МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

УРАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ЛЕСОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Часовских В.П., Воронов М.П., Акчурина Г.А., Кох Е.В., Нохрина Г.Л.

ПОСТРОЕНИЕ КОРПОРАТИВНЫХ СИСТЕМ ИНФОРМАЦИОННОГО ОБСЛУЖИВАНИЯ УПРАВЛЕНЧЕСКОЙ ДЕЯТЕЛЬНОСТИ

Екатеринбург 2018

УДК 004.42

Рецензент:

Доросинский Леонид Григорьевич, профессор, доктор технических наук, директор департамента радиоэлектроники и связи Уральского федерального университета им. первого Президента России Б.Н. Ельцина

Часовских В.П., Воронов М.П., Акчурина Г.А., Кох Е.В., Нохрина Г.Л.
Построение корпоративных систем информационного обслуживания управленческой деятельности. Учебное пособие. – Екатеринбург: Уральский государственный лесотехнический университет, 2018. 200 с.

ISBN 978-5-6041352-6-6

В рамках данного учебного пособия рассматриваются основные аспекты проектирования и реализации корпоративных информационных систем. Представлены понятие и описание корпоративных информационных систем; их классификация; типовой модульный состав корпоративных информационных систем; дано описание и практическое руководство по созданию элементов корпоративных информационных систем в средах SPoD и 1С:Предприятие; представлены принципы проектирования баз данных и основы программирования, необходимые для создания и эксплуатации корпоративных систем управления.

Структура и содержание учебного пособия полностью соответствует требованиям федеральных государственных образовательных стандартов высшего образования (ФГОС ВО). Результаты данной работы были использованы при создании Программного обеспечения адаптации сайта образовательного процесса к реализуемой образовательной программе - Свидетельство о государственной регистрации программы для ЭВМ № 2015616260 зарегистрировано в Реестре программ для ЭВМ 04 июня 2015 г.; а также Программы управления динамически настраиваемым сайтом основных сведений об образовательной организации высшего профессионального образования - Свидетельство о государственной регистрации программы для ЭВМ № 2015617724 зарегистрировано в Реестре программ для ЭВМ 21 июля 2015 г. (см. Приложение).

Для студентов очной и заочной форм обучения направлений 38.03.02 Менеджмент (бакалавриат), 38.04.02 Менеджмент (магистратура), 09.03.03 Прикладная информатика (бакалавриат), 09.04.03 Прикладная информатика (магистратура), 27.03.02 Управление качеством (бакалавриат), 27.04.02 Управление качеством (магистратура).

Печатается по решению редакционно-издательского совета
Уральского государственного лесотехнического университета

ISBN 978-5-6041352-6-6

Оглавление

Информация об авторах

Часовских Виктор Петрович – профессор, доктор технических наук, заведующий кафедрой менеджмента и управления качеством, директор Института экономики и управления Уральского государственного лесотехнического университета.

Воронов Михаил Петрович – доцент, кандидат технических наук, заведующий кафедрой информационных технологий и моделирования Уральского государственного лесотехнического университета.

Акчурина Галия Абдулазисовна – доцент кафедры менеджмента и управления качеством Уральского государственного лесотехнического университета.

Кох Елена Викторовна – кандидат сельскохозяйственных наук, доцент кафедры информационных технологий и моделирования Уральского государственного лесотехнического университета.

Нохрина Галина Львовна – старший преподаватель кафедры информационных технологий и моделирования Уральского государственного лесотехнического университета.

Корпоративные системы информационного обслуживания управленческой деятельности

1. Понятие корпоративных информационных систем

Управление предприятием представляет собой совокупность трех основных функций хозяйственной деятельности - планирования, учета и оперативного регулирования. Этими функциями характеризуется управленческая работа административного персонала предприятия в сферах материально-технического снабжения, производства изделий и услуг, сбыта продукции.

В современных условиях перечисленные процессы, с целью повышения эффективности управления и обеспечения оперативности обработки информации, моделируются в рамках автоматизированных систем управления предприятием (АСУП).

Концепция АСУП состоит в том, что на каждую отдельно взятую функцию (планирование, учет, оперативное регулирование и т.п.), либо часть такой функции (технико-экономическое планирование, оперативно-производственное планирование, оперативный учет, бухгалтерский учет, управление материально-техническим снабжением и т.д.) в составе АСУП выделяют функциональную подсистему, обслуживающую все сферы промышленного производства и непромышленную деятельность предприятия. Комплексный подход к созданию функциональных подсистем АСУП обеспечивает сокращение информационных потоков при наиболее полном удовлетворении потребностей управляющих и управляемых подразделений обработанной информацией.

Функциональные подсистемы АСУП основаны на так называемом комплексе информационного обеспечения (ИО). Элементами ИО являются нормативы, классификаторы, документация, массивы информации. К основам информационной системы относится также программное обеспечение подсистем - алгоритмы и программы.

Программное обеспечение (ПО) АСУП выступает в виде комплекса алгоритмов и машинных программ обработки экономической информации, а также инструкций по их применению и корректировке. Различают внешнее ПО, предназначенное для решения специфических задач планирования, учета

и оперативного регулирования, и внутреннее ПО, обеспечивающее автоматическое управление счетом.

В настоящее время, наиболее современной и эффективной формой АСУП является корпоративная информационная система (КИС). КИС промышленного предприятия - это информационная система, обеспечивающая производственные и бизнес-процессы предприятия.

Под КИС подразумевают системы для крупных, часто территориально распределенных организаций, обычно имеющих несколько уровней управления. Обязательным свойством КИС является ее системность, то есть понимание КИС как объекта, имеющего новое системное свойство, которым не обладают отдельные компоненты.

Компоненты КИС предназначены для автоматизации большей части бизнес-процессов, которые в любой организации можно разделить на две группы:

– основные бизнес-процессы (бизнес-процессы, которые дают результат для клиента);

– вспомогательные бизнес-процессы (бизнес-процессы, дающие результат для основного бизнес-процесса или организации).

Основные бизнес-процессы часто идентичны для организаций, принадлежащих одной отрасли, но могут являться и уникальными. Для автоматизации таких процессов используются отраслевые решения или специально разработанные под заказ системы.

Вспомогательные же бизнес-процессы в общем виде являются схожими во всех организациях (бухгалтерия, логистика, складской учет, кадровая служба и т.п.), различия же проявляются на уровнях детализации.

Существует ряд свойств, которыми должна обладать любая корпоративная система:

– Интегрируемость системы, т.е. изменение в одной части системы (например, изменения запасов на складе) должны автоматически изменять показатели в других ее частях (в данном случае, в бухгалтерских проводках).

– Автоматизация процедур, т.е. все процедуры, выполнявшиеся вручную или на бумаге, должны автоматически, или по запросу пользователя выполняться в рамках информационной системы.

– Моделирование бизнес-процессов и процедур, которые существуют в организации.

– Предоставление руководителям оперативной информации в объеме, достаточном для принятия оперативных решений.

– Простота системы в обучении и использовании, т.е. рядовой сотрудник должен научиться выполнять свои обязанности при помощи системы за максимально короткое время.

– Адаптивность системы, т.е. возможность редактирования сотрудниками, наделенными соответствующими правами, отчетов и документов, изменения их формы и создания собственных форматов.

– Наличие процедур контроля работы пользователей, сводящие ошибки к минимуму.

– Автоматическое создание контрольных точек системы, и отражение информации о каждом пользователе, изменяющем или вносящем каждую запись.

– Наличие системы защиты данных и системы распределения прав доступа к данным.

2. История развития корпоративных информационных систем

Истоки возникновения КИС относятся к началу 60-х гг., когда уже был накоплен некоторый опыт решения задач обработки экономической информации средствами технологий файловых систем и языка COBOL. Это были технологии, основанные на использовании магнитных лент с их последовательным доступом. Появление таких устройств памяти прямого доступа, как магнитные диски вычислительных систем IBM/360 или ICL-1900, открыло принципиально новые возможности и стимулировало поиски новых эффективных методов организации хранения слож-ноструктурированных данных большого объема.

В начале 60-х гг. были созданы первые системы управления базами данных. Среди них СУБД общего назначения IDS (Integrated Data Storage, 1963 г.), разработанная в компании General Electric под руководством Чарльза Бахмана. Эта система интересна не только тем, что она была одной из первых коммерческих СУБД. Реализованные в ней принципы организации базы данных и манипулирования данными стали впоследствии основой сетевой модели данных. В этот же период начинают формироваться основы

методологии построения систем баз данных, которая вскоре стала играть основополагающую роль в разработке АСУП и АСУТП как в нашей стране, так и за рубежом. Одним из ключевых элементов этой методологии является концепция модели данных.

Появление сетевой и иерархической моделей данных создают условия для возникновения первых систем решения задач управления на предприятии, ставших прототипами современных корпоративных информационных систем. Такие системы в основном охватывали сферу складского или материального учета (IC – Inventory Control).

В конце 60-х гг. начала формироваться и индустрия программного обеспечения систем баз данных. В это время были созданы широко известные коммерческие СУБД общего назначения, которые вот уже более трех десятилетий не сходят со сцены. В этот период компания IBM разработала свою знаменитую систему IMS (1969 г.), основанную на иерархической модели данных. Развитие систем баз данных стимулировало развитие систем автоматического управления предприятиями. Идея создания модели данных в рамках организации стала привлекать внимание международных промышленных компаний, которые искали способ упростить управление производственными процессами.

Первым шагом в данном направлении стала разработка концепции MRP, планирование материальных ресурсов (Materials Resource Planning), включавшее только планирование материалов для производства. Основная концепция MRP в том, чтобы минимизировать издержки, связанные со складскими запасами (в том числе и на различных участках в производстве). В основе этой концепции лежит следующее понятие - Bill Of Material (BOM - спецификация изделия, за которую отвечает конструкторский отдел), который является основой для выявления спроса на сырье, полуфабрикаты и пр. в зависимости от плана выпуска (бюджета реализации) готовой продукции. На основании плана выпуска продукции, BOM и технологической цепочки осуществляется расчет потребностей в материалах, относительно задаваемых сроков. При этом количество производственных мощностей, их загрузка, оплата труда рабочих и прочие факторы не учитывались.

Период 70-х гг. чрезвычайно богат новыми идеями и подходами в области управления данными, фундаментальными исследованиями, затрагивающими фактически все важнейшие аспекты организации и функционирования систем баз данных, исследовательскими прототипами и коммерческими программными продуктами. В 70-е гг. было выпущено довольно большое число коммерческих СУБД, основанных на различных моделях данных, и многие из них имели значительное число установок. Так, был создан ряд СУБД для иерархических систем, наиболее распространенными из которых были System 2000 (MRI Systems Corp.), а также несколько новых версий флагманского продукта того времени компании IBM - системы IMS, среди которых IMS/VS. Появился также ряд новых СУБД типа CODASYL - DMS-1100 (UNIVAC), IDS/II (Honeywell), DBMS-10/20 (Digital), новая версия системы IDMS (Cullman). В этот период были выпущены также известные системы - ADABAS (Software AG), использующая технику инвертированных списков, и TOTAL (Cincom), основанная на модели связанных файлов и обычно относимая к категории сетевых систем.

В 70-е гг. довольно многое было сделано в разработке и исследовании различных подходов к моделированию данных. В связи с этим нужно, прежде всего, отметить инициированный циклом работ Э. Кодда ряд публикаций по математической теории реляционных баз данных. К числу их главных итогов можно отнести: создание теории зависимостей и базирующейся на ней теории нормализации отношений, которая стала основой проектирования реляционных баз данных; разработку алгоритмов редукции выражений реляционного исчисления в реляционную алгебру; первые шаги в исследовании неопределенных значений и проблемы неполноты информации; ряд результатов, связанных с понятием универсального отношения.

Благодаря успешному созданию исследовательских прототипов и первых коммерческих СУБД конца 70-х гг. реляционная теория стала активно применяться на практике. В течение 80-х гг. быстро увеличивалось число коммерческих реализаций реляционных СУБД для различных программно-аппаратных платформ. В 1981 г. поставку своей первой коммерческой реляционной СУБД SQL/DS начала компания IBM. Спустя два

года компания выпустила новую систему DB2, положившую начало линии SQL-серверов баз данных, поставляемой IBM до настоящего времени. Был создан ряд новых коммерческих версий системы Ingres, в том числе поддерживающих язык SQL. Было выпущено несколько новых версий Oracle, появилась реляционная СУБД Rdb компании Digital для платформы VAX/MVS.

В 80-х гг. на основе расширенной концепции MRP была разработана концепция планирования всех производственных ресурсов предприятия MRPII (Manufacturing Resource Planing - планирование производственных ресурсов). В соответствии с концепцией MRPII информационная система должна выполнять следующие функции:

1. Планирование развития бизнеса (составление и корректировка бизнес-плана);
2. Планирование деятельности предприятия;
3. Планирование продаж;
4. Планирование потребностей в сырье и материалах;
5. Планирование производственных мощностей;
6. Планирование закупок;
7. Выполнение плана производственных мощностей;
8. Выполнение плана потребности в материалах;
9. Осуществление обратной связи.

Каждая из перечисленных функций обеспечивается в системе соответствующим программным модулем.

К началу 90-х гг. реляционные СУБД стали составлять доминирующую долю установок СУБД практически на всех распространенных аппаратно-программных платформах. Реляционные СУБД стали использоваться для создания приложений разного масштаба в самых различных областях применения - от офисных до крупных корпоративных. Однако, в 90-х получают развитие и многомерные модели данных, самыми распространенными из которых являются постреляционная и объектно-ориентированная модели данных. Образованная в 1991 году компания Arbor Software (теперь Hyperion Solutions) в качестве своей специализации выбрала создание многопользовательских серверов многомерных баз данных; результатом этих работ стала система Essbase. Позже Arbor лицензировала

базовую версию Essbase корпорации IBM, которая в дальнейшем легла в основу DB2.

В начале 90-х сложилась еще одна важная концепция — крупные хранилища данных, которые, как правило, базируются на схемах «звезда» и «снежинка». При таком подходе для реализации многомерных баз удается использовать технологию реляционных баз данных. И уже в 1993 году Э.Ф. Кодд ввел термин системы оперативной аналитической обработки (online analytical processing - OLAP).

Появление и совершенствование технологий хранения и обработки данных привели к изменениям в подходе к построению КИС. Концепция MRPII была усовершенствована и дополнена возможностями по учету оставшихся затрат предприятия. Таким образом, появилась концепция ERP (Enterprise Resource Planning - Планирование ресурсов предприятия). В основе концепции ERP лежит принцип создания единого хранилища данных, содержащего всю деловую информацию, накопленную организацией в процессе ведения деловых операций, включая финансовую информацию, данные, связанные с производством, управлением персоналом, или другие сведения. При этом любая часть информации, которой располагает данная организация, становится одновременно доступной для всех работников, обладающих соответствующими полномочиями. В настоящее время практически все современные западные производственные системы и основные системы управления производством базируются на концепции ERP и отвечают её рекомендациям.

Наиболее полной и современной концепцией КИС является CSRP (Customer Synchronized Resource Planning). Она разработана в конце XX столетия на основе концепции ERP, и дополнена процедурами взаимодействия с клиентами (оформление заказа, поддержка заказчика на местах и пр). Если MRP, MRPII, ERP ориентировались на внутреннюю организацию предприятия, то CSRP включил в себя полный цикл от проектирования будущего изделия, с учетом требований заказчика, вплоть до гарантийного и сервисного обслуживания после продажи. Основная суть концепции CSRP в том, чтобы интегрировать Заказчика (Клиента, Покупателя и пр.) в систему управления предприятием.

3. Классификация корпоративных информационных систем

Обобщая опыт создания КИС, отметим, что основой для разработки любых корпоративных информационных систем промышленных предприятий являются концепции и методологии планирования и управления различными видами ресурсов предприятия. Основными концепциями являются:

1. MPS (Master Planning Shedule) – календарное планирование, база для планово-ориентированной методологии.

2. MRP (Material Requirements Planning) - планирования материальных ресурсов.

3. CRP (Capacity Requirements Planning) – планирование производственных ресурсов.

4. FRP (Finite Requirements Planning) - планирование производственных ресурсов в условиях ограниченных мощностей.

5. FRP (Finance Requirements Planning) - планирование финансовых ресурсов.

6. MRPII (Manufacturing Resource Planning) - интегрированная методология планирования производственных ресурсов (включает MRP и CRP, а также часто использует MPS и FRP).

7. ERP (Enterprise Planning) – интегрированная система планирования ресурсов предприятия (включает MPS, MRP, CRP, FRP обеспечивая возможностью динамического изменения плана). Фактически концепция ERP является усовершенствованной, более современной версией MRPII.

8. CSRP (Customer Synchronized Resource Planning) – планирование ресурсов (включает концепцию ERP) и цикл взаимодействия с клиентами.

Реализация любой из перечисленных концепций в рамках КИС промышленного предприятия требует капиталовложений. Основными статьями затрат являются затраты на программное обеспечение, аппаратного обеспечения, внедрение, администрирование, техническое обеспечение КИС. Также, требуются дополнительные затраты на проведение модификации КИС в случаях появления новых информационных и производственных технологий; освоения предприятием новых видов производств; создания или ликвидации структурных подразделений предприятия; изменения способов и

методов проведения расчетов или изменение стандартов отчетных форм; прочих структурных изменений предприятия.

Все системы на российском рынке могут быть условно разделены на три группы:

1. Локальные системы. Предназначены, в основном, для автоматизации учета по одному или нескольким направлениям (бухгалтерия, сбыт, склады, учет кадров и т.д.). Локальной системой может воспользоваться практически любое предприятие, нуждающееся в управлении финансовыми потоками и в автоматизации учетных функций. Локальные системы по многим критериям универсальны, но ряд разработчиков предлагает отраслевые решения, например, особые способы начисления налогов и т.п. Стоимость локальных систем колеблется в диапазоне $5 000 - $50 000.

2. Финансово-управленческие системы. Такие системы гибко настраиваются на нужды конкретного предприятия, хорошо интегрируют деятельность предприятия и предназначены, в первую очередь, для учета и управления ресурсами непроизводственных компаний. Как правило, они универсальны, однако возможности для отражения специфики деятельности конкретной компании в таких системах шире, чем у локальных. Во многих системах данного класса присутствуют базовые возможности управления производством. Стоимость финансово-управленческих систем можно условно определить в диапазоне от $50 000 до $200 000.

3. Средние интегрированные системы. Предназначены для управления производственным предприятием и интегрированного планирования производственного процесса. Цепочка планирования "сбыт - производство - закупки" является ядром этих систем. Подразделения предприятия (финансы, бухгалтерия, маркетинг и пр.) строят свою деятельность, опираясь на данные этой цепочки. Средние системы значительно сложнее в установке: цикл внедрения занимает от 6 месяцев до полутора лет и более. Причина в том, что система покрывает потребности подразделений и полностью интегрирует производственное предприятие, что требует значительных совместных усилий сотрудников предприятия и поставщика КИС. Средние системы по многим параметрам значительно превосходят финансово-управленческие. Стоимость внедрения средних систем начинается, как и у финансово-

управленческих систем, в районе $50 000, но, в зависимости от охвата проекта, может достигать $500 000 и более.

4. Крупные интегрированные системы. Отличаются от средних набором вертикальных рынков и глубиной поддержки процессов управления большими многофункциональными группами предприятий (холдингов или ФПГ). Такие системы имеют наибольшую функциональность, включая управление производством, управление сложными финансовыми потоками, корпоративную консолидацию, глобальное планирование и бюджетирование и пр. Сходные функции присутствуют и во многих финансово-управленческих и средних интегрированных системах, однако, с более низкой степенью проработки. Сроки внедрения крупных интегрированных систем обычно занимают более года, а стоимость проекта - более $500 000.

