Костанайский государственный педагогический институт

Естественно-математический факультет

Кафедра информатики и компьютерных технологий

Сухов М.В.

Базы данных и информационные системы

Учебное пособие

Костанай, 2016

УДК 004.65
ББК 32.972.134
С 91

Рецензенты:
Медетов Н.А. – доктор физико-математических наук, доцент (Костанайский государственный университет им. А. Байтурсынова)
Клименко И.С. – доктор технических наук, доцент (Костанайский государственный педагогический институт)

Сухов М.В.
С91 Базы данных и информационные системы: учебное пособие / М.В. Сухов. – Костанай, 2016. – 180 с.

ISBN 978-601-7839-15-4

Учебное пособие посвящено вопросам проектирования и использования информационных систем организационного уровня – базам данных (БД). Подробно изложены теоретические основы построения баз данных, их структура, специфика и возможности применения. Дана характеристика моделей представления данных, рассмотрены CASE-системы, реляционная модель данных и проектирование реляционных баз данных, защита и администрирование БД. Основной целью данного пособия является формирование концептуальных представлений об основных принципах построения БД и СУБД, принципах проектирования БД; а также анализ основных технологий реализации БД. Особое внимание уделяется описанию возможностей современных систем управления базами данных, технологиям разработки и использования информационных систем, в том числе в учебных целях.

Учебное пособие предназначено студентам и преподавателям высшей школы, научным работникам, интересующимся вопросами создания и использования баз данных и информационных систем.

УДК 004.65
ББК 32.972.134

Печатается по решению Ученого совета
Костанайского государственного педагогического института

ISBN 978-601-7839-15-4

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Применение новых информационных и цифровых технологий в современном обществе стало делом повседневным. Практически на каждом шагу нас окружают процессы и явления, которые связанны с компьютерной техникой и программным обеспечением. Поскольку количество компьютерных программ решающих различные задачи большое множество существуют целые классы программ ориентированные на различные цели. Базы данных относятся к числу программ предназначенных для организации централизованного накопления, хранения, обработки и передачи информации. Информация может быть представлена как небольшой программой ведущей учет коммунальных платежей для дома, так и комплексом аппаратно-программных средств позволяющих организовать деятельность крупных предприятий.

От правильного выбора инструментальных средств создания информационных систем, определения подходящей модели данных, обоснования рациональной схемы построения базы данных, организации запросов к хранимым данным и ряда других моментов во многом зависит эффективность функционирования разрабатываемых систем. Все это требует осознанного применения теоретических положений и инструментальных средств разработки баз данных и информационных систем.

Цель учебного пособия заключается в систематическом изложении теоретических основ построения баз данных, возможностей современных систем управления баз данных, технологии применения их для разработки и использования информационных систем, в том числе в сетях Интернет и интранет.

Учебное пособие состоит из двух частей. В первой части (раздел 1 – 11) приводятся теоретические основы построения баз данных. Рассматриваются базы данных и информационные системы. Описываются основные понятия баз данных и систем управления базами данных, дается описание логической модели данных. Дается характеристика вариантов организации информационной системы с использованием различных архитектур. Приводится классификация СУБД, и описываются основные их функции. Рассматриваются варианты создания приложений и организации взаимодействия пользователей с информационными системами.

В учебном пособии рассматривается наиболее распространенная реляционная модель представления данных, дается определение реляционной модели и характеристика ее элементов. Описываются операции индексирования, связывания таблиц и контроль целостности связей. Рассматриваются теоретические основы построения языков запросов: реляционная алгебра и реляционное исчисление; а также дается характеристика языков QBE и SQL, формат операторов.

Во второй части (раздел 12 – 21) учебного пособия рассматриваются основные возможности работы с системой управления базами данных Microsoft Access. Дано описание основных элементов СУБД, таких как таблицы, формы,

отчеты, запросы, макросы. Описаны механизмы организации работы сетевой базы данных, способах защиты базы данных. Даются практические рекомендации по проектированию и обслуживанию базы данных в MS Access.

В результате изучения изложенного в учебном пособии материала ожидается достижение следующих результатов:

1. Формирование у студентов общего кругозора в области современных компьютерных технологий, ознакомление с типовыми пакетами программ, обеспечивающими широкие возможности обработки специальной информации, освоение студентами приемов и методов проектирования баз данных.

2. Освоение методов проектирования и реализации реляционных систем обработки информации. Привитие навыков использования современных инструментальных средств и навыков работы с ними в среде разработки конкретных систем управления базами данных (СУБД).

Учебное пособие будет интересно как преподавателям, студентам занимающихся изучением вопросов проектирования систем управления баз данных, так и специалистам в области IT–индустрии.

1 БАЗЫ ДАННЫХ И ИНФОРМАЦИОННЫЕ СИСТЕМЫ

1.1 Основные понятия

В основе решения многих задач лежит обработка информации.

Определений термина информация на сегодняшний день много, дадим наиболее обобщенную формулировку данного понятия: **Информация** – совокупность знаний о практических данных и взаимосвязь между ними.

Основные стадии работы с информацией:

Сбор информации – это результат просмотра материалов и документов, уточнения, дополнения и формализации информации.

Накопление информации – это результат интеграции, систематизации, уточнения и учета информации в определенных системах.

Хранение информации – это результат централизации, коррекции, обновления и сбережения банков данных.

Обработка информации – это результат преобразования (сортировка, группировка, обогащение, сравнение и т.д.) в формы, удобные для работы.

Передача информации – это передача информации потребителю в режиме сигнального информирования или в соответствии с программой, указанной в запросе.

С точки зрения информатики, наиболее важными представляются следующие общие свойства: объективность, достоверность, полнота, точность, актуальность, полезность, ценность, своевременность, понятность, доступность, краткость и пр.

1. Объективность информации. Объективный – существующий вне и независимо от человеческого сознания. Информация – это отражение внешнего объективного мира. Информация объективна, если она не зависит от методов ее фиксации, чьего-либо мнения, суждения.

2. Объективную информацию можно получить с помощью исправных датчиков, измерительных приборов. Отражаясь в сознании человека, информация может искажаться (в большей или меньшей степени) в зависимости от мнения, суждения, опыта, знаний конкретного субъекта, и, таким образом, перестать быть объективной.

3. Достоверность информации. Информация достоверна, если она отражает истинное положение дел. Объективная информация всегда достоверна, но достоверная информация может быть как объективной, так и субъективной. Достоверная информация помогает принять нам правильное решение. Недостоверной информация может быть по следующим причинам:

- преднамеренное искажение (дезинформация) или непреднамеренное искажение субъективного свойства;
- искажение в результате воздействия помех («испорченный телефон») и недостаточно точных средств ее фиксации.

4. Полнота информации. Информацию можно назвать полной, если ее достаточно для понимания и принятия решений. Неполная информация может привести к ошибочному выводу или решению.

5. Точность информации определяется степенью ее близости к реальному состоянию объекта, процесса, явления и т. п.

6. Актуальность информации – важность для настоящего времени, злободневность, насущность. Только вовремя полученная информация может быть полезна.

7. Полезность (ценность) информации. Полезность может быть оценена применительно к нуждам конкретных ее потребителей и оценивается по тем задачам, которые можно решить с ее помощью.

Для облегчения обработки информации создаются информационные системы (ИС).

Автоматизированными системами называются информационные системы, в которых применяют технические средства, в частности ЭВМ. В широком понимании под определение информационной системы попадает любая система обработки информации.

Автоматизированными информационными системами принято называть совокупность информационных, экономико-математических, программных, технических средств, и предназначены для обработки информации и принятия решения.

Информационные системы классифицируются по разным признакам. Рассмотрим наиболее часто используемые способы классификации.

По масштабу информационные системы подразделяются на следующие группы:

- одиночные;
- групповые;
- корпоративные.

Одиночные информационные системы реализуются, как правило, на автономном персональном компьютере (сеть не используется). Такая система может содержать несколько простых приложений, связанных общим информационным фондом, и рассчитана на работу одного пользователя или группы пользователей, разделяющих по времени одно рабочее место.

Групповые информационные системы ориентированы на коллективное использование информации членами рабочей группы и чаще всего строятся на базе локальной вычислительной сети. При разработке таких приложений используются серверы баз данных (называемые также SQL–серверами) для рабочих групп.

Корпоративные информационные системы являются развитием систем для рабочих групп, они ориентированы на крупные компании и могут поддерживать территориально разнесенные узлы или сети. В основном они имеют иерархическую структуру из нескольких уровней. Для таких систем характерна архитектура клиент-сервер со специализацией серверов или же многоуровневая архитектура.

Классификация по сфере применения

По сфере применения информационные системы обычно подразделяются на четыре группы:

- системы обработки транзакций;
- системы принятия решений;
- информационно–справочные системы;
- офисные информационные системы.

Системы обработки транзакций, в свою очередь, по оперативности обработки данных, разделяются на пакетные информационные системы и оперативные информационные системы. В информационных системах организационного управления преобладает режим оперативной обработки транзакций – OLTP (OnLine Transaction Processing), для отражения актуального состояния предметной области в любой момент времени, а пакетная обработка занимает весьма ограниченную часть. Для систем OLTP характерен регулярный (возможно, интенсивный) поток довольно простых транзакций, играющих роль заказов, платежей, запросов и т.п. Важными требованиями для них являются:

- высокая производительность обработки транзакций;
- гарантированная доставка информации при удаленном доступе к БД по телекоммуникациям.

Системы поддержки принятия решений – DSS (Decision Support System) – представляют собой другой тип информационных систем, в которых с помощью довольно сложных запросов производится отбор и анализ данных в различных разрезах: временных, географических и по другим показателям.

Обширный класс *информационно-справочных систем* основан на гипертекстовых документах и мультимедиа. Наибольшее развитие такие информационные системы получили в сети Интернет.

Класс *офисных информационных* систем нацелен на перевод бумажных документов в электронный вид, автоматизацию делопроизводства и управление документооборотом.

Классификация по области применения:

- ИС в производстве
- ИС в образовании
- ИС в здравоохранении
- ИС в науке
- ИС в торговле, ИС социальной сфере, военное дело и любых других отраслях.

Классификация по способу организации

По способу организации групповые и корпоративные информационные системы подразделяются на следующие классы:

- системы на основе архитектуры файл-сервер;
- системы на основе архитектуры клиент-сервер;
- системы на основе многоуровневой архитектуры;
- системы на основе Интернет/ интранет – технологий.

Архитектура файл-сервер

Архитектура файл-сервер использует компьютер для функций отображения, что облегчает построение графического интерфейса. Файл-сервер только извлекает данные из файлов, так что дополнительные пользователи и приложения добавляют лишь незначительную нагрузку на центральный процессор. Каждый новый клиент добавляет вычислительную мощность к сети.

Архитектура клиент-сервер

Архитектура клиент-сервер предназначена для разрешения проблем файл-серверных приложений путем разделения компонентов приложения и размещения их там, где они будут функционировать наиболее эффективно. Особенностью архитектуры клиент-сервер является использование выделенных серверов баз данных, понимающих запросы на языке структурированных запросов SQL (Structured Query Language) и выполняющих поиск, сортировку и агрегирование информации.

Многоуровневая архитектура

Многоуровневая архитектура стала развитием архитектуры клиент-сервер и в своей классической форме состоит из трех уровней:

- нижний уровень представляет собой приложения клиентов, имеющие программный интерфейс для вызова приложения на среднем уровне;
- средний уровень представляет собой сервер приложений, на котором выполняется прикладная логика и с которого логика обработки данных вызывает операции с базой данных;
- верхний уровень представляет собой удаленный специализированный сервер базы данных, выделенный для услуг обработки данных и файловых операций (без риска использования хранимых процедур).

Подобную концепцию обработки данных пропагандируют, в частности, фирмы Oracle, Sun, Borland и др.

Интернет/интранет-технологии

В развитии технологии Интернет/интранет основной акцент пока что делается на разработке инструментальных программных средств. В то же время наблюдается отсутствие развитых средств разработки приложений, работающих с базами данных. Компромиссным решением для создания удобных и простых в использовании и сопровождении информационных систем, эффективно работающих с базами данных, стало объединение Интернет/интранет-технологии с многоуровневой архитектурой. При этом структура информационного приложения приобретает следующий вид: браузер – сервер приложений – сервер баз данных – сервер динамических страниц – web-сервер.

Благодаря интеграции Интернет/интранет–технологии и архитектуры клиент-сервер процесс внедрения и сопровождения корпоративной информационной системы существенно упрощается при сохранении достаточно высокой эффективности и простоты совместного использования информации.

По ***типу хранимых данных*** ИС делятся на фактографические и документальные.

Фактографические системы предназначены для хранения и обработки структурированных данных в виде чисел и текстов. Над такими данными можно выполнять различные операции.

В *документальных системах* информация представлена в виде документов, состоящих из наименований, описаний, рефератов и текстов. Поиск по неструктурированным данным осуществляется с использованием семантических признаков. Отобранные документы предоставляются пользователю, а обработка данных в таких системах практически не производится.

Основные понятия баз данных (БД):

Банк данных – является разновидностью ИС, в которой реализованы функции централизованного хранения и накопления обрабатываемой информации, ограниченных в одну или несколько БД.

Банк данных в общем случае состоит из следующих компонентов: БД (одной или нескольких), системы управления базами данных (СУБД), словаря данных, вычислительной системы, администраторов и обслуживающего персонала.

База данных – представляет собой совокупность специальным образом организованных данных, хранимых в памяти вычислительной системы и отображающих состояние объектов и их взаимосвязей в рассматриваемой предметной области.

Логическую структуру хранимых в БД называют моделью представления данных.

К основным моделям представления данных можно отнести следующие модели:

1. Иерархическая модель.
2. Сетевая модель.
3. Реляционная модель.
4. Постреляционная модель.
5. Многомерная модель.
6. Объектно-ориентированная модель.

Система управления базами данных (СУБД) – это комплекс языковых и программных средств, предназначенный для создания, ведения и совместного использования БД многими пользователями. Обычно СУБД различают по используемой модели данных. Так, СУБД, основанные на использовании реляционной модели данных, называют реляционными СУБД.

Одними из первых СУБД являются следующие системы: IMS (IBM, 1968 г.), IDMS (Cullinet, 1971 г.), ADABAS (Software AG, 1969 г.) и ИНЭС (ВНИИСИ АН СССР, 1976 г.). Количество современных систем управления базами данных исчисляется тысячами.

Приложение представляет собой программу или комплекс программ, обеспечивающих автоматизацию обработки информации для прикладной задачи. Нами рассматриваются приложения, использующие БД. Приложения могут создаваться в среде или вне среды СУБД – с помощью системы программирования, использующей средства доступа к БД, к примеру, Delphi или C++

Builder. Приложения, разработанные в среде СУБД, часто называют *приложениями СУБД,* а приложения, разработанные вне СУБД, – *внешними приложениями.*

Для работы с базой данных зачастую достаточно средств СУБД и не нужно использовать приложения, создание которых обычно требует программирования. Приложения разрабатывают главным образом в случаях, когда требуется сделать работу пользователей более удобной или автоматизировать рутинные операции с БД.

Словарь данных (СД) представляет собой подсистему Банка Данных, предназначенную для централизованного хранения информации о структурах данных, взаимосвязях файлов БД друг с другом, типах данных и форматах их представления, принадлежности данных пользователям, кодах защиты и разграничения доступа и т. п.

Словарь данных, иначе называемый системным каталогом, как следует из определения, является хранилищем служебной информации о данных в базе («данных о данных», или метаданных).

Функционально СД присутствует во всех банках данных, но не всегда выполняющий эти функции компонент имеет именно такое название. Чаще всего функции СД выполняются СУБД и вызываются из основного меню системы или реализуются с помощью ее утилит.

Если СД является частью БД, то его называют *интегрированным СД,* в противном случае СД является *автономным.* Автономные словари данных обычно используют не только в интересах собственно данных базы, но и в целях управления другими информационными ресурсами организаций при разработке структур баз данных на этапе проектирования, для ведения документации, управления проектами и т. д.

Стандартизация интерфейса СД привела к разработке службы словаря информационных ресурсов (Information Resource Dictionary System – IRDS). Служба IRDS имеет четыре интерфейса: графический, командный язык, экспорта/импорта и прикладных программ. Реализация IRDS представляет собой программный инструмент для унифицированного управления различными информационными ресурсами организации группами пользователей и приложениями. Введение IRDS может быть целесообразно на ранних этапах проектирования БД организации, когда необходимо отложить привязку БД к конкретной СУБД. Кроме того, с помощью служб IRDS можно переносить информацию между IRDS-совместимыми СД различных СУБД (независимо от используемой в них модели данных).

Администратор базы данных (АБД) есть лицо или группа лиц, отвечающих за выработку требований к БД, ее проектирование, создание, эффективное использование и сопровождение. В процессе эксплуатации АБД обычно следит за функционированием информационной системы, обеспечивает защиту от несанкционированного доступа, контролирует избыточность, непротиворечивость, сохранность и достоверность хранимой в БД информации. Для одно-

пользовательских информационных систем функции АБД обычно возлагаются на лиц, непосредственно работающих с приложением БД.

В вычислительной сети АБД, как правило, взаимодействует с *администратором сети*. В обязанности последнего входят контроль над функционированием аппаратно-программных средств сети, реконфигурация сети, восстановление программного обеспечения после сбоев и отказов оборудования, профилактические мероприятия и обеспечение разграничения доступа.

Вычислительная система (ВС) представляет собой совокупность взаимосвязанных и согласованно действующих ЭВМ или процессоров и других устройств, обеспечивающих автоматизацию процессов приема, обработки и выдачи информации потребителям. Поскольку основными функциями банка данных являются хранение и обработка данных, то используемая ВС, наряду с приемлемой мощностью центральных процессоров (ЦП) должна иметь достаточный объем оперативной и внешней памяти прямого доступа.

Обслуживающий персонал выполняет функции поддержания технических и программных средств в работоспособном состоянии. Он проводит профилактические, регламентные, восстановительные и другие работы по планам, а также по мере необходимости.

1.2 Модели представления данных

Иерархическая модель данных

В иерархической модели связи между данными можно описать с помощью упорядоченного графа (или дерева).

Для описания структуры (схемы) иерархической БД на некотором языке программирования используется тип данных «дерево» (рисунок 1).

Тип «дерево» схож с типами данных «структура» языков программирования ПЛ/1, С и «запись» языка Паскаль. В них допускается вложенность типов, каждый из которых находится на некотором уровне.

Тип «дерево» является составным. Он включает в себя подтипы («поддеревья»), каждый из которых, в свою очередь, является типом «дерево». Каждый из типов «дерево» состоит из одного «корневого» типа и упорядоченного набора (возможно, пустого) подчиненных типов. Каждый из элементарных типов, включенных в тип «дерево», является простым или составным типом «запись». Простая «запись» состоит из одного типа, например числового, а составная «запись» объединяет некоторую совокупность типов, например, целое, строку символов и указатель (ссылку).

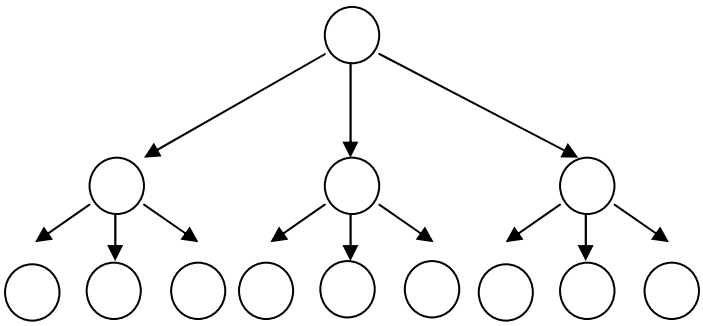


Рисунок 1 – Иерархическая модель данных

Корневым называется тип, который имеет подчиненные типы и сам не является подтипом. Подчиненный тип (подтип) является потомком по отношению к типу, который выступает для него в роли предка (родителя). Потомки одного и того же типа являются близнецами по отношению друг к другу.

В целом тип «дерево» представляет собой иерархически организованный набор типов «запись».

Иерархическая БД представляет собой упорядоченную совокупность экземпляров данных типа «дерево» (деревьев), содержащих экземпляры типа «запись» (записи). Часто отношения родства между типами переносят на отношения между самими записями. Поля записей хранят собственно числовые или символьные значения, составляющие основное содержание БД. Обход всех элементов иерархической БД обычно производится сверху вниз и слева направо.

В иерархических СУБД может использоваться терминология, отличающаяся от приведенной. Так, в системе IMS понятию «запись» соответствует термин «сегмент», а под «записью БД» понимается вся совокупность записей, относящаяся к одному экземпляру типа «дерево».

Для организации физического размещения иерархических данных в памяти ЭВМ могут использоваться следующие группы методов:

– представление линейным списком с последовательным распределением памяти (адресная арифметика, левосписковые структуры);

– представление связными линейными списками (методы, использующие указатели и справочники).

К основным операциям манипулирования иерархически организованными данными относятся следующие:

– поиск указанного экземпляра БД;

– переход от одного дерева к другому;

– переход от одной записи к другой внутри;

– вставка новой записи в указанную позицию;

– удаление текущей записи и т. д.

В соответствии с определением типа «дерево», можно заключить, что между предками и потомками автоматически поддерживается контроль целостности связей. Основное правило контроля целостности формулируется следующим образом: потомок не может существовать без родителя, а у некоторых

родителей может не быть потомков. Механизмы поддержания целостности связей между записями различных деревьев отсутствуют.

К достоинствам иерархической модели данных относятся эффективное использование памяти ЭВМ и неплохие показатели времени выполнения основных операций над данными. Иерархическая модель данных удобна для работы с иерархически упорядоченной информацией.

Недостатком иерархической модели является ее громоздкость для обработки информации с достаточно сложными логическими связями, а также сложность понимания для обычного пользователя.

На иерархической модели данных основано сравнительно ограниченное количество СУБД, в числе которых можно назвать зарубежные системы IMS, PC/Focus, Team–Up и Data Edge, а также отечественные системы Ока, ИНЭС и МИРИС.

Сетевая модель данных

Сетевая модель данных позволяет отображать разнообразные взаимосвязи элементов данных в виде произвольного графа, обобщая тем самым иерархическую модель данных (рисунок 2). Наиболее полно концепция сетевых БД впервые была изложена в Предложениях группы КОДАСИЛ (KODASYL).

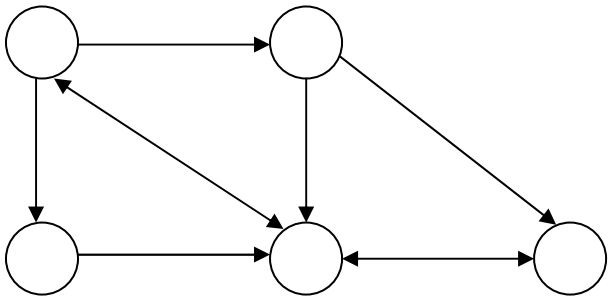


Рисунок 2 – Сетевая модель данных

Для описания схемы сетевой БД используется две группы типов: «запись» и «связь». Тип «связь» определяется для двух типов «запись»: предка и потомка. Переменные типа «связь» являются экземплярами связей.

Сетевая БД состоит из набора записей и набора соответствующих связей. На формирование связи особых ограничений не накладывается. Если в иерархических структурах запись–потомок могла иметь только одну запись–предка, то в сетевой модели данных запись–потомок может иметь произвольное число записей–предков (сводных родителей).

В различных СУБД сетевого типа для обозначения одинаковых по сути понятий зачастую используются различные термины. Например, такие как элементы и агрегаты данных, записи, наборы, области и т. д.

Физическое размещение данных в базах сетевого типа может быть организовано практически теми же методами, что и в иерархических базах данных.

К числу важнейших операций манипулирования данными баз сетевого типа можно отнести следующие:

– поиск записи в БД;

– переход от предка к первому потомку;
– переход от потомка к предку;
– создание новой записи;
– удаление текущей записи;
– обновление текущей записи;
– включение записи в связь;
– исключение записи из связи;
– изменение связей и т. д.

Достоинством сетевой модели данных является возможность эффективной реализации по показателям затрат памяти и оперативности. В сравнении с иерархической моделью сетевая модель предоставляет большие возможности в смысле допустимости образования произвольных связей.

Недостатком сетевой модели данных является высокая сложность и жесткость схемы БД, построенной на ее основе, а также сложность для понимания и выполнения обработки информации в БД обычным пользователем. Кроме того, в сетевой модели данных ослаблен контроль целостности связей вследствие допустимости установления произвольных связей между записями.

Системы на основе сетевой модели не получили широкого распространения на практике. Наиболее известными сетевыми СУБД являются следующие: IDMS, db_VistaIII, СЕТЬ, СЕТОР и КОМПАС.

Реляционная модель данных

Реляционная модель данных предложена сотрудником фирмы IBM Эдгаром Коддом и основывается на понятии отношение (relation).

Отношение представляет собой множество элементов, называемых кортежами. Наглядной формой представления отношения является привычная для человеческого восприятия двумерная таблица.

Таблица 1. Реляционная модель данных

№накладной	**№покупателей**
0373	8723
8374	8232
7364	8723

Таблица 2. Реляционная модель данных

№накладной	**Товар**	**Кол-во**
0373	Тетрадь (24 листа)	300
0373	Тетрадь (48 листов)	200
8374	Тетрадь (96 листов)	1000
8374	Блокнот (обложка пластик)	600
8374	Блокнот (обложка картон)	200
7364	Ручка черная шариковая	10000

Таблица имеет строки (записи) и столбцы (колонки). Каждая строка таблицы имеет одинаковую структуру и состоит из полей. Строкам таблицы соответствуют кортежи, а столбцам – атрибуты отношения.

С помощью одной таблицы удобно описывать простейший вид связей между данными, а именно деление одного объекта (явления, сущности, системы и проч.), информация о котором хранится в таблице, на множество подобъектов, каждому из которых соответствует строка или запись таблицы. При этом каждый из подобъектов имеет одинаковую структуру или свойства, описываемые соответствующими значениями полей записей. Например, таблица может содержать сведения о группе обучаемых, о каждом из которых известны следующие характеристики: фамилия, имя и отчество, пол, возраст и образование. Поскольку в рамках одной таблицы не удаются описать более сложные логические структуры данных из предметной области, применяют связывание таблиц.

Физическое размещение данных в реляционных базах на внешних носителях легко осуществляется с помощью обычных файлов.

Достоинство реляционной модели данных заключается в простоте, понятности и удобстве физической реализации на ЭВМ. Именно простота и понятность для пользователя явились основной причиной их широкого использования. Проблемы же эффективности обработки данных этого типа оказались технически вполне разрешимыми.

Основными недостатками реляционной модели являются следующие: отсутствие стандартных средств идентификации отдельных записей и сложность описания иерархических и сетевых связей.

Примерами зарубежных реляционных СУБД для ПЭВМ являются следующие: dBase III Plus и dBase IY (фирма Ashton–Tate), DB2 (IBM), R:BASE (Microrim), FoxPro ранних версий и FoxBase (Fox Software), Paradox и dBASE for Windows (Borland), FoxPro более поздних версий, Visual FoxPro и Access (Microsoft), Clarion (Clarion Software), Ingres (ASK Computer Systems) и Oracle (Oracle).

К отечественным СУБД реляционного типа относятся системы: ПАЛЬМА (ИК АН УССР), а также система HyTech (МИФИ).

Заметим, что последние версии реляционных СУБД имеют некоторые свойства объектно-ориентированных систем. Такие СУБД часто называют объектно-реляционными. Примером такой системы можно считать продукты Oracle 8.x. Системы предыдущих версий вплоть до Oracle 7.x считаются «чисто» реляционными.

Постреляционная модель данных

Классическая реляционная модель предполагает неделимость данных, хранящихся в полях записей таблиц. Это означает, что информация в таблице представляется в первой нормальной форме. Существует ряд случаев, когда это ограничение мешает эффективной реализации приложений.

Постреляционная модель данных представляет собой расширенную реляционную модель, снимающую ограничение неделимости данных, хранящихся в записях таблиц. Постреляционная модель данных допускает многозначные поля – поля, значения которых состоят из подзначений. Набор значений многозначных полей считается самостоятельной таблицей, встроенной в основную таблицу.

Таблица 3. Постреляционная модель данных

Накладные	**№покупателей**	**Товар**	**Кол–во**
0373	8723	Тетрадь (24 листа)	300
		Тетрадь (48 листов)	200
8374	8232	Тетрадь (96 листов)	1000
		Блокнот (обложка пластик)	600
		Блокнот (обложка картон)	200
7364	8723	Ручка черная шариковая	10000

Помимо обеспечения вложенности полей постреляционная модель поддерживает ассоциированные многозначные поля (множественные группы). Совокупность ассоциированных полей называется ассоциацией. При этом в строке первое значение одного столбца ассоциации соответствует первым значениям всех других столбцов ассоциации. Аналогичным образом связаны все вторые значения столбцов и т. д.

На длину полей и количество полей в записях таблицы не накладывается требование постоянства. Это означает, что структура данных и таблиц имеет большую гибкость.

Поскольку постреляционная модель допускает хранение в таблицах ненормализованных данных, возникает проблема обеспечения целостности и непротиворечивости данных. Эта проблема решается включением в СУБД механизмов, подобных хранимым процедурам в клиент-серверных системах.

Для описания функций контроля значений в полях имеется возможность создавать процедуры (коды конверсии и коды корреляции), автоматически вызываемые до или после обращения к данным. Коды корреляции выполняются сразу после чтения данных, перед их обработкой. Коды конверсии, наоборот, выполняются после обработки данных.

К числу СУБД, основанных на постреляционной модели данных, относятся системы Bubba, uniVers и Dasdb.

Многомерная модель данных

Многомерный подход к представлению данных в базе появился практически одновременно с реляционным. Реально работающих многомерных СУБД (МСУБД) было очень мало. С середины 90-х годов интерес к ним стал приобретать массовый характер.

Толчком послужила в 1993 году программная статья одного из основоположников реляционного подхода Э. Кодда. В ней сформулированы 12 основных требований к системам класса OLAP (OnLine Analytical Processing –

оперативная аналитическая обработка), важнейшие из которых связаны с возможностями концептуального представления и обработки многомерных данных. Многомерные системы позволяют оперативно обрабатывать информацию для проведения анализа и принятия решения.

В развитии концепций ИС можно выделить следующие два направления:

– системы оперативной (транзакционной) обработки;

– системы аналитической обработки (системы поддержки принятия решений).

Реляционные СУБД предназначались для информационных систем оперативной обработки информации и в этой области были весьма эффективны. В системах аналитической обработки они показали себя несколько неповоротливыми и недостаточно гибкими. Более эффективными здесь оказываются многомерные СУБД (МСУБД).

Многомерные СУБД являются узкоспециализированными СУБД, предназначенными для интерактивной аналитической обработки информации. Раскроем основные понятия, используемые в этих СУБД: агрегируемость, историчность и прогнозируемость данных.

Агрегируемость данных означает рассмотрение информации на различных уровнях ее обобщения. В информационных системах степень детальности представления информации для пользователя зависит от его уровня: аналитик, пользователь–оператор, управляющий, руководитель.

Историчность данных предполагает обеспечение высокого уровня статичности (неизменности) собственно данных и их взаимосвязей, а также обязательность привязки данных ко времени.

Статичность данных позволяет использовать при их обработке специализированные методы загрузки, хранения, индексации и выборки.

Временная привязка данных необходима для частого выполнения запросов, имеющих значения времени и даты в составе выборки. Необходимость упорядочения данных по времени в процессе обработки и представления, данных пользователю накладывает требования на механизмы хранения и доступа к информации. Так, для уменьшения времени обработки запросов желательно, чтобы данные всегда были отсортированы в том порядке, в котором они наиболее часто запрашиваются.

Прогнозируемость данных подразумевает задание функций прогнозирования и применение их к различным временным интервалам.

Многомерность модели данных означает не многомерность визуализации цифровых данных, а многомерное логическое представление структуры информации при описании и в операциях манипулирования данными.

По сравнению с реляционной моделью многомерная организация данных обладает более высокой наглядностью и информативностью.

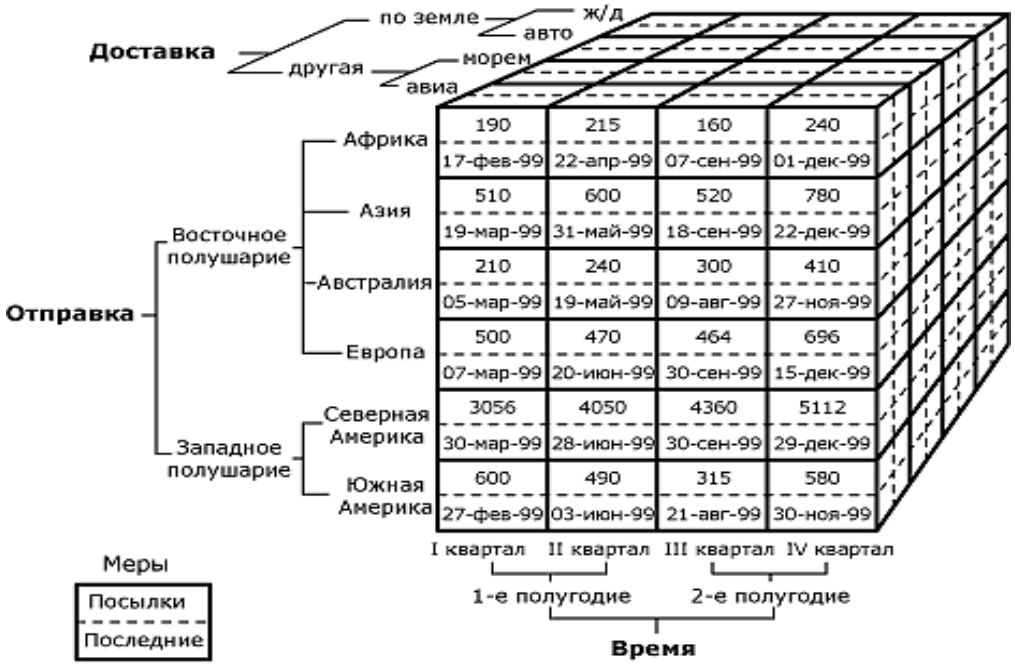


Рисунок 3 – Многомерная модель данных

Если речь идет о многомерной модели с мерностью больше двух, то не обязательно визуально информация представляется в виде многомерных объектов (трех-, четырех- и более мерных гиперкубов). Пользователю в этих случаях более удобно иметь дело с двухмерными таблицами или графиками. Данные при этом представляют собой «вырезки» (точнее, «срезы») из многомерного хранилища данных, выполненные с разной степенью детализации.

Рассмотрим основные понятия многомерных моделей данных, к числу которых относятся измерение и ячейка.

Измерение (Dimension) – это множество однотипных данных, образующих одну из граней гиперкуба. Примерами наиболее часто используемых временных измерений являются Дни, Месяцы, Кварталы и Годы. В качестве географических измерений широко употребляются Города, Районы, Регионы и Страны. В многомерной модели данных измерения играют роль индексов, служащих для идентификации конкретных значений в ячейках гиперкуба.

Ячейка (Cell) или показатель – это поле, значение которого однозначно определяется фиксированным набором измерений. Тип поля чаще всего определен как цифровой. В зависимости от того, как формируются значения некоторой ячейки, обычно она может быть переменной (значения изменяются и могут быть загружены из внешнего источника данных или сформированы программно) либо формулой (значения, подобно формульным ячейкам электронных таблиц, вычисляются по заранее заданным формулам).

В существующих МСУБД используются два основных варианта (схемы) организации данных: гиперкубическая и поликубическая.

В поликубической схеме предполагается, что в БД может быть определено несколько гиперкубов с различной размерностью и с различными измерениями в качестве граней. Примером системы, поддерживающей поликубический вариант БД, является сервер Oracle Express Server.

В случае гиперкубической схемы предполагается, что все показатели определяются одним и тем же набором измерений. Это означает, что при наличии нескольких гиперкубов БД все они имеют одинаковую размерность и совпадающие измерения. Очевидно, в некоторых случаях информация в БД может быть избыточной (если требовать обязательное заполнение ячеек).

В случае многомерной модели данных применяется ряд специальных операций, к которым относятся: формирование «среза», «вращение», агрегация и детализация.

«Срез» (Slice) представляет собой подмножество гиперкуба, полученное в результате фиксации одного или нескольких измерений. Формирование «срезов» выполняется для ограничения используемых пользователем значений, так как все значения гиперкуба практически никогда одновременно не используются.

Операция «вращение» (Rotate) применяется при двухмерном представлении данных. Суть ее заключается в изменении порядка измерений при визуальном представлении данных.

Операцию «вращение» можно обобщить и на многомерный случай, если под ней понимать процедуру изменения порядка следования измерений.

В простейшем случае, например, это может быть взаимная перестановка двух произвольных измерений.

Операции «агрегация» (Drill Up) и «детализация» (Drill Down) означают соответственно переход к более общему и к более детальному представлению информации пользователю из гиперкуба.

Основным достоинством многомерной модели данных является удобство и эффективность аналитической обработки больших объемов данных, связанных со временем. При организации обработки аналогичных данных на основе реляционной модели происходит нелинейный рост трудоемкости операций в зависимости от размерности БД и существенное увеличение затрат оперативной памяти на индексацию.

Недостатком многомерной модели данных является ее громоздкость для простейших задач обычной оперативной обработки информации.

Примерами систем, поддерживающих многомерные модели данных, являются Essbase (Arbor Software), Media Multi–matrix (Speedware), Oracle Express Server (Oracle) и Cache (InterSystems). Некоторые программные продукты, например Media/MR (Speedware), позволяют одновременно работать с многомерными и с реляционными БД. В СУБД Cache, в которой внутренней моделью данных является многомерная модель, реализованы три способа доступа к данным: прямой (па уровне узлов многомерных массивов), объектный и реляционный.

Объектно-ориентированная модель данных

В объектно-ориентированной модели при представлении данных имеется возможность идентифицировать отдельные записи базы. Между записями базы данных и функциями их обработки устанавливаются взаимосвязи с по-

мощью механизмов, подобных соответствующим средствам в объектно-ориентированных языках программирования.

Стандартизованная объектно-ориентированная модель описана в рекомендациях стандарта ODMG–93 (Object Database Management Group – группа управления объектно-ориентированными базами данных). Реализовать в полном объеме рекомендации ODMG–93 пока не удастся. Для иллюстрации ключевых идей рассмотрим несколько упрощенную модель объектно-ориентированной БД.

Структура объектно-ориентированной БД графически представима в виде дерева, узлами которого являются объекты. Свойства объектов описываются некоторым стандартным типом (например, строковым – string) или типом, конструируемым пользователем (определяется как class).

Значением свойства типа string является строка символов. Значение свойства типа class есть объект, являющийся экземпляром соответствующего класса. Каждый объект–экземпляр класса считается потомком объекта, в котором он определен как свойство. Объект–экземпляр класса принадлежит своему классу и имеет одного родителя. Родовые отношения в БД образуют связную иерархию объектов.

Пример логической структуры объектно-ориентированной БД библиотечного дела приведен на рисунке 4.

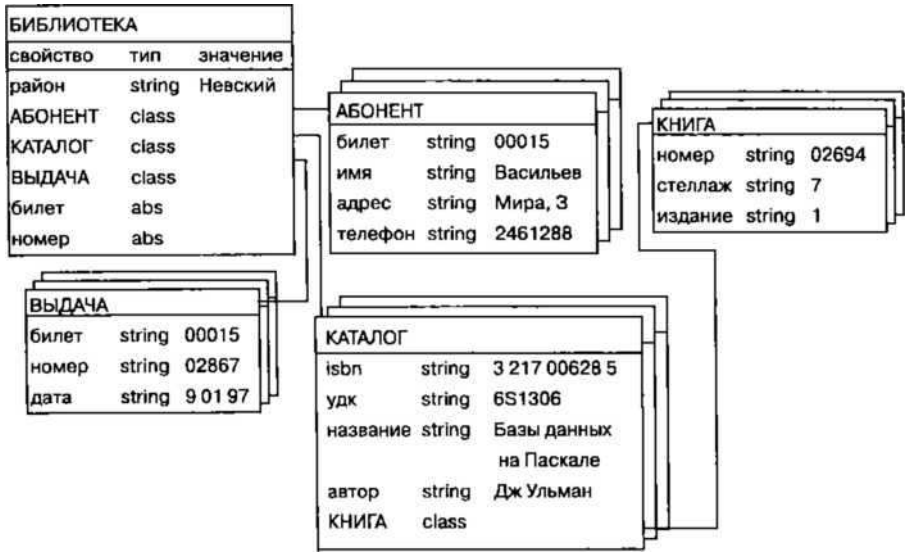


Рисунок 4 – Объектно-ориентированная модель данных

Здесь объект типа БИБЛИОТЕКА является родительским для объектов–экземпляров классов АБОНЕНТ, КАТАЛОГ и ВЫДАЧА. Различные объекты типа КНИГА могут иметь одного или разных родителей. Объекты типа КНИГА, имеющие одного и того же родителя, должны различаться по крайней мере инвентарным номером (уникален для каждого экземпляра книги), но имеют одинаковые значения свойств isbn, удк, название и автор.

Логическая структура объектно-ориентированной БД внешне похожа на структуру иерархической БД. Основное отличие между ними состоит в методах манипулирования данными.

Для выполнения действий над данными в рассматриваемой модели БД применяются логические операции, усиленные объектно-ориентированными механизмами инкапсуляции, наследования и полиморфизма. Ограниченно могут применяться операции, подобные командам SQL (например, для создания БД).

Создание и модификация БД сопровождается автоматическим формированием и последующей корректировкой индексов (индексных таблиц), содержащих информацию для быстрого поиска данных.

Рассмотрим кратко понятия инкапсуляции, наследования и полиморфизма применительно к объектно-ориентированной модели БД.

Инкапсуляция ограничивает область видимости имени свойства пределами того объекта, в котором оно определено. Так, если в объект типа КАТАЛОГ добавить свойство, задающее телефон автора книги и имеющее название телефон, то мы получим одноименные свойства у объектов АБОНЕНТ и КАТАЛОГ. Смысл такого свойства будет определяться тем объектом, в который оно инкапсулировано.

Наследование, наоборот, распространяет область видимости свойства на всех потомков объекта. Так, всем объектам типа КНИГА, являющимся потомками объекта типа КАТАЛОГ, можно приписать свойства объекта–родителя: isbn, удк, название и автор. Если необходимо расширить действие механизма наследования на объекты, не являющиеся непосредственными родственниками (например, между двумя потомками одного родителя), то в их общем предке определяется абстрактное свойство типа abs. Так, определение абстрактных свойств билет и номер в объекте БИБЛИОТЕКА приводит к наследованию этих свойств всеми дочерними объектами АБОНЕНТ, КНИГА и ВЫДАЧА. Не случайно поэтому значения свойства билет классов АБОНЕНТ и ВЫДАЧА, показанных на рисунке 4, будут одинаковыми – 00015.

Полиморфизм в объектно-ориентированных языках программирования означает способность одного и того же программного кода работать с разнотипными данными. Другими словами, он означает допустимость в объектах разных типов иметь методы (процедуры или функции) с одинаковыми именами. Во время выполнения объектной программы одни и те же методы оперируют с разными объектами в зависимости от типа аргумента. Применительно к нашей объектно-ориентированной БД полиморфизм означает, что объекты класса КНИГА, имеющие разных родителей из класса КАТАЛОГ, могут иметь разный набор свойств. Следовательно, программы работы с объектами класса КНИГА могут содержать полиморфный код.

Поиск в объектно-ориентированной БД состоит в выяснении сходства между объектом, задаваемым пользователем, и объектами, хранящимися в БД. Определяемый пользователем объект, называемый объектом – целью (свойство объекта имеет тип goal), в общем случае может представлять собой подмножество всей хранимой в БД иерархии объектов. Объект–цель, а также результат выполнения запроса могут храниться в самой базе.

Основным достоинством объектно-ориентированной модели данных в сравнении с реляционной является возможность отображения информации о сложных взаимосвязях объектов. Объектно-ориентированная модель данных позволяет идентифицировать отдельную запись базы данных и определять функции их обработки.

Недостатками объектно-ориентированной модели являются высокая понятийная сложность, неудобство обработки данных и низкая скорость выполнения запросов.

В 90-е годы существовали экспериментальные прототипы объектно-ориентированных систем управления базами данных. В настоящее время такие системы получили достаточно широкое распространение, в частности, к ним относятся следующие СУБД: G–Base (Grapael), GemStone (Servio–Logic совместно с OG1), Statice (Symbolics), ObjectStore (Object Design), Objectivity /DB (Objectivity), Versant (Versant Technologies), 02 (Ardent Software), ODB– Jupiter (научно-производственный центр «Интеллект Плюс»), а также Iris, Orion и Postgres.

1.3 Архитектура информационной системы

Эффективность функций ИС во многом зависит от ее архитектуры. В настоящее время перспективной является архитектура клиент – сервер. В достаточно распространенном варианте она предполагает наличие компьютерной сети и распределенной БД, включающей корпоративную БД (КБД) и персональные БД (ПБД). КБД размещаются на компьютере – сервере, ПБД размещаются на компьютерах сотрудников подразделений, являются клиентами корпоративной.

Сервером определенного ресурса в компьютерной сети называют компьютер (программа) управляющий этим ресурсом, клиентом – компьютер (программа), использующая этот ресурс.

В качестве **ресурса** компьютерные сети могут выступать, к примеру, БД, файловые системы, службы печати, почтовые службы и т.д.

Тип сервера определяется видом ресурса, которым он управляет. Например: сервер БД, сервер печати, почтовый сервер. Достоинством организации информационной системы по архитектуре клиент-сервер является удачное сочетание централизованного управления, обслуживания и коллективного доступа к использованию информации с индивидуальной работой пользователей над персональной информацией.

Архитектура клиент-сервер допускает различные варианты реализации.

Архитектура ФАЙЛ–СЕРВЕР

Исторически первыми появились архитектуры файл–сервер. В таких ИС по запросам пользователей файлы БД передаются на ПК, где и проводится их обработка. Недостаток такого варианта архитектуры является высокая интенсивность передачи обрабатываемых данных. Причем зачастую передают-

ся избыточные данные: вне зависимости от того сколько записей из БД требуется пользователю, файлы БД передаются целиком.

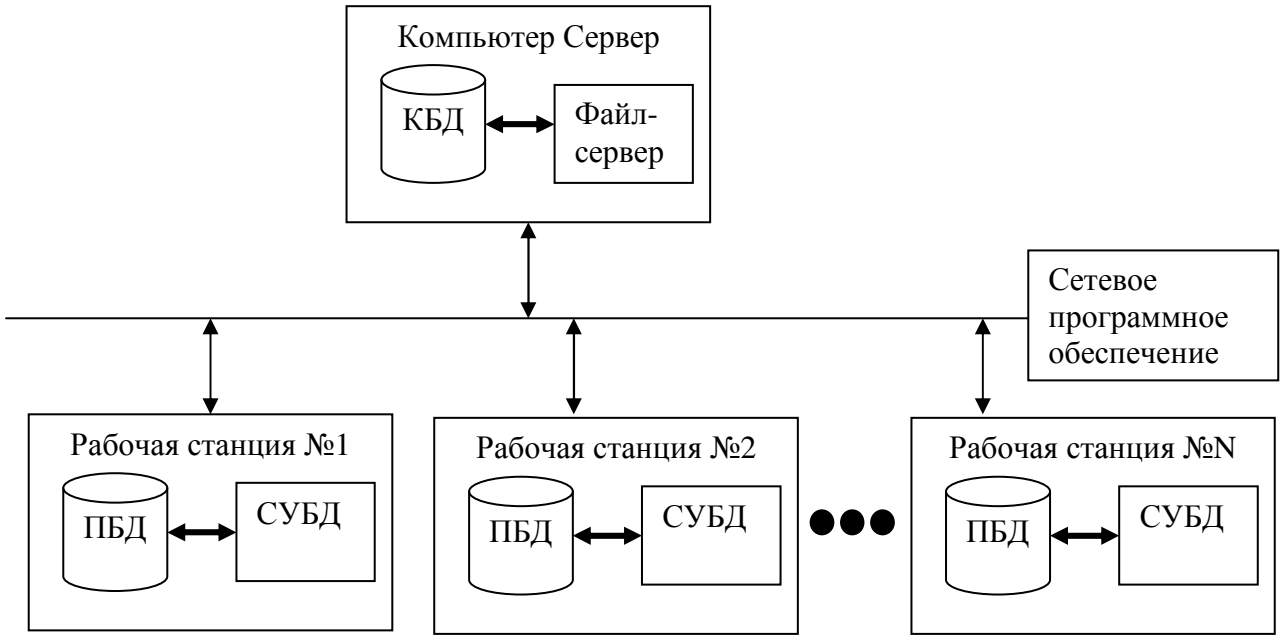


Рисунок 5 – Архитектура клиент-сервер

При такой архитектуре сервер БД обеспечивает основное выполнение основного объема обработки данных. Формируемые пользователем или приложением, запросы поступают к серверу БД в виде инструкции языка SQL (Structured Query Language) – структурный язык запросов. Сервер БД выполняет поиск и извлечение нужных данных, которые затем передаются на компьютер пользователя.

