Лабораторная работа N 4

Проектирование моделей данных в ERWin

Задание. Построить модель данных информационной системы по теме проекта. Предусмотреть в модели связи многие-ко-многим, один-ко многим, категориальные связи. На основе логической модели получить физическую модель. Выбрать сервер БД и сгенерировать базу данных для СУБД ACCESS/

Сведения из теории

Моделирование данных

Case-метод Баркера

Цель моделирования данных состоит в обеспечении разработчика ИС концептуальной схемой базы данных в форме одной модели или нескольких локальных моделей, которые относительно легко могут быть отображены в любую систему баз данных.

Наиболее распространенным средством моделирования данных являются диаграммы "сущность-связь" (ERD). С их помощью определяются важные для предметной области объекты (сущности), их свойства (атрибуты) и отношения друг с другом (связи). ERD непосредственно используются для проектирования реляционных баз данных.

Нотация ERD была впервые введена П. Ченом (Chen) и получила дальнейшее развитие в работах Баркера. Метод Баркера будет излагаться на примере моделирования деятельности компании по торговле автомобилями. Ниже приведены выдержки из интервью, проведенного с персоналом компании.

Главный менеджер: одна из основных обязанностей - содержание автомобильного имущества. Он должен знать, сколько заплачено за машины и каковы накладные расходы. Обладая этой информацией, он может установить нижнюю цену, за которую мог бы продать данный экземпляр. Кроме того, он несет ответственность за продавцов и ему нужно знать, кто что продает и сколько машин продал каждый из них.

Продавец: ему нужно знать, какую цену запрашивать и какова нижняя цена, за которую можно совершить сделку. Кроме того, ему нужна основная информация о машинах: год выпуска, марка, модель и т.п.

Администратор: его задача сводится к составлению контрактов, для чего нужна информация о покупателе, автомашине и продавце, поскольку именно контракты приносят продавцам вознаграждения за продажи.

Первый шаг моделирования - извлечение информации из интервью и выделение сущностей.

Сущность (Entity) - реальный либо воображаемый объект, имеющий существенное значение для рассматриваемой предметной области, информация о котором подлежит хранению (рисунок 1).

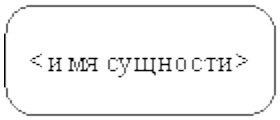


Рис. 1. Графическое изображение сущности

Каждая сущность должна обладать уникальным идентификатором. Каждый экземпляр сущности должен однозначно идентифицироваться и отличаться от всех других экземпляров данного типа сущности. Каждая сущность должна обладать некоторыми свойствами:

- каждая сущность должна иметь уникальное имя, и к одному и тому же имени должна всегда применяться одна и та же интерпретация. Одна и та же интерпретация не может применяться к различным именам, если только они не являются псевдонимами;

- сущность обладает одним или несколькими атрибутами, которые либо принадлежат сущности, либо наследуются через связь;
- сущность обладает одним или несколькими атрибутами, которые однозначно идентифицируют каждый экземпляр сущности;
- каждая сущность может обладать любым количеством связей с другими сущностями модели.

Обращаясь к приведенным выше выдержкам из интервью, видно, что сущности, которые могут быть идентифицированы с главным менеджером - это автомашины и продавцы. Продавцу важны автомашины и связанные с их продажей данные. Для администратора важны покупатели, автомашины, продавцы и контракты. Исходя из этого, выделяются 4 сущности (автомашина, продавец, покупатель, контракт), которые изображаются на диаграмме следующим образом (рисунок 2).

Рис. 2.

Следующим шагом моделирования является идентификация связей.

Связь (Relationship) - поименованная ассоциация между двумя сущностями, значимая для рассматриваемой предметной области. Связь - это ассоциация между сущностями, при которой, как правило, каждый экземпляр одной сущности, называемой родительской сущностью, ассоциирован с произвольным (в том числе нулевым) количеством экземпляров второй сущности, называемой сущностью-потомком, а каждый экземпляр сущности-потомка ассоциирован в точности с одним экземпляром сущности-родителя. Таким образом, экземпляр сущности-потомка может существовать только при существовании сущности родителя.

Связи может даваться имя, выражаемое грамматическим оборотом глагола и помещаемое возле линии связи. Имя каждой связи между двумя данными сущностями должно быть уникальным, но имена связей в модели не обязаны быть уникальными. Имя связи всегда формируется с точки зрения родителя, так что предложение может быть образовано соединением имени сущности-родителя, имени связи, выражения степени и имени сущности-потомка.

Например, связь продавца с контрактом может быть выражена следующим образом:

- продавец может получить вознаграждение за 1 или более контрактов;
- контракт должен быть инициирован ровно одним продавцом.

Степень связи и обязательность графически изображаются следующим образом (рисунок 3).

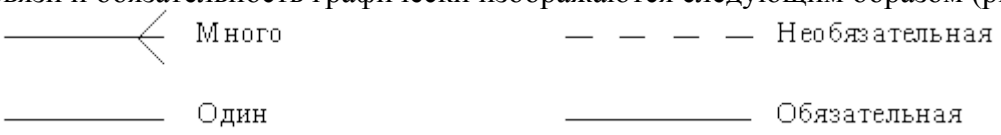


Рис. 3.

Таким образом, 2 предложения, описывающие связь продавца с контрактом, графически будут выражены следующим образом (рисунок 3).

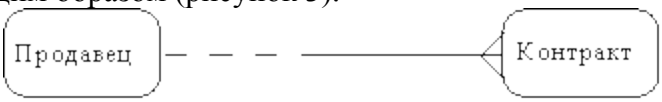


Рис. 4.

Описав также связи остальных сущностей, получим следующую схему (рисунок 5).

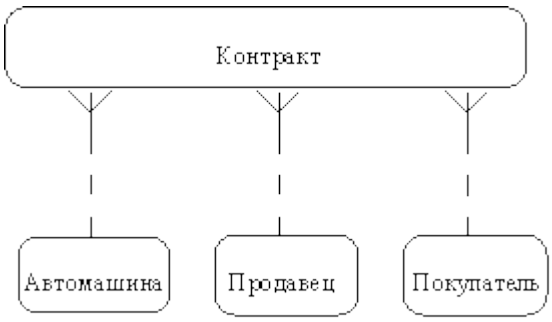


Рис. 5.

Последним шагом моделирования является идентификация атрибутов.

