Лабораторная работа.
ОСНОВЫ ПРОЕКТИРОВАНИЯ БАЗ ДАННЫХ.

КОНЦЕПТУАЛЬНОЕ ПРОЕКТИРОВАНИЕ

Целью разработки любой базы данных является последующее хранение и использование информации о какой-либо предметной области. При разработке базы данных обычно выделяется несколько уровней моделирования, при помощи которых происходит переход от предметной области к конкретной реализации базы данных средствами конкретной СУБД. Можно выделить следующие уровни:

- сама предметная область;
- модель предметной области (концептуальная модель);
- логическая модель данных;
- физическая модель данных;
- собственно база данных и приложения.

Предметная область – это часть реального мира, данные о которой мы хотим отразить в базе данных. Например, в качестве предметной области можно выбрать бухгалтерию какого-либо предприятия, отдел кадров, банк, магазин и т. д.

Модель предметной области – это наши знания о предметной области. Знания могут быть представлены в виде неформальных знаний в мозгу эксперта или выражены формально при помощи каких-либо средств. В качестве таких средств могут выступать текстовые описания предметной области, наборы должностных инструкций, правила ведения дел в компании и т. п. Опыт показывает, что текстовый способ представления модели предметной области крайне неэффективен. Гораздо более информативными и полезными при разработке баз данных являются описания предметной области, выполненные при помощи специализированных графических нотаций. Основным средством разработки концептуальной модели в настоящий момент являются различные варианты ER-диаграмм (Entity Relationship, диаграммы «объект-связь»). Концептуальная модель представляет процессы, происходящие в предметной области, и данные, используемые этими процессами. От того, насколько правильно смоделирована предметная область, зависит успех дальнейшей разработки приложений.

Логическая модель данных. На следующем, более низком уровне, находится логическая модель данных. Логическая модель данных является начальным прототипом будущей базы данных. Логическая мо-

дель строится в терминах информационных единиц, но без привязки к конкретной СУБД. Более того, логическая модель данных необязательно должна быть выражена средствами именно реляционной модели данных. Одну и ту же ER-диаграмму, созданную на предыдущем уровне, 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, можно считать, что логическая модель формулируется в терминах реляционной модели данных.

Решения, принятые на предыдущем уровне при разработке концептуальной модели, определяют некоторые границы, в пределах которых можно развивать логическую модель данных, в пределах же этих границ можно принимать различные решения. При разработке логической модели данных важными являются вопросы:

1. Хорошо ли спроектированы отношения?
2. Правильно ли они отражают концептуальную модель и саму предметную область?

Физическая модель данных. На еще более низком уровне находится физическая модель данных. Физическая модель данных описывает данные средствами конкретной СУБД. Мы будем рассматривать физическую модель данных, реализованную средствами СУБД Microsoft Access (реляционной СУБД), хотя, как уже было сказано ранее, это необязательно. Отношения, разработанные на стадии формирования логической модели данных, преобразуются в таблицы, атрибуты становятся полями таблиц, для ключевых атрибутов создаются уникальные индексы.

При этом опять-таки решения, принятые на уровне логического моделирования, определяют некоторые границы, в пределах которых можно развивать физическую модель данных. Точно так же в пределах этих границ можно принимать различные решения. Например, отношения, содержащиеся в логической модели данных, должны быть преобразованы в таблицы, но для каждой таблицы можно дополнительно объявить различные индексы, повышающие скорость обращения к данным. Многое тут зависит от конкретной СУБД.

При разработке физической модели данных важными являются вопросы:

1. Хорошо ли спроектированы таблицы?
2. Правильно ли выбраны индексы?

3. Насколько много программного кода в виде триггеров и хранимых процедур необходимо разработать для поддержания целостности данных?

Собственно база данных и приложения. И наконец, как результат предыдущих этапов появляется собственно сама база данных. База данных реализована на конкретной программно-аппаратной основе, и выбор этой основы позволяет существенно повысить скорость работы с БД. Например, можно выбирать различные типы компьютеров, менять количество процессоров, объем оперативной памяти, дисковые подсистемы и т. п. Очень большое значение имеет также настройка СУБД в пределах выбранной программно-аппаратной платформы.

Но опять решения, принятые на предыдущем уровне – уровне физического проектирования, определяют границы, в пределах которых можно принимать решения по выбору программно-аппаратной платформы и настройки СУБД.

Таким образом, ясно, что решения, принятые на каждом этапе моделирования и разработки базы данных, будут сказываться на дальнейших этапах. Поэтому особую роль играет принятие правильных решений на ранних этапах моделирования.

Создание модели «объект-связь»

В реальном проектировании структуры базы данных применяется метод, называемый ***семантическим моделированием***. Семантическое моделирование представляет собой моделирование структуры данных, опираясь на смысл этих данных. В качестве инструмента семантического моделирования используются различные варианты ***диаграмм «объект-связь» (ER – Entity Relationship)***. Все такие диаграммы используют графическое изображение объектов предметной области, их свойств и взаимосвязей между объектами.

Основные понятия.

Объект (сущность) – это то, что существует в реальном мире, ограниченном предметной областью, это то, что существенно для разрабатываемой информационной системы, информация об этом обязательно должна быть учтена в модели. Каждый объект должен иметь имя, выраженное существительным в единственном числе. Примерами объектов могут быть «Строение», «Клиент», «Кадастровая книга». Каждый объект в модели изображается в виде прямоугольника с наименованием (рис. 1).

Рис. 1. Изображение объекта

Экземпляр объекта – это конкретный представитель данной сущности. Например, представителем сущности «Клиент» может быть «Клиент Иванов».

Свойство (атрибут) объекта – это именованная характеристика объекта. Наименование атрибута должно быть выражено существительным в единственном числе (возможно, с характеризующими прилагательными). Примерам свойств сущности «Клиент» могут быть такие атрибуты, как «Номер паспорта», «Фамилия», «Имя», «Отчество», «Должность», «Зарплата» и т. п. Атрибуты изображаются овалом, соединенным линией связи с объектом (рис. 2).

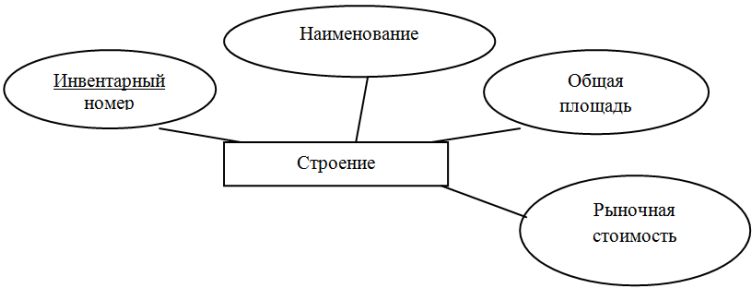

Рис. 2. Изображение атрибутов

Идентификатор объекта – свойство, значение которого является *уникальным* для каждого экземпляра объекта. Идентифицирующее свойство выделяется на диаграмме подчеркиванием.

