

ВВЕДЕНИЕ. СУБД. ПРОЕКТИРОВАНИЕ И СОЗДАНИЕ ТАБЛИЦ.

1.1. Цель работы

Усвоение методов определения предметных областей и формализации информации.

1.2. Организация баз данных

Любые объекты в окружающем мире характеризуются совокупностью знаний о них. Так, например, мы знаем о человеке, как его зовут, когда он родился, сколько у него детей, и кто его дети, кличка любимой собаки, семейное положение, болезни, образование, привычки, домашний адрес, место работы, черты характера и т.п. Когда в качестве объекта выступает какая-либо организация или технологический процесс, то им соответствует свой специфический и тоже бесконечный набор атрибутов знания (информации). Этот набор знаний постоянно увеличивается, он не связан с какой-либо областью их приложения, а содержит все то, что известно об объекте – предмете исследования. Такая полная совокупность знаний об объекте окружающего мира и составляет *предметную область объекта*. Однако, для обработки информации об объекте полный охват его предметной области не только не требуется, но даже и вреден. Более того, даже не просто вреден, а невозможен. Поскольку предметная область содержит бесконечное множество знаний об объекте, то их определение, фиксация и обработка потребует бесконечного объема ресурсов: бумаги, чернил, памяти машины, времени работы оператора и т.п. При организации информационного обеспечения любого процесса необходимо, прежде всего, выделить из предметной области необходимый минимум знаний. Таких знаний должно быть как можно меньше, ибо каждый новый атрибут – это дополнительные затраты ресурсов, нарастающие при этом отнюдь не линейно. С другой стороны, этих знаний должно быть достаточно, для того чтобы удовлетворять информационные потребности процесса. Таким образом из предметной области необходимо изъять:

–знания не используемые для информационного обеспечения конкретного процесса (например, информация о любимой собаке работника);

–знания, которые могут быть получены путем логических либо математических операций из других (например, если есть информация о дате рождения человека, то нет необходимости выделять его возраст, поскольку он может быть получен исходя из текущей даты и даты рождения).

–повторяющиеся знания.

Оставшийся минимум атрибутов предметной области, помещенный на определенный носитель называется *базой данных*. Вид базы данных определяется выбранным исходя из возможностей организации и спецификой объекта носителем. Наиболее общее подразделение баз данных на ручные и машинные.

Ручная база данных – это база данных, выполненная без применения средств вычислительной техники, организованная и обрабатываемая целиком и полностью человеком. Домашняя фонотека – типичный пример ручной базы данных, хотя информация представлена на электронных носителях – магнитных кассетах.

Машинная (автоматизированная) база данных – система, в которой организация хранения и обработки информации производится с использованием компьютеров. Особенно эффективными показали себя такие базы данных для хранения и обработки текстовой и числовой информации. Современные системы управления базами данных (СУБД) позволяют хранить информацию в любом формате, поддерживаемом приложениями операционной системы, и обрабатывать ее средствами этих приложений. В машинной базе данных, например, можно хранить информацию о структуре и содержании стандартов ИСО 9000, фотографии работников организации, технологическую и конструкторскую документацию, разработанную средствами любых систем автоматического проектирования, программы для станков с ЧПУ, рекламные видеоролики об организации и т.д.

Одной из проблем, не решенных на сегодняшний день, является передача информации между ручными и машинными базами. Единственным способом такого взаимодействия является схема **машинная (автоматизированная) база данных – человек – ручная база данных**. В результате информация обрабатывается иногда даже хуже, чем при использовании любой из них в отдельности. Решение проблемы многие видят в постепенном отмирании ручных баз данных и полном переходе на машинные.

Машинные базы данных действительно несоизмеримо более эффективно удовлетворяют информационные потребности, однако для их использования необходимо соблюдать определенные правила взаимодействия с искусственным интеллектом. Значительная часть этих правил связана с тем, чтобы обеспечить «понимание» компьютером человека и наоборот. Русский человек испытывает определенные затруднения в общении с японцем. Но ведь и японцы и русские относятся к одному и тому же виду *Homo Sapiens*. И японец и русский в принципе мыслят одинаково – символьными образами – «чанками». Компьютер же по своей сути существо чуждое человеку – он мыслит кодами. Процесс его «мышления» заключается в формальном выполнении фиксированного набора команд, представленных средствами двоичной логики. Само содержание таких команд крайне сложно для понимания человеком, особенно для того, кто связан профессионально с вычислительной техникой. Поэтому необходимо иметь средства обеспечивающие «перевод» запросов с человеческого языка на машинный, а результатов их обработки – обратно на язык понятный оператору (пусть даже и английский). Одним из аспектов такого перевода является представление предметных областей на машинных носителях – перевод информации, представленной чанками – в двоичный код.

1.3. Представление данных

Работы по созданию универсальных средств представления знаний ведутся едва ли не сначала 60-х годов. В результате появились, так называемые, *модели данных*, представляющие собой упрощенные способы отображения знаний об объекте и использующие определенные принципы организации данных. На базе таких моделей данных были разработаны программные системы обработки информации – системы управления базами данных или СУБД. Интересная деталь: практически все известные на сегодняшний день модели данных (кроме объектных) были разработаны почти

одновременно (с 1967 по 1971 годы). Однако на том или ином историческом этапе рынок СУБД захватывался одной максимум двумя моделями данных. На сегодняшний день имеются системы, реализующие все известные модели данных. Причем реляционная модель является самой универсальной, поскольку позволяет реализовывать любые схемы данных и структуры. На сегодняшний день известны следующие модели данных:

- ✓ системы управления файлами;
- ✓ иерархические модели данных;
- ✓ сетевые модели данных;
- ✓ реляционные модели данных;
- ✓ постреляционные модели данных;
- ✓ многомерные модели данных;
- ✓ объектно-ориентированные модели данных.

Поиски решения проблемы представления знаний с целью их формализации и последующей обработки начали осуществляться задолго до создания автоматизированных систем управления базами данных. Информацию необходимо группировать, сортировать, преобразовывать. Для осуществления такой обработки традиционно осуществляется расчленение информации на совокупность характеристик – *атрибутов*. Например, информация о покупном изделии может включать в себя поставщика, стоимость, наличие на складе, в какое изделие входит и т.п. Все это атрибуты, позволяющие удовлетворять информационные запросы в рамках процесса управления поставщиками. Каждый конкретный информационный объект имеет уникальный набор *значений атрибутов*. Так, например, приведенный выше набор атрибутов для покупного изделия «Болт» может иметь следующие значения: поставщик – ЗАО «Галс», стоимость – 10 коп, наличие на складе – 100 кг, в какое изделие входит – ДТ-700. Такая организация хранения информации позволяет производить её эффективную обработку. Например, можно определить, какие ещё изделия поставляет ЗАО «Галс» или подсчитать стоимость всех покупных изделий, входящих в определенную продукцию. Такой подход к обработке информации посредством её расчленения на пары атрибут – значение имеет длительную историю, существенно превышающую возраст автоматизированных систем управления данными. Точная дата возникновения такого принципа неизвестна, но можно с уверенностью сказать, что он появился почти одновременно с языком и письменностью. Примечательно, что и в наш информационный век ничего более эффективного изобрести так не удалось. И хотя этот принцип наиболее адекватен для табличных моделей данных, тем не менее он является основой и для всех других, в том числе и объектно-ориентированных, которые вроде бы должны бы быть основаны не на расщеплении, а на рассмотрении объекта в совокупности его характеристик.

1.4. Практическое задание

Определить необходимый набор атрибутов для создания базы данных, обеспечивающей решение следующих задач.

1. Управление документацией.

2. Обслуживание пациентов поликлиники.
3. Регистрация автотранспорта.
4. Прием студентов в учебное заведение.
5. Контроль качества продукции.
6. Изготовление детали (любой на выбор).
7. Управление кадрами.
8. Разработка домашней видеотеки.
9. Организация работы оптовой фирмы.
10. Разработка графика работы персонала предприятия.
11. Инвентаризация оборудования подразделения.
12. Управление измерительным оборудованием.

1.5. Указания по выполнению работы

1. Представленный набор атрибутов должен обеспечивать удовлетворение любого информационного запроса в рамках поставленной задачи.
2. Полученная база данных не должна содержать избыточной информации (кроме, возможно повторяющейся).
3. Результат оформить в виде таблицы:

Информационная задача: {привести задание}

№ пп	Название атрибута	Пример значения атрибута	Примечание
1, 2,3,...	Имя атрибута	2 – 3 примера значения	Для удовлетворения каких информационных запросов он необходим (1 – 2 примера)

1.6. Контрольные вопросы

1. Понятие предметной области.
2. Понятие избыточных знаний.
3. Понятие базы данных.

4. Ручная и автоматизированная база данных.
5. Модели данных.
6. Определение атрибута.
7. Определение значения атрибута.

2. Лабораторная работа №2. Начальные данные о реляционной СУБД. Основы работы с СУБД Access

2.1. Цель работы

Изучение принципов организации хранения информации в реляционной СУБД. Основы работы с реляционной СУБД на примере СУБД Access.

2.2. Программное обеспечение

Для выполнения работы необходим MS Office 97. Рекомендуется MS Office 2000/XP

2.3. Теоретические сведения

2.3.1. Реляционная модель данных

Недостатки иерархической и сетевой моделей привели к появлению новой, реляционной модели данных, созданной Тэдом Коддом в 1970 году. Реляционная модель была попыткой упростить структуру базы данных. В ней отсутствовали явные указатели на предков и потомков, а все данные были представлены в виде простых таблиц, разбитых на строки и столбцы.

К сожалению, практическое определение понятия «реляционная база данных» оказалось гораздо более расплывчатым, чем точное математическое определение, данное этому термину Коддом в 1970 году. В первых реляционных СУБД не были реализованы некоторые из ключевых частей модели Кодда, и этот пробел был восполнен только впоследствии. По мере роста популярности реляционной концепции реляционными стали называться многие базы данных, которые на деле таковыми не являлись.

В ответ на неправильное использование термина "реляционный" Кодд в 1985 году написал статью, где сформулировал 12 правил, которым должна удовлетворять любая база данных, претендующая на звание реляционной. С тех пор двенадцать правил Кодда считаются определением реляционной СУБД. Однако можно сформулировать и более простое определение:

Реляционной называется база данных, в которой все данные, доступные пользователю, организованы в виде таблиц, а все операции над данными сводятся к операциям над этими таблицами.

Приведенное определение не оставляет места встроенным указателям, имеющимся в иерархических и сетевых СУБД. Несмотря на это, реляционная СУБД также способна реализовать отношения предок/потомок, однако эти отношения представлены исключительно значениями данных, содержащихся в таблицах.

2.3.2. Принципы построения реляционной базы данных

В реляционной базе данных информация организована в виде таблиц, разделённых на строки и столбцы, на пересечении которых содержатся значения данных. У каждой таблицы имеется уникальное имя, описывающее её содержимое. Более наглядно структуру таблицы иллюстрирует рис. 2.1, на котором изображена таблица OFFICES. Каждая горизонтальная строка этой таблицы представляет

Офисы

Офис №	Город	Рук.	Кол. служащих	Продажи
22	Москва	108	20	9,086,042.00
11	Ярославль	106	8	4,692,000.00
12	Кострома	104	9	1,739,000.00
13	Череповец	105	15	5,735,157.00
21	Рыбинск	107	5	1,835,915.00

Данные об офисе в Ярославле

Данные об офисе в Рыбинске

Город, в котором расположен офис

Идентификатор служащего, управляющего офисом

Объём продаж офиса с начала года

Рис. 2.1. Структура реляционной таблицы

отдельную физическую сущность – один офис. Пять строк таблицы вместе представляют все пять офисов компании. Все данные, содержащиеся в конкретной строке таблицы, относятся к офису, который описывается этой строкой.

Каждый вертикальный столбец таблицы «Офисы» представляет один элемент данных для каждого из офисов. Например, в столбце «Город» содержатся названия городов, в которых расположены офисы. В столбце «Продажи» содержатся объёмы продаж, обеспечиваемые офисами.

На пересечении каждой строки с каждым столбцом таблицы содержится в точности одно значение данных. Например, в строке, представляющей ярославский офис, в столбце «Город» содержится значение «Ярославль». В столбце «Продажи» той же строки содержится значение 4,692,000.00, которое является объёмом продаж ярославского офиса с начала года.

Все значения, содержащиеся в одном и том же столбце, являются данными одного типа. Например, в столбце «Город» содержатся только слова, в столбце «Продажи» содержатся денежные суммы, а в столбце «Рук.» содержатся целые числа, представляющие идентификаторы служащих. Множество значений, которые могут содержаться в столбце, называется доменом этого столбца. Доменом столбца «Город» является множество названий городов. Доменом столбца «Продажи» является любая денежная сумма.

У каждого столбца в таблице есть своё имя, которое обычно служит заголовком столбца. Все столбцы в одной таблице должны иметь уникальные имена, однако разрешается присваивать одинаковые имена столбцам, расположенным в различных таблицах. На практике такие имена столбцов, как **ИМЯ**, **АДРЕС**, **СТОИМОСТЬ**, **ПРОДАЖИ** часто встречаются в различных таблицах одной базы данных.

Столбцы таблицы упорядочены слева направо, и их порядок определяется при создании таблицы. В любой таблице всегда есть как минимум один столбец. В

стандарте ANSI/ISO не указывается максимально допустимое число столбцов в таблице, однако почти во всех коммерческих СУБД этот предел существует и обычно составляет примерно 255 столбцов.

В отличие от столбцов, строки таблицы не имеют определённого порядка. Это значит, что если последовательно выполнить два одинаковых запроса для отображения содержимого таблицы, нет гарантии, что оба раза строки будут перечислены в одном и том же порядке.

В таблице может содержаться любое количество строк. Вполне допустимо существование таблицы с нулевым количеством строк. Такая таблица называется пустой. Пустая таблица сохраняет структуру, определённую её столбцами, просто в ней не содержится данные. Стандарт ANSI/ISO не накладывает ограничений на количество строк в таблице, и во многих СУБД размер таблиц ограничен лишь свободным дисковым пространством компьютера. В других СУБД имеется максимальный предел, однако он весьма высок – около двух миллиардов строк, а иногда и больше.

Поскольку строки в реляционной таблице не упорядочены, нельзя выбрать строку по ее номеру в таблице. В таблице нет «первой», «последней» или «тринадцатой» строки. Тогда каким же образом можно указать в таблице конкретную строку, например строку для офиса, расположенного в Череповце?

В правильно построенной реляционной базе данных в каждой таблице есть один или несколько столбцов, значения в которых во всех строках разные. Этот столбец (столбцы) называется первичным ключом таблицы. Давайте вновь посмотрим на базу данных, показанную на рис.2.1. На первый взгляд, первичным ключом таблицы «Офисы» могут служить и столбец «Офис», и столбец «Город». Однако в случае, если компания будет расширяться и откроет в каком-либо городе второй офис, столбец «Город» больше не сможет выполнять роль первичного ключа. На практике в качестве первичных ключей таблиц обычно следует выбирать идентификаторы, такие как идентификатор офиса («Офис» в таблице «Офисы»).

Таблица «Товары», фрагмент которой показан на рис. 2.2, является примером

Таблица Товары

Иzg.	Тип	Название	Цена	Кол. в наличии
IBM	2A45C	Системная плата	\$90.00	210
Acer	4100Y	Монитор	\$210.00	25
Osaka inc.	2A45C	Видеоадаптер	\$55.00	38
IBM	41672	Винчестер	\$135.00	0

Первичный ключ

Рис. 2.2. Пример таблицы с составным первичным ключом

таблицы, в которой первичный ключ представляет собой комбинацию столбцов. Такой первичный ключ называется составным. Столбец «Иzg.» содержит идентификаторы производителей всех товаров, перечисленных в таблице, а столбец «Тип» содержит

номера, присвоенные товарам производителями. Может показаться, что столбец «Тип» мог бы и один выполнять роль первичного ключа, однако ничто не мешает двум различным производителям присвоить своим изделиям одинаковые номера. Таким образом, в качестве первичного ключа таблицы «Товары» необходимо использовать комбинацию столбцов «Изг.» и «Тип». Для каждого из товаров, содержащихся в таблице, комбинация значений в этих столбцах будет уникальной.

Первичный ключ для каждой строки таблицы является уникальным, поэтому в таблице с первичным ключом нет двух совершенно одинаковых строк. Таблица, в которой все строки отличаются друг от друга, в математических терминах называется отношением. Именно этому термину реляционные базы данных и обязаны своим названием, поскольку в их основе лежат отношения (таблицы с отличающимися друг от друга строками).

Хотя первичные ключи являются важной частью реляционной модели данных, в первых реляционных СУБД (System/R, DB2, Oracle и других) не была обеспечена явным образом их поддержка. Как правило, проектировщики базы данных сами следили за тем, чтобы у всех таблиц были первичные ключи, однако в самих СУБД не было возможности определить для таблицы первичный ключ. И только в СУБД DB2 Version 2, появившейся в апреле 1988 года, компания IBM реализовала поддержку первичных ключей. После этого подобная поддержка была добавлена в стандарт ANSI/ISO.

Одним из отличий реляционной модели от первых моделей представления данных было то, что в ней отсутствовали явные указатели, используемые для реализации отношений предок/потомок в иерархической модели данных. Однако вполне очевидно, что отношения предок/потомок существуют и в реляционных базах данных. Например, в приведенной выше базе данных каждый из служащих закреплен за конкретным офисом, поэтому ясно, что между строками таблицы «Подразделения» и таблицы «Сотрудники» существует отношение. Не приводит ли отсутствие явных указателей в реляционной модели к потере информации?

Как следует из рис.2.3, ответ на этот вопрос должен быть отрицательным. На рисунке изображено несколько строк из таблиц «Подразделения» и «Сотрудники». Обратим внимание на то, что в столбце «Подразделение» таблицы «Сотрудники» содержится идентификатор офиса, в котором работает служащий. Доменом этого столбца (множеством значений, которые могут в нем храниться) является множество идентификаторов офисов, содержащихся в столбце «Код» таблицы «Подразделения». То, в каком офисе работает инженер Шустров, можно узнать, определив значение столбца «Подразделение» в строке таблицы «Сотрудники» для Шустрова (число 2) и затем отыскав в таблице «Подразделения» строку с таким же значением в столбце «Код» (это для службы внутреннего аудита качества). Таким же образом, чтобы найти всех работников службы внутреннего аудита качества, следует запомнить значение столбца «Код» для нее (число 2), а потом просмотреть таблицу «Сотрудники» и найти все строки, в столбце «Подразделение» в которых содержится число 2 (это строки для Шустрова Е.М. и Стрелкова Г.А.).

Таблица Подразделения

Название	Руководитель	Кол-во сотрудников	Код
Отдел главного технолога	Седов А.А.	150	1
Служба внутреннего аудита качества	Мотылев Г.У.	10	2
Отдел охраны	Димаков П.С.	35	3

Таблица Сотрудники

Ф.И.О.	Образование	Должность	Стаж	Под-раз-деле-ние
Жерликов П.М.	Среднее проф.	Техник-технолог	8 лет	1
Стрелков Г.А.	Высшее	Нач. бюро	12 лет	2
Шустров Е.М.	Высшее	Инженер	2 года	2
Седов А.А.	Высшее	Главный	10 лет	1

Рис.2.3. Отношения предок/потомок

Отношение предок/потомок, существующее между офисами и работающими в них людьми, в реляционной модели не потеряно; просто оно реализовано в виде одинаковых значений данных, хранящихся в двух таблицах, а не в виде явного указателя. Все отношения, существующие между таблицами реляционной базы данных, реализуются в таком виде.

Столбец одной таблицы, значения в котором совпадают со значениями столбца, являющегося первичным ключом другой таблицы, называется внешним ключом. На рис.2.3 столбец «Подразделение» представляет собой внешний ключ для таблицы «Подразделения». Значения, содержащиеся в этом столбце, представляют собой идентификаторы офисов. Эти значения соответствуют значениям в столбце «Код», который является первичным ключом таблицы «Подразделения». Совокупно первичный и внешний ключи создают между таблицами, в которых они содержатся, такое же отношение предок/потомок, как и в иерархической базе данных.

Внешний ключ, как и первичный ключ, тоже может представлять собой комбинацию столбцов. На практике внешний ключ всегда будет составным (состоящим из нескольких столбцов), если он ссылается на составной первичный ключ в другой таблице. Очевидно, что количество столбцов и их типы данных в первичном и внешнем ключах совпадают.

Если таблица связана с несколькими другими таблицами, она может иметь несколько внешних ключей. Отношения предок/потомок, созданные с помощью нескольких внешних, это те же самые отношения, что и в сетевой базе данных. Как показывает пример, реляционная модель данных обладает всеми возможностями сетевой модели по части выражения сложных отношений.

Внешние ключи являются неотъемлемой частью реляционной модели, поскольку реализуют отношения между таблицами базы данных. К сожалению, как и в случае с первичными ключами, поддержка внешних ключей отсутствовала в первых

реляционных СУБД. Она была введена в системе DB2 Version 2 и теперь имеется во всех коммерческих СУБД.

2.3.3. Правила Кодда

В статье, опубликованной в журнале «Computer World», Тэд Кодд сформулировал двенадцать правил, которым должна соответствовать настоящая реляционная база данных. Двенадцать правил Кодда приведены ниже. Они являются полуофициальным определением понятия реляционная база данных. Перечисленные правила основаны на теоретической работе Кодда, посвященной реляционной модели данных.

1. *Правило информации.* Вся информация в базе данных должна быть предоставлена исключительно на логическом уровне и только одним способом - в виде значений, содержащихся в таблицах.

2. *Правило гарантированного доступа.* Логический доступ ко всем и каждому элементу данных (атомарному значению) в реляционной базе данных должен обеспечиваться путём использования комбинации имени таблицы, первичного ключа и имени столбца.

3. *Правило поддержки недействительных значений.* В настоящей реляционной базе данных должна быть реализована поддержка недействительных значений, которые отличаются от строки символов нулевой длины, строки пробельных символов, и от нуля или любого другого числа и используются для представления отсутствующих данных независимо от типа этих данных.

4. *Правило динамического каталога, основанного на реляционной модели.* Описание базы данных на логическом уровне должно быть представлено в том же виде, что и основные данные, чтобы пользователи, обладающие соответствующими правами, могли работать с ним с помощью того же реляционного языка, который они применяют для работы с основными данными.

5. *Правило исчерпывающего подъязыка данных.* Реляционная система может поддерживать различные языки и режимы взаимодействия с пользователем (например, режим вопросов и ответов). Однако должен существовать по крайней мере один язык, операторы которого можно представить в виде строк символов в соответствии с некоторым четко определенным синтаксисом и который в полной мере поддерживает следующие элементы:

- определение данных;
- определение представлений;
- обработку данных (интерактивную и программную);
- условия целостности;
- идентификация прав доступа;
- границы транзакций (начало, завершение и отмена).

6. *Правило обновления представлений.* Все представления, которые теоретически можно обновить, должны быть доступны для обновления.

7. *Правило добавления, обновления и удаления.* Возможность работать с отношением как с одним операндом должна существовать не только при чтении данных, но и при добавлении, обновлении и удалении данных.

8. *Правило независимости физических данных.* Прикладные программы и

утилиты для работы с данными должны на логическом уровне оставаться нетронутыми при любых изменениях способов хранения данных или методов доступа к ним.

9. *Правило независимости логических данных.* Прикладные программы и утилиты для работы с данными должны на логическом уровне оставаться нетронутыми при внесении в базовые таблицы любых изменений, которые теоретически позволяют сохранить нетронутыми содержащиеся в этих таблицах данные.

10. *Правило независимости условий целостности.* Должна существовать возможность определять условия целостности, специфические для конкретной реляционной базы данных, на подязыке реляционной базы данных и хранить их в каталоге, а не в прикладной программе.

11. *Правило независимости распространения.* Реляционная СУБД не должна зависеть от потребностей конкретного клиента.

12. *Правило единственности.* Если в реляционной системе есть низкоуровневый язык (обрабатывающий одну запись за один раз), то должна отсутствовать возможность использования его для того, чтобы обойти правила и условия целостности, выраженные на реляционном языке высокого уровня (обрабатывающем несколько записей за один раз).

2.3.4. СУБД Microsoft Access и ее реляционная база данных

СУБД Microsoft Access является системой управления реляционной базой данных, включающей все необходимые инструментальные средства для создания локальной базы данных, общей базы данных в локальной сети с файловым сервером или создания приложения пользователя, работающего с базой данных на SQL-сервере.

В качестве учебной системы она выбрана по следующим соображениям.

✓ Во-первых, в Microsoft Access наиболее полно реализована концепция реляционных баз данных в том виде, в котором она была разработана Коддом.

✓ Во-вторых, Microsoft Access является элементом Microsoft Office и, следовательно, его не надо специально приобретать.

✓ В-третьих, Microsoft Access наиболее хорошо работает с другими приложениями Microsoft Office, а также другими форматами баз данных, например популярным до недавнего времени форматом DBase .dbf.

✓ В-четвертых, в Access 2000 получили значительное развитие два технологических направления, составляющих основу корпоративных сетей:

- Технология клиент/сервер, для реализации которой в Access включены средства создания *проекта-приложения*, работающего в качестве клиента баз данных SQL-сервера. Подключение к серверу реализуется с помощью нового интерфейса OLE DB без использования ядра баз данных Microsoft Jet. В Microsoft SQL Server 7.0 этот интерфейс является базовым. Благодаря этому Access становится универсальной основой для построения клиентских приложений, работающих с SQL-сервером.

- Internet-технология, позволяющая эффективно распространять и получать доступ к разнородной информации в глобальных и корпоративных сетях. Эта технология обеспечивает унифицированный доступ к данным

различных приложений в разнородных сетях. Для реализации Internet-технологии в Access включены новые интерактивные средства конструирования Web-страниц доступа к данным в базах Access и SQL-серверов. При этом Web-браузер используется как универсальный интерфейс для доступа и работы с информацией из внешней среды вне зависимости от аппаратно-программной платформы компьютера пользователя и компьютера — источника информации. Страницы могут использоваться подобно формам Access — для ввода и редактирования данных или подобно отчетам Access — для отображения иерархически сгруппированных записей.

К сожалению, система Access не лишена недостатков.

