Министерство образования Республики Беларусь

Учреждение образования
«Гомельский государственный технический
университет имени П. О. Сухого»

Институт повышения квалификации
и переподготовки кадров

Кафедра «Информатика»

Н. В. Водополова, Е. Г. Стародубцев

ОРГАНИЗАЦИЯ И ПРОЕКТИРОВАНИЕ БАЗ ДАННЫХ

ПОСОБИЕ
по одноименному курсу
для слушателей специальности 1-40 01 73
«Программное обеспечение информационных систем»
заочной формы обучения

Гомель 2012

УДК 681.3.06(075.8)
ББК 30.2-5-05я73
В62

Рекомендовано кафедрой «Информатика» ГГТУ им. П. О. Сухого
(*протокол № 12 от 06.06.2012 г.*)

Рецензент: доц. каф. «Инфорационные технологии» ГГТУ им. П. О. Сухого *В. И. Токочаков*

Водополова, Н. В.

В62 Организация и проектирование баз данных : пособие по одноим. курсу для слушателей специальности 1-40 01 73 «Программное обеспечение информационных систем» заоч. формы обучения / Н. В. Водополова, Е. Г. Стародубцев. – Гомель : ГГТУ им. П. О. Сухого, 2012. – 29 с. – Систем. требования: PC не ниже Intel Celeron 300 МГц ; 32 Mb RAM ; свободное место на HDD 16 Mb ; Windows 98 и выше ; Adobe Acrobat Reader. – Режим доступа: http://alis.gstu.by/StartEK/. – Загл. с титул. экрана.

Рассмотрены основы организации и проектирования баз данных, информационных систем на основе баз данных, использования систем управления базами данных (СУБД). Дана характеристика современных технологий представления и хранения структурированных данных, рассмотрены основные этапы проектирования реляционных баз данных на основе принципов нормализации реляционных таблиц. Приведены примеры построения информационных моделей и разработки баз данных (на примере СУБД MS Access) для различных предметных областей.

Для слушателей ИПК и ПК специальности 1-40 01 73 «Программное обеспечение информационных систем».

УДК 681.3.06(075.8)
ББК 30.2-5-05я73

1. ОСНОВЫ ПРЕДСТАВЛЕНИЯ И ХРАНЕНИЯ ДАННЫХ

1.1. Информационные системы и базы данных

Современные автоматизированные информационные системы (АИС) основаны на концепции интеграции данных, характеризуются большими объемами и сложной организацией хранимых данных, необходимостью удовлетворять разнообразным требованиям многочисленных пользователей. Разработано много программного обеспечения, позволяющего с помощью современных информационных технологий решать задачи, связанные с *формированием*, *хранением*, *обновлением* и *воспроизведением данных* самого разнообразного назначения и структуры в форме так называемых баз данных (БД).

Одно из возможных определений термина *«база данных»* – совокупность взаимосвязанных, хранящихся вместе данных, которая содержит сведения о различных *сущностях* одной *предметной области* (иногда в качестве синонима термина «сущность» используется термин «информационный объект»).

В определении понятия БД фигурируют 3 основных и очень важных особенности, учитываемых при проектировании БД:

1. БД – это совокупность *сущностей*.
2. БД – это совокупность *взаимосвязанных* сущностей.
3. БД – это средство *хранения* сведений о различных сущностях одной *предметной области*.

Под *предметной областью* понимают часть реального мира, подлежащую изучению с целью организации управления и автоматизации. Это может быть предприятие, вуз, служба управления городом и т.д. Предметная область представляется *множеством фрагментов*: например, предприятие – бухгалтерией, отделом кадров, планово-экономической службой и т.д. Каждый фрагмент предметной области, в свою очередь, характеризуется *множеством объектов* и *процессов*, использующих объекты, а также *множеством пользователей* с единым «взглядом» на предметную область. В частности, для бухгалтерии: объекты – разные финансовые документы (ценники, договоры, трудовые соглашения и др.); процессы – расчет заработной платы, материальный учет, банковские операции. Пользователи этого фрагмента предметной области – сотрудники бухгалтерии, работники финансовых органов.

Сложившийся в прошлом подход к проектированию АИС состоял в автоматизации отдельных процессов в рамках фрагментов предметной области, т.е. в создании множества *локальных приложений*. В силу значительной независимости приложений одни и те же данные многократно дублировались. При переходе от автоматизации отдельных процессов предметной области к созданию АИС требуется не только «взаимоувязка» приложений, но и качественно новый подход к организации данных. Это подход состоит в использовании единого хранилища данных – *базы данных.* При этом отдельные пользователи перестают быть владельцами тех или иных данных. Все данные накапливаются и хранятся *централизованно*. В памяти компьютера создается динамически обновляемая *модель предметной области*.

Для работы с БД используется *система управления базами данных* (СУБД) – комплекс языковых и программных средств, предназначенный для создания, ведения и совместного использования БД многими пользователями.

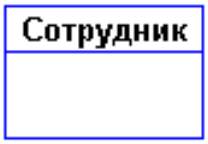
1.2. Проектирование структуры базы данных

Один из основных методов проектирования структуры БД – *метод семантического проектирования*, который опирается на смысл данных, которые будут храниться в БД. В качестве инструмента семантического моделирования применяются различные варианты так называемых *ER-диаграмм*. Такие диаграммы используют графические изображения сущностей (*Essences*) предметной области, их свойств (*атрибутов*), и взаимосвязей (*Relations*) между сущностями. Поэтому данный метод проектирования также называется: *метод ER-диаграмм* или *метод сущность-связь*.

Рассмотрим основные понятия ER-диаграмм (терминология и графические изображения сущностей и связей могут отличаться для разных вариантов метода ER-диаграмм).

Определение 1. *Сущность* – это класс однотипных объектов, информация о которых должна быть учтена в модели предметной области.

Каждая сущность должна иметь наименование, выраженное существительным в единственном числе. Например, «Сотрудник», «Студент», «Поставщик», «Накладная». Каждая сущность в модели изображается в виде прямоугольника с наименованием:



Определение 2. *Экземпляр сущности* – это конкретный представитель данной сущности. Например, представителем сущности «Сотрудник» может быть «Сотрудник Иванов».

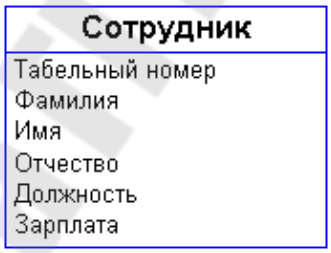
Экземпляры сущностей должны быть *различимы*, т.е. сущности должны иметь некоторые свойства, уникальные для каждого экземпляра этой сущности.

Определение 3. *Атрибут сущности* – это именованная характеристика, являющаяся некоторым свойством сущности.

Наименование атрибута должно быть выражено существительным в единственном числе (возможно, с характеризующими прилагательными).

