Лабораторная работа №5. Применение доменов при разработке модели данных

ЦЕЛЬ РАБОТЫ: Ознакомиться с понятием логического домена, особенностями использования доменов при моделировании данных, научиться разрабатывать новые домены, а также познакомиться с особенностями использования отношений «многие-ко-многим» и рекурсивных связей и применить полученные знания для разработки модели базы данных и реализации ее для СУБД Microsoft Access.

Использование доменов в ER*win*

В издательских программах Вы можете быстро присвоить абзацу набор характеристик для форматирования, присвоив ему именованный *Style*. В ER*win* имеется аналогичная возможность, помогающая экономить время, - *домен.* Он позволяет Вам сохранить набор характеристик колонки вместе под одним именем.

При использовании ER*win,* домен может включать в себя одну или более характеристик колонки, ориентированных на СУБД, таких как тип данных, режим нулевых значений, значение по умолчанию и правило валидации. Как правило, использование доменов ускоряет процесс проектирования базы данных и упрощает работу с моделью данных. Вместо того, чтобы задавать ограничения для каждой колонки в отдельности, Вы можете создать домен и использовать его для задания нескольких характеристик одновременно. Если после этого понадобится изменить характеристику колонки, то можно просто изменить домен, и все связанные с ним колонки будут автоматически изменены.

Например, если модель данных содержит несколько разных атрибутов с номерами телефонов (например, *home-phone, business-phone* и т.д.), Вы можете создать домен под именем «AREA CODE», в котором имя колонки будет определено как *area_code*, а тип и длина данных - как CHAR(3). Кроме того, Вы можете связать с доменом список допустимых AREA CODE в правиле валидации (201, 202, 203, 204, и т.д.), и присвоить местный код в качестве значения по умолчанию (609). Наконец, Вы можете задать формат изображения, при котором код местности заключается в апострофы. Когда Вы связываете домен «AREA CODE» с колонкой Вашей базы данных, которая содержит информацию о телефонных кодах, она автоматически наследует весь набор характеристик колонки, который задан в домене.

В ER*win* домены модели организованы в иерархически упорядоченную систему, называемую *словарем доменов*. Для управления параметрами доменов в ER*win* применяется

специализированный редактор – редактор доменов (*Domain Dictionay Edtior*). Этот редактор позволяет задавать имя домена, родительский домен, имя колонки, тип данных, режим нулевых значений, значение по умолчанию, правило валидации и другие характеристики, которые можно скомбинировать и сохранить в домене. Режимы и возможности, доступные в редакторе доменов, различаются в зависимости от того, насколько выбранная СУБД поддерживает домены.

В редакторе доменов (см. рис. 1) на экран выводятся имя домена, тип данных и определение для всех доменов из списка, который находится в левой части окна-диалога. Когда Вы выделяете домен в этом списке, ER*win* сразу же показывает в остальных окнах редактора все значения характеристик колонки, связанные с выбранным доменом. Чтобы просмотреть характеристики колонки, связанные с другим доменом, просто выделите имя домена в левом списке, щелкнув по нему.