Атрибут - любая характеристика сущности, значимая для рассматриваемой предметной области и предназначенная для квалификации, идентификации, классификации, количественной характеристики или выражения состояния сущности. Атрибут представляет тип характеристик или свойств, ассоциированных со множеством реальных или абстрактных объектов (людей, мест, событий, состояний, идей, пар предметов и т.д.). Экземпляр атрибута - это определенная характеристика отдельного элемента множества. Экземпляр атрибута определяется типом характеристики и ее значением, называемым значением атрибута. В ER-модели атрибуты ассоциируются с конкретными сущностями. Таким образом, экземпляр сущности должен обладать единственным определенным значением для ассоциированного атрибута.

Атрибут может быть либо обязательным, либо необязательным (рисунок 6). Обязательность означает, что атрибут не может принимать неопределенных значений (null values). Атрибут может быть либо описательным (т.е. обычным дескриптором сущности), либо входить в состав уникального идентификатора (первичного ключа).

Уникальный идентификатор - это атрибут или совокупность атрибутов и/или связей, предназначенная для уникальной идентификации каждого экземпляра данного типа сущности. В случае полной идентификации каждый экземпляр данного типа сущности полностью идентифицируется своими собственными ключевыми атрибутами, в противном случае в его идентификации участвуют также атрибуты другой сущности-родителя (рисунок 7).

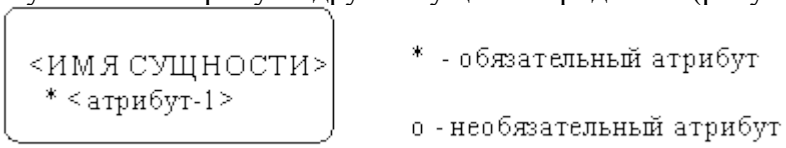


Рис. 8.

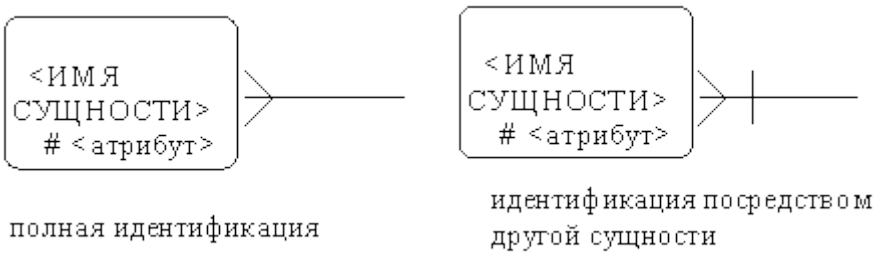


Рис. 9.

Каждый атрибут идентифицируется уникальным именем, выражаемым грамматическим оборотом существительного, описывающим представляемую атрибутом характеристику. Атрибуты изображаются в виде списка имен внутри блока ассоциированной сущности, причем каждый атрибут занимает отдельную строку. Атрибуты, определяющие первичный ключ, размещаются наверху списка и выделяются знаком "#".

Каждая сущность должна обладать хотя бы одним возможным ключом. Возможный ключ сущности - это один или несколько атрибутов, чьи значения однозначно определяют каждый экземпляр сущности. При существовании нескольких возможных ключей один из них обозначается в качестве первичного ключа, а остальные - как альтернативные ключи.

С учетом имеющейся информации дополним построенную ранее диаграмму (рисунок 10).

Помимо перечисленных основных конструкций модель данных может содержать ряд дополнительных.

Подтипы и супертипы: одна сущность является обобщающим понятием для группы подобных сущностей (рисунок 11).

Взаимно исключающие связи: каждый экземпляр сущности участвует только в одной связи из группы взаимно исключающих связей (рисунок 12).

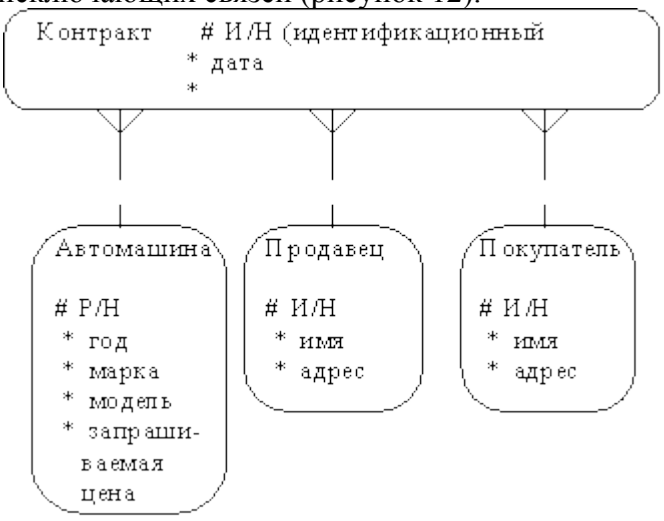


Рис. 13.

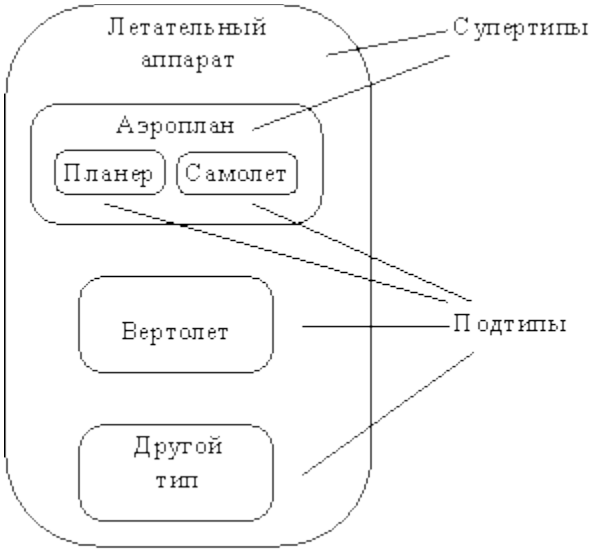


Рис.14. Подтипы и супертипы

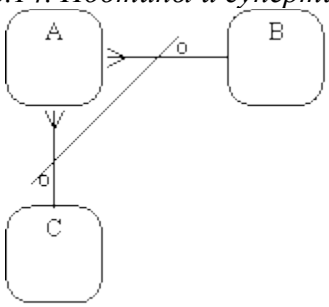


Рис. 15. Взаимно исключающие связи

Рекурсивная связь: сущность может быть связана сама с собой (рисунок 16).

Неперемещаемые (non-transferrable) связи: экземпляр сущности не может быть перенесен из одного экземпляра связи в другой (рисунок 17).