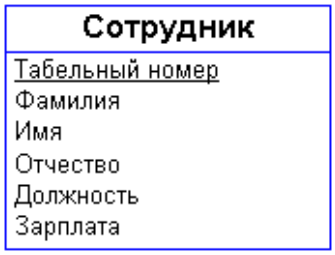
Примерами атрибутов сущности «Сотрудник» могут быть такие атрибуты как «Табельный номер», «Фамилия», «Имя», «Отчество», «Должность», «Зарплата» и т.п.

Атрибуты изображаются в пределах прямоугольника, определяющего сущность:



Определение 4. *Ключ сущности* – это *неизбыточный* набор атрибутов, значения которых в совокупности являются *уникальными* для каждого экземпляра сущности. Неизбыточность заключается в том, что при удалении любого атрибута из ключа нарушается его уникальность.

Сущность может иметь несколько различных ключей. Ключевые атрибуты могут разными способами изображаться на ER-диаграмме, например, выделяться подчеркиванием:



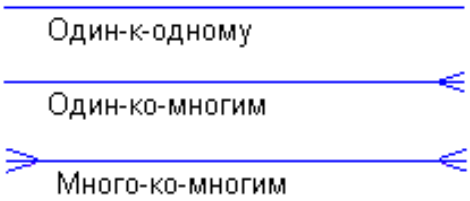
Определение 5. *Связь* – это некоторая ассоциация между двумя сущностями. Одна сущность может быть связана с другой сущностью или сама с собою.

Связи позволяют по одной сущности находить другие сущности, связанные с нею. Например, связи между сущностями могут выражаться следующими фразами – «СОТРУДНИК может иметь несколько ДЕТЕЙ», «каждый СОТРУДНИК обязан числиться ровно в одном ОТДЕЛЕ».

Графически связь изображается линией, соединяющей две сущности:

Каждая связь имеет два конца и одно или два наименования. Наименование обычно выражается в неопределенной глагольной форме: «иметь», «принадлежать» и т.п. Каждое из наименований относится к своему концу связи. Иногда наименования не пишутся ввиду их очевидности.

Каждая связь может иметь один из следующих типов связи:



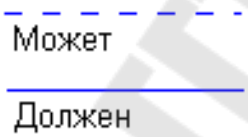
Связь типа *один-к-одному* (используется обозначение 1:1) означает, что один экземпляр первой сущности (левой) связан с одним экземпляром второй сущности (правой). Связь *один-к-одному*,

как правило, свидетельствует о том, что на самом деле имеется всего одна сущность, неправильно разделенная на две. На практике эта связь используется для разделения данных на общедоступные и конфиденциальные.

Связь типа *один-ко-многим* (1:∞ или 1:M) означает, что один экземпляр первой сущности (левой) связан с несколькими экземплярами второй сущности (правой). Это наиболее часто используемый тип связи. Левая сущность (со стороны «один») называется родительской, правая (со стороны «много») – дочерней. Пример такой связи представлен на рисунке определения 5.

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

Каждая связь может иметь одну из двух *модальностей связи*:



Модальность «может» означает, что экземпляр одной сущности может быть связан с одним или несколькими экземплярами другой сущности, а может быть и не связан ни с одним экземпляром.

Модальность «должен» означает, что экземпляр одной сущности должен быть связан не менее чем с одним экземпляром другой сущности.

Связь может иметь разную модальность с разных концов (как на рисунке определения 5).

Описанный графический синтаксис позволяет однозначно читать диаграммы, пользуясь следующей схемой построения фраз:

<Каждый экземпляр *СУЩНОСТИ 1*>
<МОДАЛЬНОСТЬ СВЯЗИ>
<НАИМЕНОВАНИЕ СВЯЗИ>
<ТИП СВЯЗИ>
<экземпляр *СУЩНОСТИ 2*>.

Каждая связь может быть прочитана как слева направо, так и справа налево. Связь на рисунке определения 5 читается так:

Слева направо: *Каждый сотрудник может иметь несколько детей*.

Справа налево: *Каждый ребенок обязан принадлежать ровно одному сотруднику*.

1.3. Пример разработки простой ER-модели

При разработке ER-моделей необходимо получить следующую информацию о предметной области:

1. Список сущностей предметной области.
2. Список атрибутов сущностей.
3. Описание взаимосвязей между сущностями.

ER-диаграммы удобны тем, что процесс выделения сущностей, атрибутов и связей является итерационным. Разработав первый приближенный вариант диаграмм, можно уточнить их, опрашивая экспертов предметной области. При этом документацией, в которой фиксируются результаты бесед, являются сами ER-диаграммы.

Задача: *разработать ER-модель для АИС некоторой оптовой торговой фирмы*.

Основные этапы решения задачи

1. *Изучение предметной области и процессов, происходящих в ней*, как правило, осуществляется путем опроса сотрудников фирмы, чтения документации, изучения формы заказов, накладных и т.п.

Например, в ходе беседы с менеджером по продажам, выяснилось, что он (менеджер) считает, что проектируемая АИС должна выполнять следующие действия:

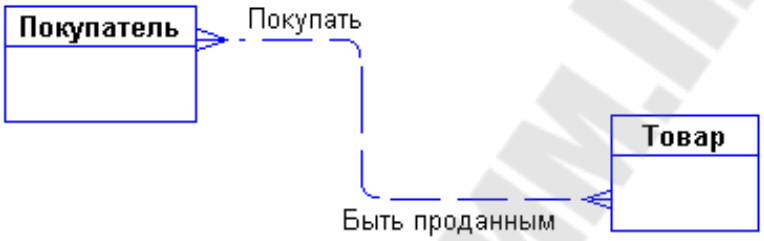
- *Хранить информацию о покупателях.*
- *Печатать накладные на отпущенные товары.*
- *Следить за наличием товаров на складе.*

2. *Определение сущностей.* Для этого необходимо выделить все существительные в предложениях, описывающих процессы, происходящие в изучаемой предметной области. Это и будут потенциальные кандидаты на сущности и их атрибуты. Проанализируем их (непонятные термины будем выделять знаком вопроса):

- *Покупатель* – кандидат на сущность.
- *Накладная* – кандидат на сущность.

- *Товар* – кандидат на сущность.
- ***(?)*** *Склад* – а вообще, сколько складов имеет фирма? Если несколько, то это будет кандидатом на новую сущность.
- ***(?)*** *Наличие товара* – это, скорее всего, атрибут, но атрибут какой сущности?

