Тема 10. Создание БД в MS Access

В Microsoft Office Access 2007 данные организуются в таблицы. Большинство БД включают несколько таблиц. Например, в одной таблице могут храниться сведения о продуктах, во второй — сведения о заказах, а в третьей — сведения о клиентах.

Каждая строка называется также записью, а каждый столбец, или тип элемента, называется также полем.

Таблицы

Таблица содержит данные по определенной теме, например, сведения о сотрудниках или товарах. Каждая запись в таблице включает данные об одном элементе, например о конкретном сотруднике. Запись состоит из полей и включает такие сведения, как имя, адрес и телефонный номер. Кроме того, запись обычно называется строкой, а поле – столбцом.

База данных может включать множество таблиц, в которых хранятся данные по различным темам. Каждая таблица может состоять из множества полей различного типа, включая текст, числа, даты и рисунки.

Формы

Формы иногда называются окнами ввода данных. Это интерфейсы, которые используются для работы с данными и часто содержат кнопки для выполнения различных команд. Большинство пользователей баз данных предпочитают просматривать, вводить и редактировать данные таблиц при помощи форм.

Формы позволяют работать с данными в удобном формате; кроме того, в них можно добавлять функциональные элементы, например кнопки команд.

Отчеты

Отчеты служат для сбора и представления данных, содержащихся в таблицах. Каждый отчет можно отформатировать так, чтобы представить сведения в наиболее удобном виде.

Обычно отчеты форматируют для печати, но их можно также просматривать на экране, экспортировать в другую программу или отправлять в виде сообщений электронной почты.

Запросы

Запросы являются основным рабочим инструментом базы данных и могут выполнять множество различных функций. Самая распространенная функция запросов — извлечение определенных данных из таблиц. Данные, которые необходимо просмотреть, как правило, находятся в нескольких таблицах; запросы позволяют представить их в одной таблице. Кроме того, поскольку

обычно не требуется просматривать все записи сразу, с помощью запросов можно, задав ряд условий, «отфильтровать» только нужные записи.

Макросы

Макросы в приложении Access можно рассматривать как упрощенный язык программирования, который позволяет добавлять функциональные возможности в базу данных. Например, кнопке команды в форме можно назначить макрос, который будет запускаться при нажатии этой кнопки. Макрос содержит последовательность действий для выполнения определенной задачи, например для открытия отчета, выполнения запроса или закрытия базы данных. Большинство операций с базой данных, выполняемых вручную, можно автоматизировать с помощью макросов, которые позволяют существенно экономить время.

Модули

Модули, как и макросы, являются объектами, которые можно использовать для добавления функциональных возможностей в базу данных. В то время как макросы создаются в приложении Access путем выбора макрокоманд из списка, модули пишутся на языке программирования Visual Basic для приложений (VBA) (VBA (Visual Basic for Applications). Версия макроязыка программирования Microsoft Visual Basic, используемая для программирования приложений для Microsoft Windows и поставляемая с некоторыми программами корпорации Майкрософт.).

2. Процесс разработки

1) Определение цели создания базы данных.

Целесообразно записать цель создания базы данных на бумаге: задачи, способы использования и список пользователей. Для базы данных небольшого объема для работы на дому можно определить следующую простую цель: «База данных клиентов содержит сведения о клиентах и используется для рассылки сообщений электронной почты и отчетов». При создании более сложной базы данных для большого количества пользователей, например в организации, описание цели может состоять из нескольких параграфов; необходимо указать время и способы использования БД различными пользователями. Основная задача — составить подробное описание цели создания БД, чтобы иметь возможность обращаться к нему в процессе проектирования.

2) Поиск и организация необходимых данных.

Процесс поиска и организации необходимых данных следует начать с записи имеющихся сведений. Соберите необходимые документы и составьте список типов данных (например, список полей в бланке). Какие данные

требуется записывать? Какие поля для заполнения необходимо создать? Определите нужные элементы и составьте их список.

Если база данных предназначена для нескольких пользователей, попросите их предложить свои варианты.

Важно помнить, что данные необходимо разбивать на минимальные элементы. Имя следует разделять на две части: имя и фамилию, чтобы фамилия была доступна для использования.

Составьте список вопросов, ответы на которые требуется получать с помощью БД.

