

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**  
**Федеральное государственное бюджетное образовательное учреждение высшего образования**  
**«Кузбасский государственный технический университет имени Т. Ф. Горбачева»**

**Кафедра прикладных информационных технологий**

**Составитель**  
**Е. В. Прокопенко**

## **КОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ И БАЗЫ ДАННЫХ**

### **Методические материалы**

Рекомендованы учебно-методической комиссией направления  
подготовки 27.03.05 Инноватика в качестве электронного издания  
для использования в образовательном процессе

**Кемерово 2018**

**Рецензенты**

Королева Т. Г., председатель учебно-методической комиссией направления подготовки 27.03.05 Инноватика

Сыркин И. С., кандидат технических наук, доцент кафедры информационных и автоматизированных производственных систем

**Прокопенко Евгения Викторовна**

**Компьютерные технологии и базы данных:** методические материалы [Электронный ресурс]: для обучающихся направления подготовки 27.03.05 Инноватика очной формы обучения / сост. Е. В. Прокопенко; КузГТУ. – Кемерово, 2018.

© КузГТУ, 2018

© Е. В. Прокопенко,  
составление, 2018

## **Компьютерные технологии. Компьютерные технологии баз данных. Базы данны.**

В современных условиях специалистам всех сфер деятельности очень часто приходится работать с информацией, полученной из разных источников, каждый из которых связан с определенным видом деятельности. Более того, сегодня, в информационную эпоху, в подавляющем большинстве случаев при решении хозяйственных, экономических и финансовых задач приходится иметь дело с обширными массивами данных. Они разнородны, специфически структурированы и взаимосвязаны друг с другом. Такие сложные наборы данных принято называть базами данных (далее – БД).

Вся работа базируется на управлении информацией. Данные решают все, и очень важно эффективно их обрабатывать. Теория управления БД как самостоятельная дисциплина начала развиваться приблизительно с начала 50-х годов XX в. За это время она приобрела черты классической и заняла достойное место в современной науке. Однако нас больше интересует не теоретический, а сугубо практический аспект информационной обработки БД.

Программное обеспечение, осуществляющее операции над БД, получило название СУБД, что означает «система управления базами данных».

Современные СУБД в основном являются приложениями Windows, так как данная среда позволяет более полно использовать возможности персональной ЭВМ, нежели среда DOS. Снижение стоимости высокопроизводительных ПК обусловил не только широкий переход к среде Windows, где разработчик программного обеспечения может в меньшей степени заботиться о распределении ресурсов, но также сделал программное обеспечение ПК в целом и СУБД в частности менее критичными к аппаратным ресурсам ЭВМ.

Среди наиболее ярких представителей систем управления базами данных можно отметить: Lotus Approach, Microsoft Access, Borland dBase, Borland Paradox, Microsoft Visual FoxPro, Microsoft Visual Basic, а также СУБД Microsoft SQL Server и Oracle, используемые в приложениях, построенных по технологии "клиент-сервер". Фактически, у любой современной СУБД существует аналог, выпускаемый другой компанией, имеющий аналогичную область применения и возможности, любое приложение способно работать со многими форматами представления данных, осуществлять экспорт и импорт данных благодаря наличию большого числа конвертеров. Общепринятыми, также, являются технологии, позволяющие использовать возможности других приложений, например, текстовых процессоров, пакетов построения графиков и т.п., и встроенные версии языков высокого уровня (чаще – диалекты SQL и/или VBA) и средства визуального программирования интерфейсов разрабатываемых приложений. Поэто-

му уже не имеет существенного значения, на каком языке и на основе какого пакета написано конкретное приложение, и какой формат данных в нем используется. Более того, стандартом "де-факто" стала "быстрая разработка приложений" или RAD (от английского Rapid Application Development), основанная на широко декларируемом в литературе "открытом подходе", то есть необходимость и возможность использования различных прикладных программ и технологий для разработки более гибких и мощных систем обработки данных. Поэтому в одном ряду с "классическими" СУБД все чаще упоминаются языки программирования Visual Basic 4.0 и Visual C++, которые позволяют быстро создавать необходимые компоненты приложений, критичные по скорости работы, которые трудно, а иногда невозможно разработать средствами "классических" СУБД. Современный подход к управлению базами данных подразумевает также широкое использование технологии "клиент-сервер".

Таким образом, на сегодняшний день разработчик не связан рамками какого-либо конкретного пакета, а в зависимости от поставленной задачи может использовать самые разные приложения. Поэтому, более важным представляется общее направление развития СУБД и других средств разработки приложений в настоящее время.

### **1. Понятие и функции системы управления базами данных**

Система управления базами данных (СУБД) – специализированная программа (чаще комплекс программ), предназначенная для организации и ведения базы данных<sup>1</sup>. Для создания и управления информационной системой СУБД необходима в той же степени, как для разработки программы на алгоритмическом языке необходим транслятор.

Основные функции СУБД:

1. Определение структуры создаваемой базы данных, ее инициализация и проведение начальной загрузки.

Как правило, создание структуры базы данных происходит в режиме диалога. СУБД последовательно запрашивает у пользователя необходимые данные. В большинстве современных СУБД база данных представляется в виде совокупности таблиц. Рассматриваемая функция позволяет описать и создать в памяти структуру таблицы, провести начальную загрузку данных в таблицы.

2. Предоставление пользователям возможности манипулирования данными (выборка необходимых данных, выполнение вычислений, разработка интерфейса ввода/вывода, визуализация).

3. Обеспечение независимости прикладных программ и данных (логической и физической независимости)<sup>2</sup>.

Важнейшим свойством СУБД является возможность поддерживать два независимых взгляда на базу данных – "взгляд пользователя", воплощаемый в логическом представлении данных, и его отражения в приклад-

ных программах; и "взгляд системы" – физическое представление данных в памяти ЭВМ. Обеспечение логической независимости данных предоставляет возможность изменения (в определенных пределах) логического представления базы данных без необходимости изменения физических структур хранения данных. Таким образом, изменение логического представления данных в прикладных программах не приводит к изменению структур хранения данных. Обеспечение физической независимости данных предоставляет возможность изменять (в определенных пределах) способы организации базы данных в памяти ЭВМ не вызывая необходимости изменения "логического" представления данных. Таким образом, изменение способов организации базы данных не приводит к изменению прикладных программ.

#### 4. Защита логической целостности базы данных.

Основной целью реализации этой функции является повышение достоверности данных в базе данных. Достоверность данных может быть нарушена при их вводе в БД или при неправомерных действиях процедур обработки данных, получающих и заносящих в БД неправильные данные. Для повышения достоверности данных в системе объявляются так называемые ограничения целостности, которые в определенных случаях "отлавливают" неверные данные. Так, во всех современных СУБД проверяется соответствие вводимых данных их типу, описанному при создании структуры. Система не позволит ввести символ в поле числового типа, не позволит ввести недопустимую дату и т.п. В развитых системах ограничения целостности описывает программист, исходя из содержательного смысла задачи, и их проверка осуществляется при каждом обновлении данных.

#### 5. Защита физической целостности.

При работе ЭВМ возможны сбои в работе (например, из-за отключения электропитания), повреждение машинных носителей данных. При этом могут быть нарушены связи между данными, что приводит к невозможности дальнейшей работы. Развитые СУБД имеют средства восстановления базы данных. Важнейшим используемым понятием является понятие "транзакции". Транзакция – это единица действий, производимых с базой данных. В состав транзакции может входить несколько операторов изменения базы данных, но либо выполняются все эти операторы, либо не выполняется ни один. СУБД, кроме ведения собственно базы данных, ведет также журнал транзакций.

Предположим, что база данных была испорчена в результате аппаратного сбоя компьютера, на котором был установлен сервер СУБД. В этом случае нужно использовать последнюю сделанную резервную копию базы данных и журнал транзакций. Причем применить к базе данных нужно только те транзакции, которые были зафиксированы после создания резервной копии. Большинство современных СУБД позволяют администратору воссоздать базу данных исходя из резервной копии и журнала тран-

закций. В таких системах в определенный момент БД копируется на резервные носители. Все обращения к БД записываются программно в журнал изменений. Если база данных разрушена, запускается процедура восстановления, в процессе которой в резервную копию из журнала изменений вносятся все произведенные изменения<sup>3</sup>.

6. Управление полномочиями пользователей на доступ к базе данных.

Разные пользователи могут иметь разные полномочия по работе с данными (некоторые данные должны быть недоступны; определенным пользователям не разрешается обновлять данные и т.п.). В СУБД предусматриваются механизмы разграничения полномочий доступа, основанные либо на принципах паролей, либо на описании полномочий.

7. Синхронизация работы нескольких пользователей.

Достаточно часто может иметь место ситуация, когда несколько пользователей одновременно выполняют операцию обновления одних и тех же данных. Такие коллизии могут привести к нарушению логической целостности данных, поэтому система должна предусматривать меры, не допускающие обновление данных другим пользователям, пока работающий с этими данными пользователь полностью не закончит с ними работать. Основным используемым здесь понятием является понятие "блокировка". Блокировки необходимы для того, чтобы запретить различным пользователям возможность одновременно работать с базой данных, поскольку это может привести к ошибкам.

Для реализации этого запрета СУБД устанавливает блокировку на объекты, которые использует транзакция. Существуют разные типы блокировок – табличные, страничные, строчные и другие, которые отличаются друг от друга количеством заблокированных записей. Чаще других используется строчная блокировка – при обращении транзакции к одной строке блокируется только эта строка, остальные строки остаются доступными для изменения.

8. Управление ресурсами среды хранения.

БД располагается во внешней памяти ЭВМ. При работе в БД заносятся новые данные (занимается память) и удаляются данные (освобождается память). СУБД выделяет ресурсы памяти для новых данных, перераспределяет освободившуюся память, организует ведение очереди запросов к внешней памяти и т.п.

9. Поддержка деятельности системного персонала.

При эксплуатации базы данных может возникать необходимость изменения параметров СУБД, выбора новых методов доступа, изменения (в определенных пределах) структуры хранимых данных, а также выполнения ряда других общесистемных действий. СУБД предоставляет возможность выполнения этих и других действий для поддержки деятельности БД

обслуживающему БД системному персоналу, называемому администратором БД<sup>4</sup>.