- Во-первых, как и у других программных продуктов фирмы Microsoft наблюдаются ошибки в работе;
- Во-вторых, Access не всегда корректно конвертирует файлы версии 97 года в формат Access 2000. Компилированные файлы (.mde) в принципе читаются только теми версиями, в которых они созданы.
- В-третьих, Access достаточно тяжело работает для сопровождения распределенных баз данных.

Таким образом, для практического использования Access можно порекомендовать в качестве средства информационного обеспечения небольшой фирмы.

Access входит в состав Microsoft Office (в варианты Professional, Premium и Developer) и, как и другие компоненты Office, работает в среде Windows 95, Windows 98, 2000, XP или Windows NT Workstation 4.0 и выше.

Практическим минимумом, предъявляемым Access к персональному компьютеру, является Pentium 75 MHz и 16 Мб оперативной памяти при работе под Windows 95 или Windows 98 или 32 Мб при работе под Windows NT Workstation. При одновременном выполнении нескольких приложений Office необходимо дополнительно для каждого приложения по 4 Мб, а для Access, Outlook и FrontPage по 8 Мб, а для PhotoDraw - 16 Мб. При стандартной установке набора приложений: Word, Excel, Outlook, PowerPoint, Access, FrontPage требуется примерно 250 Мб на жестком диске. В зависимости от конфигурации приложений требования к объему жесткого диска изменятся. Рекомендуется монитор SVGA, возможно использование VGA. При установке приложений Office на локальном компьютере требуется дисковод CD-ROM.

2.3.5. Объекты Access

Access ориентирована на работу с объектами, к которым относятся таблицы базы данных, запросы, а также объекты приложений для работы с базой данных: формы, отчеты, страницы, макросы и модули.

Для типовых процессов обработки данных — просмотра, обновления, поиска по заданным критериям, получения отчетов — в Access имеются средства конструирования форм, запросов, отчетов и страниц. Объекты приложений состоят из графических элементов, называемых элементами управления. Основные элементы управления служат для связи объектов с записями таблиц, являющихся источниками данных.

При создании приложений пользователя также используются средства программирования, реализуемые объектами другого типа — макросами и модулями на

языке программирования Visual Basic for Applications (VBA).

Каждый объект и элемент управления имеет свои свойства, определяя которые, можно настраивать объекты и элементы управления. С каждым объектом и элементом управления связывается набор событий, которые могут обрабатываться макросами или процедурами на VBA.

Объекты представлены в окне базы данных Access. Все операции по работе с объектами базы данных и приложений начинаются в этом окне.

Таблицы (Tables) создаются пользователем для хранения данных об одном информационном объекте модели данных предметной области. Таблица состоит из полей (столбцов) и записей (строк). Каждое поле содержит одну характеристику объекта предметной области. В записи собраны сведения об одном экземпляре этого объекта.

Запросы (Queries) создаются пользователем для выборки нужных данных из одной или нескольких связанных таблиц. Результатом выполнения запроса является таблица, которая может быть использована наряду с другими таблицами БД при обработке данных. Запрос может формироваться в виде запросов по образцу (QBE) или с помощью инструкции SQL — языка структурированных запросов. С помощью запроса можно также обновить, удалить или добавить данные в таблицы или создать новые таблицы на основе уже существующих.

Формы (Forms) являются основным средством создания диалогового интерфейса приложения пользователя. Форма может создаваться для ввода и просмотра взаимосвязанных данных базы на экране в удобном виде, который соответствует привычному для пользователя документу. Формы также могут использоваться для создания панелей управления в приложении.

Отчеты (Reports) предназначены для формирования выходных документов, содержащих результаты решения задач пользователя, и вывода их на печать.

Страницы (Pages) - Страницы Доступа к данным являются диалоговыми Web-страницами, которые поддерживают динамическую связь с базой данных и позволяют просматривать, редактировать и вводить данные в базу работая в окне браузера.

Макросы (Macros). Макрос является программой, которая содержит описание последовательности действий, выполняемых при наступлении некоторого события в объекте или элементе управления приложения. Каждое действие реализуется макрокомандой. Создание макросов осуществляется в диалоговом режиме путем выбора нужных макрокоманд и задания параметров, используемых ими при выполнении.

Модули (Modules) содержат процедуры на языке VBA. Могут создаваться процедуры-функции, которые разрабатываются пользователем для реализации нестандартных функций в приложении пользователя, и процедуры для обработки событий.

В Access 2000 для удобства пользователя объекты базы данных могут быть объединены в группы по функциональному или иному признаку. Группы содержат ссылки на объекты базы данных различных типов.

2.4. Размещение базы данных

Все таблицы базы данных, а также другие объекты Access: формы, запросы, отчеты, макросы и модули, построенные для этой базы, и внедренные объекты могут размещаться на диске в одном файле формата .mdb. Это упрощает технологию ведения базы данных и приложения пользователя. Обеспечивается высокая компактность размещения всех объектов БД на диске и эффективность обработки данных. Страницы доступа к данным Access сохраняются в отдельных файлах, в файле БД размещаются только ссылки на них.

Приложение базы данных, которое содержит программы VBA, может быть скомпилировано и сохранено в файле с расширением .mde. При этом исходные программы на VBA удаляются, а база данных сжимается, что значительно сокращает размер файла. После компиляции объекты БД не могут быть модифицированы.

При работе с базой данных в сети с файловым сервером и размещении на нем базы данных для коллективного использования Access предоставляет возможность отделить от нее объекты, составляющие приложение пользователя, в отдельный файл. Этот файл размещается на всех компьютерах пользователей, которые будут работать с общей базой данных, и приложение можно модифицировать в соответствии с потребностями пользователя.

В Access включены средства разработки проекта — приложения, обеспечивающие работу с базой данных, размещенной на SQL-сервере. Проект размещается в файле .adp на компьютере пользователя. При создании проекта пользователь может создать базу данных на сервере SQL или использовать существующую.

2.5. Интерфейс Access

Access имеет характерный для всех приложений Microsoft Windows удобный графический интерфейс, ориентированный на комфортную работу пользователя. Для работы с таблицами базы данных и другими объектами Access предоставляет многочисленные команды меню и контекстно-зависимые панели инструментов. Поскольку интерфейс приложений Microsoft Office унифицирован, пользователю требуется меньше времени на освоение приложения.

Пользователь имеет возможность переносить объекты БД и их элементы с помощью мыши. Например, любую таблицу или запрос можно перенести из окна базы данных в окно схемы данных. Для установления связи между объектами можно в схеме данных переместить поле из одной таблицы в другую. Для размещения подчиненной формы в главной, достаточно перенести в нее ранее созданную форму или даже просто таблицу — источник в конструируемую форму.

С помощью мыши можно переносить объекты между различными базами данных. При этом необходимо запустить две задачи Microsoft Access. Возможен перенос таблиц и запросов Access в другие приложения, например, в Microsoft Word и Microsoft Excel. Можно выделить данные в форме или в объекте в режиме таблицы и перенести только их. Можно создать таблицу путем переноса с помощью мыши диапазона ячеек Microsoft Excel в окно базы данных Microsoft Access. Объекты других приложений (объекты OLE, Object Linking and Embedding) могут быть перенесены в поле объекта OLE в таблицу или форму в режиме формы, а также в форму или отчет в режиме конструктора.

В Access предусмотрено широкое использование технологии IntelliSense, которая помогает пользователю ориентироваться в выборе необходимых действий и обеспечивает высокую производительность труда пользователя за счет автоматизации выполнения основных функций. Выдача Помощником (Office Assistant) контекстно-зависимой справочной информации помогает принять решение, как лучше выполнить то или иное действие, или найти нужный инструмент в Access. Справочная система Access 2000 построена на основе HTML и поэтому работа с ней не отличается от работы в программах просмотра Internet. К сожалению, в большинстве версий продукта русификация справочной системы неполная. Справочная система Access 97 хорошо русифицирована представляется более удобной. При вызове справки ее окно теперь не заслоняет окно Access, которое автоматически изменяет размер, освобождая пространство для окна справки.

Всплывающие подсказки кратко описывают назначение кнопок панели инструментов при установке курсора мыши на них. Кроме того, предусмотрены более содержательные всплывающие определения. Они появляются при использовании кнопки Контекстная справка (Что это такое?) (What's This) знак вопроса которой можно переносить на любой элемент в окне. В диалоговых окнах для получения всплывающего определения об их элементах используется кнопка называемая Справка.

Пользователь может создать всплывающие подсказки для элементов управления

форм и отчетов. Выполняется это простым заполнением строки свойств этих элементов управления. Кроме того, могут быть созданы контекстные справки и меню, привязанные к элементам управления форм и отчетов.

2.6. Диалоговые средства конструирования объектов

Access предоставляет в распоряжение непрограммирующего пользователя разнообразные диалоговые средства, которые позволяют ему создавать приложения, не прибегая к разработке запросов на языке SQL или к программированию макросов или модулей на языке VBA.

Для автоматизации создания объектов БД - таблиц, запросов по примеру (Query By Example, QBE), схемы базы данных, и объектов приложения (форм, отчетов, страниц) используются специализированные диалоговые средства, называемые конструктором (Design). Конструктор предоставляет пользователю набор инструментов, с помощью которых можно быстро создать и модифицировать объект. Для конструирования макета форм, отчетов и страниц используется панель элементов, которая появляется при вызове конструктора.

Предусмотрено автоматическое конструирование форм, запросов, отчетов, страниц и их элементов с помощью программ-мастеров и команд, начинающихся с приставки «авто».

В Access 2000 для упрощения внесения изменений в объекты базы данных разработана технология интеллектуальной замены имен объектов в базе данных. При этом автоматически исправляются ошибки, вызванные переименованием таблиц, полей, форм, отчетов, запросов, текстовых блоков или других элементов управления. Реализуется за счет того, что каждый именуемый объект (или элемент) базы данных имеет внутренний уникальный идентификатор, имя является только псевдонимом. При переименованиях изменяется лишь псевдоним и при необходимости корректируются все ссылки на объект из других объектов. Для применения этой технологии следует до создания объектов установить соответствующие параметры в разделе **Автозамена имен** (Name AutoCorrect) на закладке **Общие** (General), открываемой через меню **Сервис/Параметры** (Tools/Options).

2.6.1. Мастера баз данных

Особую роль играют мастера баз данных, которые предлагают шаблоны многих типовых приложений. С их помощью можно практически сразу приступить к работе с базой данных выбранного приложения. Для этого после выбора шаблона приложения необходимо выполнить автоматическую генерацию этого приложения. Типовое приложение может быть при необходимости доработано пользователем.

Для предметных областей различных сфер деловой и личной жизни Access содержит шаблоны *типовых баз данных*, включающих все необходимые таблицы, формы, запросы и отчеты. Мастер баз данных создает на основе выбранного шаблона базу данных и приложение в диалоге с пользователем.

Для отображения списка шаблонов типовых БД, с которыми работает мастер, достаточно нажать кнопку **Создать** (New) в окне Access и далее в окне **Создание** (New) выбрать вкладку **Базы данных** (Database).

Мастер, создавая базу данных в соответствии с шаблоном, предлагает включить в таблицы базы данных дополнительные поля, выбрать оформление экрана в формах и отчетах. Пользователю остается только ввести данные.

Мастера баз данных максимально упрощают разработку типовой БД, автоматически создавая ее. Типовые базы данных позволят начинающему пользователю познакомиться с основными принципами построения таблиц БД, связей между ними, получить навыки практической работы в среде Access. Работая с типовой базой, пользователь научится просматривать и изменять данные через формы, создавать запросы для получения сведений из связанных таблиц, готовить отчеты.

Однако, используя типовую базу данных, трудно рассчитывать, что она в полной мере удовлетворит потребности пользователя. Базу данных, созданную мастером, можно изменить и расширить, но эта работа требует от пользователя практически тех же знаний, что и создание новой БД.

2.6.2. Средства программирования

Наряду с диалоговыми средствами создания объектов базы данных и объектов приложения, которые позволяют решить многие задачи пользователя Access располагает мощными средствами программирования для разработки приложений пользователя. Эти средства могут использоваться как для доработки приложений, созданных диалоговыми средствами, так и для реализации сложных задач и создания приложений в целом с необходимым пользователю интерфейсом.

Одним из средств программирования в Access является язык макрокоманд. Программы, созданные на этом языке, называются *макросами* и позволяют легко связывать отдельные действия, реализуемые с помощью форм, запросов, отчетов. Макросы управляются событиями, которые вызываются действиями пользователя при диалоговой работе с данными через формы или системными событиями.

Макросы позволяют разработать меню приложения для выбора и выполнения функциональных компонентов приложения. Простой язык макрокоманд и диалоговая среда разработки макросов позволяют при малой трудоемкости интегрировать объекты приложения и организовать обработку данных.

Наряду с языком макрокоманд Access включает развитую интегрированную среду объектно-ориентированного программирования Visual Basic for Applications (VBA), позволяющую реализовать любые программные решения.

Ключевой идеей объектно-ориентированного программирования (ООП) является объединение данных и оперирующих ими функций в один объект. В VBA база данных рассматривается как совокупность объектов (таблиц, форм, отчетов, их элементов и т. д.), имеющих свойства и методы, реализующие заранее определенные действия над объектами. Структурированность объектов БД упрощает освоение этого языка и создание приложений. Управление выполнением программ в диалоговых приложениях VBA осуществляется в соответствии с событиями, вызываемыми действиями пользователей или системы.

Среда VBA объединяет разнообразные наглядные графические инструменты: редактор VBA, окно разрабатываемого проекта, окно свойств объектов проекта, окно просмотра объектов, отладчик и др. Все инструменты унифицированы и являются общими для всех продуктов Microsoft Office, для Visual Basic, а также продуктов ряда

других фирм. Такая практически единая для различных приложений Office среда позволяет совместно использовать их объекты.

Приложения, разрабатываемые на VBA, могут выполняться только в той среде, в которой поддерживается VBA, в то время как Visual Basic ориентирован на полностью самостоятельную разработку автономно выполняющихся приложений. Язык VBA является производным от самостоятельной системы программирования Visual Basic и имеет с ним много общего. Их синтаксис и интерфейс практически одинаков. Для пользователя, знакомого с программированием на Visual Basic, освоение VBA не вызовет трудностей.

В Access макросы могут быть преобразованы в программы на языке VBA. Это позволяет упростить подготовку программ на VBA, которые затем могут быть усовершенствованы при использовании более мощных средств VBA.

2.6.3. Многопользовательская база данных Access

База данных, как правило, содержит данные, необходимые многим пользователям. Создание многопользовательской БД Access и получение одновременного доступа нескольких пользователей к общей базе данных возможно в локальной одноранговой сети персональных компьютеров или в сети – с файловым сервером.

Сеть обеспечивает аппаратную и программную поддержку обмена данными между компьютерами. Access следит за разграничением доступа разных пользователей к БД и обеспечивает защиту данных при одновременной работе пользователей с общими данными. Автоматически обеспечивается защита данных от одновременной их корректировки несколькими пользователями сети. Разграничение доступа осуществляется в соответствии с правами, предоставленными отдельным пользователям в сетевой операционной системе.

Широкое распространение получили сети, поддерживающие концепцию файлового сервера. База данных Access в такой сети размещается на компьютере, выделенном в качестве *файлового сервера*. СУБД Access может быть установлена или на файловом сервере или на каждой рабочей станции, но выполняется она всегда на рабочей станции пользователя. Обработка данных базы в обоих случаях также осуществляется на *рабочих станциях* пользователей. Для пользователя работа в сети со средствами Access практически не зависит от конфигурации сети и размещения СУБД Access.

Концепция файлового сервера в локальной сети обеспечивается рядом сетевых операционных систем. Наиболее популярными являются Microsoft Windows NT и NetWare Novell. Windows NT имеет версию Windows NT Server, предназначенную для управления файловым и другими серверами сети, и версию Windows NT Workstation, которая устанавливается на рабочей станции. Windows NT Workstation является полностью 32-разрядной операционной системой, под управлением которой могут выполняться различные приложения, в том числе и Microsoft Access. Заметим, что Windows NT Workstation может работать не только на процессорах Intel, но и на ряде Rise-процессоров.

Для обеспечения защиты данных от одновременной корректировки несколькими пользователями сети в Access предусматривается блокировка страниц памяти размером в 4 Кб, при которой блокируются все записи на странице. Access 2000

дополнительно к блокировке страничного уровня поддерживает блокировку на уровне записи. Выбор режима блокировки на уровне записи в текущей базе данных управляется параметром **Блокировка записей при открытии базы данных** (Open databases using record level locking) меню **Сервис/Параметры/Другие** (Tools/Options/Advanced). Если этот флажок сброшен, по умолчанию будет использована блокировка на уровне страницы.

2.6.4. Репликация баз данных

Для пользователей, которые совместно работают с приложением, но не всегда имеют возможность подключаться к сети, Access предлагает использование репликации базы данных. *Репликацией* называют создание специальных копий-реplik общей БД Access, с которыми пользователи могут одновременно работать на разных рабочих станциях. Например, при работе в командировке или дома, когда невозможно подключиться к сети, или когда необходимо уменьшить загрузку сети. Отличие репликации от обычного копирования файлов баз данных заключается в том, что для копий БД возможна синхронизация изменений. При репликации ядро базы данных Microsoft Jet вносит в базу данных ряд изменений.

Преобразование базы данных в реплицированную выполняется командой меню **Сервис/Репликация/Создать реплику** (Tools/Replication/Create Replica). При этом Access присваивает базе данных статус основной *реплики* и создает одну новую реплику.

При проведении сеанса синхронизации изменения, сделанные одним пользователем, могут автоматически вноситься в общую реплику и реплики других пользователей и наоборот. В процессе синхронизации производится обмен обновленными записями и объектами между репликами.

После внесения изменений в реплики становится возможной их синхронизация, которая выполняется с помощью команды **Синхронизация** (Synchronize Now). Если пользователи двух разных реплик по-разному изменили одну и ту же запись, то при синхронизации реплик создается конфликтная таблица. Для того чтобы просмотреть и исправить конфликтующие записи, следует выполнить команду **Устранить конфликты** (Resolve Conflicts).

Могут реплицироваться не все объекты базы данных. Часть объектов может использоваться локально. Часть объектов может реплицироваться группами пользователей. При проведении сеанса синхронизации работа с базой данных может продолжаться.

2.7. Практическое задание

Определить структуру таблиц для создания базы данных, обеспечивающей решение следующих задач (см. л/р №1).

1. Управление документацией.
2. Обслуживание пациентов поликлиники.
3. Регистрация автотранспорта.
4. Прием студентов в учебное заведение.
5. Контроль качества продукции.
6. Изготовление детали (любой на выбор).

7. Управление кадрами.
8. Разработка домашней видеотеки.
9. Организация работы оптовой фирмы.
10. Разработка графика работы персонала предприятия.
11. Инвентаризация оборудования подразделения.
12. Управление измерительным оборудованием.

2.8. Указания по выполнению работы

4. Все атрибуты должны быть распределены в таблицы в соответствии с информационными объектами, которые они характеризуют.
5. Полученная база данных не должна содержать избыточной информации.
6. Результат оформить в виде группы таблиц:

Информационная задача: {привести задание}

Таблица (название)	Атрибут	Статус атрибута
Имя таблицы, в которую входит атрибут	Имя атрибута	Поле связи и/или ключевое поле (если таковым является)

2.9. Контрольные вопросы

8. Определение реляционной базы данных.
 9. Организация таблиц реляционной базы данных.
 10. Отношения предок/потомок в реляционной системе.
 11. Реализация сетевых структур в реляционной базе данных.
 12. Правила Кодда.
 13. Достоинства и недостатки СУБД Access.
 14. Объекты Access.
 15. Мастера баз данных Access.
 16. Средства программирования Access.
- Репликация баз данных.

3. Лабораторная работа №3. Создание базы данных в среде ACCESS

3.1. Цель работы

Выработка навыков создания реляционных баз данных в пользовательском режиме Ms Access.

3.2. Программное обеспечение

Для проведения лабораторной работы требуется Ms Access 97/2000. Рекомендуется Ms Access 2000.

3.3. Начало работы в Microsoft Access. Запуск Access

На компьютере пользователя, который будет работать с СУБД Access, должна быть установлена операционная система Windows 95/98/NT и СУБД Access. Для того чтобы начать работу в СУБД Access, можно, например, после загрузки операционной системы в нижней части рабочего стола на Панели задач нажать кнопку **Пуск** и, открыв в главном меню Windows пункт Программы, выбрать программу Microsoft Access и запустить ее.

3.3.1. Окно Access

Рассмотрим основные элементы окна Access.

3.3.2. Строка заголовка окна

В строке заголовка содержится кнопка системного меню, название приложения — Microsoft Access и три кнопки, управляющие представлением окна на экране.

Кнопка (**Свернуть**, Minimize) позволяет свернуть окно Access. После чего Access наряду с другими выполняющимися программами представляется только в виде кнопки на панели задач Windows:

В свернутом окне Access продолжает работать, и достаточно нажать его кнопку на панели задач, чтобы вернуть окно на экран.

Кнопка (**Развернуть**, Maximize) в окне полноэкранного размера позволяет развернуть окно на весь экран.

Кнопка (**Закреть**, Close) позволяет закрыть окно. При этом работа Access завершается.

В полноэкранном окне кнопка **Развернуть** (Maximize) заменяется кнопкой **Восстановить** (Restore), которая позволяет уменьшить размер окна до размера, предшествующего разворачиванию его на весь экран.

Системное меню может открываться не только щелчком мыши на соответствующей кнопке в строке заголовка, но и нажатием клавиш <Alt>+<Пробел> или <Alt>+<Минус>. Его команды могут использоваться для управления размерами окна, его положением на экране. Системное меню содержит также команду, позволяющую закрыть приложение.

3.3.3. Строка меню

Строка меню содержит несколько пунктов, каждый из которых имеет собственное ниспадающее меню, открываемое щелчком мыши на этом пункте. Ниспадающее меню содержит список команд Access. Команды определяют операции, позволяющие пользователю выполнять нужную работу с помощью графического интерфейса. Каждому средству Access соответствует строка меню и набор операций, дающих пользователю возможность в диалоговом режиме выполнять необходимые действия.

3.3.4. Панели инструментов

Панель инструментов состоит из значков-кнопок, представляющих наиболее часто используемые команды из меню. Кнопки панели позволяют выполнить, не обращаясь к командам, те же операции, но более быстрым и удобным способом.

Набор команд и их графических представлений на панели инструментов является основой интерфейса пользователя и позволяет в диалоговом режиме выполнить основные работы в среде Access.

Устанавливая курсор на кнопках, с помощью всплывающих подсказок можно найти нужную операцию и для ее выполнения щелкнуть на кнопке мышью.

Access имеет набор встроенных панелей инструментов, обеспечивающий удобный интерфейс пользователя при выполнении работ в каждом из режимов базы данных. Эти панели выводятся в окне Access по умолчанию в соответствии с текущим режимом работы. Например, когда активно окно базы данных, выводится панель инструментов **База данных (Database)**. Панель может быть перемещена курсором мыши в любое место окна. Для этого надо установить курсор мыши на свободное от кнопок место на панели и перетащить ее в нужное место. При этом она представляется отдельным окном, которое может находиться в любом месте экрана. В заголовке такого окна панели находится название панели и кнопка **Другие кнопки (More Buttons)**, которая открывает доступ к команде **Добавить или удалить кнопки (Add or Remove Buttons)**. С помощью этой кнопки открывается список доступных для этой панели команд. Двойной щелчок на заголовке перемещенной панели возвращает ее на первоначальное место.

Чтобы получить информацию о кнопке панели, щелкните мышью на кнопке **Справка** и, переместив появившийся знак вопроса на нужную кнопку, снова щелкните мышью. В каждом режиме доступно несколько панелей инструментов. Просмотреть список доступных в текущем режиме панелей и вывести нужные на экран можно через контекстное меню, открываемое щелчком правой кнопки мыши на строке меню или любой выведенной на экран панели. Список доступных панелей инструментов можно получить и по команде **Вид/Панели инструментов (View/Toolbars)**. Команда **Настройка (Customize...)** в списке панелей инструментов открывает окно настройки, в котором на вкладке **Панели инструментов (Toolbars)** представлен список всех панелей Access и созданных пользователем. Для вывода нужной панели на экран достаточно пометить ее строку.

Правая кнопка мыши используется для вызова динамического контекстного меню, содержащего наиболее часто используемые команды, и работает для большинства компонентов Access.

В любой момент пользователю предоставляется возможность настроить панели инструментов по своему усмотрению и изменить состав и вид кнопок на панели.

Настройка производится в соответствующем окне на вкладке **Команды** (Commands). Следует выбрать кнопку одной из категорий и перетащить ее на панель инструментов. Заметим, что панель инструментов, на которую нужно перенести кнопку, должна быть выведена в окне Access. Важно также, чтобы в режиме, для которого предназначена эта панель, могла выполняться функция, реализуемая через эту кнопку. Для удаления кнопки с панели ее нужно переместить в группу кнопок одной из категорий окна настройки.

Выбрав нужную кнопку на панели инструментов или в окне настройки, можно получить ее описание, нажав соответствующую кнопку на вкладке **Команды**. Кнопка **Изменить выделенный объект** (Modify Selection) становится доступной только после выбора одной из кнопок, размещенной на панели инструментов. Нажатием этой кнопки открывается список ее возможных преобразований. При этом можно изменить текст всплывающей подсказки кнопки, выбрать новый значок, отображаемый на ней, отредактировать вид значка, изменить свойства кнопки и т. д.

Для того чтобы отображались всплывающие подсказки кнопок, в окне **Настройка** на вкладке **Параметры** (Options) должен быть установлен флажок **Отображать подсказки для кнопок** (Show Screen Tips on Toolbars).

3.3.5. Строка состояния

Информация, представленная в строке состояния, зависит от того, какие объекты в окне являются текущими. Здесь выводится текст сообщения, связанный с элементом, на котором установлен курсор. Длина отображаемого текста зависит от выбранного для сообщения шрифта и размеров окна Access. Здесь может быть выведен индикатор состояния объекта, индикатор выполнения программы, различные подсказки.

Сообщение "Готово" (Ready) означает, что Access готов принять команду пользователя.

