Лабораторная работа 1
Проектирование базы данных

Цель работы: научиться создавать модели данных: определять сущности предметной области, их атрибуты, первичные и альтернативные ключи, устанавливать отношения между сущностями, разрабатывать концептуальную, логическую и физическую модели данных.

1.1. Концептуальное проектирование базы данных

База данных представляет собой совокупность специальным образом организованных данных, хранимых в памяти вычислительной системы и отражающих состояние объектов и их взаимосвязи в рассматриваемой предметной области [1]. Другими словами, база данных – это совокупность таблиц, связанных отношениями между собой.

Первый этап проектирования базы данных состоит в разработке концептуальных моделей данных для каждого из существующих типов пользователей создаваемого приложения (локальных концептуальных моделей). Представление пользователя включает в себя данные, необходимые конкретному пользователю для принятия решений или выполнения некоторого задания. Обычно представление пользователя отражает некоторую функциональную область в общем поле деятельности предприятия – например, производство, маркетинг, сбыт, управление кадрами или складской учет. Пользователь может быть как отдельным работником, так и группой лиц, которые будут непосредственно работать с создаваемым приложением [8].

Каждая локальная концептуальная модель данных включает:

- типы сущностей;
- типы связей;
- атрибуты;
- домены атрибутов;
- потенциальные ключи;
- первичные ключи.

1.1.1. Определение типов сущностей

Сначала необходимо определить основные объекты, которые могут интересовать пользователя. Эти объекты являются типами ***сущностей***, входящих в модель базы данных. Сущность – это абстракция множества объектов реального мира, в котором все предметы множества имеют одинаковые характеристики и согласованы с одним и тем же набором правил и линий поведения. Один из методов идентификации сущностей состоит в изучении спецификаций по выполнению конкретных функций пользователя на данном предприятии. Из этих спецификаций следует извлечь все используемые в них существительные или сочетания существительного и прилагательного. Для нашего случая – разработки базы данных учета товара в обувном магазине – можно выделить следующие сущности: «Товар», «Страна-производитель», «Количество товара», «Цена товара», «Цвет товара», «Адрес производителя», «Директор фирмы-производителя».

Затем среди выделенных сущностей выбираются самые крупные объекты или представляющие интерес концепции и исключаются все существительные, которые просто определяют другие объекты. Например, сущности «Адрес производителя» и «Директор фирмы-производителя» могут быть объединены в сводном объекте под названием «Производитель», тогда как сущности «Цена товара», «Цвет товара», «Количество товара» можно объединить в сущности под названием «Товар».

Таким образом, получается, что для нашего примера выделено только две сущности: «Товар» и «Производитель».

1.1.2. Определение типов связей

После выделения сущностей следующим этапом разработки концептуальной модели данных будет установление всех существующих между сущностями ***связей.*** Связь – это идентификатор требований, в соответствии с которыми сущность вовлекается в отношение. Отношения именуются с помощью глагола. Необходимо выбирать все выражения со словами-сущностями, в которых содержатся глаголы. Например: фирма-производитель *производит* товар. Нас интересуют только те связи между сущностями, которые необходимы для удовлетворения требований к проекту. В

большинстве случаев связи являются парными – другими словами, связи существуют только между двумя сущностями. Однако следует проявлять осторожность и тщательно проверять наличие в проекте комплексных связей, объединяющих более двух сущностей различных типов, а также рекурсивных связей, существующих между сущностями одного и того же типа [8].

Особое внимание следует уделять проверке того, были ли выделены *все* связи, явно или неявно присутствующее в спецификациях на проект. В принципе, каждую из возможных пар сущностей было бы полезно проверить на наличие между ними некоторой связи, что может оказаться чрезвычайно трудоемкой задачей. Но вообще отказываться от выполнения подобных проверок неразумно [8].

Установив связи, которые будут иметь место в создаваемой модели, необходимо определить кардинальность каждой из них. Каждая связь может иметь кардинальность либо «один-к-одному» (1:1), либо «один-ко-многим» (1:M), либо «многие-ко-многим» (M:N). Для нашего примера верно следующее:

- производитель может производить несколько товаров;
- один товар (с уникальным артикулом) может быть произведен только одним производителем.

