Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего профессионального образования
«Казанский (Приволжский) федеральный университет»

ИНСТИТУТ МАТЕМАТИКИ И МЕХАНИКИ ИМ. Н.И.ЛОБАЧЕВСКОГО
КАФЕДРА ТЕОРИИ И ТЕХНОЛОГИИ ПРЕПОДАВАНИЯ МАТЕМАТИКИ И ИНФОРМАТИКИ

Специальность: 050202.65 информатика с доп. специальностью

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Создание дистанционного курса по дисциплине «Информационные системы»

Работа завершена:
«___» _____2015г. _____(К. О. Екатеринин)
Работа допущена к защите:
Научный руководитель
к.т.н., доцент
«___» _____2015г. _____(Т. Ю. Гайнутдинова)
Заведующий кафедрой:
д.п.н., профессор
«__» _____2015г. _____(Л.Р.Шакирова)

Казань 2015

Оглавление

ВВЕДЕНИЕ

Потоки информации, циркулирующие в мире, который нас окружает, огромны. Во времени они имеют тенденцию к увеличению. Поэтому в любой организации, как большой, так и маленькой, возникает проблема такой организации управления данными, которая обеспечила бы наиболее эффективную работу. Некоторые организации используют для этого шкафы с папками, но большинство предпочитают компьютеризированные способы - базы данных, позволяющие эффективно хранить, структурировать и систематизировать большие объемы данных. И уже сегодня без баз данных невозможно представить работу большинства финансовых, промышленных, торговых и прочих организаций.

Существует много веских причин перевода существующей информации на компьютерную основу. Базы данных позволяют хранить, структурировать информацию и извлекать ее оптимальным для пользователя образом. Использование клиент/серверных технологий позволяют сберечь значительные средства, а главное и время для получения необходимой информации, а также упрощают доступ и введение, поскольку они основываются на комплексной обработке данных и централизации их хранения. ЭВМ позволяет хранить любые форматы данных, текст, чертежи, данные в рукописной форме, фотографии, записи голоса.

Актуальность. Для использования огромных объемов хранимой информации, помимо развития системных устройств, средств передачи данных, памяти, необходимы средства обеспечения диалога человек - ЭВМ, которые позволяют пользователю вводить запросы, читать файлы, модифицировать хранимые данные, добавлять новые данные или принимать решения на основании хранимых данных. Для обеспечения этих функций созданы специализированные средства - системы управления базами данных (СУБД). Современные СУБД - многопользовательские системы управления базой данных, которые специализируется на управлении массивом

информации одним или множеством одновременно работающих пользователей.

Современные СУБД обеспечивают: набор средств для поддержки таблиц и отношений между связанными таблицами; развитый пользовательский интерфейс, который позволяет вводить и модифицировать информацию, выполнять поиск и представлять информацию в графическом или текстовом режиме; средства программирования высокого уровня, с помощью которых можно создавать собственные приложения.

Целью выпускной квалификационной работы является – создание дистанционного курса по изучению дисциплины «Информационные системы».

Для достижения цели исследования решались следующие задачи:

- изучить основные тенденции развития современной работы с базами данных: цели, содержание, формы, методы и средств обучения;

- провести анализ научной и учебно-методической литературы по выбранной теме;

- изучение и анализ современного программного обеспечения и тенденции его развития;

- разработать дистанционный курс по дисциплине «Информационные системы» для студентов.

Практическая значимость работы заключается в определении содержания теоретического материала, направленного на формирование специальных знаний и умений по изучению дисциплины «Информационные системы» и его практического применения.

Работа состоит из введения, двух глав, заключения и списка использованной литературы. В первой главе представлен теоретический материал по изучению баз данных: MS Access 2013, SQL. Во второй главе разработан практикум по созданию и обработке баз данных.

ГЛАВА I

ИНФОРМАЦИОННЫЕ СИСТЕМЫ

1.1.Информационная система, классификации

Информация в современном мире превратилась в один из наиболее важных ресурсов, а информационные системы (далее - ИС) стали необходимым инструментом практически во всех сферах деятельности.

В самом широком смысле ***информационная система*** представляет собой программный комплекс, функции которого состоят в поддержке надежного хранения информации в памяти компьютера, выполнении специфических для данного приложения преобразований информации и/или вычислений, предоставлении пользователям удобного и легко осваиваемого интерфейса. Данные, с которыми работают такие системы, обычно довольно большого размера, а также имеют сложную структуру. Классическими примерами *информационных систем* являются банковские системы, системы резервирования авиационных или железнодорожных билетов, мест в гостиницах.

Различие задач, решаемых с помощью ИС, привело к появлению большого количества систем разных типов, с различным построением и различными методами обработки информации.

Информационные системы **классифицируют** по различным признакам. В основе данной классификации рассмотрены наиболее существенные признаки, определяющие функции и особенности строения рассматриваемых систем. Информационные системы делят на классы в зависимости от того на сколько велик объем данных задач, технические средства, используемые для решения этих задач, организации функционирования.

По типу хранимых данных ИС делятся на фактографические и документальные. Фактографические системы используются для хранения и обработки данных в виде текстов и чисел. Над этими данными выполняется множество различные операции. Информация в документальных системах представляется в виде документов, которая состоит из наименования, описания, реферата и текста. В таких системах обработка данных в принципе не производится. Пользователю предоставляются отобранные документы, найденные с помощью семантических признаков.

Информационные системы можно поделить на ручные, автоматические и автоматизированные, основываясь на степени автоматизации информационных процессов в системе управления фирмой.

Ручным информационным системам характерно отсутствие современных технических средств обработке информации и выполнение всех операций человеком.

Автоматические информационные системы выполняют обработку данных без участия человека.

В автоматизированные информационных системах обработка информации производится как человеком, так и техническими средствами, главная роль по выполнению рутинных операций обработки данных отводится компьютеру, современному понятию "информационная система" соответствует именно этот класс систем.

Информационные системы делят на информационно-поисковые и информационно-решающие, в зависимости от характера обработки данных.

Информационно-поисковые производят хранение, систематизацию, ввод и выдачу информации по запросу пользователя без сложных преобразований данных. (Примером такой системы является библиотечное обслуживание, резервирование и продажа билетов на транспорте, бронирование мест в гостинице.)

В Информационно-решающих системах осуществляется, кроме того, операции переработки информации по определенному алгоритму. По характеру использования выходной информации такие системы принято делить на управляющие и советующие.

В управляющих информационных системах результативная информация трансформируется в решение принимаемые человеком. Для этих систем характерны задачи расчетного характера и обработка больших объемов данных. (Например, ИС планирования производства или заказов, бухгалтерского учета.)

В советующие информационных системах вырабатывает информация, которая принимается человеком к сведению и учитывается при формировании управленческих решений, а не инициирует конкретные действия. Эти системы имитируют интеллектуальные процессы обработки знаний, а не данных.

Различают следующие классы ИС, в зависимости от сферы применения.

ИС организационного управления - предназначены для автоматизации функций управленческого персонала как промышленных предприятий, так и непромышленных объектов.

Основными функциями подобных систем являются: оперативный контроль и регулирование, оперативный учет и анализ, перспективное и оперативное планирование, бухгалтерский учет, управление сбытом, снабжением и другие экономические и организационные задачи.

ИС управления технологическими процессами – служат для автоматизации функций производственного персонала по контролю и управлению производственными операциями. В таких системах обычно предусматривается наличие развитых средств измерения параметров технологических процессов, процедур контроля допустимости значений параметров и регулирования технологических процессов.

Информационные системы автоматизированного проектирования – предназначены для автоматизации функций инженеров-проектировщиков, конструкторов, архитекторов, дизайнеров при создании новой техники или технологии. Основными функциями подобных систем являются: инженерные расчеты, создание графической документации (чертежей, схем, планов), создание проектной документации, моделирование проектируемых объектов.

Интегрированные информационные системы - используются для автоматизации всех функций фирмы и охватывают весь цикл работ от планирования деятельности до сбыта продукции. Они включают в себя ряд модулей (подсистем), работающих в едином информационном пространстве и выполняющих функции поддержки соответствующих направлений деятельности. Типовые задачи, решаемые модулями корпоративной системы, приведены в таблице 1.

Таблица 1

Функциональное назначение модулей корпоративной ИС.				
Подсистема маркетинга	**Производственные подсистемы**	**Финансовые и учетные подсистемы**	**Подсистема кадров (человеческих ресурсов)**	**Прочие подсистемы (например, ИС руководства)**
Исследование рынка и прогнозирование продаж	Планирование объемов работ и разработка календарных планов	Управление портфелем заказов	Анализ и прогнозирование потребности в трудовых ресурсах	Контроль за деятельностью фирмы
Управление продажами	Оперативный контроль и управление производством	Управление кредитной политикой	Ведение архивов записей о персонале	Выявление оперативных проблем
Рекомендации по производству новой продукции	Анализ работы оборудования	Разработка финансового плана	Анализ и планирование подготовки кадров	Анализ управленческих и стратегических ситуаций
Анализ и установление цены	Участие в формировании заказов поставщикам	Финансовый анализ и прогнозирование		Обеспечение процесса выработки стратегических решений
Учет заказов	Управление запасами	Контроль бюджета, бухгалтерский учет и расчет зарплаты		

Существует *классификация ИС* в зависимости от уровня управления, на котором система используется.

ИС оперативного уровня - поддерживая исполнителя, обрабатывает данные о сделках и событиях (счета, накладные, зарплата, кредиты, поток сырья и материалов). ИС оперативного уровня является связующим звеном между внешней средой и фирмой.

На оперативном уровне источники информации, алгоритмы обработки, задачи и цели заранее определены и в высокой степени структурированы.

ИС специалистов – повышают производительность и продуктивность работы инженеров и проектировщиков, поддерживая работу с знаниями и данными. Задачей таких ИС является интеграция новых сведений в организацию и помощь в обработке бумажных документов.

ИС уровня менеджмента - используются работниками среднего управленческого звена для контроля, принятия решений, мониторинга, и администрирования. Основные функции этих ИС:

- сравнение текущих показателей с прошлыми;
- составление периодических отчетов за определенное время, а не выдача отчетов по текущим событиям, как на оперативном уровне;
- обеспечение доступа к архивной информации.

Стратегическая ИС - компьютерная информационная система, обеспечивающая поддержку принятия решений по реализации стратегических перспективных целей развития организации.

ИС стратегического уровня помогают высшему звену управленцев решать неструктурированные задачи, осуществлять долгосрочное планирование. Основной задачей информационной системы стратегического уровня является сравнение изменений, происходящих во внешнем окружении с потенциалом фирмы. В неожиданно возникающих ситуациях они призваны создать общую среду компьютерной телекоммуникационной поддержки решений. Данные системы используют самые современные и совершенные программы, они способны предоставить информацию из многих источников в любой момент.

Выделяют ряд типовых архитектур информационных систем, с точки зрения программно-аппаратной реализации.

Традиционные архитектурные решения основаны на использовании выделенных серверов баз данных или файл-серверов. Существуют также варианты архитектур корпоративных информационных систем, базирующихся на технологии Internet. Следующая разновидность

архитектуры ИС основывается на концепции "хранилища данных" – встроенная информационная среда, которая включает в себя разнородные информационные ресурсы. Для построения глобальных распределенных информационных приложений используют архитектуру интеграции информационно-вычислительных компонентов на основе объектно-ориентированного подхода.

1.2. Этапы проектирования и создания информационной системы

Основным подходом в *проектировании ИС* на первом этапе был метод "снизу-вверх", когда система создавалась как набор приложений для поддержки деятельности предприятия в данный момент. Основной целью этих проектов было обслуживание текущих потребностей конкретного учреждения, а не создание тиражируемых продуктов. В рамках "лоскутной автоматизации" достаточно хорошо обеспечивается поддержка отдельных функций, но практически полностью отсутствует стратегия развития комплексной системы автоматизации, а объединение функциональных подсистем превращается в самостоятельную и достаточно сложную проблему.

Предприятия пытались обустроиться своими силами, создавая свои отделы и управления автоматизации. Однако периодические изменения технологий работы и должностных инструкций, сложности, связанные с разными представлениями пользователей об одних и тех же данных, приводили к непрерывным доработкам программных продуктов для удовлетворения все новых и новых пожеланий отдельных работников. Как следствие - и работа программистов, и создаваемые ИС вызывали недовольство руководителей и пользователей системы.

Второй этап связан с осознанием того факта, что существует потребность в достаточно стандартных программных средствах

автоматизации деятельности различных учреждений и предприятий. Из всего спектра проблем разработчики выделили наиболее заметные: автоматизацию ведения бухгалтерского аналитического учета и технологических процессов. Системы начали проектироваться "сверху-вниз", т.е. в предположении, что одна программа должна удовлетворять потребности многих пользователей.

Сама идея использования универсальной программы накладывает существенные ограничения на возможности разработчиков по формированию структуры базы данных, экранных форм, по выбору алгоритмов расчета. Заложенные "сверху" жесткие рамки не дают возможности гибко адаптировать систему к специфике деятельности конкретного предприятия: учесть необходимую глубину аналитического и производственно-технологического учета, включить необходимые процедуры обработки данных, обеспечить интерфейс каждого рабочего места с учетом функций и технологии работы конкретного пользователя.

Решение этих задач требует серьезных доработок системы. Таким образом, материальные и временные затраты на внедрение системы и ее доводку под требования заказчика обычно значительно превышают запланированные показатели.

Согласно статистическим данным, собранным Standish Group (США), из 8380 проектов, обследованных в США в 1994 году, неудачными оказались более 30% проектов, общая стоимость которых превышала 80 миллиардов долларов. При этом оказались выполненными в срок лишь 16% от общего числа проектов, а перерасход средств составил 189% от запланированного бюджета.

В то же время, заказчики информационных систем стали выдвигать все больше требований, направленных на обеспечение возможности комплексного использования корпоративных данных в управлении и планировании своей деятельности.

Таким образом, возникла насущная необходимость формирования новой методологии построения информационных систем.

Цель такой методологии заключается в регламентации процесса проектирования информационных систем и обеспечении управления этим процессом с тем, чтобы гарантировать выполнение требований как к самой информационной системе, так и к характеристикам процесса разработки. Основными задачами, решению которых должна способствовать методология проектирования корпоративных информационных систем, являются следующие:

- обеспечивать создание корпоративных информационных систем, отвечающих целям и задачам организации, а также предъявляемым требованиям по автоматизации деловых процессов заказчика;
- гарантировать создание системы с заданным качеством в заданные сроки и в рамках установленного бюджета проекта;
- поддерживать удобную дисциплину сопровождения, модификации и наращивания системы;
- обеспечивать преемственность разработки, т.е. использование в разрабатываемой информационной системе существующей информационной инфраструктуры организации.

Внедрение методологии должно приводить к снижению сложности процесса создания информационной системы за счет полного и точного описания этого процесса, а также применения современных методов и технологий создания информационных систем на всем жизненном цикле ИС - от замысла до реализации.

Проектирование информационной системы охватывает три основные области:

- проектирование объектов данных, которые будут реализованы в базе данных;

- проектирование программ, экранных форм, отчетов, которые будут обеспечивать выполнение запросов к данным;
- учет конкретной среды или технологии, а именно: топологии сети, конфигурации аппаратных средств, используемой архитектуры (файл-сервер или клиент-сервер), параллельной обработки, распределенной обработки данных и т.п.

Проектирование информационных систем всегда начинается с определения цели проекта. В общем виде цель проекта можно определить, как решение ряда взаимосвязанных задач, включающих в себя обеспечение на момент запуска системы и в течение всего времени ее эксплуатации:

- требуемой функциональности системы и уровня ее адаптивности к изменяющимся условиям функционирования;
- требуемой пропускной способности системы;
- требуемого времени реакции системы на запрос;
- безотказной работы системы;
- необходимого уровня безопасности;
- простоты эксплуатации и поддержки системы.

Согласно современной методологии, процесс создания информационных систем представляет собой процесс построения и последовательного преобразования ряда согласованных моделей на всех этапах жизненного цикла ИС. На каждом этапе жизненного цикла создаются специфичные для него модели - организации, требований к ИС, проекта ИС, требований к приложениям. Модели формируются рабочими группами команды проекта, сохраняются и накапливаются в репозитории проекта. Создание моделей, их контроль, преобразование и предоставление в коллективное пользование осуществляется с использованием специальных программных инструментов - CASE-средств.

Процесс создания ИС делится на ряд этапов (стадий), ограниченных некоторыми временными рамками и заканчивающихся выпуском конкретного продукта (моделей, программных продуктов, документации и пр.).

Обычно выделяют следующие *этапы создания ИС*: формирование требований к системе, проектирование, реализация, тестирование, ввод в действие, эксплуатация и сопровождение. (Последние два этапа далее не рассматриваются, поскольку выходят за рамки тематики книги.)

Начальным этапом процесса создания ИС является моделирование бизнес-процессов, протекающих в организации и реализующих ее цели и задачи. Модель организации, описанная в терминах бизнес-процессов и бизнес-функций, позволяет сформулировать основные требования к ИС. Это фундаментальное положение методологии обеспечивает объективность в выработке требований к проектированию системы. Множество моделей описания требований к ИС затем преобразуется в систему моделей, описывающих концептуальный проект ИС. Формируются модели архитектуры ИС, требований к программному обеспечению (ПО) и информационному обеспечению (ИО). Затем формируется архитектура ПО и ИО, выделяются корпоративные БД и отдельные приложения, формируются модели требований к приложениям и проводится их разработка, тестирование и интеграция.

Целью начальных *этапов создания ИС*, выполняемых на стадии анализа деятельности организации, является формирование требований к ИС, корректно и точно отражающих цели и задачи организации-заказчика. Чтобы специфицировать процесс создания ИС, отвечающей потребностям организации, нужно выяснить и четко сформулировать, в чем заключаются эти потребности. Для этого необходимо определить требования заказчиков к

ИС и отобразить их на языке моделей в требования к разработке проекта ИС так, чтобы обеспечить соответствие целям и задачам организации.

Задача формирования требований к ИС является одной из наиболее ответственных, трудно формализуемых и наиболее дорогих и тяжелых для исправления в случае ошибки. Современные инструментальные средства и программные продукты позволяют достаточно быстро создавать ИС по готовым требованиям. Но зачастую эти системы не удовлетворяют заказчиков, требуют многочисленных доработок, что приводит к резкому удорожанию фактической стоимости ИС. Основной причиной такого положения является неправильное, неточное или неполное определение требований к ИС на этапе анализа.

На этапе проектирования прежде всего формируются модели данных. Проектировщики в качестве исходной информации получают результаты анализа. Построение логической и физической моделей данных является основной частью проектирования базы данных. Полученная в процессе анализа информационная модель сначала преобразуется в логическую, а затем в физическую модель данных.

Параллельно с проектированием схемы базы данных выполняется проектирование процессов, чтобы получить спецификации (описания) всех модулей ИС. Оба эти процесса проектирования тесно связаны, поскольку часть бизнес-логики обычно реализуется в базе данных (ограничения, триггеры, хранимые процедуры). Главная цель проектирования процессов заключается в отображении функций, полученных на этапе анализа, в модули информационной системы. При проектировании модулей определяют интерфейсы программ: разметку меню, вид окон, горячие клавиши и связанные с ними вызовы.

Конечными продуктами этапа проектирования являются:

- схема базы данных (на основании ER-модели, разработанной на этапе анализа);
- набор спецификаций модулей системы (они строятся на базе моделей функций).

Кроме того, на этапе проектирования осуществляется также разработка архитектуры ИС, включающая в себя выбор платформы (платформ) и операционной системы (операционных систем). В неоднородной ИС могут работать несколько компьютеров на разных аппаратных платформах и под управлением различных операционных систем. Кроме выбора платформы, на этапе проектирования определяются следующие характеристики архитектуры:

- будет ли это архитектура "файл-сервер" или "клиент-сервер";
- будет ли это 3-уровневая архитектура со следующими слоями: сервер, ПО промежуточного слоя (сервер приложений), клиентское ПО;
- будет ли база данных централизованной или распределенной. Если база данных будет распределенной, то какие механизмы поддержки согласованности и актуальности данных будут использоваться;
- будет ли база данных однородной, то есть, будут ли все серверы баз данных продуктами одного и того же производителя (например, все серверы только Oracle или все серверы только DB2 UDB). Если база данных не будет однородной, то какое ПО будет использовано для обмена данными между СУБД разных производителей (уже существующее или разработанное специально как часть проекта);.
- будут ли для достижения должной производительности использоваться параллельные серверы баз данных (например, Oracle Parallel Server, DB2 UDB и т.п.).

Этап проектирования завершается разработкой технического проекта ИС.

На этапе реализации осуществляется создание программного обеспечения системы, установка технических средств, разработка эксплуатационной документации.

Этап тестирования обычно оказывается распределенным во времени.

После завершения разработки отдельного модуля системы выполняют автономный тест, который преследует две основные цели:

- обнаружение отказов модуля (жестких сбоев);
- соответствие модуля спецификации (наличие всех необходимых функций, отсутствие лишних функций).

После того как автономный тест успешно пройдет, модуль включается в состав разработанной части системы, и группа сгенерированных модулей проходит тесты связей, которые должны отследить их взаимное влияние.

Далее группа модулей тестируется на надежность работы, то есть проходят, во-первых, тесты имитации отказов системы, а во-вторых, тесты наработки на отказ. Первая группа тестов показывает, насколько хорошо система восстанавливается после сбоев программного обеспечения, отказов аппаратного обеспечения. Вторая группа тестов определяет степень устойчивости системы при штатной работе и позволяет оценить время безотказной работы системы. В комплект тестов устойчивости должны входить тесты, имитирующие пиковую нагрузку на систему.

Затем весь комплект модулей проходит системный тест - тест внутренней приемки продукта, показывающий уровень его качества. Сюда входят тесты функциональности и тесты надежности системы.

Последний тест информационной системы - приемо-сдаточные испытания. Такой тест предусматривает показ информационной системы заказчику и должен содержать группу тестов, моделирующих реальные бизнес-процессы, чтобы показать соответствие реализации требованиям заказчика.

Необходимость контролировать процесс создания ИС, гарантировать достижение целей разработки и соблюдение различных ограничений

(бюджетных, временных и пр.) привело к широкому использованию в этой сфере методов и средств программной инженерии: структурного анализа, объектно-ориентированного моделирования, CASE-систем.

1.3. Модели данных в информационных системах

Одной из наиболее важных сфер применения первых СУБД было планирование производства для компаний, занимающихся выпуском продукции. Например, если автомобильная компания хотела выпустить 10 000 машин одной модели и 5000 - другой, ей необходимо было знать, сколько деталей следует заказать у своих поставщиков. Чтобы ответить на этот вопрос, необходимо определить, из каких деталей состоят эти части и т.д. Например, машина состоит из двигателя, корпуса и ходовой части; двигатель в свою очередь состоит из клапанов, цилиндров, свеч и т.д. Работа со списками составных частей была как будто специально предназначена для компьютеров. Список составных частей изделия по своей природе является иерархической структурой. Для хранения данных, имеющих такую структуру, была разработана *иерархическая* модель данных, которую иллюстрирует рис. 1.

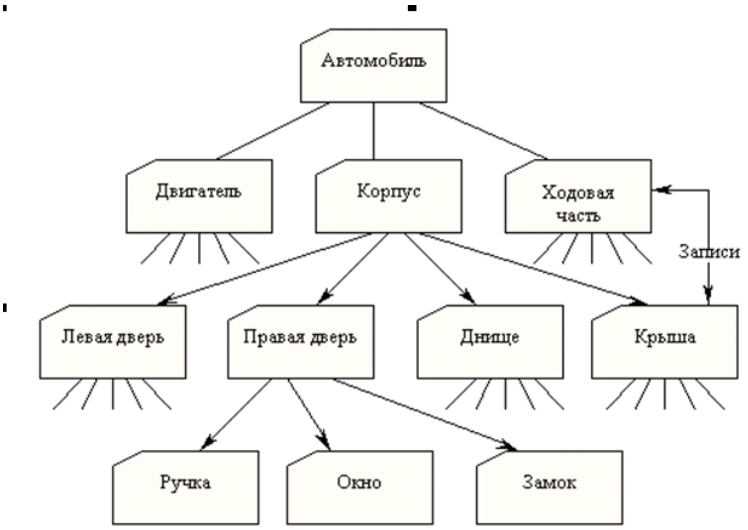


Рис. 1. Иерархическая база данных, содержащая информацию о составных частях

В этой модели каждая *запись* базы данных представляла конкретную деталь. Между записями существовали *отношения предок/потомок*, связывающие каждую часть с деталями, входящими в неё.

Чтобы получить доступ к данным, содержащимся в базе данных, программа могла:

· найти конкретную деталь (правую дверь) по её номеру;

· перейти "вниз" к первому потомку (ручка двери);

· перейти "вверх" к предку (корпус);

· перейти "в сторону" к другому потомку (правая дверь).

Таким образом, для чтения данных из иерархической базы данных требовалось перемещаться по записям, за один раз переходя на одну запись вверх, вниз или в сторону.

Одной из наиболее популярных иерархических СУБД была Information Management System (IMS) компании IBM, появившаяся в 1968 году. Ниже перечислены преимущества IMS и реализованной в ней иерархической модели.

· *Простота модели*. Принцип построения IMS был легок для понимания. Иерархия базы данных напоминала структуру компании или генеалогическое дерево.

· *Использование отношений предок/потомок*. СУБД IMS позволяла легко представлять отношения предок/потомок, например: "А является частью В" или "А владеет В".

· *Быстродействие*. В СУБД IMS отношения предок/потомок были реализованы в виде физических указателей из одной записи на другую, вследствие чего перемещение по базе данных происходило быстро. Поскольку структура данных в этой СУБД отличалась простотой, IMS могла размещать записи предков и потомков на диске рядом друг с другом, что позволяло свести к минимуму количество операций записи-чтения.

Если структура данных оказывалась сложнее, чем обычная иерархия, простота структуры иерархической базы данных становилась её недостатком.

Например, в базе данных для хранения заказов один заказ мог участвовать в трёх *различных* отношениях предок/потомок, связывающих заказ с клиентом, разместившим его, со служащим, принявшим его, и с заказанным товаром.

Такие структуры данных не соответствовали строгой иерархии IMS. В связи с этим для таких приложений, как обработка заказов, была разработана новая *сетевая* модель данных. Она являлась улучшенной иерархической моделью, в которой одна запись могла участвовать в нескольких отношениях предок/потомок, как показано на рис. 2.

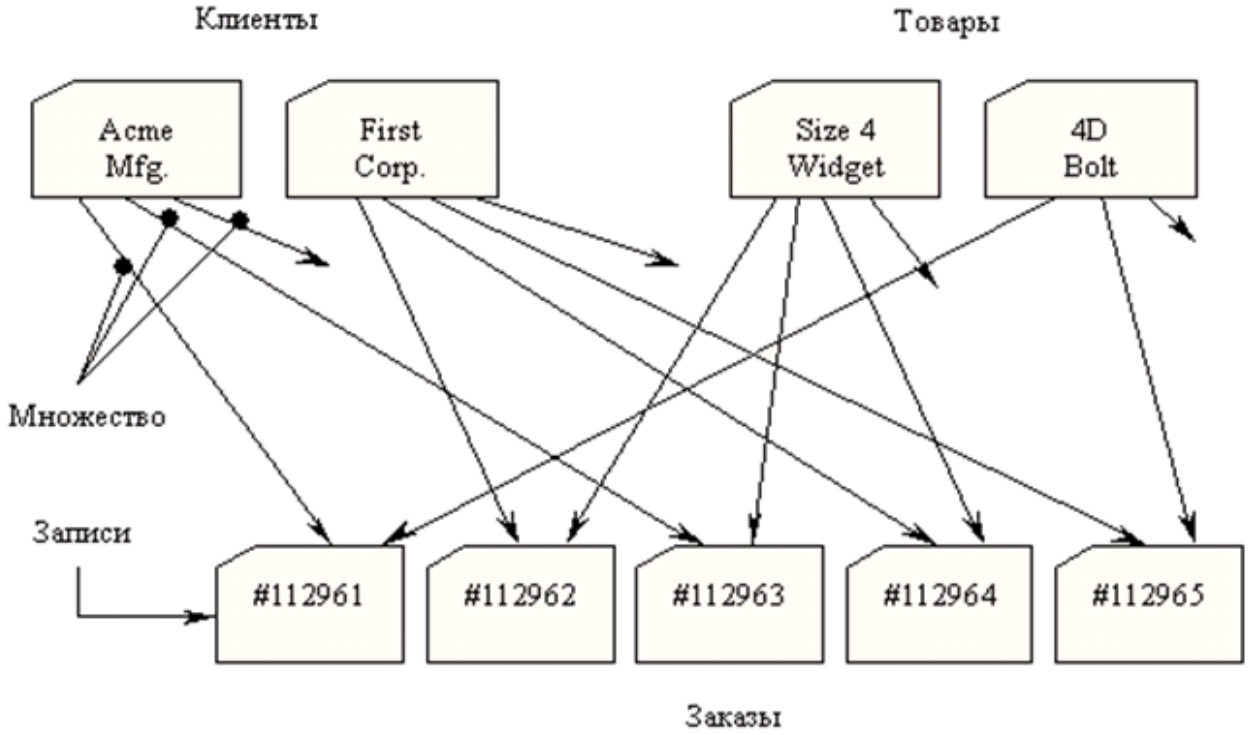


Рис. 2. Сетевая база данных, содержащая информацию о заказах

В сетевой модели такие отношения назывались *множествами*. Сетевые базы данных обладали рядом преимуществ:

· *Гибкость.* Множественные отношения предок/потомок позволяли сетевой базе данных хранить данные, структура которых была сложнее простой иерархии.

· *Быстродействие.* Вопреки своей большой сложности, сетевые базы данных достигали быстродействия, сравнимого с быстродействием иерархических баз данных. Множества были представлены указателями на

физические записи данных, и в некоторых системах администратор мог задать кластеризацию данных на основе множества отношений.

У сетевых баз данных есть недостатки. Как и иерархические, сетевые базы данных были очень жесткими. Наборы отношений и структуру записей приходилось задавать наперёд. Изменение структуры базы данных обычно означало перестройку всей базы данных.

Как иерархическая, так и сетевая база данных были инструментами программистов. Чтобы получить ответ на вопрос типа "Какой товар наиболее часто заказывает компания Acme Manufacturing?", программисту приходилось писать программу для навигации по базе данных. Реализация пользовательских запросов часто затягивалась на недели и месяцы, и к моменту появления программы информация, которую она предоставляла, часто оказывалась бесполезной.

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

Таблица PRODUCTS

DESCRIPTION	PRICE	QTY_ON_HAND
Ratcher Link	$79,00	210
Widget Remover	$2.750,00	25
Reducer	$355,00	38

Таблица ORDERS

ORDER_NUM	ORDER_DATE	PRODUCT	QTY
112961	17-DEC-89	2A44L	7
113012	11-JAN-90	41003	35
112989	03-JAN-90	114	6
113051	10-FEB-90	XK47	4
112968	12-OCT-89	41001	34

Таблица CUSTOMERS

COMPANY	CUST_REP	CREDIT_LIMIT
JSP Inc.	103	$50,000.00
First Corp.	101	$65,000.00

Рис. 3. Реляционная база данных, содержащая информацию о заказах

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

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

В реляционной базе данных информация организована в виде *таблиц*, разделённых на строки и столбцы, на пересечении которых содержатся значения данных. У каждой таблицы имеется уникальное имя, описывающее её содержимое. Более наглядно структуру таблицы иллюстрирует рис. 4, где изображена таблица OFFICES.

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

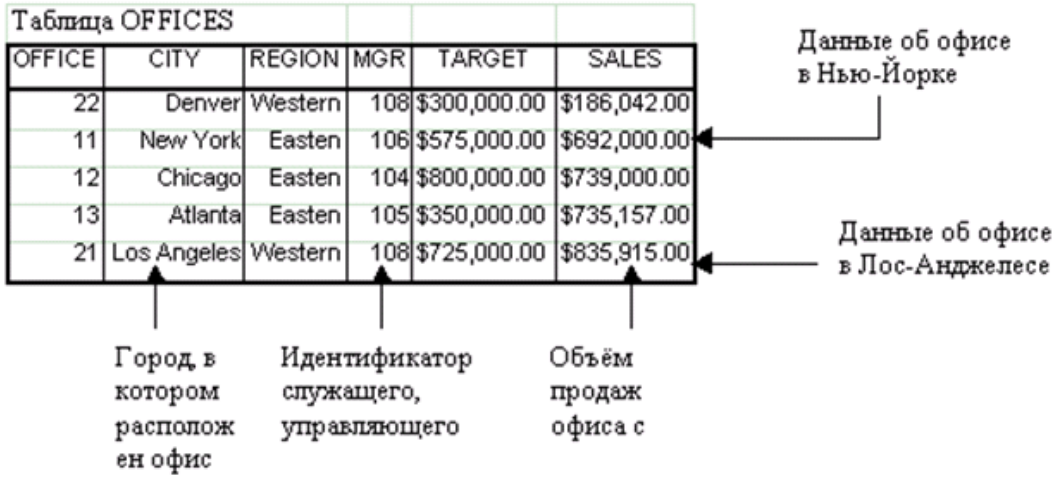


Рис. 4. Структура реляционной таблицы

Каждый вертикальный *столбец* таблицы OFFICES представляет один элемент данных для каждого из офисов. Например, в столбце CITY

содержатся названия городов, в которых расположены офисы. В столбце SALES содержатся объёмы продаж, обеспечиваемые офисами.

На пересечении каждой строки с каждым столбцом таблицы содержится в точности одно значение данных. Например, в строке, представляющей нью-йоркский офис, в столбце CITY содержится значение "New York". В столбце SALES той же строки содержится значение $692.000.000, которое является объёмом продаж нью-йоркского офиса с начала года.

Все значения, содержащиеся в одном и том же столбце, являются данными одного типа. Например, в столбце CITY содержатся только слова, в столбце SALES денежные суммы, а в столбце MGR целые числа, представляющие идентификаторы служащих. Множество значений, которое может содержаться в столбце, называется *доменом* этого столбца. Доменом столбца CITY является множество названий городов. Доменом столбца SALES является любая денежная сумма. Домен столбца REGION состоит всего из двух значений, "Eastern" и "Western", поскольку у компании всего два торговых региона.

У каждого столбца в таблице есть своё *имя*, которое обычно служит заголовком столбца. Все столбцы в одной таблице должны иметь уникальные имена, однако разрешается присваивать одинаковые имена столбцам, расположенным в различных таблицах. На практике такие имена столбцов, как NAME, ADDRESS, QTY, PRICE и SALES часто встречаются в различных таблицах одной базы данных.

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

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

В таблице может содержаться любое количество строк. Вполне допустимо существование таблицы с нулевым количеством строк. В таком случае она называется *пустой***.** Пустая таблица сохраняет структуру, определённую её столбцами, просто в ней не содержится данные.

Первичные ключи

Поскольку строки в реляционной таблице не упорядочены, нельзя выбрать строку по ее номеру в таблице. Здесь "первой", "последней" или "тринадцатой" строки. Тогда каким же образом можно указать в таблице конкретную строку, например, строку для офиса, расположенного в Денвере?

В правильно построенной реляционной базе данных в каждой таблице есть один или несколько столбцов, значения в которых во всех строках разные. Этот столбец (столбцы) называется *первичным ключом* таблицы. Для иллюстрации можно посмотреть на базу данных, показанную на рис. 4. На первый взгляд, первичным ключом таблицы OFFICES могут служить и столбец OFFICE, и столбец CITY. Однако в случае, если компания будет расширяться и откроет в каком-либо городе второй офис, столбец CITY больше не сможет выполнять роль первичного ключа. На практике в качестве первичных ключей таблиц обычно следует выбирать идентификаторы, такие как идентификатор офиса (OFFICE в таблице OFFICES), служащего (EMPL_NUM в таблице SALESREPS) и клиента (CUST_NUM в таблице CUSTOMES). А в случае; с таблицей ORDERS выбора нет - единственным столбцом, содержащим уникальные значения, является номер заказа (ORDER_NUM).

Таблица PRODUCTS

MFR_ID	PRODUCT_ID	DESCRIPTION	PRICE	QTY_ON_HAND
REI	2A45C	Ratchet Link	$79.00	210
ACI	4100Y	Widget Remover	$2,750.00	25
QSA	KX47	Reducer	$355.00	38
BIC	41672	Plate	$180.00	0

Первичный ключ

Рис. 5. Пример таблицы с составным первичным ключом

Таблица PRODUCTS, фрагмент которой показан на рисунке 5, является примером таблицы, в которой первичный ключ представляет собой *комбинацию* столбцов. Такой первичный ключ называется *составным.* Столбец MRF_ID содержит идентификаторы производителей всех товаров, перечисленных в таблице, а столбец PRODUCT_ID содержит номера, присвоенные товарам производителями. Может показаться, что столбец PRODUCT_ID мог бы и один выполнять роль первичного ключа, однако ничто не мешает двум различным производителям присвоить своим изделиям одинаковые номера. Таким образом, в качестве первичного ключа таблицы PRODUCTS необходимо использовать комбинацию столбцов MRF_ID и PRODUCT_ID. Для каждого из товаров, содержащихся в таблице, комбинация значений в этих столбцах будет уникальной.

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

Отношения предок/потомок (один-ко-многим)

Одним из отличий реляционной модели от первых моделей представления данных было то, что в ней отсутствовали явные указатели, используемые для реализации отношений предок/потомок в иерархической модели данных. Однако вполне очевидно, что отношения предок/потомок существуют и в реляционных базах данных. Например, в базе данных, показанной на рисунке 6, каждый из служащих закреплен за конкретным офисом, поэтому ясно, что между строками таблицы OFFICES и таблицы

SALESREPS существует отношение. Не приводит ли отсутствие явных указателей в реляционной модели к потере информации?

Как следует из рисунка 6, ответ на этот вопрос должен быть отрицательным. На рисунке изображено несколько строк из таблиц OFFICES и SALESREPS. Необходимо обратить внимание на то, что в столбце REP_OFFICE таблицы SALESREPS содержится идентификатор офиса, в котором работает служащий. Доменом этого столбца (множеством значений, которые могут в нем храниться) является множество идентификаторов офисов, содержащихся в столбце OFFICE таблицы OFFICES.

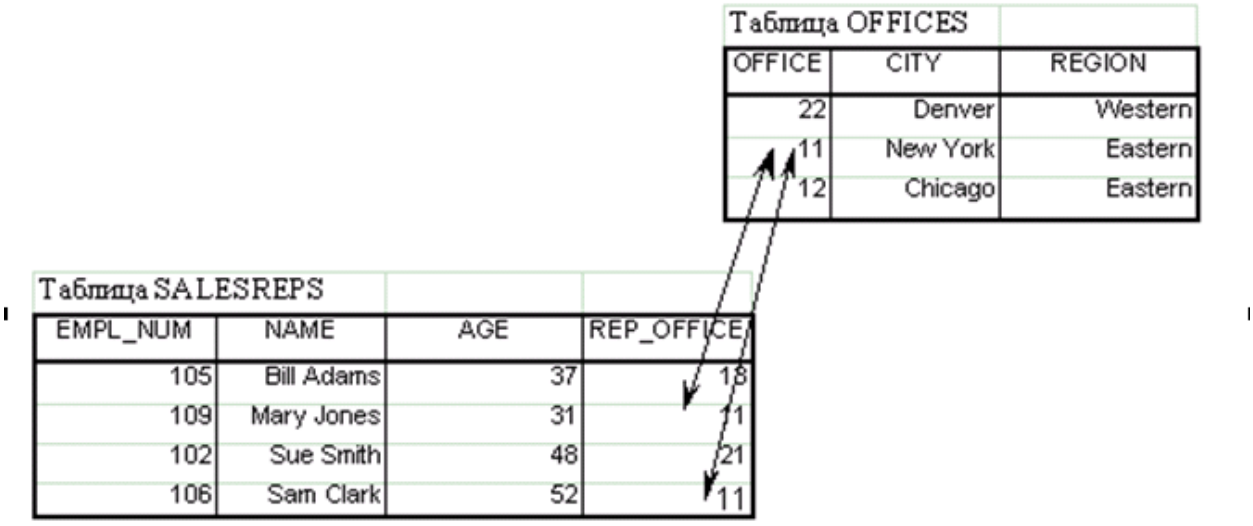


Рис. 6. Отношение предок/потомок в реляционной базе данных

То, в каком офисе работает Мэри Джонс (Mary Jones), можно узнать, определив значение столбца REP_OFFICE в строке таблицы SALESREPS для Мэри Джонс (число II) и затем отыскав в таблице OFFICES строку с таким же значением в столбце OFFICE (это для офиса в Нью-Йорке). Таким же образом, чтобы найти всех служащих нью-йоркского офиса, следует запомнить значение столбца OFFICE для Нью-Йорка (число II), а потом просмотреть таблицу SALESREPS и найти все строки, в столбце REP_OFFICE которых содержится число 11 (это строки для Мэри Джонс и Сэма Кларка (Sam Clark)).

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

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

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

Если таблица связана с несколькими другими таблицами, она может иметь несколько внешних ключей. На рисунке 7 показаны три внешних ключа таблицы ORDERS из учебной базы данных:

· столбец REP является внешним ключом для таблицы SALESREPS связывает каждый заказ со служащим, принявшим его;

· столбец CUST является внешним ключом для таблицы CUSTOMES и связывает каждый заказ с клиентом, разместившим его;

· столбцы MRF и PRODUCT совокупно представляют собой составной внешний ключ для таблицы PRODUCTS, который связывает каждый заказ с заказанным товаром.

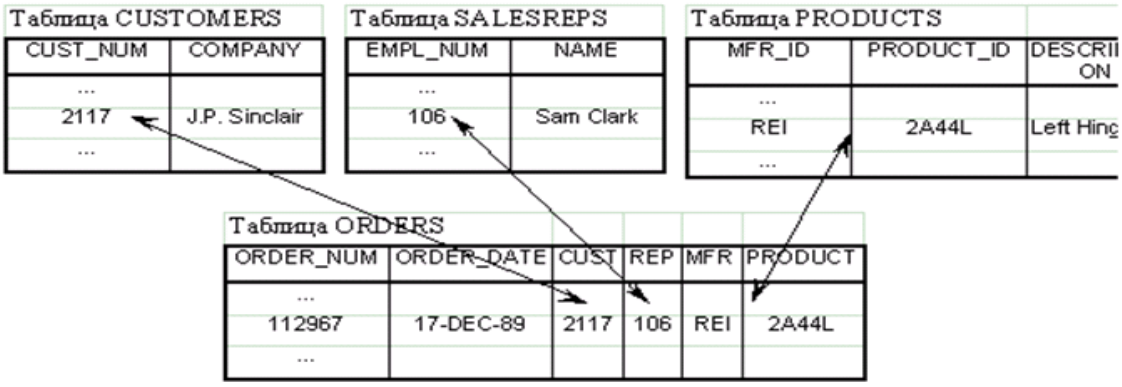


Рис. 7. Множественные отношения предок/потомок в реляционной базе данных (отношение многие-ко-многим)

Отношения предок/потомок, созданные с помощью трех внешних ключей в таблице ORDERS, могут показаться знакомыми. И действительно, это те же самые отношения, что и в сетевой базе данных, представленной на рис. 2.

В ходе информационного процесса данные преобразуются из одного вида в другой с помощью различных методов. Обработка данных включает в себя множество операций. По мере развития научно-технического прогресса и общего усложнения связей в человеческом обществе возрастают неуклонно трудозатраты на обработку данных. Прежде всего, это связано с постоянным усложнением условий управления производством и обществом. Второй фактор, также вызывающий общее увеличение объемов обрабатываемых данных, связан с научно-техническим прогрессом, а именно с быстрыми темпами появления и внедрения новых носителей данных, средств их хранения и доставки.

В структуре возможных операций с данными можно выделить следующие:

• *сбор* - накопление информации с целью обеспечения достаточной полноты для принятия решений;

• *формализация* - приведение данных, поступающих из разных источников, к одинаковой форме, чтобы сделать их сопоставимыми между собой, то есть повысить их уровень доступности;

• *фильтрация* - отсеивание «лишних» данных, в которых нет необходимости для принятия решений; при этом должен уменьшаться уровень «шума», а достоверность и адекватность данных должны возрастать;

• *сортировка* - упорядочение данных по заданному признаку с целью удобства использования; эта процедура повышает доступность информации;

• *архивация* - организация хранения данных в удобной и легкодоступной форме; служит для снижения экономических затрат по хранению данных и повышает общую надежность информационного процесса в целом;

• *защита* - комплекс мер, направленных на предотвращение утраты, воспроизведения и модификации данных;

• *транспортировка* - прием и передача (доставка и поставка) данных между удаленными участниками информационного процесса; при этом источник данных в информатике принято называть *сервером,* а потребителя - *клиентом;*

• *преобразование данных* - перевод данных из одной формы в другую или из одной структуры в другую. Преобразование данных часто связано с изменением типа носителя, например, книги можно хранить в обычной бумажной форме, но можно использовать для этого и электронную форму, и микрофотопленку. Необходимость в многократном преобразовании данных возникает также при их транспортировке, особенно если она осуществляется средствами, не предназначенными для транспортировки данного вида данных. В качестве примера можно упомянуть, что для транспортировки цифровых потоков данных по каналам телефонных сетей (которые изначально были ориентированы только на передачу аналоговых сигналов в узком диапазоне частот) необходимо преобразование цифровых данных в некое подобие звуковых сигналов, чем и занимаются специальные устройства - *телефонные модемы.*

Приведенный здесь список типовых операций с данными далеко не полон. Миллионы людей во всем мире занимаются созданием, обработкой, преобразованием и транспортировкой данных, и на каждом рабочем месте

выполняются свои специфические операции, необходимые для управления социальными, экономическими, промышленными, научными и культурными процессами. Полный список возможных операций составить невозможно, да и не нужно. Сейчас нам важен другой вывод: *работа с информацией может иметь огромную трудоемкость, и ее надо автоматизировать.*

Процедура доступа к данным может быть инициирована как самим компьютером (для решения каких-либо своих технических задач), так и конечным пользователем. В последнем случае пользователь формирует запрос, куда включает, в частности, обозначение требуемого вида доступа или действия и указание на то, над какими данными это действие надо выполнить. Как отмечалось ранее, идентификация данных осуществляется с помощью ключей. В качестве же требуемого действия может производиться одно из следующих: добавление, удаление, изменение, просмотр элемента или обработка данных из элемента.

При добавлении элемента информационный массив пополняется новыми данными в виде записи файла или файла в целом, соответственно, для структурированных и неструктурированных данных. В запросе в этом случае, помимо указанной выше информации, приводится и сам новый элемент. При этом объем информационного массива увеличивается.

Удаление, наоборот, является обратным действием, вызывающим исключение упомянутых данных. Это действие приводит к уменьшению объема информационного массива.

Изменение относится не к элементу, а к его составляющим – полям записи файла или тексту, хранящемуся в файле, и означает, в свою очередь, удаление прежних значений полей или строк текста и/или добавление новых. В запрос включается дополнительная информация, указывающая на требуемые составляющие изменяемого элемента, а также сами новые значения этих составляющих. Объем информационного массива при этом не меняется для структурированных данных и, возможно, меняется для неструктурированных;

Просмотр связан с предоставлением данных пользователю на устройстве вывода компьютера, как правило, на дисплее. В запросе в этом случае дополнительно указывается, какие составляющие элемента требуется просмотреть (по умолчанию просматривается весь элемент).

Обработка предусматривает выполнение некоторых арифметических операций над данными элемента, например, накопление суммы и т.д., и относится только к структурированным данным, а потому далее не рассматривается.

Чтобы выполнить любое их указанных выше действий, нужный элемент должен быть предварительно найден в информационном массиве, для чего выполняется его поиск (для добавления нового элемента тоже делается попытка его поиска, которая заканчивается неудачно, и тогда элемент добавляется). Под поиском элемента понимается определение его местонахождения в информационном массиве. Таким образом, любой доступ включает поиск, что делает эту фазу доступа наиболее значимой.

Технологии доступа при выполнении действий изменения элемента показана на рис. 8.

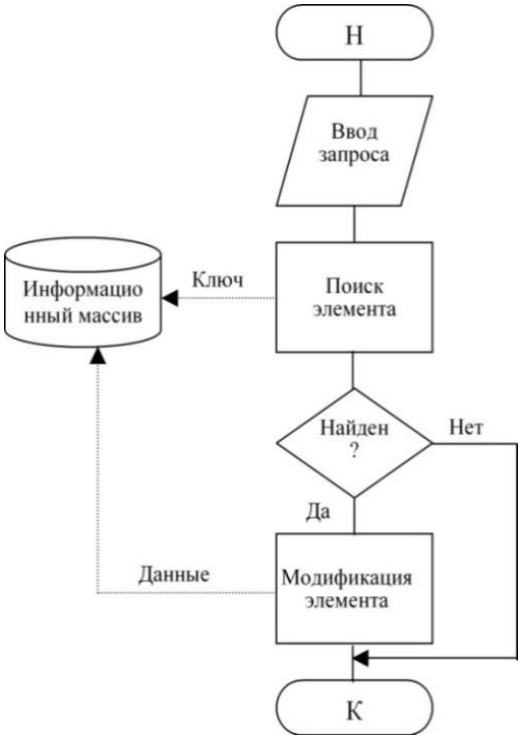


Рис. 8. Технологии доступа при выполнении действий изменения элемента

Технологии доступа при выполнении действий добавления элемента показаны на рис. 9: Технология удаления изображена на рис. 10.

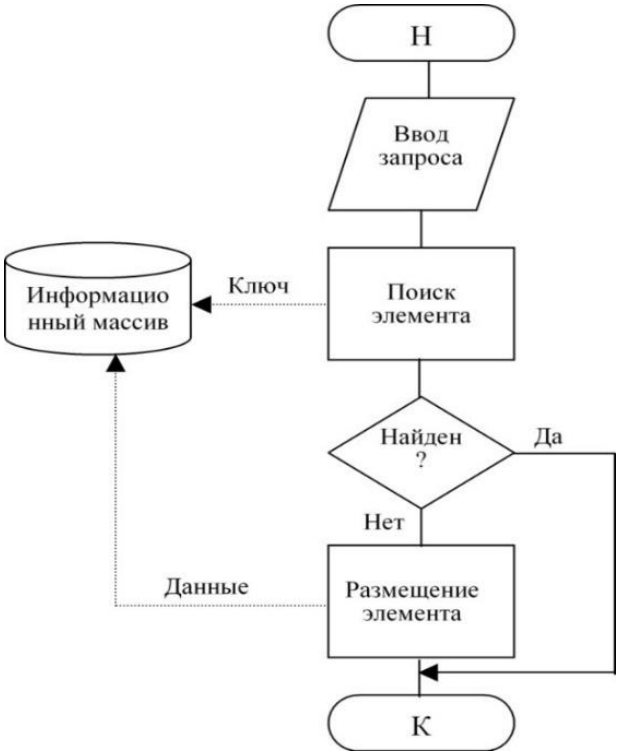


Рис. 9. Технологии доступа при выполнении действий добавления элемента

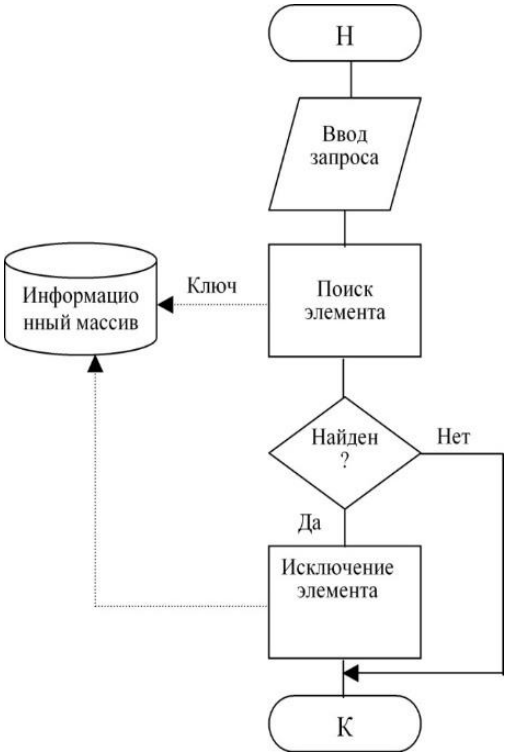


Рис. 10. Технология удаления элемента

Технология просмотра элемента приведена на рис. 11. Различие в схемах состоит в том, что по технологии рис. 8 и 9 выполняется воздействие на информационный массив с целью его изменения, для чего в него передаются данные, по технологии рис. 10 воздействие не связано с передачей данных, а по схеме рис. 11 данные выводятся из информационного массива без его изменения.