3) Распределение данных по таблицам

Чтобы распределить данные по таблицам, выделите основные группы или темы.

При разработке БД следует стремиться к однократному сохранению каждого элемента данных.

Каждая таблица должна содержать столбец для однозначного определения каждой строки таблицы. Как правило, в этих целях используется уникальный идентификационный номер, например код сотрудника или серийный номер. В базе данных такие сведения носят название первичного ключа таблицы. В Access первичные ключи служат для быстрого связывания данных из нескольких таблиц и их отображения для пользователя.

Первичный ключ не должен содержать повторяющихся значений. Например, не следует использовать в качестве первичного ключа имена людей, т. к. они не являются уникальными. В одной таблице могут существовать две записи с одинаковыми именами.

Первичный ключ должен всегда иметь значение.

4) Создание связей между таблицами.

После распределения данных по таблицам необходимо получить возможность объединять их. Для объединения данных используются связи между таблицами. Далее необходимо определить условия целостности данных.

Условиями целостности данных называется набор правил для поддержания связей между записями в связанных таблицах. Эти правила делают невозможным случайное удаление или изменение связанных данных.

Для того чтобы преодолеть ограничения на удаление или изменение связанных записей, сохраняя при этом *целостность данных*, следует включить *режимы каскадного обновления и каскадного удаления*. При установленном флажке *Каскадное обновление связанных полей* изменение

значения в ключевом поле главной таблицы приводит к автоматическому обновлению соответствующих значений во всех связанных записях. При установленном флажке *Каскадное удаление связанных полей* удаление записи в главной таблице приводит к автоматическому удалению связанных записей в подчиненной таблице.

3. Виды связей между отношениями

Количественный характер участия экземпляров сущностей (один или многие) задается *типом связи.*

1. Связь "один – к - одному" представляет собой простейший вид связи данных, когда первичный ключ таблицы является в то же время внешним ключом, ссылающимся на первичный ключ другой таблицы.

Например, вы делаете *БД для магазина, торгующего автомобилями*. У вас будет таблица с общими характеристиками автомобиля, например: цвет, стоимость, дата выпуска и тип (иномарка или отечественная). При описании реального проекта характеристик будет значительно больше. А теперь представьте, что поскольку машины могут быть отечественного производства и иномарки, то у машин отечественного производства есть свои параметры, которых нет у иномарок, например, гарантия завода изготовителя. В то же время и у иномарок есть свои параметры, которых нет у отечественных машин, например, из какой страны импортирована, пошлина, размещение руля (слева или справа). Здесь приходит на выручку отношение *"один - к – одному"*, т.е. мы реализуем своего рода подтипы (см. рис. 4.4).

Рисунок 4.4 – Пример связи «один-к-одному»

Любой записи в таблице Автомобили в зависимости от значения поля *тип* соответствует запись либо в таблице Иномарки, либо в таблице Отечественные.

2. Связь "один – ко - многим" в большинстве случаев отражает реальную взаимосвязь сущностей в предметной области. Она реализуется уже описанной парой "внешний ключ - первичный ключ", т.е. когда определен

внешний ключ, ссылающийся на первичный ключ другой таблицы. Например, район – город. В одном *районе* находится много *городов*.

Или, например, если есть 2 сущности "Студент" и "Преподаватель" и связь между ними – *руководство дипломными проектами*, то каждый студент имеет только одного руководителя, но один и тот же преподаватель может руководить несколькими студентами-дипломниками (рис. 4.5).

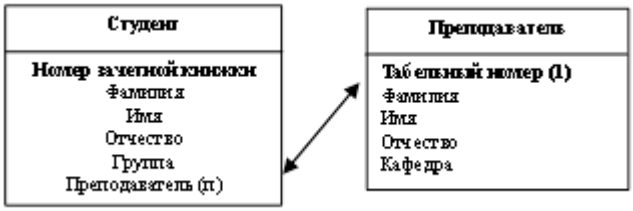

Рисунок 4.5 - Пример связи «один-ко-многим»

3. Связь "многие – ко - многим" в явном виде в реляционных БД не поддерживается. Однако имеется ряд способов косвенной реализации такой связи, которые с успехом возмещают ее отсутствие. Один из наиболее распространенных способов заключается во введении дополнительной таблицы, строки которой состоят из внешних ключей, ссылающихся на первичные ключи двух таблиц.