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: «КЛИЕНТ может сделать несколько ЗАКАЗОВ», «каждый СОТРУДНИК обязан числиться только в одном ОТДЕЛЕ». Графически связь изображается линией, соединяющей две сущности (рис. 3).

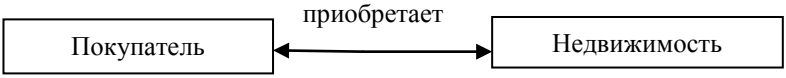

Рис. 3. Графическая связь

Каждая связь имеет два конца и одно или два наименования. Наименование обычно выражается в неопределенной глагольной форме: «иметь», «принадлежать», «приобретать» 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*** означает, что один экземпляр первой сущности связан с одним экземпляром второй сущности. Связь один-к-одному чаще всего свидетельствует о том, что на самом деле мы имеем всего одну сущность, разделенную на две.

Связь типа ***один-ко-многим*** означает, что один экземпляр первой сущности связан с несколькими экземплярами второй сущности. Это наиболее часто используемый тип связи. Сущность со стороны «один» называется ***родительской,*** со стороны «много» – ***дочерней*.**

Связь типа ***многие-ко-многим*** 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 многие-ко-многим является *временным* типом связи, допустимым на ранних этапах разработки модели. В дальнейшем этот тип связи должен быть заменен двумя связями типа один-ко-многим путем создания промежуточной сущности.

Описанный графический синтаксис позволяет *однозначно* читать диаграммы.

Каждая связь может быть прочитана как слева направо, так и справа налево. Слева направо: «Каждый сотрудник может иметь несколько детей». Справа налево: «Каждый ребенок обязан принадлежать хотя бы одному сотруднику».

Пример разработки простой модели «объект-связь».

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

- список сущностей предметной области;
- список атрибутов сущностей;
- описание взаимосвязей между сущностями.

ER-диаграммы удобны тем, что процесс выделения сущностей, атрибутов и связей является итерационным. Разработав первый приближенный вариант диаграмм, мы уточняем их, опрашивая экспертов

предметной области. При этом отображением документации, в которой фиксируются результаты бесед, являются сами ER-диаграммы.

Предположим, что перед нами стоит задача разработать информационную систему по заказу некоторого агентства, занимающегося продажей недвижимости. В первую очередь мы должны изучить предметную область и процессы, происходящие в ней. Для этого мы опрашиваем сотрудников агентства, читаем документацию, изучаем формы заявок, актов купли-продажи и т. п.

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

- хранить информацию о предлагаемых к продаже строениях или помещениях;
- печатать акты на сделку;
- хранить документацию по строениям (паспорта строений) и сделкам. Выделим все существительные в этих предложениях – это будут потенциальные кандидаты на сущности и атрибуты, и проанализируем их (непонятные термины будем выделять знаком вопроса).

Покупатель (клиент) – явный кандидат на сущность.

Паспорт строения – явный кандидат на сущность.

Акт – явный кандидат на сущность.

Строение – явный кандидат на сущность

(?) Сделка – сколько сделок может одновременно совершать один покупатель? Если несколько, то это будет кандидат на новую сущность.

(?) Количество сделок – это, скорее всего, атрибут, но атрибут какой сущности?

Сразу возникает очевидная связь между сущностями – «покупатели могут покупать несколько строений» и «строения могут продаваться нескольким покупателям». Первый вариант диаграммы представлен на рис. 4.

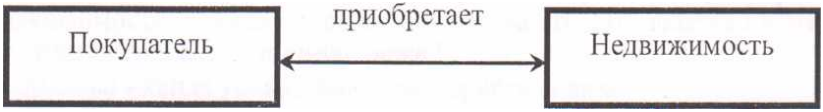

Рис. 4. Диаграмма

Задав дополнительные вопросы менеджеру, мы выяснили, что агентство имеет несколько сотрудников, занимающихся продажами объектов недвижимости. Причем каждое строение может быть продано любым сотрудником.

Куда поместить сущности «Сделка» и «Сотрудник» и с чем их связать? Спросим себя, как связаны эти сущности между собой и с сущностями «Покупатель» и «Строение»?

Покупатели приобретают недвижимость, получая при этом акт, в который внесены данные о приобретенных объектах недвижимости, необходимые для государственной регистрации этого имущества. Каждый покупатель может получить несколько актов.

Каждый акт должен быть оформлен на одного покупателя, не бывает пустых актов. Кроме того, каждый акт должен быть оформлен определенным сотрудником, и любой сотрудник может продавать несколько объектов недвижимости. Таким образом, после уточнения диаграмма будет выглядеть, как показано на рис. 5.

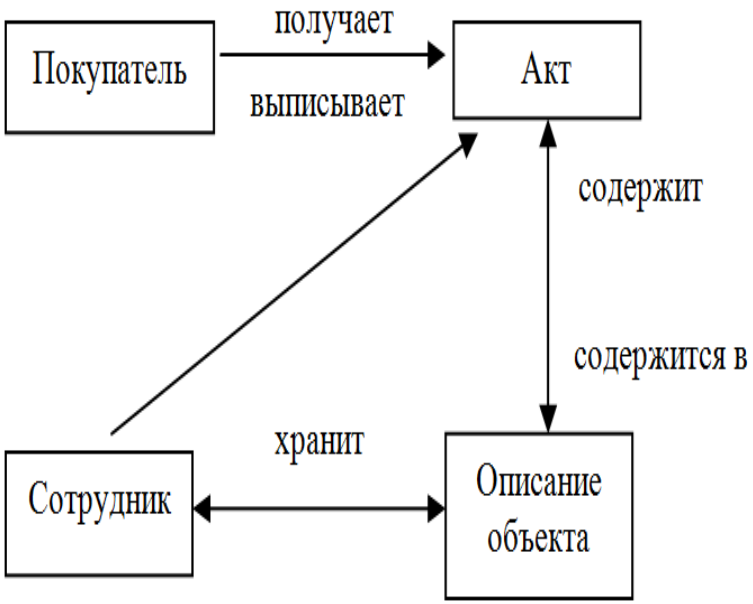

Рис. 5. Уточненная диаграмма

Далее рассмотрим атрибуты сущностей. Беседуя с сотрудниками агентства, мы выяснили следующее:

- каждый покупатель может быть как юридическим (имеет наименование, адрес, банковские реквизиты), так и физическим лицом;
- каждое строение имеет кадастровый номер, наименование, цену и характеризуется общей площадью занесенных в паспорт строений;
- каждый акт имеет уникальный номер, дату выписки, описание объекта недвижимости, а также сумму сделки;
- акт выписывается определенным сотрудником и на определенного покупателя;
- для каждого сотрудника известны фамилия, имя, отчество, должность и дополнительные данные, например, контактный телефон, время работы.