При выполнении рассмотренных действий над элементами информационного массива на практике важны два фактора, противоречащие друг другу: временной фактор, в соответствии с которым запрос пользователя должен обрабатываться в минимальные сроки, и фактор минимизации требуемого объема памяти для хранения данных.

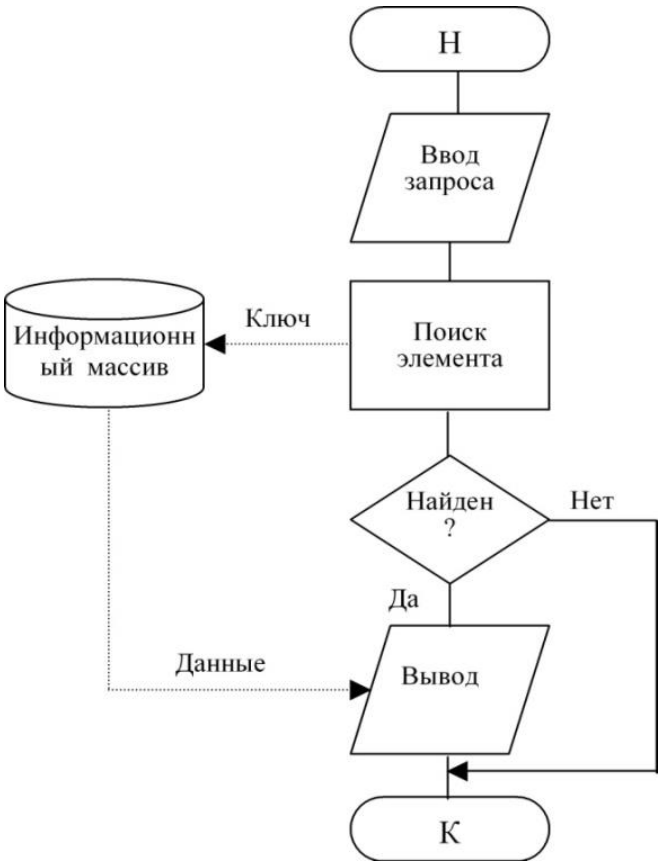


Рис. 11. Технология просмотра элемента

Для уменьшения времени обработки запроса особые усилия прилагаются к применению таких структур хранения данных, которые позволяли бы оптимизировать поисковые операции, возможно, за счет дополнительных описаний данных. Это, очевидно, повышает расход памяти. Поэтому при проектировании моделей данных учитывается предполагаемый режим эксплуатации информационного массива: если это интерактивный режим, то основное внимание уделяется минимизации времени доступа к данным, если же режим пакетный, то минимизируют требуемую память. Кроме того, на выбор модели влияют особенности той предметной области, которая отражается в структурах хранения.

В силу вышесказанного, основное внимание в данном разделе уделено задачам организации хранения данных разных видов и поиска по ключам, входящим в запросы пользователей, поскольку поисковые операции и определяют, в основном, продолжительность различных действий над

информационным массивом. Из приведенных типов действий в рассмотрение включены добавление и просмотр элементов данных, поскольку добавление связано с воздействием на информационный массив и изменением его объема (напомним, что удаление является обратным действием по отношению к добавлению), а просмотр - это наиболее часто выполняемые действия на практике. При этом рассматриваются общие вопросы работы с текстовой и структурированной информацией, методы и модели, используемые при организации хранения, поиска и добавления данных.

Излагаемые модели данных и алгоритмы доступа к ним составляют “brainware” современной информатики, носят универсальный характер и применяются в большинстве систем, связанных с хранением и обработкой информационных массивов.

1.4. Операторы реляционной алгебры

Реляционная алгебра – это набор операций, которые принимают отношения в качестве операндов и возвращают отношение в качестве результата. Первая версия этой алгебры была определена Э. Коддом.

В основе всех реляционных БД лежит использование реляционной алгебры, которая обеспечивает запись выражений для реализации на некотором языке, например, SQL. Если возможности языка как минимум соответствуют возможностям, обеспеченным алгебраическими операциями, то его называют *реляционно полным*.

Обычно выделяют 8 основных операторов реляционной алгебры и несколько дополнительных (их количество меняется со временем). Мы рассмотрим два дополнительных оператора: расширения и подведения итогов.

Основные операторы реляционной алгебры

Реляционная алгебра включает восемь основных операций, которые подразделяются на две группы, в каждую из которых входит четыре операции.

1. Традиционные операции с множествами, модифицированные для таблиц (отношений).

1) Объединение: **A UNION B**

Результатом операции объединения является отношение, содержащее все кортежи, принадлежащие одному из двух или обоим отношениям.

В отличие от объединения множеств, результатом объединения отношений должно стать отношение, а не набор разнородных кортежей.

При объединении должны соблюдаться два условия:

- отношения должны быть совместимы по типу, т.е. иметь одно и то же множество атрибутов, определённых на одних и тех же доменах;
- результатом каждой операции должно быть также отношение (свойство замкнутости).

Пусть имеется отношение *Students*:

StudentID	Name	GroupID	BirthDate
1	Казаков Петр	2	23.04.1990
2	Васильев Иван	1	11.05.1991
4	Шишкина Дарья	2	23.09.1991

И отношение *Teachers*:

TeacherID	**Name**	**BirthDate**
1	Кислицын О.П.	1.2.1970
2	Царев С.М.	10.03.1964
4	Пестов Д.Н.	2.05.1980

Нам требуется вывести список всех преподавателей и студентов с указанием их дня рождения.

Результатом выполнения будет следующее отношение:

Name	**BirthDate**
Казаков Петр	23.04.1990
Васильев Иван	11.05.1991
Шишкина Дарья	23.09.1991
Кислицын О.П.	1.2.1970
Царев С.М.	10.03.1964
Пестов Д.Н.	2.05.1980

2) Пересечение: **A INTERSECT B**

Результатом операции пересечения является отношение, содержащее кортежи, которые принадлежат обоим отношениям.

Для операции пересечения необходимы те же два условия, что и для объединения: совместимость по типу и замкнутость.

Пример: рассмотрим отношение *Teachers* (преподаватели) и *Supervisors* (кураторы, могут руководить преподавателями):

Teachers:

TeacherID	**Name**	**BirthDate**
1	Кислицын О.П.	1.2.1970
2	Царев С.М.	10.03.1964
4	Пестов Д.Н.	2.05.1980

Supervisors:

SupervisorID	**Name**	**BirthDate**
1	Кислицын О.П.	1.2.1970
2	Царев С.М.	10.03.1964
4	Нечаев Н.В.	12.08.1970

Требуется вывести всех преподавателей (из таблицы *Teachers*), которые одновременно являются кураторами.

Результатом выполнения запроса будет:

Name
Кислицын О.П.
Царев С.М.

3) Вычитание: **A MINUS B**

Результатом операции вычитания является отношение, которое содержит все кортежи, принадлежащие первому отношению и не принадлежащие второму отношению.

Условия необходимы те же: совместимость по типу и замкнутость.

Пример: рассмотрим отношение *Teachers* (преподаватели) и *Supervisors* (кураторы, могут руководить преподавателями):

Teachers:

TeacherID	Name	BirthDate
1	Кислицын О.П.	1.2.1970
2	Царев С.М.	10.03.1964
4	Пестов Д.Н.	2.05.1980

Supervisors:

SupervisorID	Name	BirthDate
1	Кислицын О.П.	1.2.1970
2	Царев С.М.	10.03.1964
4	Нечаев Н.В.	12.08.1970

Требуется вывести всех преподавателей (из таблицы *Teachers*), которые одновременно являются кураторами.

Результатом выполнения запроса будет:

Name
Пестов Д.Н.

4) Произведение: **A TIMES B**

Результатом является отношение, содержащее все возможные кортежи, которые являются сочетанием двух кортежей, принадлежащих двум отношениям.

Для произведения необходимо свойство замкнутости, т.е. результатом является не просто множество пар кортежей, а множество целых кортежей. В реляционной алгебре каждая пара кортежей заменяется одним благодаря сцеплению (*сцепление* – это объединение в смысле теории множеств, а не реляционной алгебры).

При произведении может возникнуть проблема одинаковых имён атрибутов. В этом случае одно из конфликтующих полей необходимо переименовать.

Пример: рассмотрим отношения *Students* и *Courses* (учебные курсы):

Students:

StudentID	**Name**	**GroupID**	**BirthDate**
1	Казаков Петр	2	23.04.1990
2	Васильев Иван	1	11.05.1991
4	Шишкина Дарья	2	23.09.1991

Courses:

CourseID	**Name**
1	Базы данных
2	Технологии программирования
3	Программирование в среде C++

Требуется для каждого студента вывести список всех доступных учебных курсов.

Результат выполнения запроса:

Name	Name
Казаков Петр	Базы данных
Васильев Иван	Базы данных
Шишкина Дарья	Базы данных
Казаков Петр	Технологии программирования
Васильев Иван	Технологии программирования
Шишкина Дарья	Технологии программирования
Казаков Петр	Программирование в среде C++
Васильев Иван	Программирование в среде C++
Шишкина Дарья	Программирование в среде C++

2. Специальные реляционные операции.

1) ***Выборка (или ограничение)***

Результатом выборки является отношение, содержащее все кортежи из исходного отношения, которые удовлетворяют определённому условию.

Пусть дано следующее отношение *Students*:

StudentID	Name	GroupID
1	Казаков Петр	2
2	Васильев Иван	1
4	Шишкина Дарья	2
5	Драгомиров Евгений	1
6	Васнецова Евгения	2

Выберем лишь тех студентов, которые принадлежат группе 2:

Результатом выполнения будет отношение:

StudentID	Name	GroupID
1	Казаков Петр	2
4	Шишкина Дарья	2
6	Васнецова Евгения	2

2) Проекция

Результатом проекции является отношение, содержащее все кортежи после удаления из них некоторых атрибутов. В этом случае результирующие кортежи называются *подкортежами*.

Обратим внимание на два момента:

1) возможно указание всех атрибутов исходного отношения для проекции – получится *тождественная проекция*;

2) возможно указать пустой список атрибутов – получится *нулевая проекция*.

Пусть дано следующее отношение *Students*:

StudentID	**Name**	**GroupID**	**BirthDate**
1	Казаков Петр	2	23.04.1990
2	Васильев Иван	1	11.05.1991
4	Шишкина Дарья	2	23.09.1991

Выберем из таблицы только имена и даты рождений:

Результатом выполнения будет отношение:

Name	**BirthDate**
Казаков Петр	23.04.1990
Васильев Иван	11.05.1991
Шишкина Дарья	23.09.1991

3) Соединение

Результат соединения – это отношение, кортежи которого являются сочетанием двух кортежей, принадлежащих двум начальным отношениям и имеющих общие значения для одного или нескольких атрибутов (общее значение в результирующем отношении появляется только один раз).

Пусть дано следующее отношение *Students*:

StudentID	**Name**	**GroupID**	**BirthDate**
1	Казаков Петр	2	23.04.1990
2	Васильев Иван	1	11.05.1991
4	Шишкина Дарья	2	23.09.1991

и отношение *Groups*:

GroupID	**GroupName**
1	ПМ-11
2	ПМ-12
3	ПМ-21

Соединим эти таблицы по полю **GroupID** и выведем имя студента и название учебной группы:

Результатом выполнения будет отношение:

Name	**GroupName**
Казаков Петр	ПМ-12
Васильев Иван	ПМ-11
Шишкина Дарья	ПМ-12

Это соединение обладает свойствами ассоциативности и коммутативности.

4) Деление: **A DIVIDEBY B**

Результатом деления двух отношений (бинарного и унарного) является отношение, содержащее все значения атрибута первого бинарного отношения, которые соответствуют всем значениям унарного отношения.

R1:

A	X
A	Y
B	Z
B	X
C	Y

R2:

X
Y

В результате получится отношение:

R:

A

Операции расширения и подведения итогов

После того, как Э. Кодд предложил восемь основных операций, многочисленные авторы предложили новые алгебраические операции. Мы рассмотрим лишь две из них – расширение и подведение итогов. Эти операции удачно дополняют основной набор и являются наиболее востребованными.

1. Расширение

С помощью операции расширения из отношения создаётся новое отношение, содержащее новый атрибут, значения которого получены посредством скалярных вычислений.

Пусть дано следующее отношение *Teachers*:

TeacherID	Name	BirthDate
1	Кислицын О.П.	1.2.1970
2	Царев С.М.	10.03.1964
4	Пестов Д.Н.	2.05.1980

Результатом выполнения будет отношение:

TeacherID	Name	BirthDate	Age
1	Кислицын О.П.	1.2.1970	39
2	Царев С.М.	10.03.1964	45
4	Пестов Д.Н.	2.05.1980	29

2. Подведение итогов

Операция подведения итогов даёт возможность "вертикальных" вычислений. Для этого используются агрегатные функции, которые для набора значений возвращают одно единственное.

Пусть дано следующее отношение *Students*:

StudentID	Name	GroupID
1	Казаков Петр	2
2	Васильев Иван	1
4	Шишкина Дарья	2
5	Драгомиров Евгений	1
6	Васнецова Евгения	2

Требуется подсчитать, сколько студентов в каждой группе:

Результатом выполнения будет отношение:

GroupID	StudentCount
1	2
2	3

Операторы обновления

Предназначены для управления данными в таблицах. Существует три операции обновления.

1) Вставка записи

Позволяет добавлять одну или несколько новых записей в отношение.

Пусть имеется отношение *Students*:

StudentID	Name	GroupID
1	Казаков Петр	2
2	Васильев Иван	1
4	Шишкина Дарья	2

После выполнения, отношение *Students* будет выглядеть следующим образом:

StudentID	Name	GroupID
1	Казаков Петр	2
2	Васильев Иван	1
4	Шишкина Дарья	2
5	Драгомиров Евгений	1
6	Васнецова Евгения	2

2) Удаление записи.

Удаляет все записи из отношения, или только записи, удовлетворяющие заданному критерию.

Пусть имеется отношение *Students*:

StudentID	Name	GroupID
1	Казаков Петр	2
2	Васильев Иван	1
4	Шишкина Дарья	2
5	Драгомиров Евгений	1
6	Васнецова Евгения	2

После выполнения, отношение *Students* будет выглядеть так:

StudentID	Name	GroupID
1	Казаков Петр	2
4	Шишкина Дарья	2
6	Васнецова Евгения	2

3) Обновление записи

Позволяет обновлять значения указанных полей всех или только определенных записей.

Пусть имеется отношение *Students*:

StudentID	Name	GroupID
1	Казаков Петр	2
2	Васильев Иван	1
4	Шишкина Дарья	2
5	Драгомиров Евгений	1
6	Васнецова Евгения	2

После выполнения, отношение *Students* будет выглядеть следующим образом:

StudentID	Name	GroupID
1	Казаков Петр	2
2	Васильев Иван	2
4	Шишкина Дарья	2
5	Драгомиров Евгений	2
6	Васнецова Евгения	2

1.5. Процесс нормализации данных.

Нормализация отношений (таблиц) — одна из основополагающих частей теории реляционных баз данных. Нормализация имеет своей целью избавиться от избыточности в отношениях и модифицировать их структуру таким образом, чтобы процесс работы с ними не был обременён различными посторонними сложностями. При игнорировании такого подхода эффективность проектирования стремительно снижается, что вкупе с прочими подобными вольностями может привести к критическим последствиям.

Первая нормальная форма

Отношение находится **в первой нормальной форме** (сокращённо 1НФ), если все его атрибуты атомарны, то есть если ни один из его атрибутов нельзя разделить на более простые атрибуты, которые соответствуют каким-то другим свойствам описываемой сущности.

Будем называть исходное отношение основным, а значение неатомарного атрибута — подчинённым.

Для того, чтобы нормализовать исходное отношение, атрибуты которого неатомарны, необходимо объединить схемы основного и подчинённого отношений. Кроме того, если, например, таблица, соответствующая

ненормализованному отношению, уже содержится в БД и заполнена информацией, задача усложняется тем, что значение неатомарного атрибута может в свою очередь содержать несколько кортежей.

Следует пояснить сказанное на примере. Рассмотрим отношение, имеющее атрибуты «Код сотрудника», «ФИО», «Должность», «Проекты». Очевидно, что один сотрудник может работать над несколькими проектами. Предположим, что проект описывается идентификатором, названием и датой сдачи.

Код сотрудника	**ФИО**	**Должность**	**Проекты**
1	Иванов Иван Иванович	Программист	ID: 123; Название: Система управления паровым котлом; Дата сдачи: 30.09.2011 ID: 231; Название: ПС для контроля и оповещения о превышениях ПДК различных газов в помещении; Дата сдачи: 30.11.2011 ID: 321; Название: Модуль распознавания лиц для защитной системы; Дата сдачи: 01.12.2011

Легко заметить, что не все атрибуты этого отношения атомарны (неделимы). В частности, атрибут «Проекты» можно разделить на три более простых атрибута: «Код проекта», «Название», «Дата сдачи», а значение этого атрибута для сотрудника Иван Иванович Иванов содержит несколько кортежей — информацию о трёх проектах.

Примечание: *с некоторой точки зрения атрибут «ФИО» можно также считать неатомарным и в таком случае его также следует разделить на более простые, как «Фамилия», «Имя», «Отчество».*

Теперь настало время рассмотреть алгоритм нормализации отношения до 1НФ.

1. Создать новое отношение, схема которого будет получена путём слияния основной и подчинённой схем исходного отношения в одну.

2. Для каждого кортежа исходного отношения включить в новое столько строк, сколько кортежей содержится в подчинённом отношении этого кортежа.

3. Заполнить значения атрибутов нового отношения, соответствующих атрибутам подчинённого отношения.

4. Заполнить строки нового отношения значениями атомарных атрибутов исходного.

Применим этот алгоритм к приведённому выше отношению. Схема нового отношения будет состоять из 6 атрибутов: «Код сотрудника», «ФИО», «Должность», «Код проекта», «Название», «Дата сдачи». Для одного единственного кортежа заданного отношения, добавим в новое три строки, по одной для каждого проекта (по количеству кортежей в подчинённом отношении). Теперь можно заполнить значения разделённых атрибутов кортежами из подчинённого отношения. Затем перенесём в каждую из этих строк значения атомарных атрибутов: «Код сотрудника», «ФИО», «Должность» (как Вы уже догадались, все три строки будут содержать одинаковые значения этих атрибутов).

Результат будет выглядеть так:

Код сотрудника	**ФИО**	**Должность**	**Код проекта**	**Название**	**Дата сдачи**
1	Иванов Иван Иванович	Программист	123	Система управления паровым котлом	30.09.2011
1	Иванов Иван Иванович	Программист	231	ПС для контроля и оповещения о превышениях ПДК различных газов в помещении	30.11.2011
1	Иванов Иван Иванович	Программист	321	Модуль распознавания лиц для защитной системы	01.12.2011

Вторая нормальная форма

Ясно, что отношение, находящееся в 1НФ, также может обладать избыточностью. Для её устранения предназначена вторая нормальная форма. Но прежде чем приступить к её описанию, сначала следует выявить недостатки первой.

Пусть исходное отношение содержит информацию о поставке некоторых товаров и их поставщиках.

Код поставщика	**Город**	**Статус города**	**Код товара**	**Количество**
1	Москва	20	1	300
1	Москва	20	2	400
1	Москва	20	3	100
2	Ярославль	10	4	200
3	Ставрополь	30	5	300
3	Ставрополь	30	6	400
4	Псков	15	7	100

Заранее известно, что в этом отношении содержатся следующие функциональные зависимости:

{ {Код поставщика, Код товара} -> { Количество},

{Код поставщика} -> {Город},

{Код поставщика} -> {Статус},

{Город} -> {Статус}}

Первичный ключ в отношении: {Код поставщика, Код товара}. Очевидно, что отношение обладает избыточностью: оно описывает две сущности — поставку и поставщика. В связи с этим возникают следующие аномалии:

- Аномалия вставки. В отношение нельзя добавить информацию о поставщике, который ещё не поставил ни одного товара.
- Аномалия удаления. Если от поставщика была только одна поставка, то при удалении информации о ней будет удалена и вся информация о поставщике.

• Аномалия обновления. Если необходимо изменить какую-либо информацию о поставщике (например, поставщик переехал в другой город), то придётся изменять значения атрибутов во всех записях о поставках от него.

Физический смысл избыточности исходного отношения заключается в том, что оно описывает *не одну* сущность, а *две —поставку* и *поставщика*.

Чтобы устранить эти аномалии, необходимо разбить исходное отношение на проекции:

1. В первую следует включить первичный ключ и все неключевые атрибуты *явно* зависимые от него.

2. В остальные проекции (в данном случае она одна) будут включены неключевые атрибуты, зависящие от первичного ключа *неявно*, вместе с той *частью* первичного ключа, от которой эти атрибуты зависят явно. В итоге будут получены два отношения:

Код поставщика	**Код товара**	**Количество**
1	1	300
1	2	400
1	3	100
2	4	200
3	5	300
3	6	400
4	7	100

Первому отношению теперь соответствуют следующие функциональные зависимости:

{Код поставщика, Код товара} -> {Количество}

Код	**Город**	**Статус города**
1	Москва	20
2	Ярославль	10
3	Ставрополь	30
4	Псков	15

Второму отношению соответствуют:

{ {Код поставщика} -> {Город},

{Код поставщика} -> {Статус},

{Город}->{Статус}}

Такое разбиение устранило аномалии, описанные выше: можно добавить информацию о поставщике, который ещё не поставлял товар, удалить информацию о поставке без удаления информации о поставщике, а также легко обновить информацию в случае если поставщик переехал в другой город.

Теперь можно сформулировать определение второй нормальной формы, до которого, скорее всего, читатель уже смог догадаться самостоятельно: отношение находится **во второй нормальной форме** (сокращённо 2НФ) тогда и только тогда, когда оно находится в первой нормальной форме и каждый его неключевой атрибут неприводимо зависим от первичного ключа.

Третья нормальная форма (3NF)

Основные критерии: Таблица находится во второй нормальной форме. Любой её не ключевой атрибут функционально зависит только от первичного ключа. Проще говоря, третье правило требует выносить все не ключевые поля, содержимое которых может относиться к нескольким записям таблицы в отдельные таблицы.

Например, есть у нас таблица:

Имя шпиона	**Государство**
Джеймс Бонд	Великобритания
Ким Филби	СССР
Штирлиц	СССР

В этой таблице ключом является имя шпиона. А не ключевым полем – государство, на которое он работает. Вполне логично предположить, что в этой таблице государства могут быть одинаковыми для нескольких записей.

И для того, чтобы эта таблица находилась в третей нормальной форме, не обходимо ее разделить на две:

ID	Государство
1	Великобритания
2	СССР

Имя шпиона	Государство
Джеймс Бонд	1
Ким Филби	2
Штирлиц	2

Благодаря этому правилу, при удалении какого-то государства, имена шпионов не будут утеряны.

Вообще, говоря, на практике, совершенствовать таблицы заканчивают на этом этапе (приведя их в третью нормальную форму).

Методы приведения к 3NF:

• Удаление полей, не зависящих от ключа

Нормальная форма Бойса-Кодда (BCNF)

Эта форма почти то же самое, что и третья. С одним небольшим дополнительным условием.

Основные критерии:

• Таблица находится в третьей нормальной форме

• В таблице должен быть только один потенциальный первичный ключ

Другими словами, в таблице должен быть только один первичный ключ и не должно быть других потенциальных вариантов (например, набор не ключевых полей — это таблицы).

Методы приведения к BCNF:

• Вынести в отдельную таблицу потенциальные первичные ключи

ER (Entity-Relationship) - метод проектирования РБД

Одним из наиболее понятных и практически используемых методов проектирования реляционных баз данных является метод семантического моделирования. В качестве инструмента используются различные варианты диаграмм.

ОПРЕДЕЛЕНИЯ:

Сущность определяется как некоторый объект рассматриваемой предметной области, информация о котором должна быть отражена в базе данных.

Этот объект должен иметь *экземпляры* - конкретные представители данной сущности, отличающиеся друг от друга и допускающие однозначную идентификацию.

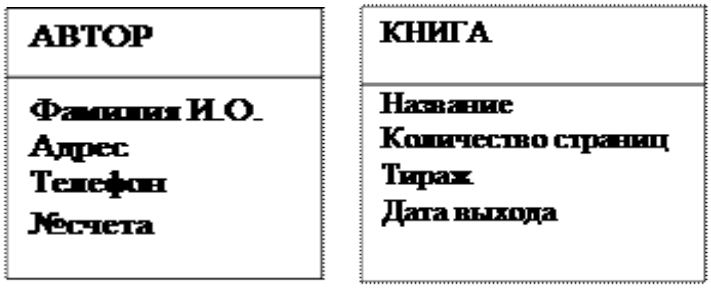
Связь – это некоторая ассоциация между двумя сущностями.

Атрибут – это свойство сущности.

Сущность – это, как правило, существительное; связь чаще всего выражается глаголом.

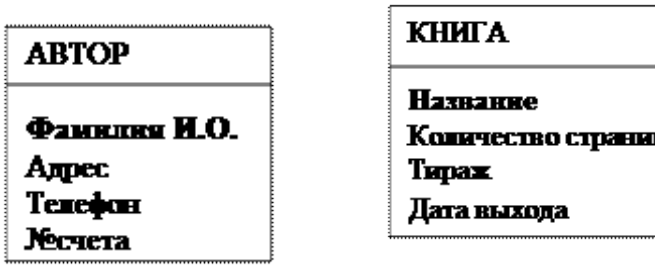
Например, проектируется база данных издательства, предназначенная для хранения информации о книгах и авторах, которые их написали. Тогда два главных объекта (две сущности, информация о которых должна быть отражена в базе данных) – это **книга** и **автор**. Эти сущности содержательно соединены с помощью связи **пишет**. Атрибуты сущности **книга** – это название, количество страниц, тираж, дата выхода сигнального экземпляра, цена и т.д. Атрибуты сущности **автор** – это фамилия, адрес, телефон, №счета и т.д.

Сущности изображаются в виде прямоугольника, атрибуты вписываются внутрь прямоугольника, изображающего сущность:



Данные определения не являются формальными, однако они приемлемы для использования при проектировании базы данных. Особенность ER-метода заключается в том, что разные проектировщики могут рассматривать одну и ту же предметную область с различных точек зрения, получая в итоге разные наборы сущностей и связей. Определение лучшего из нескольких возможных наборов является вопросом личного предпочтения.

Атрибут или набор атрибутов, используемый для идентификации экземпляра сущности, называется *ключом сущности*. Таким образом, ключ сущности должен быть уникальным для каждого экземпляра этой сущности. Ключ каждой сущности не должен быть избыточным, т.е. удаление любого атрибута из этого набора будет нарушать его уникальность. Ключевые атрибуты каким-либо образом выделяются на диаграмме (например, подчеркиванием или более жирным шрифтом).



В нашем примере в качестве ключевого атрибута сущности АВТОР было решено взять фамилию, а в качестве ключевого атрибута сущности КНИГА взять её название. Первое решение заведомо не является

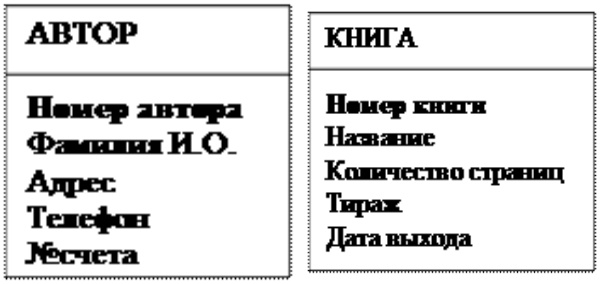
бесспорным: возможно появление авторов-однофамильцев (тогда атрибут **Фамилия И.О.** теряет уникальность и не может быть использован в качестве ключа). В принципе, допустимо появление книг с одинаковыми названиями.

Если есть хотя бы минимальное подозрение, что атрибут, выбираемый в качестве ключевого, может потерять свою уникальность, нужно отказаться от его использования в качестве ключа и попытаться подобрать на эту роль другой атрибут.

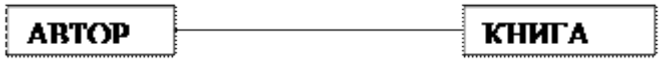
Если окажется, что ни один содержательный атрибут не может быть использован как ключевой, то существует (по меньшей мере) два способа решения этой проблемы:

- подобрать набор атрибутов, значения которых будут уникальными для каждого экземпляра сущности;
- ввести еще один атрибут, который не будет отражать какое-либо свойство сущности, но будет пригоден в качестве ключевого. Обычно таким атрибутом становится номер экземпляра анализируемой сущности.

Для нашего примера выберем второй способ:



Связь между двумя сущностями может быть представлена графически в виде ER-диаграммы:



Важной характеристикой связи является *степень связи*. Возможны следующие степени связи: *один-к-одному*, *один-ко-многим* и *многие-ко-многим*.

Кроме того, надо выявить к*ласс принадлежности сущности*, который характеризует обязательность включения каждого экземпляра сущности в связь.

Связь один-к-одному:

Связь один-к-одному подразумевает, что каждый экземпляр сущности, расположенной как в левой, так и в правой частях диаграммы, связывается не более, чем с одним экземпляром сущности, расположенной в противоположной части диаграммы.

Если все экземпляры сущности должны участвовать в связи, то участие называется обязательным, и изображается на ER-диаграмме кружком, помещенным в блок, изображающий сущность (при словесной формулировке такой связи обычно используется глагол «должен»):

В этой диаграмме отражено правило: «каждый автор должен писать книгу, причем только одну, и каждую книгу должен писать только один автор», т.е. в базе данных не будет информации об авторах, не пишущих ни одной книги, а также информации о книгах, которые пока никто не пишет.

Если не все экземпляры сущности должны участвовать в связи, то участие называется необязательным, и кружок на ER-диаграмме располагается вне блока сущности (при словесной формулировке такой связи обычно используется глагол «может»):

В этой диаграмме отражено правило: «каждый автор может писать не более одной книги, и каждую книгу должен писать только один автор», т.е. в базе данных допускается наличие авторов, не пишущих ни одной книги, но нет информации о книгах, которые пока никто не пишет.

Для сущностей АВТОР - КНИГА возможны еще два типа связи один-к-одному, отражающих два оставшихся варианта обязательности включения экземпляров:

«Каждый автор должен писать книгу, причем только одну, и каждую книгу пишет не более чем один автор (один или никто)», т.е. в базе данных не будет храниться информация об авторах, которые в данный момент не пишут ни одной книги, но допускаются книги, которые еще никто не пишет.

«Каждый автор пишет не более одной книги, и каждая книга пишется не более чем одним автором», т.е. в базе данных допускается наличие авторов, не пишущих сейчас ни одной книги, и наличие книг, которые еще никто не пишет.

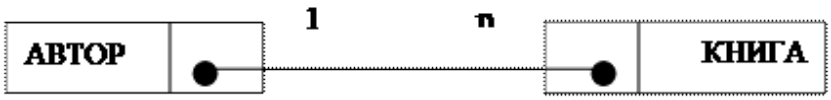
Каждая диаграмма представляет некоторый набор правил, принятых в данной предметной области (например, в издательстве) по поводу того, какого рода информация должна храниться в базе данных.

Связь один-ко -многим:

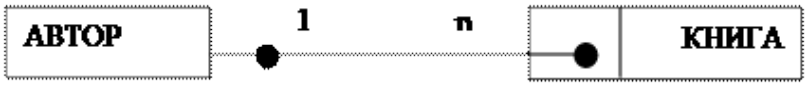
Связь один-ко-многим подразумевает, что один экземпляр сущности, расположенной в левой части диаграммы, связан с несколькими экземплярами сущности, расположенной в правой части диаграммы.

По-прежнему, если все экземпляры сущности должны участвовать в связи, то участие является обязательным, и изображается на ER-диаграмме кружком, помещенным в блок, изображающий сущность. Если не все экземпляры сущности должны участвовать в связи, то участие является необязательным, и кружок на ER-диаграмме располагается вне блока сущности.

Следующая диаграмма отражает связь один-ко-многим сущностей АВТОР – КНИГА, где экземпляры обеих сущностей вступают в обязательную связь:



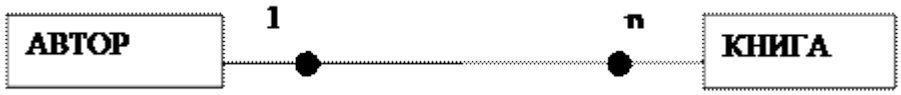
Таким образом, в этой диаграмме отражено правило: «каждую книгу пишет только один автор, но каждый автор может писать несколько книг»; обязательность включения экземпляров говорит о том, что в базе данных не будет информации об авторах, не пишущих ни одной книги, а также информации о книгах, которые пока никто не пишет.



В этой диаграмме отражено правило: «каждую книгу пишет только один автор, каждый автор может писать несколько книг либо не писать их вовсе», т.е. в базе данных допускается наличие авторов, не пишущих ни одной книги, но нет информации о книгах, которые пока никто не пишет.

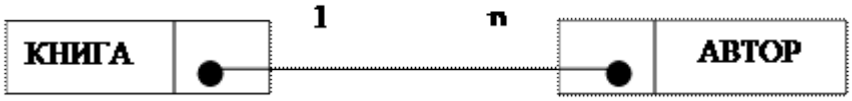
Для сущностей АВТОР - КНИГА возможны еще два типа связи один-ко-многим, отражающих два оставшихся варианта обязательности включения экземпляров:

«Каждую книгу может писать не более чем один автор; каждый автор может писать несколько книг (но должен писать хотя бы одну)», т.е. в базе данных не будет информации об авторах, которые не пишут ни одной книги, но допускается хранить информацию о книгах, которые еще никто не пишет.



«Каждую книгу может писать не более чем один автор; каждый автор может писать несколько книг либо не писать их вовсе», т.е. в базе данных допускается наличие авторов, не пишущих сейчас ни одной книги, и наличие книг, которые еще никто не пишет.

Аналогично анализируются и фиксируются все варианты связи один-ко-многим сущностей КНИГА - АВТОР (с учетом обязательности или необязательности участия в связи всех экземпляров этих сущностей).



Например, в этой диаграмме отражено правило: «каждую книгу пишут несколько авторов, каждый автор должен писать только одну книгу»; обязательность включения экземпляров говорит о том, что в базе данных не будет информации об авторах, не пишущих ни одной книги, а также информации о книгах, которые пока никто не пишет.

Связь многие-ко -многим:

Связь многие-ко-многим подразумевает, что один экземпляр сущности, расположенной в левой части диаграммы, связан с несколькими экземплярами сущности, расположенной в правой части диаграммы, и один экземпляр сущности, расположенной в правой части диаграммы, связан с несколькими экземплярами сущности, расположенной в левой части диаграммы.

По-прежнему, если все экземпляры сущности должны участвовать в связи, то участие является обязательным, и изображается на ER-диаграмме кружком, помещенным в блок, изображающий сущность. Если не все экземпляры сущности должны участвовать в связи, то участие является необязательным, и кружок на ER-диаграмме располагается вне блока сущности.

Следующая диаграмма отражает связь многие-ко-многим сущностей АВТОР – КНИГА, где экземпляры обеих сущностей вступают в обязательную связь:

Таким образом, в этой диаграмме отражено правило: «каждую книгу пишут несколько авторов, каждый автор пишет несколько книг»; обязательность включения экземпляров говорит о том, что в базе данных не будет информации об авторах, не пишущих ни одной книги, а также информации о книгах, которые пока никто не пишет.

В этой диаграмме отражено правило: «каждую книгу пишут несколько авторов, каждый автор может писать несколько книг либо не писать их вовсе», т.е. в базе данных допускается наличие авторов, не пишущих ни одной книги, но нет информации о книгах, которые пока никто не пишет.

Для сущностей АВТОР - КНИГА возможны еще два типа связи многие-ко-многим, отражающих два оставшихся варианта обязательности включения экземпляров:

«Каждую книгу может писать несколько авторов; каждый автор может писать несколько книг (но должен писать хотя бы одну)», т.е. в базе данных не будет информации об авторах, которые не пишут ни одной книги, но допускается хранить информацию о книгах, которые еще никто не пишет.

«Каждую книгу может писать несколько авторов; каждый автор может писать несколько книг либо не писать их вовсе», т.е. в базе данных допускается наличие авторов, не пишущих сейчас ни одной книги, и наличие книг, которые еще никто не пишет.

ГЛАВА II

СОЗДАНИЕ БАЗ ДАННЫХ В СИСТЕМЕ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ MS ACCESS

2.1. Основные возможности Access 2013

База данных (далее БД) представляет собой взаимосвязанные данные, которые хранятся в одном файле и используются для одной конкретной области (предмет, организация или прикладная задача).

Система управления базами данных (далее СУБД) – специализированные пакеты программного обеспечения, которые отвечают за ответы на пользовательские запросы БД и отчеты, а также поиск, хранение, изменение, пополнение и ввод данных.

У СУБД также имеется возможность использовать данные другими программными средствами.

Одна из самых популярных систем управления базами данных для персональных компьютеров является продукт Microsoft Access. Информация в данной БД имеет вид одной или нескольких таблиц. В программное обеспечение входит ввод и отбор данных по каким-либо критериям, изменение их структуры и вывод данных или результат решения задач.

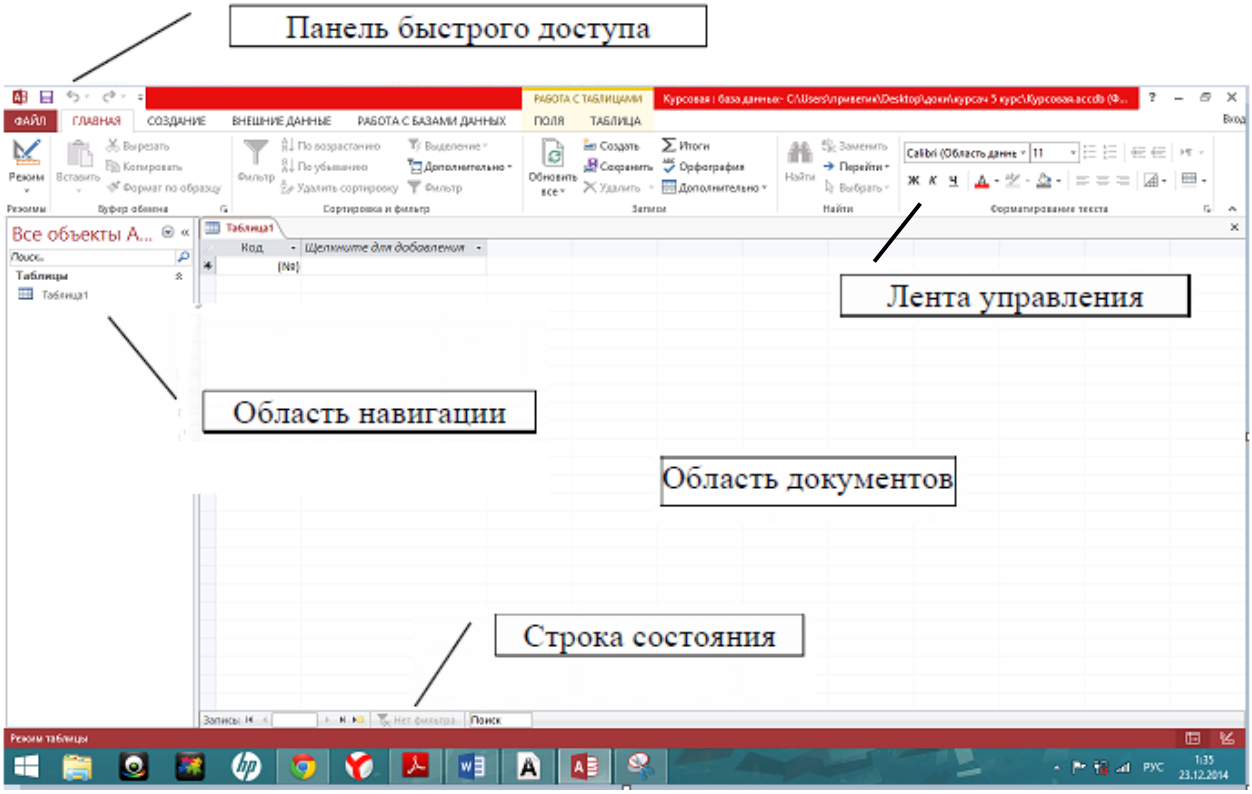
Также Microsoft Access устанавливает ссылки на данные и связывает объекты. В этом случае под данными имеется в виду правила Microsoft Access, которые автоматически изменяются все связанные объекты при изменении одного объект.

1. Создания новой базы данных**: Microsoft Access → Пустая база данных → Имя файла → Папка → Создать.**

Интерфейс Microsoft Access

Пользовательский интерфейс представляет собой Ленту, находящиеся в верхней части окна приложения. На ней находятся вкладки: Файл, Главная, Создание, Внешние данные, Работа с базами данных. Каждая вкладка связана с определенным видом действия.

Панель быстрого доступа расположена в верхней левой части окна Access на ней находится четыре кнопки управления.



Различные режимы работы с активным объектом отображаются в строке состояния

2.2. Создание таблиц

Таблица — это объект БД, в котором находятся данные по определенной теме, например, о работниках или зарплате. Таблица состоит из записей и полей.

В режиме конструктора таблиц создадим таблицу ГРУППА. Начнем с определения ее структуры. Основные параметры таблицы ГРУППА, представлены в табл. 1.1.

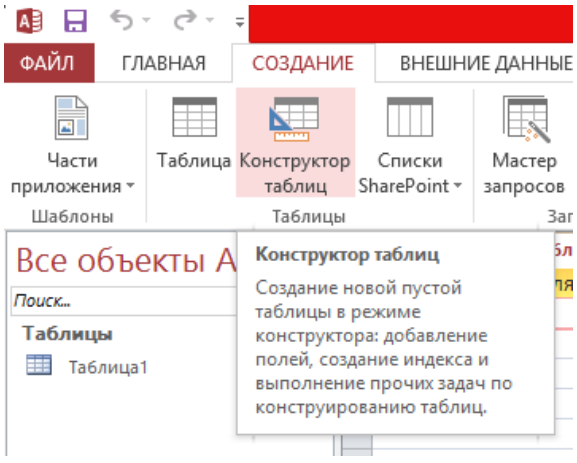
Таблица 1.1. Основные параметры структуры таблицы ГРУППА

Имя поля	Ключевое поле	Уникальное поле	Обязательное поле	Тип данных	Размер
НГ	да	да	да	Короткий текст	3
КОЛ			нет	числовой	байт
ПБАЛЛ			нет	числовой	Одинарное с плавающей точкой

Продолжение **таблицы 1.1**

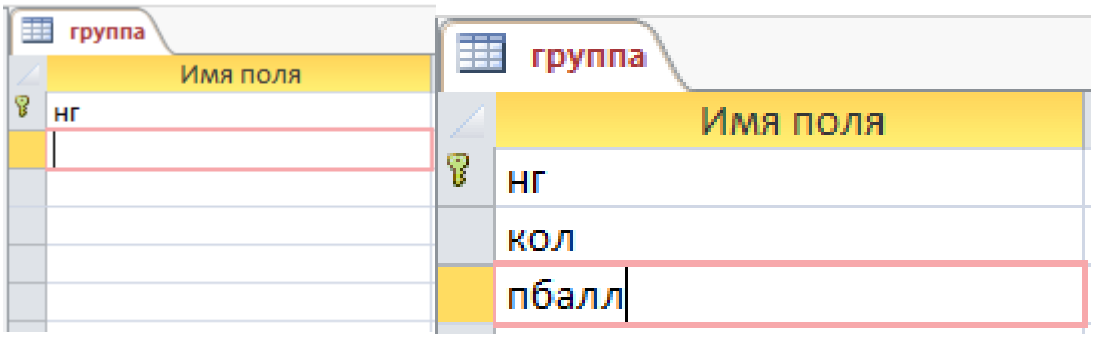
Имя поля	Число десятичных знаков	Подпись поля	Правило проверки	Сообщение об ошибке
НГ		Номер группы		
КОЛ		Количество студентов в группе	>=0 And <=35	Количество студентов больше допустимого
ПБАЛЛ	2	Проходной бал	>2 And <5 Or 0	Ошибка в оценке

Для перехода в режим *конструктора таблиц* перейдем во вкладку *создание* на ленте базы дынных.

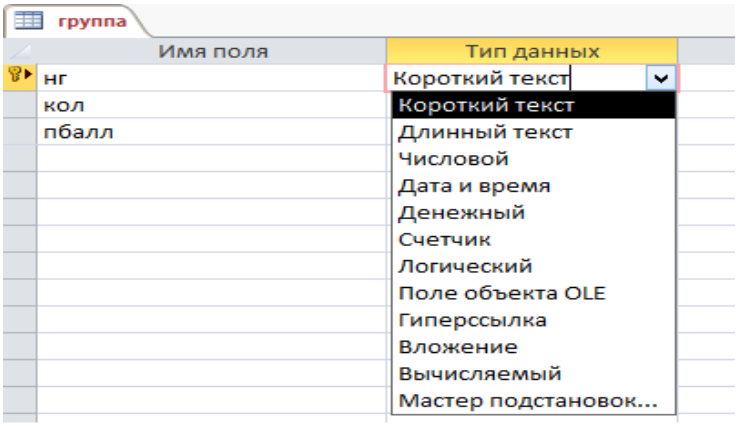


Для того чтобы задать параметры структуры таблицы ГРУППА необходимо:

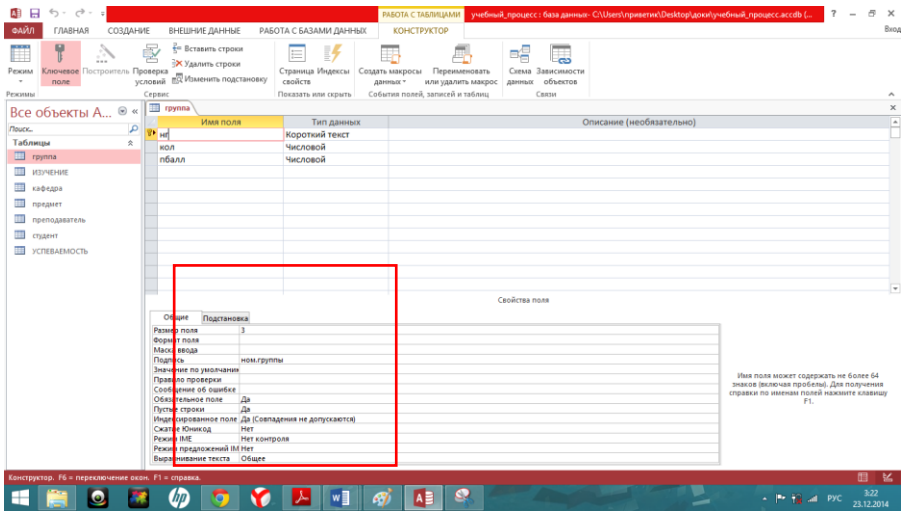
1) Ввести имена полей НГ, КОЛ, ПБАЛЛ в столбец *Имя поля*.



2) В следующем столбце *Тип данных* выбрать нужный тип, указанный в таб. 1.1 для всех полей.

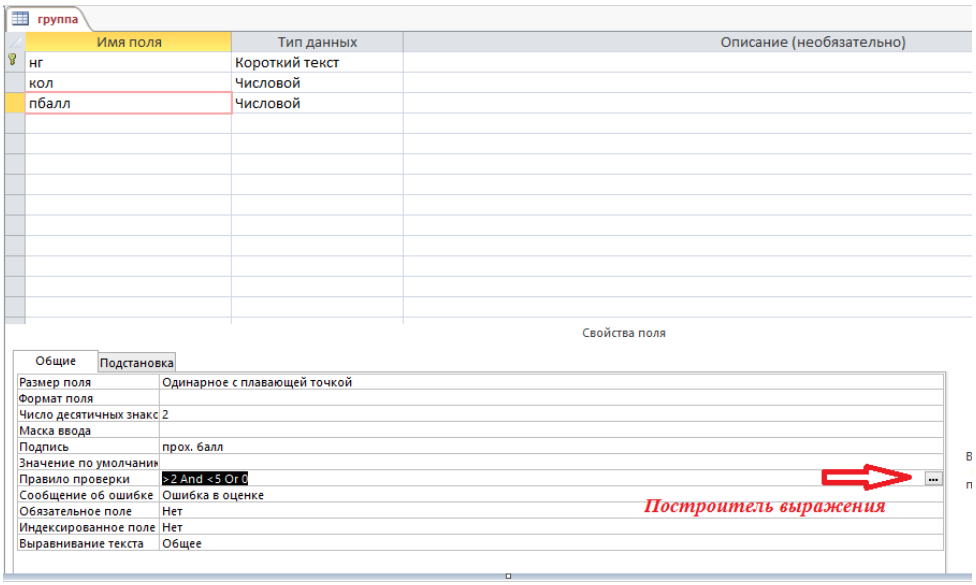


3) В нижней вкладке *Общие* указать свойства полей, приведенные в таб. 1.1



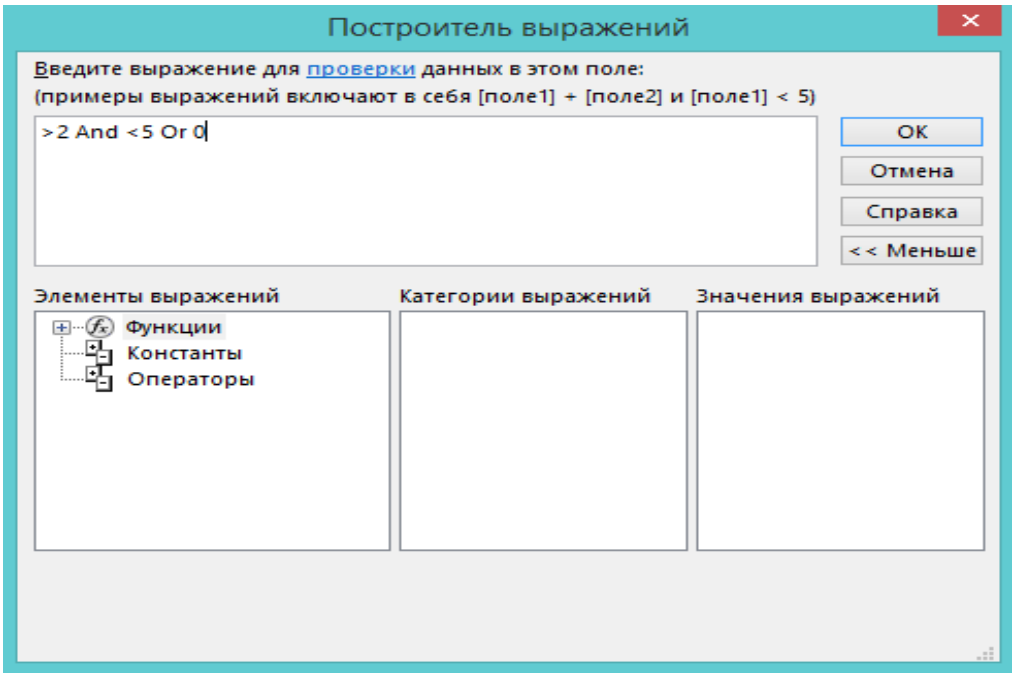
В строку *правило проверки*, вводится выражение, которое дано в таб. 1.1, так же оно может быть построено с помощью *построителя выражений*.