Например, имеются две таблицы: КЛИЕНТ и ГРУППА_ИНТЕРЕСОВ. Один человек может быть включен в различные группы, в то время как группа может объединять различных людей. Для реализации такой связи *"многие – ко - многим"* вводится дополнительная таблица, назовем ее КЛИЕНТЫ_В_ГРУППЕ, строка которой будет иметь два внешних ключа: один будет ссылаться на первичный ключ в таблице КЛИЕНТ, а другой - на первичный ключ в таблице ГРУППА_ИНТЕРЕСОВ. Таким образом, в таблицу КЛИЕНТЫ_В_ГРУППЕ можно записывать любое количество людей и любое количество групп.

Другим примером связи *"многие – ко - многим"* является следующий: имеются две таблицы: СТУДЕНТ и ПРЕПОДАВАТЕЛЬ и между ними установлена связь - *лекции*. Один студент слушает лекции разных преподавателей, а преподаватель читает лекции многим студентам.

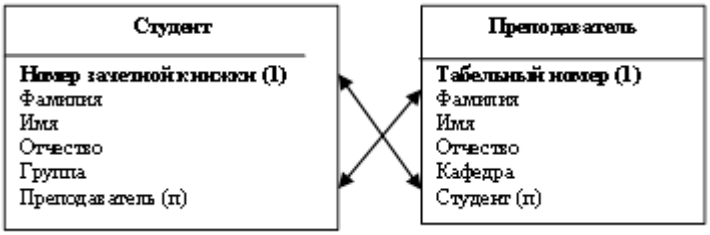

Рисунок 4.6 – Пример связи «многие ко многим»

Для реализации такой связи введем дополнительную таблицу ЛЕКЦИИ .

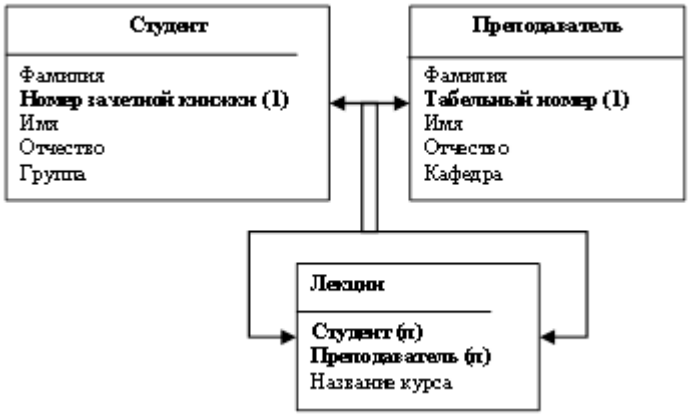

Рисунок 4.7 – Пример связи «многие ко многим»

Инструмент связей – это средство представления *сложных объектов*, каждый из которых может рассматриваться как множество взаимосвязанных *простых объектов*. Деление на простые и сложные объекты, также как и характер взаимосвязи, является условным и определяется особенностями анализа предметной области, т. е. характером использования данных о предметах в решаемых прикладных задачах. При этом с точки зрения, например, конструктора, ДЕТАЛЬ является сложным объектом, а с точки зрения ПОСТАВЩИКА – простым.

4. Нормализация отношений

Нормализация таблиц - исключение избыточности данных.

После определения таблиц, полей, индексов и связей между таблицами следует посмотреть на проектируемую БД в целом и проанализировать ее, используя *правила нормализации*, с целью устранения логических ошибок.

С практической точки зрения *нормализацией* называется процесс удаления избыточных данных. Каждый элемент должен храниться в БД в одном и только одном экземпляре.