Обычно современная СУБД содержит следующие компоненты:

- ядро, которое отвечает за управление данными во внешней и оперативной памяти и журнализацию;
- процессор языка базы данных, обеспечивающий оптимизацию запросов на извлечение и изменение данных, и создание, как правило, машинно-независимого исполняемого внутреннего кода;
- подсистему поддержки времени исполнения, которая интерпретирует программы манипуляции данными, создающие пользовательский интерфейс с СУБД;
- а также сервисные программы (внешние утилиты), обеспечивающие ряд дополнительных возможностей по обслуживанию информационной системы.

## 2. Средства организации баз данных и работа с ними

Управление базой данных позволяет организовать работу с территориально удаленными подразделениями в рамках единого информационного пространства с целью получения оперативной информации о состоянии дел в территориально-удаленных подразделениях организации, а сотрудникам своевременно получать необходимую информацию из центральной ЭВМ.

Требования, предъявляемые к базам данных:

- контроль целостности передаваемых данных;
- использование различных каналов связи;
- обеспечение эффективной загрузки системы в целом;
- полнота представления данных, т.е. данные в базе должны адекватно представлять всю информацию об объекте и их должно быть достаточно для систем обработки данных;
- обеспечение сохранности информации при их обработке;
- обеспечение разграничения доступа к данным;
- целостность баз данных, т.е. данные должны сохраняться при обработке их системами обработки данных;
- гибкость структуры данных, т.е. БД должна позволять изменить структуры данных, не нарушая своей целостности и полноты при изменении внешних условий;
- реализуемость. Должно быть объективное представление разнообразных объектов, их свойств и отношений;
- доступность, т.е. БД должна обеспечить разграничение доступа пользователей к данным<sup>5</sup>.

СУБД обрабатывает информацию, которая находится в БД. К ней предъявляются следующие требования:

Независимость данных, универсальность, защита данных;

Должность обеспечить поддержку централизованной и распределенной БД;

Предохранять БД от рассогласований в режиме коллективного доступа.

Основными средствами работы в СУБД являются:

- Средства задания (описания) структуры БД;
- Средства конструирования экранных форм, предназначенных для ввода данных, просмотра и их обработки в диалоговом режиме;
- Средства создания запросов для выборки данных при заданных условиях;
- Средства создания отчетов из БД для вывода на печать результатов обработки в удобном для пользователя виде;
- Средства создания отчетов из БД для вывода на печать результатов обработки в удобном для пользователя виде:
- Языковые средства – макросы, встроенный алгоритмический язык (Dbase, Visual Basic и др.), язык запросов (SQL) и т.п.;
- Средства создания приложений пользователя (генераторы приложений, средства создания меню и панели управления приложениями), позволяющие определить различные операции работы с базой данных в единый технологический процесс. СУБД может иметь включающий или базовый язык программирования. В СУБД с базовым языком применяется собственный алгоритмический язык, позволяющий кроме операций манипулирования данными выполнять различные вычисления и обработку данных. Стандартным реляционным языком запросов является язык структурированных запросов SQL<sup>6</sup>.

### 3. Системы управления базами данных

Microsoft Access – это только одна из многочисленных «персональных» СУБД, которые успешно используются в различных областях. Кроме персональных, существуют также профессиональные (промышленные) СУБД. Именно они первоначально получили наибольшее распространение до появления персональных компьютеров, да и сейчас используются в самых важных областях. На их основе создаются комплексы управления и обработки информации крупных предприятий, банков и даже целых отраслей. Профессиональные СУБД поддерживают совместную работу с базой большого количества пользователей; обеспечивают масштабируемость, т.е. возможность роста системы пропорционально увеличению запросов к ней; являются максимально устойчивыми к сбоям различного рода и могут работать круглосуточно в течение многих лет. Пожалуй, самой известной профессиональной СУБД сейчас является Oracle, которая вот уже долгие годы обрабатывает информацию для ФБР и ЦРУ (по их заказу и разрабатывалась эта система). Кстати, основатель фирмы Oracle Ларри Эллисон входит в число самых богатых людей мира, лишь немного уступая по размеру своего состояния Биллу Гейтсу.



Персональные СУБД сильно отличаются от профессиональных. Данные программы предназначены для обслуживания небольшой группы пользователей или вообще одного человека. Их фактически можно считать офисным программным обеспечением. Неудивительно, что СУБД Access входит в комплект MS Office, точнее, в его профессиональную версию MS Office Professional.

**База данных, БД** – совокупность взаимосвязанных, хранящихся вместе сведений о различных сущностях одной предметной области (реальных объектах, процессах, явлениях или событиях), обеспечивающая наличие такой минимальной избыточности, которая допускает их использование оптимальным образом для одного или нескольких приложений или пользователей; одним из основных свойств баз данных можно считать независимость данных от использующих их прикладных программ. Под независимостью данных подразумевается то, что изменения в данных не приводит к изменению программ. Разработка программ длительный, трудоемкий и дорогостоящий процесс, поэтому при возникновении потребности модифицировать структуру данных, необходимости сохранять уже созданные прикладные программы.

Для обеспечения действительной независимости данных (хотя полностью независимые данные бывают очень редко) предлагается создавать структуры двух видов: логические и физические. Логические структуры описывают, как данные представляются прикладному программисту или пользователю данных. Физические структуры определяют способ физической записи данных на внешней памяти. Логические структуры могут не совпадать с физическими. Программное обеспечение преобразует логические структуры в физические.

**Системы управления базами данных, СУБД** – то программные средства, предназначенные для ввода, наполнения, удаления, фильтрации и поиска данных.

**Модель данных** – является фундаментом технологий баз данных; на ней базируется конкретная СУБД. Модель описывает набор понятий и признаков, которыми должна обладать конкретная СУБД и управляемые ими базы данных, если они основываются на этой модели. Наличие такой модели позволяет сравнивать конкретные реализации СУБД и оценивать их соответствие модели.

История создания и развития СУБД насчитывает около сорока лет. За этот период были разработаны многочисленные модели данных, прежде всего это сетевые, иерархические, реляционные и объектные модели данных. Сетевые и иерархические модели в настоящее время считаются устаревшими, но существует множество баз данных созданных на их основе и требующих поддержания их работы.

Одним из крупнейших достижений в этой области является создание **реляционной модели данных** и базирующейся на ней теории реляционных баз данных, которая позволила получить важные результаты для развития теории баз данных. Как отмечают многие исследователи, своим успехом реляционная модель данных во многом обязана, в первую очередь тому, что опиралась на строгий математический аппарат теории множеств, отношений и логики первого порядка. Разработчики любой конкретной реляционной системы считали своим долгом показать соответствие своей конкретной модели данных общей реляционной модели, которая выступала в качестве меры "реляционности" системы. Существует широкий спектр реляционных СУБД для приложений различного масштаба. Разработан международный стандарт языка запросов SQL, ставший универсальным интерфейсом коммерческих реляционных СУБД. По оценкам специалистов, примерно 99% мирового рынка баз данных занимают в настоящий момент реляционные СУБД. Несмотря на то, что подавляющее большинство приложений базируется на реляционной технологии, их роль начинает ослабевать.

Вместе с тем в последние годы четко обозначилась тенденция развития СУБД в объектном направлении. Объектная (объектно-ориентированная) модель не противоречит реляционной модели данных, а дополняет и развивает последнюю (точнее сказать – реляционная модель является частным случаем объектной формы представления данных). Однако, трудности развитого математического аппарата, на который могла бы опираться общая объектная модель данных, не существует, как нет и признанной базовой объектной модели. С другой стороны, некоторые авторы утверждают, что общая объектная модель данных в классическом смысле и не может быть определена по причине непригодности классического понятия модели данных к парадигме объектной ориентированности.

**Парадигма** – это пространство идей и законы движения в этом пространстве. В рамках парадигмы определены аксиомы, на которых выстраивается своя логика. Решения, вырабатываемые в рамках парадигмы, непротиворечивы и логичны.

Преимуществами объектных СУБД можно считать:

объектные СУБД – открытые системы. Несложно добавить новый тип данных;

Большинство производителей ООБД предоставляют визуальные средства создания прикладных программ ОСУБД. Если раньше созданием прикладных программ для ОСУБД занимались специалисты в C++, Smaltalk, то теперь использовать ООБД стало намного проще

Объектные СУБД быстрее, чем реляционные, если в программе многократно осуществляется переход от объекта к объекту по ссылке. Поскольку ссылка на объект есть идентификатор, однозначно определяющий

его расположение в базе, то переход по такой ссылке происходит быстрее, чем ссылка между кортежами отношений по первичному ключу. ОСУБД устраняют необходимость в языке запросов

Традиционные области применения ОСУБД – САПР, моделирование, мультимедиа. ОСУБД широко используются в телекоммуникациях, различных аспектах автоматизации предприятия, издательском деле, геоинформационных проектах.

**Интеграция неоднородных информационных ресурсов.** Информационная неоднородность ресурсов заключается в разнообразии понятий, словарей; отображаемых реальных объектов; правил, определяющих адекватность моделируемых объектов реальности; видов данных, способов их сбора и обработки; интерфейсов пользователей и т.д.

Реализационная неоднородность источников проявляется в использовании разнообразных компьютерных платформ, средств управления базами данных, моделей данных и знаний, средств программирования, операционных систем, и т.п. Системы обеспечивающие ..., называются **интероперабельными системами**.

Традиционные системы баз данных, используемые в информационных системах для сопровождения бизнес – процессов поддерживают большие объемы информации с помощью технологий оперативная обработка транзакций – OLTP. В OLTP-технологии обрабатываются детализированные данные, главные свойства данных здесь, их полнота и актуальность.

Для поддержки принятия решений нужны другие технологии. Необходимо объединять данные из различных источников (как из корпоративной информационной системы, так и из внешней среды), накапливать данные, делая их срезы во времени. Анализ таких данных позволяет оценивать состояние и динамику развития организации, делать обоснованные прогнозы и принимать обоснованные решения. Программные продукты, необходимые для обеспечения управленческих решений, должны обеспечивать хранение больших объемов данных, эффективный доступ к ним, а так же располагать развитыми средствами анализа данных и представления результатов в удобной для специалистов и руководства форме.