4. Модульный состав корпоративных информационных систем

В общем виде модель КИС промышленного предприятия (а также любого из программных модулей КИС) может быть представлена в виде схемы (рис. 1.).

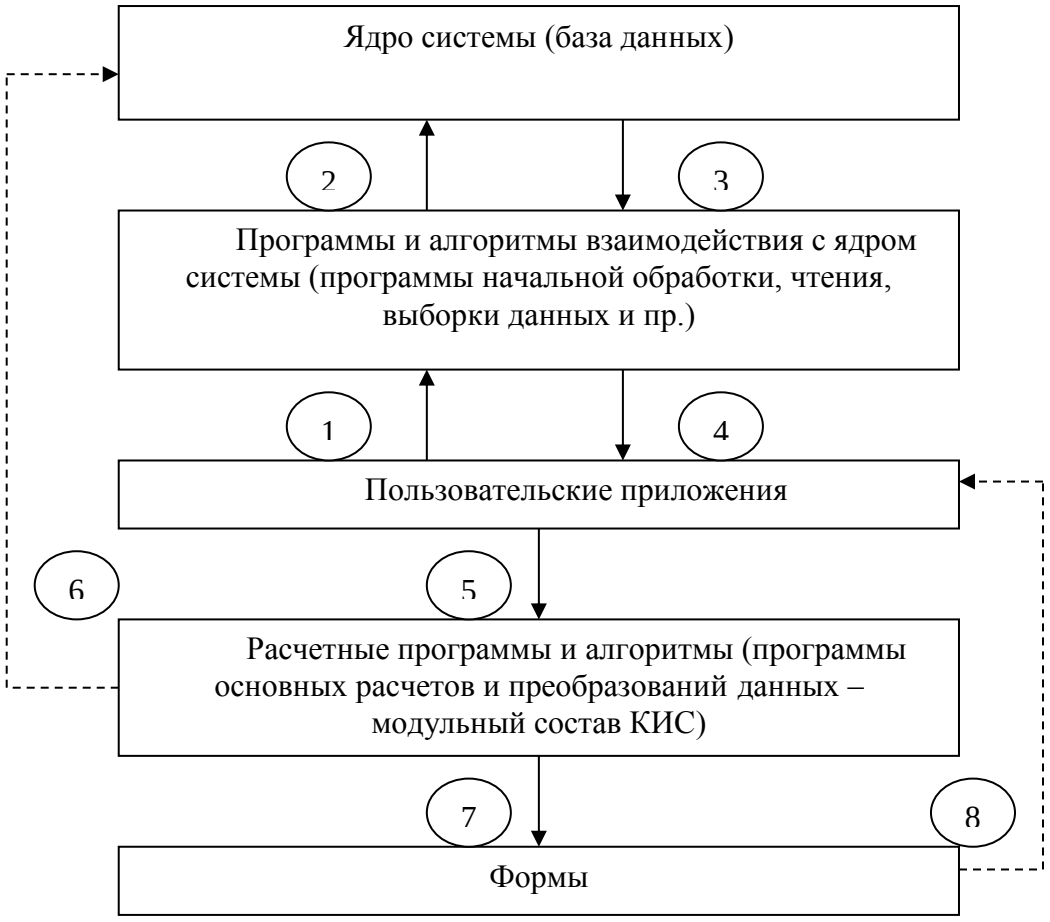


Рис 1. Компоненты программных модулей КИС

На рис. 1. введены следующие условные обозначения:

1 – обращение к программам управления данными (запуск программ).

2 – запросы данных, проведение выборки данных, внесение данных и пр.

3 – данные.

4 – данные, результаты выборки, результаты.

5 – обращение к программам расчета величин и показателей на основе полученных данных (запуск программ).

6 – результаты проведения анализа.

7 – представление результатов расчетов в виде форм, результаты.

8 – формы.

Компоненты программных модулей КИС разделяются на 5 видов:

Ядро системы (БД). В файлах структуры БД фиксируются структура хранения данных и свойства обработки для каждой единицы данных (для каждого поля данных).

Программы и алгоритмы взаимодействия с ядром системы. Основное назначение данных программ – сообщение с БД. Все программы и алгоритмы можно подразделить на следующие группы:

– программы первичной обработки данных и ввода данных в БД (в т.ч. программы проверки вводимых данных на соответствие);

– программы, осуществляющие чтение данных в БД в заданной последовательности;

– программы, осуществляющие выборку данных из БД по заданным параметрам (критериям);

– программы интерпретации показателей и индикаторов;

– программы расчета значений на основе запрашиваемых данных.

Пользовательские приложения. Назначение пользовательских приложений в рамках КИС промышленного предприятия – управление данными. Пользовательские приложения обеспечивают сотрудников предприятия возможностями:

– Получения данных и результатов выборки за различные периоды.

– Получения форм ввода данных для внесения в БД и дальнейшей обработки. Также, в целях избежания ошибок ввода рекомендуется

предварительная проверка введенных данных на соответствие требованиям к формату перед осуществлением процедуры внесения данных в БД.

– Получения форм редактирования ранее введенных данных.

– Получения форм удаление ранее вводимых данных.

– Получения форм отображения полученных данных.

– Получения форм манипулирования данными, а также возможность вызова программ дальнейшего расчета.

– Получения и печати отчетных форм.

Расчетные программы и алгоритмы. Расчетные программы и алгоритмы осуществляют основные расчеты и преобразования и образуют модульный состав КИС промышленного предприятия. Это математические и статистические методы, оптимизационные модели, модель «дерево решений» и т.д.

Формы. Все результаты расчетов отображаются в виде стандартных форм (в соответствии ГОСТам, либо внутренним стандартам).

КИС промышленного предприятия, разрабатываемая в соответствии с концепцией ERP, должна состоять из следующих функциональных модулей, каждый из которых формируется определенным набором компонентов КИС:

1. Программный модуль управления производством (отражает специфику предприятия).
2. Программный модуль управления производственными запасами.
3. Программный модуль управления сбытом готовой продукции.
4. Программный модуль управления учетной деятельностью.
5. Программный модуль управления планово-аналитической деятельностью.

Вопросы для самопроверки

1. Сущность и назначение автоматизированных систем управления предприятием (АСУП).
2. Понятие и свойства корпоративных информационных систем (КИС).
3. Возникновение и развитие моделей данных и систем управления базами данных.
4. Эволюция концепции корпоративных информационных систем.
5. Концепции и методологии планирования и управления ресурсами.
6. Классификация КИС.

7. Компоненты программных модулей КИС.
8. Перечень функциональных модулей КИС промышленного предприятия.

Литература

1. Часовских В.П., Воронов М.П. Исследование системных связей и закономерностей функционирования корпоративной информационной системы лесопромышленного предприятия в среде ADABAS и Natural: Монография, электронное издание. 2 изд. испр. и доп. – Екатеринбург: Урал. гос. лесотехн. ун-т, 2012. 180 с.
2. Информационные технологии в бизнесе/Под ред. М. Железны - СПб: Питер, 2002. – 1120 с.
3. Советов Б.Я. Цехановский В.В. Информационные технологии. 6-е изд. Учебник для бакалавров. – М.: Юрайт, 2013. – 272 с.
4. Бочаров Е.П. Интегрированные корпоративные информационные системы. Учебное пособие. – М.: Финансы и статистика, 2007. - 288 с.
5. Душин В.К. Теоретические основы информационных процессов и систем: Учебник. – М.: Издательство «Дашков и К», 2012. – 348 с.
6. Самардак А.С. Корпоративные информационные системы: Учебное пособие. - Владивосток: ТИДОТ ДВГУ, 2003. - 252 с.
7. Глинских А.А. Мировой рынок КИС // Компьютер-Информ. Май 2000. № 10(80).
8. Баранов В.В. и др. Информационные технологии и управление предприятием. – М.: Компания АйТи, 2004. – 328 с.
9. Сухомлин В.А. Введение в анализ информационных технологий: Учебник. – М.: Горячая линия – Телеком, 2003. – 427 с.
10. Козырев А.А. Информационные технологии в экономике и управлении: Учебник. – СПб.: Изд-во Михайлова В.А., 2000. – 360 с.
11. Гасанов Э.Э., Кудрявцев В.Б. Теория хранения и поиска информации. – М.: ФИЗМАТЛИТ, 2002. – 288 с.
12. Бажин И.И. Информационные системы менеджмента. – М.: ГУ-ВШЭ, 2000. – 668 с.
13. Титоренко Г.А. Автоматизированные информационные технологии в экономике. – М.: Юнити, 2006. – 400 с.

14. Хомоненко А.Д., Цыганков В.М., Мальцев М.Г. Базы данных: Учебник/ Под ред. Проф. А.Д. Хомоненко. – СПб.: Корона, 2000. – 416 с.

15. Горшков А.Ф. и др. Компьютерное моделирование менеджмента: Учебное пособие. – М.: Экзамен, 2004. – 528 с.

16. Свириденко С.С. Современные информационные технологии. – М.: Радио и связь, 1989. – 304 с.

17. Дж. Лодон, К. Лодон. Управление информационными системами. 7-е изд./Пер. с англ. – СПб.: Питер, 2005. – 912 с.

18. В.В. Дик Методология формирования решений в экономических системах и инструментальные среды их поддержки. – М: Финансы и статистика, 2001. – 300 с.

Состав и инструментарий единой среды разработки приложений SPoD

SPoD (Single Point of Development) – единая среда разработки прикладных приложений, позволяющая конструировать приложения для различных платформ (Windows, Unix, Mainframe), для различных СУБД (Adabas, Tamino, Oracle, DB2 и пр.), а также с использованием различных технологий (DCOM, SOAP, Web Services, и пр.)

SPoD удовлетворяет следующим требованиям и предлагает следующие преимущества:

– Клиент/сервер архитектуры, поддерживающий одну единственную удаленную среду разработки для всех платформ.

– Единый, знакомый образ всех объектов включаемых в прикладную разработку.

– Расширенная управляемая помощь/документация для дистанционных сред разработки.

– Увеличение производительности разработки благодаря мощной графической рабочей среде.

– Управление над заданиями разработки.

– Низкие издержки для разработки программного обеспечения, эксплуатации и администрирования.

5. Архитектура среды SPoD

Технология разработки приложений SPoD использует редактор Natural Studio в среде Microsoft Windows в качестве клиента разработки, Natural Development Server и Natural на серверной платформе, а также TCP/IP в качестве транспортного протокола. Архитектура системы SPoD показана на рис. 2.

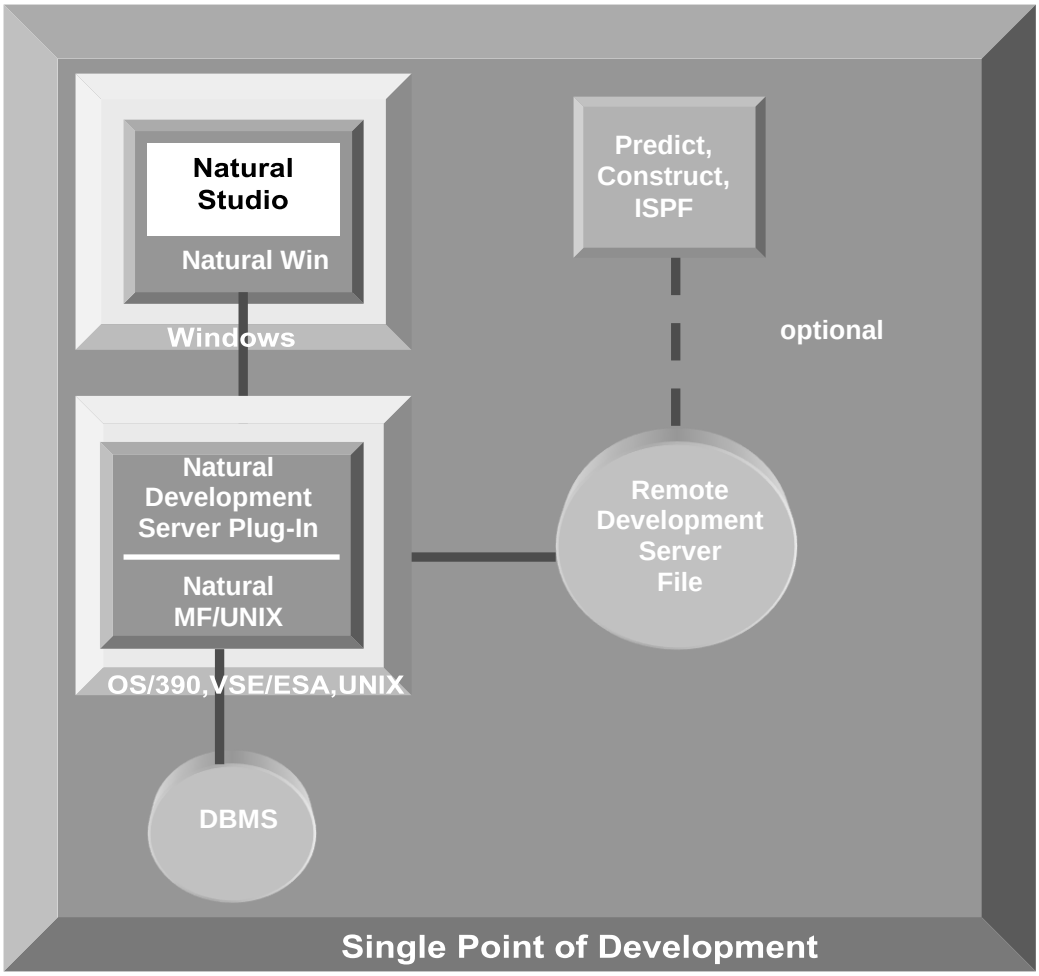


Рис. 2. Архитектура системы SPoD

Система SPoD включает следующие средства:

– Natural для Windows – клиент удаленной разработки;

– Natural Development Server – сервер разработки;

– Development Server File – файл хранения данных;

– Database Management System – система управления базами данных;

– Predict and Natural Construct – идентичность файла хранения данных.

Natural для Windows – клиент удаленной разработки. При использовании в качестве клиента удаленной разработки, Natural Studio является рабочей станцией центральной разработки для всех остальных платформ. Все Natural-связанные прикладные разработки, включая шаги конфигурации, можно осуществлять из среды Natural Studio, независимо от платформы, на которой находится серверная среда разработки. Большинство функций, необходимых для удалённой разработки приложений, таких как администрирование, удаленное редактирование, компиляция, отладка, и т.п., интегрировано в Natural Studio.

Natural Development Server – сервер разработки. Используется для удаленной разработки на серверной платформе. Сервером разработки является Natural вместе с добавленным и установленным продуктом Natural Development Server (NDV).

Использование удаленного сервера разработки обеспечивает следующие функции:

– предоставляет доступ к системным файлам;

– обеспечивает доступ к прикладным данным;

– гарантирует согласованность модификаций прикладных объектов;

– выполняет дистанционные команды, задаваемые клиентом разработки.

Development Server File – файл хранения данных. Файл сервера разработки (Development Server File) хранит прикладные данные.

Database Management System – система управления базами данных. Дистанционная среда разработки Natural может использоваться с любым компонентом, поддерживающим систему управления базы данных, которая доступна для серверной платформы и выполняется под существующей операционной системой. В целях обеспечения максимальной производительности рекомендуется использовать СУБД ADABAS и Tamino, которые, как и система SPoD, являются продуктами фирмы Software AG.

Predict and Natural Construct – идентичность файла хранения данных. Predict является встроенным словарем данных, основная функция которого – автоматическая генерация программного кода на основе свойств задаваемых объектов. Natural Construct – это генератор прикладных систем, предназначенный для обеспечения гибкости структуры пользовательских приложений.

6. Функции и свойства среды SPoD

Основными функциями среды SPoD являются:

– Определение и генерация базы данных (ADABAS, или иная профессиональная СУБД).

– Разработка программ с использованием сервисов систем обработки транзакций (CICS, Complete).

– Генерация программ.

– Документирование приложений.

– Реинжениринг и поддержка приложений.

– Работа в среде Natural Studio с объектами операционной системы (очередями заданий, наборами данных и т.д.).

Основными свойствами среды SPoD являются:

– Дистанционная обработка объекта. Т.е. возможность манипулирования программными объектами, независимо от физического размещения.

– Дистанционное редактирование. Возможность извлечения исходных файлов, размещенных на сервере и редактирования их на рабочей станции пользователя с последующим сохранением их в месте физического размещения.

– Дистанционная компиляция. В случае компиляции с рабочей станции пользователя, исполнение команды на сервере.

– Удаленная отладка. Возможность отладки приложений, выполняемых на сервере, которая может располагаться в той же системе или в универсальных Natural средах сервера.

– Информация перекрестных ссылок. Тиражирование информации об объектах и их ссылках сохраняется в файл сервера.

– Блокировка объектов. В случае доступа к дистанционному серверу разработки, предотвращение параллельной коррекции посредством блокировки. Блокировка информации держится в файле сервера разработки.

– Поддержка окна эмуляции терминалов. Эксплуатация универсальных приложений часто включает испытание выхода на терминалы. Для этой цели окно эмуляции терминалов выдается автоматически.

7. Компоненты среды SPoD

Рабочее место разработчика в среде Microsoft Windows использует встроенную систему Natural Studio со следующими компонентами:

– Program Editor – редактор программ.

– DDM Editor – редактор DDM.

– Data Editor – редактор областей данных.

– Map Editor – редактор шаблонов.

– Dialog Editor – редактор диалога.

– Navigator – средство поиска и управления объектами.

– Debugger – отладчик программ.

– Natural Data Browser – средство просмотра данных в БД.

– Component Browser – просмотр объектов библиотек DCOM.

– Natural Reporter – средство составления отчетов.

В качестве дополнительных компонент могут быть активизированы другие средства разработки:

– Predict - словарь данных.

– Natural Construct - генератор прикладных систем.

– Natural Engineer - средство реинжиниринга приложений.

– Natural ISPF - встроенное структурное средство программирования.

Natural Studio включает ряд специализированных редакторов, каждый из которых отвечает за определенную функцию, что способствует эффективной и удобной разработке и модификации программных объектов Natural. Эти редакторы, а также средства просмотра библиотек и прочие инструменты объединены в интегрированную систему, называемую *Natural Studio*.

Program Editor используется для редактирования программ, подпрограмм, вложенных процедур, копикодов и текстов, разрабатываемых средствами Natural.

DDM Editor - редактор описаний таблиц баз данных (Data Definition Modules).

Data Editor - редактор описаний областей данных, таких как Local Data Area, Global Data Area, Parameter Data Area и Object Data Area. Области данных допускают использование в различных программных объектах.

Map Editor - редактор разработки экранов. Экраны Natural могут содержать фиксированный текст, области ввода и области вывода, а также ограниченный набор элементов графики, таких как меню, кнопки переключения, поля прокрутки и изображения. При проектировании панели допускается импортировать поля из других программных объектов Natural, что уменьшает разночтения в написании имен переменных и определений типов.

Dialog Editor - редактор разработки стандартизированных GUI-диалогов, которые управляются событийно-ориентированными средствами Natural. Также редактор может быть использован как редактор пользовательских форм и панелей.

Navigator – обеспечивает поиск и управление программными объектами Natural в пределах библиотек. Кроме того, Navigator предлагает функции массовой обработки, рассчитанные на целые наборы программных объектов.