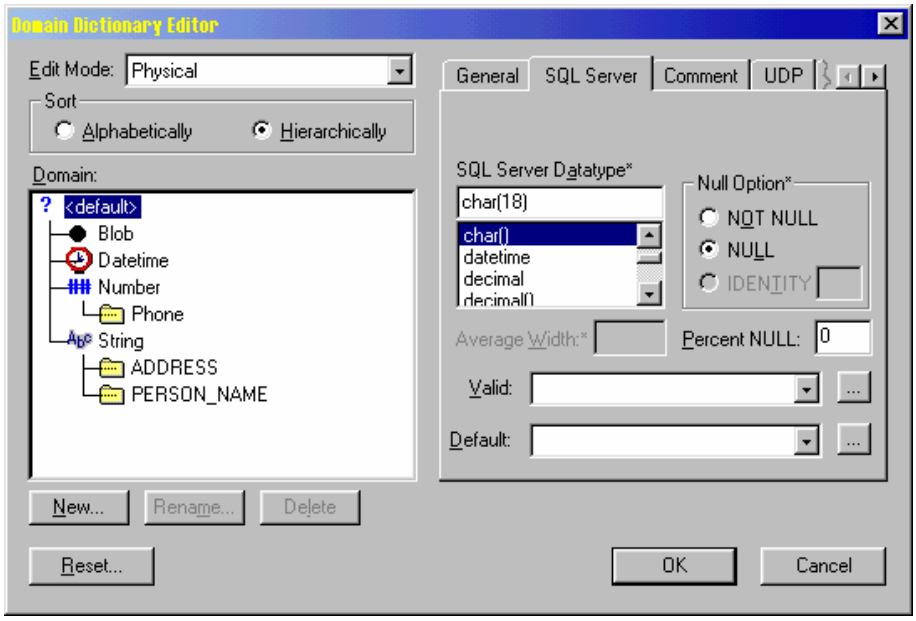

Рис. 1. Окно редактора словаря доменов ER*win*

Новый домен можно создать только из уже существующего домена. Когда Вы создаете новый домен, то он автоматически наследует все характеристики, присвоенные его родительскому домену. Чтобы создать домен, выберите домен в списке *Domain* и выберите родительский домен в списке *Inheritance Hierarchy*. Щелкнув кнопку *New* Вы в открывшемся окне вводите логическое и физическое имя для домена, после чего, нажав *OK* создается новый домен. Он наследует все свойства родительского домена, поэтому разработка нового домена заключается лишь в изменении некоторых характеристик, в частности изменить пиктограмму для нового домена.

Кнопки *Default...* и *Validation...* в редакторе доменов служат для входа в редакторы *Default* и *Validation Rule*, так что Вы можете перейти в определенный редактор, задать новую характеристику колонки, например значение по умолчанию, а затем вернуться в редактор доменов и присвоить ее существующему домену, не обращаясь при этом к главному меню.

Кнопка *Reset...* открывает окно-диалог, в котором Вы можете восстановить значения домена по умолчанию для одной или более характеристик колонки. Например, если Вы изменили режим нулевых значений для домена <*default*> с *NULL* на *NOT NULL*, то Вы можете использовать кнопку *Reset* для восстановления начального значения *NULL*.

Изначально в ER*win* определены четыре домена (*String, Number, Datetime, Memo*) и специальный домен <*default*>. Последний домен, в отличии от остальных, служит прототипом для создания новых атрибутов и не должен оставаться в разработанной модели.

Домен *String* служит для описания атрибутов, чьими значениями будут являться строки символов (например, имя человека). Принципиально то, что атрибуты такого домена (а, следовательно, и всех его потомков) ограничены по своей длине (обычно максимальной длиной атрибутов этого домена является 255 символов). На физическом уровне этот домен обычно реализуется через типы данных *CHAR* и *VARCHAR*.

Домен *Number* используется для описания атрибутов, чьими значениями являются численные величины. На практике это могут целые, дробные, десятичные и пр. значения. Этот домен также может применяться для хранения некоторых временных характеристик (например, номер года). Для домена *Number* на физическом уровне используется большое количество типов данных, но наиболее часто это *NUMBER*, *NUMERIC*, *INT* и т.п.

Домен *Datetime* введен специально для хранения временных величин (даты и время). Использование этого домена предпочтительнее домена *String* ввиду того, что на физическом уровне он реализуется более эффективно и предлагает дополнительные возможности для поиска информации (например, по дням недели). На физическом уровне для реализации домена применяются такие типы данных, как *DATE*, *TIME*, *DATETIME* и др.

Домен *BLOB* (*Binary Large Object*) предназначен для описания тех домен, которые хранят значения произвольной (или очень большой) длины. Это могут быть текстовые или бинарные данные, изображения, звуки и т.п. В отличии от других доменов домен BLOB отрицает использование индексов на физическом уровне, поэтому атрибуты этого домена обычно выполняют роль описательных атрибутов (например, комментарии).

Домен <default>

ERwin поставляется с предопределенным доменом, имя которого <*default*>, который автоматически задает характеристики для новых колонок. Эти характеристики, присваиваемые новой колонке доменом *default*, изначально основываются на значениях типов данных и режима нулевых значений, присваиваемых по умолчанию, которые задаются в редакторе *Target Server*. Кроме таких характеристик, как тип данных по умолчанию и режим нулевых значений, получаемых из редактора *Target Server*, домен *default* использует макрокоманду *%AttName* для присваивания колонке имени логического атрибута в качестве имени колонки.

Как и в случае любого другого домена, Вы можете изменить в редакторе доменов любые характеристики, присвоенные домену *default*, за исключением его имени.