**OLAP-технология, On-Line Analytical Processing, оперативная аналитическая обработка** – информационная технология, которая предоставляет руководителям различного уровня возможность получения необходимой информации для принятия управленческих, финансовых и кадровых решений. OLAP-технологии базируются на технологиях хранилищ данных (Data warehouses).

**Хранилище данных, Data warehouses** – обеспечивает накопление с течением времени данные для содействия в принятии решений. Хранилище это данных репозиторий (склад) информации содержащий объединенные, проверенные данные, отражающие работа организации за длительный

период. Объемы данных в хранилищах данных в несколько раз превосходят объемы данных в OLTP-системах.

Хранилища данных отличаются от баз данных или систем оперативной обработки транзакций (OLTP-систем) своим назначением и устройством:

- хранилище содержит данные, позволяющие проводить анализ деловых операций;

- хранилища обычно представляют собой системы, доступные только для чтения;

- в хранилищах же накапливаются данные, не меняющиеся со временем и избавленные от ошибок.

Из-за большого объема данных в хранилищах одной из основных проблем создания хранилищ является обеспечение высокой производительности обработки запросов. Запросы в хранилище отличаются высоким уровнем сложности.

Создание хранилищ данных – трудоемкий и длительный процесс. Наряду с хранилищами данных существуют и часто используются компаниями витрины данных (Data Mart), называемые также киосками данных. Такие системы создаются для отдельных подразделений компаний или для обеспечения отдельных видов деятельности. Объемы данных и требования к вычислительным ресурсам в витринах данных существенно меньше по сравнению с хранилищами. Витрины данных могут строиться как независимо, так и на основе хранилищ данных компании. Хранилища данных имеют двухуровневую или трехуровневую архитектуру. В двухуровневых хранилищах на верхнем уровне поддерживается объединенная информация. На нижнем уровне – различные источники баз данных. В трехуровневой архитектуре предусматривается поддержка витрин данных для отдельных подразделений компании над ее единым хранилищем.

В современных условиях в большинстве случаев для доступа к базам данных используются компьютерные сети. Наиболее перспективной архитектурой при организации работы в сетях является "клиент-сервер". В клиент-серверных СУБД их программы функционально разделены на две части, называемые *сервером* и *клиентом*.

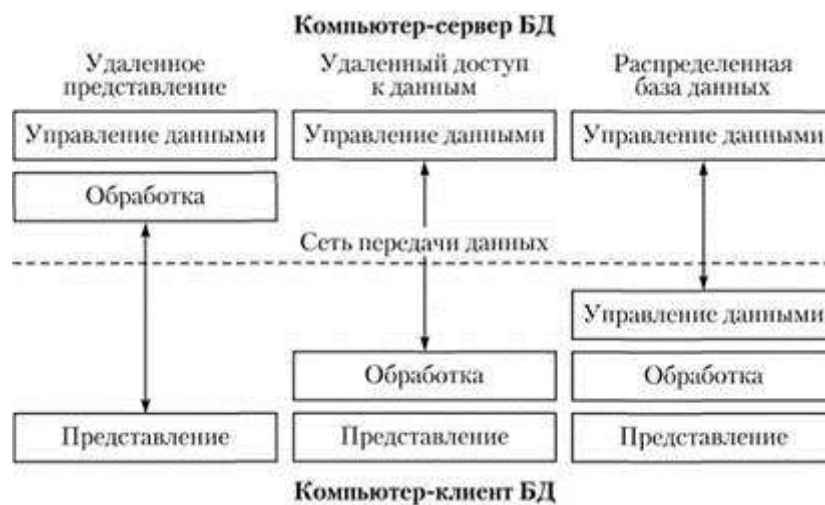
**Серверная часть** СУБД – это основная программа, выполняющая функции управления и защиты данных в БД, находящейся на компьютерном сервере. В качестве сервера может использоваться ядро профессиональной реляционной СУБД (Informix, Sybase) или некоторый SQL-сервер (Novell Netware SQL, Microsoft SQL Server).

**Клиентская часть** СУБД исполняется на компьютере пользователя и обеспечивает интерфейс пользователя с БД: преобразует запросы пользователя в команды запросов к серверной части, при получении результатов выполняет преобразование и отображение информации. Клиентской

программой может быть любая программа, имеющая интерфейс с серверной программой или СУБД (Access, FoxPro, Paradox).

Между клиентской и серверной частями системы возможны различные варианты распределения функций. При этом основные функции СУБД (управление данными; обработка с помощью прикладных программ; пользовательские представления) могут выполняться на одном или нескольких узлах сети. На рис. 11.2 представлены три основные схемы разделения функций и данных между двумя узлами сети.

**Модель удаленного представления** иначе называют моделью сервера БД (DBS – DataBase Server). В этой модели функции компьютера-клиента заключаются только в представлении полученной с сервера информации, а все управление и прикладные функции сосредоточены на компьютере-сервере. Приложения реализуются в виде так называемых хранимых процедур. Все основные действия по коллективному доступу к БД и обработке выполняются на мощном сервере, а клиентам пересылаются только необходимые данные. Достоинством такого подхода является централизованное администрирование БД и приложений на всех этапах разработки, сопровождения и модификации системы, а также низкая нагрузка коммуникационных каналов.



**Схемы моделей архитектуры баз данных клиент-сервер**

Модель DataBase Server используется в таких СУБД, как Ingress, Sybase и Oracle.

**Удаленный доступ к данным** (RDA – Remote Data Access) предполагает выполнение обработки данных на компьютерах пользователей (клиентах), а обращение к серверу производится с помощью SQL-запросов или вызовом функций специальной библиотеки API (Application Program Interface). Основным достоинством такой схемы можно считать тот факт, что большинство существующих СУБД поддерживают SQL-интерфейсы и существует большое количество систем разработки приложений клиент-

ской части. Недостатками являются большой поток данных, передаваемых по сети, и отсутствие возможности централизованного администрирования.

В модели *распределенной БД* данные хранятся как на компьютере-сервере, так и на компьютере-клиенте. При этом в локальной и удаленной базах данных могут храниться различные части единой БД или же локальная и удаленная БД являются синхронизируемыми друг с другом копиями. Системы с распределенной БД обладают большой гибкостью и живучестью, однако сложны в разработке и требуют больших затрат коммуникационных и вычислительных ресурсов.

Классификация СУБД:

- По архитектуре организации хранения данных:
  - локальные (все части базы данных размещаются на 1 компьютере);
  - распределенные (размещаются на нескольких компьютерах).
- По способу доступа к базе данных:
  - файл-серверные (например, Microsoft Access);
  - клиент-серверные (например, MySQL);
  - встраиваемые (например, Sybase SQL Anywhere).
- По типу управляемой базы данных:
  - иерархические (с древовидной структурой элементов, например, структура файлов и папок на компьютере);
  - сетевые (каждый элемент базы данных может быть связан с любым другим элементом);
  - реляционные (на базе двумерных массивов);
  - объектно-ориентированные (элементами являются модели объектов, включающих прикладные программы, которые управляются внешними событиями).

### **MS Access как пример СУБД Основные понятия MS Access**

#### **Понятие базы данных**

Хранение информации – одна из важнейших функций компьютера. Одним из распространенных средств такого хранения являются базы данных. *База данных* – это файл специального формата, содержащий информацию, структурированную заданным образом.

#### **Структура базы данных**

Большинство баз данных имеют табличную структуру. Как мы знаем, в табличной структуре адрес данных определяется пересечением строк и столбцов. В базах данных столбцы называются полями, а строки – запи-

сями. Поля образуют структуру базы данных, а записи составляют информацию, которая в ней содержится.

Для того чтобы легко усвоить понятие структуры базы данных, надо представить себе пустую базу, в которой пока еще нет никаких данных. Несмотря на то, что данных в базе нет, информация в ней все таки есть. Это структура базы, то есть набор полей. Они определяют, что будет записано в эту базу и в каком виде.

### **Простейшие базы данных**

Простейшие базы можно создавать, не прибегая к специальным программным средствам. Чтобы файл считался базой данных, информация в нем должна иметь структуру (поля) и быть форматирована так, чтобы содержимое соседних полей легко различалось. Простейшие базы можно создавать даже в текстовом редакторе Блокнот, то есть обычный текстовый файл при определенном форматировании тоже может считаться базой данных.

Существует, по крайней мере, два формата текстовых баз данных:

- с заданным разделителем;
- с фиксированной длиной поля.

Несмотря на «примитивность» таких текстовых баз данных, мощные системы управления базами данных позволяют импортировать подобные файлы и преобразовывать их в «настоящие» базы данных. Поэтому если в организации пока нет системы управления базами данных, данные можно хранить в текстовом файле, а потом, когда такая система появится, данные не пропадут и будут успешно импортированы.

### **Свойства полей. Типы полей**

Поля – это основные элементы структуры базы данных. Они обладают свойствами. От свойств полей зависит, какие типы данных можно вносить в поле, а какие нет, а также то, что можно делать с данными, содержащимися в поле.

Например, данные, содержащиеся в поле Цена, можно просуммировать, чтобы определить итоговый результат. Суммировать данные, содержащиеся в поле Номер телефона, совершенно бессмысленно, даже если номера телефонов записаны цифрами. Очевидно, что эти поля обладают разными свойствами и относятся к разным типам.

Основным свойством любого поля является его длина. Длина поля выражается в символах или, что то же самое, в знаках. От длины поля зависит, сколько информации в нем может поместиться. Мы знаем, что символы кодируются одним или двумя байтами, поэтому можно условно считать, что длина поля измеряется в байтах.

Очевидным уникальным свойством любого поля является его Имя. Разумеется, одна база данных не может иметь двух полей с одинаковым именем, поскольку компьютер запутается в их содержимом. Но кроме имени у поля есть еще свойство Подпись. Подпись – это та информация,

которая отображается в заголовке столбца. Ее не надо путать с именем поля, хотя если подпись не задана, то в заголовке отображается имя поля. Разным полям, например, можно задать одинаковые подписи. Это не мешает работе компьютера, поскольку поля при этом по-прежнему сохраняют разные имена.