Debugger - развитая система символьной отладки, которая дает пользователям возможность анализировать программу по шагам, расставлять точки прерывания и наблюдать за изменениями значений переменных. Отладчик может работать также в удаленном режиме.

Natural Data Browser обеспечивает быстрый доступ к данным ADABAS. Позволяет просматривать данные ADABAS для проверки, выбирать файлы и поля, генерировать отчеты.

Component Browser дает пользователям возможность просматривать библиотеку объектов для получения описания интерфейсов объектов, элементов управления или объектов автоматизации, для их использования в программе Natural. Component Browser автоматически создает код образца, который готов к копированию в программу и к немедленному использованию без выполнения явной процедуры импорта.

Natural Reporter - полнофункциональное средство составления отчетов, совместимое с любой СУБД, поддерживаемой ODBC. Natural Reporter позволяет создавать отчеты, в которых объединены данные, тексты, графики и рисунки. Редактором поддерживаются различные варианты шрифтов, обрамления и затенения. Предлагаются и специальные функции, в числе которых сортировка, выбор по критерию, поля формул, верхние и нижние колонтитулы для одной или нескольких страниц, разрывы группировок страниц, подсчет итогов в группе или во всем документе.

8. Система управления базами данных ADABAS

8.1. Общее описание

ADABAS — это постреляционная СУБД, спроектированная для управления сверхбольшими базами данных и решения жизненно-важных задач в рамках информационных систем больших и средних корпораций и организаций. Работая в субсекундном диапазоне времени отклика, ADABAS, тем не менее, может обслуживать десятки тысяч параллельно запрашивающих его услуги пользователей. Сегодня ADABAS — это наиболее эффективная СУБД с точки зрения приведенных затрат при обработке онлайновых деловых транзакций (операций). Она работает на

мэйнфреймах под управлением операционных систем IBM (OS/390, VSE/ESA, VM/CMS, OS/400), SNI (OSD) и Fujitsu (MSP). ADABAS также доступен на всех ведущих платформах UNIX, на OpenVMS, Windows 2000, Windows XP, Windows 2003.

Мультимодельность ADABAS в совокупности с рядом дополнительных возможностей, позволяет строить как традиционные иерархические, сетевые и реляционные базы данных, так и сложные текстовые информационно-поисковые и интегрированные системы и системы обработки изображений, постреляционные структуры для моделирования человеческой деятельности, экспертного анализа сложных производственных процессов и т.д. При этом сочетаются преимущества различных подходов. Можно спроектировать БД ADABAS в третьей нормальной форме и при этом, в целях повышения производительности часть связей преобразовать в иерархию. Таким образом, проектировщику предоставляется возможность выйти за рамки ограничений определенного подхода и, объединив преимущества всех подходов, достичь большей производительности и гибкости создаваемой системы на конкретных запросах.

Многообразие видов информационных систем и технологий становится возможным благодаря тому, что ADABAS обеспечивает поддержку следующих моделей и типов данных.

Непервая нормальная форма (NF2 - Non-First Normal Form). Эта модель данных традиционна для ADABAS с 1969 года, когда он впервые вышел на рынок систем обработки данных.

Традиционная реляционная модель данных. Эта модель соответствует ANSI/ISO стандарту SQL92 и реализована в виде либо надстройки над ADABAS, либо как неотъемлемая часть ADABAS D.

Модель данных сущность-связь (E/R модель). В ADABAS предусмотрено расширение до E/R модели (Entity-Relationship модель) для управления сложными структурами данных с высокой степенью связности, а также рекурсивные структуры данных (когда элемент структуры данных содержит один или несколько указателей на элементы такого же типа). Объединяя эту модель с другими моделями ADABAS можно строить чрезвычайно мощные интегрированные базы данных и, соответственно, прикладные системы. Предпочтительные области для применения E/R

моделей — системы представления знаний, моделирование поведения сложных технических и биологических систем, расчеты потребностей, планирование материальных ресурсов различного вида и назначения (Bills of Materials). Например, традиционные для последней предметной области проблемы информационного взрыва и управления циклами легко разрешимы с помощью возможностей E/R расширения ADABAS.

Обработка и управление произвольными текстами. Этот тип данных (и соответствующие средства манипулирования ими) обеспечивает доступ к документам, обрабатываемым ADABAS, как к произвольным текстам.

Изображения и аудиоинформация. Поддержка изображений и аудиоинформации в ADABAS основана на функции гипердескрипции (hyperdescriptor). Она позволяет использовать множественные значения индексов, выработанных внешним математическим обеспечением, или определенных пользователем, для концептуально однородных объектов (тезаурус). Эти значения обрабатываются далее ADABAS, делая, таким образом, возможным обращение к объектам и работу с ними при помощи других процедур, предоставляемых СУБД.

ADABAS имеет мощные и удобные в работе средства администрирования баз данных, реализованные как в интерактивном, так и в пакетном режимах, на всех серверных платформах.

На основе ADABAS в мире построено множество больших прикладных систем, как OLTP, так и OLAP. В то же время ADABAS может служить основой не только "традиционных" БД, но и хранилищ данных (Data Warehouse).

8.2. Основные возможности и характеристики СУБД ADABAS

Основные возможности и характеристики СУБД ADABAS

На рис. 3. показаны основные возможности СУБД ADABAS.

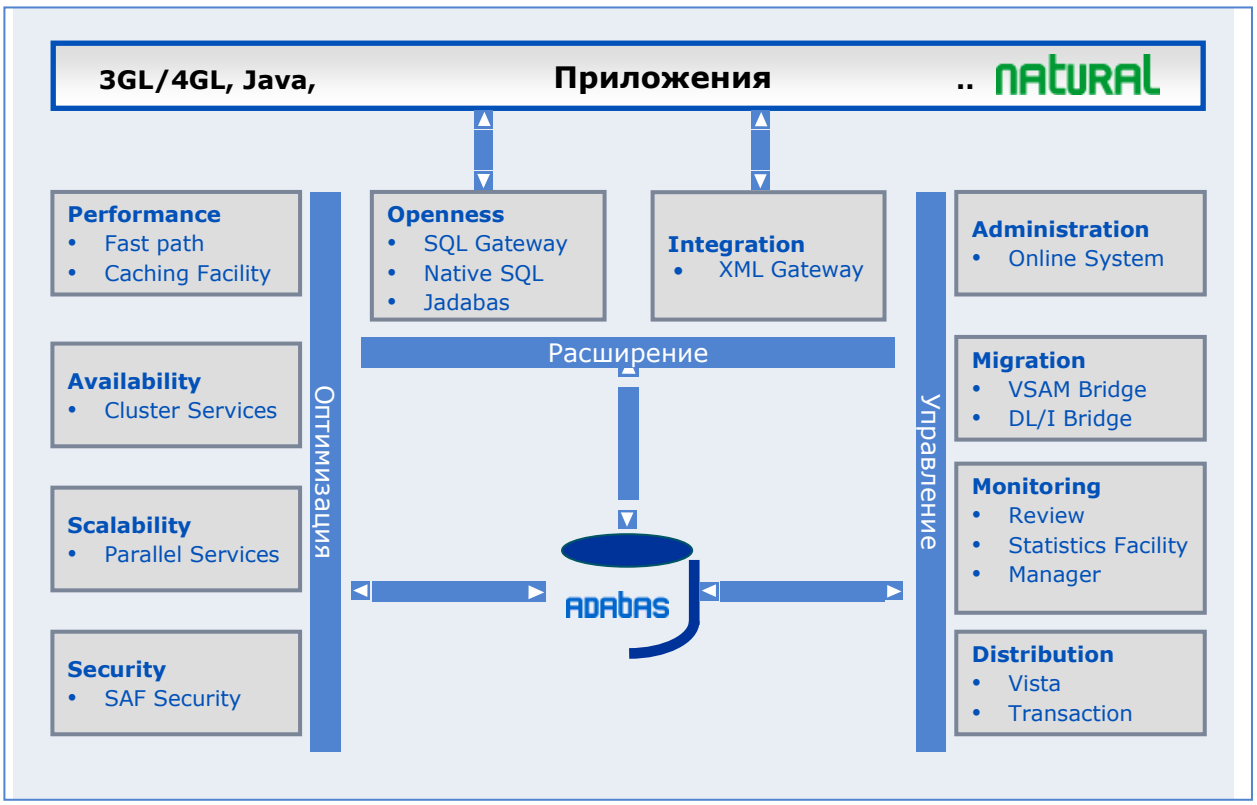


Рис. 3. Основные возможности СУБД ADABAS

Результаты тестов, проводившихся в лабораториях IBM:

- 5.000 транзакций в секунду.
- 160.000 вызовов в секунду.
- 40.000 параллельных пользователей.

Главные конструктивные особенности, обеспечивающие высокую эффективность ADABAS:

–Вложенные отношения позволяют уменьшать схему базы данных и увеличивать количество данных, передаваемых за одну операцию ввода-вывода.

–Автоматическое, независимое от платформы сжатие данных требует меньше объема памяти для хранения и позволяет оптимизировать процедуры ввода-вывода.

–Обусловленное спецификой приложений использование различных типов памяти ЭВМ сокращает время обработки данных и уменьшает число операций ввода-вывода.

–Блокировки на уровне строки (записи) в многопользовательском режиме максимально снижают проблемы доступа (клинча) к базам данных и улучшают условия для параллельной обработки данных.

–Способы хранения и доступа к данным в ADABAS отделены от особенностей конкретных физических носителей, что делает его гибким и эффективным.

8.3. Базовая модель и структура данных системы

Структура данных базовой модели

База данных ADABAS определяется как совокупность взаимосвязанных файлов и ассоциаций записей файлов.

Файл представляет собой именованную совокупность записей, имеющих одинаковую структуру, компонентами которой являются *атрибут*, *множественный атрибут*, *простая*, *составная* и *периодическая группа*. Каждой записи файла назначается системный ключ, представляющий собой число в диапазоне от 1 до 16 * 10^6, которое однозначно идентифицирует экземпляр записи и определяется в ADABAS как внутрисистемный номер.

Связи между файлами - типа 1:1, 1:M или M:N - являются непоименованными парными связями и предназначены для моделирования сетевых и иерархических отношений между объектами предметной области, а также групповых отношений в объектах иерархической структуры.

Ассоциация записей файлов (далее *ассоциация*) представляет собой совокупность записей файла, обладающих общим свойством. В качестве такого свойства может выступать, в частности, значение некоторого атрибута, которое совместно, с его именем определяет однозначно ассоциацию и может использоваться как ключ для доступа к ассоциации записей файла.

Ассоциации в ADABAS организуются в БД в виде списков *внутрисистемных номеров* (*ВСН*) записей файлов. Если ассоциация образуется на основе равенства значений некоторого атрибута, список ВСН записей, входящих в ассоциацию, называется *инвертированным списком,* а сам атрибут — *поисковым.* В том случае, когда ассоциация образуется на основе *равенства значений атрибута записей* одного файла, значению атрибута некоторой записи другого файла, соответствующий список ВСН называется *списком связи,* а указанные атрибуты файлов - *атрибутами связи.*

Использование ассоциаций реализует, по существу, разбиение записей файлов на *классы*, поддерживаемое в ADABAS динамически в процессе модификации БД. При этом ассоциации в виде инвертированных списков

обеспечивают возможность ускоренной селекции записей файлов по их содержимому, а ассоциации в виде списков связь-возможность поддержания связей между файлами и возможность селекции записей с применением этих связей.

Описание структуры базы данных на *языке описания данных (ЯОД)* определяется как *схема БД*. *Схема базы данных* включает схемы составляющих ее файлов и схемы связей между ними.

Как уже отмечалось, к числу компонентов структуры файла относятся *атрибут* и *группа*. *Атрибут* является наименьшей именованной единицей данных, каждый экземпляр которой в записи файла представляется одним или несколькими значениями. *Группа* также рассматривается в виде атрибута, который называется *множественным.*

Для *атрибута* в схеме файла задаются его короткое *двухсимвольное имя* (определяемое на ЯОД как код имени атрибута), *тип* и *максимальная длина значения*; для *множественного атрибута* дополнительно может быть определено максимальное количество экземпляров его значения в записи.

Тип значения атрибута определяет стандартный (принятый по умолчанию в командах *языка манипулирования данными* - *ЯМД*) вид представления значения этого атрибута в прикладной программе. К числу допустимых типов значений относится *символьный*, *упакованный* и *распакованный десятичный*, *битовый* и *двоичный*.

Атрибут в схеме файла может быть описан как *уникальный*, при этом его значения однозначно определяют экземпляры записей в файле БД и могут быть использованы в качестве ключей записей.

Группа представляет собой именованную совокупность атрибутов и, возможно, других групп. *Простая группа* состоит только из атрибутов, *составная группа* может содержать как атрибуты, так и простые и составные группы.

Периодической называется группа, которая может иметь в записи несколько экземпляров. Периодические группы в структуре файла не должны быть вложенными, т. е. не должны входить в состав других групп. Простые и составные группы могут быть иерархически подчинены одна другой и могут входить в состав повторяющихся групп, образуя древовидную структуру.

При описании группы на ЯОД задаются ее *имя* и *двухсимвольный код имени*, для повторяющихся групп может быть задано *максимальное количество экземпляров* этой группы в пределах экземпляра записи.

Использование группы упрощает доступ к входящим в ее состав атрибутам и группам. В команде выборки достаточно указать код имени группы, чтобы получить значения всех входящих в нее атрибутов. В случае периодической группы необходимо также указать номер (индекс) требуемого экземпляра.

Периодическая группа является средством реализации связи типа 1:M в пределах записи. Пример использования периодической группы в составе структуры файла СОТРУДНИКИ представлен на рис. 4. и рис. 5.

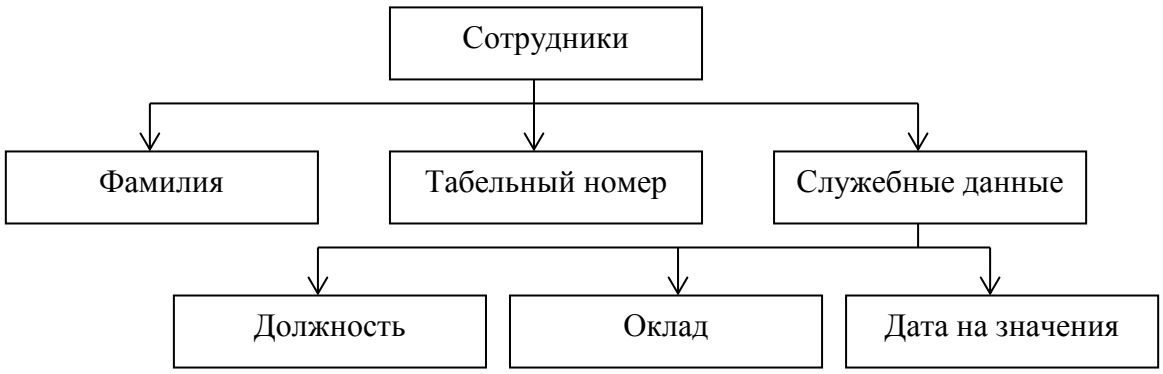


Рис. 4. Структура файла СОТРУДНИКИ с повторяющейся группой СЛУЖЕБНЫЕ ДАННЫЕ

		Инженер	140	17.08.2005
Николаев	983	Лаборант	90	20.03.2003

		Ст. инженер	160	23.02.2001
		Инженер	140	19.06.2000
Петров	2013	Техник	80	06.04.1997

Рис. 5. Записи файла СОТРУДНИКИ с повторяющейся группой СЛУЖЕБНЫЕ ДАННЫЕ

Как отмечалось ранее, ассоциации записей файла в базе данных ADABAS организуются в виде *инвертированных списков* по значениям поисковых атрибутов и в виде списков связи - по значениям атрибутов связи.

Признак *поисковый* для соответствующего атрибута указывается при его описании и структуре файла, а *коды имен атрибутов связи* - при определении связи двух файлов на ЯОД. Предусматривается также возможность организации ассоциаций по значениям *производных атрибутов*.

Значение производного атрибута может быть определено в схеме файла как часть значения обычного (непроизводного) атрибута (усеченный производный атрибут) или как конкатенация значений и/или частей значений обычных атрибутов (составной атрибут). Значения производных атрибутов используются только для организации ассоциаций и в записях файлов БД не хранятся.

Производные атрибуты по своей сущности являются поисковыми и могут использоваться также в качестве атрибутов связи.

Ассоциации, организованные по значениям производных атрибутов, расширяют возможности динамической классификации записей файла, позволяя моделировать классификацию объектов предметной области не только по их свойствам, но и по комбинации свойств.

Ассоциации записей файлов БД, реализованные посредством инвертированных списков и списков связи, автоматически создаются и корректируются при вводе, корректировке и удалении записей файлов, содержащих значения поисковых атрибутов, атрибутов связи и значения атрибутов, входящих в состав значений производных атрибутов.

Пример организации и ведения инвертированных списков иллюстрируется рис. 6, рис. 7 и рис. 8. Каждая запись представлена с ее внутрисистемным номером**,** который не входит в состав структуры записи. Поисковыми в схеме файла КАДРЫ являются атрибуты ГОД-РОЖДЕНИЯ и ПОЛ с кодами имен соответственно GR и PL. Файл КАДРЫ первоначально содержит восемь записей. По значениям поискового атрибута ГОД-РОЖДЕНИЯ для представленных записей организовано три инвертированных списка, по значениям поискового атрибута ПОЛ— два.

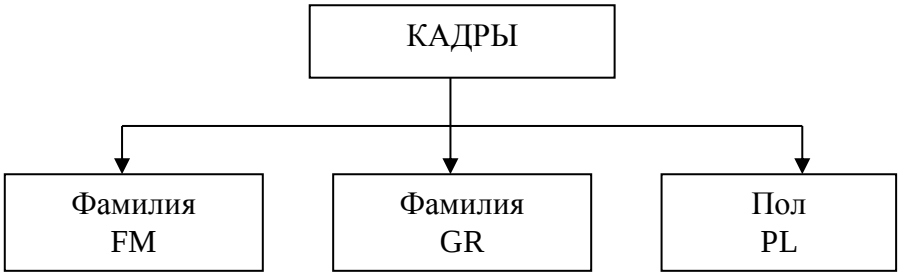


Рис. 6. Схема файла КАДРЫ

Индексные таблицы, обеспечивающие быстрый доступ к инвертированным спискам по коду имени и значению поискового атрибута, изображены на рис. 6, рис. 7 и рис. 8 упрощенно. Модификация файла, которая заключалась в удалении записи 40, добавлении записей 46, 47 и 48 и корректировке записи 42 (замена ошибочного значения 1948 на значение 1944), привела к созданию нового инвертированного списка с соответствующей модификацией индексной таблицы и корректировке всех инвертированных списков, кроме одного.

Рассмотрим далее организацию связей между записями файлов БД. Между любыми двумя файлами БД возможна организация одного типа связи, причем каждый файл может быть связан с несколькими другими файлами.

При описании связи на ЯОД указываются номера связываемых файлов и имена атрибутов связи (по одному в каждом файле). Запись любого из этих файлов объявляется *связанной* с одной или несколькими записями второго файла, если атрибут связи в каждой из этих записей имеет значение, совпадающее со значением атрибута связи в записи первого файла.

Для записи одного файла, связанной с множеством записей другого файла, в БД строится список связи, включающий совокупность ВСН этих записей. Доступ к списку связи осуществляется по ВСН записи первого файла, с которой связаны записи, представленные в списке связи. Первичное создание и удаление списков связи производится в автономном режиме с помощью утилит.

