

ЗАДАНИЕ 3 ДЛЯ САМОСТОЯТЕЛЬНОЙ ПРАКТИЧЕСКОЙ РАБОТЫ ПО ДИСЦИПЛИНЕ «ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА БАЗ И ХРАНИЛИЩ ДАННЫХ»

1. Изучить методологию моделирования данных.
2. Построить логическую модель данных на основе разработанной функциональной модели предметной области IDEF0 и модели DFD.

МЕТОДОЛОГИИ МОДЕЛИРОВАНИЯ ДАННЫХ.

МОДЕЛЬ "СУЩНОСТЬ-СВЯЗЬ"

3.1. Основные понятия модели «сущность-связь»

На этапе инфологического проектирования базы данных должна быть построена модель предметной области, не привязанная к конкретной СУБД, понятная не только разработчикам информационной системы, но и экономистам, менеджерам и другим специалистам. В то же время модель предметной области должна максимально точно отражать семантику предметной области и позволять легко перейти к модели данных конкретной СУБД.

Такими моделями являются *модели "сущность-связь"*. Известно несколько методологий построения моделей "сущность-связь". Наибольшее распространение получила методология IDEF1X. Рассмотрим построение моделей "сущность-связь", ориентируясь на продукт СА ERwin Data Modeler. ERwin имеет два уровня представления модели:

- *Логический уровень*, соответствующий инфологическому этапу проектирования и не привязанный к конкретной СУБД. Модели логического уровня оперируют с понятиями сущностей, атрибутов и связей, которые на этом уровне именуются на естественном языке (в нашем случае – на русском) так, как они называются в реальном мире.
- *Физический уровень* – это отображение логической модели на модель данных конкретной СУБД. Одной логической модели может соответствовать несколько физических моделей. Причем, Erwin (как и другие CASE-системы проектирования баз данных) позволяет автоматизировать отображение логической модели на физическую.

Модель "сущность-связь" строится в виде диаграммы "сущность-связь", основными компонентами которой являются *сущности* (Entity) и *связи* (Relationship). Отсюда происходят часто используемые названия модели (ER-модель) и диаграммы (ER-диаграмма).

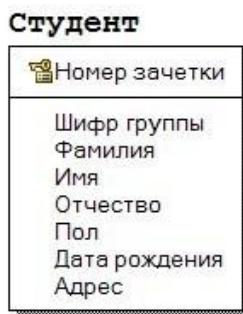
3.1.1. Сущности и атрибуты

Сущность – это абстракция множества предметов или явлений реального мира, информацию о которых надо сохранить. Все экземпляры сущности имеют одинаковые характеристики и подчиняются одним и тем же правилам поведения. Например, можно

выделить сущность **СТУДЕНТ**. Экземплярами сущности **СТУДЕНТ** будут данные о конкретных студентах. Сущность должна иметь *имя* – существительное в единственном числе.

Сущности обладают определенными свойствами – атрибутами. *Атрибут* – это абстракция *одной* характеристики объекта или явления реального мира. Каждый атрибут должен иметь *имя* – существительное в единственном числе, и получать значение из некоторого множества допустимых значений – *домена*.

У каждой сущности должен быть выделен идентификатор, или первичный ключ. *Первичный ключ* – это один или несколько атрибутов, однозначно определяющих каждый экземпляр сущности. Если первичный ключ состоит из нескольких атрибутов, то он называется *составным*. Первичный ключ не должен изменяться и принимать неопределенное значение (NULL). Ключ должен быть *компактным*, т. е. не содержать слишком много атрибутов. Сущность может иметь несколько *потенциальных ключей*, из которых должен быть выбран первичный ключ. Иногда приходится использовать *искусственный первичный ключ* (некоторый номер или код), когда ключ содержит слишком много атрибутов (в пределе каждый экземпляр сущности может определяться всем множеством атрибутов). Используется также понятие *внешнего ключа*. Внешний ключ – это первичный ключ другой сущности, который миграирует (копируется) в сущность и служит для связи сущностей.



Пример сущности

Каждая сущность должна сопровождаться *описанием*. Описание сущности должно объяснять ее смысл, а не то, как будет использоваться информация данной сущности. Описание должно быть ясным, полным и непротиворечивым, понятным специалистам предметной области.

Сущности и атрибуты выделяются в результате анализа требований к системе. При выборе атрибутов целесообразно придерживаться следующих правил (не входящих в IDEF1X), позволяющих перейти к физической модели, находящейся в третьей нормальной форме:

1. Атрибуты должны быть неделимыми.

2. Каждый неключевой атрибут должен полностью зависеть от составного ключа, а не от части ключа.
3. Не должно существовать транзитивных зависимостей атрибутов от ключа. Например, если ключ **ТАБЕЛЬНЫЙ_НОМЕР** определяет атрибут **НОМЕР_ОТДЕЛА**, а **НОМЕР_ОТДЕЛА** определяет **ТЕЛЕФОН**, то **ТАБЕЛЬНЫЙ_НОМЕР** транзитивно определяет **ТЕЛЕФОН**.

3.1.2. Связи

Связи между объектами реального мира отражаются в виде связей (*отношений, ассоциаций*) между сущностями.

Отношение – это ассоциация или "связь" между двумя сущностями. Отношение представляется в модели линией, соединяющей две сущности, и именем отношения – глагольной конструкцией, которая описывает, как две сущности зависят друг от друга. Имена сущностей, соединенные именем отношения, должны образовывать осмысленную фразу, описывающую бизнес-правило отношения. Например, **СТУДЕНТ <Обучается в> УЧЕБНАЯ ГРУППА**. В примере имя отношения показано в угловых скобках. Отношения двунаправлены, поэтому должны иметь имена в каждом направлении.

Отношение обладает следующими *свойствами*:

- степень;
- направленность;
- тип;
- мощность;
- обязательность.

Степень отношения представляет собой число сущностей, ассоциированных с отношением. Чаще всего используются *бинарные отношения*, связывающие две сущности. *Унарные* или *рекурсивные отношения* представляют случаи, когда экземпляр сущности связан с другим экземпляром той же самой сущности. Часто унарные или рекурсивные отношения рассматриваются как бинарные рекурсивные отношения, связывающие экземпляр сущности с другим ее экземпляром.

Направленность отношения указывает на исходную сущность в отношении.

Родительская сущность - сущность, из которой отношение исходит.

Подчиненная (дочерняя) сущность – сущность, которой отношение заканчивается.

Направленность отношения определяется взаимосвязью между сущностями и зависит от типа и мощности отношения. В отношении между независимой и зависимой сущностями отношение исходит из независимой сущности и заканчивается в зависимой сущности. Если обе сущности независимые, отношение симметрично. В отношении *один-ко-многим* родительской является сущность, входящая в отношение однократно. Отношения *многие-ко-многим* симметричны. Ключ родительской сущности *мигрирует*

(повторяется) в дочерней сущности. Такой мигрировавший ключ в дочерней сущности называется *внешним ключом*. Внешний ключ в зависимости от типа связи может стать частью составного ключа дочерней сущности или неключевым атрибутом дочерней сущности. С помощью внешнего ключа экземпляр дочерней сущности *ссылается* на соответствующий экземпляр родительской сущности.

В ERwin отношение между двумя сущностями или сущности самой с собой может принадлежать к одному из следующих *типов*:

- идентифицирующее отношение;
- неидентифицирующее отношение;
- типизирующее отношение;
- отношение многие-ко-многим;
- рекурсивное отношение.

Каждый тип отношений определяет поведение атрибутов первичного ключа, когда они мигрируют из родительской сущности в подчиненную.

Идентифицирующим является отношение между двумя сущностями, в котором каждый экземпляр подчиненной сущности идентифицируется значениями атрибутов родительской сущности. Это означает, что экземпляр подчиненной сущности зависит от родительской сущности и не может существовать без экземпляра родительской сущности. В идентифицирующем отношении единственный экземпляр родительской сущности связан с множеством экземпляров подчиненной. Атрибуты первичного ключа родительской сущности мигрируют в атрибуты подчиненной, чтобы стать там атрибутами первичного ключа.



Пример идентифицирующего отношения

На рисунке представлено идентифицирующее отношение между сущностями **Заказ** и **Состав заказа**. В сущности **Заказ** использован искусственный первичный ключ **ID заказа** (Идентификатор заказа), который мигрировал в дочернюю сущность **Состав заказа** и стал там первичным ключом. В реальной диаграмме атрибут **Заказчик**, скорее всего, будет являться мигрировавшим ключом сущности, описывающей заказчиков.

Пример показывает, что дочерняя сущность не может существовать без родительской. Этот пример также демонстрирует, как показать переменный состав заказа: в заказ может входить произвольное количество товаров.

Неидентифицирующим называется отношение между двумя сущностями, в котором каждый экземпляр подчиненной сущности не зависит от значений атрибутов родительской сущности. Это означает, что экземпляр подчиненной сущности не зависит от родительской сущности и может существовать без экземпляра родительской сущности. В неидентифицирующем отношении единственный экземпляр родительской сущности связан с множеством экземпляров подчиненной. Атрибуты первичного ключа родительской сущности мигрируют в подчиненную, чтобы стать там *неключевыми* атрибутами.



Пример обязательного неидентифицирующего отношения

На рисунке показано неидентифицирующее отношение между сущностями **Группа** и **Студент**. Каждое из этих отношений имеет собственный первичный ключ. Первичного ключа родительской сущности **Группа** мигрировал в подчиненную сущность **Студент** и стал там неключевым атрибутом.

На данном примере рассмотрим также понятие *обязательности* отношения. Неидентифицирующее отношение называется *обязательным* (No Nulls), если все экземпляры дочерней сущности должны участвовать в отношении.

Неидентифицирующее отношение называется *необязательным* (Nulls Allowed), если некоторые экземпляры дочерней сущности могут не участвовать в отношении.

Очевидно, что студент должен принадлежать одной из учебных групп.



Пример необязательного неидентифицирующего отношения

На рисунке показан пример необязательного отношения: допускается, что сотрудник может не принадлежать ни одному из отделов.

Типизирующими называются отношения между родительской и одной или более подчиненными сущностями, когда сущности разделяют общие характеристики. Такие отношения называются еще *иерархией наследования* или *иерархией категорий*. Типизирующие отношения используются в том случае, когда экземпляр родительской сущности определяет различные наборы атрибутов в подчиненных сущностях.

Например, имеются различные категории сотрудников, отличающиеся только небольшим количеством атрибутов. Для каждой категории необходимо указать *дискриминатор* – атрибут родительской сущности, показывающий, как отличить одну категориальную сущность от другой. На рисунке дискриминатором является атрибут **Тип**.

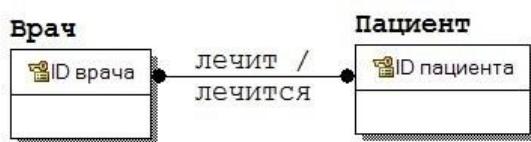


Пример полной категории иерархии наследования

На рисунке показана *полная категория*, т. е. каждый экземпляр сущности сотрудник относится к одной из перечисленных категорий. Возможна *неполная категория*, когда существуют экземпляры родительской сущности, не имеющие

соответствующих экземпляров в дочерних сущностях (значок категории содержит одну горизонтальную линию).

Отношения *многие-ко-многим* возникают тогда, где один экземпляр одной сущности связан с несколькими экземплярами другой, и один экземпляр этой другой сущности также связан с несколькими экземплярами первой сущности. Эти отношения также называют неспецифическими. Отношения *многие-ко-многим* *используются только на логическом уровне*. На физическом уровне эти отношения реализуются за счет использования *ассоциативной сущности*, содержащей ключи родительских сущностей и, возможно, дополнительные атрибуты. Для большей наглядности диаграммы желательно ввести ассоциативные сущности на логическом уровне.



Пример связи "многие-ко-многим"



Пример использования ассоциативной сущности

Рекурсивное отношение – это *неидентифицирующее* отношение между двумя сущностями, которое указывает, что экземпляр сущности может быть связан с другим экземпляром той же самой сущности. При рекурсивном отношении родительская и подчиненная сущности совпадают.



Примеры реализации рекурсивных отношений с использованием имени роли и без него в сущности **Сотрудник**

На рисунке показаны примеры двух реализаций рекурсивного отношения для сущности **Сотрудник**, с использованием *имени роли* и без него. Имя роли показывает,

какую роль играет внешний ключ в сущности. В данном примере внешний ключ задает табельный номер. Обратите внимание, что ERwin "унифицирует" атрибуты внешнего ключа и первичного ключа, когда имя роли не используется. Использование имени роли приводит к размещению внешнего ключа в качестве неключевого атрибута.

Мощность отношения задает максимальное число экземпляров одной сущности, которые могут быть связаны с экземплярами другой сущности. Мощность отношения определяется для обеих сторон отношения – для исходной и завершающей сущностей. Количество элементов определяет максимальное количество экземпляров сущностей, участвующих в отношении, в то время как обязательность определяет минимальное число экземпляров. Количество элементов часто выражается как один или много. Один и много могут появляться в трех различных комбинациях:

Один-к-одному (1:1) – один и только один экземпляр сущности связан с одним и только одним экземпляром другой сущности.

Один-ко-многим (1:N) – один и только один экземпляр родительской сущности связан со многими экземплярами подчиненной сущности.

Многие-ко-многим (M:N) – много экземпляров одной сущности связаны со многими экземплярами другой сущности (также называется неспецифическим отношением).

В отношении *один-к-одному* один и только один экземпляр сущности связан с одним и только одним экземпляром другой сущности. Это редкий случай отношения, следует рассмотреть возможность объединения двух отношений в одно. Но, например, в отношении сущностей **Факультет** и **Сотрудник** целесообразнее установить связь *один-к-одному*, чем заносить данные о декане в сущность **Факультет**.

В отношении *один-ко-многим* один и только один экземпляр родительской сущности связан со многими экземплярами дочерней сущности. Это наиболее распространенное отношение.

Отношение *многие-ко-многим* уже было рассмотрено.

В ERwin отношение изображается линией с точкой на конце "много". Кроме общего случая мощности отношения 0, 1 или много можно задать частные случаи отношений:

- 1 или много (обозначается буквой **p**),
- 0 или 1 (обозначается буквой **z**),
- конкретное число экземпляров дочерней сущности (обозначается числом экземпляров, например, 6).

В ERwin реализована также методология IE (Information Engineering), которая принципиально не отличается от методологии IDEF1X.

3.1.3. Правила ссылочной целостности

Целостность данных является понятием базы данных. ERwin позволяет в модели "сущность-связь" задать *правила ссылочной целостности* (возможно также задание ограничений на значения атрибутов). При генерации схемы базы данных ERwin генерирует правила декларативной ссылочной целостности и триггеры. То есть, правила ссылочной целостности задаются в модели "сущность-связь", а реализуются в построенной по этой модели реляционной базе данных. Соответствие между моделью и базой данных легко устанавливается, учитывая, что сущностям модели соответствуют отношения реляционной базы данных, а экземплярам сущностей – кортежи отношений.

Так как внешние ключи кортежей дочернего отношения служат ссылками на соответствующие кортежи родительского отношения, то эти ссылки не должны указывать на несуществующие кортежи. Это определяет следующее *правило целостности внешних ключей* (правило ссылочной целостности): для каждого значения внешнего ключа должно существовать соответствующее значение первичного ключа в родительском отношении.

Ссылочная целостность может нарушаться в результате операций с кортежами отношений. Таких операций три: вставка, обновление и удаление кортежей в отношениях. Рассмотрим эти операции для родительского и дочернего отношений.

При операциях с кортежами *родительского отношения* возможны следующие ситуации:

1. *Вставка кортежа в родительское отношение.* Так как допустимо существование кортежей родительского отношения, на которые нет ссылок из дочерних отношений, то *вставка кортежа в родительское отношение не нарушает ссылочной целостности*.

2. *Обновление кортежа родительского отношения.* При обновлении кортежа родительского отношения может измениться значение ключа. Если есть экземпляры дочернего отношения, ссылающиеся на обновляемый кортеж родительского отношения, то значения их внешних ключей станут некорректными. *Обновление кортежа в родительском отношении может привести к нарушению ссылочной целостности, если это обновление затрагивает значение ключа.*

3. *Удаление кортежа родительского отношения.* При удалении кортежа родительского отношения удаляется значение ключа. Если есть кортежи дочернего отношения, ссылающиеся на удаляемый кортеж родительского отношения, то значения их внешних ключей станут некорректными. *Удаление кортежа родительского отношения может привести к нарушению ссылочной целостности.*

При операциях с кортежами *дочернего отношения* возможны следующие ситуации:

1. Вставка кортежа дочернего отношения может привести к нарушению ссылочной целостности, если вставляемое значение внешнего ключа некорректно.
2. Обновление кортежа дочернего отношения может привести к нарушению ссылочной целостности при некорректном изменении значения внешнего ключа.
3. При удалении кортежа дочернего отношения ссылочная целостность не нарушается.

Таким образом, ссылочная целостность в принципе может быть нарушена при выполнении одной из четырех операций:

- 1) обновление кортежа в родительском отношении;
- 2) удаление кортежа в родительском отношении;
- 3) вставка кортежа в дочернее отношение;
- 4) обновление кортежа в дочернем отношении.

Существуют две основные стратегии поддержания ссылочной целостности:

- 1) **RESTRICT (ОГРАНИЧИТЬ)** – не разрешать выполнение операции, приводящей к нарушению ссылочной целостности. Это самая простая стратегия, требующая только проверки, имеются ли кортежи дочернего отношения, связанные с некоторыми кортежами родительского отношения.
- 2) **CASCADE (КАСКАД)** – разрешить выполнение требуемой операции, но внести при этом необходимые поправки в других кортежах отношений так, чтобы не допустить нарушения ссылочной целостности и сохранить все имеющиеся связи. Изменение начинается в родительском отношении и каскадно выполняется в дочернем отношении. Так как дочернее отношение может быть родительским для некоторого третьего отношения, то может потребоваться выполнение каскадной стратегии и для этой связи и т.д. Это самая сложная стратегия, но она хороша тем, что при этом не нарушается связь между кортежами родительского и дочернего отношений.

Эти стратегии являются стандартными и присутствуют во всех СУБД, в которых имеется поддержка ссылочной целостности.

ERwin реализует также дополнительные стратегии поддержания ссылочной целостности (если они реализованы в целевой СУБД):

- 1) **NONE (НИКАКОЙ)** – никаких операций по поддержке ссылочной целостности не выполняется. В этом случае в дочернем отношении могут появляться некорректные значения внешних ключей, и вся ответственность за целостность базы данных ложится на приложение.
- 2) **SET NULL (УСТАНОВИТЬ В NULL)** – разрешить выполнение требуемой операции, но все возникающие некорректные значения внешних ключей заменять на неопределенные значения (null-значения). При этом кортежи дочернего отношения теряют всякую связь с кортежами родительского отношения.

3) **SET DEFAULT (УСТАНОВИТЬ ПО УМОЛЧАНИЮ)** – разрешить выполнение требуемой операции, но все возникающие некорректные значения внешних ключей изменять на некоторое значение, принятое по умолчанию. При этом должен существовать кортеж родительского отношения, первичный ключ которого принят как значение по умолчанию для внешних ключей. Этот кортеж нельзя удалять из родительского отношения, и в этом кортеже нельзя изменять значение ключа. Кроме того, как и в предыдущем случае, кортежи дочернего отношения теряют всякую связь с кортежами родительского отношения.

Рассмотрим применение стратегии поддержания ссылочной целостности.

При обновлении кортежа в родительском отношении допустимы следующие стратегии:

1. **RESTRICT** – не разрешать обновление, если имеется хотя бы один кортеж дочернего отношения, ссылающийся на обновляемый кортеж родительского отношения.
2. **CASCADE** – выполнить обновление и каскадно изменить значения внешних ключей во всех кортежах дочернего отношения, ссылающихся на обновляемый кортеж.
3. **SET NULL** – выполнить обновление и во всех кортежах дочернего отношения, ссылающихся на обновляемый кортеж, изменить значения внешних ключей на null-значение.
4. **SET DEFAULT** – выполнить обновление и во всех кортежах дочернего отношения, ссылающихся на обновляемый кортеж, изменить значения внешних ключей на некоторое значение, принятое по умолчанию.
5. **NONE** – выполнить обновление, не обращая внимания на нарушения ссылочной целостности.

При удалении кортежа в родительском отношении допустимы стратегии:

1. **RESTRICT** – не разрешать удаление, если имеется хотя бы один кортеж в дочернем отношении, ссылающийся на удаляемый кортеж.
2. **CASCADE** – выполнить удаление и каскадно удалить кортежи в дочернем отношении, ссылающиеся на удаляемый кортеж.
3. **SET NULL** – выполнить удаление и во всех кортежах дочернего отношения, ссылающихся на удаляемый кортеж, изменить значения внешних ключей на null-значение.
4. **SET DEFAULT** – выполнить удаление и во всех кортежах дочернего отношения, ссылающихся на удаляемый кортеж, изменить значения внешних ключей на некоторое значение, принятое по умолчанию.
5. **NONE** – выполнить удаление, не обращая внимания на нарушения ссылочной целостности.