Объектами разработки в файл–серверном приложении являются компоненты приложения, определяющие логику диалога, а также логику обработки и управления данными. Разработанное приложение реализуется либо в виде законченного загрузочного модуля, либо в виде специального кода для интерпретации.

Однако такая архитектура имеет существенный недостаток: при выполнении некоторых запросов к базе данных клиенту могут передаваться большие объемы данных, загружая сеть и приводя к непредсказуемости времени реакции. Значительный сетевой трафик особенно сильно сказывается при организации удаленного доступа к базам данных на файл–сервере через низкоскоростные каналы связи. Одним из вариантов устранения данного недостатка является удаленное управление файл–серверным приложением в сети. При этом в локальной сети размещается сервер приложений, совмещенный с телекоммуникационным сервером (обычно называемым сервером доступа), в среде которого выполняются обычные файл–серверные приложения. Особенность состоит в том, что диалоговый ввод–вывод поступает от удаленных клиентов через телекоммуникации. Приложения не должны быть слишком сложными, иначе велика вероятность перегрузки сервера, или же нужна очень мощная платформа для сервера приложений.

Архитектура КЛИЕНТ-СЕРВЕР

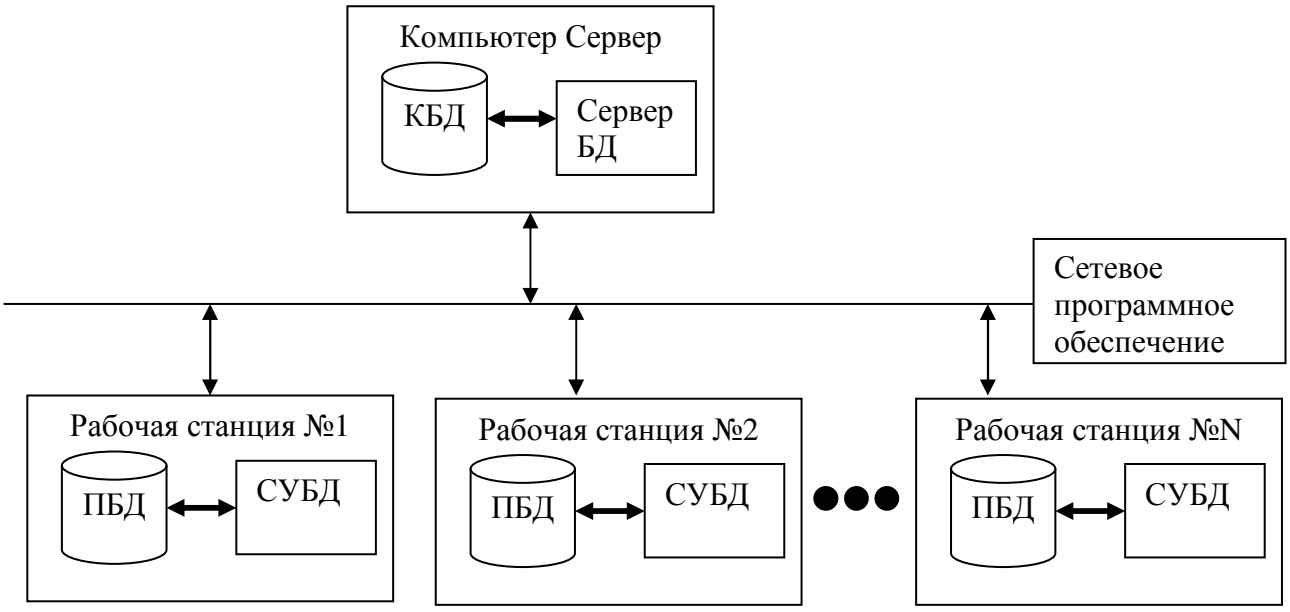


Рисунок 6 – Архитектура клиент-сервер

Отличительная черта серверов БД – наличие справочника данных, в котором записана структура БД, ограничения целостности данных, форматы и даже серверные процедуры обработки данных по вызову или по событиям в программе. Объектами разработки в таких приложениях помимо диалога и логики обработки являются, прежде всего, реляционная модель данных и связанный с ней набор SQL–операторов для типовых запросов к базе данных.

Большинство конфигураций клиент-сервер использует двухуровневую модель, в которой клиент обращается к услугам сервера. Предполагается, что диалоговые компоненты размещаются на клиенте, что позволяет обеспечить графический интерфейс. Компоненты управления данными размещаются на сервере, а диалог и логика – на клиенте. Двухуровневое определение архитектуры клиент-сервер использует именно этот вариант: приложение работает у клиента, СУБД – на сервере. Поскольку эта схема предъявляет наименьшие требования к серверу, она обладает наилучшей масштабируемостью. Однако сложные приложения, вызывающие большое взаимодействие с БД, могут жестко загрузить как клиента, так и сеть. Результаты SQL–запроса должны вернуться клиенту для обработки, потому что там находится логика принятия решения. Такая схема приводит к дополнительному усложнению администрирования приложений, разбросанных по различным клиентским узлам.

Для сокращения нагрузки на сеть и упрощения администрирования приложений компонент логики можно разместить на сервере. При этом вся логика принятия решений оформляется в виде хранимых процедур и выполняется на сервере БД. Хранимая процедура – процедура с операторами SQL для доступа к БД, вызываемая по имени с передачей требуемых параметров и выполняемая на сервере БД. Хранимые процедуры могут компилироваться, что повышает скорость их выполнения и сокращает нагрузку на сервер.

Создание архитектуры клиент-сервер возможно и на основе многотерминальной системы. В этом случае в многозадачной среде сервера приложений выполняются программы пользователей, а клиентские узлы вырождены и представлены терминалами. Подобная схема информационной системы характерна для UNIX. В настоящее время архитектура клиент-сервер получила признание и широкое распространение как способ организации приложений для рабочих групп и информационных систем корпоративного уровня. Подобная организация работы повышает эффективность выполнения приложений за счет использования возможностей сервера БД, разгрузки сети и обеспечения контроля целостности данных.

Двухуровневые схемы архитектуры клиент-сервер могут привести к некоторым проблемам в сложных информационных приложениях с множеством пользователей и запутанной логикой. Решением этих проблем может стать использование многоуровневой архитектуры.

1.4 Принцип организации функционирования информационной системы на одном компьютере

Функциональные части ИС могут размещаться на одном или несколько персональных компьютерах (ПК). Рассмотрим вариант организации ИС на одном ПК. Организация функционирования локальной ИС на одном компьютере в среде некоторой ОС возможны с помощью следующих вариантов использования программных средств:

1. «Полной» СУБД
2. Приложение и усеченной СУБД (ядро)
3. Независимого приложения.

Первый способ обычный применяется в случаях, когда в дисковой памяти компьютера помещается вся СУБД, и она часто используется для доработки приложения (рисунок 7).

Взаимодействие пользователя с СУБД происходит на прямую, через пользовательский интерфейс СУБД, либо с помощью приложения. Приложение выполняется в режиме **интерпретации** – непосредственное выполнение исходной программы в ходе просмотра ее текста.

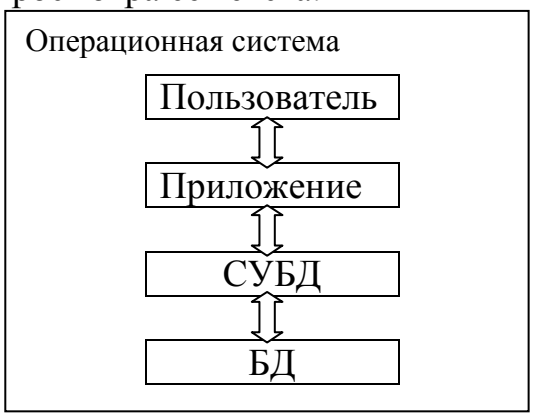


Рисунок 7 – Организация функционирования локальной ИС по схеме «полной» СУБД

Использование приложения и СУБД

Основное достоинство схемы – простота разработки и сопровождения БД и приложений при наличии развитых соответствующих средств разработки и сервисных средств (рисунок 8). Недостатком этой схемы является затраты дисковой памяти на хранение программы СУБД.

Приложение с ядром СУБД используется для достижения следующих целей:

– уменьшение любого занимаемого СУБД пространства дискового пространства и оперативной памяти;

– повышение скорости работы приложения;

– защиты приложения от модификации со стороны пользователя (обычное ядро не содержит средств разработки приложений).

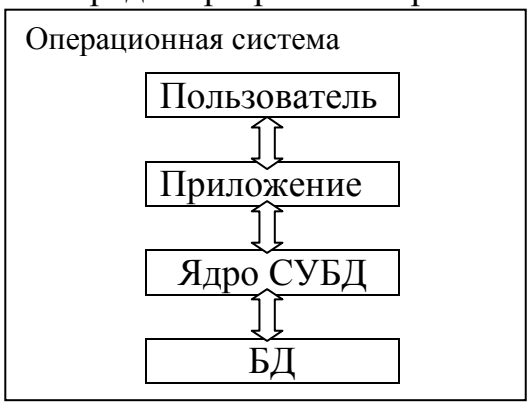


Рисунок 8 – Организация функционирования локальной ИС
по схеме приложение и усеченной СУБД

Достоинства данной схемы:

– меньшее потребление ресурсов ПК, ускорение работы приложения, возможности защиты приложения от модификации.

Недостатки:

– значение любой дисковой памяти для хранения ядра СУБД и недостаточно высокое быстродействие приложений.

При третьем способе организации ИС исходная программа компилируется – преобразуется в последовательность исполняемых машинных команд. В результате получения готовая к выполнению независимая программа, не требующая ни СУБД, ни ядра.

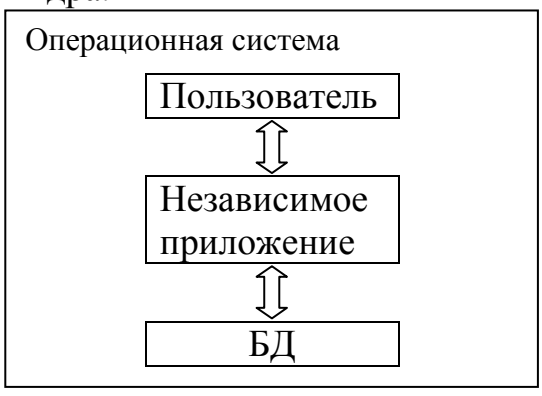


Рисунок 9 – Организация функционирования локальной ИС
по схеме независимое приложение

Достоинства данной схемы:

–экономия внешней и оперативной памяти ускорения выполнения приложения и полная защита от модификации.

Недостатки:

–трудоемкость доработки приложений и отсутствие возможных использующих стандартные средства СУБД по обслуживанию БД.

Важнейшим достоинством применения БД в ИС является обеспечение независимости данных от прикладных программ.

2 СИСТЕМЫ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ. КЛАССИФИКАЦИЯ И ОСНОВНЫЕ ФУНКЦИИ

Классификация СУБД

В общем случае под СУБД можно понимать любой программный продукт, поддерживающий процессы создания, ведения и использования БД. К СУБД относятся следующие основные виды программ:

–полнофункциональные СУБД

–серверы БД

–клиенты БД

–средства разработки программ работы с БД

Полнофункциональные СУБД (ПФСУБД) представляют собой традиционные СУБД, которые сначала появились для больших машин, затем для мини-машин и для ПЭВМ. Из числа всех СУБД современные ПФСУБД являются наиболее многочисленными и мощными по своим возможностям. К ПФСУБД относятся, например, такие пакеты, как Clarion Database Developer, DataEasc, DataFlcx, dBase IV, Microsoft Access, Microsoft FoxPro и Paradox R:BASE.

Обычно ПФСУБД имеют развитый интерфейс, позволяющий с помощью команд меню выполнять основные действия с БД: создавать и модифицировать структуры таблиц, вводить данные, формировать запросы, разрабатывать отчеты, выводить их на печать и т. п. Для создания запросов и отчетов не обязательно программирование, а удобно пользоваться языком QBE (Query By Example – формулировки запросов по образцу). Многие ПФСУБД включают средства программирования для профессиональных разработчиков.

Некоторые системы имеют в качестве вспомогательных и дополнительные средства проектирования схем БД или CASE–подсистемы. Для обеспечения доступа к другим БД или к данным SQL–серверов полнофункциональные СУБД имеют факультативные модули.

Серверы БД предназначены для организации центров обработки данных в сетях ЭВМ. Эта группа БД в настоящее время менее многочисленна, но их количество постепенно растет. Серверы БД реализуют функции управления базами данных, запрашиваемые другими (клиентскими) программами обычно с помощью операторов SQL.

Примерами серверов БД являются следующие программы: NetWare SQL (Novell), MS SQL Server (Microsoft), InterBase (Borland), SQLBase Server (Gupta), Intelligent Database (Ingress).

В роли *клиентских программ* для серверов БД в общем случае могут использоваться различные программы: ПФСУБД, электронные таблицы, текстовые процессоры, программы электронной почты и т. д. При этом элементы пары «клиент – сервер» могут принадлежать одному или разным производителям программного обеспечения.

В случае, когда клиентская и серверная части выполнены одной фирмой, естественно ожидать, что распределение функций между ними выполнено ра-

ционально. В остальных случаях обычно преследуется цель обеспечения доступа к данным «любой ценой». Примером такого соединения является случай, когда одна из полнофункциональных СУБД играет роль сервера, а вторая СУБД (другого производителя) – роль клиента. Так, для сервера БД SQL Server (Microsoft) в роли клиентских (фронтальных) программ могут выступать многие СУБД, такие как dBASE IV, Blyth Software, Paradox, DataEase, Focus, 1–2–3, MDBS III, Revelation и другие.

Средства разработки программ работы с БД могут использоваться для создания разновидностей следующих программ:

- клиентских программ;
- серверов БД и их отдельных компонентов;
- пользовательских приложений.

Программы первого и второго вида малочисленны, так как предназначены, главным образом, для системных программистов. Пакетов третьего вида гораздо больше, но меньше, чем полнофункциональных СУБД.

К средствам разработки пользовательских приложений относятся системы программирования, например Clipper, разнообразные библиотеки программ для различных языков программирования, а также пакеты автоматизации разработок (в том числе систем типа клиент-сервер). В числе наиболее распространенных можно назвать следующие инструментальные системы: Delphi и Power Builder (Borland), Visual Studio (Microsoft), SILVERRUN (Computer Advisers Inc.), S–Designor (SDP и Powersoft) и ERwin (LogicWorks).

Если говорить о конкретных системах программирования (для языков C++, C#, Visual Basic, Java и др.), то все они содержат некоторые средства доступа к наиболее широко используемым БД.

Кроме перечисленных средств, для управления данными и организации обслуживания БД используются различные дополнительные средства, например мониторы транзакций.

По характеру использования СУБД делят на *персональные* и *многопользовательские.*

Персональные СУБД обычно обеспечивают возможность создания персональных БД и недорогих приложений, работающих с ними. Персональные СУБД или разработанные с их помощью приложения зачастую могут выступать в роли клиентской части многопользовательской СУБД. К персональным СУБД, например, относятся Visual FoxPro, Paradox, Clipper, dBase, Access и др.

Многопользовательские СУБД включают в себя сервер БД и клиентскую часть и, как правило, могут работать в неоднородной вычислительной среде (с разными типами ЭВМ и операционными системами). К многопользовательским СУБД относятся, например, СУБД Oracle и Informix.

В зависимости от способа хранения и обработки БД (централизованного или децентрализованного) СУБД можно разделить на два класса: *централизованные* (или обычные) СУБД и *децентрализованные* (или распределенные) СУБД. В обычных СУБД данные хранятся в том же месте, где и программы их управления. В распределенных СУБД как программное обеспечение, так и

данные распределены по узлам сети. Распределенные СУБД могут быть *однородными* или *неоднородными*. Неоднородность СУБД может проявляться в отличии поддерживаемых моделей данных, типов данных, языков запросов, фирм–разработчиков и т. д.

Одной из разновидностей распределенных СУБД являются *мультибазовые системы*, в которых управление каждым из узлов осуществляется автономно. В мультибазовых СУБД производится такая интеграция локальных систем, при которой не требуется изменение существующих СУБД и в то же время конечным пользователям предоставляется доступ к совместно используемым данным. Пользователи локальных СУБД получают возможность управлять данными собственных узлов без централизованного контроля, который присутствует в обычных распределенных СУБД. Примером мультибазовой СУБД является система UniSQL компании Cincom Corporation.

В зависимости от возможности распараллеливания процесса обработки данных выделяют *СУБД с последовательной и параллельной обработкой* (*параллельные СУБД)*. Параллельные СУБД функционируют в многопроцессорной вычислительной системе (как правило, с множеством устройств хранения данных) или в сети компьютеров.

По используемой модели данных СУБД (как и БД), разделяют на иерархические, сетевые, реляционные, объектно-ориентированные и другие типы. Некоторые СУБД могут одновременно поддерживать несколько моделей данных.

С точки зрения пользователя, СУБД реализует ***функции*** хранения, изменения (пополнения, редактирования и удаления) и обработки информации, а также разработки и получения различных выходных документов.

Для работы с хранящейся в базе данных информацией СУБД предоставляет программам и пользователям следующие два типа *языков:*

– язык описания данных – высокоуровневый непроцедурный язык декларативного типа, предназначенный для описания логической структуры данных;

– язык манипулирования данными – совокупность конструкций, обеспечивающих выполнение основных операций по работе с данными: ввод, модификацию и выборку данных по запросам.

Названные языки в различных СУБД могут иметь отличия. Наибольшее распространение получили два стандартизованных языка: QBE (Query By Example) – язык запросов по образцу и SQL (Structured Query Language) – структурированный язык запросов. QBE в основном обладает свойствами языка *манипулирования* данными, SQL сочетает в себе свойства языков обоих типов – *описания* и *манипулирования* данными.

Перечисленные выше функции СУБД, в свою очередь, используют следующие основные функции более низкого уровня, которые назовем ***низкоуровневыми:***

– управление данными во внешней памяти;

– управление буферами оперативной памяти;

– управление транзакциями;

– ведение журнала изменений в БД;

– обеспечение целостности и безопасности БД.

Дадим краткую характеристику необходимости и особенностям реализации перечисленных функций в современных СУБД.

Реализация функции *управления данными во внешней памяти* в разных системах может различаться и на уровне управления ресурсами (используя файловые системы ОС или непосредственное управление устройствами ПЭВМ), и по логике самих алгоритмов управления данными. В основном методы и алгоритмы управления данными являются «внутренним делом» СУБД и прямого отношения к пользователю не имеют. Качество реализации этой функции наиболее сильно влияет на эффективность работы специфических ИС, например, с огромными БД, со сложными запросами, большим объемом обработки данных.

Необходимость буферизации данных и как следствие реализации функции *управления буферами* оперативной памяти обусловлено тем, что объем оперативной памяти меньше объема внешней памяти.

Буферы представляют собой области оперативной памяти, предназначенные для ускорения обмена между внешней и оперативной памятью. В буферах временно хранятся фрагменты БД, данные из которых предполагается использовать при обращении к СУ БД или планируется записать в базу после обработки.

Механизм транзакций используется в СУБД для поддержания целостности данных в базе. *Транзакцией* называется некоторая неделимая последовательность операций над данными БД, которая отслеживается СУБД от начала и до завершения. Если по каким–либо причинам (сбои и отказы оборудования, ошибки в программном обеспечении, включая приложение) транзакция остается незавершенной, то она отменяется.

В зависимости от времени, требуемого для выполнения, выделяют *обычные и продолжительные транзакции.* Продолжительные транзакции могут охватывать часы, дни и даже месяцы. Такие транзакции могут возникать в процессе проектирования и разработки сложных систем крупным коллективом людей. Кроме того, помимо обычных *плоских транзакций,* используется модель *вложенных транзакций*. В последнем случае транзакция рассматривается как набор взаимосвязанных подзадач (субтранзакций), каждая из которых также может состоять из произвольного количества субтранзакций.

Говорят, что транзакции присущи три основных свойства:

– атомарность (выполняются все входящие в транзакцию операции или ни одна);

– сериализуемость (отсутствует взаимное влияние выполняемых в одно и то же время транзакций);

– долговечность (даже крах системы не приводит к утрате результатов зафиксированной транзакции).

Примером транзакции является операция перевода денег с одного счета на другой в банковской системе. Здесь необходим, по крайней мере, двух шаговый процесс. Сначала снимают деньги с одного счета, затем добавляют их к другому счету. Если хотя бы одно из действий не выполнится успешно, результат операции окажется неверным и будет нарушен баланс между счетами.

Контроль транзакций важен в однопользовательских и в многопользовательских СУБД, где транзакции могут быть запущены параллельно. В последнем случае говорят о сериализуемости транзакций. Под *сериализацией* параллельно выполняемых транзакций понимается составление такого плана их выполнения (сериального плана), при котором суммарный эффект реализации транзакций эквивалентен эффекту их последовательного выполнения.

При параллельном выполнении смеси транзакций возможно возникновение конфликтов (блокировок), разрешение которых является функцией СУБД. При обнаружении таких случаев обычно производится «откат» путем отмены изменений, произведенных одной или несколькими транзакциями.

Ведение журнала изменений в БД (журнализация изменений) выполняется СУБД для обеспечения надежности хранения данных в базе при наличии аппаратных сбоев и отказов, а также ошибок в программном обеспечении.

Журнал СУБД – это особая БД или часть основной БД, непосредственно недоступная пользователю и используемая для записи информации обо всех изменениях базы данных. В различных СУБД в журнал могут заноситься записи, соответствующие изменениям в СУБД на разных уровнях: от минимальной внутренней операции модификации страницы внешней памяти до логической операции модификации БД (например, вставки записи, удаления столбца, изменения значения в поле) и даже транзакции.

Для эффективной реализации функции ведения журнала изменений в БД необходимо обеспечить повышенную надежность хранения и поддержания в рабочем состоянии самого журнала. Иногда для этого в системе хранят несколько копий журнала.

Обеспечение целостности БД составляет необходимое условие успешного функционирования БД, особенно для случая использования БД в сетях. *Целостность БД* есть свойство базы данных, означающее, что в ней содержится полная, непротиворечивая и адекватно отражающая предметную область информация. Поддержание целостности БД включает проверку целостности и ее восстановление в случае обнаружения противоречий в базе данных. Целостное состояние БД описывается с помощью *ограничений целостности* в виде условий, которым должны удовлетворять хранимые в базе данные. Примером таких условий может служить ограничение диапазонов возможных значений атрибутов объектов, сведения о которых хранятся в БД, или отсутствие повторяющихся записей в таблицах реляционных БД.

Обеспечение безопасности достигается в СУБД шифрованием прикладных программ, данных, защиты паролем, поддержкой уровней доступа к базе данных и к отдельным ее элементам (таблицам, формам, отчетам и т. д.).

3 СПОСОБЫ РАЗРАБОТКИ И ВЫПОЛНЕНИЯ ПРИЛОЖЕНИЙ

Современные СУБД позволяют решать широкий круг задач по работе с БД без разработки приложения. Тем не менее, есть случаи, когда целесообразно разработать приложение.

Для разработки приложений СУБД должны иметь программный интерфейс, основу которого составляют функции и/или процедуры соответствующего языка программирования.

Существующие СУБД поддерживают следующие технологии (и их комбинации) разработки приложений:

– ручное кодирование программ (Clipper, VFP, Paradox);

– создание текстов приложений с помощью генераторов (FoxApp в FoxPro, Personal Programmer в Paradox);

– –автоматическая генерация готового приложения методами визуального программирования (Delphi, Access, Paradox for Windows).

При **ручном кодировании** программисты вручную набирают текст программ приложений, после чего выполняют их отладку.

Использование **генераторов** упрощает разработку приложений, поскольку при этом можно получать программный код без ручного набора. Генераторы приложений облегчают разработку основных элементов приложений.

Средства визуального программирования приложений являются дальнейшим развитием идеи использования генераторов приложений. Приложение при этом строится из готовых «строительных» блоков с помощью удобной интегрированной среды. При необходимости пользователь может вставить свой код.

Интегрированная среда, как правило, предоставляет мощные средства создания, отладки и модификации приложений. Интегрированная среда, как правило, позволяет создавать в кратчайшие сроки эффективные и надежные приложения.

Разработанное приложение обычно состоит из одного или нескольких файлов ОС.

Если основным файлом приложения является **исполняемый файл** (например: exe – файл), то это приложение, скорее всего является независимым приложением, которое выполняется автономно от среды СУБД.

Получение независимого приложения на практике осуществляется путем компилирования исходных текстов программы, полученных различными способами: путем набора текста вручную, а также полученных с помощью генератора приложений или среды визуального программирования.

Во многих случаях приложения не могут исполняться без среды СУБД. Выполнение приложения состоит в том, что СУБД анализирует содержимое файлов приложения (в частном случае – текст исходной программы) и автоматически строит необходимые исполняемые машинные команды. Другими словами, приложение выполняется методом **интерпретаций** – непосредственное исполнение текста исходной программы в ходе просмотра ее текста. Режим

интерпретации реализован во многих современных СУБД, например Access, VFP, Paradox

Кроме того, существуют системы, использующие **промежуточный вариант** между компиляцией и интерпретацией – так называемую **псевдокомпиляции.**

В таких системах исходная программа путем компиляции преобразуется в промежуточный код (псевдокод) и записывается на диск. В этом виде ее в некоторых системах разрешается даже редактировать, но главная цель псевдокомпиляции – преобразовать программу к виду, ускоряющему процесс ее интерпретации.

Такой прием широко используется в программных работах под управлением ОС DOS.

В СУБД работает под управлением Windows, псевдокод, чаще используется, для того чтобы запретить модифицировать приложение. Это полезно для защиты от случайной или преднамеренной порчи работающей программы.

Некоторые СУБД предоставляют пользователю возможность выбора варианта разработки приложения: как интерпретируемого СУБД программного кода или как независимой программы.

Достоинством применения независимых приложений является то, что время выполнения машинной программы обычно меньше, чем при интерпретации. Такие приложения целесообразно использовать на слабых машинах и в случае установки машин «под ключ», когда необходимо закрыть приложение от доработок со стороны пользователя.

Важным достоинством приложения интерпретируемых приложений является легкость их модификации. Если готовая программа подвергается частым изменениям, то для их внесения нужна инструментальная система, т.е. СУБД или аналогичная среда. Для интерпретируемых приложений такой инструмент всегда под рукой, что очень удобно.

Другим серьезным достоинством систем с интерпретацией является то, что хорошие СУБД обычно имеют мощные средства контроля целостности данных и защиты от несанкционированного доступа, чего не скажешь о системе компилирующего типа. В последних упомянутых функции приходится программировать вручную или оставлять на совесть администраторов.

При выборе средств разработки приложения следует учитывать три основных фактора:

– ресурсы компьютера;

– особенности приложения (потребность в модификации функции программы) время на разработку, необходимого контроля доступа и поддержания целостности информации);

– цель разработки.

Для пользователя имеющего мощный компьютер и планирующего создать несколько приложений, по всей видимости, больше подойдет СУБД интерпретирующего типа.

При использовании компьютера со слабыми характеристиками лучше установить свой выбор на систему со средствами разработки независимых приложений. При этом следует иметь в виду, что малейшие изменения в приложении влечет за собой циклическое повторение этапов программирования, компиляции и отладки программы. Разница в выполнении независимого приложения и выполнения приложения в режиме интерпретации колеблется в пределах миллисекунд в пользу независимого приложения. В то же время разница во времени подготовки приложения к его использованию обычно составляет величины порядка минуты–часы в пользу систем с интерпретацией.

Схемы обмена данными при работе с БД

Пользователю любой категории (администратору БД, разработчику или обычному пользователю) для грамотного решения задач полезно представлять вычислительный процесс, происходящий в ОС при работе с БД. Рассмотрим внутренние механизмы этого процесса на примере, когда пользователь работает с « полной» версией СУБД.

При работе пользователя с БД над ее содержимым выполняет следующие основные операции: выбор, добавление, модификация (замена) и удаление данных. Рассмотрим, как происходит обмен данными между отдельным пользователем и персональной СУБД при выполнении наиболее часто используемой операции выбора данных. Обмен данными между пользователем и БД для других операций отличаются несущественно.

Схематично обмен данными при работе пользователя с БД можно представить следующим образом, где обычными стрелками обозначены связи по управлению, утолщенными – связи по информации.

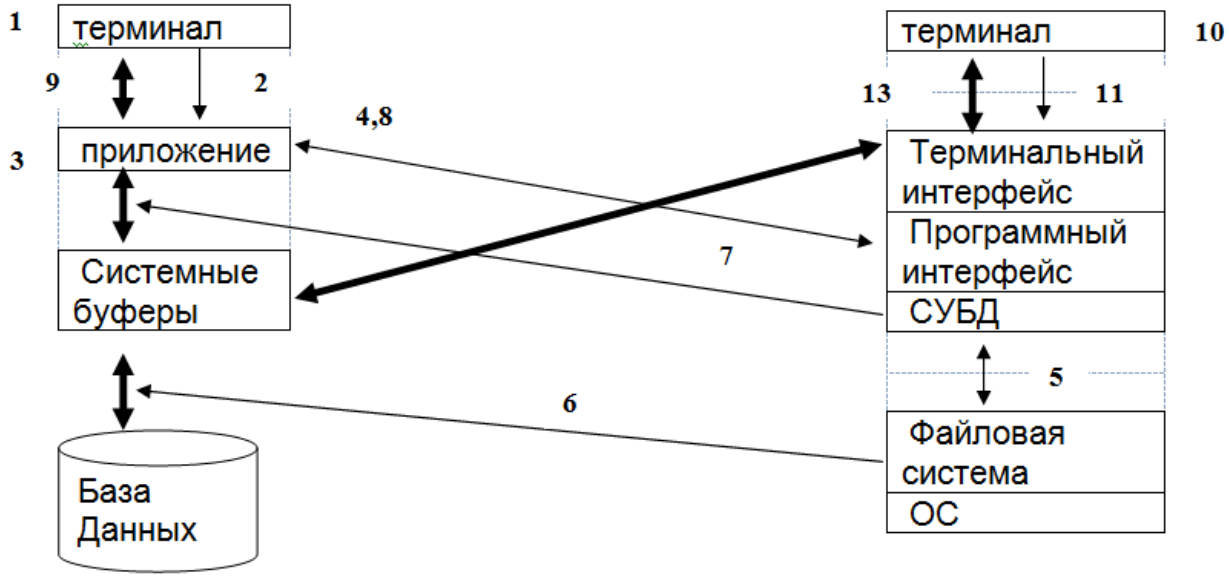


Рисунок 10 – Схема обмена данными при работе с БД

Цикл взаимодействия пользователя с БД с помощью приложения можно разделить на основные этапы:

1.Пользователь терминала (1) в процессе диалога с приложением формирует запрос (2) на некоторые данные из БД

2. Приложение (3) на программном уровне средствами языка манипулирования данными (SQL, QBE) формулирует запрос (4), с которым обращается к СУБД.

3. Используя свои системные управляющие блоки и таблицы, СУБД с помощью словаря данных определяет место положение требуемых данных и обращается (5) за ними к ОС.

4 . Программы методов доступа файловой системы ОС считывают (6) из внешней памяти искомые данные и помещают их в системные буфера СУБД

5. Преобразуя полученные данные е требуемому формату, СУБД пересылает их (7) в соответствующую область программы и сигнализирует (8) о завершении операции каким либо образом (например кодом возврата).

6. Результаты выбора данных из базы приложение (3) отображает (9) на терминале пользователя (1).

В случае работы пользователя в диалоговом режиме с СУБД (без приложения) цикл взаимодействия упрощается. Его можно (цикл) представить следующим образом:

1. Пользователь терминала (10) формирует на языке запросов СУБД, по связи (11) требования на выборку некоторых данных из базы.

2. СУБД определяет местоположение требуемых данных и обращается (5) за ними к ОС, которая считывает (6) из внешней памяти искомые данные и помещает их в системные буферы СУБД.

3. Информация из системных буферов преобразуется (12) к требуемому формату, после чего отображается (13) на терминале пользователя (10)

Эта схема описывает работу одного пользователя.

Если ПК и ОС поддерживающий многопользовательский режим работы, то в такой вычислительной системе может функционировать многопользовательская СУБД, т.е. одновременная работа нескольких пользователей или приложений.

Иногда к вычислительной системе подключается так называемый удаленный пользователь на некотором удалении от ЭВМ и соединенный с ней при помощи, какой либо передающей среды.

В многопользовательской СУБД при выполнении различных операций параллельно протекают процессы подобные схеме.

При обслуживании нескольких параллельных источников запросов (от пользователя и приложения) СУБД так планирует использование своих ресурсов и ресурсов ЭВМ, чтобы обеспечить независимое или почти независимое выполнение операции, порождаемых запросами.

Многопользовательская СУБД часто применяются на больших и средних ЭВМ, где основным режимом использования ресурсов является коллективный доступ.

4 РЕЛЯЦИОННАЯ МОДЕЛЬ ДАННЫХ

Реляционная модель данных (РМД), некоторой предметной области представляет собой набор отношений, изменяющихся во времени. При создании информационной системы совокупность отношений позволяет хранить данные об объектах предметной области и моделировать связи между ними.

Элементы РМД и формы их представления приведены в таблице 4.

Таблица 4. Элементы реляционной модели данных

Элементы РМД	**Форма представления**
Отношение	Таблица
Схемы отношения	Строка заголовков столбцов таблицы (заголовок таблицы)
Кортеж	Строка табл.
Сущность	Описание свойств объекта
Атрибут	Заголовок столбца таблицы
Домен	Множество допустимых значений атрибута
Значение атрибута	Значение поля в записи
Первичный ключ	Один или несколько атрибутов
Тип данных	Тип значений элементов таблицы

Отношение – является важнейшим понятием и представляет собой двумерную таблицу, содержащую некоторые данные (таблица 5).

Сущность – есть объект любой природы, данные о котором хранятся в БД. Данные о сущности хранятся в отношении.

Атрибуты – представляют собой свойства, характеризующие сущность. В структуре таблицы каждый атрибут именуется и ему соответствует заголовок некоторого столбца таблицы.

Математически отношение можно описать следующим образом.

Пусть даны n множеств D1,D2,D3,…,Dn, тогда отношение R есть множество упорядоченных кортежей <d1, d2, d3 ,..., dn>,где dk ∈ Dk, dk – атрибут, а Dk –домен отношения R.

Таблица 5. Пример представления отношения СОТРУДНИК

ФИО	**Отдел**	**Должность**	**Дата рождения**
Иванов И.И.	002	Начальник	27.09.1949
Петров П.П.	001	Заместитель	15.05.1955
Сидоров И.П.	002	Инженер	31.05.1972

В общем случае порядок кортежей в отношении, как и в любом множестве, не определен. Однако в реляционных СУБД для удобства картежи упорядочивают. Чаще всего для этого выбирают некоторый атрибут, по которому система автоматически сортирует кортежи по возрастанию или убыванию. Если

пользователь не назначает атрибута упорядочения, система автоматически присваивает номер кортежам в порядке их ввода.

Формально, если переставить атрибуты в отношении, то получится новое отношение. Однако в реляционных БД перестановка атрибутов не приводит к образованию нового отношения.

Домен – представляет собой множество всех возможных значений определенного атрибута отношения.

Отношение СОТРУДНИК содержит 4 домена:

ДОМЕН 1: содержит фамилии всех сотрудников

ДОМЕН 2: номера всех отделов фирмы

ДОМЕН 3: название всех должностей

ДОМЕН 4: даты рождения всех сотрудников

Каждый домен образует значения одного тира данных, например, числовые или символьные и т.д.

Отношение СОТРУДНИК содержит 3 кортежа. Кортеж рассматриваемого отношения состоит их 4–х элементов, каждый из которых выбирается из соответствующего домена. Каждому кортежу соответствует строка таблицы.

Схема отношений (заголовок отношения) представляет собой список имен атрибутов. Например, для приведенного примера схемы отношения СОТРУДНИК имеет вид. С (ФИО, Отдел, Должность, Дату рождения). Множество кортежей отношения часто называют содержимым (телом) отношения.

Первичным ключом (ключом отношения, ключевым атрибутом) называется атрибут отношения, однозначно идентифицирующий каждый из его кортежей. Например, в отношении СОТРУДНИК ключевым является ФИО. Ключ может быть составным (сложным) т.е. состоять их нескольких атрибутов.

Каждое отношение обязательно имеет комбинацию атрибутов, которая может служить ключом, Ее существование гарантируется тем, что отношение это множество, которое не содержит одинаковых элементов кортежей. Т.е. в отношении нет повторяющихся кортежей. А это значит, что, по крайней мере, вся совокупность атрибутов обладает свойством однозначной идентификации кортежей отношения. Во многих СУБД допускается, создавать отношения не определяя ключи.

Возможны случаи, когда отношение имеет несколько комбинаций атрибутов, каждая из которых однозначно определяет все кортежи отношения. Все эти комбинации атрибутов являются **возможными ключами** отношения. Любой из возможных ключей может быть выбран как первичный.

Если выбранный ключ состоит из минимально необходимого набора атрибутов, говорят что, он является не избыточным.

Ключи обычно используют для достижения следующий целей:

1.Исключения дублирования значений в ключевых атрибутах (остальные атрибуты в расчет не принимаются).

2.Упорядочения кортежей. Возможно упорядочение по возрастанию или убыванию значений всех ключевых атрибутов, а так же смещения упорядочения (по одним – возрастание, а по другим – убывание).

3. Ускорение работы с кортежами отношения

4. Организация связывания таблиц.

Пусть в отношении R1 имеется неключевой атрибут A, значения которого являются значениями ключевого атрибута **B** другого отношения R, тогда говорят, что атрибут A отношения R1 есть ВНЕШНИЙ КЛЮЧ.

С помощью внешних ключей устанавливаются связи между отношениями.

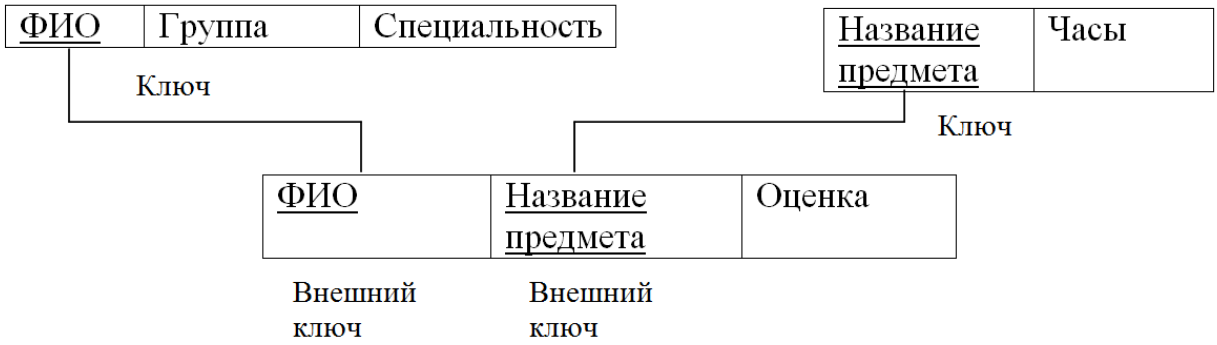


Рисунок 11 – Пример связи отношений

В связующем отношении атрибуты ФИО и название предмета образуют составной ключ. Эти атрибуты представляют собой внешние ключи, являющиеся первичными ключами других отношений. Реляционная модель накладывает на внешние ключи ограничения для обеспечения целостности данных, которая называется – ССЫЛОЧНАЯ ЦЕЛОСТНОСТЬ. Это означает, что каждому значению внешнего ключа должны соответствовать строки в связываемых отношениях.

Поскольку не всякой таблице можно поставить в соответствие отношение, приведем условия, выполнение которых позволяет таблицу считать отношением.

1. Все строки таблицы должны быть уникальными, т.е. не может быть строк с одинаковыми первичными ключами.

2. Имена столбцов таблицы должны быть различны, а значения их простыми, т.е. недопустима группа значений в одном столбце одной строки.

3. Все строки одной таблицы должны иметь одну структуру, соответственно именам и типам столбцов.

4. Порядок размещения строк в таблице может быть произвольным.

Наиболее часто таблицы с отношением размещается в одной файле.

В общем случае можно считать, БД включенной в одну или несколько таблиц объединенных смысловым содержанием, а также процедурами контроля целостности и обработки информации в интересах решения некоторой прикладной задачи.

Если задаваемое таблицей отношение имеет ключ, то считается, что таблица тоже имеет ключ, и ее называют ключевой или таблицей с ключевыми полями.

У большинства СУБД файл таблицы включает управляющую часть (описание типов полей, имени полей и др. информация) и область размещения

записей. К отношениям можно применять системные операции позволяющие получать одни отношения из других. Например: результатом запроса может быть новое отношение, вычисляемое на основе имеющихся отношений. Поэтому можно разделить обрабатывающие данные на хранимую и вычисляемую части. Основной единицей обработки данных в реляционных БД является отношения, а не отдельные его кортежи (записи).

Индексирование

Как отмечалось выше, определение ключа, для таблиц означает автоматическую сортировку записей, контроль отсутствия повторений значений в ключевых полях записей и повышение скорости выполнения операций поиска в таблице. Для реализации этих функций в СУБД применяют **индексирование**.

Термин « индекс» тесно связан с понятием « ключ», хотя между ними есть и некоторое отличие.

Под индексом понимают средство ускорения операции поиска записей в таблице, а, следовательно, и других операций, использующих поиск: извлечение, модификация, сортировка и т.д.

Таблицу, для которой используется индекс, называют индексированной. Индекс выполняет роль оглавления таблицы, просмотр которого предшествует обращению к записям таблицы.

Варианты решения проблемы организации физического доступа к информации зависят в основном от следующих факторов:

- вида содержимого в поле ключа записей индексного файла.
- типа используемых ссылок (указателей) на запись основной таблицы.
- методы поиска нужных записей.

В поле ключа индексного файла можно хранить значения ключевых полей индексируемой таблицы либо свертку ключа (так называемый ХЕШ–код).

Для организации ссылки на запись таблицы могут использоваться три типа адресов:

- абсолютный (действительный)
- относительный
- символический идентификатор

На практике чаще всего используются два метода поиска:

- последовательный
- бинарный (основан на делении интервала поиска пополам)

Существуют различные схемы индексации таблиц. Рассмотрим одно уровневую и двухуровневую систему.

При одноуровневой схеме в индексной файле хранятся короткие записи, имеющие два поля: после содержимого старшего ключа (ХЕШ – код ключа) адресуемого блока и поле адреса начала этого блока. В каждом блоке записи располагаются в порядке возрастания значения ключа. Старшим ключом каждого блока является ключ его последней записи (рисунок 12).

Если в индексном файле хранятся ХЕШ–коды ключевых полей индексированных таблиц, по алгоритмам поиска нужной записи (с указанным ключом) в таблице включает в себя следующие 3 этапа:

1. Образование свертки значения ключевого поля искомой записи
2. Поиск в индексном файле записи о блоке, значение первого поля, которого больше полученной свертки.
3. Последовательный просмотр записей блока до совпадения сверток (ключей) искомой записи и записи блока файла. В случае коллизий сверток ищется запись, значение ключа которой совпадает со значением ключа искомой записи.

Основным недостатком одноуровневой схемы является то, что ключи (свертки) записей хранятся вместе с записями. Это приводит к увеличению времени поиска записей из за большой длины просмотра.

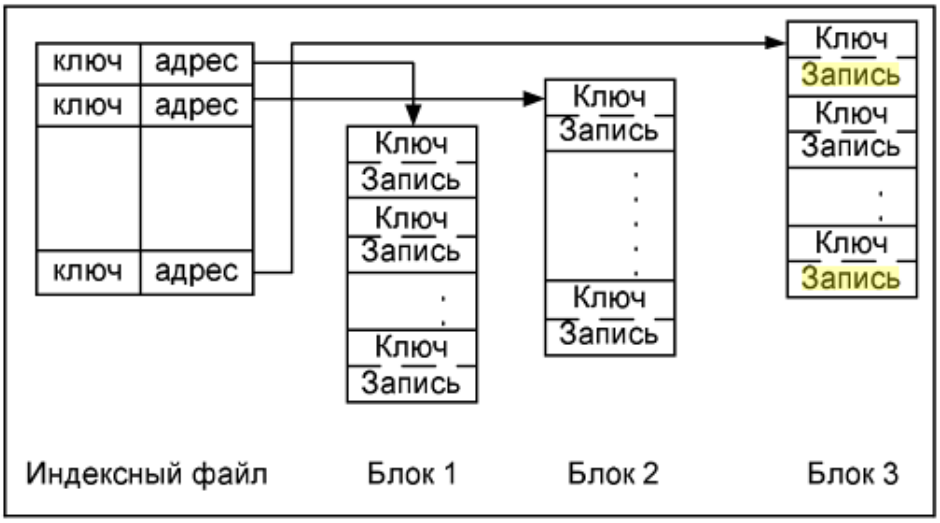


Рисунок 12 – Одноуровневая схема индексации

В ряде случаев оказывается более рациональной, в ней ключи (свертки) записей отделены от содержимого записей (рисунок 13) В этой схеме индекс основной таблицы распределен по совокупности файлов: одному файлу главного индекса и множеству файлов с блоками ключей.

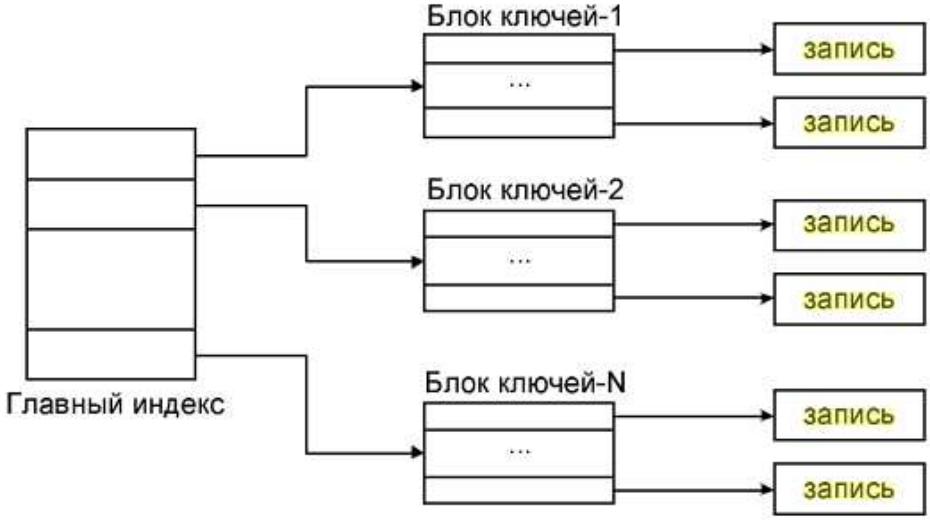


Рисунок 13 – Двухуровневая схема индексации

На практике для создания индекса для некоторой таблицы БД пользователь указывает поле таблицы, которое требует индексации. Ключевые поля таблицы во многих СУБД, как правило, индексируются автоматически.

Индексы, создаваемые пользователем для неключевых полей, иногда называют ***вторичными (пользовательскими) индексами.*** Введение таких индексов не изменяет физического расположения записей таблицы, но влияет на последовательность просмотра записей. Индексные файлы, создаваемые для поддержания вторичных индексов таблицы, обычно называются *файлами вторичных индексов.*

Связь вторичного индекса с элементами данных базы может быть установлена различными способами. Один из них – использование вторичного индекса как входа для получения первичного ключа, по которому затем с использованием первичного индекса производится поиск необходимых записей. Некоторыми СУБД, например Access, деление индексов на первичные и вторичные не производится. В этом случае используются автоматически создаваемые индексы и индексы, определяемые пользователем по любому из неключевых нолей.