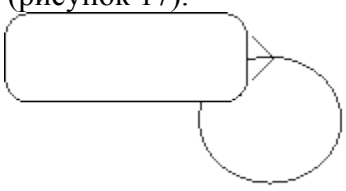


Рис. 16. Рекурсивная связь

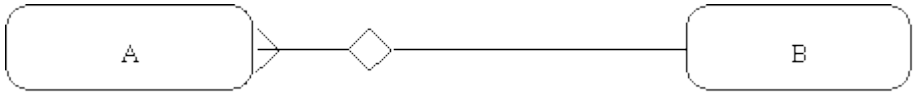
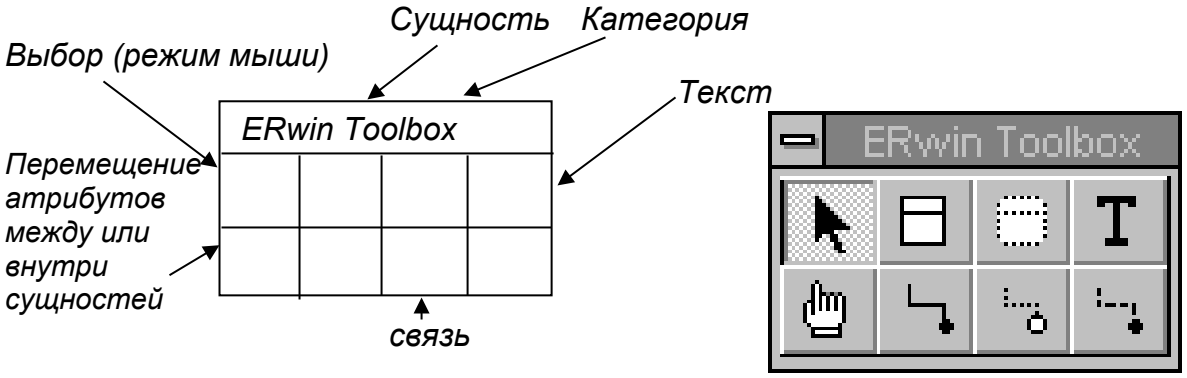


Рис. 17. Неперемещаемая связь

Работа с ERWin

Палитра инструментов на логическом уровне



Используются две нотации создания моделей - IDEF1X (армия США и госучреждения, финансовые и промышленные корпорации), IE (промышленность).
Переключение: *Option / Preferences / Methodology*

ERwin имеет несколько уровней отображения диаграммы

- уровень сущностей
- уровень атрибутов
- уровень определений
- уровень первичных ключей
- уровень иконок

Переключение - через контекстное меню (ПЩ на свободном месте, пункт Display Level) или через кнопки палитры инструментов (первые 3 уровня)

Уровни отображения модели

Уровень отображения	*Представление модели*
Сущность (Entity)	Деталь
Атрибуты (Attribute)	Сотрудник Табельный номер Номер паспорта (AK1.1) Фамилия (AK2.1) Имя (AK2.2) Отчество (AK2.3) Дата рождения (AK2.4) Отдел (IE1.1)
Первичный ключ (Primary Key)	Деталь Номер детали
Определение (Definition)	Деталь Содержит информацию о деталях

Сущности с изображением малых иконок	$ счет
Сущности с изображением больших иконок	 $ счет

Установка цвета и шрифта - панель инструментов *Font and Color Toolbar*

Создание логической модели в ERWin

Ключи

Каждая сущность должна иметь по крайней мере один *потенциальный ключ* (поле или набор полей с уникальными значениями)
Первичный ключ (Primary key) - определяет экземпляр сущности уникальным образом.
Альтернативный ключ (Alternate Key) - потенциальный ключ, не ставший первичным. При генерации схемы БД по всем атрибутам альтернативного ключа генерируется уникальный индекс.

Можно вводить неуникальные индексы (допускаются совпадения). Атрибуты, участвующие в неуникальных индексах, называются *инверсионные входы* (Inversion Entries). Они не определяет экземпляр сущности уникальным образом, но используются для частого обращения к экземплярам сущности.

Денормализация в ERwin

В результате нормализации все взаимосвязи данных становятся правильно определены., исключаются аномалии при оперировании с данными, модель данных становится легче поддерживать. Однако часто нормализация данных не ведет к повышению производительности БД.

Пример. Производные атрибуты, ведущие к нарушению первой НФ.
Сотрудник= {Таб номер, Фамилия, …, Код Должности}
Должность= { Код Должности, Название должности, Оклад}
Связь необязательная один ко многим

Денормализуем - будет работать быстрее
Сотрудник= {Таб номер, Фамилия, …, Должность, Оклад}
Должность= { Должность, Оклад}

Аномалии обновления: если меняется оклад должности - нужно менять у всех сотрудников. Решение: обновлять - только в таблице Должность, выбирать данные - только из таблицы Сотрудник. Нужна процедура синхронизации таблиц.

Домены

Домен - совокупность значений, из которых берутся значения атрибутов. Каждый атрибут может быть определен только на одном домене. На каждом домене может быть определено множество атрибутов.

Домен имеет уникальное имя и может использоваться как на логическом, так и на физическом уровне.

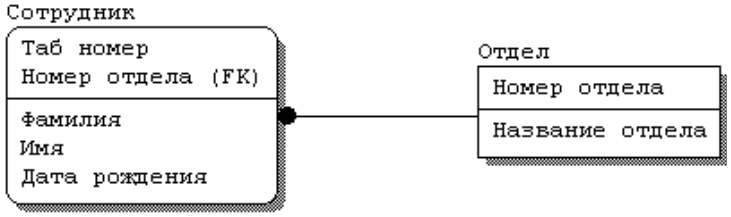
На логическом уровне домены описываются без физических свойств. На физическом уровне они автоматически получают специфические свойства.