Построитель выражений вызывается с помощью кнопки расположенной справа от строки *Правило проверки*.



Внимание

При вводе операторов «больше равно» и «меньше равно» в программе имеются определенные символы (>= и <=). При вводе выражения не должны присутствовать пробелы. Построитель выражений сам выведет нужную ему форму.



После того как выражения было введено в окно построителя необходимо нажать клавишу «Enter». При этом Access начнет выполнять синтаксический анализ данного выражения и выведет его в строке *условие на значение*.

Следующем шагом будет определение первичного ключа для таблицы. После выделения поля НГ, необходимо щелкнуть правой кнопкой мыши по маркированной области слева от имени поля, нажимаем кнопку *Ключевое поле* во вкладке конструктор. Слева от имени поля появится изображение в виде ключа, это означает что ключевое поле было установлено. Далее определяем свойство ключевого поля.

В конце проделанной работы нужно сохранить таблицу и присвоить имя новой таблице – “ГРУППА”. Для этого выполняем команду *Файл* → *Сохранить* в появившемся диалоговом окне вводим имя.

Упражнение

Создать базу данных «Учебный процесс» состоящая из таблиц КАФЕДРА, ПРЕДМЕТ, ПРЕПОДАВАТЕЛЬ, СТУДЕНТ. Параметры структуры необходимые для создания таблиц расположены в приложении 1.

Познакомимся поближе с некоторыми особенностями создания структуры таблиц в базе данных «Учебный процесс».

Использование данных типа Поле объекта OLE (OLE Object)

Тип данных *поле объекта OLE* необходим, если в таблице нужно расположить данные в графическом формате (.bmp), например, фото работника какой-либо организации. Для таких случаев используется тип данных как поле объекта OLE.

Размещение этого объекта в поле производится на этапе заполнения полей таблицы. Объект может быть внедренным или связанным.

Замечание

OLE (Object Linking and Embedding – связывание и внедрение объекта) – это метод передачи информации между приложениями Windows в виде объектов. Тип данных поле объекта OLE позволяет устанавливать связь с объектами других приложении или внедрять объекты в базу данных. К ним

относятся: простые и форматированные тексты, графические объекты, графики и диаграммы, аудио дорожки (.WAV, .MIDI), анимация(.FLI, .MMM), видео дорожки (.AVI), электронные таблицы из других приложений, поддерживающих это средство. Access, поддерживая OLE, полностью взаимосвязан с другими продуктами Microsoft Office.

Объект, внедренный из другого приложения, как и все другие данные сохраняется в базы данных и всегда доступен для редактирования. Для этого необходимо щелкнуть два раза мышью по ячейке в которой находиться объект. Режим редактирования будет использовать программу, в которой был создан объект. Редактировать и просматривать объект можно во время работы с базой данных.

Использование данных типа Поле MEMO

Тип данных *поле MENO* необходим, если поле будет содержать текстовые данные большой длины. Данные можно ввести в ручную непосредственно в таблице, либо в область вызванную с помощью нажатия клавиш «Shift+F2».

Если же текс был подготовлен заранее, например, в Microsoft Word, то удобнее будет с помощью типа данных *поле объекта OLE* внедрить файл в таблицу.

Использование данных типа Гиперссылка (Hyperlink)

Тип данных Гиперссылка позволяет работать с гиперссылками в полях таблиц базы данных. Такие поля предназначены для перехода к другой БД или внутри БД в которой вы находитесь. Так же можно переходить к документам, находящимся в приложениях Microsoft Office, которые расположены на локальных или сетевых дисках, а также к ресурсам в сети Internet. Гиперссылка открывается щелчком мыши приложением, в котором оно было создано.

Упражнение

Создать в базе данных «Учебный процесс» таблицы ИЗУЧЕНИЕ, УСПЕВАЕМОСТЬ с параметрами структуры, которые расположены в приложении 1.

Ввод данных в таблицы

В правой вкладке *Область навигации* двойным щелчком откроем таблицу КАФЕДРА, в соответствии с данными представленными в таблице 2, вписываем эти данные в строки нашей таблицы.

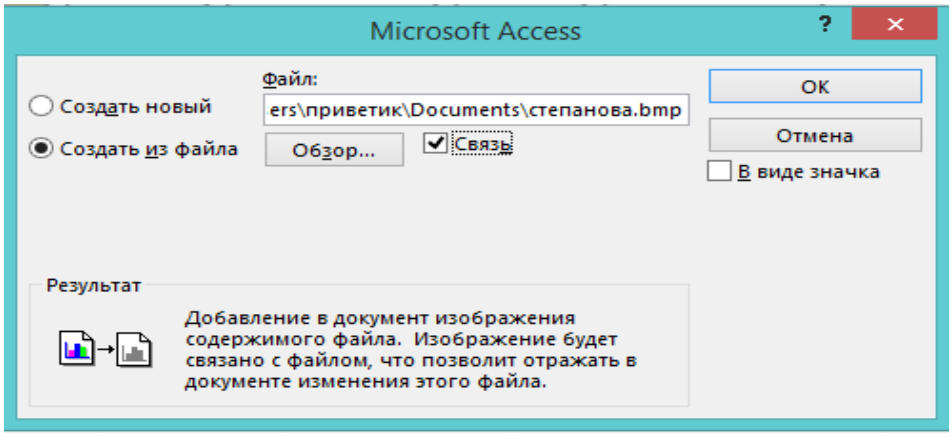
Таблица 2

Данные таблицы КАФЕДРА

К	**Название**	**ТЕЛ.**	**ФИО зав. кафедрой**
0	ИНФОРМАТИКА	310-47-14	Игнатьева В. В.
0	МАТЕМАТИКИ	310-47-15	Иванов И. И.
0	ИСТОРИИ	310-47-16	Смирнова И. В.
0	ИНОСТРАННОГ	310-47-17	Жданова А.Е.
0	ФИЗКУЛЬТУРЫ	310-47-67	Плетнев В.А.
0	ФИЛОСОФИИ	310-47-18	Бондаренко В.В.

Размещение объекта OLE

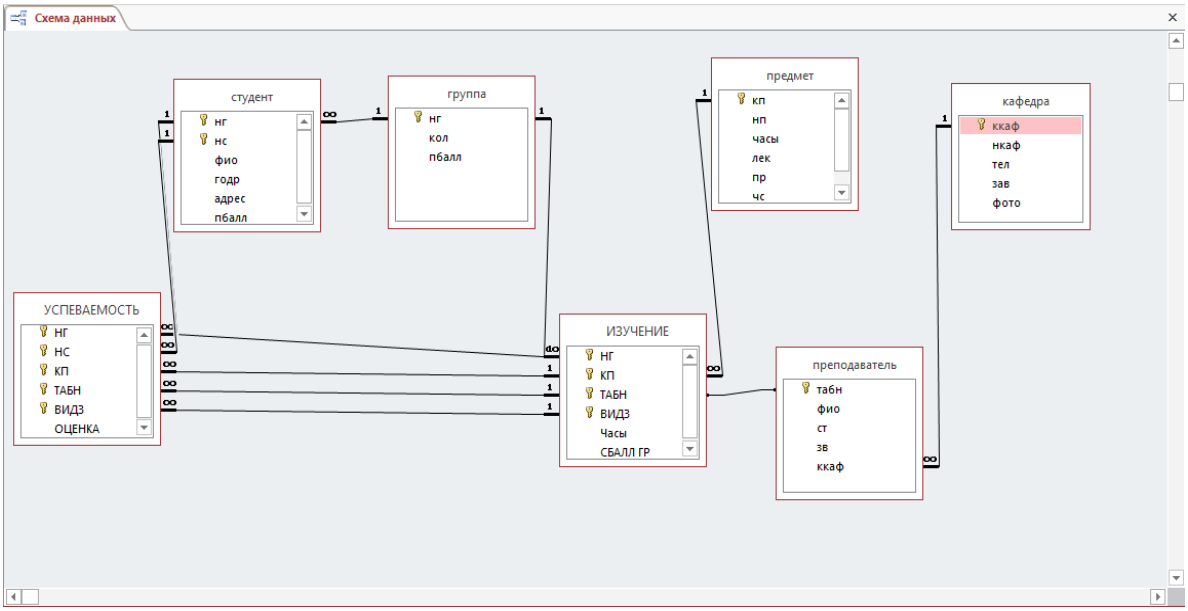
Для размещения объекта OLE нужно установить курсор в поле куда будет внедрен объект. Правой кнопкой мыши щелкаем по нужному полю в появившемся меню выбираем пункт *Вставка объекта*. Далее устанавливаем переключатель на *Создать из файла.* Так же в данном окне можно вписать путь к файлу вручную или воспользоваться кнопкой *обзор*, после нажатия которой появится диалоговое окно, позволяющее найти файл на компьютоpe.



Флажок *связь* позволяет ввести в поле связанный объект, что дает возможность отображать изменения, проделанные с файлом.

Включение таблиц в схему данных

Для этого необходимо перейти на ленте во вкладку *работа с базами данных*, в появившемся меню выберем *схема данных*. В открывшееся окно перетаскиваем из *области навигации* наши таблицы.



Определение связей между таблицами схемы данных

Определение связей по простому ключу. Установим связь между таблицами ГРУППА и СТУДЕНТ по простому ключу НГ.

Для этого в окне Схемы данных установим курсор мыши на ключевом поле НГ главной таблицы ГРУППА и перетащим его на поле НГ в подчиненной таблице СТУДЕНТ. В открывшемся окне Изменение связей в строке Тип отношения установится значение один-ко-многим. Отметим параметр Обеспечение целостности данных. Если таблица ГРУППА и СТУДЕНТ ранее были заполнены корректными данными, между таблицами будет установлена связь, обозначенная на схеме как 1:∞. Это свидетельствует

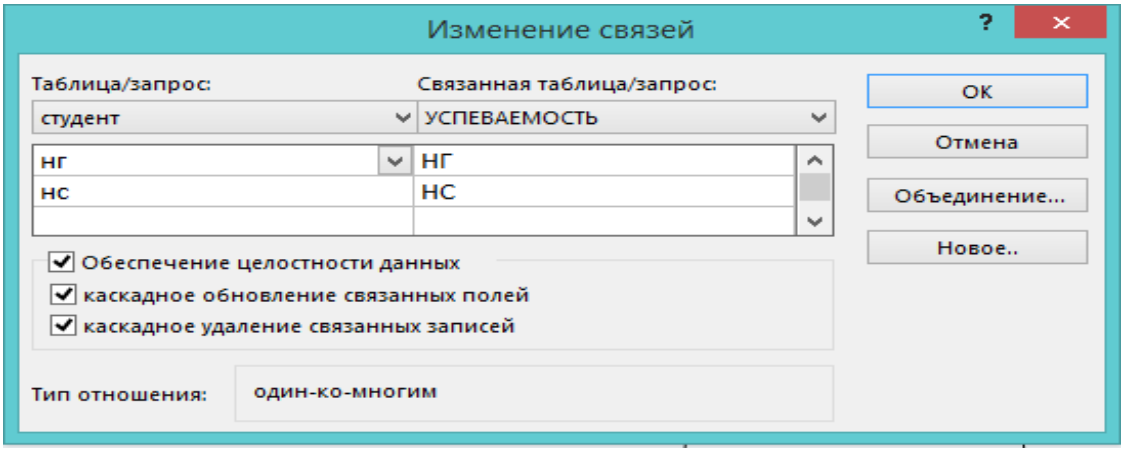
о регистрации связи типа 1:M с параметром поддержания целостности. В противном случае появится сообщение о невозможности установить этот тип отношения.

Для обеспечения автоматической корректировки данных во взаимосвязанных таблицах установим флажок каскадное обновление связанных полей и каскадное удаление связанных записей.

Аналогичные действия выполняются для других пар таблиц КАФЕДРА → ПРЕПОДАВАТЕЛЬ (ключ ККАФ), ПРЕДМЕТ → ИЗУЧЕНИЕ (ключ КН), ПРЕПОДАВАТЕЛЬ → ИЗУЧЕНИЕ (ключ ТАБН), ГРУППА → ИЗУЧЕНИЕ (ключ НГ).

Определение связей по составному ключу. Определим связи между таблицами СТУДЕНТ → УСПЕВАЕМОСТЬ, которые связаны по составному ключу НГ+НС. Для этого в главной таблице СТУДЕНТ выделим оба этих поля, удерживая клавишу Ctrl. Перетащим оба поля на поле НГ в подчиненной таблице УСПЕВАЕМОСТЬ.

В окне **Изменение связи** (рис. 3.3) для ключевого поля НС главной таблицы ТАБЛИЦА/ЗАПРОС выберем соответствующее поле подчиненной таблица СВЯЗАННАЯ ТАБЛИЦА/ЗАПРОС. В этом же окне установим режимы **Обеспечение целостности данных** и другие параметры связи.



Аналогично определяются связи между парой таблиц ИЗУЧЕНИЕ → УСПЕВАЕМОСТЬ (составной ключ связи – НГ+КП+ТАБН+ВИДЗ).

После определения связей таблицы могут перемещаться в пределах рабочего пространства окна схемы данных. Перемещения и изменения размеров таблиц осуществляются принятыми в Windows способами.

Проверка работоспособности схемы данных, поддержание целостности осуществляется при конструировании форм, запросов, отчетов и их использовании, а также при непосредственной корректировке таблиц.

2.3. Создание форм

Однотабличные формы

Форма построенная на основе таблицы бывает для самостоятельной загрузки, просмотра и корректирования таблиц, а также для вспомогательного включения в какую-либо составную форму.

Для просмотра, ввода и редактирования записей БД, форма должна быть предварительно спроектированной и далее сконструированной с помощью средств Access.

Для того чтобы создать форму можно воспользоваться мастерами Access. Но для точного формирования макетов форм с требованиями пользователя необходимо воспользоваться *конструктором форм*.

Конструирование формы

Для того чтобы сконструировать форму воспользуемся конструктором форм. В начале конструирования определим таблицу базы данных на основе которой будет создаваться форма, далее выбираем поля в таблице, которые будут представлены в форме, размещаем их на макете формы, после создаем графические элементы, которые включают в себя кнопки, элементы оформления, поясняющие тексты, выключатели и рисунки. Настройка этих элементов производится с помощью их свойств.

Области и элементы формы в режиме конструктора

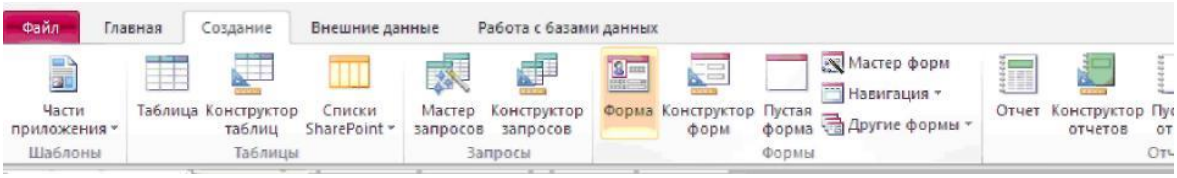
Форма в режиме Конструктора форм имеет три области: *область данных, заголовок формы*, и *примечание формы*, которые могут быть образованы по команде меню **Вид|Заголовок|примечание формы**. Области формы заполняются множеством различных графических объектов.

Графические объекты, которые связывают записи таблиц и предназначаются для того чтобы отображать данные некоторых полей, называются *элементами управления*. Основные типы элементов управления: *поле, поле со списком, список*. Тип элемента управления, выбираемый для поля *по умолчанию*, определяется в свойствах поля таблицы базы данных, с которым связано поле формы. Задается это свойство при определенных типах данных поля в режиме конструктора таблиц на вкладке *подстановка*.

Графические объекты, которые не связаны с запросами и таблицами, предназначаются в первую очередь для создания макета формы и содержат *надписи* полей.

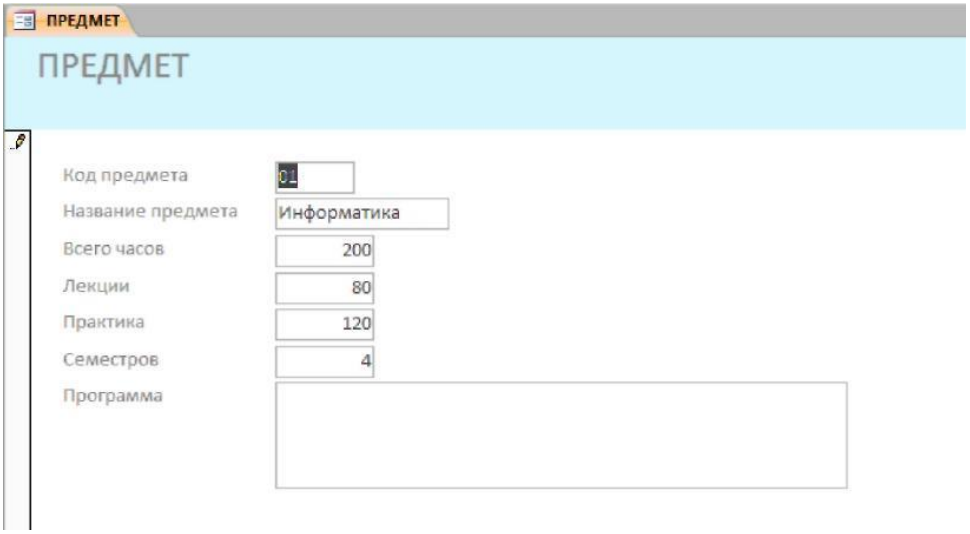
Создание однотабличной формы с помощью инструмента «Форма»

В *области навигации* выберем запрос или таблицу, которые хотим отобразить в форме. На ленте во вкладке *создать* нажмем на кнопку *форма*.



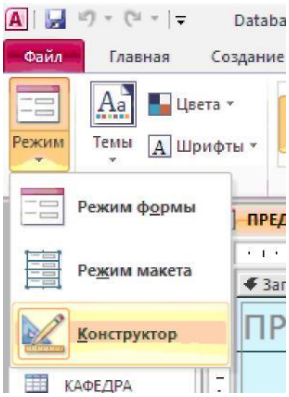
Создастся новая форма, отображенная в макетном режиме. При одновременном отображении данных в режиме макета можно вносить изменения в структуру макета.

Завершением работы команды *форма* является отображение самой формы. *Режим формы* выводит панель инструментов, кнопки которой аналогичны по значению кнопкам *режима таблицы*.

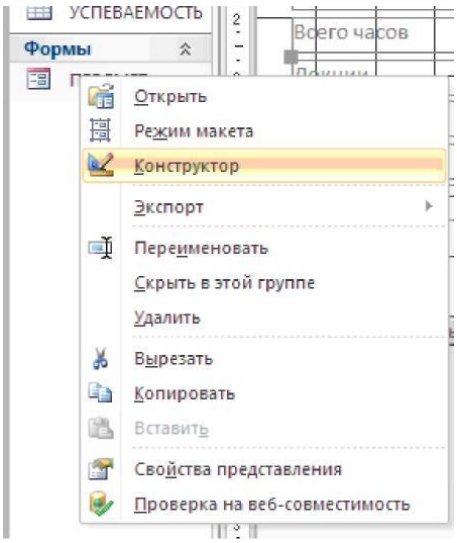


Редактирование формы

Переход в режим *конструктора форм* необходим для уточнения параметров отображения элементов формы таких как надписи, местоположение, размер, шрифт. Переход при открытой форме в режим конструктора осуществляется путем нажатия на кнопку *режим* в вкладке *режимы*, после чего появится список в котором можно выбрать нужный режим.



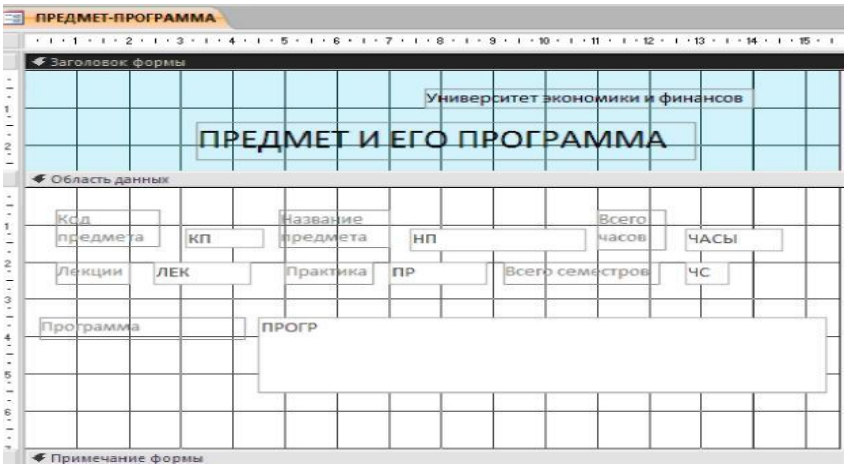
Так же переход в режим конструктора осуществляется также с помощью *контекстного меню* данной формы.



После выбора режима конструктора в окне Access появляются панель *конструктор форм* и *панель элементов*. Панель форматирования *формат (форма/ отчет)* может быть вызвана при активном окне формы по команде меню *Вид|Панель инструментов|Формат (Форма/Отчет).*

Создание заголовка

Для создания заголовка полученной формы в *режиме конструктор* нужно расширить область заголовка формы для этого необходимо установить курсор на границу области данных и перетащить ее на необходимое расстояние.



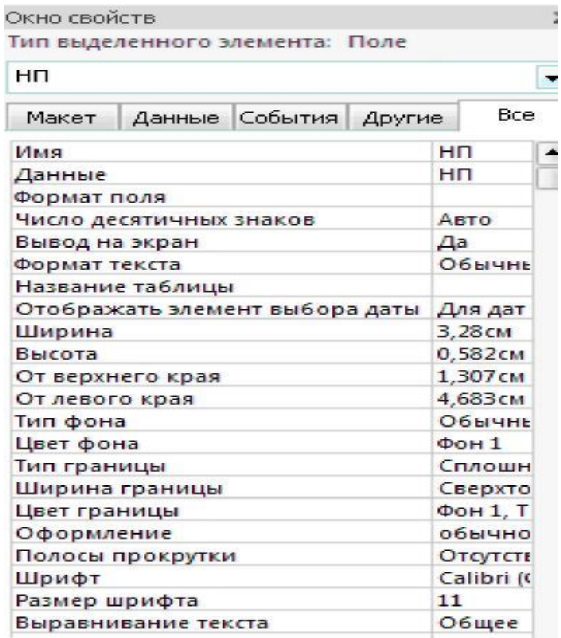
Для того чтобы ввести текст заголовка создадим графический элемент *надпись*. На панели инструментов щелкнем мышью по элементу *надпись*. Выберем на панели форматирования нужные нам параметры оформления, например, размер шрифта. Далее растягиваем рамку элемента *надпись* до

нужного нам размера, вводим наш заголовок и завершаем работу с элементом *надпись* нажатием на клавишу «Enter» или щелчком мыши вне рамки элемента.

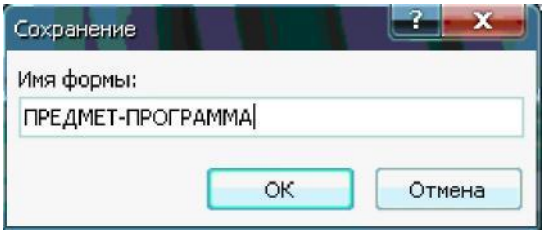
Так же, как и с другими элементами корректирование элемента *надпись* может выполнятся в любой момент. Это выполняется с помощью щелчка мыши по элементу для изменения его размеров, либо по тексту внутри рамки для его форматирования. Удаление происходит с помощью выделения элемента и нажатия клавиши «Del».

Изменение надписей и отображения значений полей. При редактировании связанных элементов *поле* и *надпись*, если между ними установлена связь, или аналогичной пары элементов, полученной с помощью кнопки *список полей* на панели конструктора форм, следует иметь в виду, что независимое перемещение поля и его надписи возможно, только если курсор примет вид указательного пальца. В противном случае оба элемента перемещаются синхронно. Остальные действия по внесению изменений в эти элементы осуществляется аналогично рассмотренному при формировании элемента в области заголовка.

Изменение свойств. Корректирование формы и ее элементов можно выполнить с помощью изменений их свойств. Для этого двойным щелчком открываем *окно свойств* нужного нам элемента.



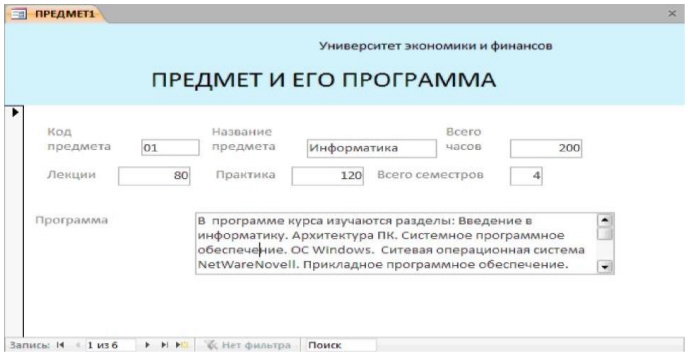
Сохранение формы после редактирования. После форматирования нужных нам элементов не забываем сохранить форму. Для этого выполним команду меню *Файл*|*Сохранить* или нажмем кнопку *сохранить* на панели быстрого доступа. Далее подтверждаем сохранение, и в появившемся диалоговом окне вводим название формы ПРЕДМЕТ-ПРОГРАММА в текстовом поле *имя формы*.



Работа с данными таблицы в режиме формы

Завершив редактирование формы, перейдем к работе с таблицей ПРЕДМЕТ через форму. Сначала выйдем из режима конструктор для этого во вкладке *режим* выберем *режим формы*.

Для работы с данными таблицы ПРЕДМЕТ в *области навигации* в группе *объекты* перейдем к строке *формы* и двойным щелчком мыши откроем сохраненную форму.



Вводимые в поля информация должна соответствовать типу данных указанных при создании структуры таблицы ПРЕДМЕТ.

Формы для загрузки двух таблиц

В данном разделе будет рассмотрена последовательность конкретных действий при создании составной формы для загрузки двух таблиц, связанных однозадачными или многозначными отношениями. Технология

разработки любой многотабличной формы включает *проектирование* макета формы (см. выше раздел *«Технология загрузки базы данных с использованием форм»)* и процесс *конструирования* средствами Access.

При проектировании составной формы выполним:

- Конструирование подсхемы данных для создаваемой формы
- Конструирование общей структуры экранной формы, т. е. ее макета в соответствии со структурой входного документа и подсхемой данных
- Конструирование размещения и состава реквизитов для каждой из частей создаваемой формы.

На основе результатов проектирования осуществим конструирование экранной формы средствами Access.

Проектирование форм на основе двух таблиц

Начнем с проектирования формы для добавления данных в таблицы СТУДЕНТ и ГРУППА, а также все возможного редактирования и просмотра этих данных. Источником такой формы является документ, содержащий «Список студентов группы». Из данного документа будут добавляться в базу данных одновременно две таблицы: СТУДЕНТ и ГРУППА, которые в совокупности будут образовывать объект загрузки.

Определение подсхемы данных для составной формы

Поскольку объект загрузки: ГРУППА → СТУДЕНТ не подчиняется в схеме данных другим таблицам, подсхема, необходимая для построения формы, не должна включать других таблиц.

Определение общей структуры составной формы

В соответствии с приведенной подсхемой определим общую структуру составной формы, которую назовем СПИСОК ГРУППЫ.

Для обеспечения удобного ввода данных из документа, в форме предусматриваем основную часть с реквизитами группы и подчиненную в которой находятся записи о студентах группы. Подчиненную форму назовем СПИСОК СТУДЕНТОВ.

Таким образом, составную форму СПИСОК ГРУППЫ определяют:

- *Тип формы* - многотабличная
- *Источник записей для основной части формы* - таблица ГРУППА
- *Включаемая подчиненная форма* - СПИСОК СТУДЕНТОВ

Подчиненную форму СПИСОК СТУДЕНТОВ определяют:

- *Тип формы* - подчиненная, многозаписевая
- *Источник записей* - таблица СТУДЕНТ

На подсхеме при создании форм показано назначение таблиц.

Размещение реквизитов основной и подчиненной формы

Размещение реквизитов в основной части формы и подчиненной форме должно соответствовать входному документу «Список студентов группы».

В основной части составной формы СПИСОК ГРУППЫ вверху размещаем реквизиты, которые соответствуют полям таблицы ГРУППА:

- Номер группы (НГ - ключ)
- Количество студентов (КОЛ)
- Средний проходной бал в группе (ПБАЛЛ)

В подчиненной форме СПИСОК СТУДЕНТОВ размещаем в роли заголовка столбцов многозаписевой формы названия реквизитов соответствующих полей таблицы СТУДЕНТ:

- Номер студента в группе (НС)
- Фамилия И. О. (ФИО)
- Год рождения (ГОДР)
- Адрес (АДРЕС)
- Средний балл при поступлении (ПБАЛЛ)

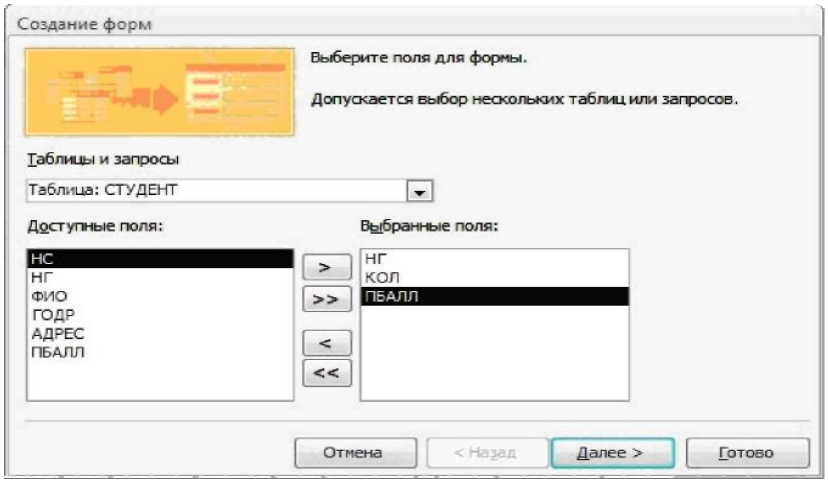
Заметим, что ключевое поле НГ не включено в подчиненную форму, так как поле связи НГ включено в основную часть формы.

Создание формы для двух таблиц с помощью мастера

Осуществим с помощью средств Access конструирование формы для одновременной корректировки и загрузки двух таблиц СТУДЕНТ и ГРУППА в базе данных «Учебный процесс».

Определение таблиц и полей для основной и включаемой частей формы

Выберем строку *формы* в группе *объекты* окна базы данных и нажмем кнопку *создать*.

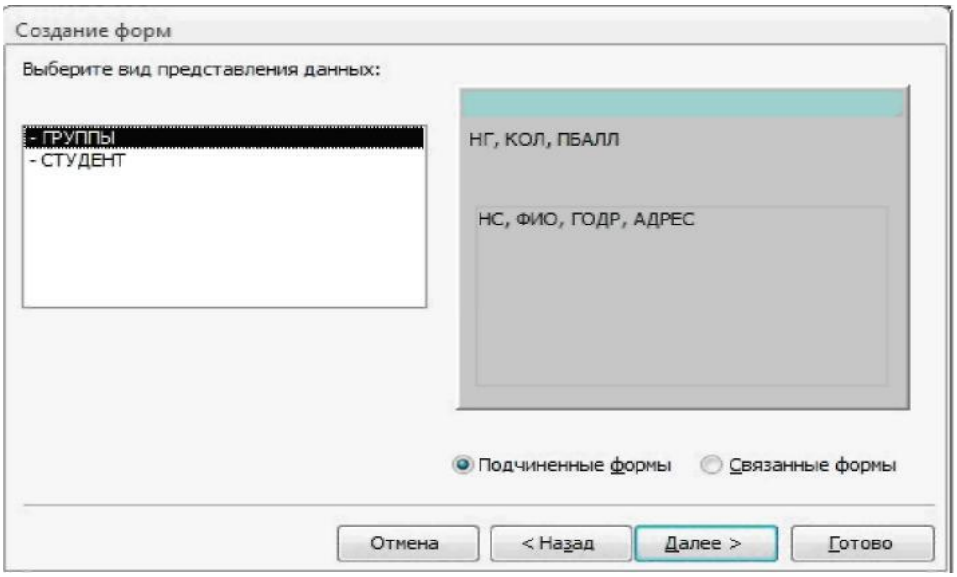


В появившемся диалоговом окне *новая форма* выберем режим *мастер форм* и таблицу ГРУППА, которая в данный момент служит источником данных для основной части, создаваемой *многотабличной формы*.

В окне *создание форм* в списке *таблицы/запросы* будет отражена ранее выбранная нами таблица ГРУППА. Далее выберем для нее из списка *доступные поля* только те поля, которые вошли в сконструированный макет формы, после перемещаем их в область *выбранные поля*. Выбираем таблицу СТУДЕНТ и ее поля. Данная таблица является источником записей подчиненной формы, связанных с записью, отображенной в основной части формы.

Выбор типа формы

В следующем окне *создание форм* отображается макет формы с перечнем полей в основной и подчиненной частях формы. В открытом окне уже будет выделена таблица ГРУППА, которая является источником записей основной части формы. Таблица СТУДЕНТ, являющееся источником записей подчиненной формы. Для непосредственного включения подчиненной формы выберем первый вариант – *подчиненные формы*.

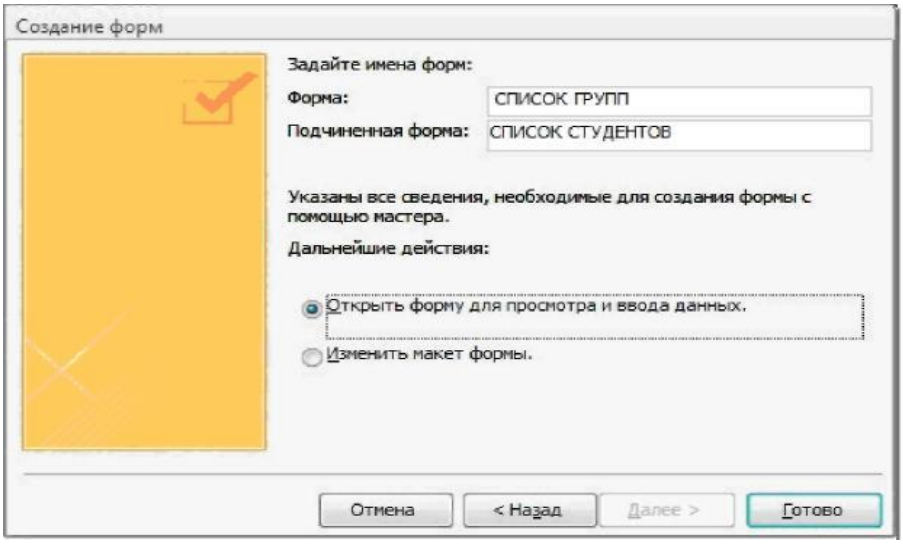


В следующем диалоговом окне мастера форм выберем внешний вид подчиненной формы *ленточный* для получения многозаписевой подчиненной формы и вывода в ней подписей полей. В очередном диалоговом окне выбираем стиль оформления *стандартный* с утопленными полями.

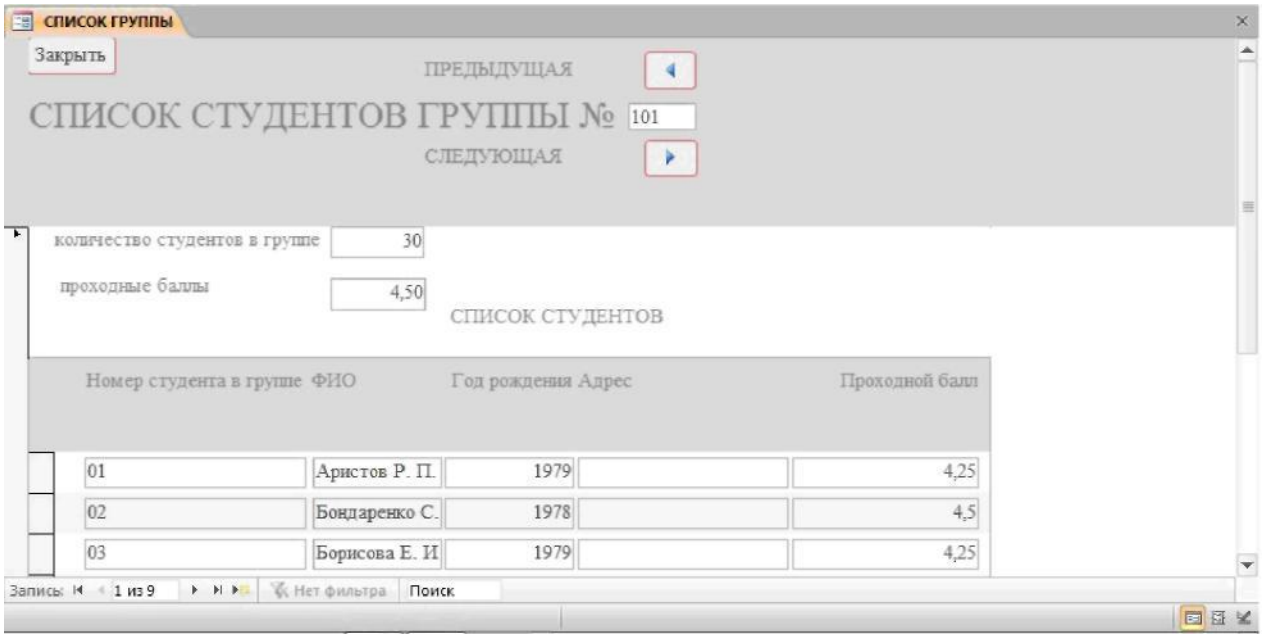
Присвоение имени форме и ее открытие

В последнем окна *создание форм* вводим имена составной формы – СПИСОК ГРУППЫ и подчиненной формы – СПИСОК СТУДЕНТОВ.

Выбираем дальнейшие действие мастера форм – *открытие формы для просмотра и ввода данных.*



После завершения работы мастера форм выводиться форма с данными из таблиц базы данных.

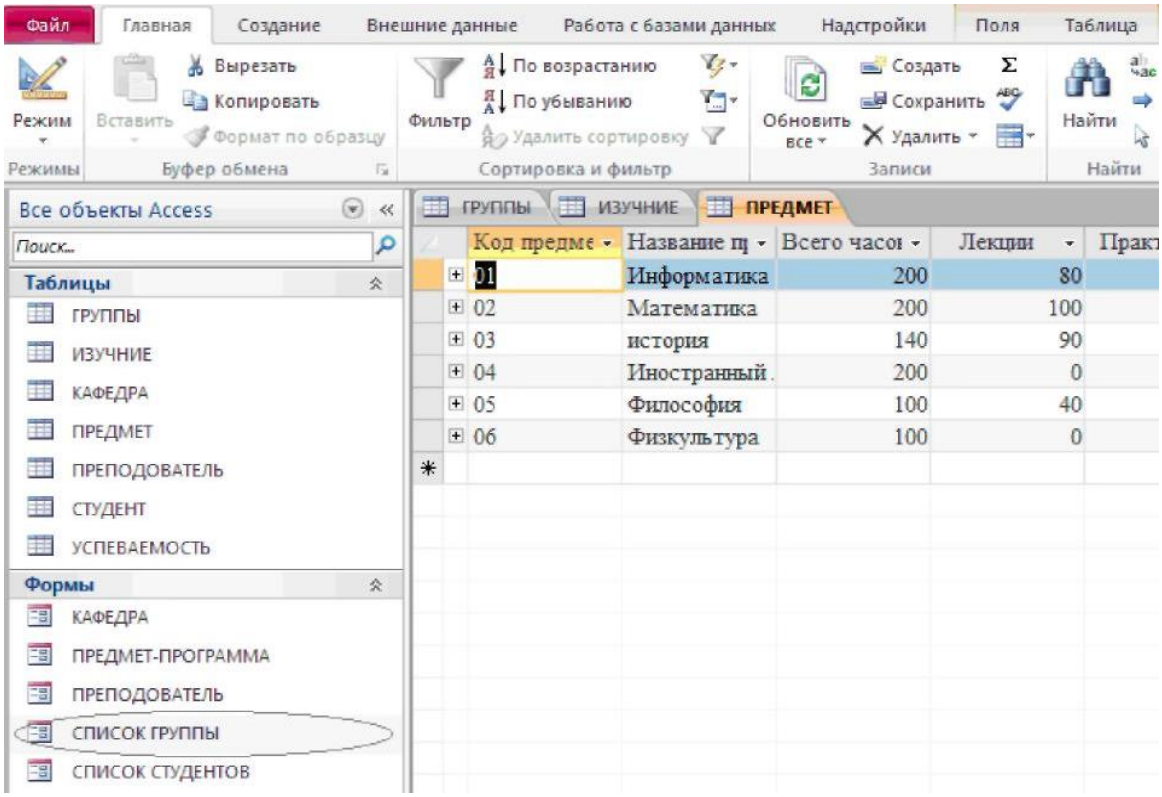


При этом в подчиненной форме выводятся те записи таблицы СТУДЕНТ, которые связаны с текущей записью таблицы ГРУППА, данные которой отображаются в основной части формы.

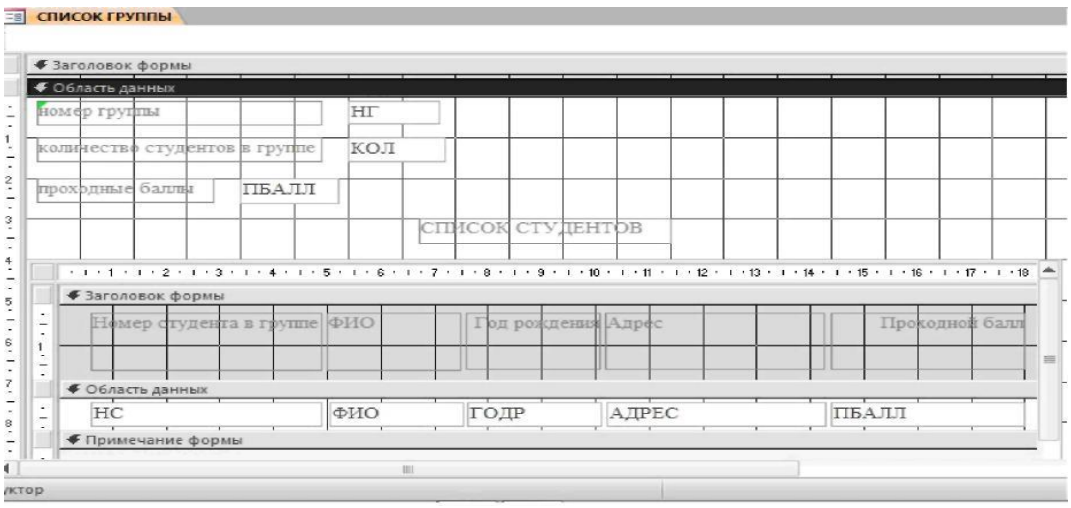
Редактирование формы в режиме конструктора

Изменение надписей и размещения полей

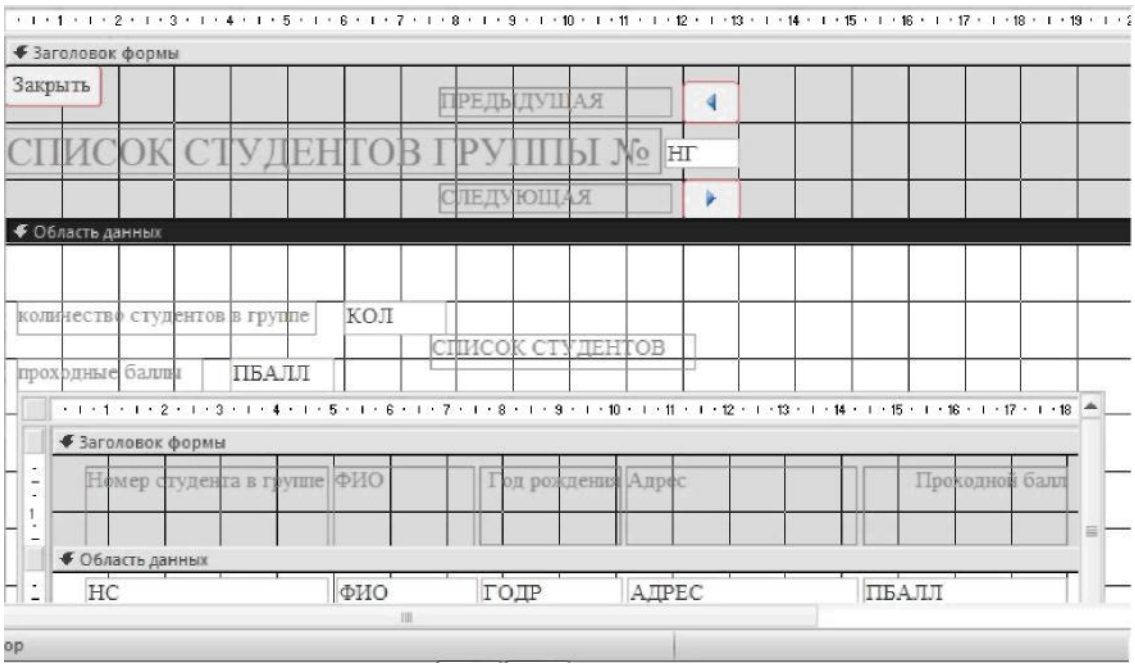
В окне *базы данных* в области *объекты* выделяем строку *формы*. Выбираем для форматирования созданную ранее многотабличную форму СПИСОК ГРУППЫ и жмем кнопку *конструктор*.



В форме открытой в режиме конструктора указано имя формы, СПИСОК ГРУППЫ, представлены поля с надписями основной части формы, расположенные в области данных, а также надписи и поля подчиненной формы СПИСОК СТУДЕНТОВ в рамке, созданной для нее мастером.



Доработаем форму СПИСОК ГРУППЫ, с помощью техники редактирования. Вводим полное название формы в область заголовка СПИСОК СТУДЕНТОВ ГРУППЫ №, которое соответствует макету документа и будет выводиться при распечатке формы.



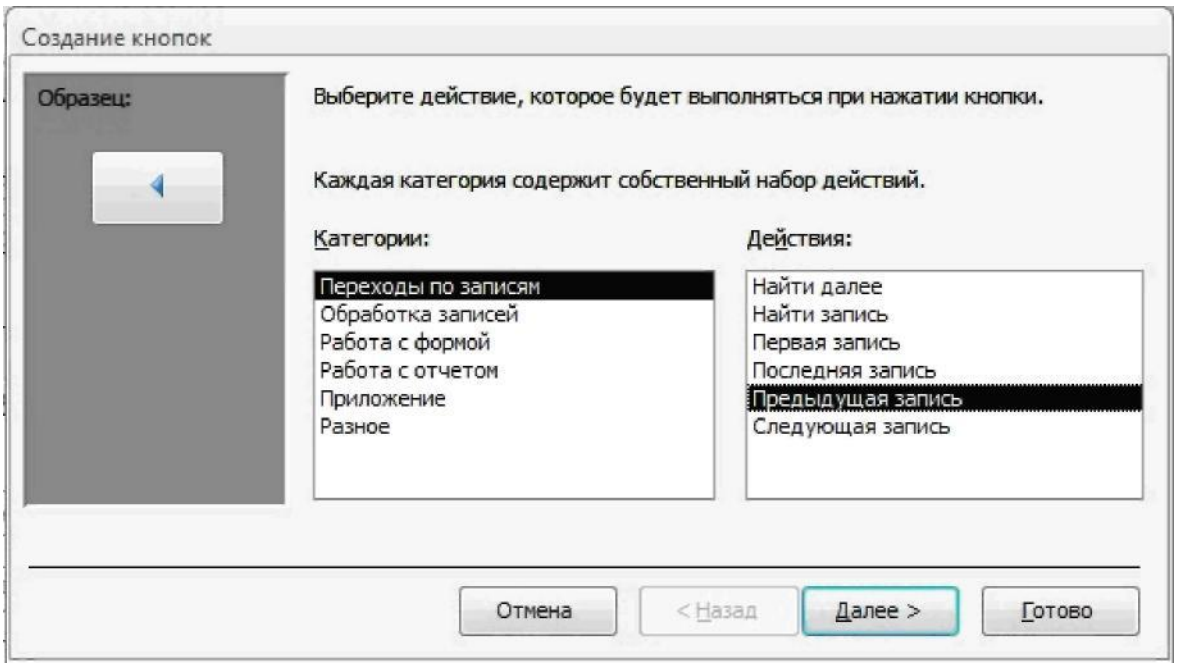
Размещаем поля и отмеченные элементы, уточняем параметры элементов как это было проделано в однотабличной форме.

Параметры подписи такие как ширина и высота можно задать, нажав кнопку *по размеру данных* на панели *конструктор форм*. Так же можно произвести выравнивание выделенных в форме элементов, например, по горизонтали с помощью команды меню *формат*|*выравнять*|*по нижнему краю*.

Создание кнопок для перехода к другой записи

Для перехода в форме от одной записи к другой, необходимо создать в основной части формы *кнопки управления*.

Для этого находим и жмем кнопку *мастера элементов* на панели инструментов Access, после воспользуемся инструментом *кнопка*. Перетаскиваем курсором в нужное место и нажимаем на нее. Далее появится окно *создание кнопок*.



В открывшемся окне выбираем действие, которое должно выполнятся при нажатии на кнопку. В группе *категории* выберем строку *переходы по записям*, в группе *действия* выберем строку *Предыдущая запись*. В следующем диалоговом окне нужно выбрать вид кнопки: *текст* или *рисунок* и выбрать его из списка. Отметим флажками *рисунок* и *показать все рисунки*. Далее выберем подходящий рисунок из списка, например, *стрелка вверх* (синяя). После нажатия кнопки *готово* кнопка с выбранным рисунком встраивается в форму. Аналогичные действия выполняются для встраивания кнопки перехода к последующей записи таблицы. При этом выбираются, соответственно, в группе *действия* – строку *следующая запись* и *рисунок стрелка вниз*. Отредактируем размер в надписи кнопок для перехода к записи другой группы, записав – «ПРЕДЫДУЩАЯ», «СЛЕДУЮЩАЯ».

Для создания кнопки закрытия формы в группе *категории* надо выбрать строку *работа с формой*, а в группе *действия – закрыть форму*. После формирования кнопки заменим название ее название на «ЗАКРЫТЬ».