Следовательно, кардинальность связи сущностей «Производитель» и «Товар» – «один-ко-многим» (1:M). Схема, представленная на рис. 1.1, называется ***диаграммой «сущность-связь»*** (или ER-диаграммой). ER-диаграмма является методом представления логической структуры базы данных в графическом виде для более простого и наглядного отображения основных компонентов конкретного проекта базы данных. Одну и ту же ER-модель можно преобразовать как в реляционную модель данных, так и в модель данных для иерархических и сетевых СУБД, или в постреляционную модель данных. Однако так как мы рассматриваем именно реляционные СУБД, то можно считать, что логическая модель данных для нас формулируется в терминах реляционной модели данных.

Существует четыре графических нотации диаграммы «сущность-связь»: Чена (Chen), IDEF1x, Мартина (Martin), Баркера (Barker). В данном лабораторном практикуме используется нотация Мартина, в которой сущности обозначены прямоугольниками,

имя связи указывается на линии, ее обозначающей, атрибуты записываются списком внутри прямоугольника сущности, первичный ключ подчеркивается.

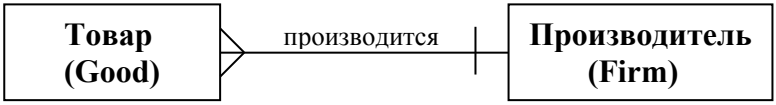

Рис. 1.1. Диаграмма “сущность-связь” на уровне сущностей

1.1.3. Определение атрибутов и связывание их с типами сущностей и связей

На следующем этапе предлагаемой методологии построения концептуальной модели необходимо выявить атрибуты сущностей, т.е. все данные, описывающие сущности и связи, выделенные в создаваемой модели базы данных. Воспользуемся тем же методом, который применялся для идентификации сущностей: выберем все существительные и содержащие их фразы, присутствующие в спецификациях на проект. Выбранное существительное представляет атрибут в том случае, если оно описывает свойство, качество, идентификатор или характеристику некоторой сущности или связи [8].

Самым простой метод выделения атрибутов – после идентификации очередной сущности или связи задать себе следующий вопрос: «Какую информацию требуется хранить о...». Каждому выявленному атрибуту следует присвоить осмысленное имя, понятное пользователям. В некоторых случаях может оказаться полезным попросить пользователей уточнить их требования к хранимой информации [8].

В нашем примере для сущности «Товар» можно выделить такие атрибуты, как: «Наименование товара», «Фирма-производитель», «Цена», «Цвет», «Количество». Для сущности «Производитель» – «Название фирмы», «Адрес», «Телефон», «ФИО директора», «№ банковского счета».

На рис. 1.2. приведена ER-диаграмма на уровне атрибутов.

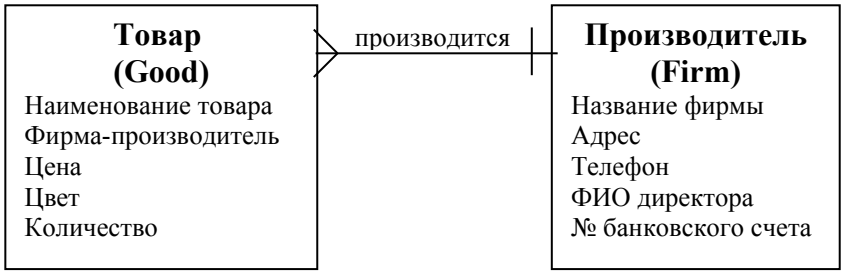

Рис. 1.2. Диаграмма “сущность-связь” на уровне атрибутов

Важно отметить, что каждый атрибут может быть либо ***простым***, либо ***составным***. Составные атрибуты представляют собой набор простых атрибутов. Например, атрибут «Адрес» может быть простым и представлять все элементы адреса как единое значение: «115409, Москва, Каширское ш., д.31». В другом варианте этот же атрибут может быть представлен как составной, т.е. состоящий из нескольких простых атрибутов, содержащих различные элементы адреса. В этом случае то же самое значение может быть разделено на такие атрибуты, как «Почтовый код» (115409), «Город» (Москва), «Улица» (Каширское ш.), «Дом» (31). Выбор способа представления адреса в виде простого или составного атрибута определяется требованиями, предъявляемыми к приложению пользователем. Если пользователь не нуждается в доступе к отдельным элементам адреса, то его целесообразно представить как простой атрибут. Но если пользователю потребуется независимый доступ к отдельным элементам адреса, то атрибут «Адрес» следует сделать составным, образованным из необходимого количества простых атрибутов.