Пример: Домен "Возраст"
Логический уровень - атрибуты получат тип Number
Физический уровень - колонкам будет присвоен тип INTEGER
Редактирование доменов - команда Edit/Domain Dictionary и кнопка

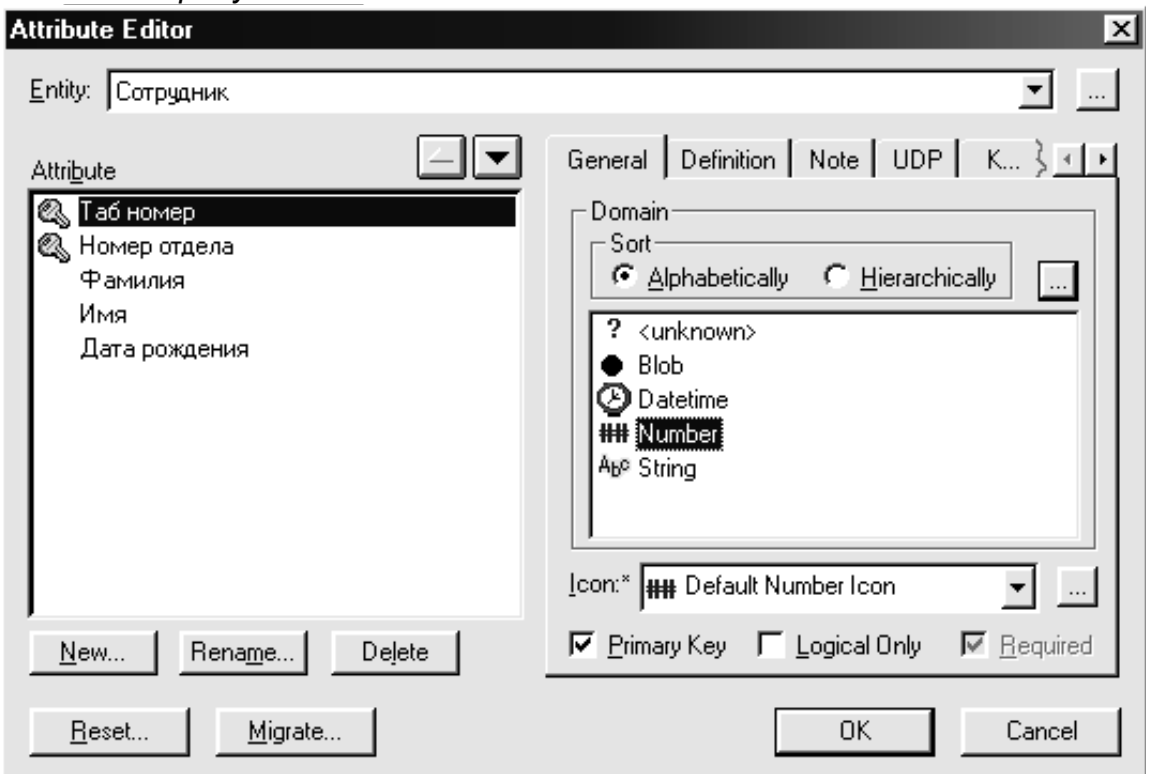
Создание новых атрибутов в модели - диалог Independent Attribute Browser
(Ключ CTRL +B)

General - задание родительского домена (Domain Parent), имени колонки, Physical Only - определяет домен только на уровне физической модели.
Comment - комментарий к домену
UDP -
Visual Basic - Power Builder - Задание специальных свойств домена для кодогенерации клиентского приложения.

Пример логической модели



Задание атрибутов модели - диалог



Создание Физической модели данных

Физическая модель содержит всю информацию, необходимую для реализации конкретной БД
Различают два уровня физической модели

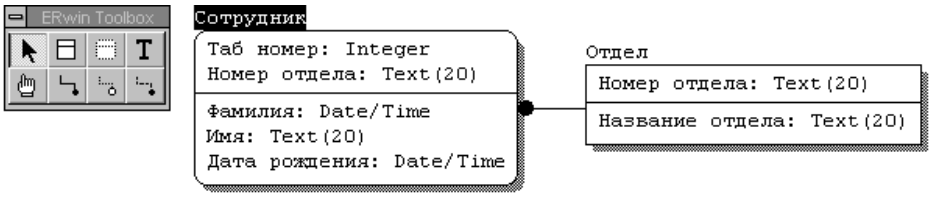
- трансформационная модель (Transformation Model)
- модель СУБД (DBMS Model)

Трансформационная модель содержит всю информацию для реализации отдельного проекта, который может быть частью общей ИС и может описывать подмножество предметной области.

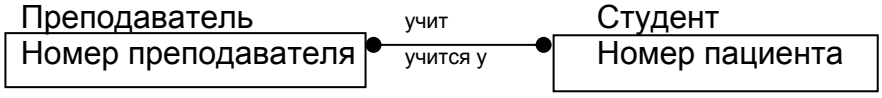
Проектировщик может выделить подмножество модели в виде предметных областей (Subject Area)

Модель СУБД автоматически генерируется из трансформационной модели и является точным отражением системного каталога СУБД

Пример физической модели

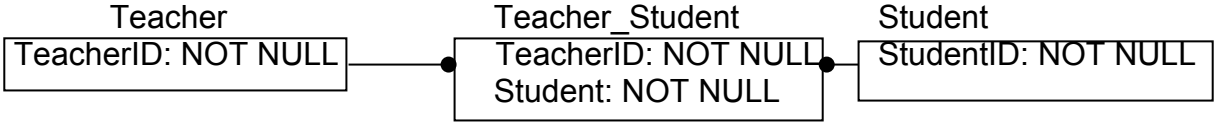


Связь - многие-ко-многим. Возможна только на уровне логической *модели* данных

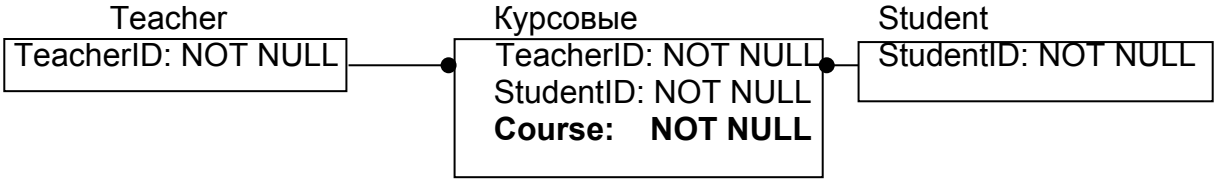


Именуется двумя фразами учит / учится у

При переходе к физическому уровню ERwin автоматически преобразует связь:



Такой таблицы недостаточно, надо дополнить таблицу (таблица "Выполнение курсовых работ") колонкой "Дисциплина" или "Дата". Таблица называется **"Курсовые".**



При этом на логическом уровне диаграмма не изменится.