Редактирование подчиненной формы

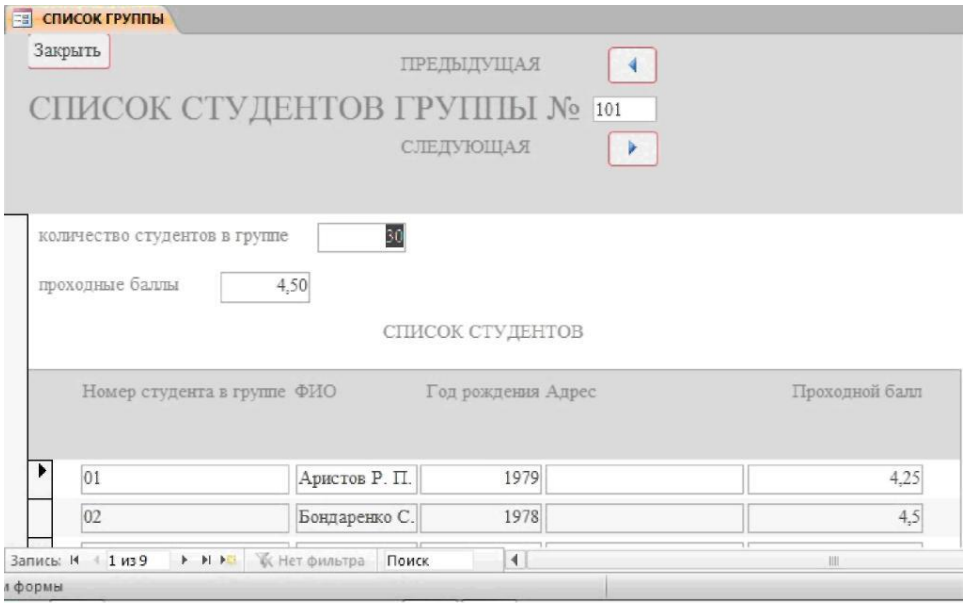
Аналогичные действия по доработке выполним для подчиненной формы СПИСОК СТУДЕНТОВ. Перейдем к редактированию подчиненной формы,

переводя курсор в область подчиненной формы или открывая подчиненную форму в окне базы данных.

Используя технику редактирования формы, удалим поле НГ, отображающее номер группы, т. к. это поле является полем связи и его достаточно сохранить в основной части формы. В подчиненной форме это поле имело бы одно и то же повторяющееся значение во всех строках. Уточним подписи полей-столбцов в заголовке формы, а также шрифт, размеры полей и подписей. После редактирования формы сохраним ее, нажав кнопку панели инструментов *сохранить*.

Переход в режим формы и загрузка таблиц

Завершив редактирование формы, перейдем из режима конструктора в *режим формы*, выбрав его на панели конструктора форм или выполнив команду меню *вид*|*режим формы*.



Если после работы форма была закрыта, то для возврата к ней нужно в *области навигации* перейти к строке формы. Открывается форма двойным щелчком мыши по названию.

Готовая форма одновременно загружает и работает с данными двух таблиц СТУДЕНТ и ГРУППА. Далее нам нужно загрузить оставшиеся данные из приложения 1.

Во время работы с формой сначала вводим значение полей группы.

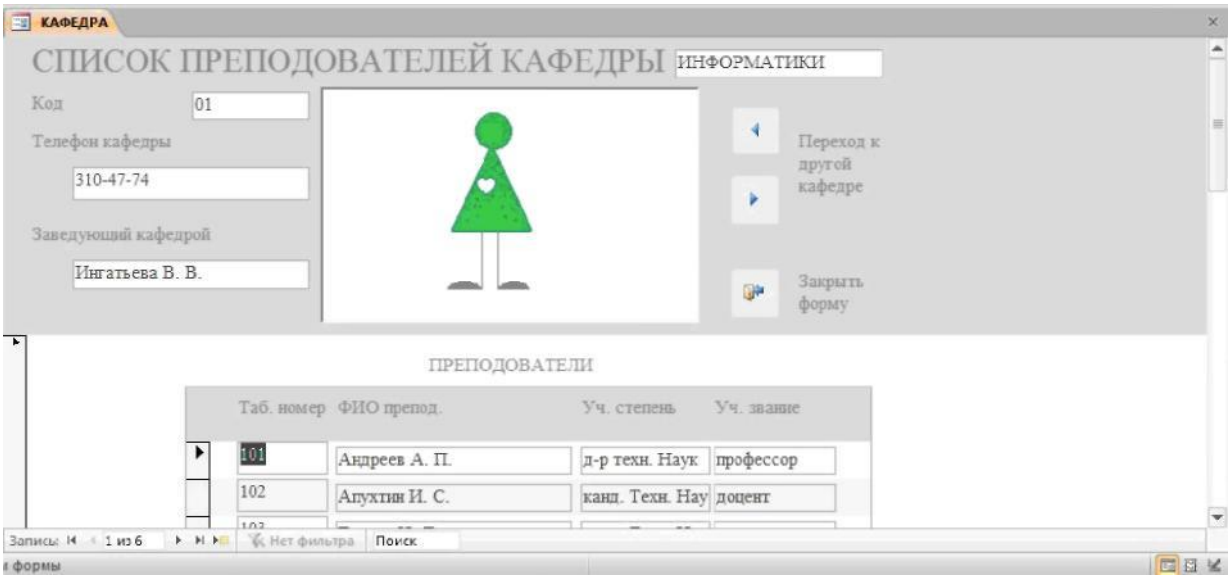
Замечание

Ключевые поля не должны оставаться пустыми, иначе вы не сможете создать запись.

Далее в подчиненную форму вводим реквизиты студентов. При этом в подчиненной форме так же имеется ключевое поле, как и в основной форме без его заполнения запись не будет сохранена. Для перехода к записи другой группы можно использовать созданные кнопки со стрелками вверх (вниз), для перемещения по записям студентов - стандартные кнопки перехода в *поле номера записи* в нижней части подчиненной формы. По окончании работы с формой воспользуемся кнопкой *закрыть* созданной на самой форме.

Упражнение

1. Создать форму аналогичным способом для загрузки данных таблиц ГРУППА и СТУДЕНТ.



2. Ознакомившись с технологией загрузки баз данных, рассмотренной в данной главе. Спроектировать формы для загрузки данных в таблицы ПРЕПОДАВАТЕЛЬ и КАФЕДРА. При этом расположив документы для более удобной записи.

3. Создать форму с помощью мастера форм и отформатировать в конструкторе форм.

Загрузите данные через форму СПИСОК ПРЕПОДОВАТЕЛЕЙ КАФЕДРЫ в таблицы КАФЕДРА И ПРЕПОДАВАТЕЛЬ, используя значения данных из Приложении 1.

Многотабличные формы

В соответствии с этапами загрузки базы данных "Учебный процесс", определенными выше, загрузка записей о занятиях текущего семестра в таблицу ИЗУЧЕНИЕ должна выполняться после загрузки таблиц со справочными данными, что обеспечит установление связей загружаемых записей с соответствующими записями этих таблиц.

На этапе проектирования определим все необходимые требования к создаваемой форме и ее макет. Процесс конструирования сложной формы средствами Access осуществим далее в соответствии с результатами этой работы.

При определении требований к форме рассмотрим особенности назначения и работы с формой, составим подсхему данных для создания формы, определим общую структуру формы и размещение реквизитов в соответствии со структурой входного документа и подсхемой данных[1].

Проектирование формы

Спроектируем формы, которые будут обеспечивать загрузку в таблицу ИЗУЧЕНИЕ данных о текущем семестре по каждой группе, и определим требования к форме, на основе которых можно перейти к ее созданию.

Документом вне машинной сферы, содержащим необходимые данные для загрузки таблицы ИЗУЧЕНИЕ, может служить "План проведения занятий в группе". Поэтому загружаемые через форму записи целесообразно

группировать в соответствии с их подчиненностью записям другой таблицы - ГРУППА.

В таблицу ИЗУЧЕНИЕ в соответствии с ее структурой наряду с номером группы и видом занятия нужно вводить идентификаторы предмета и преподавателя. В документе-источнике указаны наименование предмета и фамилия преподавателя. Для того чтобы при загрузке правильно вводить только идентификаторы, предусмотрим отображение в форме расшифровывающей информации: *наименования предмета* (НП) и *фамилии преподавателя* (ФИО) из таблиц ПРЕДМЕТ и ПРЕПОДАВАТЕЛЬ, которым подчинена загружаемая таблица.

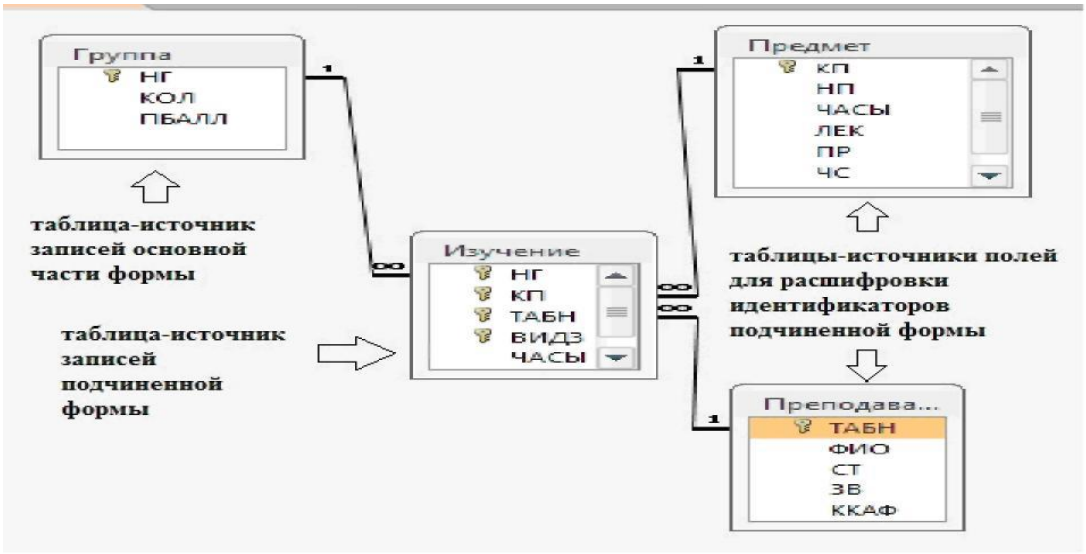
Поскольку форма служит не только для загрузки, но и для просмотра, включим в форму и другие описательные реквизиты из таблиц ПРЕДМЕТ и ПРЕПОДАВАТЕЛЬ для их отображения.

Определение подсхемы данных

В результате загрузки данных о занятиях в группах должны формироваться только записи таблицы ИЗУЧЕНИЕ, которую необходимо включить в подсхему данных для формы ввода-вывода данных о занятиях в группах.

В подсхему данных включим таблицу ГРУППА для того чтобы более удобно производить загрузку и просмотр данных по каждой группе в отдельности.

По этой причине выберем таблицу ГРУППА в качестве главного источника основной части составной формы. Обратим внимание на то, что данные взятые из таблицы ГРУППА должны отображаться на форме, но не как не вводится в нее. Предусмотрим отображение описательных данных о преподавателе и предмете, поэтому включаем в подсхему таблицы ПРЕПОДАВАТЕЛЬ и ПРЕДМЕТ.



Определение общей структуры формы

На основе полученной подсхемы определим общую структуру формы, которую назовем ПЛАН ЗАНЯТИЙ.

Основная часть формы.

Многотабличная форма ПЛАН ЗАНЯТИЙ будет содержать основную часть на основе таблицы ГРУППА для группировки вводимых данных о занятиях по каждой группе студентов. Для ввода данных в таблицу ИЗУЧЕНИЕ предусмотрим *непосредственное включение подчиненной формы* ИЗУЧЕНИЕ. Таким образом, форму ПЛАН ЗАНЯТИЙ определяют:

- *Тип формы* - многотабличная
- *Источник записей* для основной части формы - таблица ГРУППА
- Включаемая подчиненная форма ИЗУЧЕНИЕ с *источником записей* - таблица ИЗУЧЕНИЕ

Подчиненная форма, включаемая в основную

Для расшифровки идентификаторов предмета и преподавателя предусмотрим включение в подчиненную форму ИЗУЧЕНИЕ полей из таблиц ПРЕДМЕТ и ПРЕПОДАВАТЕЛЬ, являющихся главными относительно таблицы, на которой строится форма ИЗУЧЕНИЕ. Таким образом, подчиненную форму ИЗУЧЕНИЕ определяют:

- *Тип формы подчиненная*

• *Источник записей для основной части формы ИЗУЧЕНИЕ - таблица ИЗУЧЕНИЕ*

• *Источники отображаемых полей формы - таблицы ПРЕДМЕТ и ПРЕПОДАВАТЕЛЕЙ*

Размещение реквизитов формы

Основная часть формы

В основной части составной формы ПЛАН ЗАНЯТИЙ, вверху будем размещать поля таблицы ГРУППА:

• *НГ номер группы (уникальный ключ)*

• *КОЛ - количество студентов*

• *ПБАЛЛ - средний балл в группе при поступлении*

Доступ к перечисленным полям должен быть ограничен *только чтением,* т.к. значения этих полей не должны вводиться и корректироваться из документа «План занятий».

Подчиненная форма ИЗУЧЕНИЕ

В подчиненной форме ИЗУЧЕНИЕ разместим: Все поля загружаемой таблицы ИЗУЧЕНИЕ, кроме ключевого реквизита НГ (номер группы), включенного в основную часть формы, что обеспечивает однократное отображение одинаковых номеров группы по форме:

• *КП - код предмета*

• *ТАБН идентификатор преподавателя*

• *ВИДЗ - вид занятий*

• *ЧАСЫ - число часов занятий*

• *СБАЛЛ-ГР - средний балл по предмету в группе*

Поля из таблиц ПРЕДМЕТ и ПРЕПОДАВАТЕЛЬ, позволяющие отобразить справочную информацию о предмете и преподавателе, ведущем занятие: поля таблицы ПРЕДМЕТ:

• *НП - название предмета*

• *ЧАСЫ - всего часов*

• *ЛЕК - часов лекций*

- *ПР - часов практики*

Поля таблицы ПРЕПОДАВАТЕЛЬ:

- *ФИО - фамилия преподавателя*
- *СТ - ученая степень*
- *УЗ - ученое звание*

Разработка формы, обеспечивающей удобный интерфейс пользователя для загрузки подчиненной таблицы

Рассмотренный выше вариант построения формы для загрузки данных о занятиях недостаточно удобен для работы пользователя. В этом варианте данные о каждом занятии были представлены компактно в виде *одиночной записи,* но не обеспечивали удобный доступ к занятиям каждой группы. Создадим форму для более удобной работы пользователя, а также удобного просмотра и загрузки старых или новых занятий.

Требования к создаваемой форме.

Для начала обеспечиваем возможность отображения на экране всей информации об одном занятии и сохраним в то же время возможность объединения записей о занятиях по группам. Кроме того, целесообразно в форме в одну группу объединить поля, в которые вводятся значения при загрузке таблицы ИЗУЧЕНИЕ, а в другую -поля, которые содержат только справочную информацию о предмете и преподавателе, которая отображается для расшифровки идентификаторов занятия.

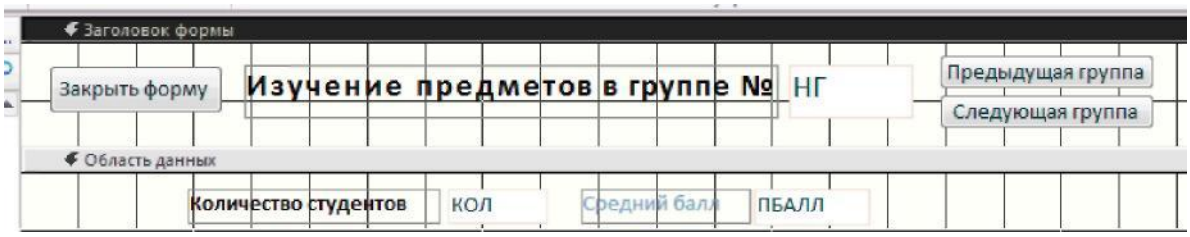
В процессе конструирования обеспечим защиту справочных данных в таблицах ГРУППА, ПРЕДМЕТ и ПРЕПОДАВАТЕЛЬ от случайных изменений при загрузке данных о занятиях в таблицу ИЗУЧЕНИЕ. Предусмотрим для удобства пользователя кнопки перехода к просмотру занятий для другой группы и кнопку закрытия формы. Для визуальной проверки правильности вводимых идентификаторов преподавателя и предмета используем поля со списком.

В соответствии с перечисленными требованиями для первоначального размещения полей и создания подчиненной формы можно воспользоваться

формой ПЛАН ЗАНЯТИЙ, полученной мастером. Откроем эту форму в режиме конструктора. Для этого в *границе области переходов (область навигации)* выберем для редактирования созданную ранее многотабличную форму ПЛАН ЗАНЯТИЙ. Если форма была открыта ранее в режиме просмотра или в режиме макета, то для перехода в режим конструктора достаточно выбрать соответствующий тип представления в группе *режимы* на вкладке *главная.*

Редактирование основной части формы.

В основной части формы разместим и отредактируем поля таблицы ГРУППА. Уточним текст подписей полей, шрифт и размеры полей и подписей, введем текст в заголовок формы. Удалим элемент с подписью подчиненной формы. Удалим разделительные линии между разделами формы: заголовком, областью данных и примечания. Для этого в свойствах формы на вкладке *макет* в строке *разделительные линии* выберем *нет*. Уберем область выделения записи, проставив в свойствах формы в соответствующей строке "Нет" Создадим две кнопки для перехода к следующей или предыдущей группе, а также кнопку для закрытия формы.



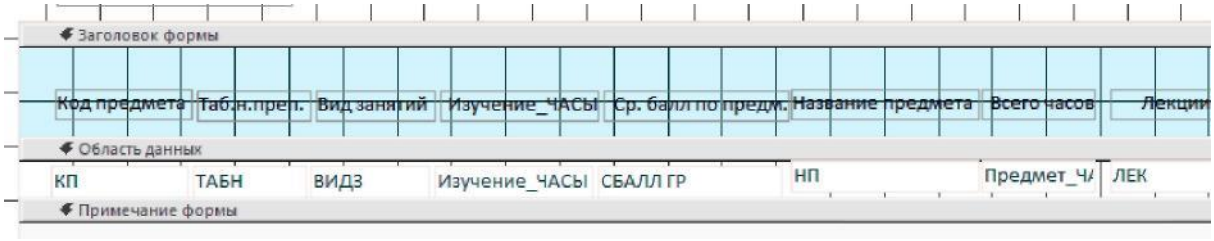
Ограничение доступа к полям таблицы-источника основной части формы.

Защитим данные записей таблицы ГРУППА от непроизвольных изменений при работе с формой, т.к. они должны использоваться только для отображения. Это все поля основной части формы. Для защиты поля выделим рамку поля и с помощью контекстно-зависимого меню вызовем свойства поля. В окне свойств на вкладке *данные* в строке *блокировка* выберем *да*. После установки этого свойства поле доступно только для чтения.

Сохраним форму под новым именем – «ПЛАН ЗАНЯТИЙ В ГРУППЕ», используя команду *сохранить как.*

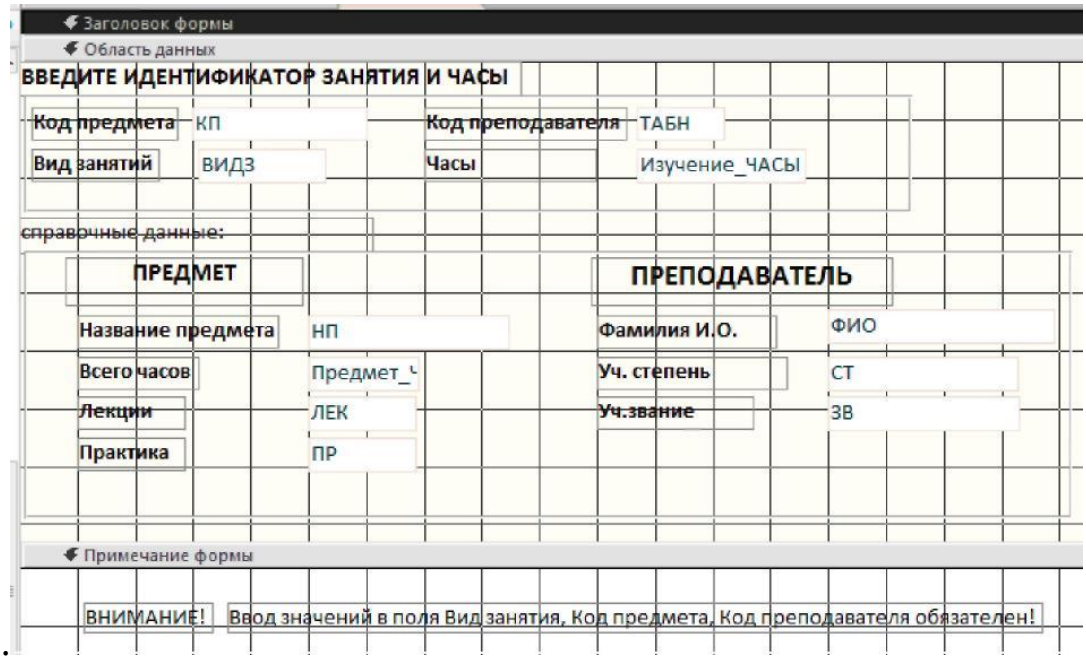
Редактирование подчиненной формы ИЗУЧЕНИЕ.

Ранее мастером было получена подчиненная ленточная форма.



Для изменения вида подчиненной формы вызовем ее свойства. На вкладке *макет* в окне *окно свойств* (прав.кнопка мыши - пункт «свойства формы») заменим в строке *режим по умолчанию* значение *ленточная форма* на *простая форма*. Это позволит отображать в подчиненной форме одну запись о занятии. Вид формы в конструкторе останется прежним.

В подчиненной форме разместим поля. После перемещения всех подписей полей из заголовка в область данных можно сократить его размер до нуля перемещением границы заголовка и области данных. Поля таблицы ИЗУЧЕНИЕ, в которой надо вводить данные из документа «План занятий», разместим в верхней части области данных. В нижней части области данных разместим поля, в которые будут автоматически выводиться справочные данные из таблиц ПРЕДМЕТ и ПРЕПОДАВАТЕЛЬ для расшифровки вводимых идентификаторов занятия. Эти поля служат только для ото-бражения сведений о предмете и преподавателе. Для создания рамок используем кнопку панели элементов управления *прямоугольник.*



Уточним текст подписей полей, шрифт и размеры полей и подписей. Выполним относительное выравнивание надписей и полей с помощью команды *главная|форматирование текста|выровнять.*

В область примечаний формы введем инструкцию пользователю, требующую обязательного ввода данных в поля, идентифицирующие занятие: код предмета — КП, номер преподавателя — ТАБН и вид занятия — ВИДЗ. Без этого не может быть создана запись в таблице ИЗУЧЕНИЕ.

Защита справочных данных от изменений.

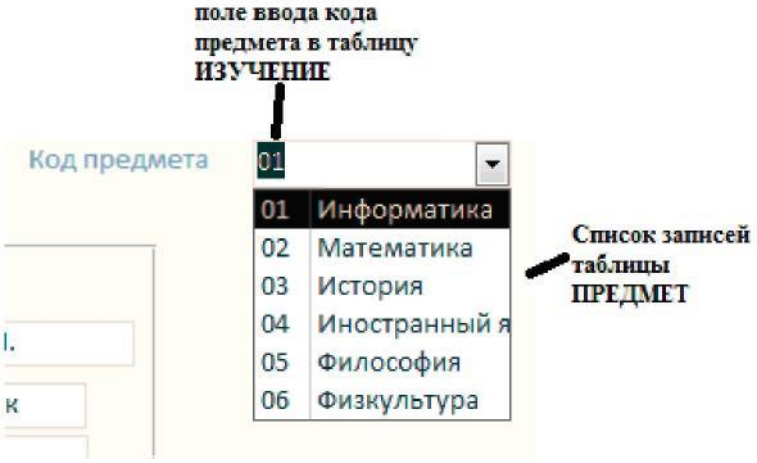
Защитим поля НП, ЧАСЫ, ЛЕК, таблицы ПРЕДМЕТ и поля ФИО, СТ, ЗВ таблицы ПРЕПОДАВАТЕЛЬ от случайных изменений при работе с формой. Для зашиты поля выделим рамку поля и с помощью контекстно-зависимого меню вызовем свойства поля. В окне свойств на вкладке *данные* в строке *блокировка* выберем *да*. После установки этого свойства поле доступно только для чтения.

Для визуального контроля правильности ввода идентификаторов занятия: КП и ТАБН можно использовать *поле со списком*. Процесс создания такого поля рассматривается ниже.

Создание полей со списком

При вводе идентификационных данных через форму в Access имеется возможность получить справочную информацию из ранее загруженных таблиц, что позволяет выбрать уже имеющиеся значения в базе и тем самым повысить достоверность вводимой информации. Отображение данных из справочных таблиц при вводе идентификатора свидетельствует о наличии в базе данных главных записей для загружаемой подчиненной записи, что необходимо для успешного завершения ввода при установленном параметре целостности в схеме данных.

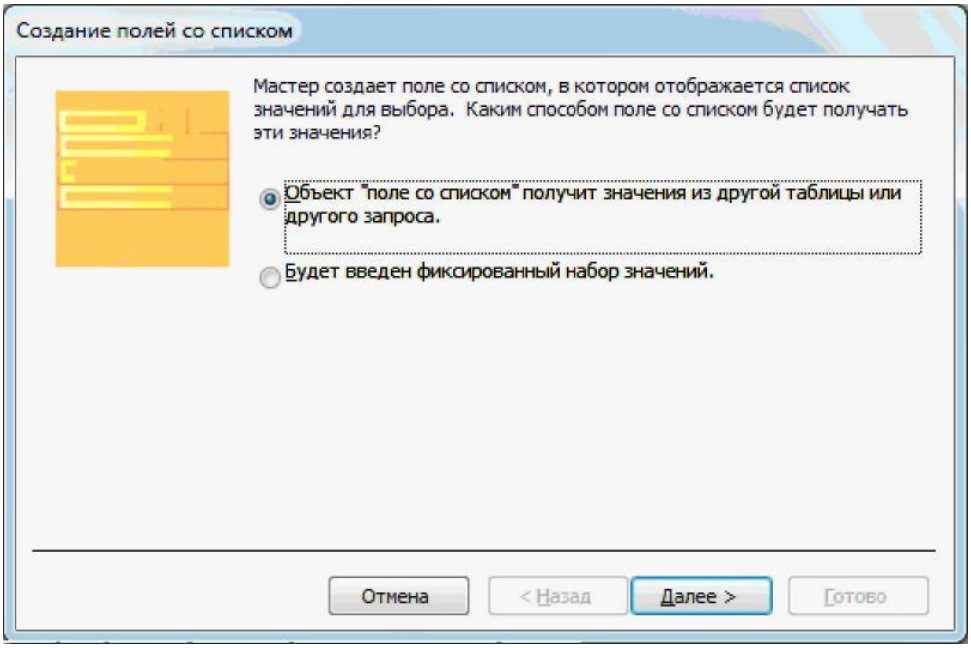
Поле со списком объединяет поле формы, в которое нужно ввести данные, и список. Список содержит записи из связанной главной таблицы. В списке можно выбрать из соответствующего поля нужное значение и ввести его в поле формы.



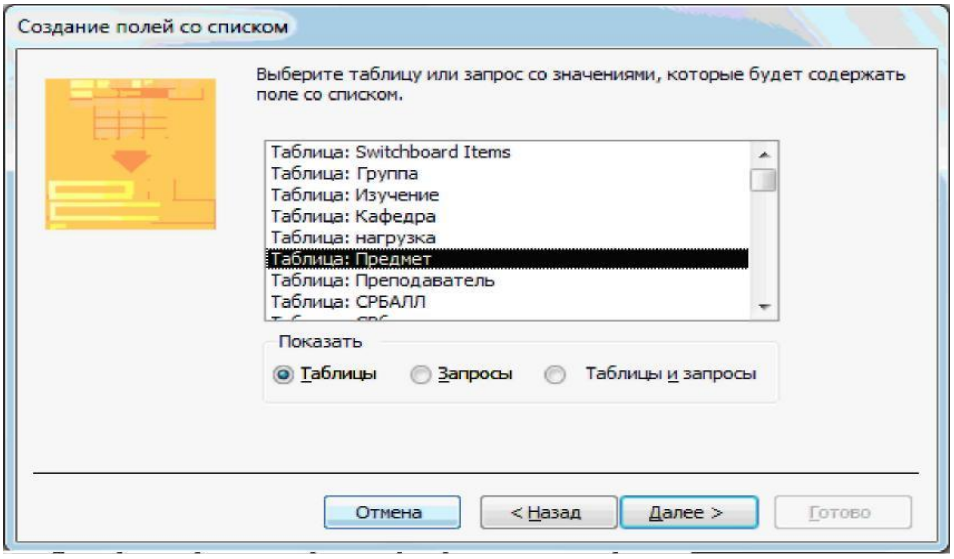
Создание поля со списком с помощью мастера

Создадим поле со списком для ввода значений кода предмета – КП в таблицу ИЗУЧЕНИЕ. Это позволит просматривать и вводить значения, которые уже имеются в главной таблице ПРЕДМЕТ, а также проверять соответствие кода и наименования предмета, имеющихся в документе-источнике загрузки «План занятий».

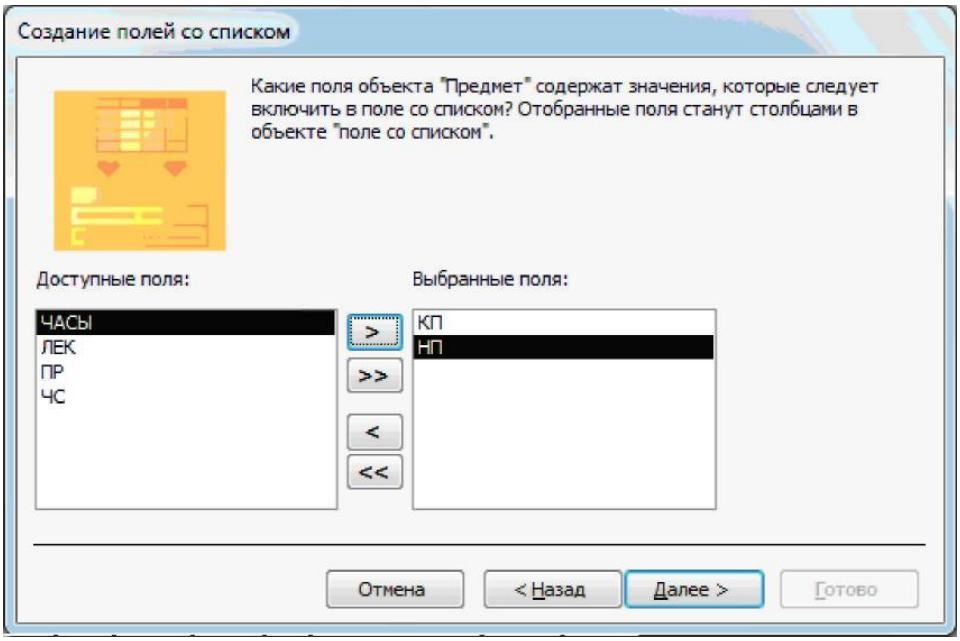
Выберем кнопку *поле со списком* в группе *элементы управления* на вкладке *конструктор,* установим курсор мыши в нужное место, нажмем кнопку мыши и, не отпуская ее, вычертим рамку элемента. После отпускания кнопки мыши запустится мастер и откроет диалоговое окно *создание полей со списком.* В этом окне определим способ, которым список поля получает свои значения. Для формирования списка из связанной таблицы выберем вариант *таблица или запрос содержат значения, которые использует поле со списком.*



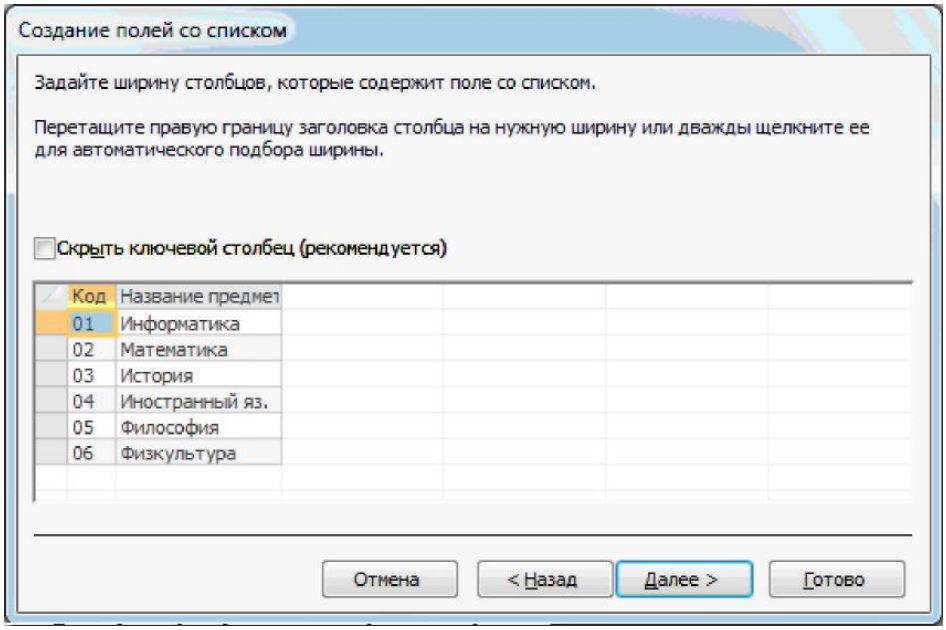
В следующем окне выберем таблицу ПРЕДМЕТ, которая будет поставлять значения в список поля.



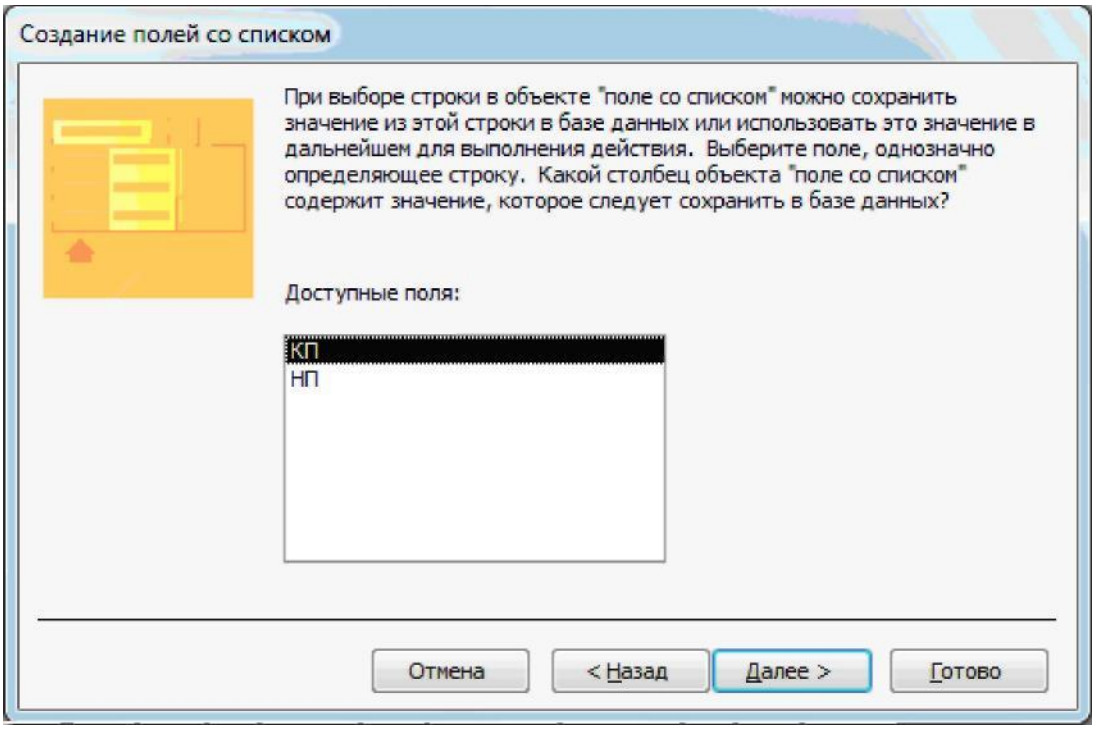
Затем выберем поле КП, а также поле НП для расшифровки кода КП. Эти поля образуют записи списка.



Далее в появившейся таблице определим ширину столбцов списка в соответствии с размером значений. Для этого курсор мыши установим на линию, разделяющую имена столбцов, и переместим ее в нужное место.

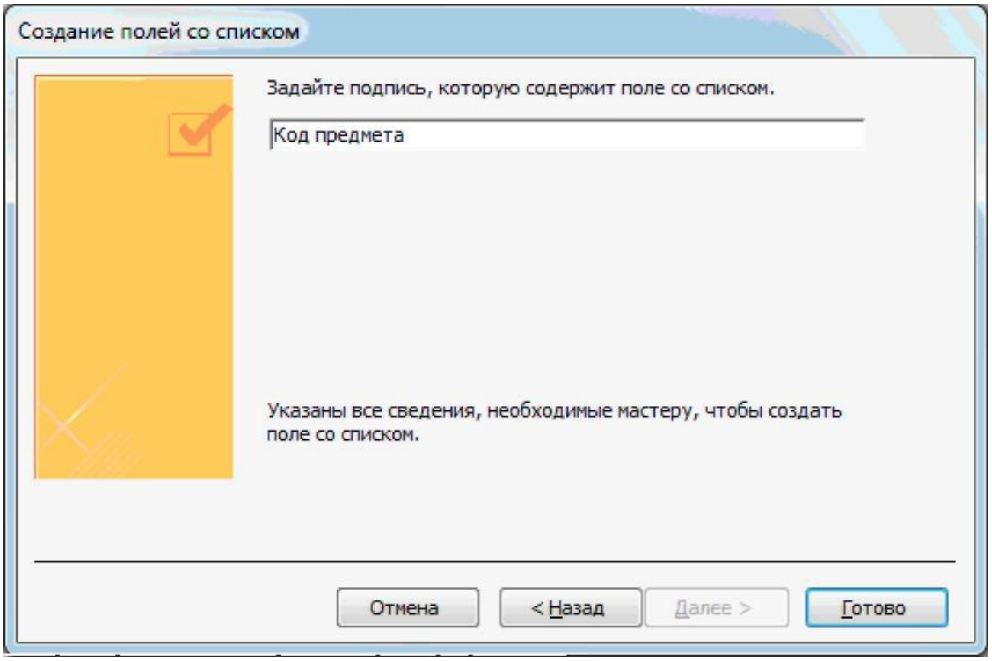


Далее выберем поле списка КП, являющееся ключом связанной таблицы ПРЕДМЕТ. Из этого поля будет выбираться значение для ввода в поле формы.



В следующем окне отметим переключатель **Сохранить в поле** и выберем поле формы КП (поле таблицы ИЗУЧЕНИЕ), в кото-рое будут вводиться значения из списка.

Далее введем подпись поля со списком – «Код предмета».



Нажмем кнопку *готово*. В результате получим поле КП со списком, которое содержится в окончательной форме.

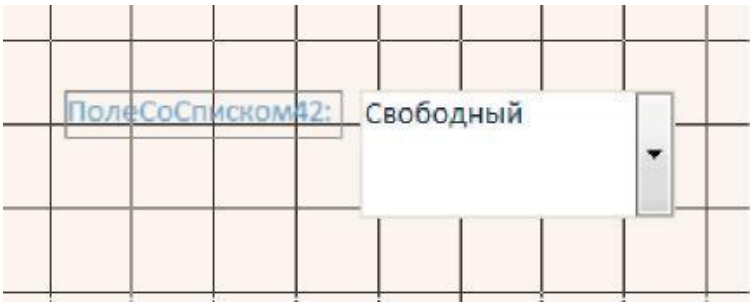
Использование поля со списком предметов возможно в режиме формы. Для удобства поиска нужного значения в списке можно воспользоваться операциями поиска и сортировки. Доступ к этим операциям возможен при помощи кнопок *найти* (группа *найти*), Сортировка по возрастанию, Сортировка по убыванию группы Сортировка и фильтр.

Создание поля со списком без использования мастера

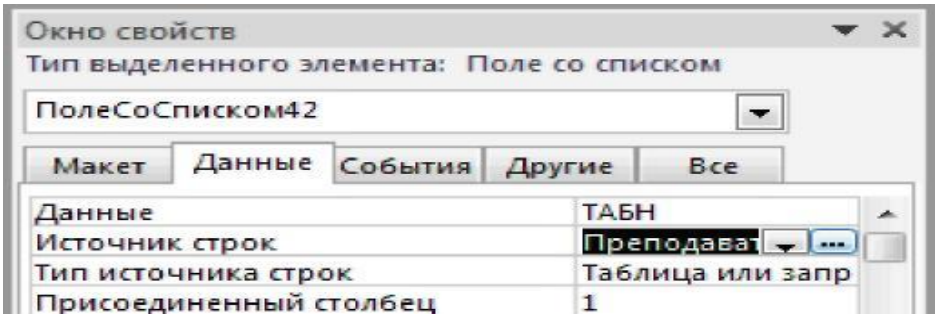
Создадим поле со списком для ввода значений идентификатора преподавателя ТАБН. Это позволит просматривать и вводить значения, которые уже имеются в главной таблице ПРЕПОДАВАТЕЛЬ, а также проверять соответствие номера и фамилии преподавателя, имеющихся в документе-источнике загрузки.

Нажмем на вкладке *конструктор* в группе Элементы управления кнопку Поле со списком.

Установим курсор мыши на появившийся элемент *свободный*.

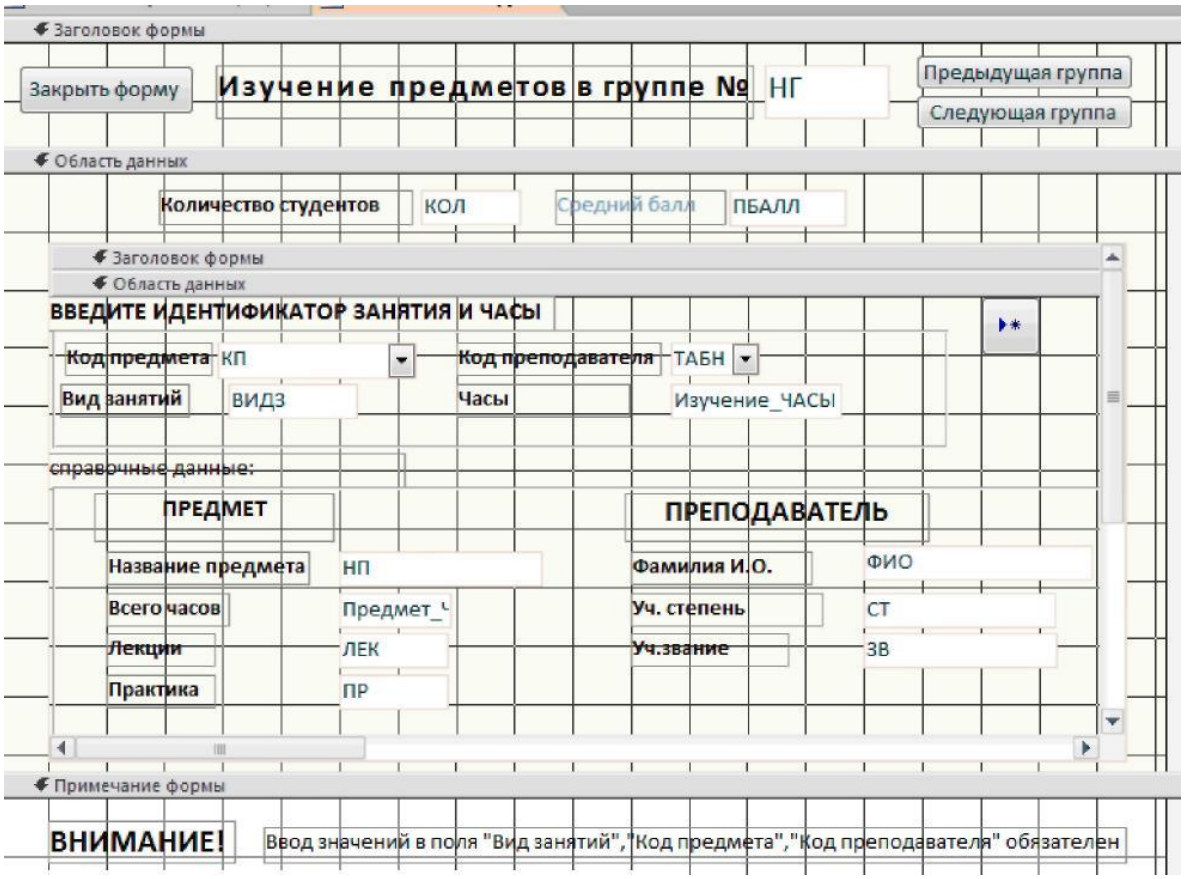


Нажмем правую кнопку мыши, чтобы вызвать контексно-зависимое меню. Выберем пункт *свойства*, затем - вкладку *данные*. В строке *данные* выберем поле ТАБН, которое необходимо заполнять через форму в таблице ИЗУЧЕНИЕ. В строке *тип источника строк* выберем элемент *таблица|запрос*, а в строке *источник строк* -таблицу ПРЕПОДАВАТЕЛЬ. Поля, включаемые в список, и их порядок определяются в построителе, который вызывается в этой же строке нажатием кнопки **-^J .**



Построитель выводит бланк запросов, в который перетащим из таблицы ПРЕПОДОВАТЕЛЬ поля ТАБН и ФИО.

Для того чтобы в списке выводились два поля, на вкладке *макет* необходимо в строке *число столбцов* указать «2». Для настройки ширины столбцов списка в строках *ширина списка* и *ширина столбцов* зададим подходящие значения. Закроем окно свойств *поле со списком*. В результате получим поле ТАБН со списком, которое содержится в окончательной форме.



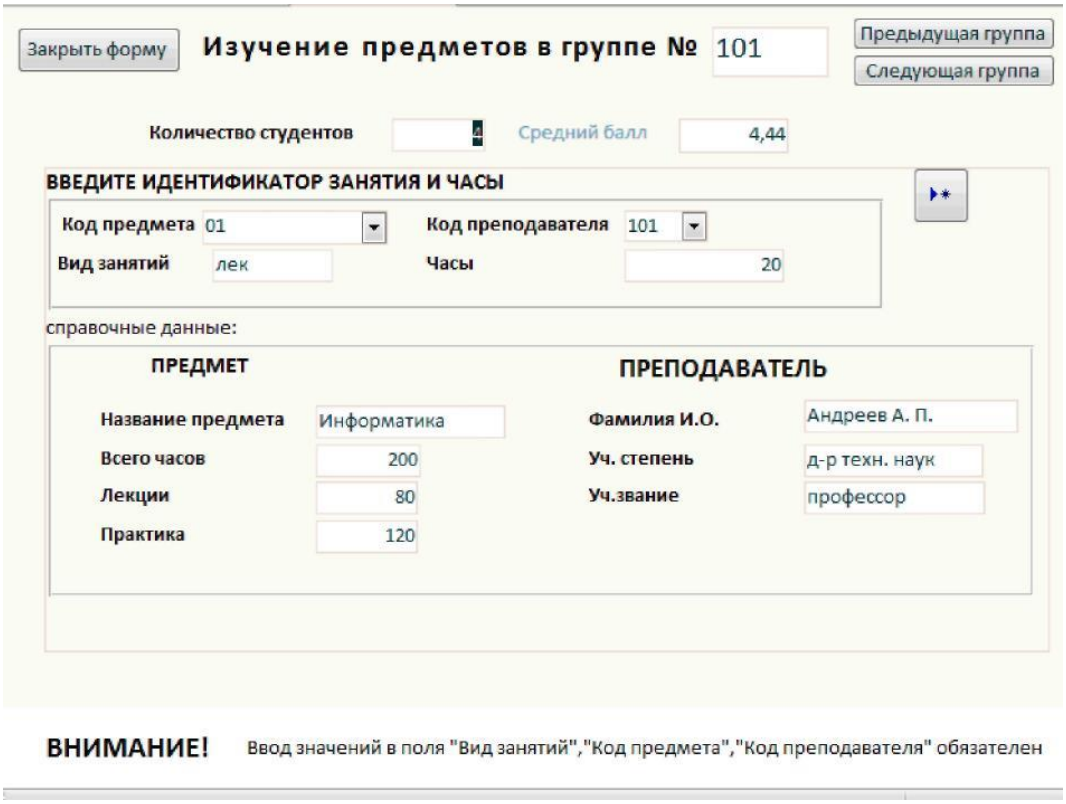
Замечание

Если необходимо преобразовать обычное поле в поле со списком , надо в контекстно-зависимом меню поля выбрать *преобразовать элемент в* и далее выбрать строку *поле со списком* при включенной кнопке *мастера элементов* элемент будет преобразован мастером. При выключенной кнопке необходимо для получения поля со списком установить свойства этого поля, как описано выше.

Загрузка данных в подчиненную таблицу через форму

Для загрузки данных в подчиненную таблицу ИЗУЧЕНИЕ через форму ПЛАН ЗАНЯТИЙ можно сразу перейти из режима конструктора в *режим формы***.** Для этого на вкладке *главная* в списке кнопки *режим* выбирается *режим формы***.**

Если форма была закрыта, необходимо в окне *область навигации* выбрать форму ПЛАН ЗАНЯТИЙ.



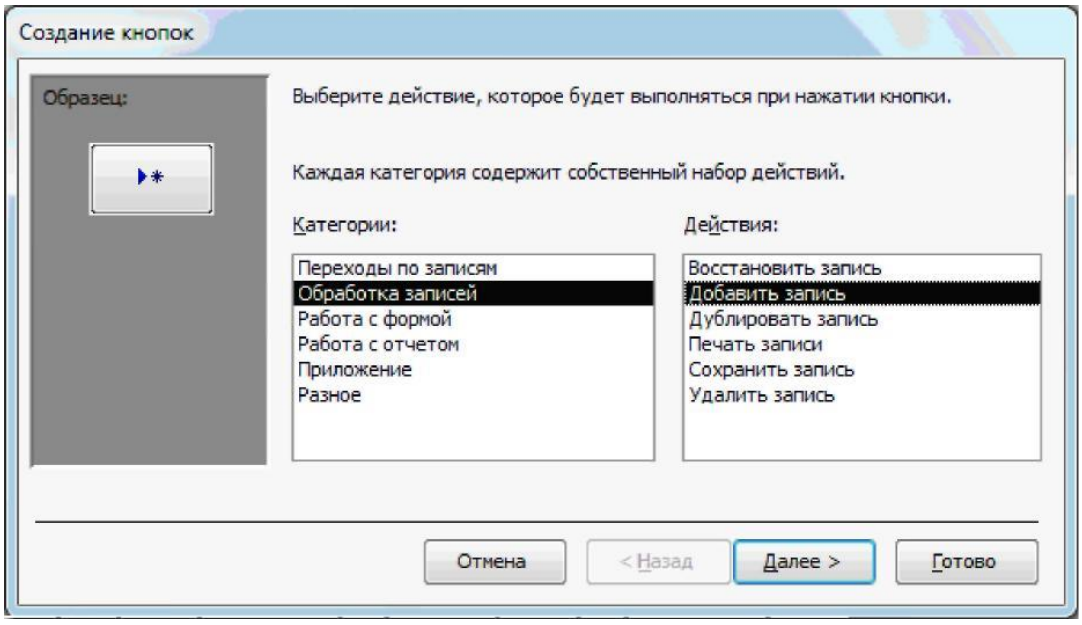
Загрузка подчиненной таблицы

Для загрузки записи нового занятия в таблицу ИЗУЧЕНИЕ через вызванную форму необходимо сделать текущим номер группы, для которой вводятся данные о занятиях из документа «План занятий». Это можно сделать путем просмотра записей групп при помощи кнопок *предыдущая группа* и *следующая группа*.