Главная причина повышения скорости выполнения различных операций индексированных таблицах состоит в том, что основная часть работы производится с небольшими индексными файлами, а не с самими таблицами. Наибольший эффект повышения производительности работы с индексированными таблицами достигается для значительных но объему таблиц. Индексирование требует небольшого дополнительного места на диске и незначительных затрат процессора на изменение индексов и процессе работы. Индексы в общем случае могут изменяться перед выполнением запросов к БД, после выполнения запросов к БД, по специальным командам пользователя или программным вызовам приложений.

5 СВЯЗЫВАНИЕ ТАБЛИЦ

При проектировании реальных БД информацию обычно размещают в нескольких таблицах. Таблицы при этом связаны семантикой информации. В Реляционных СУБД для указания связей таблиц проводят операцию их связывания.

Укажем достоинства, обеспечиваемые в результате связывания таблиц. Многие СУБД при связывании таблиц выполняют контроль целостности вводимых в БД в соответствии с установленными связями. В конечном итоге это повышает достоверность хранимой в БД информации.

Кроме того, установление связей между таблицами облегчает доступ к данным. Связывание таблиц при выполнении таких операций как поиск, просмотр, редактирование, выборка и подготовка отчетов обычно обеспечивает возможность обращения к производственным полям связанных записей. Это уменьшает количество явных обращений к таблицам данных и число манипуляции в каждой из них.

Основные виды связи таблиц

Между таблицами могут устанавливаться бинарные (между двумя таблицами), тернарные (между 3-мя), и в общем случае n-нарные связи. Рассмотрим наиболее часто встречающиеся бинарные связи.

При связывании двух таблиц выделяют основную и дополнительную (подчиненную) таблицы. Логическое связывание таблиц производится с помощью ключа связи.

Ключ связи, по аналогии с обычным ключом таблицы, состоит из одного или нескольких полей, которые в данном случае называют полями связи (ПС).

Суть связывания состоит в установлении соответствия полей связи основной и дополнительной таблиц. Поля связи основной таблицы могут быть обычными и ключевыми. В качестве полей связи подчиненной таблицы чаще всего используют ключевые поля.

В зависимости от того как определены поля связей основной и дополнительной таблицы (как соотношения ключевые поля с полями связи), между двумя таблицами в общем случае могут быть следующие 4 основные вида связи (таблица 6):

- один–один (1:1)
- один–много (1:M)
- много–один (M:1)
- много–много (M:M или M:N)

Таблица 6. Характеристика видов связей таблиц

Характеристика полей связи по видам	**1:1**	**1:M**	**M:1**	**M:M**
Поля связи основной таблицы	Является ключом	Является ключом	Не является ключом	Не является ключом
Поля связи дополнительной таблицы	Является ключом	Не является ключом	Является ключом	Не является ключом

Дадим характеристику названным видам связей между двумя таблицами и приведем примеры их использования.

Связь вида 1:1

Связь вида **1**:1 образуется в случае, когда все поля связи основной таблицы и дополнительной являются ключевыми. Поскольку в ключевых полях обеих таблиц не повторяются, обеспечивается взаимно-однозначное соответствие записей из этих таблиц.

Сами таблицы, по сути, становятся равноправными.

ПРИМЕР 1. Пусть имеются основная «О1» и дополнительная «Д1» таблицы. Ключевые поля обозначим « *», используемые для связи поля обозначим «+» (таблица 7).

Таблица 7. Пример связи вида 1:1

Таблица О1

Поле 11 *+	**Поле 12**
а	10
б	40
в	3
г	15

Таблица Д1

Поле 21 *+	**Поле 22**
а	стол
в	книга

В приведенных таблицах установлена связь между записью (а ,10) табл. О1 и записью (а, стол) таблицы Д1. Основанием этого является совпадение значений в полях связи. Аналогичная связь существует и между записями (в,3) и (в, книга) этих же таблиц. В таблицах записи отсортированы по значениям в ключевых полях.

Сопоставление записей двух таблиц по существу означает образование новых « виртуальных записей» (псевдозаписей). Так, первую пару записей логически можно считать новой псевдозаписью вида (а,10,стол), а вторую пару – псевдозаписью вида (в,3,книга).

На практике связи вида 1:1 используются сравнительно редко, так как хранимую в двух таблицах информацию легко объединить в одну таблицу, которая займет гораздо меньше места в памяти ЭВМ. Возможны случаи, когда удобнее иметь не одну, а две и более таблицы. Причинами этого может быть необходимость ускорить обработку, повысить удобство работы нескольких пользователей с общей информацией, обеспечить более высокую степень защиты информации и т.д. Приведем пример, иллюстрирующий последнюю из приведенных причин.

ПРИМЕР 2

Пусть имеются сведения о выполняемых в некоторой организации НИР. Эти данные включают в себя следующую информацию по каждой из работ:

Тему (девиз или наименование работ), шифр (код), даты начала и завершения работы, количество этапов, головного исполнителя и др. дополнительную информацию. Все работы имеют гриф « Для служебного использования» или «секретно».

В такой ситуации всю информацию целесообразно хранить в двух таблицах: в одной из них всю секретную (например: шифр, наименование и

головной исполнитель), а в другой всю оставшуюся не секретную информацию. Обе таблицы можно связать по шифру работы. Первую из таблиц целесообразно защитить от несанкционированного доступа.

СВЯЗЬ ВИДА 1:М

Связь 1:М имеет место в случае, когда одной записи основной таблицы соответствует несколько записей вспомогательной таблицы.

ПРИМЕР 3.

Пусть имеются две связанные таблицы О2 и Д2. В таблице О2 содержится информация о видах мультимедиа–устройств ПЭВМ, а в таблице Д2 – сведения о фирмах – производителях этих устройств, а также о наличии на складе хотя бы одного устройства (таблица 8).

Таблица 8. Пример связи вида 1:М

О2

Код	**Вид устройства**
A	CD – ROM
B	CD – Recorder
C	Sound Blaster

Д2

Код	**Фирма производитель**	**Наличие**
A	Acer	10
A	Sony	15
A	Samsung	20
A	NEC	0
A	Mitsumi	15
B	LG	10
B	Sony	18
B	BENQ	4
B	ASUS	25
C	Creative Labs	40
C	Genius	30

Таблица Д2 имеет 2 ключевых поля, т.к. одна и та же фирма может производить устройства различных видов. Например, фирма Sony производит продукцию CD – ROM, CD – R. Сопоставление записей обеих табл. По полю «код» порождает псевдозаписи вида (A, CD–ROM,Acer,10),(A,CD–ROM, Sony, 15) и т.д. Если свести все данные (псевдозаписи) в одну таблицу, то получим полную информацию обо всех видах мультимедийных устройств ПЭВМ, фирмах их производителей, а так же сведения о наличии конкретных видов устройств на складе.

СВЯЗЬ ВИДА М:1

Связь М:1 имеет место в случае, когда одной или нескольким записям основной таблицы ставится в соответствие одна запись дополнительной таблицы.

ПРИМЕР 4

Рассмотрим связь таблиц **характеристики и приобретение.**

В основной таблице О1 информация о названии деталей, видах материалов из которых изготовлены детали, и марка материала. В дополнительной таблице Д1 содержатся сведения о названии деталей, даты покупки, количество (таблица 9).

Таблица 9. Пример связи вида М:1

О3

Название	**Материал**	**Марка**
Д1	Чугун	М1
Д1	Чугун	М2
Д2	Сталь	М1
Д2	Сталь	М3
Д3	Алюминий	М4
Д4	Чугун	М1
Д4	чугун	М2

Д3

Название	**Дата–покупки**	**Стоим.1 ед.**
Д1	21.01.15	900
Д2	01.04.15	200
Д3	31.05.15	500
Д4	11.09.15	1000

Связывание этих таблиц обеспечивает такое установление соответствия между записями, которое эквивалентно образованию следующих псевдозаписей (д1,чугун,м1,21.01.13,900)…(д4,чугун,м2,11.09.13,1000).

Полученная псевдо таблица может быть полезна при планировании или принятии управленческих решений. Отметим, что основная таблица не имеет ключей и в ней возможно повторение записей. Если дополнительную таблицу сделать основной, а основную дополнительной то получим связь типа 1:М. Аналогично предшествующему примеру (Оборудование, Фирма, Наличие) 1:М=>М:1

Отсюда следует, что вид связи (1:М или М:1) зависят от того, какая таблица является главной, а какая дополнительной.

Связь вида М:М

Самый общий вид связи М:М возникает в случаях, когда нескольким записям основной таблицы соответствует несколько записей данной таблицы

Пример 5.

Пусть в основной таблице О4 содержится информация о том, на каких станках могут работать рабочие некоторой бригады. Д4 содержит сведения о том, кто из бригады ремонтников какие станки обслуживает (таблица 10).

Таблица 10. Пример связи вида М:М

О4

Работает	**На станке**
Иванов А.В.	Станок 1
Иванов А.В.	Станок 2
Петров Н.Г.	Станок 1
Петров Н.Г.	Станок 3
Сидоров В.К.	Станок 2

Д4

Обслуживает	**Станок**
Голубев Б. С.	Станок 1
Голубев Б.С.	Станок 3
Зыков А.Ф.	Станок 2
Зыков А.Ф.	Станок 3

Первой и 3-ей записям О4 соответствует 1–я запись таблицы Д4 4-ой записи Д4 соответствует 2-ая и 4-ая запись Д4.

Исходя из определений полей связи этих таблиц можно составить новую таблицу с именем « О4+ Д4», записями которой будут псевдозаписи. Записям полученной таблицы можно придать смысл возможных смен, составляемых при планировании работы (таблица 11). Для удобства поля новой таблицы переименованы (кстати такую операцию предлагают многие из современных СУБД).

Приведенную таблицу можно использовать для получения ответа на вопрос: «Кто обслуживает станки, на которых работает?»

Очевидно, аналогично 1:1, связь М:М не устанавливает подчиненности таблиц. Для проверки этого можно поменять таблицы местами и выполнить объединение информации. Результаты таблиц могут отличаться лишь порядком следования полей или записей.

Таблица 11. Пример связи вида М:М

О4+ Д4

работает	**станок**	**обслуживает**
Иванов А.В.	Станок 1	Голубев Б. С.
Иванов А.В.	Станок 2	Зыков А.Ф.
Петров Н.Г.	Станок 1	Голубев Б. С.
Петров Н.Г.	Станок 3	Голубев Б. С.
Петров Н.Г.	Станок 3	Зыков А.Ф.
Сидоров В.К.	Станок 2	Зыков А.Ф.

Примечание.

На практике в связь обычно вовлекается сразу несколько таблиц. При этом одна из таблиц может иметь различного рода связи с другими таблицами. В случаях, когда связывание таблицы, в свою очередь, имеют связи с другими таблицами, образуются иерархия или дерево связей.

Контроль целостности связей

Из перечисленных видов связи наиболее широко используется связь вида 1:М. Связь вида 1:1 можно считать частным случаем связи 1:М, когда одной записи главной таблицы соответствует одна запись вспомогательной таблицы. Связь М:1 , по сути, является «зеркальным отображением» связи 1:М. Остальной вид связи М:М характеризуется как слабый вид связи или даже как отсутствие связей.

Следовательно, связь 1:М образуется, когда одна запись главной таблицы (главная, родительская запись) оказывается связанной с несколькими записями дополнительно (дополнительные, подчиненные записи) и имеет место схема (рисунок 14).

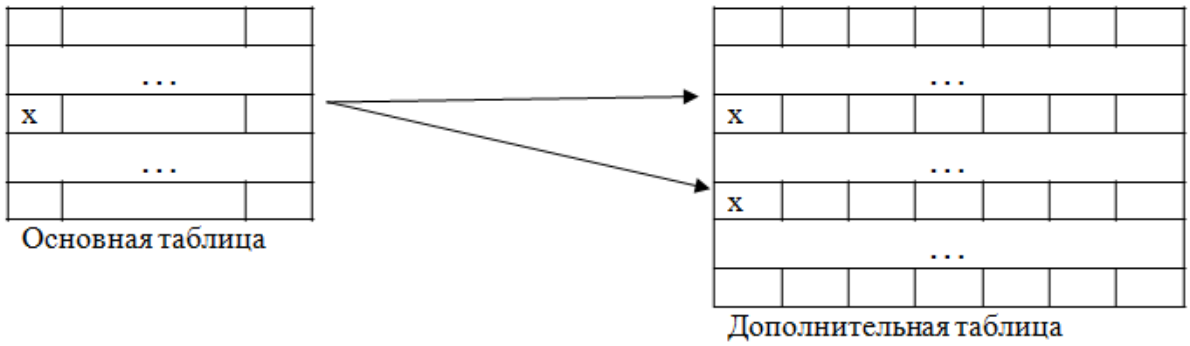


Рисунок 14 – Связь 1:М двух таблиц

Контроль целостности связей обычно означает анализ содержимого двух таблиц на соблюдение следующих правил:

– каждой записи основной таблицы соответствует нуль или более записей дополнительной таблицы

– в дополнительной таблице нет записей, которые не имеют родительских записей в основной таблице

– Каждая запись дополнительной таблицы имеет только одну родительскую запись основной таблицы.

Опишем действия контроля целостности при манипулировании данными в таблице. Рассмотрим 3 основных операции.

– ввод новых записей;

– модификацию записей;

– удаление записей.

1.При рассмотрении попытаемся охватить все возможные методы организации контроля целостности. В реальных СУБД могут быть использованы собственные методы, подобные описываемым.

При вводе новых записей возникает вопрос определения последовательности ввода записей в таблицы такой, чтобы не допустить нарушение целостности.

2.Модификация записей. Изменение содержимого полей связанных записей, не относящихся к полям связи, очевидно, должно происходить обычным образом.

Редактирование поля связи основной таблицы разумно подчинить одному из следующих правил:

– редактировать записи, у которых нет подчиненных записей. Если есть подчиненные записи, то блокировать модификацию полей связи.

– изменения в полях связи основной таблицы мгновенно передается во все поля дополнительной таблицы

3. В операциях удаления записей связанных таблиц большую свободу имеют записи дополнительной таблицы. Удаление их должно происходить практически бесконтрольно.

6 ТЕОРЕТИЧЕСКИЕ ЯЗЫКИ ЗАПРОСОВ

Операции, выполняемые над отношениями, можно разделить на две группы.

Первую группу составляют операции над множествами, к которым относятся операции: объединения, пересечения, разности, деления и декартова произведения. Вторую группу составляют специальные операции над отношениями, к которым, в частности, относятся операции: проекции, соединения, выбора.

В различных СУБД реализована некоторая часть операции над отношениями, определяющая в какой-то мере возможности данной СУБД и сложность реализации запросов к БД.

В реляционных СУБД для выполнения операции над отношениями используются две группы языков имеющие в качестве своей математической основы теоретические языки запросов, предложенные Э.Коддом: (Edgar Frank 'Ted Cood).

- реляционная алгебра;
- реляционное отношение.

Эти языки представляют минимальные возможности реальных языков манипулирования данными в соответствии с реляционной моделью, эквивалентны друг другу по своим выразительным возможностям. Существуют не очень сложные правила преобразования запросов между ними. **В реляционной алгебре** операнды и результаты всех действий являются отношениями. Языки реляционной алгебры являются процедурными, так как отношение, являющееся результатом запроса. К реляционным БД, вычисляются при выполнении последовательности реляционных операторов, применяемых к отношениям. Операторы состоят их операндов, в роли которых выступают отношения и реляционных операций.

Результатом реляционной операции является отношение.

Языки исчислений, в отличие от реляционной алгебры, являются непроцедурными (описательными или декларативными) и позволяют выражать запросы с помощью предикаты первого порядка (высказывание в виде функции), которому должны удовлетворять кортежи или домены отношений. Запрос к БД, выполненный с использованием подобного языка, содержит лишь информацию о желаемом результате. Для этих языков характерно наличие наборов правил для записи запросов. В частности, к языкам этой группы относятся SQL.

6.1 Реляционная алгебра

Реляционная алгебра как теоретический язык запросов по сравнению с реляционным исчислением более наглядно описывает выполняемые над отношениями действия.

Примером языка запросов, основанного на реляционной алгебре, является ISBL (Information System Base Language–базовый язык информационных систем).

Языки запросов, построенные на основе реляционной алгебре, в современных СУБД широко распространения не получили. Однако знакомство с ней полезно для понимания сети реляционных операций, выражаемых другими используемыми языками.

Вариант, предложенный Коддом, включает в себя следующие основные операции: объединение, разность (вычитание), пересечение, декартово (прямое) произведение (произведение), выборка (селекция, ограничение), проекция, деление и соединение.

Реляционная алгебра Кодда обладает несколькими недостатками. Во-первых, восемь перечисленных операций по охвату своих функций, с одной стороны, избыточны, так, на минимально необходимый набор составляют 5 операций: объединение, вычитание, произведение, проекция и выборка. Остальные 3 операции (пересечение, соединение и деление) можно определить через 5 минимально необходимых.

Во-вторых, эти 8 операций недостаточны для построения реальной СУБД на принципах реляционной алгебры. Требуются расширения, включения операции: переименования атрибутов, образования новых вычисляемых атрибутов и т.д.

Рассмотрим более подробно операции:

ОБЪЕДИНЕНИЕ

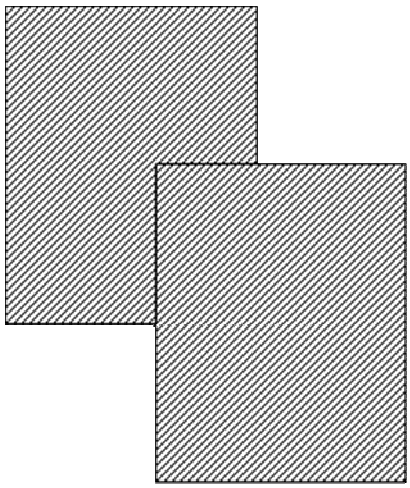


Рисунок 15 – Графическая интерпретация операции объединение

Объединение двух совместимых отношений R1 и R2 одинаковой размерности (R1 UNION R2) является отношение R содержащее все элементы исходных отношений (с исключением повторений)

Пример 1. Объединение отношений.

Пусть отношение обозначает R1 множество поставщиков из Астаны, а отношение R2 – множество поставщиков, которые поставляют деталь P1. Тогда отношение R обозначает поставщиков, находящихся в Астане, или поставщиков, выпускающих деталь P1, либо тех и других.

Таблица 12. Пример операции объединение отношений

Отношение R1

П№	Имя	Статус	Город_П
S1	Сергей	20	Астана
S4	Николай	20	Астана

Отношение R2

П№	Имя	Статус	Город_П
S1	Сергей	20	Астана
S2	Иван	10	Алматы

Отношение R1 UNION R2

П№	Имя	Статус	Город_П
S1	Сергей	20	Астана
S2	Иван	10	Алматы
S4	Николай	20	Астана

РАЗНОСТЬ

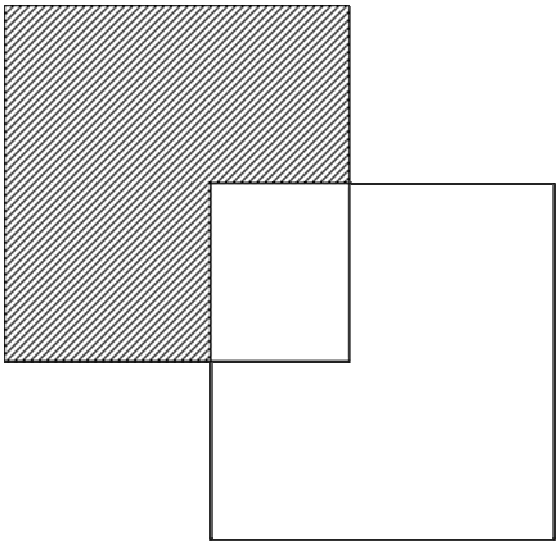


Рисунок 16 – Графическая интерпретация операции разность

Разность совместимых отношений R1 и R2 (R1 MINUS R2) одинаковой размерности, есть отношение, тело которого состоит из множества кортежей принадлежащих R1, но не принадлежащих R2.

Для тех же отношений R1 и R2 из предыдущего примера отношение R будет представлять собой множество поставщиков, находящихся в Астане, но не выпускающих деталь P1, то есть R={(S4, Николай, 20, Астана)}.

Заметим, что результат операции вычитания зависит от порядка следования операндов, то есть R1 MINUS R2 и R2 MINUS R1 – не одно и то же.

ПЕРЕСЕЧЕНИЕ

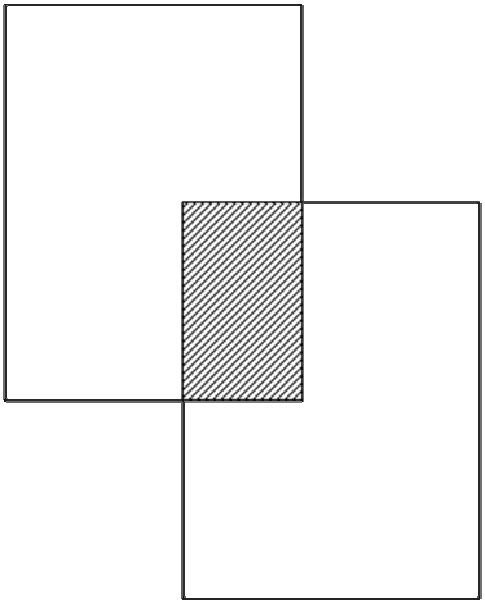


Рисунок 17 – Графическая интерпретация операции пересечение

Пересечение двух совместимых отношений R1 и R2 одинаковой размерности (R1 INTERSECT R2) порождает отношение R с телом, включающим в себя кортежи, одновременно принадлежащие обоим исходным отношениям.

Для отношений R1 и R2 результирующее отношение R будет означать всех производителей из Астаны, выпускающих деталь P1. Тело отношения R состоит из единственного элемента (S1, Сергей, 20, Астана).

ПРОИЗВЕДЕНИЕ

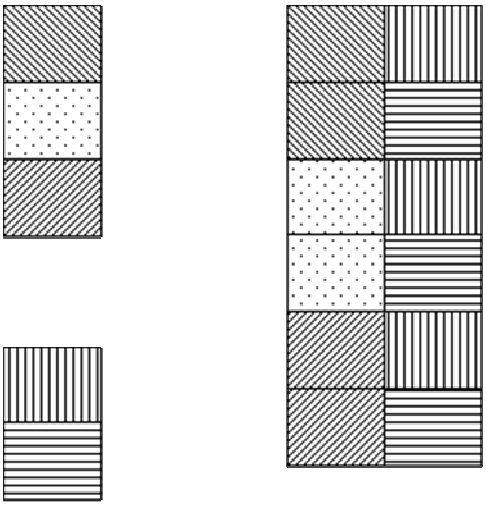


Рисунок 18 – Графическая интерпретация операции произведение

Произведение отношения R1 степени к1 и отношения R2 степени к2 (R1 TIMES R2), которые не имеют одинаковых имен атрибутов, есть такое отношение R степени (к1+к2), заголовок которого представляет сцепление заголовков отношений R1 и R2, а тело имеет кортежи такие, что первые к1 элементов кортежей принадлежат множеству R1, а последние к2 элементов – множеству R2. При необходимости получить произведение двух отношений, имеющих

одинаковые имена одного или нескольких атрибутов, применяется операция переименования RENAME.

Пример 2. Произведение отношений.

Пусть отношение R1 представляет собой множество номеров всех текущих поставщиков {S1, S2, S3, S4, S5}, а отношение R2 – множество номеров всех текущих деталей {P1, P2, P3, P4, P5, P6}. Результатом операции R1 TIMES R2 является множество всех пар типа «поставщик – деталь», то есть {(S1,P1), (S1,P2), (S1,P3), (S1,P4), (S1,P5), (S1,P6), (S2,P1),..., (S5,P6)}.

Заметим, что в теории множеств результатом операции прямого произведения является множество, каждый элемент которого является парой элементов, первый из которых принадлежит R1, а второй – принадлежит R2. Поэтому кортежами декартова произведения бинарных отношений будут кортежи вида: ((а, б), (в, г)), где кортеж (а, б) принадлежит отношению R1, а кортеж (в, г) – принадлежит отношению R2. В реляционной алгебре применяется расширенный вариант прямого произведения, при котором элементы кортежей двух исходных отношений сливаются, что при записи кортежей результирующего отношения означает удаление лишних скобок, то есть (а, б, в, г).

ВЫБОРКА

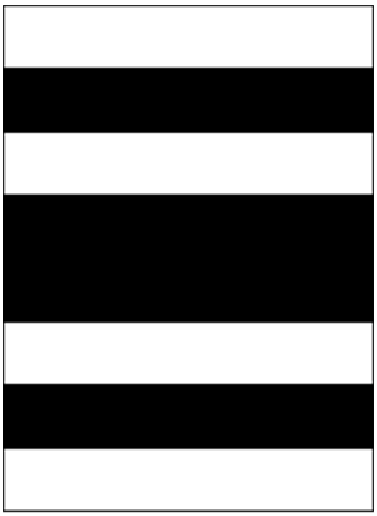


Рисунок 19 – Графическая интерпретация операции выборка

Выборка (R WHERE f) отношение R по формуле f представляет собой новое отношение с таким же заголовком и телом, состоящим из таких кортежей отношения R, которые удовлетворяют истинности логического выражения, заданного формулой f.

Для записи формулы используются операнды – имена атрибутов (или номера столбцов), константы, логические операции (AND – ***И,*** OR – ИЛИ, NOT – НЕ), операции сравнения и скобки.

Пример 3. Операция выборки (таблица 13).

Таблица 13. Пример операции выборки

P WHERE Вес <14

Д№	Названи	Тип	Вес	Город_Д
P1	гайка	каленый	12	Астана
P5	палец	твердый	12	Алматы

ПРОЕКЦИЯ

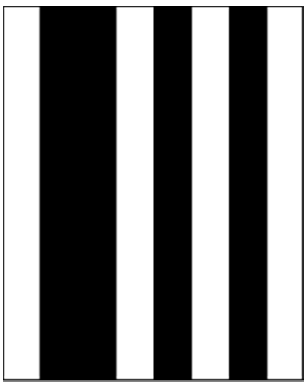


Рисунок 20 – Графическая интерпретация операции проекция

Отношение A на атрибуты X,Y,…,Z (A[X,Y,…,Z]), где множество {X,Y,…,Z} является подмножеством полного списка атрибутов заголовка отношения A , представляет собой отношение с заголовком X,Y,…,Z и телом содержащим кортежи отношений A, за исключением повторяющихся кортежей.

Операция проекции допускает следующие дополнительные варианты записи:

– отсутствие списка атрибутов подразумевает указание всех атрибутов (операция тождественной проекции);

– выражение вида R[ ] означает ***пустую*** проекцию, результатом которой является пустое множество;

– операция проекции может применяться к произвольному отношению, в том числе и к результату выборки.

Пример 4. Проекция (таблица 14).

Таблица 14. Пример операции проекции

P [Тип, Город_Д]

Тип	Город_Д
каленый	Астана
мягкий	Алматы
твердый	Костанай
твердый	Алматы

ДЕЛЕНИЕ

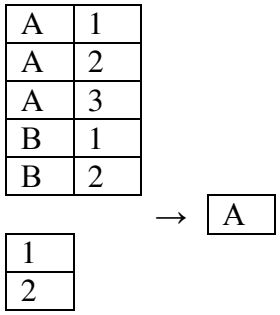


Рисунок 21 – Графическая интерпретация операции деление

Результатом деления отношения R1 с атрибутом A и B на отношение R2 с атрибутом B (R1 DIVIDEBY R2), где A и B простые или составные атрибуты, причем атрибут B – общий атрибут, определенный на одном и том же домене, является отношение R с заголовком A и телом, состоящим из кортежей r таких, что в отношении r имеются кортежи (r, s) причем множество значений s включает множество значений атрибута B отношения R2.

Пример 5. Деление отношения.

Пусть R1 – проекция SP [П№, Д№], а R2 – отношение с заголовком Д№ и телом {P2, P4}, тогда результатом деления R1 на R2 будет отношение R с заголовком П№, и телом {S1,S4} (таблица 15).

Таблица 15. Пример операции деления

R1

П№	**Д№**
S1	P1
S1	P2
S1	P3
S1	P4
S1	P5
S1	P6
S2	P1
S2	P2
S3	P2
S4	P2
S4	P4
S4	P5

R2

Д№
P2
P4

R1 DIVIDE BY R2

П№
S1
S4

СОЕДИНЕНИЕ (естественное)

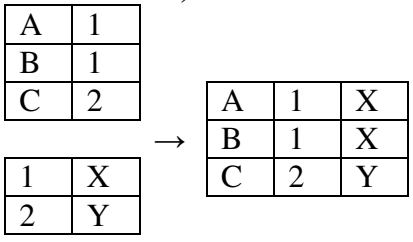


Рисунок 22 – Графическая интерпретация операции соединение

Cf (R1,R2) отношение R1и R2 по условию, заданному формулой f , представляет собой отношение R, которое можно получить, путем Декартова произведения отношений R1 и R2 с последующим применением к результату операции выборки по формуле f.

Другими словами, соединением отношения R1 по атрибуту A с отношением R2 по атрибуту B (отношения не имеют общих имен атрибутов) является результат выполнения операции вида:

(Rl TIMES R2) WHERE A Θ B,

где Q–логическое выражение над атрибутами, определенными на одном (нескольких – для составного атрибута) домене. Соединение C_f(Rl, R2), где формула f имеет произвольный вид (в отличие от частных случаев, рассматриваемых далее), называют также *Q–соединением.*

Важными с практической точки зрения частными случаями соединения являются эквисоединение и естественное соединение.

Операция *эквисоединения* характеризуется тем, что формула задает равенство операндов. Приведенный выше пример демонстрирует частный случай операции эквисоединения по одному столбцу. Иногда эквисоединение двух отношений выполняется по таким столбцам, атрибуты которых в обоих отношениях имеют соответственно одинаковые имена и домены. В этом случае говорят об эквисоединении по общему атрибуту.

Операция *естественное соединение* (операция JOIN) применяется к двум отношениям, имеющим общий атрибут (простой или составной). Этот атрибут в отношениях имеет одно и то же имя (совокупность имен) и определен на одном и том же домене (доменах).

Результатом операции естественного соединения является отношение R, которое представляет собой проекцию эквисоединения отношений R1 и R2 по общему атрибуту на объединенную совокупность атрибутов обоих отношений.

Дополнительные операции реляционной алгебры, предложенные Дейтом, *включают следующие операции.*

Операция ***переименования*** позволяет изменить имя атрибута отношения и имеет вид:

RENAME <исходное отношение> <старое имя атрибута> AS <новое имя атрибута>,

где <исходное отношение> задается именем отношения либо выражением реляционной алгебры. В последнем случае выражение заключают в круглые скобки.

Операция ***расширения*** порождает новое отношение, похожее па исходное, по отличающееся наличием добавленного атрибута, значения которого получаются путем некоторых скалярных вычислений. Операция расширения имеет вид:

EXTEND <исходное отношение> ADD <выражение> AS <новый атрибут>, где к исходному отношению добавляется (ключевое слово ADD) <новый атрибут> подсчитываемый по правилам, заданным <выражением>. Исходное отношение может быть задано именем отношения и с помощью выражения

реляционной алгебры, заключенного в круглые скобки. При этом имя нового атрибута не должно входить в заголовок исходного отношения и не может использоваться в <выражении>. Помимо обычных арифметических операций и операций сравнения, в выражении можно использовать различные функции, называемые итоговыми, такие как: COUNT (количество), SUM (сумма), AVG (среднее), MAX (максимальное), MIN (минимальное).

Результатом операции SUMMARIZE является отношение R с заголовком, состоящим из атрибутов списка, расширенного новым атрибутом. Для получения тела отношения R сначала выполняется проецирование (назовем проекцию R1) исходного отношения на атрибуты A_l, A_2,..., A_N, после чего каждый кортеж проекции расширяется новым (N+1)-м атрибутом. Поскольку проецирование, как правило, приводит к сокращению количества кортежей по отношению к исходному отношению (удаляются одинаковые кортежи), то можно считать, что происходит своеобразное группирование кортежей исходного отношения: одному кортежу отношения R1 соответствует один или более (если было дублирование при проецировании) кортежей исходного отношения. Значение (N+l)-го атрибута каждого кортежа отношения R формируется путем вычисления выражения

К основным операторам, позволяющим изменять тело существующего отношения, отнесем операции реляционного ***присвоения, вставки, обновления и удаления.***

Операцию ***присвоения*** можно представить следующим образом:

<выражение–цель> := <выражение–источник>,

где оба выражения задают совместимые (точнее, эквивалентные) по структуре отношения. Типичный случай выражений: в левой части – имя отношения, а в правой – некоторое выражение реляционной алгебры. Выполнение операции присвоения сводится к замене предыдущего значения отношения на новое (начальное значение, если тело отношения было пустым), определенное выражением–источником.

С помощью операции присвоения можно не только полностью заменить все значения отношения–цели, но и добавить или удалить кортежи.

Более удобными операциями изменения тела отношения являются операции вставки, обновления и удаления.

Операция ***вставки*** INSERT имеет следующий вид:

INSERT <выражение–источник> INTO <выражение–цель>,

где оба выражения должны быть совместимы по структуре. Выполнение операции сводится к вычислению <выражение–источник> и вставке полученных кортежей в отношение, заданное <выражение–цель>.

Операция ***обновления*** UPDATE имеет следующий вид:

UPDATE <выражение–цель> <список элементов>,

где <список элементов> представляет собой последовательность разделенных запятыми операций присвоения <атрибут>:= <скалярное выражение>. Результатом выполнения операции обновления является отношение, полученное после присвоения соответствующих значений

атрибутам отношения, заданного целевым выражением.

Операция ***удаления*** DELETE имеет следующий вид:

DELETE <выражение–цель>,

где <выражение–цель> представляет собой реляционное выражение, описывающее удаляемые кортежи.

Операция ***реляционного сравнения*** может использоваться для прямого сравнения двух отношений. Она имеет синтаксис:

<выражение1> 0 <выражение2>,

где оба выражения задают совместимые по структуре отношения, а знак 0 – один из следующих операторов сравнения: = (равно), <> (не равно), < (собственное подмножество), < (подмножество), > (надмножество), > (собственное надмножество).

6.2 Реляционное исчисление

Как отмечалось ранее, принципиальное различие между реляционной алгеброй и реляционным исчислением состоит в том, что в первом случае процесс получения искомого результата описывается явным образом путем указания набора операций, которые надо выполнить для получения результата, а во втором – указываются свойства искомого отношения без конкретизации процедуры его получения.

Внешне подходы сильно различаются: один из них предписывающий (реляционная алгебра), а другой описательный (реляционное исчисление). На более низком уровне рассмотрения подходы эквивалентны, так как любые выражения реляционной алгебры могут быть преобразованы в семантически эквивалентные выражения реляционного исчисления и наоборот. Возможность такого преобразования доказывалась многими авторами, в частности, для этого можно использовать ***алгоритм редукции*** Кодда.

Преимуществом реляционного исчисления перед реляционной алгеброй можно считать то, что пользователю не требуется самому строить алгоритм выполнения запроса. Программа СУБД (при достаточной ее интеллектуальности) сама строит эффективный алгоритм.

Математической основой реляционного исчисления является исчисление предикатов – один из разделов математической логики. Понятие реляционного исчисления как языка работы с базами данных впервые предложено Коддом. Им же был разработан язык ALPHA – прототип программно реализованного языка QUEL, который некоторое время конкурировал с языком SQL.

Существует два варианта исчислений: ***исчисление кортежей*** и ***исчисление доменов.*** В первом случае для описания отношений используются переменные, допустимыми значениями которых являются кортежи отношения, а во втором случае – элементы домена.

Реляционное исчисление, основанное на кортежах (***исчисление кортежей***), предложено и реализовано при разработке упоминавшегося языка

ALPHA. В нем, как и в процедурных языках программирования, сначала нужно описать используемые переменные, а затем записывать некоторые выражения.

Описательную часть исчисления *можно представить в виде:*

RANGE OF <переменная> IS <список>,

где прописными буквами записаны ключевые слова языка, <переменная> – идентификатор переменной кортежа (области значений), а <список> – последовательность одного или более элементов, разделенных запятыми, то есть конструкция вида: x_1, [, x_2 [..., x_j]...].

Вся конструкция RANGE указывает идентификатор переменной и область ее допустимых значений. Список элементов x_1, [, x_2 [..., x_j]...].содержит элементы, каждый из которых является либо отношением, либо выражением над отношением. Все элементы списка должны быть совместимы по типу, то есть соответствующие элементам отношения должны иметь идентичные заголовки. Область допустимых значений <переменной> образуется путем объединения значений всех элементов списка. Так, запись вида RANGE OF T IS X1,X2 означает, что область определения переменной T включает в себя все значения из отношения, которое является объединением отношений X1 и X2.

Общий смысл записи выражения состоит в перечислении атрибутов результирующего (целевого) отношения, атрибуты которого должны удовлетворять условию истинности формулы wff (well formulated formula – правильно построенная формула). Список атрибутов целевого отношения, или ***целевой список***, в терминах реляционной алгебры по существу определяет операцию проекции, а формула wff – селекцию кортежей.

В паре <переменная>.<атрибут> первая составляющая служит для указания переменной кортежа (определенной конструкцией RANGE), а вторая – для определения атрибута отношения, на котором изменяется переменная кортежа. Необязательная часть «AS <атрибут>» используется для переименования целевого отношения. Если она отсутствует, то имя атрибута целевого отношения наследуется от соответствующего имени атрибута исходного отношения.

Употребление в качестве элемента целевого отношения просто имени переменной T равносильно перечислению в списке всех соответствующих атрибутов, т. е. $T.A_1$, $T.A_2$, ... , $T.A_n$, где A_1, A_2, ... A_n – атрибуты отношения, сопоставляемого с переменной T.

Ключевые слова NOT, AND и OR обозначают логические операции соответственно: И, НЕ и ИЛИ. Ключевые слова IF и THEN переводятся соответственно «если» и «то». И наконец, ключевые слова EXISTS и FORALL называются ***кванторами.*** Первый из них – квантор существования, а второй – квантор всеобщности.

Формула wff вида: EXISTS x (f) означает: «Существует по крайней мере одно такое значение переменной x, что вычисление формулы f дает значение истина». Выражение вида: FORALL x (f) интерпретируется как высказывание: «Для всех значений переменной x вычисление формулы f даст значение истина». В общем случае переменные кортежей в формулах могут быть ***свободны-***

ми или ***связанными.*** В формулах EXISTS x (f) и FORALL x (f) переменные кортежей x всегда являются связанными.

Вариант ***реляционного исчисления, основанного на доменах*** (***исчисление доменов),*** предложен Лакроиксом и Пиротте (Lacroix and Pirotte), которые также разработали на его основе соответствующий язык ILL. Другими языками, основанными на исчислении доменов, являются: FQL, DEDUCE, а также QBE с некоторыми оговорками.

По утверждению Дейта, язык QBE включает элементы исчисления кортежей и исчисления доменов, но более близок ко второму. Он не является реляционно полным, так как не поддерживает операцию отрицания квантора существования (NOT EXISTS). Несмотря на этот недостаток, язык QBE получил широкое распространение в современных СУБД. Тем более, что реализации этого языка, как правило, шире исходного языка.

Исчисление доменов имеет много сходства с исчислением кортежей. В отличие от исчисления кортежей, в ***исчислении доменов*** основой любого выражения запроса выступают переменные доменов. ***Переменная домена*** – это скалярная переменная, значения которой охватывают элементы некоторого домена.

Большая часть различий рассматриваемых исчислений заключается в том, что исчисление доменов поддерживает дополнительную форму условия, называемую ***условием принадлежности.*** В общем виде условие принадлежности записывается в виде:

$R(A_1: \vartheta_1, A_2: \vartheta_2, \ldots)$,

где A. – атрибут отношения R, а ϑ – переменная домена или литерал. Проверяемое условие истинно, если и только если существует кортеж в отношении R, имеющий атрибуты A, равные заданным в выражении соответствующим значениям.

6.3 Язык запросов по образцу QBE

Хранимые в БД, данные можно обрабатывать вручную, последовательно просматривая и редактируя данные в таблицах, с помощью имеющихся в СУБД соответствующих средств. Для повышения эффективности применяют запросы, позволяющие производить множественную обработку данных, т.е. одновременно вводить, редактировать и удалять множество записей, а также выбирать данные из таблиц.

Запрос

Представляет собой специальным образом описанное требование, определяющее состав производимых над БД операций по выборке, удалению или модификации хранимых данных.

Два основных языка описания запросов.

– QBE
– SQL

По возможности манипулирования данными при описании запросов языки практически эквивалентны. Главное отличие в способах формирования запросов:

– QBE предлагает ручное или визуальное формирование запроса.

– SQE означает программирование запроса.

Характеристика языка QBE

Теоретической основой языка QBE является реляционное исчисление с переменными–доменами.

Язык QBE позволяет задавать сложные запросы к БД путем заполнения предлагаемой СУБД запрошенной формы. Такой способ создания запросов обеспечивает высокую наглядность и не требует указания алгоритма выполнения операции – достаточно описать образец ожидаемого результата. В каждой из современных реляционных СУБД имеется свой вариант QBE.

На языке QBE можно задать запросы однотабличные и многотабличные (выбирающие или обрабатывающие данные из нескольких таблиц).

С помощью запросов на языке QBE можно выполнять следующие основные операции:

- выборку данных
- вычисления над данными
- вставку новых записей
- удаление записей
- модификацию (изменение) данных.

Результатом выполнения запроса является новая таблица, называемая ответной (первые две операции) или обновленная исходная таблица (остальные операции).

Выборка, вставка, удаление и модификация могут производиться, безусловно, или в соответствии с условиями, которые задаются с помощью логических условий.

Вычисления над данными задаются с помощью арифметических выражений и порождают в ответных таблицах новые поля, называемые **вычисляемыми**

Запросная форма имеет вид таблиц, имя и названия полей которой совпадают с именами и названиями полей соответствующей исходной таблицы. Чтобы узнать имена доступных таблиц БД, в языке QBE предусмотрен запрос на выборку имен таблиц. Названия полей исходной таблицы могут вводиться в шаблон вручную или автоматически.

6.4 Структурированный язык запросов SQL

Язык SQL предназначен для выполнения операции над таблицами (создание, удаление, изменение структуры) и над данными таблиц (выборка, изменение, добавление и удаление), а так же некоторых сопутствующих операций. SQL является непроцедурным языком и не содержит операторов управления, организации подпрограмм, ввода-вывода и т.п. В связи с этим SQL авто-

номно не используется, обычно он погружен в среду встроенного языка программирования СУБД (FoxPro – VFP, Object PAL – СУБД PARADOX, VBA СУБД Access).

В современных СУБД с интерактивным интерфейсом можно создавать запросы, используя средства QBE. Однако применение SQL зачастую позволяет повысить эффективность обработки данных в базе.

Язык SQL не обладает функциями полноценного языка разработки, а ориентирован на доступ к данным, поэтому его включают в состав средств разработки программ, В этом случае его называют встроенным SQL. Стандарт языка SQL поддерживают современные реализации следующих языков программирования:

PL/1, Ada, C, Cobol, Fortran, MUMPS, и Pascal.

Различают два основных метода использования встроенного SQL: статический и динамический.

При **статическом** использовании языка (статический SQL) в тексте программы имеются вызовы функции языка SQL, которые жестко включаются в выполняемый модуль после компиляции. Изменения в вызываемых функциях могут быть на уровне отдельных параметров вызовов с помощью переменных языка программирования.

При **динамическом** использовании языка (динамический SQL) предполагается динамическое построение вызовов SQL функции и интерпретация этих вызовов.

Основным назначением языка SQL является подготовка и выполнение запросов.

Основные группы команд:

1. DDL (Data Definition Language) –язык определения данных (создание и редактирование таблиц)
2. DML (Data Manipulation Language) – язык манипулирования данными
3. DCL (Data Control Language) – язык управления доступа к данным
4. DQL (Data Query Language) – язык запроса данных

Опишем минимальное подмножество языка SQL, опираясь на его реализацию в стандартном интерфейсе ODBC– (Open Database Connectivity – совместимость открытых баз данных) фирмы Microsoft.

Рассмотрим формат и основные возможности важнейших операторов:

Группа команд DDL

Оператор ***создания таблицы*** имеет формат вида:

CREATE TABLE имя_таблицы(
имя_поля_1 тип_данных;
имя_поля_2 тип_данных;
. . .
имя_поля_n тип_данных)

Обязательными операндами оператора являются имя создаваемой таблицы и имя хотя бы одного столбца (поля) с указанием типа данных, хранимых в этом столбце.

При создании таблиц могут использоваться ограничения:

1. Ограничение типа NOT NULL – поле не может содержать пустого значения

CREATE TABLE имя_таблицы (поле_1 тип_данных NOT NULL)

2. Ограничение первичного ключа PRIMARY KEY – установление первичного ключа

CREATE TABLE имя_таблицы (поле_1 тип_данных PRIMARY KEY NOT NULL)

3. Ограничение UNIQUE – поля д.б. заполнены (может содержать пустое значение)и уникальны

CREATE TABLE имя_таблицы(поле_1 тип_данных UNIQUE)

4. Ограничение внешнего ключа (рисунок 23)

PORIGEN KEY имя_внешнего_ключа (поле_1)

REFERENCES имя_родительской_таблицы (список_родительских_ полей)

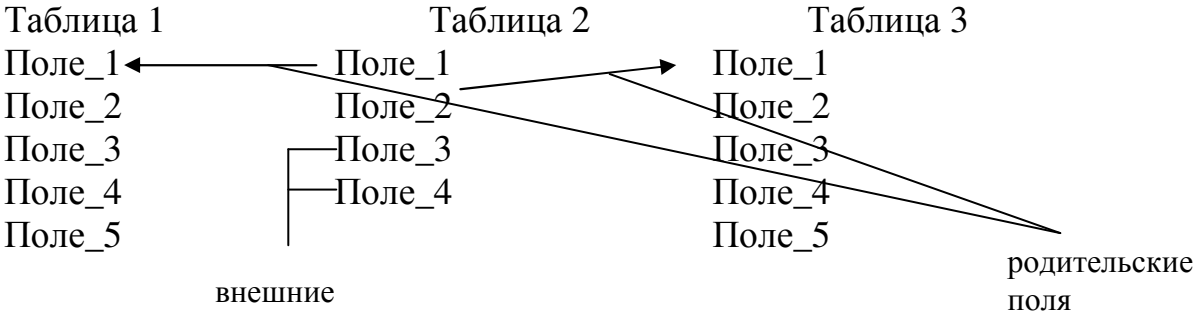


Рисунок 23 –Ограничение внешнего ключа

Оператор ***изменения структуры таблицы*** имеет формат вида:

ALTER TABLE имя_таблицы режим,

где

режим– параметр, который может принимать следующие значения:

– MODIFY – изменить столбец таблицы;

– ADD – добавить столбец в таблицу;

– DROP – удалить столбец таблицы.

Оператор ***удаления таблицы*** имеет формат вида:

DROP TABLE имя_таблицы [RESTRICT | CASCADE],

где:

- RESTRICT – выдается сообщение, если есть связь с другой таблицей
- CASCADE – удаляет все связи

Оператор ***создания индекса*** имеет формат вида:

CREATE [UNIQUE] INDEX <имя индекса>

ON <имя таблицы>

(<имя столбца> [ASC | DESC]

[,<имя столбца> [ASC | DESC] ...)

Оператор позволяет создать индекс для одного или нескольких столбцов заданной таблицы с целью ускорения выполнение запросных и поисковых операций с таблицей. Для одной таблицы можно создать несколько индексов.

Задав необязательную опцию UNIQUE, можно обеспечить уникальность значений во всех указанных в операторе столбцах. По существу, создание индекса с указанием признака UNIQUE означает определение ключа в созданной ранее таблице.

При создании индекса можно задать порядок автоматической сортировки значений в столбцах – в порядке возрастания ASC (по умолчанию), или в порядке убывания DESC. Для разных столбцов можно задавать различный порядок сортировки.