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

- *Юридическое лицо* – термин, определяющий тип покупателя (не является определяющим).
- *Наименование покупателя* – явная характеристика покупателя.
- *Адрес* – явная характеристика покупателя.
- *Банковские реквизиты* – явная характеристика покупателя.
- *Наименование строения* – явная характеристика строения.
- *(?) Стоимость строения* – похоже, что это характеристика продаваемого объекта недвижимости, возможно, рыночная стоимость. Отличается ли эта характеристика от стоимости в акте купли-продажи?
- *Номер акта* – явная уникальная характеристика акта.
- *Дата оформления акта* – явная характеристика акта.
- *(?) Стоимость строения в акте* – опять же это должна быть не просто характеристика продаваемого объекта недвижимости, а характеристика строения в акте купли-продажи. Но стоимость строения уже встречалась выше – это одно и то же?
- *Ф. И. О. сотрудника* – явная характеристика сотрудника (рис. 6).

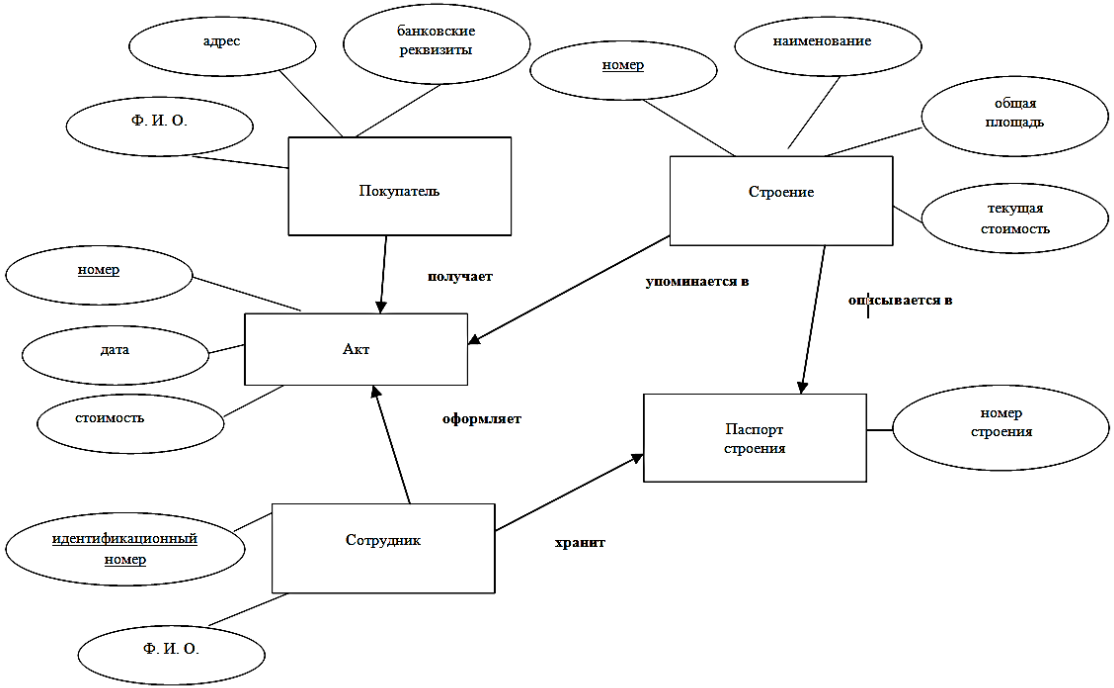

Рис. 6. Общая диаграмма

Практическое задание

Описать предметную область и построить концептуальную схему работы какой-либо предметной области.

Описание предметной области и построение схемы выполнить в текстовом процессоре Microsoft Word с использованием графических объектов, применяя грамотные приемы форматирования текстовых и нетекстовых элементов.

Вариант 1. Описать предметную область и разработать концептуальную схему функционирования какого-либо факультета высшего учебного заведения.

Вариант 2. Описать предметную область и разработать концептуальную схему функционирования какого-либо учреждения культуры (театр, художественная галерея, филармония и т. д.) с точки зрения репертуарного плана.

Вариант 3. Описать предметную область и разработать концептуальную схему функционирования авторемонтной мастерской сельхозпредприятия с точки зрения формирования заказов на ремонт автотехники.

Вариант 4. Описать предметную область и разработать концептуальную схему функционирования фермерского хозяйства с точки зрения поставки производимых товаров на рынок.

Вариант 5. Описать предметную область и разработать концептуальную схему проведения спортивной олимпиады студентов Республики Беларусь для ведения протоколов проводимых соревнований.

Вариант 6. Описать предметную область и разработать концептуальную схему функционирования оптовой товарной базы для ведения документации обработки заказов.

Вариант 7. Описать предметную область и разработать концептуальную схему учета пропусков занятий студентами вашей группы.

Вариант 8. Описать предметную область и разработать концептуальную схему работы с физическими лицами филиала банка.

Вариант 9. Описать предметную область и разработать концептуальную схему функционирования страховой компании.

ЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ

На этапе логического проектирования разрабатывается логическая модель БД. Так как в настоящее время наибольшее распространение получила реляционная модель (реляционные БД составляют 99 % всех используемых к настоящему времени БД), в этой работе рассматривается проектирование реляционной БД.

Реляционная модель состоит из трех частей:
– структурной части;
– целостной части;
– манипуляционной части.

Структурная часть описывает, какие объекты рассматриваются реляционной моделью. Считается, что единственной структурой данных, используемой в реляционной модели, являются взаимосвязанные отношения. Отношение можно представить в форме таблицы особого вида. *Целостная часть* описывает ограничения, которые должны выполняться для любых отношений в любых реляционных БД. *Манипуляционная часть* описывает способы манипулирования данными. В данной работе рассматривается структурная часть реляционной модели.

Фундаментальными понятиями реляционной модели данных являются: *отношение, атрибут, кортеж, первичный и внешний ключи*.