При вставке кортежа в дочернее отношение возможны стратегии:

1. **RESTRICT** – не разрешать вставку, если внешний ключ во вставляемом кортеже не соответствует ни одному значению потенциального ключа родительского отношения.
2. **SET NULL** – вставить кортеж, но в качестве значения внешнего ключа занести не предлагаемое пользователем некорректное значение, а null-значение.
3. **SET DEFAULT** – вставить кортеж, но в качестве значения внешнего ключа занести не предлагаемое пользователем некорректное значение, а некоторое значение, принятое по умолчанию.
4. **NONE** – вставить кортеж, не обращая внимания на нарушения ссылочной целостности

При обновлении кортежа в дочернем отношении возможны стратегии:

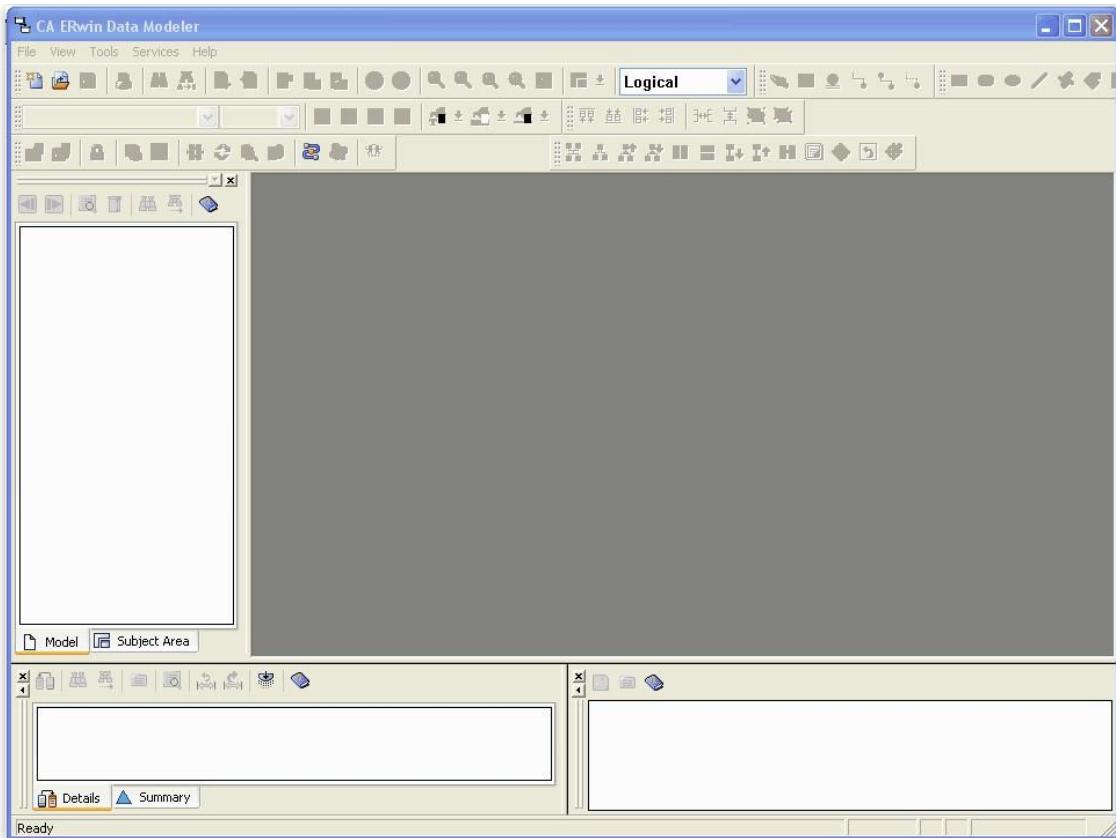
1. **RESTRICT** – не разрешать обновление, если внешний ключ в обновляемом кортеже становится не соответствующим ни одному значению потенциального ключа родительского отношения.
2. **SET NULL** – обновить кортеж, но в качестве значения внешнего ключа занести не предлагаемое пользователем некорректное значение, а null-значение.
3. **SET DEFAULT** – обновить кортеж, но в качестве значения внешнего ключа занести не предлагаемое пользователем некорректное значение, а некоторое значение, принятое по умолчанию.
4. **NONE** – обновить кортеж, не обращая внимания на нарушения ссылочной целостности.

3.2. Создание логической модели данных

3.2.1. Начало работы

Запуск программы осуществляется через меню **Пуск>Все программы>СА>ERwin>ERwin Data Modeler r7.3>ERwin Data Modeler** или через ярлык на рабочем столе.

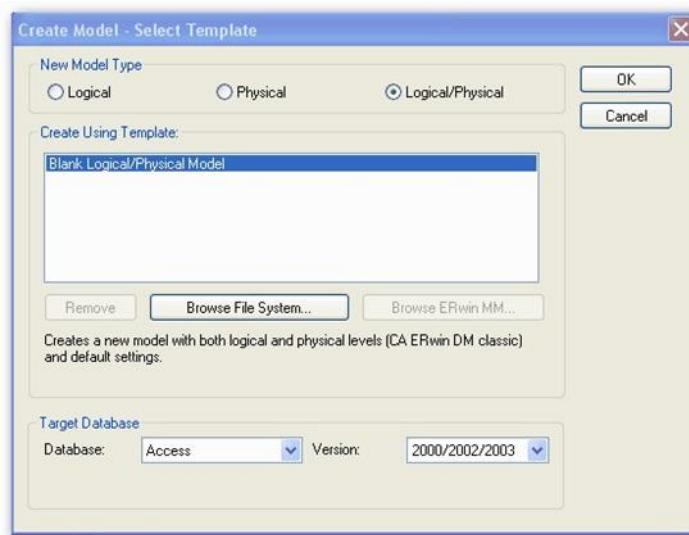
Порядок построения модели данных в среде ERwin рассмотрим на примере автоматизированной информационной системы "Реализация средств вычислительной техники", предназначеннной для учета продаж настольных компьютеров по заказам клиентов.



Главное окно программы **CA ERwin Data Modeler**

Создание новой модели начинается с выбора типа модели: *логическая*, *физическая*, *логико-физическкая*. При проектировании базы данных целесообразнее выбирать *логико-физическую* модель.

Выбор типа новой модели осуществляется в диалоге **Create Model**, вызываемом через **File>New** или кнопку создания нового файла .

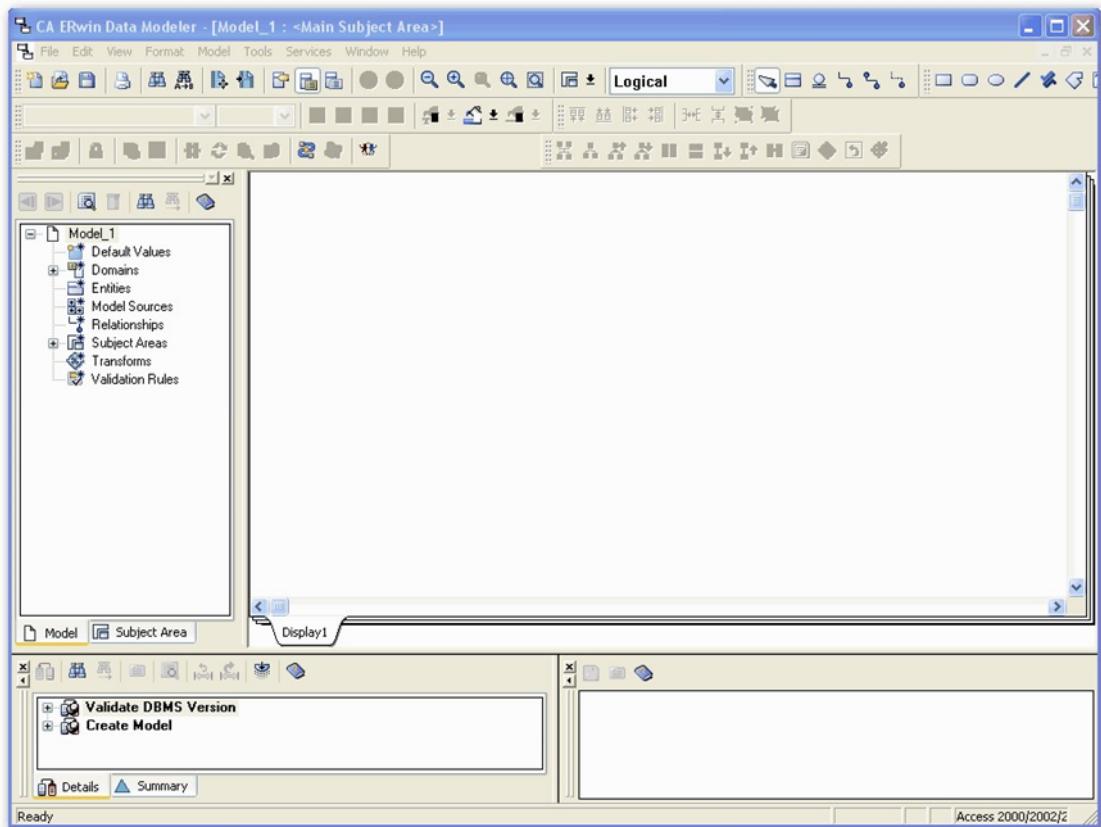


Диалог **Create Model**

При выборе физической или логико-физической модели в диалоге **Create Model** необходимо также указать необходимую СУБД и номер ее версии (в полях **Database** и **Version**).

Данный диалог позволяет также открыть существующую модель данных указанного типа посредством активизации кнопки **Browse File System...**.

В соответствии с выбранным действием при активизации кнопки **OK** открывается окно для построения новой модели данных (рис. 6.3) или окно редактирования модели, созданной ранее.



Окно для построения новой модели данных

3.2.2. Подуровни логического уровня модели данных

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

Если предварительно был выбран логико-физический уровень представления модели данных, то в *списке выбора для переключения между логической и физической моделью* (пункт **Logical**, расположенный на панели инструментов) автоматически будет выбрана логическая модель.



Пункт **Logical** на панели инструментов

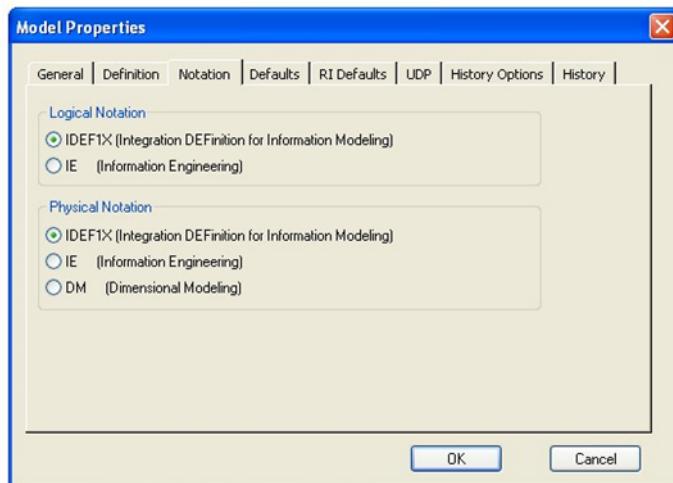
В противном случае тип модели необходимо выбрать из выпадающего меню данного пункта.



Выпадающее меню пункта **Logical**

На логическом уровне ERwin поддерживает две нотации (IE и IDEF1X), на физическом уровне – три нотации (IE, IDEF1X и DM). По умолчанию используется нотация IDEF1X (*Integration DEFinition for information modeling*).

Смену нотации можно сделать на вкладке **Notation** диалога **Model Properties**, вызываемого через меню **Model>Model Properties**.



Смена нотации на вкладке **Notation** диалога **Model Properties**

Для построения ER-модели используется панель инструментов, состав которой (условные графические обозначения) зависит от выбранного стандарта представления моделей и типа модели: логическая или физическая.

Палитра инструментов логического уровня

Вид кнопки	Назначение кнопки
	Указатель (режим мыши) – в этом режиме можно установить фокус на каком-либо объекте модели
	Создание новой сущности – для создания сущности необходимо щелкнуть левой кнопкой мыши по кнопке и один раз по свободному пространству на модели
	Создание категории – для установления категориальной связи (специальный тип связи между сущностями) необходимо щелкнуть левой кнопкой мыши по кнопке категории, затем один раз щелкнуть по родительской сущности и затем один раз по сущности-потомку
	Создание идентифицирующей связи
	Создание связи «многие ко многим»
	Создание неидентифицирующей связи

Поскольку стандарт IDEF1X используется по умолчанию, то на панели инструментов автоматически будут отображаться виды кнопок данного стандарта, описание которых приведено в таблице.

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

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

Диаграмма сущность-связь включает сущности и взаимосвязи, отражающие основные бизнес-правила предметной области.

Модель данных, основанная на ключах – более подробное представление данных. Данная модель включает описание всех сущностей и первичных ключей, необходимых для подробного описания предметной области.

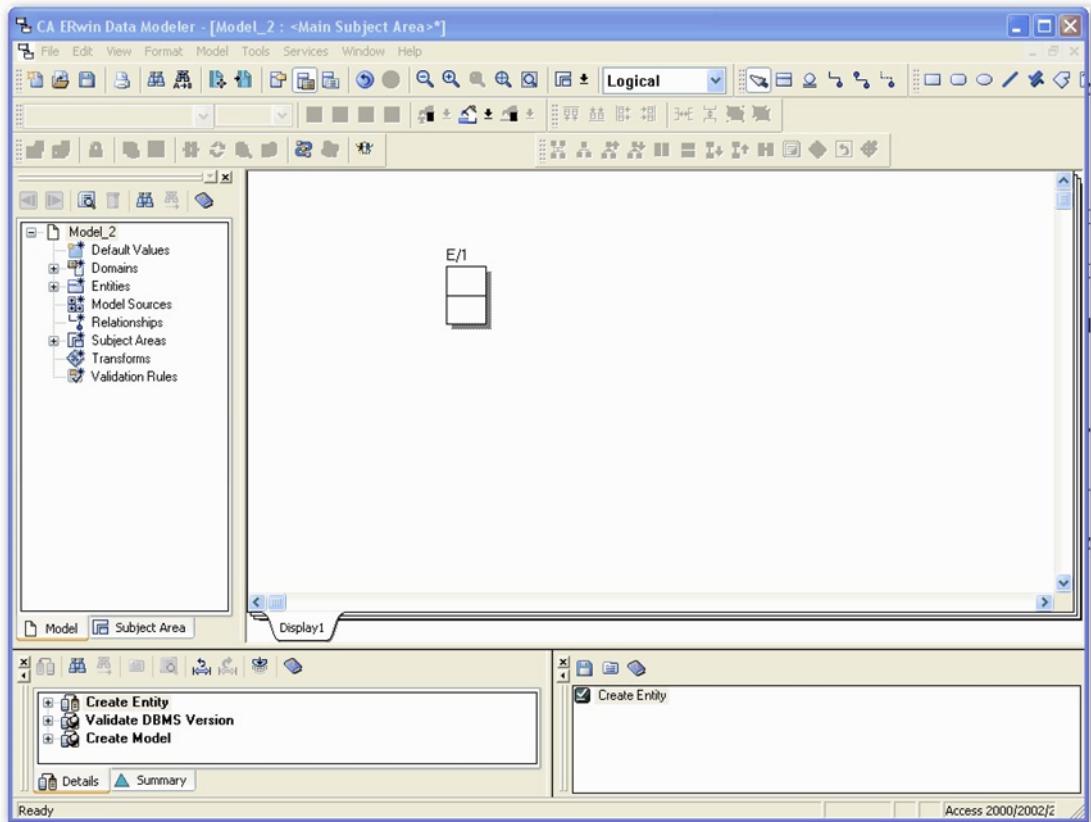
Полная атрибутивная модель данных – наиболее детальное представление структуры данных предметной области. Данная модель представляет данные в третьей нормальной форме и включает все сущности, атрибуты и связи.

3.2.3. Создание сущностей и атрибутов

Построение логической модели данных предполагает определение сущностей и атрибутов, т. е. необходимо определить, какая информация будет храниться в конкретной сущности и в конкретном атрибуте. Сущность можно определить как объект, событие или концепцию, информация о которых должна сохраняться.

Например, сущность **Клиент** (но не **Клиенты!**) может иметь атрибуты **Номер клиента**, **Фамилия клиента**, **Адрес клиента** и **Телефон клиента**. На уровне физической модели данных данной сущности может соответствовать таблица (отношение) **Client** с колонками **Client_number**, **Client_name**, **Client_address** и **Client_telephone**.

Для *внесения сущности в модель* необходимо щелкнуть левой кнопкой мыши по кнопке сущности , расположенной на панели инструментов, и затем щелкнуть один раз в поле проектирования модели на том месте, где необходимо расположить новую сущность. В результате в поле проектирования появится сущность с именем **E/1** по умолчанию.



Размещение в поле проектирования сущности с именем **E/1** по умолчанию

Определение имени сущности осуществляется через диалог **Entities**, который открывается через пункт **Entity Properties** (*свойства сущности*) контекстного меню по щелчку правой кнопкой мыши по выделенной сущности или пункта главного меню **Model/Entities**.

Диалог **Entities** содержит вкладки, которые позволяют назначить и редактировать свойства сущности: имя (**Name**), определение (**Definition**), объем (**Volumetrics**), примечания (**Note, Note2, Note3**), определяемые пользователем свойства (**UDP – User definition properties**), иконки (**Icon**), историю (**History**).



Диалог **Entities**

Вкладка **Definition** используется для ввода определения сущности в виде ее текстового описания. На логическом уровне определения позволяют четче понять объект, а на физическом уровне – их можно экспорттировать как часть схемы и использовать в реальной базе данных.

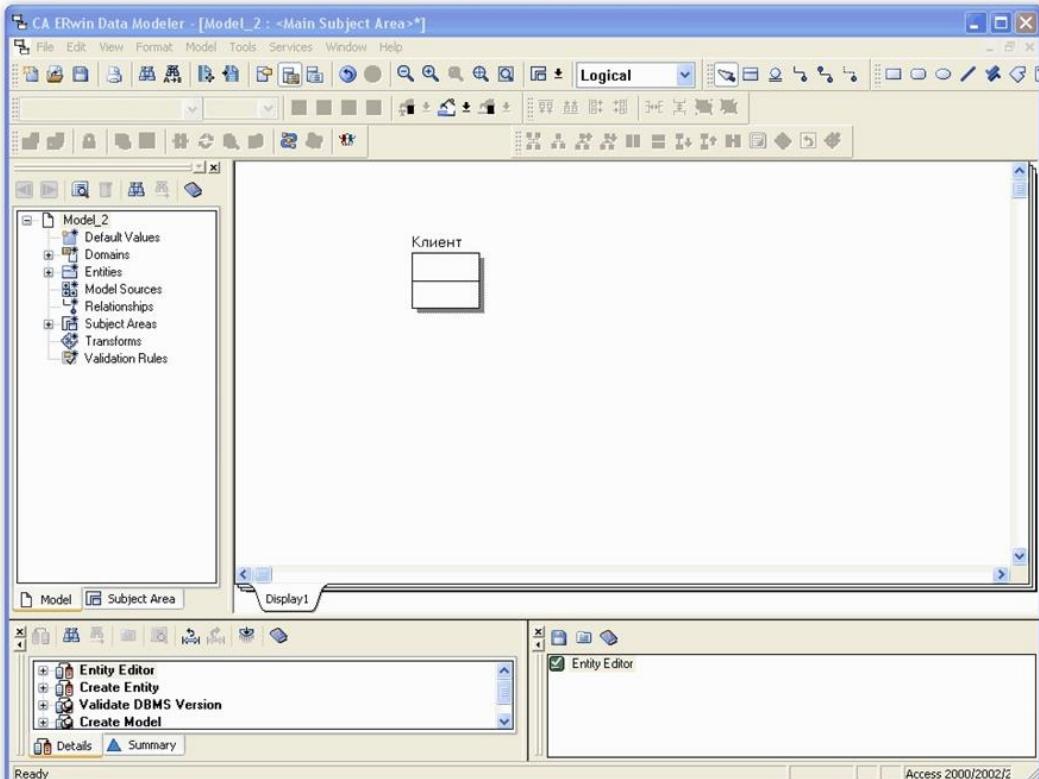
Вкладка **Volumetrics** позволяет указать предполагаемое начальное и максимальное количество экземпляров сущности и возможное их увеличение.

Вкладки **Note**, **Note2**, **Note3**, **UDP** служат для внесения дополнительных комментариев и определений к сущности, в частности:

- вкладка **Note** позволяет добавлять дополнительные замечания о сущности, которые не были отражены в определении (*Definition*), например, описать бизнес – правило или соглашение по организации диаграммы;
- вкладка **Note2** позволяет задокументировать некоторые возможные запросы, которые предполагается использовать по отношению к сущности в базе данных;
- вкладка **Note3** позволяет в произвольной форме вводить примеры данных для сущности;
- вкладка **UDP** позволяет пользователю определить свойства сущности и объем хранимых данных.

Вкладка **Icon** позволяет каждой сущности поставить в соответствие изображение, которое будет отображаться в режиме просмотра модели на уровне иконок.

Вкладка **History** позволяет просмотреть историю всех изменений, связанных с сущностью и добавить комментарий к изменению в окне **Comment**.