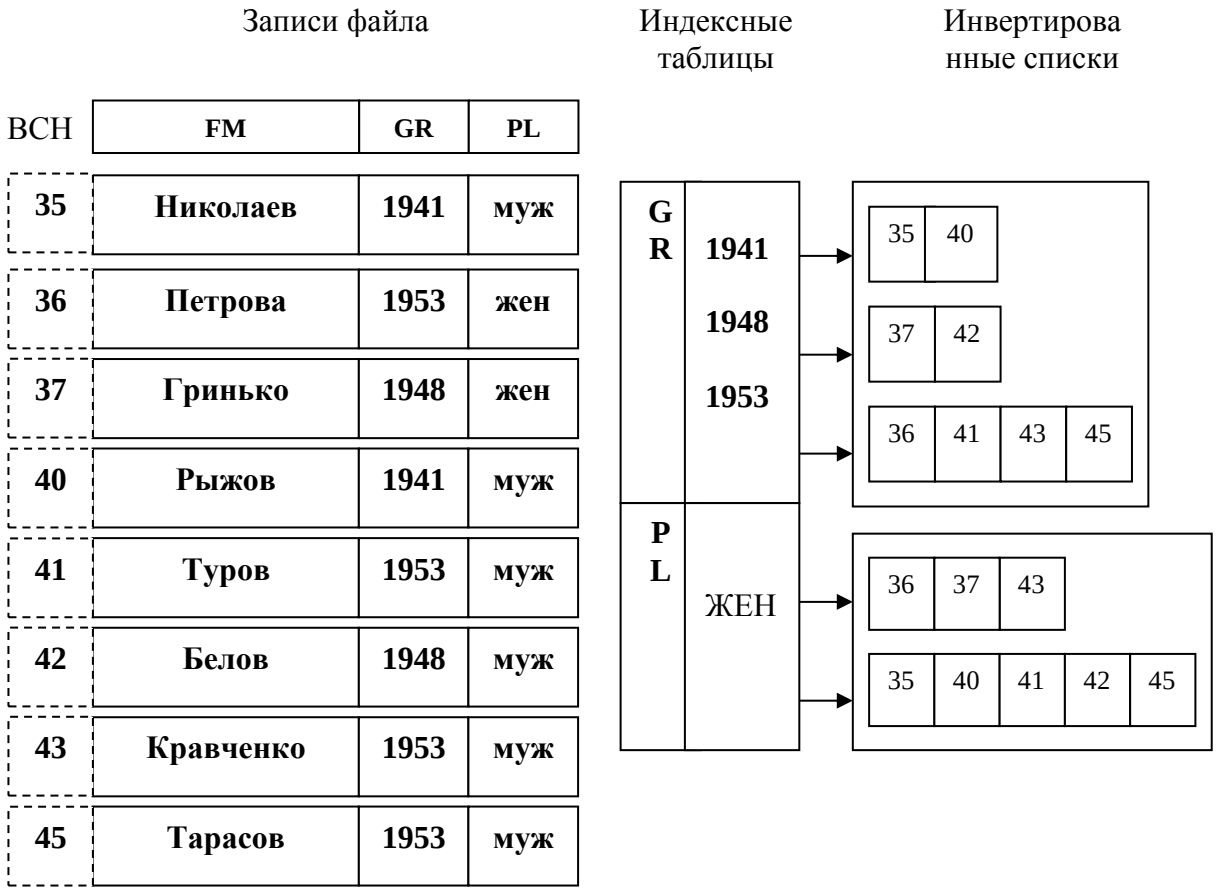


Рис. 7. Записи файла и инвертированные списки до модификации файла

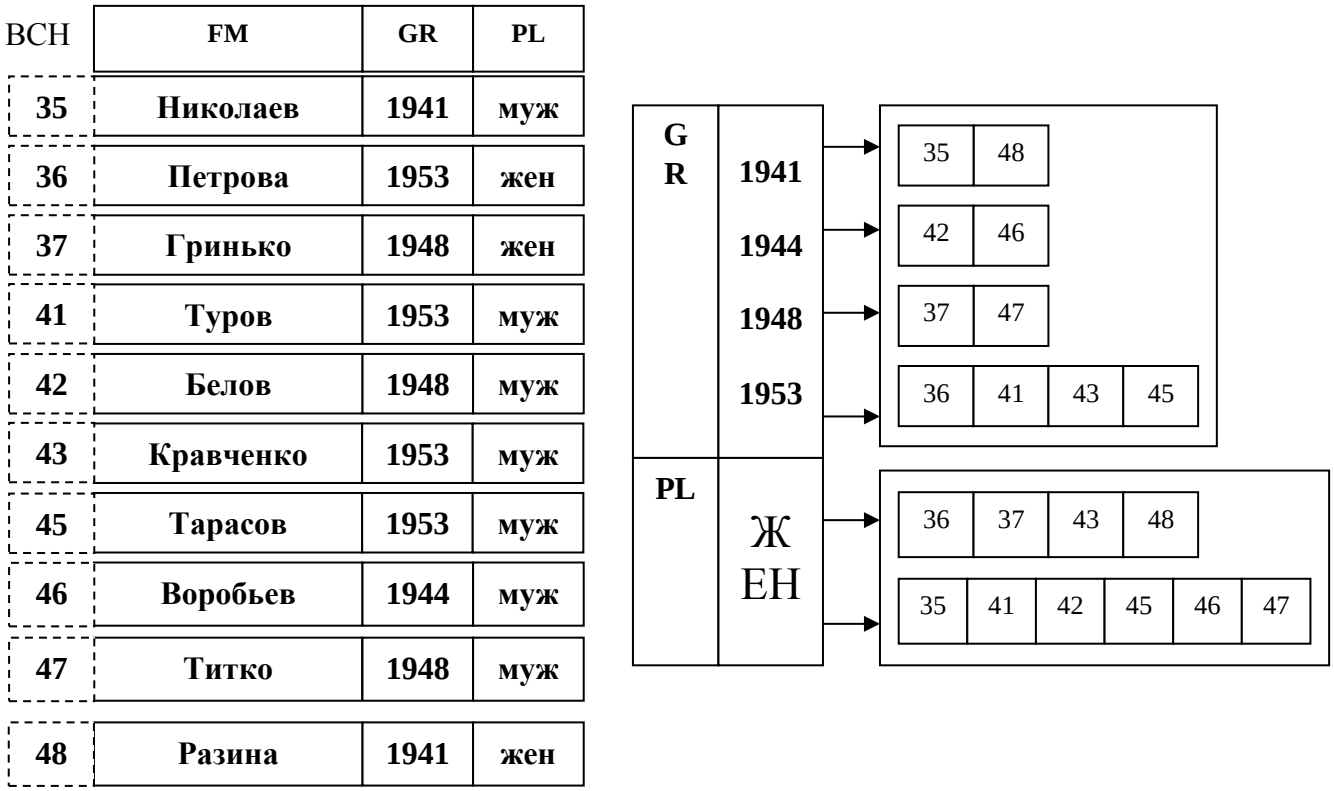


Рис. 8. Записи файла и инвертированные списки после модификации файла

Существенно, что атрибуты связи в каждом из связываемых файлов должны быть *поисковыми* (объявленными дескрипторами), поскольку формирование списков связи осуществляется с помощью инвертированных списков, которые до связывания должны быть организованы по каждому из этих атрибутов. Для организации *связи типа один-к-одному* в качестве атрибутов связи в каждом из файлов должны быть объявлены уникальные атрибуты. *Связь типа один-ко-многим* и *типа многие-к-одному* будет организована, если только один из атрибутов связи будет уникальным, а второй - простым атрибутом.

Связь типа многие-ко-многим может быть организована при объявлении атрибутом связи в одном файле уникального атрибута, а во втором — множественного атрибута. На атрибуты связи накладывается ряд ограничений, в число которых входят следующие:

– тип и длина значения атрибутов связи s обоих файлах не должны отличаться;

– множественным может быть атрибут связи только в одном из файлов;

– атрибут связи не должен входить в повторяющуюся группу;

– производный атрибут, используемый в качестве атрибута связи, не может быть сформирован из атрибутов, входящих в повторяющиеся группы.

Пример организации списков связи для связи типа 1:M представлен на рис. 9, рис. 10 и рис. 11. Файл СТУДЕНТ содержит атрибуты ФАМИЛИЯ (код имени FM) и ГРУППА (код имени GR), а файл КАРТОЧКА — атрибуты КНИГА (код имени KN) и ЧИТАТЕЛЬ (код имени CH). Связывание производится по значениям уникального атрибута ФАМИЛИЯ в первом файле и атрибута ЧИТАТЕЛЬ — во втором. Списки связи формируются путем последовательного просмотра инвертированных списков обоих файлов. При этом для каждого ВСН очередного инвертированного списка файла по значению поискового атрибута этого списка (он же является атрибутом связи) происходит обращение к соответствующему инвертированному списку второго файла, который заносится в БД как список связи с ключом, в качестве которого используется исходный ВСН записи первого файла. В результате последовательного выполнения указанных операций формируются списки связи для каждого из связываемых файлов, при этом «прямые» списки для первого файла будут «обратными» для второго.

Списки связи хранятся так же, как инвертированные списки, при этом для ускорения доступа к спискам связи создается индекс, отсортированный по значениям ВСН, используемым в качестве ключей списков связи.

Связи между записями файлов, соответствующие списку связи с ключом 4, на рис. 9 помечены пунктиром.

Файл СТУДЕНТ

BCH	**FM**	**GR**
1.	ПЕТРОВА	241
2.	ЗОТОВА	332
3.	ФРОЛОВА	211
4.	ГРАДОВ	343
5.	КВИТКО	423

Файл КАРТОЧКА

BCH	**KN**	**CH**
17.	РАЗГРОМ	ГРАДОВ
19.	ГОЙЯ	КВИТКО
20.	РЕВИЗОР	ГРАДОВ
21.	ОБРЫВ	ЗОТОВА
25.	ОВОД	ПЕТРОВА
26.	БЕРЕГ	ГРАДОВ
27.	СПАРТАК	ПЕТРОВА

Рис. 9. Записи файлов

FM	ГРАДОВ	→	4
	ЗОТОВА	→	2
	КВИТКО	→	5
	ПЕТРОВА	→	1
	ФРОЛОВА	→	3

CH	ГРАДОВ	→	17	20	26
	ЗОТОВА	→	21		
	КВИТКО	→	19		
	ПЕТРОВА	→	25	27	

Рис. 10. Инвертированные списки с индексными таблицами

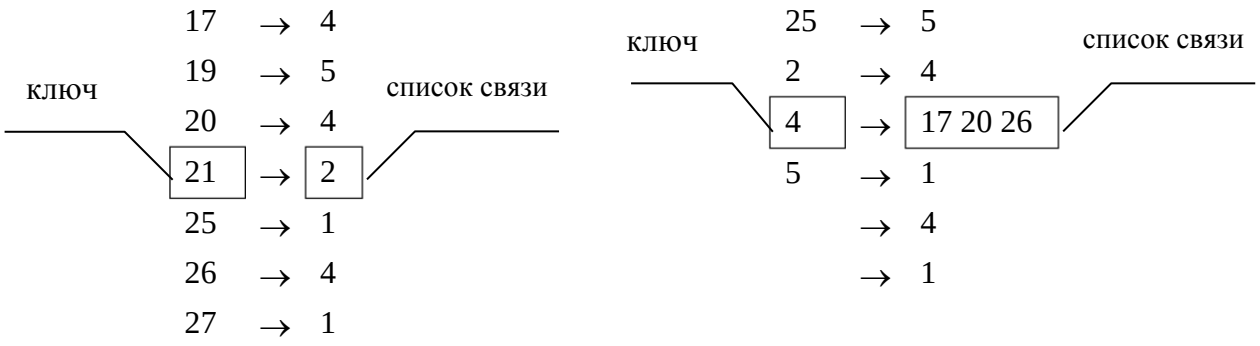


Рис. 11. Списки связи с ключами

Более сложный пример связывания файлов представлен на рис. 12., рис. 13 и рис. 14. Схема файла СТУДЕНТ включает атрибуты НОМЕР-ЗАЧЕТНОЙ-КНИЖКИ, ФАМИЛИЯ и ГРУППА с кодами имен соответственно NZ, FM и GR, а схема файла СЕМИНАР—атрибуты РУКОВОДИТЕЛЬ, КАФЕДРА и УЧАСТНИК с кодами имен RK, KF и UK. Атрибутами связи файлов являются атрибуты НОМЕР-ЗАЧЕТНОЙ-КНИЖКИ и УЧАСТНИК.

В данном случае реализуется *связь типа многие-ко-многим*, и алгоритм формирования «прямых» списков второго файла несколько сложнее, так как каждый список связи формируется в результате объединения отдельных одноэлементных «частных» списков.

При организации связи между двумя файлами БД записи считаются равноправными, и связь может использоваться в обоих направлениях.

Файл1 СТУДЕНТ

BCH	**NZ**	**FM**	**GR**
18	171	ПЕТРОВ	241
19	302	ТИТОВ	242
20	281	ИЛЬИНА	241
21	138	СТОЛЯР	243
22	544	ФРОЛОВ	242
25	217	ТЮРИНА	241

Файл2 СЕМИНАР

BCH	**RK**	**KF**	**UK**
1	УРЛАНИС	РПС	171 302 138 544
2	ГЛЕБОВА	ИВС	302 217
3	РОДИН	РПУ	
4	ПЕТРОВ	ИВС	138 544 217

Рис. 12. Записи файлов

NZ	138	→	21	**UK**	138	→	1	4
	171	→	18		171	→	1	
	217	→	25		217	→	2	3 4
	281	→	20		281	→	3	
	302	→	19		302	→	1	2
	544	→	22		544	→	1	4

Рис. 13. Инвертированные списки с индексными таблицами

ключи списков саязи

$$\begin{cases} 1 \to 18\ \ 19\ \ 21\ \ 22 \\ 2 \to 19\ \ 25 \\ 3 \to 20\ \ 25 \\ 1 \to \underbrace{21\ \ 22\ \ 25}_{\text{список связи}} \end{cases}$$

$21 \to 1\ \ 4$
$18 \to 1$
$25 \to 2\ \ 3\ \ 4$
$20 \to 3$
$19 \to 1\ \ 2$
$22 \to 1\ \ 4$

Рис. 14. Списки связи с ключами

По способу реализации связь между файлами ADABAS может быть отнесена к типу связей, не несущих информации, поскольку все, что семантически связывает записи двух файлов между собой, содержится в значениях соответствующих атрибутов записей независимо от того, созданы или нет списки связи.

Исходя из вышеизложенных принципов организации связей между записями файлов БД системы ADABAS, работа с логически связанными записями различных файлов возможна только с помощью инвертированных списков, без организации списков связи. Однако такой способ связывания файлов в процессе обращения к БД требует дополнительных временных затрат, поэтому использование списков связи при работе с сетевыми и иерархическими структурами повышает производительность системы.

Независимая от записей файлов организация ассоциаций в виде инвертированных списков и списков связи в ADABAS обеспечивает возможность создания новых ассоциаций и уничтожения старых без перезагрузки файлов БД.

Операции базовой модели данных

Объектами операций модели данных концептуального уровня ADABAS являются БД и составляющие ее компоненты, рассмотренные в предыдущем разделе.

Операция создания БД заключается в формировании внутримашинного представления схемы, т. е. загрузочной схемы БД. Остальные операции, выполняемые над схемой БД и значениями составляющих БД компонентов, относятся к *операциям модификации* и *доступа к данным*.

Операции модификации схемы БД предусматривают:

– добавление схемы нового файла;

– удаление существующей схемы файла;

– включение в структуру существующего файла новых атрибутов, групп и производных атрибутов;

– установление новых связей между файлами;

– удаление существующих связей между файлами;

– определение непоискового атрибута поисковым (объявление поля дескриптором);

– отмену признака поисковости у атрибута (отмена объявления поля дескриптором).

Операции модификации данных, хранящихся в БД, обеспечивают ввод, корректировку и удаление отдельных записей файлов БД. Для выполнения указанных операций в ЯМД концептуального уровня ADABAS предусмотрены соответствующие команды. Массовый ввод и обновление записей файлов, а также удаление файлов БД осуществляются утилитами автономного ведения данных. Операции модификации данных включают:

– ввод записей в БД;

– корректировку записей файла;

– удаление записей файла;

– доступ к данным:

  – чтение записей файла;

  – поиск записей файла;

– селекция по связям записей файла;

– рандомизированный доступ к записям.

Ввод записей в БД. Запись, вводимая в БД программой пользователя, должна содержать не менее одного атрибута. Каждый атрибут в составе записи представляется при вводе *кодом имени* и *значением*, *длиной* и *типом значения* атрибута.

Вводимой записи автоматически присваивается ВСН, который после окончания операции ввода передается прикладной программе. Предусмотрена также возможность назначения ВСН программой пользователя по известному ему алгоритму, однако назначаемый подобным образом ВСН не должен превышать максимально допустимого числа записей в файле, которое задается параметрически при создании файла.

Если в вводимой записи имеются значения атрибутов связи и поисковых атрибутов, а также значения атрибутов, входящих в состав производных атрибутов, то модификация списков связи и инвертированных списков, т. е. создание новых и корректировка существующих списков, выполняется автоматически.

Все атрибуты, которые не вводятся в составе новой записи и в схеме файла предшествуют последнему вводимому атрибуту, после загрузки получают нулевое значение для числового и значение «пробел» — для символьного типа значения. Это означает, что в качестве признаков «неопределенных» значений в записях файлов БД системы ADABAS используются нуль или пробел в зависимости от типа значения атрибута.

Если атрибут с «неопределенным» значением является *поисковым* (или *атрибутом связи*), то при описании его на ЯОД без признака С ПОДАВЛЕНИЕМ, его значение (нуль или пробел) используется для модификации инвертированных списков и списков связи. В противном случае, т. е. если поисковый атрибут, значения которого не вводятся, определен С ПОДАВЛЕНИЕМ, модификация инвертированных списков и списков связи при выполнении операций ввода не производится. Это относится также к невводимым поисковым атрибутам, которые в схеме файла находятся за атрибутом, значение которого в составе вводимой записи является последним.

Если в составе записи находится значение уникального атрибута, то при выполнении операции ввода производится проверка на отсутствие этого значения в ранее введенных записях файла.

Корректировка записей файла. Единичная запись как объект корректировки идентифицируется ее внутрисистемным номером. ВСН записи получается в прикладной программе с помощью одной из операций селекции данных. Логика операции корректировки записей файла основана на сопоставлении значений атрибутов, заданных программой пользователя, и значений этих атрибутов в корректируемой записи файла БД. Корректировка записей файла заключается в обновлении значений атрибутов этих записей. При этом обеспечиваются вставка, замена и исключение значений атрибутов ранее введенных записей. Если значение обновляемого атрибута задано в прикладной программе, а в корректируемой записи значение этого атрибута отсутствует, производится вставка нового значения атрибута в состав существующей записи файла.

При задании значения атрибута в прикладной программе и отличном от него значении этого атрибута в корректируемой записи существующее значение атрибута записи файла заменяется новым значением. Значения атрибута записи удаляются путем замены существующего значения атрибута на «неопределенное» (нулевое — для числовых или пробелы — для символьных атрибутов), которые задаются для удаляемого атрибута прикладной программой. Если в число обновляемых атрибутов при корректировке записей файла входят поисковые атрибуты, производятся необходимые изменения в инвертированных списках и списках связи. Модификация списков связи и инвертированных списков, построенных по значениям производных атрибутов, осуществляется при обновлении значений атрибутов, из которых сформированы значения производных атрибутов.