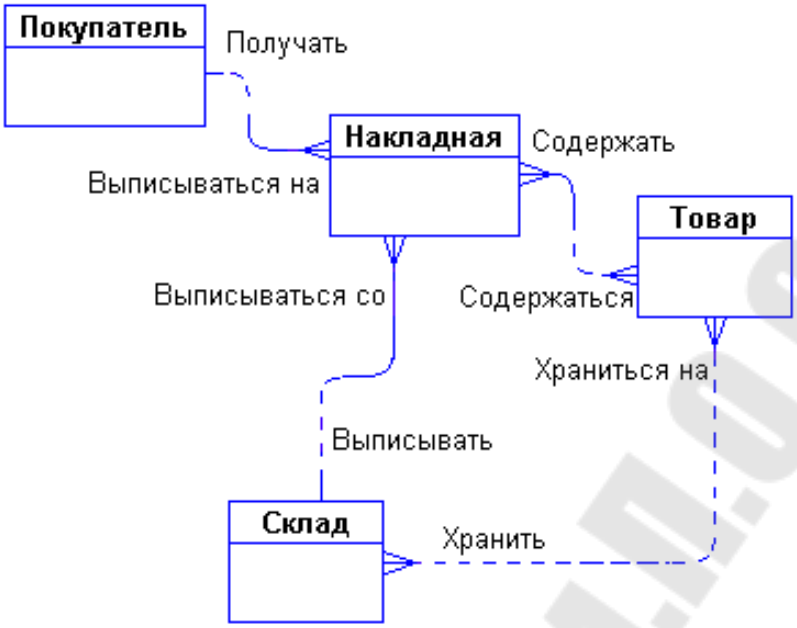
3. *Определение связей между сущностями.* Для рассматриваемого примера сразу возникает очевидная связь между сущностями: «*Покупатели* могут покупать много *Товаров*» и «*Товары* могут продаваться многим *Покупателям*». Первый вариант диаграммы выглядит так:



Задав дополнительные вопросы менеджеру, были выявлены новые данные о том, что:

- фирма имеет несколько складов, а каждый товар может: храниться на нескольких складах; быть проданным с любого склада;
- покупатели покупают товары, получая при этом накладные, в которые внесены данные о количестве и цене купленного товара;
- каждый покупатель может получить несколько накладных;
- каждая накладная выписывается на одного покупателя;
- каждая накладная содержит хотя бы один товар (не бывает пустых накладных);
- каждый товар, в свою очередь, может быть продан нескольким покупателям через несколько накладных;
- каждая накладная должна быть выписана с определенного склада, и с любого склада может быть выписано много накладных.

Учитывая новые сведения, диаграмма примет следующий вид:



4. *Определение атрибутов сущностей.* Беседуя с сотрудниками фирмы, были выяснены следующие обстоятельства:

- каждый покупатель является юридическим лицом и имеет наименование, адрес, банковские реквизиты;
- каждый товар имеет наименование, цену, а также характеризуется единицами измерения;
- каждая накладная имеет уникальный номер, дату выписки, список товаров с количествами и ценами, а также общую сумму накладной; накладная выписывается с определенного склада и на определенного покупателя;
- каждый склад имеет свое наименование.

Снова выпишем все существительные, которые будут потенциальными атрибутами, и проанализируем их:

- *Юридическое лицо* – т.к. фирма работает только с юридическими лицами (не работает с физическими лицами), то такой атрибут выделять нет смысла.
- *Наименование покупателя* – характеристика покупателя.
- *Адрес* – характеристика покупателя.
- *Банковские реквизиты* – характеристика покупателя.
- *Наименование товара* – характеристика товара.
- **(?)** *Цена товара* – похоже, что это характеристика товара. Отличается ли эта характеристика от цены в накладной?
- *Единица измерения* – характеристика товара.

- *Номер накладной* – уникальная характеристика накладной.
- *Дата накладной* – характеристика накладной.
- **(?)** *Список товаров в накладной* – список не может быть атрибутом. Вероятно, нужно выделить этот список в отдельную сущность.
- **(?)** *Количество товара в накладной* – это характеристика, но характеристика чего? Это характеристика не просто "товара", а "товара в накладной".
- **(?)** *Цена товара в накладной* – характеристика товара в накладной. Но цена товара уже встречалась выше – это одно и то же?
- *Сумма накладной* – характеристика накладной. Эта характеристика не является независимой. Сумма накладной равна сумме стоимостей всех товаров, входящих в накладную.
- *Наименование склада* – характеристика склада.

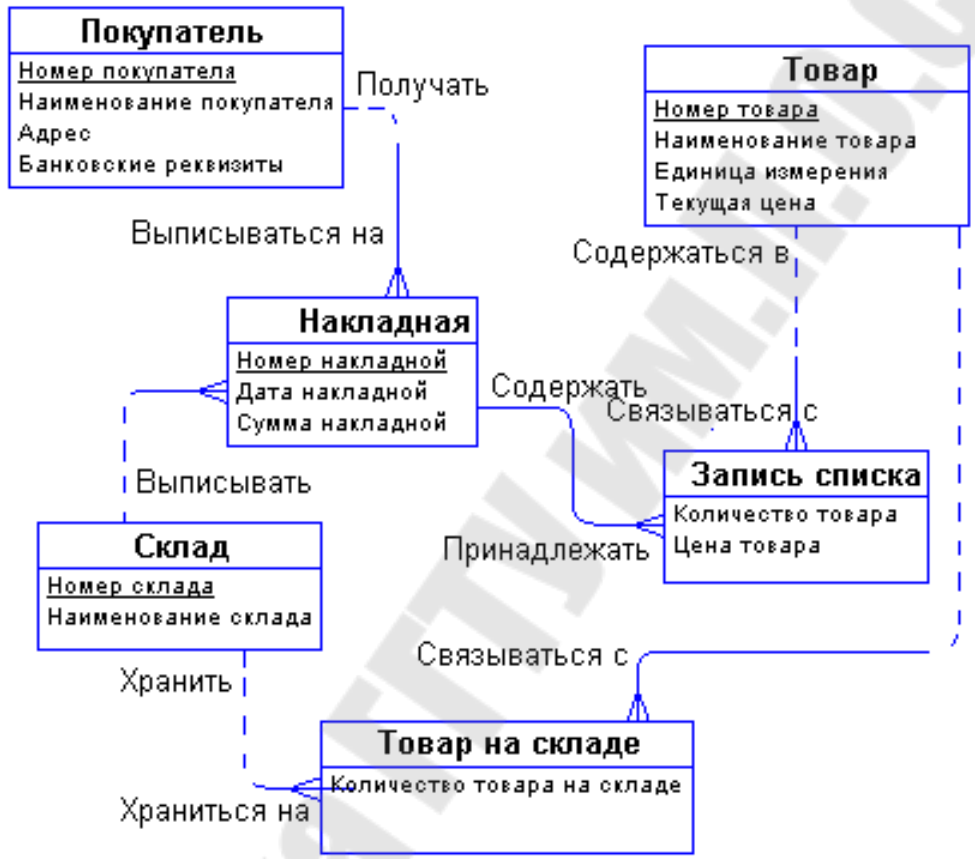
В ходе дополнительной беседы с менеджером удалось прояснить различные понятия цен. Оказалось, что каждый товар имеет некоторую текущую цену. Эта цена, по которой товар продается в данный момент. Естественно, что эта цена может меняться со временем. Цена одного и того же товара в разных накладных, выписанных в разное время, может быть различной. Таким образом, имеется *две цены* – цена товара в накладной и текущая цена товара.