Рассмотрим задачу проектирования БД на базе следующей таблицы:

Таблица 4.6 – Таблица «Сессия», находящаяся в ненормализованной форме

ФИО студента	Семестр	Дисциплина	Форма отчетности	Оценка	
Иванов М.А.	1	Математика	Зачет	4	
		История	Экзамен	5	
		Философия	Зачет	4	
Петрова А.П.	1	Английский язык	Зачет	3	
		Математика	Зачет	4	
Сидоров К.К.	3	Информатика	Зачет	5	
		Экономика	Экзамен	3	

Этот вариант таблицы «Сессия» не является отношением, т. к. большинство ее столбцов не атомарны. Значение любого атрибута реляционной таблицы является *атомарным*, если само это значение, в свою очередь, не является реляционной таблицей (отношением). Атомарными являются значения столбцов ФИО СТУДЕНТА, СЕМЕСТР. Остальные столбцы таблицы – множественные.

Эта таблица называется таблицей в *ненормализованной форме (ННФ)* или *ненормализованной таблицей*, т. к. содержит одну или несколько повторяющихся групп данных.

Для преобразования данных в отношение необходимо реконструировать таблицу, например, с помощью процесса вставки. Результат имеет вид:

Таблица 4.7 - Универсальное отношение «СЕССИЯ»

ФИО студента	Семест	Дисциплина	Форма отчетности	Оценка	Количест часов
Иванов М.А.	1	Математика	Зачет	4	28
Иванов М.А.	1	История	Экзамен	5	32

Иванов М.А.	1	Философия	Зачет	4	24
Петрова А.П.	1	Английский язык	Зачет	3	60
Петрова А.П.	1	Математика	Зачет	4	28
Сидоров К.К.	3	Информатика	Зачет	5	60
Сидоров К.К.	3	Экономика	Экзамен	3	32

Такое преобразование приводит к возникновению большого объема избыточных данных. Но полученную таблицу называют *универсальным отношением* проектируемой БД. В одно универсальное отношение включаются все представляющие интерес атрибуты, и оно может содержать все данные, которые предполагается размещать в БД. При проектировании БД универсальное отношение используется в качестве отправной точки.

При использовании универсального отношения возникают следующие проблемы:

1) *Избыточность данных*. Значения столбцов таблицы многократно повторяются. Повторяются также и некоторые наборы значений столбцов, например, данные о дисциплине.

2) *Потенциальная противоречивость*. Если при вводе данных, например, количества часов для дисциплины АНГЛИЙСКИЙ ЯЗЫК, была допущена ошибка, то для ее исправления необходимо найти все строки, содержащие сведения об этой дисциплине, и во всех этих строках произвести изменения.

3) *Аномалии модификации:*

- *аномалии вставки*. В БД не может быть записан новый преподаватель, если он не ведет ни одну дисциплину по данному учебному плану. По аналогичным причинам нельзя ввести и новую дисциплину, если она не изучается в рамках данного учебного плана.

- *аномалии удаления*. Обратная проблема возникает при необходимости удаления записей, содержащих информацию об успеваемости всех студентов по конкретной дисциплине. При таком удалении будут утрачены и сведения о преподавателе, ведущем данную дисциплину.

- *аномалии обновления*. При попытке изменения одного из атрибутов для некоторого преподавателя, например, номера телефона, необходимо обновить соответствующие значения в строках для всех студентов, которых обучает этот преподаватель. Если такой модификации будут подвергнуты не все требуемые строки, то в этом случае БД будет содержать противоречивые сведения.

Решение этих проблем состоит в разделении данных и связей (или *декомпозиции*), т. е. в выделении в отдельные таблицы сведений о студентах, преподавателях, дисциплинах и результатах сдачи экзаменов:

Студенты

№	ФИО студента
1.	Иванов В.П.
2.	Петрова А.П.
3.	Сидоров К.К.

Преподаватели

№	ФИО преподавателя
1.	
2.	
3.	
4.	

Относительно таблиц *нормализация* - это разбиение таблицы на две или более, обладающих лучшими свойствами при включении, изменении и удалении данных. Окончательная цель нормализации сводится к получению такого проекта БД, в котором *каждый факт появляется лишь в одном месте*, т.е. исключена избыточность информации. Это делается не столько с целью экономии памяти, сколько для исключения возможной противоречивости хранимых данных.

Люди также интересуются этой лекцией: 1.2. Основные программные компоненты сети.

После применения правил нормализации логические группы данных располагаются не более, чем в одной таблице. Это дает следующие *преимущества*:

¾ данные легко обновлять или удалять;

¾ исключается возможность рассогласования копий данных;

¾ уменьшается возможность введения некорректных данных.