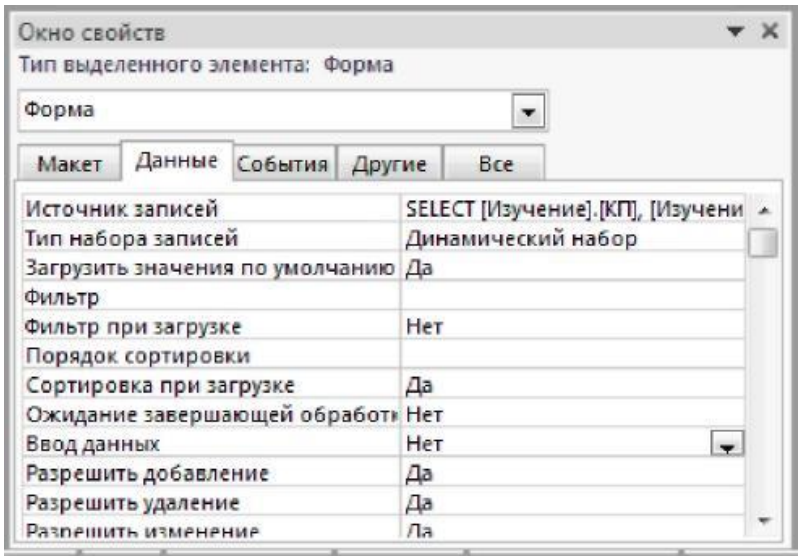
При большом числе групп целесообразно использовать функцию поиска нужной записи. Чтобы осуществить поиск, установим курсор в поле номера группы, нажмем на вкладке *главная* в группе *найти* кнопку *найти* и в открывшемся окне *поиск и замена* зададим в качестве образца поиска номер нужной группы. После нажатия в этом окне кнопки *найти далее* в форме отобразятся данные занятиях заданной группы.

Для ввода новых записей должен быть обеспечен переход в режим добавления новой записи в подчиненной форме. Для этого может быть создана специальная кнопка перехода к пустой записи *добавить запись*.

Для формирования такой кнопки используем мастер кнопок, в диалоговом окне которого *создание кнопок* надо выбрать соответствующую категорию *обработка записей* и действие *добавить запись*, которые обеспечат формирование нужной процедуры обработки события.



Установка свойств *разрешить добавление*, *разрешить удаление, разрешить изменение,* обеспечивающих возможность добавления, удаления и изменения записей при загрузке и корректировке записей таблицы ИЗУЧЕНИЕ – источника записей, показана в окне свойств подчиненной формы.

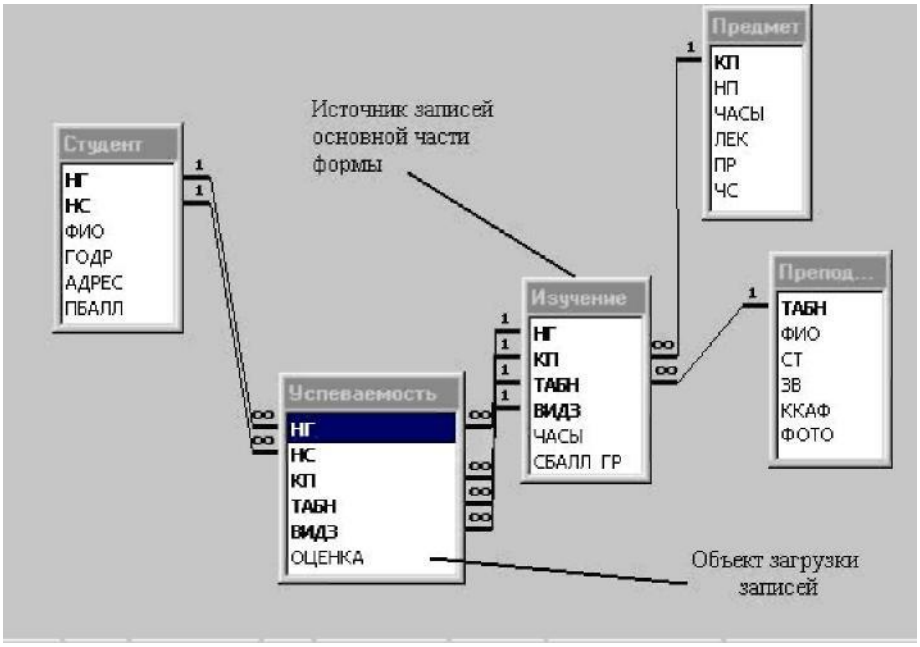


Упражнение

Создайте многотабличную форму, для загрузки результатов сдачи экзаменов в таблицу УСПЕВАЕМОСТЬ и их просмотра. При создании формы произведите действия, аналогичные рассмотренным для таблицы ИЗУЧЕНИЕ.

В соответствии с технологией загрузки базы данных, рассмотренной в начале настоящей главы, осуществите проектирование формы для загрузки данных в подчиненную таблицу УСПЕВАЕМОСТЬ из документа-источника «Экзаменационная ведомость».

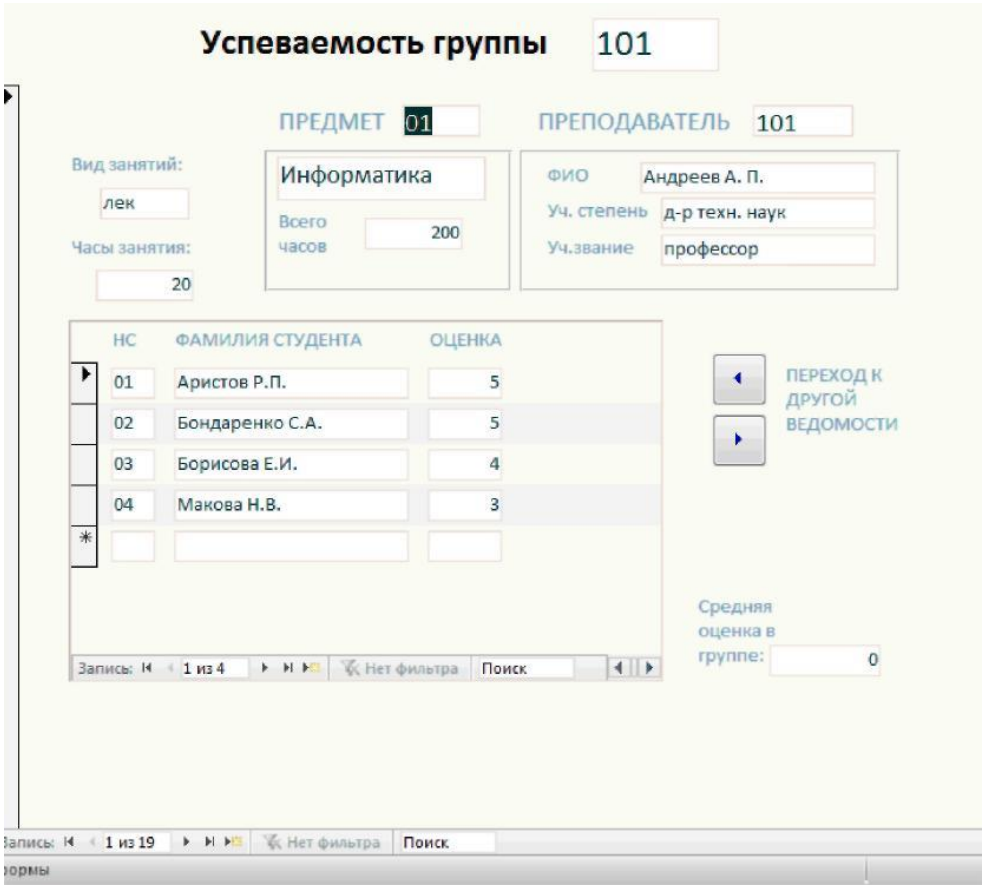
Определите общую структуру составной формы для ввода (просмотра) данных об оценках студентов группы по предмету в соответствии с подсхемой данных для составной формы.



В результате загрузки в БД данных об оценках студентов группы по предмету в БД должны формироваться только записи таблицы УСПЕВАЕМОСТЬ (объект загрузки). Загрузку и просмотр этих данных удобно производить по каждому проведенному в группе занятию в отдельности из соответствующей «Экзаменационной ведомости». Поэтому в подсхему для формы ввода включается таблица ИЗУЧЕНИЕ, которую по этой причине целесообразно выбрать в качестве источника основной части составной формы. Кроме того, в форме предусмотрен вывод (отображение) данных о предмете и преподавателе, проводящем занятие, а в списке студентов необходимо предусмотреть отображение его фамилии. Для этого в подсхему включены таблицы СТУДЕНТ, ПРЕДМЕТ и ПРЕПОДАВАТЕЛЬ.

Спланируйте размещение реквизитов в макете формы так, чтобы обеспечить удобный ввод данных в таблицу УСПЕВАЕМОСТЬ из документа «Экзаменационная ведомость», а также отображение справочной информации о студенте, предмете и преподавателе, в списке студентов предусмотрите вывод фамилии студента.

Выполните конструирование экранной формы, через которую будет осуществляться ввод, добавление и изменение записей таблицы базы данных УСПЕВАЕМОСТЬ.



Загрузите через построенную форму данные из документа «Экзаменационная ведомость» в таблицу УСПЕВАЕМОСТЬ. Используйте значения данных, приведенные в Приложении 2.

2.4. Создание запросов

В этом разделе на конкретных примерах рассматривается технология конструирования запросов различного вида. Дано описание конкретных действий пользователя в процессе конструирования запросов. Подробно описан процесс конструирования однотабличного и многотабличного запроса, показано формирование вычисляемых полей, использование групповых операций и параметров запроса. Рассмотрено обновление таблиц с помощью запроса. Кроме того, в конце раздела приведен пример решения задачи на основе нескольких запросов.

Конструирование однотабличного запроса на выборку

Рассмотрим процесс конструирования однотабличного запроса на выборку на примере получения информации из таблицы ПРЕДМЕТ базы данных «Учебный процесс».

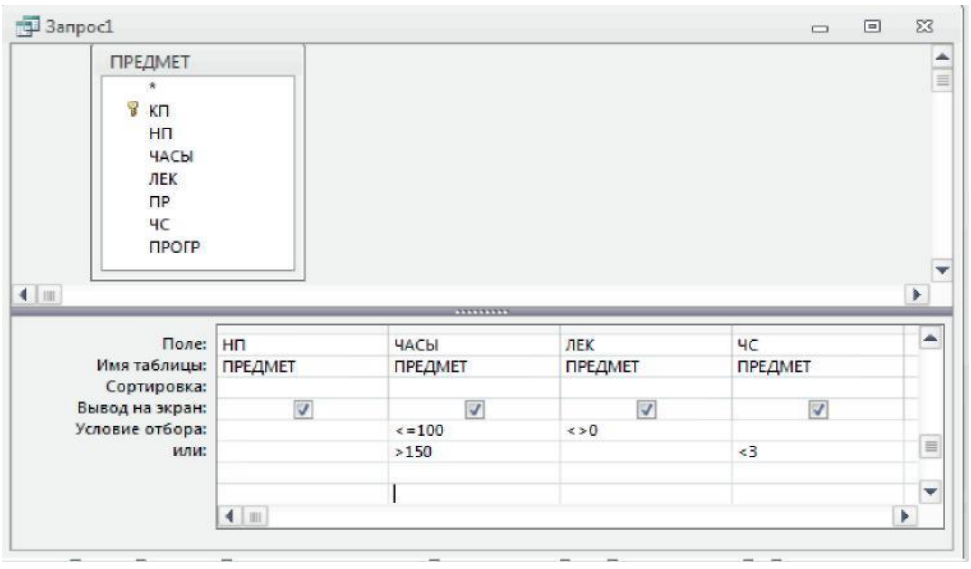
Использование логических операций в условии отбора

Пусть надо выбрать предметы, по которым общее число изучения не более 100, и есть лекции, а также выбрать предметы, по которым общее число часов больше 150 и число семестров изучения не более двух. Результат должен содержать наименование предмета (НП), общее число часов по предмету (ЧАСЫ), количество лекционных часов (ЛЕК) и число семестров (ЧС).

Для создания запроса в режиме конструктора выберем вкладку *создание* на панели быстрого доступа и нажмем кнопку *конструктор запросов*.

После нажатия кнопки появляется окно запроса на выборку в режиме конструктора *запрос1* и диалоговое окно *добавление таблицы*. В диалоговом окне выберем таблицу ПРЕДМЕТ и нажмем кнопку *добавить*. Выбранная таблица будет отображена в области схемы данных запроса. Закроем окно *добавление таблицы*.

В окне конструктора перетащим из списка полей таблицы ПРЕДМЕТ поля НП, ЧАСЫ, ЛЕК и ЧС в столбцы бланка запроса в строку *поле.*



Сформулированные в задаче условия требуют формирования следующего логического выражения:

(ЧАСЫ<=100 AND ЛЕК<>0) OR (ЧАСЫ>150 AND ЧС<3)

Здесь ЛЕК<>0 (число лекций не равно нулю), соответствует заданному в задаче условию выбрать предметы, в которых есть лекции.

Условия из первых скобок запишем в соответствующих полях ЧАСЫ и ЛЕК первой строки *условия отбора*. Между условиями в разных полях одной строки выполняется логическая операция. Условия из вторых скобок запишем в соответствующих полях ЧАСЫ и ЧС второй строки *условие отбора*. Между условиями, записанными в разных строках, выполняется логическая операция.

Выполним запрос, нажав на панели конструктора запросов кнопку *выполнить*.

На экране появится окно запроса в режиме таблицы с записями из таблицы ПРЕДМЕТ, отвечающими заданным условиям отбора.

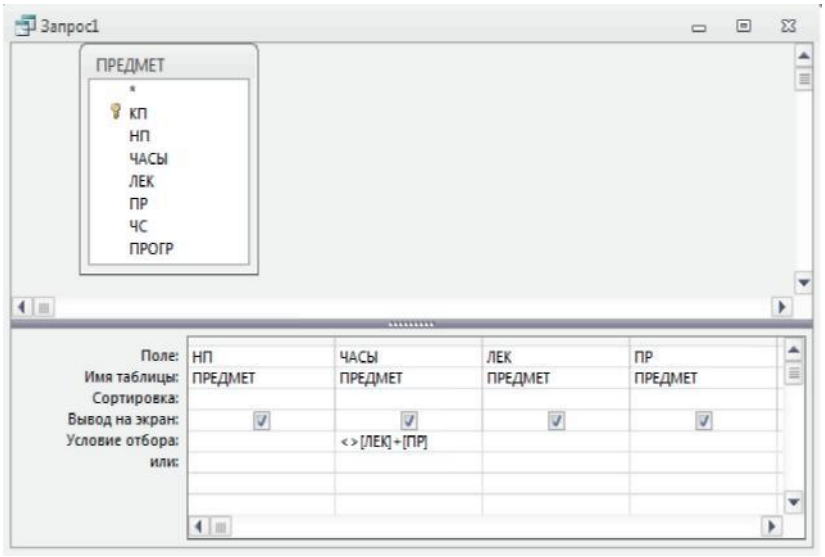
Сохраним запрос, нажав кнопку *сохранить* на вкладке *файл* и задав нужное имя запроса. Закроем текущий запрос нажав кнопку окна запроса *закрыть*. Сохраненный запрос можно выполнить, выделив запрос в окне *все объекты Access*, нажатием правой кнопки мыши и выбрав кнопку *открыть*.

Использование в условии отбора выражений с именами полей

В предыдущем примере в условии отбора в качестве операндов использовались только значения для отбора по конкретным полям. Создадим запрос, в условии отбора которого сравниваются значения в разных полях.

Пусть необходимо проверить правильность задания общих часов в таблице ПРЕДМЕТ. По запросу должны отбираться только те записи, в которых значение в поле ЧАСЫ не равно значению, получаемому при сложении значений полей ПР и ЛЕК.

Такое условие записывается в бланке запроса в столбце ЧАСЫ и в нем используются имена полей [ПР] и [ЛЕК].



Конструирование многотабличного запроса на выборку

Рассмотрим технологию конструирования многотабличного запроса на выборку на примере получения информации об успеваемости студентов из БД «Учебный процесс».

Запрос на основе нескольких взаимосвязанных таблиц

Пусть необходимо получить информацию об оценках полученных студентами по всем предметам. Результат должен содержать фамилию студента, наименования сданных предметов и оценки.

Для создания запроса на панели быстрого доступа выберем вкладку *создание* и нажмем кнопку *конструктор запросов*.

Формирование схемы данных запроса

В окне Добавление таблицы выберем таблицы:

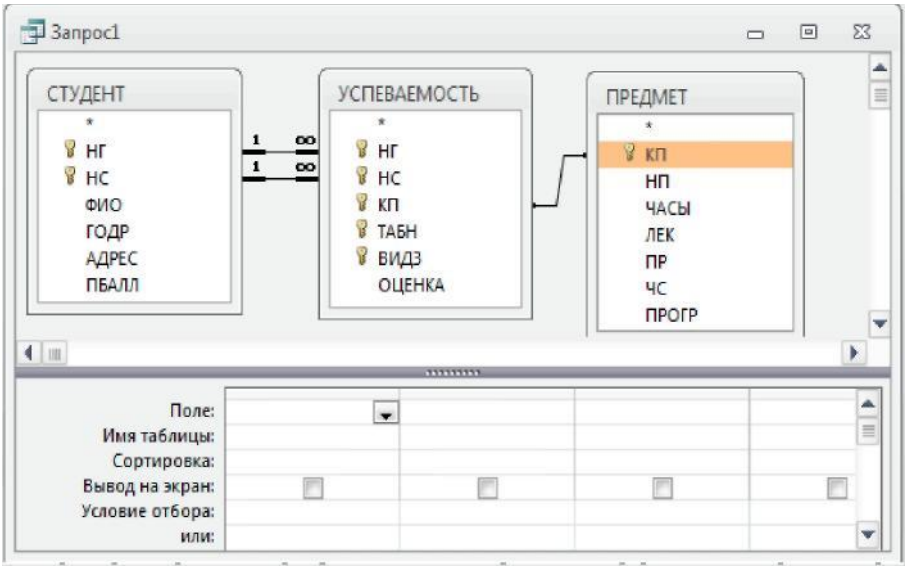
- СТУДЕНТ- для выборки фамилия студента из поля ФИО
- УСПЕВАЕМОСТЬ- для определения кодов предметов (поле КП), по которым студент сдал экзамены, выборки оценок по предмету (из поля ОЦЕНКА).
- ПРЕДМЕТ- для выборки наименования предмета (из поля НП), представленного кодом КП в таблице УСПЕВАЕМОСТЬ.

Закроем окно *добавление таблицы*.

В окне конструктора запросов представлена схема данных запроса, содержащая выбранные таблицы. Между таблицами автоматически установлены необходимые связи:

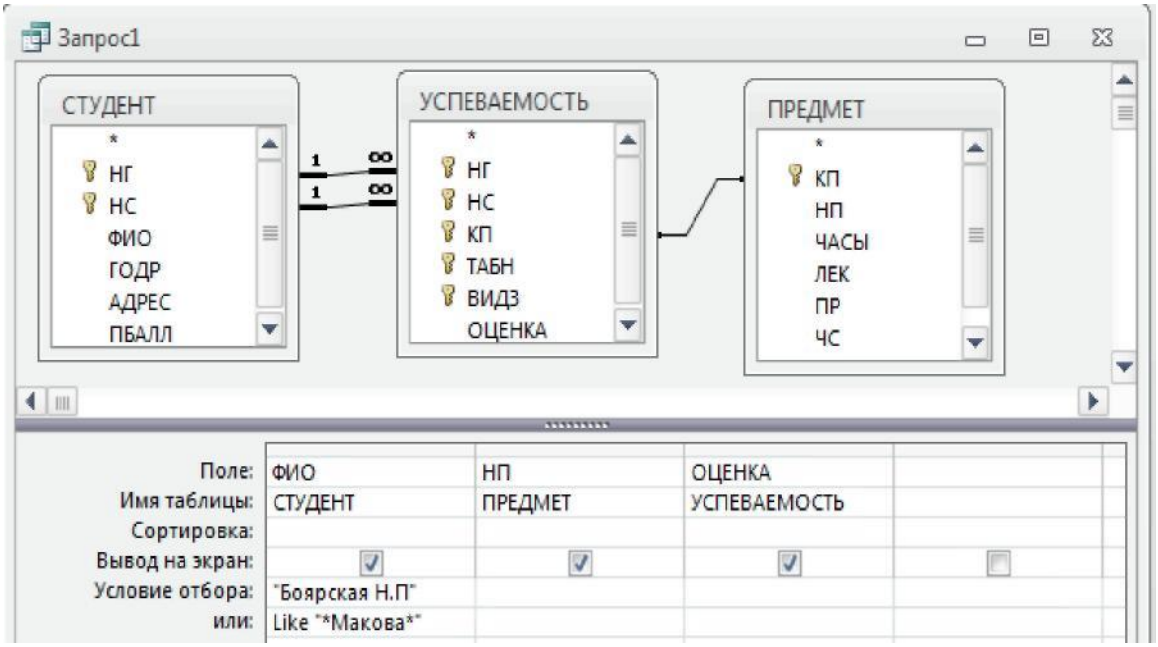
• *Одно-многозначная* связь между таблицами СТУДЕНТ и УСПЕВАЕМОСТЬ по составному ключу НГ+НС в соответствии с построенной ранее схемой данных.

• *Связь-объединение* между УСПЕВАЕМОСТЬ и ПРЕДМЕТ поскольку эти таблицы имеют поля с одинаковым именем КП и одинаковым типом данных.



Подготовка бланка запроса

Поскольку в запросе используется несколько таблиц, в бланке запроса удобно видеть имя таблицы наряду с именем поля. Для отображения имен таблиц в бланке запроса нажмем кнопку *имена таблиц* на панели инструментов конструктора или нажмем соответствующую кнопку в контекстном меню, который вызовем правой кнопкой мыши.



Перетащим с помощью мыши поля, включаемые в результат выполнения запроса, в строку бланка запроса *поле*:

- ФИО- из таблицы СТУДЕНТ
- НП- из таблицы ПРЕДМЕТ
- ОЦЕНКА- из таблицы УСПЕВАЕМОСТЬ

Ввод значений в условия отбора записей

Пусть необходимо получить информацию об успеваемости конкретных студентов: Боярской Н.П. и Маковой.

Зададим в строке *условие отбора* их фамилии. Запишем фамилии студентов в разных строках бланка запроса, поскольку необходимо выбрать записи со значением в поле ФИО-Боярская или Макова. Поскольку инициалы студентки Маковой неизвестны, ее фамилию зададим с использованием символа шаблона «*». Заметим, что фамилия с инициалами содержит точки, поэтому ее надо брать в кавычки. После ввода фамилии с символом шаблона система сама вставляет оператор Like, определяющий поиск по образцу.

Выполним запрос, нажав на панели конструктора запросов кнопку *выполнить*.

Замечание

Записи о заданном студенте появятся в результирующей таблице запроса только в том случае, если запись об этом студенте содержится в таблице СТУДЕНТ, а в таблице УСПЕВАЕМОСТЬ имеются записи, связанные с записью о студенте.

Формирование записей результата при выполнении запроса

Результат выполнения запроса об оценках заданных студентов.

По заданной фамилии студента- Боярская Н.П.- в таблице СТУДЕНТ отыскивается запись. По значению ключа связи НГ+НС осуществляется выборка подчиненных записей из таблицы УСПЕВАЕМОСТЬ с оценками данного студента по разным предметам (в поле ОЦЕНКА). Для каждой из этих записей по значению ключа связи КП выбирается одна запись с наименованием предмета (НП) из таблицы ПРЕДМЕТ.

Таким образом, таблица с результатом запроса будет содержать по одной записи о каждом предмете, сданном студентом. Аналогично формируются записи для второго заданного в запросе студента- Маковой.

Ввод параметров в запрос

В предыдущем примере для задания фамилии конкретного студента необходимо было корректировать бланк запроса. Чтобы избежать этого, целесообразно использовать в запросе параметры. При этом Access перед выполнением запроса через диалоговое окно будет запрашивать у пользователя конкретные значения параметров и введет их в условия отбора.

Пусть необходимо получить информацию об оценке студента по заданному предмету.

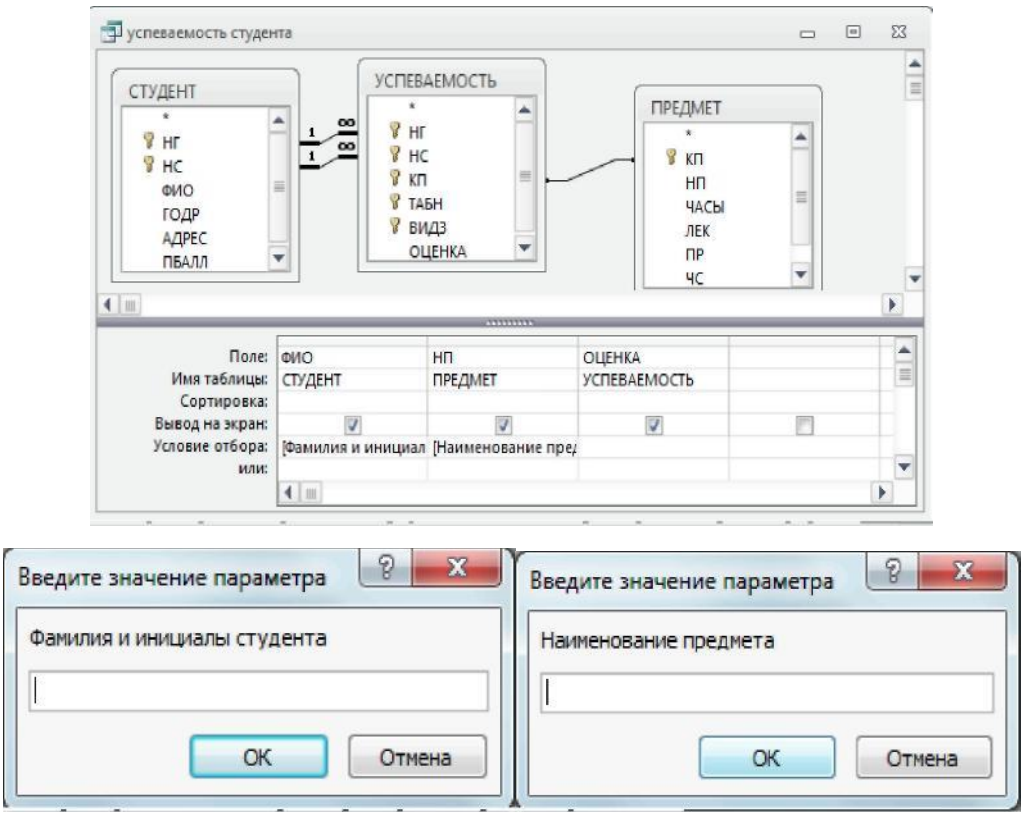
В условие отбора поля ФИО вместо конкретной фамилии введем название параметра, по которому будет запрашиваться фамилия при

выполнении запроса. Название параметра введем как текст, заключенный в квадратные скобки:

[Фамилия и инициалы студента]

Этот текст Access воспринимает как имя параметра. В условие отбора поля НП введем второй параметр запроса:

[Наименование предмета]



При выполнении запроса Access выведет диалоговые окна, в которые пользователь сможет ввести нужные значения параметров.

Использование имен полей различных таблиц в условии отбора

Пусть необходимо выбрать записи из таблицы ИЗУЧЕНИЕ, в которых часы практических занятий по информатике не соответствуют равномерному распределению по семестрам всех часов практики.

Для решения этой задачи необходимо использовать таблицы:

- ИЗУЧЕНИЕ, в которой содержатся сведения о плановых занятиях в группах (в текущем семестре), в том числе о продолжительности (поле ЧАСЫ) каждого вида занятия (поле ВИДЗ).

• ПРЕДМЕТ, в которой содержатся сведения о наименовании (поле НП), общей продолжительности изучения предмета (поле ЧАСЫ), числа часов практики (ПР) и числе семестров изучения (ЧС).

Для отбора записей о практических занятиях по информатике из таблицы ИЗУЧЕНИЕ надо в строке *условие отбора* для поля НП (ТАБЛИЦА ПРЕДМЕТ) задать значение «Информатика», а для поля ВИД3 (таблицы ИЗУЧЕНИЕ) задать значение «пр» (практическое занятие).

При равномерном распределении практики по семестрам общее число часов практических занятий по предмету (ПР) должно равняться произведению часов практики (ЧАСЫ) из таблицы ИЗУЧЕНИЕ на число семестров (ЧС) из таблицы ПРЕДМЕТ. Для решения рассматриваемой задачи надо включить в результат только те записи, для которых число часов не соответствует этому произведению. Для этого запишем в *условие отбора* поля ПР (таблицы ПРЕДМЕТ) выражение:

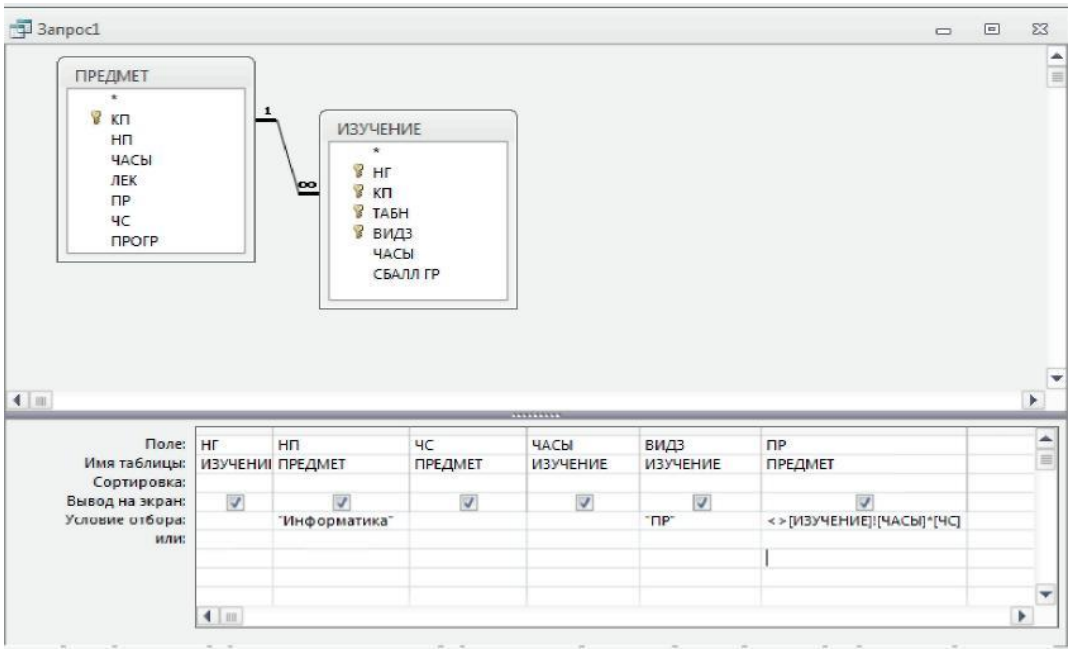
[ИЗУЧЕНИЕ] ! [ЧАСЫ]*[ЧС]

Замечание

Указывать таблицу ИЗУЧЕНИЕ для поля ЧАСЫ обязательно, потому что поле с таким же именем имеется и в таблице ПРЕДМЕТ.

Замечание

Если результат выполнения запроса не содержит записей, то это означает, что для заданного предмета в каждой из студенческих групп часы практических занятий соответствуют равномерному распределению по семестрам всех часов практики.



ЗАПРОС НА ВЫБОРКУ

Ном. группы	Название п	Семестров	Всего часов	Вид занятий	Практика
101	Информатика	4	60	пр	120
102	Информатика	4	180	пр	120
201	Информатика	4	180	пр	120
*					

Создание вычисляемых полей в запросах

Вычисляемое поле, включенное в запрос, позволяет получить новое поле с результатами вычисления, отображаемыми только в таблице запроса, и не создает полей в исходных таблицах базы данных.

Рассмотрим технологию создания запроса с вычисляемым полем на примере таблицы ПРЕДМЕТ.

Пусть необходимо найти записи о предметах, в которых общее число часов по предмету не совпадает с суммой часов лекций и практики. Для решения этой задачи рассчитаем разность между общим числом часов по предмету (поле ЧАСЫ) и суммой часов лекций (поле ЛЕК) и практики (поле ПР). в ответ включим только те записи, для которых эта разность не равна нулю.

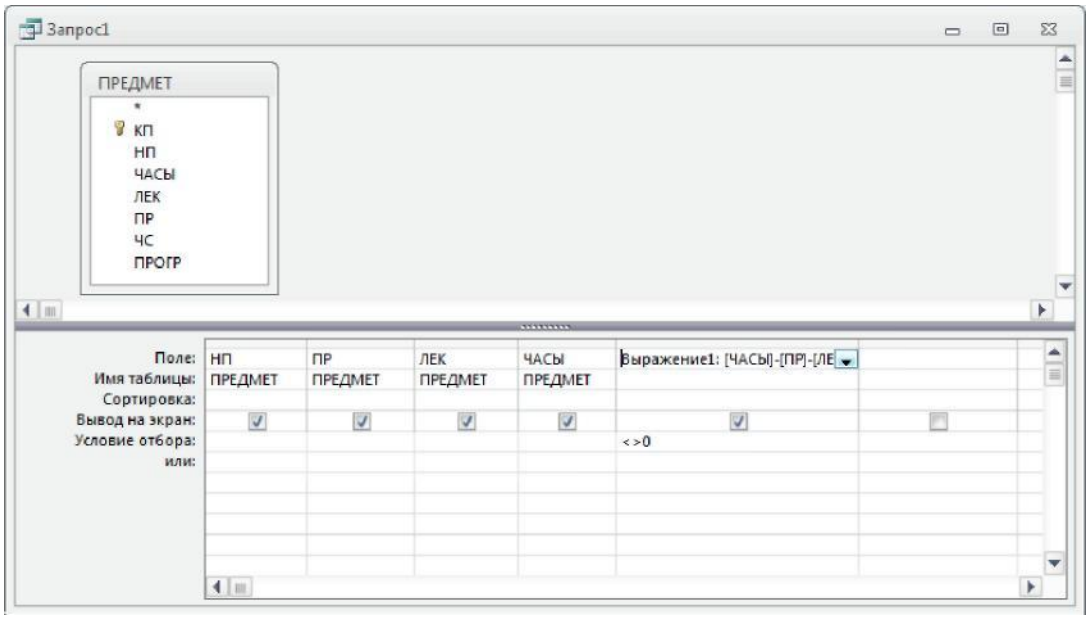
Создадим запрос на выборку для таблицы ПРЕДМЕТ. Перетащим в бланк запроса поля НП, ПР, ЛЕК, ЧАСЫ.

Создание вычисляемого поля

Для получения разности создадим вычисляемое поле в пустой ячейке строки *поле*, записав туда выражение:

[ЧАСЫ] - [ПР] - [ЛЕК]

Для отбора записей с ненулевым значением разности в вычисляемом поле в строку *условие отбора* введем <>0 (не равно 0).



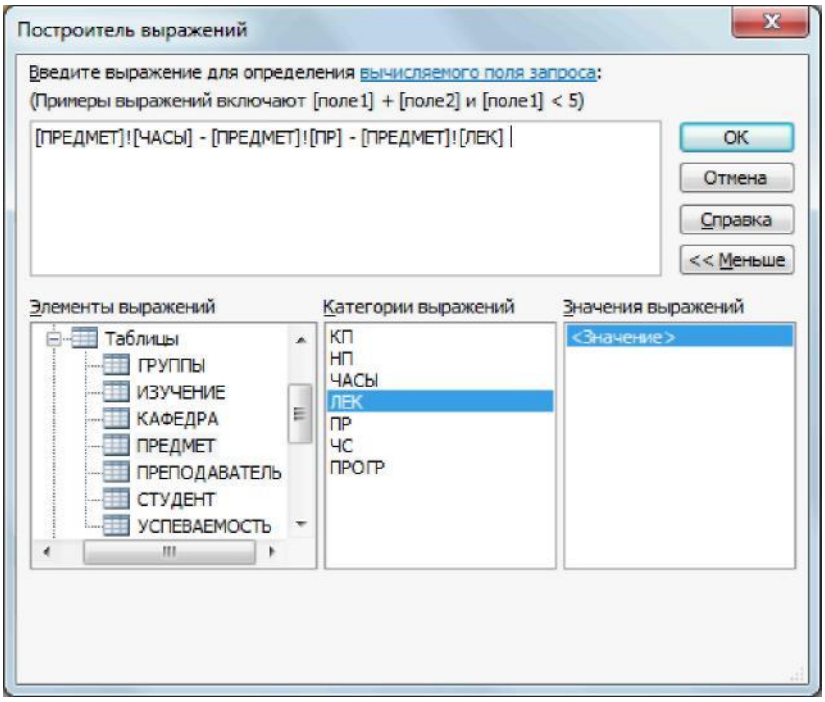
После ввода выражения система формирует имя вычисляемого поля по умолчанию- «Выражение 1». Это имя вставится перед выражением. Для изменения имени установим курсор мыши в вычисляемом поле бланка запроса и нажмем правую кнопку мыши. В контекстно- зависимом меню выберем *свойства* поля, а в строку *подпись* введем новое имя поля- «ЧАСЫ не равны ПР+ЛЕК». Имя поля может быть исправлено также непосредственно в бланке запроса.

Использование построителя выражений

Для формирования сложного выражения в вычисляемом поле целесообразно использовать построитель выражений. Построитель позволяет выбрать необходимые имена полей из таблиц, запросов, форм, знаки операций, функции.

Вызовем построитель выражений, нажав команду *построить* в контекстно- зависимом меню (курсор мыши должен быть установлен на строке *поле* вычисляемого поля).

В левой части окна *построитель выражений* выберем таблицу ПРЕДМЕТ, на которой построен запрос. Справа отобразится список ее полей. Последовательно выберем нужные поля, добавляя их двойным щелчком мыши, знаки операций вводятся с клавиатуры. При этом в верхней части окна сформируется выражение.



Сохраним запрос под именем «Разность часов по предмету». Сохранить. Сохраненный запрос можно выполнить, выделив запрос в окне *все объекты Access*, нажатием правой кнопки мыши и выбрав кнопку *открыть*.

Построенный запрос может быть использован для проверки правильности заполнения поля ЧАСЫ в таблице ПРЕДМЕТ.

Использование групповых операций в запросах

Назначение групповых операций

Групповые операции позволяют выделить группы записей с одинаковыми значениями в указанных полях и использовать для этих групп

одну из статистических функций. В Access предусмотрено девять статистических функций:

- Sum - сумма значений некоторого поля для группы
- Avg - среднее от всех значений поля в группе
- Max, Min - максимальное, минимальное значение поля в группе
- Count - число значений поля в группе без учета пустых значений
- Stdev - среднеквадратичное отклонение от среднего значения поля в группе
- Var - дисперсия значений поля в группе
- First и Last - значение поля из первой или последней записи в группе

Результат запроса с использованием групповых операций содержит по одной записи для каждой группы. В запрос включаются поля, по которым производится группировка, и поля, для которых выполняются групповые функции.

Порядок создания запроса с использованием групповых операций

Для создания запроса с использованием групповых операций формируется запрос на выборку. В бланк запроса включаются поля, по которым надо произвести группировку, и поля, по которым надо произвести статистические вычисления.

Выполните команду *создать/конструктор запросов* и на панели инструментов конструктора запросов нажмите кнопку *итоги*.

Для групповых вычислений по некоторому полю нужно заменить в нем слово *группировка* на нужную статистическую функцию. Выбрать нужную функцию можно через раскрывающийся в поле список.

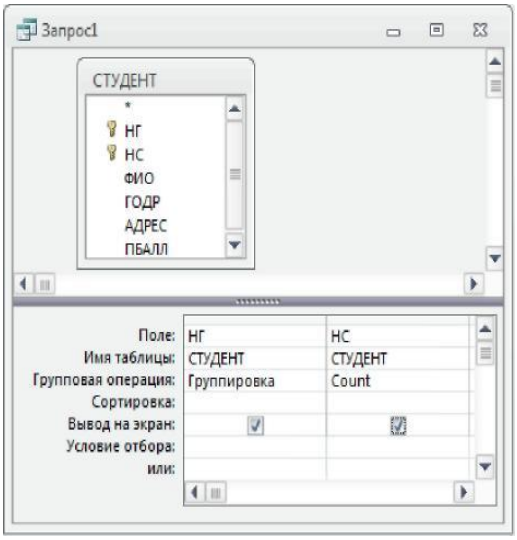
Конструирование однотабличного запроса с групповой операцией

Рассмотрим технологию конструирования однотабличного запроса с групповой операцией на примере таблицы СТУДЕНТ.

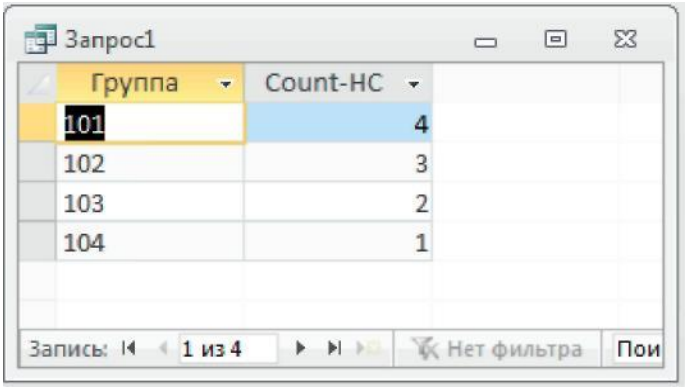
Запрос с функцией Count

Определим фактическое число студентов в группе. Создадим запрос на выборку из таблицы СТУДЕНТ. Из списка таблицы СТУДЕНТЫ перетащим в бланк запроса поле НГ (номер группы). Таким образом мы укажем, что по этому полю должна производиться группировка. Перетащим в бланк запроса поле НС, по которому будет вычисляться функция Count для подсчета числа студентов в группе.

Нажмем кнопку *итоги*. Заменим слово "Группировка" в столбце НС на функцию Count. Для этого вызовем список и выберем эту функцию. Бланк запроса примет вид.



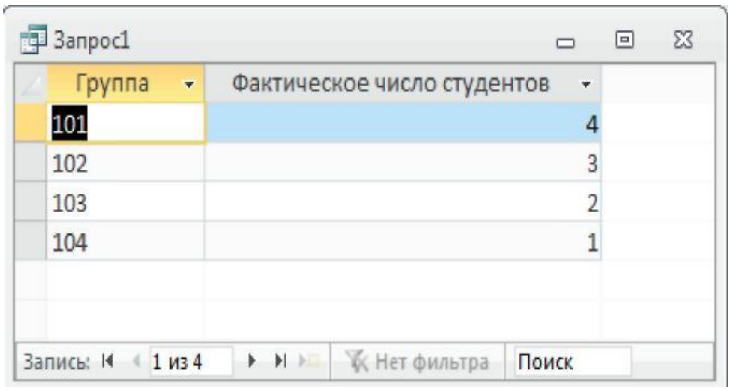
Результат запроса



Подпись поля "Count_НС" можно заменить на "Фактическое число студентов". Для ввода этой подписи в бланке запроса установим на поле НС курсор мыши и нажмем правую кнопку. В контекстно-зависимом меню

выберем команду *свойства* . В *окне свойств* наберем в строке *подпись* "Фактическое число студентов".

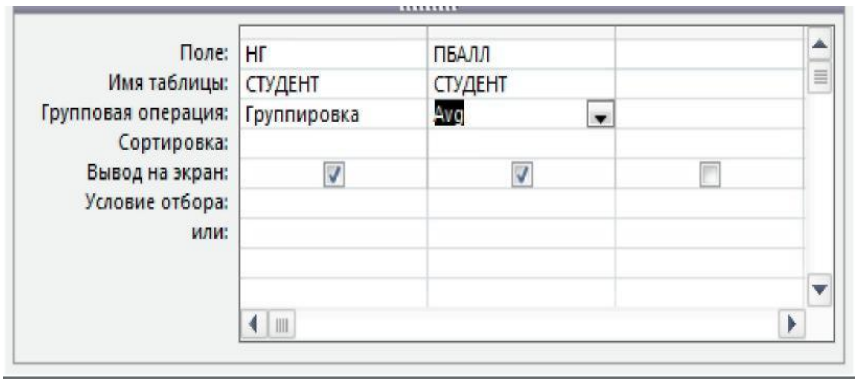
Таблица результата после доработки запроса



Сохраним запрос на выборку под именем "Число студентов в группах".

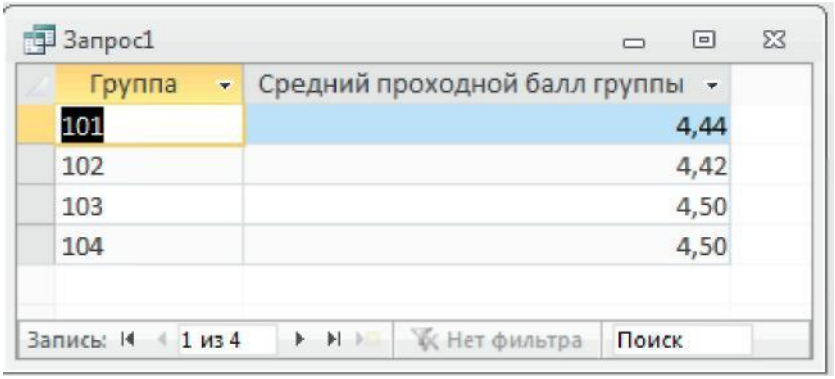
Запрос с функцией Avg

Подсчитаем средний проходной балл в группе. Сформируем запрос на выборку для таблицы СТУДЕНТ с функцией Avg для поля ПБАЛЛ (проходной балл студента). В бланке запроса заполним поля.



Для ограничения точности результата двумя знаками выберем в *окне свойств* в строке *формат поля* значение *фиксированный.*

Результат выполнения запроса



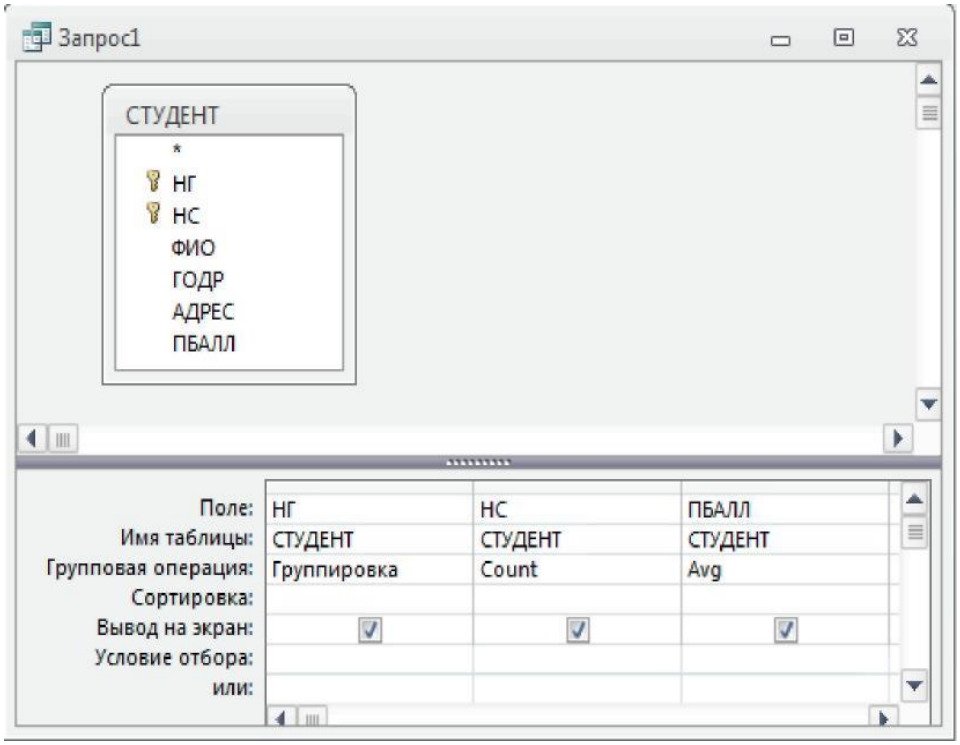
Сохраним этот запрос под именем "Средний проходной балл группы"

Запрос с несколькими групповыми функциями

Выполним расчет числа студентов и среднего проходного балла в группе в одном запросе. Это возможно, т. к. группы записей в обоих случаях формируются одинаково. Сохраним этот запрос под именем "Число студентов и средний ПБАЛЛ группы".

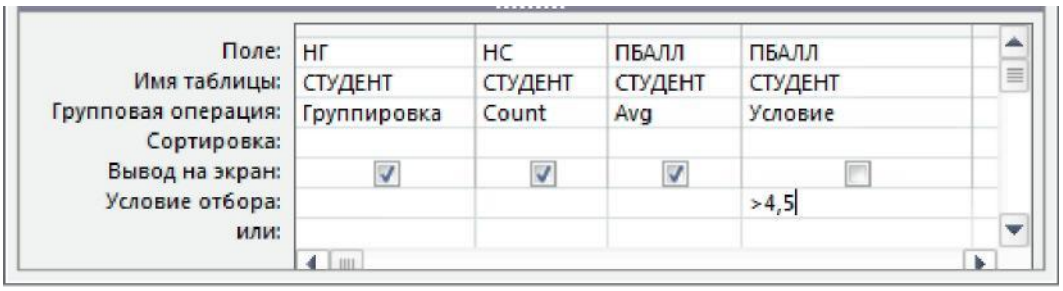
Задание условий отбора в запросах с групповыми операциями

В запрос с групповыми операциями можно включать поля для задания условий отбора записей из таблиц.



Подсчитаем число студентов в каждой из групп с проходным баллом больше 4,7.

Для этого в запрос *число студентов и средний пбалл группы* вторично включим поле ПБАЛЛ и в строке *групповые операции* заменим значение *группировка* на значение *условие*, выбрав его из списка. После этого введем в строку *условие отбора* ">4,5".



Заметим, что средний балл в этом запросе также вычисляется только для студентов с проходным баллом, превосходящим 4,7.

Условие отбора, заданное в поле, по которому проводится группировка, или в поле, где записана функция группировки, позволяет отобрать только нужные группы записей, например, группы студентов с заданным номером или с заданным средним проходным баллом.

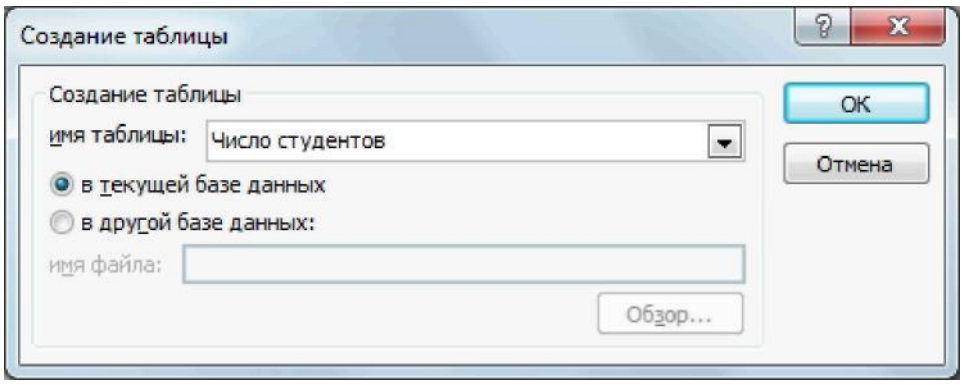
Конструирование запроса на создание таблицы

Запрос на *создание таблицы* используется для сохранения результата запроса. Этот вид запроса основан на *запросе на выборку*, но, в отличие от него, сохраняет таблицу с результатами запроса.

Необходимость в сохранении результатов запроса возникает, например, когда невозможно построить запрос непосредственно на другом запросе. К этому случаю относится, в частности, построение запроса на обновление полей на основе запроса с операцией группировки.

Сформируем запрос на создание таблицы на примере ранее полученного запроса на выборку с групповыми вычислениями *число студентов в группах.*

В области навигации вызовем названный запрос в режиме конструктора запросов. Преобразуем этот запрос в запрос на создание таблицы, выбрав тип запроса на панели конструктора *создание таблицы.* В окне *создание таблицы* введем имя создаваемой таблицы "Число студентов".