С возникающим понятием «Список товаров в накладной» все довольно ясно. Сущности *Накладная* и *Товар* связаны друг с другом отношением типа *много-ко-многим*. Такая связь должна быть разделена на две связи типа *один-ко-многим.* Для этого требуется дополнительная сущность. Этой сущностью и будет сущность *Список товаров в накладной*. Связь ее с сущностями *Накладная* и *Товар* характеризуется следующими фразами: «Каждая накладная обязана иметь несколько записей из списка товаров в накладной», «Каждая запись из списка товаров в накладной обязана включаться ровно в одну накладную», «Каждый товар может включаться в несколько записей из списка товаров в накладной», «Каждая запись из списка товаров в накладной обязана быть связана ровно с одним товаром».

Атрибуты *Количество товара в накладной* и *Цена товара в накладной* являются атрибутами сущности *Список товаров в накладной*.

Точно также поступим со связью, соединяющей сущности *Склад* и *Товар*. Введем дополнительную сущность *Товар на складе*. Атрибутом этой сущности будет *Количество товара на складе*. Таким образом, товар будет числиться на любом складе и количество его на каждом складе будет свое.

В результате ER-диаграмма примет вид:



1.4. Модели данных

Модель предметной области поддерживается в памяти компьютера с помощью СУБД. В силу этого результат моделирования зависит не только от предметной области, но и от используемой СУБД, т.к. каждая СУБД предоставляет свой инструментарий для отображения предметной области. Этот инструментарий принято называть *моделью данных*.

Модель данных определяется тремя компонентами:

- допустимой организацией данных;
- ограничениями целостности (семантической);
- множеством операций, допустимых над объектами модели данных.

Допустимая организация данных определяется разнообразием и количеством типов объектов модели данных, ограничениями на структуру данных.

Ограничения целостности поддерживаются средствами, предусмотренными в модели данных для выражения ограничений на значения данных и ассоциации, которые (ограничения) характеризуют достоверные состояния БД.

Ряд ограничений целостности поддерживается моделью данных по умолчанию и распространяется на все типовые ситуации, возникновение которых возможно при внесении изменений в БД. Например, если между записями типа ГРУППА и СТУДЕНТ установлена иерархическая связь, то СУБД воспрепятствует удалению сведений о студенческой группе, если с ней связана хотя бы одна запись о студенте.

Другие ограничения целостности могут задаваться явно и также распространяться на множество однотипных ситуаций или на значения отдельных полей. Например, предположим, что при определении записи для какого-либо поля задано свойство уникальности значений, тогда СУБД воспрепятствует появлению в БД двух записей с одинаковым значением этого поля.

Множество операций определяет виды обработок, которым могут подвергаться объекты модели данных. Прежде всего, это – операции выборки данных и операции, изменяющие состояние БД.

Поддерживаемые СУБД модели данных традиционно разбиваются на: *сетевые*, *иерархические*, *реляционные* (имеются также другие модели данных). Эта классификация условна, т.к. каждая СУБД поддерживает свою оригинальную модель данных. И даже если модель данных относится к одной из выделенных разновидностей, она может содержать достаточно много отличительных особенностей. Наиболее распространенными моделями являются *реляционные модели данных* (РМД).

1.5. Реляционная модель данных

В реляционной модели данные организованы в виде двумерных таблиц (*реляционных таблиц*) или *отношений* (*relations*), а РМД некоторой предметной области представляет собой *набор отношений, изменяющихся во времени*. Совокупность отношений позволяет хранить данные об объектах предметной области и моделировать связи между ними.

Каждая реляционная таблица представляет собой двумерный массив и обладает следующими свойствами:

- *каждый элемент таблицы – один элемент данных*;
- *все столбцы в таблице однородные*, т.е. все элементы в столбце имеют одинаковый тип данных и длину;
- *каждый столбец имеет уникальное имя*;
- *порядок следования строк и столбцов может быть произвольным*.

Строки (записи) реляционной таблицы соответствуют *кортежам*, а столбцы (поля) – *атрибутам* с наборами *значений атрибутов*. Поле, каждое значение которого однозначно определяет (идентифицирует) соответствующую запись, называется *простым ключом* (ключевым полем). Если записи идентифицируются значениями нескольких полей, то таблица имеет *составной ключ*.

Элементы РМД и формы их представления в СУБД приведены в таблице.

Элементы реляционной модели

Элемент реляционной модели	Форма представления
Отношение	Таблица
Схема отношения	Строка заголовков столбцов таблицы (заголовок таблицы)
Кортеж	Строка таблицы
Сущность	Описание свойств объекта
Атрибут	Заголовок столбца таблицы
Домен	Множество допустимых значений атрибута
Значение атрибута	Значение поля в записи
Первичный ключ	Один или несколько атрибутов
Тип данных	Тип значений элементов таблицы

На рис. 1 дан пример представления отношения СТУДЕНТ в РМД.

Рисунок 1 – Представление отношения СТУДЕНТ в РМД

В РМД также используются следующие основные термины.

Сущность – объект любой природы, данные о котором хранятся в БД. Данные о сущности хранятся в отношении.

Атрибуты – свойства, характеризующие сущность. В структуре таблицы каждый атрибут именуется и ему соответствует заголовок некоторого столбца таблицы.

Математически отношение можно описать следующим образом. Пусть даны n множеств $D_1, D_2, ..., D_n$, тогда отношение R есть множество упорядоченных *кортежей* $<d_1, d_2, ..., d_n>$, где $d_k \in D_k$, d_k – *атрибут*, а D_k – *домен* отношения R.

В общем случае порядок кортежей в отношении, как и в любом множестве, не определен. Однако в реляционных СУБД для удобства кортежи все же упорядочивают. Чаще всего для этого выбирают некоторый атрибут, по которому СУБД автоматически сортирует кортежи по возрастанию или убыванию. Если пользователь не назначает атрибута упорядочения, система автоматически присваивает номер кортежам в порядке их ввода. Формально, если переставить атрибуты в отношении (переставить столбцы в таблице, описывающей отношение), то получается новое отношение. Однако в реляционных БД *перестановка атрибутов не приводит к образованию нового отношения*.

Домен – множество всех возможных значений определенного атрибута отношения. Например, отношение СТУДЕНТ (рис. 1) включает 5 доменов:

- *домен 1* содержит номера всех студентов (например, номера личных дел, хранящихся в деканате);

– *домен 2* – фамилии студентов;
– *домен 3* – имена студентов;
– *домен 4* – сокращенные наименования факультетов;
– *домен 5* – даты рождения студентов.