Оператор ***удаления индекса*** имеет формат вида:

DROP INDEX <имя индекса>

Этот оператор позволяет удалять созданный ранее индекс с соответствующим именем.

Оператор ***создания представления*** имеет формат вида:

CREATE VIEW <имя представления>
[(<имя столбца> [,<имя столбца>]...)]
AS <оператор SELECT>

Данный оператор позволяет создать представление. Если имена столбцов в представлении не указываются, то будут использоваться имена столбцов из запроса, описываемого соответствующим оператором SELECT.

Оператор ***удаления представления*** имеет формат вида:

DROP VIEW <имя представления>

Оператор позволяет удалить созданное ранее представление. Заметим, что при удалении представления таблицы, участвующие в запросе, удалению не подлежат.

Группа команд DQL

Оператор ***выборки записей*** имеет формат вида:

SELECT [ALL | DISTINCT]
<список данных>
FROM <список таблиц>
[WHERE <условие выборки>]
[GROUP BY <имя столбца> [,<имя столбца>]...]
[HAVING <условие поиска>]
[ORDER BY <спецификация> [,<спецификация>] ...]

SELECT это наиболее важный оператор из всех операторов SQL. Функциональные возможности его огромны.

Оператор SELECT позволяет производить выборку и вычисления над данными из одной или нескольких таблиц. Результатом выполнения оператора является ответная таблица, которая может иметь (ALL), или не иметь (DISTINCT) повторяющиеся строки. По умолчанию в ответную таблицу

включаются все строки, в том числе и повторяющиеся. В отборе данных участвуют записи одной или нескольких таблиц, перечисленных в списке операнда FROM.

Список данных может содержать имена столбцов, участвующих в запросе, а также выражения над столбцами. В простейшем случае в выражениях можно записывать имена столбцов, знаки арифметических операций (+, –,*,/), константы и круглые скобки. Если в списке данных записано выражение, то наряду с выборкой данных выполняются вычисления, результаты которого попадают в новый (создаваемый) столбец ответной таблицы.

При использовании в списках данных имен столбцов нескольких таблиц для указания принадлежности столбца некоторой таблице применяют конструкцию вида: <имя таблицы>.<имя столбца>.

Операнд WHERE задает условия, которым должны удовлетворять записи в результирующей таблице. Выражение <условие выборки> является логическим. Его элементами могут быть имена столбцов, операции сравнения, арифметические операции, логические связки (И, ИЛИ, НЕТ), скобки, специальные функции LIKE, NULL, IN и т. д.

Логические операции применяемые в конструкции WHERE:

1. IS NULL – для сравнения текущего значения со значением NULL
2. BETWEEN ... AND ... для отбора записей, в которых значение поля находится внутри заданного диапазона.
3. IN – для отбора записей, в которых значение поля соответствует хотя бы одному из значений заданного списка
4. LIKE – применяется для сравнения значения поля со значением заданным при помощи шаблона

% – как *

_ – как ?

5. EXIST – используется для отбора записей соответствующих заданному критерию
6. UNIQUE– используется для отбора записей соответствующих заданному критерию, только используется проверки записей таблицы на уникальность. Запрос с использованием этого оператора должен возвращать не более одной записи
7. ALL – для сравнения исходного значения со всеми другими значениями, входящими в некоторый набор данных.

Операнд GROUP BY позволяет выделять в результирующем множестве записей группы. ***Группой*** являются записи с совпадающими значениями в столбцах, перечисленных за ключевыми словами GROUP BY. Выделение групп требуется для использования в логических выражениях операндов WHERE и HAVING, а также для выполнения операций (вычислений) над группами.

В логических и арифметических выражениях можно использовать следующие групповые операции (функции): +, –, *, /, AVG (среднее значение в группе), MAX (максимальное значение в группе), MIN (минимальное значение в группе), SUM (сумма значений в группе), COUNT (число значений в группе).

Операнд HAVING действует совместно с операндом GROUP BY и используется для дополнительной селекции записей во время определения групп. Правила записи <условия поиска> аналогичны правилам формирования <условия выборки> операнда WHERE.

Операнд ORDER BY задает порядок сортировки результирующего множества. Обычно каждая <спецификация> аналогична соответствующей конструкции оператора CREATE INDEX и представляет собой пару вида: <имя столбца> [ASC | DESC].

Оператор SELECT может иметь и другие более сложные синтаксические конструкции. Одной из таких конструкций, например, являются так называемые ***подзапросы.*** Они позволяют формулировать ***вложенные запросы,*** когда результаты одного оператора SELECT используются в логическом выражении условия выборки операнда WHERE другого оператора SELECT.

Вторым примером более сложной формы оператора SELECT является оператор, в котором отобранные записи в дальнейшем предполагается модифицировать (конструкция FOR UPDATE OF). СУБД после выполнения такого оператора обычно блокирует (защищает) отобранные записи от модификации их другими пользователями.

Еще один случай специфического использования оператора SELECT – выполнение объединений результирующих таблиц при выполнении нескольких операторов SELECT (операнд UNION).

Оператор ***изменения записей*** имеет формат вида:

UPDATE <имя таблицы>
SET <имя столбца> = {<выражение> , NULL }
[, SET <имя столбца> = {<выражение> , NULL } ...]
[WHERE <условие>]

Выполнение оператора UPDATE состоит в изменении значений в определенных операндом SET столбцах таблицы для тех записей, которые удовлетворяют условию, заданному операндом WHERE.

Новые значения полей в записях могут быть пустыми (NULL), либо вычисляться в соответствии с арифметическим выражением. Правила записи арифметических и логических выражений аналогичны соответствующим правилам оператора SELECT.

Оператор ***вставки новых записей*** имеет форматы двух видов:

INSERT INTO <имя таблицы>
[(<список столбцов>)]
VALUES (<список значений>)

и

INSERT INTO <имя таблицы>
[(<список столбцов>)]
<предложение SELECT>

В первом формате оператор INSERT предназначен для ввода новых записей с заданными значениями в столбцах. Порядок перечисления имен столбцов должен соответствовать порядку значений, перечисленных в списке

операнда VALUES. Если <список столбцов> опущен, то в <списке значений> должны быть перечислены все значения в порядке столбцов структуры таблицы.

Во втором формате оператор INSERT предназначен для ввода в заданную таблицу новых строк, отобранных из другой таблицы с помощью предложения SELECT.

Оператор ***удаления записей*** имеет формат вида:

DELETE FROM <имя таблицы>

[WHERE <условие>]

Результатом выполнения оператора DELETE является удаление из указанной таблицы строк, которые удовлетворяют условию, определенному операндом WHERE. Если необязательный операнд WHERE опущен, то есть условие отбора удаляемых записей отсутствует, удалению подлежат все записи таблицы.

7 ПРОЕКТИРОВАНИЕ БАЗ ДАННЫХ

7.1 Проблемы проектирования БД

Проектирование ИС, включает в себя БД, осуществляется на физическом и логическом уровнях. Решение проблем проектирования на физическом уровне во многом зависит от используемой СУБД, зачастую автоматизировано и скрыто от пользователя. В ряде случаев пользователю предоставляется возможность настройки отдельных параметров системы, которая не составляет большой проблемы.

Логическое проектирование заключается в определении числа и структуры таблиц, формировании запросов к БД, определении типов отчетных документов, разработке алгоритмов обработки информации создании форм для ввода и редактирования данных в базе и решения рада других задач.

Решение задач логического проектирования БД в основном определяется спецификой задач предметной области. Наиболее важной здесь является проблема структуризации данных.

При проектировании структур данных для автоматизированных систем можно выделить 3 основных подхода:

1. Сбор информации об объектах решаемой задачи в рамках одной таблицы (одного отношения) и последующая декомпозиция ее на несколько взаимосвязанных таблиц на основе процедуры нормализации отношений.

2. Формулирование знаний о системе (определение типов исходных данных и их взаимосвязей) и требований к обработке данных, получение с помощью CASE–системы (системы автоматизации проектирования и разработки БД) готовой схемы БД или даже готовой прикладной ИС.

3. Структурирование информации для использования в ИС в процессе проведения системного анализа на основе совокупности правил и рекомендации.

Рассмотрим первый из названных подходов. Прежде всего, охарактеризуем **основные проблемы,** имеющие место **при определении структур данных в отношениях** реляционной модели.

– Избыточное дублирование данных и аномалии.

Следует различать простое (не избыточное) и избыточное дублирование данных. Наличие первого из них допускается в БД, а избыточное дублирование может приводить к проблемам при обработке данных (таблица 16).

Таблица 16. Пример простого (не избыточного) и избыточного дублирования данных

С_Т (не избыточное)

Сотрудники	Телефон
Иванов	3721
Петров	4328
Сидоров	4328
Егоров	4328

С_Т_Н (избыточное)

Сотрудники	тел	Н–ком
Иванов	3721	109
Петров	4328	111
Сидоров	4328	111
Егоров	4328	111

Возможен выход декомпозиции исходного отношения (таблица 17).

Таблица 17. Декомпозиция исходного отношения

T_H

тел	Номер
3721	109
4328	111

C_H

Сотрудники	номер
Иванов	109
Петров	111
Сидоров	111
Егоров	111

Процедура декомпозиции отношения является основной процедурой нормализации отношений.

Избыточное дублирование данных создает проблемы при обработке кортежей отношения, названные Эдгаром Коддом «аномалиями обновления отношения». Он показал, что для некоторых отношений проблемы возникают при попытке удаления, добавления или редактировании их кортежей.

Аномалиями будем называть такую ситуацию в таблицах БД, которая приводит к противоречиям в БД, либо существенно усложняет обработку данных.

Выделяют 3 основные вида аномалий: аномалии модификации (или редактирования), аномалии удаления и аномалии добавления.

Аномалии модификации проявляются в том, что изменение значения одного данного может повлечь за собой просмотр всей таблицы и соответствующее изменение некоторых других записей таблицы.

Аномалия удаления состоит в том, что при удалении какого либо данного из таблиц может пропасть и другая информация, которая не связана напрямую с удаляемым данным.

Аномалия добавления возникает в случаях, когда информацию в таблицу нельзя поместить до тех пор, пока она не полная, либо вставка новой записи требует дополнительного просмотра таблицы.

Формирование исходного отношения.

Проектирование БД начинается с определения всех объектов, сведения о которых будут включены в базу данных, и определения их атрибутов. Затем атрибуты сводятся в одну таблицу – исходное отношение.

Если исходное отношение содержит избыточное дублирование данных, то средством исключения избыточности в отношениях и как следствие аномалии является нормализация отношений.

7.2 Зависимости между атрибутами

Проектирование БД является одним из этапов жизненного цикла системы. Основной задачей, решаемой в процессе проектирования БД, является задача нормализации ее отношений. Метод нормализации форм является классическим методом проектирования реляционной БД. Этот метод основан на фундаментальном в теории реляционных БД понятии зависимости между атрибутами отношений.

Зависимости между атрибутами

Рассмотрим основные виды зависимостей между атрибутами отношений:

- функциональные
- транзитивные
- многозначные
- частичные

Понятие функциональной зависимости является базовым, так как на его основе формулируется определение всех остальных видов зависимостей.

Атрибут B функционально зависит от атрибута A, если каждому значению A соответствует в точности одно значение B.

Математически A –>B

Функциональная взаимозависимость.

Если существует функциональная зависимость вида A–>B и B–>A, то между A и B имеются взаимно однозначное соответствие или функциональная зависимость.

Частичной зависимостью (*частичной функциональной зависимостью*) называется зависимость неключевого атрибута от части составного ключа.

Атрибут C зависит от атрибута A **транзитивно** (существует транзитивная зависимость), если для атрибутов A, B, C выполняется условие A–>B и B–>C, но обратная зависимость отсутствует.

В отношении R атрибут B **многозначно** зависит от атрибута A, если каждому значению a соответствует множество значений B, не связанных с другими атрибутами из R.

Альтернативным вариантом является ***полная функциональная зависимость*** неключевого атрибута от всего составного ключа

Взаимно независимые атрибуты. Два или более атрибута называются взаимно независимыми, если ни один из этих атрибутов не является функционально зависимым от других атрибутов.

Выявление зависимостей между атрибутами необходимо для выполнения проектирования БД методом нормальных форм.

Основной способ определения наличия функциональных зависимостей – внимательный анализ семантики атрибутов. Для каждого отношения существует, но не всегда, определенное множество функциональных зависимостей между атрибутами. Причем если в некотором отношении существует одна или несколько функциональных зависимостей, можно вывести другие функциональные зависимости, существующие в этом отношении.

8 НОРМАЛЬНЫЕ ФОРМЫ

8.1 Нормализация отношений

Процесс проектирования БД с использованием метода нормальных форм (НФ) является итерационным и заключается в последовательном переводе отношений из первой НФ в НФ более высокого порядка по определенным правилам. Каждая следующая НФ ограничивает определенный тип функциональных зависимостей, устраняет соответствующие аномалии при выполнении операции над отношениями БД и сохраняет свойства предыдущих нормальных форм.

Выделяют следующую последовательность НФ:

- первая НФ (1 НФ)
- вторая НФ (2 НФ)
- Третья НФ (3 НФ)
- Усиленная третья НФ или НФ Бейса–Кодда (БК НФ)
- Четвертая НФ (4 НФ)
- Пятая НФ (5 НФ)

Первая НФ

Отношение находится в 1НФ, если все его атрибуты являются простыми (имеют единственное значение). Исходное отношение строится таким образом, чтобы оно было в 1НФ.

Перевод отношения в следующую нормальную форму осуществляется методом «декомпозиций без потерь». Декомпозиция должна обеспечивать то, что запросы (выборка данных по условию) к исходному отношению и к отношениям, получаемым в результате декомпозиции. Дадут одинаковый результат. Основной операцией метода является операция проекции.

Основной операцией метода является операция проекции. Поясним ее на примере. Предположим, что в отношении R(A,B,C,D,E,...) устранение функциональной зависимости C–>D позволит перевести его в следующую нормальную форму. Для решения этой задачи выполним декомпозицию отношения R па два новых отношения R1(A,B,C,E,...) и R2(C,D). Отношение R2 является проекцией отношения R на атрибуты C и D.

Вторая НФ

Отношение находится во 2НФ, если оно находится в 1НФ, и каждый неключевой атрибут функционально полно зависит от первичного ключа (составного).

Для устранения частичной зависимости и перевода отношения в 2НФ необходимо, используя операцию проекции, разложить его на несколько отношений следующим образом:

– построить проекцию без атрибутов, находящихся в частичной функциональной зависимости от первичного ключа.

– построить проекции на части составного первичного ключа и атрибуты, зависящие от этих частей.

Для дальнейшего совершенствования отношения необходимо преобразовать его в 3НФ.

Третья НФ

Определение 1. Отношение находится в 3НФ, если оно находится в 2НФ, и каждый неключевой атрибут не транзитивно зависит от первичного ключа.

Существует и альтернативное определение.

Определение 2. Отношение находится в 3НФ в том и только в том случае, если все неключевые атрибуты отношения взаимно независимы и полностью зависят от первичного ключа.

Доказать справедливость этого утверждения несложно. Действительно, то, что неключевые атрибуты полностью зависят от первичного ключа, означает, что данное отношение находится в форме 2НФ. Взаимная независимость атрибутов (определение приведено выше) означает отсутствие всякой зависимости между атрибутами отношения, в том числе и транзитивной зависимости между ними. Таким образом, второе определение 3НФ сводится к первому определению.

На практике построение 3НФ схем отношений в большинстве случаев является достаточным и приведением к ним процесс проектирования реляционной БД заканчивается.

Если в отношении имеется зависимость атрибутов составного ключа от неключевых атрибутов, то необходимо перейти к усиленной 3НФ.

Усиленная 3НФ и НФ Бейса–Кодда (БКНФ)

Отношение находится в БК НФ, если оно находится в 3НФ и в нем отсутствуют зависимости ключей (атрибутов составного ключа) от неключевых атрибутов.

Четвертая НФ

Отношение R находится в 4НФ в том и только в том случае, когда существует многозначная зависимость A–>B, а все остальные атрибуты R функционально зависят от A.

Дальнейшая нормализация отношений, , основывается на следующей теореме ***Фейджина***.

Теорема Фейджина (Fagin R.). Отношение R(A, B, C) можно спроецировать без потерь в отношения R1(A, B) и R2(A, C) в том и только том случае, когда существует зависимость A=>B |C .

Под проецированием без потерь здесь понимается такой способ декомпозиции отношения, при котором исходное отношение полностью и без избыточности восстанавливается путем ***естественного соединения*** полученных отношений.

Пятая НФ

Отношение R находятся в 5НФ в том и только том случае, когда любая зависимость соединения в R следует из существования некоторого возможного ключа в R.

На практике обычно ограничиваются структурой БД, соответствующей 3НФ или БКНФ. Поэтому процесс нормализации отношений методом нормальных форм предполагает последовательное удаление из исходного отношения следующих межатрибутных зависимостей:

- частичных зависимостей неключевых атрибутов от ключа (удовлетворение требований 2НФ);
- транзитивных зависимостей неключевых атрибутов от ключа (удовлетворение требований 3НФ);
- зависимости ключей (атрибутов составных ключей) от неключевых атрибутов (удовлетворение требований БКНФ).

Кроме метода нормальных форм Кодда, используемого для проектирования небольших БД, применяют и другие методы, например, метод ER–диаграмм (метод «Сущность–связь»), Этот метод используется при проектировании больших БД, па нем основан ряд средств проектирования БД.

8.2 Рекомендации по разработке структур

Основное правило при создании таблиц сущностей – это «каждой сущности – отдельную таблицу» (как в популярном лозунге: «каждой семье – отдельную квартиру»),

Поля таблиц сущностей могут быть двух видов: ***ключевые*** и ***неключевые.*** Введение ключей в таблице практически во всех реляционных СУБД позволяет обеспечить уникальность значений в записях таблицы по ключу, ускорить обработку записей таблицы, а также выполнить автоматическую сортировку записей по значениям в ключевых полях.

Обычно достаточно определения ***простого*** ключа, реже – вводят ***составной*** ключ. Таблицей с составным ключом может быть, например, таблица хранения списка сотрудников (фамилия, имя и отчество), в котором встречаются однофамильцы. В некоторых СУБД пользователям предлагается определить автоматически создаваемое ключевое поле нумерации (в Access – это поле типа «счетчик»), которое упрощает решение проблемы уникальности записей таблицы.

Иногда в таблицах сущностей имеются поля описания свойств или характеристик объектов. Если в таблице есть значительное число повторений по этим полям и эта информация имеет существенный объем, то лучше их выделить в отдельную таблицу (придерживаясь правила: «каждой сущности – отдельную таблицу»). Тем более, следует образовать дополнительную таблицу, если свойства взаимосвязаны.

В более общем виде последние рекомендации можно сформулировать так: информацию о сущностях следует представить таким образом, чтобы не-

ключевые поля в таблицах были взаимно независимыми и полностью зависели от ключа (см. определение третьей нормальной формы).

В процессе обработки таблиц сущностей надо иметь в виду следующее. Новую сущность легко добавить и изменить, но при удалении следует уничтожить все ссылки на нее из таблиц связей, иначе таблицы связей будут некорректными. Многие современные СУБД блокируют некорректные действия в подобных операциях.

Организация связи сущностей

Записи ***таблицы связей*** предназначены для отображения связей между сущностями, информация о которых находится в соответствующих таблицах сущностей.

Обычно одна таблица связей описывает взаимосвязь двух сущностей. Поскольку таблицы сущностей в простейшем случае имеют по одному ключевому полю, то таблица связей двух таблиц для обеспечения уникальности записей о связях должна иметь два ключа. Можно создать таблицу связей, как и таблицу объектов, и без ключей, но тогда функции контроля над уникальностью записей ложатся на пользователя.

Более сложные связи (не бинарные) следует сводить к бинарным. Для описания взаимосвязей N объектов требуется N–1 таблиц связей. Транзитивных связей не должно быть. Избыток связей приводит к противоречиям.

Не следует включать в таблицы связей характеристики сущностей, иначе неизбежны аномалии. Их лучше хранить в отдельных таблицах сущностей.

С помощью таблиц связей можно описывать и несколько специфичный вид связи – линейную связь, или слабую связь. Примером линейной связи можно считать отношение принадлежности сущностей некоторой другой сущности более высокого порядка (системы, состоящие из узлов; лекарства, состоящие из компонентов; сплавы металлов и т. д.). В этом случае для описания связей достаточно одной таблицы связей.

При работе с таблицами связей следует иметь в виду, что любая запись из таблицы связей легко может быть удалена, поскольку сущности некоторое время могут обойтись и без связей. При добавлении или изменении содержимого записей таблицы надо контролировать правильность ссылок на существующие объекты, так как связь без объектов существовать не может. Большинство современных СУБД контролируют правильность ссылок на объекты.

8.3 Обеспечение целостности

Под ***целостностью*** понимают свойство базы данных, означающее, что она содержит полную, непротиворечивую и адекватно отражающую предметную область информацию.

Различают ***физическую*** и ***логическую*** целостность. Физическая целостность означает наличие физического доступа к данным и то, что данные не утрачены. Логическая целостность означает отсутствие логических ошибок в базе данных, к которым относятся нарушение структуры БД или ее объектов,

удаление или изменение установленных связей между объектами и т. д. В дальнейшем речь будем вести о логической целостности.

Поддержание целостности БД включает проверку (контроль) целостности и ее восстановление в случае обнаружения противоречий в базе. Целостное состояние БД задается с помощью ***ограничений целостности*** в виде условий, которым должны удовлетворять хранимые в базе данные.

Среди ограничений целостности можно выделить два основных типа ограничений: ***ограничения значений*** атрибутов отношений и ***структурные ограничения*** на кортежи отношений.

Примером ***ограничений значений*** атрибутов отношений является требование недопустимости пустых или повторяющихся значений в атрибутах, а также контроль принадлежности значений атрибутов заданному диапазону.

Наиболее гибким средством реализации контроля значений атрибутов являются ***хранимые процедуры*** и ***триггеры,*** имеющиеся в некоторых СУБД.

Структурные ограничения определяют требования ***целостности сущностей*** и ***целостности ссылок.*** Каждому экземпляру сущности, представленному в отношении, соответствует только один его кортеж. Требование ***целостности сущностей*** состоит в том, что любой кортеж отношения должен быть отличим от любого другого кортежа этого отношения, т. е., иными словами, любое отношение должно обладать первичным ключом.

Формулировка требования целостности ссылок тесно связана с понятием ***внешнего ключа.*** Напомним, что внешние ключи служат для связи отношений (таблиц БД) между собой. При этом атрибут одного отношения (родительского) называется ***внешним ключом*** данного отношения, если он является ***первичным*** ключом другого отношения (дочернего). Говорят, что отношение, в котором определен внешний ключ, ссылается на отношение, в котором этот же атрибут является первичным ключом.

Требование целостности ссылок состоит в том, что для каждого значения внешнего ключа родительской таблицы должна найтись строка в дочерней таблице с таким же значением первичного ключа.

Во многих современных СУБД имеются средства обеспечения контроля целостности БД.

9 МЕТОД СУЩНОСТЬ – СВЯЗЬ

Метод сущность–связь называют так же методом «ER–диаграмм»:

– во-первых, ER аббревиатура от слов Essence (сущность) и Relation (связь)

– во-вторых, метод основан на использовании диаграмм, называемых соответственно диаграммами ER–экземпляров и диаграммами ER–типа.

9.1 Основные понятия метода

Сущность, представляет собой объект, информация о котором хранится в БД.

Экземпляры сущности отличаются друг от друга и однозначно идентифицируются с названиями сущностей, как правило, являются существительные, ПРЕПОДАВАТЕЛЬ, ДИСЦИПЛИНА, КАФЕДРА.

Атрибут представляет собой свойство сущности. Это понятие аналогично понятию атрибута в отношении – свойство, характеризующее сущность.

Ключ сущности – атрибут или набор атрибутов, используемый для идентификации экземпляра сущности.

Связь двух или более сущностей – предполагает зависимость между атрибутами этих сущностей. Название связи обычно представляется глаголом.

С целью повышения наглядности и удобства проектирования для представления сущностей, экземпляров сущностей и связей между ними используются следующие графические средства:

*диаграммы ER – экземпляров

* диаграмма ER – типа или ER – диаграмма

Рассмотрим пример: диаграммы ER – экземпляров для сущностей ПРЕПОДАВАТЕЛЬ и ДИСЦИПЛИНА со связью ведет.

Диаграмма ER – типа, соответственно рассмотрим диаграмму ER – экземпляр (рисунок 24).

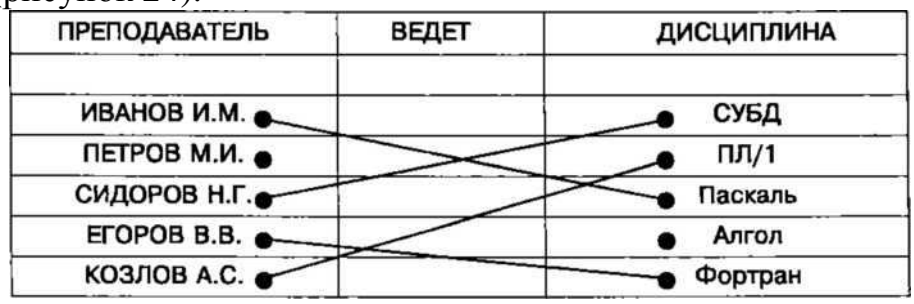


Рисунок 24 – Диаграмма ER – экземпляров

Диаграмма конкретно показывает, кто, что ведет (рисунок 25).

Рисунок 25 – Диаграмма ER – типа

На начальном этапе проектирования БД выделяются атрибуты, составляющие ключи сущностей.

На основе анализа диаграмм ER –тира формируются отношения проектируемой БД. При этом учитывается степень связей сущностей и класс их принадлежности, которые в свою очередь, определяются на основе анализа диаграмм ER – экземпляров соответствующих сущностей.

Степень связи является характеристикой связи между сущностями, которая может быть типа 1:1, 1:M, M:1, M:M.

Класс принадлежности (КП) может быть обязательным и не обязательным. КП сущности является обязательным, если все экземпляры этой сущности обязательно участвуют в рассмотрении связи, в противном случае класс принадлежности сущности является не обязательным.

9.2 Этапы проектирования

Процесс проектирования БД является итерационным – допускающим возврат к предыдущим этапам для пересмотра ранее принятых решений и включает следующие этапы:

1. Выделение сущностей и связей между ними
2. Построение диаграмм ER–типа с учетом всех сущностей и их связей
3. Формирование набора предварительных отношений с указанием предполагаемого первичного ключа для каждого отношения и использованием диаграмм ER – типа.
4. Добавление неключевых атрибутов в отношения.
5. Приведение предварительных отношений к НФ БК.
6. Просмотр ER – диаграмм в следующих случаях:

– некоторые отношения не приводятся к НФ БК

– некоторым атрибутам не находится логически обоснованных мест в предварительных отношениях.

После преобразования ER –диаграмм осуществляется повторное выполнение предыдущих этапов проектирования (возврат к этапу 1)

Одним из условных этапов проектирования является этап формирования отношений.

9.3 Правила формирования отношений

Правила формирования отношения основываются на учете следующего:

– степени связи между сущностями (1:1, 1:M, M:1, M:M).

– класса принадлежности (КП) экземпляров сущностей (обязательный и необязательный)

Существуют 6 правил формирования отношений на основе ER–типа.

Связь вида 1:1

Правило № 1.

Если степень бинарной связи 1:1 и класс принадлежности обеих сущностей обязательный, то формируется одно отношение. Первичным ключом этого отношения может быт ключ любой из двух сущностей.

Правило № 2.

Если степень связи 1:1 и (класс) КП принадлежит одной сущности обязательный, а второй не обязательный, то под каждую из сущностей формируется по отношению с первичными ключами. Является ключами соответствующих сущностей.

Далее к отношению, сущность которого не имеет обязательный КП, добавляется в качестве атрибута ключ сущности с обязательным КП.

Правило № 3.

Если степень связи 1:1 и КП обеих сущностей является необязательным, то необходимо использовать 3 отношения 2 отношения соответствующие связываемым сущностям, ключи, которых являются первичными в этих отношениях. Третье отношение является связным между двумя, поэтому его ключ объединяет ключевые атрибуты связываемых отношений.

Связь вида 1:M

Правило № 4.

Если степень связи между отношением 1:M (или M:1) и класс КП M – связной сущности обязательный, то достаточно формирование двух отношений (по одному на каждую из сущностей). При этом первичными ключами этих отношений являются ключи их сущностей. Кроме того, ключ 1 – связной сущности добавляется как атрибут (внешний ключ) в отношение соответствии M–связной сущности.

Правило № 5.

Если степень связи 1:M (M:1) и КП M – связной сущности является необязательным, то необходимо формирование трех отношений. Два отношения соответствует связываемым сущностям, ключи которых являются первичными в этих отношениях. Третье отношение является связным между первыми двумя (его ключ объединяет ключевой атрибут связываемых отношений).

Связь вида M:M

Правило № 6.

Если степень связи M:M , то независимо от класса принадлежности сущностей формируется три отношения. Два отношения соотв. Связываемым сущностям и их ключи являются первичными ключами этих отношений. Третье отношение является связным между первыми двумя, а его ключ объединяет ключевые атрибуты связываемых отношений.

10 ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ СИСТЕМ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ

Системы управления базами данных стали доминирующим инструментом хранения больших массивов информации. Сколько-нибудь развитые информационные приложения полагаются не на файловые структуры операционных систем, а на многопользовательские СУБД, выполненные в технологии клиент/сервер. В этой связи обеспечение информационной безопасности СУБД, и в первую очередь их серверных компонентов, приобретает решающее значение для безопасности организации в целом.

Для СУБД важны все три основных аспекта информационной безопасности – конфиденциальность, целостность и доступность. Общая идея защиты баз данных состоит в следовании рекомендациям, сформулированным для класса безопасности С2 в «Критериях оценки надежных компьютерных систем». Для иллюстрации излагаемых понятий и средств будут использоваться СУБД INGRES, Informix и Oracle.

Идентификация и проверка подлинности пользователей

Обычно в СУБД для идентификации и проверки подлинности пользователей применяются либо соответствующие механизмы операционной системы, либо SQL–оператор CONNECT. Например, в случае СУБД Oracle оператор CONNECT имеет следующий вид:

CONNECT пользователь[/пароль] [@база_данных];

Так или иначе, в момент начала сеанса работы с сервером баз данных, пользователь идентифицируется своим именем, а средством аутентификации служит пароль. Детали этого процесса определяются реализацией клиентской части приложения.

Управление доступом

Обычно в СУБД применяется произвольное управление доступом, когда владелец объекта передает права доступа к нему (чаще говорят – привилегии) по своему усмотрению. Привилегии могут передаваться субъектам (отдельным пользователям), группам, ролям или всем пользователям.

Группа – это именованная совокупность пользователей. Объединение субъектов в группы облегчает администрирование баз данных и, как правило, строится на основе формальной или фактической структуры организации. Каждый пользователь может входить в несколько групп. Когда пользователь тем или иным способом инициирует сеанс работы с базой данных, он может указать, от имени какой из своих групп он выступает. Кроме того, для пользователя обычно определяют подразумеваемую группу.

Роль – это еще один возможный именованный носитель привилегий. С ролью не ассоциируют перечень допустимых пользователей – вместо этого роли защищают паролями. В момент начала сеанса с базой данных можно специфицировать используемую роль (обычно с помощью флагов или эквивалентного механизма) и ее пароль, если таковой имеется. Привилегии роли имеют приоритет над привилегиями пользователей и групп. Иными словами, пользо-

вателю как субъекту не обязательно иметь права доступа к объектам, обрабатываемым приложениям с определенной ролью.

Основные категории пользователей

Пользователей СУБД можно разбить на три категории:

- администратор сервера баз данных. Он ведает установкой, конфигурированием сервера, регистрацией пользователей, групп, ролей и т.п. Прямо или косвенно он обладает всеми привилегиями, которые имеют или могут иметь другие пользователи.
- администраторы базы данных. К этой категории относится любой пользователь, создавший базу данных, и, следовательно, являющийся ее владельцем. Он может предоставлять другим пользователям доступ к базе и к содержащимся в ней объектам. Администратор базы отвечает за ее сохранение и восстановление. В принципе в организации может быть много администраторов баз данных.
- прочие (конечные) пользователи. Они оперируют данными, хранящимися в базах, в рамках выделенных им привилегий.

Поручать администрирование различных баз данных разным людям имеет смысл только тогда, когда эти базы независимы и по отношению к ним не придется проводить согласованную политику выделения привилегий или резервного копирования. В таком случае каждый из администраторов будет знать ровно столько, сколько необходимо.

Виды привилегий

Привилегии в СУБД можно подразделить на две категории: привилегии безопасности и привилегии доступа. Привилегии безопасности позволяют выполнять административные действия. Привилегии доступа, в соответствии с названием, определяют права доступа субъектов к определенным объектам.

Привилегии безопасности

Привилегии безопасности всегда выделяются конкретному пользователю (а не группе, роли или всем) во время его создания или изменения характеристик. Таких привилегий пять:

- security – право управлять безопасностью СУБД и отслеживать действия пользователей. Пользователь с этой привилегией может подключаться к любой базе данных, создавать, удалять и изменять характеристики пользователей, групп и ролей, передавать права на доступ к базам данным другим пользователям, управлять записью регистрационной информации, отслеживать запросы других пользователей и, наконец, запускать команды от имени других пользователей. Привилегия security необходима администратору сервера баз данных, а также лицу, персонально отвечающему за информационную безопасность. Передача этой привилегии другим пользователям (например, администраторам баз данных) увеличивает число потенциально слабых мест в защите сервера баз данных.
- createdb – право на создание и удаление баз данных. Этой привилегией, помимо администратора сервера, должны обладать пользователи, которым отводится роль администраторов отдельных баз данных.

– operator – право на выполнение действий, которые традиционно относят к компетенции оператора. Имеются в виду запуск и остановка сервера, сохранение и восстановление информации. Помимо администраторов сервера и баз данных этой привилегией целесообразно наделить также администратора операционной системы.

– maintain_locations – право на управление расположением баз баз данных и операционной системы.

– trace – право на изменение состояния флагов отладочной трассировки. Данная привилегия полезна администратору сервера баз данных и другим знающим пользователям при анализе сложных, непонятных ситуаций.

Привилегии доступа

Привилегии доступа выделяется пользователям, группам, ролям владельцем соответствующих объектов (он же – администратор базы данных) или обладатель привилегии security (обычно администратор сервера баз данных). Прежде чем присваивать привилегии группам и ролям, их (группы и роли) необходимо создать. При совершении подобных действий необходимо иметь подключение к базе данных iidbdb, в которой хранятся сведения о субъектах и их привилегиях. Привилегии доступа можно подразделить в соответствии с видами объектов, к которым они относятся:

– таблицы и представления
– процедуры
– базы данных
– сервер баз данных
– события

Применительно к таблицам и представлениям можно управлять следующими правами доступа:

SELECT	– право на выборку данных
INSERT	– право на добавление данных
DELETE	– право на удаление данных
UPDATE	– право на обновление данных (можно указать определенные столбцы, разрешенные для обновления)
REFERENCES	– право на использование внешних ключей, ссылающихся на данную таблицу (можно указать определенные столбцы)

Иерархия прав доступа Средства управления доступом СУБД позволяют реализовать следующие виды ограничения доступа:

– операционные ограничения
– ограничения по значениям (за счет механизма представлений);
– ограничения на ресурсы (за счет привилегий доступа к базам данных).

При обработке запроса СУБД сначала проверяет права доступа к объектам. Если операционные ограничения оказываются нарушенными, запрос отвергается с выдачей соответствующей диагностики. Нарушение ограничений на значения влияет только на количество результирующих строк; никакой диагностики при этом не выдается. Наконец, после учета двух предыдущих ограни-

чений, запрос поступает на обработку оптимизатору. Если тот обнаружит превышение ограничений на ресурсы, запрос будет отвергнут с выдачей соответствующей диагностики. На иерархию привилегий можно посмотреть и с другой точки зрения. Каждый пользователь, помимо, собственных, имеет привилегии PUBLIC. Кроме этого, он может входить в различные группы и запускать приложения с определенными ролями. Как соотносятся между собой права, предоставленные различным именованным носителям привилегий? Иерархия авторизации выглядит для СУБД следующим образом:

- роль (высший приоритет)
- пользователь
- группа
- PUBLIC (низший приоритет)

Метки безопасности и принудительный контроль доступа

Средства произвольного управления доступом не могут помешать авторизованному пользователю законным образом получить секретную информацию и затем сделать ее доступной для других, неавторизованных пользователей. Нетрудно понять, почему это так. При произвольном управлении доступом привилегии существуют отдельно от данных. В результате данные оказываются «обезличенными», и ничто не мешает передать их кому угодно даже средствами самой СУБД. В «Критериях оценки надежных компьютерных систем», применительно к системам уровня безопасности B, описан механизм меток безопасности. Применять эту версию на практике имеет смысл только в сочетании с операционной системой и другими программными компонентами того же уровня безопасности. В СУБД к каждой реляционной таблице неявно добавляется столбец, содержащий метки безопасности строк таблицы. Метка безопасности состоит из трех компонентов:

- *Уровень секретности.* Смысл этого компонента зависит от приложения. В частности, возможен традиционный спектр уровней от «совершенно секретно» до «несекретно».
- *Категории.* Понятие категории позволяет разделить данные на «отсеки» и тем самым повысить надежность системы безопасности. В коммерческих приложениях категориями могут служить «финансы», «кадры», «материальные ценности» и т.п.
- *Области.* Является дополнительным средством деления информации на отсеки. На практике компонент «область» может действительно иметь географический смысл, обозначая, например, страну, к которой относятся данные.

Каждый пользователь СУБД характеризуется степенью благонадежности, которая также определяется меткой безопасности, присвоенной данному пользователю. Пользователь может получить доступ к данным, если степень его благонадежности удовлетворяет требованиям соответствующей метки безопасности. Более точно:

- уровень секретности пользователя должен быть не ниже уровня секретности данных;

– набор категорий, заданных в метке безопасности данных, должен целиком содержаться в метке безопасности пользователя;

– набор областей, заданных в метке безопасности пользователя, должен целиком содержаться в метке безопасности данных.

При добавлении или изменении строк они, как правило, наследуют метки безопасности пользователя, инициировавшего операцию. Таким образом, даже если авторизованный пользователь перепишет секретную информацию в общедоступную таблицу, менее благонадежные пользователи не смогут ее прочитать.

Поддержание целостности данных в СУБД

Для коммерческих организаций обеспечение целостности данных по крайней мере не менее важно, чем обеспечение конфиденциальности.

Известно, что главными врагами баз данных являются не внешние злоумышленники, а ошибки оборудования, администраторов, прикладных программ и пользователей. Защита от подобных ошибок – главная тема этого раздела. С точки зрения пользователя СУБД, основными средствами поддержания целостности данных являются ограничения и правила.

Ограничения

Ограничения могут относиться к таблицам или отдельным столбцам. Ограничения на столбцы задаются при создании таблицы. Ссылочные ограничения отвечают за целостность связей между таблицами. Подобное ограничение требует, чтобы каждому значению в столбце или группе столбцов одной таблицы соответствовало ровно одно значение в другой таблице.

Ограничения всех видов накладываются владельцем таблицы и влияют на исход последующих операций с данными. Перед завершением выполнения SQL–оператора производится проверка имеющихся ограничений. При обнаружении нарушений СУБД сигнализирует о ненормальном завершении и аннулирует внесенные оператором изменения.

Ограничения можно не только накладывать, но и отменять. При этом между ограничениями могут существовать зависимости, и отмена одного из них может потребовать ликвидации других (ссылочных) ограничений, зависящих от первоначального.

Правила

Правила позволяют вызывать выполнение заданных действий при определенных изменениях базы данных. Обычно действие – это вызов процедуры. Правила ассоциируются с таблицами и срабатывают при изменении этих таблиц.

В отличие от ограничений, которые являются лишь средством контроля относительно простых условий, правила позволяют проверять и поддерживать сколь угодно сложные соотношения между элементами данных в базе. Как и в случае ограничений, проверка правил отключается при массовых операциях копирования.

СУБД обеспечивает автоматическое удаление правил в тех случаях, когда удаляется соответствующая таблица. Тем самым поддерживается целостность системы таблиц и правил.

В контексте информационной безопасности важно отметить, что создать правило, ассоциируемое с таблицей, может владелец этой таблицы, имеющий право на выполнение соответствующей процедуры. Пользователь, действия которого вызывают срабатывание правила, должен обладать лишь необходимыми правами доступа к таблице. Тем самым правила неявно расширяют привилегии пользователей. Подобные расширения нуждаются в строгом административном контроле, поскольку даже незначительное изменение правила или ассоциированной процедуры может кардинально повлиять на защищенность данных. Ошибка же в сложной системе правил вообще чревата непредсказуемыми последствиями.

Средства поддержания высокой готовности

В коммерческих приложениях высокая готовность аппаратно-программных комплексов является важнейшим фактором. Применительно к СУБД средства поддержания высокой готовности должны обеспечивать нейтрализацию аппаратных отказов, особенно касающихся дисков, а также восстановление после ошибок обслуживающего персонала или прикладных программ. Подобные средства должны с самого начала закладываться в архитектуру комплекса. Например, необходимо использовать тот или иной вид избыточных дисковых массивов. Конечно, это сделает аппаратно-программное решение более дорогим, но зато убережет от возможных убытков во время эксплуатации.

Угрозы, специфичные для СУБД

Главный источник угроз, специфичных для СУБД, лежит в самой природе баз данных. Основным средством взаимодействия с СУБД является язык SQL – мощный непроцедурный инструмент определения и манипулирования данными. Хранимые процедуры добавляют к этому репертуару управляющие конструкции. Механизм правил дает возможность выстраивать сложные, трудные для анализа цепочки действий, позволяя попутно неявным образом передавать право на выполнение процедур, даже не имея, строго говоря, полномочий на это. В результате потенциальный злоумышленник получает в свои руки мощный и удобный инструментарий, а все развитие СУБД направлено на то, чтобы сделать этот инструментарий еще мощнее и удобнее.

Получение информации путем логических выводов

Нередко путем логического вывода можно извлечь из базы данных информацию, на получение которой стандартными средствами у пользователя не хватает привилегий. Если для реализации контроля доступа используются представления, и эти представления допускают модификацию, с помощью операций модификации/вставки можно получить информацию о содержимом базовых таблиц, не располагая прямым доступом к ним.

Основным средством борьбы с подобными угрозами, помимо тщательно проектирования модели данных, является механизм размножения строк. Суть его в том, что в состав первичного ключа, явно или неявно, включается метка

безопасности, за счет чего появляется возможность хранить в таблице несколько экземпляров строк с одинаковыми значениями «содержательных» ключевых полей. Наиболее естественно размножение строк реализуется в СУБД, поддерживающих метки, однако и стандартными SQL–средствами можно получить удовлетворительное решение.

Агрегирование данных

Агрегирование – это метод получения новой информации путем комбинирования данных, добытых легальным образом из различных таблиц. Агрегированная информация может оказаться более секретной, чем каждый из компонентов, ее составивший.

Покушения на высокую готовность (доступность)

Если пользователю доступны все возможности SQL, он может довольно легко затруднить работу других пользователей (например, инициировав длительную транзакцию, захватывающую большое число таблиц). Современные многопотоковые серверы СУБД отражают лишь самые прямолинейные атаки, состоящие, например, в запуске в «часы пик» операции массовой загрузки данных. Настоятельно рекомендуется не предоставлять пользователям непосредственного SQL– доступа к базе данных, используя в качестве фильтров серверы приложений.

11 РАСПРЕДЕЛЕННЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ И БАЗЫ ДАННЫХ

Под распределенной (Distributed DataBase – DDB) обычно подразумевают базу данных, включающую фрагменты из нескольких баз данных, которые располагаются на различных узлах сети компьютеров, и, возможно управляются различными СУБД. Распределенная база данных выглядит с точки зрения пользователей и прикладных программ как обычная локальная база данных. В этом смысле слово «распределенная» отражает способ организации базы данных, но не внешнюю ее характеристику. («распределенность» базы данных невидима извне).

Определение распределенных баз данных (DDB) предложил Дэйт (C.J. Date). Он установил 12 свойств или качеств идеальной DDB:

Локальная автономия (local autonomy)

– Независимость узлов (no reliance on central site)
– Непрерывные операции (continuous operation)
– Прозрачность расположения (location independence)
– Прозрачная фрагментация (fragmentation independence)
– Прозрачное тиражирование (replication independence)
– Обработка распределенных запросов (distributed query processing)
– Обработка распределенных транзакций (distributed transaction processing)
– Независимость от оборудования (hardware independence)
– Независимость от операционных систем (operationg system independence)
– Прозрачность сети (network independence)
– Независимость от баз данных (database independence)

Локальная автономия. Это качество означает, что управление данными на каждом из узлов распределенной системы выполняется локально. База данных, расположенная на одном из узлов, является неотъемлемым компонентом распределенной системы. Будучи фрагментом общего пространства данных, она, в то же время функционирует как полноценная локальная база данных; управление ею выполняется локально и независимо от других узлов системы.

Независимость от центрального узла. В идеальной системе все узлы равноправны и независимы, а расположенные на них базы являются равноправными поставщиками данных в общее пространство данных. База данных на каждом из узлов самодостаточна – она включает полный собственный словарь данных и полностью защищена от несанкционированного доступа.

Непрерывные операции. Это качество можно трактовать как возможность непрерывного доступа к данным (известное «24 часа в сутки, семь дней в неделю») в рамках DDB вне зависимости от их расположения и вне зависимости от операций, выполняемых на локальных узлах. Это качество можно выра-

зить лозунгом «данные доступны всегда, а операции над ними выполняются непрерывно».

Прозрачность расположения. Это свойство означает полную прозрачность расположения данных. Пользователь, обращающийся к DDB, ничего не должен знать о реальном, физическом размещении данных в узлах информационной системы. Все операции над данными выполняются без учета их местонахождения. Транспортировка запросов к базам данных осуществляется встроенными системными средствами.

Прозрачная фрагментация. Это свойство трактуется как возможность распределенного (то есть на различных узлах) размещения данных, логически представляющих собой единое целое. Существует фрагментация двух типов: горизонтальная и вертикальная. Первая означает хранение строк одной таблицы на различных узлах (фактически, хранение строк одной логической таблицы в нескольких идентичных физических таблицах на различных узлах). Вторая означает распределение столбцов логической таблицы по нескольким узлам.

Прозрачность тиражирования. Тиражирование данных – это асинхронный (в общем случае) процесс переноса изменений объектов исходной базы данных в базы, расположенные на других узлах распределенной системы. В данном контексте прозрачность тиражирования означает возможность переноса изменений между базами данных средствами, невидимыми пользователю распределенной системы. Данное свойство означает, что тиражирование возможно и достигается внутрисистемными средствами.

Обработка распределенных запросов. Это свойство DDB трактуется как возможность выполнения операций выборки над распределенной базой данных, сформулированных в рамках обычного запроса на языке SQL. То есть операцию выборки из DDB можно сформулировать с помощью тех же языковых средств, что и операцию над локальной базой данных.

Обработка распределенных транзакций. Это качество DDB можно трактовать как возможность выполнения операций обновления распределенной базы данных (INSERT, UPDATE, DELETE), не разрушающее целостность и согласованность данных. Эта цель достигается применением двухфазового или двухфазного протокола фиксации транзакций (two–phase commit protocol), ставшего фактическим стандартом обработки распределенных транзакций. Его применение гарантирует согласованное изменение данных на нескольких узлах в рамках распределенной (или, как ее еще называют, глобальной) транзакции.

Независимость от оборудования. Это свойство означает, что в качестве узлов распределенной системы могут выступать компьютеры любых моделей и производителей – от мэйнфреймов до «персоналок».

Независимость от операционных систем. Это качество вытекает из предыдущего и означает многообразие операционных систем, управляющих узлами распределенной системы.

Прозрачность сети. Доступ к любым базам данных может осуществляться по сети. Спектр поддерживаемых конкретной СУБД сетевых протоколов не должен быть ограничением системы с распределенными базами данных.

Данное качество формулируется максимально широко – в распределенной системе возможны любые сетевые протоколы.

Независимость от баз данных. Это качество означает, что в распределенной системе могут мирно сосуществовать СУБД различных производителей, и возможны операции поиска и обновления в базах данных различных моделей и форматов.