Удаление записей файла. Идентификация удаляемых из файла записей осуществляется по ВСН этих записей, причем одна операция рассчитана на удаление одной записи.

Занятая удаляемой записью физическая память файла объявляется свободной для дальнейшего использования, а ВСН удаленной записи отмечается как незанятый и в дальнейшем может быть назначен вновь вводимым записям.

Удаление записи сопровождается модификацией соответствующих списков связи и инвертированных списков.

Доступ к данным. Операции доступа к данным представлены в модели данных концептуального уровня ADABAS операциями *чтения* и *поиска записей* файлов БД.

Операция чтения обеспечивает селекцию записи по ее позиции в файле или его части и выборку требуемых значений атрибутов, т. е. операция чтения эквивалентна последовательности операций селекции и выборки данных.

Операция поиска обеспечивает селекцию некоторой совокупности записей файла по значениям атрибутов этих записей или с учетом связей между файлами и, при необходимости,— выборку значений атрибутов из первой отобранной записи. Частным случаем селекции по значениям атрибутов в ADABAS является рандомизированный доступ к записям файлов.

Чтение записей. В зависимости от вида упорядоченности множества записей операция чтения обеспечивает доступ к записям файла:

– по списку ВСН записей;

– в порядке возрастания ВСН записей файла;

– в логической последовательности по значениям заданного поискового атрибута;

– в физической последовательности расположения записей в БД.

При *доступе к записям файла по списку ВСН* в качестве упорядоченного множества выступает часть записей файла БД, представленная списком ВСН, причем возможно обращение к первой заданной и к следующей записи этого множества. Сам список должен быть отсортирован по возрастанию значений ВСН, он может быть получен с использованием селекции по значениям атрибутов или селекции по связям файлов. Доступ к записи файла по ВСН осуществляется через системную таблицу, называемую преобразователем адреса. Преобразователь адреса организуется для каждого файла БД и отображает каждый ВСН в относительный номер блока набора данных, в котором размещается запись файла с этим ВСН.

При *доступе к записям файла в порядке возрастания их ВСН* возможно обращение к записи с минимальным ВСН, к записи с ВСН, ближайшим «большим» указанного, и обращение к записи со следующим (точнее, ближайшим — большим) ВСН независимо от физического размещения

записей. Следует отметить, что наряду со значениями требуемых атрибутов очередной записи файла пользовательской программе становится доступным ВСН этой записи.

При *доступе к записям в логической последовательности значений поискового атрибута* в качестве упорядоченного множества, в котором производится селекция записей, выступает вся совокупность ассоциаций, построенных по значениям заданного поискового атрибута файла. Порядок в данном множестве определяется сортировкой полей индексных таблиц по значениям поисковых атрибутов и сортировкой каждого из инвертированных списков по значениям ВСН. В данном случае обеспечивается возможность обращения к *первой записи* (в смысле порядка, указанного выше), к *конкретной записи* и к *следующей записи множества*.

Для обращения к *первой записи* достаточно указать в условии селекции код имени поискового атрибута. *Первой* в данном случае считается первая запись первой ассоциации из всех организованных в данном файле ассоциаций по значениям атрибута, указанного в условии селекции. Следует отметить, что в условии селекции в данном случае, как и при обращении к конкретной записи, допускается использовать атрибуты, не входящие в периодическую группу, а также производные атрибуты, не содержащие атрибутов из повторяющихся групп.

Обращение к *конкретной записи* выполняется по коду имени и значению поискового атрибута. В случае отсутствия заданного значения в записях файла выдается запись с ближайшим *большим* (в смысле сортировки значений поискового атрибута) *значением атрибута*. Имеется дополнительная возможность указания конкретной записи путем задания ВСН, но при этом доступ обеспечивается к записи с ближайшим *большим* заданного ВСН.

Возможность обращения к *следующей записи* логически упорядоченного по значениям поискового атрибута множества записей файла позволяет просмотреть все ассоциации данного атрибута с начала или с указанной записи. При этом для каждой очередной прочитанной из БД записи выдается ее ВСН. Следует иметь в виду, что если в условии селекции используется множественный атрибут, то в процессе просмотра одна и та же запись может встретиться несколько раз.

Пример доступа к записям файла, логически упорядоченного по значениям поискового атрибута, представлен на рис. 15. Селекция записей осуществляется с использованием инвертированных списков и позволяет просмотреть все записи ассоциаций записей, организованных по поисковому атрибуту ОК, начиная с заданной записи. Доступ к записям файла в порядке физического размещения записей в БД основан на последовательном считывании блоков физической памяти файла в порядке их расположения в наборе данных. При этом возможно чтение первой записи первого физического блока файла, чтение записи в указанной части памяти файла и чтение записи, физически следующей за указанной.

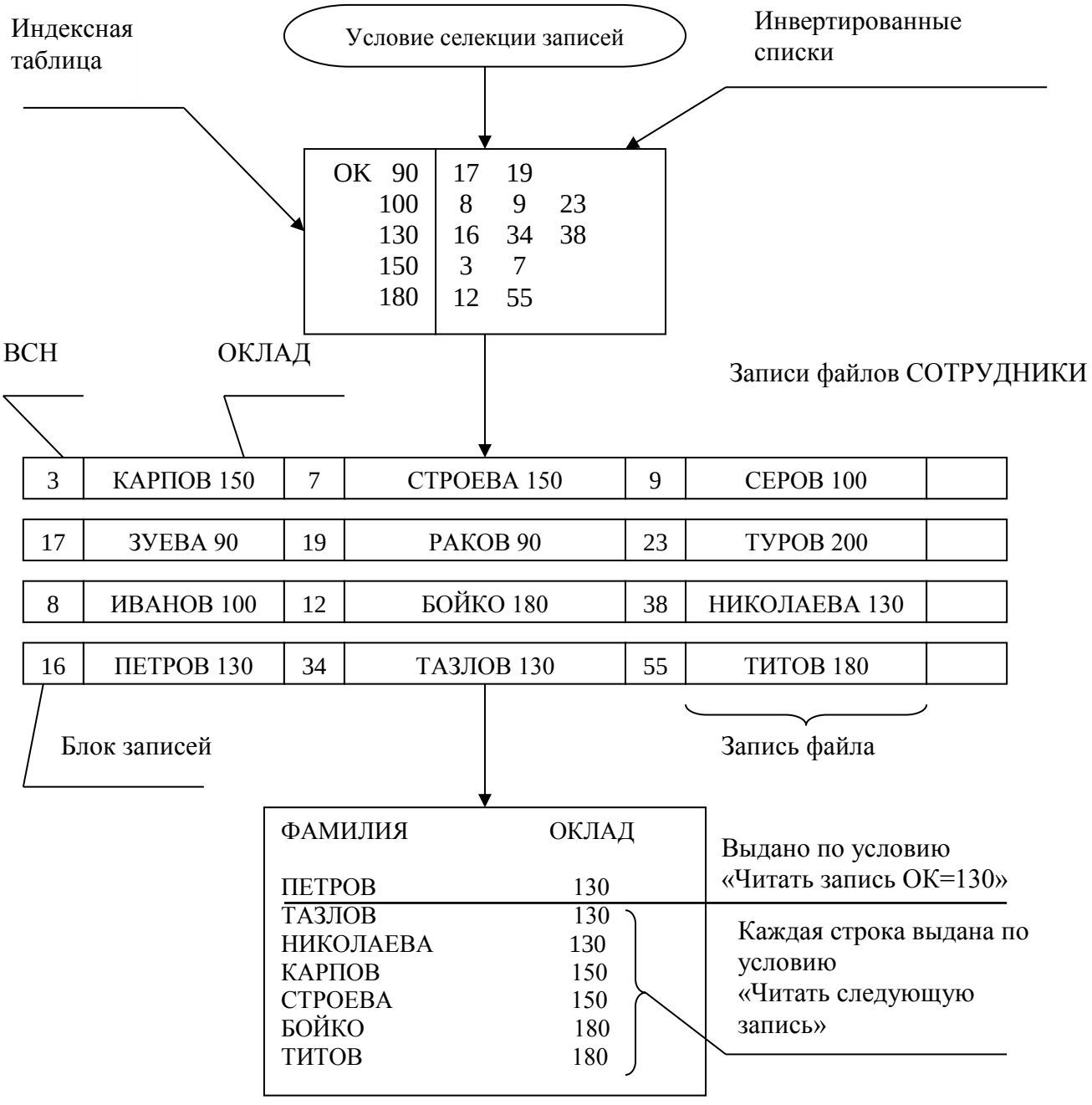


Рис. 15. Доступ к записям файла в логической последовательности по значениям поискового атрибута ОКЛАД (код имени ОК)

Поиск записей файла. Поиск записей в ADABAS, основанный на использовании ассоциаций, представленных в виде инвертированных списков и списков связи, определяется как *ассоциативный* поиск.

Ассоциативный поиск обеспечивает возможность селекции записей файла БД по значениям атрибутов, а также селекцию записей файла с учетом их связей с записями другого файла.

Условие селекции определяется в ADABAS как условие поиска, которое включает одно или несколько простых условий поиска. Простые условия поиска в пределах условия поиска соединяются логическими операциями И.

Простое условие задает для поискового атрибута (в том числе производного) одно или несколько значений с использованием логических операций И, ИЛИ, ОТ ДО, КРОМЕ. Логическая операция ОТ ДО задает диапазон значений поискового атрибута, исключение значения из которого определяется логической операцией КРОМЕ.

Селекция по значениям атрибутов в ADABAS не требует обращения к записям файлов, поскольку каждое простое условие определяет ключи соответствующих ассоциаций записей файла, созданных в процессе загрузки файла. При этом селекция по значениям атрибутов сводится к формированию списка ВСН записей по каждому из простых условий и пересечению этих списков в соответствии с логикой, заданной в условии поиска. Итогом указанных операций является представленное результирующим списком ВСН множество записей файла, удовлетворяющих условию поиска.

Формирование списка ВСН по простому условию заключается в чтении ряда инвертированных списков по коду имени и значениям поискового атрибута, заданным перечислением конкретных значений или границами диапазона значений, и в дальнейшей обработке этих списков путем выполнения необходимых логических операций пересечения, объединения и пересечения с отрицанием.

На рис. 16 представлена схема ассоциативного поиска с использованием инвертированных списков. Условие поиска состоит из двух простых условий, одно из них задано диапазоном значений атрибутов, а другое — перечнем значений. Результирующий список, полученный как пересечение двух спис-

ков, сформированных по простым условиям, содержит четыре ВСН записей, удовлетворяющих условию поиска.

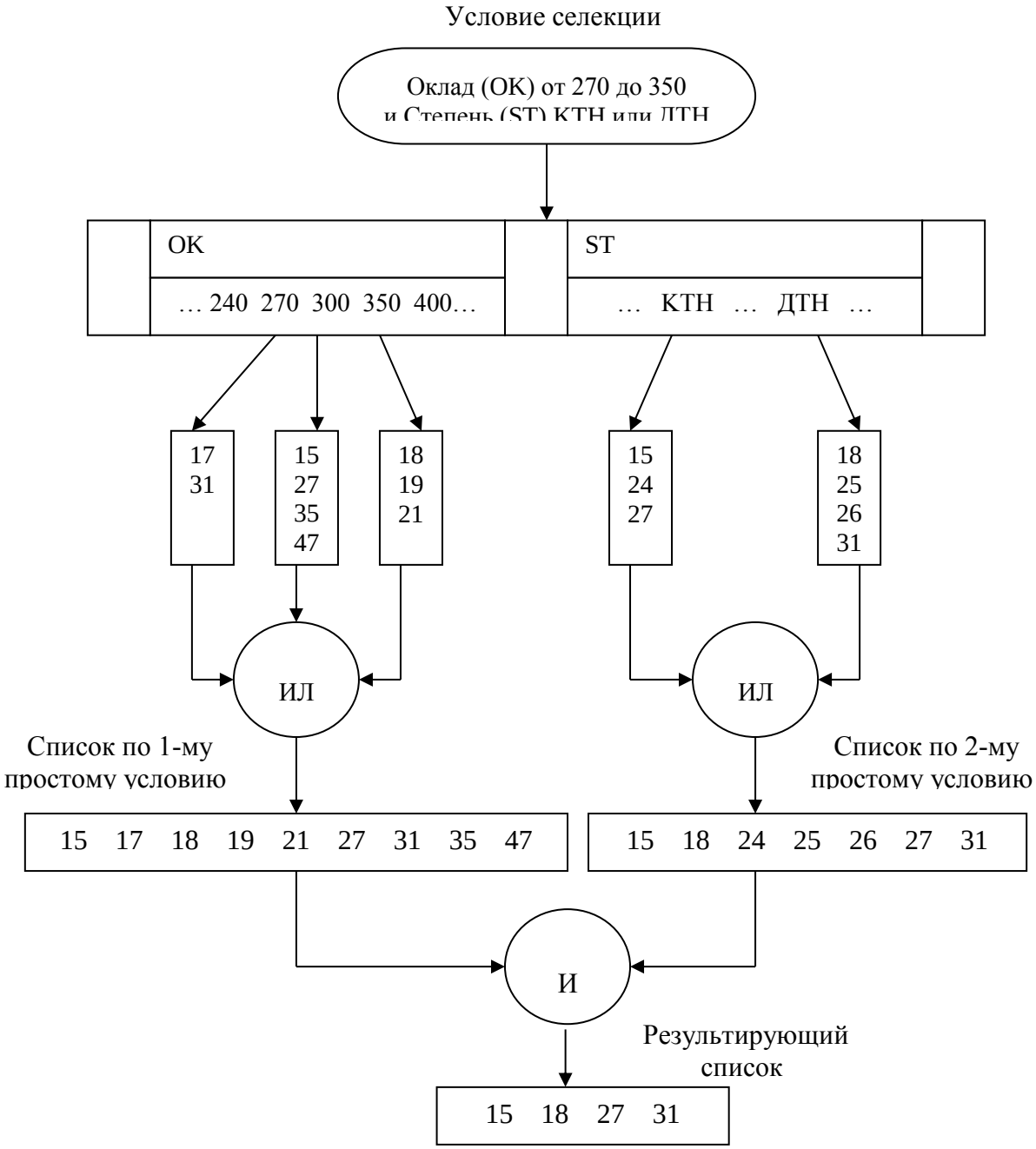


Рис. 16. Ассоциативный поиск записей файла

Если полагать, что чтение каждого инвертированного списка с учетом многоуровневости индексных таблиц требует в среднем четырех операций ввода-вывода, то практически независимо от количества записей файла БД ассоциативный поиск в данном случае потребует 20 операций ввода-вывода. Это дает ощутимый выигрыш во времени по сравнению с методами использования цепочек и прямого просмотра файла БД.

При использовании инвертированных списков некоторая зависимость времени поиска от количества записей файла проявляется в больших БД, содержащих 10^5—10^6 записей, поскольку возникает необходимость обработки больших списков ВСН по частям.

Селекция по связям записей файла. Как уже отмечалось, связь между записями двух файлов устанавливается по равенству значений атрибутов связи в этих записях и фиксируется путем построения *прямых* и *обратных* списков связи, при этом понятия *исходный* и *связанный* файлы являются относительными.

В модели данных концептуального уровня ADABAS реализуются две разновидности селекции по связям между записями файлов:

– по наличию связей записей файла с записью другого файла;

– по значениям атрибутов в записях связанных файлов.

Первая разновидность селекции, называемая *поиском связанных записей,* использует списки связи для отображения записи одного файла в одну или более записей второго файла. В качестве *ключа списка* при этом используется ВСН отображаемой записи. Данная разновидность селекции является спецификационной операцией и может быть использована, например, в иерархической структуре для перехода от *исходного узла* к *порожденному* без обращения к экземплярам порожденного узла.

Вторая разновидность селекции, называемая *поиском в связанных файлах,* основана на использовании предыдущей разновидности селекции и селекции по значениям атрибутов. В качестве условия селекции при этом выступает *составное условие поиска*, включающее до пяти условий поиска, одно из которых относится к исходному файлу, а остальные - к связанным файлам.

Результатом поиска в связанных файлах является список ВСН записей исходного файла, которые удовлетворяют заданному условию поиска и связаны с записями нескольких, но не более четырех файлов с заданными свойствами.

Селекция записей файла *по значениям атрибутов* связанного файла выполняется поэтапно:

- селекция по значениям атрибутов в исходном файле;
- селекция по значениям атрибутов в связанном файле;

– отображение результата поиска в связанном файле па исходный файл;

– получение результирующего списка ВСН путем пересечения списков ВСН исходного файла, полученных в результате операций селекции и отображения.

На рис. 17. представлена схема ассоциативного поиска в связанных файлах. В данном примере используются файлы СТУДЕНТ (файл 1) и СЕМИНАР (файл 2), организация связи между которыми была описана ранее (рис. 12, рис. 13, и рис. 14).

Составное условие поиска в примере на рис. 17 соответствует требованию найти в файле 1 все записи о студентах учебных групп 241 и 242, участвующих в семинарах преподавателей кафедры информационно-вычислительных систем (ИВС). Файл 1 объявлен как исходный, поэтому результат поиска в виде списка из трех ВСН записей относится к файлу СТУДЕНТ.

Возможности ассоциативного поиска в БД системы ADABAS могут быть расширены благодаря использованию операций над ассоциациями записей файлов. Данные операции обеспечивают возможность логической обработки и сортировки списков ВСН записей файла, которые могут быть получены в результате ассоциативного поиска или сформированы программой пользователя.

Операции логической обработки используют в качестве операндов два списка ВСН записей, принадлежащих одному файлу. Допускается три вида логических операций - *пересечение списков*, их *объединение* и *пересечение с отрицанием*.

Сортировка списка ВСН может быть выполнена в порядке возрастания значений ВСН в списке, а также в порядке возрастания или убывания значений от одного до трех поисковых атрибутов записей файла, которые представлены своими ВСН в сортируемом списке.

Рандомизированный доступ к записям. Суть *метода рандомизации* заключается в алгоритмическом преобразовании ключа записи в некоторое число, которое является физическим адресом этой записи. В качестве ключа в ADABAS употребляется значение одного из атрибутов записи, которое должно быть уникальным в пределах файла, а физическим адресом записи

является относительный номер блока, в котором находится эта запись. Ключом записи может быть также ее ВСН.