Каждый домен образует значения одного типа данных, например, числовые или символьные. Отношение СТУДЕНТ содержит 3 кортежа. Кортеж рассматриваемого отношения состоит из 5 элементов, каждый из которых выбирается из соответствующего домена. Каждому кортежу соответствует строка таблицы (рис. 1).

Схема отношения (*заголовок отношения*) – список имен атрибутов. Например, для приведенного примера схема отношения имеет вид

СТУДЕНТ (*Номер, Фамилия, Имя, Факультет, Дата_рождения*).

Первичным ключом (*ключом отношения, ключевым атрибутом*) называется атрибут отношения, идентифицирующий каждый из его кортежей. Например, в отношении СТУДЕНТ ключевым является атрибут *Номер* (если номера личных дел студентов не совпадают).

Ключ может быть *составным* (*сложным*), то есть состоять из нескольких атрибутов. Каждое отношение обязательно имеет комбинацию атрибутов, которая может служить ключом. Ее существование гарантируется тем, что отношение – это множество, которое не содержит одинаковых элементов – кортежей. Если в отношении нет повторяющихся кортежей, то, по крайней мере, вся совокупность атрибутов обладает свойством однозначной идентификации кортежей отношения. Во многих СУБД допускается создавать отношения (таблицы), не определяя ключи.

Отношение может иметь несколько комбинаций атрибутов, каждая из которых идентифицирует все кортежи отношения. Все эти комбинации атрибутов являются *возможными ключами* отношения. Любой из возможных ключей может быть выбран как первичный. Если выбранный первичный ключ состоит из минимально необходимого набора атрибутов, говорят, что он является *не избыточным*.

Ключи обычно используют для достижения следующих целей: 1) исключения дублирования значений в ключевых атрибутах (остальные атрибуты в расчет не принимаются); 2) упорядочения

кортежей (возможны различные виды упорядочения); 3) ускорения работы с кортежами отношения; 4) организации связывания таблиц.

Пусть в отношении R1 имеется *неключевой* атрибут A, значения которого являются значениями ключевого атрибута B другого отношения R2. Тогда говорят, что атрибут A отношения R1 есть *внешний ключ*. С помощью внешних ключей устанавливаются связи между отношениями. Например, имеются отношения: СТУДЕНТ (рис. 1) и ПРЕДМЕТ(*Название*, *Семестр*, *Часы*), связанные отношением СТУДЕНТ_ПРЕДМЕТ(*Номер*, *Название*, *Оценка*) (рис. 2).

Рисунок 2 – Пример связи отношений

Атрибуты *Номер* и *Название* отношений СТУДЕНТ и ПРЕДМЕТ – первичные ключи (обозначены ПК на рис. 2), аналогичные атрибуты отношения СТУДЕНТ_ПРЕДМЕТ являются внешними ключами (обозначены ВК).

Отношения и соответствующие им реляционные таблицы могут быть связаны между собой с помощью ранее рассмотренных типов связей: один–к–одному (1:1); один–ко–многим (1:∞ или 1:M); многие–ко–многим (∞:∞ или M:M).

Связь 1:1 предполагает, что в каждый момент времени одному экземпляру сущности A соответствует не более одного экземпляра сущности B и наоборот. При этом каждая запись (строка) в таблице, характеризующей объект A, соответствует только одной записи (строке) в таблице, характеризующей объект B. Пример: связь между сущностями **Студент** и **Посещения занятий**, если каждый студент имеет определённый уникальный набор посещенных занятий по разным предметам за семестр.

При связи 1:∞ одному экземпляру сущности A соответствует 0, 1 или более экземпляров объекта B, но каждый экземпляр сущности B связан не более чем с одним экземпляром сущности A. При этом

каждая запись в таблице, описывающей объект А, соответствует многим записям в таблице, описывающей объект В. Примером связи 1:∞ служит связь между сущностями **Заработная плата** и **Сотрудники**, когда установленный размер заработной платы может повторяться многократно для различных сотрудников.

Связь ∞:∞ предполагает, что в каждый момент времени одному экземпляру сущности А соответствует 0, 1 или более экземпляров сущности В и наоборот. Пример такой связи – связь между сущностями **Студент** и **Преподаватель**, когда один студент обучается у многих преподавателей, а один преподаватель обучает многих студентов. Связь **∞:∞** не реализуется в реляционных БД непосредственно (для двух таблиц). Для организации такой связи используется промежуточная (третья) таблица-связка для «замены» одной связи **∞:∞** на «сумму» двух связей 1:∞ и ∞:1.

Связи между реляционными таблицами устанавливаются *при помощи совпадающих значений связующих полей*. Например, таблицы, соответствующие отношениям СТУДЕНТ и ПРЕДМЕТ (рис. 2), связаны по полям *Номер* и *Название* с таблицей, характеризующей отношение СТУДЕНТ_ПРЕДМЕТ. При этом реализуются две связи типа 1:∞ (при наличии одинаковых значений номеров и названий предметов в соответствующих парах таблиц), так как одному студенту соответствует набор оценок по разным предметам, а одному предмету – группа студентов.

1.6. Нормализация отношений

Одни и те же данные могут группироваться в реляционные таблицы различными способами. Группировка полей в таблицах должна быть рациональной, что означает сведение к минимуму дублирования данных и упрощение процедуры их обработки и обновления.

Для решения этих вопросов автором и разработчиком концепции реляционной модели данных Е. Коддом был разработан аппарат, называемый *нормализацией отношений* (*таблиц*). И хотя идеи нормализации сформированы в терминологии реляционной модели данных, они применимы и для других моделей данных.

Нормализация отношений – это формальный аппарат ограничений на формирование таблиц, который позволяет устранить дублирование данных, обеспечивает непротиворечивость хранимых данных, уменьшает трудозатраты на ведение БД.

При практическом проектировании реляционных БД обычно используют *три нормальные формы таблиц*. Самая совершенная – третья. Существует алгоритм, позволяющий любое отношение (таблицу) преобразовать к третьей нормальной форме. В процессе таких преобразований могут выделяться новые отношения.

Первая нормальная форма. Отношение называется нормализованным к первой нормальной форме (1НФ), если все его атрибуты *простые* (далее неделимы).

Простым является атрибут, если его значения *атомарны*, т.е. неделимы.

Сложный атрибут может иметь значение, представляющее собой конкатенацию (объединение) нескольких значений одного или разных доменов. Аналогами сложного атрибута может быть вектор, набор или массив значений.

Согласно 1НФ, в каждой позиции (клетке таблицы), определяемой строкой и столбцом, должна быть только одна величина, *но не массив или список величин*.

Ненормализованное отношение легко сделать нормализованным. Такое преобразование может привести к увеличению *мощности отношения* (количества строк соответствующей реляционной таблицы) и изменению ключа отношения (таблицы).