Для нашего примера атрибут ФИО директора будет являться составным, так как при обращении в письме, например, необходимы только имя и отчество, а в почтовом адресе инициалы. Атрибут адрес предполагается использовать только как почтовый адрес, поэтому этот атрибут будет простым.

О каждом атрибуте в отчет помещаются следующие сведения:

- ***имя атрибута и его описание***;
- значение, принимаемое для атрибута ***по умолчанию*** (если таковое имеется);

- является ли атрибут ***обязательным*** (т.е. может ли он отсутствовать или иметь значение NULL);
- является ли атрибут ***составным*** и, если это так, из каких простых атрибутов он состоит;
- является ли данный атрибут ***вычисляемым*** и, если это так, какой метод следует использовать для вычисления его значения;
- является ли данный атрибут ***множественным***, т.е. может ли атрибут иметь одновременно несколько значений для одного и того же экземпляра сущности.

1.1.4. Определение доменов атрибутов

Доменом атрибута называется некоторый набор значений, элементы которого выбираются для присвоения значений атрибуту. Домены должны содержать следующие данные:

- набор допустимых значений для атрибута;
- сведения о размере и формате каждого из полей атрибутов.

Определим домены для каждого атрибута из нашего примера (табл. 1.1).

Таблица 1.1. Домены атрибутов

Название атрибута	Домен
Наименование товара	Любое строковое (текстовое) выражение, размером 15 символов
Фирма-производитель	Любое строковое (текстовое) выражение, размером 15 символов
Цена	Денежное выражение, больше нуля
Цвет	Любое строковое (текстовое) выражение, размером 8 символов
Количество	Числовое поле, целое, больше нуля
Адрес	Любое строковое (текстовое) выражение, размером 20 символов
Телефон	Числовое поле, длиной 7 символов, больше нуля
Фамилия директора	Любое строковое (текстовое) выражение, размером 15 символов
Имя директора	Любое строковое (текстовое) выражение, размером 10 символов

Продолжение табл. 1.1

Название атрибута	Домен
Отчество директора	Любое строковое (текстовое) выражение, размером 15 символов
№ банковского счета	Любое строковое (текстовое) выражение, размером 20 символов

1.1.5. Определение атрибутов, являющихся потенциальными и первичными ключами

На этом этапе для каждой сущности устанавливается потенциальный ключ (или ключи), после чего осуществляется выбор первичного ключа. ***Потенциальным ключом*** называется атрибут или минимальный набор атрибутов заданной сущности, позволяющий уникальным образом идентифицировать каждый ее экземпляр. Для некоторых сущностей возможно наличие нескольких потенциальных ключей. В этом случае среди них нужно выбрать один ключ, который будет называться ***первичным ключом***. Первичный ключ – минимальный набор атрибутов, по значениям которых можно однозначно найти требуемый экземпляр сущности. Все остальные потенциальные ключи будут называться ***альтернативными ключами***. При выборе первичного ключа среди нескольких потенциальных ключей будем руководствоваться приведенными ниже рекомендациями [8]:

- выбирается потенциальный ключ с минимальным набором атрибутов;
- выбирается тот потенциальный ключ, вероятность изменения значений которого минимальна;
- выбирается тот потенциальный ключ, который имеет минимальную вероятность потери уникальности значений в будущем;
- выбирается потенциальный ключ, значения которого имеют минимальную длину (в случае текстовых атрибутов);
- выбирается потенциальный ключ, с которым будет проще всего работать (с точки зрения пользователя).

В нашем случае потенциальными ключами будут являться:

- для сущности «Товар»: «Название товара» + «Фирма-производитель» + «Цвет»;
- для сущности «Производитель»: «Название фирмы».