Для того, чтобы просмотреть, какие записи будут помещены в новую таблицу, щелкните по кнопке панели инструментов **Выполнить**. Выполните запрос, чтобы таблица ЧИСЛО СТУДЕНТОВ была сохранена в базе данных. Теперь эту таблицу можно увидеть в списке таблиц окна БД.

Упражнение

Преобразуйте запрос на выборку Средний проходной балл группы в запрос на создание таблицы, а создаваемую таблицу назовите “СРБАЛЛ”.

Конструирование запроса на обновление

Обновление полей значениями, рассчитанными с использованием групповых операций

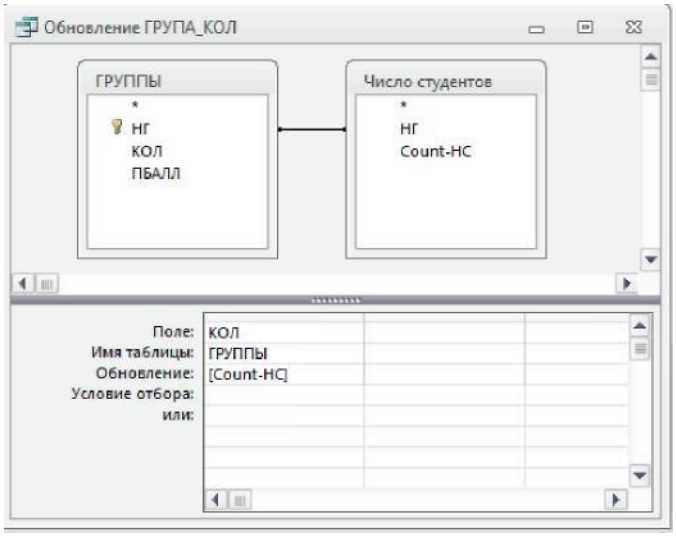
Рассмотрим технологию создания запроса на обновление на примере обновления поля КОЛ (количество студентов группы в таблице ГРУППА).

Количество студентов в группах ранее было подсчитано в запросе на выборку *Число студентов в группах* с использованием статистической функции Count. Запрос на обновление непосредственно на таком запросе построить нельзя. Поэтому используем для обновления не сам запрос, а таблицу ЧИСЛО СТУДЕНТОВ, полученную по запросу на создание таблицы в предыдущем пункте.

Для формирования запроса на обновление сначала создадим запрос на выборку на основе двух таблиц: обновляемой таблицы ГРУППА и таблицы

ЧИСЛО СТУДЕНТОВ, содержащей данные для обновления. Заметим, что в подсхеме данных запроса автоматически устанавливается связь этих таблиц по полю с именем НГ. Для преобразования запроса на выборку в

запрос на обновление выберем на панели конструктора тип запроса *обновление.*



Заполним бланк запроса. Перетащим обновляемое поле КОЛ из списка таблицы ГРУППА. В строке *обновление* введем имя поля "Count_НС" (таблицы ЧИСЛО СТУДЕНТОВ), из которого выбираются значения для обновления. Имя поля вводится в квадратных скобках.

Запрос можно выполнить, не выходя из режима конструктора. Содержимое обновляемого поля КОЛ можно просмотреть в режиме таблицы до и после выполнения запроса. Для последующего использования подготовленного запроса сохраним его под именем "Обновление ГРУППА_КОЛ".

Упражнение

1. Произведите обновление поля ПБАЛЛ – средний проходной балл в таблице ГРУППА значениями из ранее созданной таблицы СРБАЛЛ.

2. Произведите обновление поля СРБАЛЛ-ГР – средняя оценка в группе по предмету в таблице ИЗУЧЕНИЕ. Для выполнения задания:

• создайте запрос к таблице УСПЕВАЕМОСТЬ для расчета средней оценки в группе по предмету и сохраните результат в таблице, для чего группировку произведите по двум полям: НГ – номер группы и КП – код предмета:

• обновите поле СРБАЛЛ-ГР в таблице ИЗУЧЕНИЕ, используя сохраненный результат.

Использование выражений в запросе на обновление

Рассмотрим формирование запроса на обновление с использованием выражения на примере заполнения поля ЧАСЫ для лекционных занятий в таблице ИЗУЧЕНИЕ. Пусть поле ЧАСЫ должно обновляться данными, вычисляемыми на основе полей ЛЕК (часы лекций) и ЧС (число семестров) из таблицы ПРЕДМЕТ. Расчетное число часов по лекциям определим по формуле ЛЕК/ЧС.

В соответствии с задачей в записях лекционных занятий таблицы ИЗУЧЕНИЕ необходимо обновить поле ЧАСЫ расчетным числом часов. Записи о лекционных занятиях можно выбрать по значению поля ВИДЗ этой таблицы, т. к. в нем указан вид занятия. Данные для расчета среднего числа часов содержатся в таблице ПРЕДМЕТ. Таким образом запрос должен строиться на основе таблиц ИЗУЧЕНИЕ и ПРЕДМЕТ.

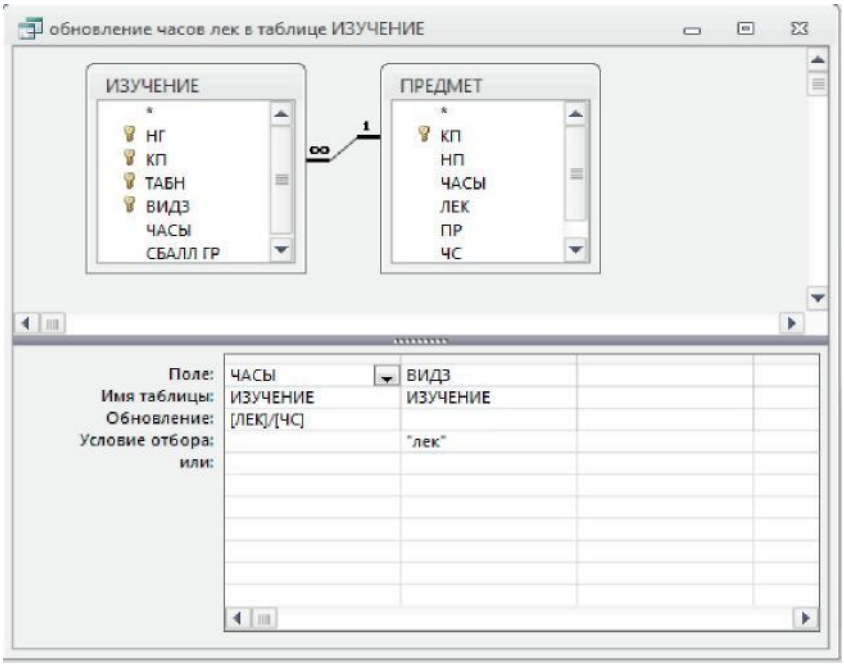
Создадим сначала запрос на выборку на основе таблиц ИЗУЧЕНИЕ и ПРЕДМЕТ. Затем преобразуем его в запрос на обновление, нажав соответствующую кнопку панели инструментов.

Включим в бланк запроса обновляемое поле ЧАСЫ таблицы ИЗУЧЕНИЕ. В строке *обновление* для этого поля введем выражение [ЛЕК]/[ЧС]. Для отбора в таблице ИЗУЧЕНИЕ обновляемых записей о лекционных занятиях в бланк запроса включим поле ВИДЗ и укажем в поле *условия отбора* значение "лек".

Окончательно сформированный запрос.

Выполним запрос, нажав кнопку *выполнить.* В диалоговом окне появится сообщение о числе обновляемых записей.

Чтобы видеть результаты обновления в таблице ИЗУЧЕНИЕ, откройте ее одновременно с запросом.



Конструирование перекрестного запроса

Создание перекрестного запроса, который позволяет получить данные в форме, подобной электронной таблице, с помощью мастера было рассмотрено выше. Однако такой запрос несложно построить полностью в режиме конструктора.

Построение запроса начинается как обычно, например, выбрать вкладку *создание/ конструктор запросов* в окне базы данных. В окне конструктора начинается создание запроса на выборку. В любой момент строящийся запрос на выборку может быть преобразован в перекрестный запрос. Для этого надо выбрать тип запроса *перекрестный* на панели.

Ниже рассмотрим действия пользователя при работе с перекрестным запросом в режиме конструктора на примере.

Воспользуемся в качестве примера перекрестным запросом *изучение предметов группами*, полученным мастером перекрестных запросов. Для большей информативности полученной перекрестной таблицы заменим в ней коды предметов их наименованиями. Полю, содержащему результат суммирования по строкам, дадим пользовательское имя "Всего часов".

Для выполнения перечисленных преобразований откроем перекрестный запрос *изучение предметов группами* в режиме конструктора.

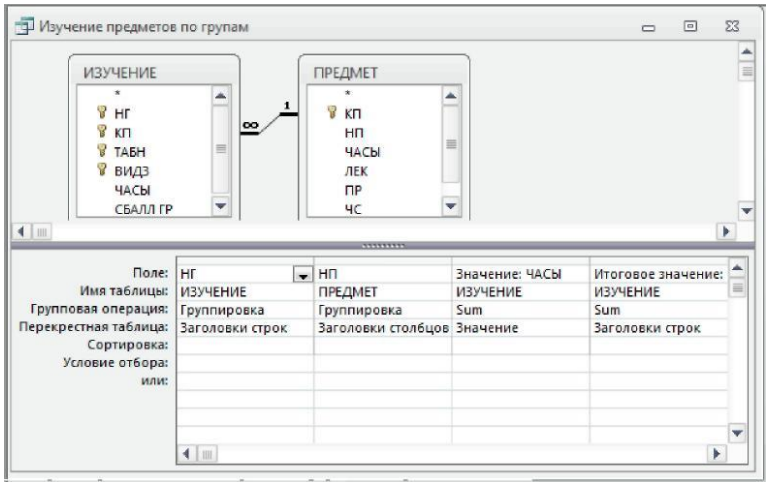
Поле НП (наименование предмета) размещено в таблице ПРЕДМЕТ, поэтому ее нужно добавить к разрабатываемому запросу. Для этого, находясь в окне конструктора, нажмем кнопку *отобразить таблицу.*

Теперь схема данных запроса состоит из таблиц ПРЕДМЕТ и ИЗУЧЕНИЕ, связанных по полю КП (код предмета) отношением один-ко-многим.

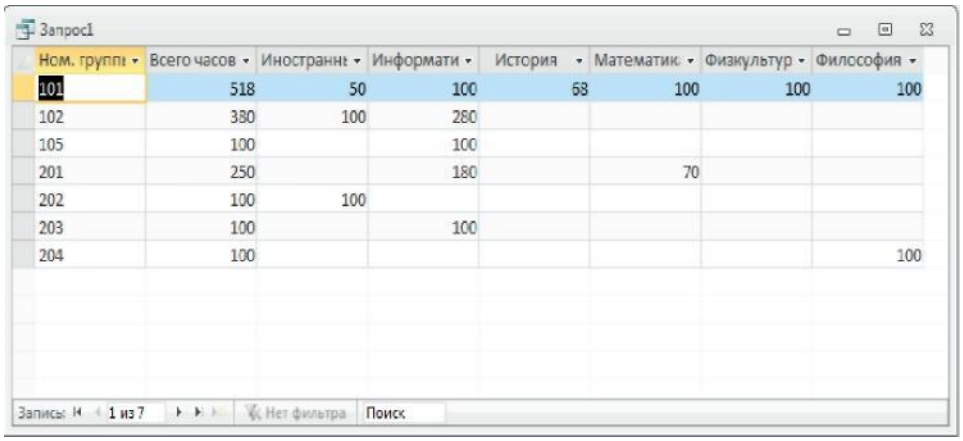
Заменим в бланке запроса поле КП на поле НП таблицы ПРЕДМЕТ. Для этого щелкнем правой кнопкой мыши на области отображения полей таблиц и выберем *имена таблиц,* чтобы получить в бланке информацию о принадлежности поля к таблице. Далее в поле КП в строке *имя таблицы* нажмем кнопку списка и выберем поле ПРЕДМЕТ, а в строке *поле* - поле НП.

Для изменения подписи поля ИТОГОВОЕ ЗНАЧЕНИЕ, содержащего сумму по строкам, щелкнем правой кнопкой мыши, находясь в зоне этого поля. В открывшемся контекстно-зависимом меню выберем пункт *свойства.* В окне *свойства* введем в строку *подпись* "Всего часов".

Окончательно сформированный перекрестный запрос

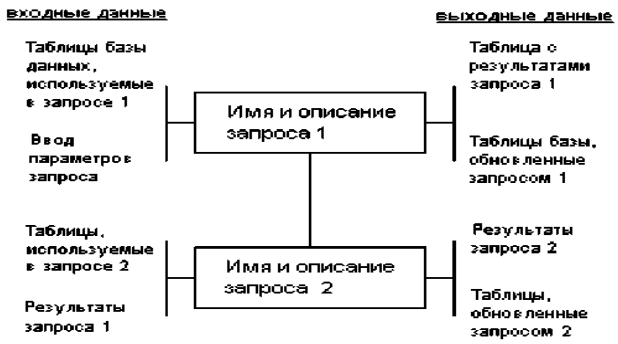


Результат выполнения полученного перекрестного запроса



Решение задач на основе нескольких запросов

Выше рассматривались примеры простых задач, решение которых осуществлялось выполнением одного запроса. В этом случае запросом реализуется весь алгоритм формирования результата на основе входных данных из таблиц базы и параметров задачи.



Для описания алгоритма задач, реализуемых одним запросом, обычно достаточно словесного описания действий. Целесообразно также использовать функционально-технологическую схему, на которой указываются входные и выходные таблицы данных.

Более сложные задачи требуют последовательного выполнения нескольких запросов. Каждый из запросов имеет свои входные и выходные данные. В простейшем случае выходные данные предшествующего запроса

являются входными для следующего построенного на нем запроса, и, только выполнив последний запрос в цепочке запросов построенных друг на друге, вы инициируете последовательное выполнение всех запросов цепочки и полное решение задачи.

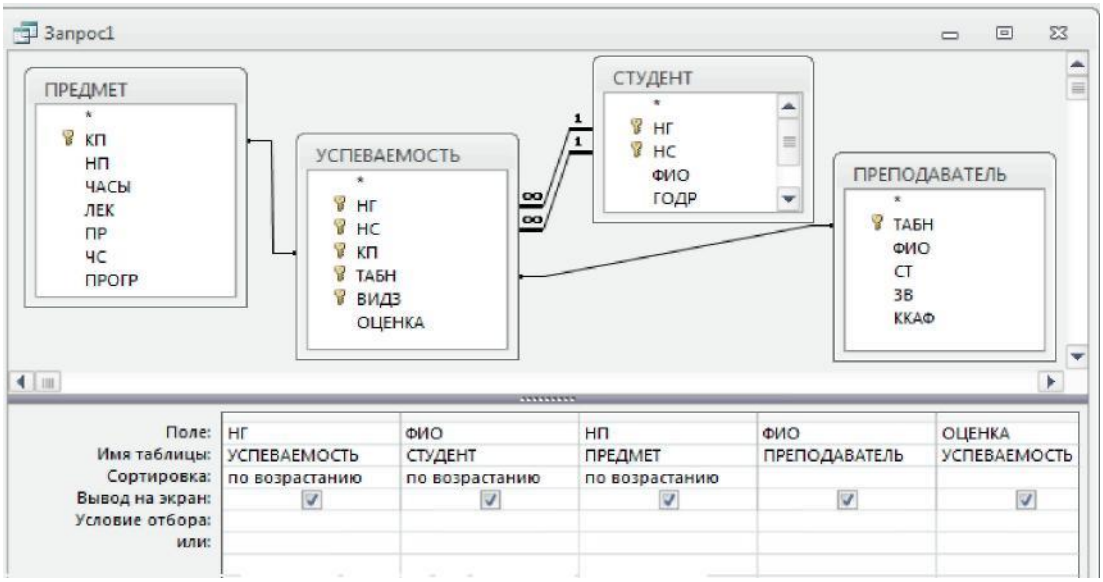
Запросы в Access являются мощным средством решения различных задач. При этом возможно построение сложных запросов, в том числе построенных на других запросах. Ниже рассматривается построение таких запросов, а также реализация задачи последовательно выполняющимися запросами.

Построение запроса на основе другого запроса

Выполним анализ оценок, полученных студентами по различным предметам. Например, подсчитаем число оценок (2,3,4,5) по каждому из предметов.

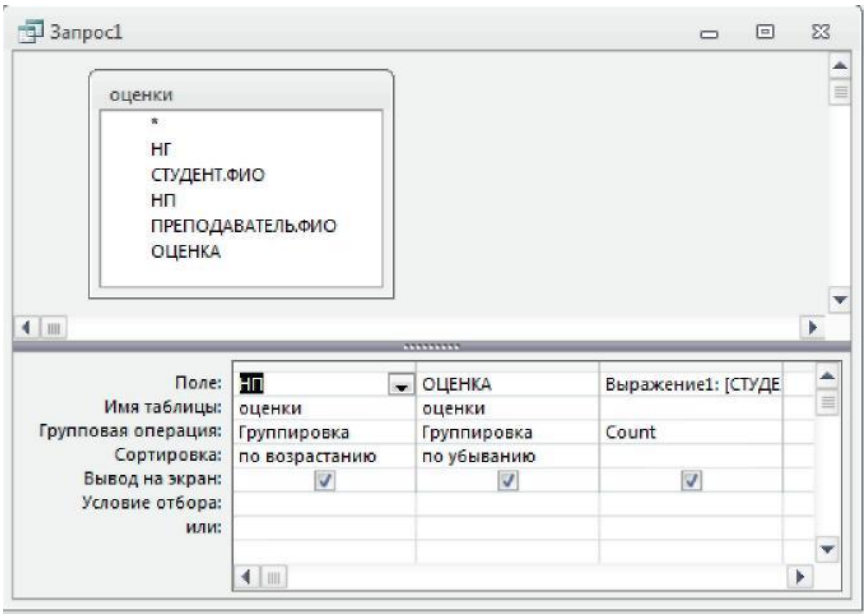
Создадим сначала многотабличный запрос на выборку на основе таблиц СТУДЕНТ, УСПЕВАЕМОСТЬ, ПРЕДМЕТ, ПРЕПОДАВАТЕЛЬ, формирующий сведения об оценках, полученных студентами по различным предметам. Для этого в режиме конструктора создадим схему данных запроса и бланк. Сохраним этот запрос с именем "оценки".

В результате выполнения этого запроса будет получена таблица, источником записей которой является таблица УСПЕВАЕМОСТЬ, а расшифровывающие данные выбираются из таблиц: ПРЕДМЕТ, СТУДЕНТ, ПРЕПОДАВАТЕЛЬ. Таким образом, каждая строка результата будет содержать информацию об одной оценке, полученной студентом по указанному в строке предмету. Число строк в таблице запроса будет равно числу строк в таблице УСПЕВАЕМОСТЬ.



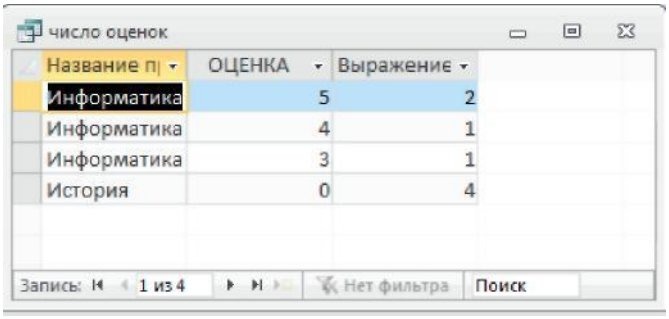
Для подсчета числа различных оценок (2,3,4,5) по каждому из предметов на основе этого запроса создадим новый запрос - *число оценок*. При создании нового запроса в окне *отразить таблицу/добавление таблицы* на вкладке **Запросы** выберем из списка запрос *оценки*. Заполним бланк запроса.

Результат выполнения запроса *число оценок*, где в столбце *выражение 1* отображено количество оценок, полученных по каждому предмету.



Замечание

Нет необходимости предварительно выполнять запрос *(оценки)*, на основе которого выполняется другой запрос *(число оценок)*. Выполнение вложенного запроса инициируется системой при выполнении запроса, построенного на нем.



Упражнения

• Создайте на основе запроса оценки, запрос для анализа оценок, выставленных каждым из преподавателей. Результат запроса должен содержать количество оценок (2,3,4,5), выставленных каждым преподавателем. Подпись столбца с результатами выполнения групповой операции Count Выражение1 замените на Количество оценок

• Создайте на основе запроса оценки запрос для определения числа студентов, получивших 2,3,4 или 5 по предмету, задаваемому в диалоге с пользователем

• Создайте на основе запроса оценки запрос для подсчета числа студентов в группе, получивших 2 (или другую заданную оценку) по каждому предмету. Предусмотрите ввод номера группы и оценки в диалоге с пользователем

• Создайте на основе запроса оценки запрос для подсчета средней оценки в группе по каждому предмету; средней величины оценок, выставленных преподавателем; средней успеваемости по каждому предмету

Решение задачи, требующей выполнения нескольких запросов и сохранения промежуточных результатов

Пусть необходимо определить среднюю нагрузку преподавателя кафедры в текущем семестре. Для этого необходимо подсчитать число преподавателей кафедры, затем общее количество часов занятий,

проводимых кафедрой, и завершить решение задачи расчетом средней нагрузки преподавателя.

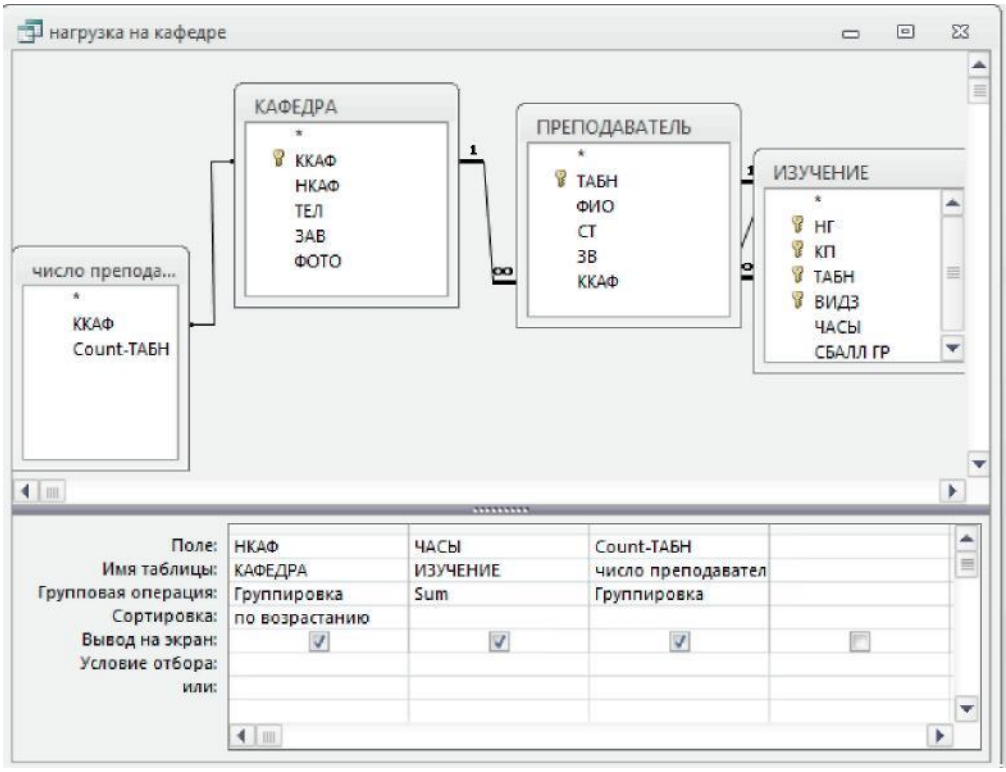
Подготовим и последовательно выполним соответствующие запросы.

Первый запрос. Создадим первый запрос на выборку, в котором по таблице ПРЕПОДАВАТЕЛЬ с помощью функции Count подсчитаем число преподавателей по кафедрам. Сохраним запрос под именем "Число преподавателей кафедры".

Второй запрос. Подготовим второй запрос на выборку для подсчета общего числа часов занятий, проводимых каждой кафедрой.

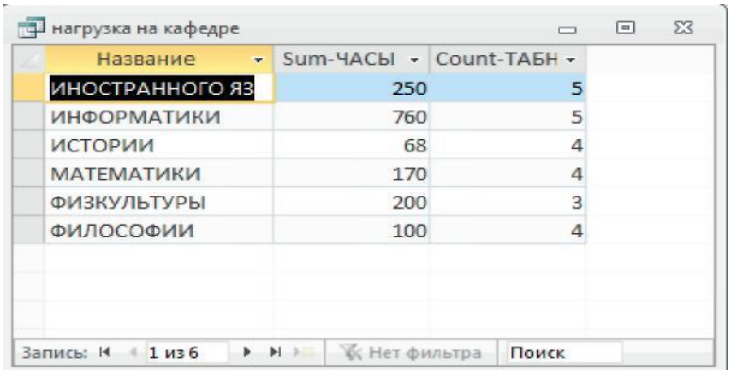
Этот запрос построим на базе таблиц ПРЕПОДАВАТЕЛЬ, ИЗУЧЕНИЕ, КАФЕДРА и запроса *число преподавателей кафедры*. Таблицы ПРЕПОДАВАТЕЛЬ и ИЗУЧЕНИЕ нужны для суммирования числа часов занятий, проводимых преподавателями каждой кафедры. Таблица КАФЕДРА необходима для включения в результат наименования кафедры, а запрос *число преподавателей кафедры* - для включения в результат числа преподавателей на кафедре.

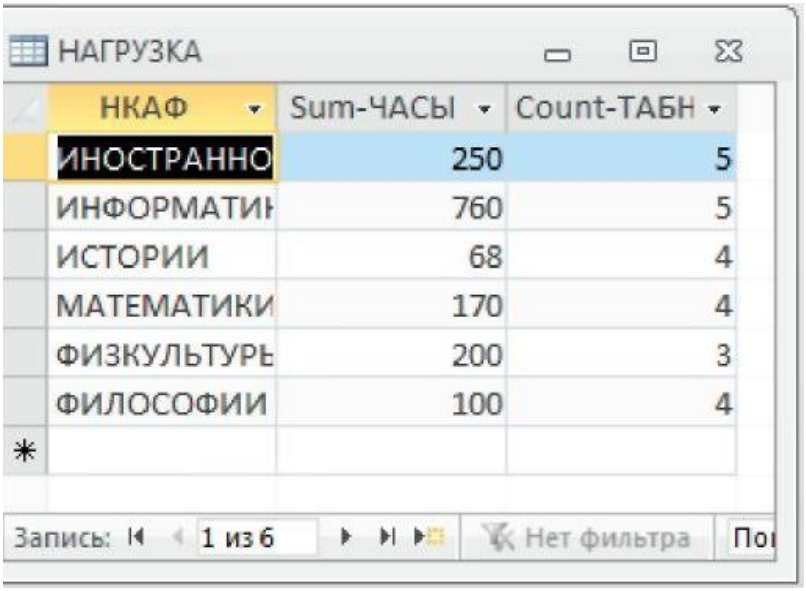
Записи этого запроса формируются на основе записей таблицы ИЗУЧЕНИЕ, причем число записей до проведения группировки равно числу записей в этой таблице. В результате запроса к каждой записи добавляется наименование кафедры НКАФ, по которому и производится группировка. Число преподавателей кафедры Count_ТАБН никак не нарушает требуемого объединения записей в группы, поскольку для каждой кафедры является единственным. Число записей в таблице результата запроса равно числу кафедр.



В рамках данного запроса нельзя сразу вычислить среднюю нагрузку преподавателя, поскольку число преподавателей кафедры является результатом выполнения групповой операции. Использование результатов выполнения запроса с групповой операцией не допускается в вычисляемых полях. Поэтому необходимо сохранить результаты выполнения второго запроса в новой таблице и на ее основе построить следующий запрос, в котором будет произведен расчет средней нагрузки преподавателя.

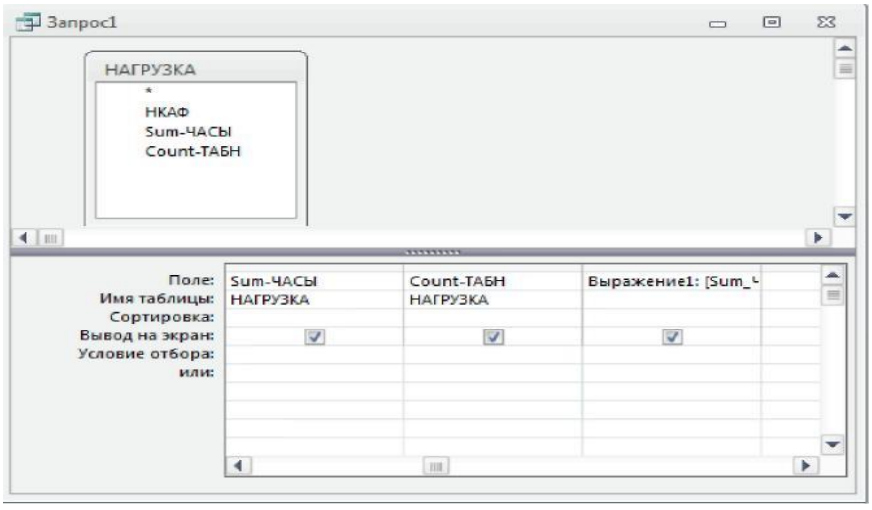
Чтобы сохранить полученные результаты, преобразуем второй запрос на выборку в запрос на создание таблицы. Таблице, которая будет создана запросом, присвоим имя "Нагрузка". Запрос сохраним под именем " Нагрузка на кафедре"



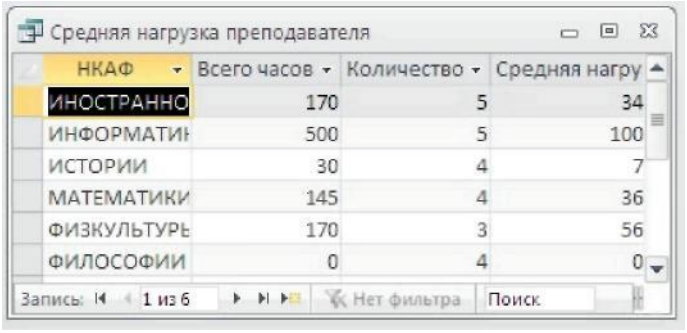


Третий запрос. Для окончательного решения задачи расчета средней нагрузки преподавателя кафедры подготовим на базе таблицы НАГРУЗКА третий запрос на выборку с вычисляемым полем. Для создания вычисляемого поля, рассчитывающего среднюю нагрузку преподавателя, в строку *поле* пустого столбца введем выражение [Sum_ЧАСЫ]/[Count_ТАБН].

В таблице результата следует изменить заголовок столбца *выражение1*, формируемый по умолчанию для вычисляемого поля, и его формат (для получения результата с округлением до целого). Для этого вызовем свойства поля с помощью контекстно-зависимого меню. Зададим в качестве подписи поля значение "Средняя нагрузка преподавателя", формат поля определим как фиксированный, а параметру *число десятичных знаков* присвоим значение "0"



Результаты выполнения запроса после изменения подписей полей в свойствах



2.5. Отчеты

Рассмотрим технологию создания однотабличного отчета на примере получения списков студентов по группам.

Пусть в результате проектирования макета отчета СПИСКИ СТУДЕНТОВ определены перечисленные ниже требования. На макете показано оформление списка студентов для одной группы. В отчете должны последовательно выводиться со своими заголовками списки студентов для каждой группы. При формировании отчета необходимо рассчитать средний проходной балл для каждой группы и отобразить его в отчете. Записи списка группы должны выводиться в порядке возрастания номера студента в группе. Название отчета должно выводиться на каждой странице отчета.

СПИСКИ СТУДЕНТОВ

(Текущая дата)

Список студентов группы _____

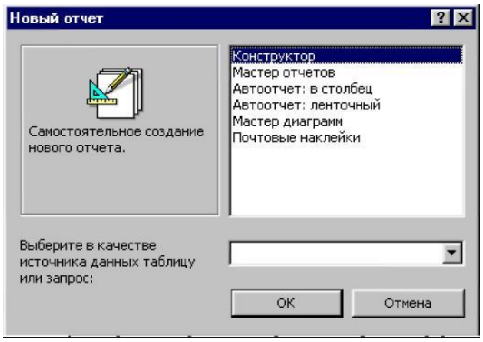
Номер	Фамилия И О	Год рождения	Прох балл
_____	_____	_____	_____
_____	_____	_____	_____
_____	_____	_____	_____
_____	_____	_____	_____
_____	_____	_____	_____

Средний проходной балл группы _____

Создание однотабличного отчета в режиме конструктора

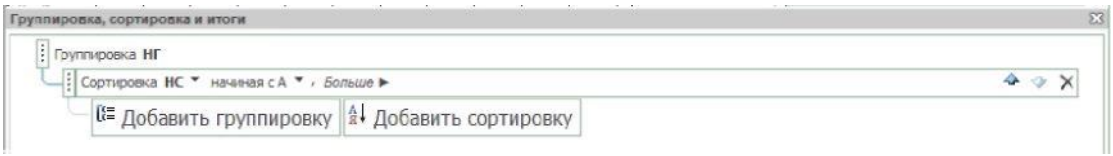
В пункте меню *создание* нажимаем кнопку *конструктор отчетов.* В области данных вызываем контекстное меню, выбираем пункт *свойства*. В открывшемся окне выбираем вкладку *данные* и заполняем соответствующее поле именем таблицы СТУДЕНТ, которая будет служить источником записей для нашего отчета. В пункте меню Конструктор нажимаем кнопку *добавить поля.*

Если в открывшемся окне конструктора отсутствует раздел Заголовок отчета, то вызываем контекстное меню и выбираем пункт *заголовок/ примечание отчета*.



Группировка и сортировка данных отчета

Для выполнения требования к группировке и сортировке данных, отображаемых в отчете, нажмем кнопку *группировка* на панели *группировка и итоги* конструктора и зададим необходимые параметры в открывшемся диалоговом окне *группировка, сортировка и итоги.*



Группировка по полю

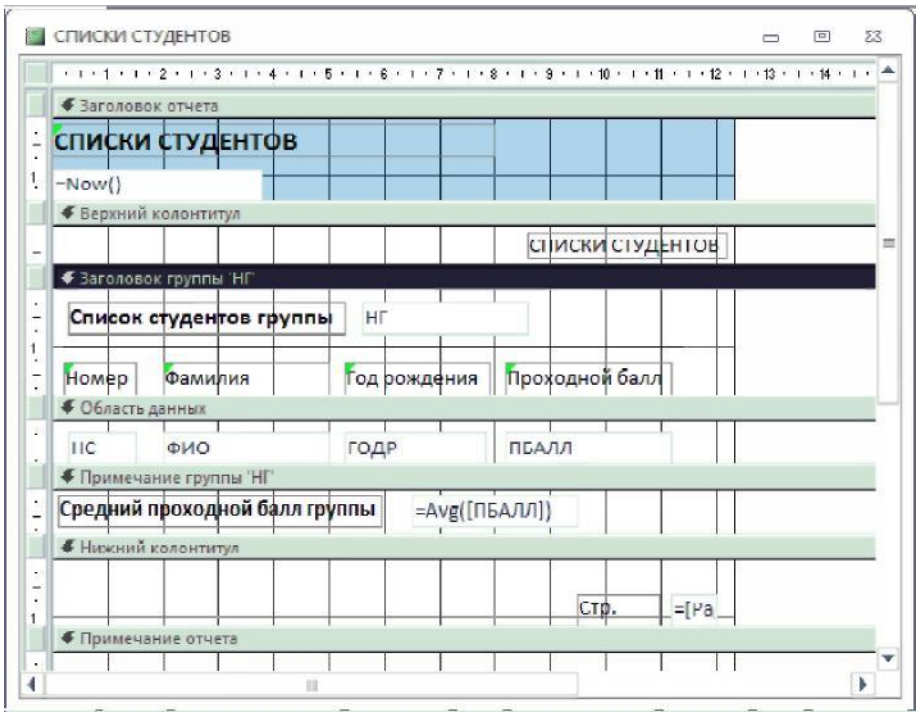
Поскольку общий список студентов в соответствии с проектом отчета должен быть разбит по группам, выберем в окне *сортировка и группировка* (Sorting and grouping) из списка поле номера группы НГ, зададим группировку по этому полю. Для этого в области *свойства группы* строках *заголовок группы* и *примечание группы* надо выбрать значения *с разделом*

заголовка, с разделом примечания. Сортировка для поля устанавливается автоматически.

Сортировка по полю

Для вывода отсортированного списка студентов в каждой группе, зададим сортировку по полю номера студента НС. Для этого в диалоговом окне выберем наряду с полем НГ поле НС. В области *свойства группы* этого поля в строках *ЗАГОЛОВОК ГРУППЫ* и *примечание группы* надо выбрать значения **б***ез раздела заголовка, без раздела примечания*, что и определяет сортировку только по этому полю.

После определения группировки в окне конструктора отчетов появляются дополнительные разделы *заголовок группы нг примечание группы нг***.**



Размещение полей из таблиц

Размещение поля группировки

Значение номера группы должно быть представлено один раз в заголовке группы. Для этого разместим поле НГ в разделе *заголовок группы нг***.**

Нажмем кнопку панели инструментов конструктора отчетов *добавить поля* и перетащим поле НГ в раздел заголовка НГ. Откорректируем подпись поля, изменив ее на “Список студентов группы”. Установим нужный шрифт в элементах. Для установки размеров рамки по размеру текста подписи выполним команду контекстного меню *размер|по размеру данных* или соответствующую кнопку панели инструментов.

Форматирование табличной части отчета

Последовательно разместим поля НС, ФИО, ДАТАР, ПБАЛЛ в области данных, которая определяет содержимое строк табличной части. Поле размещается вместе с подписью, которую система берет из свойств полей таблицы СТУДЕНТ. Подписи полей надо перенести в область заголовка путем вырезания и вставки. Если они не совпадают с названиями столбцов в проекте макета, их надо откорректировать. Заметим, что подписи также можно создать заново, воспользовавшись кнопкой панели элементов *надпись*.

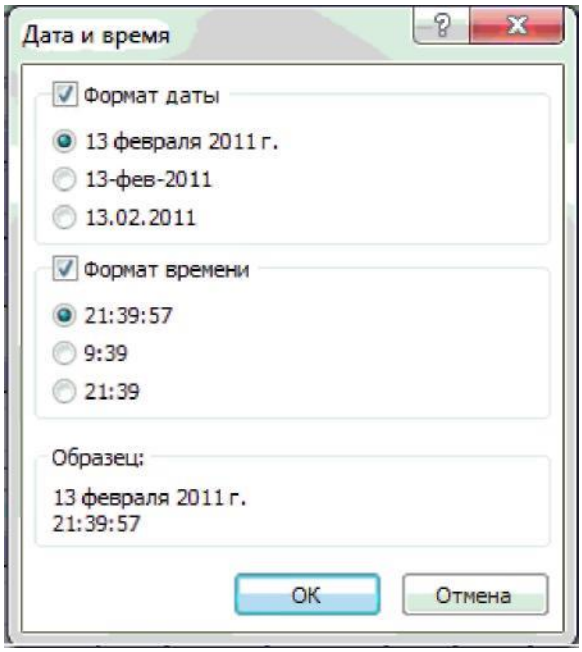
Для включения расчетного реквизита *средний проходной балл группы* нажмем кнопку *поле* на панели элементов и разместим элементы *свободный* в раздел *примечание группы нг*. Определим в свойствах этого элемента выражение для расчета среднего значения. Для этого запишем на вкладке *данные* в строку *данные* функцию =Avg ([ПБАЛЛ]), в строку *число десятичных знаков* – “2”, на вкладке *макет* в строку *формат поля* поместим значение “Фиксированный”. Отредактируем подпись поля. Для этого выделим подпись и вызовем ее свойства. В свойствах на вкладке *макет* в строке *подпись* запишем: “Средний проходной балл группы”. Такие действия, как изменение подписи или ввод выражения в поле можно выполнить, и не обращаясь к свойствам элементов.

Добавление текущей даты к странице

Для добавления в отчет *текущей даты* воспользуемся встроенной функцией Now. Для этого создадим в заголовке отчета свободный элемент, нажав кнопку *поле*, и зададим в окне его свойств на вкладке *данные* в строке *данные* выражение =Now. На вкладке *макет* в строке *формат поля* выберем значение *полный формат даты*. Подпись этого поля выделим и удалим.

Для добавления *номера страницы* в нижний колонтитул создадим свободный элемент и заполним в его свойствах на вкладке *данные* строку *данные* выражением =[Page]. Отредактируем подпись этого поля, записав в его свойствах на вкладке *макет* в строке *подпись* значение “Стр”.

Рассмотрим другие способы формирования поля даты и номера страницы. Поле текущей даты и времени можно добавить в отчет, выполнив в режиме конструктора команду *дата и время*. Установка в диалоговом окне *дата и время* флажков *формат даты* и /или *формат времени* позволяет вставить текущую дату и/или текущее время и выбрать нужный формат.



В отчет будет добавлено поле, в свойствах которого на вкладке *данные* в строке *данные* будет записано соответствующее выражение. Если в отчете имеется раздел заголовка, поле добавляется в этот раздел. В противном случае поле вносится в раздел данных. В качестве выражения записывается

функция Format, которая формирует значение на основе заданных ей аргументов – функции Date, возвращающей текущую системную дату, и формата, в котором должна выводится дата. Например, при выборе параметров, функция примет вид =Format(Date();”Long Date”).

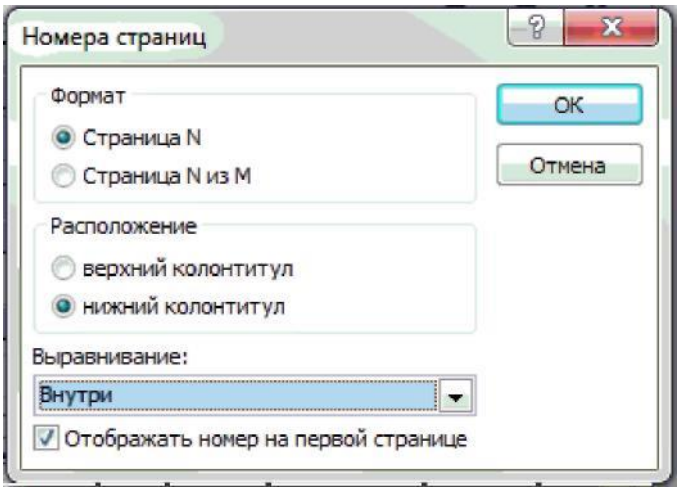
Поле нумерации страниц можно добавить в отчет, выполнив в режиме конструктора команду *номера страниц*. В окне диалога *номера страниц* выбираются параметры, определяющие формат, расположение и выраынивание номеров страниц. Для печати номера страницы на первой странице устанавливается флажок *отображать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

Замечание

Выражение, определяющее вывод номеров страниц, записывается в свойствах поля на вкладке *Данные* в строке *Данные*. Выражение может иметь вид: =”Страница”&[Page]

или:

=”Страница” & [Page] & ”из” & [Pages],что соответствует выбору *Страница N* или *Страница N из M*.



Завершение оформления отчета

Для окончательного оформления введем в раздел *заголовок отчета*, вставить раздел *верхний колонтитул* и выбрать нужный шрифт. Далее надо указать в свойствах отчета на вкладке *макет* в строке *верхний*

колонтитул: ”Без заголовка” . Свойства отчета могут быть вызваны при установке курсора на пересечении линеек.

Создадим линии в соответствии с макетом, воспользовавшись кнопкой панели элементов *линия*.

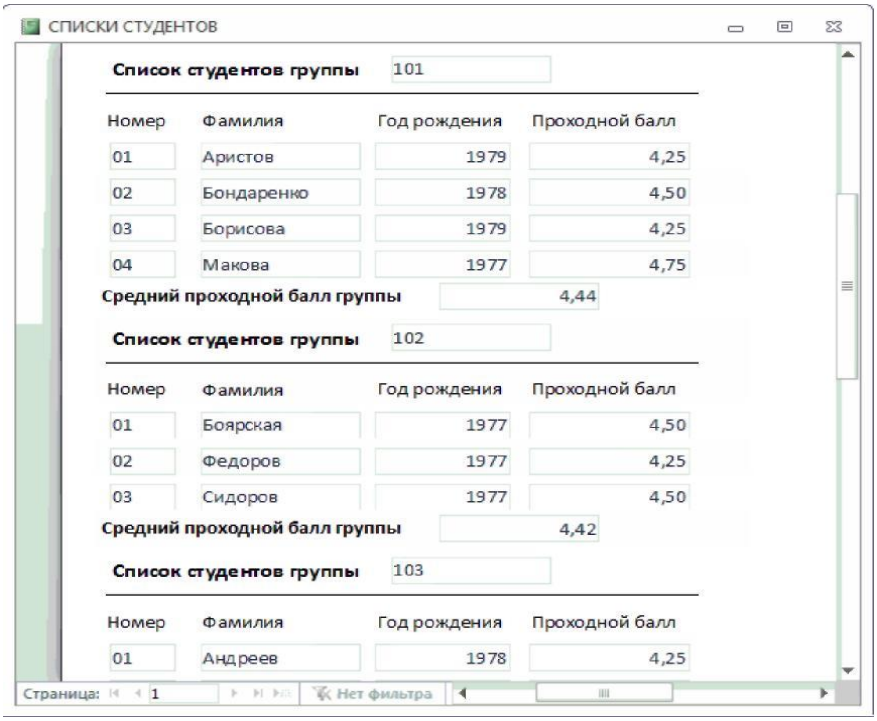
Просмотр и печать отчета

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

В режиме предварительного просмотра имеется своя панель инструментов.

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

Кнопка *печать* панели инструментов режима предварительного просмотра позволяет вывести отчет на печать.



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

Отчеты по двум таблицам

Наряду с однотабличными отчетами Access позволяет создавать более сложные отчеты, обеспечивающие вывод данных из нескольких взаимосвязанных данных в многотабличный отчет автоматически используются связи, установленные в схеме данных БД.

Многотабличные отчеты могут содержать основную часть и включаемую часть, т.е. подчиненный. Для каждой из этих частей в качестве источника данных выбирается своя таблица или несколько таблиц и при этом не иметь подчиненных отчетов.

Рассмотрим далее построение отчета для двух таблиц, находящихся в одно-многозначных отношениях. В этих отношениях одна таблица является главной, а другая – подчиненной. Построение такого отчета имеет свои особенности в зависимости от выбора главной или подчиненной таблицы в качестве подчиненного отчета.

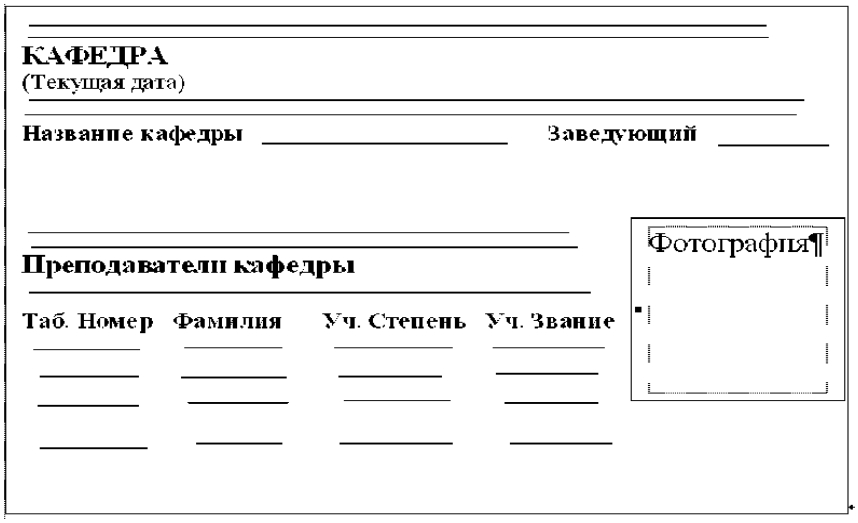
Многотабличный отчет с основной частью на базе главной таблицы

Рассмотрим технологию создания многотабличного отчета на основе таблиц КАФЕДРА и ПРЕПОДАВАТЕЛЬ. Пусть необходимо подготовить отчет, содержащий сведения о кафедрах и включающий списки преподавателей по кафедрам.

Проект макета отчета

В соответствии с проектом макета в отчет предполагается выводить данные по каждой кафедре, включая название, код и телефон, а также

фамилию и фотография заведующего. Эти данные содержатся в таблице КАФЕДРА. В табличной части по каждой кафедре необходимо вывести данные о преподавателях кафедре, которые содержатся в таблице ПРЕПОДАВАТЕЛЬ. Проект макета дает основание выбрать в качестве основной таблицы отчета таблицу КАФЕДРА, а таблицу ПРЕПОДАВАТЕЛЬ – в качестве источника данных для подчиненного отчета со списком преподавателей.

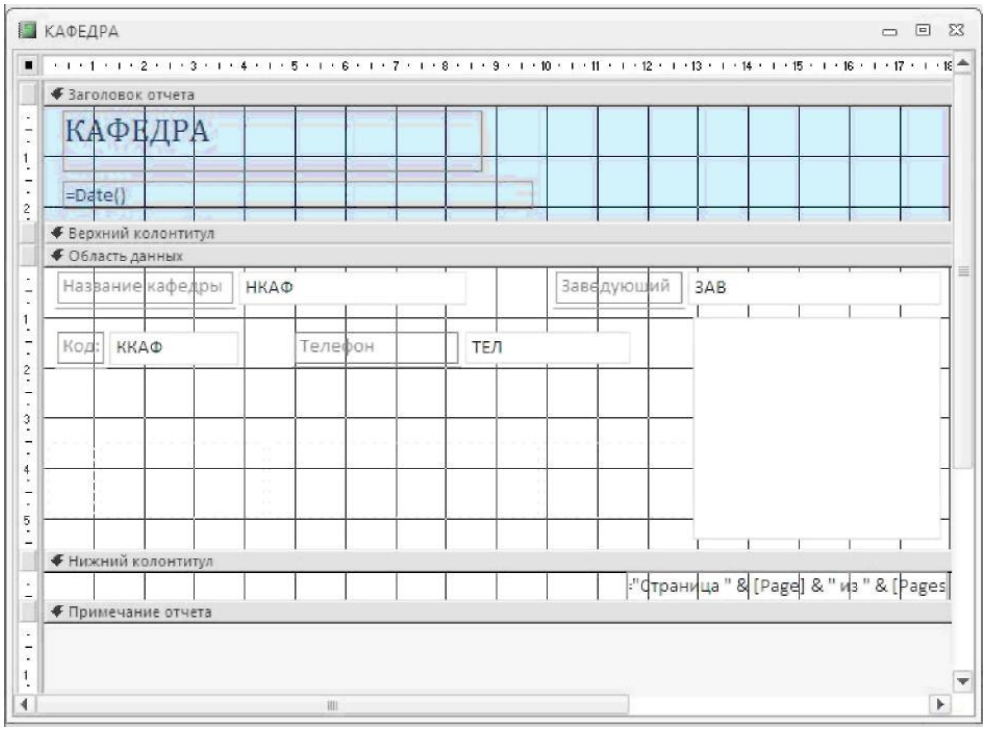


Создание основной части отчета

В окне базы данных *все объекты Access* выберем таблицу КАФЕДРА, которая будет источником данных для основной части отчета. Далее выберем вкладку *создание* и нажмем на кнопку *отчет*. Щелкаем правой кнопкой мыши по появившемуся отчету КАФЕДРА и выбираем *конструктор* для построения отчета в режиме конструктора.

В область данных перетащим поля таблицы КАФЕДРА: НКАФ, ККАФ, ТЕЛ, ЗАВ и ФОТО из верхнего колонтитула. Разместим поля и подписи, а также отредактируем их в соответствии с проектом макета.