Рассмотрим пример таблицы, **нарушающей** требования 1НФ (звездочкой будем обозначать ключевое поле):

Заказы1

КодЗаказа*	**КодПокупателя**	**СодержаниеЗаказа**
1	4	5 молотков, 3 дрели, 6 ключей
2	23	1 молоток, 4 ключа
3	15	2 ключа

Таблица Заказы1 служит для хранения данных о заказах на складе инструментов. Эта таблица имеет простой ключ, состоящий из одного поля *КодЗаказа*. Данные в столбце *СодержаниеЗаказа* содержат ***списки величин*** и ***не являются атомарными***. Так как в столбце *СодержаниеЗаказа* находится слишком много информации, то получить упорядоченную информацию из этой таблицы будет

трудно. Например, трудоёмким окажется составление отчёта о суммарных закупках инструментов различных видов. Таблицу Заказы1 можно преобразовать (*без потери данных*) в таблицу Заказы2, удовлетворяющую требованиям 1НФ:

Заказы2

КодЗаказа*	КодСодержанияЗаказа	КодПокупател	Количеств	СодержаниеЗаказ
1	1	4	5	молоток
1	2	4	3	дрель
1	3	4	6	ключ
2	1	23	1	молоток
2	3	23	4	ключ
3	3	15	2	ключ

Эта таблица имеет составной ключ из двух полей: *КодЗаказа* и *КодСодержанияЗаказа*. Данные всех столбцов таблицы Заказы2 являются атомарными.

Прежде чем перейти ко второй нормальной форме отношений, рассмотрим следующие основные понятия, используемые в аппарате нормализации отношений.

Функциональная зависимость атрибутов – это зависимость, при которой определенному значению ключа соответствует *только одно значение описательного атрибута*.

Например, табельный номер функционально зависит от ФИО сотрудника, и одновременно ФИО сотрудника функционально зависит от табельного номера. Действительно, каждому сотруднику ставится в соответствие только один табельный номер, а конкретный табельный номер в определенный момент времени соответствует только одному сотруднику.

Полная функциональная зависимость имеет место в том случае, когда неключевой (не входящий в ключ) атрибут зависит от всего составного ключа, но не находится в функциональной зависимости ни от какой части составного ключа.

Например, отношение Сотрудники находится в 1НФ и имеет составной ключ *Табельный номер родителя – Имя ребенка*:

Сотрудники

Табельный номер родителя	Имя ребенка	Возраст	ФИО	Оклад
211	Саша	10	Иванов И.И.	120
211	Женя	7	Иванов И.И.	120
211	Вася	5	Иванов И.И.	120
368	Таня	6	Петров П.П.	200
299	Петя	2	Сидоров С.С.	180

Атрибуты ФИО и Оклад не находятся в полной функциональной зависимости от ключа отношения, т.к. они функционально зависят только от части ключа – столбца *Табельный номер родителя*. Рассмотрим атрибут Оклад. Нерационально хранить эту информацию в отношении, каждый кортеж которого соответствует ребенку сотрудника. Во-первых, в этом случае имеет место дублирование информации, поскольку у сотрудника может быть несколько детей, и для каждого из них будут храниться сведения об окладе родителя. Кроме того, при изменении оклада понадобится корректировать не один, а несколько кортежей отношения (несколько строк таблицы). Во-вторых, проблема возникает с сотрудниками, у которых нет детей. Им не соответствует ни один кортеж отношения, а значит, БД не может содержать сведений о таких сотрудниках. Очевидно, что рассматриваемое отношение следует преобразовать.

Вторая нормальная форма. Отношение находится во второй нормальной форме (2НФ), если оно находится в 1НФ и каждый неключевой (*описательный*) атрибут *функционально полно* зависит от всего ключа (простого или составного).

Если таблица имеет простой первичный ключ, состоящий только из одного столбца, то она автоматически находится во 2НФ.

Если же первичный ключ составной, то таблица необязательно находится во 2НФ. В этом случае необходимо преобразовать таблицу или разделить ее на несколько таблиц так, чтобы первичный ключ однозначно определял значение в любом неключевом столбце. Для выполнения требований 2НФ приведенное выше отношение Сотрудники следует разделить на два отношения, которые можно назвать Дети и Родители, связанных по полям *Табельный номер родителя* и *Табельный номер* (имена связующих полей могут не совпадать):

Дети

Табельный номер родителя	Имя ребенка	Возраст
211	Саша	10
211	Женя	7
211	Вася	5
368	Таня	6
299	Петя	2

Родители

Табельный номер	ФИО	Оклад
211	Иванов И.И.	120
368	Петров П.П.	200
299	Сидоров С.С.	180

Таким образом, приведение таблицы к 2НФ позволяет *избежать повторения одних и тех же данных*, которое может иметь место в случае 1НФ, однако это может потребовать создания дополнительных отношений (таблиц).

Одну и ту же таблицу можно привести к 2НФ разными способами, не теряя данных.

Например, таблица Заказы2 не находится во 2НФ. Эта таблица имеет составной ключ, включающий 2 поля: *КодЗаказа* и *КодСодержанияЗаказа*. В то же время, неключевое поле *СодержаниеЗаказа* однозначно определяется только полем *КодСодержанияЗаказа*, т.е. *только одним* из двух полей, входящих в составной ключ. При заданной кодировке инструментов на складе для заполнения значения поля *СодержаниеЗаказа* достаточно знать только значение поля *КодСодержанияЗаказа* (1 – молоток, 2 – дрель и т. д.). Таким образом, для поля *СодержаниеЗаказа* отсутствует функциональная зависимость от *всего составного ключа* и таблица Заказы2 не находится во 2НФ. Таблицу Заказы2 можно преобразовать без потери данных в таблицу Заказы3, удовлетворяющую требованиям 2НФ:

Заказы3

КодЗаказа*	СодержаниеЗаказа*	КодПокупателя	Количество
1	молоток	4	5
1	дрель	4	3
1	ключ	4	6
2	молоток	23	1
2	ключ	23	4
3	ключ	15	2

У таблицы Заказы3 составной ключ содержит 2 поля: *КодЗаказа* и *СодержаниеЗаказа*, а неключевые или описательные поля *КодПокупателя* и *Количество* функционально зависят от *всего составного ключа*. Из данного примера также видно, как уменьшается повторение одинаковых данных при переходе от 1НФ ко 2НФ («исчезает» один столбец из двух, содержащих повторяющиеся данные в таблице Заказы2).

Третья нормальная форма. Отношение находится в третьей нормальной форме (3НФ), если оно находится во 2НФ, и каждый неключевой атрибут *нетранзитивно* зависит от первичного ключа.

Транзитивная зависимость атрибутов имеет место в том случае, если один из двух описательных атрибутов зависит от ключа, а другое описательное поле зависит от первого описательного поля. Таким образом, таблица соответствует 3НФ, если она соответствует 2НФ, и все неключевые атрибуты *взаимно независимы*.