Отношение – это структурная единица реляционной модели, содержащая информацию об одном объекте предметной области. Имя отношения уникально в БД. *Отношение* состоит из двух частей: заголовка и тела. *Заголовок отношения* содержит фиксированное количество атрибутов. *Атрибут* определяет одно из свойств объекта. Имена атрибутов должны быть уникальны в пределах отношения. Запись вида *Имя отношения (список атрибутов)* называют *схемой отношения*.

Тело отношения – содержит множество *кортежей* отношения. Каждый кортеж представляет экземпляр объекта.

Первичный ключ – это атрибут (группа атрибутов), значение которого уникально в данном отношении, что обеспечивает уникальность каждого кортежа. Первичный ключ не может содержать нулевое значение (null-значение). *Внешний ключ* – это атрибут (группа атрибутов), вводимый в отношение для установки связи. Назначение внешнего ключа – обеспечение связи с другими отношениями. Реляционная модель поддерживает три типа связи между отношениями: 1:1, 1:M, M:1. Ключи отношения могут быть простыми – состоящими из одного ат-

рибута или составными – содержащими несколько атрибутов, а также естественными – определяемыми из состава атрибутов отношения или искусственными – вводимыми в отношение разработчиками БД.

Рассмотрим отношение *Сотрудники*, состоящее из атрибутов НомерСотрудника, Фамилия, Зарплата, НомерОтдела.

Пусть в данный момент отношение содержит три кортежа:

(1, Иванов, 1000, 1);

(2, Петров, 2000, 2);

(3, Сидоров, 3000, 1).

Такое отношение естественным образом представляется в виде табл. 1.

Таблица 1. **Отношение Сотрудники**

НомерСотрудника	Фамилия	Зарплата	НомерОтдела
1	Иванов	1000	1
2	Петров	2000	2
3	Сидоров	3000	1

Первичный ключ отношения – НомерСотрудника, *внешний ключ* – НомерОтдела (*обеспечивает связь с отношением* Отделы (НомерОтдела, Наименование)). *Схема отношения имеет вид*: Сотрудники (НомерСотрудника, Фамилия, Зарплата, НомерОтдела).

Реляционной БД называется набор отношений.

Схемой реляционной БД называется набор схем отношений, входящих в БД.

Термины, которыми оперирует реляционная модель данных, имеют «естественные синонимы» (табл. 2).

Таблица 2. **Термины реляционной модели**

Реляционный термин	Соответствующий «табличный» термин
База данных	Набор таблиц
Схема базы данных	Набор заголовков таблиц
Отношение	Таблица
Заголовок отношения	Заголовок таблицы
Тело отношения	Тело таблицы
Атрибут отношения	Наименование столбца таблицы
Кортеж отношения	Строка таблицы

Хотя любое отношение можно изобразить в виде таблицы, нужно четко понимать, что *отношения не являются таблицами.* Это близкие,

но не совпадающие **понятия.** Отношение обладает следующими свойствами:

– в отношении нет одинаковых кортежей;

– кортежи не упорядочены;

– атрибуты не упорядочены;

– все значения атрибутов атомарны, т. е. неделимы.

В этих свойствах в основном и состоят различия между отношениями и таблицами.

Особенности проектирования реляционной базы данных.

Проектирование реляционной БД проходит в том же порядке, что и проектирование БД других моделей, но имеет свои особенности. Проблема логического проектирования реляционной БД состоит в обоснованном принятии решений о том: из каких отношений должна состоять БД и какие атрибуты должны быть у этих отношений.

Проектирование схемы БД должно решать задачи минимизации дублирования данных и упрощения процедур их обработки и обновления. При неправильно спроектированной схеме БД могут возникнуть аномалии модификации данных, для решения подобных проблем проводится нормализация отношений.

Нормализация отношения – это пошаговый процесс декомпозиции (разбиения на более простые). Каждый этап данного процесса приводит схему базы данных к определенной нормальной форме. Каждая форма обладает «лучшими свойствами», чем предыдущая. Окончательная цель нормализации сводится к получению такой БД, в которой каждый факт появляется лишь в одном месте, т. е. исключена избыточность данных.

В теории реляционных БД принято выделять следующую последовательность нормальных форм:

– первая нормальная форма (1НФ);

– вторая нормальная форма (2НФ);

– третья нормальная форма (3НФ);

– нормальная форма Бойса-Кодда (НФБК);

– четвертая нормальная форма (4НФ);

– нормальные формы более высоких порядков.

Каждой нормальной форме соответствует некоторый набор ограничений. Отношение находится в определенной нормальной форме, если оно удовлетворяет набору ограничений этой формы.

Процесс нормализации основан на понятии функциональной зависимости атрибутов. Атрибут *Y функционально зависит* от атрибута *X*

(обозначается $X \rightarrow Y$, читается как «X функционально (или однозначно) определяет Y»), если в любой момент времени каждому значению атрибута X соответствует единственное значение атрибута Y. *Неключевым атрибутом* называется любой атрибут отношения, не входящий в состав первичного ключа. Если неключевой атрибут зависит от составного первичного ключа в целом и не зависит от его части, то говорят о *полной функциональной зависимости* атрибута от составного первичного ключа. Два или более атрибута *взаимно независимы*, если ни один из этих атрибутов не является функционально зависимым от других.

Отношение находится в 1НФ, если значения атрибутов атомарны и все неключевые атрибуты функционально зависят от первичного ключа.

Отношение находится в 2НФ, если выполняются ограничения 1НФ и каждый неключевой атрибут находится в полной функциональной зависимости от составного первичного ключа.

Для того чтобы привести отношение к 2НФ, нужно:

1) исключить из исходного отношения неключевые атрибуты, которые не находятся в полной функциональной зависимости от составного первичного ключа;

2) создать новое отношение (отношения), включив в него часть составного первичного ключа и атрибуты, функционально зависящие от этой части.

Отношение с простым первичным ключом автоматически находится в 2НФ.

Отношение находится в 3НФ, если выполняются ограничения 2НФ и все неключевые атрибуты взаимно независимы и полностью зависят от первичного ключа.

Для того чтобы привести отношение к 3НФ, нужно:

1) исключить из исходного отношения атрибуты, которые функционально зависят от неключевого атрибута;

2) создать новое отношение (отношения), включив в него функционально зависимые неключевые атрибуты.

Отношение с одним неключевым атрибутом автоматически находится в 3НФ.

Третья нормальная форма БД в большинстве случаев считается достаточной для достижения целей нормализации. Приведением отношений к третьей нормальной форме процесс проектирования реляционной БД обычно заканчивается.

Пример проектирования реляционной БД методом нормализации отношений.