Исключение связей “многие-ко-многим”

С помощью инструмент “неопределенная связь” можно создавать связи “многие-ко-многим” между двумя сущностями. Отношение “многие-ко-многим” означает, что каждый экземпляр первой сущности может быть связан с несколькими экземплярами второй сущности и наоборот. Например, можно создать связь”многие-ко-многим” между сущностями CUSTOMER и MOVIE, что означает, что каждый клиент может взять напрокат много фильмов и каждый фильм можно дать напрокат многим клиентам. Однако, заметьте, что если бы эта связь “многие-ко-многим” была единственной связью между клиентами и фильмами, то было бы очень трудно отслеживать выдачу фильмов клиентам. Как проследить, какие фильмы брал напрокат какой-то конкретный клиент? Предположите, что у Вас есть две копии фильма и обе даны напрокат двум разным клиентам. Где Вы запишете, какую копию взял каждый из клиентов? Как видно, связь “многие-ко-многим” не позволяет решить многие проблемы проектирования базы данных.

Для того, чтобы избежать конфликтов, создаваемых связями “многие-ко-многим”, проектировщики баз данных часто добавляют в модель данных промежуточные сущности, которые служат “посредниками” между двумя сущностями, между которыми имеется неопределенная связь. Добавляя промежуточную сущность между двумя сущностями, состоящими в связи “многие-ко-многим”, можно превратить связь в последовательность связей “один-ко-многим” и использовать эти связи для того, чтобы яснее выразить, как работает модель данных.

Для того, чтобы улучшить проект базы данных, необходимо искать способы избавления от связей “многие-ко-многим”, которыми обычно используются на более ранних

стадиях процесса проектирования, и заменять их промежуточными сущностями-”посредниками”, между которыми существуют связи “один-ко-многим”

Процесс избавления от связей «многие-ко-многим» иногда называют *разрешением связи «многие-ко-многим»*. Для разрешения связи «многие-ко-многим» применяется пункт *Resolve many-to-many* контекстного меню этой связи. После применения этой команды связь «многие-ко-многим» автоматически распадется на две связи «один-ко-многим», связанные вспомогательной сущностью (см. рис. 2).

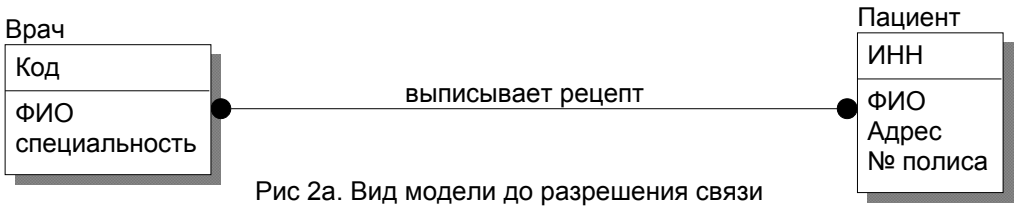

Рис 2а. Вид модели до разрешения связи "многие-ко-многим"

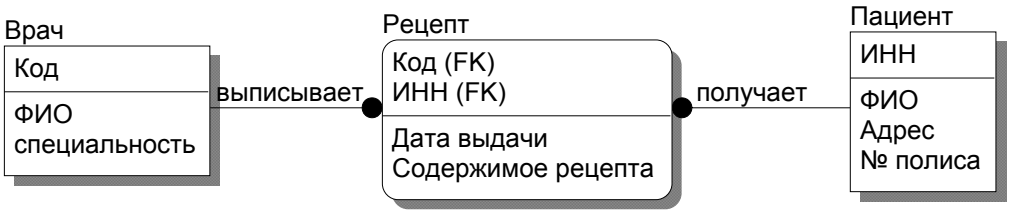

Рис 2б. Вид модели после разрешения связи и добавления атрибутов к ней

Задание рекурсивных связей

Рекурсивная связь - такая связь, при которой одна и та же сущность является и родительской, и дочерней. Такая ситуация часто встречается в “реальном мире” (например, руководители управляют руководителями). Существует два типа рекурсии, оба распространены и оба отражены в ER*win*. Первый тип называется *“иерархической рекурсией”* (другое название - однотабличная рекурсия или “петля”), и задает иерархию связей между родительской и дочерней сущностью, при которой родитель может породить любой количество дочерних сущностей, но дочерняя сущность может иметь только одного родителя. Второй тип - *сетевая рекурсия* (другое название - двухтабличная рекурсия), она задает “сеть” или “паутину” отношений между родительскими и дочерними сущностями, когда родитель может иметь любое количество дочерних сущностей и дочерняя сущность может иметь любое количество родителей. В обоих случаях все связи представляют собой пары первичных ключей в одной и той же таблице, но каждый из двух случаев имеет свое собственное значение.