Типы сущностей и иерархия наследования

Связи определяют, является ли сущность независимой или зависимой. Зависимые сущности

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

Пример. Сотрудник ———● Увлечения

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

Пример - таблица связи "Курсовые"

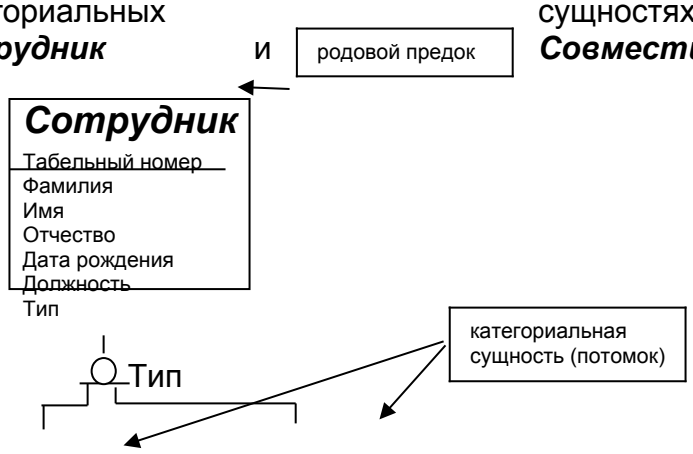
Именующая - частный случай ассоциативной сущности, не имеющей собственных атрибутов (только атрибуты родительских сущностей, мигрирующих в качестве внешнего ключа)

Пример - таблица Teacher_Student

Категориальная - дочерняя сущность в иерархии наследования

Иерархии наследования (или иерархия категорий) - представляет собой особый тип объединения сущностей, которые разделяют общие характеристики. Например, в организации работают служащие, занятые полный рабочий день (постоянные служащие) и совместители. Из их общих свойств можно сформировать обобщенную

сущность ***Сотрудник,*** чтобы представить информацию, общую для всех типов служащих. Специфическая для каждого типа информация может располагаться в категориальных сущностях (потомках) ***Постоянный сотрудник*** и ***Совместитель.***



Неполная категория

Сущность консультант не снесена в иерархию наследования

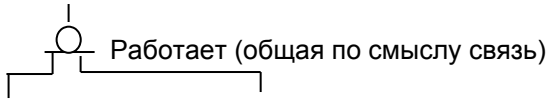
Постоянный сотрудник

Табельный номер (FK)

Совместитель

Табельный номер (FK)
Ставка

Иерархию наследования создают, когда несколько сущностей имеют общие по смыслу атрибуты, либо когда сущности имеют общие по смыслу связи



Особенность: Одна таблица через одно поле связана в двумя или более таблицами !

Постоянный сотрудник

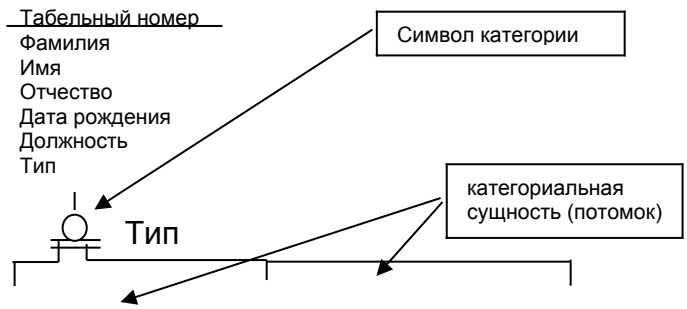
Совместитель

Полные и неполные категории

В ***полной*** категории каждому экземпляру родового предка обязательно соответствует экземпляр в каком -либо потомке, т.е. каждый служащий обязательно является либо совместителем, либо консультантом, либо постоянным сотрудником.

Категория может быть еще не выстроена полностью. Тогда в родовом предке могут существовать экземпляры, которые не имеют соответствующих экземпляров в потомках. Такая категория будет неполной.

Сотрудник



Полная категория

Все Сущности внесены в иерархию наследования

Постоянный сотрудник

Табельный номер (FK)

Совместитель

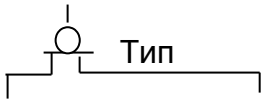
Табельный номер (FK)
Ставка

Консультант

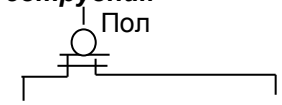
Табельный номер (FK)
Организация

Комбинация полной и неполной категорий

Сотрудник



Постоянный сотрудник ***Совместитель***



Создание категориальной связи

- установить курсор на кнопке o в палитре инструментов и ЛК мыши
- щелкнуть сначала по родовому предку, потом - по потомку
- для установления второй связи в иерархии категорий сначала щелкнуть по символу категории, потом по второму потомку

Редактирование категории - ПЩ по символу категории В контекстном меню - пункт - Subtype Relationship Editor. Указать дискриминатор категории (Discriminator Attribute Choice) (например - атрибут *Тип* в родовом предке) и тип категории - полная/неполная (Complete/Incomplete).

Стадии построения иерархии наследования

1. *Определение сущностей с общими (по определению) атрибутами*
   пример 1: Постоянный сотрудник и Совместитель
   пример 2:Транзисторы низкочастотные, транзисторы высокочастотные

2. *Перенос общих атрибутов в сущность - родовой предок*
   пример 1: Постоянный сотрудник и Совместитель - > Сотрудник
   пример 2:Транзисторы низкочастотные, транзисторы высокочастотные -> транзистор

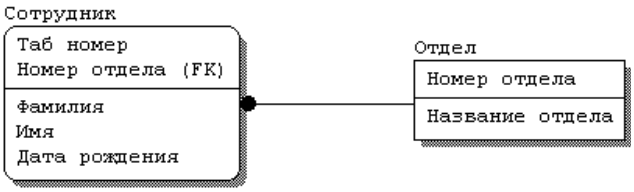
3. *Создание неполной структуры категорий*
   Создается категориальная связь от новой сущности - родового предка - к старым сущностям - потомкам. Новая сущность дополняется атрибутом-дискриминатором категории - **тип**

4. *Создание полной структуры категорий*
   Производится дополнительный поиск сущностей, имеющих общие по смыслу атрибуты с родовым предком. Общие атрибуты переносятся в родового предка и категория преобразуется в полную. Некоторые не общие атрибуты могут быть перенесены в сущность-потомка.
   пример1: консультант
   пример 2 : транзисторы сверхвысокочастотные.