Рассмотрим в качестве предметной области некоторую организацию, выполняющую некоторые проекты. Модель предметной области опишем следующим неформальным текстом:

- Сотрудники организации выполняют проекты.
- Проекты состоят из нескольких заданий.
- Каждый сотрудник может участвовать в одном или нескольких проектах или временно не участвовать ни в каких проектах.
- Над каждым проектом может работать несколько сотрудников, или временно проект может быть приостановлен, тогда над ним не работает ни один сотрудник.
- Над каждым заданием в проекте работает только один сотрудник.
- Каждый сотрудник числится в одном отделе.
- Каждый сотрудник имеет телефон, находящийся в отделе.

В ходе дополнительного уточнения того, какие данные необходимо учитывать, выяснилось следующее:

- О каждом сотруднике необходимо хранить табельный номер и фамилию. Табельный номер является уникальным для каждого сотрудника.
- Каждый отдел имеет уникальный номер.
- Каждый проект имеет номер и наименование. Номер проекта является уникальным.
- Каждая работа из проекта имеет номер, уникальный в пределах проекта. Работы в разных проектах могут иметь одинаковые номера.

В ходе логического проектирования на первом шаге предлагается хранить данные в одном отношении, имеющем следующую схему:

СотрудникиОтделыПроекты (НомерСотрудника, Фамилия, НомерОтдела, Телефон, НомерПроекта, Проект, НомерЗадания), где:

НомерСотрудника – табельный номер сотрудника;

Фамилия – фамилия сотрудника;

НомерОтдела – номер отдела, в котором числится сотрудник;

Телефон – телефон сотрудника;

НомерПроекта – номер проекта, над которым работает сотрудник;

Проект – наименование проекта, над которым работает сотрудник;

НомерЗадания – номер задания, над которым работает сотрудник так как каждый сотрудник в каждом проекте выполняет ровно одно задание, то в качестве первичного ключа отношения необходимо взять пару атрибутов {*НомерСотрудника, НомерПроекта*}.

В текущий момент состояние предметной области отражается следующими фактами:

– сотрудник Иванов, работающий в 1-м отделе, выполняет в первом проекте «Космос» задание 1 и во втором проекте «Климат» задание 1;

– сотрудник Петров, работающий в 1-м отделе, выполняет в первом проекте «Космос» задание 2;

– сотрудник Сидоров, работающий в 2-м отделе, выполняет в первом проекте «Космос» задание 3 и во втором проекте «Климат» задание 2.

Электронное состояние отражается в таблице (курсивом выделены ключевые атрибуты, составной первичный ключ) (табл. 3).

Таблица 3. **Электронное состояние предметной области**

Номер сотрудника	Фамилия	Номер отдела	Телефон	Номер проекта	Проект	Номер задания
1	Иванов	1	11-22-33	1	Космос	1
1	Иванов	1	11-22-33	2	Климат	1
2	Петров	1	11-22-33	1	Космос	2
3	Сидоров	2	33-22-11	1	Космос	3
3	Сидоров	7	33-22-11	2	Климат	2

Первая нормальная форма.

Отношение СотрудникиОтделыПроекты находится в 1НФ. Все атрибуты имеют атомарные значения. Определен составной первичный ключ.

Одного взгляда на таблицу отношения СотрудникиОтделыПроекты достаточно, чтобы увидеть, что данные хранятся в ней с *большой избыточностью*. Во многих строках повторяются фамилии сотрудников, номера телефонов, наименования проектов. Кроме того, в данном отношении хранятся вместе независимые друг от друга данные и данные о сотрудниках, об отделах, проектах, работах по проектам. Пока никаких действий с отношением не производится, это не столь важно. Но как только состояние предметной области изменяется, то при попытке соответствующим образом изменить состояние БД возникает большое количество проблем.

Исторически эти проблемы получили название *аномалий*. Понятие аномалии в данном случае рассматривается как неадекватность модели данных предметной области (что говорит на самом деле о том, что логическая модель данных попросту неверна!).

Так как аномалии проявляют себя при выполнении операций, изменяющих состояние БД, то различают следующие виды аномалий:

– аномалии добавления;

– аномалии обновления;

– аномалии удаления.

Аномалии добавления. В отношение **СотрудникиОтделыПроекты** нельзя добавить данные о сотруднике, который пока не участвует ни в одном проекте. Действительно, если, например, во втором отделе появляется новый сотрудник Пушников, и он пока не участвует ни в одном проекте, то мы должны вставить в отношение кортеж (4, Пушников, 2, 33-22-11, *null,* null, null). Это сделать невозможно, так как атрибут *НомерПроекта* (номер проекта) входит в состав первичного ключа и, следовательно, не может содержать null-значений.

Аномалии обновления. Фамилии сотрудников повторяются во многих кортежах отношения. Поэтому если сотрудник меняет фамилию, то такие изменения необходимо *одновременно* выполнить во всех местах, где эта фамилия встречается, иначе отношение станет некорректным (например, один и тот же сотрудник в разных кортежах будет иметь разные фамилии). Таким образом, обновление БД одним действием реализовать невозможно.

Аномалии удаления. При удалении некоторых данных может произойти потеря другой информации. Например, если закрыть проект «Космос» и удалить все строки, в которых он встречается, то будут потеряны все данные о сотруднике Петрове.

Для устранения указанных аномалий продолжим процесс нормализации.

В отношении **СотрудникиОтделыПроекты** можно привести следующие примеры функциональных зависимостей.

Зависимость атрибута от ключа отношения:

{НомерСотрудника,НомерПроекта} → **Фамилия**;

{НомерСотрудника,НомерПроекта} → **НомерОтдела**;

{НомерСотрудника,НомерПроекта} → **Телефон**;

{НомерСотрудника,НомерПроекта} → **Проект**;

{НомерСотрудника,НомерПроекта} → **НомерЗадания**.

Зависимость атрибутов, характеризующих сотрудника, от табельного номера сотрудника:

НомерСотрудника → **Фамилия**;

НомерСотрудника → **НомерОтдела;**

НомерСотрудника → **Телефон**.

Зависимость наименования проекта от номера проекта:

НомерПроекта → Проект.

Зависимость номера телефона от номера отдела:

НомерОтдела → Телефон.

Замечание. Приведенные функциональные зависимости *не выведены* из внешнего вида отношения, приведенного в таблице. Эти зависимости отражают взаимосвязи, обнаруженные между объектами предметной области, и являются дополнительными ограничениями, определяемыми предметной областью. Таким образом, функциональная зависимость – это *семантическое понятие*. Она возникает, когда по значениям одних данных в предметной области можно определить значения других данных. Например, зная табельный номер сотрудника, можно определить его фамилию, по номеру отдела можно определить номер телефона. Функциональная зависимость *задает дополнительные ограничения* на данные, которые могут храниться в отношениях.