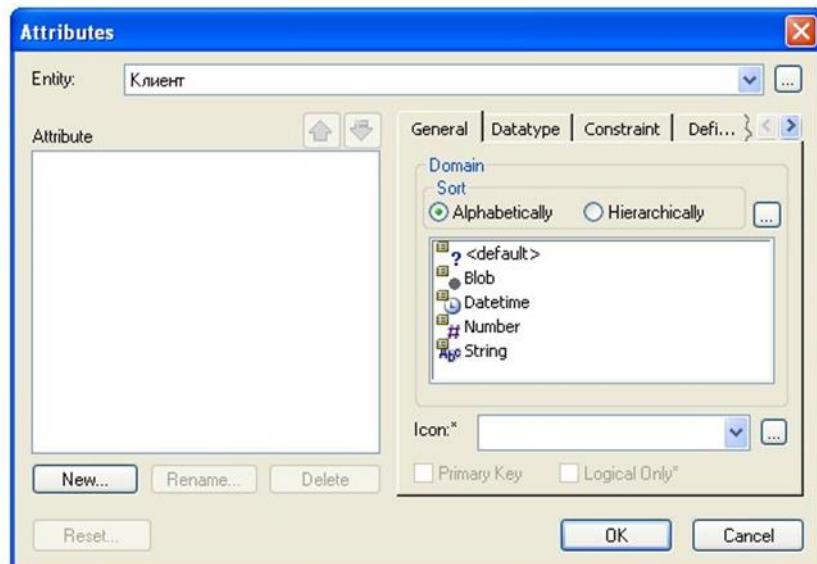
Размещение в поле проектирования сущности с именем **Клиент**

По активизации кнопки **OK** на поле проектирования отобразится сущность с заданным именем и определенными свойствами.

При разработке ER-модели можно установить *параметры шрифтов и цветовое оформление сущностей* для облегчения обзора и понимания модели. При этом можно изменять установки параметров, назначенные по умолчанию, при добавлении нового объекта на поле диаграммы, как для всех объектов диаграммы, так и для групп выделенных объектов или отдельных сущностей.

Параметры шрифта для названия сущности и цвет для заполнения поля блока редактируются с помощью пункта **Object Font & Color** контекстного меню (всплывающего меню по щелчку правой кнопкой мыши на поле выделенной сущности). Аналогичным образом изменяются параметры и для групп выделенных объектов диаграммы.

Определение атрибутов сущностей и их характеристик осуществляется в диалоговом окне **Attributes**, которое открывается с помощью пункта **Attributes...** контекстного меню и позволяет ввести данные об атрибутах выбранной сущности.



Диалоговое окно **Attributes**

При активизации кнопки **New...** диалогового окна **Attributes** открывается диалог **New Attribute**, позволяющий добавить новый атрибут для выделенной сущности.



Диалоговое окно New Attribute

В диалоговом окне **New Attribute** указывается имя нового атрибута (поле **Attribute Name**), имя, соответствующее ему в физической модели колонки (поле **Column Name**) и домен (тип колонки на уровне физической модели).

Имена атрибутов можно задавать с помощью шрифтов типа "Кириллица" или "Латиница". При этом следует учитывать возможности СУБД, которые будут использоваться для сопровождения базы данных.

Например, при использовании СУБД типа Access можно использовать имена сущностей и атрибутов для физической модели данных на русском языке.

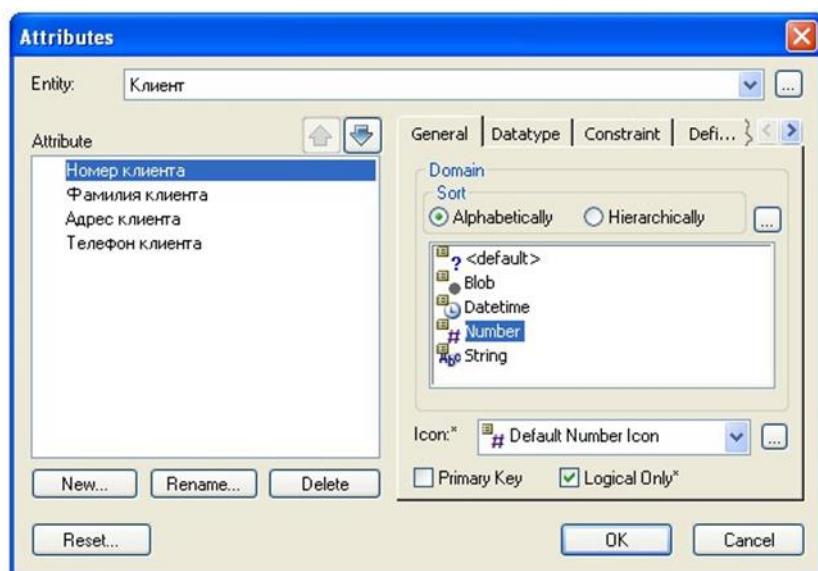


Определение нового атрибута для сущности **Клиент**

Атрибуты должны именоваться в единственном числе и иметь четкое смысловое значение, что позволяет частично решить проблему нормализации данных на этапе определения атрибутов. Например, создание для сущности **Клиент** атрибута **Телефоны клиента** противоречит требованиям нормализации, т. к. атрибут должен быть атомарным, т.е. не содержать множественных значений.

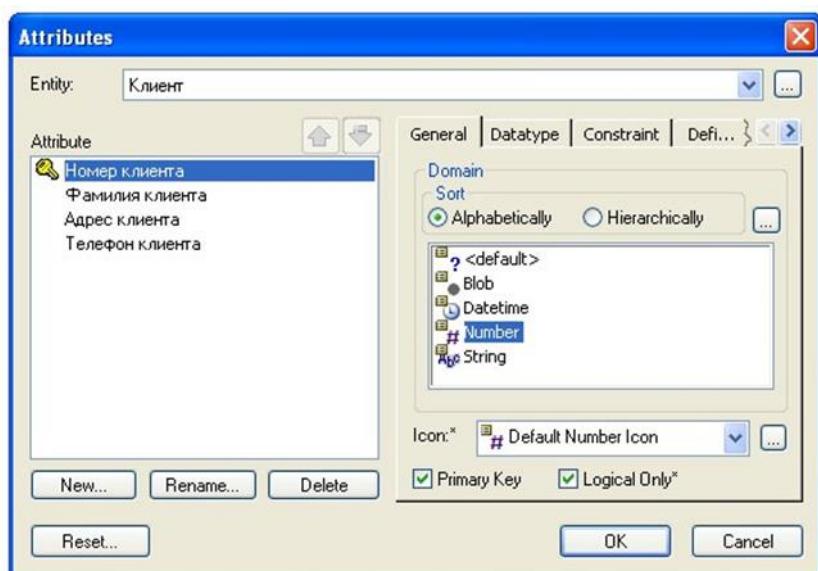
Согласно синтаксису нотации, IDEF1X имя атрибута должно быть уникально в рамках модели, поэтому новый атрибут, в случае совпадения его имени с уже существующим в рамках модели, должен быть переименован.

При активизации кнопки **OK** новый атрибут добавляется в список атрибутов выделенной сущности, отражающийся в поле **Attribute** диалогового окна **Attributes**.



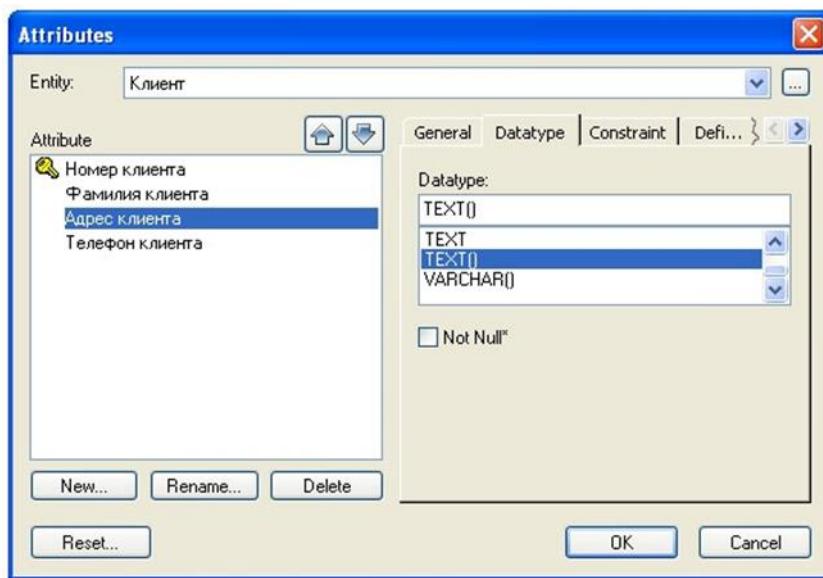
Атрибуты сущности **Клиент**

Для определения первичного ключа выделенной сущности необходимо перейти на вкладку **General** и сделать пометку в окне выбора **Primary Key**.



Определение первичного ключа в окне выбора **Primary Key**

Вкладка **Datatype** позволяет с помощью выпадающего списка на странице уточнить тип данных для выделенного атрибута.



Уточнение типа данных атрибута **Адрес клиента**

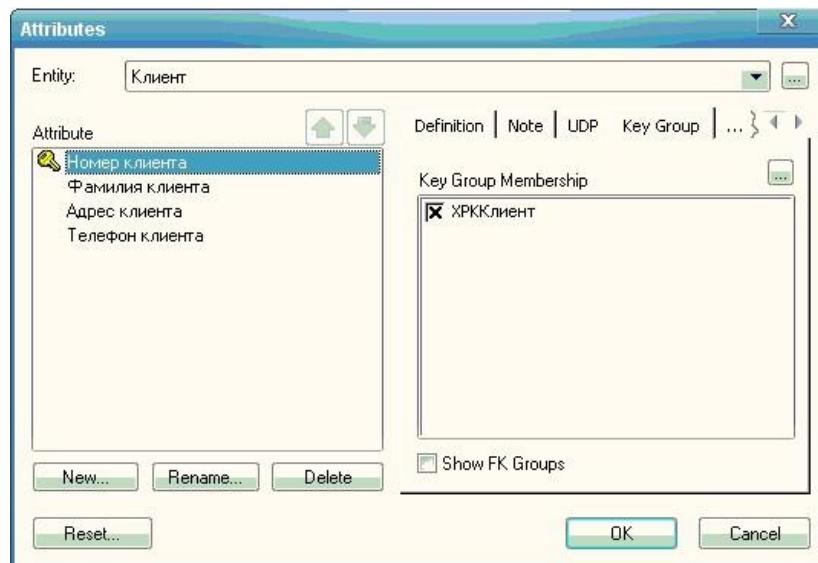
Вкладка **Constraint** позволяет задать ограничения для выделенного атрибута.

Вкладка **Definition** позволяет записать определения отдельных атрибутов.

Вкладка **Note** позволяет добавлять замечания об одном или нескольких атрибутах сущности, которые не вошли в определения.

Вкладка **UDP** предназначена для задания значений свойств, определяемых пользователем.

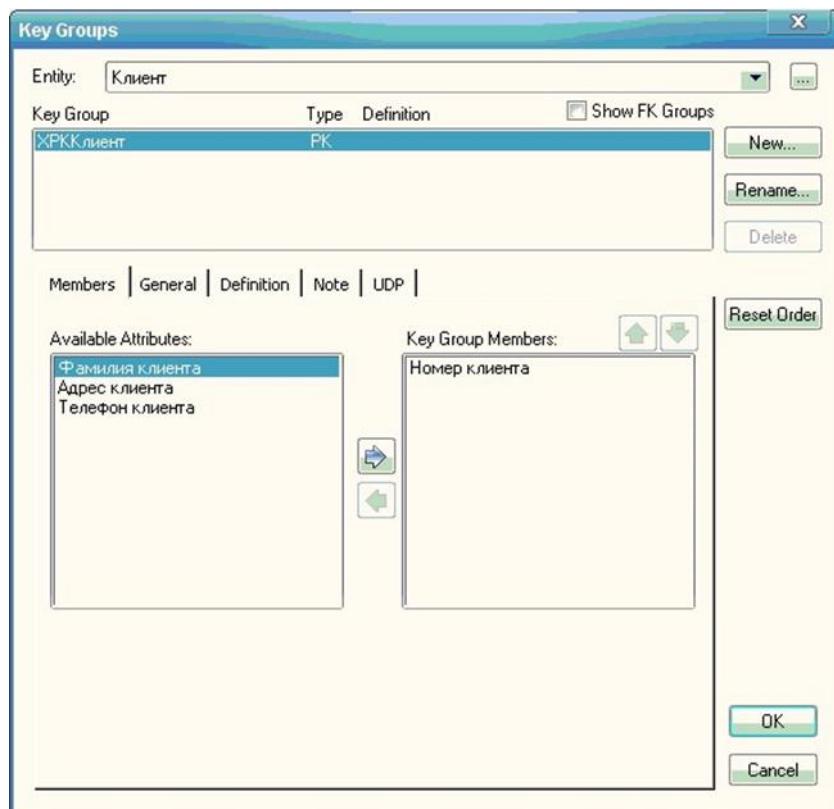
Вкладка **Key Group** предназначена для отображения атрибутов, входящих в группу ключевых атрибутов сущности. В частности, на вкладке показано, что первичным ключом сущности **Клиент** является атрибут **Номер клиента**.



Вкладка **Key Group** диалога **Attributes**

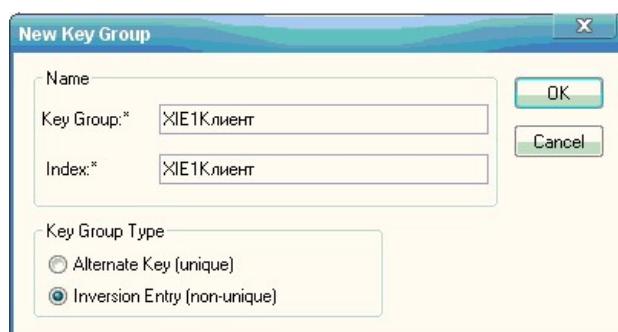
Сущность может иметь несколько возможных (*потенциальных*) **ключей**. Один потенциальный ключ объявляется первичным, а остальные могут быть объявлены **альтернативными ключами (Alternate Key)**. ERwin по умолчанию на физическом уровне генерирует уникальные индексы по первичному и альтернативным ключам. ERwin позволяет также строить *неуникальные индексы* по атрибутам, называемым **инверсными входами (Inversion Entries)**. Неуникальный индекс обеспечивает быстрый доступ не к одной строке таблицы, а к нескольким, имеющим одинаковое значение инверсного входа. Это ускоряет доступ к данным.

Альтернативные ключи и инверсные входы создаются в диалоге **Key Groups**, который появляется после выбора соответствующего пункта контекстного меню.



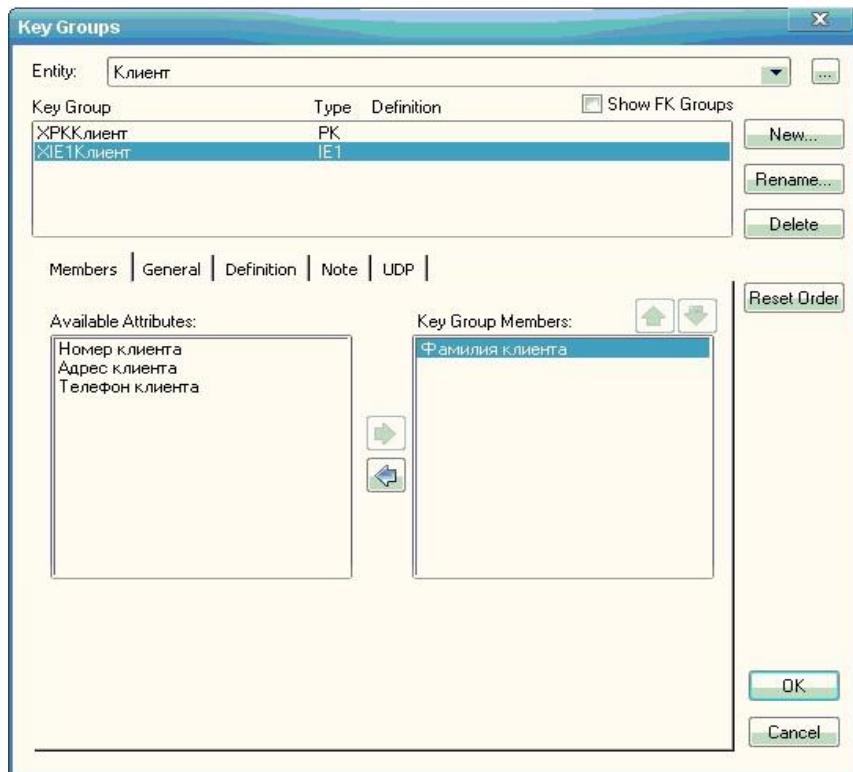
Диалог **Key Groups**

Видно, что имеется первичный ключ (**PK**), включающий атрибут **Номер клиента**. Создадим инверсный вход по атрибуту **Фамилия клиента**. Для этого выделяем атрибут **Фамилия клиента** и нажимаем на кнопку **New**. Появляется диалог **New Key Group**.



Диалог **New Key Group**

В этом диалоге выбираем **Inversion Entry**. Имя ключевой группы (**Key Group**) и индекса (**Index**) присваиваются автоматически. Щелкаем по кнопке **OK** и возвращаемся в диалог **Key Groups**.

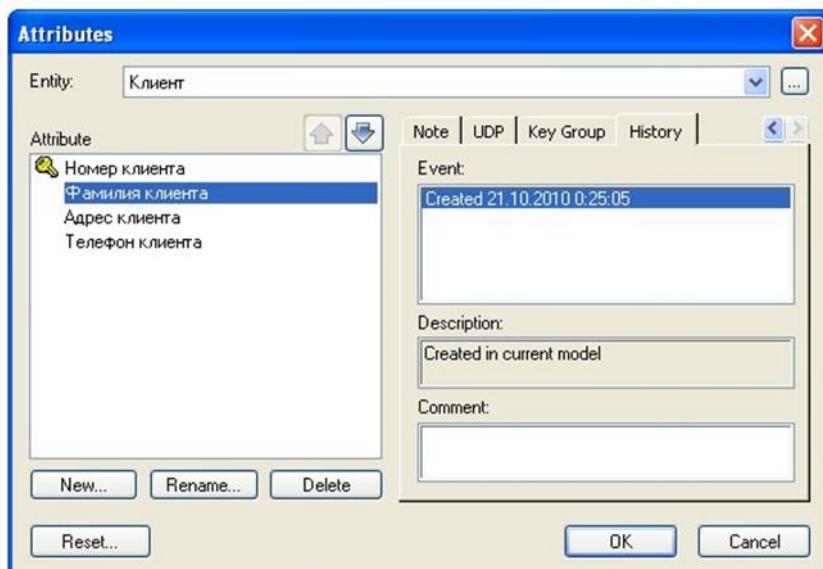


Диалог **Key Groups** при построении инверсного входа

В верхнем окне **Key Group** вкладки выбираем ключевую группу **XIE1Клиент**, являющуюся инверсным входом (**IE**). В окне **Available Attributes** выбираем атрибут **Фамилия клиента** и с помощью кнопки включаем его в состав ключевой группы.

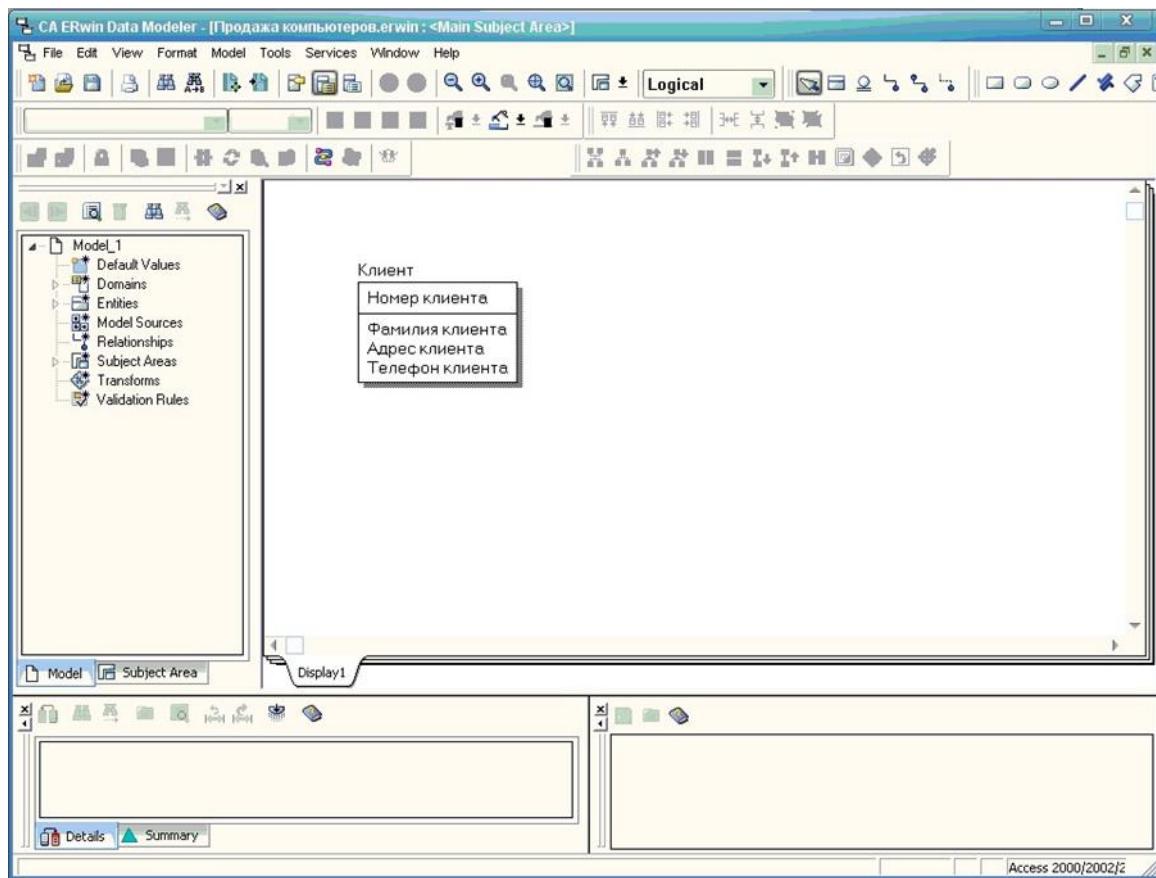
Альтернативные ключи создаются аналогично.

