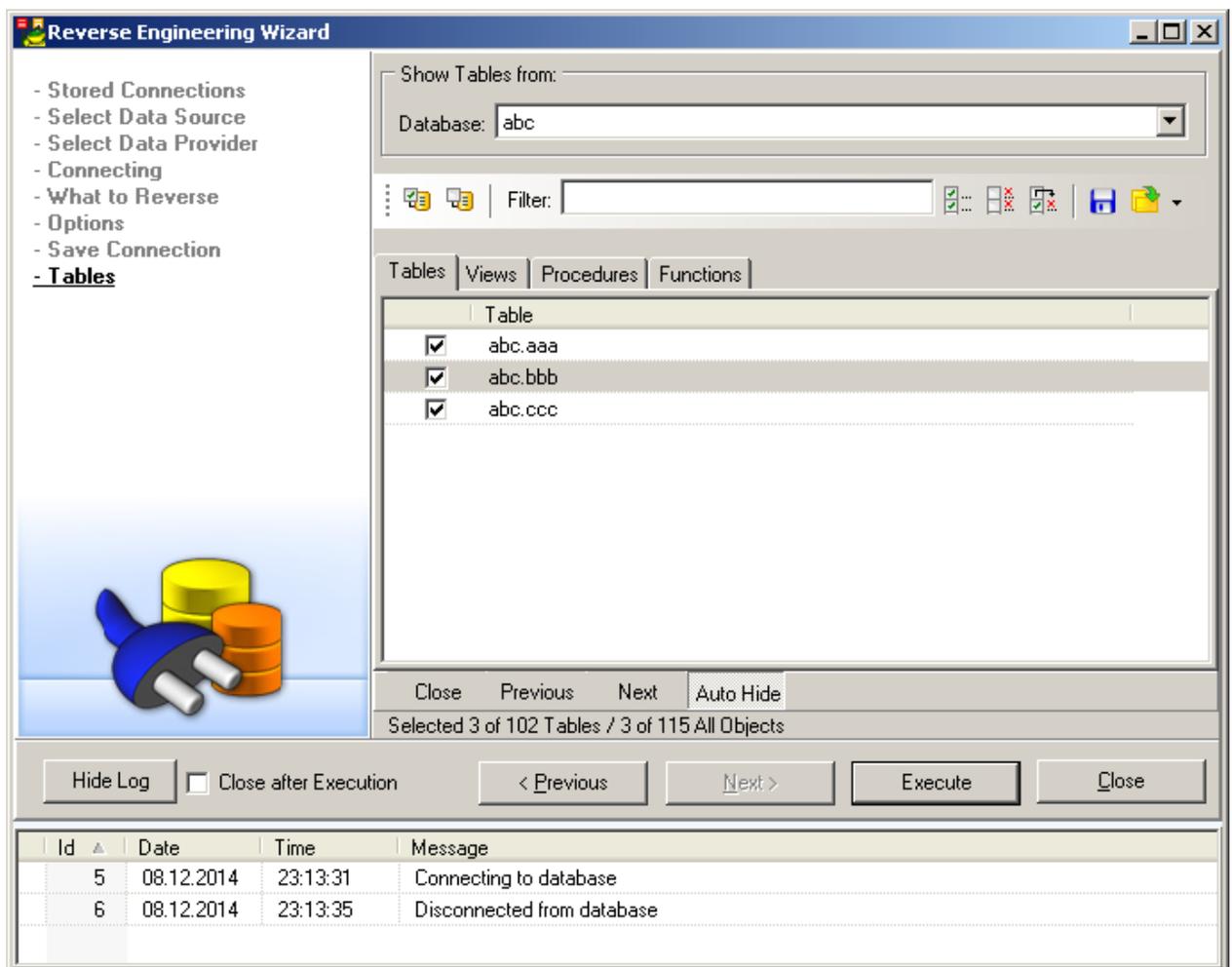
10. Получите модель базы данных путем реинжиниринга существующей на сервере.