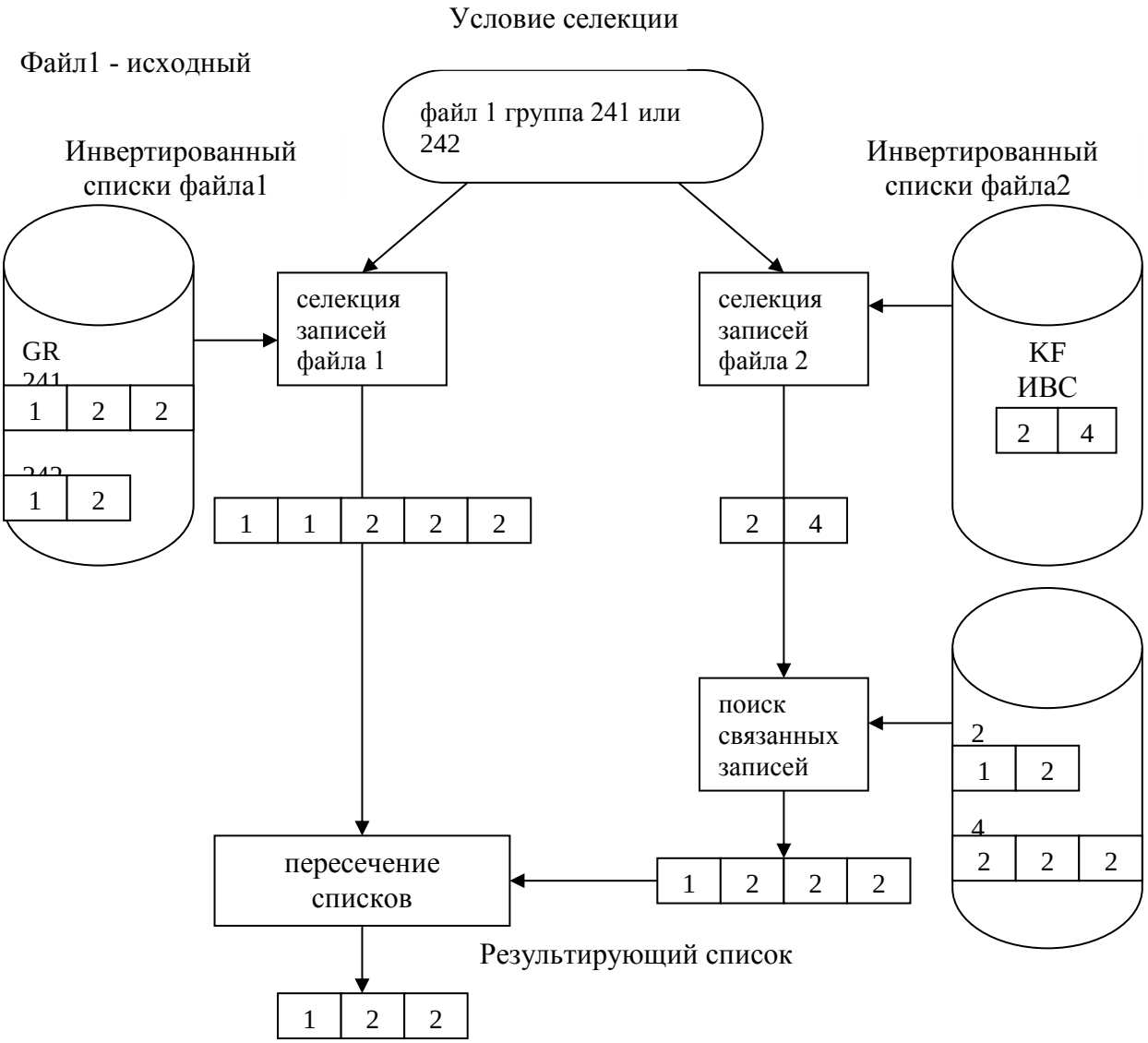


Рис. 17. Ассоциативный поиск записей в связанных файлах

Алгоритм преобразования ключа в адрес включает *нормализацию шестнадцатеричного представления значения атрибута* путем его усечения до параметрически заданной длины и деление результата нормализации на простое число, ближайшее меньшее *числа блоков экстента* набора данных, выделенного для размещения записей данного файла. В качестве физического адреса записи при ее занесении в файл используется остаток от деления. При рандомизированном доступе физический адрес записи вычисляется алгоритмически по значению ключевого атрибута (или ВСН) точно так же, как это делается при вводе записи. Это означает, что независимо от количества записей в файле рандомизированный доступ к записи требует одной операции ввода-вывода.

При использовании метода рандомизации размещение вводимых записей в пределах файла носит случайный характер и различным ключам записей может соответствовать один и тот же вычисленный физический адрес. Это означает, что в одном блоке набора данных необходимо размещать несколько записей файла. При этом может возникнуть такая ситуация, когда блок, в котором должна быть размещена очередная запись, не имеет достаточно свободного пространства. В подобном случае очередная запись заносится в ближайший свободный блок набора данных и для нее строится обычный инвертированный список по значению уникального атрибута.

Наличие подобных записей в файле ухудшает эффективность рандомизированного доступа к файлу, однако при правильно выбранных параметрах переполнение блоков происходит редко и эффективность этого вида селекции ухудшается незначительно.

Ограничения целостности

Ограничения целостности модели данных концептуального уровня ADABAS могут быть разделены на *синтаксические* и *семантические*.

Статические синтаксические ограничения, определяющие возможности ADABAS по отображению структурных свойств объектов предметной области и отношений между ними в схеме БД приведены ниже:

Число файлов в БД	255
Число записей файла	$16{,}7*10^6$
Число компонентов схемы файла	500
Число уровней в схеме файла	7
Число экземпляров повторяющейся группы	99
Число экземпляров множественного атрибута	191
Число поисковых атрибутов файла	200
Число элементов составного атрибута	5
Длина значения атрибута:	
символьного, байт	253
десятичного целого, разрядов	27
упакованного, байт	14
битового, байт	126
двоичного целого, байт	4
символьного или битового поискового, байт	126

производного, байт 256

Число файлов, связанных с данным файлом 18

Динамическим синтаксическим ограничением целостности является запрет на использование в операции модификации более чем одной записи файла БД. Это ограничение минимизирует объем данных, передаваемых между прикладной программой и ADABAS в процессе ввода, корректировки или удаления записей файлов БД, что повышает динамичность работы ADABAS в режиме обслуживания многих пользователей.

Статические семантические ограничения целостности усиливают действие структурных ограничений, устанавливая дополнительные рамки на значения атрибутов в соответствии с семантикой моделируемых объектов предметной области и тем самым, ограничивая число возможных экземпляров объектов, представленных в БД. К данному виду ограничений можно отнести *задание* на ЯОД *числа экземпляров множественного атрибута и периодических групп*, а также *задание длин значений атрибутов в пределах допустимых максимальных значений*, определяемых структурными ограничениями. Возможность *определения уникальных атрибутов в схеме файла* также относится к числу статических семантических ограничений. Кроме того, статические семантические ограничения в ADABAS представлены возможностью *контроля по словарю значений символьных атрибутов* и *контроля по диапазону значений для числовых атрибутов*. Требование на указанные виды контроля определяется в словаре данных путем описания атрибутов, подлежащих контролю.

Динамические семантические ограничения, определяющие допустимые переходы БД из одного целостного состояния в другое в зависимости от свойств объектов предметной области, могут быть заданы путем использования специальных команд ЯМД. К таким командам относятся команды управления транзакциями, с помощью которых определяется допустимая последовательность операций модификации данных, переводящая БД в повое целостное состояние.

8.4. Структура СУБД ADABAS

Многоуровневая архитектура

Для современной концепции баз данных характерно многоуровневое представление данных, при котором каждый уровень играет определенную роль в процессе проектирования, эксплуатации и использования БД.

Многоуровневая архитектура СУБД обеспечивает независимость данных и способность информационных систем, построенных на основе СУБД, к эволюции. Наибольшее признание в настоящее время получила *трехуровневая архитектура СУБД*, включающая *внешний*, *концептуальный* и *внутренний уровни*.

Внешний уровень поддерживает частные представления о данных отдельных задач пользователей.

Концептуальный уровень является глобальным представлением о предметной области вне зависимости от того, как данные хранятся и используются приложениями, внутренний уровень связан с представлением данных в памяти ЭВМ.

Каждый уровень представления данных в архитектуре СУБД является, по существу, самостоятельным уровнем абстракции данных и определяется соответствующей моделью данных, поэтому современные СУБД с многоуровневой архитектурой характеризуются не одной, а несколькими моделями данных.

Трехуровневая архитектура СУБД обладает высокой степенью адаптации к происходящим изменениям в процессе эволюции информационных систем вследствие того, что обеспечивает возможность независимой модификации внешних и внутренних представлений о БД благодаря существованию стабильной концептуальной схемы и действию соответствующих отображений.

ADABAS является системой управления базами данных с трехуровневой архитектурой. Она предоставляет в распоряжение пользователей концептуальный и внутренний уровни, ориентированные на АБД, и внешний уровень, ориентированный на конечных пользователей, администратора прикладных задач и прикладных программистов. Кроме того, прикладные программисты могут обращаться к БД на *языке манипулирования данными концептуального уровня*. Однако необходимо отметить, что в системе

ADABAS отсутствует четкая граница между концептуальным и внутренним уровнями.

Описанные выше функциональные возможности системы ADABAS реализуются совокупностью программных компонентов, объединенных архитектурой, отраженной на рис. 18.

В состав ADABAS входят программные средства, обеспечивающие:

- управление видеотерминалами (МВТ);
- конструирование интерактивных технологий (КИТ);
- диалоговую обработку данных (ДИОД);
- диалоговую подготовку справок (ДПС);
- программирование интерфейсных модулей (СПРИНТ);
- генерацию отчетов (СГО);
- управление базой данных (УБД);
- мультипрограммный интерфейс (МПИ);
- автономное ведение данных (АВД);
- ведение данных.

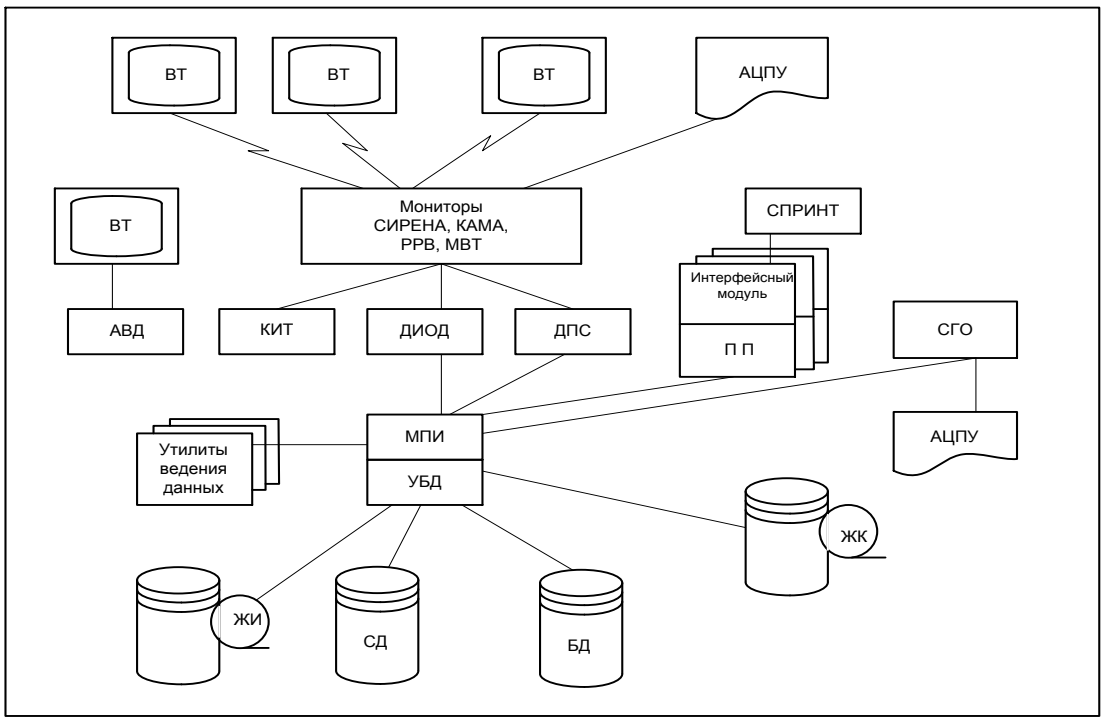


Рис. 18. Архитектура ADABAS

ВТ — видеотерминал;

АЦПУ — алфавитно-цифровое печатающее устройство;

РРВ — режим разделения времени;

ПП - прикладные программы;

ЖК — журнал команд;

ЖИ -журнал изменения;

СД — словарь данных;

БД—база данных;

МВТ - монитор видеотерминалов;

АВД — автономное ведение данных;

КИТ — конструирование интерактивных технологий;

ДИОД— диалоговая обработка данных;

ДПС — диалоговая подготовка справок;

МПИ — мультипрограммный интерфейс;

УБД — управление базой данных;

СГО — средства генерации отчетов;

СПРИНТ — средства программирования интерфейсных модулей.

Концептуальный и внутренний уровни

Концептуальный и *внутренний уровни* реализуются *программой УБД* и *утилитами ведения данных*.

Утилиты ведения данных включают *утилиты создания и ведения базы данных*, *утилиты ведения словаря данных* и *утилиту обеспечения сохранности данных*.

Утилиты ведения данных в интерактивном режиме их использования получают параметры от *средств АВД и КИТ*. Эти утилиты могут запускаться также в пакетном режиме работы.

Утилиты ведения словаря данных осуществляют накопление, обновление и выдачу сведений о данных, программах, пользователях и их связях с данными и программами.

Программа УБД и утилиты ведения данных обеспечивают механизм отображения модели данных концептуального уровня на модель данных, поддерживаемую на внутреннем уровне.

На концептуальном уровне ADABAS с помощью языка описания данных даст представление о БД как о совокупности взаимосвязанных файлов, состоящих из однотипных записей, включающих атрибуты (простые и повторяющиеся), которые моделируют реальные информационные объекты предметной области, их свойства и взаимосвязи.

Операции манипулирования БД обеспечивают возможность создания, модификации и доступа к записям файлов с учетом их взаимосвязей и объявленных в схеме БД ассоциаций.

На концептуальном уровне ADABAS допускается возможность обращения к БД из прикладных программ, написанных на языках ассемблера, Кобол, Фортран или ПЛ/1. Особенности операционной среды, под управлением которой выполняются прикладные программы, учитываются с помощью специализированных интерфейсов.

Модель данных, используемая в ADABAS на концептуальном уровне, определяется в системе как *базовая модель данных*.

На внутреннем уровне дается представление о БД с использованием таких понятий среды хранения, как *набор данных*, *экстент*, *блок*, *хранимая запись*, *индекс* и т. д.

Механизм отображения концептуального представления БД на ее внутреннее представление обеспечивает преобразование различных типов данных, определенных на них ограничений и операций базовой модели в понятия, ограничения и операции внутренней модели.

Внешний уровень

Внешний уровень в архитектуре ADABAS представлен рядом компонентов, в число которых входят программные средства ДИОД, СГО, ДПС, СПРИНТ и утилита форматированного ввода данных.

Каждый из перечисленных компонентов внешнего уровня ADABAS поддерживает некоторый уровень абстракции БД, ориентированный на соответствующую категорию пользователей системы, и может рассматриваться как *механизм отображения внешнего уровня в концептуальный*.

Представление данных, поддерживаемое каждым из компонентов внешнего уровня, характеризуется структурными свойствами этого представления, совокупностью операций над элементами соответствующей структуры и ограничениями целостности. Языковые средства описания структуры данных и манипулирования ими для каждого из представлений данных внешнего уровня позволяют определить внешний уровень ADABAS как *управляемый*.

Модель данных, поддерживаемая средствами ДИОД, СГО и ДПС и соответствующими языками, практически совпадает с моделью данных, используемой на концептуальном уровне. Близость представлений данных, применяемых в средствах ДИОД, ДПС и СГО, к представлениям данных концептуального уровня обеспечивает сравнительно простое и эффективное отображение внешних моделей данных, поддерживаемых этими компонентами, в базовую модель данных концептуального уровня.

Представление данных средств ДИОД

Представление о структуре БД пользователя средств ДИОД включает все структурные понятия, употребляемые на концептуальном уровне, поэтому с точки зрения администратора прикладных задач БД является *совокупностью взаимосвязанных файлов,* а элементами ее структуры - *атрибут* и *группа*.

Схема связей файлов концептуального уровня БД отображается на рассматриваемый уровень представления данных в полном объеме, а структура каждого файла БД описывается его подсхемой, называемой в ADABAS *внешней схемой*.

Внешняя схема файла вводится в словарь данных и определяет свойства групп и атрибутов, входящих в структуру файла. При этом структура файла внешнего уровня может использовать все атрибуты файла, описанные на концептуальном уровне, включая *поисковые* и *производные атрибуты*.

Атрибуту может быть назначено новое имя, при необходимости могут быть изменены тип и длина значения, указаны имена заголовков столбцов и шаблоны для вывода значений атрибутов в составе табличных отчетов. Допускается новый тип значения — дробное десятичное число с фиксированной точкой.

Состав групп внешней схемы файла определяется исходя из требований прикладной задачи, и может отличаться от состава групп концептуального уровня вплоть до отмены существующих в концептуальной схеме групп и образования новых групп. Каждый файл получает символьное имя в отличие от числового номера файла концептуального уровня, причем один и тот же файл может быть описан многократно под разными именами и с изменением

его структуры. Связи между файлами и ассоциации записей файлов при этом остаются неизменными.

Операции в рамках рассматриваемого представления данных обеспечивают ассоциативный поиск записей файлов с учетом их связывания, чтение записей с использованием логической и физической упорядоченности записей файла, выборку значений атрибутов записей, ввод, корректировку записей и их удаление. Указанные операции выполняются в ADABAS с помощью языка диалогового программирования задач.

Представление данных средств генерации отчетов

Средства генерации отчетов функционируют только в пакетном режиме и позволяют формировать сложные отчеты в условиях ограниченных аппаратных ресурсов. СГО использует структуру данных, совпадающую со структурой данных средств ДИОД, однако состав операций этого компонента системы является подмножеством операций средств ДИОД, в частности средства генерации отчетов не предусматривают возможности обновления файлов БД.

Представление данных средств ДПС

Средства диалоговой подготовки справок обеспечивают диалоговый режим работы конечных пользователей с помощью языка запросов ДПС.

Структуры данных, используемые ДПС, аналогичны структурам, определяемым на внешнем уровне для средств диалоговой обработки данных.

Особенностью ДПС является возможность динамического определения *составной записи*. Составная запись принадлежит двум связанным файлам БД. Каждый экземпляр составной записи включает запись исходного файла и совокупность связанных с ней записей подчиненного файла БД.

Средства ДПС обеспечивают операции поиска записей, включая составные записи с учетом связей между файлами, а также выборку атрибутов записей. При выборке доступны все атрибуты составной записи; результат выборки выдастся на экран видеотерминала в виде иерархической таблицы. Модификация данных в языке ДПС не предусмотрена.

Представление данных средств программирования интерфейсных модулей

Интерфейсные модули предназначены для повышения производительности труда программистов при решении прикладных задач.

Интерфейсные модули разрабатываются администратором прикладных задач. Каждый интерфейсный модуль специфицирует некоторую структуру данных, отображает ее на структуру БД и определяет в рамках этой структуры операции, эквивалентные последовательности ряда операций концептуального уровня.

В дополнение к структурам файлов и связям между ними, определенным в базовой модели, в рамках интерфейсных модулей допускается определение иерархических структур в виде цепочки последовательно связанных файлов, которые содержат логические записи.

Логическая запись — это совокупность записей БД, одна из которых является записью исходного файла, а остальные записи принадлежат связанным друг с другом файлам, находящимся на других уровнях цепочки.

Таким образом, для того чтобы описать схему логической записи, необходимо задать цепочку связанных файлов БД, из которой выбираются и обрабатываются данные.

Представление данных средств форматированного ввода

Форматированный ввод данных осуществляется одной из утилит ведения БД. Единица загружаемых данных определяется как *документ,* который включает *реквизиты*, *множественные реквизиты*, *простые* и *периодические группы реквизитов*. При этом обеспечивается возможность распределения реквизитов и групп реквизитов по файлам БД, включая распределение экземпляров периодических групп реквизитов по записям файла БД.

Представление данных средств КИТ

Внешним по отношению к концептуальному уровню системы является и *компонент КИТ* - средство для описания и интерпретации автоматизированных технологий эксплуатации БД.