Среди потенциальных ключей выберем первичные ключи:

- для сущности «Товар»: так как потенциальный ключ сущности «Товар» сложен (состоит из 3 атрибутов), то целесообразно создать «искусственное» ключевое поле, например «Уникальный индекс товара»;
- для сущности «Производитель»: «Название фирмы».

1.2. Логическое проектирование базы данных (для реляционной модели)

1.2.1. Преобразование локальной концептуальной модели данных в локальную логическую модель

На этом этапе структура данных преобразуется в такую форму, которая не вызовет затруднений при реализации в СУБД.

На данном этапе выполняются следующие действия.

- ***Удаление связей типа M:N***. Если в концептуальной модели присутствуют связи типа M:N, то их следует устранить путем определения некоторой промежуточной сущности. Связь типа M:N заменяется двумя связями типа 1:M, устанавливаемыми со вновь созданной сущностью.
- ***Удаление сложных связей***. Сложной называется связь, существующая между тремя и больше типами сущностей. Если в концептуальной модели присутствует сложная связь, ее следует устранить с помощью промежуточной сущности. Сложная связь заменяется необходимым количеством бинарных связей типа 1:M, устанавливаемых со вновь созданной сущностью.
- ***Удаление рекурсивных связей***. Рекурсивными называются такие связи, в которых сущность некоторого типа взаимодействует сама с собой. Если концептуальная модель содержит рекурсивные связи, они должны быть устранены посредством определения некоторой промежуточной сущности.
- ***Удаление связей с атрибутами***. Если в концептуальной модели присутствуют связи, имеющие собственные атрибуты, они должны быть преобразованы путем создания новой сущности.

•***Удаление множественных атрибутов***. Если в концептуальной модели присутствует множественный атрибут, его следует преобразовать путем определения новой сущности.

•***Перепроверка связей типа 1:1.*** В процессе определения сущностей могли быть созданы две различные сущности, которые на самом деле представляют один и тот же объект в предметной области приложения. В подобном случае следует объединить эти две сущности в одну. Если первичные ключи объединяемых сущностей различны, необходимо выбрать один из них в качестве первичного, а другой указать как альтернативный ключ.

•***Удаление избыточных связей***. Связь является избыточной, если одна и та же информация может быть получена не только через нее, но и с помощью другой связи. Всегда следует стремиться создавать минимальные модели данных, и поэтому, если избыточная связь не является очевидно необходимой, ее следует удалять. Установить, что между двумя сущностями имеется больше одной связи, довольно просто. Однако из этого еще не следует, что одна из двух связей обязательно является избыточной, поскольку обе они могут представлять различные объединения, реально существующие в организации.

В нашем примере связь одна, тип связи – «один-ко-многим», сложных, избыточных, рекурсивных и связей с атрибутами нет, множественных атрибутов нет (предполагается, что телефон и банковский счет у фирмы может быть только один), поэтому этот этап пропускается.

1.2.2. Определение набора отношений, исходя из структуры локальной логической модели данных

На данном этапе необходимо на основе созданных локальных логических моделей данных определить наборы отношений, необходимые для представления сущностей и связей, входящих в представления отдельных пользователей о предметной области приложения.

Связи, которые сущность имеет с другими типами сущностей, представляются с помощью механизма ***первичных и внешних ключей***. Для принятия решения о том, откуда взять и куда поместить значения атрибута (атрибутов) внешнего ключа, предварительно следует установить, какая из участвующих в связи сущно-

стей является родительской, а какая – дочерней. ***Родительской*** считается сущность, которая передает копию набора значений своего первичного ключа в отношение, представляющее ***дочернюю сущность***, где эти значения будут играть роль внешнего ключа.

Для каждой присутствующей в логической модели данных бинарной связи типа 1:1, установленной между сущностями E1 и E2, необходимо переслать атрибуты первичного ключа сущности E1 в отношение, представляющее сущность E2.