Вкладка **History** диалога **Attributes** отображает историю создания и изменения свойств атрибута и позволяет добавить комментарии к изменению в окне **Comment**.



История создания и изменения свойств атрибута **Фамилия клиента**

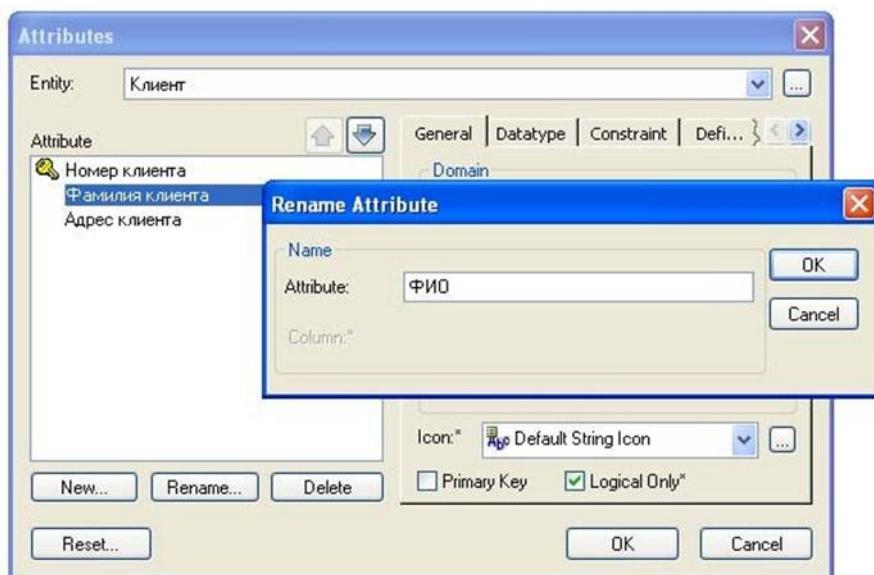
При нажатии на кнопку **OK** в поле проектирования модели отобразится сущность с атрибутами, определенными пользователем.



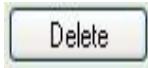
Сущность Клиент с заданными атрибутами

Диалоговое окно **Attributes** позволяет пользователю редактировать имена атрибутов выделенной сущности и корректировать список атрибутов путем выполнения операции удаления ошибочно определенного атрибута.

Редактирование имени атрибута осуществляется в диалоге **Rename Attribute**, который открывается при активизации кнопки **Rename...** диалогового окна **Attributes**.



Редактирование имени атрибута Фамилия клиента



При активизации кнопки **Delete** диалогового окна **Attributes** пользователю предоставляется возможность удаления выделенного атрибута из списка атрибутов. При этом необходимо внимательно относиться к данному действию, т.к. **ERwin** не требует подтверждения удаления атрибута.

Для описания свойств сущности часто приходится создавать *производные атрибуты*, т.е. атрибуты, значение которых можно вычислить из других атрибутов. Примером производного атрибута может служить атрибут **Сумма заказа**, значение которого может быть вычислено из атрибутов **Количество заказанного товара** и **Цена за единицу товара**.

Многие производные атрибуты часто приводят к конфликтам.

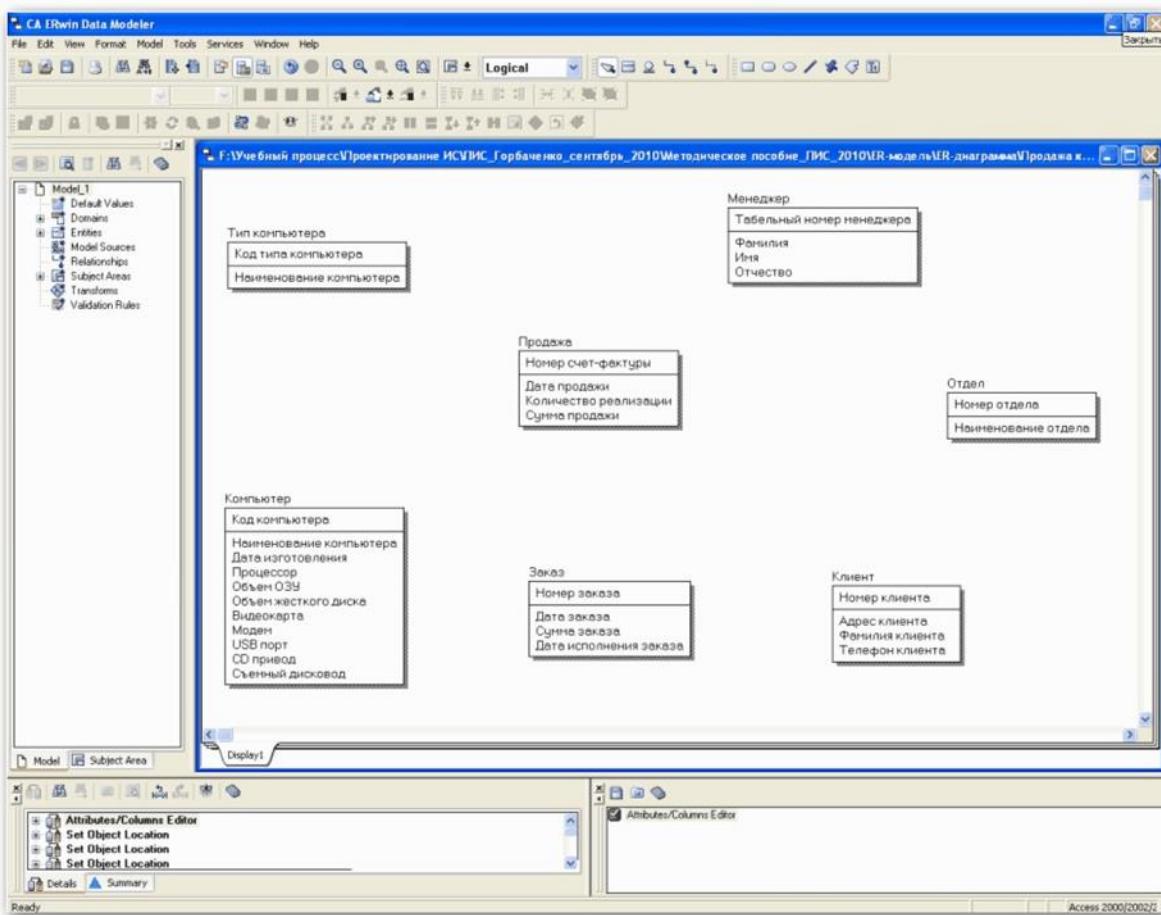
Например, значение производного атрибута **Возраст сотрудника** может быть вычислено из атрибута **Дата рождения сотрудника**. Возникновение конфликтной ситуации здесь возможно в случае несвоевременного обновления значения атрибута **Возраст сотрудника**, что приводит к противоречию значению атрибута **Дата рождения сотрудника**.

Производные атрибуты – ошибка нормализации. Нарушение синтаксиса нормализации допустимо в случаях необходимости повышения производительности системы.

Например, хранение **Итоговой суммы заказа** позволяет повысить производительности системы при формировании документов на оплату заказа за счет непосредственного обращения к соответствующему атрибуту и исключения проведения предварительных вычислений, которые могут быть достаточно сложными, особенно при использовании специальных методик расчета с введением коэффициентов различного назначения.

ERwin позволяет *переносить атрибуты* внутри и между сущностями. Для этого необходимо щелкнуть правой кнопкой мыши по атрибуту. При этом указатель приобретает вид *кисти руки*, после чего можно перетащить выделенный атрибут на новое местоположение внутри сущности или в поле другой сущности диаграммы.

Для описания объектов предметной области по реализации настольных компьютеров по заказам клиентов выделены следующие сущности: **Тип компьютера**, **Компьютер**, **Клиент**, **Заказ**, **Продажа**, **Менеджер**, **Отдел**.



Сущности, выделенные для описания объектов предметной области по реализации настольных компьютеров по заказам клиентов

3.2.4. Создание связей

Логическим соотношением между сущностями является *связь*. Каждому виду связи соответствует определенная кнопка, расположенная на панели инструментов. Имя связи выражает некоторое ограничение или правило и облегчает чтение диаграммы. Каждая связь должна именоваться глаголом или глагольной фразой.

Например, связь между сущностями **Компьютер**, **Продажа** и **Менеджер** показывает, какой компьютер продан и какой менеджер оформил сделку продажи компьютера:

- каждый **Компьютер** <продается> **Продажа**;
- каждая **Продажа** <оформляется> **Менеджер**.



В нотации 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 и показывается на диаграмме сплошной линией с жирной точкой на дочернем конце связи. При установлении идентифицирующей связи ERwin автоматически преобразует дочернюю сущность в зависимую сущность. Зависимая сущность на диаграмме изображается прямоугольником со скругленными углами, например, сущность **Продажа**. Экземпляр зависимой сущности определяется только через отношение к родительской сущности.

Например, информация о продаже не может быть внесена и не имеет смысла без информации о проданном компьютере. При установлении *идентифицирующей связи* атрибуты первичного ключа родительской сущности автоматически переносятся в состав первичного ключа дочерней сущности. Операция дополнения атрибутов дочерней сущности при создании связи называется *миграцией атрибутов*. В дочерней сущности новые (мигрированные) атрибуты помечаются как внешний ключ (**FK**). В случае идентифицирующей связи при генерации схемы базы данных атрибутам внешнего ключа присваивается признак **NOT NULL**, что означает невозможность внесения записи в таблицу, соответствующей дочерней сущности, без идентификационной информации из таблицы, соответствующей родительской сущности. *Например*, невозможность внесения записи в таблицу продаж без информации об идентификационном номере проданного компьютера, определенного в таблице описания имеющихся в наличии компьютеров.

Неидентифицирующая связь показывается на диаграмме пунктирной линией с жирной точкой и служит для установления связи между независимыми сущностями. При установлении неидентифицирующей связи дочерняя сущность остается независимой, а атрибуты первичного ключа мигрируют в состав неключевых атрибутов родительской сущности.



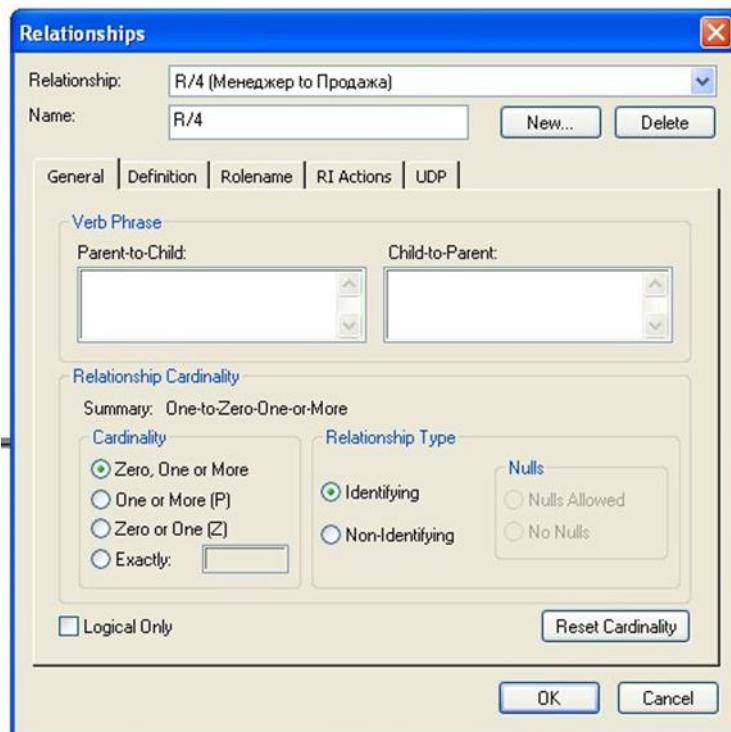
Пример неидентифицирующей связи между независимыми сущностями
Менеджер и **Отдел**

Для создания новой связи необходимо:

- установить курсор на кнопке, расположенной на палитре инструментов и соответствующей требуемому виду связи, и нажать левую кнопку мыши;
- щелкнуть левой кнопкой мыши сначала по родительской, а затем по дочерней сущности.

Изменить форму линии связи и ее местоположение между связанными сущностями можно путем захвата мышью выделенной линии связи и переноса ее с места на место, пока линия не примет необходимые местоположение и форму.

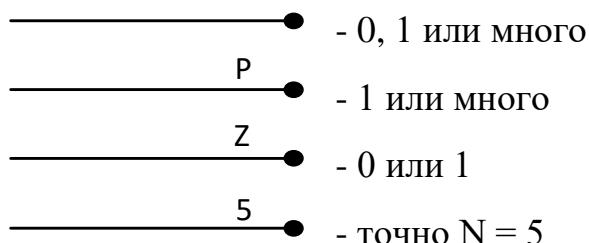
Редактирование свойств связи осуществляется в диалоговом окне **Relationship**, которое открывается через пункт **Relationship Properties** контекстного меню, активизируемого посредством нажатия правой кнопки мыши на выделенной связи.



Диалоговое окно **Relationship**

Вкладка **General** диалогового окна **Relationship** позволяет задать мощность, имя и тип связи.

Мощность связи (Cardinality) служит для обозначения отношения числа экземпляров родительской сущности к числу экземпляров дочерней сущности. Различают 4 типа мощности связи:

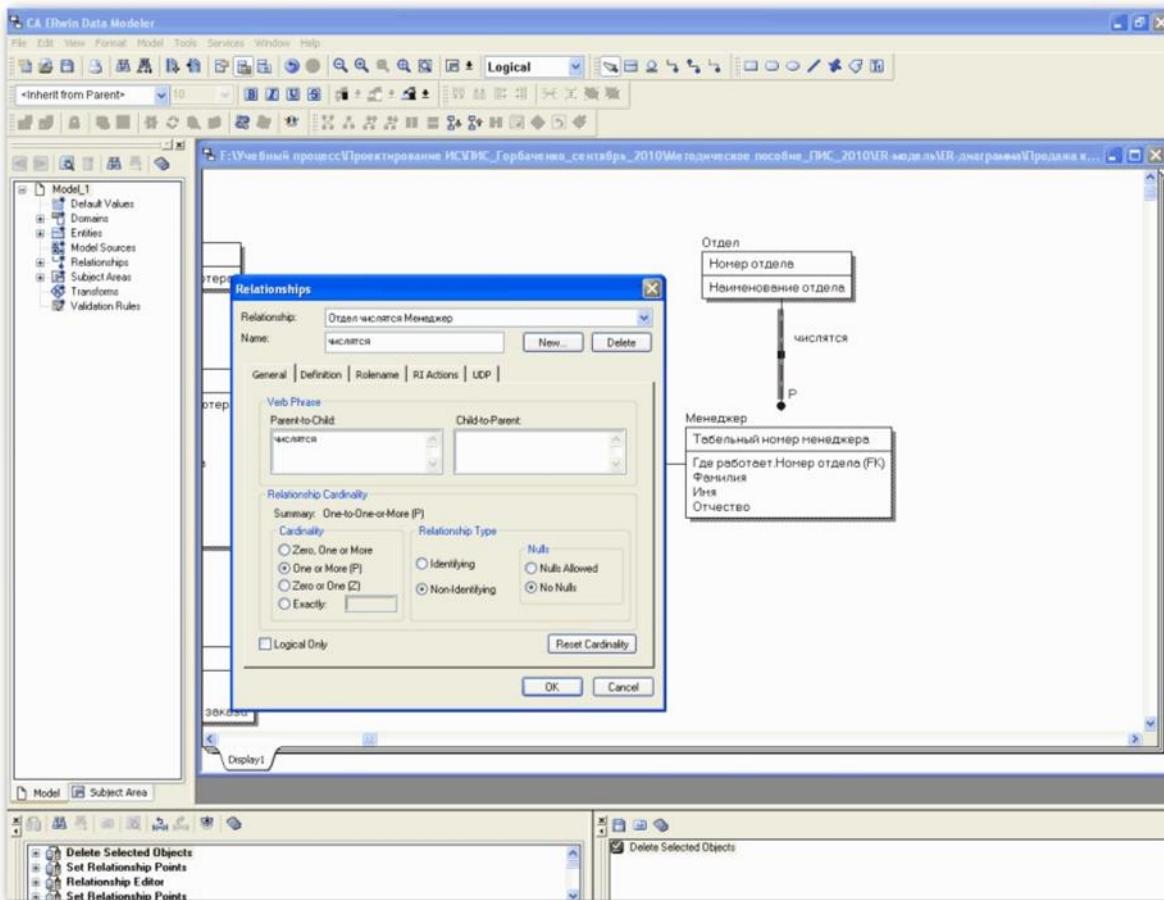


Обозначение мощности связей

- **общий случай** – не помечается каким-либо символом и соответствует ситуации, когда одному экземпляру родительской сущности соответствуют 0, 1 или много экземпляров дочерней сущности;
- **символом *P* помечается случай**, когда одному экземпляру родительской сущности соответствуют 1 или много экземпляров дочерней сущности, т.е. исключено нулевое значение;
- **символом *Z* помечается случай**, когда одному экземпляру родительской сущности соответствуют 0 или 1 экземпляр дочерней сущности, т.е. исключены множественные значения;
- **цифрой помечается случай точного соответствия**, когда одному экземпляру родительской сущности соответствует заранее заданное число экземпляров дочерней сущности.

По умолчанию символ, обозначающий мощность связи, не показывается на диаграмме. Для отображения мощности связи следует включить опцию **Cardinality** пункта **Relationship Display** контекстного меню, которое появляется по щелчку правой кнопкой мыши по любому месту диаграммы, не занятому объектами модели.

Имя связи (*Verb Phrase*) – фраза, характеризующая отношение между родительской и дочерней сущностями. Для связи "один ко многим" (идентифицирующей или неидентифицирующей) достаточно указать имя, характеризующее отношение от родительской к дочерней сущности (*Parent-to-Child*), а для связи "многие ко многим" необходимо указывать имя связи как от родительской к дочерней сущности (*Parent-to-Child*), так и от дочерней к родительской сущности (*Child-to-Parent*).

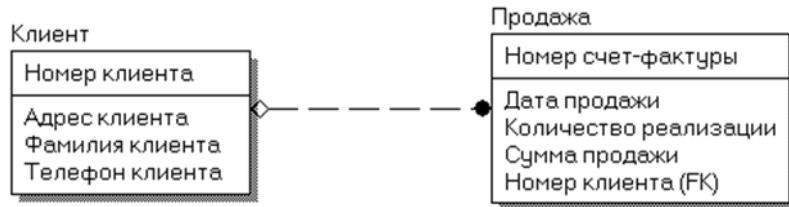


Пример определения мощности и имени связи между сущностями **Отдел** и **Менеджер**

Тип связи (*Relationships Type*) позволяет обозначить идентифицирующую (*Identifying*) или неидентифицирующую (*Non-Identifying*) связь. Для неидентифицирующей связи необходимо указать обязательность связи (*Nulls Allowed* или *No Nulls*). В случае обязательной связи (*No Nulls*), 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. Необязательная неидентифицирующая связь помечается прозрачным ромбом со стороны родительской сущности.



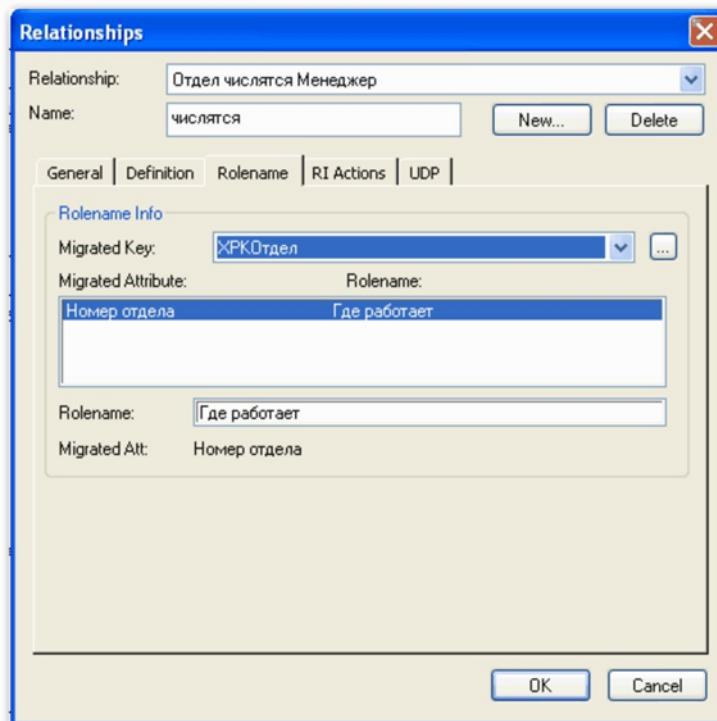
Графическое представление необязательной неидентифицирующей связи
Например, при оформлении сделки по продаже компьютера информация о покупателе (клиенте) не всегда участвует в оформлении расходных документов.



Пример реализации необязательной неидентифицирующей связи

Вкладка **Definition** позволяет дать более полное определение связи для того, чтобы в дальнейшем иметь возможность на него ссылаться.

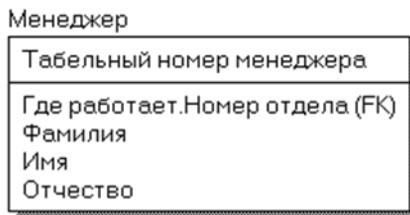
Вкладка **Rolename** открывает окно диалога **Relationships**, которое позволяет задать имя роли атрибута внешнего ключа.