Технологические процессы по эксплуатации БД выполняются в интерактивном режиме путем последовательного исполнения операций над объектами предметной области АБД.

Особенностью представления данных, обеспечиваемого средствами КИТ, является отвлечение от структуры обрабатываемых объектов и переключение внимания на их поведение (изменение состояния объектов при выполнении операций).

Центральным понятием представления данных средств КИТ является *технологический процесс,* определяющий допустимую последовательность выполнения операций над объектами предметной области АБД, приводящую к успешному решению поставленной технологической задачи. Для описания технологического процесса используются следующие понятия: *технологическая операция*, *параметр операции*, *состояние обрабатываемых объектов* и *шаг диалога*.

В ходе выполнения технологического процесса АБД последовательно выполняет технологические операции, задает конкретные значения параметров операции, анализирует свершившееся в результате каждой операции событие и соотносит его с одним из возможных изменений состояния обрабатываемого объекта. Очередная операция может однозначно определяться новым состоянием объектов после выполнения предыдущей операции. Однако некоторые состояния объектов позволяют в рамках одного технологического процесса выполнять не одну, а несколько операций, выбираемых АБД в процессе диалога со средствами КИТ. Для описания таких ситуаций используется понятие *шага диалога.*

Полное описание технологического процесса включает описания всех используемых в нем технологических операций и шагов диалога, которые хранятся в словаре данных в виде соответствующих спецификаций.

Спецификация шага диалога содержит перечень возможных вариантов продолжения. *Спецификация операции* включает описание задаваемых параметров и перечень всех возможных послеоперационных состояний, отражающих события, которые могут произойти в ходе ее выполнения. *Спецификация технологического процесса* фиксируется посредством ссылок между спецификациями операций и шагов диалога.

Каждой технологической операции соответствует как спецификация, хранимая в словаре данных, так и *реализация*, представляющая собой программу на языке ДИОД. Поскольку от отдельной операции зависит ряд событий, которые могут произойти с объектами некоторого типа, то

совокупность всех связанных с объектами операций полностью определяет их поведение, достаточное для выполнения функций ЛБД по эксплуатации БД.

К объектам, представляющим интерес для АБД, в первую очередь относятся база данных, файл базы данных, словарь данных, промежуточный набор данных, журнал администратора базы данных. Именно для этих объектов имеется набор готовых к использованию операций, входящих в состав КИТ. Он используется как при выполнении стандартных технологических процессов эксплуатации БД, так и при конструировании новых технологий. Для новых операций множество типов обрабатываемых объектов можно расширять путем создания новых спецификаций и программных реализаций.

Таким образом, все данные в ADABAS управляются, контролируются и накапливаются на основе общих принципов и механизмов, логика работы которых определяется единым концептуальным уровнем. Взаимодействие всех компонентов программного обеспечения системы с БД осуществляется через концептуальный уровень посредством мультипрограммного интерфейса МПИ, который обеспечивает адаптацию системы к вычислительной среде функционирования, типу операционной системы и телемониторам.

Модульность построения ADABAS, при которой каждый компонент системы является независимым и обладает собственными языковыми средствами, предоставляет пользователям широкие возможности по компоновке конкретной версии ADABAS для реальной вычислительной среды в соответствии с имеющимися у пользователя ресурсами.

8.5. Структура хранения данных СУБД ADABAS

Организация среды хранения данных

Среда хранения данных должна обеспечивать эффективное использование вычислительных средств при существующем уровне развития запоминающих устройств. Физическое представление данных и расположение их на запоминающих устройствах определяют *физическую организацию данных*. Она зависит от используемых методов поиска записей, ввода новых и удаления старых записей.

База данных ADABAS размещается на устройствах прямого доступа. Записи БД запоминаются в блоках *устройств прямого доступа*. Размер

блока выбирается с учетом условия эффективного размещения целого числа блоков на дорожке используемых типов устройств прямого доступа.

Размещение записей в блоке обеспечивает более эффективное использование внешней памяти. Для экономии внешней памяти, занимаемой данными, существуют разнообразные методы сжатия:

– исключение из записей полей с отсутствующими значениями атрибутов;

– хранение чисел в упакованном формате;

– подавление повторяющихся символов;

– кодирование часто используемых значений атрибутов;

– посимвольное кодирование и т. д.

Применение того или иного способа представления данных в конкретной СУБД в большой степени зависит от требований, предъявляемых к данной системе, и принятых в ней методов организации поиска данных. Для адекватного отображения структур концептуального уровня на внутренний в системах используются различные *вспомогательные данные* (указатели, индексы и т. д.), которые хранятся вместе с данными или отдельно от них. В ADABAS принято раздельное хранение вспомогательных данных и самих данных.

Структурными элементами базы данных ADABAS на внутреннем уровне являются (рис. 19): *накопитель*, *ассоциатор*, *рабочий набор* и *вспомогательные наборы данных*.

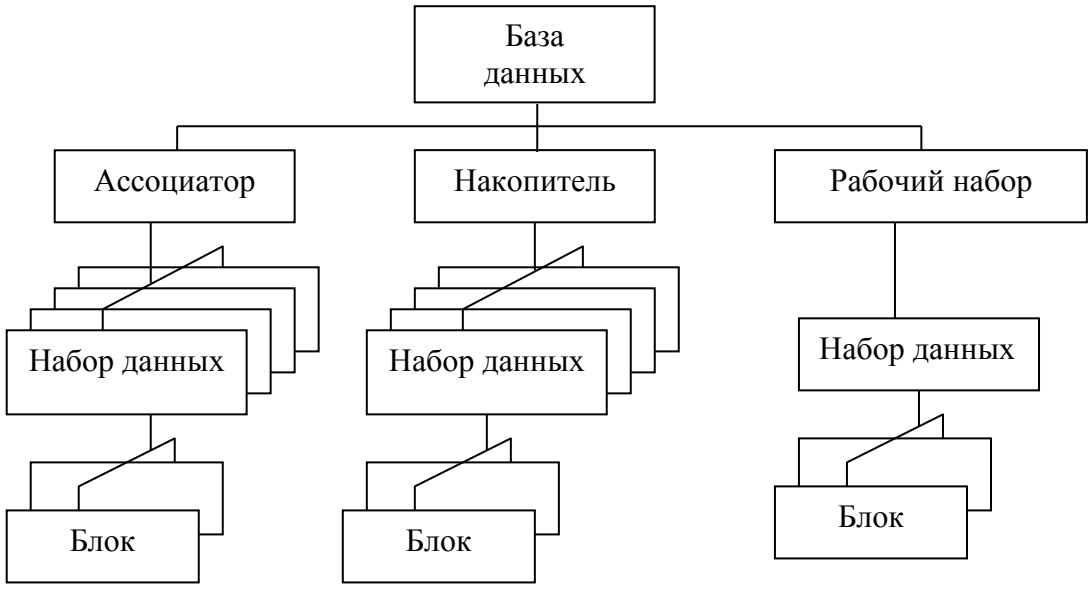


Рис. 19. Структура базы данных

Накопитель, ассоциатор и рабочий набор размещаются в наборах данных ОС ЕС, которые имеют прямую организацию и могут быть многотомными. Накопитель и ассоциатор занимают 1-5 наборов данных. Рабочий набор размещается только в одном наборе данных. Набор данных состоит из блоков фиксированной длины. В системе используются также вспомогательные наборы данных: *журнал команд*, *журнал изменений*, *временный набор*, *набор для сортировки*.

Структура накопителя

Накопитель, который занимает не более пяти наборов данных, предназначен для хранения записей файла БД. Каждому файлу выделяется до пяти *экстентов*, размещение которых отмечается в *таблице размещения файла (ТРФ)*, находящейся в *ассоциаторе*.

Размер блока зависит от типа устройства внешней памяти. *Блок набора данных* накопителя состоит из поля длины, содержащего общую длину занятой части блока, и записи файла. Размер записи не должен превышать размера блока. В блоке может находиться несколько записей.

Неиспользованная память в блоках учитывается в *таблице свободной памяти (ТСП)*, находящейся в ассоциаторе, и используется при вводе и корректировке записей.

Запись содержит *поле длины* и поле *ВСН (ISN)*, за которыми следуют значения атрибутов в соответствии с описанием на ЯОД. ВСН является уникальным ключом записи. Физический адрес блока, в котором находится запись с данными ВСН, определяется посредством *преобразователя адреса*, размещенного также в ассоциаторе.

Атрибуты представлены в записи с указателями длины значения за исключением атрибутов, определенных на ЯОД с постоянной длиной. Атрибуты без признака *Подавление*, значения которых отсутствуют в записи, представляются двумя байтами, первый из которых содержит значение длины, а второй - признак «пустого» значения (пробел для символьных значений и нуль - для числовых).

Атрибуты с признаком *Подавление*, значения которых отсутствуют в записи, представлены *счетчиком отсутствующих значений*. Если в записи содержится подряд несколько таких атрибутов, то счетчик будет содержать

количество этих атрибутов. Неопределенные значения атрибутов, являющиеся последними в записи, в памяти никак не представляются. Значения множественного атрибута хранятся как последовательность значений с указателем длины и предшествующим однобайтовым счетчиком числа экземпляров. *Периодическая группа* начинается с однобайтового счетчика, указывающего самый большой номер из хранимых в группе экземпляров.

Изменение значения множественного атрибута на «0» или «_» уменьшает значение счетчика на 1, а изменение значений экземпляра группы на «0» или «_» - не уменьшает.

В процессе работы с БД производятся *включение, удаление* и *модификация записей*. Удаление записи вызывает освобождение памяти, которая становится доступной для дальнейшего использования и отмечается в соответствующем поле таблицы свободной памяти. Модификация записи может вызвать уменьшение или увеличение ее размера. Уменьшение длины записи также ведет к освобождению памяти и корректировке поля для соответствующего блока в таблице свободной памяти.

Увеличение длины записи производится за счет свободной памяти в блоке. В случае ее отсутствия запись переносится в блок, имеющий достаточно свободной памяти. Перемещение записей снижает производительность системы, так как требует дополнительных операций ввода-вывода. С целью сокращения количества перемещений записей при их модификации вводится коэффициент заполнения блоков при первоначальной загрузке файла БД. Величина резерва свободной памяти в блоках выбирается исходя из условий использования данного файла. Такой способ размещения записей снижает частоту *реорганизации файла*.

Вводимая новая запись помещается в блок с достаточным объемом свободной памяти. В случае использования *рандомизированного метода доступа* запись помещается в блок с требуемым адресом, и если в этом блоке запись разместиться не может, то она помещается так же, как в случае *нерандомизированного доступа*. Доступ к такой записи будет осуществляться посредством инвертированных списков. Схема размещения записи в блоке показана на рис. 20.

При создании файла БД в накопителе выделяется *экстент памяти* требуемых размеров для размещения записей.

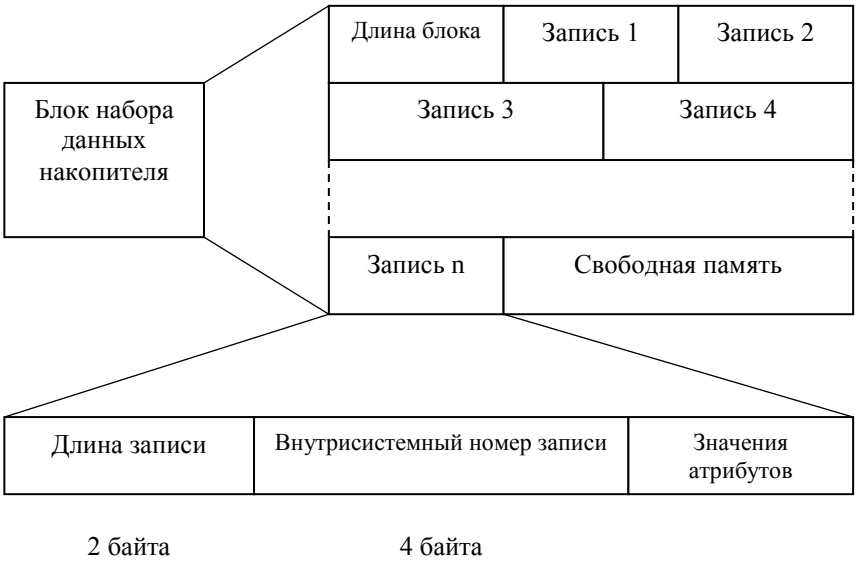


Рис. 20. Размещение записей в блоке

В процессе эксплуатации система может выделить файлу, по мере необходимости, дополнительно 4 раза по одному экстенту памяти. Увеличение количества экстентов снижает производительность системы. При снижении производительности до неприемлемых значений следует провести реорганизацию БД средствами АБД.

На рис. 21. представлена структура записи файла СТУДЕНТ для следующей карточки студента:

НОМЕР КАРТОЧКИ	115573
ФАМИЛИЯ	Ухов
ИМЯ-ОТЧЕСТВО	Петр Иванович
УЧЕБНАЯ ГРУППА	241
СЕССИЯ	1
ЭКЗАМЕНЫ	математика, физика, электротехника
ОЦЕНКИ	5, 4, 4
СЕССИЯ	2
ЭКЗАМЕНЫ	математика, физика, электротехника

ОТМЕТКИ 4, 4, 5

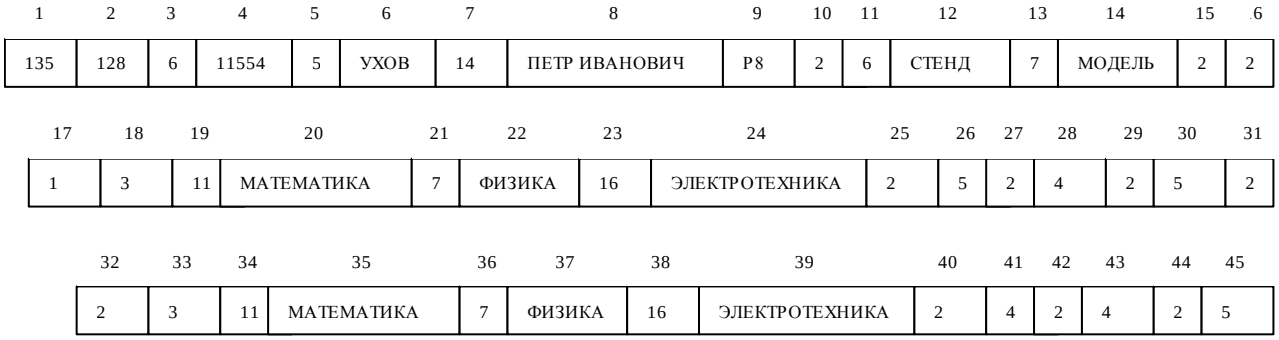


Рис. 21. Структура записи файла СТУДЕНТ

Поля записи (для удобства они пронумерованы) содержат:

1 - общую длину записи;

2 - ВСН;

3, 5, 7, 11, 13, 2, 19, 21, 23, 25, 27, 29, 31, 34, 36, 38, 40, 42, 44 - длину значений атрибутов;

4, 6, 8, 9, 12, 14, 17, 20, 22, 24, 26, 28, 30, 32, 35, 37, 39, 41, 43, 45 - значения атрибутов;

10, 18, 33 — счетчики экземпляров множественного атрибута;

15 - счетчик экземпляров повторяющейся группы.

Структура ассоциатора

Ассоциатор содержит сведения о структуре данных *концептуального* и *внутреннего уровней* БД в виде *таблиц, списков* и т. д., помещенных в блоки, и служит для взаимного отображения этих структур и выполнения операций над ними.

Таблицы и списки ассоциатора формируются средствами АБД при создании и ведении БД с помощью *словаря данных* и модифицируются системой во время ее функционирования. Основная информация о размещении БД сосредоточена в *таблице распределения памяти (ТРП)*.

ТРП занимает один блок ассоциатора и имеет постоянный адрес. Она содержит

– имя и номер БД;

– максимальное число файлов, которые могут быть загружены в БД;

– число загруженных файлов;

– номера системных файлов;

– адреса экстентов, выделенных для ассоциатора, накопителя, рабочего набора;

– типы устройств, на которых они размещены, неиспользованные области памяти и т. д.

Структура ТРП фиксированная, т. е. каждой характеристике отведено постоянное поле.

Данные о размещении файла в БД содержит *таблице размещения файла (ТРФ)*. Каждому файлу соответствует своя ТРФ, которая занимает один блок ассоциатора. Ее местоположение в ассоциаторе определяется номером файла.

ТРФ составляется средствами АБД во время создания файла БД. В ней содержатся:

– имя файла;

– номер файла;

– имя атрибута рандомизированного доступа;

– загрузочная схема связей файлов;

– коэффициент заполнения блоков файла в накопителе;

– коэффициент заполнения блоков инвертированных списков в ассоциаторе;

– сведения о размещении файла в накопителе, инвертированных списков и их индексов в ассоциаторе, преобразователя ВСН записей в адрес блока файла, таблицы учета свободной памяти в блоках накопителя и т. д.

Описание *логической структуры файла* на ЯОД транслируется в *таблицу определения данных (таблица FDT)*, которая представляет собой загрузочную форму схемы файла.

Таблица FDT для каждого файла имеет фиксированный адрес и занимает 4 блока ассоциатора, поля этой таблицы содержат признаки свойств каждого атрибута файла. Таблица используется программой УВД для взаимного отображения структур концептуального и внутреннего уровней. Прикладным программам обеспечивается доступ к таблице FDT с помощью специальной команды ЯМД.

Блоки накопителя, отведенные под данные, имеют различную степень заполнения. Ввод, корректировка и удаление записей требуют учета свободной памяти в блоках. Учет производится с использованием ТСП. Эта

таблица состоит из полей длиной в 1 байт. Номер поля от начала таблицы соответствует номеру блока в накопителе.

Число в соответствующем поле, умноженное на 16, определяет размер свободной памяти (в байтах) в данном блоке накопителя. ТСП формируется при создании БД средствами АБД и модифицируется во время работы системы. Размер ТСП в байтах численно равен количеству блоков накопителя.

Свободные области памяти ассоциатора и накопителя учитываются в *таблице свободных областей (ТСО)*. Она занимает один блок ассоциатора и имеет фиксированный адрес. В этой таблице производится отметка о свободных *экстентах памяти* накопителя и ассоциатора. Таблица формируется при создании БД средствами АБД и модифицируется во время функционирования системы.

Ряд операций, инициируемых командами ЯМД, выполняется с помощью *инвертированных списков*, состоящих из ВСН записей файла.

Инвертированные списки хранятся в блоке набора данных ассоциатора совместно с заголовочной частью, состоящей из значения атрибута, которому соответствует список, длины этого значения и длины инвертированного списка. В одном блоке размещается несколько списков, но все они должны соответствовать значениям одного атрибута. Список отсортирован в порядке возрастания значений ВСН, поэтому первый ВСН оказывается младшим элементом списка.

С целью уменьшения перемещения списков при вводе, корректировке и удалении записей блоки при загрузке файлов заполняются не полностью. Величина резерва задается параметрически при создании файла и может быть изменена утилитой реорганизации ассоциатора в процессе эксплуатации БД. При переполнении блока в результате расширения списка автоматически выделяется новый блок, и часть списка переносится в него так, чтобы в старом блоке остался резерв памяти, установленный АБД.

Доступ к требуемому инвертированному списку осуществляется с помощью *многоуровневого индекса*, содержащего *индекс значений* и *старшие индексы* (рис. 22.).