Сохраним отчет под именем «Кафедры»



Разработка подчиненного отчета

Для вывода в отчет **Кафедры** списка преподавателей из таблицы ПРЕПОДАВАТЕЛЬ подготовим отдельный отчет, который будет включен в основную часть отчета в качестве подчиненного.

Создание автоотчета

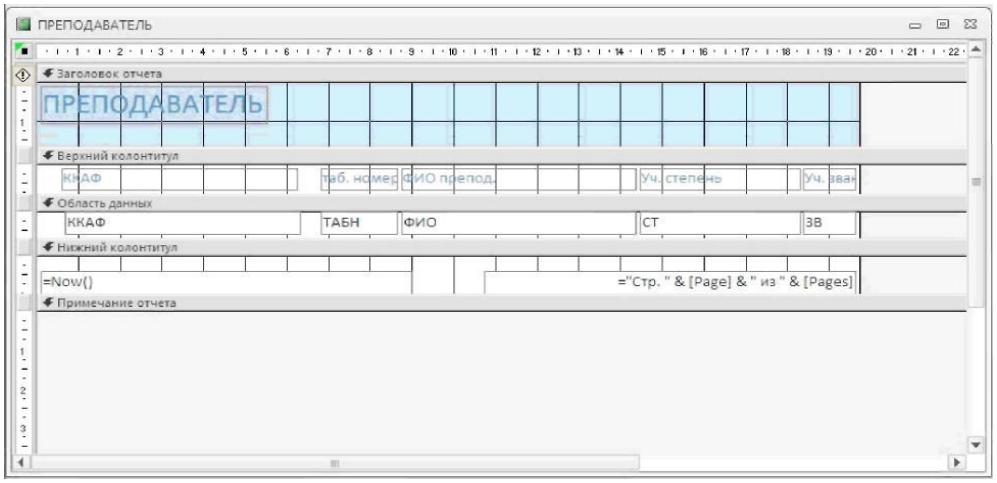
Подчиненный отчет создается как обычный однотабличный отчет. Для автоматического создания отчета во вкладке **Создание/Отчет** выберем опцию **Мастер отчетов.** В качестве источника данных этого отчета выберем таблицу ПРЕПОДАВАТЕЛЬ. Создание автоотчета завершается выводом на экран отчета, в котором в качестве заголовка фигурирует имя таблицы. В отчет включены все поля таблицы, а заголовками столбцов являются подписи этих полей, заданные в свойствах таблицы.

Доработка подчиненного отчета в режиме конструктора

Для того чтобы подчиненный отчет можно было включить в основную часть отчета в нужном виде в соответствии с проектом макета, доработаем его в режиме конструктора.

Верхний колонтитул, в котором после работы мастера оказались размещенными записи полей, и нижний колонктитул, где размещаются дата и номер страницы, не отобразятся при встраивании отчета как подчиненного.

Для отображения в полном отчете заголовков столбцов табличной части перенесем название столбцов из верхнего колонтитула в заголовок подчиненного отчета.

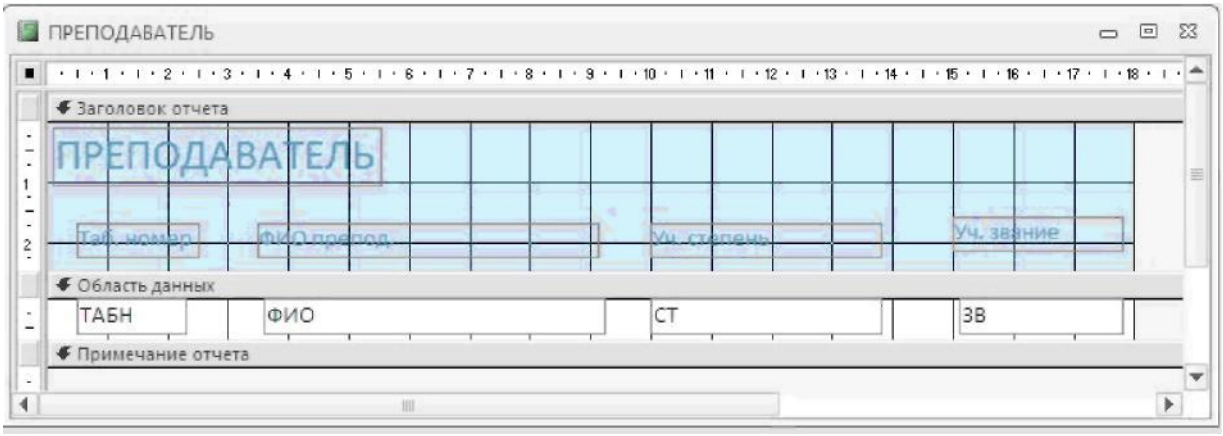


Для этого расширим раздел заголовка отчета, выделим все подписи полей в верхнем колонтитуле и перетащим их. Поскольку отчет будет использоваться только как подчиненный, можно удалить оба колонтитула.

Для этого нажмем правой кнопкой мыши на *верхний* или *нижний колонтитул* и выберем *колонтитулы страницы.* После этого в открывшемся окне нажимаем ДА. После этого оба колонтитула удалятся.

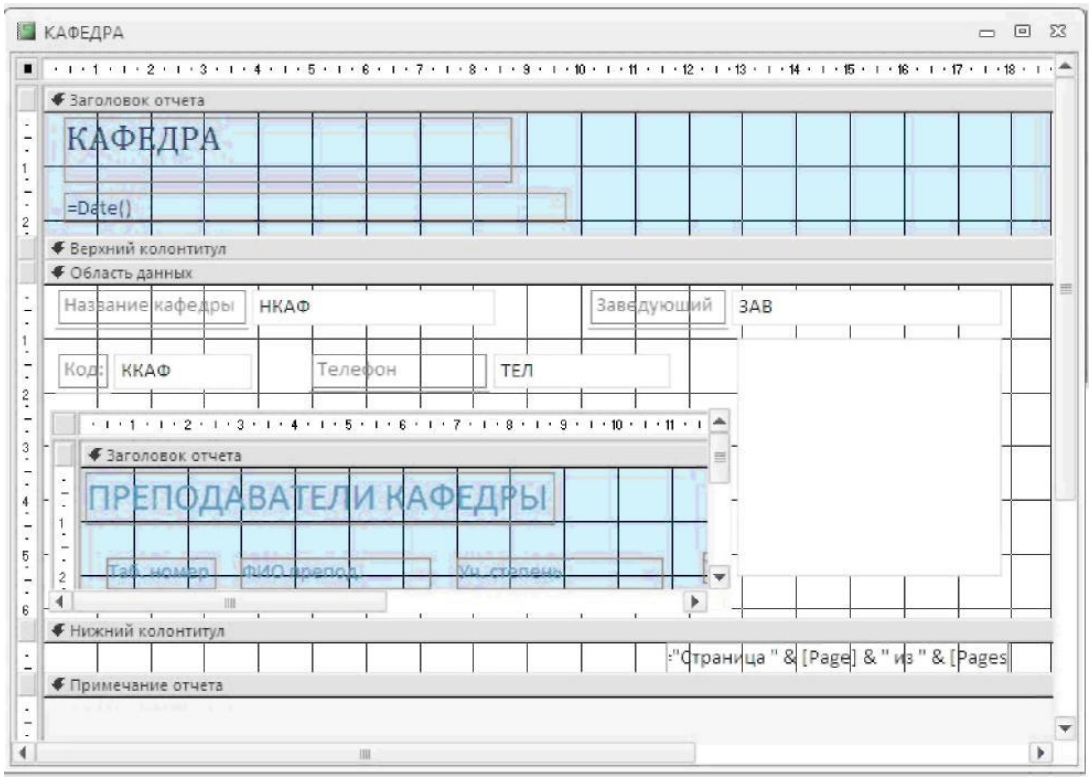
Удалим элемент поля ККАФ и его надпись «Код кафедры», т.к. в подчиненном отчете значения этого поля будут повторяться во всех строках о преподавателях, а однократное отображение кода кафедры предусмотрено в основной части отчета.

Окончательно сконструированный подчиненный отчет. Сохраним подчиненный отчет под именем «Преподаватели».



Включение подчиненного отчета

Воспользуемся самым простым способом включения подчиненного отчета в основной. Разместим на экране окно базы данных рядом с окном отчета *кафедры* в режиме конструктора. Перейдем в окно базы данных. Выберем из перечня имен отчетов подчиненный отчет *преподаватели* и перетащим его в область данных отчета *кафедры***.** Удалим элемент надписи подчиненного отчета, выделив его и нажав <Del>. Отчет *кафедры* после внедрения подчиненного отчета.



Подчиненный отчет отображен внутри отчета *кафедры***,** где доступен для редактирования, как и основной отчет. Для того чтобы подчиненный отчет при отображении не был взят в рамку, выделим его, как видно на этом рисунке, откроем его свойства и на вкладке *макет* в строке *тип границы* выберем значение ***отсутствует*.**

Просмотрим содержимое отчета *кафедры*, нажав на него 2 раза левой кнопкой мыши, и убедимся, что его вид соответствует проекту макета отчета.

Сортировка записей

Для вывода записей отчета в нужном порядке задается критерий сортировки. Для этого в режиме конструктора надо нажать кнопку **Итоги.** При необходимости сортировки и группировки записей подчиненного отчета необходимо выполнить для него такие же действия.

Многотабличные отчеты

Рассмотрим технологию разработки отчета, основным источником которого является подчиненная таблица, когда данные главных таблиц относительно этого источника тоже включаются в ответ. Мастер отчетов позволяет построить многотабличный отчет для взаимосвязанных таблиц, выбрать из них нужные поля в заданной последовательности и указать, какая таблица из участвующих в отчете, будет записи образующей, т.е. основным источником данных. Кроме того, мастер предоставляет возможность определить группировку и сортировку записей отчета по различным полям, подсчитать итоговые значения.

Пусть необходимо получить отчет, в котором выводятся в виде списка данные о занятиях, проводимых в каждой группе. Строки отчета должны быть упорядочены по коду предмета. При выводе данных в отчете должны также формироваться расчетные суммарные часы по занятиям для каждой группы.

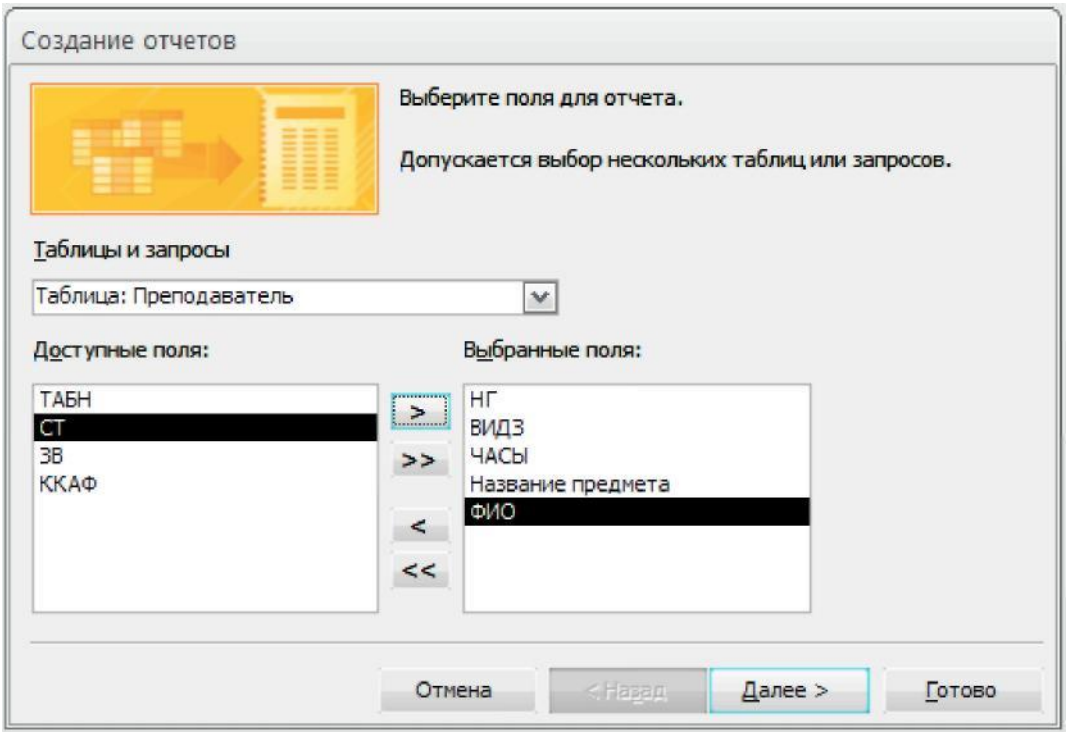
Проект макета отчета, который должен быть создан для вывода данных о занятиях, проводимых в каждой группе.

Выбор таблиц для отчета и варианта его создания.

Основные сведения о занятиях, проводимых в группах, содержит подчиненная таблица ИЗУЧЕНИЕ. Из таблицы ИЗУЧЕНИЕ можно получить перечень идентификаторов занятий, проводимых в каждой группе, с указанием часов по занятию. Эта таблица является основным источником записей для создаваемого отчета. Наименования предметов содержит таблица ПРЕДМЕТ, а фамилии преподавателей – таблица ПРЕПОДАВАТЕЛЬ. Обе эти таблицы являются главными по отношению к таблице ИЗУЧЕНИЕ. Одно-многозначные связи между этими таблицами и подчиненной таблицей ИЗУЧЕНИЕ установлены в схеме данных базы по соответствующим идентификаторам занятия: коду предмета и табельному номеру преподавателя.

Начиная создание отчета во вкладке *создание***,** выберем *мастер отчетов***.**

В окне *создание отчетов* выберем из таблицы ИЗУЧЕНИЕ поля, включаемые в отчет: НГ, ВИДЗ, ЧАСЫ.



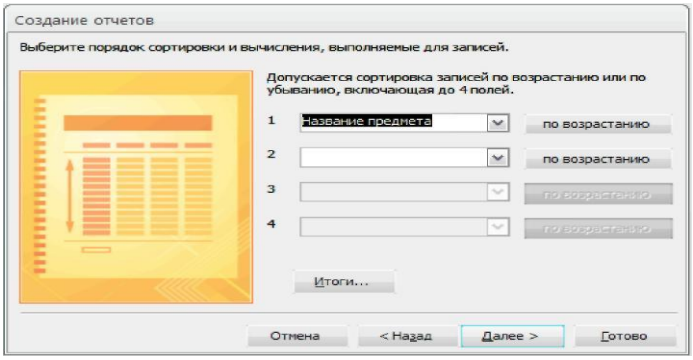
Из таблицы ПРЕДМЕТ выберем поле с наименованием предмета НП, а из таблицы ПРЕПОДАВАТЕЛЬ поле ФИО.

Замечание

Для того чтобы включать поля в нужной последовательности, следует иметь в виду, что поле вставляется в след за выделенным в списке полей уже включенных в ответ. Например, для включения поля НП сразу за полем НГ последнее должно быть предварительно выделено.

В следующем сеансе окна мастера в строке *выберите тип представления данных*. Далее задаются уровни группировки, которые позволяют вывести записи, объединенные по разным полям. Зададим один уровень группировки по полю НГ. При этом в окне отображается общая структура формируемого макета отчета.

В следующем сеансе диалогового окна мастера *создание отчета* задается порядок сортировки записей.

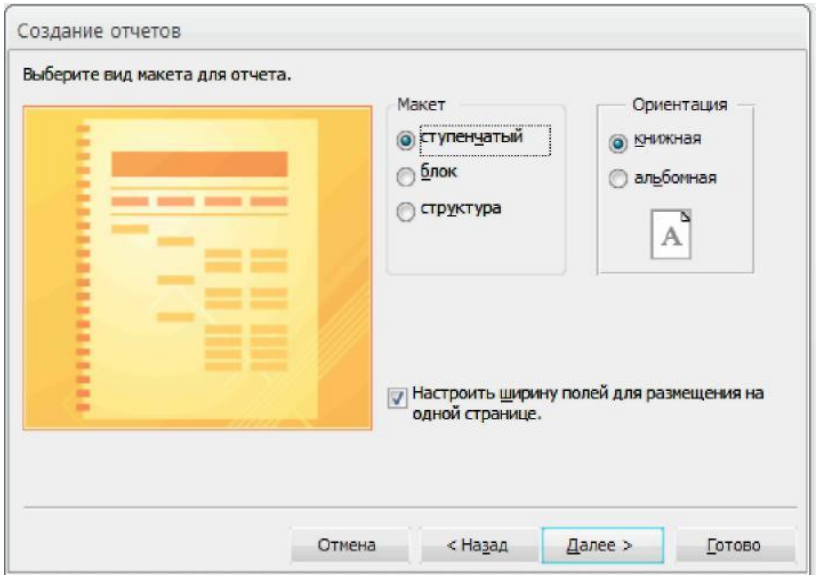


Мастер позволяет производить сортировку по четырем полям в порядке возрастания или убывания значений поля. Выберем поле НП (Название предмета), по которому нужно произвести сортировку по возрастанию. Чтобы произвести подсчет итоговых значений, нажмем кнопку *итоги***.** Для числовых полей в открывшемся окне *итоги* будут выведены строки, в которых можно выбрать статистическую функцию (Sum, Avg, Min, Max) для подсчета значений в итоговой строке группы. В соответствии с проектом макета отчета должно быть выведено суммарное число часов в поле отчета *итого по группе***.** Поэтому выберем для поля ЧАСЫ функцию Sum.

Замечание

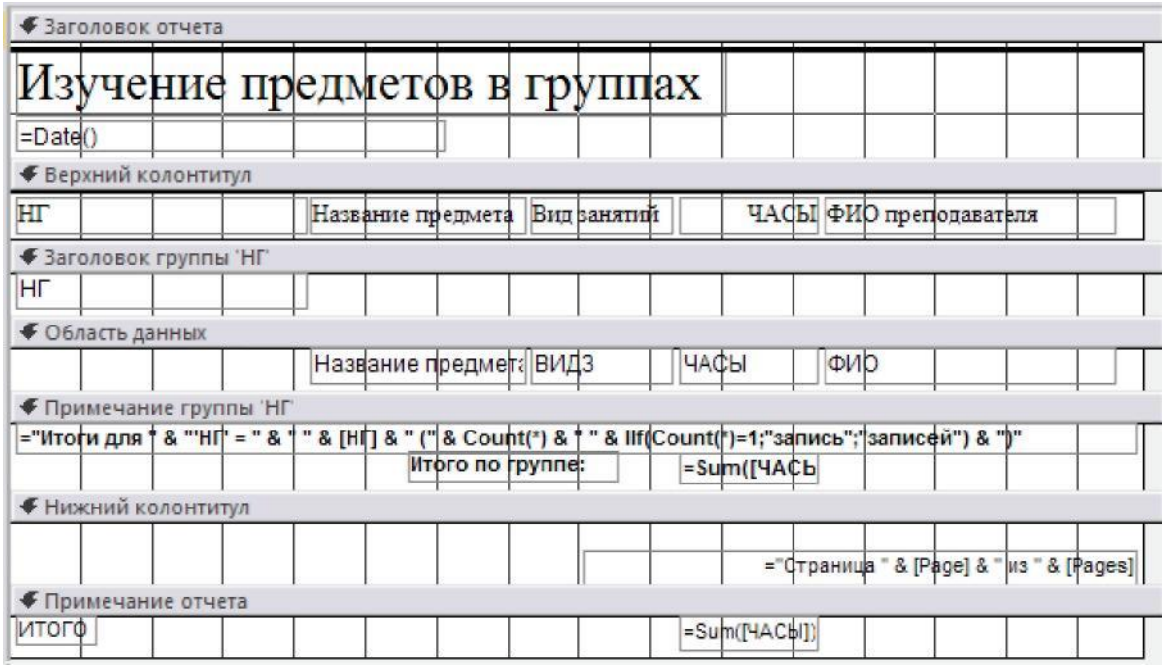
Если необходимо подсчитать долю суммарных часов группы от общих часов всех групп, нужно отметить флажок *вычислить проценты*.

Далее в следующем сеансе окна мастера выберем из шести предлагаемых видов макета отчета *ступенчатый* с *книжной* ориентацией. Отметим флажок *настроить ширину полей для размещения на одной странице***.**

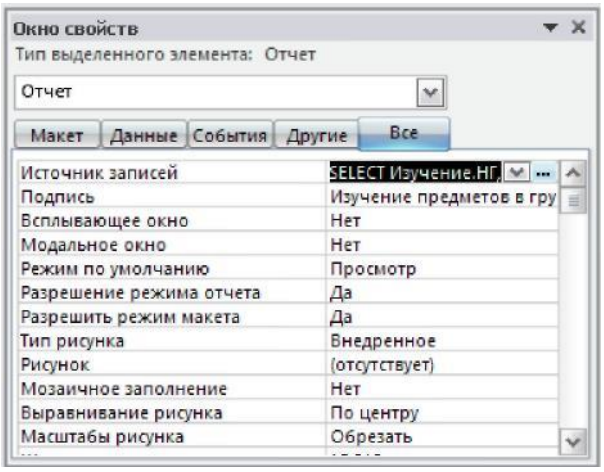


Потом зададим имя отчета – «Изучение предметов в группах», которое также отобразиться в заголовке отчета. Под этим именем мастер автоматически сохраняет отчет в базе данных. Для того чтобы сразу доработать отчет, выберем дальнейшие действия – *изменить макет отчета***.** Отчет отобразиться на экране в режиме конструктора.

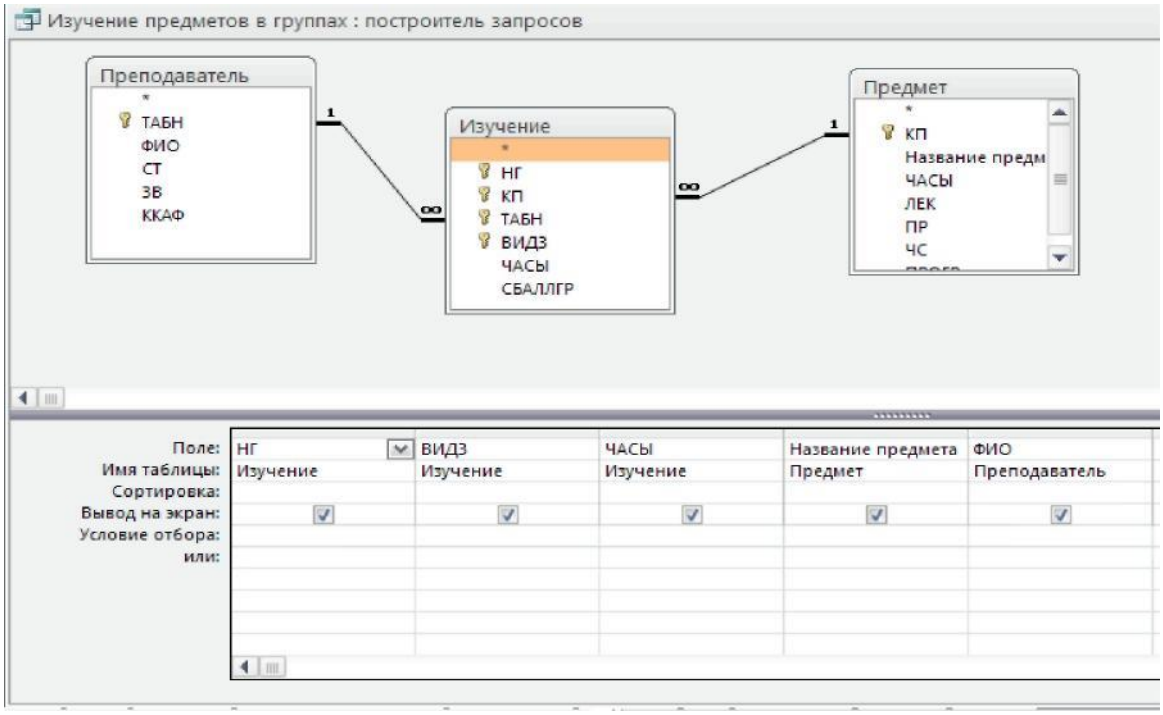
Этот отчет по основным параметрам соответствует проекту макета отчета. В него включены поля из трех взаимосвязанных таблиц. Причем пользователь не потребовалось задавать связи между таблицами и включать в отчет поля КП (код предмета) и ТАБН (номер преподавателя), являющиеся полями связи.



Отметим, что на основе информации, сохраняемой в схеме данных, и заданных пользователем полей при создании макета отчета, мастер сам строит необходимый запрос. По этому запросу формируются записи из полей нескольких взаимосвязанных таблиц. В свойствах отчета в качестве источника записей мастер записывает инструкцию SQL, реализующую запрос и определяющую выборку заданных полей из различных таблиц.



Запрос, созданный мастером при подготовке макета отчета, можно просмотреть и при необходимости откорректировать. Чтобы отобразить запрос на экране, нужно в диалоговом окне свойств отчета в строке *источник записей* нажать кнопку *построитель***,** которая вызовет построитель запросов. Открывающееся окно построителя запросов



Редактирование макета отчета в режиме конструктора.

В отчет, построенный мастером), в заголовок группы автоматически было включено поле номера группы НГ. В примечание группы мастером включены итоги по группе: поле для подсчета числа записей в группе и поле для подсчета суммы часов по группам. Отредактируем надписи этих полей.

Поместим поле с выражением =Date(), определяющим текущую дату, в заголовок отчета.

Элементы заголовков столбцов размещены в верхнем колонтитуле отчета и, следовательно, будут печататься на каждой странице отчета. Чтобы заголовки печатались в каждой группе, как предусмотрено в проекте макета отчета , переместим их в заголовок группы.

Если необходимо, чтобы сведения о каждой группе печатались на отдельной странице, вставьте в примечание группы разрыв страницы. Для этого воспользуйтесь кнопкой *разрыв страницы* на панели *конструктора отчетов*.

Отредактируем текст заголовков в соответствии с проектом макета.

Замечание

В отчетах можно вывести значения некоторого поля записи или итогового поля группировки нарастающим итогом. Например, можно накапливать сумму часов от группы к группе. Так, если в первой группе суммарное число часов равно 262, во второй – 150, а в третьей – 130, то задав свойство поля *сумма с накоплением = для всего*, можно получить значения: для первой группы 262, для второй 412, для третьей 542. Чтобы накапливать сумму значений поля для записей в группе, нужно установить свойство *сумма с накоплением = для группы***.** Это свойство размещено на вкладке *данные.*

Разработка отчета на основе запроса.

Запрос является мощным и удобным средством выборки взаимосвязанных данных. Поэтому с помощью запроса можно подготовить данные для сложного отчета.

Рассмотрим технологию создания сложного отчета с использованием запроса на примере формирования бланка “Экзаменационная ведомость” для группы студентов по заданному предмету. Бланк должен иметь форму документа, используемого для ввода данных после внесения оценок преподавателем в этот бланк.

Проект макета отчета. Бланк “Экзаменационная ведомость”, которая должна выводиться из базы данных в режиме просмотра и печати отчета.

Экзаменационная ведомость

Название предмета Группа

Преподаватель

Вид сдачи 29-май-03

N п/п	Фамилия И.О.	Отметка о сдаче	Подпись преподавателя

Итого: Отлично_____

Хорошо_____

Удовлетворительно_____

Неудовлетворительно_____

Неявка_____

Подпись преподавателя _____

Рассмотрим подготовку запроса, обеспечивающего выборку информации, необходимой для формирования бланка экзаменационной ведомости.

Создание запроса для подготовки данных в отчет

Определение схемы данных запроса

Для вывода в отчет реквизитов, указанных в проекте макета нужно определить таблицы – источники и их взаимосвязи.

Таблицы источники данных

Для вывода в шапку бланка экзаменационной ведомости значений реквизитов *вид сдачи*, *группа* необходимы данные из полей ВИДЗ, НГ таблицы ИЗУЧЕНИЕ. Для вывода значений реквизитов с общей надписью *преподаватель (фамилия, ученое звание)* необходимы данные из полей ФИО, ЗВ таблицы ПРЕПОДАВАТЕЛЬ. Для вывода реквизита *название предмета* необходимы данные из поля НП таблицы ПРЕДМЕТ.

Для вывода в табличную часть бланка экзаменационной ведомости значений реквизитов *n n/n, фамилия и.о.* необходимы данные из полей НС, ФИО таблицы СТУДЕНТ. Таким образом, запрос для выборки этих

взаимосвязанных данных должен быть построен на основе таблицы ИЗУЧЕНИЕ, ПРЕПОДАВАТЕЛЬ, ПРЕДМЕТ, СТУДЕНТ.

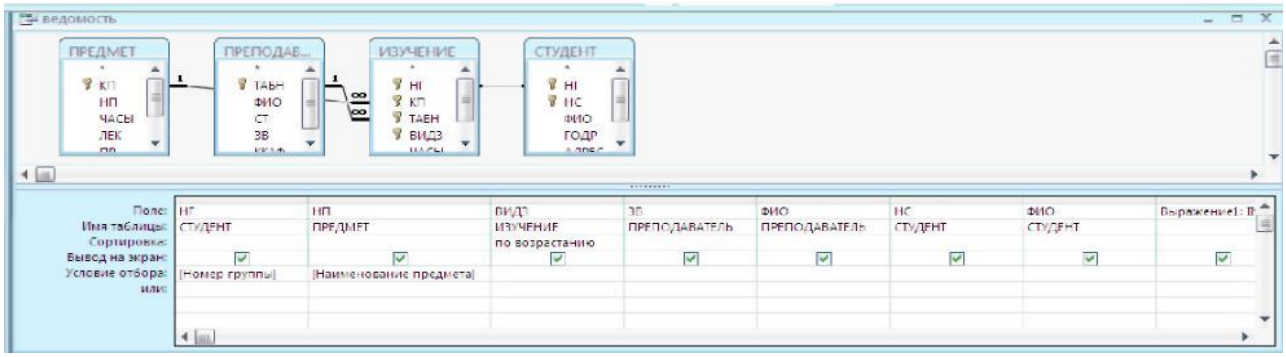
Связи между таблицами запроса

При создании запроса связи между таблицами установятся автоматически. Связи таблиц ИЗУЧЕНИЕ, ПРЕДМЕТ, ПРЕПОДАВАТЕЛЬ определяются в соответствии со схемой данных БД.

При создании запроса между таблицами СТУДЕНТ и ИЗУЧЕНИЕ автоматически установится также связь по одноименному полю НГ (номер группы). Эта связь является *связью – объединением*, которой нет в схеме данных базы. Заметим, что эти таблицы находятся в отношениях многие-ко-многим, поскольку один студент изучает много предметов, и один предмет изучается многими студентами. Связь, установленная между таблицами СТУДЕНТ и ИЗУЧЕНИЕ, определяет операцию симметричного объединения. При этом записи из этих таблиц объединяются и добавляются в результат только в том случае, если связанные поля содержат одинаковые значения.

Конструирование запроса для подготовки макета отчета.

Процесс включения в запрос необходимых таблиц и полей из них является достаточно простым и выполняется по технологии. В строке бланка запроса *условие отбора* определим параметры запроса |Номер группы| и |Наименование предмета| для аналогового ввода их значений при выполнении запроса. Это позволяет получить данные для конкретной ведомости.



В отчете необходимо выводить значения реквизита *вид сдачи: экзамен, зачет,* которых нет непосредственно в таблицах БД, но они могут быть получены на основе значений поля ВИДЗ таблицы ИЗУЧЕНИЕ. Два значения реквизита *вид сдачи***:** ”Экзамен” и ”Зачет” соответствуют двум возможным значениям поля ВИДЗ: ”лек” и ”пр”. Фактически нужно вместо значения ”лек” формировать слово ”Экзамен”, а вместо ”пр” – слово “Зачет”. Для этого надо в запрос ввести новое поле, которое формируется как вычисляемое. Это поле должно содержать встроенную функцию управления.

Выражение 1 : **Iif** ([ВИДЗ] = «лек»; «Экзамен»; «Зачет»)

Именно этого вычисляемого поля по умолчанию является ”Выражение 1:” Функция Iif (”immediate if” – мгновенное условие) аналогична инструкции if… Then… Else и имеет следующий формат: **Iif** (условие; если Истина; если Ложь)

В соответствии с форматом этой функции, если выполнится условие (ВИД)= «лек» (т.е. в поле ВИДЗ находится значение «лек»), ТО РЕЗУЛЬТАТОМ ФУНКЦИИ БУДЕТ «Экзамен». В противном случае, т.е. если (ВИДЗ) = «пр» результатом функции будет слово «Зачет».

В предыдущем рисунке приведен в окончательном виде запрос для подготовки данных отчета-бланка экзаменационной ведомости. Этот запрос сохранен под именем «Ведомость».

Формирование записей результатов в запросе.

В процессе формирования записей результата из таблицы СТУДЕНТ последовательно выбираются записи с *заданным номером группы* (НГ).

С каждой записью таблицы СТУДЕНТ объединяются связанные по номеру группы НГ записи из таблицы ИЗУЧЕНИЕ. Из таблицы ИЗУЧЕНИЕ в этом объединении участвуют только записи с *заданным наименованием предмета.*

Дополнение формируемых запросом записей полями из таблицы ПРЕДМЕТ и ПРЕПОДАВАТЕЛЬ не добавляет новых записей. Эти таблицы

представляют сторону *один* в отношении *один-ко-многим* и служат лишь для расшифровки кодов КП и ТАБН из таблицы ИЗУЧЕНИЕ.

Если по заданному предмету проводится не одно, а два вида занятий: лекции и практика, то число формируемых записей удваивается. В этом случае должны быть сформированы две ведомости – одна для сдачи экзамена, другая для сдачи зачета. Заметим, что таблица результатов запроса содержит ненормализованные данные, что проявляется в повторяемости значений в столбцах ГРУППА, НАИМЕНОВАНИЕ ПРЕДМЕТА, ФИО ПРЕПОД., УЧ. ЗВАНИЕ, ВИД ЗАНЯТИЙ и ВЫРАЖЕНИЕ1. Поэтому данная таблица не соответствует проекту макета отчета Экзамена ведомость не может непосредственно использоваться как выходной документ.

Использование средств Access для подготовки отчета обеспечивает преобразование получаемой таблицы результатов запроса в выходной документ нужной формы в соответствии с проектом макета отчета.

ведомость

Группа	Название пр	вид занятий	уч.звание	ФИО препо	Номер студ	ФИО	Выражение1
101	информатика	лек	ассистент	Новиков П.Н.	04	Макова Н.В.	Экзамен
101	информатика	лек	ассистент	Новиков П.Н.	03	Борисова Е.И.	Экзамен
101	информатика	лек	ассистент	Новиков П.Н.	02	Бондаренко С	Экзамен
101	информатика	лек	ассистент	Новиков П.Н.	01	Аристов Р.П.	Экзамен
101	информатика	пр	доцент	Апухтин И.С.	04	Макова Н.В.	Зачет
101	информатика	пр	доцент	Апухтин И.С.	03	Борисова Е.И.	Зачет
101	информатика	пр	доцент	Апухтин И.С.	02	Бондаренко С	Зачет
101	информатика	пр	доцент	Апухтин И.С.	01	Аристов Р.П.	Зачет

Конструирование отчета на основе запроса

Рассмотрим технологию отчета на основе запроса *ведомость* подготовленного выше. Для конструирования отчета во вкладке *создание* в группе *отчеты* нажмем кнопку *конструктор отчетов***.** В окне *новый отчет* выберем запрос *ведомость*, который будет источником данных для отчета.

Размещение данных в разделах отчета

Результаты запроса содержат много полей с повторяющимися значениями НГ, НП, ФИО преподавателя, ЗВ, ВИДЗ, Выражение 1: Данные в отчете должны быть размещены, как показано на макете отчета.

Экзаменационная ведомость

Значение каждого из этих полей должно быть представлено в отчете в заголовке один раз.

Заметим, что результаты для отчета получены по заданной группе и предмету. Поскольку по предмету может быть два вида занятий и для каждого вида занятий предполагается вывод отдельной ведомости, необходимо выполнить группировку по полю ВИДЗ. Для группировки по виду занятия нажмем кнопку *сортировка и группировка* на панели инструментов конструктора отчетов и заполним поля открывшегося окна.

После определения группировки в окне конструктора отчета появятся разделы *заголовок группы* «ВИДЗ» и *примечание группы* «ВИДЗ» .

Чтобы каждая группировка могла быть оформлена как самостоятельная экзаменационная ведомость, будем формировать шапку макета ведомости в заголовке группы ВИДЗ, а строки о подведении итогов сдачи экзамена (или зачета) в примечании группы ВИДЗ.

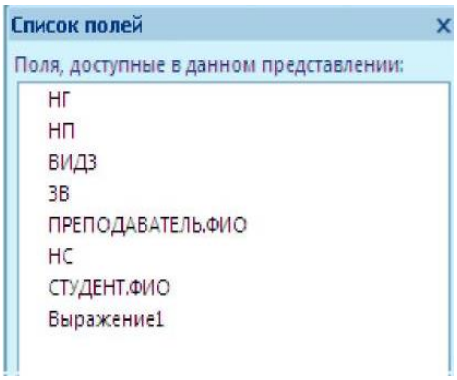
Создание текстовых элементов отчета.

Для создания в заголовке и примечании группы ВИДЗ текстовых элементов за исключением названий реквизитов выводимых полей, используем кнопку *надпись*. Для оформления текста можно воспользоваться кнопками панели форматирования. Кроме того, можно задавать параметры надписей в окне свойств этих элементов. Окно свойств элемента открывается кнопками панели инструментов *свойства*. Элемент предварительно должен быть выделен.

При размещении элементов может понадобиться расширить раздел отчета. Для этого установить курсор мыши на его нижнюю границу так, чтобы появилась двунаправленная стрелка, и переместить эту стрелку вниз.

Оформление табличной части отчета.

Для размещения в отчете полей вызовем окно списка полей запроса *ведомость*, нажав на панели конструктора отчетов кнопку *добавить поля***.**



В область данных разместим поля НС и ФИО, на основе содержимого которых будут формироваться строки табличной части отчета в соответствии с проектом макета отчета. Для этого из списка полей перетащим поля НС (номер студента) и СТУДЕНТ. ФИО в область данных.

Каждое поле представляется двумя элементами: собственно, полем и его подписью. Например, поле ФИО таблицы СТУДЕНТ представляется элементом СТУДЕНТ. ФИО, отображающим значение поля, и элементом, содержащим подпись этого поля из таблицы базы данных - Фамилия И.О. Установим нужный шрифт в элементах. Используем команду меню *формат* |*размер*| *по размеру данных* для установки размеров элемента по размеру текста подписи.

Замети, что подпись поля из табличной части должна быть перемещена для формирования заголовка столбцов в раздел *заголовок группы* «ВИДЗ». Для этого выделим подпись поля. Затем вырежем его. Активизируем раздел *заголовок группы* «ВИДЗ» и вставим подпись в этот раздел.

Оформим элементы в соответствии с требованиями проекта макета отчета. В частности, заменим подпись поля «Номер студента на N п/п». Для того чтобы в бланке экзаменационной ведомости были пустые столбцы для оценок и подписи (заполняемые преподавателем вручную на экзамене), добавим в строку с заголовками столбцов новые текстовые элементы «Отметка о сдаче» и «Подпись преподавателя».

Размещение полей в заголовке

Значения полей НГ, НП, ФИО преподавателя, ЗВ, Выражение 1 (экзамен, зачет) повторяются во всех записях группировки. Поэтому они должны размещаться в заголовке группы. Перетащим эти поля из списка полей в раздел *заголовок группы* «ВИДЗ».

Добавление текущей даты и номера страницы

Для добавления в отчет текущей даты воспользуемся встроенной функцией Now (). Для этого создадим несвязанный элемент, нажав кнопку *поле* на панели элементов и разместим его в нужном месте. Зададим в окне его свойств на вкладке *данные* в строке *данные* выражение = Now (), а на вкладке *макет* в строке *формат поля* выберем значение – *средний формат даты***.**

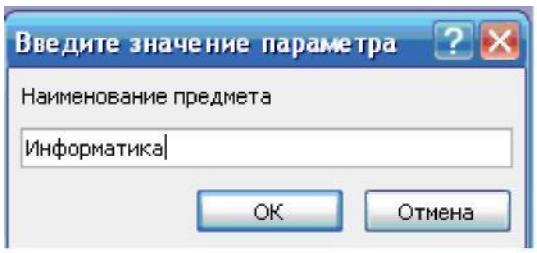
Для добавления номера страницы в раздел *нижний колонтитул* создадим несвязанный элемент и заполним в его свойствах строку *данные* выражением = «Страница» @ [Page]. Создадим горизонтальные и вертикальные линии в соответствии с макетом, воспользовавшись кнопкой *линии*. Установим нужную толщину линий в свойствах этого элемента. Сохраним отчет под именем «Экзаменационная ведомость».

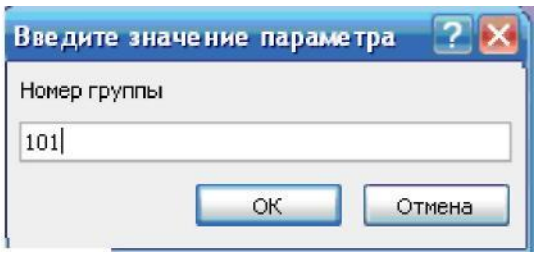
Предварительный просмотр отчета.

При подготовке к просмотру отчета, построенного на базе запроса с параметрами, Acceess предварительно выполняет запрос и выводит диалоговые окна ввода параметров отчета.

Просмотр отчета

Для отображения отчета *экзаменационная ведомость* на экране в том виде, в котором он будет напечатан, выйдем из режима конструктора отчетов, нажав кнопку *вид*. Последовательно появляются диалоговые окна для ввода параметров запроса, которые являются в то же время параметрами отчета.





Введем значение параметра «Номер группы» – «101» и значение параметра «Наименование предмета» – «Информатика». Для данных значений параметров отчет будет состоять из двух страниц, на первой из которых будет представлена экзаменационная ведомость для экзамена по информатике, на второй – для сдачи зачета.

2.6. Язык структурированных запросов SQL

Одним из языков, появившихся в результате разработки реляционной модели данных, является Structured Query Language (SQL), который в настоящее время получил очень широкое распространение и фактически превратился в стандартный язык реляционных баз данных. SQL является примером языка преобразования данных, или же языка, предназначенного для работы с таблицами с целью преобразования входных данных к требуемому выходному виду.

Язык SQL, который определен стандартом ISO, имеет два основных компонента;

• язык DDL (Data Definition Language), предназначенный для определения структур базы данных и управления доступом к данным;

• язык DML (Data Manipulation Language), предназначенный для выборки и обновления данных.

Язык SQL – это непроцедурный язык, поэтому в нем необходимо указывать, какая информация должна быть получена, а не как ее можно получить. Иначе говоря, язык SQL не требует указания методов доступа к данным.

Структура команд задается набором ключевых слов, представляющих собой обычные слова английского языка, такие как CREATE TABLE

(Создать таблицу), INSERT (Вставить), SELECT (Выбрать), UPDATE (Обновить), Delete (удалить).

В настоящее время для языка SQL существуют международные стандарты, формально определяющие его как стандартный язык создания и манипулирования реляционными базами данных, каковым он фактически и является.

Преимущества SQL.

Язык SQL является основой многих СУБД, т.к. отвечает за физическое структурирование и запись данных на диск, а также за чтение данных с диска, позволяет принимать SQL-запросы от других компонентов СУБД и пользовательских приложений. Таким образом, SQL – мощный инструмент, который обеспечивает пользователям, программам и вычислительным системам доступ к информации, содержащейся в реляционных базах данных.

Основные достоинства языка SQL заключаются в следующем:

- стандартность – как уже было сказано, использование языка SQL в программах стандартизировано международными организациями;

- независимость от конкретных СУБД – все распространенные СУБД используют SQL, т.к. реляционную базу данных можно перенести с одной СУБД на другую с минимальными доработками;

- возможность переноса с одной вычислительной системы на другую – СУБД может быть ориентирована на различные вычислительные системы, однако приложения, созданные с помощью SQL, допускают использование как для локальных БД, так и для крупных многопользовательских систем;

- реляционная основа языка – SQL является языком реляционных БД, поэтому он стал популярным тогда, когда получила широкое распространение реляционная модель представления данных. Табличная структура реляционной БД хорошо понятна, а потому язык SQL прост для изучения;

- возможность создания интерактивных запросов – SQL обеспечивает пользователям немедленный доступ к данным, при этом в интерактивном

режиме можно получить результат запроса за очень короткое время без написания сложной программы;

- возможность программного доступа к БД – язык SQL легко использовать в приложениях, которым необходимо обращаться к базам данных. Одни и те же операторы SQL употребляются как для интерактивного, так и программного доступа, поэтому части программ, содержащие обращение к БД, можно вначале проверить в интерактивном режиме, а затем встраивать в программу;

- обеспечение различного представления данных – с помощью SQL можно представить такую структуру данных, что тот или иной пользователь будет видеть различные их представления. Кроме того, данные из разных частей БД могут быть скомбинированы и представлены в виде одной простой таблицы, а значит, представления пригодны для усиления защиты БД и ее настройки под конкретные требования отдельных пользователей;

- возможность динамического изменения и расширения структуры БД – язык SQL позволяет манипулировать структурой БД, тем самым обеспечивая гибкость с точки зрения приспособленности БД к изменяющимся требованиям предметной области;

- поддержка архитектуры клиент-сервер – SQL – одно из лучших средств для реализации приложений на платформе клиент-сервер. SQL служит связующим звеном между взаимодействующей с пользователем клиентской системой и серверной системой, управляющей БД, позволяя каждой из них сосредоточиться на выполнении своих функций.

2.7. Операторы SQL

Создание новой таблицы

Инструкция CREATE TABLE создает новую таблицу и используется для описания ее полей и индексов. Если для поля добавлено ограничение NOT NULL, то при добавлении новых записей это поле должно содержать допустимые данные. Синтаксис:

CREATE TABLE таблица (поле_1 тип [(размер)]

[NOT NULL] [индекс_1] [, поле_2 тип [(размер)]

[NOT NULL] [индекс_2] [, ...]] [, CONSTRAINT составной Индекс

[, ...]]),

где таблица — имя создаваемой таблицы;

поле_1, поле_2 — имена одного или нескольких полей, создаваемых в новой таблице. Таблица должна содержать хотя бы одно поле;

тип — тип данных поля в новой таблице;

размер — размер поля в символах (только для текстовых и двоичных полей);

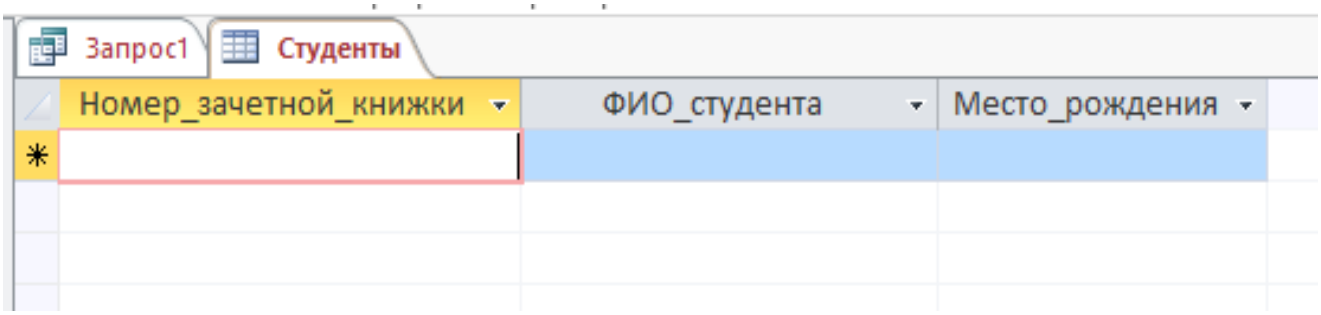
индекс_1, индекс_2 — предложение CONSTRAINT, предназначенное для создания простого индекса;

составной Индекс — предложение CONSTRAINT, предназначенное для создания составного индекса.

В следующем примере создается новая таблица с двумя полями:

CREATE TABLE Студенты (Номер_зачетной_книжки integer PRYMARY KEY, ФИО_студента TEXT (50), Место_рождения TEXT (50));

В результате выполнения этого запроса будет создана таблица со следующей схемой:



Оператор Select.

Оператор SELECT – один из наиболее важных и самых распространенных операторов SQL. Он позволяет производить выборки данных из таблиц и преобразовывать к нужному виду полученные

результаты. Будучи очень мощным, он способен выполнять действия, эквивалентные операторам реляционной алгебры, причем в пределах единственной выполняемой команды. При его помощи можно реализовать сложные и громоздкие условия отбора данных из различных таблиц.

Оператор SELECT – средство, которое полностью абстрагировано от вопросов представления данных, что помогает сконцентрировать внимание на проблемах доступа к данным. Примеры его использования наглядно демонстрируют один из основополагающих принципов больших (промышленных) СУБД: средства хранения данных и доступа к ним отделены от средств представления данных. Операции над данными производятся в масштабе наборов данных, а не отдельных записей.

Оператор SELECT имеет следующий формат:

```
SELECT [ALL | DISTINCT ] {*|[имя_столбца
  [AS    новое_имя]]} [,...n]
  FROM  имя_таблицы [[AS] псевдоним] [,...n]
  [WHERE  <условие_поиска>]
  [GROUP BY  имя_столбца [,...n]]
  [HAVING <критерии выбора групп>]
  [ORDER BY имя_столбца [,...n]]
```

Оператор SELECT определяет поля (столбцы), которые будут входить в результат выполнения запроса. В списке они разделяются запятыми и приводятся в такой очередности, в какой должны быть представлены в результате запроса. Если используется имя поля, содержащее пробелы или разделители, его следует заключить в квадратные скобки. Символом * можно выбрать все поля, а вместо имени поля применить выражение из нескольких имен.

Если обрабатывается ряд таблиц, то (при наличии одноименных полей в разных таблицах) в списке полей используется полная спецификация поля, т.е. Имя_таблицы.Имя_поля.

Предложение FROM задает имена таблиц и просмотров, которые содержат поля, перечисленные в операторе SELECT. Необязательный параметр псевдонима – это сокращение, устанавливаемое для имени таблицы.

Обработка элементов оператора SELECT выполняется в следующей последовательности:

FROM – определяются имена используемых таблиц;

WHERE – выполняется фильтрация строк объекта в соответствии с заданными условиями;

GROUP BY – образуются группы строк, имеющих одно и то же значение в указанном столбце;

HAVING – фильтруются группы строк объекта в соответствии с указанным условием;

SELECT – устанавливается, какие столбцы должны присутствовать в выходных данных;

ORDER BY – определяется упорядоченность результатов выполнения операторов.

Порядок предложений и фраз в операторе SELECT не может быть изменен. Только два предложения SELECT и FROM являются обязательными, все остальные могут быть опущены. SELECT – закрытая операция: результат запроса к таблице представляет собой другую таблицу.

В дальнейшем изложении в качестве примера будет использоваться небольшая база данных, отражающая процесс поставки или продажи некоторого товара постоянным клиентам.

Исходя из анализа предметной области, можно выделить два типа сущностей – ТОВАР и КЛИЕНТ, которые связаны между собой отношением "многие–ко–многим", т.к. каждый покупатель может купить много

наименований товара, а каждый товар может быть куплен многими покупателями. Однако реляционная модель данных требует заменить отношение "многие–ко-многим" на несколько отношений "один–ко-многим". Добавим еще один тип сущностей, отображающий процесс продажи товаров, – СДЕЛКА.

Установим связи между объектами. Один покупатель может неоднократно покупать товары, поэтому между объектами КЛИЕНТ и СДЕЛКА имеется связь "один–ко–многим". Каждое наименование товара может неоднократно участвовать в сделках, в результате между объектами ТОВАР и СДЕЛКА имеется связь "один-ко-многим".