Например, отношение

ДанныеСотрудников

Табельный номер	***ФИО***	***Оклад***	***Комната***	***Телефон***
211	Иванов И.И.	120	14	191
368	Петров П.П.	200	14	191
299	Сидоров С.С	180	21	623

находится во 2НФ, но не в 3НФ, т.к. атрибут *Телефон* зависит от номера комнаты, в которой работает сотрудник. Это отношение можно преобразовать в два отношения, связанных по полю *Комната*:

Сотрудники

Табельный номер	***ФИО***	***Оклад***	***Комната***
211	Иванов И.И.	120	14
368	Петров П.П.	200	14
299	Сидоров С.С.	180	21

Телефоны

Комната	***Телефон***
14	191
21	623

Транзитивные зависимости полей создают проблемы при *добавлении*, *обновлении* и *удалении* записей из таблицы. Например,

если в какой-то комнате поменялся номер телефона, то для отношения (таблицы) ДанныеСотрудников необходимо изменить столько записей, сколько сотрудников работает в этой комнате. В случае связанных отношений Сотрудники и Телефоны, удовлетворяющих 3НФ, нужно будет изменить *только одну запись* в таблице Телефоны.

Таблица Заказы3 также находится в 3НФ, т.к. неключевые поля *КодПокупателя* и *Количество* взаимно независимы, т.е. в этой таблице нет транзитивных зависимостей полей.

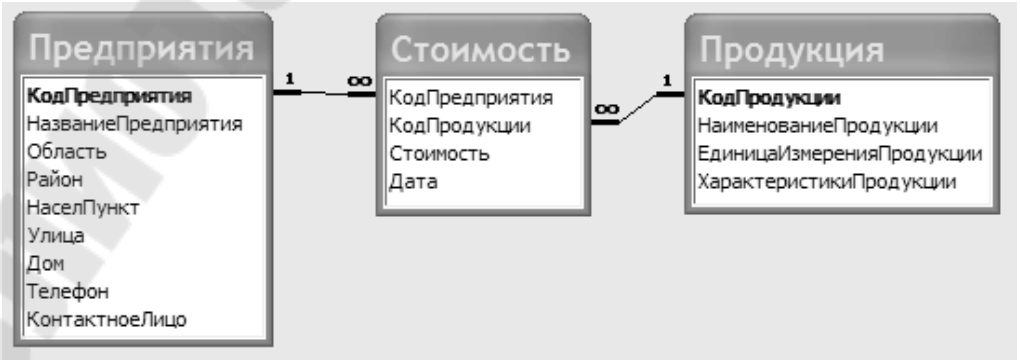
Таким образом, «плата» за нормализацию таблиц – добавление в таблицы новых полей, возможное увеличение количества таблиц в процессе нормализации. «Выгода» от нормализации – более простая и быстрая обработка данных; экономия памяти и машинного времени для хранения и обработки данных.

2. ПРИМЕРЫ ПРОЕКТИРОВАНИЯ БАЗ ДАННЫХ В СУБД MS ACCESS

Рассмотрим примеры проектирования реляционных БД с учетом требований нормализации средствами СУБД MS Access (ниже приведены окна создания схемы данных, вызываемые из основного меню MS Access командами **Сервис** / **Схема данных**).

Пример 1. Спроектировать БД «Каталог продукции производителей Республики Беларусь». БД должна содержать следующую информацию: сведения о продукции, сведения о предприятии-производителе, стоимость продукции на определенную дату.

Требуемая схема данных может иметь вид:



Между таблицами используются связи типа «один ко многим» (1:∞). Поля на стороне отношения «один» (на схеме данных выделяются полужирным шрифтом) являются первичными ключами таблиц. Первичные ключи *КодПредприятия*, *КодПродукции* в таблицах Предприятия, Продукция имеют тип данных *Счетчик*, а соответствующие поля на стороне отношения «многие» (внешние ключи) в таблице Стоимость имеют тип данных *Числовой* (при этом свойство *Размер поля* имеет значение *Длинное целое*). Остальные поля имеют типы данных *Текстовый*, *Числовой*, *Дата/время* в зависимости от заносимых данных. При этом задается ***минимально необходимое*** значение свойства *Размер поля* (например, нет необходимости для текстового поля *НазваниеПредприятия* задавать размер, больший 20-30 символов).

Отметим, что структуру реляционной БД, соответствующей требованиям нормализации, можно быстро создать, используя ***подход «справочных» и «оперативных» таблиц***. В данном примере к справочным таблицам (*справочникам*) относятся таблицы Предприятия, Продукция (содержат по одной строке для каждого предприятия или вида продукции соответственно; данные в этих таблицах слабо зависят от времени и изменяются относительно редко). *Оперативная таблица* Стоимость содержит много строк для каждого предприятия и вида продукции, ее данные сильно зависят от времени и быстро изменяются, так как стоимость продукции постоянно обновляется. Как правило, с течением времени объем данных оперативных таблиц становится существенно больше объема данных справочников. Между таблицами-справочниками и оперативными таблицами устанавливается связь 1:∞ (чаще всего используется в реляционных БД), для этого в таблицы нужно добавить соответствующие первичные и внешние ключи. В результате, разместив поля БД по справочным и оперативным таблицам (после анализа предметной области и выделения интересующих атрибутов отношений) и задав связи, получаем нормализованную БД, как правило, соответствующую 3НФ.

Пример 2. Разработать БД «Энергетик», в которой хранится и обрабатывается информация об энергетическом оборудовании предприятия (объединения предприятий). Для каждой единицы оборудования в БД должна храниться следующая информация:

1) инвентарный и серийный номера, марка, наименование;

2) данные о подразделении, где установлено оборудование: адрес (город, улица, дом, корпус); реквизиты главного энергетика;

3) организация – производитель оборудования (наименование, основные реквизиты, контактное лицо);

4) год выпуска, дата (год, месяц) ввода в эксплуатацию;