Разные типы полей имеют разное назначение и разные свойства.

1. Основное свойство текстового поля – размер.
2. Числовое поле служит для ввода числовых данных. Оно тоже имеет размер, но числовые поля бывают разными, например, для ввода целых чисел и для ввода действительных чисел. В последнем случае кроме размера поля задается также размер десятичной части числа.
3. Поля для ввода дат или времени имеют тип Дата/время. Для ввода логических данных, имеющих только два значения (Да или Нет; 0 или 1; Истина или Ложь и т. п.), служит специальный тип – Логическое поле. Нетрудно догадаться, что длина такого поля всегда равна 1 байту, поскольку этого более чем достаточно, чтобы выразить логическое значение.
4. Особый тип поля – Денежный. Из названия ясно, какие данные в нем хранят. Денежные суммы можно хранить и в числовом поле, но в денежном формате с ними удобнее работать. В этом случае компьютер изображает числа вместе с денежными единицами, различает рубли и копейки, фунты и пенсы, доллары и центы, в общем, обращается с ними элегантнее.
5. В современных базах данных можно хранить не только числа и буквы, но и картинки, музыкальные клипы и видеозаписи. Поле для таких объектов называется полем объекта OLE.
6. У текстового поля есть недостаток, связанный с тем, что оно имеет ограниченный размер (не более 256 символов). Если нужно вставить в поле длинный текст, для этого служит поле типа МЕМО. В нем можно хранить до 65 535 символов. Особенность поля МЕМО состоит в том, что реально эти данные хранятся не в поле, а в другом месте, а в поле хранится только указатель на то, где расположен текст.
7. Очень интересно поле Счетчик. На первый взгляд это обычное числовое поле, но оно имеет свойство автоматического наращивания. Если в базе есть такое поле, то при вводе новой записи в него автоматически вводится число, на единицу большее, чем значение того же поля в предыдущей записи. Это поле удобно для нумерации записей.

### **Связанные таблицы**

Примеры, которые мы привели выше, можно считать простейшими базами данных, но на самом деле это не совсем базы, а только таблицы. Если бы информация хранилась в таких простых структурах, то для работы с ней можно было бы обойтись без специальных систем управления базами данных. На практике приходится иметь дело с более сложными структурами, которые образованы из многих связанных таблиц.



Базы данных, имеющие связанные таблицы, называют также реляционными базами данных.

Рассмотрим пример работы малого предприятия, занимающегося прокатом компакт-дисков с компьютерными играми. Для того чтобы знать, кто какой диск взял, когда должен возвратить и сколько дисков каждого наименования осталось на складе, предприятию необходима база данных. Но если все сведения о покупателях и о дисках хранить в одной таблице, то таблица станет очень неудобной для работы. В ней начнутся повторы данных. Всякий раз, когда гражданин Новиков В. П. будет брать очередной диск, придется вписывать его домашний адрес, телефон и паспортные данные. Так никто не работает. Это долго, трудно и чревато многочисленными ошибками.

Гораздо удобнее сделать несколько таблиц. В одной хранить сведения о клиентах со всеми их паспортными данными, в другой – сведения о выданных дисках, чтобы в любой момент узнать, что выдано клиенту и когда наступает срок возврата, а в третьей таблице – остаток дисков на складе, чтобы вовремя пополнять запасы. После этого отдельные поля таблиц связывают. Если из таблицы Прокат известно, что клиент НВП взял диск D001, то система управления базой данных мгновенно найдет в таблице Клиенты все паспортные данные этого человека, а в таблице Склад все данные об этом диске.

Разделение базы на связанные таблицы не только удобно, но иногда и необходимо. Например, для увеличения числа заказов менеджер фирмы, занимающейся прокатом компакт-дисков, решил поставить в общем зале компьютер, на котором каждый клиент может просмотреть список имеющихся дисков с иллюстрациями из игр.

Если база состоит только из одной таблицы, то вместе с информацией о дисках случайный посетитель получит доступ к информации о других клиентах фирмы. Вряд ли это понравится заказчикам. Такой менеджер не только не приобретет новых клиентов, но и растеряет тех, которых имел. Если данные в разных записях начинают повторяться, это может говорить о том, что база имеет плохую структуру. Надо подумать о том, нельзя ли разбить таблицу на группу связанных таблиц. Если заданы связи между таблицами, то работать с разными таблицами можно, как с одной цельной базой данных.

### **Поля уникальные и ключевые**

Создание базы данных всегда начинается с разработки структуры ее таблиц. Структура должна быть такой, чтобы при работе с базой требовалось вводить в нее как можно меньше данных. Если ввод каких-то данных приходится повторять неоднократно, базу делают из нескольких связанных таблиц. Структуру каждой таблицы разрабатывают отдельно.

Для того чтобы связи между таблицами работали надежно, и по записи из одной таблицы можно было однозначно найти записи в другой таблице, надо предусмотреть в таблице уникальные поля.

*Уникальное поле* – это поле, значения в котором не могут повторяться.

Если из таблицы Прокат известно, что клиент Новиков просрочил возврат взятого диска, то он должен уплатить штраф. Но в таблице Клиенты фирмы может быть несколько разных Новиковых, и компьютер не разберется, кто же из них должен платить штраф. Это означает, что поле Фамилия не является уникальным и потому его нельзя использовать для связи между таблицами. Поле номера телефона – более удачный кандидат на звание уникального поля, но, как вы понимаете, и одним телефоном могут пользоваться несколько разных людей.

Если ни одно поле таблицы не приемлемо в качестве уникального, его можно создать искусственно. В нашем примере в таблице Клиенты фирмы создано поле Шифр, которое образовано первыми тремя буквами фамилии и последними двумя цифрами номера телефона. Его и использовали для связи между таблицами.

Скорее всего, поле Шифр окажется уникальным, и проблем со связями между таблицами не возникнет, но было бы неплохо, если бы компьютер мог просигнализировать в том случае, если вдруг записи в этом поле повторятся. Для этого существует понятие ключевое поле. При создании структуры таблиц одно поле (или одну комбинацию полей) можно назначить ключевым. С ключевыми полями компьютер работает особо. Он проверяет их уникальность и быстрее выполняет сортировку по таким полям. Ключевое поле – очевидный кандидат для создания связей. Иногда ключевое поле называют первичным ключом.

Если при создании таблицы автор не задал ключевое поле, система управления базой данных вежливо напомнит о том, что поле первичного ключа таблице не помешает.

В качестве первичного ключа в таблицах часто используют поле, имеющее тип Счетчик. Ввести два одинаковых значения в такое поле нельзя по определению, поскольку приращение значения поля производится автоматически. Структура связей между таблицами называется схемой данных

### **СУБД Access**

Системы управления базами данных (СУБД) – это программные средства, с помощью которых можно создавать базы данных, наполнять их и работать с ними. В мире существует немало различных систем управления базами данных. Многие из них на самом деле являются не законченными продуктами, а специализированными языками программирования, с помощью которых каждый, освоивший данный язык, может сам создавать такие структуры, какие ему удобны, и вводить в них необходимые элемен-

ты управления. К подобным языкам относятся Clipper, Paradox, FoxPro и другие.

Необходимость программировать всегда сдерживала широкое внедрение баз данных в малом бизнесе. Крупные предприятия могли позволить себе сделать заказ на программирование специализированной системы «под себя». Малым предприятиям зачастую не по силам было не только решить, но даже и правильно сформулировать эту задачу.

Положение изменилось с появлением в составе пакета Microsoft Office системы управления базами данных Access.

С помощью Access обычные пользователи получили удобное средство для создания и эксплуатации достаточно мощных баз данных без необходимости что-либо программировать. В то же время работа с Access не исключает возможности программирования. При желании систему можно развивать и настраивать собственными силами. Для этого надо владеть основами программирования на языке Visual Basic.

Еще одним дополнительным достоинством Access является интегрированность этой программы с Excel, Word и другими программами пакета Office. Данные, созданные в разных приложениях, входящих в этот пакет, легко импортируются и экспортируются из одного приложения в другое.

### **Объекты Access**

Исходное окно Access отличается простотой и лаконичностью. Шесть вкладок этого окна представляют шесть видов объектов, с которыми работает программа.

1. Таблицы – основные объекты базы данных. С ними мы уже знакомы. В них хранятся данные. Реляционная база данных может иметь много взаимосвязанных таблиц.

2. Запросы – это специальные структуры, предназначенные для обработки данных базы. С помощью запросов данные упорядочивают, фильтруют, отбирают, изменяют, объединяют, то есть обрабатывают.

3. Формы – это объекты, с помощью которых в базу вводят новые данные или просматривают имеющиеся.

4. Отчеты – это формы «наоборот». С их помощью данные выдают на принтер в удобном и наглядном виде.

5. Макросы – это макрокоманды. Если какие-то операции с базой производятся особенно часто, имеет смысл сгруппировать несколько команд в один макрос и назначить его выделенной комбинации клавиш.

6. Модули – это программные процедуры, написаны на языке Visual Basic. Если стандартных средств Access не хватает, программист может расширить возможности системы, написав для этого необходимые модули.

### **Режимы работы с Access**

С организационной точки зрения в работе с любой базой данных есть два разных режима:

1. проектировочный

## 2. эксплуатационный (пользовательский).

*Создатель* базы имеет право создавать в ней новые объекты (например, таблицы), задавать их структуру, свойства полей, устанавливать необходимые связи. Он работает со структурой базы и имеет полный доступ к базе. У одной базы может быть один, два или несколько разработчиков.

*Пользователь* базы – это лицо, которое наполняет ее информацией с помощью форм, обрабатывает данные с помощью запросов и получает результат в виде результирующих таблиц или отчетов. У одной базы могут быть миллионы пользователей, и, конечно, доступ к структуре базы для них закрыт.

1. Взгляните на стартовое окно базы данных. Кроме шести вкладок для основных объектов оно содержит три командные кнопки: Открыть, Конструктор, Создать. С их помощью и выбирается режим работы с базой.