В клетках, представленных в правой части строки состояния, отображается состояние клавиш <Caps Lock>, <Num Lock>, <ScrollLock> и некоторые режимы.

3.3.6. Диалоговые окна

Выполнение практически любой команды начинается с вывода диалогового окна, которое позволяет уточнить операцию, передать параметры для выполнения запрашиваемой команды. При открытом диалоговом окне нельзя перейти к выполнению других действий в данном приложении. Диалоговое окно имеет постоянные размеры, но может быть перемещено в другое место.

3.4. Окно базы данных

После запуска Microsoft Access одновременно с его окном выводится первое диалоговое окно, позволяющее начать создание новой базы данных или открыть существующую.

Диалоговое окно появляется, если в окне **Параметры** (Options), вызываемом по команде меню **Сервис / Параметры** (Tools/Options), на вкладке **Вид** (View) в группе **Отображать** установлен флажок **Окно запуска** (Startup Dialog Box).

Существующую базу данных можно открыть, выбрав ее из списка в диалоговом окне. Если это окно не появляется при запуске Access, для открытия БД выполняется

команда **Файл/Открыть** (File/Open) или нажимается кнопка **Открыть** (Open). После выполнения этой команды открывается окно базы данных.

Все операции по обработке объектов БД начинаются в окне базы данных.

Окно базы данных, как и окно Access, имеет строку заголовка, содержащую слева кнопку системного меню, название — <имя БД> : **база данных** (Database) и справа три кнопки управления размерами окна.

В окне базы данных представлены два раздела:

- Раздел **Объекты** (Objects) с основными типами объектов базы данных: **Таблицы** (Tables), **Запросы** (Queries), **Формы** (Forms), **Отчеты** (Reports), **Страницы** (Pages), **Макросы** (Macros), Модули (Modules).

- Раздел **Группы** (Groups), где создаются пользовательские группы объектов, предназначенные для хранения ссылок на объекты различных типов, объединенные, например, по функциональному назначению.

Рабочее поле окна базы данных предназначено для отображения списка объектов выбранного типа. Кроме того, в Access 2000 здесь размещены ярлыки, открывающие возможность сразу приступить к созданию объекта в основных режимах: в режиме конструирования и с помощью мастера.

В этом окне представлен ряд кнопок:

- Первой является кнопка **Открыть** (Open), если выбрана таблица, форма или запрос, либо кнопка **Просмотреть** (Preview), если выбран отчет, либо кнопка **Запустить** (Run), если выбран макрос

- Вторая кнопка — **Конструктор** (Design) — позволяет перейти в режим доработки любого ранее созданного объекта

- Третья кнопка — **Создать** (New) — позволяет приступить к созданию нового объекта любого выбранного типа

Остальные кнопки окна предназначены для удаления выбранного объекта и выбора вида отображения объектов в окне.

При создании объекта предоставляется возможность выбора режима его разработки. Это может быть мастер или конструктор или некоторый другой режим, зависящий от выбранного типа объекта.

При открытии окна БД по умолчанию выводится панель инструментов **База данных** (Database). Эта же панель инструментов выводится в окне Access до открытия базы данных, но большинство ее кнопок являются недоступными.

3.5. Создание новой базы данных

Создание новой нормализованной реляционной базы данных Access осуществляется в соответствии с ее структурой, полученной в результате проектирования. Структура реляционной базы данных определяется составом таблиц и их взаимосвязями. Взаимосвязи между двумя таблицами типа один-ко-многим или один-к-одному реализуются через ключ связи, входящий в состав полей связываемых

таблиц.

Создание реляционной базы данных с помощью системы управления базами данных (СУБД) начинается с формирования **структуры таблиц**. При этом формируется состав полей и задается их описание. После определения структуры таблиц создается схема данных, в которой устанавливаются связи между таблицами. Access запоминает и использует эти связи при заполнении таблиц и обработке данных.

При создании базы данных (БД) важно задать параметры, в соответствии с которыми Access будет автоматически поддерживать целостность данных. Для этого при определении структуры таблицы должны быть указаны ограничения на допустимые значения данных, а при создании схемы данных на основе нормализованных таблиц должны быть заданы параметры поддержания целостности связей БД.

Создание таблицы завершается процедурой загрузки, т.е. заполнением таблиц конкретными данными. Особое значение имеет технология загрузки взаимосвязанных данных. При этом пользователь может начинать работу с базой при любом количестве созданных таблиц еще до создания полной базы, отображающей все объекты модели данных предметной области. База данных может создаваться поэтапно, и в любой момент ее можно дополнять новыми таблицами и вводить связи между ними в схему данных.

После запуска Microsoft Access выводит первое диалоговое окно, позволяющее начать создание БД выбором параметра **Новая база данных** или открыть ранее созданную БД из предложенного списка. Если окно не выводится, то можно начать создание БД с помощью команды **Файл/Создать** или кнопки Создать на панели инструментов **База данных**.

В этом случае Access выведет окно **Создание**.

Для создания одной из типовых БД с помощью мастера на основе существующих шаблонов необходимо выбрать вкладку **Базы данных**. На закладке представлены шаблоны таких БД. Для запуска мастера достаточно щелкнуть на значке нужной БД.

Вкладка **Общие** содержит значок **База данных**, позволяющий приступить к созданию новой оригинальной БД.

Access 2000 дополнен по сравнению со старыми версиями новыми средствами, предназначенными для создания *проекта приложения*, работающего с БД SQL-сервера, и *страниц доступа к данным* Access. На вкладке **Общие** имеются соответствующие значки **Проект (новая база данных)**, **Проект (существующая база данных)**, **Страница доступа к данным**.

Проект (новая база данных) и **Проект (существующая база данных)** позволяют работать с проектом-приложением пользователя, которое работает с БД, размещенной на SQL-сервере. Причем *проект* можно создать для уже существующей на сервере БД или одновременно с проектом начать создание новой БД.

Страница доступа к данным позволяет создавать Web-страницы специального типа, предназначенные для просмотра и работы с данными в базах Microsoft Access или Microsoft SQL Server из Internet или intranet. Страница доступа к данным сохраняется в отдельном файле вне БД Microsoft Access. При создании этого файла в окне БД на вкладке **Страницы** Access автоматически создает ярлык к этому файлу. Разработка страницы ведется в интерактивном режиме средствами, похожими на те,

что используются при конструировании форм или отчетов, однако и в разработке и во взаимодействии со страницами имеются существенные отличия.

Страницы могут быть различного типа. Одни позволяют только просматривать информацию из баз данных, обеспечивая при этом сортировку и фильтрацию данных. Причем информация эта является статической, опубликованной на Web-странице в момент ее создания. Поскольку такие страницы не имеют непосредственной связи с базой, отображаемые данные могут быть недостаточно актуальными, а их изменение невозможно. Другие страницы обеспечивают просмотр данных, которые получаются из базы динамически именно в момент открытия страницы. Получение таких актуальных данных обеспечивается за счет инструкций SQL, размещаемых на этих страницах и обрабатываемых Web-сервером, который обращается к базе за данными. Кроме того, возможно создание страниц, которые обеспечивают не только просмотр актуальной информации, но и ее редактирование, удаление и добавление записей в БД.

Можно открыть страницу доступа к данным и работать через нее со связанной базой данных с помощью программы MS Internet Explorer. Кроме того, возможна работа со страницей доступа к данным непосредственно в Access. Страницы могут использоваться в приложении БД наряду с формами и отчетами.

Создание доступа к данным, открытие их в режиме конструктора или для просмотра в Access, а также просмотр и работа со страницами в Internet и intranet возможны только при наличии на компьютере MS Internet Explorer 5.

3.6. Создание файла базы данных Access

Access хранит все таблицы БД, а также другие объекты в одном файле. Прежде чем приступить к созданию таблиц БД, необходимо создать файл пустой БД.

Для создания файла новой БД надо в окне **Создание** выбрать вкладку **Общие**, дважды щелкнуть на значке **База данных**. В раскрывающемся списке **Папка** появившегося окна **Файл новой базы данных** нужно выбрать каталог, в котором будет размещен файл, задать имя файла новой базы данных и нажать кнопку **Создать**. В результате открывается окно новой БД **<имя БД>: база данных**.

Задавая имя БД, следует иметь в виду, что его предельная длина составляет 255 символов, включая пробелы. Имена файлов не должны содержать следующих символов: \ ? : * " < > | .

Тип файла по умолчанию имеет значение *Базы данных Microsoft Access* (расширение *.mdb, Microsoft Access Databases). Это расширение является зарегистрированным в Windows для данного типа файлов и связывается с программой Access.

Задание 1.

Открыть учебную базу данных «Борей.mdb» (устанавливается вместе с Ms Access). Просмотреть возможности базы данных по созданию таблиц, запросов, форм и отчетов. Расположение базы данных: c:\Program Files\Microsoft Office\Office\Samples\Борей.mdb

3.7. Окно файла новой базы данных

В окне **Файл новой базы данных** вертикальный ряд больших кнопок предназначен для быстрого открытия доступных папок. Кнопка **Журнал** открывает служебную папку, в которой создаются и сохраняются ярлыки к последним открывавшимся документам. Для отображения в папке только файлов БД в окне списка **Тип файла** устанавливается тип **Базы данных Microsoft Access (*.mdb)**. Кнопки **Мои документы**, **Рабочий стол**, **Избранное** позволяют просмотреть содержимое соответствующих папок. В папку **Избранное** целесообразно помещать те файлы и папки (каталоги), с которыми приходится постоянно работать. Делается это командой **Добавить в папку избранное** в разделе **Сервис** на панели управления при выбранной соответствующей папке или файле.

Кнопки на панели инструментов окна обеспечивают удобный поиск и переход к папке, в которой необходимо создать файл базы данных. Назначение каждой из кнопок можно прочитать во всплывающей подсказке при наведении на нее указателя мыши.

3.8. Окно базы данных

В результате выполнения команды **Создать** открывается окно новой БД **<имя БД>: база данных**. Причем **<имя БД>** соответствует заданному в окне **Файл новой базы данных**.

В окне новой базы данных в разделе **Объекты** вертикальным рядом кнопок представлены все объекты, которые могут быть созданы в БД: таблицы, запросы, отчеты, страницы, макросы и модули. При нажатии кнопки в рабочем поле окна отображается список имен объектов данного типа. При создании новой БД список для любого выбранного типа объекта отсутствует.

Объекты различных типов могут объединяться в группы, которые представлены в разделе **Группы**. Группы позволяют в больших БД объединить объекты одной темы. Изначально в разделе **Группы** существует единственная группа **Избранное**. Для создания новой группы необходимо нажать правую кнопку на строке этой группы и выбрать из контекстного меню команду **Новая группа**. Для внесения объектов в группу выделяется нужный объект, вызывается контекстное меню, выбирается команда **Добавить в группу** и в ней группа, в которую включается объект. Объекты представлены в группе ярлыками, ссылающимися на включенный в группу объект. При выполнении этой команды также можно создать новую группу.

3.9. Создание таблицы базы данных

Создание таблицы БД состоит из двух этапов. На первом этапе определяется ее структура: состав полей, их имена, последовательность размещения полей в таблице, тип данных каждого поля, размер поля, ключи, индексы таблицы и другие свойства полей. На втором этапе производится создание записей таблицы и заполнение их данными.

Для создания новой таблицы надо в окне базы данных выбрать объект **Таблицы** и нажать кнопку **Создать**. В открывшемся окне **Новая таблица** нужно выбрать один из режимов создания таблицы. В Access 2000 основные первые три режима вынесены в рабочее поле, предназначенное для отображения списка таблиц. Это позволяет сразу перейти в нужный режим создания таблицы, сократив число выполняемых

пользователем операций.

Строка Создание таблицы в режиме конструктора в рабочем поле окна БД или Конструктор в окне Новая таблица определяет выбор основного способа создания новой таблицы, при котором создание таблицы начинается с определения ее структуры в режиме конструктора таблиц. В режиме конструктора пользователь может сам установить параметры всех элементов структуры таблицы.

3.9.1. Определение структуры новой таблицы в режиме конструктора

При выборе *режима конструктора таблиц* появляется окно **Таблица1:Таблица**, в котором определяется структура таблицы базы данных. При переходе в режим конструктора таблиц меняется состав команд меню, а панель инструментов БД заменяется на панель инструментов **Конструктор таблиц**.

3.9.2. Определение полей таблицы

Для определения поля в окне **Таблица** задаются **Имя поля**, **Тип данных**, **Описание** – краткий комментарий, а также свойства поля в разделе **Свойства поля**. На вкладке **Общие** представлены строки свойств поля, в том числе максимальный размер, подпись, которая выводится в заголовке столбца, значение по умолчанию и др. На вкладке **Подстановка** выбирается *тип элемента управления*: поле, список или поле со списком.

3.9.3. Имена полей и типы данных

❖ **Имя поля.** Каждое поле в таблице должно иметь уникальное имя, удовлетворяющее соглашениям об именах объектов в Access. Оно является комбинацией из букв, цифр, пробелов и специальных символов, за исключением символов . ! ' []. Имя не может начинаться с пробела и содержать управляющие символы с кодами ASCII от 00 до 31. Максимальная длина имени 64 символа.

❖ **Тип данных.** Тип данных определяется значениями, которые предполагается вводить в поле, и операциями, которые будут выполняться с этими значениями. В Access допускается использование девяти типов данных. Список возможных типов данных вызывается нажатием кнопки списка при выборе типа данных каждого поля:

➤ **Текстовый** – тип данных по умолчанию. Допускается ввод текста или цифр, не участвующих в расчетах. Число символов в поле не должно превышать 255. Максимальное число символов, которое можно ввести в поле, задается в свойстве **Размер поля**. Пустые символы в неиспользуемой части поля не сохраняются.

➤ **Поле МЕМО.** Длинный текст, например, некоторое описание или примечание. Максимальная длина 64 000 символов.

➤ **Числовой.** Числовые данные, используемые в математических вычислениях. Конкретные варианты числового типа и их длина задаются в свойстве **Размер поля**.

➤ **Денежный.** Денежные значения и числовые данные, используемые в расчетах, проводящихся с точностью до 15 знаков в целой и до 4 знаков в дробной части. Длина поля 8 байт. При обработке числовых значений из

денежных полей выполняются вычисления с фиксированной точкой, более быстрые, чем вычисления для полей с плавающей точкой, кроме того, при вычислениях предотвращается округление. В свете сказанного для полей, в которых планируется хранить числовые значения с указанной точностью, рекомендуется использовать денежный тип данных.

➤ **Дата/время.** Значения даты или времени, относящиеся к годам с 100 по 9999 включительно. Длина поля 8 байт.

➤ **Счетчик.** Тип данных поля, в которое для каждой новой записи автоматически вводятся уникальные целые, последовательно возрастающие на 1, или случайные числа. Значения этого поля нельзя изменить или удалить. Длина поля 4 байта для длинного целого, для кода репликации – 128 байт. По умолчанию в поле вводятся последовательные значения. В таблице не может быть более одного поля этого типа. Используется для определения уникального ключа таблицы.

➤ **Логический.** Логические данные, которые могут иметь одно из двух возможных значений Да/Нет; Истина/Ложь; Вкл./Выкл. Длина поля 1 бит.

➤ **Поле объекта OLE.** Объект (например, электронная таблица Microsoft Excel, документ Microsoft Word, рисунок, звукозапись и другие данные в двоичном формате), *связанный* или *внедренный* в таблицу Access. Длина поля – до 1 Гигабайта (ограничивается объемом диска). Для полей OLE и MEMO не допускается сортировка и индексирование.

➤ **Гиперссылка.** В качестве гиперссылки можно указывать путь к файлу на жестком диске, путь UNC или адрес URL. Если щелкнуть мышью на поле гиперссылки, Access выполнит переход на соответствующий объект, документ, страницу Web или другое место назначения. Максимальная длина 64 000 символов.

➤ **Мастер подстановок...** Выбор этого типа данных запускает мастера подстановок. Мастер строит для поля список значений на основе полей из другой таблицы. Значения в такое поле будут вводиться из одного из полей списка. Возможно также определение *поля со списком постоянных значений*.

3.9.4. Общие свойства поля

Общие свойства задаются для каждого поля на вкладке **Общие** и зависят от выбранного типа данных. Для отображения свойств поля необходимо установить курсор на строке соответствующего поля. К наиболее важным свойствам полей следует отнести:

❖ **Размер поля** задает максимальный размер данных, сохраняемых в поле. Для поля с типом данных **Текстовый** задается размер от 1 до 255 байтов (по умолчанию 50 байт). Для поля с типом данных **Счетчик** можно задать:

- **Длинное Целое** – 4 байта.
- **Код репликации** – 128 байт.

Для поля с типом данных **Числовой** можно задать:

- **Байт** для целых чисел от 0 до 255, длина поля 1 байт.
- **Целое** для целых чисел от –32.768 до +32.767, занимает 2 байта.

- **Длинное целое** для целых чисел от $-2.147.483.648$ до $+2.147.483.647$, занимает 4 байта.
- **Дробные с плавающей точкой 4 байта** для чисел от $-3,4 \times 10^{38}$ до $+3,4 \times 10^{38}$ с точностью до 7 знаков.
- **Дробные с плавающей точкой 8 байт** для чисел от $-1,797 \times 10^{308}$ до $+1,797 \times 10^{308}$ с точностью до 15 знаков.
- **Действительное** для целых чисел от $-10^{38} - 1$ до $10^{38} - 1$ (при работе с проектами, которые хранятся в файлах типа *.adp) и от $-10^{28} - 1$ до $10^{28} - 1$ (*.mdb) с точностью до 28 знаков, занимает 12 байт.
- **Код репликации.** Глобальный уникальный идентификатор, занимает 16 байт. Поля такого типа используются Access для создания системных уникальных идентификаторов реплик, наборов реплик, таблиц, записей и других объектов при репликации баз данных.

Рекомендуется задавать минимально допустимый размер поля, который понадобится для сохраняемых значений, так как сохранение таких полей требует меньше памяти, и обработка данных меньшего размера выполняется быстрее. Изменение размера поля с большего на меньший в таблице, имеющей данные, может привести к их искажению или полной потере. Изменения в данных, которые происходят вследствие изменения свойства **Размер поля**, нельзя отменить после их сохранения в конструкторе таблиц.

❖ **Формат поля** является форматом отображения заданного типа данных при выводе их на экран или на печать. В Access определены встроенные стандартные форматы отображения для полей с типами данных Числовой, Дата/время, Логический и Денежный. Ряд этих форматов совпадает с настройкой национальных форматов, определяемых в окне Язык и стандарты панели управления Microsoft Windows. Пользователь может создать собственный формат для всех типов данных, кроме OLE с помощью символов форматирования. Для указания конкретного формата отображения необходимо выбрать в раскрывающемся списке одно из значений **Формат поля**. Формат поля используется для отображения данных в режиме таблицы, а также применяется в форме или отчете при отображении этих полей.

❖ **Маска ввода** используется для форматирования данных и управления вводимыми значениями. Маска ввода состоит из текстовых символов (таких как точки, тире, скобки), разделяющих пустые интервалы, предназначенные для заполнения. Свойство **Маска ввода** состоит из текстовых и специальных символов, определяющих тип значений, которые могут быть введены в данную конкретную позицию. В основном маски ввода используются в текстовых полях и полях даты/времени, а также в числовых и денежных.

В приведенной ниже таблице указаны описания некоторых масок ввода и примеры значений, которые в них могут быть введены.

Описание маски ввода	Примеры введенных значений
(000) 000-0000	(206) 555-0248
(999) 999-9999!	(206) 555-0248
	() 555-0248
(000) AAA-AAAA	(206) 555-TELE
#999	-20
	2000
>L????L?000L0	GREENGR339M3
	МАЙ Р 452Ю7
>L0L 0L0	T2Ф 8M4
00000-9999	98115- 98115-3007
>L<??????????????	Мария
	Иван
ISBN 0-#####-0	ISBN 1-55615-507-7
	ISBN 0-13-964262-5
>LL00000-	DB51392-0493

Определение маски ввода может состоять из трех разделов, разделенных знаком точка с запятой, например, (999) 000-0000!;0;" ". Первый раздел – собственно маска ввода, второй указывает, следует ли сохранять текстовые символы.

0 означает, что текстовые символы сохраняются вместе с введенными значениями,

1 или пустое значение – сохраняются только введенные символы.

Третий раздел – это символ, выводящийся в маске ввода на месте пустых символов. Допускается использование любого символа; для отображения пробела. Если данный раздел описания оставить пустым, то для представления пустых символов используется символ подчеркивания “_”.

В приведенной ниже таблице указано, как Microsoft Access интерпретирует символы, содержащиеся в первой части описания в свойстве Маска ввода (InputMask). Чтобы включить в маску текстовые константы, отличные от представленных в таблице, в том числе символы и пробелы, следует просто ввести их в нужную позицию. Чтобы включить один из следующих символов в качестве текстовой константы, необходимо перед ним ввести символ обратной косой черты \.

Символ	Описание
0	Цифра (от 0 до 9, ввод обязателен; символы плюс [+] и минус [-] не допускаются).
9	Цифра или пробел (ввод не обязателен; символы плюс и минус не допускаются).
#	Цифра или пробел (ввод не обязателен; пустые символы преобразуются в пробелы, допускаются символы плюс и минус).
L	Буква (от A до Z или от A до Я, ввод обязателен).
?	Буква (от A до Z или от A до Я, ввод не обязателен).

- A Буква или цифра (ввод обязателен).
- a Буква или цифра (ввод необязателен).
- & Любой символ или пробел (ввод обязателен).
- C Любой символ или пробел (ввод необязателен).

., : ; - / Десятичный разделитель и разделители тысяч, значений дат и времени. (Отображаемый символ зависит от настроек языка и стандартов на панели управления Windows.)

< Указывает перевод всех следующих символов на нижний регистр.

> Указывает перевод всех следующих символов на верхний регистр.

! Указывает заполнение маски ввода справа налево, а не слева направо. Заполнение маски символами всегда происходит слева направо. Восклицательный знак в маске ввода можно помещать в любую позицию.

\ Указывает ввод любого следующего символа в качестве текстовой константы. Используется для отображения всех перечисленных в данной таблице символов как текстовых констант (например, \A выводится как символ «А»).

Пароль Значение **Пароль**, заданное для свойства **Маска ввода**, создает поле для ввода пароля. Любой символ, введенный в поле, сохраняется как символ, но отображается как звездочка (*).

❖ **Число десятичных знаков** задает для числового и денежного типов данных число знаков после запятой. Можно задать число от 0 до 15. По умолчанию (значение **Авто**) это число определяется установкой в свойстве **Формат поля**. Следует иметь в виду, что установка этого вида не действует, если свойство **Формат поля** не установлено или выбрано значение **Основной**. Свойство **Число десятичных знаков** влияет только на количество десятичных знаков, отображаемых на экране, и не влияет на число сохраняемых десятичных знаков. Для изменения числа сохраняемых знаков нужно изменить свойство **Размер поля**.

❖ **Подпись** поля задает текст, который выводится в таблицах, формах, отчетах.

❖ **Условие на значение** позволяет осуществлять контроль ввода, задает ограничения на вводимые значения, при нарушении условий запрещает ввод и выводит текст, заданный свойством **Сообщение об ошибке**.

❖ **Сообщение об ошибке** задает текст сообщения, выводимый на экран при нарушении ограничений, заданных свойством **Условие на значение**.

3.9.5. Тип элемента управления

На вкладке **Подстановка** в окне конструктора таблиц задается свойство **Тип элемента управления**. Это свойство определяет, будет ли отображаться поле в таблице и в форме в виде **Поля**, **Списка** или **Поля со списком**. Таким образом определяется вид элемента управления, используемого по умолчанию для отображения поля.

Если для поля выбран тип элемента управления **Список** или **Поле со списком**, на вкладке **Подстановка** появляются дополнительные свойства, которые определяют источник данных для строк списка и ряд других характеристик списка. В качестве источника данных для списка выбирается таблица, с которой осуществляется

постоянная связь, что обеспечивает актуальное состояние списка.

3.9.6. Определение первичного ключа

Каждая таблица в реляционной БД должна иметь уникальный первичный ключ, который может быть простым или составным, включающим несколько полей (до 10). Для определения ключа необходимо выделить поля, составляющие ключ, и на панели инструментов **Конструктор таблиц** нажать кнопку **Ключевое поле** либо выполнить команду меню **Правка/Ключевое поле**.

Для ключевого поля автоматически строится индекс. В этом можно убедиться, посмотрев информацию об индексах таблицы. Окно **Индексы** вызывается щелчком на кнопке просмотра и редактирования индексов **Индексы** на панели инструментов или выполнением команды меню **Вид/Индексы**. Индекс ключевого поля всегда уникален и не допускает пустых полей в записях.

Если первичный ключ не установлен пользователем до сохранения вновь созданной таблицы, Access спросит о необходимости создания первичного ключа. При ответе «Да» Access создаст первичный ключ с типом данных **Счетчик**.

3.10. Сохранение таблицы

После определения структуры таблицы ее надо сохранить. Для этого можно воспользоваться командой меню **Файл/Сохранить** или кнопкой панели инструментов конструктора **Сохранить**. В появившемся окне **Сохранение** вводится имя таблицы.

При сохранении таблицы происходит обновление файла базы данных, в которую помещается созданная таблица.

После сохранения таблицы становится доступным режим, позволяющий перейти ко второму этапу создания таблицы – созданию записей. Переход в этот режим, называемый *режимом таблицы*, возможен только после сохранения структуры таблицы и осуществляется нажатием кнопки **Вид** на панели инструментов конструктора таблиц или выбором этого режима при открытии списка на этой кнопке.

Задание 2

Создать базу данных «Предприятие». Создать структуру таблицы «Работники» базы данных «Предприятие» в режиме конструктора (см. ниже).

Имя поля	Ключевое поле	Обязательное поле	Тип данных	Размер	Число десятичных знаков	Подпись поля
Подр	Да	Да	Текстовый	10		Подразделение

Таб№	Да	Да	Тексто- вый	3		Табельный номер в подразделе нии
ФИО		Да	Тексто- вый	15		Ф.И.О.
ГОД РОЖД		Нет	Число- вой	Це- лое		Год рождения
Образование		Нет	Тексто- вый	25		
Оклад		Нет	Число- вой	С плав. Точ- кой 4 байта	2	Оклад

Перейти в режим таблицы, заполнить ее следующими данными.