Для каждой бинарной связи типа 1:M, установленной в логической модели данных между сущностями E1 и E2, необходимо переслать копию атрибутов первичного ключа сущности E1 в отношение, представляющее сущность E2, где они будут играть роль внешнего ключа. Сущность, представляющая «единичную» сторону связи определяется как родительская, а сущность, представляющая «множественную» сторону, – как дочерняя. Для представления данной связи необходимо скопировать первичный ключ родительской сущности в отношение, представляющее дочернюю сущность, где этот ключ должен быть описан как внешний.

В нашем случае родительской является сущность «Производитель», а дочерней – «Товар», связь между сущностями – «один-комногим», следовательно, в сущности «Товар» должен быть атрибут, описывающий фирму-производителя. В нашем примере такой атрибут уже предусмотрен, это атрибут «Фирма-производитель», он будет являться внешним ключом.

1.2.3. Проверка модели с помощью правил нормализации

Нормализация используется для улучшения модели данных для того, чтобы модель удовлетворяла различным ограничениям, позволяющим исключить нежелательное дублирование данных. Нормализация гарантирует, что полученная в результате ее применения модель данных будет наилучшим образом отображать особенности использования информации на предприятии, не содержать противоречий, иметь минимальную избыточность и максимальную устойчивость.

В теории реляционных баз данных обычно выделяется следующая последовательность нормальных форм:

- первая нормальная форма (1NF);
- вторая нормальная форма (2NF);

- третья нормальная форма (3NF);
- нормальная форма Бойса–Кодда (BCNF);
- четвертая нормальная форма (4NF);
- пятая нормальная форма, или нормальная форма проекции-соединения (5NF или PJ/NF).

Основные свойства нормальных форм:

- каждая следующая нормальная форма в некотором смысле лучше предыдущей;
- при переходе к следующей нормальной форме свойства предыдущих нормальных свойств сохраняются.

Таким образом, каждая нормальная форма является в некотором смысле более ограниченной, но и более *желательной*, чем предшествующая. Это связано с тем, что (N+1)-я нормальная форма не обладает некоторыми непривлекательными особенностями, свойственным N-й нормальной форме. Общий смысл дополнительного условия, налагаемого на (N+1)-ю нормальную форму по отношению к N-й нормальной форме, состоит в исключении этих непривлекательных особенностей. Определения нормальных форм приведены в приложении 1.

В нашем примере база данных приведена к первой нормальной форме (так как ни одна из строк таблиц не содержит в любом своем поле более одного значения, т.е. все значения атрибутов атомарны), ко второй нормальной форме (так как нет атрибутов, частично зависящих от ключевого поля) и к третьей нормальной форме (так как нет транзитивных зависимостей).

1.2.4. Создание диаграмм «сущность-связь» логической модели данных

После всех преобразований итоговая логическая модель базы данных представлена на рис. 1.3.

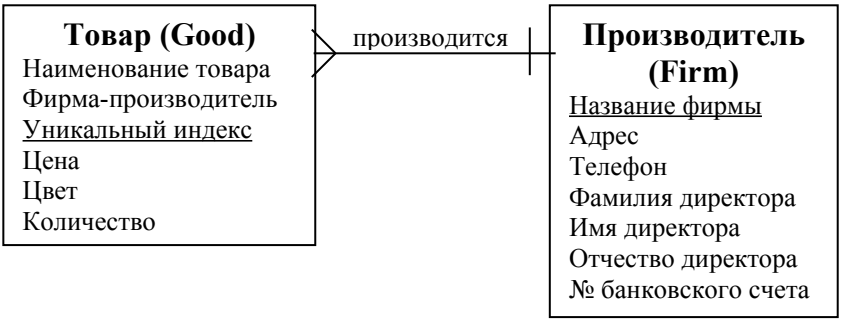

Рис. 1.3. Итоговая логическая модель базы данных учета товара в обувном магазине

1.3. Физическое проектирование базы данных (с использованием реляционной СУБД)

Физическое проектирование заключается в переносе логической модели данных в выбранную СУБД (в нашем случае в СУБД Visual FoxPro).