Вторая нормальная форма.

Отношение *СотрудникиОтделыПроекты* не соответствует требованиям 2НФ, так как есть атрибуты, зависящие от части составного первичного ключа.

Зависимость атрибутов, характеризующих сотрудника, от табельного номера сотрудника является зависимостью от части составного первичного ключа. Зависимость наименования проекта от номера проекта является зависимостью от части составного первичного ключа.

Для того чтобы устранить зависимость атрибутов от части составного первичного ключа, нужно произвести ***декомпозицию*** (разбиение) отношения на несколько отношений. При этом *те атрибуты, которые зависят от части сложного ключа,* выносятся в отдельное отношение.

Отношение **СотрудникиОтделыПроекты** *декомпозируем на три отношения* – **СотрудникиОтделы**, **Проекты, Задания.** Отношение **СотрудникиОтделы** (*НомерСотрудника*, Фамилия, НомерОтдела, Телефон) представлено в табл. 4.

Таблица 4. **Отношение СотрудникиОтделы**

НомерСотрудника	Фамилия	НомерОтдела	Телефон
1	Иванов	1	11-22-33
2	Петров	1	11-22-33
3	Сидоров	2	33-22-11

Отношение Проекты (*НомерПроекта* Проект) приведено в табл. 5.

Таблица 5. **Отношение Проекты**

НомерПроекта	Проект
1	Космос
2	Климат

Отношение Задания (*НомерСотрудника*, *НомерПроекта*, НомерЗадания) представлено в табл. 6.

Таблица 6. **Отношения задания**

НомерСотрудника	НомерПроекта	НомерЗадания
1	1	1
1	2	1
2	1	2
3	1	3
3	2	2

Отношения, полученные в результате декомпозиции, находятся в 2НФ. Действительно, отношения **СотрудникиОтделы** и **Проекты** имеют простые ключи, следовательно, автоматически находятся во 2НФ, отношение **Задания** имеет составной первичный ключ, но единственный неключевой атрибут **НомерЗадания** функционально полно зависит от первичного ключа ***{НомерСотрудника, НомерПроекта}***. Часть аномалий обновления устранена.

Третья нормальная форма.

Отношение **СотрудникиОтделы** не находится в 3НФ, так как имеется функциональная зависимость неключевых атрибутов (зависимость номера телефона от номера отдела):

НомерОтдела → Телефон.

Для того чтобы устранить зависимость неключевых атрибутов, нужно произвести декомпозицию отношения на несколько отношений. При этом *те неключевые атрибуты, которые являются зависимыми*, выносятся в отдельное отношение.

Отношение **СотрудникиОтделы** декомпозируем на два отношения – **Сотрудники** и **Отделы** (табл. 7).

Таблица 7. **Отношение СотрудникиОтделы**

НомерСотрудника	Фамилия	НомерОтдела
1	Иванов	1
2	Петров	1
3	Сидоров	2

Отношение Отделы (*НомерОтдела*, Телефон) приведено в табл. 8.

Таблица 8. **Отношение Отделы**

НомерОтдела	Телефон
1	11-22-33
2	33-22-11

Обратим внимание на то, что атрибут **НомерОтдела**, *не являвшийся ключевым* в отношении **СотрудникиОтделы**, *становится первичным ключом* в отношении **Отделы**. Именно за счет этого устраняется избыточность, связанная с многократным хранением одних и тех же номеров телефонов. Атрибут **НомерОтдела** в отношении **Сотрудники** становится внешним ключом для обеспечения связи между отношениями.

Таким образом, все обнаруженные аномалии устранены. Реляционная модель состоит из четырех отношений: **Сотрудники, Отделы, Проекты, Задания**. Каждое отношение соответствует третьей нормальной форме. База данных является адекватной описанной модели предметной области.

Схема БД имеет следующее описание:

Отделы (НомерОтдела, Телефон);

Сотрудники (НомерСотрудника, Фамилия, НомерОтдела);

Проекты (НомерПроекта, Проект);

Задания (НомерСотрудника, НомерПроекта, НомерЗадания).

Схема БД может иметь также графическое представление (рис. 7).

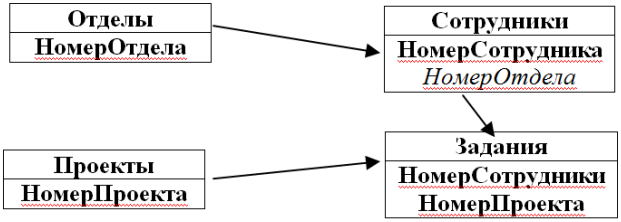

Рис. 7. Графическое представление БД

Получение реляционной схемы из модели «объект-связь».

Шаг 1. Каждый объект превращается в отношение. Имя объекта становится именем отношения.

Шаг 2. Каждое свойство объекта становится атрибутом отношения. Уточняется имя атрибута.

Шаг 3. Идентификатор объекта превращается в первичный ключ отношения. Если у объекта идентификатор не определен, первичный ключ отношения определяется из состава атрибутов или вводится искусственный первичный ключ.

Шаг 4. Связи «один-к-одному» становятся ключами. Для этого первичный ключ одного отношения, участвующего в связи, вводится в другое отношение в качестве внешнего ключа.

Шаг 5. Связи «один-ко-многим» становятся внешними ключами. Для этого первичный ключ отношения, участвующего в связи, со стороны «один» вводится в отношение, участвующее в связи, со стороны «многие» в качестве внешнего ключа.

Шаг 6. Связи «многие-ко-многим» становятся новым отношением, состоящим из первичных ключей отношений, участвующих в связи. Таким образом, связь «многие-ко-многим» преобразуется в связи «один-ко-многим».

Шаг 7. Выполняются шаги по нормализации полученных отношений и приведение их к третьей нормальной форме.

Задание.

Разработать схему БД в третьей нормальной форме для предметной области «Риелтерское агентство». База данных должна содержать сведения о клиенте (Ф. И. О., адрес), типе сделке (продажа, покупка, аренда), оплате сделки и информацию о риелторе (Ф. И. О., квалификация – брокер, агент, поверенный).

Примечание. Риелтор выступает в качестве агента – реализацию данного статуса он осуществляет на основе агентского договора, который он заключает с потребителем услуг. Риелтор выступает в качестве поверенного – при этом заключается договор поручения, который возлагает на риэлтора обязательства осуществить от имени и за счет доверителя определенные юридические действия в отношении принадлежащего ему или используемого им объекта недвижимости. Риелтор выступает в качестве брокера между потребителем услуг, и риелтором заключается договор комиссии; в соответствии с ним риелтор принимает на себя обязательства от своего собственного имени и за счет комитента совершить одну или несколько сделок с объектом недвижимости.