Сетевая рекурсия - это такая ситуация, когда сущность находится сама с собой в связи “многие-ко-многим”. Например, поскольку компания может одновременно быть владельцем

и дочерней компанией по отношению к нескольким другим компаниям, то может существовать связь “многие-ко-многим” между сущностью COMPANY и этой же сущностью. Когда возникает проблема рекурсии “многие-ко-многим”, то есть сетевой рекурсии, Вы можете прояснить ситуацию, создав промежуточную сущность и превращая связь “многие-ко-многим” в две связи “один-ко-многим”, что описывается ниже.

В диаграмме ER*win* рекурсивная связь, описывающая тот факт, что одна компания может владеть другой, показан на обеих конструкциях, одна из которых представляет отношения единоличного владения (один родитель), а другая - конгломерат (множество родителей). В обоих случаях необходимо присвоить имена ролей мигрировавшим внешним ключам для того, чтобы передать значение рекурсивной связи.

Рекурсивные связи должны быть неидентифицирующими. Почему? Легче будет ответить, по какой причине им не имеет смысла быть идентифицирующими. В идентифицирующей связи первичный ключ родительской сущности становится подмножеством первичного ключа дочерней, и ключевые атрибуты никогда не могут принимать значение NULL. Подумав об этом и нарисовав простую таблицу экземпляров на основе примера из “реального мира”, Вы сможете убедить себя, что идентифицирующая связь не имела бы смысла, потому что она бы утверждала, что некоторая сущность является своим собственным родителем. Мигрирующий ключ мигрировал бы бесконечное число раз. Пример иерархической рекурсии приведен на рис. 3.

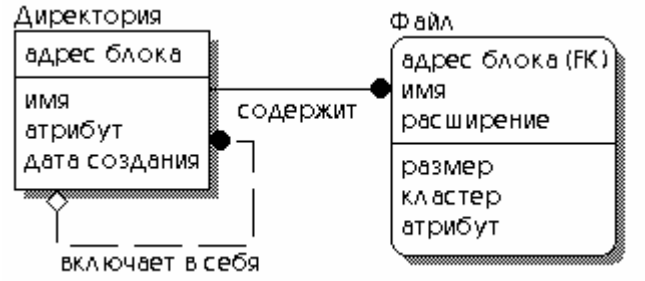

Рис. 3. Использование рекурсивных связей в модели.

Задание на лабораторную работу

1. Внимательно ознакомиться с материалами введения.
2. Разработать модель данных, позволяющую решить следующую задачу. Пусть имеется книжный магазин, торгующий книгами. Магазин хочет внедрить автоматизированную систему поиска литературы по каталогу, которая позволяла бы найти книгу, основываясь на ее реквизитах – названии, автору (авторам) книги, дате и месту издания, жанру, цене. Всю эту информацию предполагается хранить в базе данных, но кроме этого магазину требуется хранить и другую информацию о книгах, а именно: внутренний код издания,

количество имеющихся экземпляров, аннотация к книге, стране издания книги, стране проживания авторов книги, их биографию, адрес издательства, выпустившего книгу.

3. Учесть в разработанной модели, что название книги, а также ряд других параметров могут иметь очень длинные названия (как минимум 200 символов). Для этого ввести в модели домен longString, основанный на string и назначить его тип данных VARCHAR(200). Назначить этот домен всем необходимым атрибутам. Для остальных атрибутов проверить выбор домен и обосновать выбор при задании из определений.
4. Если в модели будут присутствовать отношения «многие-ко-многим», то их необходимо разрешить и дать им соответствующее название и определение.
5. Провести физическое моделирование логической модели и сгенерировать базу данных для СУБД Microsoft Access с использованием интерфейса ODBC.
6. Сохранить разработанную модель в свою папку под именем **task5-1.er1**.

Контрольные вопросы

1. Каковы сходные и различные черты домена и типа данных? Что из них используется на логическом, а что на физическом уровне?
2. В чем различие между доменами *string* и *blob*? Что у них общего?
3. Как вручную составить связь «многие-ко-многим» не прибегая к инструменту ER*win*?
4. Какие правила ссылочной целостности можно назначить для отношения «многие-ко-многим» ?
5. Какие типы рекурсивных связей бывают? Каковы особенности каждой разновидности?