Подразде- ление	Таб номер в подраз- делении	Ф.И.О.	Год рождения	Образова- ние	Оклад
ОГМ	01	Аристов Р.П.	1964	Высшее проф.	3000
ОГМ	02	Бондаренко С.А.	1973	Среднее	2500
ОГМ	03	Борисова Е.И.	1971	Среднее проф.	3000
ОГМ	04	Макова Н.В.	1977	Начальное проф.	2850
ОК	01	Боярская Н.П.	1969	Высшее проф.	3500
ОК	02	Федоров Д.К.	1966	Высшее проф.	4000
ОК	03	Сидоров И.Р.	1974	Высшее проф.	2780
УКСС	01	Андреев Г.М.	1970	Высшее проф.	4560
УКСС	02	Петров О.К.	1975	Высшее проф.	3980
Цех № 4	01	Иванов К.К.	1960	Началь ное проф.	36 00

Добавить поле «Паспорт» самостоятельно определив его параметры и с использованием маски ввода.

3.11. Создание новой таблицы в режиме таблицы

В режиме таблицы пользователь может создать таблицу, не определяя предварительно ее структуры.

Создание новой таблицы в режиме таблицы осуществляется выбором строки **Создание таблицы путем ввода данных** в рабочем поле окна базы данных или строки **Режим таблицы** в окне **Новая таблица**.

После выбора этого режима сразу открывается пустая таблица, в которую можно ввести данные, при сохранении этой таблицы Access проанализирует данные и автоматически присвоит соответствующий тип данных каждому полю, т.е. автоматически создаст структуру таблицы. Таблица, предлагаемая Access в рассматриваемом режиме для заполнения данными имеет 10 столбцов и 21 строку. Полям таблицы по умолчанию присваиваются имена Поле1, Поле2 и т.д.

Любое поле этой таблицы можно переименовать в соответствии с требованиями пользователя, непосредственно редактируя имена в заголовке столбцов. Для этого нужно дважды щелкнуть мышью на области выделения столбца, содержащей его имя. Тот же эффект достигается командой **Формат/Переименовать столбец**.

Если требуется создать таблицу, содержащую более 20 полей, то можно вставить новые столбцы. Для этого следует перейти в столбец, слева от которого требуется вставить новый столбец, и выполнить команду меню **Вставка/Столбец**.

Для удаления определенного столбца следует пометить его, щелкнув мышью на заголовке, и выполнить команду **Правка/Удалить столбец**. Для перемещения столбца нужно пометить его и, оставив курсор на заголовке, нажать кнопку мыши и, удерживая его, перетащить столбец в нужное место.

В каждый столбец вводятся данные определенного типа. При вводе данных, для которых определены стандартные форматы, необходимо использовать форматы из установленного для этого типа списка. Это позволяет Access автоматически определить тип данных. Свойство **Формат поля** остается пустым и при отображении данных используется формат по умолчанию. Все столбцы, оставленные пустыми, будут удалены при сохранении таблицы.

По окончании ввода данных во все нужные столбцы необходимо сохранить таблицу, выполнив соответствующую команду.

При сохранении таблицы выводится приглашение для создания ключевого поля. При выборе **Да** в таблицу будет добавлено ключевое поле типа **Счетчик**. Если в таблице имеются поля с однозначной идентификацией каждой записи, то целесообразно одно из таких полей сделать ключевым. В этом случае нужно нажать **Нет** и самостоятельно назначить ключ в режиме конструктора.

Только после сохранения таблицы Access создает ее структуру и делает возможным переход в режим конструктора таблиц. Тип данных каждого поля определяется форматом введенных данных. Если в дальнейшем требуется изменить определение поля, например, задать другой тип данных, формат, указать значения по

умолчанию, создать маски ввода или определить условие на значение, добавить новые поля, то эти действия можно выполнить в режиме конструктора.

Вне зависимости от способа создания таблицы режим конструктора позволяет в любой момент изменить структуру таблицы.

Задание 3

В режиме таблицы создать таблицу «Подразделения» базы данных «Предприятие»

Аббревиатура	Численность работающих	Руководитель	Площадь помещений
ОГМ	30	Железнов А.В.	100,25
ОК	32	Сальникова Е.М.	130,3
УКСС	15	Шитов А.А.	50,6
Цех №4	150	Асадов К.Н.	600
Цех №16	80	Петров И.И.	480,5
БРИЗ	10	Соколов П.А.	50,5
ОТК	30	Викторов К.К.	80,25
Бухгалтерия	40	Воробьев П.И.	150,4
ОГК	100	Кузнецов Е.Н.	450,8

3.12. Создание таблицы с помощью мастера таблиц

Мастер таблиц автоматически создаст таблицу по одному из шаблонов. Начать работу мастера можно выбрав строку Создание таблицы с помощью мастера в рабочем поле объекта **Таблицы** окна **База данных**. Для создания таблицы с помощью мастера можно также выбрать Мастер таблиц в окне **Новая таблица**. Мастер автоматически создает таблицу по одному из шаблонов. Пользователю предлагается для выбора более 50 образцов таблиц, предназначенных для использования в различных целях. Каждая таблица шаблона содержит соответствующий набор полей, из которых пользователь может выбрать нужные для включения в создаваемую таблицу.

Включаемые в таблицу поля могут быть переименованы. Мастер определит ключ таблицы. Может быть создана связь новой таблицы с уже существующими в базе данных. При этом ключ новой таблицы будет включен в таблицу, с которой устанавливается связь. По запросу пользователя мастер создает форму для ввода данных в таблицу. После создания таблицы мастером можно в любое время доработать структуру таблицы в режиме конструктора.

Задание 4

Создайте таблицу в режиме мастера на интересующую Вас тематику и введите в нее 6 – 7 записей.

3.13. Непосредственный ввод данных в таблицу

Непосредственный ввод данных в таблицу осуществляется в *режиме таблицы*. Переход в этот режим из окна базы данных выполняется нажатием кнопки **Открыть** при выделенной таблице не рабочем поле окна базы данных. Переход в режим таблицы из режима конструктора выполняется нажатием кнопки **Вид** на панели инструментов или выполнением команды **Вид/Режим таблицы**.

В режиме таблицы предоставляется возможность вводить новые записи в таблицу, заполняя ее значениями ее поля. Дополнение таблицы новыми записями и редактирование записей разрешено по умолчанию, т.к. по умолчанию пользователь имеет имя Admin и ему предоставлены все *права доступа* к объектам базы данных:

- ❖ *Права доступа к данным таблицы*. Просмотреть и изменить права доступа можно по команде **Сервис/Защита/Разрешения**.

- ❖ *Режим ввода записи*. Выполнение команды **Записи/Ввод данных** позволяет перейти в режим, при котором видна только вводимая запись. Возврат к просмотру всей таблицы по команде **Записи/Удалить фильтр**.

- ❖ *Требования корректности вводимых данных*. При заполнении таблиц, для связей между которыми не определены параметры целостности в схеме данных, обеспечение корректности вводимой информации зависит только от пользователя. Так, например, при одно-многочисленных связях таблиц и вводе записей в подчиненную таблицу необходимо отслеживать наличие записей с вводимыми значениями ключевых полей в главной таблицы. При изменении или удалении ключевых полей в записях главной таблицы необходимо изменять или удалять связанные с ними записи в подчиненных таблицах.

Вводимые в поля таблицы значения данных должны соответствовать типам данных, определенным в структуре, а способ их записи – допустимым в текущий момент форматам данных. Значения должны удовлетворять накладываемым ограничениям. После ввода значения в ячейку поля при попытке перейти к другой ячейке Access проверяет, являются ли введенные данные допустимыми для этого поля. Если введено значение, не соответствующее типу данных поля, Access пытается преобразовать его в правильный тип данных. Если значение не является допустимым и преобразование невозможно, например, нельзя преобразовать текст в число, появляется предупреждающее сообщение. Для того, чтобы выйти из ячейки следует ввести правильное значение или отменить внесенные изменения.

3.14. Макет таблицы

Для удобства работы с таблицей можно изменить ее представление на экране. При этом можно менять ширину столбца, высоту строки, шрифт данных таблицы, цвет текста, линий сетки и фона, оформление, которое может быть обычным, приподнятым или утопленным. Можно выводить на экран только те столбцы, которые нужны для текущей работы, можно зафиксировать столбец при просмотре широких таблиц. Эти параметры отображения таблицы на экране называются *макетом* таблицы и сохраняются вместе с ней.

Настройка макета выполняется в режиме таблицы. При этом могут быть использованы команды меню **Формат** или панель инструментов **Формат** (режим таблицы), которую можно вызвать из контекстного меню на доступной панели

инструментов.

Многие операции настройки макета можно выполнить непосредственно в таблице с помощью мыши:

✓ *Изменение ширины столбца.* Курсор мыши устанавливается на линию, разделяющую имена столбцов. При этом он превращается в планочку со стрелками в обе стороны. Далее границу столбца можно перетащить в нужное место.

✓ *Изменение высоты строки.* Курсор мыши устанавливается в области маркировки записи, расположенной слева, на границе между записями. Граница строки перетаскивается на требуемое расстояние. При этом меняется высота всех строк таблицы.

✓ *Удаление столбцов с экрана.* Убрать столбец можно, перетащив его правую границу влево до исчезновения столбца.

✓ *Изменение порядка расположения столбцов на экране.* Столбец выделяется щелчком мыши на его имени. Протащив курсор мыши поперек столбцов можно выделить несколько столбцов. Выделенный столбец перетаскивается в новое место при установке курсора на его имени (области маркировки столбца).

✓ *Скрыть столбцы, не нужные для текущей работы, закрепить столбцы, а также отменить эти действия можно при помощи соответствующих команд меню **Формат**.*

✓ *Сохранение макета таблицы* выполняется командой **Файл/Сохранить** при закрытии таблицы после утвердительного ответа на вопрос «Сохранить изменения макета таблицы '<имя таблицы>'?».

Задание 5

Открыть любую из созданных таблиц и опробовать все команды создания макета.

Зачетное задание

Разработать структуру и создать базу данных, содержащую информацию об изделиях, производимых определенным подразделением предприятия. База должна состоять минимум из двух таблиц, имеющих поле связи. Каждая таблица должна иметь первичный ключ и содержать в себе минимум три поля и четыре записи. Наличие маски ввода и условия на значения (домена) обязательно. База может содержать информацию об ассортименте изделий, параметрах их качества и методах испытаний, бригадах, производящих ту или иную продукцию, ответственном за сопровождение изделия и т.п.

4. Лабораторная работа №4. Создание схемы данных в среде ACCESS

4.1. Цель работы

Выработка навыков организации схемы данных в пользовательском режиме СУБД Ms Access.

4.2. Программное обеспечение

Для проведения лабораторной работы требуется Ms Access 97/2000. Рекомендуется Ms Access 2000.

4.3. Ввод логически связанных записей

Объекты логически взаимосвязанных записей в таблицах могут быть связаны одно-многочисленными отношениями, но пока не создана схема данных и связи между таблицами не установлены, система не может контролировать логическую взаимосвязь вводимых данных. Поэтому для получения целостной базы, в которой все записи подчиненной таблицы имеют логически связанную с ней главную запись, пользователю необходимо самому отслеживать логические связи записей. При вводе подчиненных записей необходимо проверять наличие записей в главной таблице, значение ключа которой совпадает со значением поля связи (внешнего ключа) вводимой подчиненной записи.

При непосредственном вводе в таблицу записей, логически связанных с записями другой таблицей, полезно отобразить на экране обе таблицы.

Для одновременного отображения открытых таблиц можно воспользоваться командой в меню **Сверху вниз** (Tile Horizontally) или **Слева направо** (Tile Vertically) в меню **Окно** (Windows).

Очевидно, что в базе данных сложной структуры при вводе данных непосредственно в таблицы не гарантируется надежное и корректное обслуживание данных. При вводе данных в таблицы нижних уровней надо отслеживать несколько вышестоящих. При большом объеме данных это весьма затруднительно. Таким образом, необходима поддержка СУБД режима автоматической проверки целостности и непротиворечивости данных.

4.4. Схема данных в ACCESS

Схема данных в ACCESS является не только средством графического отображения логической структуры базы данных, При любой обработке данных из нескольких таблиц нет необходимости сообщать системе о наличии той или иной связи, т.к. однажды заданные связи между таблицами используются автоматически.

Реляционная база данных, созданная в соответствии с проектом канонической модели данных, состоит из нормализованных таблиц. В такой базе данных обеспечивается отсутствие дублирования данных во взаимосвязанных таблицах и,

соответственно, минимизируется объем сохраняемых данных. В процессе загрузки и корректировки базы данных для вывода отчетов и получения информации по запросам, а также для решения большинства задач необходим одновременный доступ к нескольким взаимосвязанным таблицам. Создание схемы данных позволяет упростить конструирование многотабличных форм, запросов и отчетов, а также обеспечить поддержание *целостности* взаимосвязанных данных при корректировке таблиц.

4.5. Взаимосвязи таблиц

При создании в ACCESS схемы данных в ней определяются и запоминаются связи между таблицами. Это позволяет системе автоматически использовать связи, один раз определенные в схеме данных, при создании форм, запросов, отчетов на основе взаимосвязанных таблиц, а пользователь освобождается от необходимости указывать эти связи при конструировании этих объектов. Схема данных базы графически отображается в своем окне, где таблицы представлены списками полей, а связи – линиями между полями разных таблиц.

4.6. Одно-многозначные (1:M) или одно-однозначные (1:1) связи

Схема данных прежде всего ориентирована на работу с таблицами, отвечающими требованиям нормализации, между которыми могут быть установлены одно-многозначные (1:M) или одно-однозначные (1:1) связи, для которых может автоматически поддерживаться связанная целостность. Поэтому схему данных целесообразно строить в соответствии с информационно-логической моделью.

При построении схемы данных ACCESS автоматически определяет по выбранному полю связи тип отношения между таблицами. Если поле по которому нужно установить связь, является уникальным ключом как в одной таблице, так и в другой, ACCESS выявляет отношения *один-к-одному*. Если поле связи является уникальным ключом в одной таблице (главной таблицы связи), а в другой таблице (подчиненной таблице связи) является не ключевым или входит в составной ключ, то есть значения его могут повторяться, ACCESS выявляет отношение *один-ко-многим* между записями главной таблицы к подчиненной. В этом случае можно задать автоматическое поддержание целостности связей.

Связи-объединения. Между двумя таблицами может быть установлена *связь-объединение* по некоторому полю связи. Для *связи-объединения* может быть выбран один из трех способов объединения записей:

- Способ 1 – объединение только тех записей, в которых связанные поля обеих таблиц совпадают (производится по умолчанию)
- Способ 2 – объединение тех записей, в которых связанные поля обеих таблиц совпадают, а также объединение всех записей из первой таблицы, для которых нет связанных со второй, с пустой записью второй таблицы.

□Способ 3 – объединение тех записей, в которых связанные поля обеих таблиц совпадают, а также объединение всех записей из второй таблицы, для которых нет связанных с первой, с пустой записью первой таблицы.

Такой тип связи может быть определен, если связь характеризуется отношением 1:1 или 1:M, а также если тип отношения не может быть определен системой, то есть если не выполняются условия для этих отношений. Например, при выборе в главной таблице в качестве поля связи неключевого поля или поля, входящего в составной ключ, ACCESS сообщает, что тип отношения не может быть определен. В этом случае между таблицами возможно установить только *связи-объединения*.

Связь-объединение обеспечивает объединение записей таблиц, имеющих одинаковые значения в поле связи. Кроме того, если выбран второй или третий вариант в результате объединения могут быть добавлены записи из таблицы, для которых нет логически связанных записей в другой таблице. Последние два варианта часто необходимы для решения практических задач. Примером такой задачи может быть формирование записей студентов с результатами успеваемости как в случае получения оценки по предмету, так и при отсутствии оценки. При отсутствии оценки соответствующее поле будет пустым.

4.7. Обеспечение целостности данных

При создании схемы данных пользователь включает в нее таблицы и устанавливает связи между ними. Для связей типа 1:1 и 1:M можно задать параметр обеспечения связной целостности данных, а также автоматическое каскадное обновление и удаление связанных записей.

Обеспечение связной целостности данных означает, что ACCESS при корректировке базы данных обеспечивает для связанных таблиц контроль за соблюдение следующих условий:

- В подчиненную таблицу не может быть добавлена запись с несуществующим в главной таблице значением ключа связи
- В главной таблице нельзя удалить запись, если не удалены связанные с ней записи в подчиненной таблице
- Изменение значений ключа связи в записи главной таблицы невозможно, если в подчиненной таблице имеются связанные с ней записи

При попытке пользователя нарушить эти условия в операциях добавления и удаления записей или обновления ключевых данных в связанных таблицах ACCESS выводит соответствующее сообщение и не допускает выполнения операции.

Установление между двумя таблицами связи типа 1:1 и 1:M и задание для нее параметров целостности данных возможно только при следующих условиях:

- Связываемые поля имеют одинаковый тип данных, причем имена полей могут быть различными
- Обе таблицы сохраняются в одной базе данных ACCESS
- Главная таблица связывается с подчиненной по первичному простому или составному ключу (уникальному индексу) главной таблицы

ACCESS автоматически отслеживает целостность связей при добавлении удалении записей и изменении значений ключевых полей, если между таблицами в схеме данных установлена связь с параметрами обеспечения целостности. При действиях, нарушающих целостность связи таблиц, выводится сообщение. ACCESS не позволяет установить параметр целостности для связи таблиц, если ранее введенные в таблицы данные не отвечают требованиям целостности.

4.8. Каскадное обновление и удаление связанных записей

Если для выбранной связи обеспечивается поддержание целостности, можно задать режим *каскадного обновления связанных полей* и режим *каскадного удаления связанных записей*.

В режиме *каскадного обновления связанных полей* при изменении значения поля связи в записи главной таблицы, ACCESS автоматически изменит значения в соответствующем поле в подчиненных записях.

В режиме *каскадного удаления связанных записей* при удалении записи из главной таблицы будут автоматически удаляться все связанные записи в подчиненных таблицах. При удалении записи из главной таблицы выполняется каскадное удаление подчиненных записей на всех уровнях, если этот режим задан на каждом уровне.

При удалении записей непосредственно в таблице или через форму выводится предупреждение о возможности удаления связанных записей.

4.9. Создание схемы данных

Создание схемы данных начинается в окне **Базы данных** (Database) с выполнения команды **Сервис/Схема данных** (Tools\Relationships) или нажатия кнопки **Схема данных** (Relationships) на панели инструментов базы данных.

4.9.1. Включение таблиц в схему данных

После нажатия кнопки **Схема данных** (Relationships) открывается окно **Добавление таблицы** (Show Table), в котором можно выбрать таблицы и запросы, включаемые в схему данных. Для размещения таблицы в окне **Схема данных** (Relationships) надо выделить ее в окне **Добавление таблицы** (Show Table) и нажать кнопку **Добавить** (Add). Для выделения нескольких таблиц надо, удерживая клавишу <Ctrl>, щелкнуть мышью на каждой из этих таблиц. Включив все нужные таблицы в схему данных, нажать кнопку **Заккрыть** (Close).

В результате в окне **Схема данных** (Relationships) будут представлены все включенные таблицы со списком своих полей. Далее можно приступить к определению связей между ними.

4.9.2. Создание связей между таблицами

При определении связей в схеме данных удобно использовать информационно-логическую модель в каноническом виде, по которой легко определить главную и подчиненную таблицу каждой одно-многозначной связи, поскольку в такой модели главные объекты всегда размещены выше подчиненных. Эти связи являются основными в реляционных базах данных, т.к. одно-однозначные связи используются лишь в редких случаях, когда приходится разделять большое количество полей, определяемых одним и тем же ключом, по разным таблицам, имеющим разный регламент обслуживания.

Устанавливая в окне схемы данных связи типа 1:М между парой таблиц, надо выделить в главной таблице уникальное ключевое поле, по которому устанавливается связь. Далее, при нажатой кнопке мыши, протащить курсор в соответствующее поле таблицы.

При создании связи по составному ключу необходимо выделить все поля, входящие в ключ главной таблицы, и перетащить их на одно из полей связи в подчиненной таблице. Для выделения всех полей, входящих в составной уникальный ключ, необходимо отмечать поля при нажатой клавише <Ctrl>. После создания связи откроется окно **Изменение связей**(Edit Relationships). При этом в строке **Тип отношения**(Relationships Type) автоматически установится тип **один-к-одному**(One-To-Many).

При составном ключе связи в окне **Изменение связей**(Edit Relationships) необходимо для каждого поля ключа в главной таблице **ТАБЛИЦА/ЗАПРОС**(Table\Query) выбрать соответствующее поле подчиненной таблицы, названной **СВЯЗАННАЯ ТАБЛИЦА/ЗАПРОС** (Related Table\Query).

4.9.3. Задание параметров целостности

В окне **Изменение связей**(Edit Relationships) можно задать параметр **Обеспечение целостности данных**(Enforce Referential Integrity) для выбранной связи. Если таблицы уже содержат данные, не отвечающие требованиям целостности, связь 1:М не устанавливается и не выдается сообщение.

После задания параметра целостности можно в окне **Изменение связей**(Edit Relationships) при необходимости отметить флажки **каскадное обновление связанных полей**(Cascade Update Related Fields) и **каскадное удаление связанных записей**(Cascade Delete Related Records).

Кнопка **Новое** в окне **Изменение связей** позволяет перейти к созданию связи между любыми двумя таблицами базы, не выходя в окно схемы данных.

Задание 1.

Создать таблицу «Кафедра» новой базы данных «Кафедры», структура которой приведена на рис.4.1.

Рис.4.1

Имя поля	Тип данных	Описание
Код кафедры	Числовой	Размер - байт
Аббревиатура кафедры	Текстовый	Размер - 20
Название	Текстовый	Размер - 50
Статус	Текстовый	Разм.-15, зн. по ум. - "Выпускающая", усл: "Выпускающая" Or "Не выпускающая"
Заведующий	Текстовый	Размер - 30

Заполнить ее данными в соответствии с рис.4.2.

Код кафедры	Аббревиатура	Название	Статус	Заведующий
0	ОКМ	Основ конструирования машин	Не выпускающая	д.т.н., проф. В.В. Петров
1	ЭПЭ	Электротехники и промышленной электроники	Выпускающая	д.т.н., проф. В.В. Мищенко
2	ВМ	Высшей математики	Не выпускающая	проф. А.М. Журавлев
3	Физики	Физики	Выпускающая	д.т.н., проф. С.А. Бурков
4	ТАД и ОМ	Технологии авиадвигателей и общего машиностроения	Выпускающая	д.т.н., проф. С.П. Суворов
5	Социологии	Социологии	Не выпускающая	д.ф.н., проф. М.Н. Воронов
6	МилП	Металловедения и литейного производства	Выпускающая	проф. А.А. Зиновьев
7	СИ	Станки и инструменты	Выпускающая	д.т.н., проф. Н.С. Раскатов

Рис.4.2

Создать таблицу «Преподаватель», структура которой приведена на рис.4.3, и заполнить ее данными, приведенными на рис.4.4.

Имя поля	Тип данных	Описание
Преподаватель	Текстовый	
Кафедра	Числовой	Размер - байт
Ученая степень	Текстовый	Размер - 50
Ученое звание	Текстовый	Размер - 50
Код преподавателя	Числовой	размер - целое

Рис.4.3

Преподаватель	Кафедра	Ученая степень	Ученое звание	Код преподавателя
Симакова А.И.	2			0
Зеленов В.А.	1	кандидат технических наук		1
Павлов В.В.	1	доктор технических наук	профессор	2
Мезенцев А.С.	4	кандидат технических наук		3
Судаков А.В.	1	кандидат технических наук		4
Семенов А.М.	2	кандидат технических наук	профессор	5
Крестовский В.А.	7	кандидат технических наук	доцент	6

Рис.4.4

Задание 2.

Организовать связь между таблицами «Кафедра» и «Преподаватель», организовав автоматическое обеспечение целостности данных и каскадное обновление связанных полей. Поля связи: «Код кафедры» – в таблице «Кафедра», «Кафедра» – в таблице «Преподаватель».

4.10. Изменение схемы данных

При модификации схемы данных осуществляется изменение состава ее таблиц – удаление, добавление таблиц и изменение связей.

Необходимость изменения связей возникает, в частности, при изменении ключей в таблицах. Изменение ключа по составу, по типу, и размеру его полей не может выполняться, пока не удалены связи таблицы в схеме данных.

При изменении типа данных для неключевых полей, задействованных в связях таблицы, также необходимо сначала открыть окно схемы данных. Для этого надо закрыть все таблицы и выполнить команду **Сервис|Схема данных** (Tools|Relationship) или нажать кнопку **Схема данных** (Relationship) на панели инструментов. При этом открывается панель инструментов **Связь** (Relationship).

Добавление таблиц выполняется при нажатии кнопки **Добавить таблицу** (Show Table) панели инструментов **Связь** (Relationship). В диалоговом окне **Добавление таблицы** (Show Table) надо выделить нужную таблицу и нажать кнопку **Добавить** (Add).

Удаление таблицы из схемы данных осуществляется переходом в окно **Схема данных** (Relationship). В этом окне надо удалить связи таблицы, сделать таблицу текущей и выполнить команду **Правка|Удалить** (Edit|Delete) или нажать клавишу **<Delete>**.

Удаление связи – к связи подвести курсор мыши и отметить связь щелчком, затем нажать правую кнопку мыши, вызвав контекстное меню, и выполнить команду **Удалить связь** (Delete Relationship). Помеченную связь можно также удалить командой **Правка|Удалить** (Edit|Delete) или нажав клавишу **<Delete>**.

Изменение параметров связи выполняется при выделенной связи в диалоговом окне, которое вызывается командой **Связи|Изменить связь** (Relationship|Edit Relationship) или соответствующей командой контекстного меню.

Задание 3.

Создать таблицу «Предмет» в соответствии со структурой на рис.4.5 и заполнить ее данными согласно рис.4.6.