Определим атрибуты и свяжем их с сущностями и связями. К объекту ТОВАР относятся такие характеристики, как название, тип, цена, сорт. К объекту КЛИЕНТ – имя, отчество, фамилия, фирма, город, телефон. Тип сущности СДЕЛКА может быть охарактеризован такими признаками, как дата и количество проданного товара.

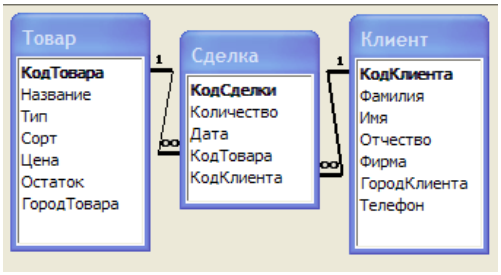
Важным этапом в создании базы данных является определение атрибутов, которые однозначно определяют каждый экземпляр сущности, т.е. выявление первичных ключей.

Для таблицы ТОВАР название не может служить первичным ключом, т.к. товары разных типов могут иметь одинаковые названия, поэтому введем первичный ключ КодТовара, под которым можно понимать, например, артикул товара. Точно так же ни Имя, ни Фирма, ни Город не могут служить первичным ключом в таблице КЛИЕНТ. Введем первичный ключ КодКлиента, под которым можно понимать номер паспорта, идентификационный номер налогоплательщика или любой другой атрибут, однозначно определяющий каждого клиента. Для таблицы СДЕЛКА первичным ключом является поле КодСделки, т.к. оно однозначно определяет дату, покупателя и другие элементы данных. В качестве первичного ключа можно было бы выбрать не одно поле, а некоторую

совокупность полей, но для иллюстрации конструкций языка ограничимся простыми первичными ключами.

Установим связи между таблицами. Один покупатель может неоднократно покупать товары. Поэтому между таблицами КЛИЕНТ и СДЕЛКА имеется связь "один–ко–многим" по полю КодКлиента.

Каждый покупатель может приобрести несколько различных товаров. Поэтому между таблицами ТОВАР и СДЕЛКА имеется связь "один–ко–многим" по полю КодТовара.



Пример 1. Составить список сведений о всех клиентах.

SELECT * FROM Клиент

В таблице могут присутствовать повторяющиеся записи (дубликаты). Предикат ALL задает включение в выходной набор всех дубликатов, отобранных по критерию WHERE. Нет необходимости указывать ALL явно, поскольку это значение действует по умолчанию.

Пример 2. Составить список всех фирм.

SELECT ALL Клиент.Фирма FROM Клиент

Или (что эквивалентно)

SELECT Клиент.Фирма FROM Клиент

Результат выполнения запроса может содержать дублирующиеся значения, поскольку в отличие от операций реляционной алгебры оператор SELECT не исключает повторяющихся значений при выполнении выборки данных.

Предикат DISTINCT следует применять в тех случаях, когда требуется отбросить блоки данных, содержащие дублирующие записи в выбранных полях. Значения для каждого из приведенных в инструкции SELECT полей должны быть уникальными, чтобы содержащая их запись смогла войти в выходной набор.

Причиной ограничения в применении DISTINCT является то обстоятельство, что его использование может резко замедлить выполнение запросов.

Откорректированный пример 2 выглядит следующим образом:

```
SELECT DISTINCT Клиент.Фирма
  FROM Клиент
```

Предложение Where.

С помощью WHERE-параметра пользователь определяет, какие блоки данных из приведенных в списке FROM таблиц появятся в результате запроса. За ключевым словом WHERE следует перечень условий поиска, определяющих те строки, которые должны быть выбраны при выполнении запроса. Существует пять основных типов условий поиска (или предикатов):

Сравнение: сравниваются результаты вычисления одного выражения с результатами вычисления другого.

Диапазон: проверяется, попадает ли результат вычисления выражения в заданный диапазон значений.

Принадлежность множеству: проверяется, принадлежит ли результат вычислений выражения заданному множеству значений.

Соответствие шаблону: проверяется, отвечает ли некоторое строковое значение заданному шаблону.

Значение NULL: проверяется, содержит ли данный столбец определитель NULL (неизвестное значение).

Сравнение

В языке SQL можно использовать следующие операторы сравнения: = – равенство; < – меньше; > – больше; <= – меньше или равно; >= – больше или равно; <> – не равно.

Пример 3. Показать все операции отпуска товаров объемом больше 20.

```
SELECT * FROM Сделка
   WHERE Количество>20
```

Более сложные предикаты могут быть построены с помощью логических операторов AND, OR или NOT, а также скобок, используемых для определения порядка вычисления выражения. Вычисление выражения в условиях выполняется по следующим правилам:

Выражение вычисляется слева направо.

Первыми вычисляются подвыражения в скобках.

Операторы NOT выполняются до выполнения операторов AND и OR.

Операторы AND выполняются до выполнения операторов OR.

Для устранения любой возможной неоднозначности рекомендуется использовать скобки.

Пример 4. Вывести список товаров, цена которых больше или равна 100 и меньше или равна 150.

```
SELECT Название, Цена
   FROM Товар
   WHERE Цена>=100 And Цена<=150
```

Пример 5. Вывести список клиентов из Москвы или из Самары.

```
SELECT Фамилия, ГородКлиента
   FROM Клиент
   WHERE ГородКлиента="Москва" Or
       ГородКлиента="Самара"
```

Диапазон

Оператор BETWEEN используется для поиска значения внутри некоторого интервала, определяемого своими минимальным и

максимальным значениями. При этом указанные значения включаются в условие поиска.

Пример 6. Вывести список товаров, цена которых лежит в диапазоне от 100 до 150 (запрос эквивалентен примеру 4.4).

```
SELECT Название, Цена
   FROM Товар
   WHERE Цена Between 100 And 150
```

При использовании отрицания NOT BETWEEN требуется, чтобы проверяемое значение лежало вне границ заданного диапазона.

Пример 7. Вывести список товаров, цена которых не лежит в диапазоне от 100 до 150.

```
SELECT Товар.Название, Товар.Цена
   FROM Товар
   WHERE Товар.Цена Not Between 100 And 150
```

Или (что эквивалентно)

```
SELECT Товар.Название, Товар.Цена
  FROM Товар
  WHERE (Товар.Цена<100) OR (Товар.Цена>150)
```

Принадлежность множеству

Оператор IN используется для сравнения некоторого значения со списком заданных значений, при этом проверяется, соответствует ли результат вычисления выражения одному из значений в предоставленном списке. При помощи оператора IN может быть достигнут тот же результат, что и в случае применения оператора OR, однако оператор IN выполняется быстрее.

Пример 8. Вывести список клиентов из Москвы или из Самары (запрос эквивалентен примеру 5).

```
SELECT Фамилия, ГородКлиента
  FROM Клиент
  WHERE ГородКлиента in ("Москва", "Самара")
```

NOT IN используется для отбора любых значений, кроме тех, которые указаны в представленном списке.

Пример 9. Вывести список клиентов, проживающих не в Москве и не в Самаре.

```
SELECT Фамилия, ГородКлиента
   FROM Клиент
   WHERE ГородКлиента
    Not in ("Москва","Самара")
```

Соответствие шаблону

С помощью оператора LIKE можно выполнять сравнение выражения с заданным шаблоном, в котором допускается использование символов-заменителей:

Символ % – вместо этого символа может быть подставлено любое количество произвольных символов.

Символ _ заменяет один символ строки.

[] – вместо символа строки будет подставлен один из возможных символов, указанный в этих ограничителях.

[^] – вместо соответствующего символа строки будут подставлены все символы, кроме указанных в ограничителях.

Пример 10. Найти клиентов, у которых в номере телефона вторая цифра – 4.

```
SELECT Клиент.Фамилия, Клиент.Телефон
   FROM Клиент
   WHERE Клиент.Телефон Like "_4%"
```

Пример 11. Найти клиентов, у которых в номере телефона вторая цифра – 2 или 4.

```
SELECT Клиент.Фамилия, Клиент.Телефон
   FROM Клиент
   WHERE Клиент.Телефон Like "_[24]%"
```

Пример 12. Найти клиентов, у которых в номере телефона вторая цифра 2, 3 или 4.

```
SELECT Клиент.Фамилия, Клиент.Телефон
   FROM Клиент
   WHERE Клиент.Телефон Like "_[2-4]%"
```

Пример 13. Найти клиентов, у которых в фамилии встречается слог "ро".

```
SELECT Клиент.Фамилия
   FROM Клиент
   WHERE Клиент.Фамилия Like "%ро%"
```

Значение NULL

Оператор IS NULL используется для сравнения текущего значения со значением NULL – специальным значением, указывающим на отсутствие любого значения. NULL – это не то же самое, что знак пробела (пробел – допустимый символ) или ноль (0 – допустимое число). NULL отличается и от строки нулевой длины (пустой строки).

Пример 14. Найти сотрудников, у которых нет телефона (поле Телефон не содержит никакого значения).

```
SELECT Фамилия, Телефон
   FROM Клиент
   WHERE Телефон Is Null
```

IS NOT NULL используется для проверки присутствия значения в поле.

Пример 15. Выборка сотрудников, у которых есть телефон (поле Телефон содержит какое-либо значение).

```
SELECT Клиент.Фамилия, Клиент.Телефон
   FROM Клиент
   WHERE Клиент.Телефон Is Not Null
```

Предложение ORDER BY

В общем случае строки в результирующей таблице SQL-запроса никак не упорядочены. Однако их можно требуемым образом отсортировать, для чего в оператор SELECT помещается фраза ORDER BY, которая сортирует данные выходного набора в заданной последовательности. Сортировка может выполняться по нескольким полям, в этом случае они перечисляются за ключевым словом ORDER BY через запятую. Способ сортировки задается ключевым словом, указываемым в рамках параметра ORDER BY следом за названием поля, по которому выполняется сортировка. По умолчанию реализуется сортировка по возрастанию. Явно она задается ключевым словом ASC. Для выполнения сортировки в обратной последовательности необходимо после имени поля, по которому она выполняется, указать ключевое слово DESC. Фраза ORDER BY позволяет упорядочить выбранные записи в порядке возрастания или убывания значений любого столбца или комбинации столбцов, независимо от того, присутствуют эти столбцы в таблице результата или нет. Фраза ORDER BY всегда должна быть последним элементом в операторе SELECT.

Пример 16.Вывести список клиентов в алфавитном порядке.

```
SELECT Клиент.Фамилия, Клиент.Фирма
   FROM Клиент
   ORDER BY Клиент.Фамилия
```

Во фразе ORDER BY может быть указано и больше одного элемента. Главный (первый) ключ сортировки определяет общую упорядоченность строк результирующей таблицы. Если во всех строках результирующей таблицы значения главного ключа сортировки являются уникальными, нет необходимости использовать дополнительные ключи сортировки. Однако, если значения главного ключа не уникальны, в результирующей таблице будет присутствовать несколько строк с одним и тем же значением старшего ключа сортировки. В этом случае, возможно, придется упорядочить строки с

одним и тем же значением главного ключа по какому-либо дополнительному ключу сортировки.

Пример 17. Вывести список фирм и клиентов. Названия фирм упорядочить в алфавитном порядке, имена клиентов в каждой фирме отсортировать в обратном порядке.

```
SELECT Клиент.Фирма, Клиент.Фамилия
   FROM Клиент
   ORDER BY Клиент.Фирма, Клиент.Фамилия DESC
```

Оператор Insert

Предназначен для добавления новых данных в таблицу. Он имеет следующий формат:

INSERT INTO TableName [(columnList)] VALUES (dataValueList);
Здесь параметр TableName (Имя таблицы) может представлять имя таблицы базы данных. Параметр columnList (Список столбцов) представляет собой список, состоящий из имен одного или более столбцов, разделенных запятыми. Параметр columnList является необязательным. Если он опущен, то предполагается использование списка из имен всех столбцов таблицы, указанных в том порядке, в котором они были описаны в операторе CREATE TABLE. Параметр dataValueList (Список значений данных) должен следующим образом соответствовать параметру columnList:

- количество элементов в обоих списках должно быть одинаковым;

- должно существовать прямое соответствие между позицией одного и того же элемента в обоих списках типы данных элементов списка dataValueList должны быть совместимы с типом данных соответствующих столбцов таблицы.

Пример использования оператора Insert.

Задание: Поместите в таблицу staff новую запись, содержащую данные во всех столбцах.

```
INSERT INTO Staff VALUES('SG16', 'Alan', 'Brown', 'Assistant’, 'M’, DATE '1957-05-25',8300, 'B003');
```

Оператор Update

Позволяет изменять содержимое уже существующих строк указанной таблицы. Этот оператор имеет следующий формат:

```
UPDATE tableName
SET columnNamel=dataValuel[,columnName2=dataValue2…]
[WHERE searchCondition];
```

Здесь параметр tableName представляет имя таблицы базы данных. В конструкции SET указываются имена одного или более столбцов, данные в которых необходимо изменить. Конструкция WHERE является необязательной. Если она опущена, значения указанных столбцов будут изменены во всех строках таблицы. Если конструкция WHERE присутствует, то обновлены будут только те строки, которые удовлетворяют условию поиска, заданному в параметре searchCondition. Параметры dataValue1, dataValue2... представляют новые значения соответствующих столбцов и должны быть совместимы с ними по типу данных.

Пример использования оператора Update.

Задание: Всем менеджерам компании повысить заработную плату на 5%.

```
UPDATE Staff
SET salary=salary*1.05
WHERE position = 'Manager’;
```

Оператор Delete

Позволяет удалять строки данных из указанной таблицы. Этот оператор имеет следующий формат:

```
DELETE FROM tableName
[WHERE searchCondition];
```

Как и в случае операторов INSERT и UPDATE, параметр TableName представляет собой таблицы базы данных. Параметр searchCondition является необязательным — если он опущен, из таблицы будут удалены все существующие в ней строки. Однако сама по себе таблица удалена не будет.

Если необходимо удалить не только содержимое таблицы, но и ее определение, следует использовать оператор DROP TABLE. Если конструкция WHERE присутствует, из таблицы будут удалены только те строки, которые удовлетворяют условию отбора, заданному параметром searchCondition

Пример использования оператора Delete.

Задание: Удалить все записи об осмотрах сдаваемого в аренду объекта с учетным номером PG4.

```
DELETE FROM Viewing
WHERE propertyNo = 'PG4';
```

```
DROP {TABLE таблица | INDEX индекс ON таблица}
```

где таблица — имя таблицы, которую следует удалить или из которой следует удалить индекс;

индекс — имя индекса, удаляемого из таблицы.

Прежде чем удалить таблицу или удалить из нее индекс, необходимо ее закрыть. Следует отметить, что таблица удаляется из базы данных безвозвратно.

Операции над отношениями в SQL.

Рассмотрим основные операции над отношениями, которые могут представлять интерес с точки зрения извлечения данных из реляционных таблиц. Это объединение, пересечение, разность, расширенное декартово произведение отношений, а также специальные операции над отношениями: выборка, проекция, соединение и деление.

Для иллюстрации теоретико-множественных операций над отношениями введем абстрактные отношения (таблицы) с некоторыми атрибутами (полями).

Отношение R

R.a1	R.a2
A	1
A	2
B	1
B	3
B	4

CREATE TABLE R

(a1 CHAR(1), a2 INT, PRIMARY KEY(a1,a2))

Отношение S

S.b1	S.b2
1	h
2	g
3	h

CREATE TABLE S

(b1 INT PRIMARY KEY, b2 CHAR(1))

Операции выборки и проекции являются унарными, поскольку они работают с одним отношением.

Операция Выборки

Операция выборки - построение горизонтального подмножества, т.е. подмножества кортежей, обладающих заданными свойствами.

Операция выборки работает с одним отношением R и определяет результирующее отношение, которое содержит только те кортежи (строки) отношения R, которые удовлетворяют заданному условию F (предикату).

Выборка $\sigma_{(a2=1)}(R)=\{(a, 1), (b, 1)\}$ записывается следующим образом:

SELECT a1, a2

FROM R

WHERE a2=1

Декартово произведение.

Декартово произведение RxS двух отношений (двух таблиц) определяет новое отношение - результат конкатенации (т.е. сцепления) каждого кортежа (каждой записи) из отношения R с каждым кортежем (каждой записью) из отношения S.

RxS={(a, 1, 1, h), (a, 2, 1, h), (b, 1, 1, h), ... }

SELECT R.a1, R.a2, S.b1, S.b2

FROM R, S

Если одно отношение имеет N записей и K полей, а другое M записей и L полей, то отношение с их декартовым произведением будет содержать NxM записей и K+L полей. Исходные отношения могут содержать поля с одинаковыми именами, тогда имена полей будут содержать названия таблиц в виде префиксов для обеспечения уникальности имен полей в отношении, полученном как результат выполнения декартова произведения.

Однако в таком виде отношение содержит больше информации, чем обычно необходимо пользователю. Как правило, пользователей интересует лишь некоторая часть всех комбинаций записей в декартовом произведении, удовлетворяющая некоторому условию. Поэтому вместо декартова произведения обычно используется одна из самых важных операций реляционной алгебры - операция соединения, которая является производной от операции декартова произведения. С точки зрения эффективности реализации в реляционных СУБД эта операция - одна из самых трудных и часто входит в число основных причин, вызывающих свойственные всем реляционным системам проблемы с производительностью.

Операция объединения.

Объединение R UNION S отношений R и S можно получить в результате их конкатенации с образованием одного отношения с исключением кортежей-дубликатов. При этом отношения R и S должны быть совместимы,

т.е. иметь одинаковое количество полей с совпадающими типами данных. Иначе говоря, отношения должны быть совместимы по объединению.

Объединением двух таблиц R и S является таблица, содержащая все строки, которые имеются в первой таблице R, во второй таблице S или в обеих таблицах сразу.

```
SELECT R.a1, R.a2
FROM R
UNION
SELECT S.b2, S.b1
FROM S
```

Операция пересечения.

Операция пересечения R INTERSECT S=R-(R-S) определяет отношение, которое содержит кортежи, присутствующие как в отношении R, так и в отношении S. Отношения R и S должны быть совместимы по объединению.

Пересечением двух таблиц R и S является таблица, содержащая все строки, присутствующие в обеих исходных таблицах одновременно.

```
SELECT R.a1, R.a2
FROM R,S
WHERE R.a1=S.b1 AND R.a2=S.b2
```

В стандарте языка SQL имеются предложения **оператора SELECT** для выполнения операций **пересечения** и **разности** запросов. Этими предложениями являются **INTERSECT** (пересечение) и **EXCEPT** (разность), которые работают аналогично предложению **UNION**. В результирующий набор попадают только те строки, которые присутствуют в обоих запросах (INTERSECT) или только те строки первого запроса, которые отсутствуют во втором (EXCEPT).

Однако многие СУБД не поддерживают эти предложения в операторе SELECT. Это справедливо и для MS SQL Server. Поэтому для выполнения операций пересечения и разности могут быть использованы другие средства. Здесь уместно заметить, что один и тот же результат можно получить с

помощью различных формулировок оператора SELECT. В случае пересечения и разности можно воспользоваться предикатом существования EXISTS.

Предикат EXISTS принимает значение TRUE, если подзапрос возвращает любое количество строк, иначе его значение равно FALSE. Для NOT EXISTS все наоборот. Этот предикат никогда не принимает значение UNKNOWN.

Обычно предикат EXISTS используется в зависимых подзапросах. Этот вид подзапроса имеет внешнюю ссылку, связанную со значением в основном запросе. Результат подзапроса может зависеть от этого значения и должен оцениваться отдельно для каждой строки запроса, в котором содержится данный подзапрос. Поэтому предикат EXISTS может иметь разные значения для каждой строки основного запроса.

Операция разности.

Разность R EXCEPT S=R-S двух отношений R и S состоит из кортежей, которые имеются в отношении R, но отсутствуют в отношении S. Причем отношения R и S должны быть совместимы по объединению.

Разностью двух таблиц R и S является таблица, содержащая все строки, которые присутствуют в таблице R, но отсутствуют в таблице S.

```
SELECT R.a1, R.a2
FROM R
WHERE NOT EXISTS
   (SELECT S.b1,S.b2
    FROM S
    WHERE S.b1=R.a2 AND S.b2=R.a1)
```

Найти тех производителей ПК-блокнотов, которые производят также и принтеры:

```
SELECT DISTINCT maker
FROM Product AS LapProduct
WHERE type = 'Laptop'
AND EXISTS (SELECT maker
FROM Product
WHERE type = 'Printer' AND maker = LapProduct.maker);
```

Найти тех производителей ПК-блокнотов, которые не производят принтеров:

```
SELECT DISTINCT maker
FROM Product AS LapProduct
WHERE type = 'Laptop'
AND NOT EXISTS (SELECT maker
FROM Product
WHERE type = 'Printer' AND maker = LapProduct.maker)
```

Использование итоговых функций

С помощью итоговых (агрегатных) функций в рамках SQL-запроса можно получить ряд обобщающих статистических сведений о множестве отобранных значений выходного набора.

Пользователю доступны следующие основные итоговые функции:

Count (Выражение) - определяет количество записей в выходном наборе SQL-запроса;

Min/Max (Выражение) - определяют наименьшее и наибольшее из множества значений в некотором поле запроса;

Avg (Выражение) - эта функция позволяет рассчитать среднее значение множества значений, хранящихся в определенном поле отобранных запросом записей. Оно является арифметическим средним значением, т.е. суммой значений, деленной на их количество.

Sum (Выражение) - вычисляет сумму множества значений, содержащихся в определенном поле отобранных запросом записей.

Чаще всего в качестве выражения выступают имена столбцов. Выражение может вычисляться и по значениям нескольких таблиц.

Все эти функции оперируют со значениями в единственном столбце таблицы или с арифметическим выражением и возвращают единственное значение. Функции COUNT, MIN и MAX применимы как к числовым, так и к нечисловым полям, тогда как функции SUM и AVG могут использоваться только в случае числовых полей, за исключением COUNT(*). При вычислении результатов любых функций сначала исключаются все пустые значения, после чего требуемая операция применяется только к оставшимся конкретным значениям столбца. Вариант COUNT(*) - особый случай использования функции COUNT, его назначение состоит в подсчете всех строк в результирующей таблице, независимо от того, содержатся там пустые, дублирующийся или любые другие значения.

Если до применения обобщающей функции необходимо исключить дублирующийся значения, следует перед именем столбца в определении функции поместить ключевое слово DISTINCT. Оно не имеет смысла для функций MIN и MAX, однако его использование может повлиять на результаты выполнения функций SUM и AVG, поэтому необходимо заранее обдумать, должно ли оно присутствовать в каждом конкретном случае. Кроме того, ключевое слово DISTINCT может быть указано в любом запросе не более одного раза.

Очень важно отметить, что итоговые функции могут использоваться только в списке предложения SELECT и в составе предложения HAVING. Во всех других случаях это недопустимо. Если список в предложении SELECT содержит итоговые функции, а в тексте запроса отсутствует фраза GROUP BY, обеспечивающая объединение данных в группы, то ни один из элементов списка предложения SELECT не может включать каких-либо ссылок на поля,

за исключением ситуации, когда поля выступают в качестве аргументов итоговых функций.

Определить первое по алфавиту название товара.

```
SELECT Min(Название) AS Min_Название
FROM Товар
```

Определить количество сделок.

```
SELECT Count(*) AS Количество_сделок
FROM Сделка
```

Определить суммарное количество проданного товара.

```
SELECT Sum(Количество) AS Количество_товара
FROM Сделка
```

Определить среднюю цену проданного товара.

```
SELECT Avg(Цена) AS Avg_Цена
FROM Товар, Сделка
WHERE Товар.КодТовара=Сделка.КодТовара
```

Подсчитать общую стоимость проданных товаров.

```
SELECT Sum(Товар.Цена*Сделка.Количество) AS Стоимость
FROM Товар, Сделка
WHERE Товар.КодТовара=Сделка.КодТовара
```

Предложение GROUP BY

Часто в запросах требуется формировать промежуточные итоги, что обычно отображается появлением в запросе фразы "для каждого...". Для этой цели в операторе SELECT используется предложение GROUP BY. Запрос, в котором присутствует GROUP BY, называется группирующим запросом, поскольку в нем группируются данные, полученные в результате выполнения операции SELECT, после чего для каждой отдельной группы создается

единственная суммарная строка. Стандарт SQL требует, чтобы предложение SELECT и фраза GROUP BY были тесно связаны между собой. При наличии в операторе SELECT фразы GROUP BY каждый элемент списка в предложении SELECT должен иметь единственное значение для всей группы. Более того, предложение SELECT может включать только следующие типы элементов: имена полей, итоговые функции, константы и выражения, включающие комбинации перечисленных выше элементов.

Все имена полей, приведенные в списке предложения SELECT, должны присутствовать и во фразе GROUP BY - за исключением случаев, когда имя столбца используется в итоговой функции. Обратное правило не является справедливым - во фразе GROUP BY могут быть имена столбцов, отсутствующие в списке предложения SELECT.

Если совместно с GROUP BY используется предложение WHERE, то оно обрабатывается первым, а группированию подвергаются только те строки, которые удовлетворяют условию поиска.

Стандартом SQL определено, что при проведении группирования все отсутствующие значения рассматриваются как равные. Если две строки таблицы в одном и том же группируемом столбце содержат значение NULL и идентичные значения во всех остальных непустых группируемых столбцах, они помещаются в одну и ту же группу.

Вычислить средний объем покупок, совершенных каждым покупателем.

```
SELECT Клиент.Фамилия, Avg(Сделка.Количество) AS Среднее_количество
FROM Клиент, Сделка
WHERE Клиент.КодКлиента=Сделка.КодКлиента
GROUP BY Клиент.Фамилия
```

Определить, на какую сумму был продан товар каждого наименования.

```
SELECT Товар.Название, Sum(Товар.Цена*Сделка.Количество) AS Стоимость
FROM Товар, Сделка
WHERE Товар.КодТовара=Сделка.КодТовара
GROUP BY Товар.Название
```

Подсчитать количество сделок, осуществленных каждой фирмой.

```
SELECT Клиент.Фирма, Count(Сделка.КодСделки) AS Количество_сделок
FROM Клиент, Сделка
WHERE Клиент.КодКлиента=Сделка.КодКлиента
GROUP BY Клиент.Фирма
```

Подсчитать общее количество купленного для каждой фирмы товара и его стоимость.

```
SELECT Клиент.Фирма, Sum(Сделка.Количество) AS Общее_Количество,
   Sum(Товар.Цена*Сделка.Количество) AS Стоимость
FROM Товар, Клиент, Сделка
WHERE Клиент.КодКлиента=Сделка.КодКлиента AND Товар.КодТовара=Сделка.КодТовара
GROUP BY Клиент.Фирма
```

Определить суммарную стоимость каждого товара за каждый месяц.

```
SELECT Товар.Название, Month(Сделка.Дата) AS Месяц,
   Sum(Товар.Цена*Сделка.Количество) AS Стоимость
FROM Товар, Сделка
WHERE Товар.КодТовара=Сделка.КодТовара
GROUP BY Товар.Название, Month(Сделка.Дата)
```

Определить суммарную стоимость каждого товара первого сорта за каждый месяц.

```
SELECT Товар.Название, Month(Сделка.Дата) AS Месяц,
   Sum(Товар.Цена*Сделка.Количество) AS Стоимость
FROM Товар, Сделка
WHERE Товар.КодТовара=Сделка.КодТовара AND Товар.Сорт="Первый"
GROUP BY Товар.Название, Month(Сделка.Дата)
```

Предложение HAVING

При помощи HAVING отражаются все предварительно сгруппированные посредством GROUP BY блоки данных, удовлетворяющие заданным в HAVING условиям. Это дополнительная возможность "профильтровать" выходной набор.

Условия в HAVING отличаются от условий в WHERE:

HAVING исключает из результирующего набора данных группы с результатами агрегированных значений;

WHERE исключает из расчета агрегатных значений по группировке записи, не удовлетворяющие условию; в условии поиска WHERE нельзя задавать агрегатные функции.

Определить фирмы, у которых общее количество сделок превысило три.

```
SELECT Клиент.Фирма, Count(Сделка.Количество) AS Количество_сделок
FROM Клиент, Сделка
WHERE Клиент.КодКлиента=Сделка.КодКлиента
GROUP BY Клиент.Фирма
HAVING Count(Сделка.Количество)>3
```

Вывести список товаров, проданных на сумму более 10000 руб.

```
SELECT Товар.Название, Sum(Товар.Цена*Сделка.Количество) AS Стоимость
FROM Товар, Сделка
WHERE Товар.КодТовара=Сделка.КодТовара
GROUP BY Товар.Название
HAVING Sum(Товар.Цена*Сделка.Количество)>10000
```

Вывести список товаров, проданных на сумму более 10000 без указания суммы.

```
SELECT Товар.Название
FROM Товар, Сделка
WHERE Товар.КодТовара=Сделка.КодТовара
GROUP BY Товар.Название
HAVING Sum(Товар.Цена*Сделка.Количество)>10000
```

2.8. Задачи для самостоятельной работы

ER-диаграммы

Ниже приведены списки атрибутов для разных предметных областей. Для каждой ПО построить ER-диаграмму и перейти от нее к предварительным отношениям. Затем все оставшиеся атрибуты приписать к полученным отношениям и проверить, находятся ли эти отношения в НФБК.

1. Физкультура

Зачетка, Фамилия, Группа, Дата рождения, Преподаватель, Специализация, Медицинская группа, Разряд, Вид, Особенности, Норматив, Результат, Дата, Оценка, Семестр, Пол.

2. Кинотеатры

Название, Адрес, Телефон, Категория, Вместимость, Число залов, Кинотеатр, Кинофильм, Время, Дата, Режиссер, Год выпуска, Страна, Число серий, Тематика, Краткое содержание.

3. Аптека

Номер, Дата, Врач, Поликлиника, Лекарство, Количество, Режим приема, Стоимость, Особые замечания, Шифр, Название, Группа, Краткая рекомендация по применению, Срок хранения рецепта, Дата поступления, Цена, Единица измерения, Количество, Срок годности.

4. Библиотека

Шифр, Автор, Название, Тематика, Издательство, Год издания, Тираж, Количество страниц, Аннотация, Билет, Фамилия, Место работы, Должность, Телефон, Возраст, Особые отметки, Дата выдачи, Срок возврата.

5. Почтовое отделение

Шифр, Название, Тип, Учреждение, Цена, Число экземпляром в год, Адрес, Фамилия, Профессия, Возраст, Дата начала, Длительность, Сумма.

6. Расписание экзаменов

Номер группы, Специальность, Число студентов, Староста, Факультет, Курс, Название дисциплины, Преподаватель, Дата консультации, Время консультации, Дата экзамена, Время экзамена, Аудитория для консультации, Аудитория для экзамена.

7. Гостиницы города

Номер, Название, Директор, Телефон, Категория, Адрес, Число мест, Стоимость, Фамилия, Адрес, Возраст, Дата заезда, Срок проживания, Оплата, Особые отметки.

8. Расписание занятий

Номер группы, Специальность, Факультет, Число студентов, Староста, Номер аудитории, Вместимость, Тип, Шифр дисциплины, Название, Фамилия преподавателя, Звание, Должность, Кафедра, Время, День, Неделя.

9. Поликлиника

Карта, Фамилия, Адрес, Возраст, Место работы, Профессия, Дата последнего посещения, Особые отметки, Номер кабинета, Название, Врач, Пропускная способность, Дата, Время, Жалобы, Диагноз, Назначение.

10. Канцелярия

Номер, Фамилия, Должность, Кафедра, Дата, Длительность, Город, Организация, Аванс, Приказ, Дата приказа, Содержание командировки, Подпись, Тип расхода, Сумма расхода, Ведомость, Дата отчета.

11. Диета

Номер, Название диеты, Диагноз, Длительность, Противопоказания, Название блюда, Жиры, Белки, Углеводы, Калорийность, Несовместимость, Особенности применения, Количество, Форма.

12. Кулинария

Название блюда, Категория, Калорийность, Стоимость, Название продукта, Единица измерения, Жиры, Белки, Углеводы, Вита мины, Цена, Поставщик, Количество, Состояние.

13. Больница

Номер палаты, Отделение, Число коек, Врач, Персонал, Фамилия Карта, Возраст, Диагноз, Адрес, Профессия, Место работы, Специализация, Оклад, Телефон, Характеристика.

14. Турнир

Название, Город, Спонсор, Тренер, Телефон, Рейтинг, Фамилия игрока, Команда, Амплуа, Возраст, Адрес, Телефон, Характеристика, Хозяева, Гости, Дата, Судья, Число зрителей, Время, Результат, Оценка.

15. Коллекционирование монет

Шифр, Название, Страна, Государство, Тираж, Сплав, Год, Вес, Оценка, История, Фамилия, Адрес, Профессия, Место работы, Телефон, Количество.

16. Телефон

Город, Индекс, Стоимость, Промежуточный пункт, Примечания, Телефон, Дата, Продолжительность, Вид оплаты, Документ об оплате, Факт оплаты, Сумма, Адрес, Фамилия, Месячная плата.

17. Шахматы

Шифр, Команда, Возраст, Место работы, Профессия, Должность, Квалификация, Рейтинг, Телефон, Белые, Черные, Очки-б, Очки-ч, Дата, Итог, Число ходов, Номер игры, Ход-б, Ход-ч, Время-б, Время-ч.

18. Домоуправление

Адрес, Квартира, Фамилия, Площадь, Число комнат, Номер ордера, Дата получения, Месячная плата, Долг, Вид услуги, Стоимость, Дата введения, Дата оплаты.

19. Управление троллейбусами

Номер маршрута, Протяженность, Время, Число остановок, Начало движения, Конец движения, Состояние, Число машин, На звание остановки, Номер маршрута, Номер остановки, Крыша, Время отправления.

20. Спортклуб

Название секции, Тренер, Число членов, Место занятий, Особенности приема, Оплата, Фамилия, Возраст, Адрес, Телефон, Рост, Вес, Личный рекорд, Достижения, Дата соревнования, Ранг, Результат, Место, Число участников.

21. Ремонт телевизоров

Марка, Тип, Завод, Адрес завода, Телефон, Характеристика телевизора, Цена, Фамилия, Адрес, Дата ремонта, Платы, Документ, Мастер, Работа.

22. Станция технического обслуживания

Фамилия, Разряд, Адрес, Телефон, Оклад, Стаж, Номер, Марка автомобиля, Цвет, Заводской номер, Пробег, Владелец, Техпаспорт, Год выпуска, Состояние, Дата поступления, Документ, Срок готовности, Дата окончания, Стоимость, Содержание ремонта.

23. Научно-исследовательская работа

Номер, Тема, Исполнитель, Заказчик, Дата заключения, Дата начала, Дата окончания, Сумма, Число этапов, Особые условия, Кафедра, Фамилия, Телефон, Число договоров, Сумма договоров, Адрес, Счет, Руководитель, Факс, Министерство.

24. Учебный план специальности

Шифр специальности, Дисциплина, Семестр, Вид испытания, Вид занятий, Форма занятий, Количество часов.

25. Расписание занятий

Группа, Аудитория, Дисциплина, Вид занятий, Преподаватель, День недели, Пара занятий.

SQL

1. Найдите номер модели, скорость и размер жесткого диска для всех ПК стоимостью менее 500 дол.
2. Найдите производителей принтеров.
3. Найдите номер модели, объем памяти и размеры экранов ПК-блокнотов, цена которых превышает 1000 дол.
4. Найдите все записи таблицы Printer для цветных принтеров.
5. Найдите номер модели, скорость и размер жесткого диска ПК, имеющих 12x или 24x CD и цену менее 600 дол.
6. Укажите производителя и скорость для тех ПК-блокнотов, которые имеют жесткий диск объемом не менее 10 Гбайт.
7. Найдите номера моделей и цены всех продуктов (любого типа) выпущенных производителем B (латинская буква).
8. Найдите производителя, продающего ПК, но не ПК-блокноты.
9. Найдите производителей ПК с процессором не менее 450 Мгц.
10. Найдите принтеры, имеющие самую высокую цену. Вывести: model, price
11. Найдите среднюю скорость ПК.

12. Найдите среднюю скорость ПК-блокнотов, цена которых превышает 1000 дол.

13. Найдите среднюю скорость ПК, выпущенных производителем A.

14. Для таблицы Product получить результирующий набор в виде таблицы со столбцами maker, pc, laptop и printer, в которой для каждого производителя требуется указать, производит он (yes) или нет (no) соответствующий тип продукции. В первом случае (yes) указать в скобках без пробела количество имеющихся в наличии (т.е. находящихся в таблицах PC, Laptop и Printer) различных по номерам моделей соответствующего типа.

15. Найдите размеры жестких дисков, совпадающих у двух и более PC. Вывести: HD

16. Найдите пары моделей PC, имеющих одинаковые скорость и RAM. В результате каждая пара указывается только один раз, т.е. (i,j), но не (j,i), Порядок вывода: модель с большим номером, модель с меньшим номером, скорость и RAM.

17. Найдите ПК-блокноты, скорость которых меньше скорости любого из ПК. Вывести: type, model, speed

18. Найдите производителей самых дешевых цветных принтеров. Вывести: maker, price

19. Для каждого производителя найдите средний размер экрана выпускаемых им ПК-блокнотов. Вывести: maker, средний размер экрана.

20. Найдите производителей, выпускающих по меньшей мере три различных модели ПК. Вывести: Maker, число моделей

21. Найдите максимальную цену ПК, выпускаемых каждым производителем. Вывести: maker, максимальная цена.

22. Для каждого значения скорости ПК, превышающего 600 МГц, определите среднюю цену компьютера с такой же скоростью. Вывести: speed, средняя цена.

23. Найдите производителей, которые производили бы как ПК со скоростью не менее 750 МГц, так и ПК-блокноты со скоростью не менее 750 МГц. Вывести: Maker

24. Перечислите номера моделей любых типов, имеющих самую высокую цену по всей имеющейся в базе данных продукции.

25. Найдите производителей принтеров, которые производят ПК с наименьшим объемом RAM и с самым быстрым процессором среди всех ПК, имеющих наименьший объем RAM. Вывести: Maker

26. Найдите среднюю цену ПК и ПК-блокнотов, выпущенных производителем A (латинская буква). Вывести: одна общая средняя цена.

27. Найдите средний размер диска ПК каждого из тех производителей, которые выпускают и принтеры. Вывести: maker, средний размер HD.

ЗАКЛЮЧЕНИЕ

В результате проделанной работы удалось создать курс дистанционного обучения по дисциплине «Информационные системы». Данный курс ориентирован для студентов, обучающихся на специальностях, связанных с изучением информационных технологий. В курс входит изучение теории информационных систем, моделей данных информационных систем, операторов для работы с информационными системами, оптимизация и нормализация данных информационных систем, изучение баз данных информационных систем, MS Access 2013 и изучение структурированного языка вопросов SQL. Данный курс позволяет обучающимся работать с более новым программным обеспечением продукта Microsoft Access 2013.

В процессе изучения данного вопроса были учтены все последние тенденции и обновления продукта Microsoft Access 2013, что в дальнейшем предоставляет возможность студентам работать с использованием новых возможностей данного продукта.

При выполнении данной квалификационной работы были решены следующие задачи:

- рассмотрены основные тенденции развития современной работы с базами данных: цели, содержание, формы, методы и средств обучения;

- проведен анализ научной и учебно-методической литературы по выбранной теме;

- проведен анализ современного программного обеспечения;

- создан дистанционный курс по дисциплине «Информационные системы» для студентов с помощью системы Moodel.

Данный курс можно использовать в учебном процессе для студентов, обучающихся на специальностях, связанных с изучением информационных технологий.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Абдикеева Н.М. Корпоративные информационные системы управления: Учебник. - М.: ИНФРА-М, 2011. - 464 с.
2. Балдин К.В. Информационные системы в экономике: Учебник– 7-е изд. – М.: Издательско-торговая корпорация «Дашков и К*», 2012 – 395с.
3. Варфоломеева А.О. Информационные системы предприятия: Учебное пособие. - М.: НИЦ ИНФРА-М, 2013. - 283 с.
4. Голицына О.Л. Информационные системы: Учебное пособие. - 2-е изд. - М.: Форум: НИЦ ИНФРА-М, 2014. - 448 с.: ил.
5. Голицына О.Л., Максимов Н.В., Попов И.И. Базы данных: Учебное пособие. - 2-е изд., испр. и доп. - М.: Форум: ИНФРА-М, 2009. - 400 с.
6. Гринченко Н.Н., Гусев Е.В., Макаров Н.П., Пылькин А.Н., Цуканова Н.И. Проектирование баз данных. СУБД Microsoft Access. Учебное пособие для высших учебных заведений. Москва: "Горячая линия - Телеком", 2010. – 432с.
7. Дейт К. Дж. Введение в системы баз данных, - СПб.: «Вильямс», 2010. – 560с.
8. Емельянова Н.З., Партыка Т.Л., Попов И.И..Проектирование информационных систем: Учебное пособие. - М.: Форум: НИЦ ИНФРА-М, 2014. - 432 с.
9. Заботина Н.Н. Проектирование информационных систем: Учебное пособие. - М.: НИЦ Инфра-М, 2013. - 331 с.
10. Избачков Ю.С., Петров В.Н. Информационные системы. – СПб.: «Питер», 2011. – 342с.
11. Кириллов В. В., Громов Г. Ю. Введение в реляционные базы данных /. — СПб.: БХВ-Петербург, 2009. — 450 с.
12. Коваленко В.В. Проектирование информационных систем: Учебное пособие. - М.: Форум: НИЦ ИНФРА-М, 2014. - 320 с.

13. Малыхина М.П. Базы данных: основы, проектирование, использование: учебное пособие. – СПб: БХВ-Петербург, 2009. – 320с.

14. Пирогов, В. Ю. Информационные системы и базы данных: организация и проектирование: учеб. пособие. — СПб.: БХВ-Петербург, 2009. — 528 с.

15. Семакин И.Г., Хеннер Е.К. Информационные системы и модели. Элективный курс: методическое пособие. - 2-е изд. - Изд.:"Бином. Лаборатория знаний", 2012. – 164с.

16. Советов Б.Я., Цехановский В.В., Чертовкский В.Д. Базы данных: теория и практика: учебник для бакалавров: для студентов вузов, обучающихся по направлениям "Информатика и вычислительная техника" и "Информационные системы". – 2-е изд. - М.:Юрайт, 2012. – 356с.

17. Хомоненко А.Д., Гофман В.Э., Мещерякова В.Е. Базы данных: Учебник для высших учебных заведений. – СПб.: «Корона-принт», 2010. – 420с.

Интернет-источники

1. Интуит - www.intuit.ru

2. Citforum - www.citforum.ru

3. Информационные системы в экономике – http://www.bibliorossica.com/book.html?currBookId=7799

4. Информационные системы и модели – http://e.lanbook.com/books/element.php?pl1_cid=25&pl1_id=8788

5. Решение задач на SQL – www.sql-ex.ru

Приложение 1.

Описание свойств полей таблиц БД «Учебный процесс»

Таблица А.1. Описание свойств полей таблицы СТУДЕНТ

Имя поля	**Ключевое Поле**	**Обязатель-ное Поле**	**Тип данных**	**Размер**	**Число дес. знаков**	**Подпись поля**
НГ	Да	Да	Короткий текст	3		Группа
НС	Да	Да	Короткий текст	2		Номер студента в группе
ФИО		Да	Короткий текст	15		ФИО
ГОДР		Нет	Числовой	Целое		Год рож-дения
АДРЕС		Нет	Короткий текст	25		
ПБАЛЛ		Нет	Числовой	Одинар-ное с плав. точкой	2	Проход-ной балл

Таблица А.2. Описание свойств полей таблицы ГРУППА

Имя поля	**Ключевое поле**	**Обязательное поле**	**Тип данных**	**Размер**	**Число десятичных знаков**	**Подпись поля**
НГ	Да	Да	Короткий текс	3		Номер группы
КОЛ		Нет	Числовой	Байт		Кол. ст. в гр.
ПБАЛЛ		Нет	Числовой	Одинарное с плав. точкой	2	Прох. балл

Таблицы А.2.

Имя поля	Правило проверки	Сообщение об ошибке
НГ		
КОЛ	>=0 And <=35	Количество студентов больше допустимого
ПБАЛЛ	>2 And <5 Or 0	Ошибка в оценке

Таблица А.3. Описание свойств полей таблицы КАФЕДРА

Имя поля	Ключевое Поле	Обязательное Поле	Тип данных	Размер	Подпись поля
ККАФ	Да	Да	Короткий текст	2	Код
НКАФ		Нет	Короткий текст	15	Название
ТЕЛ		Нет	Короткий текст	9	
ЗАВ		Нет	Короткий текст	15	ФИО зав. кафедры
ФОТО		Нет	Поле объекта OLE		Фото заве-дующего

Таблица А.4. Описание свойств полей таблицы ПРЕПОДАВАТЕЛЬ

Имя поля	Ключевое Поле	Обязательное Поле	Тип данных	Размер	Подпись поля
ТАБН	Да	Да	Короткий текст	4	Таб. номер
ФИО		Да	Короткий текст	30	ФИО препод.
СТ		Нет	Короткий текст	15	Уч. степень
ЗВ		Нет	Короткий текст	10	Уч. звание
ККАФ		Да	Короткий текст	2	Код кафедры

Таблица А.5. Описание свойств полей таблицы ПРЕДМЕТ

Имя поля	**Ключе-вое поле**	**Обяза-тельное поле**	**Тип данных**	**Размер**	**Подпись поля**
КП	Да	Да	Короткий текст	2	Код предмета
НП		Нет	Короткий текст	15	Название предмета
ЧАСЫ		Нет	Числовой	Целое	Всего часов
ЛЕК		Нет	Числовой	Целое	Лекция
ПР		Нет	Числовой	Целое	Практика
ЧС		Нет	Числовой	Целое	Семестров
ПРОГР			Поле МЕМО		Программа

Продолжение **Таблица А.5.**

Имя поля	**Правило проверки**	**Сообщение об ошибке**
КП		
НП		
ЧАСЫ	>0 And <=300	Число часов должно быть не более 300
ЛЕК		
ПР		
ЧС		
ПРОГР		

Таблица А.6. Описание свойств полей таблицы ИЗУЧЕНИЕ

Имя поля	Ключевое Поле	Обязательное Поле	Тип данных	Размер	Число дес. знаков	Подпись поля
НГ	Да	Да	Короткий текст	3		Ном. группы
КП	Да	Да	Короткий текст	2		Код. предмета
ТАБН	Да	Да	Короткий текст	4		Таб. н. преп.
ВИДЗ	Да	Да	Короткий текст	3		Вид занятий
ЧАСЫ		Нет	Числовой	Целое	0	Ср. балл по предм.
СБАЛЛ. ГР		Нет	Числовой	Одинарное с плав. точкой	2	

Таблица А.7. Описание свойств полей таблицы УСПЕВАЕМОСТЬ

Имя поля	Ключевое поле	Обязательное поле	Тип данных	Размер	Число десятичных знаков	Подпись поля
НГ	Да	Да	Короткий текст	3		Номер группы
НС	Да	Да	Короткий текст	2		Ном. студента
КП	Да	Да	Короткий текст	2		Код предм.
ТАБН	Да	Да	Короткий текст	4		Таб. н. препод.
ВИДЗ	Да	Да	Короткий текст	3		Вид занятия
ОЦЕНКА		Нет	Числовой	Целое	0	

Приложение 2.

Данные таблиц БД «Учебный процесс»

группа		
ном.группы	**кол. ст. в группе**	**прох. балл**
101	30	4,5
102	32	4,5
103	29	4,8
104	35	4,4
105	35	4,8
201	35	3,9
202	30	4
203	28	4,7
204	25	4

ИЗУЧЕНИЕ					
Ном.группы	**Код предмета**	**Таб.н.преп.**	**Вид занятий**	**Часы**	**Ср.балл по предм.**
101	01	101	лек	40	
101	01	102	пр	60	
101	02	201	лек	50	
101	02	202	пр	50	
101	03	301	лек	48	
101	03	302	пр	20	
101	04	401	пр	50	
101	05	501	лек	50	
101	05	502	пр	50	
101	06	601	лек	100	
102	01	101	лек	100	
102	01	103	пр	180	
102	04	401	лек	100	
105	01	101	лек	100	
201	01	102	пр	180	
201	02	201	пр	70	
202	04	403	пр	100	
203	01	101	лек	100	
204	05	503	пр	100	

кафедра				
код	**название**	**тел**	**фио зав. каф**	**фото заведуещего**
01	информатики	310-47-74	Игнатьева В. В.	
02	маиематики	310-47-15	Иванов И. И.	
03	истории	310-47-16	Смирнова И. В.	
04	иностранного яз	310-47-17	Жданова И. Е.	
05	физкультуры	310-47-67	Плетнев В. А.	
06	философии	310-47-18	Бондоренко В.В.	

предмет						
Код предмета	**Название предмета**	**Всего часов**	**Лекции**	**Практика**	**Семестров**	**Программа**
01	информатика	200	80	120	4	
02	математика	200	100	100	4	
03	история	140	90	50	3	
04	иностраееый яз	200	0	200	4	
05	философия	100	40	60	2	
06	физкультура	100	0	100	2	

студент					
группа	**номер студента в группе**	**фио**	**год рождения**	**адрес**	**проходной балл**
101	01	Аристов Р. П.	1979		4,25
101	02	Бондаренко С.А	1978		4,5
101	03	Борисова Е.И.	1979		4,25
101	04	Макова Н.В.	1977		4,75
102	01	Боярская Н.П.	1977		4,5
102	02	Федоров Д. К.	1977		4,25
102	03	Сидоров И. Р.	1977		4,5
103	01	Андреев Г.М.	1978		4,25
103	02	Петров О.К.	1979		4,75
104	01	Иванов К. К.	1977		4,5

УСПЕВАЕМОСТЬ					
Номер группы	**Ном.студента**	**Код предм.**	**Таб.н.препод**	**Вид занятия**	**ОЦЕНКА**
101	01	01	101	лек	5
101	01	03	302	пр	0
101	02	01	101	лек	5
101	02	03	302	пр	0
101	03	01	101	лек	4
101	03	03	302	пр	0
101	04	01	101	лек	3
101	04	03	302	пр	0

преподаватель				
таб. номер	**фио препод.**	**уч. степень**	**уч. звание**	**код кафедры**
102	Апухтин И. С.	канд. техн. наук	доцент	01
103	Глухое И. Н	канд. техн. наук	доцент	01
104	Сеченов И. Б.	канд. техн. наук	доцент	01
105	Чернов И. Л.	канд. техн. наук	доцент 01	01
201	Блюмкина И. П.	р физ.мат. Наук	профессор	02
202	Львова И. Л.		ассистент	02
203	Шапошников Н. Д.	д-р техн. Наук	профессор	02
204	Новиков П. К.		ассистент	02
301	Ильясов И.Т	канд. фил. наук	доцент	03
302	Пустыцев Н. О.	канд. ист. Наук	доцент	03
303	Романов Н. А.	канд. ист. Наук	доцент	03
304	Цветков А. И.	канд. ист. Наук	доцент	03
401	Сорокина М. Ф.	канд. фил. наук.	доцент	04
402	Богомолов П. В.	канд. фил. наук.	доцент	04
403	Лысова Н. К.	канд. фил. наук.	доцент	04
404	Шаповалова	канд. фил. наук.	доцент	04
405	Кудряшова Р. К.		ассистент	04
501	Жигарева К. Е.	канд. пед. Наук	доцент	05
502	Егорова И. Н.		ст. пред.	05
503	Ермолин Р. П.		ассистент	05
601	Логинов А. М.	канд. фил. наук	доцент	06
602	Яковлев В. П.	канд. фил. наук	доцент	06
603	Раков А. Н.	канд. фил. наук	доцент	06