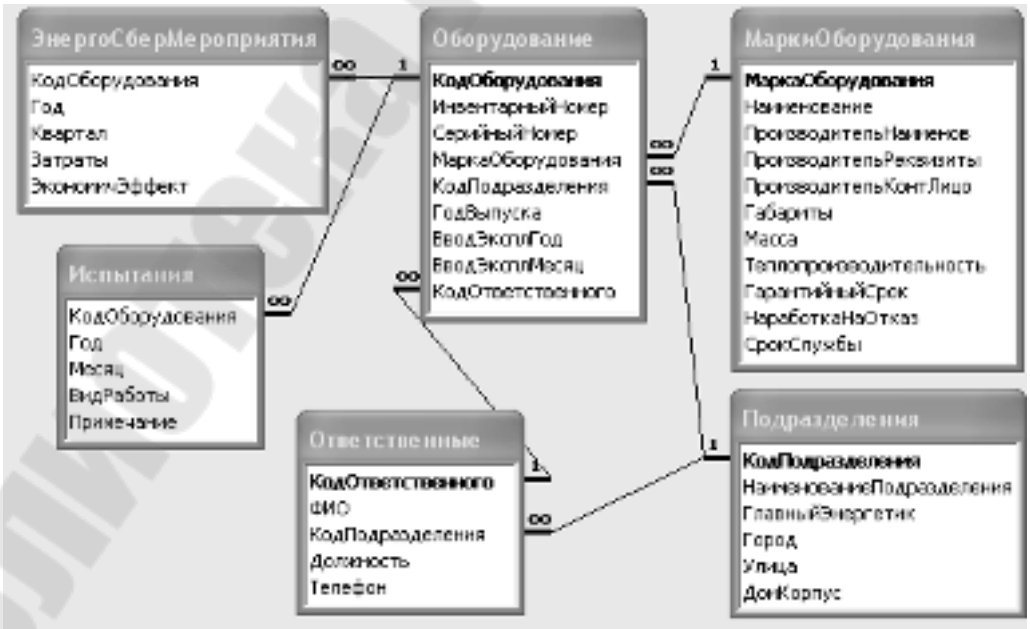
5) даты испытаний (год, месяц);

6) технические характеристики: масса (кг); номинальная теплопроизводительность (кВт, для соответствующего оборудования); габариты (длина Х ширина Х высота, м); гарантийный срок эксплуатации (месяцев); средняя наработка на отказ (ч); полный назначенный срок службы (лет);

7) лицо, ответственное за эксплуатацию оборудования (ФИО, подразделение, должность, телефон);

8) энергосберегающие мероприятия, связанные с оборудованием: наименование мероприятия, дата выполнения (год, квартал), затраты на внедрение (млн. руб.); экономический эффект от внедрения мероприятия за квартал (млн. руб.); для одной единицы оборудования могут периодически выполняться энергосберегающие мероприятия.

Требуемую информацию об оборудовании можно разными способами распределить по реляционным таблицам. Одну из возможностей проектирования БД иллюстрирует следующая схема данных:



Первичные ключи *КодОборудования*, *КодОтветственного*, *КодПодразделения* имеют тип данных *Счетчик*, а соответствующие поля на стороне отношения «многие» имеют тип данных *Числовой* (при этом свойство *Размер поля* имеет значение *Длинное целое*). Для таблиц *Оборудование*, *МаркиОборудования* поле *МаркаОборудования* имеет тип данных *Текстовый*. Остальные поля имеют типы данных *Текстовый*, *Числовой* в зависимости от заносимых данных. При этом задается ***минимально необходимое*** значение свойства *Размер поля* (например, 10-15 символов для текстового поля *МаркаОборудования*).

Отметим, что в приведенных БД некоторые поля могут содержать много текстовых данных: поле *ХарактеристикиПродукции* в таблице Продукция – Пример 1; поле *ПроизводительРеквизиты* в таблице МаркиОборудования – Пример 2). С учетом анализа предметных областей, эти данные атомарны – нас интересуют полные значения этих полей. Если бы требовалось хранить и обрабатывать по отдельности, например, каждую их характеристик вида продукции (масса, размеры, цвет и т. д.) или каждый из реквизитов производителя определенной марки оборудования (адрес, расчетный счет и т. д.), то указанные поля пришлось бы разбить на несколько полей согласно требованиям нормализации.

На приведенной схеме данных таблица Оборудование выступает в качестве оперативной таблицы по отношению к справочным таблицам МаркиОборудования, Подразделения, Ответственные. Таблица Ответственные, в свою очередь, является оперативной по отношению к таблице Подразделения. Кроме этого, таблица Оборудование является справочной таблицей по отношению к таблицам ЭнергоСберМероприятия и Испытания. Соответствующие первичные и внешние ключи обеспечивают связи 1:∞ между таблицами БД.

СПИСОК ЛИТЕРАТУРЫ

1. Хомоненко А.Д., Цыганков В.М., Мальцев М.Г. Базы данных: Учебник для высших учебных заведений / Под ред. проф. А.Д. Хомоненко. – М.: Бином-Пресс; СПб.: КОРОНА принт, 2006. -736 с.
2. Дейт К. Введение в системы баз данных. – М.: Вильямс, 2001. – 1072 с.
3. Крёнке Д. Теория и практика построения баз данных. – СПб.: Питер, 2003. – 800 с.
4. Гетц К., Гунделой М., Литвин П. Разработка корпоративных приложений в Access 2002. – СПб.: Питер, 2003. – 400 с.
5. Информатика. Учебник. / Под ред. проф. Н.В. Макаровой. – М.: Финансы и статистика, 2006. -768 с.
6. Гладких Б.А. Информатика. Введение в специальность: Учебное пособие для вузов – Томск: Изд-во научно-техн. литературы, 2002. – 350 с.
7. Исаченко А.Н. Модели данных и системы управления базами данных / А. Н. Исаченко, С. П. Бондаренко. – Минск: БГУ, 2007. – 220 с.
8. Пушников А.Ю. Введение в системы управления базами данных. Часть 1. Реляционная модель данных: Учебное пособие / Изд-е Башкирского ун-та. – Уфа, 1999. – 108 с.
9. Пушников А.Ю. Введение в системы управления базами данных. Часть 2. Нормальные формы отношений и транзакции: Учебное пособие / Изд-е Башкирского ун-та. – Уфа, 1999. – 138 с.
10. Асенчик О.Д., Стародубцев Е.Г. Практическое пособие по теме «СУБД MS Access» для студентов экономических специальностей дневного и заочного отделений. – Гомель: ГГТУ им. П. О. Сухого, 2001, 2005 (2-е стереотипное издание). – 44 с.
11. Стародубцев Е.Г. Системы управления базами данных: пособие по дисциплинам «Базы данных», «Технологии организации, хранения и обработки данных», «Разработка приложений баз данных для информационных систем» для студентов специальности 1-40 01 02 «Информационные системы и технологии (по направлениям)». – Гомель: ГГТУ им. П. О. Сухого, 2010. – 30 с.

СОДЕРЖАНИЕ

Водополова Наталия Виталиевна
Стародубцев Евгений Генрихович

ОРГАНИЗАЦИЯ И ПРОЕКТИРОВАНИЕ БАЗ ДАННЫХ

Пособие
по одноименному курсу
для слушателей специальности 1-40 01 73
«Программное обеспечение информационных систем»
заочной формы обучения

Подписано в печать 9.11.12.
Формат 60x84/$_{16}$. Бумага офсетная. Гарнитура «Таймс».
Ризография. Усл. печ. л. 1,86. Уч.-изд. л. 1,7.
Изд. № 2.
http://www.gstu.by

Отпечатано на цифровом дуплекаторе
с макета оригинала авторского.
Учреждение образования «Гомельский государственный
технический университет имени П.О. Сухого».
246746, г. Гомель, пр. Октября, 48