2. Кнопка Открыть открывает избранный объект. Если это таблица, ее можно просмотреть, внести новые записи или изменить те, что были внесены ранее.

3. Кнопка Конструктор тоже открывает избранный объект, но по-другому. Она открывает его структуру и позволяет править не ее содержимое, а устройство. Если это таблица, в нее можно вводить новые поля или изменять свойства существующих полей. Если это форма, в ней можно изменять или создавать элементы управления. Очевидно, что этот режим служит не для пользователей базы, а для ее разработчиков.

4. Действие командной кнопки Создать соответствует ее названию. Она служит для создания новых объектов. Этот элемент управления тоже предназначен для проектировщиков базы. Таблицы, запросы, формы и отчеты можно создавать несколькими разными способами: автоматически, вручную или с помощью Мастера.

## Работа в MS Access

### Таблицы. Создание таблиц

Таблицы – основные объекты базы данных. Без запросов, форм, отчетов и прочего можно обойтись, но если нет таблиц, то данные некуда записывать, а значит, нет и базы.

Создание базы начинается с создания первой таблицы. Создание таблицы состоит в задании ее полей и назначении их свойств. Оно начинается с щелчка на кнопке Создать в окне База данных.

1. Есть несколько способов создания новой таблицы, отличающихся уровнем автоматизации.

2. Самый «автоматичный» способ состоит в импорте таблиц из другой базы, может быть, даже созданной в другой системе. В зависимости от обстоятельств из импортируемой таблицы может поступить структура полей, их названия и свойства, а также и содержимое базы. Если что-то им-

портируется не совсем так, как надо, необходимые правки (например, в свойства полей) вносят вручную.

3. В тех случаях, когда речь идет о чужой таблице, которая находится на удаленном сервере и которую нельзя импортировать целиком, пользуются режимом Связь с таблицами. Это напоминает подключение к таблице для совместного использования ее данных.

4. Опытные разработчики пользуются Мастером таблиц. Это программа, ускоряющая создание структуры таблицы. Мастер задает ряд вопросов и, руководствуясь полученными ответами, создает структуру таблицы автоматически. Несмотря на то, что этот режим служит для упрощения работы, начинающим пользоваться им не рекомендуется, поскольку, не владея всей терминологией, легко запутаться в вопросах и ответах. Первые таблицы стоит попробовать создать вручную.

5. Пункт Режим таблицы открывает заготовку, в которой все поля имеют формальные имена: Поле1, Поле2... и т. д. и один стандартный текстовый тип. Такую таблицу можно сразу наполнять информацией.

6. Наиболее универсальный ручной метод предоставляет пункт Конструктор. В этом режиме можно самостоятельно задать имена полей, выбрать их тип и настроить свойства.

Для изменения свойств полей надо перейти в режим Конструктор щелчком на кнопке Вид. Чтобы вставить новое поле, надо установить указатель мыши на маркер поля и нажать клавишу INSERT. Чтобы удалить поле, его надо выделить и нажать клавишу DELETE. Закончив создание структуры, можно щелкнуть на кнопке Вид и перейти в Режим таблицы для заполнения ее данными.

### **Особенности таблиц баз данных**

Прежде чем мы приступим к изучению приемов работы с таблицами баз данных, надо обратить внимание на одну особенность всех баз данных, связанную с сохранением информации. Тех, кто привык работать с другими классами программ, она поначалу обескураживает.

Обычно с документом в программах можно делать все что угодно, пока не настала пора его сохранять. Испортив неаккуратными действиями исходный документ, можно отказаться от сохранения и вернуться к работе с прежней копией. В базах данных это не так.

Таблицы баз данных не являются самостоятельными документами. Сама база – это документ. Ей соответствует файл на диске, и мы можем сделать его копию. Структура таблиц – тоже документ. В некоторых системах она имеет отдельный файл, а в некоторых (например, в Access) такого файла нет, но структура таблиц входит в состав общего файла базы данных наряду с запросами, формами, отчетами и другими объектами. При изменении структуры таблицы система управления базой данных всегда выдает запрос на сохранение изменений.

Но содержание таблиц – это совсем другое дело. Его нельзя сохранить принудительной командой или, наоборот, отказаться от его сохранения. Все изменения в таблицах сохраняются автоматически в режиме реального времени. Режим реального времени означает, что, пока мы работаем с таблицей, происходит ее непрерывное сохранение. Как только заканчивается ввод данных в одно поле и происходит переход к следующему полю, данные немедленно записываются на жесткий диск.

Профессионалы высоко ценят эту особенность систем управления базами данных, а начинающих она иногда вводит в заблуждение. Экспериментируя с таблицами, надо знать, что все изменения, которые вносятся в их содержание, имеют необратимый характер. Нельзя что-то изменить, удалить, а потом отказаться от сохранения и вернуться к исходному варианту.

Эта особенность систем управления базами данных требует аккуратного отношения к работе с таблицами. Для экспериментов надо создавать отдельные копии базы или таблиц и работать с ними.

### **Надежность и безопасность баз данных**

Надежность баз данных имеет особую важность. Последствия утраты документа, созданного в текстовом процессоре или графическом редакторе, можно оценить затратами времени, необходимого для его воспроизведения. Утрата базы данных может привести к остановке целой отрасли промышленности и иметь глобальные последствия. Существуют базы данных, от которых зависит движение транспорта, работа банков и промышленных предприятий. Есть базы, содержащие жизненно важные сведения медицинского характера.

Создатели систем управления базами данных не могут полагаться на то, что конкретный пользователь не забудет своевременно дать команду Сохранить. Они учитывают и то, что во время работы может произойти аварийное отключение электричества. Ни при каких условиях информация не должна теряться, поэтому все изменения данных немедленно и автоматически сохраняются на диске.

### **Совместное использование данных**

Системы управления базами данных должны учитывать, что с базами могут одновременно работать много людей. Если бы с базами работали как с документами в текстовом процессоре, то один человек, открывший файл для редактирования, монополизировал бы этот файл и блокировал бы к нему доступ других пользователей до тех пор, пока файл не будет закрыт и сохранен.

В базах данных один пользователь, вносящий изменения в базу, блокирует только одну запись, с которой он работает, причем ненадолго. Например, известно, что службы автомобильной инспекции имеют базы данных угнанных автомобилей. Тот факт, что где-то в центральной службе идет ввод новых записей об угнанных автомобилях, не мешает инспекто-

рам на местах обращаться к базе по компьютерной сети и наводить необходимые справки. Как только ввод очередной записи завершается, она становится доступной всем инспекторам для просмотра, а некоторым (кому это положено по должности) и для редактирования.

Если в локальной или глобальной сети с одной базой работают несколько пользователей, то каждый может видеть в режиме реального времени те изменения, которые вносят в базу его коллеги.

### **Приемы работы с таблицами баз данных**

С базой данных можно работать обычными приемами управления с помощью мыши.

1. Строка состояния в нижней части окна в Access называется полем номера записи. Это поле содержит кнопки перехода, с помощью которых можно эффективно перемещаться по таблице.

2. Каждая запись имеет слева кнопку (маркер записи). Щелчок на этом маркере выделяет всю запись и готовит ее к копированию, перемещению, удалению.

3. Щелчок правой кнопкой на выделенной записи открывает контекстное меню для операций с записью.

4. Маркер, находящийся в левом верхнем углу таблицы, – это маркер таблицы. Щелчок на нем выделяет всю таблицу, а правый щелчок открывает контекстное меню для операций с таблицей в целом.

5. Поля базы данных представлены в таблице столбцами. Каждый столбец имеет заголовок, в котором записано имя поля или то значение, которое задано в свойстве Подпись.

6. Если содержимое поля не полностью уместается в ячейке таблицы, столбец можно расширить. При наведении указателя мыши на границу между столбцами указатель меняет форму. Теперь границу можно перемещать методом перетаскивания, а двойной щелчок, выполненный в этот момент, автоматически устанавливает ширину столбца равной длине самого длинного значения в данном поле.

7. Щелчок на заголовке столбца выделяет весь столбец, а щелчок правой кнопкой на выделенном столбце открывает контекстное меню. В нем есть очень интересные пункты, позволяющие отсортировать записи по данному полю, вставить новый столбец, скрыть столбец и прочее.

8. Скрытый столбец не исчезает из базы, а только перестает отображаться на экране. Чтобы снова его отобразить, надо привести указатель на границу между столбцами в том месте, где был скрыт столбец, и выполнить двойной щелчок. Скрытый столбец опять станет видимым.

### **Создание связей между таблицами**

Основные преимущества систем управления базами данных реализуются при работе не с отдельными таблицами, а с группами взаимосвязанных таблиц. Для создания связей между таблицами СУБД Access имеет специальное диалоговое окно, которое называется Схема данных.

1. Окно Схема данных открывают щелчком на одноименной кнопке панели инструментов или командой Сервис – Схема данных.

2. Если ранее никаких связей между таблицами базы не было, то при открытии окна Схема данных одновременно открывается окно Добавление таблицы, в котором можно выбрать нужные таблицы для включения в структуру межтабличных связей.

3. Если связи между таблицами уже были заданы, то для введения в схему данных новой таблицы надо щелкнуть правой кнопкой мыши на схеме данных и в контекстном меню выбрать пункт Добавить таблицу.

4. Введя в схему данных все таблицы, которые надо связать, можно приступать к созданию связей между полями таблиц.

5. Связь между полями устанавливают путем перетаскивания имени поля из одной таблицы в другую на соответствующее ему связанное поле.

6. После перетаскивания открывается диалоговое окно Связи, в котором можно задать свойства образующейся связи.

7. Включение флажка Обеспечение условия целостности данных позволяет защититься от случаев удаления записей из одной таблицы, при которых связанные с ними данные других таблиц останутся без связи.

8. Чтобы условие целостности могло существовать, поле основной таблицы должно обязательно быть ключевым и оба поля должны иметь одинаковый тип.