Окно диалога **Relationships** вкладки **Rolename**

Имя роли (Rolename, функциональное имя) – это синоним атрибута внешнего ключа, который показывает, какую роль играет атрибут в дочерней сущности.

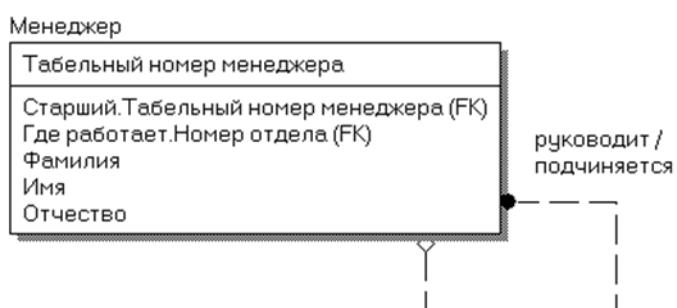
Например, в сущности **Менеджер** внешний ключ **Номер отдела** имеет функциональное имя "*Где работает*", которое показывает, какую роль играет этот атрибут в данной сущности. По умолчанию в списке атрибутов показывается только имя роли. Для отображения полного имени атрибута (как функционального имени, так и имени роли необходимо включить опцию **Rolename/Attribute** пункта **Entity Display** контекстного меню, которое появляется по щелчку правой кнопки мыши по любому месту диаграммы, не занятому объектами модели. В результате отобразится *Полное имя* атрибута, включающее функциональное имя и базовое имя, разделенные между собой точкой.



Пример *Полного имени* внешнего атрибута **Номер отдела** сущности **Менеджер**

Обязательным является применение имен ролей в том случае, когда два или более атрибута одной сущности определены по одной и той же области, т. е. они имеют одинаковую область значений, но разный смысл. Другим примером обязательности присвоения имен ролей являются *рекурсивные связи* (иногда их называют "рыболовным крючком" – *fish hook*), когда одна и та же сущность является и родительской, и дочерней одновременно. При задании рекурсивной связи атрибут миграирует в качестве внешнего ключа в состав неключевых атрибутов той же сущности. Атрибут не может появиться дважды в одной сущности под одним именем, поэтому обязательно должен получить имя роли. Например, сущность **Менеджер** содержит атрибут первичного ключа **Табельный номер**. Информация о старшем менеджере (руководителе) содержится в той же сущности, поскольку старший менеджер работает в этой же организации. Чтобы сослаться на старшего менеджера, необходимо создать рекурсивную связь **руководит/подчиняется** и присвоить имени роли значение **старший** (рис. 6.33). Рекурсивная связь может быть только необязательной неидентифицирующей. В противном случае внешний ключ должен был бы войти в состав первичного ключа и получить при генерации схемы признак **NOT NULL**, что делает невозможным построение иерархии – у дерева подчиненности должен быть корень – менеджер, который никому не подчиняется в рамках данной организации.

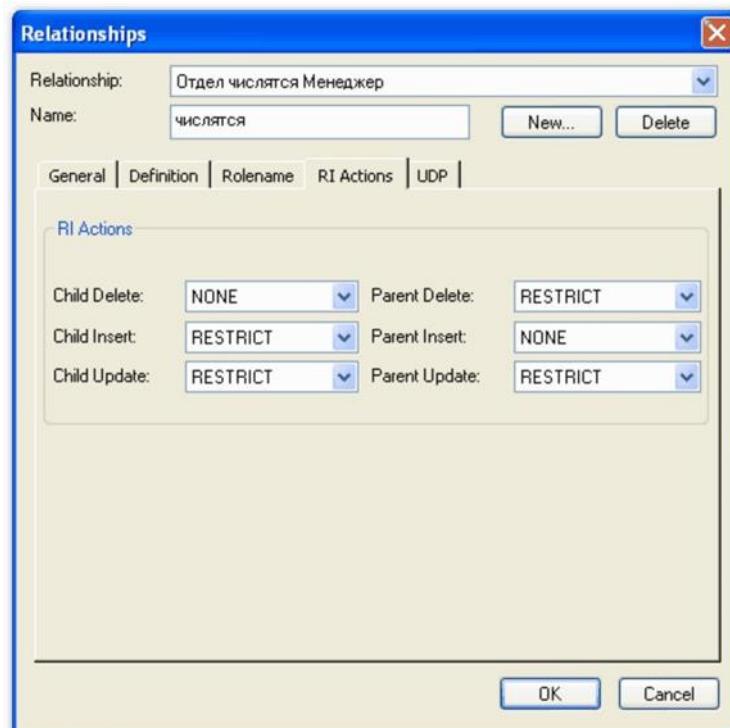
Связь **руководит/подчиняется** позволяет хранить древовидную иерархию подчиненности сотрудников. Такой вид рекурсивной связи называется *иерархической рекурсией* (*hierarchical recursion*) и задает связь, когда руководитель (экземпляр родительской сущности) может иметь множество подчиненных (экземпляров дочерней сущности), но подчиненный имеет только одного руководителя.



Пример рекурсивной связи

Другим видом рекурсии является *сетевая рекурсия (network recursion)*, когда руководитель может иметь множество подчиненных и, наоборот, подчиненный может иметь множество руководителей. Сетевая рекурсия задает паутину отношений между экземплярами родительской и дочерней сущностей. Это случай, когда сущность находится сама с собой в связи "многие-ко- многим". Для разрешения связи "многие-ко-многим" необходимо создать дополнительную сущность (подробнее связь "многие-ко-многим" рассматривается ниже).

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** контекстного меню **Relationships**. Заданные на вкладке **RI Actions** опции логической модели используются при генерации схемы базы данных для определения правил декларативной ссылочной целостности, которые предписываются для каждой связи, и триггеров, обеспечивающих ссылочную целостность.



Окно диалога **Relationships** вкладки **RI Actions**

На рисунке показан пример установленных по умолчанию правил ссылочной целостности для неидентифицирующей связи между сущностями **Отдел** и **Менеджер**. На удаление экземпляра родительской сущности (**Parent Delete**) устанавливается ограничение **RESTRICT**. Это означает, что экземпляр родительской сущности нельзя удалить, если с ней связан экземпляр дочерней. С экземпляром родительской сущности может быть не связан ни один экземпляр дочерней сущности. Поэтому при вставке

экземпляра родительской сущности (**Parent Insert**) не проводится никакой проверки (ограничение **NONE**). При изменении ключевых атрибутов родительской сущности (**Parent Update**) нарушится связь с дочерними сущностями, так как внешний ключ дочерних сущностей не равен ключу родительской сущности. Поэтому по умолчанию такое изменение запрещено (ограничение **RESTRICT**). Отметим, что в данном случае можно применить ограничение **CASCADE**, приводящее к изменению внешних ключей дочерних сущностей при изменении ключа родительской сущности. При удалении экземпляра дочерней сущности (**Child Delete**) ссылочная целостность не нарушается, поэтому по умолчанию применено ограничение **NONE**. Если при вставке экземпляра дочерней сущности (**Child Insert**) ее внешний ключ будет отличаться от ключа родительской сущности, то нарушится ссылочная целостность. Такая ситуация проверяется ограничением **RESTRICT**. Аналогично, ограничение **RESTRICT** не допускает присвоения внешнему ключу дочерней сущности значения, отличного от значения родительского ключа (**Child Update**).

ERwin автоматически присваивает каждой связи значение ссылочной целостности, устанавливаемой по умолчанию, прежде чем добавить ее в диаграмму. Режимы **RI**, присваиваемые ERwin по умолчанию, могут быть изменены на вкладке **RI Defaults** редактора **Model Properties** (меню **Model/Model Properties**). Правила ссылочной целостности можно изменить на вкладке **RI Actions** диалогового окна **Relationships**.

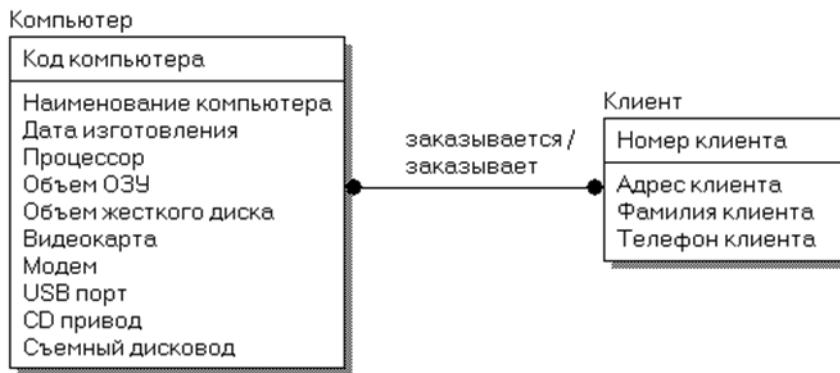
Правила ссылочной целостности реализуются специальными программами – *триггерами*, хранящимися в базе данных и выполняемыми всякий раз при выполнении команд вставки, замены или удаления (**INSERT**, **UPDATE** или **DELETE**). На логическом уровне ERwin создает шаблоны триггеров, построенные с использованием специальных макрокоманд. Шаблоны триггеров и расширенный код, зависящий от выбранной СУБД, можно увидеть на вкладке **Relationships Templates** контекстного меню **Relationships**.

Связь "многие-ко-многим" может быть создана только на уровне логической модели. Такая связь обозначается сплошной линией с двумя точками на концах. Установление связи "многие-ко-многим" осуществляется при активизации соответствующей кнопки на панели инструментов посредством щелчка левой кнопки мыши сначала по одной, а затем по другой сущности.

С целью облегчения чтения диаграммы связь "многие-ко-многим" должна иметь двунаправленное имя: от родительской к дочерней сущности (*Parent-to-Child*) и от дочерней к родительской сущности (*Child-to-Parent*).

Примером связи "многие-ко-многим" может служить связь между сущностями **Компьютер** и **Клиент**, т.к. один тот же вид компьютера может быть заказан многими клиентами (покупателями), а один и тот же клиент организации может заказать разные виды компьютеров:

- **Компьютер** <заказывается> **Клиент**;
- **Клиент** <заказывает> **Компьютер**.



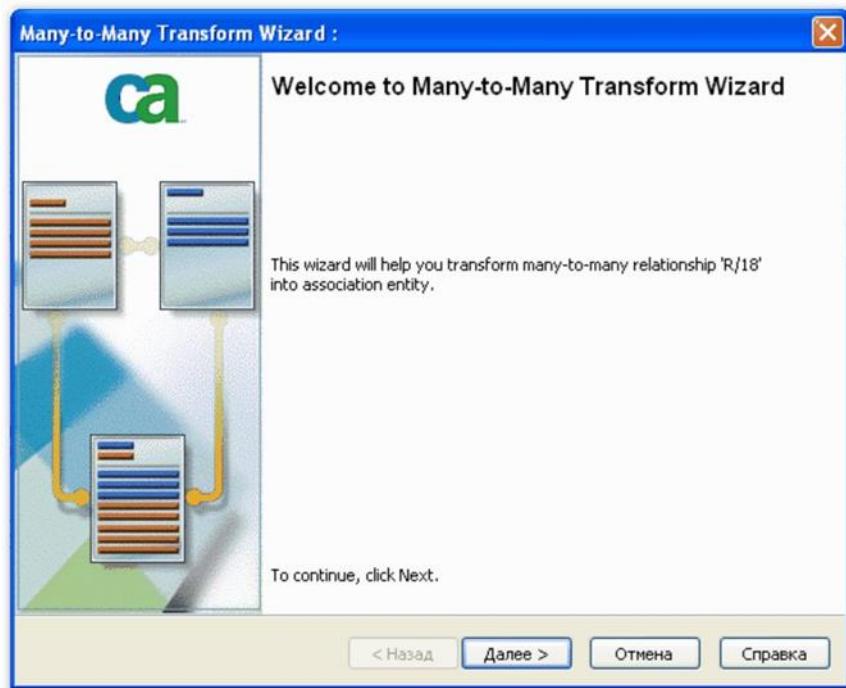
Пример реализации связи "многие-ко-многим"

Нотация IDEF1X требует, чтобы на физическом уровне связь "многие-ко-многим" была преобразована, так как СУБД не поддерживают связь "многие-ко-многим". По умолчанию при переходе к физическому уровню ERwin автоматически не преобразует связь "многие-ко-многим". В этом случае на физическом уровне диаграмма выглядит так же, как и на логическом, однако при генерации схемы базы данных системы такая связь игнорируется. Поэтому уже на логическом уровне предпочтительно таких связей избегать. Кроме того, этого требует и синтаксис нормализации до третьей нормальной формы.

ERwin позволяет реализовать принудительное преобразование связи "многие-ко-многим", которое включает создание новой (связывающей) таблицы и двух новых связей "один-ко-многим" от старых таблиц к новой таблице. При этом имя новой таблице может присваиваться как автоматически, так и определяться пользователем. В случае автоматического присвоения имени связывающей таблице назначается имя, включающее имена обеих сущностей.

Например, в случае автоматического разрыва связи "многие-ко-многим" между сущностями **Компьютер** и **Клиент** связывающая таблица получит имя **КомпьютерКлиент**.

Для осуществления принудительного преобразования связи "многие-ко-многим" необходимо выбрать пункт контекстного меню **Create Association Entity**. В результате откроется диалог **Many-To-Many Transform Wizard**.



Окно диалога **Many-To-Many Transform Wizard**

Диалог **Many-To-Many Transform Wizard** предлагает выполнить четыре шага для преобразования связи. Для перехода к следующему шагу необходимо щелкнуть по кнопке **Далее**. На втором и третьем шаге следует задать имя вновь создаваемой таблицы и имя преобразования. По окончании выполнения действий диалога **Many-To-Many Transform Wizard** на диаграмме будет добавлена новая таблица с заданным пользователем именем, а связь "многие-ко-многим" заменится идентифицирующими связями "один-ко-многим" от старых таблиц к новой таблице. При этом новые связи автоматически получат соответствующие имена от связи "многие-ко-многим".

Пример результата принудительного преобразования связи "многие-ко-многим" между сущностями **Компьютер** и **Клиент**, реализованного с помощью диалога **Many-To-Many Transform Wizard**, приведен на рисунке.



Пример результата принудительного преобразования связи "многие ко многим"

Принудительного решения проблемы связи "многие-ко-многим" не всегда оказывается достаточно. Часто для описания сущности согласно бизнес-логике требуется добавление дополнительных атрибутов, позволяющих идентифицировать новую сущность.

Например, для описания сущности **Заказ** необходимо добавить такие атрибуты, как **Дата заказа**, **Дата исполнения заказа** и **Сумма заказа**.

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. Чтобы представить информацию, общую для всех типов экземпляров сущности, из их общих свойств можно сформировать обобщенную сущность (родовой предок). При этом специфическая для каждого типа экземпляра сущности информация будет располагаться в категориальных сущностях (потомках).

Например, в организации работают менеджеры и консультанты, которые имеют сходную по смыслу связь "**работает в**" с сущностью **Отдел**. Чтобы представить информацию, общую для всех типов служащих, из их общих свойств можно сформировать обобщенную сущность **Сотрудник**. При этом категориальными сущностями будут являться **Менеджер** и **Консультант** (рис. 6.38).



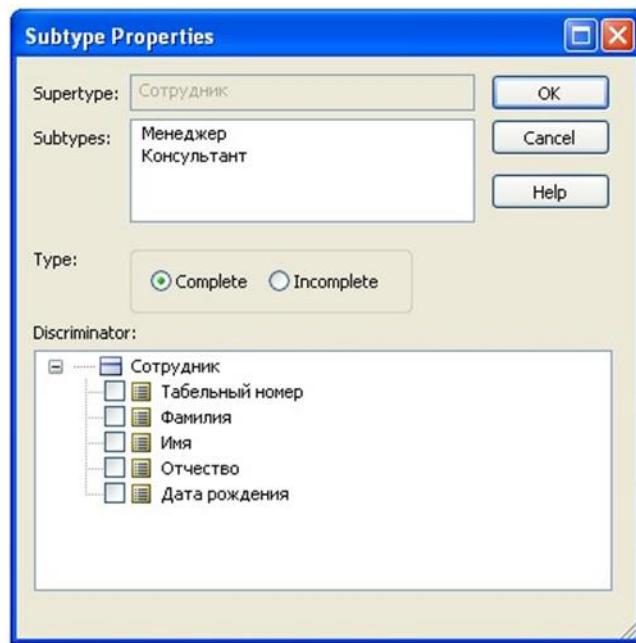
Пример иерархии наследования (категориальной связи)

Для создания категориальной связи следует:

- установить курсор на кнопке , расположенной на палитре инструментов и нажать левую кнопку мыши;
- щелкнуть сначала по родовому предку, а затем – по потомку;
- установить вторую связь в иерархии категории, выбрав кнопку и щелкнув сначала по символу категории на диаграмме, затем – по второму потомку.

Редактирование категорий осуществляется в диалоговом окне **Subtype Properties**, которое открывается при выборе пункта **Subtype Properties...** контекстного меню, отображаемого по щелчку правой кнопкой мыши по символу категории (рис. 6.39).

Поля диалогового окна позволяют указать дискриминатор – атрибут категории (список **Discriminator**) и тип категории – полная/неполная (радиокнопки **Complete/Incomplete**).



Диалоговое окно Subtype Properties

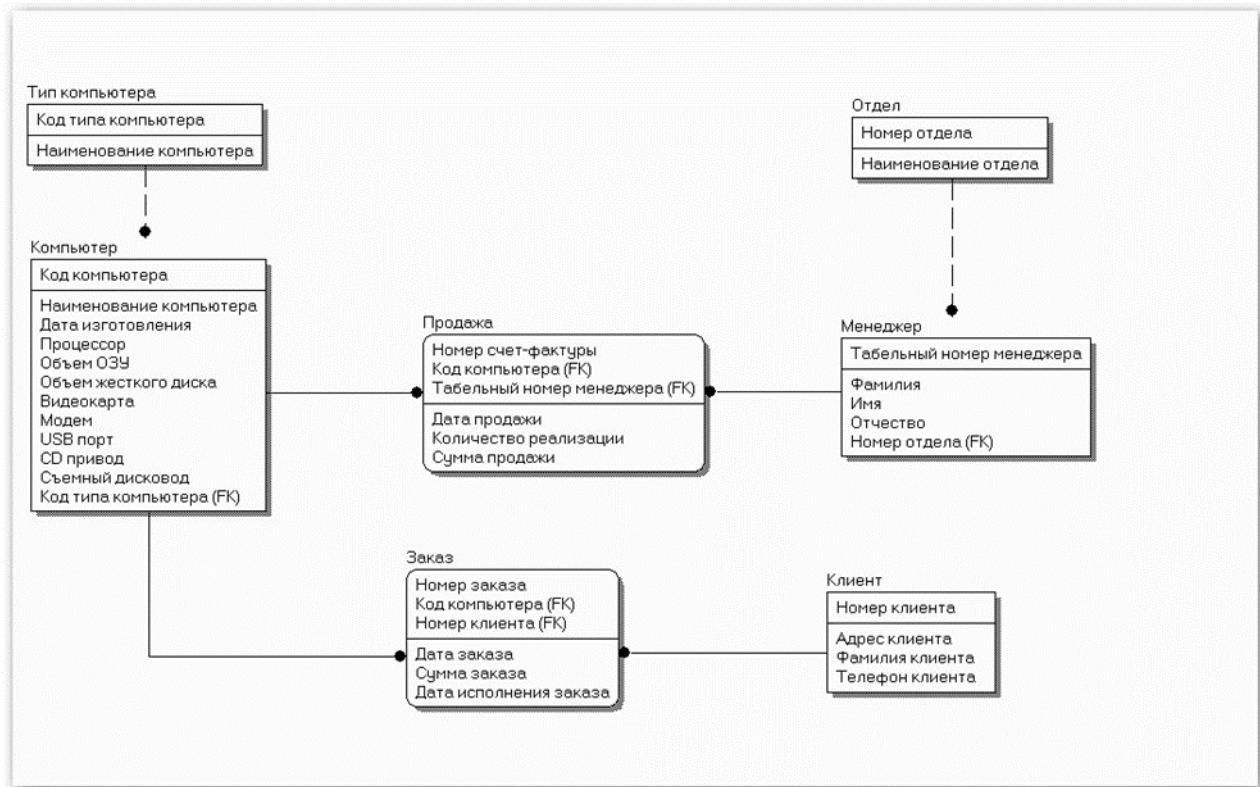
Дискриминатор категории – это атрибут родового предка, который показывает, как отличить одну категориальную сущность от другой.

Иерархии категорий делятся на два типа – полные и неполные. В полной категории одному экземпляру родового предка обязательно соответствует экземпляр в каком-либо потомке. Например, сотрудник обязательно является либо менеджером, либо консультантом. Если категория еще не выстроена полностью и в родовом предке могут существовать экземпляры, которые не имеют соответствующих экземпляров в потомках, то такая категория будет неполной. Например, сотрудник может быть не только менеджером или консультантом, но и совместителем. В случае неполной категории сущность **Совместитель** еще не внесена в иерархию наследования (рис. 6.38). На рис. 6.40 представлен пример полной категории. Возможна также комбинация полной и неполной категорий.



Пример полной категории

Результат разработки логической модели данных системы "Реализация средств вычислительной техники", предназначеннной для учета продаж настольных компьютеров по заказам клиентов приведен на рис. 6.40.



Логическая модель данных системы "Реализация средств вычислительной техники"

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. При этом физические модели данных могут соответствовать СУБД разных производителей, например, Oracle, MS SQL Server, MySQL, SYBASE, Informix, MS Access, Progress и т.д. Возможность синхронизации логического уровня модели с несколькими моделями, имеющими разный физический уровень, позволяет повысить эффективность разработки гетерогенных информационных систем.