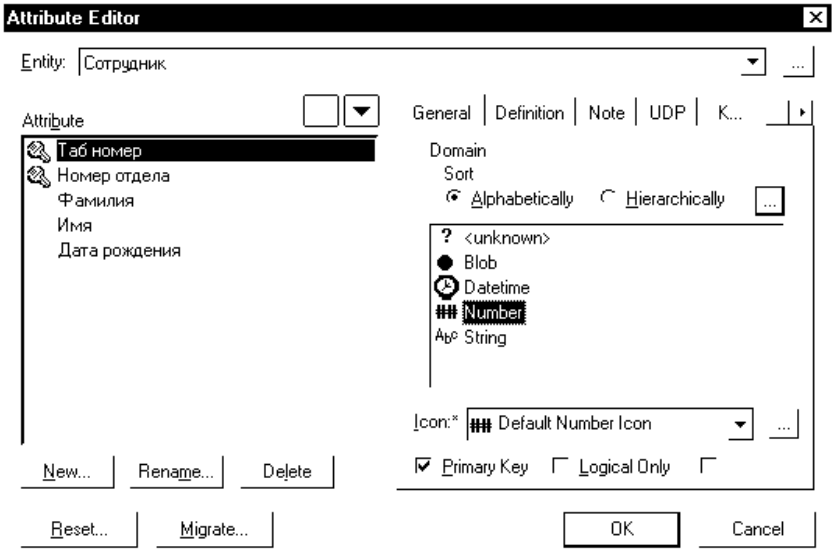
5. *Комбинация полной и неполной структур категорий*
   При необходимости создание иерархии категорий можно продолжить