9. Флажки Каскадное обновление связанных полей и Каскадное удаление связанных записей обеспечивают одновременное обновление или удаление данных во всех подчиненных таблицах при их изменении в главной таблице. Если клиент Соколова выйдет замуж и изменит фамилию на Воронову, то придется внести изменение только в поле Фамилия таблицы Клиенты. В прочих таблицах изменения произойдут автоматически.

Диалоговое окно Схема данных наглядно отображает связи между таблицами. Чтобы удалить связь, надо щелкнуть на линии связи правой кнопкой мыши и воспользоваться командой Удалить контекстного меню.

### **Запросы**

Предположим, что на крупном предприятии есть огромная база данных Кадры, содержащая подробнейшие сведения о каждом сотруднике. Кроме формальной информации база может содержать и конфиденциальную, например, сведения о заработной плате. Вся эта информация хранится в базовых таблицах.

Работать с базой данных Кадры могут разные подразделения предприятия, и всем им нужны разные данные. Не все то, что положено знать службе безопасности предприятия, должно быть доступно главному врачу, и наоборот. Поэтому доступ пользователей к базовым таблицам закрывают.



Для доступа к данным есть другое, гораздо более гибкое и удобное средство – запросы. Для одной и той же таблицы можно создать множество разных запросов, каждый из которых сможет извлекать из таблицы лишь малую часть информации, но именно ту часть, которая в данный момент необходима. У сотрудника бухгалтерии должен быть запрос, который позволит определить, сколько дней в году по болезни отсутствовал тот или иной работник, но у него не должно быть запроса, позволяющего узнать, чем он болел и где лечился, а у главного врача такой запрос быть должен.

В результате работы запроса из общей исходной базы формируется результирующая таблица, содержащая часть общей информации, соответствующую запросу.

Важным свойством запросов является то, что при создании результирующей таблицы можно не только выбирать информацию из базы, но и обрабатывать ее. При работе запроса данные могут упорядочиваться (сортироваться), фильтроваться (отсеиваться), объединяться, разделяться, изменяться, и при этом никаких изменений в базовых таблицах может не происходить. Результаты обработки сказываются только на содержании результирующей таблицы, а она имеет временный характер, и иногда ее даже называют моментальным снимком.

И еще одним ценным свойством запросов является их способность выполнять итоговые вычисления. Запрос может не только выдать результирующую таблицу, но и найти, например, среднее (наибольшее, наименьшее, суммарное и т. п.) значение по какому-то полю.

Начинающих пользователей баз данных часто удивляет отсутствие в таблицах элементарной возможности вставить новую запись между другими. Записи всегда добавляются только в конец базы. На вопрос, почему так происходит, ответ простой: потому что в упорядочении таблиц нет необходимости. Для этого существуют запросы. Совершенно неважно, под каким номером внесена в таблицу та или иная запись. Если нужно видеть ее в строго определенном месте (например, рядом с другими аналогичными), значит нужно создать запрос, который сгруппирует записи по заданному признаку.

### **Запросы на выборку**

Существует немало различных видов запросов, но самые простые из них и, к тому же, используемые наиболее часто – это запросы на выборку.

Цель запроса на выборку состоит в создании результирующей таблицы, в которой отображаются только нужные по условию запроса данные из базовых таблиц.

Как и другие объекты Access, запросы можно создавать автоматически с помощью Мастера или вручную. И, как обычно, на этапе обучения лучше не пользоваться Мастером, чтобы почувствовать работу с запросами «кончиками пальцев».

Для создания запросов к базам данных существует специальный язык запросов. Он называется SQL (Structured Query Language – структурированный язык запросов). К счастью, те, кто пользуются СУБД Access, могут позволить себе не изучать этот язык. Вместо него в Access есть простое средство, которое называется бланком запроса по образцу. С его помощью можно сформировать запрос простыми приемами, перетаскивая элементы запроса между окнами.

#### Выбор базовых таблиц для запроса

1. Создание запроса к базе начинается с открытия вкладки Запросы диалогового окна База данных и щелчка на кнопке Создать.

2. В открывшемся диалоговом окне Новый запрос задают ручной режим создания запроса выбором пункта Конструктор.

3. Создание запроса в режиме Конструктора начинают с выбора тех таблиц базы, на которых будет основан запрос.

4. Выбор таблиц выполняют в диалоговом окне Добавление таблицы. В нем отображаются все таблицы, имеющиеся в базе.

5. Выбранные таблицы заносят в верхнюю половину бланка запроса по образцу щелчком на кнопке Добавить.

6. В окне Добавление таблицы обратите внимание на наличие трех вкладок: Таблицы, Запросы, Запросы и таблицы. Они говорят о том, что запрос не обязательно основывать только на таблицах. Если ранее уже был создан запрос, то новый запрос можно основывать и на нем. Какие именно таблицы использовать в качестве базовых, решает сам создатель запроса.

#### **Заполнение бланка запроса по образцу**

Бланк запроса по образцу – удивительно изящное и удобное средство создания запросов.

1. Бланк запроса по образцу имеет две панели. На верхней панели расположены списки полей тех таблиц, на которых основывается запрос.

2. Строки нижней панели определяют структуру запроса, то есть структуру результирующей таблицы, в которой будут содержаться данные, полученные по результатам запроса.

3. Строку Поле заполняют перетаскиванием названий полей из таблиц в верхней части бланка. Каждому полю будущей результирующей таблицы соответствует один столбец бланка запроса по образцу.

4. Строка Имя таблицы заполняется автоматически при перетаскивании поля.

5. Если щелкнуть на строке Сортировка, появится кнопка раскрывающегося списка, содержащего виды сортировки. Если назначить сортировку по какому-то полю, данные в результирующей таблице будут отсортированы по этому полю.

6. Бывают случаи, когда поле должно присутствовать в бланке запроса по образцу, но не должно отображаться в результирующей таблице.

В этом случае можно запретить его вывод на экран, сбросив соответствующий флажок.

7. Самая интересная строка в бланке запроса по образцу называется Условие отбора. Именно здесь и записывают тот критерий, по которому выбирают записи для включения в результирующую таблицу. По каждому полю можно создать свое условие отбора. В нашем примере назначены два условия отбора: повесу игрока (более 80 кг) и по росту (менее 190 см).

8. Запуск запроса выполняют щелчком на кнопке Вид. При запуске образуется результирующая таблица.

9. Чтобы выйти из результирующей таблицы и вернуться к созданию запроса в бланке запроса по образцу, нужно еще раз щелкнуть на кнопке Вид.

### **Зачем нужен флажок Вывод на экран?**

Возникает один закономерный вопрос. Зачем нужен флажок Вывод на экран?

Если содержимое поля не надо выводить на экран, то, может быть лучше вообще не включать это поле в бланк запроса по образцу?

Случаи, когда присутствие поля в бланке необходимо, обычно связаны с использованием этого поля для сортировки. Но если при этом сведения в данном поле конфиденциальные, то поле скрывают.

В студии, занимающейся прокатом видеокассет, посетителям могут предложить для просмотра базу видеофильмов, отсортированную в порядке убывания популярности. Чем чаще кассету берут в прокат, тем выше она находится в общем списке.

Но если владелец студии не желает, чтобы любой посетитель мог точно узнать, как часто берутся в прокат те или иные кассеты, то поле, по которому выполнена сортировка, делают скрытым.

### **Запросы с параметром**

Во многих случаях пользователю надо предоставить возможность выбора того, что он хочет найти в таблицах базы данных. Для этого существует специальный вид запроса – запрос с параметром.

Предположим, что в базе данных есть таблица, в которой содержатся все результаты чемпионатов мира по футболу. Наша задача – создать запрос, с помощью которого пользователь может определить, в каком году та или иная команда занимала первое место, причем выбор этой команды – его личное дело. Для этой цели служит специальная команда языка SQL, которая выглядит так: LIKE [...]

В квадратных скобках можно записать любой текст, обращенный к пользователю, например:

LIKE [Введите название страны]

1. Команду LIKE надо поместить в строке Условие отбора и в том поле, по которому производится выбор. В нашем случае это столбец сборных, занимавших первые места в чемпионатах мира по футболу.

2. После запуска запроса открывается диалоговое окно, в котором пользователю предлагается ввести параметр. Если в качестве параметра ввести слово Бразилия, то выдается результирующая таблица, содержащая записи по тем чемпионатам, когда сборная Бразилии становилась чемпионом.

Если в качестве параметра ввести слово Италия, то результирующая таблица будет иной.

Разумеется, в нашей небольшой таблице и без запроса нетрудно найти сборные, занимавшие призовые места. Но без запроса не обойтись, если в базе содержатся сотни тысяч записей, причем расположенные в разных таблицах.

### **Вычисления в запросах**

Поле, содержимое которого является результатом расчета по содержимому других полей, называется вычисляемым полем.

Вычисляемое поле существует только в результирующей таблице. В исходных (базовых) таблицах такое поле не создается, и при работе обычного запроса таблицы не изменяются. Неправда ли, это очень разумно? Каждый, кто обращается к базе, может с помощью запросов как угодно манипулировать данными и получать любые результаты, но при этом исходные таблицы остаются неизменно одинаковыми для всех пользователей.

1. Для создания запроса, производящего вычисления, служит тот же самый бланк запроса по образцу. Разница только в том, что в одном из столбцов вместо имени поля записывают формулу. В формулу входят заключенные в квадратные скобки названия полей, участвующих в расчете, а также знаки математических операций, например, так: **Стоимость:[Количество]\*[Цена]**

2. В узкий столбец непросто записать длинную формулу, но если нажать комбинацию клавиш SHIFT F2, то открывается вспомогательное диалоговое окно, которое называется Область ввода. В нем можно ввести сколь угодно длинную формулу, а потом щелчком на кнопке ОК перенести ее в бланк запроса по образцу.

3. Если включить отображение вычисляемого поля, результаты расчетов будут выдаваться в результирующей таблице.

4. Ничто не мешает сделать вычисляемое поле полем сортировки, чтобы не только получать новые результаты, но и анализировать их.

### **Итоговые запросы**

Запросы позволяют не только отбирать нужную информацию из таблиц и обрабатывать ее путем создания новых (вычисляемых) полей, но и производить так называемые итоговые вычисления.