Исходя из определения Дэйта, можно рассматривать DDB как слабосвязанную сетевую структуру, узлы которой представляют собой локальные базы данных. Локальные базы данных автономны, независимы и самоопределены; доступ к ним обеспечиваются СУБД, в общем случае от различных поставщиков. Связи между узлами – это потоки тиражируемых данных. Топология DDB варьируется в широком диапазоне – возможны варианты иерархии, структур типа «звезда» и т.д. В целом топология DDB определяется географией информационной системы и направленностью потоков тиражирования данных.

Посмотрим, во что выливается некоторые наиболее важные свойства DDB, если рассматривать их практически.

Целостность данных

В DDB поддержка целостности и согласованности данных, ввиду свойств 1–2, представляет собой сложную проблему. Ее решение – синхронное и согласованное изменение данных в нескольких локальных базах данных, составляющих DDB – достигается применением протокола двухфазной фиксации транзакций. Если DDB однородна – то есть на всех узлах данные хранятся в формате одной базы и на всех узлах функционирует одна и та же СУБД, то используется механизм двухфазной фиксации транзакций данной СУБД. В случае же неоднородности DDB для обеспечения согласованных изменений в нескольких базах данных используют менеджеры распределенных транзакций. Это, однако, возможно, если участники обработки распределенной транзакции – СУБД, функционирующие на узлах системы, поддерживают XA–интерфейс, определенный в спецификации DTP консорциума X/Open. В настоящее время XA–интерфейс имеют CA–OpenIngres, Informix, Microsoft SQL Server, Oracle, Sybase.

Если в DDB предусмотрено тиражирование данных, то это сразу предъявляет дополнительные жесткие требования к дисциплине поддержки целостности данных на узлах, куда направлены потоки тиражируемых данных. Проблема в том, что изменения в данных инициируются как локально – на данном узле – так и извне, посредством тиражирования. Неизбежно возникают конфликты по изменениям, которые необходимо отслеживать и разрешать.

Прозрачность расположения

Это качество DDB в реальных продуктах должно поддерживаться соответствующими механизмами. Разработчики СУБД придерживаются различных подходов.

Обработка распределенных запросов

Выше уже упоминалось это качество DDB. Обработка распределенных запросов (Distributed Query –DQ) – задача, более сложная, нежели обработка ло-

кальных и она требует интеллектуального решения с помощью особого компонента – оптимизатора DQ.

Межоперабельность

В контексте DDB межоперабельность означает две вещи.

Во-первых, – это качество, позволяющее обмениваться данными между базами данных различных поставщиков. Как, например, тиражировать данные из базы данных Informix в Oracle и наоборот. Известно, что штатные средства тиражирования в составе данной конкретной СУБД позволяют переносить данные в однородную базу. Так, средствами CA–Ingres/Replicator можно тиражировать данные только из Ingres в Ingres. Как быть в неоднородной DDB. Ответом стало появление продуктов, выполняющих тиражирование между разнородными базами данных.

Во-вторых, это возможность некоторого унифицированного доступа к данным в DDB из приложения. Возможны как универсальные решения (стандарт ODBC), так и специализированные подходы. Очевидный недостаток ODBC – недоступность для приложения многих полезных механизмов каждой конкретной СУБД, поскольку они могут быть использованы в большинстве случаев только через расширения SQL в диалекте языка данной СУБД, но в стандарте ODBC эти расширения не поддерживаются.

Технология тиражирования данных

Принципиальная характеристика тиражирования данных (Data Replication – DR) заключается в отказе от физического распределения данных. Суть DR состоит в том, что любая база данных (как для СУБД, так и для работающих с ней пользователей) всегда является локальной; данные размещаются локально на том узле сети, где они обрабатываются; все транзакции в системе завершаются локально.

Тиражирование данных – это асинхронный перенос изменений объектов исходной базы данных в базы, принадлежащим различным узлам распределенной системы. Функции DR выполняет, как правило, специальный модуль СУБД – сервер тиражирования данных, называемый репликатором (так устроены СУБД CA–OpenIngres и Sybase). В Informix–OnLine Dynamic Server репликатор встроен в сервер, в Oracle 7 для использования DR необходимо приобрести дополнительно к Oracle7 DBMS опцию Replication Option.

Специфика механизмов DR зависит от используемой СУБД.

12 ОСНОВНЫЕ ПОНЯТИЯ СУБД MS ACCESS

Под базой данных следует понимать унифицированную совокупность специально организованных данных, совместно используемая различными задачами в рамках некоторой единой автоматизированной информационной системы. БД хранятся на каком–либо носителе, например на бумаге, на магнитном диске и т.д. Программы для хранения, обработки и поиска большого объема информации называются системами управления базой данных (СУБД).

Microsoft Access – это функционально полная реляционная СУБД. В ней предусмотрены все необходимые средства для определения и обработки данных, а также для управления ими при работе с большими объемами информации. MS Access – это набор инструментальных средств создания и эксплуатации информационных систем.

Средствами MS Access производятся следующие операции:

1. Проектирование баз данных – двухмерных таблиц, с разными типами данных, включая логические типы, поля объектов OLE и гиперссылки.

2. Установление связей между таблицами, с поддержкой целостности данных, каскадного обновления полей и каскадного удаления записей.

3. Ввод, хранение, просмотр, сортировка, модификация и выборка данных из таблиц с использованием различных средств контроля информации, индексирования таблиц и аппарата алгебры и логики (для фильтрации данных).

4. Создание, модификация и использование производных объектов ИС (форм, запросов и отчетов).

Объекты базы данных

СУБД MS Access ориентирована на работу с объектами, к которым относятся таблицы базы данных, формы, запросы, отчеты, макросы и модули. Для типовых процессов обработки данных – ввода, просмотра, обновления, поиска по заданным критериям, получения отчетов – MS Access позволяет конструировать в диалоговом режиме такие объекты, как формы, запросы, отчеты.

Таблица – это базовый объект MS Access. Все остальные объекты являются производными и создаются только на базе ранее подготовленных таблиц. Таблицы создаются пользователем для хранения данных по одному объекту модели данных предметной области.

Форма не является обязательным объектом MS Access. Формы предназначены для упрощения ввода, просмотра и корректировки взаимосвязанных данных базы данных.

Запросы – это производная таблица, в которую собираются данные из уже существующих таблиц. Они создаются пользователем для выборки нужных данных из одной или нескольких связанных таблиц, над которыми можно производить различные операции. Запрос может формироваться с помощью запросов по образцу (QBE) или с помощью языка структурированных запросов SQL. С помощью запроса можно также обновить, удалить или добавить данные в таблицы или создать новые таблицы на основе уже существующих.

Отчеты фактически тот же запрос, но оформленный так, чтобы его можно было напечатать на бумаге. Т.е. отчеты предназначены для формирования выходного документа, предназначенного для вывода на печать.

Запросы и отчеты выполняют самостоятельные функции: выбирают, группируют, отображают, печатают информацию.

Общие рекомендации при создании базы данных Microsoft Access

Предварительные действия перед созданием базы данных

Перед созданием базы данных необходимо ответить на следующие вопросы.

- Каково назначение базы данных и кто будет ею пользоваться?
- Какие таблицы (данные) будет содержать база данных?
- Какие запросы и отчеты могут потребоваться пользователям этой базы данных?
- Какие формы может потребоваться создать?

Отвечая на эти вопросы, можно разработать проект базы данных и создать полезную и удобную в использовании базу данных.

Удачная разработка базы данных обеспечивает простоту ее поддержки. Данные следует сохранять в таблицах, причем каждая таблица должна содержать информацию одного типа. Тогда достаточно будет обновить конкретные данные, только в одном месте, чтобы обновленная информация отображалась во всей базе данных.

Правильно спроектированная база данных обычно содержит разнообразные запросы, позволяющие отображать нужную информацию. В запросах может выводиться подмножество данных, например перечень заказчиков из определенного города, или комбинированные данные из нескольких таблиц, например сведения о заказах совместно со сведениями о заказчиках.

Прежде чем приступить в Microsoft Access к фактической разработке таблиц, запросов, форм и других объектов, рекомендуется предварительно спланировать структуру на бумаге.

Определение цели создания базы данных

На первом этапе разработки базы данных необходимо определить ее назначение и как она будет использоваться.

- Посоветуйтесь с будущими пользователями базы данных. Вместе с ними сформулируйте вопросы, ответы на которые вы и они хотите получать с помощью базы данных. В процессе проектирования рекомендуется обращаться к пользователям для того чтобы уточнять, корректировать и улучшать базу данных.
- Создайте эскизы отчетов, которые хотелось бы получить.
- Соберите формы, которые вы уже используете для ввода данных.

По мере определения предназначения базы данных начнет формироваться перечень необходимых данных. Зная это, можно определить, какие фактические данные следует сохранять в базе данных и по каким темам распределяются эти данные. Темам должны соответствовать таблицы, а данным – поля (столбцы) в этих таблицах.

Определение нужных полей в базе данных

Каждое поле содержит определенные фактические данные. Для каждого типа сведений следует создать отдельное поле. При составлении схемы полей, учитывайте следующее.

- Включайте все необходимые сведения.
- Разбивайте информацию на минимальные логические компоненты.
- Не создавайте поля для данных, состоящих из нескольких элементов.
- Не рекомендуется включать в таблицу данные, которые являются результатом вычисляемого выражения.
- Не создавайте поля, содержащие аналогичные данные.

Определение таблиц, которые должна содержать база данных

Каждая таблица должна содержать информацию только на одну тему. Список нужных полей подскажет, какие требуются таблицы.

Определение таблиц, к которым относятся поля

При решении вопроса, к какой таблице должно относиться каждое поле, необходимо учитывать следующие принципы разработки.

- Включайте каждое поле только в одну таблицу.
- Не включайте поле в таблицу, если в результате его добавления одни и те же данные будут появляться в нескольких записях этой таблицы. Если оказывается, что поле таблицы содержит много повторяющихся данных, это поле, вероятно, помещено не в ту таблицу.

Данные, хранящиеся только в одной таблице, обновляются только один раз. Это более эффективно и, кроме того, исключает возможность дублирования записей, содержащих разные сведения.

Определение полей с уникальными значениями в каждой записи

Для связывания в Microsoft Access сведений, хранящихся в разных таблицах – например для связывания клиента со всеми его заказами – каждая таблица базы данных должна содержать поля или набор полей, однозначно определяющих каждую запись. Такое поле или набор полей называют первичный ключ или ключевое поле.

Определение связей между таблицами

После разбиения сведений на таблицы и определения ключевых полей, необходимо выбрать способ, которым Microsoft Access будет вновь объединять связанные сведения. Для этого следует определить виды связей между таблицами базы данных Microsoft Access.

Может оказаться полезным изучить связи в существующей базе данных с хорошо организованной структурой.

Усовершенствование структуры базы данных

После создания нужных таблиц, полей и взаимосвязей таблиц необходимо еще раз просмотреть структуру базы данных и выявить возможные недочеты. Желательно это сделать на данном этапе, пока таблицы не заполнены данными.

Создайте таблицы в Microsoft Access, создайте между ними связи и введите в таблицы достаточный объем данных для проверки структуры. Чтобы

проверить связи в базе данных, посмотрите, удается ли создать запросы для получения нужных сведений. Создайте черновые формы и отчеты и посмотрите, отображаются ли в них те данные, что ожидалось. Найдите излишние повторы данных и исключите их.

Ввод данных и создание других объектов базы данных

Если структуры таблиц отвечают поставленным требованиям, то можно ввести все данные. Затем можно создать все необходимые объекты базы данных – запросы, отчеты, формы и другие элементы. Совокупность описаний, инструкций и процедур, сохраненная под общим именем и будет представлять собой базу данных.

Использование средств анализа Microsoft Access

В Microsoft Access существуют два инструмента, помогающие усовершенствовать структуру базы данных.

– Мастер анализа таблиц позволяет проанализировать структуру таблицы, предложить подходящие новые структуры и связи таблиц, а также разделить таблицу на новые связанные таблицы, если это имеет смысл.

– Анализатор быстродействия исследует всю базу данных и дает рекомендации по ее улучшению. Мастер может также выполнить эти рекомендации.

13 РАЗРАБОТКА ТАБЛИЦ

13.1 Создание таблиц

В Microsoft Access поддерживаются четыре метода создания баз данных:

1. Создание базы данных с помощью мастера.
2. Создание базы данных с помощью шаблона.
3. Создание пустой базы данных без помощи мастера.
4. Импорт данных из другого источника.

Создание базы данных с помощью мастера

Можно создать базу данных с помощью мастера создания баз данных. Этот мастер позволяет выбирать один из встроенных шаблонов и настраивать его требуемым образом. Затем создается набор таблиц, запросов, форм и отчетов, а также кнопочная форма базы данных. Таблицы не содержат данных. Этот метод используется, если встроенный шаблон полностью удовлетворяет требованиям.

Можно воспользоваться мастером баз данных для создания всех необходимых таблиц, форм и отчетов для базы данных выбранного типа – это простейший способ начального создания базы данных. Мастер предлагает ограниченный набор параметров для настройки базы данных.

1. Нажмите кнопку **Создать** на панели инструментов.
2. В области задач **Создание файла** в группе **Шаблоны** выберите **На моем компьютере**.
3. Выберите значок подходящего шаблона базы данных на вкладке **Базы данных** и нажмите кнопку **ОК**.
4. В диалоговом окне **Файл новой базы данных** введите имя базы данных и укажите ее расположение, а затем нажмите кнопку **Создать**.
5. Следуйте инструкциям мастера баз данных.

Если мастер не запускается, то причиной этого может быть то, что Access работает в ограниченном режиме, а на компьютере не установлено ядро Microsoft Jet 4.0 с пакетом обновления 8 (SP8) или более поздним. Ядро Jet 4.0 с пакетом обновления 8 или более поздним необходимо для полноценной работы Access при включенном ограниченном режиме.

Следует помнить, что с помощью мастера баз данных нельзя добавлять новые таблицы, формы и отчеты в существующую базу данных.

При использовании Microsoft Access 2003, можно выполнить поиск шаблонов Access на веб–узле <Office Online>. Загрузка шаблона – это самый быстрый способ создания базы данных. Если удалось найти шаблон, полностью удовлетворяющий требованиям, используйте метод, описанный выше. Шаблон представлен файлом Access (*.mdb) и содержит таблицы, запросы, формы и отчеты. Таблицы не содержат данных. После открытия базы данных можно настроить ее и ее объекты.

Создание базы данных с помощью шаблона

Это самый быстрый способ создания базы данных. Этот метод работает лучше остальных, если удастся найти и использовать шаблон, наиболее подходящий требованиям.

1. Нажмите кнопку **Создать** на панели инструментов.

2. В области задач **Создание файла** в списке **Шаблоны** либо выполните поиск конкретного шаблона, либо выберите пункт **Шаблоны на веб–узле Office Online** для обзора всех шаблонов.

3. Выберите требуемый шаблон Access и нажмите кнопку **Загрузить**.

Если требуется начать создание базы данных по собственному проекту, создайте пустую базу данных, а затем добавьте в нее таблицы, формы, отчеты и другие объекты – это наиболее гибкий способ, но он требует отдельного определения каждого элемента базы данных.

Создание пустой базы данных без помощи мастера

1. Нажмите кнопку **Создать** на панели инструментов.

2. В области задач **Создание файла** выберите в группе **Создание** ссылку **Новая база данных**.

3. В диалоговом окне **Файл новой базы данных** введите имя базы данных и укажите ее расположение, а затем нажмите кнопку **Создать**.

После открытия рабочей области можно создать требуемые объекты базы данных.

Последующие действия

После создания собственной базы данных может потребоваться выполнить следующие действия.

- Добавить данные в базу данных.
- Импортировать или связать данные с источником данных.

Импорт данных из другого источника

В Access можно импортировать данные из другой программы и поместить эти данные в новую таблицу или добавить в существующую. При совместной работе с пользователями, которые хранят данные в других программах, их данные тоже можно использовать в Access, задав связи. В обоих случаях работа с данными из других источников в Access не представляет сложности. Импортировать данные можно из листа Excel, таблицы в другой базе данных Access и других источников. Процесс импорта для различных источников немного различается, однако всегда начинается с приведенной ниже процедуры.

1. В приложении Access на вкладке **Внешние данные** в группе **Импорт и связывание** выберите команду для типа файла, который необходимо импортировать.

Например, чтобы импортировать данные из таблицы Excel, нажмите кнопку **Excel**. Если нужного типа программы в группе нет, нажмите кнопку **Дополнительно**.

2. В диалоговом окне **Внешние данные** нажмите кнопку **Обзор**, чтобы найти файл данных источника, или введите в поле **Имя файла** полный путь к файлу данных источника.

3. Выберите нужный параметр в разделе **Укажите, когда и где сохранять данные в текущей базе данных** (все программы позволяют импортировать данные, а некоторые поддерживают также добавление и связывание данных). Можно использовать импортируемые данные для создания новой таблицы или добавить их (в некоторых программах) в существующую таблицу. Также можно создать связанную таблицу, содержащую ссылку на данные в исходной программе.

4. Если будет запущен мастер, следуйте инструкциям на экране. На последней странице нажмите кнопку **Готово**.

При импорте объектов или связывании таблиц из базы данных Access открывается диалоговое окно **Импорт объектов** или **Связь с таблицами**. Выберите нужные элементы и нажмите кнопку **ОК**.

Точная последовательность действий зависит от выбранного способа обработки данных: импорт, добавление или связывание.

5. Появится предложение сохранить сведения о только что завершенной операции импорта. Если эту операцию импорта в будущем планируется выполнять снова, нажмите кнопку **Сохранить шаги импорта** и введите подробные сведения. Позже для повторения этой операции достаточно будет нажать кнопку **Сохраненные операции импорта** на вкладке **Внешние данные** в группе **Импорт и связывание**. Если сохранять сведения об операции не нужно, нажмите кнопку **Закрыть**.

Если импортируется таблица, программа Access импортирует данные в новую таблицу и отображает эту таблицу в группе **Таблицы** в области навигации. Если выбрано добавление данных к существующей таблице, данные добавляются к этой таблице. Если выбрано связывание данных, в группе **Таблицы** в области навигации создается связанная таблица.

13.2 Типы данных таблиц

При создании таблиц в приложениях Access, необходимо выбрать тип данных для каждого столбца данных. Наименование поля используется для ссылки на данные таблицы. Для определения типа хранимых данных используется тип данных. Тип данных поля вводится в поле ввода столбца **Тип данных**. В следующей таблице 18 показаны типы данных, доступные для приложения Access.

Таблица 18. Типы данных в приложениях Access

Тип данных	**Параметр свойства подтипа**	**Описание**
Счетчик	отсутствует	Уникальное значение, создаваемое Access для каждой новой записи.
Короткий текст	отсутствует	Буквенно-цифровые данные, состоящие из 1–4000 символов (ограничение по умолчанию –255 символов).

Продолжение таблицы 18

Тип данных	**Параметр свойства подтипа**	**Описание**
Длинный текст	отсутствует	Буквенно-цифровые данные объемом не более 2^30–1 байт.
Числовой	Целое число (без десятичных знаков).	Числовые данные.
Числовой	Число с плавающей запятой (переменное количество десятичных знаков).	Числовые данные.
Числовой	Число с фиксированной запятой (6 десятичных знаков).	Числовые данные.
Дата/время	Дата	Значения дат.
Дата и время	Время	Значения времени.
Дата и время	Дата и время	Значения даты и времени.
Денежный	отсутствует	Денежные данные.
Логический	отсутствует	Логические данные (да/нет).
Гиперссылка	отсутствует	Адрес ссылки на документ или файл в Интернете или интрасети.
Изображение	отсутствует	Цифровые изображения.
Вычисляемый	отсутствует	Результат выражения, созданного с использованием данных из одного или нескольких полей таблицы.
Мастер подстановок	Список значений	Использует содержимое списка значений для проверки содержимого поля.
Мастер подстановок	Другие таблица или запрос	Использует поле идентификатора другой таблицы или запроса для проверки содержимого поля.

Свойства полей Access

Каждое поле отношения имеет определенные свойства, которые определяют состав, структуру поля и в дальнейшем содержание значений данного поля, ниже приведена таблица, которая содержит описание основных свойств полей (таблица 19).

Таблица 19. Свойства полей Access

Свойства поля	**Описание**
Размер поля	Определяет максимальную длину текстового или числового поля, так как если размер подобран неоптимальное, расходуется, лишня память.
Формат поля	Устанавливает формат отображения данных в форме, запросе, отчете.
Число десятичных знаков	Количество знаков после запятой в десятичном числе.
Маска ввода	Задает маску (шаблон), при вводе данных в таблицу или форму.
Значение по умолчанию	Содержит значение, установленное по умолчанию, для всех новых записей таблицы.
Подпись	Задает подпись поля, которое выводиться в формах, отчетах, таблицах (не путать с именем поля).
Условие на значение	Позволяет задать то условие, которое проверяется при вводе данных в поле.
Сообщение об ошибке	Задается текст, сообщение выводится в диалоговом окне, если вводимые данные не соответствуют, заданному условию на значение.
Обязательное поле	Определяет, может ли поле быть пустым или нет.
Пустые строки	Определяет возможность ввода в поля пустых строк с пробелами.
Индексированное поле	Задает индексы, для ускоренного поиска информации в таблице.

13.3 Создание списка подстановки

Столбец (или поле) подстановок – это поле в таблице, значение которого загружается из другой таблицы или из списка значений. Столбец подстановок можно использовать для отображения списка выбора в списке или поле со списком. Значения могут быть взяты из таблицы, запроса или введены пользователем. Столбец подстановок можно создать вручную, задав полю свойства поля подстановок, или автоматически путем заполнения полей мастера подстановок. По возможности следует использовать мастера подстановок для создания столбца подстановок. Мастер подстановок упрощает процесс и автоматически заполняет соответствующие свойства поля, а также создает соответствующие связи между таблицами.

Мастер подстановок запускается в следующих случаях: при создании столбца подстановок в режиме таблицы, при перетаскивании поля из области Список полей в таблицу, открытую в режиме таблицы, и в режиме конструктора при выборе пункта Мастер подстановок в столбце Тип данных. Этот мастер

помогает пройти все шаги, необходимые для создания столбца подстановок, и автоматически задает свойства полей, соответствующие выбору пользователя. Мастер также создает связи между таблицами и (при необходимости) индексы для поддержки столбца подстановок.

Значения для раскрывающегося списка могут поступать из определенной таблицы или запроса, либо список создается специально для этой цели.

Раскрывающийся список для таблицы создается следующим образом:

1. В окне конструктора таблиц для необходимого поля выберите тип данных «Мастер подстановок»;

2. На первом шаге выберите один из переключателей: «Будет введен фиксированный набор значений» либо «Объект «столбец подстановки» будет использовать данные из таблицы или запроса»;

3. Задайте количество столбцов раскрывающегося списка (по умолчанию – 1). Введите значения списка.

Внести изменения в раскрывающийся список можно следующим образом:

1. Открыть таблицу в режиме конструктора и перезапустить «Мастер подстановок»;

2. Отредактировать свойства поля подстановки на вкладке «Подстановка» в нижней части окна конструктора.

13.4 Создание значений по умолчанию и маски ввода

Если в базу данных вносят данные несколько человек, можно указать, как следует вводить их в те или иные поля. Это позволит обеспечить целостность и упростить управление базой данных. Например, можно задать для формы маску ввода, чтобы пользователи могли вводить номера телефонов только в шведском формате или адреса во французском. Для маски ввода можно задать один формат, а для отображения этих же данных – другой.

Маски ввода выполняют значительный объем работы по проверке данных, предотвращая ввод пользователями неверных данных (например, номера телефона в поле даты). Маски ввода также позволяют обеспечить единообразный вид вводимых данных, что упрощает поиск сведений и управление базой данных.

Три компонента маски ввода

Маски ввода состоят из одного обязательного и двух необязательных компонентов, разделенных точками с запятой. Назначение каждого из компонентов описано ниже.

Первый компонент является обязательным. Он представляет собой знак или строку (последовательность знаков) маски с заполнителями и литералами, например круглыми скобками, точками и дефисами.

Второй компонент не является обязательным и определяет способ хранения встроенных знаков маски в поле. Если для этого компонента задано значение **0**, знаки сохраняются вместе с данными, а если **1**, то знаки отображаются

без сохранения. Выбрав значение **1**, можно сэкономить место для хранения базы данных.

Третий компонент маски ввода также не является обязательным и определяет знак, используемый в качестве заполнителя. По умолчанию в Access используется знак подчеркивания (_). Чтобы задать другой знак, введите его в третьем компоненте маски.

– В маске используются два заполнителя – 9 и 0. Заполнитель 9 обозначает необязательные цифры (код города можно не вводить), а 0 – обязательные.

– Значение 0 во втором компоненте маски ввода указывает на то, что знаки маски следует хранить вместе с данными.

– Третий компонент маски ввода указывает на то, что вместо знака подчеркивания (_) в качестве заполнителя будет использоваться дефис (**–**).

– Знаки, определяющие маски ввода

В приведенной ниже таблице 20 перечислены заполнители и литералы, используемые в масках ввода, и описано их влияние на ввод данных.

Таблица 20. Заполнители и литералы, используемые в масках ввода

ЗНАК	**ОПИСАНИЕ**
0	Пользователь должен ввести цифру (от 0 до 9).
9	Пользователь может ввести цифру (от 0 до 9).
#	Пользователь может ввести цифру, пробел, знак "плюс" или "минус". Если ничего не ввести, будет вставлен пробел.
L	Пользователь должен ввести букву.
?	Пользователь может ввести букву.
A	Пользователь должен ввести букву или цифру.
a	Пользователь может ввести букву или цифру.
&	Пользователь должен ввести какой–либо знак или пробел.
C	Пользователь может ввести знаки или пробелы.
. , : ; – /	Разделитель целой и дробной части, групп разрядов, значений дат и времени. Выбираемый знак зависит от региональных параметров Windows.
>	Все последующие знаки будут переведены в верхний регистр.
<	Все последующие знаки будут переведены в нижний регистр.
!	Маска ввода заполняется слева направо, а не справа налево.
\	Знаки, следующие непосредственно за обратной косой чертой, отображаются без изменений.
""	Знаки, заключенные в двойные кавычки, отображаются без изменений.

Создание маски ввода

Чтобы быстро добавить маску ввода, можно воспользоваться мастером или задать ее вручную, введя для свойства поля "Маска ввода" настраиваемую маску.

Добавление масок ввода с помощью мастера

Далее описано, как с помощью мастера масок ввода добавить предопределенную маску ввода к полю таблицы, запросу либо элементу управления формы или отчета.

Добавление маски ввода в поле таблицы

Маски ввода можно использовать для полей с типом данных "Текстовый", "Числовой" (кроме кода репликации), "Денежный" и "Дата/время".

Если применить маску ввода к полю типа "Дата/время", для этого поля невозможно будет использовать элемент управления **Выбор даты**.

1. В области навигации щелкните таблицу правой кнопкой мыши и выберите в контекстном меню команду **Конструктор**.
2. Выберите поле, к которому необходимо применить маску ввода.
3. В разделе **Свойства поля** на вкладке **Общие** щелкните поле свойства **Маска ввода**.
4. Чтобы запустить мастер масок ввода, нажмите кнопку **Построить** ....
5. Выберите нужный тип маски ввода из списка.

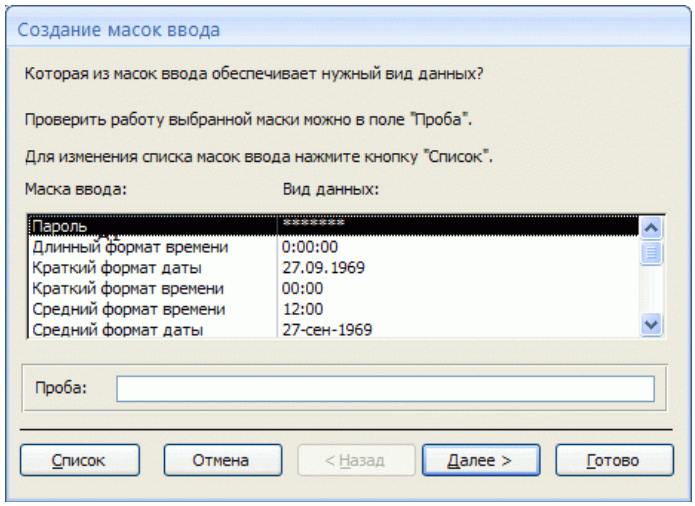


Рисунок 26 - Выбор типа маски

6. Щелкните поле **Проба** и введите данные, чтобы проверить работу маски.
7. Чтобы оставить маску ввода без изменений, нажмите кнопку **Далее**.
8. Выберите способ хранения данных.
9. Чтобы сохранить изменения, нажмите кнопку **Готово**.

Добавление маски ввода в запрос

1. В области переходов щелкните правой кнопкой мыши запрос, который требуется изменить, и выберите в контекстном меню команду **Конструктор**.
2. В бланке запроса установите курсор в столбец поля, которое требуется изменить.

Курсор можно поместить в любую строку для этого поля.

3. Нажмите клавишу F4, чтобы открыть окно свойств поля.

4. В разделе **Свойства поля** на вкладке **Общие** щелкните поле свойства **Маска ввода**.

5. Чтобы запустить мастер масок ввода, нажмите кнопку **Построить** ... Следуйте инструкциям мастера.

Добавление маски ввода в элемент управления формы или отчета

1. В области переходов щелкните правой кнопкой мыши форму или отчет, которые требуется изменить, и выберите в контекстном меню команду **Конструктор**.

2. Щелкните правой кнопкой мыши элемент управления, который требуется изменить, а затем выберите в контекстном меню команду **Свойства**.

3. На вкладке **Все** щелкните поле свойства **Маска ввода**.

4. Чтобы запустить мастер масок ввода, нажмите кнопку **Построить** ... Следуйте инструкциям мастера.

Создание настраиваемых масок ввода

В мастере предусмотрены маски ввода для наиболее распространенных вариантов форматирования, однако в некоторых случаях требуется настроить маску под особые нужды. Чтобы сделать это, можно либо изменить одну из предопределенных масок мастера ввода, либо вручную задать свойство "Маска ввода" для соответствующего поля.

Настройка масок ввода в мастере

1. Откройте объект в конструкторе и щелкните поле, к которому нужно добавить настраиваемую маску ввода.

2. Чтобы запустить мастер масок ввода, нажмите кнопку **Построить** ...

3. Нажмите кнопку **Изменить список**.

Откроется диалоговое окно **Настройка масок ввода**.

4. Введите новое описание в поле **Описание**, используя знаки и заполнители из таблицы.

5. Щелкните стрелку списка **Тип маски:** и выберите подходящий тип маски.

6. Нажмите кнопку **Закрыть**. В списке появится новая маска ввода.

Настройка масок ввода в свойстве поля

1. В области навигации щелкните объект правой кнопкой мыши и выберите в контекстном меню команду **Конструктор**.

2. Выберите поле, для которого необходимо создать настраиваемую маску ввода.

3. В области "Свойства поля" щелкните поле **Маска ввода** и введите собственную маску.

4. Чтобы сохранить изменения, нажмите сочетание клавиш CTRL+S.

Для получения дополнительных сведений о том, как определить маску ввода, щелкните поле свойства **Маска ввода** и нажмите клавишу F1. Для полей с типом данных "Числовой" и "Денежный" определение маски ввода необходимо задавать вручную.

14 РЕДАКТИРОВАНИЕ ТАБЛИЦ, СВЯЗИ

После создания таблицы для каждого объекта в базе данных нужно предоставить Office Access средства, с помощью которых эти данные возвращаются, когда это необходимо. Это осуществляется с помощью добавления общих полей в связанные таблицы и определения межтабличных связей. После этого можно создавать запросы, формы и отчеты, отображающие сведения одновременно из нескольких таблиц.

Создать связь между таблицами можно с помощью окна «Связи» или с помощью перетаскивания поля из области Список полей в таблицу. При создании связи между таблицами общие поля могут иметь различные имена, однако часто они имеют одинаковые. Очевидно, что общие поля должны иметь одинаковый тип данных. Однако, если поле первичного ключа имеет тип «Счетчик», поле внешнего ключа может также быть числовым полем, если свойство Размер поля (FieldSize) обоих полей совпадает. Например, можно сопоставить поля с типами «Счетчик» и «Числовой», если свойство Размер поля (FieldSize) обоих полей имеет значение «Длинное целое». Если оба общих поля являются числовыми, у них должно совпадать значение свойства Размер поля (FieldSize).

14.1 Определение отношений между таблицами, связи

Связь между таблицами устанавливается автоматически в следующих случаях:

– если при создании таблиц используется Мастер баз данных или Мастер таблиц;

– когда с помощью Мастера подстановок создается раскрывающийся список со значениями поля другой таблицы.

Установление связи. Связать таблицы самостоятельно можно в окне «Схем данных» (команда Сервис – Схема данных):

Создание связи между таблицами с помощью вкладки «Схема данных»

1. В диалоговом окне Открыть выберите и откройте базу данных.

2. На вкладке Работа с базами данных в группе Отображение выберите пункт Схема данных.

3. Если ни одной связи еще не определено, автоматически откроется диалоговое окно Добавить таблицу. Если окно не открылось, на вкладке Структура в группе Связи нажмите кнопку Добавить таблицу.

а. В диалоговом окне Добавить таблицу отображены все таблицы и запросы, содержащиеся в базе данных. Чтобы отобразить только таблицы, выберите пункт Таблицы. Чтобы отобразить только запросы, выберите пункт Запросы. Чтобы отобразить и таблицы и запросы, выберите пункт Таблицы и запросы.

4. Выберите одну или несколько таблиц или запросов и нажмите кнопку Добавить. После добавления таблиц и запросов на вкладку «Схема данных» нажмите кнопку Закрыть.

5. Перетащите поле (как правило, поле первичного ключа) из одной таблицы на общее поле (поле внешнего ключа) в другой таблице. Чтобы перетащить сразу несколько полей, нажмите клавишу CTRL и, удерживая ее, выберите каждое поле.

а. Откроется диалоговое окно Изменение связей (рисунок 27).

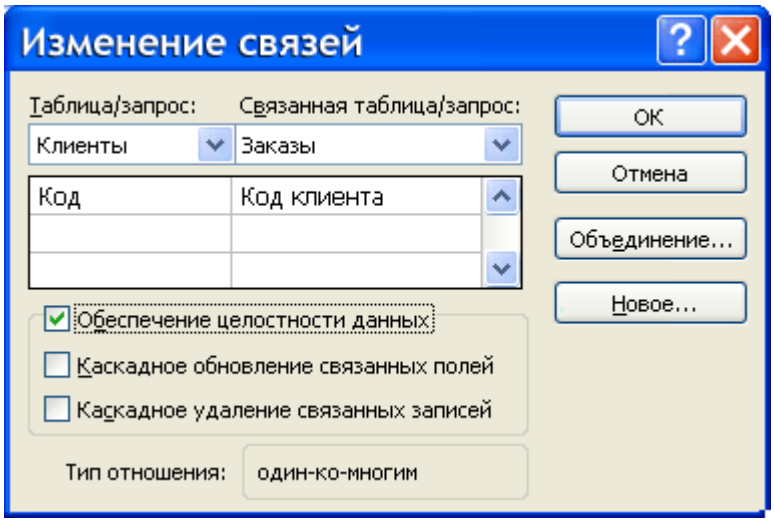


Рисунок 27 – Изменение связей

6. Убедитесь, что поля, имена которых отображены в диалоговом окне, являются общими полями данной связи. Если имя поля неверно, выделите это имя и выберите верное поле из списка.

Для поддержания целостности данных для этой связи установите флажок Обеспечение целостности данных.

7. Нажмите кнопку Создать.

Чтобы создать отношение «один-к-одному» Оба общих поля (как правило, поля первичного ключа и внешнего ключа) должны иметь уникальный индекс. Это означает, что свойства Индексированное (Indexed) этих полей должны иметь значения Да (Совпадения не допускаются). Если оба поля имеют уникальный индекс, Access создаст отношение «один-к-одному».

Чтобы создать отношение «один ко многим» Поле на одной стороне отношения (как правило, поле первичного ключа) должно иметь уникальный индекс. Это означает, что свойство Индексировано (Indexed) этого полей должны иметь значения Да (Совпадения не допускаются). Поле на стороне «многие» не должно иметь уникального индекса. У этого поля может быть индекс, но он должен допускать совпадения. Это означает, что свойство Индексировано (Indexed) этого поля должно иметь значение Нет либо Да (Допускаются совпадения). Если одно поле имеет уникальный индекс, а другое – нет, Access создаст отношение «один ко многим».

Access проведет линию связи между двумя таблицами. Если был установлен флажок Обеспечение целостности данных, линия будет толще на каж-

дом конце. Кроме того, если был установлен флажок Обеспечение целостности данных, над широкой частью на одном конце линии связи будет отображено число 1, а над широкой частью на другом конце линии – символ бесконечности (∞), как показано на следующем рисунке.

Создание связи между таблицами с помощью области «Список полей»

В Office Access можно добавить поле в существующую таблицу, открытую в режиме таблицы, с помощью перетаскивания этого поля из области Список полей. Область Список полей отображает поля, доступные в связанных таблицах, а также поля, доступные в других таблицах базы данных. При перетаскивании поля из «другой» (несвязанной) таблицы и выполнении инструкций мастера подстановок автоматически создается новое отношение «один ко многим» между таблицей в области Список полей и текущей таблицей. Эта созданная Access связь не поддерживает целостность данных по умолчанию. Для поддержания целостности данных следует изменить связь.

Откройте таблицу в режиме таблицы

1. В диалоговом окне «Открыть» выберите и откройте базу данных.

2. В области переходов щелкните правой кнопкой мыши таблицу, к которой нужно добавить поле и создать связь, а затем выберите в контекстном меню команду Режим таблицы.

Открытие области «Список полей» (рисунок 28)

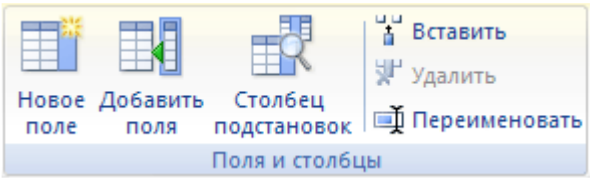


Рисунок 28 - Рабочая область «Список полей»

Появится область Список полей.

В области Список полей отображены все другие таблицы базы данных, сгруппированные по категориям. При работе с таблицей в режиме таблицы в области Список полей отображаются поля в одной их двух категорий: Доступные поля в связанной таблице и Доступные поля в другой таблице. К первой категории относятся все поля, связанные с текущей таблицей. Во второй категории перечислены все таблицы, с которыми не связана данная таблица.

Чтобы просмотреть список всех полей таблицы, щелкните знак плюс (+) рядом с именем таблицы в области Список полей. Чтобы добавить поле в таблицу, перетащите нужное поле из области Список полей в таблицу в режиме таблицы.

Добавление поля и создание связи из области «Список полей»

На вкладке Таблица в группе Поля и столбцы щелкните Добавить существующие поля (рисунок 28).

Появится область Список полей.

1. Чтобы отобразить список полей в таблице, в группе Доступные поля в другой таблице щелкните знак плюс (+) рядом с именем таблицы.

2. Перетащите нужное поле из области Список полей в таблицу, открытую в режиме таблицы.

3. Когда появится линия вставки, поместите поле в выбранное место.

Будет запущен мастер подстановок.

4. Следуйте инструкциям мастера подстановок для добавления поля.

Поле будет отображено в таблице в режиме таблицы.

При перетаскивании поля из «другой» (несвязанной) таблицы и выполнении инструкций мастера подстановок автоматически создается новое отношение «один ко многим» между таблицей в области Список полей и текущей таблицей. Эта созданная Access связь не поддерживает целостность данных по умолчанию. Для поддержания целостности данных следует изменить связь.

Изменение связи между таблицами

Чтобы изменить связь между таблицами следует выбрать ее на вкладке «Схема данных», а затем изменить эту связь.

1. Установите указатель на линии связи и щелкните линию, чтобы выделить ее.

При выделении линия связи становится толще.

2. Дважды щелкните выделенную линию связи.

–или–

На вкладке Конструктор в группе Инструменты выберите команду Изменение связей.

Откроется диалоговое окно Изменение связей.

Открытие диалогового окна «Изменение связей»

1. В диалоговом окне Открыть выберите и откройте базу данных.

2. На вкладке Работа с базами данных в группе Отображение выберите пункт Связи.

Откроется вкладка «Схема данных».

Если ни одной связи еще не определено и это первое открытие вкладки «Схема данных», откроется диалоговое окно Добавление таблицы. Если диалоговое окно открылось, нажмите кнопку Закрыть.

3. На вкладке Конструктор в группе Связи нажмите кнопку Все связи.

Будет отображены все таблицы со связями вместе с линиями связи. Обратите внимание, что скрытые таблицы (таблицы, у которых установлен флажок скрытый в диалоговом окне Свойства) и их связи не будут отображены, если не выбран параметр «Показывать скрытые объекты» в диалоговом окне Параметры переходов.

4. Щелкните линию связи, которую требуется изменить. При выделении линия связи становится толще.

5. Дважды щелкните линию связи.

–или–

Откроется диалоговое окно Изменение связей (рисунок 27).

6. Внесите изменения и нажмите кнопку ОК.

С помощью диалогового окна Изменение связей можно изменять связь между таблицами. Например, изменить таблицы или запросы, а также поля на любой стороне отношения. Можно также задать тип объединения или включить обеспечение целостности данных и выбрать каскадные параметры.

Удаление связи между таблицами

Чтобы удалить связь между таблицами следует удалить линию связи на вкладке «Схема данных». Осторожно установите указатель на линии связи и щелкните линию, чтобы выделить ее. При выделении линия связи становится толще. При выделенной линии связи нажмите клавишу DEL. Обратите внимание, что при удалении связи также отключается обеспечение целостности данных для этой связи, если оно было включено. В результате Access больше не будет автоматически предотвращать создание изолированных записей на стороне «многие» отношения.

Нажмите кнопку Microsoft Office, а затем выберите команду Открыть.

1. В диалоговом окне Открыть выберите и откройте базу данных.

2. На вкладке Работа с базами данных в группе Отображение выберите Связи.

Откроется вкладка «Схема данных».

Если ни одной связи еще не определено и это первое открытие вкладки «Схема данных», откроется диалоговое окно Добавление таблицы. Если диалоговое окно открылось, нажмите кнопку Закрыть.

На вкладке Конструктор в группе Связи щелкните Все связи.

Будет отображены все таблицы со связями вместе с линиями связи. Обратите внимание, что скрытые таблицы (таблицы, у которых установлен флажок скрытый в диалоговом окне Свойства) и их связи не будут отображены, если не выбран параметр «Показывать скрытые объекты» в диалоговом окне Параметры переходов.

3. Щелкните линию связи, которую требуется удалить. При выделении линия связи становится толще.

4. Нажмите клавишу DEL.

–или–

Щелкните правой кнопкой мыши и выберите команду Удалить.

5. Может быть отображено сообщение Подтвердите удаление выделенной связи из базы данных. Если сообщение о подтверждение отображено, нажмите кнопку Да.

Если одна из таблиц, участвующих в связи между таблицами используется в текущий момент – возможно, третьим лицом или процессом, либо в открытом объекте базы данных, например в форме, – то удалить связь между таблицами будет невозможно. Перед попыткой удаления связи между таблицами следует закрыть все открытые объекты, использующие эти таблицы.

14.2 Определение параметров целостности данных и типа объединения

Обеспечение целостности данных

Целью обеспечения целостности данных является предотвращение появления изолированных записей и поддержание синхронизации ссылок таким образом, чтобы не могли появиться записи, ссылающиеся на несуществующие записи. Целостность данных обеспечивается включением обеспечения целостности данных для связи между таблицами. После наложения Access отменяет все действия, которые могут нарушить целостность данных для этой связи между таблицами. Это означает, что Access отменит и обновления, изменяющие назначение связи, и удаления назначения связи.

Включение и отключение обеспечения целостности данных

1. В диалоговом окне Открыть выберите и откройте базу данных.

Откроется вкладка «Схема данных».

Если ни одной связи еще не определено и это первое открытие вкладки «Схема данных», откроется диалоговое окно Добавление таблицы. Если диалоговое окно открылось, нажмите кнопку Закрыть.

Будет отображены все таблицы со связями вместе с линиями связи. Обратите внимание, что скрытые таблицы (таблицы, у которых установлен флажок скрытый в диалоговом окне Свойства) и их связи не будут отображены, если не выбран параметр «Показывать скрытые объекты» в диалоговом окне Параметры переходов.

2. Щелкните линию связи, которую требуется изменить. При выделении линия связи становится толще.

3. Дважды щелкните линию связи.

–или–

Откроется диалоговое окно Изменение связей.

4. Выберите необходимое значение параметра Обеспечение целостности данных.

5. Внесите дополнительные изменения в связь и нажмите кнопку ОК.

После включения обеспечения целостности данных будут применены следующие правила.

– Не допускается ввод в поле внешнего ключа связанной таблицы значение, не содержащееся в поле первичного ключа главной таблицы – это приведет к созданию изолированных записей.

– Не допускается удаление записи из главной таблицы, если в подчиненной таблице существуют связанные с ней записи. Например, невозможно удалить запись из таблицы «Сотрудники», если в таблице «Заказы» имеются заказы, относящиеся к данному сотруднику. Тем не менее, можно удалить главную запись и все связанные записи одним действием, установив флажок Каскадное удаление связанных записей.

– Не допускается изменение значения первичного ключа в главной таблице, если это повлечет создание изолированных записей. Например, нельзя

изменить порядковый номер в таблице «Заказы», если в таблице «Заказано» имеются элементы строк, относящиеся к этому заказу. Тем не менее, можно обновить главную запись и все связанные записи одним действием: установив флажок «Каскадное обновление связанных полей».

Если при включении обеспечения целостности данных возникли трудности, обратите внимание, что должны быть выполнены следующие условия.

– Общее поле главной таблицы должно быть первичным ключом и иметь уникальный индекс.

– Общие поля должны иметь одинаковый тип данных. Единственным исключением является то, что поле с типом «Счетчик» можно связать с полем «Числовой», если свойство Размер поля (FieldSize) этого поля имеет значение Длинное целое.

– Обе таблицы существуют в одной базе данных Access. Обеспечение целостности данных не может быть включено для связанных таблиц. Однако, если исходные таблицы имеют формат Microsoft Access, то можно открыть базу данных, в которой они хранятся и включить обеспечение целостности данных в этой базе данных.

Задание каскадных параметров

Возможно возникновение ситуации, когда требуется изменить значение только на стороне «один» отношения. В этом случае необходимо, чтобы Access автоматически обновил все затронутые строки как часть одного действия. Таким образом обновление будет полностью завершено, а база данных не будет находиться в несовместном состоянии – когда некоторые строки обновлены, а некоторые – нет. Access помогает избежать данной проблемы с помощью поддержки параметра «Каскадное обновление связанных полей». Если при включении обеспечения целостности данных был включен параметр «Каскадное обновление связанных полей», то при последующем обновлении первичного ключа Access автоматически обновит все поля, связанные с первичным ключом.

Также может потребоваться удаление строки и всех связанных с ней записей – например запись в таблице «Поставщики» и все связанные с этим поставщиком заказы. Для этого в Access поддерживается параметр «Каскадное удаление связанных записей». Если включить обеспечение целостности данных и устранить флажка Каскадное удаление связанных записей Access будет автоматически удалять все записи, связанные с первичным ключом при удалении записи, содержащей первичный ключ.

Включение и отключение каскадного обновления и/или каскадного удаления

1. В диалоговом окне Открыть выберите и откройте базу данных.

На вкладке Средства базы данных в группе Скрыть/Отобразить выберите команду Отношения.

Откроется вкладка «Схема данных».

Если ни одной связи еще не определено и это первое открытие вкладки «Схема данных», откроется диалоговое окно Добавление таблицы. Если диалоговое окно открылось, нажмите кнопку Закрыть.

Будет отображены все таблицы со связями вместе с линиями связи. Обратите внимание, что скрытые таблицы (таблицы, у которых установлен флажок скрытый в диалоговом окне Свойства) и их связи не будут отображены, если не выбран параметр «Показывать скрытые объекты» в диалоговом окне Параметры переходов.