Созданная логическая модель данных системы является основанием для создания физической модели данных под выбранную СУБД.

ЗАДАНИЕ 4 ДЛЯ САМОСТОЯТЕЛЬНОЙ ПРАКТИЧЕСКОЙ РАБОТЫ ПО ДИСЦИПЛИНЕ «ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА БАЗ И ХРАНИЛИЩ ДАННЫХ»

1. Изучить методологию моделирования данных.
2. Построить физическую модель данных на основе разработанной логической модели данных.

4.1. Создание физической модели данных

4.1.1. Выбор физического уровня представления модели данных

Создание физической модели является вторым шагом построения модели данных системы. ERwin позволяет построить физическую модель как независимую, так и зависимую от логического уровня представления модели данных.

Построение физической модели, независимой от логического уровня представления модели данных, начинается с активизации диалога Create Model через пункты основного меню File>New или кнопку нового файла, при этом в окне выбора нового типа модели New Model Type следует выбрать физический уровень представления (кнопка Physical).



Рисунок 4.1. Окно выбора нового типа модели New Model Type

Физический уровень представления модели зависит от конкретной реализации СУБД, поэтому необходимо предварительно осуществить ее выбор. CA ERwin Data Modeler поддерживает 17 наиболее распространенных СУБД. Выбор СУБД осуществляется в окне Target Database диалога Create Model. Для выбора СУБД необходимо открыть выпадающий список с перечнем поддерживаемых СУБД и щелкнуть по соответствующему имени. При этом в поле Version отобразится версия выбранной СУБД.

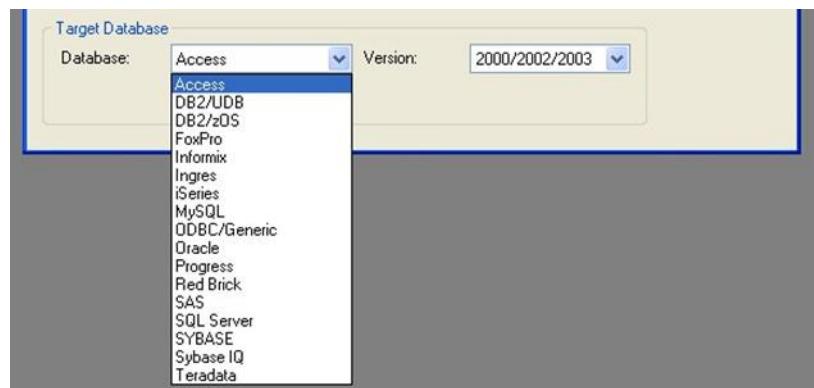


Рисунок 4.2. Окно выбора СУБД Target Database

При активизации кнопки OK диалога Create Model откроется поле для построения физической модели данных.

При выборе логико-физической модели для проектирования базы данных преобразование логической модели в физическую модель осуществляется автоматически по переходу к пункту Physical в списке выбора для переключения между логической и физической моделью, расположенным на панели инструментов.



Рисунок 4.3. Переход к физической модели через пункт Physical панели инструментов

В случае автоматического перехода 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, также принимаются по умолчанию, т.е. сохраняются.

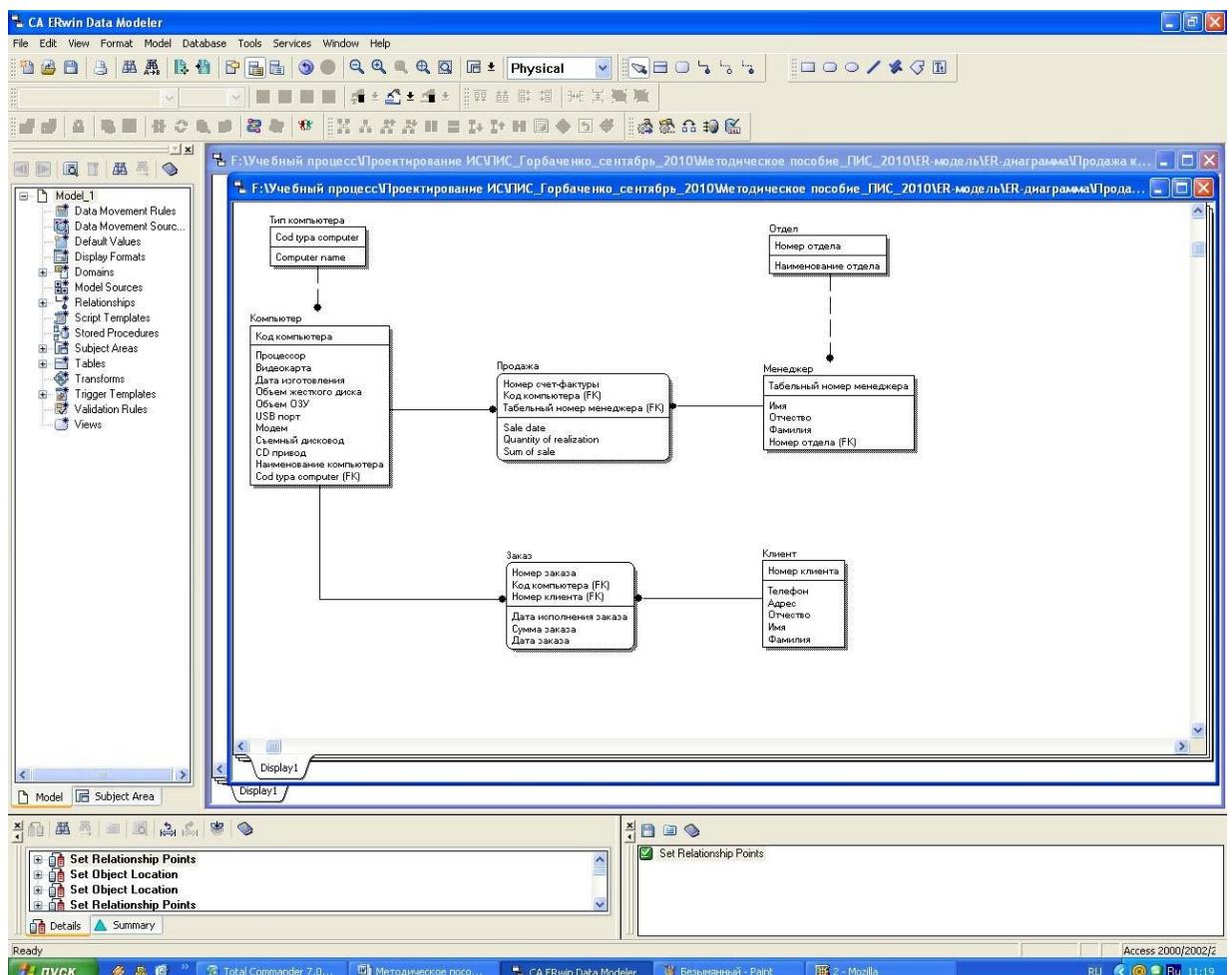


Рисунок 4.4. Физический уровень логико-физической модели данных системы

При генерации имени таблицы или колонки по умолчанию ERwin по возможности (частично) преобразовывает их названия в соответствии с англоязычным

представлением. При реализации базы данных с помощью СУБД Access допускается сохранять названия таблиц и колонок на русском языке.

4.1.2. Добавление/редактирование таблиц

Для построения/редактирования ER-модели физического уровня используется панель инструментов, содержащая кнопки редактирования модели, условные графические обозначения которых приведены в таблице 4.1.

Таблица 4.1. Палитра инструментов физического уровня

Вид кнопки	Назначение кнопки
	Указатель (режим мыши) – в этом режиме можно установить фокус на каком-либо объекте модели
	Создание новой таблицы – для создания таблицы необходимо щелкнуть левой кнопкой мыши по кнопке и один раз по свободному пространству на модели
	Создание нового представления (view table) – для создания представления необходимо щелкнуть левой кнопкой мыши по кнопке и один раз по свободному пространству на модели
	Создание идентифицирующей связи
	Создание связи между представлением и временной таблицей
	Создание неидентифицирующей связи

Для внесения новой таблицы в модель на физическом уровне необходимо щелкнуть по кнопке, расположенной на палитре инструментов, а затем на поле проектирования модели в том месте, где необходимо расположить новую таблицу. В результате в поле проектирования будет размещена таблица с именем по умолчанию Е/1.

Определение/редактирование имени таблицы осуществляется через вкладку General пункта Table Properties (свойства таблицы) контекстного меню, отображаемого по щелчку правой кнопкой мыши по выделенной сущности (рис. 4.5, 4.6), или пункт главного меню Model/Tables....

При этом открывается диалог Tables, отражающий имя выбранной для реализации СУБД. Например, при выборе для реализации СУБД Access диалог Tables будет называться Access Tables (рис. 4.7).

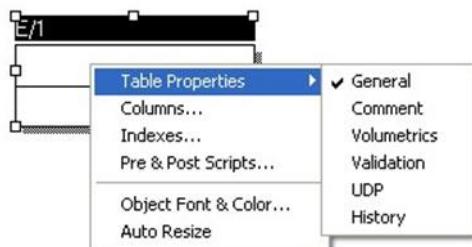


Рисунок 4.5. Контекстное меню добавленной таблицы

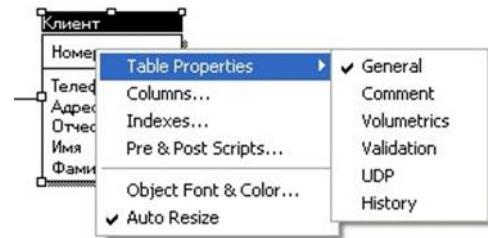


Рисунок 4.6. Контекстное меню таблицы Клиент

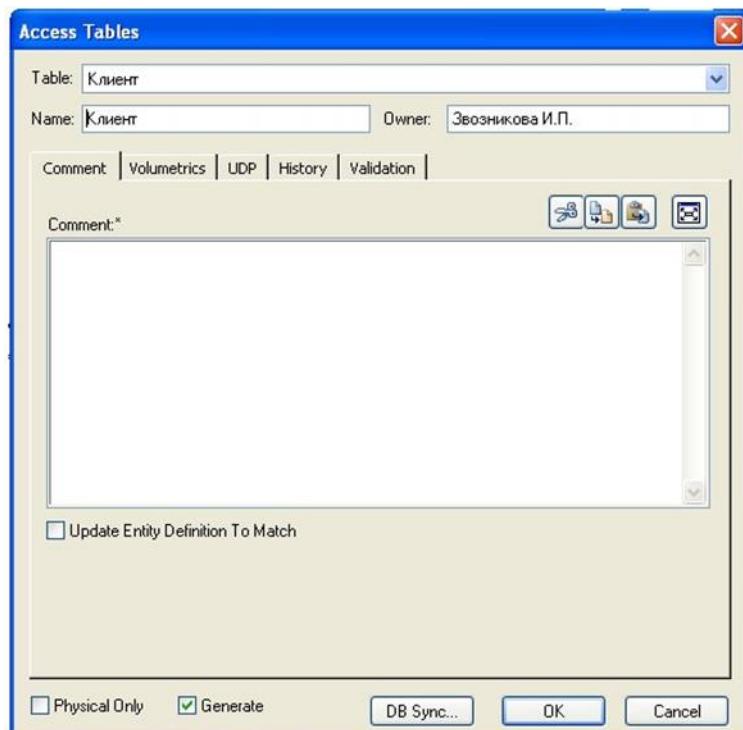


Рисунок 4.7. Диалог Access Tables

Диалог Tables (в нашем случае Access Tables) позволяет переключаться с одной таблицы на другую и задать свойства любой таблицы модели, отличные от значения по умолчанию.

Переключиться на другую таблицу можно при помощи раскрывающегося списка выбора таблиц для редактирования окна Table, расположенного в верхней части диалога.

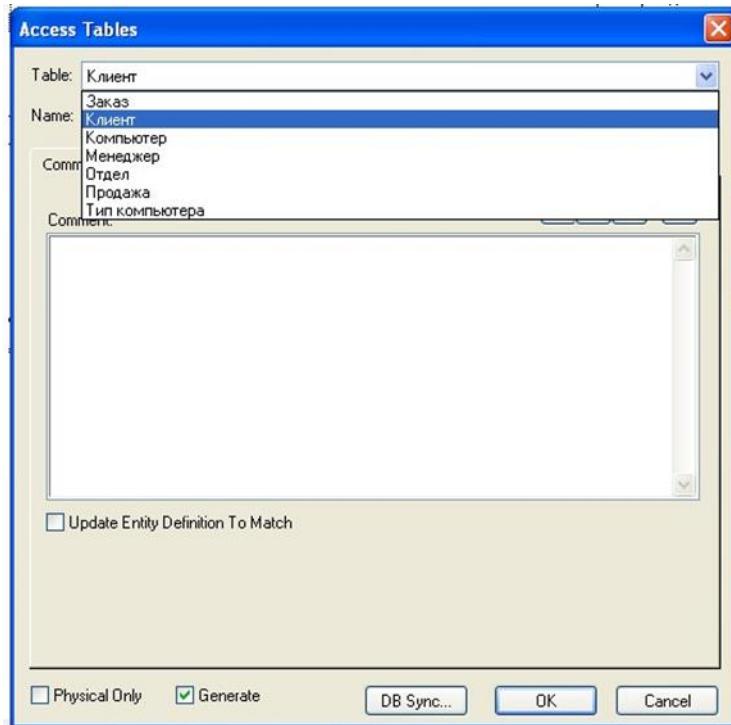


Рисунок 4.8. Список выбора таблиц для редактирования окна Table

Окно Name служит для задания/изменения имени текущей таблицы. Окно Owner позволяет внести/изменить имя владельца таблицы, отличное от имени пользователя, производящего генерацию логической модели базы данных. Окно выбора Physical Only предназначено для создания объектов только на физическом уровне. Выбор опции Generate позволяет сгенерировать логическую модель базы данных путем выполнения команды CREATE TABLE (создать таблицу). Кнопка DB Sync предназначена для синхронизации модели с системным каталогом базы данных (рис. 4.9). Диалог Tables содержит следующие вкладки:

- Comment – предназначена для внесения комментария к таблице;
- Volumetrics – предназначена для оценки размера БД;
- UDP – предназначена для задания свойств, определяемых пользователем;
- Validation – предназначена для задания правил валидации;
- History – отображает историю создания и изменения свойств таблицы и позволяет добавить комментарии к изменению в окне Comment.

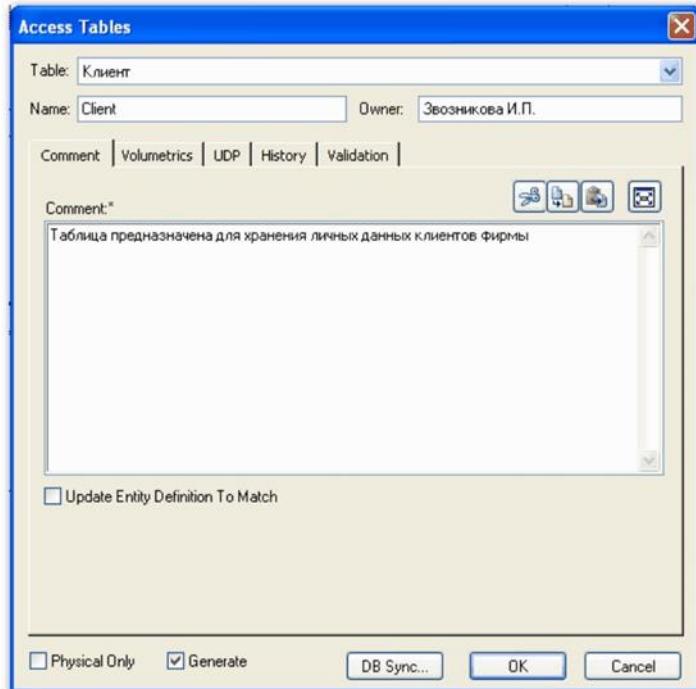


Рисунок 4.9. Изменение имени и внесение имени владельца таблицы Клиент

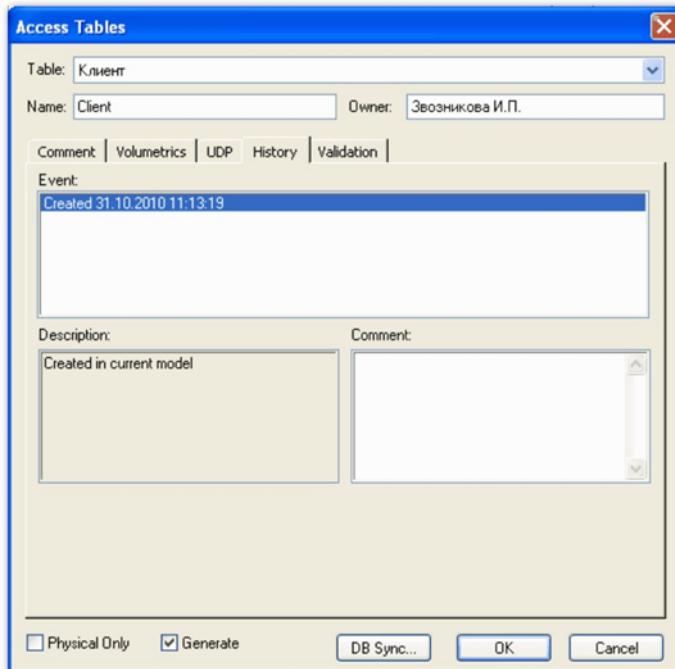


Рисунок 4.10. История создания и изменения свойств таблицы Клиент

Таблицы физической модели данных определяются в соответствии с семантическим анализом предметной области. При этом каждому объекту (сущности) предметной области ставится в соответствие таблица, характеристикам объекта (атрибутам) соответствуют колонки (поля) таблицы, а идентификатору объекта соответствует ключ (ключевое поле) таблицы.

4.1.3. Определение свойств колонок таблицы

Задание/редактирование свойств колонок таблицы осуществляется с помощью редактора диалогового окна Columns, которое открывается через пункт Columns

контекстного меню выделенной таблицы или пункт главного меню Model/Columns.... При этом редактор предлагает установить типы данных согласно выбранной СУБД.

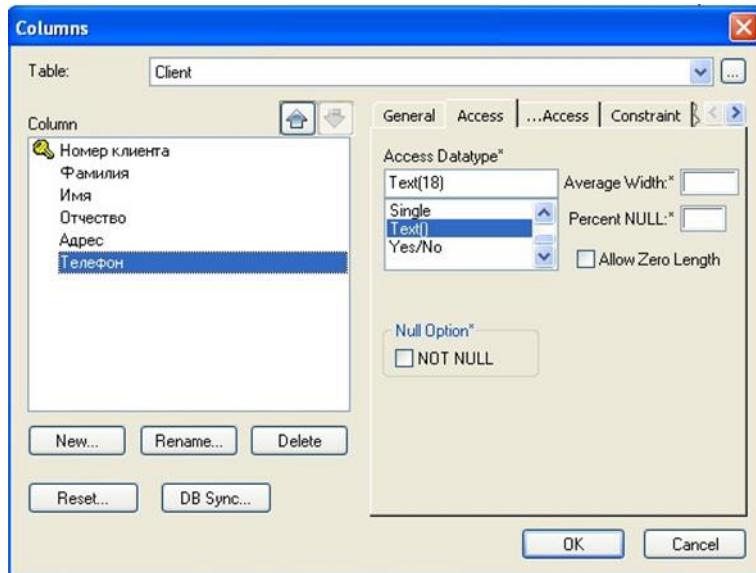


Рисунок 4.11. Диалоговое окно Columns

По умолчанию ERwin присваивает пустые значения (NULL) всем неключевым колонкам, а для колонок первичного ключа и альтернативных ключей устанавливает режим NOT NULL. Режим NOT NULL не присваивается автоматически инверсным входам (Inversion Entry).

Диалоговое окно Columns по своему внешнему виду напоминает диалоговое окно редактора свойств атрибутов Attributes и содержит окна и вкладки, позволяющие добавлять, редактировать и удалять колонки выделенной таблицы.

Вкладка General позволяет присвоить колонку определенному домену, создать колонку только на физическом уровне и включить ее в состав первичного ключа.

Вкладка, соответствующая выбранной СУБД, называется по имени СУБД и позволяет задать тип данных, опцию NULL и значение по умолчанию (рис. 4.11). Имя вкладки устанавливается автоматически и соответствует названию выбранной для реализации СУБД. Например, при выборе для реализации базы данных системы СУБД Access вкладка будет называться Access. Для СУБД Access создается дополнительная вкладка для задания свойств колонки (рис. 4.13).