Имя поля	Тип данных	Описание
Предмет	Текстовый	Размер 50
Преподаватель	Числовой	Целое

Рис.4.5

Предмет	Преподаватель
Основы обеспечения качества	4
Законодательство по качеству	1
Аналитическая геометрия	0
Математический анализ	0
Информационное обеспечение, базы и банки данных	1
Информатика	1
Общая электротехника и электроника	1
Основы обеспечения качества	3
Всеобщее управление качеством	3

Рис.4.6

Задание 4.

Установить связь таблицы «Предмет» и таблицы «Преподаватель» организовав автоматическое обеспечение целостности данных и каскадное обновление связанных полей. Поля связи: «Код преподавателя» – в таблице «Преподаватель», «Преподаватель» – в таблице «Предмет».

В результате должна получиться схема данных, приведенная на рис.4.7.

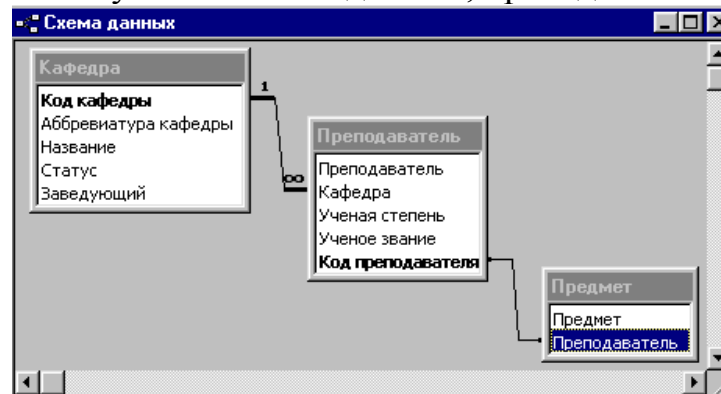


Рис.4.7

4.11. Ввод и корректировка данных во взаимосвязанных таблицах

Если построена схема данных и установлены параметры целостности, то при вводе и корректировке данных во взаимосвязанных таблицах пользователь должен учитывать требования связной целостности, рассмотренные в предыдущем разделе.

Если для одно-многочисленных связей, установленных в схеме данных, не задан параметр **Обеспечение целостности данных** (Enforce Referential Integrity), то пользователь при добавлении, удалении записей и изменения значений ключевых полей должен сам отслеживать непротиворечивость данных и целостность связей.

Если задан только параметр **Обеспечение целостности данных** (Enforce Referential Integrity), то ACCESS разрешает пользователю:

- Добавить запись в таблицу, которая не подчинена никакой другой таблице в схеме данных (находится на верхнем уровне иерархии)
- Добавить запись в подчиненную таблицу, когда в главной имеется запись с вводимым значением вторичного ключа
- Удалить запись в таблице, если нет связанных с ней записей в подчиненных таблицах

При попытке добавить в таблицу запись, для которой нет соответствующей главной записи, или удалить запись, для которой имеется связанная с ней подчиненная, или изменить значение ключа записи при наличии связанной подчиненной записи система выдаст сообщение о невозможности внесения изменений в таблицу.

Если наряду с параметром **Обеспечение целостности данных**(Enforce Referential Integrity) задан параметр **каскадное удаление связанных записей**(Cascade Delete Related Records), то пользователь может удалить запись, и при этом автоматически будут удалены все подчиненные записи.

Если наряду с параметром **Обеспечение целостности данных**(Enforce Referential Integrity) задан параметр **каскадное обновление связанных полей**(Cascade Update Related Fields), то пользователь может изменять значение ключевого поля, и при этом автоматически будут обновлены все значения внешних ключей во всех связанных подчиненных записях.

Таким образом, при загрузке таблиц базы данных при установленных параметрах целостности система отслеживает корректность связей для вводимых записей.

В ACCESS 2000 имеется возможность отображения записей подчиненных таблиц при просмотре таблицы. Благодаря этому пользователь может при добавлении, удалении и корректировке записей осуществлять контроль за корректностью связей в отображаемой цепочке таблиц.

4.11.1. Отображение записей подчиненных таблиц в главной

В таблице базы данных ACCESS 2000 одновременно можно просматривать данные взаимосвязанных таблиц нескольких уровней.

При просмотре таблицы, которая имеет простой ключ, только одну подчиненную таблицу, и в схеме данных между ними уже определена связь, отображается столбец со значками (+) в каждой записи. Достаточно щелкнуть на этом значке в строке записи, чтобы отобразить записи подчиненной таблицы, связанные текущей записью. При этом плюс на значке преобразуется в (-). Щелчком на минусе подчиненные записи закрываются. Таким образом могут быть открыты подчиненные записи каждой записи главной таблицы. Открыв все записи главной таблицы, вы увидите все записи подчиненной таблицы, разбитые на подмножества, связанные с конкретными записями главной таблицы. Воспользовавшись командой меню **Формат\Подтаблица\Развернуть все** (Format\Subdatasheet\Expand All) или **Свернуть все**(Collapse All), можно открыть или закрыть все подчиненные записи¹.

При этом свойстве главной таблицы **Имя подчиненной таблицы**(Subdatasheet Name) по умолчанию устанавливается значение **Авто**(Auto), которое и определяет единственную связь таблицы и вывод столбца со значками «+» для открытия записей по этой связи².

В общем случае открывать связанные записи нужной подчиненной таблицы можно только в том случае, если в свойствах таблицы была определена связь с этой подчиненной таблицей и задан ключ этой связи. Для определения связи можно задать соответствующие свойства таблицы. Для удобства пользователя в режиме таблицы предусмотрена команда меню **Вставка\Подтаблица** (Insert\Subdatasheet), которая позволяет в диалоговом режиме определить связь. Кроме того, предусмотрен ряд кнопок, которые могут быть выведены на панель инструментов:

- Подтаблица** (Subdatasheet);
- Развернуть все** (Expand All);
- Свернуть все** (Collapse All);
- Удалить** (Remove).

Итак, если таблица не имеет столбца с полюсами, надо выполнить команду **Подтаблица**(Subdatasheet), которая определит связь таблицы с подчиненной и даст возможность открывать связанные данные в подчиненном окне. После выполнения команды выводится окно **Вставка подтаблицы** (Insert Subdatasheet), в котором следует выбрать подчиненную таблицу или запрос и указать ключ связи.

Если главная и подчиненная таблицы имеют разные имена полей, составляющих ключ связи, и в схеме данных не установлена связь между таблицами, то в окне **Вставка подтаблицы** (Insert Subdatasheet) можно ввести имена полей связи, разделяя

¹ В подчиненной таблице не отображается поле внешнего ключа.

² Дополнительный столбец, в котором стоит значок «+», автоматически отображается только в таблице, имеющей простой ключ, единственную подчиненную таблицу и определенную в схеме данных связь с этой подчиненной таблицей.

их точкой с запятой. При этом система автоматически добавляет данную связь в схему базы данных. Установленная таким образом связь действует во время текущего просмотра главной таблицы.

Если сохранить таблицу, то установленная связь с подчиненной таблицей делается постоянной, т.к. только после этого записывается в свойства главной таблицы³.

Командой **Подтаблица**(Subdatasheet) можно изменить ранее установленную и сохраненную связь с некоторой подчиненной таблицей. Для этого в окне **Вставка подтаблицы** (Insert Subdatasheet) надо установить новую связь таблицы, которая будет действовать только в течение сеанса. Сохранив таблицу, можно заменить ранее установленную связь на новую постоянную. При этом запись в свойствах таблицы изменяется автоматически.

Пользуясь командой **Подтаблица**(Subdatasheet), можно в течение одного сеанса подключать для работы различные подчиненные таблицы или запросы.

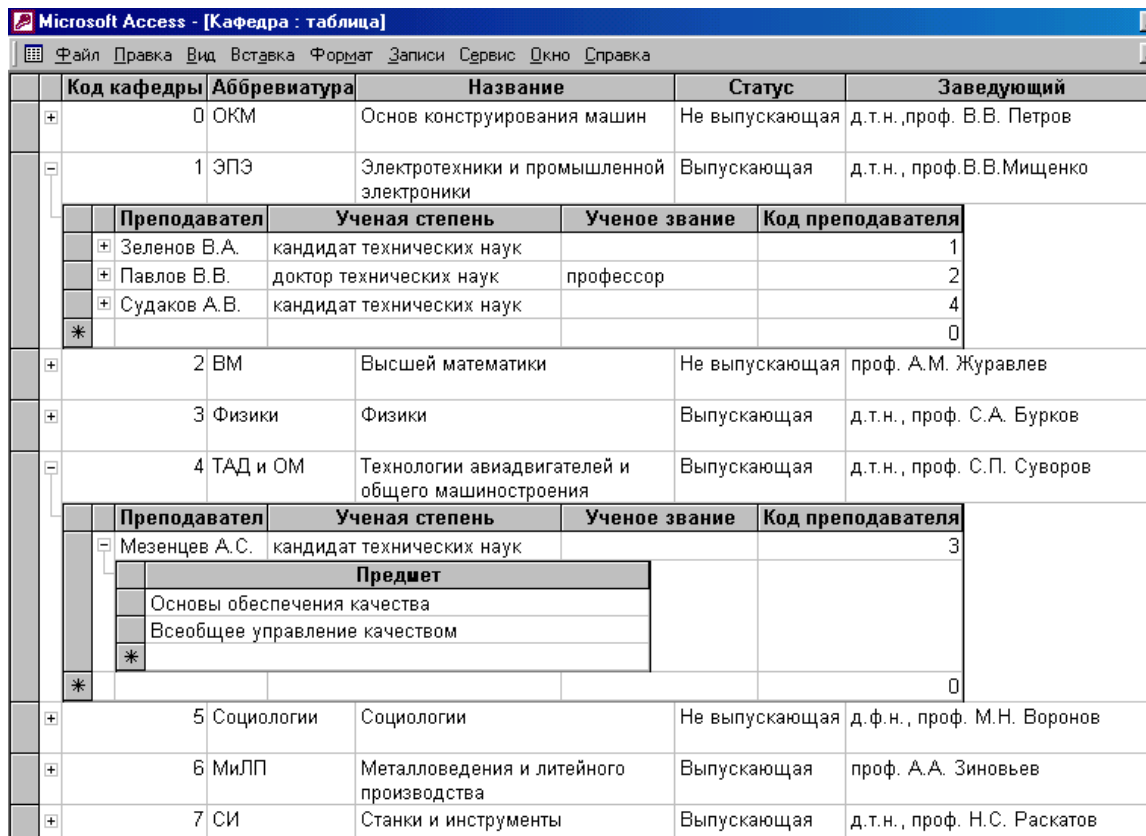
Для установления постоянной связи таблицы с конкретной подчиненной можно обратиться непосредственно к свойствам таблицы в режиме конструктора. В списке свойства **Имя подчиненной таблицы**(Subdatasheet Name) выбирается подчиненная таблица, при этом в свойствах **Подчиненные поля**(Link Child Fields) и **Основные поля**(Link Master Fields) отобразятся поля ключа (простого и составного) главной таблицы, по которым осуществляется связь.

Для удаления связи, жестко зафиксированной в свойствах таблицы, можно выбрать параметр **Авто**(Auto) в качестве значения свойства **Имя подтаблицы**(Subdatasheet Name). Для того чтобы в таблице не выводился столбец, позволяющий открывать подчиненные записи, в качестве значения свойства должно быть установлено **Нет** (None). Это можно сделать в режиме таблицы с помощью команды меню **Формат\Подтаблица\Удалить** (Format\Subdatasheet\Remove) или соответствующей кнопки панели инструментов.

В подчиненной таблице можно выполнять те же действия, что и в главной: работать с данными или отображать подчиненную более низкого уровня.

Несмотря на то, что основным назначением команды является предоставление просмотра связанных записей подчиненных таблиц, система позволяет устанавливать связь и просматривать связанную запись главной таблицы из подчиненной.

³ Нет необходимости в сохранении выполненных в таблицах корректировок данных, т.к. корректировки сохраняются в таблице автоматически сразу после перехода к следующей записи таблицы.



Код кафедры	Аббревиатура	Название	Статус	Заведующий
0	ОКМ	Основ конструирования машин	Не выпускающая	д.т.н., проф. В.В. Петров
1	ЭПЭ	Электротехники и промышленной электроники	Выпускающая	д.т.н., проф. В.В. Мищенко
	Преподаватель	Ученая степень	Ученое звание	Код преподавателя
	Зеленов В.А.	кандидат технических наук		1
	Павлов В.В.	доктор технических наук	профессор	2
	Судаков А.В.	кандидат технических наук		4
	*			0
2	ВМ	Высшей математики	Не выпускающая	проф. А.М. Журавлев
3	Физики	Физики	Выпускающая	д.т.н., проф. С.А. Бурков
4	ТАД и ОМ	Технологии авиадвигателей и общего машиностроения	Выпускающая	д.т.н., проф. С.П. Суворов
	Преподаватель	Ученая степень	Ученое звание	Код преподавателя
	Мезенцев А.С.	кандидат технических наук		3
	Предмет			
	Основы обеспечения качества			
	Всёобщее управление качеством			
	*			0
5	Социологии	Социологии	Не выпускающая	д.ф.н., проф. М.Н. Воронов
6	МилП	Металловедения и литейного производства	Выпускающая	проф. А.А. Зиновьев
7	СИ	Станки и инструменты	Выпускающая	д.т.н., проф. Н.С. Раскатов

Рис.4.8

Задание 5.

Открыть таблицу «Кафедра» (см. рис.4.8) и дополнить базу данными следующей информацией:

1. Кандидат технических наук, доцент Сидоров Вадим Николаевич работает на кафедре КиПРА («Конструирование и производство радиоаппаратуры») и читает курс «Теоретические основы электротехники».
2. Кандидат технических наук Барулев Сергей Петрович работает на кафедре КиПРА и читает курсы: «Метрология», «Источники вторичного электропитания», «Информатика».
3. Кандидат филологических наук, доцент Селезнева Галина Александровна является заведующим кафедрой иностранных языков (ИнЯз) и преподаёт немецкий язык.
4. Кандидат филологических наук, доцент Павлова Елена Анатольевна работает на невыпускающей кафедре иностранных языков и преподаёт английский.
5. Заведующим кафедрой философии является доктор философских наук, профессор Путилов Виктор Андреевич.
6. Кандидат социологических наук Соболев Игорь Николаевич работает на выпускающей кафедре философии и преподаёт курс философии.

4.12. Модификация структуры базы данных

К модификации структуры базы данных относится изменение структуры отдельных таблиц, добавление и удаление таблиц, а также изменение схемы данных.

4.12.1. Изменение структуры таблиц

Для изменения структуры таблиц основным является режим конструктора таблиц, хотя некоторые изменения можно произвести в режиме таблицы. Такие простые операции, как изменение имени, добавление, удаление неключевых полей, могут быть выполнены в обоих режимах. Однако в режиме таблицы имеется ряд ограничений. Например, можно в режиме таблицы изменить имя поля, дважды щелкнув мышью на его заголовке и введя новое значение. Это значение становится именем поля, но при этом теряется значение подписи поля. Для точного определения типа данных также потребуется определять свойства поля в режиме конструктора.

4.12.1.1. Изменение полей, которые не являются ключами или полями связи

Состав и последовательность, а также тип данных, свойства или имена таких полей можно изменять независимо от наличия связей таблицы с другими таблицами базы данных. Для заполненной таблицы беспрепятственно выполняется изменение имени поля, Добавление нового поля, удаление поля и изменение последовательности полей. При изменении типа данных, размера и других свойств выполняется преобразование данных. Однако, если преобразования не допустимы, попытка изменения типа данных может привести к их потере. Система запрашивает у пользователя подтверждение его действий.

4.12.1.2. Изменение или удаление ключевого поля

Для загруженной и несвязанной таблицы при попытке изменить свойства или удалить ключевое поле система предупреждает о возможной потере данных и удалении ключа. Если, например, удаляется поле в составном ключе, с других полей этого ключа будет снят признак ключа.

Для отказа от определения первичного ключа в таблице достаточно нажать кнопку **Ключевое поле** (Primary Key) или удалить индекс ключа в окне **Индексы** (Indexes), которое открывается после нажатия соответствующей кнопки на панели инструментов. При попытке назначить в качестве ключа другое поле, имеющее повторяющиеся значения в загруженной таблице, операция не будет завершена с выдачей соответствующего сообщения. Если надо изменить ключ таблицы, которая имеет связи с другими таблицами в схеме данных, необходимо предварительно разорвать связи.

Задание 7.

Создать базу данных, отображающую структуру документации качества. Создать соответствующую схему данных.

Зачетное задание.

Создать базу данных, отражающую иерархическую структуру, приведенную на рис.4.9.

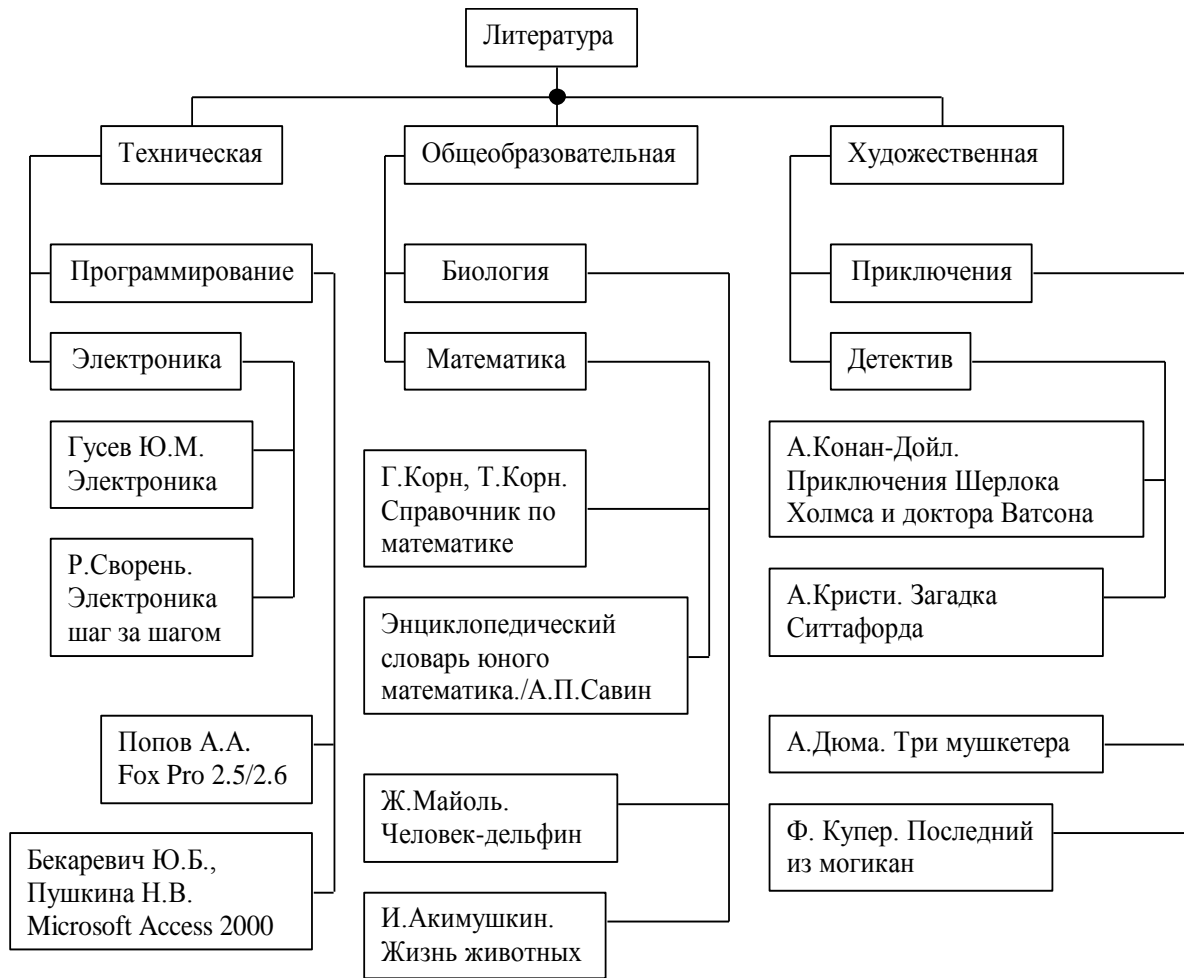


Рис.4.9.

5. Лабораторная работа №5. Создание форм для однотоабличных и многотоабличных структур данных в среде Access

5.1. Цель работы

Выработка навыков разработки простых и сложных форм в пользовательском режиме СУБД Ms Access.

5.2. Программное обеспечение

Для проведения лабораторной работы требуется Ms Access 97/2000. Рекомендуется Ms Access 2000.

5.3. Форма - диалоговый графический интерфейс пользователя для работы с базой данных

Access предоставляет широкие возможности по конструированию графического диалогового интерфейса пользователя для работы с базой данных. Основой такого интерфейса являются формы.

Формы являются основой разработки диалоговых приложений пользователя для работы с базой данных. Через формы может осуществляться первоначальная загрузка данных во взаимосвязанные таблицы БД, просмотр данных, а также их корректировка. Работая с формой, пользователь может добавлять и удалять записи в таблицах, изменять значения в полях, получать расчетные данные. В форме может осуществляться контроль вводимых данных, могут устанавливаться ограничения на доступ к данным, выводиться необходимые сообщения, возможна обработка событий, инициируемых пользователем или наступающих в процессе работы с формой. Типовые процедуры формируются автоматически при создании элементов формы. Такими элементами, например, являются, графические кнопки, с которыми могут связываться события следующих категорий:

- Переходы по записям, обработка записей (добавление, удаление, печать, восстановление)
- Работа с формой (закрытие, открытие, изменение фильтра, обновление данных, печать формы)
- Работа с отчетом (печать, просмотр, отправка, вывод в файл), приложение (запуск приложения, выход из приложения, запуск Word, Excel, блокнота)
- Запуск запроса, макроса, печать таблицы, набор номера.

Пользователь может сам программировать макросы и процедуры VBA (Visual Basic Applications) для обработки различных событий, наступающих при работе в форме. Для формы и ее элементов управления в Access определен набор событий, для которых могут быть разработаны процедуры обработки события. Примерами таких событий являются: "Открытие формы", "Закрытие формы", "Текущая запись", "До обновления", "После обновления поля", "Нажатие кнопки", "Двойное нажатие кнопки".

5.4. Технология загрузки базы данных с использованием форм

При наличии схемы данных, состоящей из нормализованных таблиц, связанных одно-многочными отношениями, могут быть созданы экранные формы, которые обеспечивают корректный ввод взаимосвязанных данных. Макет таких форм целесообразно делать адекватным формам первичных документов — источников данных для загрузки справочных и оперативных учетных данных. При этом обеспечивается комфортная работа пользователя и обеспечивается важнейший аспект технологии работы с базой данных - однократный ввод данных. Такие формы позволяют в любой момент посмотреть содержимое ранее введенных документов.

Технология создания целостной базы, в которой между таблицами установлены связи, предполагает *упорядочение загрузки* взаимосвязанных таблиц при обеспечении пользователя удобным интерфейсом. Такая технология может строиться на использовании соответствующих экранных форм ввода/вывода.

Перед конструированием форм в Access целесообразно на этапе проектирования

выполнить подготовительную работу для определения последовательности загрузки базы данных. Это необходимо для получения рационально сконструированных форм, обеспечивающих удобную работу пользователя, корректный ввод взаимосвязанных данных при создании и корректировке целостной базы данных.

5.5. Последовательность загрузки таблиц базы данных

При разработке форм, обеспечивающих загрузку взаимосвязанных таблиц базы данных, следует придерживаться определенных требований к последовательности их загрузки в соответствии со схемой данных. Эти требования можно сформулировать следующим образом:

- ✓ Независимо могут загружаться таблицы, которые не подчинены каким-либо другим таблицам в одно-многочисленных связях
- ✓ Таблицы, подчиненные каким-либо другим таблицам, могут загружаться либо одновременно с ними, либо после загрузки главных таблиц, в противном случае не могут установиться связи загружаемых в подчиненную таблицу записей с записями главных таблиц
- ✓ В базу данных сначала загружаются из соответствующих документов справочные данные, а затем учетные

Перед конструированием форм для загрузки базы данных Access, необходимо проводить подготовительную работу по определению этапов загрузки БД.

5.6. Этапы загрузки базы данных и проектирования форм

В процессе определения этапов загрузки базы данных и проектирования форм целесообразно выполнить приведенную ниже последовательность действий:

1. Определение документов-источников немашинной сферы, содержащих необходимые данные для загрузки таблиц базы данных.
2. Определение таблиц-объектов загрузки на отдельных этапах ввода данных и соответствующего документа-источника.
3. Определение последовательности этапов загрузки.
4. Определение подсхемы данных для каждого этапа загрузки (фрагмент схемы данных), необходимой для построения экранной формы ввода из документа.

В подсхему данных могут входить:

- Таблица-объект загрузки.
- Таблица, связанная с таблицей-объектом загрузки и содержащая данные только для отображения (вывода) в форме.
- Таблица, главная относительно загружаемой, позволяющая группировать вводимые (выводимые) записи.

5. Определение общей структуры экранной формы, т. е. ее макета в соответствии со структурой входного документа и подсхемой данных. При этом:

- Для многотабличной (составной) формы выбирается таблица, которая будет источником записей основной части этой формы, и задается название формы.

- Выбираются таблицы, которые будут источником записей подчиненных форм, включаемых в составную форму, и определяется название каждой подчиненной формы.

- Распределяется пространство формы для размещения ее основной части и включаемых подчиненных форм.

- Если подчиненная форма в свою очередь имеет включаемую (подчиненную 2-го уровня) форму, для нее выполняется работа как для любой составной формы.

б. Определение состава размещаемых реквизитов для каждой из частей составной формы. При этом:

- Ключевые поля таблицы-источника основной части составной формы надо вводить в основную часть формы.

- Предусмотреть в подчиненной форме ключевые поля таблицы-источника подчиненной формы, которых нет в таблице-источнике основной части.

После выполнения перечисленных пунктов осуществляется конструирование экранной формы средствами Access.

5.7. Основы создания однотабличных форм

Форма на основе одной таблицы может быть построена как самостоятельная для загрузки, просмотра и корректировки таблицы, а также как вспомогательная для включения в какую либо составную форму.