Примером итогового вычисления может служить сумма всех значений в какой-то группе записей или их среднее значение, хотя кроме суммы и среднего значения существуют и другие итоговые функции.

Поскольку итоговые функции для одной записи не имеют смысла и существуют только для группы записей, то предварительно записи надо сгруппировать по какому-либо признаку.

1. Рассмотрим работу салона, занимающегося продажей подержанных автомобилей. Результаты работы салона за последнюю неделю содержатся в таблице. В ней можно выделить несколько групп по разному признаку. Записи можно сгруппировать по моделям автомобилей (ВАЗ – отдельно и БМВ – отдельно) или по году выпуска (1989, 1993 и т. д.). Для каждой из групп можно провести итоговое вычисление по полю Цена.

2. Итоговые запросы создают на основе известного нам бланка запроса по образцу, только теперь в нем появляется дополнительная строка – Группировка.

3. Для введения этой строки в бланк надо щелкнуть на кнопке Групповые операции на панели инструментов программы Access .

Далее все происходит очень просто.

4. В тех полях, по которым производится группировка, надо установить (или оставить) функцию Группировка.

5. В тех полях, по которым следует провести итоговое вычисление, надо в строке Группировка раскрыть список и выбрать одну из нескольких итоговых функций.

6. Щелчок на кнопке Вид или Запуск запускает запрос и выдает результирующую таблицу с необходимыми итоговыми данными.

7. В строке Группировка можно указать лишь одну итоговую функцию. А как быть, если надо найти и сумму, и среднее, и максимальное значение, и еще что-то? Решение простое: одно и то же поле можно включить в бланк запроса по образцу несколько раз.

### **Запросы на изменение**

Выше мы говорили о том, что все виды запросов на выборку создают временные результирующие таблицы. Базовые таблицы при этом не изменяются. Тем не менее, специально для разработчиков баз данных существует особая группа запросов, которые называются запросами на изменение. Они позволяют автоматически создавать новые таблицы или изменять уже имеющиеся. Логика использования запросов на изменение такая:

- создается запрос на выборку, который отбирает данные из разных таблиц или сам создает новые данные путем вычислений;
- после запуска запроса образуется временная результирующая таблица;
- данные из этой временной таблицы используют для создания новых таблиц или изменения существующих.

Существует несколько видов запросов на изменение. Самый простой и понятный – это запрос на создание таблицы. Вернемся к примеру с расчетом среднего количества забитых мячей.

1. Предположим, что разработчик таблицы Итоги по командам захотел включить в нее поле Результативность. Конечно, он может рассчитать среднее количество мячей, забитых за игру каждой командой, но если ввести в таблицу такое поле, то его придется заполнять его вручную. Для таблиц, содержащих много записей, это решение неприемлемо.

2. Проще создать запрос на выборку, в который войдут все поля базовой таблицы плюс новое вычисляемое поле.

3. Щелчок на кнопке Вид позволяет убедиться, что запрос работает как положено и создает результирующую таблицу, более полную чем базовая. Теперь можно дать команду на создание новой базовой таблицы, равной результирующей.

4. Эта команда находится в меню Запрос, которое доступно только в режиме Конструктора.

5. В том же меню присутствуют команды для создания запросов на обновление данных, на добавление записей и на удаление записей. Все они относятся к запросам на изменение и работают аналогично, изменяя базовые таблицы в соответствии с данными результирующих таблиц.

### **Формы**

Обычно разработчик базы данных создает структуру таблиц и запросов, но заполнением таблиц информацией он не занимается. Для этого есть специальные кадры (обычно малоквалифицированные), выполняющие функции наборщиков. Для упрощения их труда разработчик базы может подготовить специальные объекты – формы.

Форма представляет собой некий электронный бланк, в котором имеются поля для ввода данных. Наборщик вводит данные в эти поля, и данные автоматически заносятся в таблицы базы.

### **Зачем нужны формы?**

Данные в таблицу можно вносить и без помощи каких-либо форм, но существуют, по крайней мере, четыре причины которые делают формы незаменимым средством ввода данных в базу.

1. Малоквалифицированному персоналу нельзя предоставлять доступ к таблицам (самому ценному из того, что есть в базе) Представьте, что будет, если новичок «наведет порядок» в таблице банка, хранящей расчетные счета клиентов.

2. Разные люди могут иметь разные права доступа к информации, хранящейся в таблицах. Например, один имеет право вводить только имена и адреса клиентов, другой – только номера их расчетных счетов, а третий – только денежные суммы, хранящиеся на этих счетах. Сговор между этими людьми должен быть исключен. Для ввода данных им предоставляют разные формы, хотя данные из форм могут поступать в одну таблицу.

3. Ввод данных в таблицу – чрезвычайно утомительное занятие. Уже после нескольких часов работы люди делают ошибки. Ввод данных в форму проще. Здесь многое можно автоматизировать. К тому же, элементы

управления форм настраивают таким образом, чтобы при вводе данных выполнялась их первичная проверка.

4. Информацию для баз данных берут из бумажных бланков: анкет, заявлений, накладных, счетов, описей, ведомостей, справок. Экранные формы можно сделать точной копией бумажных бланков, с которых происходит ввод данных. Благодаря этому во много раз уменьшается количество ошибок при вводе и значительно снижается утомляемость персонала.

### **Создание форм**

Как и другие объекты Access, формы можно создавать вручную или автоматически, причем несколькими способами. При создании таблиц и запросов мы рекомендовали на первых порах автоматическими средствами не пользоваться, чтобы вникнуть в терминологию и подготовить себя к работе с Мастером, задающим непонятные для начинающих вопросы. С формами дело обстоит иначе. Они состоят из многочисленных элементов управления, и от того, насколько аккуратно эти элементы расположены на экране, зависит внешний вид формы.

*Автоматические средства позволяют создавать аккуратные формы и не задают пользователю лишних вопросов. Начинать работу лучше с них.*

### **Автоформы**

Автоформы – самый простой вид автоматических форм. Для создания автоформы надо открыть вкладку Формы в диалоговом окне База данных и щелкнуть на кнопке Создать – откроется окно Новая форма.

В диалоговом окне Новая форма выбирают в качестве источника данных для формы какую-либо таблицу или запрос, после чего создают автоформу двойным щелчком в списке выбора вида автоформы (табличная, ленточная или в столбец).

### **Создание формы с помощью Мастера**

1. С помощью Мастера форма создается всего в четыре этапа:

1. выбор полей, данные для которых можно будет вводить в форме;

2. выбор внешнего вида формы (один из четырех);

3. выбор фонового рисунка формы (один из десяти);

4. задание имени формы.

5. Все эти пункты достаточно хорошо объяснены в Мастере и не требуют никаких пояснений.

2. Готовую форму можно сразу же использовать для просмотра существующих записей или для ввода новых.

### **Структура форм**

Создавая формы автоматическими средствами, можно не задумываться над их структурой, но при разработке формы вручную со структурой приходится иметь дело.

Структуру формы составляют ее разделы, а разделы содержат элементы управления.

### **Разделы формы**

1. Самый простой способ познакомиться с разделами формы состоит в том, чтобы взять готовую форму, например, созданную с помощью Мастера, и посмотреть ее устройство в режиме Конструктора. Как мы уже знаем, для этого надо щелкнуть на кнопке Вид на панели управления Access.

2. При просмотре в Конструкторе мы видим структуру формы. Обратите внимание на то, что рядом с ней открывается панель элементов, содержащая заготовки и инструменты для создания элементов управления формы.

3. В структуре формы четко видны три раздела: раздел заголовка формы, область данных и раздел примечания формы.

4. Все, что содержится в области данных, является элементами управления. В нашем случае здесь присутствуют элементы управления только двух типов: связанное поле (то, что в него вводится, поступает и в одноименное поле таблицы, на базе которой создана форма) и присоединенная надпись (называется так, поскольку перемещается вместе со своим элементом управления). В нашем случае содержание присоединенной надписи совпадает с названием связанного поля, но, как вы понимаете, это можно и изменить.

5. Фоновый рисунок, лежащий под элементами управления, показывает размер рабочего поля формы.

6. Размеры разделов и размеры рабочего поля формы можно изменять с помощью мыши. При наведении на границу раздела указатель меняет форму. В этот момент границу можно перемещать методом перетаскивания.

### **Создание надписей**

Редактирование форм состоит в создании новых или изменении имеющихся элементов управления, а также в изменении их взаимного расположения.

При рассмотрении приемов создания новых элементов управления мы воспользуемся тем фактом, что Мастер, создавший форму, не заполнил ее раздел заголовка.

1. Перетаскив вниз разделительную границу между заголовком и областью данных, мы можем освободить сверху достаточно места для создания крупной надписи.

2. На панели элементов существует специальный элемент управления для создания заголовков, который называется Надпись.

3. Щелкнув на нем, а потом на форме, мы получаем текстовую рамку, в которую можно вводить произвольный текст. При вводе текста не надо заботиться о его форматировании. Неважно, как он выглядит и где



расположен. Закончив ввод, надо нажать клавишу ENTER, после чего можно приступить к оформлению текста.

4. Для форматирования элемента управления его надо сначала выделить. Для этого служит инструмент Выбор объектов.

5. При выделении элемента управления вокруг него образуется рамка с восемью маркерами (по углам и по центрам сторон рамки). Рамку можно растягивать или сжимать методом перетаскивания границ. При наведении на маркер указатель мыши меняет форму, принимая изображение открытой ладони. В этот момент рамку можно перемещать.

6. Особую роль играет левый верхний маркер рамки. При наведении на него указатель мыши принимает форму указательного пальца. О роли этого маркера мы расскажем чуть позже.

7. Когда объект выделен, можно изменять параметры шрифта, метод выравнивания текста и другие элементы форматирования. Это выполняют обычными средствами форматирования, доступными через соответствующую панель инструментов Access.

8. Если щелкнуть на выделенном элементе правой кнопкой мыши, откроется его контекстное меню, в котором имеются дополнительные возможности изменения оформления. В нашем случае, например, применено Оформление с тенью

### **Создание и редактирование связанных полей**

1 Заголовок таблицы, который мы только что создали, не связан ни с одним из полей таблицы. Поэтому элемент управления Надпись еще называют свободным полем. Текст, введенный в него, остается неизменным независимо от того, какую запись в этот момент просматривают в форме.