2. Щелкните линию связи, которую требуется изменить. При выделении линия связи становится толще.

3. Дважды щелкните линию связи.

–или–

На вкладке Конструктор в группе Сервис щелкните Изменение связей.

Откроется диалоговое окно Изменение связей.

4. Установите флажок Обеспечение целостности данных.

5. Установите флажок Каскадное обновление связанных полей или Каскадное удаление связанных записей, или установите оба.

6. Внесите дополнительные изменения в связь и нажмите кнопку ОК.

Если первичным ключом является поле Тип данных "Счетчик", установка флажка Каскадное обновление связанных полей не приведет к каким-либо результатам, поскольку изменить значение поля счетчика невозможно.

Задание типа объединения

При определении связи между таблицами сведения о связи влияют на структуру запросов. Например, при определении связи между двумя таблицами и создании запроса, работающего с этими двумя таблицами, в Access автоматически выбираются сопоставляемые поля по умолчанию на основе полей, указанных в связи. Эти исходные значения в запросе можно изменить, но часто значения, определенные связью между таблицами являются верными. Так как сопоставление и объединение данных из двух таблиц являются часто воспроизводимыми действиями во всех базах данных, параметры по умолчанию, определенные межтабличными связями, могут быть полезны и экономить время.

Используя запрос по нескольким таблицам можно комбинировать данные из нескольких таблиц путем сопоставления значений в общих полях. Действие сопоставления и комбинирования называется объединением.

Одно из значений, которое можно задать для каждой связи – это тип объединения. Тип объединения определяет, какие записи будут включены в результаты запроса.

Следует определить, какие результаты наиболее часто требуются от запроса, объединяющего таблицы в конкретной связи, и в соответствии с этим выбрать тип объединения.

Задание типа объединения

1. В диалоговом окне Изменение связей нажмите кнопку Тип объединения.

Откроется диалоговое окно Параметры объединения.

2. Выберите нужные параметры и нажмите кнопку ОК.

Внесение изменений в окне «Параметры объединения»

1. В диалоговом окне Открыть выберите и откройте базу данных.

Откроется вкладка «Схема данных».

Если ни одной связи еще не определено и это первое открытие вкладки «Схема данных», откроется диалоговое окно Добавление таблицы. Если диалоговое окно открылось, нажмите кнопку Закрыть.

Будет отображены все таблицы со связями вместе с линиями связи. Обратите внимание, что скрытые таблицы (таблицы, у которых установлен флажок скрытый в диалоговом окне Свойства) и их связи не будут отображены, если не выбран параметр «Показывать скрытые объекты» в диалоговом окне Параметры переходов.

2. Щелкните линию связи, которую требуется изменить. При выделении линия связи становится толще.

3. Дважды щелкните линию связи.

–или–

Откроется диалоговое окно Изменение связей.

4. Нажмите кнопку Тип объединения

5. В диалоговом окне Параметры объединения выберите нужные параметры и нажмите кнопку ОК (рисунок 29).

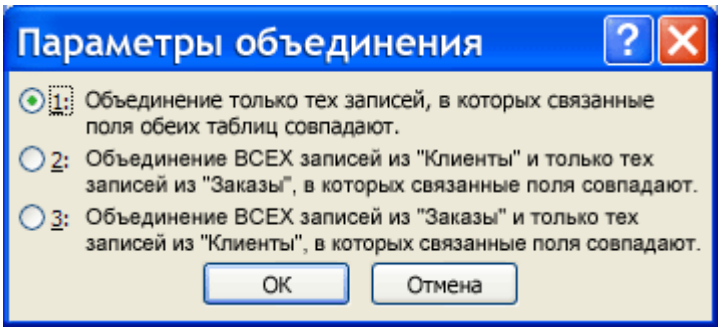


Рисунок 29 – Параметры объединения

6. Внесите дополнительные изменения в связь и нажмите кнопку ОК.

14.3 Перемещение по таблице и просмотр записей. Подтаблицы

Кроме обычных средств перемещения по таблицам (полосы прокрутки, кнопки перехода по записям, и т.д.) существует панель инструментов «Формат». Для открытия этой панели щелкните правой кнопкой на любой панели инструментов и в контекстном меню выберите «Формат».

При большом количестве полей в таблице перейти к нужному полю можно с помощью команды «Перейти к полю» панели инструментов «Формат».

Чтобы просмотреть связанные записи в подтаблице, можно щелкнуть на кнопке со знаком «+». Можно развернуть все подтаблицы сразу. Для этого необходимо выбрать команду ***Формат – Подтаблица – Развернуть все***.

Добавление новой подтаблицы. Если между двумя таблицами не определено никакой связи, к таблице можно добавить подтаблицу, если есть общее поле.

Удаление подтаблицы. Для удаления подтаблицы выберите команду ***Формат – Подтаблица – Удалить***.

Поиск и замена. В Access существует возможность сразу изменить значение нескольких полей. Эти действия можно произвести стандартным для приложений Ms Office способом – с помощью команды меню «Правка» – «Найти». При этом необходимо выбрать один из параметров совпадения в списке «Совпадение».

14.4 Редактирование специальных полей

Специальными являются поля следующих типов:

- Поле MEMO;
- Гиперссылка;
- Поле объекта OLE.
- Дата/время;
- Логическое;

Поле MEMO

Вводить данные в поля этого типа можно непосредственно с клавиатуры. Однако удобнее воспользоваться окном «Область ввода», в котором можно видеть текст целиком. Открывается окно «Область ввода» следующим образом: перейдите в поле типа MEMO и нажмите клавиши Shift+F2. Откроется окно «Область ввода».

Примечание. Чтобы в поле MEMO перейти на новую строку, используйте сочетание клавиш Ctrl+Enter, т.к. при нажатии клавиши Enter окно «Область ввода» закрывается.

Гиперссылка

Поле гиперссылки – это специальное поле, содержащее адрес гиперссылки, используемой Access для перехода к указанному в адресе объекту базы данных, документу Office или странице в Web.

Адрес гиперссылки может состоять максимум из трех частей:

- необязательный текст;
- адрес, который представляет собой путь к объекту перехода;
- необязательный дополнительный адрес (например, закладка в документе Word или объект в базе данных).

Существует несколько способов ввода адреса в поле гиперссылки:

- кнопка «Добавление гиперссылки» на панели инструментов «Таблица»;
- перетаскивание гиперссылки;
- ввод адреса с клавиатуры.

Объекты OLE

Поля типа «Поле объекта OLE» содержат такие данные, как рисунки, документы WORD, звукозаписи или объекты иного формата, созданные в других приложениях. Вы можете создать:

– связанный объект, который содержит ссылку на объект OLE в таблице Access. При редактировании или просмотре этого объекта открывается приложение, в котором был создан объект;

– внедренный объект, который содержит копию объекта OLE в таблице. Изменения, вносимые в исходный объект, не отражаются на объекте, хранящемся в базе данных.

Вставка связанного или внедренного объекта OLE:

1. в режиме таблицы установите курсор в поле той записи, которую вы редактируете;

2. в контекстном меню выберите команду «Добавить объект» или выберите команду меню «Вставка» – «Объект». Откроется диалоговое окно «Вставка объекта»;

3. установите переключатель «Создать из файла». С помощью кнопки «Обзор» выберите необходимый для связи файл;

4. установите флажок «Связь». Если вы хотите хранить копию объекта в таблице, необходимо снять этот флажок;

5. чтобы объект в поле отображался в виде значка, установите флажок «В виде значка», в противном случае в поле будет отображаться название типа объекта.

Тип данных «Дата/время»

В Microsoft Access можно оставить формат, установленный по умолчанию, использовать встроенный формат или создать пользовательский формат.

Приложение Access автоматически отображает дату и время в полном формате даты и длинном формате времени. Для английского языка (США) используется формат даты мм/дд/гггг, а для русского – дд/мм/гггг, где мм – это месяц, дд – день и гггг – год. Время отображается в формате чч:мм:сс, где чч – это часы, мм – минуты и сс – секунды. Однако можно выбрать и другие форматы даты и времени.

Эти автоматические форматы даты и времени отличаются в зависимости от географического расположения, указанного в компоненте «Язык и региональные стандарты» Microsoft Windows на конкретном компьютере. Например, в Европе и многих странах Азии дата и время могут отображаться в формате 28.11.2006 12:07:12 PM или 28/11/2006 12:07:12 PM, а в США – 11/28/2006 12:07:12 PM. Эти автоматические настройки можно изменить при помощи пользовательских форматов отображения. Выбранный формат отображения не изменяет данные во время ввода и сохранения в Access. Например, можно ввести дату в европейском формате 28.11.2006, но настроить ее отображение в таблицах, формах или отчетах в виде 11/28/2006.

Access поддерживает частичную проверку форматирования даты и времени. Например, при попытке ввода неправильных дат, таких как 32.11.2006, появляется окно с предложением ввести новое значение или изменить тип данных поля с даты/времени на текст. Допустимые значения даты входят в диапазон от –657 434 (1 января 1100 г.) до 2 958 465 (31 декабря 9999 г.). Допустимые значения времени входят в диапазон от .0 до .9999, или 23:59:59.

Логическое поле

В Access в логическом поле может храниться только два значения: "Да" и "Нет". Если для отображения логического поля вы применяете текстовое поле, значение выводится как –1 для "Да" и 0 для "Нет". Такие значения не очень понятны для большинства пользователей, поэтому в Access предлагаются флажки, переключатели и выключатели, которые можно использовать для отображения и ввода значений "Да" и "Нет". Эти элементы управления обеспечивают графическое представление значений "Да" и "Нет", которое легко использовать и понимать.

14.5 Создание индексов

Создание и использование индекса для увеличения производительности

Если необходимо постоянно выполнять поиск в таблице или сортировать записи по определенному полю, можно ускорить эти операции, создав индекс для этого поля. В таблицах Microsoft Office Access индексы используются точно так же, как предметные или именные указатели в книгах. Чтобы найти данные, Office Access 2007 проверяет местоположение этих данных по индексу. В некоторых случаях, например при определении первичного ключа, Access формирует индекс автоматически. В других случаях самому пользователю может потребоваться создание индекса.

Индексы способствуют более быстрым поиску и сортировке записей в Microsoft Office Access 2007. В индексе хранится местоположение записей на основе одного или нескольких полей, которые были выбраны для индексирования. После того как Access получает сведения о местоположении данных, эти данные могут загружаться путем перемещения непосредственно в нужное местоположение. Благодаря этому использование индекса гораздо эффективнее просмотра всех записей для поиска необходимых данных.

Выбор полей для индексирования

Можно создавать индексы, основанные на одном или нескольких полях. В основном требуется индексировать поля, в которых часто осуществляется поиск, сортируемые поля и поля, объединенные с полями в других таблицах, что часто используется в запросах по нескольким таблицам. Индексы ускоряют поиск и построение запросов, однако они могут привести и к снижению производительности при добавлении или обновлении данных. При вводе данных в таблицу, содержащую один или несколько индексов, приложение Access должно обновлять индексы при каждом добавлении или изменении записи. Добавление записей с помощью запроса на добавление или с помощью импортирования записей также будет происходить более медленно, если таблица–получатель содержит индексы.

Индексирование полей с типом данных «Объект OLE» и «Вложение» невозможно. Индексировать другие поля следует в тех случаях, когда одновременно выполняются следующие условия:

– Типом данных поля является «Текстовый», «Поле Memo», «Числовой», «Дата/время», «Счетчик», «Денежный», «Логический» или «Гиперссылка».

– Предполагается поиск значений в поле.

– Предполагается сортировка значений в поле.

– Предполагается сортировка большого числа различных значений в поле. Если поле содержит много одинаковых значений, то применение индекса, возможно, незначительно ускорит выполнение запросов.

Составные индексы

Если предполагается частое выполнение одновременной сортировки или одновременного поиска в нескольких полях, можно создать для этих полей составной индекс.

При сортировке таблицы по составному индексу Access сначала выполняет сортировку по первому полю, определенному для данного индекса. Последовательность полей определяется при создании составного индекса. Если в первом поле содержатся записи с повторяющимися значениями, то выполняется сортировка по второму полю, определенному для данного индекса, и т. д.

В составной индекс можно включить до 10 полей.

Создание индекса

Перед созданием индекса необходимо решить, следует ли создать индекс для одного поля или составной индекс. Индекс для одного поля создается с помощью установки свойства Индексированное поле. В следующей таблице 21 приведены возможные параметры свойства Индексированное поле.

При создании уникального индекса невозможно ввести новое значение в определенном поле, если такое значение уже существует в том же поле другой записи. В Access уникальный индекс автоматически создается для первичных ключей, однако может понадобиться, чтобы создание значений, совпадающих со значениями в других полях, было невозможным. Например, можно создать уникальный индекс для поля, в котором содержатся серийные номера, чтобы двум продуктам не мог быть присвоен один и тот же серийный номер.

Таблица 21. Параметр свойства «Индексированное поле»

Параметр свойства «Индексированное поле»	**Значение**
Нет	Не создавать индекс для этого поля (или удалить существующий индекс)
Да (Допускаются совпадения)	Создать индекс для этого поля
Да (Совпадения не допускаются)	Создать уникальный индекс для этого поля

При создании уникального индекса невозможно ввести новое значение в определенном поле, если такое значение уже существует в том же поле другой записи. В Access уникальный индекс автоматически создается для первичных ключей, однако может понадобиться, чтобы создание значений, совпадающих

со значениями в других полях, было невозможным. Например, можно создать уникальный индекс для поля, в котором содержатся серийные номера, чтобы двум продуктам не мог быть присвоен один и тот же серийный номер.

Создание индекса для одного поля

1. В области переходов щелкните правой кнопкой мыши название таблицы, в которой необходимо создать индекс, затем в контекстном меню выберите Конструктор.

2. Щелкните Имя поля для поля, которое следует индексировать.

3. В разделе Свойства поля откройте вкладку Общие.

4. В свойстве Индексированное поле щелкните значение Да (Допускаются совпадения), если следует разрешить повторяющиеся значения, или значение Да (Совпадения не допускаются), чтобы создать уникальный индекс.

5. Чтобы сохранить изменения, щелкните Сохранить на панели быстрого доступа или нажмите сочетание клавиш CTRL+S.

Создание составного индекса

Чтобы создать составной индекс, необходимо включить в него строку для каждого поля в индексе и поместить индекс только в самой первой строке. Все строки обрабатываются как часть одного индекса до тех пор, пока не будет обнаружена строка с другим названием индекса. Чтобы вставить строку, щелкните правой кнопкой мыши место, куда следует вставить строку, затем в контекстном меню щелкните команду Вставить строки.

1. В области переходов щелкните правой кнопкой мыши название таблицы, в которой необходимо создать индекс, затем в контекстном меню выберите Конструктор.

2. На вкладке Конструктор в группе Показать или скрыть щелкните Индексы.

Появится окно «Индексы». Измените размеры этого окна, чтобы отображались пустые строки и свойства индекса.

3. В первой пустой строке столбца Индекс введите имя индекса. Для индекса можно использовать либо имя одного из индексируемых полей, либо другое подходящее имя.

4. В столбце Имя поля щелкните стрелку, затем щелкните первое поле, которое следует использовать в индексе.

5. Следующую строку столбца Индекс оставьте пустой, затем в столбце Имя поля укажите второе индексируемое поле. Повторите этот шаг для всех полей, которые необходимо включить в индекс.

По умолчанию устанавливается сортировка по возрастанию.

6. Чтобы изменить порядок сортировки значений полей, в столбце Порядок сортировки окна «Индексы» щелкните По возрастанию или По убыванию.

7. В окне Свойства индекса окна Индексы установите свойства индекса для строки в столбце Имя индекса, содержащем индекс. Установите свойства в соответствии со следующей таблицей 22.

Таблица 22. Установка свойств индекса

Надпись	**Значение**
Первичный	Если Да, то индекс является первичным ключом.
Уникальный	Если Да, то каждое индексируемое значение должно быть уникальным.
Пропуск пустых полей	Если Да, то записи с пустыми значениями в индексируемых полях будут исключены из индекса.

1. Чтобы сохранить изменения, щелкните Сохранить на панели быстрого доступа.

Клавиши быстрого доступа Нажмите сочетание клавиш CTRL+S.

2. Закройте окно «Индексы».

Удаление индекса

Если индекс становится ненужным или сильно снижает производительность, его можно удалить. Удаление затрагивает только сам индекс, а не поля, которые были в него включены.

1. В области переходов щелкните правой кнопкой мыши название таблицы, в которой необходимо удалить индекс, затем в контекстном меню выберите пункт Конструктор.

2. На вкладке Конструктор в группе Показать или скрыть щелкните Индексы.

Появится окно «Индексы». Измените размеры этого окна, чтобы отображались пустые строки и свойства индекса.

3. В окне «Индексы» выделите строки, содержащие индекс, который следует удалить, и нажмите клавишу DELETE.

4. Чтобы сохранить изменения, щелкните Сохранить на панели быстрого доступа.

Клавиши быстрого доступа Нажмите сочетание клавиш CTRL+S.

5. Закройте окно Индексы

Просмотр или редактирование индексов

Может понадобиться просмотр индексов в таблице, чтобы оценить их влияние на производительность или убедиться в том, что необходимые поля индексированы.

1. В области переходов щелкните правой кнопкой мыши название таблицы, в которой необходимо изменить индекс, затем в контекстном меню выберите пункт Конструктор.

2. На вкладке Конструктор в группе Показать или скрыть щелкните Индексы.

Появится окно «Индексы». Измените размеры этого окна, чтобы отображались пустые строки и свойства индекса.

3. Просмотрите или измените индексы и свойства индексов в соответствии со своими задачами.

4. Чтобы сохранить изменения, щелкните команду Сохранить на панели быстрого доступа.

Клавиши быстрого доступа Нажмите сочетание клавиш CTRL+S.

5. Закройте окно «Индексы».

Автоматическое создание индексов

В некоторых случаях индексы создаются автоматически. Например, индексы создаются для любых полей, которые определяется пользователем как первичный ключ таблицы.

Другим способом автоматического создания индекса является использование параметра Автоиндекс при импорте и создании, находящегося в диалоговом окне Параметры Access. Будут проиндексированы все поля, которые начинаются с указанных в поле Автоиндекс при импорте и создании символов или заканчиваются ими, например: ID, ключ, код или число. Чтобы просмотреть или изменить текущие параметры, выполните следующие шаги:

1. Нажмите кнопку Microsoft Office, а затем выберите команду Параметры Access.

2. Щелкните пункт Конструкторы объектов, затем под заголовком Конструктор таблицы добавьте, измените или удалите значения в поле Автоиндекс при импорте и создании. Используйте точку с запятой (;) для разделения значений.

Примечание Если имя поля начинается со значения, указанного в списке, или заканчивается им, это поле будет автоматически индексировано.

3. Нажмите кнопку ОК.

Так как каждый дополнительный индекс требует дополнительной обработки, производительность при добавлении или обновлении данных снижается. Поэтому следует оценить изменение значений, указанных в поле Автоиндекс при импорте и создании, или уменьшить количество этих значений, чтобы сократить количество создаваемых индексов.

15 РАБОТА С ФОРМАМИ

Форма Access – это объект, с помощью которого пользователи могут добавлять, редактировать и отображать данные, хранящиеся в базе данных классического приложения Access, ее внешний вид играет важную роль. Если база данных классического приложения Access используется несколькими пользователями, хорошо продуманные формы – залог точности данных и эффективности работы с ними.

Создать форму на компьютере в базе данных Access можно несколькими способами.

Создание формы с помощью инструмента «Форма»

Создание разделенной формы при помощи инструмента «Разделенная форма»

– Создание формы, в которой отображается несколько записей, при помощи инструмента «Несколько элементов»
– Создание формы при помощи мастера форм
– Создание формы при помощи инструмента «Пустая форма»
– Режим макета и режим конструктора
– Доработка формы в режиме макета
– Доработка формы в режиме конструктора

15.1 Создание формы с помощью инструмента «Форма»

С помощью инструмента "Форма" можно создать форму одним щелчком мыши. При использовании этого средства все поля базового источника данных размещаются в форме. Можно сразу же начать использование новой формы либо при необходимости изменить ее в режиме макета или конструктора.

Использование инструмента "Форма" для создания новой формы

1. В области навигации щелкните таблицу или запрос с данными, которые должны отображаться в форме.

2. На вкладке Создать в группе Формы нажмите кнопку Форма.

Будет создана новая форма и отображена в режиме макета. В режиме макета можно внести изменения в структуру формы при одновременном отображении данных. Например, при необходимости можно настроить размер полей в соответствии с данными.

Если приложение Access обнаруживает одну таблицу, связанную отношением "один-ко-многим" с таблицей или запросом, который использовался для создания формы, оно добавляет таблицу данных в форму, основанную на связанной таблице или запросе. Если таблица данных в форме не нужна, ее можно удалить. Если имеется несколько таблиц, связанных отношением "один-ко-многим" с таблицей, которая использовалась для создания формы, то Access не добавляет таблицы данных в форму.

15.2 Создание разделенной формы при помощи инструмента «Разделенная форма»

Разделенная форма позволяет одновременно отображать данные в двух представлениях – в режиме формы и в режиме таблицы.

Разделенная форма отличается от сочетания формы и подчиненной формы тем, что эти два представления связаны с одним источником данных и всегда синхронизированы друг с другом. При выделении поля в одной части формы выделяется то же поле в другой части. Данные можно добавлять, изменять или удалять в любой части (при условии, что источник записей допускает обновление, а параметры формы не запрещают такие действия).

Разделенная форма позволяет использовать преимущества обоих типов форм в одной форме. Например, можно воспользоваться табличной частью формы, чтобы быстро найти запись, а затем просмотреть или изменить запись в другой части формы.

Чтобы создать разделенную форму при помощи инструмента «Разделенная форма», выполните указанные ниже действия.

1. В области навигации щелкните таблицу или запрос с данными, которые должны отображаться в форме, либо откройте таблицу или запрос в режиме таблицы.

2. На вкладке Создать в группе Формы нажмите кнопку Форма.

Приложение Access создаст форму и отобразит ее в режиме макета. В режиме макета можно внести изменения в структуру формы при одновременном отображении данных. Например, при необходимости можно настроить размер полей в соответствии с данными.

Разделенную форму можно добавить в веб-базу данных, однако для использования этой формы придется открыть базу данных в приложении Access (иначе говоря, эта форма не будет работать в браузере).

15.3 Создание формы, в которой отображается несколько записей, при помощи инструмента «Несколько элементов»

В форме, созданной с помощью средства «Форма», одновременно отображается только одна запись. Если нужна форма, в которой отображается сразу несколько записей, и при этом требуются более широкие возможности настройки, чем у таблицы, можно воспользоваться инструментом «Несколько элементов».

1. В области навигации щелкните таблицу или запрос с данными, которые должны отображаться в форме.

2. На вкладке Создание в группе Формы нажмите кнопку Другие формы и выберите пункт Несколько элементов.

Приложение Access создаст форму и отобразит ее в режиме макета. В режиме макета можно внести изменения в структуру формы при одновремен-

ном отображении данных. Например, можно настроить размер полей в соответствии с данными.

Создаваемая при помощи инструмента «Несколько элементов» форма внешне напоминает таблицу. Данные расположены в строках и столбцах, одновременно отображается несколько записей. Однако форма «Несколько элементов» предоставляет больше возможностей настройки, чем таблица. Например, в нее можно добавлять графические элементы, кнопки и другие элементы управления.

15.4 Создание формы при помощи мастера форм

Для получения большей свободы выбора полей, отображаемых на форме, вместо упомянутых выше инструментов можно воспользоваться мастером форм. Кроме того, можно указать способ группировки и сортировки данных, а также включить в форму поля из нескольких таблиц или запросов при условии, что заранее заданы отношения между этими таблицами и запросами.

1. На вкладке Создание в группе Формы нажмите кнопку Мастер форм.
2. Следуйте инструкциям на страницах мастера форм.

Для добавления в форму полей из нескольких таблиц или запросов не нажимайте кнопки Далее или Готово после выбора полей из первой таблицы или запроса на первой странице мастера форм. Повторите действия для выбора другой таблицы или запроса и щелкните все дополнительные поля, которые требуется включить в форму. Чтобы продолжить, нажмите кнопку Далее или Готово.

3. На последней странице мастера нажмите кнопку Готово.

15.5 Создание формы при помощи инструмента «Пустая форма»

Если мастер или инструменты создания форм не подходят, для создания формы можно воспользоваться инструментом «Пустая форма». Так можно очень быстро построить форму, особенно если на ней будет лишь несколько полей.

1. На вкладке Создать в группе Формы нажмите кнопку Пустая форма.

Access откроет пустую форму в режиме макета и отобразит область Список полей.

2. В области Список полей щелкните знак «плюс» (+) рядом с таблицей или таблицами, содержащими поля, которые нужно включить в форму.
3. Чтобы добавить поле в форму, дважды щелкните его и перетащите на форму.

Примечание:

– После добавления первого поля можно добавить одновременно несколько полей. Для этого необходимо щелкнуть несколько полей, удерживая при этом нажатой клавишу CTRL, а затем одновременно перетащить их на форму.

– Порядок таблиц в области Список полей может изменяться в зависимости от того, какая часть формы выделена в текущий момент. Если поле, которое требуется добавить, не отображается, попробуйте выделить другую часть формы и повторите попытку.

4. Используя инструменты группы Колонтитулы на вкладке Конструктор, можно добавить в форму эмблему компании, заголовок или дату и время.

5. Для добавления в форму элементов управления других типов используйте инструменты группы Элементы управления на вкладке Конструктор.

Немного больший выбор элементов управления доступен в режиме конструктора после щелчка формы правой кнопкой мыши и выбора пункта Режим конструктора.

Элементы управления, добавляемые в режиме конструктора могут быть несовместимыми с функцией веб-публикации. Если форму планируется публиковать в Интернете, необходимо использовать только возможности, доступные в режиме макета.

15.6 Режим макета и режим конструктора

Режим макета

Режим макета представляет собой наиболее наглядный режим изменения форм. Его можно использовать для внесения практически любых изменений в форму в Access. При создании базы данных с помощью команды Пустая веб-база данных в представлении Microsoft Backstage режим макета является единственным режимом, в котором можно проектировать формы.

В режиме макета форма уже запущена. Таким образом, данные отображаются практически так же, как при реальном использовании формы. При этом в данном режиме можно изменять структуру формы. Поскольку при изменении формы отображаются реальные данные, в режиме макета удобно задавать размер элементов управления и выполнять иные задачи, влияющие на вид и удобство использования форм.

При разработке стандартной базы данных для настольных компьютеров (не являющейся веб-базой данных) некоторые задачи невозможно выполнить в режиме макета. В этом случае можно переключиться в режим конструктора. В некоторых случаях приложение Access выводит сообщение о необходимости переключиться в режим конструктора для внесения определенных изменений.

Режим конструктора

Режим конструктора обеспечивает более подробное представление структуры формы. В нем отображаются разделы колонтитулов и данных формы. В этом режиме форма не выполняется, поэтому при внесении изменений невозможно просмотреть соответствующие данные. Однако некоторые задачи удобнее выполнять в режиме конструктора, а не макета, например следующие:

– добавление в форму дополнительных элементов управления, таких как границы привязанных объектов, разрывы страниц и диаграммы;

– изменение источников элемента управления «текстовое поле» непосредственно в самом поле, без использования окна свойств;

– изменение размеров разделов формы, таких как «Заголовок формы» или «Область данных»;

– изменение определенных свойств формы, которые нельзя изменить в режиме макета.

15.7 Доработка формы в режиме макета

Создав форму, можно легко доработать ее в режиме макета. Ориентируясь на фактические данные формы, можно изменить расположение элементов управления и подобрать их размеры. Можно добавить в форму новые элементы управления, а также задать свойства формы и входящих в нее элементов управления.

Чтобы переключиться в режим макета, щелкните правой кнопкой мыши имя формы в области навигации и выберите команду Режим макета.

Форма будет открыта в режиме макета.

Изменить свойства формы, ее разделов и элементов управления можно с помощью окна свойств. Чтобы отобразить его, нажмите клавишу F4.

Из области Список полей можно добавить в макет формы поля из базовой таблицы или базового запроса. Для отображения области Список полей выполните одно из следующих действий.

– На вкладке Конструктор в группе Сервис нажмите кнопку Добавить существующие поля.

– Поля можно перетащить в форму непосредственно из области Список полей.

– Чтобы добавить одно поле, дважды щелкните его или перетащите его из области Список полей в тот раздел формы, где оно должно отображаться.

– Чтобы добавить сразу несколько полей, щелкните их, удерживая нажатой клавишу CTRL. Затем перетащите выбранные поля в форму.

15.8 Доработка формы в режиме конструктора

В случае баз данных для настольных компьютеров доработать форму можно в режиме конструктора. Можно добавить в форму новые элементы управления и поля, разместив их на сетке макета. В окне свойств доступны многие свойства настройки формы.

Чтобы переключиться в режим конструктора, в области навигации щелкните форму правой кнопкой мыши и выберите команду Режим конструктора.

Форма будет отображена в режиме конструктора.

Изменить свойства формы, ее разделов и элементов управления можно с помощью окна свойств. Чтобы отобразить его, нажмите клавишу F4.

Из области Список полей можно добавить в макет формы поля из базовой таблицы или базового запроса. Для отображения области Список полей выполните одно из следующих действий.

– На вкладке Конструктор в группе Сервис нажмите кнопку Добавить существующие поля.

– Поля можно перетащить в форму непосредственно из области Список полей.

– Чтобы добавить одно поле, дважды щелкните его или перетащите его из области Список полей в тот раздел формы, где оно должно отображаться.

– Чтобы добавить сразу несколько полей, щелкните их, удерживая нажатой клавишу CTRL. Затем перетащите выбранные поля в форму.

16 ФИЛЬТРЫ

Отличие фильтра от запроса. У этих двух средств есть свои особенности по настройке и применению:

– Фильтр следует использовать, если из набора записей, которые вы в текущий момент просматриваете, необходимо оперативно сделать выборку в соответствии с определенными правилами.

– Запрос предназначен для выборки и просмотра записей из одной или нескольких таблиц, его можно сохранить, чтобы позднее вновь использовать или создать на его базе форму.

16.1 Выборка записей по одному значению в одном поле

Есть следующие средства для выборки записей по одному значению:

– «Фильтр по выделенному»  позволяет быстро найти необходимые записи, выполнив два действия, – выделение значения в качестве условия отбора и щелчок на кнопке.

– Исключить выделенное. Выделите значение и выберите в контекстном меню команду «Исключить выделенное».

– «Фильтр для». Эта команда используется в случае, если вы работаете с большим набором данных. Правой кнопкой щелкните в любом месте поля, к которому необходимо применить фильтр, и введите значение поля.

– «Изменить фильтр»  – средство для отбора записей по нескольким полям, дает возможность выбрать критерий отбора в раскрывающемся списке, содержащем все значения поля. Затем нажмите на кнопку Применить фильтр.

16.2 Выборка записей по нескольким значениям в нескольких полях

Для составления фильтра по выделенному для одновременного выполнения нескольких условий придется поочередно применить фильтр для каждого условия в отдельности. Команды «Изменить фильтр» и «Фильтр для» напротив дают возможность создать фильтр, в котором соединяется несколько условий.

Если необходимо просмотреть все записи, отвечающие сразу нескольким критериям или одному из них, то следует сформулировать фильтр, в котором условия будут связаны с помощью оператора *конъюнкции* And и *дизъюнкции* Or.

16.3 Составление расширенного фильтра

Возможности расширенного фильтра позволяют свободно объединять в разных полях разные условия фильтрации.

Для создания расширенного фильтра необходимо сначала открыть окно фильтра командой *Записи – Фильтр – Расширенный фильтр*. (Данное диалоговое окно аналогично окну запросов.)

Создавать расширенные фильтры можно с помощью оператора *конъюнкции* (при одновременной фильтрации нескольких полей) и *дизъюнкции* (при фильтрации одного из нескольких полей).

16.4 Сохранение фильтра

После того как фильтр создан, его надо применить с тем, чтобы отобразить в окне результат отбора. Для этого есть соответствующая команда «Применить фильтр». Чтобы удалить фильтр, необходимо щелкнуть на кнопке «Удалить фильтр».

Фильтр нельзя сохранить в качестве отдельного объекта базы данных. Существуют другие способы сохранения результатов.

Сохранение фильтра в режиме таблицы. При открытии таблицы в очередной раз, Access «запоминает» тот фильтр, который был применен последним.

Использование фильтра, сохраненного в форме. При закрытии окна формы фильтр, который был использован последним, автоматически сохраняется вместе с формой. Если форма создается на основе таблицы, с которой был сохранен фильтр, форма *наследует* этот фильтр.

Активизация унаследованного фильтра в отчете. Аналогично форме, отчет может наследовать фильтр таблицы, на которой он основан. Для этого необходимо выполнить следующие действия:

1. открыть отчет в режиме конструктора;
2. в списке Объект на панели инструментов выбрать пункт Отчет и щелкнуть на кнопке Свойства;
3. открыть вкладку Данные, изменить значение свойства Фильтр включен;
4. если необходимо поменять значение свойства Источник записи; в строке свойства Фильтр появятся условия отбора унаследованного фильтра.

17 ЗАПРОСЫ

Запросы упрощают просмотр, добавление, удаление или изменение данных в базе данных Access. Среди других целей использования запросов можно отметить:

– быстрый поиск определенных данных путем фильтрации с применением определенных критериев (условий);

– вычисление или сведение данных;

– автоматизированное управление данными, например регулярный просмотр актуальных данных.

В хорошо структурированной базе данных сведения, которые требуется представить с использованием формы или отчета, зачастую хранятся в разных таблицах. Запрос может извлечь информацию из разных таблиц и собрать ее для отображения в виде формы или отчета. Запрос может представлять собой обращение к данным для получения информации из базы данных или выполнения действий с данными. Запрос можно использовать для получения ответа на простой вопрос, выполнения расчетов, объединения данных из разных таблиц, а также для добавления, изменения или удаления данных в таблице. Это очень гибкий инструмент: существует много типов запросов, и каждый тип создается с учетом задачи (таблица 23).

Таблица 23. Основные типы запросов

Основные типы запросов	Использование
На выборку	Получение данных из таблицы и выполнение вычислений.
Действие	Добавление, изменение или удаление данных. Для каждой задачи существует специальный тип запроса на изменение. Запросы на изменение недоступны в веб-приложениях Access.

17.1 Создание запроса на выборку

Запрос на выборку позволяет просматривать данные только из определенных полей таблицы или из нескольких таблиц одновременно, или же находить данные, которые соответствуют определенным условиям. С помощью запроса на выборку можно получить только нужные данные и отобразить их в режиме таблицы.

Для этого можно создать запрос на выборку с помощью мастера запросов. На вкладке **Создание** нажмите кнопку **Мастер запросов** и следуйте инструкциям мастера для создания и выполнения запроса.

Однако если требуется добавить в запрос условия отбора, нужно использовать конструктор запросов.

1. Откройте вкладку **Создание** и выберите пункт **Конструктор запросов**.

2. В диалоговом окне **Добавление таблицы** дважды щелкните необходимую таблицу и нажмите кнопку **Закрыть**.

3. Дважды щелкните на поля, которые будут участвовать в запросе, чтобы добавить их на бланк запроса.

4. В строке условий отбора под полем поля добавьте условия отбора. В следующей таблице 24 показаны примеры условий и описано, как они работают.

Таблица 24. Примеры условий запросов

Чтобы добавить записи, которые...	**Используйте это условие**	**Результат запроса**
Точно соответствуют определенному значению, например "Китай"	"Китай"	Возвращает записи, в которых поле "СтранаРегион" содержит значение "Китай".
Не соответствуют определенному значению, например "Мексика"	Not "Мексика"	Возвращает записи, в которых значением поля "СтранаРегион" не является "Мексика".
Начинаются с заданной строки символов, например "С"	Like С*	Возвращает записи всех стран или регионов, названия которых начинаются с буквы "С", таких как Словакия и США. **Примечание** Символ "звездочка" (*****) в выражении обозначает любую строку символов. Он также называется подстановочным знаком.
Не начинаются с заданной строки символов, например "С"	Not Like С*	Возвращает записи всех стран или регионов, названия которых не начинаются с буквы "С".
Содержат заданную строку, например "Корея"	Like "*Корея*"	Возвращает записи всех стран или регионов, названия которых содержат строку "Корея".
Не содержат заданную строку, например "Корея"	Not Like "*Корея*"	Возвращает записи всех стран или регионов, названия которых не содержат строку "Корея".
Заканчиваются заданной строкой, например "ина"	Like "*ина"	Возвращает записи всех стран или регионов, названия которых заканчиваются на "ина", таких как "Украина" и "Аргентина".
Не заканчиваются заданной строкой, например "ина"	Not Like "*ина"	Возвращает записи всех стран или регионов, названия которых не заканчиваются на "ина", как в названиях "Украина" и "Аргентина".
Содержат пустые значения (или значения отсутствуют)	Is Null	Возвращает записи, в которых это поле не содержит значения.
Не содержат пустых значений	Is Not Null	Возвращает записи, в которых это поле содержит значение.
Содержат пустую строку	"" (прямые кавычки)	Возвращает записи, в которых поле имеет пустое значение (но не значение NULL). Например, записи о продажах другому отделу могут содержать пустое значение в поле "СтранаРегион".
Не содержат пустых строк	Not ""	Возвращает записи, в которых поле "СтранаРегион" имеет непустое значение.

Продолжение таблицы 24

Содержит нулевые значения или пустые строки	"" Or Is Null	Возвращает записи, в которых значение в поле отсутствует или является пустым.
Ненулевые и непустые	Is Not Null And Not ""	Возвращает записи, в которых поле "СтранаРегион" имеет непустое значение, не равное NULL.
При сортировке в алфавитном порядке следуют за определенным значением, например "Мексика"	>= "Мексика"	Возвращает записи с названиями стран и регионов, начиная с Мексики и до конца алфавита.
Входят в определенный диапазон, например от А до Г	Like "[А–Г]*"	Возвращает страны и регионы, названия которых начинается с букв от "А" до "Г".
Совпадают с одним из двух значений, например "Словакия" или "США"	"Словакия" Or "США"	Возвращает записи для США и Словакии.
Содержат одно из значений, указанных в списке	In("Франция", "Китай", "Германия", "Япония")	Возвращает записи всех стран или регионов, указанных в списке.
Содержат определенные знаки в заданном месте значения поля	Right([СтранаРегион], 1) = "а"	Возвращает записи всех стран или регионов, названия которых заканчиваются на букву "а".
Соответствуют заданной длине	Len([СтранаРегион]) > 10	Возвращает записи стран или регионов, длина названия которых превышает 10 символов.
Соответствуют заданному шаблону	Like "Лив??"	Возвращает записи стран или регионов, названия которых состоят из пяти символов и начинаются с "Лив", например Ливия и Ливан.

Примечание Символы **?** и **_** в выражении обозначают один символ. Они также называются подстановочными знаками. Знак **_** нельзя использовать в одном выражении с символом **?**, а также с подстановочным знаком ***. Вы можете использовать подстановочный знак **_** в выражении, где есть подстановочный знак **%**.

17.2 Одновременный просмотр данных из нескольких связанных таблиц

Рассмотрим пример базы данных магазина пищевых продуктов, в которой требуется просмотреть заказы, полученные от клиентов из определенного города. Допустим, данные о заказах и данные о клиентах хранятся в двух таблицах под названием "Заказчики" и "Заказы" соответственно. Если в каждой таблице имеется поле "Код заказчика", которое является основой отношение "один-ко-многим" между этими двумя таблицами, вы можете создать запрос, который возвратит сведения о заказах клиентов, живущих в определенном городе, например в Костанае, используя следующую процедуру.

1. Откройте базу данных. На вкладке **Создание** в группе **Запросы** нажмите кнопку **Конструктор запросов**.
2. В диалоговом окне **Добавление таблицы** на вкладке **Таблицы** дважды щелкните элементы **Заказчики** и **Заказы**.
3. Закройте диалоговое окно **Добавление таблицы**. Обратите внимание на линию (называемую соединением), которая соединяет поле "Код" в таблице "Заказчики" с полем "Код заказчика" в таблице "Заказы". Эта линия отображает связь между двумя таблицами.
4. В таблице "Клиенты" дважды щелкните элементы **Организация** и **Город**, чтобы добавить эти поля в бланк запроса.
5. В бланке запроса в столбце **Город** снимите флажок в строке **Показать**.
6. В строке **Условие отбора** столбца **Город** введите **Костанай**.

Если снять флажок **Показать**, в результатах запроса не будет отображаться город, а слово **Костанай** в строке **Условие отбора** означает, что требуется просмотреть только те записи, для которых в поле "Город" указано значение "Костанай". В этом случае запрос возвращает данные только о тех клиентах, которые находятся в Тюмени. Для использования поля в условии отбора показывать его на экране не обязательно.

7. В таблице "Заказы" дважды щелкните элементы **Код заказа** и **Дата размещения**, чтобы добавить эти поля в два следующих столбца в бланке запроса.
8. На вкладке **Конструктор** в группе **Результаты** нажмите кнопку **Выполнить**. Происходит выполнение запроса и отображается список заказов клиентов из Тюмени.
9. Нажмите клавиши CTRL+S, чтобы сохранить запрос.

17.3 Создание запроса с параметрами

Если часто требуется выполнять варианты определенного запроса, можно использовать запрос с параметрами. При выполнении запроса с параметрами у пользователя запрашиваются значения полей, которые затем используются для создания условий для запроса.

В продолжение предыдущего примера, где было показано, как создавать запрос на выборку, который возвращает информацию о заказах для клиентов из Костаная, можно изменить этот запрос таким образом, чтобы при каждом его запуске выводилось приглашение на ввод названия города. Откройте базу данных, созданную в предыдущем примере.

1. В области навигации щелкните правой кнопкой мыши запрос **Заказы по городу** (созданный в предыдущем разделе) и выберите в контекстном меню пункт **Конструктор**.

2. В бланке запроса в строке **Условие отбора** столбца "Город" удалите слово **Костанай** и введите [Для какого города?].

Строка **[Для какого города?]** является предложением ввести параметр. Квадратные скобки показывают, что при выполнении запроса должно появиться предложение ввести данные, а текст (в данном случае **Для какого города?**) представляет собой вопрос, отображаемый в предложении.

В предложении ввести параметр нельзя использовать точку (**.**) или восклицательный знак (**!**).

1. Установите флажок в строке **Показать** столбца "Город", чтобы в результатах запроса отображался город.

2. На вкладке **Конструктор** в группе **Результаты** нажмите кнопку **Выполнить**. Запрос предложит ввести значение в строке "Город".

3. Введите слово **Астана** и нажмите клавишу ВВОД, чтобы увидеть заказы для клиентов в Астане.

Но что делать, если значения, которые можно указать, неизвестны? В приглашении на ввод можно использовать подстановочные знаки.

4. На вкладке **Главная** в группе **Представления** нажмите кнопку **Представление** и выберите пункт **Конструктор**.

5. В бланке запроса в строке **Условие отбора** столбца **Город** введите **Like [Для какого города?]&"*"**.

В этом предложении ввести параметр ключевое слово **Like**, амперсанд (**&**) и звездочка (*****), заключенная в кавычки, позволяют ввести сочетание знаков, включая подстановочные знаки, для получения разных результатов. Например, если пользователь вводит *****, запрос возвращает все города; если пользователь вводит **A**, запрос возвращает все города, начинающиеся на букву "А"; если пользователь вводит ***c***, запрос возвращает все города, в названиях которых имеется буква "с".

6. На вкладке **Конструктор** в группе **Результаты** нажмите кнопку **Выполнить**, и в строке приглашения запроса введите **Создать** и нажмите клавишу ВВОД.

В результате выполнения запроса будет отображен список заказов от клиентов из Астаны.

Указание типов данных для параметра

Можно также указать, данные какого типа разрешается вводить в качестве значения параметра. Тип данных можно настроить для любого параметра, но особенно важно сделать это для числовых и денежных данных, а также дан-

ных о дате и времени. Когда для параметра указан тип данных, пользователи получают более понятные сообщения об ошибках в случае ввода данных неправильного типа, например ввода текста, когда ожидаются денежные данные.

Если параметр настроен таким образом, чтобы принимать текстовые данные, любое введенное значение интерпретируется как текст и сообщение об ошибке не отображается.

Чтобы указать тип данных для параметра в запросе, выполните процедуру, описанную ниже.

1. Когда запрос открыт в конструкторе, на вкладке **Конструктор** в группе **Показать или скрыть** нажмите кнопку **Параметры**.

2. В диалоговом окне **Параметры запроса** в столбце **Параметр** введите текст запроса на ввод значения для каждого параметра, для которого требуется указать тип данных. Убедитесь, что каждый из параметров соответствует запросу, который используется в строке **Условие отбора** в бланке запроса.

3. В столбце **Тип данных** выберите тип данных для каждого параметра.

17.4 Создание итогового запроса

Строка "Итог" в таблице очень удобна, но для ответа на более сложные вопросы используется запрос итоговых значений. Такой запрос представляет собой запрос на выборку, позволяющий группировать данные и составлять сводку данных, например, когда требуется просмотреть итоги продаж каждого товара. В запросе итоговых значений можно использовать статистическую функцию Sum для просмотра итогов продаж каждого товара.

Чтобы получить итоговые значения промежуточных сумм для товаров, можно следующим образом изменить запрос "Промежуточные суммы для товаров", созданный в предыдущем примере.

1. На вкладке **Главная** щелкните **Вид > Представления конструктора**.

Запрос "Промежуточные суммы для товаров" будет открыт в конструкторе.

2. На вкладке **Конструктор** в группе **Показать или скрыть** нажмите кнопку **Итоги**.

В бланке запроса отобразится строка **Итоги**.

Несмотря на схожие названия, строка **Итоги** в бланке и строка **Итог** в таблице – не одно и то же, а именно:

- С помощью строки **Итоги** в бланке можно группировать данные по значениям полей.
- Строку **Итог** из таблицы можно добавить в результаты запроса итоговых значений.
- При использовании строки **Итоги** в бланке необходимо выбрать статистическую функцию для каждого поля. Если выполнять вычисления с полем не требуется, можно сгруппировать данные по этому полю.

Во втором столбце бланка в строке **Итог** выберите в раскрывающемся списке вариант **Sum**.

На вкладке **Конструктор** в группе **Результаты** нажмите кнопку **Выполнить**. Происходит выполнение запроса, а затем отображается список товаров с промежуточными суммами.

Нажмите клавиши CTRL+S, чтобы сохранить запрос. Оставьте запрос открытым.

17.5 Выполнение расчетов на основе данных

Обычно таблицы не используются для хранения вычисляемых значений, например промежуточных сумм, даже если они основаны на данных из одной базы данных. В некоторых случаях вычисленные значения могут устареть, поскольку данные, на основе которых они были рассчитаны, изменились. Например, не стоит хранить чей-либо возраст в таблице, поскольку придется обновлять это значение каждый год; вместо этого можно хранить дату рождения, а затем использовать запрос для расчета возраста.

Например, существует база данных с информацией о товарах, которые вы хотите продать. Она содержит таблицу под названием "Сведения о заказе", в которой находится информация о товарах, например цена и количество каждого товара. Можно вычислить промежуточные суммы с помощью запроса, который умножает количество каждого товара на цену за единицу этого товара, количество каждого товара на цену за единицу этого товара и скидку этого товара, а затем вычитает общую скидку из общей цены. Если в предыдущем примере была создана база данных, откройте ее и выполните следующие действия.

1. На вкладке **Создание** нажмите кнопку **Конструктор запросов**.
2. В диалоговом окне **Добавление таблицы** на вкладке **Таблицы** дважды щелкните **Сведения о заказе**.
3. Закройте диалоговое окно **Добавление таблицы**.
4. В таблице "Сведения о заказе" дважды щелкните **Код товара**, чтобы добавить это поле в первый столбец бланка запроса.
5. Во втором столбце бланка щелкните правой кнопкой мыши строку **Поле**, а затем выберите в контекстном меню команду **Область ввода**.
6. В диалоговом окне **Область ввода** введите или вставьте следующее выражение: Всего: ([Количество]*[Цена за единицу])–([Количество]*[Цена за единицу]*[Скидка])
7. нажмите кнопку **ОК**.
8. На вкладке **Конструктор** нажмите кнопку **Выполнить**. Происходит выполнение запроса, а затем отображается список товаров с промежуточными суммами для каждого заказа.
9. Нажмите клавиши CTRL+S, чтобы сохранить запрос, и назовите его **Промежуточные суммы для товаров**.