Самостоятельная работа

Разработать в представленных ниже задачах схему БД в третьей нормальной форме.

Задание 1. Построить БД, описывающую результаты сессии. Информация должна содержать номер семестра, сведения о студенте

(Ф. И. О., группа, специальность), сведения о сдаваемом предмете (название, семестр), дату сдачи экзамена, оценку и Ф. И. О. экзаменатора.

Задание 2. Построить БД, описывающую работу библиотеки с читателем. Информация должна содержать сведения о читателе (Ф. И. О., адрес, телефон), информацию о выданной книге (название, автор, издательство) и дату выдачи книги.

Задание 3. Построить БД, описывающую обращение больных в поликлинику. Информация должна содержать сведения о больных (Ф. И. О., адрес, дату рождения), враче (Ф. И. О., специальность), дате осмотра и заключение врача.

Задание 4. Построить БД, описывающую работу с заказами некоторой оптовой базы. Информация должна содержать сведения о заказчике (название фирмы, адрес, телефон), сведения о заказываемом товаре (наименование, фирма-изготовитель, год выпуска, стоимость единицы продукции), а также количество заказанного товара.

Задание 5. Построить БД, описывающую формирование фонда сети магазинов некоторой фирмы. Информация должна содержать сведения о магазине (название, адрес, телефон), сведения о поставщике (наименование, адрес, телефон) сведения о товаре (наименование, количество) и дату поставки.

Задание 6. Построить БД, описывающую работу с клиентами фирмы по техническому обслуживанию торгового оборудования. Информация должна собираться о мастерах, выполняющих ремонтные работы (Ф. И. О., квалификация, телефон), о магазинах, подающих заявки на ремонт оборудования (наименование оборудования, магазин, адрес, телефон), и о выполнении заказа с указанием даты выполнения и оплаты.

Задание 7. Построить БД, описывающую репертуарную политику театра. Информация собирается об актерах (Ф. И. О., звание, дата рождения, адрес, телефон), о пьесе (авторы, название, список ролей с указанием их характеристики, т. е. возраст, амплуа и т. д.) и о репертуаре на следующий месяц с указанием даты спектакля.

ФИЗИЧЕСКОЕ ПРОЕКТИРОВАНИЕ

Концептуальная модель (модель «объект-связь») описывает процессы, происходящие в предметной области, и информацию, используемую этими процессами, но она не подходит для создания структуры БД. Ло-

гическая модель представляет описание структуры БД. В ней отражена одна из возможных моделей (иерархическая, сетевая, в нашем случае реляционная модель) данных. Но описание структуры БД, полученное на этапе логического проектирования, является формальным. Оно выполнено в терминах реляционной модели БД для общего случая и не учитывает особенности среды реализации, т. е. используемой СУБД. Логическая модель БД является основой для создания физической модели.

На этапе физического проектирования осуществляется выбор СУБД для реализации информационной системы и, с учетом этого разрабатывается СУБД-ориентированная модель БД. В учебном процессе используется СУДБ Microsoft Access, поэтому в данной работе будет рассмотрена разработка базы данных Microsoft Access.

Независимо от выбора СУБД процесс разработки физической модели реляционной базы данных выполняется по следующей схеме:

1. Каждое отношение преобразуется в таблицу БД. Имя отношения становится именем таблицы, если оно соответствует требованиям к именам таблиц выбранной СУБД. В противном случае определяется новое имя, соответствующее требованиям.

2. Каждый атрибут становится полем таблицы, уточняется тип поля, определяются дополнительные характеристики.

3. Первичный ключ отношения преобразуется в первичный ключ таблицы.

4. Внешние ключи отношения становятся внешними ключами таблицы.

5. Для каждой таблицы определяется набор индексов, ускоряющих процесс обработки данных (операции поиска в базе данных). Для ключевых полей (внешних и первичных) обязательно определение индексов. Индекс первичного ключа должен обладать свойством уникальности.

6. Связи между отношениями отражаются в связи между таблицами БД. Для каждой связи задаются параметры, обеспечивающие целостность (достоверность) данных.

Таким образом, процесс проектирования физической модели можно рассматривать как отражение элементов реляционной БД, разработанной на этапе логического проектирования в элементы СУБД-ориентированной БД. Для этого необходимо знать:

- Какие требования предъявляет СУБД к именам таблиц и полей?
- Какие типы данных поддерживает СУБД?

• Какие дополнительные свойства можно устанавливать для полей таблицы?

• Каким образом в СУБД формируются индексы?

• Каким образом в СУБД поддерживается целостность данных?

Получив ответы на эти вопросы, можно приступить к разработке физической модели. Результатом данного этапа проектирования является описание таблиц базы данных, описание связей между таблицами и схема базы данных.

При выполнении процесса преобразования реляционной модели данных в БД Microsoft Access необходимо знать правила формирования имен таблиц и полей, поддерживаемые типы данных, возможности обеспечения целостности данных в СУБД Microsoft Access.

База данных Microsoft Access.

Имена полей.

Система управления базами данных Microsoft Access предъявляет следующие требования к именам полей и элементам управления (поле со списком, кнопка, флажок и т. п.):

– имя должно отражать характер информации, хранимой в поле, и в то же время быть довольно коротким;

– длина имени – максимум 64 символа;

– имя – это любая комбинация букв, цифр, пробелов и специальных символов. Исключение составляют точка (.), восклицательный знак (!), надстрочный символ (‘), квадратные скобки ([ ]) и прямые кавычки («);

– имя не должно начинаться с символа «пробел»;

– имя не должно содержать управляющие символы (с кодами ASCII от 0 до 31);

Типы данных.

Для каждого поля определяется тип данных, которые будут размещаться в этом поле. Система управления базами данных Microsoft Access поддерживает типы данных, перечисленные в табл. 9.

При выборе типа данных необходимо ответить на вопросы.

1. Какие данные предполагается вводить в поле? Например, невозможно сохранить текстовую информацию в поле с числовым типом данных.

2. Какой размер поля необходим для сохранения и работы с данными?

3. Какие операции будут производиться над данными этого поля? Например, нельзя суммировать значения полей текстового типа и объекта OLE.

4. Будет ли производиться сортировка или индексирование данных? Для объекта OLE нельзя делать ни того, ни другого, а в текстовом поле числа рассматриваются как символы строки и могут сортироваться неправильно, это же характерно для многих форматов даты.