Пример создания модели

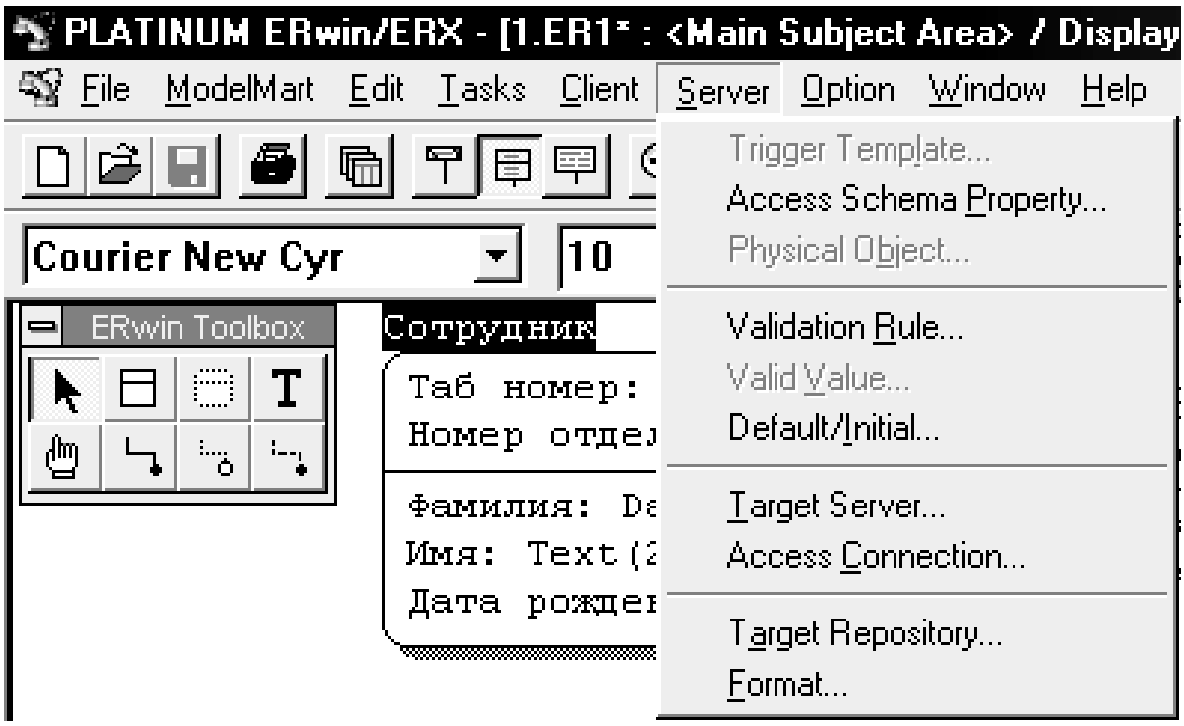
Логическая модель



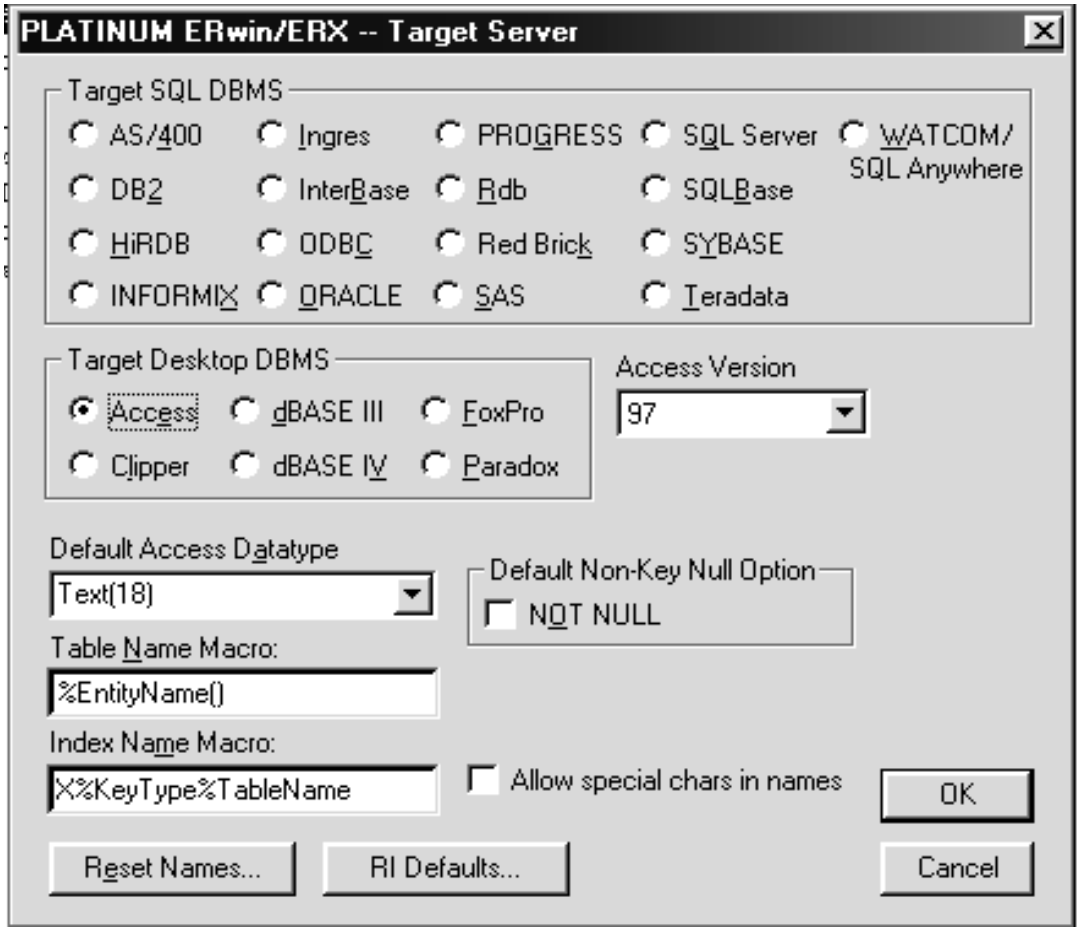
Создание модели - диалог



Настройка сервера БД
Меню – Server (сервер)



Выбор сервера
Физический уровень представления модели зависит от выбранного сервера. ERwin поддерживает более 20 реляционных и нереляционных СУБД.



Default <имя сервера> Datatype - задает тип данных по умолчанию. Зависит от выбранного сервера.

Default Non Key Option - позволяет разрешить или запретить значения NULL для неключевых колонок.

Allow special chars in names - позволяет разрешить или запретить использование специальных символов и пробелов в именах таблиц. Опция действует только для тех СУБД, которые поддерживают использование специальных символов.

По умолчанию ERwin генерирует имена таблиц и индексов по шаблону на основе имен соответствующих сущностей и ключей логической модели. Окна Table Name Macro и Index Name Macro позволяют изменить шаблон генерации имен, заданный по умолчанию. В дальнейшем имена таблиц и индексов можно изменять вручную.

Кнопка Reset Names вызывает диалог Globally Reset DBMS Property который позволяет заменить все имена таблиц, связей, индексов, колонок и соответствующих свойств, заданных вручную, на значения по умолчанию. Имена таблиц и колонок по умолчанию будут сгенерированы на основе имен сущностей и атрибутов логической модели. Пробел заменяется на символ подчеркивания.

Редактирование таблиц (Table Editor)

Можно задать имя таблицы, синонимы, правила валидации (проверки), процедуры, отличные от значений по умолчанию.

Name - имя текущей таблицы

Owner - имя владельца таблицы, возможно, отличное от имени пользователя

Physical Only - создание объектов только на физическом уровне

Generate [V] - если выбрана эта опция, то будет выполняться команда CREATE TABLE.

Кнопка DB Sync - немедленная синхронизация модели с системным каталогом БД

Закладки

Dimensional - для моделирования хранилищ данных

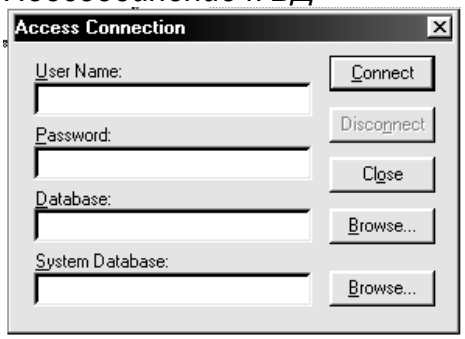
Comment - комментарии к таблице

Volumetrics - оценка размера БД

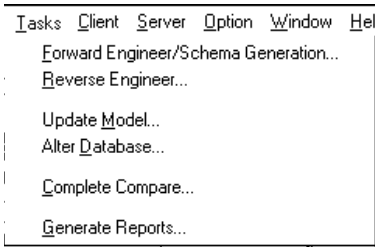
Physical Props - задает физические свойства таблицы
Partitions - задает значения разделения (только для Oracle)
UDP - свойства, определяемые пользователем
Validation - задание правил валидации
Synonym - задание синонимов таблицы (если сервер это поддерживает)
Stored Procedure - связывание с таблицей хранимых процедур
Pre& Post Script - создание скриптов (наборов команд) , которые будут выполняться до и после создания таблицы при генерации схемы БД
PowerBuilder - задание расширенных атрибутов для генерации кода клиентского приложения на PB