17.6 Просмотр сводных данных и статистических показателей

При использовании таблиц для записи операций или хранения постоянно встречающихся числовых данных удобно иметь возможность просмотреть статистические показатели для этих данных, например суммарные или средние значения. В Access в таблицу можно добавлять итоговую строку. Итоговая строка – это строка внизу таблицы, которая отображает итоговое или другое статистическое значение.

1. Выполните запрос "Промежуточные суммы для товаров" и оставьте результаты открытыми в режим таблицы.
2. На вкладке **Главная** щелкните **Итоги**. В нижней части таблицы появится новая строка со словом **Итог** в первом столбце.
3. Щелкните ячейку в последней строке с именем **Итог**.
4. Щелкните стрелку для просмотра доступных статистических функций. Поскольку в столбце содержатся текстовые данные, предлагается только два варианта: **Нет** и **Количество**.
5. Выберите **Количество**. Содержимое ячейки изменится с **Итог** на число значений в столбце.
6. Щелкните соседнюю ячейку (второй столбец). Обратите внимание на стрелку, которая появилась в ячейке.
7. Щелкните стрелку и выберите **Sum**. В поле будет отображаться сумма значений в столбце.
8. Оставьте запрос открытым в режиме таблицы.

17.7 Создание перекрестного запроса

Пусть теперь необходимо просмотреть промежуточные суммы для товаров, а также статистические показатели по месяцам, т. е. в каждой строке должны отображаться промежуточные суммы для товара, а в каждом столбце – промежуточные суммы за месяц. Для вывода промежуточных сумм для товара и промежуточных сумм за месяц используйте перекрестный запрос.

Вы можете снова изменить запрос "Промежуточные суммы для товаров", чтобы он возвращал строки промежуточных сумм для товаров и столбцы промежуточных сумм по месяцам.

1. На вкладке **Главная** в группе **Представления** нажмите кнопку **Представление** и выберите пункт **Конструктор**.
2. В группе **Настройка запроса** щелкните элемент **Добавление таблицы**.
3. В диалоговом окне **Добавление таблицы** дважды щелкните элемент **Заказы** , а затем нажмите кнопку **Закрыть**.
4. На вкладке **Конструктор** в группе **Тип запроса** выберите **перекрестный**. В бланке скрытые **Показать** строки и строки **Перекрестная таблица** отображается.

5. В третьем столбце бланка щелкните правой кнопкой мыши строку **Поле**, а затем выберите в контекстном меню пункт **Область ввода**. Откроется окно **Область ввода**.

6. В диалоговом окне **Область ввода** введите или вставьте следующее выражение: **Месяц: "Месяц" & DatePart("м", [Дата заказа])**

7. Нажмите кнопку **ОК**.

8. В строке **Перекрестная таблица** выберите следующие значения в раскрывающемся списке: **Заголовки строк** для первого столбца, **Значение** для второго столбца и **Заголовки столбцов** для третьего.

9. На вкладке **Конструктор** в группе **Результаты** нажмите кнопку **Выполнить**. Происходит выполнение запроса, а затем отображаются промежуточные суммы, собранные по месяцам.

10. Нажмите клавиши CTRL+S, чтобы сохранить запрос.

17.8 Создание запроса на создание таблицы

Для создания новой таблицы на основе данных, которые хранятся в других таблицах, можно использовать запрос на создание таблицы.

Например, пусть требуется отправить данные о заказах в Костанае партнеру из Костаная, который использует Access для подготовки отчетов. Вместо отправки всех данных о заказах можно отправить только те данные, которые относятся к заказам в Костанае.

Можно создать запрос на выборку, содержащий данные о заказах в Костанае, а затем использовать этот запрос для создания новой таблицы. Для этого используйте описанную ниже процедуру.

1. Откройте базу данных из предыдущего примера.

Для выполнения запроса на создание таблицы может потребоваться включить содержимое базы данных.

Если под лентой появится сообщение о включении базы данных, щелкните **Включить содержимое.** Если база данных уже находится в надежном расположении, строка сообщения не появится.

2. На вкладке **Создание** в группе **Запросы** нажмите кнопку **Конструктор запросов**.

3. В диалоговом окне **Добавление таблицы** дважды щелкните **Сведения о заказе** и **Заказы** и закройте диалоговое окно **Добавление таблицы**.

4. В таблице **Заказы** дважды щелкните поля **Код заказчика** и **Город получателя**, чтобы добавить их в бланк.

5. В таблице **Сведения о заказе** дважды щелкните элементы **Код заказа**, **Код товара**, **Количество**, **Цена за единицу** и **Скидка**, чтобы добавить эти поля в бланк.

6. В столбце **Город получателя** бланка снимите флажок в строке **Показать**. В строке **Условие отбора** введите **'Костанай'** (включая одинарные кавычки). Проверьте результаты выполнения запроса, прежде чем использовать их для создания таблицы.

7. На вкладке **Конструктор** в группе **Результаты** нажмите кнопку **Выполнить**.

8. Нажмите клавиши CTRL+S, чтобы сохранить запрос.

9. В поле **Имя запроса** введите **Запрос по заказам в Костанае** и нажмите кнопку **ОК**.

10. На вкладке **Главная** в группе **Представления** нажмите кнопку **Представление** и выберите пункт **Конструктор**.

11. На вкладке **Конструктор** в группе **Тип запроса** нажмите кнопку **Создание таблицы**.

12. В диалоговом окне **Создание таблицы** в поле **Имя таблицы** введите **Заказы в Костанае** и нажмите кнопку **ОК**.

13. На вкладке **Конструктор** в группе **Результаты** нажмите кнопку **Выполнить**.

14. В диалоговом окне подтверждения нажмите кнопку **Да**, и в области навигации отобразится новая таблица.

Если таблица с указанным именем уже существует, она удаляется перед выполнением запроса.

17.9 Создание запроса на добавление

Для извлечения данных из одной или нескольких таблиц и добавления их в другую таблицу можно использовать запрос на добавление.

Предположим, вы создали таблицу для совместной работы с партнером из Ростова, но вспомнили, что этот партнер работает также с клиентами из Казани. Необходимо добавить в эту таблицу строки с данными по Казани. Используя следующую процедуру, можно добавить эти данные в таблицу "Заказы в Ростове".

1. Откройте "Запрос по заказам в Костанае" в конструкторе.

2. На вкладке **Конструктор** в группе **Тип запроса** нажмите кнопку **Добавить**. Откроется диалоговое окно **Добавление**.

3. В диалоговом окне **Добавление** щелкните стрелку в поле **Имя таблицы** и выберите **Заказы в Костанае** в раскрывающемся списке, а затем нажмите кнопку **ОК**.

4. В бланке в строке **Условие отбора** столбца "Город получателя" удалите значение 'Костанай' и введите **'Алматы'**.

5. В строке **Добавление записей в таблицу** выберите соответствующее поле для каждого столбца.

В этом примере значения в строке **Добавление записей в таблицу** должны соответствовать значениям в строке **Поле**, но это не требуется для нормальной работы запросов на добавление.

6. На вкладке **Конструктор** в группе **Результаты** нажмите кнопку **Выполнить**.

При выполнении запроса, который возвращает большое количество данных, может появится сообщение об ошибке, в котором будет сказано, что отме-

нить запрос не удастся. Попробуйте увеличить ограничение сегмента памяти до 3 МБ, чтобы выполнить запрос до конца.

17.10 Создание запроса на обновление

Для изменения данных в таблицах, а также для ввода условий, указывающих, какие строки следует обновить, можно использовать запрос на обновление. Запрос на обновление позволяет просмотреть обновленные данные перед выполнением обновления.

Запрос на изменение невозможно отменить. Возможно, перед обновлением следует создать резервные копии всех таблиц, которые будут обновлены запросом на обновление.

В предыдущем примере вы добавили строки в таблицу "Заказы в Костанае". В ней в поле "Код товара" отображается числовой код товара. Чтобы данные отчетов были более информативными, замените коды товаров их названиям. Используйте следующую процедуру.

1. Откройте таблицу "Заказы в Костанае" в конструкторе.
2. В строке "Код товара" измените тип данных **Числовой** на **Текстовый**.
3. Сохраните и закройте таблицу "Заказы в Костанае".
4. На вкладке **Создание** в группе **Запросы** нажмите кнопку **Конструктор запросов**.
5. В диалоговом окне **Добавление таблицы** дважды щелкните **Заказы в Ростове** и **Товары** и закройте диалоговое окно **Добавление таблицы**.
6. На вкладке **Конструктор** в группе **Тип запроса** нажмите кнопку **Обновить**.
7. В бланке больше не будут отображаться строки **Сортировка** и **Показать** и появится строка **Обновление**.
8. В таблице **Заказы в Костанае** дважды щелкните элемент **Код товара**, чтобы добавить это поле в бланк.
9. В бланке в строке **Обновление** столбца **Код товара** введите или вставьте следующее: **[Товары].[Наименование]**

Запрос на обновление можно использовать для удаления значений полей; для этого используется пустая строка (**""**) или значение NULL в строке **Обновление**.

10. В строке **Условие отбора** введите или вставьте следующее: **[Код продукта] Like ([Товары].[Код])**
11. Можно узнать, какие значения будут изменены запросом на обновление, просмотрев запрос в режиме таблицы.
12. На вкладке **Конструктор** меню **Вид** > **Представления таблицы данных**. Запрос возвращает список идентификаторов продуктов, будут обновлены.
13. На вкладке **Конструктор** нажмите кнопку **Выполнить**.

При открытии таблицы "Заказы в Костанае" можно будет увидеть, что числовые значения в поле "Код товара" заменены наименованиями из таблицы "Товары". Читайте статью Создание запроса на обновление.

17.11 Создание запроса на удаление

Для удаления данных из таблиц, а также для ввода условий, указывающих, какие строки следует удалить, можно использовать запрос на удаление. Запрос на удаление позволяет просмотреть удаляемые строки перед выполнением удаления.

Предположим, готовясь отправить таблицу "Заказы в Костанае" из предыдущего примера партнеру в Костанай, вы заметили, что некоторые строки содержат пустые поля. Перед отправкой таблицы их необходимо удалить. Можно открыть таблицу и удалить строки вручную, но если их много и есть четкие условия отбора, удобнее использовать запрос на удаление.

Вы можете использовать запрос для удаления из таблицы "Заказы в Костанае" строк, в которых отсутствует значение "Код заказа". Для этого выполните описанную ниже процедуру.

1. На вкладке **Создание** нажмите кнопку **Конструктор запросов**.
2. В диалоговом окне **Добавление таблицы** дважды щелкните **Заказы в Костанае**, затем закройте диалоговое окно **Добавление таблицы**.
3. На вкладке **Конструктор** в группе **Тип запроса** нажмите кнопку **Удалить**. В бланке исчезают строки **Сортировка** и **Показать** и появится строка **Удалить**.
4. В таблице **Заказы в Костанае** дважды щелкните поле **Код заказа**, чтобы добавить его в бланк.
5. В бланке в строке **Условие отбора** в столбце "Код заказа" введите **Is Null**.
6. На вкладке **Конструктор** в группе **Результаты** нажмите кнопку **Выполнить**.

8 ОТЧЕТЫ

При работе с базой данных для просмотра, форматирования и обобщения данных обычно используются отчеты. Отчет – это объект базы данных, который используется для отображения и обобщения данных. С помощью отчетов можно распространять и архивировать мгновенные снимки данных в печатном виде, в виде PDF– или XPS–файлов и файлов других форматов. Создавать полезные отчеты намного проще, если структура таблиц базы данных и отношения между ними хорошо продуманы.

Отчеты могут содержать подробные сведения об отдельных записях, сводные сведения о большой группе записей либо и то, и другое. Кроме того, отчеты Access также можно использовать при создании наклеек для списков рассылок и многого другого.

Можно создавать «свободные» отчеты, не отображающие никаких данных, однако в этой статье предполагается, что отчет привязан к некоторому источнику данных, например к таблице или запросу.

18.1 Составные части отчета

В приложении Access макет отчета разбит на разделы. В клиентской базе данных разделы отчета можно просмотреть в режиме конструктора. В режиме макета разделы выделены не так четко, однако они все же обозначены и их можно выбирать в раскрывающемся списке в группе **Выделенный фрагмент** на вкладке **Формат**. Чтобы создавать полезные отчеты, нужно понимать назначение каждого раздела. Например, от выбора раздела для размещения вычисляемого элемента управления зависит способ вычисления результата. Ниже перечислены типы разделов и указано назначение каждого из них.

- **Заголовок отчета.** Печатается только один раз в начале отчета. В заголовок включается информация, обычно размещаемая на титульной странице, например эмблема компании, название отчета или дата. Если в нем содержится вычисляемый элемент управления, в котором используется агрегатная функция суммирования, сумма рассчитывается для всего отчета. Заголовок отчета печатается перед верхним колонтитулом.
- **Верхний колонтитул.** Выводится на печать вверху каждой страницы. Верхний колонтитул используется, например, когда нужно, чтобы название отчета повторялось на каждой странице.
- **Заголовок группы.** Печатается перед каждой новой группой записей. Этот раздел используется для печати названия группы. Например, если отчет сгруппирован по товарам, в заголовках групп можно указать их названия. Если поместить в заголовок группы вычисляемый элемент управления, в котором используется агрегатная функция суммирования, сумма будет рассчитываться для текущей группы. В отчете может быть несколько разделов заголовков групп в зависимости от количества уровней группировки.

– **Область данных.** Этот раздел печатается один раз для каждой строки данных из источника записей. В нем размещаются элементы управления, составляющие основное содержание отчета.

– **Примечание группы.** Выводится на печать в конце каждой группы записей. В примечании группы отображаются сводные сведения о данной группе. В отчете может быть несколько разделов примечаний групп в зависимости от количества уровней группировки.

– **Нижний колонтитул.** Печатается внизу каждой страницы. Используется для нумерации страниц и для печати постраничной информации.

– **Примечание отчета.** Печатается один раз в конце отчета. Примечание отчета можно использовать для отображения итогов и другой сводной информации по всему отчету.

18.2 Создание отчета

Действие 1. Выбор источника записей

Источником записей для отчета может быть таблица, именованный или внедренный запрос. Источник записей должен содержать все строки и столбцы данных, которые требуется отобразить в отчете.

– Если нужные данные содержатся в существующей таблице или запросе, выделите эту таблицу или запрос в области навигации и перейдите к действию 2 "Выбор инструмента отчета".

– Если источник записей еще не создан, выполните одно из перечисленных ниже действий.

Перейдите к действию 2 "Выбор инструмента отчета" и воспользуйтесь инструментом **Пустой отчет**.

Создайте таблицы или запрос, которые будут содержать нужные данные, выберите их в области навигации и перейдите к действию 2 "Выбор инструмента отчета".

Действие 2. Выбор инструмента отчета

Инструменты отчетов расположены на вкладке **Создание** ленты в группе **Отчеты**. В таблице ниже кратко описано их назначение.

Клиентские отчеты

Если добавить клиентские отчеты в веб–базу данных, ее можно будет опубликовать, однако отчеты не будут доступны в браузере. При этом с ними можно будет работать, если открыть базу данных в приложении Access.

Таблица 25. Описание инструментов «Отчеты»

Изображение кнопки	Инструмент	Описание
	Отчет	Позволяет создать простой табличный отчет, содержащий все поля из источника записей, который выбран в области навигации.

Продолжение таблицы 25

	Конструктор отчетов	Позволяет открыть пустой отчет в режиме конструктора и добавить в него нужные поля и элементы управления.
	Пустой отчет	Позволяет открыть пустой отчет в режиме макета и отобразить область задач "Список полей". При перетаскивании полей из этой области в отчет в приложении Access создается внедренный запрос, который сохраняется в его свойстве "Источник записей".
	Мастер отчетов	Служит для вызова пошагового мастера, с помощью которого можно задать поля, уровни группировки и сортировки и параметры макета. В результате работы мастера будет создан отчет на базе выбранных параметров.
	Наклейки	Вызывает мастер, в котором можно выбрать стандартный или настраиваемый размер подписей, набор отображаемых полей и порядок их сортировки. В результате мастер создает отчет с подписями на базе выбранных параметров.

Действие 3. Создание отчета

1. Выберите нужный инструмент, нажав соответствующую кнопку. Если появится окно мастера, следуйте инструкциям в нем, а на последней странице нажмите кнопку **Готово**.

Отчет будет открыт в приложении Access в режиме макета.

2. Отформатируйте отчет, чтобы добиться желаемого внешнего вида, выполнив действия, указанные ниже.

- Измените размер полей и подписей, выделяя их и перетаскивая края.
- Расположите поля в нужном порядке, выделяя их (и соответствующие подписи, если они есть) и перетаскивая в нужное место.
- Также можно, щелкая поля правой кнопкой мыши, с помощью команд контекстного меню объединять или разбивать ячейки, удалять и выделять поля и выполнять другие задачи форматирования.

18.3 Добавление группировки, сортировки и итогов

Чтобы быстро добавить в отчет группировку, сортировку или итоги, щелкните правой кнопкой мыши поле, к которому необходимо применить соответствующую функцию, и выберите нужную команду в контекстном меню.

Кроме того, когда отчет открыт в режиме макета или конструктора, можно добавить эти функции с помощью области "Группировка, сортировка и итоги".

1. Если область "Группировка, сортировка и итоги" не открыта, на вкладке **Конструктор** в группе **Группировка и итоги** выберите команду **Группировка**.

2. Нажмите кнопку **Добавить группировку** или **Добавить сортировку** и выберите поле, по которому требуется сгруппировать или отсортировать данные.

3. Чтобы задать дополнительные параметры или добавить итоги, в строке группировки или сортировки выберите команду **Больше**.

18.4 Добавление изображений

1. В области навигации щелкните правой кнопкой мыши отчет, в который требуется добавить изображение, и выберите команду **Режим макета**.

2. Щелкните то место отчета, где нужно поместить изображение.

3. На вкладке **Конструктор** в группе **Элементы управления** нажмите кнопку **Вставить изображение**.

4. Выполните одно из указанных ниже действий.

– **Используйте существующее изображение.** Если нужное изображение уже есть в коллекции, щелкните его, чтобы добавить в отчет.

– **Добавьте новое изображение.** Нажмите кнопку **Обзор** в нижней части коллекции. В диалоговом окне **Выбор рисунка** перейдите к нужному изображению и нажмите кнопку **Открыть**.

Выбранное изображение будет добавлено в отчет.

Добавление фонового изображения

1. В области навигации щелкните правой кнопкой мыши отчет, в который требуется добавить фоновое изображение, и выберите команду **Режим макета**.

2. На вкладке **Формат** в группе **Фон** нажмите кнопку **Фоновый рисунок**.

3. Выполните одно из указанных ниже действий.

– **Используйте существующее изображение.** Если нужное изображение уже есть в коллекции, щелкните его, чтобы добавить в отчет.

– **Добавьте новое изображение.** Нажмите кнопку **Обзор** в нижней части коллекции. В диалоговом окне **Выбор рисунка** перейдите к нужному изображению и нажмите кнопку **Открыть**.

Выбранное изображение будет добавлено в отчет.

18.5 Просмотр и печать отчета

Предварительный просмотр отчета

1. Откройте отчет, который требуется просмотреть, или просто выберите его в области навигации.

2. На вкладке **Файл** выберите команду **Печать**, а затем – **Предварительный просмотр**.

Отчет будет открыт в режиме предварительного просмотра. С помощью команд на вкладке **Предварительный просмотр** можно выполнить следующие действия:

- напечатать отчет;
- изменить размер или макет страницы;
- изменить масштаб или просмотреть сразу несколько страниц;
- обновить данные в отчете;
- экспортировать отчет в другой формат файла.

3. Чтобы вернуться в рабочую область базы данных, на вкладке **Предварительный просмотр** в группе **Закрыть** выберите команду **Закрыть окно предварительного просмотра**.

Печать отчета

Отчет можно вывести на печать не только из режима предварительного просмотра.

1. Откройте отчет, который требуется распечатать, или просто выберите его в области навигации.

2. На вкладке **Файл** выберите команду **Печать**.

- Чтобы сразу отправить отчет на принтер по умолчанию, не настраивая его параметры, выберите команду **Быстрая печать**.
- Чтобы открыть диалоговое окно, в котором можно выбрать принтер, указать число копий и задать другие параметры, выберите команду **Печать**.

19 МАКРОСЫ

Макрос – это средство для автоматизации задач и добавления функциональных возможностей в формы, отчеты и элементы управления. Например, при добавлении командной кнопки в форму событие кнопки **OnClick** связывается с макросом, который содержит команды, выполняемые при каждом нажатии кнопки.

В приложении Access макросы можно рассматривать как упрощенный язык программирования, на котором программа записывается в виде списка макрокоманда для выполнения. При создании макроса каждая макрокоманда выбирается из раскрывающегося списка, после чего к ней добавляется необходимая информация. Макросы позволяют добавлять функциональные возможности в формы, отчеты и элементы управления без необходимости написания кода в модуле Visual Basic для приложений (VBA). Макросы обеспечивают выполнение части команд, доступных в VBA, и для большинства пользователей создание макроса оказывается проще, нежели написание кода VBA.

Термин «макрос» часто используется по отношению к изолированным макрообъектам (то есть объектам, отображаемым в области переходов в разделе **Макросы**), но на самом деле, один макрообъект может содержать несколько макросов. В этом случае он называется *группой макросов*. Группа макросов отображается в панели переходов как один объект, хотя в действительности содержит несколько макросов. Разумеется, каждый макрос может быть создан как отдельный макрообъект, но зачастую имеет смысл сгруппировать несколько связанных макросов в один макрообъект. Имя в столбце **Имя макроса** идентифицирует каждый макрос.

Макрос состоит из отдельных макрокоманд. Для большинства макрокоманд требуется один или несколько аргументов. Каждому макросу в группе может быть присвоено имя и добавлены условия для контроля за выполнением каждой макрокоманды.

Предположим, требуется запустить отчет непосредственно из формы ввода данных. Можно добавить в форму кнопку и затем создать макрос, который будет запускать отчет. Макрос может быть изолированным (отдельный объект в базе данных), который затем связывается с событием **OnClick** для кнопки, или же он может быть внедрен непосредственно в событие кнопки **OnClick** – это новая возможность в Office Access 2007. В любом случае, при нажатии кнопки выполняется макрос, который запускает отчет.

Макрос создается с помощью построителя макросов, показанного на следующем рисунке 30.

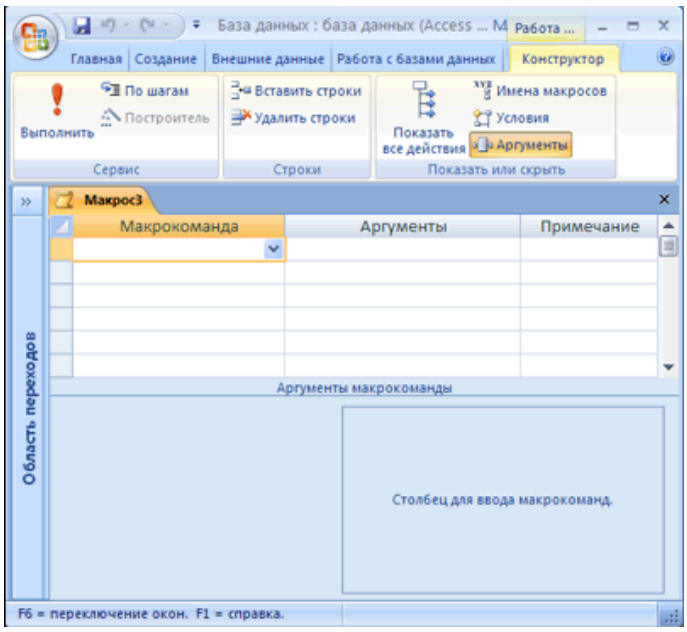


Рисунок 30 – Окно построителя макросов

Чтобы отобразить построитель макросов

– На вкладке **Создание** в группе **Другие** щелкните **Макрос**. Если эта команда недоступна, щелкните стрелку либо под кнопкой **Модуль**, либо под кнопкой **Модуль класса**, а затем щелкните **Макрос**.

Имена макросов

Если макрообъект содержит только один макрос, имя макроса излишне. Для макроса может использоваться имя макрообъекта. Однако в случае группы макросов необходимо присвоить каждому макросу уникальное имя. Если столбец **Имя макроса** не отображается в построителе макросов, нажмите кнопку **Имена макросов** в группе **Отображение** на вкладке **Конструктор**.

Аргументы

Аргумент – это значение, которое обеспечивает необходимую для макрокоманды информацию, например, какая строка должна отображаться в окне сообщения, с каким элементом управления следует выполнять действия и т. п. Некоторые аргументы являются обязательными, другие – нет. Аргументы отображаются в области **Аргументы макрокоманды** в нижней части окна построителя макросов (рисунок 31).

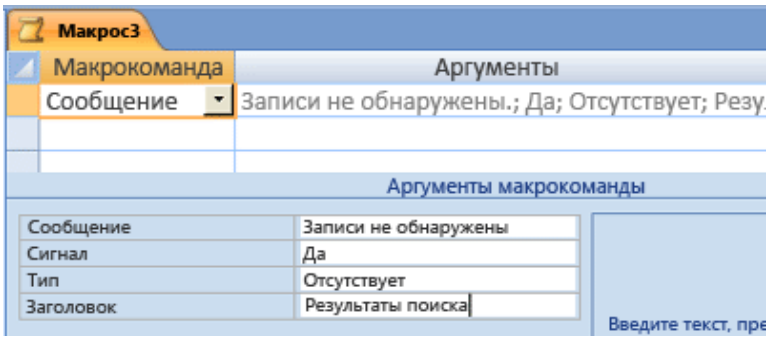


Рисунок 31 – Аргументы макрокоманды

В построителе макросов в Office Access 2007 имеется столбец **Аргументы**, позволяющий просматривать (но не изменять) аргументы макрокоманды в строке макрокоманды. Так легче читать макрос, так как уже не нужно выбирать каждую макрокоманду, чтобы увидеть ее аргументы. Чтобы отобразить столбец **Аргументы**, нажмите кнопку **Аргументы** в группе **Отображение** на вкладке **Конструктор**.

Условия

Условие определяет требования, которые должны быть соблюдены, для того чтобы была выполнена макрокоманда. Можно использовать любое выражение, результатом которого являются значения «Истина» или «Ложь» либо «Да» или «Нет». Если выражение вычисляется как «Ложь», «Нет» или 0 (нуль), макрокоманда не будет выполнена. При любом другом значении выражения макрокоманда будет выполнена.

Одно условие может управлять несколькими макрокомандами, если в столбце **Условие** ввести многоточие (...) для каждой следующей по порядку макрокоманды, к которой будет применяться данное условие. Если выражение вычисляется как «Ложь», «Нет» или 0 (нуль), ни одна из макрокоманд не будет выполнена. При любом другом значении выражения будут выполнены все макрокоманды.

Для отображения в построителе макросов столбца **Условия** на вкладке **Конструктор** в группе **Отображение** нажмите кнопку **Условия** (таблица 26).

Таблица 26. Примеры приводимых условий для макрокоманд

Выражение, используемое как условие	**Условие, при котором выполняется макрокоманда**
[Город]="Париж"	Поле «Город» в форме, из которой запускается макрос, имеет значение «Париж».
DCount("[КодЗаказа]", "Заказы")>35	Количество записей в поле «КодЗаказа» таблицы «Заказы» превышает 35.
DCount("*", "Заказано", "[КодЗаказа]=Forms![КодЗаказа]![КодЗаказа]")>3	В таблице «Заказано» содержится более 3 записей, у которых поле «КодЗаказа» в таблице соответствует полю «КодЗаказа» в форме «КодЗаказа».
[ДатаИсполнения] Between #2–фев–2015# And #2–мар–2015#	Значение поля «ДатаИсполнения» в форме, из которой запускается макрос, попадает в интервал со 2 февраля 2015 по 2 марта 2015 г.
Forms![Товары]![На складе]<5	Значение поля «На складе» в форме «Товары» меньше 5.
IsNull([Имя])	Поле «Имя» в форме, из которой запускается макрос, имеет значение Null (не заполнено). Это выражение эквивалентно следующему: [Имя] Is Null.

[Страна]="UK" And Forms![СуммыПродаж]![ВсегоЗаказов]>100	Поле «Страна» в форме, из которой запускается макрос, имеет значение «Великобритания», и значение поля «ВсегоЗаказов» в форме «СуммыПродаж» больше 100.
[Страна] In ("Франция", "Италия", "Испания") And Len([Индекс])<>5	Поле «Страна» в форме, из которой запускается макрос, имеет значение «Франция», «Италия» или «Испания», и почтовый индекс содержит не 5 символов.
MsgBox("Подтвердить изменения?",1)=1	Нажата кнопка **ОК** в диалоговом окне, в котором функция **MsgBox** отображает текст «Подтвердить изменения?». Если в диалоговом окне нажата кнопка **Отмена** эта макрокоманда будет пропущена.
[TempVars]![MsgBoxResult]=2	Временная переменная, которая используется для хранения результата окна сообщения, равна 2 (vbCancel=2).

Макрокоманды

Макрокоманды – это простейшие элементы, из которых строится макрос. В приложении Access предусмотрен большой выбор макрокоманд, которые позволяют выполнять разнообразные действия. Например, наиболее часто используются макрокоманды для открытия отчета, поиска записи, отображения окна сообщения или применения фильтра к форме или отчету.

19.1 Новые возможности макросов в Office Access 2007

В предыдущих версиях Access для выполнения наиболее часто используемых функций требовалось написание кода VBA. В Office Access 2007 включены новые возможности и макрокоманды, которые позволяют избежать этого. Благодаря им процесс добавления функциональных возможностей в базу данных стал проще и надежнее.

- **Внедренные макросы**

В новой версии появилась возможность внедрять макросы в любые события, предусмотренные в форме, отчете или элементе управления. Внедренный макрос не отображается в области переходов, он становится компонентом формы, отчета или элемента управления, в которых он был создан. При создании копии формы, отчета или элемента управления, содержащих внедренные макросы, в этой копии также будут содержаться макросы.

- **Усиленная защита**

Если кнопка **Отобразить все макрокоманды**  в построителе макросов не выделена, будут доступны только те макрокоманды и аргументы макрокоманды «Выполнить Команду», для выполнения которых не требуется присваивать им состояние надежных. Макрос, построенный из таких макрокоманд, будет выполняться, даже если база данных находится в режиме блокировки

выполнения программ (когда выполнение кода VBA запрещено). Базам данных с макрокомандами, не включенными в список надежных, – или базам данных с кодом VBA – необходимо явно присвоить состояние надежных баз данных.

– **Обработка ошибок и отладка**

В Office Access 2007 включены новые макрокоманды, в том числе **При Ошибке** (аналог оператора On Error в VBA) и **Устранить Ошибку Макроса**, которые позволяют выполнять определенные макрокоманды при ошибке выполнения макроса. Кроме того, новая макрокоманда **Шаг** позволяет включить пошаговый режим в любом месте макроса и наблюдать за последовательным выполнением отдельных макрокоманд.

– **Временные переменные**

Три новые макрокоманды (**Задать Врем Переменную**, **Удалить Врем Переменную** и **Удалить Все Врем Переменные**) дают возможность создавать и использовать в макросе временные переменные. Они могут использоваться в условных выражениях для управления выполнением макроса, для передачи данных в формы или отчеты и обратно, а также для любых других целей, которые требуют временного места для хранения значения. Эти временные переменные доступны также в VBA, поэтому могут использоваться для обмена данными с модулями VBA.

19.2 Создание макроса

В Office Access 2007 макрос или группа макросов могут быть заключены в объект макроса (называемый иногда изолированным макросом), или макрос может быть внедрен в любое свойство события в форме, отчете или элементе управления. Внедренные макросы становятся частью объекта или элемента управления, в которые они внедрены. Изолированные макросы отображаются в области переходов в разделе **Макросы**, в отличие от внедренных.

Функции построителя макросов

Построитель макросов служит для создания и изменения макросов. Он открывается следующим образом.

- На вкладке **Создание** в группе **Другие** щелкните **Макрос**. Если эта команда недоступна, щелкните стрелку либо под кнопкой **Модуль**, либо под кнопкой **Модуль класса**, а затем щелкните **Макрос**. 

Приложение Access откроет построитель макросов.

Построитель макросов используется для создания списка макрокоманд, которые должны выполняться при запуске макроса. При первом открытии построителя макросов отображаются столбцы **Макрокоманда**, **Аргументы** и **Примечание**.

В области **Аргументы макрокоманды** при необходимости вводятся или изменяются аргументы для каждой макрокоманды. В поле описания отображается краткое описание каждой макрокоманды и аргумента. Для просмотра описания щелкните макрокоманду или аргумент.

В следующей таблице 27 показаны команды, доступные на вкладке **Конструктор** в построителе макросов.

Таблица 27. Команды, доступные на вкладке Конструктор

Группа	Команда	Описание
Сервис	**Выполнить**	Выполнение макрокоманд, перечисленных в макросе.
	Пошагово	Включение режима пошагового выполнения макроса. При запуске макроса в этом режиме происходит поочередное выполнение каждой макрокоманды. После завершения каждой макрокоманды отображается диалоговое окно **Пошаговое исполнение макроса**. Для перехода к следующей макрокоманде нажмите в этом диалоговом окне кнопку **Шаг**. Нажмите кнопку **Остановить все макросы**, чтобы остановить исполнение этого и всех остальных макросов. Нажмите кнопку **Далее**, чтобы выйти из пошагового режима и выполнить остальные макрокоманды без остановки.
	Построитель	Эта кнопка активируется при вводе аргумента макрокоманды, содержащего выражение. Щелкните кнопку **Построитель**, чтобы открыть диалоговое окно **Построитель выражений**, в котором можно создать выражение.
Строки	**Вставить строки**	Добавление одной или нескольких пустых строк макрокоманд над выбранными строками.
	Удалить строки	Удаление выбранных строк макрокоманд.
Отображение	**Отобразить все макрокоманды**	Отображение большего или меньшего количества макрокоманд в раскрывающемся списке **Макрокоманда**. – Для отображения более длинного списка макрокоманд нажмите кнопку **Отобразить все макрокоманды**. Если выбран полный список макрокоманд, кнопка **Отобразить все макрокоманды** выделена. При выборе макрокоманды в полном списке перед ее выполнением может потребоваться предоставить базе данных надежное состояние явным образом. – Для отображения краткого списка, включающего только макрокоманды, которые могут использоваться в базе данных, не имеющей состояния надежной, кнопка **Отобразить все макрокоманды** не должна быть выделена. **Совет** Если кнопка **Отобразить все макрокоманды** выделена, снимите выделение, нажав эту кнопку. Если кнопка **Отобразить все макрокоманды** не выделена, доступен краткий список макрокоманд для надежной базы данных.
	Имена макросов	Отображение или скрытие столбца **Имя макроса**. Имена макросов необходимы в группе, чтобы можно было различать отдельные макросы, в противном случае они не обязательны.
	Условия	Отображение или скрытие столбца **Условие**. Этот столбец служит для ввода выражений, которые определяют условия выполнения макрокоманды.
	Аргументы	Отображение или скрытие столбца **Аргументы**. В этом столбце отображаются аргументы для каждой макрокоманды, что упрощает просмотр макроса. Если столбец **Аргументы** не отображается, приходится щелкать каждую макрокоманду и просматривать аргументы в разделе **Аргументы макрокоманды**. Ввод аргументов в столбце **Аргументы** невозможен.

19.3 Создание изолированного макроса

1. На вкладке **Создание** в группе **Другие** щелкните **Макрос**. Если эта команда недоступна, щелкните стрелку либо под кнопкой **Модуль**, либо под кнопкой **Модуль класса**, а затем щелкните **Макрос**.

Будет открыт построитель макросов.

2. Добавьте в макрос макрокоманду.

– В построителе макросов щелкните первую пустую ячейку в столбце **Макрокоманда**.

– Введите нужную макрокоманду или щелкните стрелку, чтобы раскрыть список действий и выбрать из него нужную макрокоманду.

– В разделе **Аргументы макрокоманды** при необходимости укажите аргументы для макрокоманды. Щелкните поле аргумента, чтобы увидеть его краткое описание справа от аргумента.

3. Чтобы добавить в макрос следующую макрокоманду, перейдите в другую строка действия и повторите шаг 2. Приложение Access выполняет макрокоманды в том порядке, в котором они перечислены.

19.4 Создание группы макросов

Для группировки нескольких связанных макросов в один объект макроса можно создать группу макросов.

1. На вкладке **Создание** в группе **Другие** щелкните **Макрос**. Если эта команда недоступна, щелкните стрелку либо под кнопкой **Модуль**, либо под кнопкой **Модуль класса**, а затем щелкните **Макрос**.

Будет открыт построитель макросов.

2. На вкладке **Конструктор** в группе **Отображение** нажмите кнопку **Имя макроса**, если она еще не была нажата. В построителе макросов будет отображен столбец **Имя макроса**.

3. В столбце **Имя макроса** введите имя первого макроса из группы.

4. Добавьте макрокоманды для исполнения в первом макросе.

– В столбце **Макрокоманда** щелкните стрелку, чтобы раскрыть список макрокоманд.

– Выберите макрокоманду, которую требуется добавить.

– В разделе **Аргументы макрокоманды** укажите аргументы макрокоманды, если они нужны. Щелкните поле аргумента, чтобы увидеть его краткое описание справа от аргумента.

Советы

– Для аргументов макрокоманд, параметры которых служат именами объектов базы данных, можно задать аргумент, перетащив объект из области переходов в поле аргумента **Имя объекта**.

– Можно также создать макрокоманду, перетащив объект базы данных из области переходов в пустую строку в построителе макросов. При перетаскивании таблицы, запроса, формы, отчета или модуля в построитель макросов

добавляется макрокоманда, открывающая таблицу, запрос, форму или отчет. При перетаскивании макроса в построитель макросов добавляется макрокоманда, запускающая макрос.

– При необходимости введите примечания к макрокоманде.

5. Перейдите к следующей пустой строке в столбце **Имя макроса** и введите имя очередного макроса из группы.

6. Добавьте макрокоманды для выполнения в этом макросе.

7. Повторите действия 5 и 6 для каждого макроса в группе.

19.5 Создание внедренного макроса

Внедренные макросы отличаются от изолированных тем, что хранятся в свойствах событий в формах, отчетах или элементах управления. Они не отображаются как объекты в разделе **Макросы** в области переходов. Это облегчает управление базой данных, поскольку отпадает необходимость отслеживать отдельные макрообъекты, содержащие макросы для формы или отчета. Используйте следующую процедуру для создания внедренного макроса.

1. Откройте форму или отчет, в которые требуется внедрить макрос, в режиме конструктора или макета. Для этого щелкните форму или отчет в области переходов правой кнопкой мыши и выберите пункт **Режим конструктора**  или **Режим макета**  .

2. Если страница свойств еще не отображена, для ее отображения нажмите клавишу F4.

3. Щелкните элемент управления или раздел, содержащий свойство события, в которое следует внедрить макрос.

Чтобы выбрать всю форму или отчет, выберите **Отчет** в раскрывающемся списке вверху страницы свойств.

4. Откройте вкладку **Событие** на странице свойств.

5. Щелкните свойство события, в которое нужно внедрить макрос, и нажмите кнопку  рядом с полем.

6. В диалоговом окне **Построитель** выделите пункт **Макросы** и затем нажмите кнопку **ОК**.

7. В построителе макросов щелкните первую строку в столбце **Макрокоманда**.

8. В раскрывающемся списке **Макрокоманда** выберите нужную макрокоманду.

9. Укажите в области **Аргументы макрокоманды** все требуемые аргументы и перейдите к следующей строке макрокоманды.

10. Повторяйте шаги 8 и 9, пока не завершите построение макроса.

11. Нажмите кнопку **Сохранить**  , затем нажмите кнопку **Закрыть**  .

Макрос будет выполняться каждый раз при возникновении события.

Приложение Access позволяет создавать группу макросов как внедренный макрос. Однако при возникновении события выполняется только первый

макрос из группы. Остальные макросы игнорируются, если они не вызываются из самого внедренного макроса (например, макрокомандой **ПриОшибке**).

Пример. Внедрение макроса в событие отчета «Отсутствие данных»

Если при запуске отчета в источнике данных отсутствуют записи, отображается пустая страница отчета – то есть страница, на которой отсутствуют данные. Иногда желательно, чтобы вместо пустой страницы отображалось окно сообщения. Идеальным решением в этой ситуации является использование внедренного макроса.

1. Откройте отчет в режиме конструктора или макета.
2. Если страница свойств еще не отображена, для ее отображения нажмите клавишу F4.
3. Откройте вкладку **Событие** на странице свойств.
4. Выберите событие **Отсутствие данных**.
5. Нажмите кнопку ....
6. В диалоговом окне **Построитель** выделите пункт **Макросы** и затем нажмите кнопку **ОК**.
7. Введите макрокоманды и аргументы из следующей таблицы 28.

Таблица 28. Макрокоманды и аргументы для примера Внедрение макроса в событие отчета «Отсутствие данных»

Макрокоманда	**Аргументы**
MsgBox	«Записи не обнаружены.»; «Да»; «Сведения»; «Нет данных»
ОтменитьСобытие	[аргументы отсутствуют]

1. Обратите внимание, что в предыдущей таблице аргументы представлены в том виде, в каком они отображаются в столбце **Аргументы**. Фактически они вводятся в разделе **Аргументы макрокоманды**, как показано в следующей таблице 29.
2.

Таблица 28. Аргументы макрокоманды

Аргумент макрокоманды	**Значение**
Сообщение	Записи не обнаружены.
Сигнал	Да
Тип	Сведения
Заголовок	Нет данных

1. Нажмите кнопку **Закрыть**.

Построитель макросов будет закрыт, а для события **Отсутствие данных** будет выведено сообщение **[Внедренный макрос]**.

2. Сохраните и закройте отчет.

Если при следующем запуске отчета записи не будут обнаружены, появится окно сообщения. При нажатии в этом окне кнопки **ОК** отчет будет отменен без отображения пустой страницы.

19.6 Изменение макроса

– **Вставка строки макрокоманды** Щелкните строку макроса, выше которой нужно вставить новую макрокоманду, и на вкладке **Конструктор** в группе **Строки** нажмите кнопку **Вставить строки**  .

– **Удаление строки макрокоманды** Щелкните строку макрокоманды, которую нужно удалить, и на вкладке **Конструктор** в группе **Строки** нажмите кнопку **Удалить строки**  .

– **Перемещение строки макрокоманды** Выберите макрокоманду, щелкнув область выделения строки слева от имени макрокоманды. Перетащите область выделения строки, чтобы переместить макрокоманду в другое положение.

Дополнительные сведения о макрокомандах

В построителе макросов можно получить дополнительные сведения о макрокоманде или аргументе, выделив нужный элемент и ознакомившись с описанием в поле, расположенном в правом нижнем углу окна построителя макросов. Кроме того, каждой макрокоманде посвящен отдельная статья справки. Для получения дополнительных сведений о макрокоманде щелкните макрокоманду в списке и затем нажмите клавишу F1.

19.7 Запуск макроса

Изолированные макросы можно запустить несколькими способами: непосредственно (например, из области переходов), из группы макросов, из другого макроса, из модуля VBA или в ответ на событие в форме, отчете или элементе управления. Для запуска макроса, внедренного в форму, отчет или элемент управления, нажмите кнопку **Выполнить** на вкладке **Конструктор**, когда макрос находится в режиме конструктора. Иначе этот макрос будет выполняться только по событию, с которым он связан.

Непосредственный запуск макроса

Для непосредственного запуска макроса выберите один из следующих вариантов.

– Найдите макрос в области переходов и дважды щелкните его имя.

– На вкладке **Инструменты базы данных** в группе **Макрос** нажмите кнопку **Выполнить макрос**  , выберите макрос в списке **Имя макроса** и нажмите кнопку **ОК**.

– Если макрос открыт в режиме конструктора, нажмите кнопку **Выполнить** на вкладке **Конструктор** в группе **Инструменты**. Чтобы открыть макрос в режиме конструктора, щелкните его правой кнопкой мыши в области переходов и нажмите кнопку **Конструктор**  .

Запуск макроса, находящегося в группе макросов

Чтобы запустить макрос, находящийся в группе макросов, выполните одно из следующих действий.

– На вкладке **Инструменты базы данных** в группе **Макрос** нажмите кнопку **Выполнить макрос**  и выберите нужный макрос в списке **Имя макроса**.

Для каждого макроса в группе макросов используется формат *Имя Группы Макросов. Имя Макроса*.

– Нажмите кнопку **ОК**.

– Укажите макрос в качестве значения свойство события в форме или отчете или в качестве аргумента «Имя макроса» макрокоманды **Выполнить Макрос**. При этом используется следующий синтаксис:

Имя Группы Макросов. Имя Макроса

Например, для запуска макроса «Категории» в группе «Кнопки кнопочной формы» используется следующее значение свойства события:

Кнопки кнопочной формы.Категории

– Запустите макрос, находящийся в группе макросов, из процедуры VBA с помощью метода **RunMacro** объекта **DoCmd**, используя при этом описанный выше синтаксис для ссылки на макрос.

Запуск макроса из другого макроса или из процедуры VBA

Добавление макрокоманды **Выполнить Макрос** в макрос или процедуру.

– Для добавления макрокоманды **ВыполнитьМакрос** в макрос выберите еев списке макрокоманд в пустой строке макрокоманды и укажите в аргументе **Имя макроса** имя запускаемого макроса.

– Для добавления макрокоманды **ВыполнитьМакрос** в процедуру VBA включите в процедуру метод **RunMacro** объекта **DoCmd** и укажите имя запускаемого макроса. Например, следующий экземпляр метода **RunMacro** запускает макрос «Мой макрос»:

DoCmd. RunMacro "My Macro"

Запуск макроса в ответ на событие в форме, отчете или элементе управления

Несмотря на новую возможность внедрять макросы непосредственно в свойства событий в формах, отчетах и элементах управления можно, как и в предыдущих версиях Access, создавать изолированные макросы и затем связывать их с событиями.

1. После создания изолированного макроса откройте форму или отчет в режиме конструктора или макета.
2. Откройте окно свойств для формы, отчета, раздела или элемента управления в форме или отчете.
3. Откройте вкладку **Событие**.
4. Выберите свойство события, которое будет запускать макрос. Например, чтобы запустить макрос в ответ на событие **Изменение**, выберите свойство **Изменение (On Change)**.
5. Выберите имя изолированного макроса в раскрывающемся списке.
6. Сохраните форму или отчет.

20 СПОСОБЫ СОВМЕСТНОГО ИСПОЛЬЗОВАНИЯ БАЗЫ ДАННЫХ ACCESS

База данных Access состоит из объектов, например таблиц, запросов и форм. В таблицах хранятся данные. Все остальные виды объектов баз данных помогают использовать данные, хранящиеся в таблицах. Если требуется совместно использовать базу данных, обычно необходимо обеспечить совместное использование таблиц, поскольку в них содержатся данные. При совместном использовании таблиц важно, чтобы все использовали одни и те же таблицы, т. е. одни и те же данные.

Другие объекты баз данных (запросы, формы, отчеты и т. д.) не содержат данных, поэтому использование одних и тех же объектов играет не такую важную роль. На самом деле может быть полезно использовать различные копии таких объектов. В зависимости от требований и ресурсов возможны различные способы совместного использования базы данных.

20.1 Способы совместного использования

– **Разделение базы данных.** Этот вариант подходит в случае отсутствия сайта SharePoint или сервера баз данных. Таблицы сохраняются в одном файле Access, а все остальное – в другом файле, который называется внешней базой данных. Во внешней базе данных содержатся ссылки на таблицы из другого файла. Каждый пользователь получает собственную копию внешней базы данных, поэтому совместно используются только таблицы.

– **Сетевая папка.** Это самый простой вариант с минимальными требованиями, но он обеспечивает наименьшую функциональность. Файл базы данных хранится на общем сетевом диске, и пользователи одновременно используют этот файл. При одновременном изменении данных несколькими пользователями могут возникать проблемы с надежностью и доступностью. Все объекты базы данных используются совместно.