Любая форма, с помощью которой можно просматривать, вводить или редактировать записи таблиц БД, должна быть предварительно спроектирована и далее сконструирована средствами Access.

Для создания формы могут быть использованы мастера Access. Однако точное формирование макета формы в соответствии с требованиями, выработанными пользователем в процессе ее проектирования, обеспечивается средствами конструирования форм. Ниже рассматриваются основные понятия и техника конструирования однотабличных форм.

5.7.1. Конструирование формы

Для конструирования форм в Access используется Конструктор форм (Form Design). При конструировании однотабличной формы определяется таблица БД, на основе которой создается форма, выбираются поля таблицы, которые должны быть представлены в форме, осуществляется их размещение в макете формы, создаются вычисляемые поля и другие графические элементы: кнопки, выключатели, элементы оформления, поясняющий текст, рисунки. Для настройки различных элементов формы используется типовой набор их свойств.

5.7.2. Области и элементы формы в режиме конструктора

Форма в режиме Конструктора форм (Form Design) имеет три области: **Область данных** (Detail) **Заголовок формы** (Form Header), и **Примечание формы** (Form Footer), которые могут быть образованы по команде меню **Вид/Заголовок/примечание формы**. Области формы наполняются различными

графическими объектами.

Элементы или графические объекты. Графические объекты, связанные с записями таблиц и предназначенные для отображения данных некоторого поля, называются *элементами управления*. Основными типами элементов управления являются: **Поле** (Text Box), **Поле со списком** (List Box), **Список** (Combo Box). Тип элемента управления, выбираемый для поля *по умолчанию*, определяется в свойствах поля таблицы базы данных, с которым связано поле формы. Задается это свойство при определении типа данных поля в сжиме конструктора таблиц на вкладке **Подстановка** (Lookup).

Графические объекты, не связанные с таблицами или запросами, предназначены прежде всего для создания макета формы и содержат *надписи* полей пользовательские названия реквизитов), внедряемые *объекты*, надписи этих объектов, заголовки. Информация об этих элементах сохраняется в макете формы.

Свойства элементов. Как форма в целом, так и каждый из ее элементов обладает своими свойствами, которые можно просматривать и корректировать, выполнив команду меню **Вид/Свойства** (View/Properties) или команду **Свойства** (Properties) в контекстно-зависимом меню, вызываемом правой кнопкой мыши. Перед вызовом контекстно-зависимого меню курсор устанавливается на элемент, свойства которого надо отобразить. Перед выполнением команды элемент должен быть выделен щелчком мыши. Свойства элемента позволяют определить его внешний вид, размер, местоположение в форме, режим ввода/вывода, привязать к элементу выражение, макрос или программу.

Для вызова контекстно-зависимого меню формы в целом курсор должен быть предварительно установлен на черном квадрате — области выделения формы, который размещен в верхнем левом углу на пересечении линеек. В этом меню можно перейти к просмотру свойств формы в целом.

При щелчке мышью на этом квадрате форма выделяется. Двойной щелчок на области выделения формы позволяет сразу вызвать свойства формы. Для отображения на экране ее свойств может быть также выполнена команда меню **Вид/Свойства** (View/Properties).

Если линейки не выводятся на экран, нужно установить курсор на серую фоновую область справа от формы. Заметим, что выделить целиком форму можно также, выполнив команду **Правка/Выделить форму** (Edit/Select Form).

Свойства формы отображаются в диалоговом окне **Форма** (Form), где для каждого свойства и его значения отводится строка.

Кнопки управления. В форме могут быть предусмотрены кнопки управления для разных целей, например, для перехода к другим записям в просматриваемой таблице, для работы с записями (добавить, дублировать, восстановить, удалить, сохранить), для работы с формой (открыть, закрыть, фильтровать, обновить)

5.8. Панели инструментов конструктора форм и форматирования

Для выполнения необходимых действий при создании формы используется панель инструментов **Конструктор форм** (Form Design) и панель **Формат (форма/отчет)** (Formatting (Form/Report)). Эти панели открываются в режиме конструктора, когда создается новая форма или открывается форма для

редактирования ее макета.

Ниже приводится основное назначение кнопок:

- ✓ **Вид (View)** - позволяет выбрать режим отображения формы из списка, включающего **Конструктор (Design View)**, **Режим формы (Form View)**, **Режим таблицы (Table View)**
- ✓ **Сохранить (Save)** — сохраняет изменения, внесенные в активный объект
- ✓ **Печать (Print)** — печать выводимых в форме данных формы
- ✓ **Предварительный просмотр (Print Preview)** отображает страницу перед ее печатью
- ✓ **Орфография (Spelling)** - подключает поиск орфографических ошибок по всем текстовым полям текущей записи (кнопка доступна в режиме формы)
- ✓ **Вырезать (Cut)** — удаление выделенных элементов и их копирование в буфер обмена
- ✓ **Копировать (Copy)** — копирование выделенных элементов в буфер обмена
- ✓ **Вставить (Paste)** — вставляет содержимое буфера обмена в выделенную область формы или элемент управления
- ✓ **Формат по образцу (Format Painter)** — предназначена для копирования параметров форматирования текущего элемента управления
- ✓ **Отменить (Undo Move)** — отменяет одно последнее действие
- ✓ **Добавить гиперссылку (Insert Hyperlink)** - создает элемент управления для перехода к объектам базы данных (той же или другой), к документам Word, Excel и другим приложениям Windows, а также к документам глобальной сети Internet или корпоративной сети intranet
- ✓ **Список полей (Field List)** — вызывает список доступных полей для выделенной области формы
- ✓ **Панель элементов (Toolbox)** — открывает панель элементов
- ✓ **Автоформат (AutoFormat)** - настройка и применение встроенных форматов к форме, разделу, или элементам управления.
- ✓ **Программа (Code)** - открытие модуля формы в редакторе VBA
- ✓ **Свойства (Properties)** - открывает окно свойств для выделенной формы раздела или элемента управления
- ✓ **Построить (Build)** - вызывает диалоговое окно, позволяющее выбрать и запустить построитель **Выражений (Expression Builder)**, редактор **Макросов (Macro Builder)** или **Программ на VBA (Code Builder)** для выделенной формы или элемента управления
- ✓ **Окно базы данных (Database Window)** - делает активным окно базы данных
- ✓ **Новый объект (New Object)** - открывает список объектов и позволяет приступить к созданию нового объекта базы данных **Таблицы (Table)** или **Запроса (Query)**, **Формы (Form)**, **Отчета (Report)**, **WEB-Страницы (Page)**, **Макроса (Macro)**, **Модуля (Module)**
- ✓ **Справка по Microsoft Access (MSAccess Help)** - вызывает помощник Access для получения справочной информации, советов или настройки параметров помощника

Панель форматирования содержит стандартные кнопки форматирования приложений Windows, доступ к которым открывается при выделении рамки элемента управления. Эти кнопки вызывают команды, позволяющие менять шрифт, параметры форматирования и оформления надписей, а также значений данных, выводимых в поля формы. Дополнительная панель, позволяющая добавить или временно удалить кнопки, вызывается нажатием кнопки панели форматирования **Другие кнопки**. Ниже дано краткое назначение кнопок панели форматирования:

- ✓ **Объект** (Object) - позволяет выбрать и выделить из списка доступные разделы и элементы управления формы
- ✓ **Шрифт** (Font) - позволяет выбрать из списка шрифт для выделенного элемента управления
- ✓ **Размер** (Font Size) — позволяет выбрать размер шрифта для выделенного элемента управления
- ✓ **Полужирный** (Bold), **Курсив** (Italic), **Подчеркнутый** (Underline) — делает шрифт жирным, наклонным или подчеркнутым для выделенного элемента управления
- ✓ **По левому краю** (Align Left), **по центру** (Center), **по правому краю** (Align Right) — делает содержимое выделенного элемента управления выровненным по левому краю, по центру или по правому краю
- ✓ **Цвет заливки/фона** (Fill/Back Color) — позволяет выбрать цвет фона для выделенного элемента управления или раздела
- ✓ **Цвет текста** (Font/Fore Color) — позволяет выбрать цвет текста
- ✓ **Цвет линии/границы** (Line/Border Color) — позволяет выбрать цвет границы для выделенного элемента управления
- ✓ **Толщина линии/границы** (Line/Border Width) — позволяет выбрать толщину границы для выделенного элемента управления
- ✓ **Оформление** (Special Effect: Flat) — позволяет выбрать вариант оформления выделенного элемента управления

5.8.1. Настройка панели инструментов

В Access панели инструментов отображаются в соответствии с режимом работы, т. е. являются контекстно-зависимыми. При переходе из окна базы данных в режим конструктора формы автоматически закрывается панель инструментов **База данных** (Database) и открываются панель **Конструктор форм** (Form Design) и **Панель элементов** (Toolbox). С помощью контекстного меню при активной панели конструктора можно открыть панель форматирования **Формат (форма/отчет)** (Formatting). При активизации окна базы данных закроются панели конструктора и элементов, а откроется панель инструментов **База данных** (Database).

Для принудительного отображения любой панели выполняется команда **Вид/Панели инструментов/Настройка** (View/Toolbars/Customize) и в окне **Настройка** (Customize) на вкладке **Панели инструментов** (Toolbars) отмечаются нужные панели. Для отображения панели конструктора форм надо отметить флажок **Конструктор форм** (Form Design). Окно настройки может быть вызвано и через контекстное меню любой панели, отображенной на экране.

Изменение состава кнопок панели инструментов. Состав доступных кнопок

панели может быть изменен. Рассмотрим, например, изменение состава кнопок (команд) панели конструктора форм. Для этого в окне **Настройка** (Customize) на вкладке **Команды** (Commands) в списке **Категории** (Categories) надо выбрать опцию **Конструктор форм или отчетов** (Form/Report Design). Далее курсором мыши можно перетаскивать любую кнопку из области **Команды** (Commands) на панель конструктора форм или удалять кнопку обратным перемещением.

Описание кнопки можно просмотреть, нажав в разделе **Выделенная команда** (Selected Command) кнопку **Описание** (Description).

Замечание

Для модификации свойств кнопки необходимо выделить ее на панели конструктора форм при открытом окне **Настройка** (Customize). Далее в окне настройки нажать кнопку **Изменить выделенный объект** (Modify Selection) и в дополнительном окне в строке **Имя** (Name) можно изменить значок, выбрав его после выполнения команды **Выбрать значок для кнопки** (Change Button Image).

Закончив настройку, нажмите кнопку **Заккрыть** (Close).

5.8.2. Панель элементов

При конструировании форм необходимо наряду с панелями инструментов использовать **Панель элементов** (Toolbox), которая вызывается из меню **Вид** (View) или нажатием кнопки **Панель элементов** (Toolbox) панели **Конструктора форм** (Form Design). Панель автоматически открывается при переходе в режим конструктора.

Панель элементов (Toolbox) позволяет создавать элементы управления в форме и осуществлять необходимые действия при конструировании:

- ✓ **Выбор объектов** (Select Objects) - по умолчанию позволяет щелчком мыши выделять элемент, раздел или форму
- ✓ **Мастера** (Control Wizards) — включает или отключает мастера для создания элементов управления
- ✓ **Надпись** (Label) — для создания текстов постоянных заголовков, примечаний, инструкций, не связанных с другими элементами управления
- ✓ **Поле** (Text Box) типа **Свободный** (Unbound) — для установления связи с полем таблицы или запроса, а также для создания вычисляемых полей
- ✓ **Группа переключателей** (Option Group) — для размещения набора флажков, переключателей или выключателей
- ✓ **Выключатель** (Toggle Button) — для создания элемента управления, который может принимать одно из двух значений 1 или 0 (Вкл/Выкл, Истина/Ложь, Да/Нет)
- ✓ **Переключатель** (Option Button) — для выбора альтернативных значений параметра
- ✓ **Флажок** (Check Box) — для выбора нескольких возможных значений
- ✓ **Поле со списком** (Combo Box) — объединяет поле и раскрывающийся список значений. Значения могут вводиться как непосредственно в поле, так и путем выбора из списка
- ✓ **Список** (List Box) — создает всегда раскрытый список значений, которые при связи с полем являются единственным источником ввода в поле

- ✓ **Кнопка** (Command Button) — для создания командной кнопки, с помощью которой может быть выполнен переход по записям, открыта форма, напечатан отчет и другие функции Access
- ✓ **Рисунок** (Image) - для отображения нередатируемого рисунка, не являющегося объектом OLE
- ✓ **Свободная рамка объекта** (Unbound Object Frame) - для отображения свободного объекта OLE, который остается неизменным при переходе по записям
- ✓ **Присоединенная рамка объекта** (Bound Object Frame) - для отображения объектов OLE, сохраненных в поле базового источника записей формы
- ✓ **Разрыв страницы** (Page Break) - для начала нового экрана в форме, новой страницы в печатной форме (отчете)
- ✓ **Набор вкладок** (Tab Control) - для создания вкладок в форме, на каждой из которых могут размещаться свои элементы управления
- ✓ **Подчиненная форма/отчет** (Subform/Subreport) - для вывода данных из таблиц, связанных с таблицей-источником формы
- ✓ **Линия** (Line) - для разграничения разделов в форме (отчете)
- ✓ **Прямоугольник** (Rectangle) - для создания рамки при оформлении
- ✓ **Другие элементы** (More Controls) — открывает обширный список дополнительных элементов, при выборе из которого в форме будет создан соответствующий элемент

При помощи кнопок этой панели пользователь создает по своему усмотрению удобный графический интерфейс для работы с базой данных через форму.

5.8.3. Переход в режим конструктора форм

В режиме конструктора создание формы полностью возлагается на пользователя, причем создание начинается с пустой формы. Чтобы начать создание формы в режиме конструктора, надо в окне **База данных** (Database) в группе **Объекты** (Objects) выбрать элемент **Формы** (Forms). Вызов конструктора может быть осуществлен в рабочем пространстве окна активизацией значка **Создание формы в режиме конструктора** (Create form in Design view). Для выбора режима конструктора при создании новой формы можно предварительно нажать кнопку **Создать** (New) на панели инструментов окна базы данных. Открывающееся после этого диалоговое окно **Новая форма** (New Form) предоставляет возможность выбрать режим создания формы — **Конструктор** (Design View).

5.8.4. Мастера создания формы

Типовой макет формы может быть легко получен с помощью мастера Access. Для получения однотабличной формы, соответствующей требованиям пользователя, бывает целесообразно сначала использовать **Мастер форм** (Form Wizard) или команду **Автоформа** (AutoForm). Полученную таким образом форму далее можно доработать средствами конструктора форм.

Чтобы начать создание формы при помощи мастера, надо в окне **База данных** (Database) в группе **Объекты** (Objects) выбрать элемент **Формы** (Forms). В рабочем поле этого объекта имеется значок **Создание формы с помощью мастера** (Create form by using wizard), после активизации которого можно приступить к созданию формы

мастером. Чтобы получить доступ к списку всех вариантов создания новой формы, можно на панели инструментов окна базы данных нажать кнопку **Создать (New)**. В этом окне мастер вызывается при выборе команды **Мастер форм (Form Wizard)**.

Кроме этого режима, простейшая форма на основе только одной таблицы может быть легко создана выбором одного из режимов: **Автоформа: в столбец (AutoForm: Columnar)**, **Автоформа: ленточная (AutoForm: Tabular)**, **Автоформа: табличная (AutoForm: Datasheet)**. Форма **в столбец (Columnar)** является однозаписевой, **табличная (Datasheet)** и **ленточная (Tabular)** формы являются многозаписевыми и размещают все поля в одной строке.

Мастер форм (Form Wizard) может создавать форму для одной таблицы или для нескольких взаимосвязанных таблиц. Мастер позволяет пользователю задать, какие поля включать в форму, и выбрать стиль ее оформления.

Команды **Автоформа: в столбец (AutoForm: Columnar)**, **Автоформа: ленточная (AutoForm: Tabular)** и **Автоформа: табличная (AutoForm: Datasheet)** создают для заданной таблицы формы, которые отличаются от форм, создаваемых мастером, тем, что включают все поля таблицы и не предоставляют возможности выбора стиля оформления. Эти команды, не вступая в диалог с пользователем и не отображая формы в режиме конструктора, выводят ее на экран в режиме формы, т. е. заполненную значениями из таблицы. Заметим, что таблица, для которой строится автоформа, выбирается предварительно в окне **Новая форма (New Form)**.

Форма, созданная мастером, как и форма, созданная любой командой **Автоформа (AutoForm)**, может быть отредактирована в соответствии с требованиями пользователя. Редактирование выполняется в режиме конструктора форм.

Задание 1.

Создайте в режиме конструктора таблиц таблицу «Абитуриент», содержащую следующие данные: специальность, рег. номер, Ф.И.О., дата рождения, паспорт (с маской ввода), изучаемый иностранный язык, оценки на вступительных экзаменах по математике и физике. В режиме таблицы введите 5 – 6 записей.

Задание 2.

Создайте в режиме мастера однотоабличную форму «Студент». Перейдите в режим конструктора форм и отредактируйте форму «Студент» так, чтобы надписи располагались наиболее компактным способом, а отображение содержимого полей было приподнятым над поверхностью листа. Перейдите в режим таблицы и введите еще 3 – 4 записи.

Задание 3.

Для упрощения ввода повторяющихся сведений об иностранном языке измените форму поля «Иностранный язык» с простого поля на поле со списком, используя сначала данные из этого поля таблицы «Студент», а затем создав список значений из 4 языков. В каждом случае добавить по 2 – 3 записи в режиме формы.

Задание 4.

Повторите зад.3, применив вместо поля со списком, список. Отметьте различия.

5.9. Основы создания многотабличных форм

Составная многотабличная форма создается для работы с несколькими взаимосвязанными таблицами. Многотабличная форма может состоять из *основной части* и одной или нескольких *подчиненных включаемых форм*, то есть быть *составной*. Подчиненная форма может быть построена на основе как подчиненной, так и главной относительно таблицы-источника основной части формы.

Многотабличная форма может не иметь включаемых форм. В этом случае в форму кроме полей таблицы, на основе которой она строится, добавляются поля из связанной таблицы, являющейся главной относительно основной таблицы формы.

Многотабличная форма может быть создана в режиме конструктора с помощью мастера форм.

При создании многотабличной формы средствами Access могут использоваться различные приемы. Наиболее технологичным является, очевидно, первоначальное создание форм с помощью мастера с последующей их доработкой в конструкторе. Мастер упрощает процесс создания формы, т. к. автоматически выполняет большинство требуемых операций.

5.9.1. Создание многотабличной формы с помощью мастера

Мастер форм предоставляет возможность выбрать включаемые в форму поля из нескольких взаимосвязанных таблиц, а также запросов. При этом используются различные способы построения многотабличной формы.

5.9.2. Способы построения многотабличной формы

Явное включение подчиненной формы

При использовании мастера форм подчиненная форма строится только таблицы, которая является подчиненной по отношению к таблице-источнику основной части. Подчиненная форма отображает данные из всех записей подчиненной таблицы, которые связаны с записью главной таблицы, отображаемой в основной части формы. С помощью мастера можно создать составную форму, включающую одну или несколько подчиненных форм стандартного вида с выбранными полями.

Вызов связанной формы по кнопке

Мастер форм позволяет создать связанные формы, которые не включаются непосредственно в главную форму. При этом вместо подчиненной формы включается лишь кнопка, представляющая ее. При необходимости связанная подчиненная форма вызывается на экран нажатием этой кнопки. Открывающееся содержимое связанной формы синхронизировано с текущей записью формы. Этот способ построения многотабличной формы удобен в сложных формах, перегруженных большим числом элементов управления, а также если пользователю нет необходимости постоянно видеть связанные данные.

5.9.3. Многотабличная форма без подчиненных и связанных форм

Составная многотабличная форма, построенная мастером, может не включать подчиненные и связанные формы. Такая форма создается, если необходимо отображать записи подчиненной таблицы, дополненные полями из одной или

нескольких главных таблиц. В этом случае записеобразующим источником данных, выводимых в форме, является запись подчиненной таблицы. В одной записи, отображаемой в форме, содержатся значения полей из записи подчиненной таблицы и полей из единственной связанной с ней записи главной таблицы.

Замечание

Данные из главной таблицы, отображаемые в такой форме, будут повторяться, т. е. в результате объединения двух нормализованных таблиц образуется ненормализованная структура данных.

5.9.4. Мнготабличная форма на основе запроса

Создание формы на основе полей из нескольких взаимосвязанных таблиц с помощью мастера является для пользователя достаточно простой процедурой. Однако, если в базе данных уже имеется запрос, в котором выбраны таблицы, нужные для получения результата, определены поля, которые необходимо включить в результат, целесообразно создавать форму на основе ранее созданного запроса. В этом случае мастер использует результаты работы, проделанной пользователем ранее.

Для запроса, в котором записи создаются объединением записей главной таблицы с каждой из связанных записей подчиненной, мастер создает форму так же, как если бы ему были заданы исходные таблицы. Это связано с тем, что форма, обеспечивающая однократное отображение данных, должна базироваться на исходных нормализованных таблицах. Заметим, что результатом мнготабличного запроса является ненормализованная таблица с поощряющимися значениями.

Полученная с помощью мастера составная форма при необходимости может быть отредактирована, в том числе дополнена другими включаемыми формами.

5.9.5. Создание формы мастером, выбор таблиц и полей

Для создания формы в окне **База данных (Database)** в группе **Объекты (Objects)** перейти к строке **Формы (Forms)** и нажать кнопку **Создать (New)**. В диалоговом окне **Новая форма (New Form)** выбрать режим создания — **Мастер форм (Form Wizard)** и можно сразу в качестве источника данных основной части формы выбрать из списка таблицу и запрос.

В открывшемся первый раз диалоговом окне **Создание форм (Form Wizard)** последовательно выбираются таблицы из списка **Таблицы/Запросы (Tables/Queries)**, начиная с главной, и **Доступные поля (Available Fields)** таблиц, включаемые в форму.

Внимание

Если выбраны поля из таблиц, для которых не определена связь в схеме данных, появится сообщение о невозможности создать форму. При этом мастер или предлагает изменить состав полей, или выводит схему данных для возможного определения нужных связей. После изменения связей мастер должен запускаться заново.

5.9.6. Выбор варианта создания мнготабличной формы, отображение данных главной и подчиненной таблиц

В новом сеансе окна **Создание форм (Form Wizard)** в рамке **Выберите тип представления данных (How do you want to view your data?)** надо выделить таблицу,

которая является источником основной части формы. Если таблица была ранее выбрана в окне **Новая форма** (New Form), то она уже выделена.

Если таблица-источник основной части формы является главной по отношению к другой таблице, тоже выбранной для формы, то в окне **Создание форм** (Form Wizard) можно выбрать один из двух возможных типов подключения формы:

□ Для **непосредственного включения** подчиненной формы надо отметить переключатель **Подчиненные формы** (Form with subform(s)).

□ Для **включения кнопки, вызывающей связанную форму**, надо установить переключатель **Связанные формы** (Linked forms).

Если был выбран первый вариант, далее предоставляется возможность в следующем сеансе диалогового окна выбрать вид **ленточная** (Tabular) или **табличная** (Datasheet) для подчиненной формы.

Замечание

Если таблица-источник основной части формы является подчиненной по отношению к другой таблице, тоже выбранной для формы, то создаваемая многотабличная форма не будет включать подчиненную форму. Однако форма будет содержать поля из главной таблицы. В окне **Создание форм** (Form Wizard) автоматически будет установлен тип формы **Одиночная форма** (Single Form). Далее предоставляется возможность выбрать вид формы; **В один столбец** (Columnar), **Ленточный** (Tabular) или **Табличный** (Datasheet).

В следующем сеансе диалогового окна предоставляется возможность выбрать стиль оформления, который определяет общий вид формы, отображение надписей и значений полей в форме.

5.9.7. Завершение создания формы мастером

В последнем сеансе диалогового окна **Создание форм** (Form Wizard) можно изменить имена основной и подчиненной формы, если был выбран вариант с непосредственным включением подчиненной формы, или имя каждой из связанных форм, если был выбран вариант включения кнопки, вызывающей связанную форму. В том же окне можно выбирать дальнейшие действия: **Открыть главную форму для просмотра и ввода данных** (Open the form to view or enter information) или **Изменить макет форм** (Modify the form's design).

Если был выбран вариант **Открыть главную форму для просмотра и ввода данных** (Open the form to view or enter information), автоматически выводится форма с данными из таблиц, с которыми связана форма. После нажатия кнопки **Готово** (Finish) мастер завершает создание формы. Сохранение форм производится автоматически.

5.9.8. Доработка формы в режиме конструктора

При выборе варианта **Изменить макет форм** (Modify the foil's design) форма выводится в режиме конструктора, позволяющем выполнить нужную доработку. Мастер уже разместил в макете форм заданные поля из таблиц-источников основной (главной) и подчиненных формы.

При установке курсора на основной форме при нажатой кнопке панели конструктора форм **Список полей** (Field List) делается доступным список полей таблицы — источника этой формы. При установке курсора на подчиненной форме

становится доступным список полей таблицы источника подчиненной формы.

В процессе доработки формы, используя технику редактирования формы, можно перемещать поля в основной форме, менять их свойства, в том числе шрифт и размеры, подпись поля, текст в заголовке формы. Аналогичные действия по доработке выполняются для подчиненной формы.

Следует отметить исключительную простоту и универсальность всех действий по изменению размеров, перемещению любого элемента, редактированию подписей и т. п. действий. На этапе доработки можно выполнить и более сложные действия по редактированию формы и настроить составную форму в соответствии с любыми требованиями к интерфейсу пользователя. Ниже рассмотрены более подробно возможности Конструктора форм (Form Design) при создании и редактировании форм.

5.9.9. Создание и редактирование многотабличной формы в режиме конструктора

Средствами Конструктора форм (Form Design) можно полностью создать многотабличную форму. Ранее разработанную и сохраненную форму можно в любой момент модифицировать этим конструктором. При создании и редактировании формы можно добавлять новые поля и надписи, включать поля со списком, создавать кнопки, добавлять подчиненные формы, внедрять объекты из других приложений, например, рисунков, диаграмм, а также изменять расположение отдельных полей, их отображение и подписи. При редактировании могут быть выполнены любые изменения и удаление имеющихся элементов, а также добавление новых.