10.1. Нажимаем на кнопку

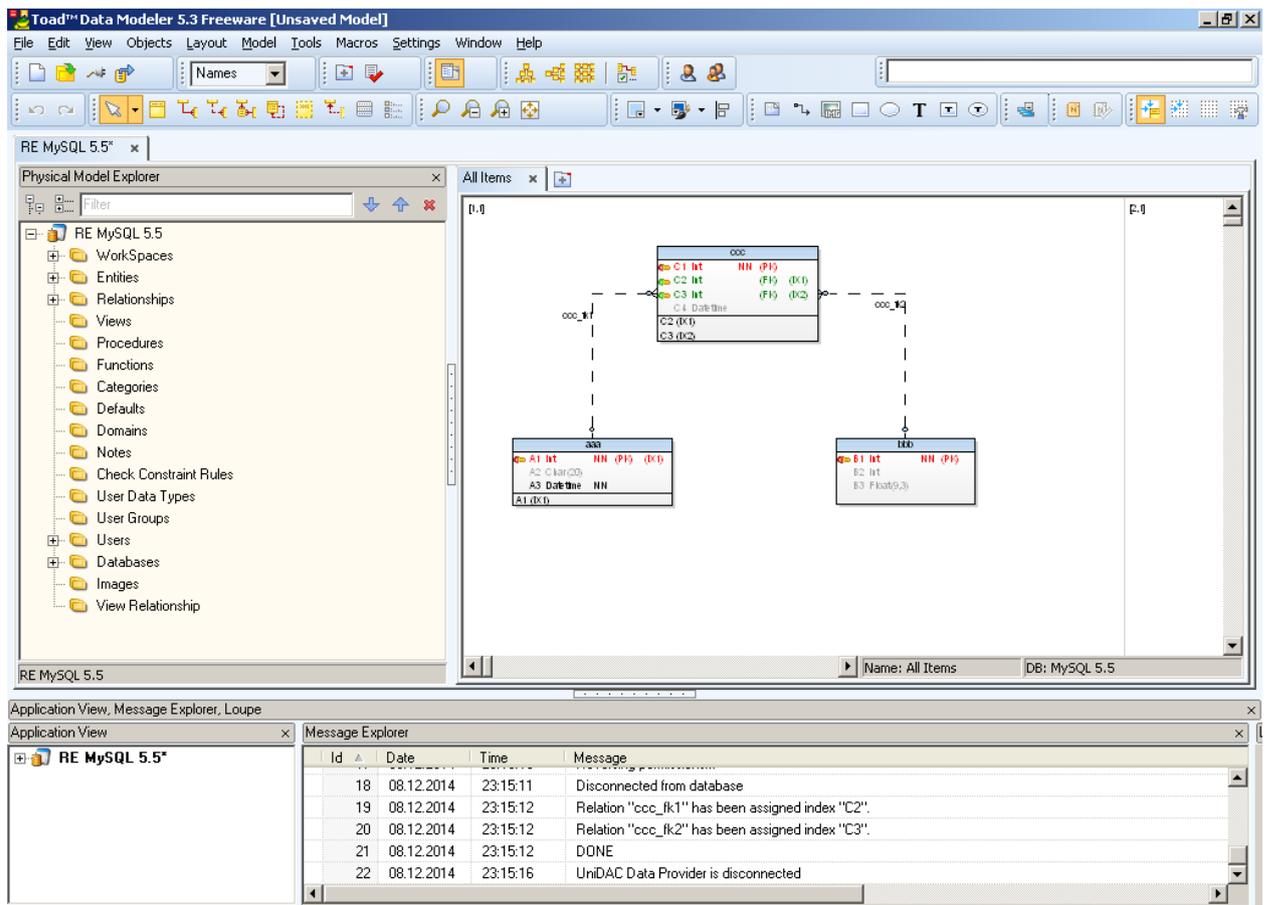
Или открываем мастер реинжиниринга

File -> Reverse Engineering -> Reverse Engineering Wizard

10.2 Выбираем объекты для реинжиниринга. Нажимаем кнопку Execute.



10.3. Сохранить полученную физическую модель.



### Задание на лабораторную работу

1. Запустить MySQL сервер.
2. Создать базу данных в студии SQL Manager Lite for MySQL.
3. Изучить интерфейс и возможности студии SQL Manager Lite for MySQL.
4. Изучить порядок создания таблиц.
5. Спроектировать таблицы в соответствии с вариантом.
6. Создать внешние ключи.
7. Создать диаграмму базы данных.
8. Внести изменения в одну из таблиц с помощью диаграммы.
9. Связать студию моделирования БД с студией управления СУБД.
10. Получить модель базы данных путем реинжиниринга существующей на сервере.