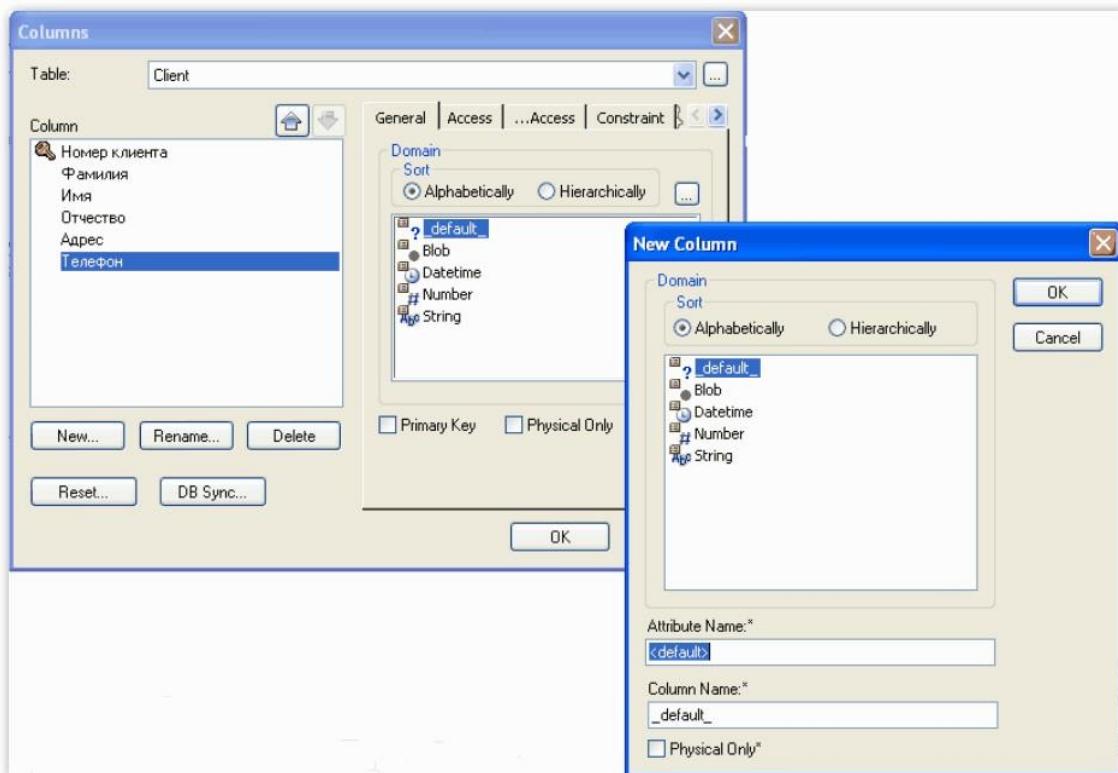


Рисунок 4.12. Вкладка General

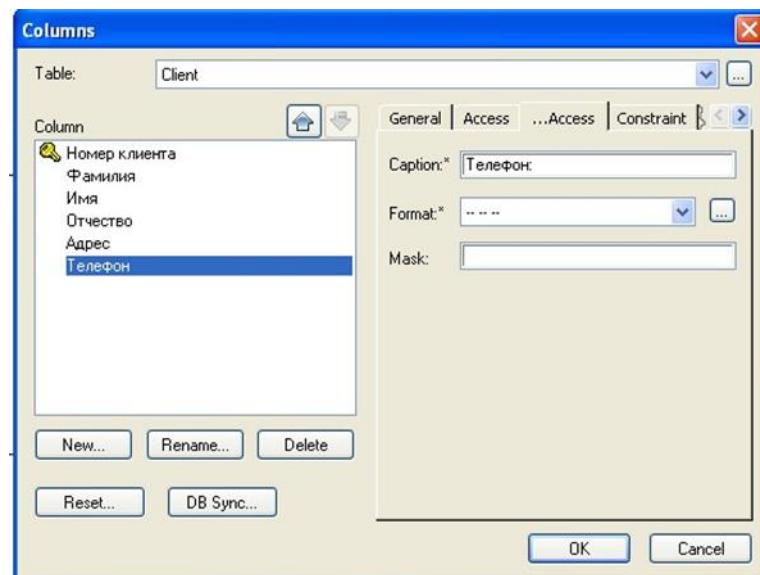


Рисунок 4.13. Вкладка ... Access

Вкладка Constraint позволяет задать имена и значения ограничений, накладываемых на поле указанной колонки таблицы, включая и значение, которое присваивается в окне Default автоматически, т.е. по умолчанию.

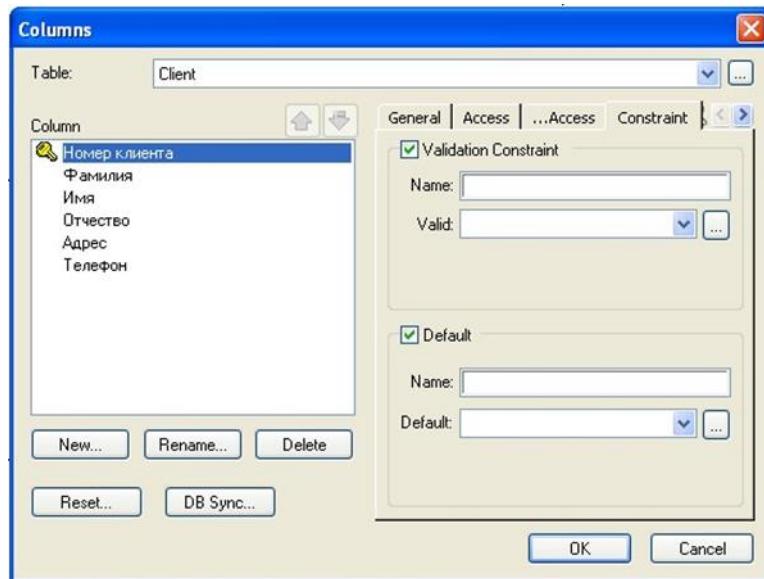


Рисунок 4.14. Вкладка Constraint

Значение по умолчанию – значение, которое нужно ввести в колонку, если никакое другое значение не задано явным образом во время ввода данных. Для создания нового значения ограничения по умолчанию необходимо перейти по кнопке поля Default в диалоговое окно Default/Initial Values, по активизации кнопки New открыть диалог New Default Value, ввести в поле Name имя правила и щелкнуть по кнопке OK.

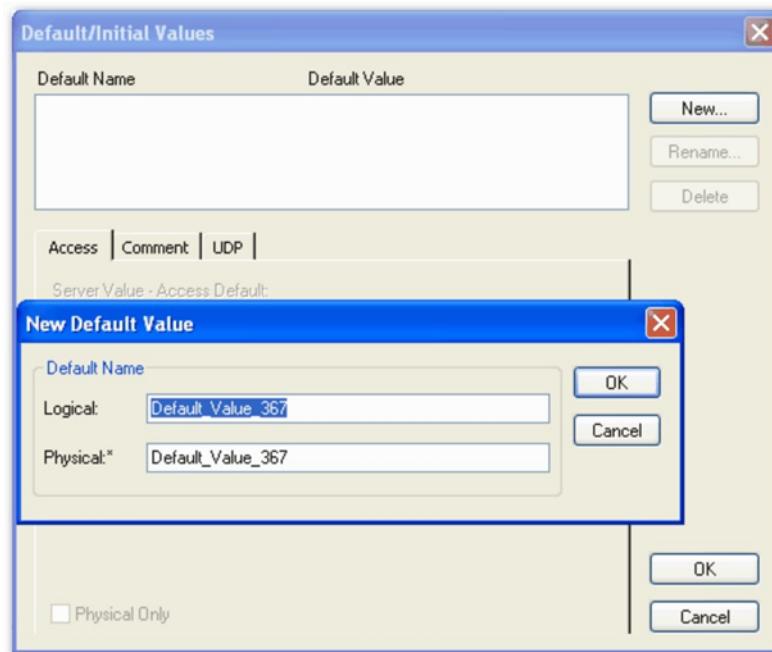


Рисунок 4.15. Создания нового значения ограничения по умолчанию

Список всех заданных имен значений по умолчанию отображается в поле Default Name диалогового окна Default/Initial Values. Для удаления и переименования значений по умолчанию используют соответственно кнопки Delete и Rename.

Вкладка Comment предназначена для внесения комментария к каждой колонке. Вкладка UDP позволяет задать свойства, определяемые пользователем. Вкладка Index служит для включения колонки в состав индексов. Вкладка History отображает историю создания и изменения свойств колонки и позволяет добавить комментарии к изменению в окне Comment.

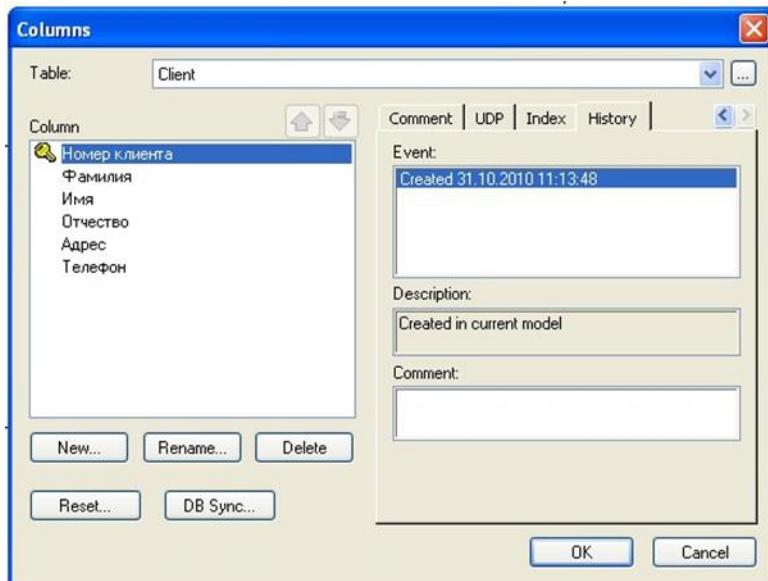


Рисунок 4.16. Вкладка History

Упорядоченный список колонок таблицы отображается в поле Column, расположенный в левой части диалогового окна Columns. Для перемещения колонок в списке на позицию вверх или вниз используют соответственно кнопки.

Кнопки New, Rename и Delete служат соответственно для создания, переименования и удаления колонки. При помощи кнопки Reset можно переустановить свойства колонки, заданные вручную, на значения по умолчанию. Кнопка DB Sync позволяет запустить процесс синхронизации модели с системным каталогом БД.

Связи между таблицами физической модели создаются так же, как и при создании логического уровня модели данных. При создании связи колонки первичного ключа родительской таблицы мигрируют в состав колонок дочерней таблицы в качестве внешнего ключа.

Изменения, внесенные в имена таблиц и колонок физической модели, не отражаются на именах сущностей и атрибутов, поскольку информация на логическом и физическом уровнях в ERwin хранится отдельно.

4.1.4. Индексы

Индекс – это особый объект, позволяющий решить проблему поиска данных в таблице БД.

Индекс содержит отсортированную по колонке или нескольким колонкам информацию и указывает на строки, в которых хранится конкретное значение колонки.

Возникновение проблемы поиска данных связано с тем, что многие реляционные СУБД имеют страничную организацию, при которой физически таблица может храниться фрагментарно в разных областях диска, причем строки таблицы располагаются на страницах неупорядоченно, т.к. данные обычно хранятся в том порядке, в котором их ввели в таблицу.

Хотя такой способ хранения позволяет быстро вводить новые данные, но для того, чтобы найти нужную строку, необходимо просмотреть всю таблицу. В промышленных системах каждая таблица может содержать миллионы строк, поэтому простой перебор ведет к катастрофическому падению производительности ИС.

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 клиента по имени, то можно создать индекс по колонке ClientName таблицы Client. В индексе имена клиентов будут отсортированы в алфавитном порядке. Для имени индекс будет содержать ссылку, указывающую, в каком месте таблицы хранится эта строка. Индекс можно создать для всех колонок таблицы, по которым часто производится поиск. Для поиска клиента серверу направляется запрос с критерием поиска (ClientName = "Иванов"). При выполнении запроса СУБД просматривает индекс, а не все по порядку строки таблицы Client. Поскольку значения в индексе хранятся в определенном порядке, то просматривать нужно гораздо меньший объем данных, что значительно уменьшает время выполнения запроса.

При генерации схемы физической БД 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 колонки наиболее часто используются для поиска данных.

Для повышения производительности системы ERwin позволяет создать собственные индексы. При этом целесообразно создавать индексы с различными колонками 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" – для альтернативного, "IEn" – для инверсионного входа. Например, по умолчанию при создании таблицы Manager будут созданы индексы XPKМенеджер – первичный ключ, в состав которого войдет колонка Number manager ID и XIF1Менеджер – внешний ключ, в состав которого войдет колонка Department number FK.

Изменить характеристики существующего индекса или создать новый индекс можно в редакторе Indexes, открывающийся при выборе пункта Indexes... контекстного меню, появляющегося по щелчку правой кнопки мыши на выделенной таблице (рис. 4.17). Редактор Indexes позволяет изменить имя индекса и его определение, а также порядок сортировки данных.

ERwin позволяет создавать индексы, которые могут содержать либо повторяющиеся, либо только уникальные значения.

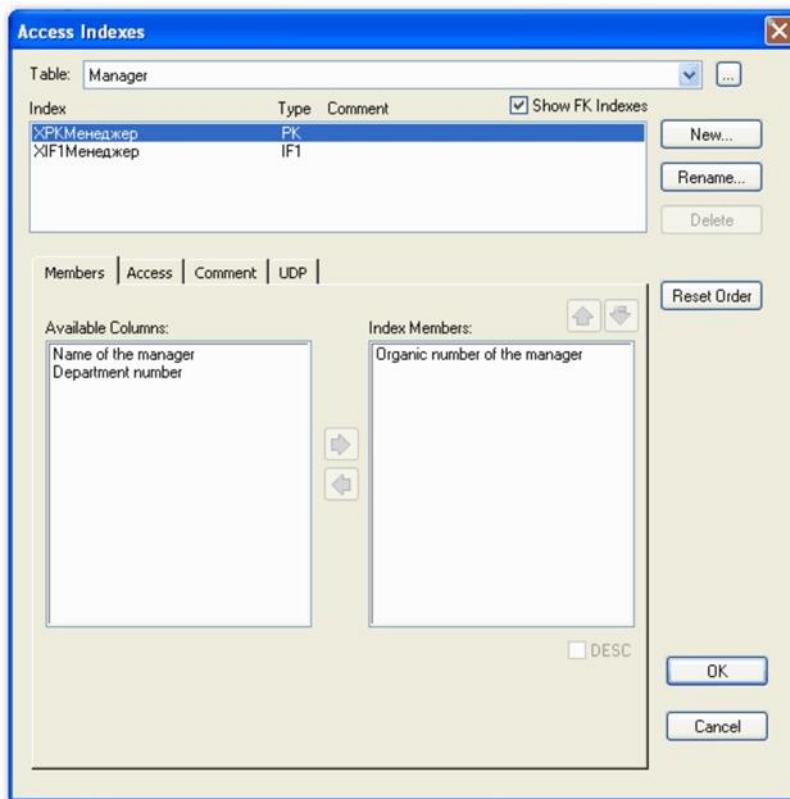


Рисунок 4.17. Редактор Indexes

Создание нового уникального индекса осуществляется через диалог New Index, открывающийся по активизации кнопки New..., при включенной опции Unique (рис. 4.18).

Уникальные индексы генерируются на основе первичного и альтернативных ключей и предотвращают попытки вставить запись с неуникальным (повторяющимся) значением посредством извещения пользователя об ошибке и игнорирования на его действия по добавлению неверного (повторяющегося) значения индекса.

Для создания индекса с повторяющимися значениями опцию Unique следует выключить. Повторяющиеся значения индекса разрешаются, если ожидается, что индексированная колонка будет с большой вероятностью содержать повторяющуюся информацию. Неуникальный индекс генерируется на основе внешнего ключа.

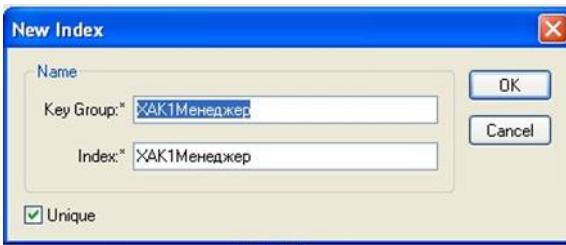


Рисунок 4.18. Создание нового уникального индекса

При создании нового индекса ERwin автоматически создает альтернативный ключ для уникального индекса и инверсионный вход для неуникального индекса, а также дает соответствующее ключу имя индекса. Имя сгенерированного индекса в дальнейшем при необходимости можно изменить вручную через диалог Rename Index, открывающийся по активизации кнопки Rename....

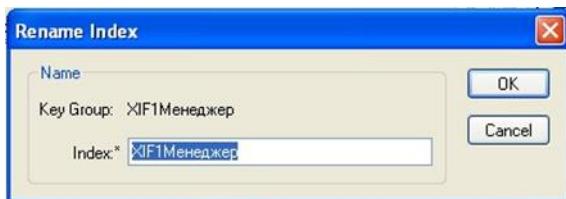


Рисунок 4.19. Диалог Rename Index

Редактор Indexes содержит следующие вкладки:

- Members – позволяет включить колонки в состав индекса;
- вкладка, соответствующая выбранной СУБД (например, Access), задает свойства индекса, специфические для выбранной СУБД;
- Comment – содержит комментарий для каждого индекса;
- UDP – позволяет связать с индексом свойства, определяемые пользователем.

Такие СУБД, как MS Access, MS SQL Server и т.п., поддерживают кластеризованные или кластеризованные хешированные индексы. Кластеризация индекса – это специальная техника индексирования, при которой данные в таблице физически располагаются в индексированном порядке. Использование кластеризованного индекса значительно ускоряет выполнение запросов по индексированной колонке. Хеширование – альтернативный способ хранения данных в заранее заданном порядке с целью ускорения поиска. Кластеризованный или хешированный индекс значительно ускоряет операции поиска и сортировки, но добавление и удаление строк замедляется из-за необходимости реорганизации данных для соответствия индексу.

Для того чтобы сделать индекс кластеризованным, необходимо включить опцию Cclustered на вкладке, соответствующей выбранной СУБД. Например, можно создать кластеризованный индекс в таблице Manager по колонке Department number FK. В

результате информация обо всех менеджерах одного отдела будет физически располагаться на диске рядом, что значительно повысит скорость выполнения запроса, который делает выборку данных по менеджерам определенного отдела. Поскольку данные в БД физически располагаются в индексированном порядке, то для каждой таблицы может существовать только один кластеризованный индекс.

Если СУБД поддерживает использование кластеризованного индекса, например, Access, то ERwin автоматически создает индекс первичного ключа кластеризованным, а при создании кластеризованного индекса не по первичному ключу автоматически снимает кластеризацию с индекса по первичному ключу.

4.1.5. Правила валидации колонок

Правило валидации задает список допустимых значений для конкретной колонки таблицы и/или правила проверки допустимых значений. Значение по умолчанию - значение, которое нужно ввести в колонку, если никакое другое значение не задано явным образом во время ввода данных.

Задание правил валидации осуществляется через диалог Validation Rules, который открывается через пункт главного меню Model/ Validation Rules... или через диалоговые окна, открывающиеся в следующем порядке:

- 1) активизировать кнопку "...", расположенную справа от раскрывающегося списка Table диалогового окна Columns (рис. 4.16);
- 2) в открывшемся окне выбрать закладку Validation и активизировать кнопку Validation Constraint... (рис. 4.20).

В результате открывается диалог Validation Rules, в котором можно задать максимальное и минимальное число колонок для всех таблиц модели, а также тип валидации: где проверять – на сервере или в клиентском приложении (рис. 4.21).

Для создания нового правила валидации необходимо активизировать кнопку New..., ввести имя правила в поле Validation Rule Name диалога New Validation Rule и активизировать кнопку OK (рис. 4.22.). Наименование правила валидации может быть разным на логическом и физическом уровне. Чтобы переименовать имеющееся правило валидации, необходимо активизировать кнопку Rename. Для удаления правила валидации предназначена кнопка Delete.

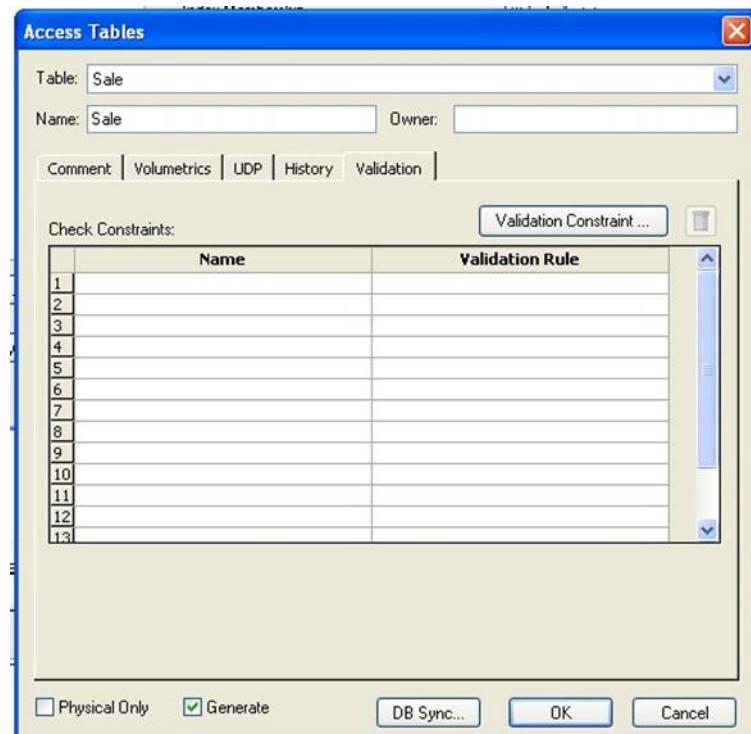


Рисунок 4.20. Выбор закладки Validation и активизация кнопки Validation Constraint...