1.3.1. Имена полей таблицы

Имя поля таблицы в СУБД Visual FoxPro может состоять из 254 символов для таблиц, включенных в базу данных, и 10 символов для свободных таблиц (не включенных в базу данных). В имени поля можно применять буквы, цифры и знак подчеркивания. Использование знаков препинания, специальных символов и пробелов в имени поля не рекомендуется. Также имя поля не должно начинаться с цифры или знака подчеркивания [2]. Выберем имена для сущностей и атрибутов логической модели данных (табл. 1.2 и табл. 1.3).

Таблица 1.2. Таблица Goods

Наименование поля	Комментарии
Good_name	Наименование товара
Firm	Фирма-производитель
Index	Уникальный индекс на складе
Price	Цена
Colour	Цвет
Amount	Количество

Таблица 1.3. Таблица Firms

Наименование поля	Комментарии
Firm_name	Название фирмы
Address	Адрес
Telephone	Телефон
Director_f	ФИО директора
Director_n	
Director_o	
Account	№ банковского счета

1.3.2. Типы данных

После определения имен сущностей и их атрибутов необходимо решить вопрос о том, каким типом данных будет представляться тот или иной атрибут (поле таблицы). Выбор типа данных зависит от домена атрибута и влияет на суммарный объем БД, скорость поиска, допустимые операции с этим атрибутом и т.д. Таким образом, выбирая тип для того или иного атрибута, необходимо учитывать следующее: какие значения может принимать тот или иной атрибут, какие операции будут выполняться с данным атрибутом, сколько физического места займет одно значение для атрибута. Возможные типы данных для полей таблиц, поддерживаемые СУБД Visual FoxPro, приведены в табл. 1.4 [2, 3].

Таблица 1.4. Типы данных

Тип	**Диапазон**	**Объем памяти, байт**	**Описание**
Array			Массив данных некоторого типа
Double	от+/-4.94065648541247E-324 до +/-1.79769313486232E+308	8	Число с плавающей точкой двойной точности
Character	Любые символы	1–254	Текстовая (символьная) строка
Date	от 01/01/100 до 12/31/9909	8	Дата
Float	от-0,9999999999x10^{+19} до 0,9999999999x10^{+20}	8	Число с плавающей точкой
General	Определяется доступной памятью	4 (в dbf)	Ссылка на OLE-объект (для хранения объектов различных сред (графических объектов, текстовых файлов и др.))
Integer	- 2147483647 до 2147483646	4	Число целое
Logical	Истина ({T,t,Y,y}), Ложь ({F,f,N,n})	1	Логическое значение
Memo	Определяется доступной памятью (в dbf)	4	Ссылка на примечание
Numeric	от-0,9999999999x10^{+19} до 0,9999999999x10^{+20}	8	Число с фиксированной точкой целое или дробное
DateTime	от 01/01/1000 до 12/31/9999 и от 00:00:00 утра до 23:59:59 вечера	8	Дата и время
Currency	от -22337203685477,5807 до 922337203685477,5807	8	Денежное значение

Для рассматриваемого примера данные будут иметь следующий тип и размер (табл. 1.5 и табл. 1.6):

Таблица 1.5. Типы данных для таблицы Goods

Наименование поля	**Тип**	**Примечание**
Good_name	Character, 15	Текстовое поле длиной 15 символов
Firm	Character, 15	Текстовое поле длиной 15 символов
Index	Numeric, 2, 0	Числовое поле длиной 2 цифры
Price	Currency	Денежный тип
Colour	Character, 8	Текстовое поле длиной 8 символов
Amount	Integer	Целое число

Таблица 1.6. Типы данных для таблицы Firms

Наименование поля	**Тип**	**Примечание**
Firm_name	Character, 15	Текстовое поле длиной 15 символов
Address	Character, 20	Текстовое поле длиной 20 символов
Telephone	Numeric, 7, 0	Числовое поле длиной 7 цифр
Director_f	Character, 15	Текстовое поле длиной 15 символов
Director_n	Character, 10	Текстовое поле длиной 10 символов
Director_o	Character, 15	Текстовое поле длиной 15 символов
Account	Character, 20	Текстовое поле длиной 20 символов

Список литературы:

Базы данных: Лабораторный практикум / Т.В. Клецова, Н.В. Овсянникова, И.В. Прохоров – М.: МИФИ, 2008 - 132 с.