5.10. Создание новой формы конструктором

Конструирование формы начинается после выбора объекта **Формы** (Forms) в окне **База данных** (Database) и нажатия кнопки **Создать** (New), которая вызывает окно **Новая форма** (New Form). В этом диалоговом окне нужно выбрать режим создания формы **Конструктор** (Design View). Можно сразу в качестве базового источника данных формы выбрать из списка нужную таблицу (запрос).

Вызов конструктора может быть осуществлен из окна базы данных также активизацией значка **Создание формы в режиме конструктора** (Create form in Design view), позволяющего открыть сразу окно конструктора форм.

После вызова конструктора форм появляется окно конструктора с именем формы по умолчанию — **Форма 1** (Form 1) и открывается доступ к списку полей таблицы, выбранной в качестве базового источника данных.

Замечание

Все элементы, добавляемые в форму, являются элементами управления. Примерами разных элементов управления служат поля, надписи, списки, переключатели, кнопки и линии. Способ создания элемента управления зависит от того, какой элемент создается: присоединенный, свободный или вычисляемый.

5.10.1. Включение полей в новую форму

Для включения нового поля базового источника записей в форму предварительно

должен быть отображен на экране список полей этого источника. Для отображения списка полей, доступных для внесения в форму при редактировании формы, следует выполнить команду меню **Вид/Список полей** (View(Field List) или нажать соответствующую кнопку панели **Конструктора форм** (Form Design). Список полей доступен только в режиме конструктора. Из списка поле можно перетаскивать в нужное место формы с помощью мыши. При этом размещается поле и связанная с ним подпись. В самом поле будет отображено имя поля таблицы базы данных, а в качестве надписи будет использовано значение из свойства поля **Подпись** (Caption), которое было определено при конструировании таблицы.

Добавление в форму полей может быть выполнено с помощью кнопок панели элементов **Поле** (Text Box) типа **Свободный** (Unbound) и **Поле со списком** (Combo Box). Для включения обычного поля надо нажать на панели элементов кнопку **Поле** (Text Box) и вычертить курсором поле в нужном месте формы. Далее надо установить связь созданного элемента с полем таблицы-источника формы.

Если заранее не была определена таблица-источник записей формы, нужно ее выбрать. Для этого надо вызвать окно свойств формы, дважды щелкнув мышью на области выделения формы. На вкладке **Данные** (Data) в окне свойств выбрать в строке **Источник записей** нужную таблицу и закрыть окно свойств.

Для установки связи создаваемого поля формы с полем таблицы-источника формы надо выделить создаваемое поле в форме и вызвать окно его свойств. В этом окне на вкладке **Данные** (Data) в строке **Данные** (ControlSource) следует выбрать нужное поле из списка доступных полей источника и закрыть окно свойств.

Включение полей нескольких источников. Для включения в форму полей из нескольких базовых источников записей формы должен быть построен и выбран запрос, включающий эти таблицы. Такой запрос может быть создан заранее. Запросы наряду с таблицами базы доступны в списке строки **Источник записей** в окне свойств формы. При отсутствии нужного запроса можно вызвать построитель нажатием кнопки, появляющейся при установке курсора за названной строкой, и построить нужный запрос.

Если в форме в качестве источника записей была определена одна таблица, то для добавления полей из другой таблицы также нужно воспользоваться построителем.

5.10.2. Добавление подчиненной формы и ее редактирование

В любую форму, в том числе однотабличную, можно добавить подчиненную форму. При этом можно использовать ранее созданную форму, либо сконструировать ее в процессе построения многотабличной формы. Можно, не создавая форму, перетащить из окна базы данных в область данных основной формы таблицу, являющуюся источником записей подчиненной формы.

Встраивание подчиненной формы без помощи мастера. Для встраивания подчиненной формы в режиме конструктора без помощи мастера необходимо нажать на панели элементов кнопку **Подчиненная форма/отчет** (Subform/Subreport). Кнопка панели элементов **Мастера** (Control Wizards) не должна быть включена. Затем установить курсор на место размещения создаваемого объекта в форме, нажать кнопку мыши и, не отпуская ее, растянуть рамку подчиненной формы до нужного размера.

В режиме конструктора в качестве подчиненной формы может быть взята форма,

построенная не только для подчиненной таблицы относительно источника основной формы, но и для главной таблицы. В последнем случае, очевидно, для одной записи основной формы будут отображаться данные только из одной записи главной таблицы.

Использование мастера для включения подчиненной формы. В процессе конструирования многотабличной формы для включения подчиненной формы можно использовать мастер, который позволяет не только включить, но и создать нужную подчиненную форму. Чтобы при нажатии кнопки **Подчиненная форма/отчет** (Subform/Subreport) панели элементов запускался мастер, необходимо предварительно на этой панели нажать кнопку **Мастера** (Control Wizards). В диалоговом окне мастера можно выбрать таблицу (или запрос), на основе которой должна быть создана подчиненная форма, или существующую форму и далее указать поля связи с подчиненной формой.

Использование метода Drag-and-Drop. Особым способом включения подчиненной формы является использование метода Drag-and-Drop (перетащить и отпустить). Таким способом можно перетащить в форму из окна базы данных готовую подчиненную форму или таблицу, для которой нужно создать подчиненную форму.

Во втором случае для создания подчиненной формы подключается мастер. В результате встраивается рамка подчиненной формы и запускается мастер построения подчиненной формы. Мастер предлагает использовать в качестве поля для связи форм внешний ключ подчиненной таблицы и автоматически создает многозаписевую форму.

Изменение дизайна формы. При определении вида как основной, так и подчиненной формы можно использовать свойства самой формы, а не отдельных ее элементов. В частности, можно убрать область маркировки (слева), полосу прокрутки, поле нумерации записей, тип границы. Для перехода к просмотру и редактированию свойств формы надо установить курсор в области выделения формы, вызвать контекстно-зависимое меню, нажав правую кнопку мыши, и выбрать опцию **Свойства** (Properties). Например, при доработке формы можно удалить линии, разделяющие области заголовка, данных и примечаний. Для этого достаточно установить значение **Нет** (No) для свойства формы **Разделительные линии** (Dividing Lines) на вкладке **Макет** (Format). Для выбора или отключения полосы прокрутки формы устанавливается соответствующее значение свойства **Полосы прокрутки** (Scroll Bars).

Задание №5.

Для базы данных «Кафедры» (л/р №4) в режиме конструктора создать форму «Кафедра».

5.11. Вычисления в форме

5.11.1. Вычисления в каждой записи формы

Чтобы произвести вычисления на основе данных одной записи в форме, необходимо создать вычисляемый элемент управления, в который записывается выражение. Для создания вычисляемого элемента управления надо открыть форму в режиме конструктора и, нажав на панели элементов кнопку **Поле** (Text Box), разместить элемент управления в нужном месте. Затем в элемент управления вводится

выражение. Выражение должно начинаться со знака равенства (=). В качестве операндов выражения чаще всего используется имена полей, константы, а в качестве операторов — знаки арифметических операций.

Рассмотрим пример вычисляемого поля. Пусть таблица или запрос, на котором строится форма, содержит данные о должностных окладах и надбавках и имеет поля: Оклад — оклад работника; Надбавка — персональная надбавка работника. В форме можно подсчитать и вывести заработную плату работника, создав вычисляемый элемент управления и записав в него выражение:

$$=([\text{Оклад}]+[\text{Надбавка}])*0.87$$

Выражение может быть введено в строку Данные (ControlSource) в окне свойств элемента управления. При этом можно воспользоваться построителем выражений, который позволяет упростить создание выражения.

Задание 6.

В форму «Абитуриент» добавить вычисляемое поле, определяющее сумму баллов, набранных студентом по математике и по физике.

5.11.2. Вычисление итоговых значений

Вычисление итоговых значений для записей формы выполняется с помощью встроенных статистических функций, записываемых в качестве выражения в вычисляемых элементах управления. Например, можно создать элемент управления для вычисления среднего оклада:

$$= \text{Mean} ([\text{Оклад}])$$

Вычисляемый элемент управления, создаваемый для расчета итогового значения, нужно размещать в области примечания формы.

В статистической функции нельзя использовать имена других вычисляемых элементов управления. При необходимости следует повторить выражение в элементе управления.

Вычисление итогового значения для записей подчиненной формы и вывод его в основной форме. Основная форма и ее подчиненная форма чаще всего строятся на основе таблиц, между которыми установлена связь типа один-ко-многим. При этом в основной форме выводится одна запись, а в подчиненной форме — несколько записей, которые подчинены ей. При расчете итогового значения для группы записей подчиненной формы вычисляемое поле может быть отображено в записи основной формы. При этом элемент управления должен помещаться в область примечаний подчиненной формы.

Замечание

Значение вычисляемого поля отображается в форме, но не может быть сохранено в таблице.

Задание 7. В форме «Кафедра» создать вычисляемое поле, информирующее о числе работников текущей кафедры (Функция подсчета числа записей count()).

5.12. Ограничения доступа к данным через форму

5.12.1. Защита данных поля от изменений

Для защиты данных поля от изменения используется свойство **Блокировка** (Locked). Блокировка может быть установлена для любого поля формы. Чтобы защитить таким образом поле, надо установить курсор в его рамке и с помощью контекстно-зависимого меню вызвать свойства поля. В окне свойств на вкладке **Данные** (Data) в строке **Блокировка** (Locked) выбрать **Да** (Yes). После установки этого свойства поле доступно только для чтения.

Установка ограничений на корректировку записей через форму

Работая с формой, можно сделать *записи* доступными только для чтения, если задать значение **Нет** (No) для свойств всей формы: **Разрешить добавление** (Allow Additions), **Разрешить удаление** (Allow Deletions) и **Разрешить изменение** (Allow Edits). Этого можно также добиться, выбрав для свойства **Тип набора записей** (Recordset Type) значение **Статический набор** (Snapshot). Указанные свойства могут устанавливаться независимо друг от друга. Например, при запрете на изменение записей могут быть разрешены добавление и удаление записей.

Замечание

Свойство формы **Блокировка записей** (Record Locks) определяет способы блокировки записей при обновлении содержимого сетевой базы данных. Это необходимо для защиты данных при попытке двух пользователей одновременно изменить одну и ту же запись. Когда один пользователь изменяет запись, запись автоматически блокируется. При этом другие пользователи не могут изменять эту запись до завершения работы с ней первого пользователя.

Свойство **Ввод данных** (Data Entry) определяет режим открытия формы, при котором разрешен только ввод новых записей, просмотр существующих записей при этом недоступен. При открытии формы будет выводиться только пустая запись, которую можно заполнять.

5.12.2. Защита данных подчиненной формы от изменений

Если в качестве подчиненной используется форма, встроенная конструктором на основе главной таблицы, содержащей справочные данные, необходимые только для расшифровки вводимых ключевых полей в основную часть формы. В этом случае целесообразно запретить обновление таких справочных данных через подчиненную форму. Защитить содержимое всех полей подчиненной формы, сделав его доступным только для чтения, позволяет свойство **Блокировка** (Locked), установленное для подчиненной формы.

Чтобы защитить данные подчиненной формы от изменений, надо вызвать свойства подчиненной формы. В окне свойств надо перейти на вкладку **Данные** (Data). В строке **Блокировка** (Locked) выбрать **Да** (Yes). При этом элемент управления функционирует нормально, но изменение, добавление и удаление данных в полях не допускаются.

Замечание

По умолчанию для всех элементов управления, кроме свободной рамки объекта, в строке **Блокировка** (Locked) устанавливается значение **Нет** (No).

Задание 8.

В базе данных «Литература» создать форму, отображающую данные всех ее таблиц, открыв в ней доступ к данным только для добавления записей.

Зачетное задание.

Создать форму «Литература», в разработанной ранее базе данных. Закрыть доступ для любого изменения данных. Форма должна представлять собой справочник.

6. Лабораторная работа №6. Создание запросов в среде Access

6.1. Цель работы

Выработка навыков разработки запросов в пользовательском режиме СУБД Ms Access.

6.2. Программное обеспечение

Для проведения лабораторной работы требуется Ms Access 97/2000. Рекомендуется Ms Access 2000.

6.3. Назначение и виды запросов

Запрос позволяет выбрать необходимые данные из одной или нескольких взаимосвязанных таблиц, произвести вычисления и получить результат в виде таблицы. Через запрос можно производить обновление данных в таблицах, добавление и удаление записей.

С помощью запроса можно выполнить следующие виды обработки данных:

- Выбрать записи, удовлетворяющие условиям отбора.
- Включить в результирующую таблицу запроса заданные пользователем поля.
- Произвести вычисления в каждой из полученных записей.
- Сгруппировать записи, которые имеют одинаковые значения в одном или нескольких полях, в одну запись с одновременным выполнением групповых операций над другими полями.
- Произвести обновление полей в выбранном подмножестве записей.
- Создать новую таблицу базы данных, используя данные из существующих таблиц.
- Удалить выбранное подмножество записей из таблицы базы данных.
- Добавить выбранное подмножество записей в другую таблицу.
- Многотабличный запрос позволяет сформировать записи результата путем объединения взаимосвязанных записей из таблиц базы данных и включения нужных полей из нескольких таблиц. В частности, при объединении двух нормализованных связанных одно-многочисленными отношениями таблиц результирующая запись образуется на основе записи подчиненной таблицы, в которую добавляются поля из связанной записи в главной таблице. Заметим, что подобное объединение формирует новую таблицу, которая не является нормализованной.

Последовательное выполнение ряда *запросов по примеру* позволяет решать достаточно сложные задачи, не прибегая к программированию.

В Access может быть создано несколько видов запроса:

- Запрос на выборку — выбирает данные из взаимосвязанных таблиц и других запросов. Результатом его является таблица, которая существует до закрытия запроса. На основе этого вида запроса строятся запросы другого вида.

Замечание

Таблицу с результатами запроса можно использовать для работы с данными в таблицах, на которых построен запрос. Например, можно корректировать данные в исходных таблицах базы данных через таблицу с результатами запроса.

- Запрос на создание таблицы — использует запрос на выборку, но в отличие от него результат запроса сохраняется в новой таблице.
- Запросы на обновление, добавление, удаление — являются запросами действия, в результате выполнения которых изменяются данные в таблицах.

6.4. Создание запроса

Основные принципы конструирования различных запросов по технологии Query By Example (QBE) заложены в технике конструирования *запроса на выборку*, являющегося основой всех видов запроса.

Запрос на выборку позволяет достаточно просто выбрать данные из одной или нескольких взаимосвязанных таблиц. Результаты выполнения запроса отображаются в виде таблицы, существующей до закрытия запроса. Записи этой таблицы формируются на основе записей в исходных таблицах и связей между этими таблицами в соответствии с задаваемыми запросом условиями отбора. Поля, составляющие записи результата, указываются пользователем в бланке запроса.

Таблица с результатами запроса может применяться при дальнейшей обработке данных. В запросе на выборку могут использоваться не только таблицы базы данных, но и ранее созданные запросы, а вернее, таблицы, являющиеся результатом их выполнения. При этом нет необходимости сохранять таблицы, получаемые в результате выполнения ранее созданных запросов.

Однако в ряде случаев непосредственное использование запроса невозможно. В этом случае включаемый запрос надо преобразовать в *запрос на создание таблицы*, который отличается от запроса на выборку тем, что результат сохраняется как таблица базы данных. После чего эта таблица может быть включена в состав других таблиц, на которых может строиться запрос.

Результаты выполнения запроса выводятся в режиме таблицы. Окно запроса в режиме таблицы аналогично окну просмотра таблицы базы данных. В этом режиме становится активной панель инструментов **Запрос в режиме таблицы** (Query Datasheet, назначение кнопок этой панели аналогично панели **Таблица в режиме таблицы** (Table Datasheet)).

Несмотря на то, что поля результирующей таблицы принадлежат, как правило, нескольким таблицам базы данных, с ними можно работать так, как если бы они принадлежали одной таблице. Можно менять данные в таблице результатов запроса на выборку, и сделанные изменения будут внесены в базовые таблицы.

6.5. Окно запроса

Для создания запроса надо в окне **База данных** (Database) перейти на вкладку **Запросы** (Queries) и нажать кнопку **Создать** (New). В открывшемся окне **Новый запрос** (New Query) выбрать **Конструктор** (Design View).

В Access 2000 можно сразу перейти к созданию нового запроса в режиме конструктора, выбрав на вкладке **Запросы (Queries)** команду **Создание запроса в режиме конструктора (Create query in Design view)**.

Далее в открывшемся окне **Добавление таблицы (Show Table)** нужно выбрать используемые в запросе таблицы и нажать **кнопку Добавить (Add)**. Для выхода из окна следует нажать **кнопку Закрывать (Close)**.

В результате появится окно конструктора запросов **Запрос <номер> (Query <number>): запрос на выборку (Select Query)**.

Окно конструктора запросов разделено на две панели. Верхняя панель содержит *схему данных запроса*, которая включает выбранные для данного запроса таблицы. Таблицы представлены списками полей. Нижняя панель является *бланком запроса QBE*, который нужно заполнить.

6.6. Схема данных запроса

В окне конструктора запроса отображаются выбранные таблицы со списком полей и одно-многозначные связи между ними, имеющиеся в схеме данных базы. Первая строка в списке полей, отмеченная звездочкой *, обозначает все множество полей таблицы. Кроме того, Access автоматически устанавливает связи для объединения таблиц, если таблицы имеют поля с одинаковыми именами и типами данных, даже если связи не были установлены в схеме данных. *Связи-объединения*, которые не были установлены Access автоматически, может установить пользователь, перетащив задействованные в связи поля из одного списка полей в другой.

При использовании в запросе других запросов или таблиц, не представленных в схеме данных базы, с ними также могут быть установлены *связи-объединения*.

6.7. Бланк запроса QBE

Бланк запроса QBE представлен в виде таблицы в нижней панели окна запроса. Такая таблица предназначена для конструирования структуры таблицы результата запроса и условий выборки данных из исходных таблиц. Первоначально эта таблица пуста.

Каждый столбец бланка относится к одному полю, с которым нужно работать в запросе. Поля могут использоваться для включения их в таблицу, являющуюся результатом выполнения запроса, для задания сортировки по ним, а также для задания условий отбора записей.

При заполнении бланка запроса:

- В строку **Поле (Field)** включаются имена полей, используемых в запросе.
- В строке **Сортировка (Sort)** выбирается порядок сортировки записей результата.
- В строке **Вывод на экран (Show)** отмечаются поля, которые должны быть включены в результирующую таблицу.
- В строке **Условие отбора (Criteria)** задаются условия отбора записей.
- В строке **или (or)** задаются альтернативные условия отбора записей.

В ряде случаев в бланке запроса наряду с именем поля нужно отображать имя соответствующей таблицы, например, если поля имеют одинаковые имена в разных

таблицах. Для отображения имен таблиц в строке бланка должна быть выполнена команда **Вид|Имена таблиц** (View|Table Names) или нажата соответствующая кнопка панели конструктора запросов. В результате выполнения команды в бланке появится строка **Имя таблицы** (Table Name).

6.8. Поля бланка запроса

Каждый столбец бланка запроса соответствует одному из полей таблиц, на которых строится запрос. Кроме того, здесь может размещаться *вычисляемое поле*, значение которого вычисляется на основе значений других полей записи результата, или итоговое поле для групп записей, использующее одну из встроенных групповых функций Access.

Для включения нужных полей из таблиц в соответствующие столбцы запроса можно воспользоваться следующими приемами:

- В первой строке бланка запроса **Поле** (Field) щелчком мыши вызвать появление кнопки списка и выбрать из списка нужное поле. Список содержит все поля таблиц, представленных в бланке запроса.
- Перетащить нужное поле из списка полей таблицы в схему данных запроса в первую строку бланка запроса.
- Дважды щелкнуть на имени поля таблицы в схеме данных запроса.
- Для включения всех полей таблицы можно перетащить или дважды щелкнуть на символе ">>" в списке полей таблицы в схеме данных запроса.

Задание 1.

Создать на основе базы данных «Кафедры» запрос на выборку данных о том, на какой кафедре работает каждый преподаватель, и какие предметы он читает.

6.9. Модификация запроса

Добавление таблицы в схему данных запроса осуществляется с помощью команды меню **Запрос Добавить таблицу** (Query Show Table) или нажатием соответствующей кнопки панели **Конструктор запросов** (Query Design). Команда добавления может быть выполнена также через контекстное меню, вызываемое на схеме данных запроса.

Добавление поля в бланк запроса осуществляется с помощью одного из действий, рассмотренных выше, например, перетаскиванием имени поля из таблицы в схему данных в нужное место бланка. Все столбцы полей справа от него передвинутся на один столбец вправо.

Удаление поля в бланке запроса требует предварительного выделения соответствующего столбца. Для этого надо переместить курсор в область маркировки столбца сверху, где он примет вид черной стрелки, направленной вниз, и щелкнуть кнопкой мыши. Далее нужно нажать клавишу <DELETE> или выполнить команду **Правка|Удалить столбцы** (Edit|Delete Columns).

Для *перемещения поля в бланке запроса* надо выделить соответствующий столбец и перетащить его в новую позицию с помощью мыши. Столбец, на место которого перемещен новый, как и все столбцы справа от него, будет сдвинут вправо.

Задание 2.

Добавить в запрос поля связи с отображением их на экране. При этом поля должны находиться на самых левых позициях.

6.10. Условия отбора записей

Условия отбора записей могут задаваться в бланке запроса для одного или нескольких полей в строке **Условие отбора** (Criteria).

Условием отбора является *выражение*, которое состоит из *операторов сравнения* и *операндов*, используемых для сравнения.

В качестве операндов выражения могут использоваться: *литералы*, *константы*, *идентификаторы (ссылки)*.

Литералами являются конкретные значения, воспринимаемые Access так, как они записаны. В качестве литералов могут быть использованы числа, текстовые строки, даты. Текстовые строки заключаются в двойные кавычки, даты — в символы "#". Например, 275, "Качество", #15-Июня-02#.

Константами являются не изменяющиеся значения, которые определены в Access, например, "True", "False", "Да", "Нет", "Null".

Идентификатор представляет собой *ссылку* на поле, элемент управления или свойство. Идентификаторами могут быть имена полей, таблиц, форм, отчетов и т. д. Они должны заключаться в квадратные скобки. Как правило, Access производит автоматическую подстановку скобок.

Во многих случаях ссылка на конкретное значение должна указывать точное его местоположение в иерархии объектов базы данных, начиная с объекта верхнего уровня. Если необходимо указать *ссылку на поле* в конкретной таблице, форме, отчете, то перед именем поля ставится имя таблицы, формы, отчета, также заключенное в квадратные скобки и отделенное от имени поля восклицательным знаком. Например, ссылка на поле в таблице примет вид: "[Имя таблицы]![Имя поля]", а ссылка на свойство DefaultValue элемента управления "Год рождения" в форме СТУДЕНТ — "Forms![Абитуриент]![Год рождения]. DefaultValue".

Операторами сравнения и логическими операторами, использование которых допускается в *выражении условия отбора*, являются операторы:

=, <, >, <>, Between, In, Like, And, Or, Not

Они определяют операцию над одним или несколькими операндами.

Если выражение в условии отбора не содержит оператора, то по умолчанию используется оператор =.

Текстовые значения в выражении, если они содержат пробелы или знаки препинания, вводятся в двойных кавычках. В противном случае кавычки можно не вводить, они будут добавлены автоматически.

Допускается использование *операторов шаблона* — * (звездочка) и ? (вопросительный знак).

Оператор Between позволяет задать интервал для числового значения. Например, Between 10 And 100 задает интервал от 10 до 100.

Оператор In позволяет выполнить проверку на равенство любому значению из списка, который задается в круглых скобках. Например, In("Математика", "Информатика", "История").

Оператор Like позволяет использовать образцы, использующие символы шаблона, при поиске в текстовых полях. Например, Like «ИСО* ».

Логические операции "И", "ИЛИ". Условия отбора, заданные в одной строке, связываются по умолчанию с помощью логической операции "И", заданные в разных строках - с помощью логической операции "ИЛИ". Эти операций могут быть также заданы явно в выражении условия отбора с помощью операторов AND и OR, соответственно.

Задание 3.

Установить условие отбора для запроса такое, чтобы на экране отображались бы только работники кафедры конструирования и производства РЭС.

6.11. Построитель выражений

Условие отбора можно сформировать с помощью построителя выражений. Перейти в окно **Построитель выражений** (Expression Builder) можно, нажав кнопку **Построить** (Build) панели инструментов конструктора запросов или выбрав команду **Построить** (Build...) в контекстно-зависимом меню. Курсор мыши должен быть установлен предварительно в ячейке ввода условия отбора.

После ввода выражения в бланк и нажатия клавиши <Enter> Access выполняет синтаксический анализ выражения и отображает его в соответствии с результатами этого анализа.

6.12. Вычисляемые поля

В запросе над полями могут производиться вычисления. Результат вычисления образует вычисляемое поле в таблице, создаваемой по запросу. При каждом выполнении запроса производится вычисление с использованием текущих значений полей.

При вычислениях могут использоваться арифметические выражения и встроенные функции Access.

6.13. Арифметические выражения

Выражение вводится в бланк запроса в пустую ячейку строки **Поле** (Field). Затем, после нажатия клавиши <Enter> или перевода курсора в другую ячейку перед выражением в этой ячейке добавляется имя поля **ВыражениеN** (ExprN), где N — целое число, увеличивающееся на единицу для каждого нового создаваемого вычисляемого поля в запросе. Имя вычисляемого поля, стоящее перед выражением, отделяется от него двоеточием. Например,

<Выражение1>: [Оклад]+[Надбавка] ,
где Оклад и Надбавка — **имена полей**.

Имя вычисляемого поля ("Выражение1") становится заголовком столбца в таблице с результатами выполнения запроса. Это имя можно изменить.

Для вычисляемых полей допускается сортировка, задание условий отбора и расчет итоговых значений, как и для любых других полей.

6.14. Встроенные функции

В Access имеются *встроенные функции*, которые можно использовать в вычисляемых полях:

- Функция **Date** формирует текущую дату
- Функция **Month** выделяет месяц из значения поля, содержащего дату
- Функция **DLookup** возвращает значение конкретного поля из записи связанной таблицы, не участвующей в запросе
- *Статистические функции* над полями подмножества записей, вычисляющие *среднее значение, сумму, минимальное, максимальное значение*

Для записи выражения может быть использован **Построитель выражений** (Expression Builder), который вызывается, кнопкой **Построить** (Build) панели инструментов.