Редактирование колонок - редактор Column Editor

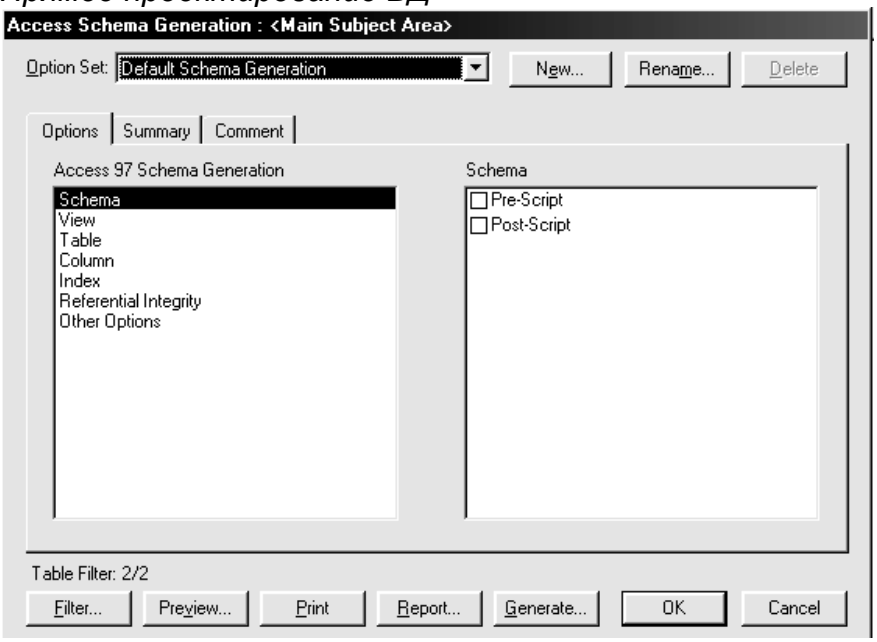
Подсоединение к БД



Задачи



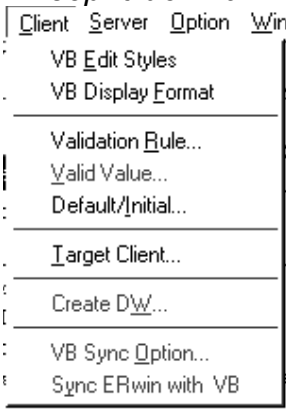
Прямое проектирование БД



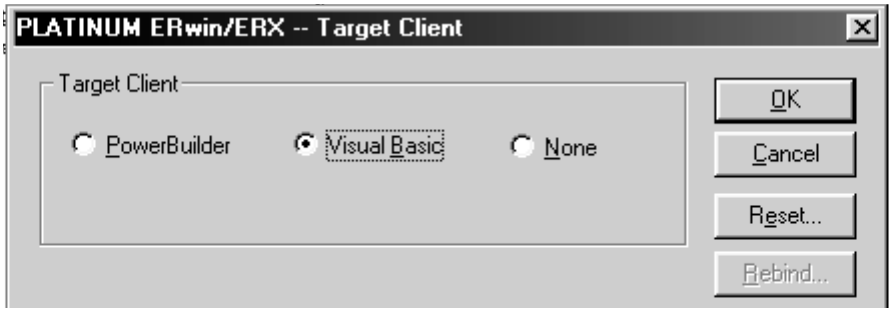
Предварительный просмотр генерации БД

```
'  Starting Access Basic DAO Session...

Dim ERwinWorkspace As Workspace
Dim ERwinDatabase As Database
Dim ERwinTableDef As TableDef
Dim ERwinQueryDef As QueryDef
Dim ERwinIndex As Index
Dim ERwinField As Field
Dim ERwinRelation As Relation

Set ERwinWorkspace = DBEngine.WorkSpaces(0)

Set ERwinDatabase = ERwinWorkspace.OpenDatabase(sERwinDatabase)

'  CREATE TABLE "Отдел"
Set ERwinTableDef = ERwinDatabase.CreateTableDef("Отдел")
Set ERwinField = ERwinTableDef.CreateField("Номер отдела", DB_TEXT, 20)
ERwinField.Required = True
ERwinTableDef.Fields.Append ERwinField
Set ERwinField = ERwinTableDef.CreateField("Название отдела", DB_TEXT, 20)
ERwinTableDef.Fields.Append ERwinField
ERwinDatabase.TableDefs.Append ERwinTableDef

'  CREATE INDEX "PrimaryKey"

Set ERwinTableDef = ERwinDatabase.TableDefs("Отдел")
Set ERwinIndex = ERwinTableDef.CreateIndex("PrimaryKey")
Set ERwinField = ERwinIndex.CreateField("Номер отдела")
ERwinIndex.Fields.Append ERwinField
ERwinIndex.Primary = True
ERwinIndex.Clustered = True
ERwinTableDef.Indexes.Append ERwinIndex

'  CREATE TABLE "Сотрудник"
Set ERwinTableDef = ERwinDatabase.CreateTableDef("Сотрудник")
Set ERwinField = ERwinTableDef.CreateField("Таб номер", DB_INTEGER)
ERwinField.Required = True
ERwinTableDef.Fields.Append ERwinField
Set ERwinField = ERwinTableDef.CreateField("Номер отдела", DB_TEXT, 20)
ERwinField.Required = True
ERwinTableDef.Fields.Append ERwinField
Set ERwinField = ERwinTableDef.CreateField("Фамилия", DB_DATETIME)
ERwinTableDef.Fields.Append ERwinField
Set ERwinField = ERwinTableDef.CreateField("Имя", DB_TEXT, 20)
ERwinTableDef.Fields.Append ERwinField
Set ERwinField = ERwinTableDef.CreateField("Дата рождения", DB_DATETIME)
ERwinTableDef.Fields.Append ERwinField
ERwinDatabase.TableDefs.Append ERwinTableDef

'  CREATE INDEX "PrimaryKey"

Set ERwinTableDef = ERwinDatabase.TableDefs("Сотрудник")
Set ERwinIndex = ERwinTableDef.CreateIndex("PrimaryKey")
Set ERwinField = ERwinIndex.CreateField("Таб номер")
ERwinIndex.Fields.Append ERwinField
Set ERwinField = ERwinIndex.CreateField("Номер отдела")
ERwinIndex.Fields.Append ERwinField
ERwinIndex.Primary = True
ERwinIndex.Clustered = True
ERwinTableDef.Indexes.Append ERwinIndex

'  CREATE RELATIONSHIP "R/1"
Set ERwinRelation = ERwinDatabase.CreateRelation("R/1", "Отдел", "Сотрудник")
Set ERwinField = ERwinRelation.CreateField("Номер отдела")
ERwinField.ForeignName = "Номер отдела"
ERwinRelation.Fields.Append ERwinField
ERwinDatabase.Relations.Append ERwinRelation

ERwinDatabase.Close
ERwinWorkspace.Close
'  Terminating Access Basic DAO Session...
```

Выбор клиента



Задание клиента



SQL-server генерация БД

```
CREATE TABLE Отдел (
     Номер_отдела         varchar(20) NOT NULL,
     Название_отдела      varchar(20) NULL
)
go


ALTER TABLE Отдел
     ADD PRIMARY KEY (Номер_отдела)
go
```

и т.д. 2 страницы текста

Как сгенерировать структуру данных из ERWIN для MS ACCESS

1. Tasks/ForwardEngineer…, потом кнопку Preview для просмотра и последующего сохранения или Report для сохранения, потом сохранить текст в файле, например, с именем "1.txt" (имя можно взять любое).
2. Создать базу MS ACCESS для запуска VBA, например "ERWIN.mdb". В этой базе создать модуль "Module1"
3. В модуле "Module1" написать текст процедуры

```
Sub m1()
Dim sERwinDatabase As String
sERwinDatabase = "ПРОБНЫЕ БАЗЫ\ПРОБА.mdb"        ' имя базы данных, куда будут генерироваться
                                                  ' таблицы и связи
Dim DB_DATETIME As Integer
DB_DATETIME = dbDate          ' устраняет несоответствие обозначений типов ERwin и
                              ' MS  ACCESS
' Вставить ниже содержимое файла "1.txt"
…
End Sub
```

Указать конкретное имя базы, в которой будут генерироваться таблицы и связи (в примере - "ПРОБА.mdb").

4. Вставить файл "1.txt" в указанное место
5. Откомпилировать проект. Исправить возможные ошибки! (несоответствие типов, имен и др.)
6. Запустить процедуру m1(). Будут сгенерированы таблицы и связи в базе "ПРОБА.mdb"

Примечание. При повторном запуске процедуры, когда таблицы уже созданы первым запуском, возникают ошибки.