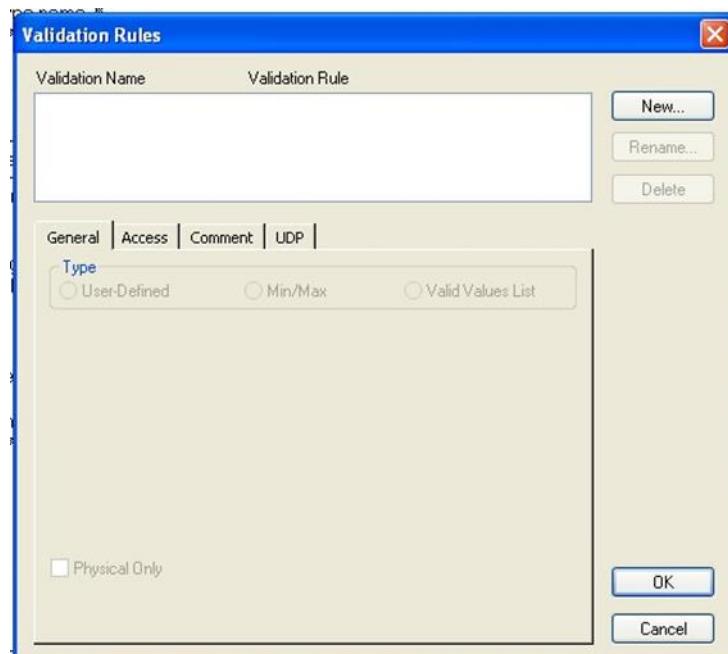


Рисунок 4.21. Диалог Validation Rules

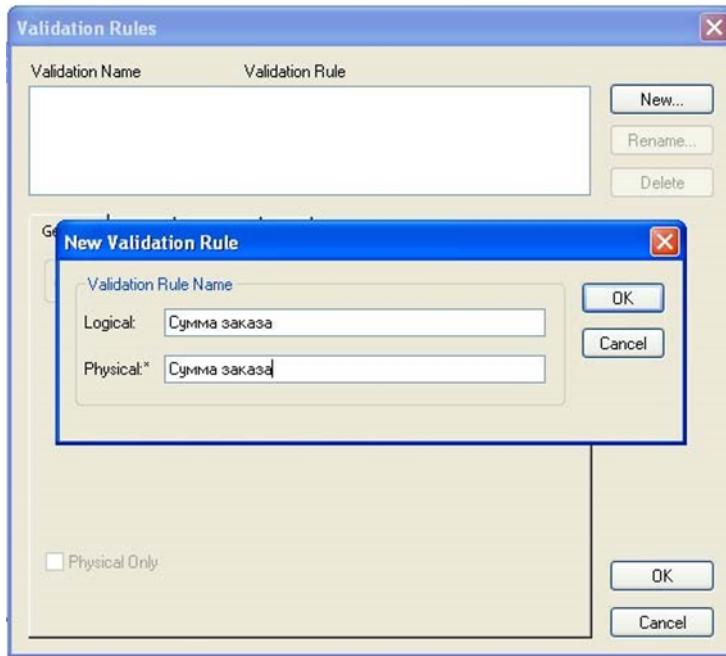


Рисунок 4.22. Создание нового правила валидации

В результате в верхней части редактора Validation Rules отобразится список всех созданных правил валидации и представится возможность редактирования правил валидации (рис. 4.23).

Закладка General редактора задания правил валидации позволяет выбрать тип правила:

- тип User-Defined позволяет задать вручную фрагмент SQL – выражения, который соответствует правилу валидации и будет использоваться при генерации схемы базы данных;
- тип Min/Max позволяет задать минимальное и максимальное значение колонки, которое будет проверяться в базе данных на вхождение в заданный диапазон;
- тип Valid Values List позволяет задать список допустимых значений.

Пример правила валидации типа Min/Max для колонки Сумма заказа таблицы Заказ приведен на рисунке 4.24.

Вкладка, соответствующая выбранной СУБД (рис. 4.24 – Access) позволяет просмотреть фрагмент SQL – выражения, соответствующего правилу валидации, которое гарантирует проверку вводимых значений (рис. 4.25). В случае выхода за границы заданного диапазона СУБД выдает сообщение об ошибке.

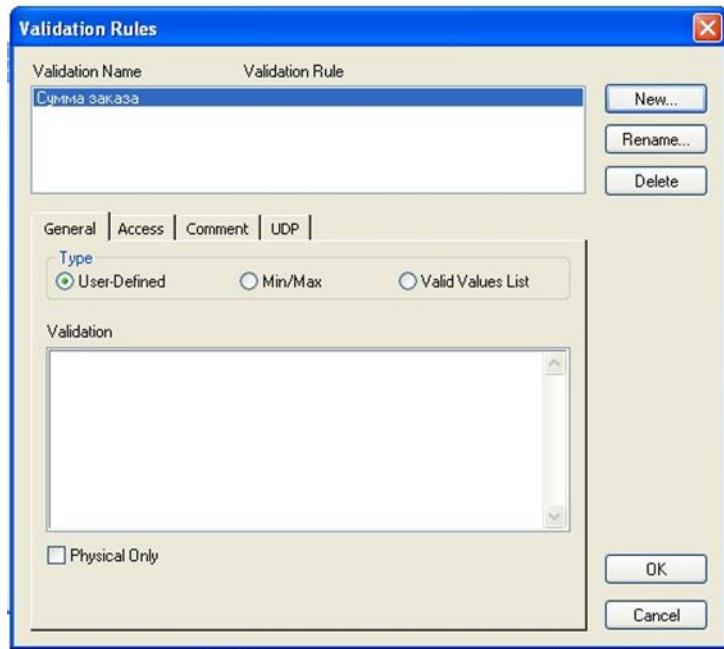


Рисунок 4.23. Создание нового правила валидации

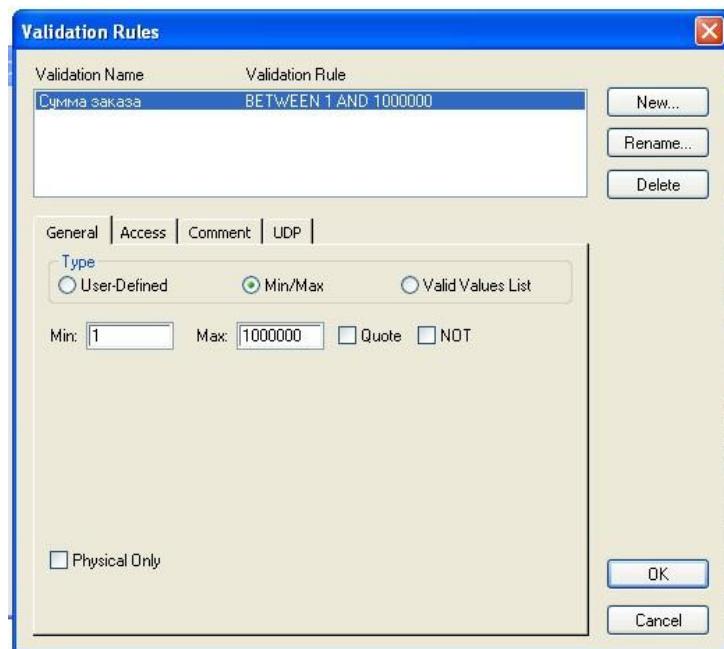


Рисунок 4.24. Пример правила валидации типа Min/Max

Закладка Comment предназначена для внесения комментария к правилу валидации.

Закладка UDP позволяет для правила валидации задать свойства, определяемые пользователем.

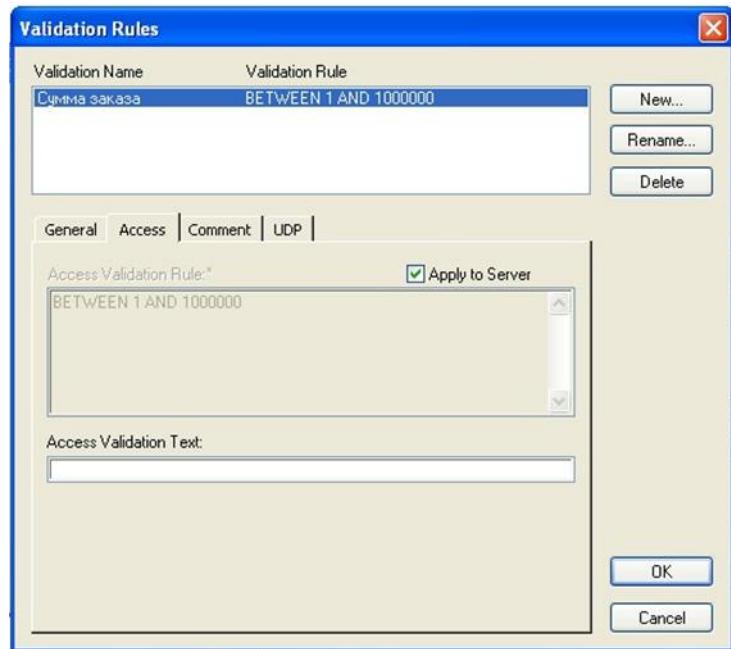


Рисунок 4.25. Просмотр фрагмента SQL – выражения через вкладку Access

4.1.6. Пример физической модели данных

Построение физической модели данных для системы "Реализация средств вычислительной техники" осуществлено путем автоматического перехода от логической модели к физической модели, так как при создании логической модели данных системы был выбран логико – физический тип модели.

Физическая модель данных системы "Реализация средств вычислительной техники" приведена на рисунке 4.26.

Созданная физическая модель данных предназначена для реализации базы данных в среде Access.

Для генерации кода создания базы данных можно использовать пункт главного меню Tools/Forward Engineer. В результате откроется окно установки свойств генерируемой схемы данных (рис. 4.27).

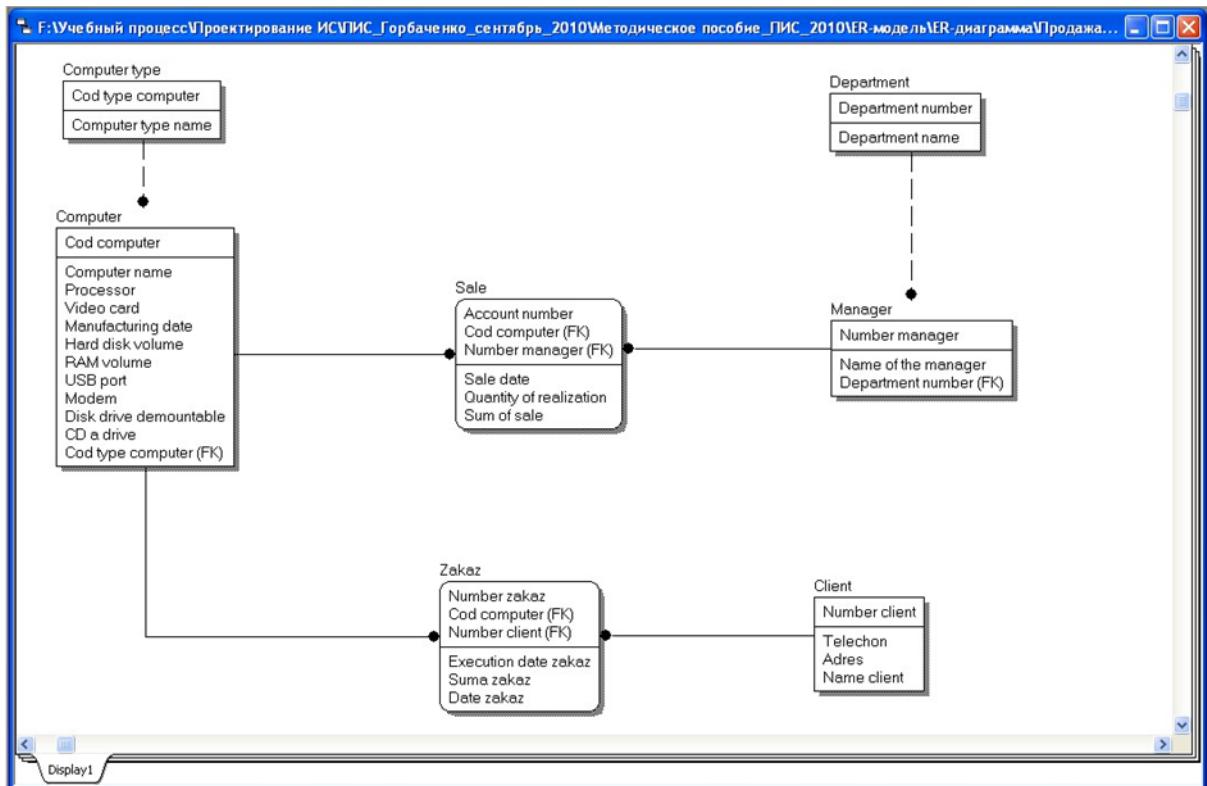


Рисунок 4.26. Физическая модель данных системы
"Реализация средств вычислительной техники"

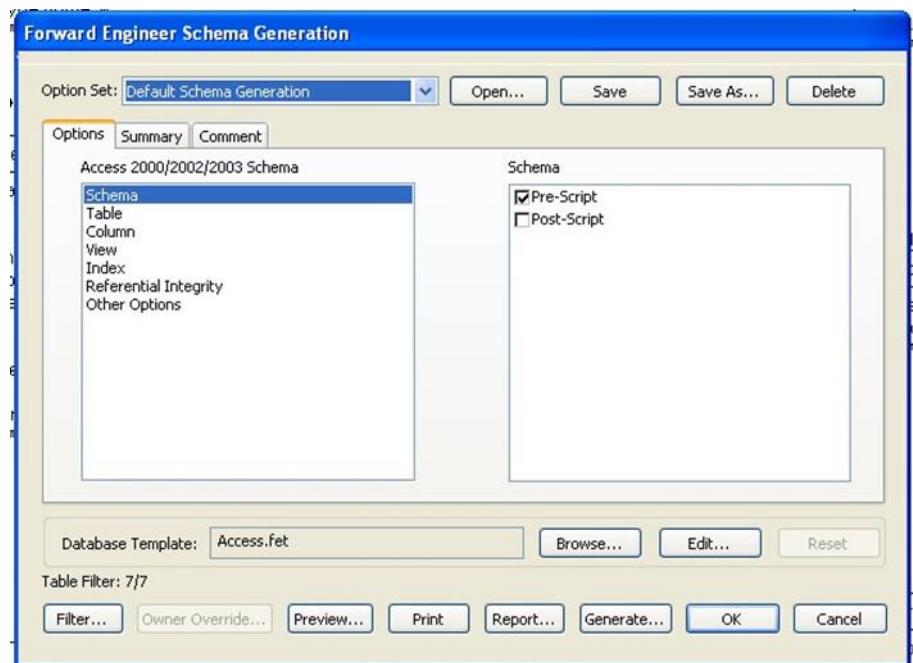


Рисунок 4.27. Окно установки свойств генерируемой схемы данных

Для предварительного просмотра SQL-скрипта служит кнопка Preview (рис. 4.28), для генерации схемы – кнопка Generate (рис. 4.29).

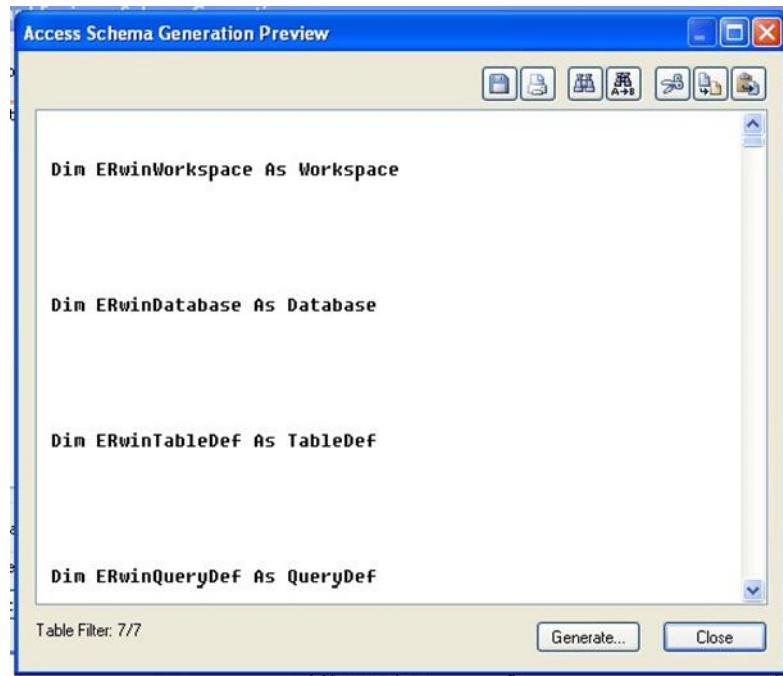


Рисунок 4.28. Окно просмотра SQL-скрипта

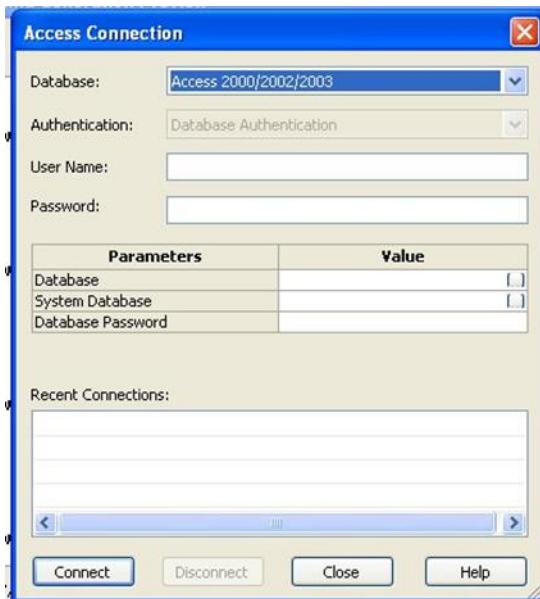


Рисунок 4.29. Окно генерации схемы данных

В процессе генерации ERwin связывается с выбранной базой данных, выполняя SQL-скрипт.

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

Окно установки свойств генерируемой схемы данных позволяет также просмотреть и править шаблон базы данных путем активизации соответственно кнопки *Browse...* или *Edit...* (рис. 4.30).

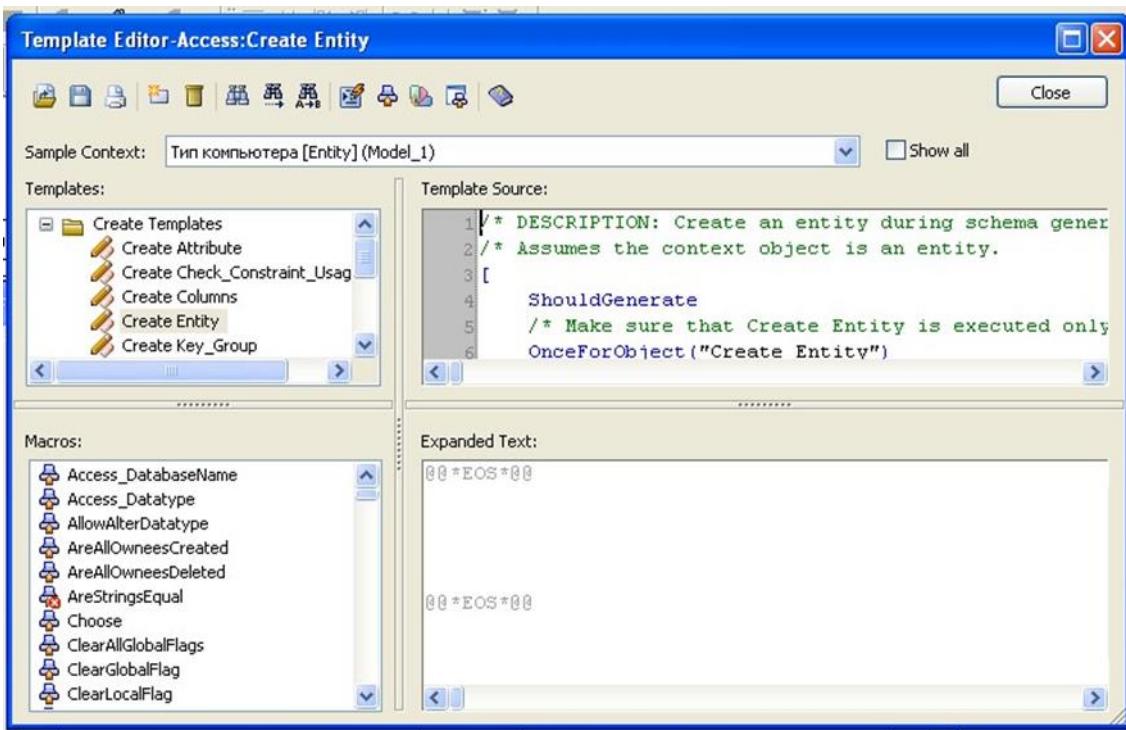


Рисунок 4.30. Окно правки шаблона базы данных

Диаграмма физической модели данных является необходимым и очень удобным материалом для разработчиков программ. Физическое проектирование позволяет обеспечить безопасность и целостность данных в выбранной для реализации СУБД.

Применение программного продукта CA ERwin Data Modeler существенно повышает эффективность деятельности разработчиков информационных систем за счет:

- существенного повышения скорости разработки базы данных с помощью мощного редактора диаграмм, возможности автоматической генерации базы данных и автоматической подготовки документации;
- наличия возможности автоматической подготовки SQL – предложений для создания базы данных;
- наличия возможности внесения изменений в модель при разработке и расширении системы;
- наличия возможности автоматической подготовки отчетов по базе данных, которые соответствуют реальной структуре базы данных;
- наличия возможности осуществления обратного проектирования, что позволяет документировать и вносить изменения в существующие информационные системы;
- поддержки однопользовательских СУБД, что позволяет использовать для персональных систем современные технологии и значительно упростить переход от настольных систем к системам с технологией клиент-сервер.