Например, для выборки всех студентов, родившихся в заданном месяце, построим в вычисляемом поле выражение

Month([Абитуриент]![ДАТАР]),

где ДАТАР — поле типа **Дата/время** (Date/Time) с датой рождения в таблице СТУДЕНТ.

Задание 4.

Получить у преподавателя базу данных «Абитуриент». Создать запрос, формирующий список совершеннолетних абитуриентов, поступающих на дневное отделение на специальность 130200 – Авиационные двигатели и энергетические установки. Специальность – поле «Napr», дата рождения – поле «Datar», форма обучения – поле «Nar_o» значение поля «1» – форма обучения дневная бюджетная, значение поля «2» – контрактная дневная.

6.15. Присвоение пользовательских имен вычисляемым полям

Пользователь имеет возможность присвоить новое имя вычисляемому полю, используя один из следующих способов:

- Изменение имени поля в запросе. В режиме конструктора запроса в бланке запроса вместо **ВыражениеN** (ExprN) ввести новое имя
- Изменение подписи поля в свойствах поля. Установить курсор на вычисляемое поле в бланке запроса и открыть окно **Свойства поля** (Field Properties), щелкнув правой кнопкой мыши и выбрав в контекстном меню опцию **Свойства** (Properties). В окне **Свойства поля** (Field Properties) на вкладке **Общие** (General) ввести нужный текст подписи в строку **Подпись поля** (Caption)

6.16. Параметры запроса

Конкретное значение поля в условии отбора может вводиться, как было показано выше, непосредственно в бланк запроса. Однако при решении практических задач необходимо вводить значение поля в диалоге с пользователем в процессе выполнения запроса. Для того чтобы выводилось диалоговое окно, обеспечивающее ввод конкретного значения поля в условия отбора, нужно определить *параметр запроса*.

6.16.1. Определение параметра запроса по значению поля

Имя параметра запроса может задаваться непосредственно в строке **Условия отбора** (Criteria) в квадратных скобках. При выполнении запроса это имя появится в диалоговом окне **Введите значение параметра** (Enter Parameter Value).

Например, если в условии отбора рассмотренного выше запроса номер месяца (5) заменить на имя параметра <номер месяца>, то при выполнении запроса будет выводиться диалоговое окно, позволяющее ввести значение этого параметра запроса.

6.16.2. Определение нескольких параметров запроса

Если в запрос вводится несколько параметров, то порядок их ввода через диалоговые окна определяется порядком расположения полей с параметрами в бланке запроса. Чтобы иметь возможность ввести несколько значений для одного поля при выполнении запроса, можно в условии отбора этого поля определить несколько параметров. Например, для отбора записей по двум группам в условии отбора поля НГ можно записать два параметра, связанных логической операцией OR: <номер группы > OR <еще один номер>.

6.16.3. Параметр запроса для ввода значения операнда выражения

Параметры запроса могут быть использованы не только для отбора по значению поля, но и для ввода значения операнда в выражениях условий отбора или вычисляемых полей. Здесь в условии отбора используется функция Year, которая выделяет год из даты. Параметр запроса - [Год рождения], фигурирует как величина, с которой сравнивается значение данной функции.

6.17. **Корректировка данных средствами запроса**

6.17.1. Запрос на обновление

Для обновления данных в полях базовых таблиц может быть использован **Запрос на обновление** (Update Query). Изменения вносятся в группу записей, отбираемых с помощью указанных пользователем условий отбора. Значения для изменений в полях определяются в бланке запроса в строке **Обновление** (Update To).

Для того чтобы создать *запрос на обновление* (Update Query) первоначально создается *запрос на выборку* (Select Query), а затем в окне конструктора запросов он преобразуется в *запрос на обновление* выбором пункта **Обновление** (Update Query) из списка **Тип запроса** (Query Type) или команды меню **Запрос|Обновление** (Query|Update). После выполнения этой команды в бланке запроса появляется строка **Обновление** (Update To).

Для отбора обновляемых записей надо включить в бланк запроса поля, требующие обновления, а также поля, по которым задаются условия отбора записей. Условия отбора задаются так же, как это делается при создании запроса на выборку.

Для обновляемого поля в строку **Обновление** (Update To) надо ввести значение или выражение, определяющее новое значение поля. Выражение можно создать с помощью построителя выражений.

После выполнения команды **Запрос|Запуск** (Query|Run) или нажатия соответствующей кнопки панели инструментов открывается диалоговое окно с сообщением о числе обновляемых записей и вопросом о продолжении операции обновления.

Переключившись в *режим таблицы* после выполнения запроса, можно просмотреть содержимое только обновленных полей. Переключение выполняется командой **Вид|Режим таблицы** (View|Datasheet View) или нажатием кнопки **Вид** (View) панели инструментов. Если переключиться в режим таблицы до выполнения запроса, можно просмотреть старое содержимое обновляемых полей.

Замечание

Если в схеме данных БД установлен параметр *Обеспечение целостности данных* (Enforce Referential Integrity) и *Каскадное обновление связанных полей* (Cascade Update Related Fields), то при обновлении ключевых полей результат зависит от взаимосвязей обновляемой таблицы с другими таблицами.

6.17.2. Запрос на добавление

С помощью *запроса на добавление* (Append Query) производится добавление записей из таблицы результата запроса в таблицу базы данных. Поэтому надо, чтобы в запросе были сформированы записи с полями, соответствующими полям в дополняемой записями таблице базы данных.

Структура записи таблицы запроса может не полностью совпадать со структурой записи таблицы, в которую добавляются записи. В записи запроса может быть меньше полей, если на поля в таблице, куда добавляются записи, не наложено требование по обязательности их заполнения. Допускается несоответствие типов полей, если возможно преобразование типа данных одного поля в тип данных другого поля.

Первоначально *запрос на добавление* (Append Query) создается как *запрос на выборку* (Select Query) на одной или нескольких взаимосвязанных таблицах.

Затем в окне конструктора запросов он превращается в запрос на добавление выбором типа запроса **Добавление** (Append Query) на панели инструментов или команды меню **Запрос|Добавление** (Query | Append Query). При этом открывается диалоговое окно **Добавление** (Append).

В этом окне в поле **Имя таблицы** (Table Name) вводится или выбирается имя таблицы, в которую надо добавить записи.

Если таблица-приемник добавляемых записей находится в открытой базе данных, надо отметить переключатель **В текущей базе данных** (Current Database). Для таблицы, находящейся в другой базе данных, надо отметить переключатель **В другой базе данных** (Another Database) и ввести имя файла БД. При необходимости укажите путь. Можно также указать базу данных Microsoft FoxPro, Paradox, dBASE или ввести строку подключения для базы данных SQL.

После преобразования запроса в запрос на добавление в его бланке появляется строка **Добавление** (Append To).

Для формирования добавляемых записей надо включить в бланк запроса поля, соответствующие определенным полям таблицы, в которую будет производиться добавление. Кроме того, в бланк запроса могут быть включены поля, по которым задаются условия отбора. Условия отбора вносятся в ячейки строки **Условие отбора** (Criteria). Если в таблице, куда добавляются записи, есть ключ, ключевые поля должны быть обязательно включены в бланк запроса.

Для указания в строке **Добавление** (Append To) имен полей таблицы-получателя, в которые будут добавляться значения из соответствующих полей таблицы запроса, надо в каждой ячейке открыть список полей и выбрать нужное имя.

Заметим, если выбранные поля имеют одни и те же имена в обеих таблицах, имена в строку **Добавление** (Append To) вносятся автоматически.

Для предварительного просмотра записей, которые планируется добавить в таблицу, надо нажать кнопку **Вид** (View) на панели инструментов. Возврат в режим конструктора запросов производится по этой же кнопке.

Для добавления записей надо нажать кнопку **Запуск** (Run) на панели инструментов. Открывается диалоговое окно с сообщением о числе обновляемых записей и вопросом о продолжении операции обновления.

Если таблица, в которую добавляются записи, содержит ключевое поле, добавляемые записи должны содержать такое же поле. Те записи, добавление которых приведет к появлению совпадающих или пустых значений в ключевом поле, не будут

добавлены. Записи не добавляются также в случае, если невозможно преобразование типа данных в добавляемых полях или не выполняются условия на значения.

Заметим, если в схеме данных БД установлен параметр **Обеспечение целостности данных** (Enforce Referential Integrity), то добавление подчиненных записей возможно только в том случае, когда в главных таблицах уже имеются связанные записи.

6.17.3. Запрос на удаление

Запрос на удаление (Delete Query) позволяет удалить записи из одной таблицы или из нескольких взаимосвязанных таблиц. В запросе указываются таблицы, из которых должны удаляться записи, и задаются условия отбора удаляемых записей.

Первоначально *запрос на удаление* (Delete Query) создается как *запрос на выборку* (Select Query), в схему данных которого включаются взаимосвязанные таблицы, из которых требуется удалить записи и для полей которых задаются, условия отбора. Затем запрос в режиме конструктора преобразуется в *запрос на удаление* (Delete Query) выбором типа запроса Удаление (Delete Query) на панели инструментов или команды меню **Запрос|Удаление** (Query|Delete Query).

После преобразования запроса в запрос на удаление в его бланке появляется строка Удаление (Delete).

Затем формируется бланк запроса. Для задания таблицы, записи которой требуется удалить, надо с помощью мыши переместить символ звездочки (*) из списка полей соответствующей таблицы в бланк запроса. В строке Удаление (Delete) в столбце этого поля появляется значение **Из** (From). Для того чтобы задать условия отбора удаляемых записей, надо переместить с помощью мыши в бланк запроса поля, для которых устанавливаются условия отбора. В строке **Удаление** (Delete) под именами этих полей появляется значение **Условие** (Where). Строку **Условие отбора** (Criteria) для этих полей надо заполнить необходимыми условиями.

Для *предварительного просмотра* удаляемых записей можно нажать кнопку **Вид** (View) панели инструментов. Для возврата в режим конструктора запроса также используется эта кнопка.

Для удаления записей нажимается кнопка **Запуск** (Run) на панели инструментов.

Результаты работы запроса на удаление зависят от установленных в схеме базы данных отношений между таблицами и параметров целостности.

Если задан параметр **Обеспечение целостности данных** (Enforce Referential Integrity) и установлен параметр **Каскадное удаление связанных записей** (Cascade Delete Related Records), то для удаления записей групп и связанных с ними записей студентов достаточно указать в запросе удаление записей главной таблицы «Кафедра». Если параметр каскадного удаления не задан, то удаление записей таблицы «Кафедра» возможно, только если в подчиненной таблице нет связанных записей.

Напомним, что подчиненные записи самого нижнего уровня могут быть удалены независимо от параметров целостности. Если параметры целостности не установлены вообще, то записи удаляются только в указанных в бланке запроса таблицах и вне зависимости от их логических связей.

Задание 5.

Создать запрос на удаление всех записей из базы данных «Абитуриент», содержащих информацию об абитуриентах, забравших документы. Такая категория имеет одиннадцатую группу (поле Gr имеет значение «11»).

6.18. Мастера создания запросов

Простейшие запросы некоторых видов могут быть созданы с помощью мастеров Access. С помощью мастера можно создать:

- Простой запрос на выборку.
- Перекрестный запрос.
- Запрос для поиска повторяющихся записей (записей с повторяющимися значениями в полях).
- Запрос для поиска записей, не имеющих подчиненных.

Мастер запросов ускоряет процесс создания запроса, автоматически выполняя первоначальные простейшие действия по подготовке запроса. Вызванный мастер запрашивает у пользователя сведения и создает запрос на основе его ответов. При необходимости можно в режиме конструктора отредактировать запрос.

Создание запроса с помощью мастера начинается с выбора в окне базы данных объекта **Запросы** (Queries) и нажатия кнопки **Создать** (New). В окне диалога **Новый запрос** (New Query) надо выбрать один из предлагаемых видов запроса: **Простой** (Simple Query Wizard), **Перекрестный запрос** (Crosstab Query Wizard), **Повторяющиеся записи** (Find Duplicates Query Wizard), **Записи без подчиненных** (Find Unmatched Query Wizard).

Далее следует выполнять инструкции, выводимые в окнах диалога мастера. В последнем окне диалога предлагается выбрать запуск запроса или просмотр структуры запроса в режиме конструктора.

Заметим, что запрос для поиска повторяющихся записей и запрос для поиска записей, не имеющих подчиненных, являются элементарными запросами на выборку.

6.19. Мастера запросов на выборку

6.19.1. Простой запрос

При создании *простого запроса* на выборку мастер предоставляет возможность выбрать поля из взаимосвязанных таблиц и запросов. Причем в подготовленном бланке запроса не формируются условия отбора и вычисляемые поля. Единственное, что может сделать мастер, — это подготовить итоговый запрос, сгруппировав записи и подсчитав для этих групп сумму, среднее значение, минимум, максимум, число записей в группе.

Перейти к созданию простого запроса мастером можно, выбрав опцию **Создание запроса с помощью мастера** (Create query by using wizard) в списке запросов базы данных.

6.19.2. Запрос для поиска повторяющихся записей

Мастер создания *запроса для поиска повторяющихся записей* строит запрос, который определяет, содержит ли таблица повторяющиеся значения в одном или нескольких полях. Мастер позволяет выбрать анализируемую таблицу, задать поля, в которых следует проверить повторяемость значений, отобрать поля, которые надо вывести наряду с повторяющимися. Запрос выводит только те записи, для которых есть хотя бы еще одна запись в таблице с одинаковыми значениями в выбранных полях. Записи выводятся в порядке возрастания значений в полях с повторяющимися значениями.

6.19.3. Запрос для поиска записей, не имеющих подчиненных

Мастер создания *запроса для поиска записей, не имеющих подчиненных*, позволяет найти в таблице записи, у которых нет связанных записей в подчиненной таблице.

6.19.4. Мастер перекрестных запросов

В *перекрестном запросе* мастер формирует таблицу, в которой левый столбец образует заголовки строк из значений одного поля, верхняя строка образует заголовки столбцов из значений другого поля, а на пересечении строк и столбцов размещаются итоговые значения, вычисленные по значениям третьего поля. Для получения итоговых значений записи группируются по полям, используемым в качестве заголовков строк и столбцов, а для значений третьего поля в полученных группах записей применяется одна из выбранных статистических функций. Можно предусмотреть выполнение статистической функции и получение общего итогового значения для каждой строки в целом.

Замечание

Нет необходимости предварительно выполнять запрос на основе которого выполняется другой запрос. Выполнение вложенного запроса инициируется системой при выполнении запроса, построенного на нем.

Зачетное задание.

Отобразить в запросе (база данных «Литература») книги технического направления.

7. Лабораторная работа №7. Создание отчетов в среде Access

7.1. Цель работы

Выработка навыков разработки отчетов в пользовательском режиме СУБД Ms Access.

7.2. Программное обеспечение

1. Для проведения лабораторной работы требуется Ms Access 97/2000. Рекомендуются Ms Access 2000.
2. Необходима установка драйвера любого принтера.

7.3. Создание отчетов

Средства разработки отчетов в Access предназначены для конструирования макета отчета, по которому может быть осуществлен вывод данных в виде выходного печатного документа. Эти средства позволяют создавать отчет сложной структуры, обеспечивающий вывод взаимосвязанных данных из многих таблиц, их группировку, вычисление данных. При этом могут быть выполнены самые высокие требования к оформлению документа.

Перед началом конструирования отчета пользователь должен спроектировать его макет. При этом определяется состав и содержание разделов отчета, размещение в нем значений, выводимых из полей таблиц базы данных, и вычисляемых реквизитов, определяются поля, по которым нужно группировать данные. Для каждого уровня группировки определяются заголовки и примечания, вычисляемые итоговые значения. Кроме того, оформляются заголовки и подписи реквизитов отчета. Определяется также порядок вывода данных в отчете.

Отчет может создаваться с помощью мастера или в режиме конструктора отчетов. Во многих случаях удобно использовать мастера отчетов. Созданный мастером отчет можно доработать в режиме конструктора.

При необходимости вывода в отчете результатов решения задачи в качестве основы для отчета может быть использован многотабличный запрос. На запрос могут быть возложены наиболее сложные виды выборки и предварительной обработки данных. Разнообразные возможности конструктора отчетов позволяют легко структурировать и оформить полученные в запросе данные.

7.4. Конструирование отчетов

Режим конструктора отчетов во многом похож на режим конструктора форм. Панель инструментов, которая активно используется при конструировании отчета, аналогична панели, используемой при разработке форм.

7.4.1. Окно конструктора отчета

7.4.1.1. Разделы отчета

Создание и изменение макета отчета осуществляется в окне конструктора отчетов. При создании отчета в режиме конструктора в окне первоначально отображаются пустые разделы отчета.

Наличие этих разделов, а также их удаление или включение определяется командами меню **Вид|Колонтитулы** (View|Page Header/Footer) и **Вид|Заголовок/примечание отчета** (View|Report Header/Footer). Для этих же целей можно использовать соответствующие кнопки панели инструментов конструктора отчетов.

При создании отчета его разделы нужно заполнить элементами в соответствии с разработанным пользователем макетом отчета. В заголовок помещается текст из шапки макета отчета. В верхний и нижний колонтитул обычно помещают заголовки, номера страниц и даты. При определении содержания этих разделов следует исходить из требований к оформлению отдельных страниц отчета. В области данных размещаются поля таблиц базы данных или запросов.

При необходимости группировки записей по полю в окно конструктора отчетов могут быть добавлены разделы **Заголовок группы** (Report Header) и **Примечание группы** (Footer). В заголовке группы, как правило, размещаются поля, по которым производится группировка. В примечании группы могут быть размещены выражения для подведения итогов по группе. Допускается до 10 уровней группировки выводимых записей.

7.4.1.2. Элементы разделов отчета

В процессе конструирования с помощью команд меню или кнопок панели инструментов конструктора отчетов и панели элементов разделы отчета заполняются элементами в соответствии с планируемым макетом отчета.

Поля с неповторяющимися значениями размещают в **Области данных** (Detail), которой можно придать вид табличной части отчета. Поля с повторяющимися значениями, по которым производится группировка записей, целесообразно размещать в заголовке группы.

Элементами разделов отчета, кроме полей таблиц или запросов, на которых строится отчет, являются также тексты подписей, кнопки управления, внедряемые объекты, линии, прямоугольники и т. п. Для каждого из элементов имеются соответствующие кнопки на панели элементов. Назначение кнопок панели элементов было показано на рис. 4.8.

Для каждого элемента, а также раздела и отчета в целом могут быть уточнены свойства. Технология размещения элементов и определения их свойств практически такая же, как и в конструкторе форм.

7.4.1.3. Панель инструментов конструктора отчетов

В процессе конструирования отчета используются команды меню, панель инструментов **Конструктор отчетов**, а также панель элементов.

Панель инструментов конструктора отчетов появляется, когда осуществляется переход в режим конструирования отчета при первоначальном создании или при внесении изменений в макет отчета.

7.4.2. Создание отчета

7.4.2.1. Создание отчета в режиме конструктора

В окне базы данных выберем объект **Отчеты (Reports)** и нажмем кнопку; **Создать (New)**. Далее в диалоговом окне **Новый отчет (New Report)** выберем таблицу **СТУДЕНТ**, которая будет источником данных для отчета. Для создания отчета в режиме конструктора выберем **Конструктор (Design)**.

Если отсутствует раздел **Заголовок отчета (Report Header)**, включим его с помощью кнопки **Заголовок/примечание отчета (Report Header/Footer)** панели конструктора отчетов.

7.4.2.2. Группировка и сортировка данных отчета

Для выполнения требований к группировке и сортировке данных, отображаемых в отчете, необходимо нажать кнопку **Сортировка и группировка (Sorting and Grouping)** на панели инструментов конструктора и задать необходимые параметры в открывшемся диалоговом окне.

Группировка по полю. Для группировки записей по какому-либо определенному признаку необходимо выбрать в окне **Сортировка и группировка (Sorting and Grouping)** из списка поле, по значениям которого нужно проводить группировку. Для этого в области **Свойства группы** в строках **Заголовок группы** и **Примечание группы** надо выбрать **Да (Yes)**. Сортировка для поля устанавливается автоматически.

Сортировка по полю. Для вывода отсортированного списка данных из базы нужно выбрать **Свойства группы** этого поля в строках **Заголовок группы** и **Примечание группы**. По умолчанию установлены значения **Нет (No)**.

7.4.2.3. Размещение полей из таблиц

Размещение поля группировки. Значение поля группировки должно быть представлено один раз в заголовке группы. Для этого нужно разместить поле в разделе **Заголовок группы**. Для этого нужно Нажать кнопку панели инструментов конструктора отчетов **Список полей** и перетащить поле в раздел заголовка. Для установки размеров рамки по размеру текста подписи нужно выполнить команду **Формат|Размер|по размеру данных** или соответствующую кнопку панели инструментов.

Формирование табличной части отчета. Для этого необходимо последовательно разместить поля в области данных, которая определяет содержимое строк табличной части. Поле размещается вместе с подписью, которую система берет из свойств полей таблицы. Подписи полей надо перенести в область заголовка путем вырезания и вставки. Если они не совпадают с названиями столбцов в проекте макета, их надо откорректировать. Подписи также можно создать заново, воспользовавшись кнопкой панели элементов **Надпись**.

7.4.2.4. Включение вычисляемого поля в отчет

Для включения расчетного реквизита нужно нажать кнопку **Поле** на панели элементов и разместить элемент **Свободный** в раздел **Примечание группы**. Затем, определить в свойствах этого элемента выражение для расчета среднего значения. Для этого запишем на вкладке **Данные** в строку **Данные** необходимую функцию. После возможно отредактировать подпись поля. Для этого нужно выделить подпись и вызвать ее свойства. В свойствах на вкладке **Макет** в строке **Подпись** записать требуемый текст. Такие действия, как изменение подписи или ввод выражения в поле можно выполнить, и не обращаясь к свойствам элементов.

7.4.2.5. Добавление текущей даты и номера страницы

Для добавления в отчет *текущей даты* имеется встроенная функция Now(). Для этого в заголовке отчета создается свободный элемент, нажатием кнопки **Поле**, и зададим в окне его свойств на вкладке **Данные** в строке **Данные** выражение =Now().

Для добавления *номера страницы* в нижний колонтитул нужно создать свободный элемент и заполнить в его свойствах на вкладке **Данные** строку **Данные** выражением =[Page].

Существуют и другие способы формирования поля даты и номера страницы.

Поле текущей даты и времени можно добавить в отчет, выполнив в режиме конструктора команду **Вставка|Дата и время**. Установка в диалоговом окне **Дата и время** флажков **Формат даты** и/или **Формат времени** позволяет вставить текущую дату и/или текущее время и выбрать нужный формат.

В отчет будет добавлено поле, в свойствах которого на вкладке **Данные** в строке **Данные** будет записано соответствующее выражение. Если в отчете имеется раздел заголовка, поле добавляется в этот раздел. В противном случае поле вносится в раздел данных. В качестве выражения записывается функция Format, которая формирует значение на основе заданных ей аргументов — функции Date(), возвращающей текущую системную дату, и формата, в котором должна выводиться дата.

Поле нумерации страниц можно добавить в отчет, выполнив в режиме конструктора команду **Вставка|Номера страниц**. В окне диалога **Номера страниц** выбираются параметры, определяющие формат, расположение и выравнивание номеров страниц. Для печати номера страницы на первой странице устанавливается флажок **Отображать номер на первой странице**.

Замечание

Выражение, определяющее вывод номеров страниц, записывается в свойствах поля на вкладке Данные (Data) в строке Данные (Control Source). Выражение может иметь вид:

= "Страница" & [Page]

или:

= "Страница" & [Page] & " из " & [Pages],

что соответствует выбору Страница N (Page N) или Страница N из M (Page N of M).

7.4.2.6. Завершение оформления отчета

Для окончательного оформления введем в раздел **Заголовок отчета** (Report Header) название отчета — "СПИСКИ СТУДЕНТОВ". Для этого воспользуемся кнопкой панели элементов **Надпись** (Label). Установим нужный шрифт надписи, с помощью кнопок панели форматирования.

Для вывода названия отчета на последующих страницах введем его также в **Верхний колонтитул** (Page Header). Для этого можно скопировать название из раздела **Заголовок отчета** (Report Header), вставить в раздел **Верхний колонтитул** (Page Header) и выбрать нужный шрифт. Далее надо указать в свойствах отчета на вкладке **Макет** (Format) в строке **Верхний колонтитул** (Page Header): "Без заголовка" (Not with Rpt Hdr). Свойства отчета могут быть вызваны при установке курсора на пересечении линеек.

Создадим линии в соответствии с макетом, воспользовавшись кнопкой панели элементов **Линия** (Line).

7.5. Просмотр и печать отчета

7.5.1. Просмотр отчета

Переход из режима конструктора отчетов в режим предварительного просмотра осуществим, нажав кнопку **Предварительный просмотр**. Для просмотра ранее созданного отчета нужно выбрать его в окне базы данных на вкладке **Отчеты** и нажать кнопку **Просмотр**. Отчет при просмотре отобразится на экране таким, каким он будет напечатан.

В режиме предварительного просмотра имеется своя панель инструментов.

Для просмотра нужных страниц отчета можно использовать стандартное *поле номера страницы* в нижнем левом углу окна отчета.

7.5.2. Печать отчета

Кнопка Печать (Print) панели инструментов режима предварительного просмотра позволяет вывести отчет на печать.

С помощью команды **Файл|Параметры страницы** можно выбрать принтер, задать формат бумаги, размер полей, расстояние между строками, ориентацию (книжная, альбомная) и т. д. Команда **Файл|Печать** позволяет выбрать для печати отдельные страницы отчета или выделенные записи, распечатать заданное число копий, вывести отчет в файл, который должен распечатываться в другое время.

Задание

Спроектировать отчеты для всех ранее созданных баз данных: «Работник», «Подразделение», «Кафедра», «Литература», «Студент». Отчет должен учитывать специфику отображаемой информации и быть представлен наиболее компактным способом.