– **Узел SharePoint.** При наличии сервера с SharePoint (особенно со службами Access – новым компонентом сервера SharePoint Server) есть несколько хороших вариантов. Возможности интеграции с SharePoint помогают обеспечить более удобный доступ к базе данных.

Существует три способа совместного использования SharePoint.

– **Публикация базы данных с помощью служб Access.** При публикации базы данных она размещается в Интернете. Можно создавать веб-формы и отчеты, запускаемые в окне браузера, а также стандартные объекты Access (их иногда называют клиентскими объектами, чтобы отличать их от веб-объектов). Для использования стандартных объектов Access необходимо установить приложение Access, однако поскольку файл базы данных хранится на сайте Share Point, все объекты базы данных используются совместно.

База данных, которая совместно используется в Интернете и содержит клиентские объекты, называется гибридной. Пользователи, у которых не

установлено приложение Access, могут использовать только объекты веб-базы данных, в то время как пользователям, у которых оно установлено, доступны все объекты базы данных.

– **Сохранение базы данных в библиотеке документов.** Этот способ похож на сохранение базы данных в сетевой папке. Чтобы повысить доступность данных, вместо таблиц Access можно использовать для хранения данных списки SharePoint.

– **Связывание со списками.** При связывании со списками SharePoint совместно используются только данные, но не объекты базы данных. Каждый пользователь получает собственную копию базы данных.

– **Сервер баз данных.** Этот способ напоминает разделение баз данных в том, что таблицы хранятся в сети, а у каждого пользователя есть локальная копия файла базы данных Microsoft Access, содержащая ссылки на таблицы, запросы, формы, отчеты и другие объекты базы данных. Этот вариант следует использовать, если доступен сервер баз данных, а у всех пользователей установлено приложение Access. Серверы баз данных оптимизированы для совместного использования данных большим количеством пользователей. Преимущества зависят от используемого программного обеспечения сервера баз данных, но в общем случае они включают наличие учетных записей пользователей и избирательный доступ к данным, отличную доступность данных и удобные встроенные средства управления данными. Более того, большинство серверных приложений для работы с базами данных нормально работают с более ранними версиями Access, поэтому не требуется, чтобы все пользователи работали с одной и той же версией. Совместно используются только таблицы.

Таблица 30. Факторы, которые следует учитывать, при организации работы с сервером баз данных

	Разделение базы данных	**Сетевая папка**	**Узел SharePoint**	**Сервер баз данных**
Необходимость установки сервера баз данных	Нет	Нет	Нет	Да
Необходимость установки SharePoint Foundation (ранее – службы Windows SharePoint)	Нет	Нет	Да	Нет

Продолжение таблицы 30

Необходимость наличия служб Access на сервере SharePoint Server	Нет	Нет	Зависит от сценария: связывание со списками и сохранение в библиотеке документов не требует наличия служб Access; публикация в виде веб-базы данных требует наличия служб Access.	Нет
Доступность данных	Хорошая	Подходит для небольших групп, если данные мало изменяются	Наилучшая. Подходит для сценариев автономного использования.	Наилучшая
Безопасность	Зависит от дополнительных мер	Наименее безопасный способ	Наилучшая	Наилучшая
Гибкость	Гибкий способ. Можно легко разрабатывать новые функции базы данных без нарушения работы. Каждый пользователь может изменять структуру объектов в собственной копии.	Менее гибкий способ. Разработку можно осуществлять с использование м автономной копии базы данных, которая затем заменяется. Отсутствует возможность индивидуального изменения структуры базы данных пользователями.	Гибкий способ. Для управления доступом и изменения структуры используются разрешения SharePoint. Можно использовать некоторые объекты базы данных (например, формы) через браузер.	Гибкий способ. Можно легко разрабатывать новые функции базы данных без нарушения работы. Каждый пользователь может изменять структуру объектов в собственной копии.

20.2 Разделение базы данных

При разделении базы данных она реорганизуется в два файла: внутреннюю базу данных, в которой содержатся таблицы данных, и внешнюю базу данных, в которой содержатся все остальные объекты базы данных (например, запросы, формы и отчеты). Каждый пользователь взаимодействует с данными путем с помощью локальной копии внешней базы данных.

Для разделения базы данных служит мастер разделения базы данных. После разделения базы данных внешняя база данных распространяется среди пользователей.

Преимущества разделения базы данных:

– **Повышенная производительность.** Производительность базы данных обычно существенно возрастает, поскольку по сети передаются только данные. В неразделенной базе данных, которая совместно используется с помощью сетевой папки, по сети передаются сами объекты базы данных (таблицы, запросы, формы, отчеты, макросы и модули), а не только данные.

– **Улучшенная доступность.** Поскольку по сети передаются только данные, транзакции базы данных (например, изменение записей) выполняются быстрее, что повышает доступность данных для изменения.

– **Повышенная безопасность.** Если база данных с таблицами хранится на компьютере с файловой системой NTFS, для защиты данных можно использовать средства обеспечения безопасности NTFS. Поскольку пользователи обращаются к внутренней базе данных с помощью связанных таблиц, снижается вероятность получения злоумышленниками несанкционированного доступа к данным путем кражи внешней базы данных или входа в качестве авторизованного пользователя.

– **Повышенная надежность.** Если у пользователя возникает проблема и база данных неожиданно закрывается, любое повреждение файла базы данных обычно ограничивается копией внешней базы данных, открытой пользователем. Поскольку пользователь обращается к данным внутренней базы данных с помощью связанных таблиц, вероятность повреждения файла внутренней базы данных существенно снижается.

– **Гибкая среда разработки.** Поскольку каждый пользователь работает с локальной копией внешней базы данных, он независимо разрабатывает запросы, формы, отчеты и другие объекты базы данных, не влияя на работу других пользователей. Аналогично можно разработать и распространить новую версию внешней базы данных без нарушения доступа к данным, которые хранятся во внутренней базе данных.

20.3 Совместное использование базы данных с помощью сетевой папки

Самый простой способ совместного использования базы данных – поместить ее в общую сетевую папку. Однако этот способ является не только самым простым и наименее требовательным, но и наименее надежным. Перед его использованием необходимо обеспечить выполнение всех условий, указанных ниже.

– Базу данных должны использовать одновременно только несколько человек.

– В базе данных не должно быть полей MEMO, а если они есть, разные пользователи не должны изменять их одновременно.

– Пользователям не требуется настраивать структуру базы данных.

Этот способ менее безопасен по сравнению с остальными способами совместного доступа к базе данных, поскольку у каждого пользователя есть полная копия файла базы данных, что повышает риск несанкционированного доступа.

Совместное использование базы данных с помощью сетевой папки

1. Если общая сетевая папка отсутствует, ее нужно настроить.

Дополнительные сведения об этом см. в справке по операционной системе компьютера, который будет использоваться для совместного доступа к базе данных. Если общая папка находится на сетевом сервере, может потребоваться помощь администратора сети.

2. Приложение Access должно быть настроено для открытия в режиме совместного доступа на компьютерах всех пользователей. Данный режим используется по умолчанию, однако это необходимо проверить: если пользователь откроет базу данных в монопольном режиме, другие пользователи не смогут работать с данными. Выполните на каждом из компьютеров действия, указанные ниже.

a. Запустите приложение Access.

b. Откройте вкладку **Файл** и нажмите кнопку **Параметры**.

c. В левой области диалогового окна **Параметры Access** щелкните элемент **Параметры клиента**.

d. В правой области диалогового окна **Параметры Access** в категории **Дополнительно** в группе **Режим открытия по умолчанию** выберите пункт **Общий доступ**.

e. Нажмите кнопку **ОК** и закройте приложение Access.

3. Скопируйте файл базы данных в общую папку. Затем настройте атрибуты файла таким образом, чтобы разрешить доступ к файлу базы данных для чтения и записи. Для использования базы данных необходим доступ к ней с правами на чтение и запись.

4. На компьютере каждого пользователя создайте ярлык для файла базы данных.

При указании пути к файлу базы данных в свойстве ярлыка **Объект** используйте вместо буквы подключенного диска UNC–адрес. Например, вместо пути **F:\sample.accdb** укажите путь **\\имя_компьютера\shared.accdb**.

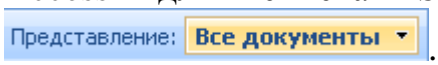
20.4 Совместное использование базы данных с помощью сайта SharePoint

Приложение SharePoint позволяет использовать преимущества централизованного размещения данных. Кроме того, можно использовать функции Share Point для управления разрешениями пользователей и предоставления доступа к данным через веб-браузер.

Сохранение файла базы данных в библиотеке документов SharePoint

Этот способ упрощает управление доступом к базе данных. Базу данных можно сохранить в любой библиотеке документов SharePoint. Это позволяет интегрировать управление файлами баз данных Access с остальными документами и бизнес-данными.

Публикация базы данных на сайте SharePoint доступна только для файлов баз данных, сохраненных в формате Office Access 2007.

Например, если на сайте SharePoint есть списки, в которых отслеживаются проблемы обслуживания клиентов и хранятся данные о сотрудниках, можно создать в приложении Access базу данных, которая будет служить интерфейсом для этих списков. Можно создавать запросы Access для анализа этих проблем и отчеты Access для форматирования и публикации письменных отчетов для собраний групп. Если у пользователей на компьютерах установлено приложение Access, можно предоставить доступ к запросам и отчетам Access для списка SharePoint с помощью меню **Представление** 

. При просмотре списка на сайте SharePoint пользователи смогут находить и открывать запросы, отчеты и другие объекты Access в меню **Представление**. Если у пользователей нет приложения Microsoft Access, они все же смогут использовать данные из списков с помощью представлений SharePoint.

1. Откройте базу данных, которую требуется использовать совместно.
2. Откройте вкладку **Файл** и нажмите кнопку **Совместный доступ**.
3. В диалоговом окне **Сохранение базы данных** в разделе **Дополнительно** выберите вариант **SharePoint**.
4. В диалоговом окне **Сохранение в SharePoint** перейдите к соответствующей библиотеке документов.
5. Проверьте имя базы данных и тип ее файла, при необходимости измените их и нажмите кнопку **Сохранить**.

Перемещение данных в списки SharePoint и связывание с ними

Этот способ позволяет каждому пользователю изменять собственную копию базы данных, поскольку совместный доступ к данным осуществляется через сайт SharePoint. Хотя в этом случае отсутствуют преимущества, получаемые при публикации базы данных на сайте SharePoint, при этом достигается выгода централизованного расположения данных. Этот способ дает те же преимущества, что и использование разделенной базы данных. Кроме того, поскольку данные находятся в списках SharePoint, к ним можно предоставлять раздельный доступ по сети с использованием функций SharePoint.

Этот способ включает три основных действия.

1. Перемещение данных в списки SharePoint.
2. Создание ссылок на эти списки.
3. Распространение файла базы данных.

Для выполнения первых двух действий можно использовать мастер переноса на сайт SharePoint, а последнее действие можно выполнить с помощью любых доступных средств.

Использование мастера экспорта таблиц в SharePoint

1. На вкладке **Работа с базами данных** в группе **Перенос данных** щелкните элемент **SharePoint**.

Этот элемент доступен только в том случае, если файл базы данных сохранен в формате ACCDB.

2. Следуйте указаниям мастера экспорта таблиц в SharePoint; в частности, укажите расположение сайта SharePoint.

Чтобы отменить процесс, нажмите кнопку **Остановить**.

3. Чтобы просмотреть дополнительные сведения о переносе, на последней странице мастера установите флажок **Подробности**.

На этой странице содержатся сведения о том, какие таблицы связаны со списками, а также сведения о расположении резервных копий и URL-адрес базы данных. Здесь также выводится предупреждение при возникновении проблем с переносом и указывается расположение таблицы журнала, в которой можно просмотреть дополнительные сведения о проблемах.

4. Когда все действия мастера будут завершены, нажмите кнопку **Готово**.

Если мастер выведет предупреждение, следует просмотреть таблицу журнала и выполнить все действия, необходимые для успешного переноса данных. Например, может потребоваться отменить перенос некоторых полей или преобразовать их в другие типы данных, совместимые со списками SharePoint.

Чтобы просмотреть списки на сайте SharePoint, щелкните в области быстрого запуска кнопку **Списки** или выберите пункт **Просмотреть все содержимое узла**. Может потребоваться обновить страницу в веб-браузере. Чтобы отобразить списки в области быстрого запуска на сайте SharePoint или изменить другие параметры (например, включить отслеживание версий), можно изменить параметры списков на сайте SharePoint.

20.5 Использование Access с сервером базы данных

Совместное использование базы данных можно организовать с помощью приложения Access и сервера баз данных (например, сервера SQL Server). Этот способ обеспечивает много преимуществ, но для него требуется дополнительное программное обеспечение – сервер баз данных.

Преимущества совместного использования базы данных с помощью сервера баз данных

– **Высокая производительность и масштабируемость.** Во многих случаях сервер баз данных обеспечивает более высокую производительность, чем простой файл базы данных Access. Большинство серверов баз данных также обеспечивают поддержку очень крупных баз данных размером до терабайта, что приблизительно в 500 раз превышает текущий предел для баз данных

Access (2 ГБ). В целом серверы баз данных работают весьма эффективно, обрабатывая запросы параллельно (с использованием нескольких естественных потоков команд в одном процессе для обработки запросов пользователя) и сводя к минимуму дополнительные требования к памяти при добавлении пользователей.

– **Повышенная доступность.** В большинстве серверов баз данных предусмотрено резервное копирование базы данных во время ее использования. В результате для резервного копирования данных пользователям не обязательно выходить из базы данных. Кроме того, обычно серверы баз данных очень эффективно работают с одновременным изменением и блокировкой записей.

– **Улучшенная безопасность.** Ни одну базу данных нельзя защитить полностью. Однако серверы баз данных обеспечивают надежную защиту, которая позволяет предотвратить несанкционированное использование данных. В большинстве серверов баз данных используются средства обеспечения безопасности на основе учетных записей, что позволяет контролировать доступ пользователей к таблицам. Даже в случае незаконного получения доступа к интерфейсу Access несанкционированное использование данных будет предотвращено средствами защиты на уровне учетных записей.

– **Автоматическое восстановление.** На случай сбоя системы (например, при аварийном завершении работы операционной системы или отключении питания) в некоторых серверах баз данных предусмотрен механизм автоматического восстановления базы данных до последнего согласованного состояния всего за несколько минут без вмешательства администратора базы данных.

– **Обработка данных на сервере.** Использование приложения Access в конфигурации «клиент-сервер» уменьшает объем сетевого трафика благодаря обработке запросов базы данных на сервере перед отправкой результатов клиенту. Обычно сервер обрабатывает данные более эффективно, особенно при работе с большими наборами данных.

Основные этапы использования Access с сервером баз данных

Точный перечень действий, которые следует выполнить для использования приложения Access с сервером баз данных, зависит от используемого сервера баз данных, однако основные действия одни и те же.

1. Перенос данных из таблиц базы данных Access в таблицы на сервере баз данных.

2. Организация связи файла базы данных Access с таблицами сервера баз данных.

3. Создание соответствующих учетных записей пользователей на сервере баз данных.

4. Распространение файла базы данных Access.

5. Установка всех необходимых драйверов баз данных на компьютерах пользователей.

21 ЗАЩИТА БАЗЫ ДАННЫХ ACCESS 2007

В Office Access 2007 предусмотрена улучшенная модель безопасности, которая упрощает процесс защиты базы данных и ее открытия с включенной защитой.

Ниже приведен список новых средств обеспечения безопасности в Office Access 2007.

– Просмотр данных даже при отключенном коде Microsoft Visual Basic для приложений (VBA) или отключенных компонентах в базе данных. Если в Microsoft Office Access 2003 устанавливается уровень безопасности «Высокий», необходимо подписать кодом базу данных и предоставить ей состояние доверенной, чтобы можно было просмотреть данные. В Office Access 2007 можно открывать базы данных и просматривать данные без запроса о включении содержимого базы данных.

– Упрощенное открытие баз данных. Если файлы базы данных (как в новом формате Office Access 2007, так и в более ранних) расположены в надежном месте, например в папке или в общем сетевом ресурсе, которые указаны как надежные, они будут открываться и обрабатываться без сообщений с предупреждениями и запроса о включении или отключении содержимого. При открытии в Office Access 2007 баз данных из более ранних версий Access, например файлов с расширениями mdb или mde, которые имеют цифровую подпись и издатель которых считается надежным, такие файлы тоже доступны без вопросов о доверии. Однако следует помнить, что код VBA в подписанных базах данных не будет работать, пока издатель не будет признан надежным, а также в том случае, если подпись станет недействительной. Подпись становится недействительной, когда кто-либо, кроме подписавшего лица, выполняет недопустимые действия с содержимым базы данных.

– Центр управления безопасностью. «Центр управления безопасностью» – это диалоговое окно, в котором можно задавать и менять параметры безопасности в Access. Оно используется для создания или изменения надежных расположений, а также для настройки параметров безопасности для Office Access 2007. Эти параметры определяют поведение новых и существующих баз данных при их открытии в Access. Программные средства центра управления безопасностью позволяют оценить компоненты базы данных и определить, безопасно ли открывать базу данных и следует ли запретить пользователю включать ее.

– Меньше сообщений с предупреждениями. В предыдущих версиях Access пользователям приходилось иметь дело с различными предупреждающими сообщениями, касающимися, например, безопасности макросов и изолированного режима. По умолчанию при открытии базы данных Office Access 2007 вне доверенного расположения появляется единое средство, называемое «Панель сообщений» (рисунок 31).

Рисунок 31 - Панель сообщений

Если точно известно, что можно доверять содержимому базы данных, используйте средство «Панель сообщений», чтобы включить все компоненты – запросы на изменение (запросы, которые добавляют, удаляют или изменяют данные), макросы, элементы управления ActiveX, выражения (функции, возвращающие одно значение) и программы на VBA – при открытии базы данных, содержащей один или несколько этих компонентов.

– Новые способы подписи и распространения файлов, созданных в формате Office Access 2007. В предыдущих версиях Access для применения сертификата безопасности к индивидуальным компонентам базы данных использовался редактор Visual Basic. В Office Access 2007 она упаковывается, а затем подписывается и распространяется. При извлечении базы данных из подписанного пакета и перемещении в надежное расположение ее открытие происходит без отображения панели сообщений. Если база данных из подписанного пакета отправляется в ненадежное расположение, но имеется надежный сертификат пакета, и подпись действительна, то нет необходимости решать вопрос о доверии. Если упаковывается и подписывается база данных, не имеющая состояния доверенной или содержащая недействительную цифровую подпись, необходимо использовать панель сообщений для предоставления ей состояния доверенной каждый раз при ее открытии, за исключением тех случаев, когда она размещена в надежном расположении.

– Более стойкий алгоритм шифрования баз данных в формате Office Access 2007 с использованием пароля базы данных. В процессе шифрования происходит перемешивание данных в таблицах, что исключает несанкционированный просмотр этих данных.

– Новый подкласс макрокоманд, выполняющихся при отключенной базе данных. Эти безопасные макрокоманды включают также возможности исправления ошибок. Макросы (даже содержащие команды, которые Access отключает) можно внедрять непосредственно в формы, отчеты или свойства элементов управления, которые будут правильно работать с модулем кода VBA или макросом из более ранних версий Access.

Начиная работу с базами данных, следует помнить следующие правила.

– При открытии базы данных в надежном расположении все компоненты запускаются без проверки на доверие.

– При упаковке, подписывании и развертывании базы данных из более ранних версий Access (файлы с расширениями mdb или mde) все компоненты запускаются без необходимости решать вопрос о доверии в том случае, если она имеет действительную цифровую подпись надежного издателя и сертификат считается надежным.

– При подписывании и развертывании базы данных, не имеющей состояния доверия, в ненадежном расположении центр управления безопасностью по умолчанию отключает ее, и каждый раз при открытии нужно включать ее содержимое.

Office Access 2007 и защита на уровне пользователя

Office Access 2007 не предусматривает защиту на уровне пользователя для баз данных, созданных в новом формате (файлы с расширением accdb или accde). Однако при открытии базы данных из более ранней версии Access, имеющей защиту на уровне пользователя, в Office Access 2007 эти параметры будут продолжать работать.

При преобразовании подобной базы данных в новый формат приложение Access автоматически удаляет все параметры безопасности и применяет правила защиты файлов ACCDB и ACCDE.

И наконец, следует помнить, что каждый раз при открытии базы данных, созданной в Office Access 2007, все пользователи имеют возможность просмотра всех ее объектов.

21.1 Структура системы безопасности Office Access 2007

Для понимания структуры системы безопасности Office Access 2007 необходимо помнить, что база данных Access не является файлом, подобным книге Microsoft Office Excel 2007 или документу Microsoft Office Word 2007. В отличие от них база данных представляет собой набор объектов – таблиц, форм, запросов, макросов, отчетов и т. д. – которые часто являются взаимозависимыми. Например, при создании формы ввода данных нельзя вводить в нее или хранить в ней данные, если элементы управления в этой форме не связаны с таблицей.

Некоторые компоненты Access могут быть небезопасны, в том числе запросы на изменение (запросы, которые добавляют, удаляют или изменяют данные), макросы, выражения (функции, возвращающие одно значение) и код VBA. Чтобы защитить данные, Office Access 2007 и центр управления безопасностью выполняют ряд проверок на безопасность всякий раз при открытии базы данных. Процесс происходит следующим образом.

– При открытии в Office Access 2007 ACCDB– или ACCDE–файла приложение Access сообщает расположение базы данных центру управления безопасностью. Если это расположение надежное, она работает с полным набором функциональных возможностей. При открытии базы данных из более ранней версии Access в Office Access 2007 в центр управления безопасностью передаются расположение и цифровая подпись, если она имеется в базе данных.

Центр управления безопасностью проверяет подлинность этого «удостоверения», чтобы определить, имеет ли база данных состояние доверенной, а затем информирует приложение Access о том, как следует ее открывать. Приложение Access либо отключает ее, либо открывает с полным набором функциональных возможностей.

– Если центр управления безопасностью отключает какое–либо содержимое, то при открытии базы данных отображается панель сообщений (рисунок 31).

Чтобы включить отключенное содержимое, щелкните **Параметры**, а затем выберите параметры в появившемся диалоговом окне. Отключенное содержимое будет включено, и база данных откроется заново с полным набором функциональных возможностей. В противном случае отключенные компоненты не будут работать.

– При открытии базы данных, созданной в более раннем формате (файлы с расширением mdb или mde), у которой нет подписи и состояния доверенной, приложение Access по умолчанию отключает любое выполняемое содержимое.

Режим отключения

Когда центр управления безопасностью определяет, что база данных не имеет состояния доверенной, Office Access 2007 открывает ее в режиме отключения – то есть отключает любое выполняемое содержимое. Это справедливо как для баз данных, созданных в новом формате Office Access 2007, так и для файлов, созданных в предыдущих версиях Access.

Office Access 2007 отключает следующие компоненты.

– Код VBA и все ссылки в нем, а также все небезопасные выражения.

– Небезопасные макрокоманды во всех макросах. «Небезопасными» являются команды, позволяющие пользователю изменять базу данных или получать доступ к ресурсам вне базы данных. Однако макрокоманды, которые Access отключает, иногда могут считаться «безопасными». Например, при наличии доверия к создателю базы данных, можно доверять и всем небезопасным макрокомандам.

– Следующие типы запросов.

Запросы на изменение Добавляют, обновляют или удаляют данные.

Управляющие запросы (DDL–запросы) Используются для создания или изменения объектов базы данных, таких как таблицы и процедуры.

SQL–запросы к серверу Отправляют команды непосредственно на сервер базы данных, поддерживающий стандарт Open Database Connectivity (ODBC). Запросы к серверу работают с таблицами на сервере, минуя ядро базы данных Access.

– Элементы управления ActiveX.

При открытии базы данных может быть предпринята попытка загрузки надстроек – программ, расширяющих функциональные возможности Access либо открытой базы данных. Кроме того, пользователь может запустить мастер, создающий объекты в базе данных. При загрузке надстройки или запуске мастера Access отправляет сведения об этом в центр управления безопасностью, который принимает дополнительные решения по доверию и либо отключает, либо включает объект или действие. Если центр управления безопасностью отключил базу данных, но пользователь не согласен с таким решением, почти всегда можно воспользоваться панелью сообщений, чтобы включить содержимое. Исключением из этого правила являются надстройки. Если в центре управления безопасностью (в области **Надстройки**) установлен флажок **Требовать подпись доверенного издателя для расширений приложений**,

приложение Access предлагает включить надстройку, но этот процесс происходит без использования панели сообщения.

21.2 Использование базы данных Office Access 2007 в надежном расположении

Если база данных Office Access 2007 размещена в надежном расположении, при ее открытии работают все коды VBA, макросы и безопасные выражения. При этом не возникает необходимость решать вопросы доверия.

Процесс использования базы данных Office Access 2007 в надежном расположении включает три основных этапа.

1. Использование центра управления безопасностью для поиска или создания надежного расположения.
2. Сохранение, перемещение или копирование базы данных Office Access 2007 в надежное расположение.
3. Открытие и использование базы данных.

Описанная ниже последовательность шагов объясняет, как найти или создать надежное расположение, а затем добавить туда базу данных.

Запуск центра управления безопасностью

1. Нажмите кнопку **Microsoft Office**, а затем выберите команду **Параметры Access**.

Примечание Открывать базу данных не требуется.

Откроется диалоговое окно **Параметры Access**.

2. Выберите пункт **Центр управления безопасностью** и в группе **Центр управления безопасностью Microsoft Office Access** нажмите кнопку **Параметры центра управления безопасностью**.
3. Выберите **Надежные расположения**, а затем выполните одно из следующих действий.

- Укажите путь к одному или нескольким надежным расположениям.
- Создайте новое надежное расположение. Для этого нажмите кнопку **Добавить новое расположение**, а затем укажите значения параметров в диалоговом окне **Надежное расположение Microsoft Office**.

Размещение базы данных в надежном расположении

- Для перемещения или копирования файла базы данных в надежное расположение можно использовать любой способ. Например, воспользоваться проводником Windows или открыть файл в Access и сохранить его в надежном расположении.

Открытие базы данных в надежном расположении

- Для открытия файла можно использовать любой привычный способ. Например, выбрать и затем дважды щелкнуть файл в проводнике Windows либо, если уже запущен Access, нажать кнопку **Microsoft Office** для поиска и открытия файла.

21.3 Упаковка, подпись и распространение базы данных Office Access 2007

Office Access 2007 упрощает и ускоряет процесс добавления подписи и распространения базы данных. После создания ACCDB– или ACCDE–файла можно упаковать его, применить к пакету цифровую подпись, а затем распространить подписанный пакет среди других пользователей. Средство подписывания пакетов помещает базу данных в файл развертывания Access (с расширением accdc), подписывает пакет, а затем помещает пакет, подписанный кодом, в указанное расположение. Пользователи затем могут извлекать базу данных из пакета и работать непосредственно в ней, а не в файле пакета.

Учитывайте следующие сведения при работе.

– Упаковка базы данных и подпись пакета являются способами передачи доверия. Когда пользователь получает пакет, подпись подтверждает, что в базу данных не были внесены несанкционированные изменения. При доверии к автору можно включить содержимое.

– Новое средство подписывания пакетов применимо только к базам данных в формате Office Access 2007. В состав Office Access 2007 входят и прежние средства для подписывания и распространения баз данных, созданных в более раннем формате. Эти средства нельзя использовать для подписывания и распространения файлов, созданных в новом формате.

– В пакет можно включить только одну базу данных.

– В этом процессе подпись кодом добавляется ко всем объектам базы данных, а не только к макросам или программным модулям. Также происходит сжатие файла пакета с целью уменьшения времени на загрузку.

– Базы данных можно извлекать из файлов пакета, расположенных на серверах Windows SharePoint Services 3.0.

Далее описаны этапы создания подписанного файла пакета и использования базы данных в таком файле.

Создание подписанного пакета

1. Откройте базу данных, для которой требуется создать пакет и подписать его.

2. Нажмите кнопку **Microsoft Office**, выберите команду **Опубликовать**, а затем команду **Упаковать и подписать**.

Откроется диалоговое окно **Выбор сертификата**.

3. Выберите цифровой сертификат, а затем нажмите кнопку **ОК**.

Откроется диалоговое окно **Создать подписанный пакет Microsoft Office Access**.

4. В списке **Сохранить в** выберите расположение для подписанного пакета базы данных.

5. В поле **Имя файла** введите имя для подписанного пакета, а затем нажмите кнопку **Создать**.

Access создаст ACCDC–файл и поместит его в выбранное расположение.

Извлечение и использование подписанного пакета

1. Нажмите кнопку **Microsoft Office**, а затем выберите команду **Открыть**.

Откроется диалоговое окно **Открыть**.

2. В списке **Тип файлов** выберите вариант **Подписанные пакеты Microsoft Office Access (*.accdc)**.

3. Воспользуйтесь списком **Папка**, чтобы найти папку, содержащую ACCDC–файл, выделите этот файл и нажмите кнопку **Открыть**.

4. Выполните одно из следующих действий.

– Если выбран параметр доверия к цифровому сертификату, примененному к развернутому пакету, появится диалоговое окно **Извлечь базу данных в**. Перейдите к следующему этапу.

– Если параметр доверия к цифровому сертификату еще не выбран, появится предупреждение (рисунок 32).

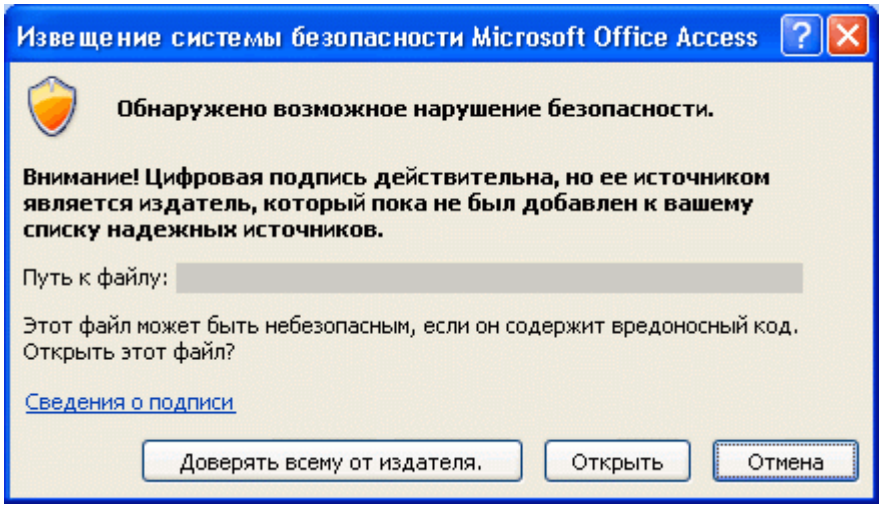


Рисунок 32 – Извещение системы безопасности

Если вы доверяете базе данных, нажмите кнопку **Открыть**. Если вы доверяете все сертификатам этого поставщика, нажмите кнопку **Доверять всему от издателя**. Откроется диалоговое окно **Извлечь базу данных в**.

5. В списке **Сохранить в** можно выбрать расположение для извлекаемой базы данных, а в поле **Имя файла** – ввести для нее другое имя.

6. Нажмите кнопку **ОК**.

21.4 Включение отключенного содержимого при открытии базы данных

По умолчанию Access отключает все выполняемое содержимое в базе данных, если она не имеет состояния доверенной или не размещена в надежном расположении. При открытии такой базы данных Access отключает это содержимое и отображает панель сообщений (рисунок 31).

В отличие от Access 2003 в Office Access 2007 при открытии базы данных не отображается набор модальных диалоговых окон (это диалоговые окна,

в которых необходимо принять какое–либо решение для того, чтобы продолжить работу). Однако при необходимости можно добавить ключ реестра, чтобы в Office Access 2007 отображались прежние модальные диалоговые окна.

Независимо от поведения Access при открытии базы данных можно предоставить разрешение выполняемому содержимому в файле в том случае, если эта база данных получена от надежного издателя.

Предоставление базе данных состояния доверия

1. На панели сообщений нажмите кнопку **Параметры**.

Откроется диалоговое окно **Параметры безопасности Microsoft Office**.

2. Выберите вариант **Включить это содержимое**, а затем нажмите кнопку **ОК**.

Если панель сообщений не отображается

– На вкладке **Работа с базами данных** в группе **Отображение** выберите параметр **Панель сообщений**.

При выполнении этих действий Access включает все отключенное содержимое, в том числе потенциально опасный код, до тех пор, пока база данных не будет закрыта. Если опасный код повредит данные или компьютер, приложение Access не сможет отменить его действия.

Закрытие базы данных

– Нажмите кнопку **Microsoft Office**, а затем выберите команду **Закрыть базу данных**.

При повторном открытии базы данных вновь отображается панель сообщений. В этом случае можно закрыть панель сообщений, оставив содержимое в отключенном состоянии либо скрыв панель. В любом случае результат будет один – отключенное содержимое останется отключенным.

Отключение содержимого

1. На панели сообщений нажмите кнопку **Параметры**.

Откроется диалоговое окно **Параметры безопасности Microsoft Office**.

2. Выберите вариант **Установить защиту от неизвестного содержимого (рекомендуется)** и нажмите кнопку **ОК**.

Access отключит все потенциально опасные компоненты.

Скрытие панели сообщений

– Не принимая решения о доверии, нажмите кнопку **Закрыть** (**X**) в верхнем углу панели сообщений.

Панель сообщений закроется.

Отображение панели сообщений

– На вкладке **Работа с базами данных** в группе **Отображение** выберите пункт **Панель сообщений**. Чтобы отобразить панель сообщений, можно также закрыть и снова открыть базу данных.

Добавление ключа реестра для отображения модальных диалоговых окон

Неверное изменение параметров реестра может привести к существенному повреждению операционной системы с необходимостью ее переустановки. Корпорация Майкрософт не гарантирует возможность разрешения проблем,

возникающих из-за изменения реестра. Перед изменением реестра выполните архивацию всех важных данных. Последние сведения о защите реестра компьютера см. в справке Microsoft Windows.

1. Нажмите кнопку **Пуск** и выберите команду **Выполнить**.

2. В поле **Открыть** введите **regedit**, а затем нажмите клавишу ВВОД.

Запустится редактор реестра.

3. Разверните папку HKEY_CURRENT_USER и укажите следующий раздел реестра:

Software\Microsoft\Office\12.0\Access\Security

4. В правой области редактора реестра щелкните правой кнопкой мыши пустое место, выберите команду **Создать**, а затем выберите вариант **Параметр DWORD**. Появится новый пустой параметр типа DWORD.

5. Введите следующее имя параметра: **ModalTrustDecisionOnly**.

6. Дважды щелкните новый параметр.

Откроется диалоговое окно **Изменение параметра DWORD**.

7. В поле **Значение** поменяйте значение **0** на **1**, а затем нажмите кнопку **ОК**.

8. Закройте редактор реестра.

Теперь при открытии базы данных, включающей небезопасное содержимое, вместо панели сообщений будет отображаться ряд диалоговых окон. Чтобы вернуться к исходному варианту, повторите эти действия и поменяйте значение **1** на **0**.

21.5 Использование пароля для шифрования базы данных Office Access 2007

Средство шифрования в Office Access 2007 представляет собой два объединенных и улучшенных средства прежних версий – кодирование и пароли баз данных. При использовании пароля для шифрования базы данных все данные становятся нечитаемыми в других программных средствах, и для того чтобы использовать эту базу данных, пользователи должны вводить пароль. При шифровании в Office Access 2007 используется более стойкий алгоритм, чем в предыдущих версиях Access.

Шифрование с использованием пароля базы данных

1. Откройте в монопольном режиме базу данных, которую требуется зашифровать.

Открытие базы данных в монопольном режиме

a. Нажмите кнопку **Microsoft Office**, а затем выберите команду **Открыть**.

b. В диалоговом окне **открыть** найдите файл, который нужно открыть, и выделите его.

c. Щелкните стрелку рядом с кнопкой **Открыть** и выберите команду **Монопольно** (рисунок 33).

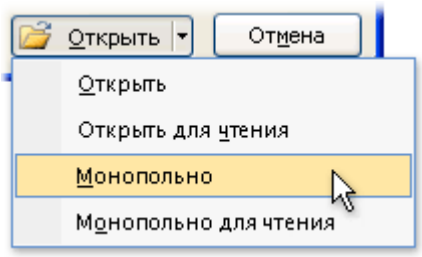


Рисунок 33. Способы открытия базы данных

2. На вкладке **Средства базы данных** в группе **Средства базы данных** щелкните **Зашифровать с помощью пароля**.

Откроется диалоговое окно **Задание пароля базы данных**.

3. Введите пароль в поле **Пароль**, а затем повторите его в поле **Проверить**.

Используйте надежные пароли, представляющие собой сочетание прописных и строчных букв, цифр и символов. Пароли, не содержащие набор таких элементов, являются ненадежными. Надежный пароль: Y6dh!et5. Ненадежный пароль: House27. Пароли должны состоять не менее чем из 8 знаков. Рекомендуется использовать фразу–пароль, состоящую из 14 или более знаков.

Важно помнить свой пароль. Если вы забыли пароль, корпорация Майкрософт не сможет его восстановить. Записывайте пароли и храните в безопасном месте отдельно от данных, для защиты которых они используются.

4. Нажмите кнопку **ОК**.

Расшифровка и открытие базы данных

1. Откройте зашифрованную базу данных точно так же, как обычно открывается любая другая.

Откроется диалоговое окно **Необходимо ввести пароль**.

2. Введите пароль в поле **Введите пароль базы данных** и нажмите кнопку **ОК**.

Удаление пароля

1. На вкладке **Средства базы данных** в группе **Средства базы данных** щелкните **Дешифровать базу данных**.

Откроется диалоговое окно **Удалить пароль базы данных**.

2. Введите пароль в поле **Пароль** и нажмите кнопку **ОК**.

21.6 О работе системы безопасности с базами данных из предыдущих версий Access, открытых в Office Access 2007

При открытии в Office Access 2007 базы данных, созданной в одной из предыдущих версий Access, все средства безопасности, примененные к ней, будут продолжать работать. Например, защита на уровне пользователя.

По умолчанию приложение Access открывает все старые базы данных, не имеющие состояния доверенных, в монопольном режиме и сохраняет их состояние. Можно включать отключенное содержимое каждый раз при открытии такой базы данных или применить цифровую подпись, воспользовавшись

сертификатом от надежного издателя, либо поместить базу данных в надежное расположение.

Для баз данных из более ранних версий, чем Office Access 2007, подпись кодом – это процесс применения цифровой подписи к компонентам базы данных. Цифровая подпись представляет собой зашифрованную электронную печать для заверения. Она подтверждает, что макросы, программные модули и другие выполняемые компоненты базы данных созданы лицом, добавившим подпись, и никто другой не изменял их после подписи.

Чтобы применить подпись к базе данных, прежде всего необходимо иметь цифровой сертификат. Если базы данных создаются для коммерческого распространения, нужно получить сертификат в коммерческом центре сертификации, например, VeriSign, Inc. или GTE. Центр сертификации наводит справки об изготовителе базы данных (издателе), чтобы удостовериться в его надежности.

Если базу данных планируется использовать в личных целях или в небольшой рабочей группе, можно воспользоваться предусмотренным в Microsoft Office профессиональный 2007 средством создания сертификатов с собственной подписью. В следующих разделах объясняется, как установить и использовать средство, называемое SelfCert.exe, для создания сертификата с собственной подписью. Этот сертификат следует добавить в список надежных источников, а затем подписать базу данных.

Создание сертификата с собственной подписью

1. Нажмите кнопку **Пуск**, выделите пункт **Все программы**, затем – пункты **Microsoft Office** и **Средства Microsoft Office** и выберите команду **Цифровое удостоверение для проектов VBA**.

–или–

Перейдите к папке, содержащей программные файлы Office профессиональный 2007. Папкой по умолчанию является папка Диск:\Program Files\Microsoft Office\Office12. В ней найдите и дважды щелкните файл **SelfCert.exe**.

Откроется диалоговое окно **Создание цифрового сертификата**.

2. В поле **Имя вашего сертификата** введите имя для нового сертификата.

3. Два раза нажмите кнопку **ОК**.

Если команда **Цифровое удостоверение для проектов VBA** не отображается либо не удается найти файл SelfCert.exe, может потребоваться установка средства SelfCert. Действия по установке описаны в конце данного раздела.

Подпись кодом базы данных

1. Откройте базу данных, к которой требуется добавить подпись.

2. На вкладке **Средства базы данных** в группе **Макрос** выберите команду **Visual Basic**, чтобы запустить редактор Visual Basic.

Клавиши быстрого доступа Нажмите клавиши ALT+F11.

3. В окне проекта выберите базу данных, макрос или модуль кода, к которым требуется добавить подпись.

4. В меню **Сервис** выберите команду **Цифровые подписи**.

Откроется диалоговое окно **Цифровые подписи**.

5. Нажмите кнопку **Выбор**, чтобы выбрать сертификат.

Откроется диалоговое окно **Выбор сертификата**.

6. Выберите необходимый сертификат.

Если выполнены действия, описанные в предыдущем разделе, то это сертификат, созданный при помощи средства SelfCert.

7. Чтобы закрыть диалоговое окно **Сертификат**, нажмите кнопку **ОК**, а затем нажмите кнопку **ОК** еще раз, чтобы закрыть диалоговое окно **Цифровая подпись**.

Следует помнить, что эти действия применимы только для баз данных, созданных в предыдущих версиях Access, при их использовании в Office Access 2007.

21.7 Выполнение небезопасных выражений (отключение изолированного режима)

При добавлении в базу данных выражения и при последующем предоставлении этой базе данных состояния доверенной либо при размещении ее в доверенном расположении приложение Access запускает это выражение в операционной среде, называемой изолированным режимом. Это действие выполняется как для баз данных в формате Office Access 2007, так и для более ранних форматов Access. Приложение Access по умолчанию применяет изолированный режим, который всегда отключает небезопасные выражения, даже после предоставления базе состояния доверенной.

Если база данных имеет состояние доверенной и необходимо выполнить выражение, отключенное изолированным режимом, нужно изменить параметр реестра и отключить этот режим. Следует помнить, что здесь необходимо прежде всего предоставить базе данных состояние доверенной.

На приведенном ниже рисунке 34 показан процесс принятия решений, который необходимо выполнить для запуска небезопасных выражений.

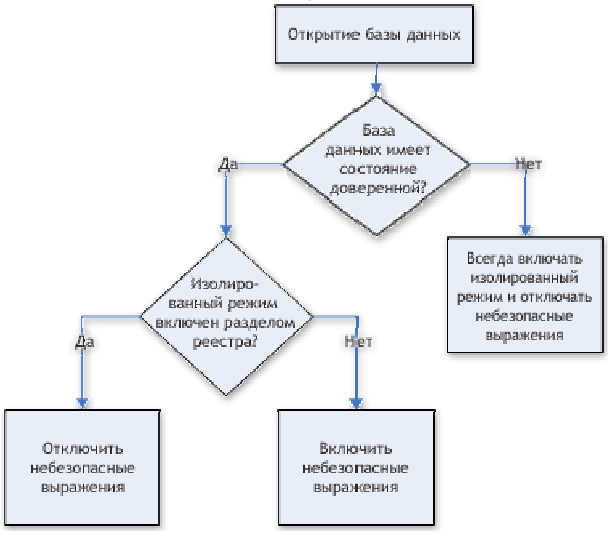


Рисунок 34 - Процесс принятия решений, который необходимо выполнить для запуска небезопасных выражений

Неверное изменение параметров реестра может привести к существенному повреждению операционной системы с необходимостью ее переустановки. Корпорация Майкрософт не гарантирует возможность разрешения проблем, возникающих из-за изменения реестра. Перед изменением реестра выполните архивацию всех важных данных. Последние сведения о защите реестра компьютера см. в справке Microsoft Windows.

При отсутствии достаточного опыта работы с реестром обратитесь за помощью к компетентному человеку или рассмотрите вариант преобразования базы данных из более ранней версии Access в формат Office Access 2007. Для изменения значений реестра необходимо также иметь разрешения администратора на данном компьютере.

21.8 Изменение параметра реестра

Важно Выполнение этих шагов позволит запускать небезопасные выражения во всех экземплярах Access всех пользователей данного компьютера.

1. Нажмите кнопку **Пуск** и выберите команду **Выполнить**.
2. В поле **Открыть** введите **regedit**, а затем нажмите клавишу ВВОД.

Запустится редактор реестра.

3. Разверните папку HKEY_LOCAL_MACHINE и укажите следующий раздел реестра:

\Software\Microsoft\Office\12.0\Access Connectivity Engine\Engines

4. В правой области редактора реестра дважды щелкните параметр **SandboxMode**.

Откроется диалоговое окно **Изменение параметра DWORD**.

5. В поле **Значение** поменяйте значение **3** на **2** и нажмите кнопку **ОК**.
6. Закройте редактор реестра.

Следует помнить, что, если база данных не имеет состояния доверенной, Access отключает любые небезопасные выражения независимо от того, изменен ли данный параметр реестра.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Базы данных: проектирование, реализация сопровождение: теория и практика: пер. с англ. / Т. Конноли, К. Бегг. – 2–е изд., испр. и доп. – М.: Вильямс, 2001. – 1111с.
2. Глушаков С.В., Ломотько Д.В. Базы данных: Учебный курс. – Харьков: ФОЛИО, М.: ООО «Издательство АСТ»; 2000. – 504 с.
3. Кузин А.В., Дёмин В.М. Разработка баз данных в системе MICROSOFT ACCESS : учеб. для вузов.– М.: ФОРУМ–ИНФРА–М., 2005. – 224с.: ил.
4. Основные концепции баз данных : пер. с англ. / Ф. Д. Ролланд.– М.: Вильямс, 2002.– 254 с.
5. Симонович С. В., Евсеев Г. А., Алексеев А. Г.. Специальная информатика: учеб. пособие / М.: АСТ–ПРЕСС: Инфорком–Пресс, 1999.– 480 с.
6. Тулегулов А. Информационные системы в производстве: учебник /.– Астана: Фолиант, 2009. – 320 с.
7. Хакимова Т. Специальные программы для работы на персональном компьютере учебное пособие.– Алматы: NURPRESS, 2007.
8. Хакимова Т. Теория и методика компьютерного моделирования для задач базы данных и глобальной сети: учеб. пособие / – Алматы: NURPRESS, 2007.
9. Харитонова И.А., Михеева В.Д. Microsoft Access 2000. – СПб.: БХВ – Санкт-Петербург, 2000. – 654с.
10. Шафрин Ю. А. Информационные технологии. В 2-х ч . Ч. 2. Офисная технология и информационные системы. – М.: БИНОМ. Лаборатория знаний, 2004. – 336 с.
11. Яворский В.В., Яворская Г.М. – Введение в информационные технологии: учеб. пособие / Астана: Фолиант, 2007. – 256 с. – (Проф.образование).
12. Гайдамакин Н.А. Автоматизированные информационные системы, базы и банки данных: Вводный курс. – «Гелиос» 2002. – 368 с.
13. Гарсиа–Молина Г., Ульман Дж., Уидом Дж. Системы баз данных. Полный курс = Database Systems: The Complete Book. – Вильямс, 2003. – 1088 с.
14. К. Дейт. Введение в системы баз данных, 6-е издание – Пер. с англ. К.; М.; СПб.: Издательский дом "Вильямс", 2000.
15. К.Дж.Дейт, Хью Дарвен Основы будущих систем баз данных. Третий манифест. Перевод: С.Д.Кузнецов, Т.А.Кузнецова/Под ред. С.Д.Кузнецова – Издательство Янус–К, 2004 г. – 656.
16. Когаловский М.Р. Энциклопедия технологий баз данных – М.: Финансы и статистика, 2002 – 800с.
17. Коннолли Т., Бегг К. Базы данных. Проектирование, реализация и сопровождение. Теория и практика. – 3-е изд. – М.: Вильямс, 2003. – 1436 с.
18. Крёнке Д. Теория и практика построения баз данных, 8-е изд. – Питер, 2003 – 800с.

19. Кузнецов С. Д. Основы баз данных. – 2-е изд. – М.: Интернет-университет информационных технологий; БИНОМ. Лаборатория знаний, 2007. – 484 с.
20. Мюллер Роберт Дж. Базы данных и UML – “Лори”, 2002 , – 420 с.
21. Пасько В. Access 97. – К.: Издательская группа BHV, 1997. – 368с.
22. Справочная система Microsort Office.