2. Совсем иначе обстоит дело с элементами управления, в которых отображается содержимое полей таблицы. Такие элементы управления называют связанными полями.

3. Для их создания служит элемент Поле на панели элементов.

4. При создании связанного поля вместе с ним одновременно образуется еще один элемент управления – присоединенная надпись. Она перемещается вместе со связанным полем и образует с ним единое целое.

5. Обратите внимание на то, что, что слово «Результативность» в присоединенной надписи записано без последней буквы. Это не ошибка. Просто Мастер, создававший форму, сделал это неаккуратно, и связанное поле «наехало» на присоединенную надпись.

6. Оторвать поле от присоединенной надписи позволяет уже упомянутый маркер, расположенный в левом верхнем углу. При наведении на него указатель мыши принимает форму указательного пальца. В этот момент связанное поле можно оторвать от присоединенной надписи и перемещать отдельно.

7. Перемещать элементы управления и изменять их размеры с помощью мыши не слишком удобно. Гораздо удобнее использовать для этой

цели курсорные клавиши в комбинации с клавишами SHIFT или CTRL. В первом случае происходит изменение размеров элемента управления, а во втором – изменение его расположения.

8. Чтобы элементы управления располагались в форме ровными рядами, существуют специальные команды выравнивания. Сначала надо выделить группу элементов управления с помощью инструмента Выбор объектов (группа выбирается при нажатой клавише SHIFT), а потом дать команду Формат – Выровнять и выбрать метод выравнивания.

### **Прочие элементы управления формы**

При создании формы вручную элементы управления размещают на ней так, как удобно проектировщику. Созданные элементы управления формы выравнивают с помощью команды Формат – Выровнять.

Кроме рассмотренных выше элементов управления Надпись и Поле, существует еще несколько полезных элементов управления.

1. *Переключатели (радиокнопки)*. С ними можно связать команды, например, выполняющие фильтрацию.

2. *Флажки*. Действуют аналогично переключателям, но в отличие от них, допускают множественный выбор. Удобны для управления режимами сортировки данных.

3. *Список*. Может содержать фиксированный набор значений или значения из заданного поля одной из таблиц. Позволяет не вводить данные, а выбирать их из списка.

4. *Поле со списком*. Применяется так же, как и список, но занимает меньше места в форме, поскольку список открывается только после щелчка на раскрывающей кнопке.

5. *Командные кнопки*. С каждой из них можно связать какую-либо полезную команду, например команду поиска записи, перехода между записями и другие.

6. *Вкладки*. Позволяют разместить много информации на ограниченной площади. На вкладках размещают другие элементы управления.

7. *Поле объекта OLE*. Служит для размещения внешнего объекта, соответствующего принятой в Windows концепции связывания и внедрения объектов. Объектом, как правило, является иллюстрация, например фотография, но это может быть и видеозапись, и музыкальный фрагмент, и голосовое сообщение.

Существуют два типа полей для размещения объектов OLE: Свободная рамка объекта и Присоединенная рамка объекта. В первом случае, рамка не связана ни с каким полем таблиц базы данных. Объект, находящийся в ней, выполняет роль иллюстрации и служит для оформления формы. С присоединенной рамкой связано одно из полей таблицы. В ней отображается содержимое этого поля. Это содержимое может меняться при переходе от одной записи к другой.

## **Отчеты**

Напомним функции основных объектов базы данных:

- таблицы служат для хранения данных;
- запросы служат для выбора данных из таблиц, а также для автоматизации операций по обновлению и изменению таблиц;
- формы служат для упрощения операций ввода данных в таблицы, но могут быть использованы и для просмотра результатов работы запросов на экране.

Из основных объектов нам осталось рассмотреть только отчеты. Отчеты во многом похожи на формы и тоже позволяют получить результаты работы запросов в наглядной форме, но только не на экране, а в виде распечатки на принтере. Таим образом, в результате работы отчета создается бумажный документ.

### **Автоотчеты**

Большая часть того, что было сказано о формах, относится и к отчетам. Выбрав в диалоговом окне База данных вкладку Отчеты и щелкнув на кнопке Создать, мы получаем диалоговое окно Новый отчет, позволяющее создать отчет автоматически (автоотчет), с помощью Мастера или вручную. Точно так же, как и с формами, с отчетами удобнее знакомиться в режиме автоматического создания. Создайте на основе любой таблицы автоотчет в столбец или ленточный. Операция настолько проста, что сводится к одному щелчку левой кнопки мыши.

Отчеты предназначены для вывода информации на принтер, поэтому для расчета расположения данных на печатной странице программа Access должна «знать» все необходимое об особенностях принтера. Эти данные Access получает от операционной системы. Соответственно, принтер в системе должен быть установлен. При отсутствии принтера отчеты создавать все-таки можно. Достаточно выполнить программную установку с помощью команды операционной системы: Пуск – Настройка Принтеры – Установка принтера, после чего зарегистрировать драйвер принтера, либо взяв его с гибкого диска, либо выбрав один из драйверов, прилагающихся к самой операционной системе.

### **Структура отчета**

Как и формы, отчеты состоят из разделов, а разделы могут содержать элементы управления. Но, в отличие от форм, разделов в отчетах больше, а элементов управления, наоборот, меньше. Со структурой отчета проще всего ознакомиться, создав какой-либо автоотчет, а затем открыв его в режиме Конструктора.

1. Структура отчета состоит из пяти разделов: заголовка отчета, верхнего колонтитула, области данных, нижнего колонтитула и примечания отчета. По сравнению с формами новыми являются разделы верхнего и нижнего колонтитулов.

2. Раздел заголовка служит для печати общего заголовка отчета.

3. Раздел верхнего колонтитула можно использовать для печати подзаголовков, если отчет имеет сложную структуру и занимает много страниц. Здесь можно также помещать и колонцифры (номера страниц), если это не сделано в нижнем колонтитуле.

4. В области данных размещают элементы управления, связанные с содержимым полей таблиц базы. В эти элементы Управления выдаются данные из таблиц для печати на принтере. Порядок размещения и выравнивания элементов управления тот же, что и при создании структуры форм.

5. Раздел нижнего колонтитула используют для тех же целей что и раздел верхнего колонтитула. В нашем случае в нем размещены два элемента управления.

6. В первом элементе управления выводится текущая дата. Для этого использована встроенная в Access функция Now(). Она возвращает текущую дату и помещает ее в поле, а отчет воспроизводит ее при печати.

7. Во втором элементе управления выводится номер страницы и общее количество страниц. Для их определения использованы встроенные функции Page() и Pages(). Тот текст, который записан в кавычках, воспроизводится «буквально», а оператор & служит для «склеивания» текста, заключенного в кавычки, со значениями, возвращаемыми функциями. Оператор & называется оператором конкатенации.

8. Раздел примечания используют для размещения дополнительной информации. В нашем примере он не использован.

### **Закрепление пройденного**

Системы управления базами данных (СУБД) – это специальные программные средства, предназначенные для работы с файлами баз данных (файлами специального формата, содержащими информацию, структурированную заданным образом).

Современные СУБД позволяют хранить в виде файлов данные любых типов: числовые, текстовые, графические, звуковые, видео и прочие.

Данные в базах хранятся в виде таблиц. Каждая таблица имеет структуру.

Структура таблицы определяется составом ее полей и их свойствами. Важнейшими свойствами полей являются: тип поля и размер поля. Для хранения разных типов данных используют поля соответствующих типов.

Данные, хранящиеся в таблице, можно изменять, удалять, сортировать, фильтровать, размножать и выполнять с ними другие операции. Для автоматизации операций по работе с данными, в частности, для отбора нужных данных, применяют специальные объекты, которые называются запросами.

В СУБД Access запросы создают с помощью специального бланка запроса по образцу.

С помощью запросов на выборку производят выбор данных из базы, их обработку, выполнение итоговых вычислений и другие операции. По результатам работы запроса создается временная результирующая таблица.

На основе результирующей таблицы, можно создавать новые таблицы или изменять существующие. Для этого служат запросы на изменение.

Для ввода данных в таблицы или для просмотра данных в наглядной форме служат специальные объекты, называемые формами. Формы – экранные объекты.

Структура форм состоит из разделов и элементов управления. Проектирование формы состоит в размещении элементов управления на бланке формы и в задании связей между этими элементами и полями таблиц или запросов базы данных.

Создание форм можно выполнять автоматически (автоформы), полуавтоматически (с помощью Мастера) или вручную (в режиме Конструктора).

Размещение элементов управления на бланке формы автоматизировано. В большинстве случаев при создании нового элемента запускается программа Мастер, с помощью которой происходит настройка свойств элемента управления.

Для создания печатных документов, которые содержат информацию из базовых таблиц или из результирующих таблиц, полученных по результатам работы запросов, служат специальные объекты – отчеты.

Отчеты отличаются от форм тем, что предназначены не для ввода данных, а только для вывода, а также тем, что создают не экранные, а печатные документы.

Структура отчетов, как и форм, состоит из разделов и элементов управления. Проектирование отчета состоит в создании структуры его разделов и в размещении элементов управления внутри этих разделов, а также в задании связей между этими элементами и полями таблиц или запросов базы данных.

Создание отчетов может выполняться автоматически (автоотчеты), полуавтоматически (с помощью Мастера) или вручную (в режиме Конструктора).

Таблицы, запросы, формы и отчеты являются основными объектами базы данных. Их разрабатывает разработчик базы. Пользователь базы использует эти объекты без вмешательства в их структуру.

Разработчик базы данных имеет также два типа дополнительных объектов: макросы и модули. Эти объекты создают в тех случаях, когда стандартных средств управления базой данных оказывается недостаточно для выполнения операций, необходимых заказчику системы. С помощью макросов создают макрокоманды, упрощающие наиболее утомительные операции с базой, а с помощью модулей, написанных на языке программи-

рования Visual Basic, создают программные процедуры для выполнения нестандартных операций.