5. Планируется ли выполнение групповых операций в запросах? Использование объектов OLE в этом случае недопустимо.

Таблица 9. **Типы данных**

Тип данных	Содержание	Размер
Текстовый	Произвольные символы	От 1 до 255 символов, пустые значения не хранятся
Поле МЕМО	Многострочный текст	До 65 535 символов
Числовой	Любые числовые данные для математических вычислений, за исключением денежных операций	1, 2, 4 или 8 байт
Дата/время	Даты и время (100–9999 годы)	8 байт
Денежный	Числа от –922 337 203 685 447,5808 до 922 337 203 687 447,5808; значения валют; невозможность округления во время вычислений; содержит до 15 символов в целой части числа и 4 в дробной	8 байт
Счетчик	Последовательные с шагом 1 или случайные числа; автоматический ввод при добавлении записи; невозможность обновлений и повторений	4 байта
Логический	Значения «Да/Нет», а также поля, содержащие только одно из двух возможных значений («Истина/Ложь», «Включено/Выключено»)	1 бит
Объекты OLE	Объекты, использующие протокол OLE и технологию связывания и внедрения	До 1 Гб (ограничено объемом диска)
Гиперссылка	Содержит как минимум 3 части: текст для показа ссылки; адрес – путь к файлу (UNC) или к странице (URL); подадрес – для ориентации внутри файла или страницы	Каждая часть гиперссылки содержит до 2048 символов
Мастер подстановок	Выбор значения из другой таблицы или из списка, используя поле со списком	Тот же размер, что и у поля, используемого в подстановке, обычно 4 байта

Свойства полей.

Описание. Определяет текст, содержащий более полное описание поля, чем его имя.

Размер поля. Определяется для данных текстового и числового типа. Размер поля должен быть минимально необходимым.

Текстовый тип данных. Размер поля, заданный по умолчанию, равен 50 символам. Минимально-достаточный размер определяется по максимальному (по количеству символов, включая пробелы) значению поля. Но, независимо от установленного размера при сохранении данных, сохраняется только значимая часть.

Для числовых данных определено несколько размерных характеристик. **Числовой тип данных** представлен в табл. 10.

Таблица 10. **Числовой тип данных**

Размер	Описание	Точность	Расход памяти
Байт	Числа в диапазоне 0–255	–	1 байт
Целое	Целые числа в диапазоне от –32768 до 32767	–	2 байта
Длинное целое	Целые числа в диапазоне от –2147483648 до 2147483647	–	4 байта
Одинарное с плавающей точкой	Действительные числа в диапазоне от –3.402823E38 до –1.401298E-45 для отрицательных и от L401298E-45 до 3.402823E38 для положительных чисел	7	4 байта
Двойное с плавающей точкой	Действительные числа в диапазоне от –1.79769313486231E308 до –4.94065645841247E-324 для отрицательных и от 1.7976931348623 LE308 до 4.94065645841247E-324 для положительных чисел	15	8 байт

Формат поля. Указывает форматы вывода текста, чисел и дат на экран и при печати. Не изменяет значение поля, влияет только на его отображение. Можно выбрать один из встроенных форматов Access или создать пользовательский формат. Встроенные форматы устанавливаются для полей числового, логического и денежного типов, счетчик, дата/время.

Для числовых полей возможны встроенные форматы: фиксированный, с разделителями разрядов, процентный и др. Для даты/время –

краткий формат даты, полный формат даты и т. д. Для текстовых полей часто используется значение «>», которое преобразует строчные символы в заглавные.

Число десятичных знаков. Определяет выводимое число десятичных знаков. Используется для числового и денежного типов данных.

Новые значения. Для полей типа Счетчик определяет формирование значений: последовательные значения (1, 2, 3, ...) или случайные числа.

Маска ввода. Устанавливает формат, используемый при вводе данных. Задает строку символов, облегчающую ввод. При создании масок можно использовать приведенные символы в табл. 11.

Таблица 11. **Символы при создании масок**

Символ	Действие, соответствующее символу
0	Цифры от 0 до 9, ввод обязателен
9	Цифры и знак пробела, ввод не обязателен
#	Цифра или знак пробела, а также знаки «+» и «–»
L	Буквы, ввод обязателен
?	Буквы, ввод не обязателен
A	Буква или цифра, ввод обязателен
a	Буква или цифра, ввод не обязателен
&	Любой символ, ввод обязателен
C	Любой символ, ввод не обязателен
.	Десятичный разделитель
,	Разделитель групп разрядов
: ; - /	Разделители даты и времени
<	Перевод в нижний регистр
>	Перевод в верхний регистр
!	Символы вводятся справа налево
«Password»	Выводит на экран звездочки вместо введенных символов

Подпись. Предоставляет для поля надпись, которая будет использоваться в формах и отчетах, а также в заголовках таблиц, которые содержат это поле.

Значение по умолчанию. Позволяет определить значение, вводимое в поле автоматически при добавлении новой записи.

Условие на значение. Определяет требования к вводимым в поле значениям. Если требование не выполнено, то выводится сообщение об ошибке ввода. Текст сообщения содержится в свойстве ***Сообщение об ошибке***.

Обязательное поле. Определяет необходимость ввода данных в поле.

Пустые строки. Определяет, допускается или не допускается ввод в поле пустых строк («»). Свойство поддерживается для полей со следующими типами данных: текстовый, поле МЕМО, гиперссылка.

Индексированное поле. Определяет наличие индекса для поля таблицы. Для индекса указывается, допускается или не допускается совпадение значений.

Обычно индексы создаются для полей, которые являются ключами связи между таблицами, служат ускорению поиска записей в таблице, выполнения многотабличных запросов, отчетов, форм. Среди всех индексов выделяется первичный индекс (Primary Key, первичный ключ или ***ключевое поле***), который обеспечивает идентификацию записей таблицы. В один индекс может входить несколько полей таблицы, для каждого поля указываются свойства:

– порядок сортировки (по возрастанию или убыванию);

– ключевое поле (только один индекс);

– уникальный индекс (возможность повторения значений индекса);

– пропуск пустых полей.

Подстановка. Для некоторых полей можно указать подстановку данных из фиксированного списка значений небольшого объема либо ссылки на значения ключевого поля другой таблицы. В таблице вместо обычного поля выводится элемент управления – поле со списком или список, которые содержат значения для выбора. При описании поля необходимо указать источник данных и некоторые параметры подстановки.

Схема данных. Система управления базами данных Access создает *схему данных*, в которой определяется состав таблиц и запросов, устанавливаются связи.