Индекс значения состоит из записей, каждая из которых включает длину атрибута, его значение, первый ВСН списка и адрес блока инвертированных

списков. Значение атрибута то же, что и в первом инвертированном списке адресованного блока, а BCH является младшим в этом списке и указывает на то, что список не имеет продолжения. Таких записей в индексе значений будет столько, сколько блоков занимают инвертированные списки данного атрибута. Запись индекса значения, указывающая на блок с инвертированным списком, начало которого находится в предыдущем блоке, вместо младшего BCH-списка этого блока содержит 0. Записи индекса значения, относящиеся к определенному атрибуту, объединяются в блоки. В блоке может находиться только целое число записей. Способ выделения новых блоков тот же, что и для инвертированных списков.

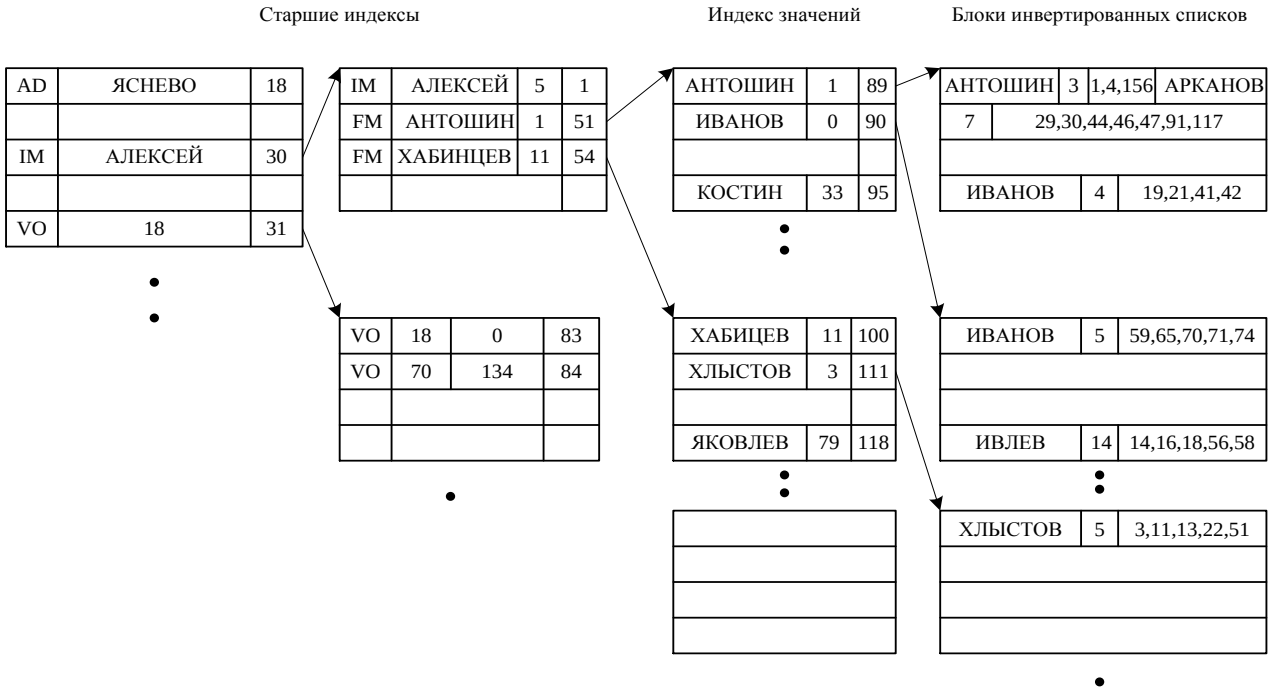


Рис 22. Инвертированные структуры

Нужный блок индекса значения, относящегося к данному атрибуту, отыскивается посредством *старшего индекса*. Блок старшего индекса состоит из записей. Каждая запись адресует один блок индекса значений. Запись включает код имени атрибута, поле признаков атрибута, длину значений атрибута, BCH и адрес блока индекса значения. На все поисковые атрибуты заведены записи в старшем индексе. Поисковые атрибуты, не имеющие значений в записях файла, представлены в записях старшего индекса нулевыми значениями полей длины, BCH и адреса блока индекса значений. В поле признаков атрибута указываются характеристики атрибута:

формат хранения, *принадлежность к повторяющейся группе*, *признак множественного атрибута*.

Каждая запись содержит указатель на один блок индекса значений. Количество блоков старшего индекса зависит от количества блоков индекса значений и, в конечном счете - от объема БД. В случае если количество блоков старшего индекса больше одного, то для поиска требуемого блока старшего индекса создается блок старшего индекса следующего уровня иерархии такой же структуры и т. д. Адрес самого верхнего уровня индекса для данного файла указывается в таблице размещения файла.

Связь ВСН записи с номером блока в накопителе, в котором находится эта запись, осуществляется посредством *преобразователя адреса*, который представляет собой таблицу, состоящую из трехбайтовых элементов. Элемент, относительный номер которого равен значению ВСН, содержит номер блока накопителя, в котором находится запись с этим ВСН. Размер таблицы определяется максимальным значением ВСН в данном файле, который задается как параметр при создании этого файла.

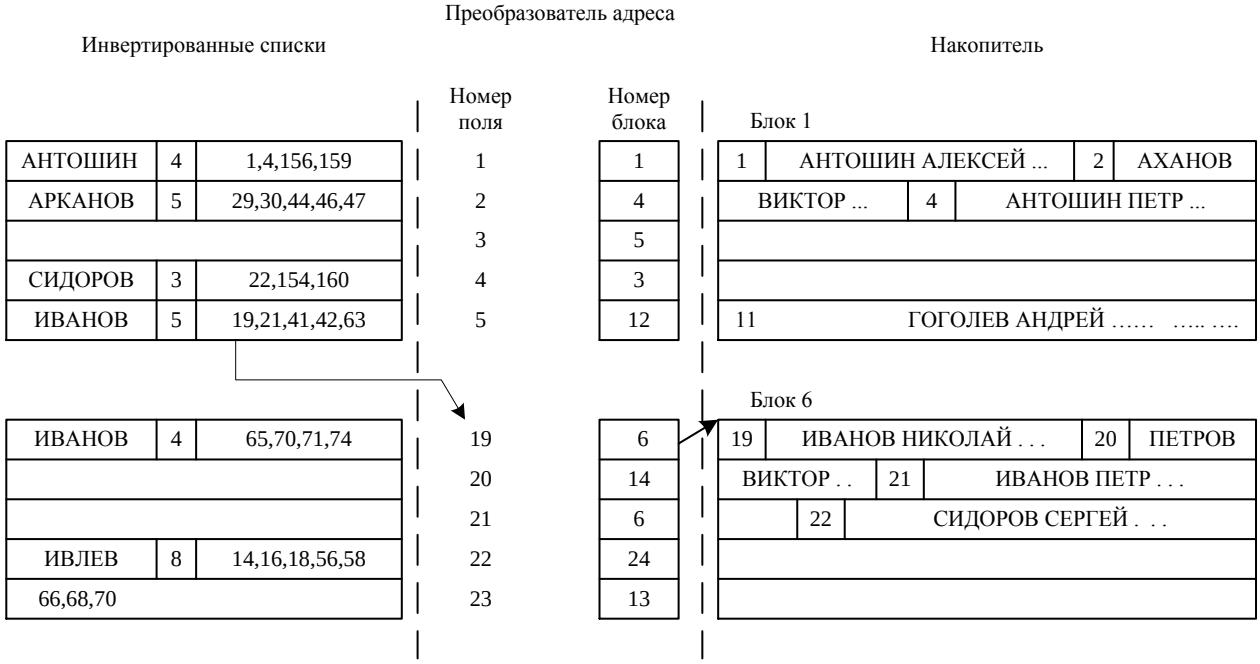


Рис. 23. Пример преобразования ВСН в адрес

При использовании всех выделенных блоков под преобразователь адреса данного файла система автоматически выделяет дополнительный экстент памяти, используя ТСО. Дополнительно система может выделить не более четырех экстентов, после чего требуется реорганизация ассоциатора. Адреса выделенных экстентов содержатся в ТРФ соответствующего файла.

Схематически связь ВСН с блоками накопителя посредством преобразователя адреса показана на рис. 23.

Структура рабочего и вспомогательных наборов данных

Рабочий набор данных занимает один экстент памяти и состоит из блоков фиксированной длины. Он предназначен для временного размещения промежуточной информации, необходимой в процессе работы с БД. В составе рабочего набора выделяется область *оперативного журнала изменений*, используемого для поддержания логической целостности БД, *области промежуточных списков ВСН*, а также *области результирующих списков ВСН*.

Область оперативного журнала используется циклически. Она должна быть достаточно большой, чтобы поместить данные за интервал времени между *контрольными точками*. Величина этой области задается при запуске системы как параметр. В область оперативного журнала помещаются записи, подлежащие корректировке, и откорректированные пользовательские данные, а также *таблица значений поисковых атрибутов*.

Область промежуточных списков ВСН предназначена для размещения списков ВСН, полученных командами поиска и используемых в логических операциях *объединения*, *пересечения*, *отрицания*. Эти списки управляются с помощью двух таблиц, расположенных в программе УБД, - *таблицы результирующих списков ВСН* и *таблицы адресов блока*.

Область результирующих списков ВСН содержит списки ВСН с идентификатором, состоящим из *идентификатора команды* и *идентификатора программы* (*пользователя*). Эти списки могут быть многократно прочитаны при выдаче команды с идентификатором той команды, по которой они были найдены.

Для работы ряда утилит предназначен *набор для сортировки* и *временный набор данных*. Эти наборы имеют прямую организацию с фиксированной длиной блока. Набор для сортировки используется при создании и модификации инвертированных списков средствами АБД. Временный набор данных применяется в качестве буферной памяти при загрузке БД для создания инвертированных списков. При функционировании системы эти наборы не используются.

Журнал изменений, в котором фиксируются все изменения, производимые в БД, размещается в последовательном наборе данных. Каждому сеансу работы системы соответствует свой набор данных, идентифицируемый номером сеанса.

Во время работы системы, выполняемые команды ЯМД протоколируются в *журнале команд*. Для журнала команд в системе отведены два прямых набора данных либо один последовательный набор данных. Прямые наборы данных применяются в цикле, т. е. после заполнения второго набора вновь используется первый.

Журнал команд может помещаться и в один последовательный набор данных. Способ размещения журнала и размер прямых наборов данных выбирает АБД при запуске системы исходя из целей использования информации журнала.

Операции на внутреннем уровне

Команды ЯМД выполняются программой УБД с использованием таблиц ТРП, ТРФ, ТОФ, ТСП**,** ТСО, инвертированных списков и преобразователя адреса. Команды ЯМД вызывают исполнение определенной последовательности элементарных операций. Операции над структурами внутреннего уровня выделены в программе УБД в специальный интерфейсный модуль. Этот модуль обеспечивает также обращение утилит к наборам системы.

Выделение функций, связанных с операциями над структурами внутреннего уровня, в отдельный модуль повышает независимость остальных модулей программы УБД и утилит от операционной системы и физических устройств, содержащих наборы данных. Обращение к интерфейсному модулю осуществляется посредством *управляющей таблицы*. Ее поля содержат следующие параметры:

– код элементарной операции;

– адреса входных и выходных данных;

– код завершения операции.

Интерфейсный модуль осуществляет открытие и закрытие БД, журнала команд, журнала изменений, временных наборов, производит чтение и запись данных, выдачу сообщений на консоль оператора. Для выполнения этих операций интерфейсному модулю указывается *код операции*, *адрес*

входных данных или *адрес буфера для выходных данных*. Интерфейсный модуль обеспечивает обращение к операционной системе для того, чтобы выполнить следующие действия:

– выдать текущую дату и время;

– осуществить запрос на ресурс;

– освободить ресурс;

– произвести запрос на оперативную память;

– освободить оперативную память;

– выдать информацию о томах наборов данных БД;

– выдать информацию о внешних устройствах наборов БД;

– дать специальную информацию для программ автономного ведения данных.

Выполнение любой команды ЯМД сопровождается записью в журнал команд *кода команды*, *времени выполнения*, *идентификатора команды* и т. д. Команды чтения могут требовать обращений к ассоциатору, накопителю и рабочему набору. Рабочий набор нужен в команде чтения только при указании «читать следующий», для получения сохраненного в нем списка ВСН. Ассоциатор используется для получения системных таблиц, преобразователя адреса и инвертированных списков, однако чтение блоков, в которых они хранятся, производится только в случае их отсутствия в оперативной памяти. Блок помещается в специально выделенную область оперативной памяти и в случае полного ее применения производится замещение блока, который давно не использовался. Таким образом, сокращается количество обращений к внешним устройствам памяти. Команды поиска используют рабочий набор для размещения промежуточных списков ВСН.

Команды модификации и ввода записи в отличие от команд чтения и поиска дополнительно обращаются к журналу изменений и оперативному журналу: прежде чем модифицированные блоки поместятся в накопитель и ассоциатор, происходит запись этих блоков до изменения и после изменения в журнал изменений и оперативный журнал. Запись в журналы производится немедленно при указании специального признака в командах ввода и модификации или при выдаче команд ET, CL и создании контрольной точки.

9. Основы программирования на языке Natural

9.1. Внесение записей в БД

Внесение записей в БД осуществляется при помощи оператора STORE. Результатом работы оператора STORE является добавление записи в существующий указанный набор данных ADABAS.

Синтаксис данного оператора описывается следующим образом (рис. 24. – структурный режим, и рис. 25. – режим отчета):

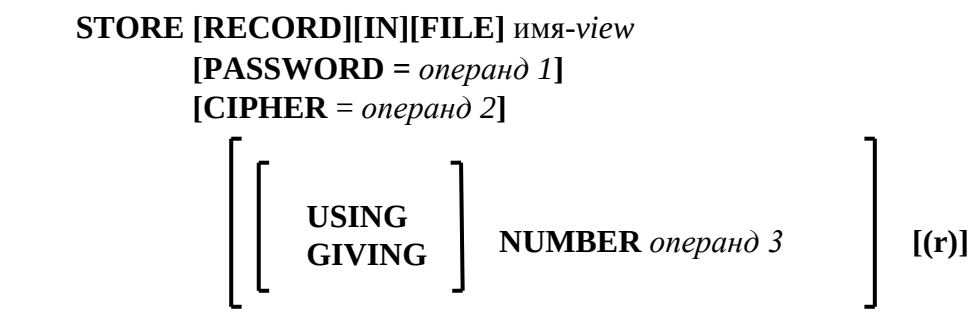


Рис. 24. Синтаксис оператора STORE (структурный режим)

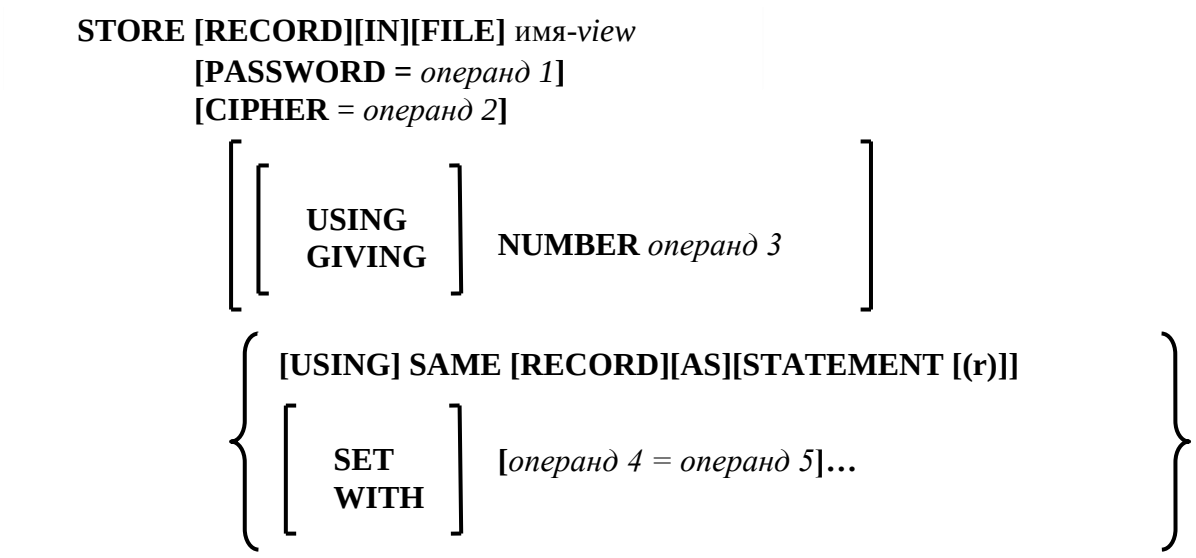


Рис. 25. Синтаксис оператора STORE (режим отчета)

Функция оператора

Оператор STORE используется для добавления записи в базу данных.

При работе с базами данных DL/I этот оператор может использоваться для добавления элемента сегмента.

Если набор данных определен с главным (primary) ключом, то должно быть предусмотрено значение для поля главного ключа.

В случае базы данных GSAM записи должны быть добавлены в конце базы данных (ограничения GSAM).

Примечание для баз данных SQL:

Этот оператор может использоваться для добавления строки в таблице. Предложения PASSWORD, CIPHER и GIVING NUMBER использоваться не могут. Для баз данных SQL оператор STORE соответствует оператору INSERT.

Примечание для баз данных VSAM:

Если набор данных определен с главным (*primary*) ключом, то должно быть предусмотрено значение для поля главного ключа.

Далее приводим описание следующих параметров оператора:

имя view

Имя представления пользователя (имя *view*) должно быть определено либо в операторе DEFINE DATA, либо вне программы в глобальной или локальной области данных (см. гл. 7.).

В режиме отчета - если нет оператора DEFINE DATA - имя *view* может также быть именем DDM.

PASSWORD/CIPHER

Данные предложения применимо только для баз данных ADABAS или VSAM.

Предложение PASSWORD используется для указания пароля при модификации данных в защищенном паролем файле.

Предложение CIPHER используется для указания шифра при модификации данных в зашифрованном файле.

USING/GIVING NUMBER

Это предложение используется только для баз данных ADABAS или VSAM.

Данная опция используется для добавления записи с заданным пользователем номером ISN ADABAS. Если запись с указанным ISN уже

существует, выдается сообщение об ошибке, и выполнение программы прерывается, если не задана обработка условия ON ERROR.

Примечание для баз данных VSAM:

Это предложение допустимо только для VSAM RRDS, для которого заданный пользователем RRN (относительный номер записи) соответствует описанному выше номеру ISN.

SET/WITH

SET/WITH используется в режиме отчета для определения полей вместе с их значениями. Любое поле файла, которое не определено в предложении SET, будет содержать нулевое значение в новой записи.

Это предложение не допускается, если используется оператор DEFINE DATA, т.к. в этом случае оператор STORE всегда ссылается на полное представление (*view*), определенное в операторе DEFINE DATA.

Соглашения для DL/I

Должны быть предусмотрены значения для последовательного поля сегмента и для всех ему предшествующих последовательных полей.

Поддерживаются только поля I/O (чувствительные).

Сегмент переменной длины записывается с минимальной длиной, необходимой для размещения всех полей, определенных в операторе STORE. Длина сегмента никогда не будет меньше минимального размера, указанного в макрокоманде SEGM в DBD.

Если множественное поле или периодическая группа определены переменной длины, то в конце сегмента записываются только реализации, заданные в операторе STORE, которые и определяют дойну сегмента.

USING SAME

USING SAME применяется в режиме отчета для того, чтобы указать, что в новой добавляемой записи должны использоваться те же значения поля, которые были прочитаны в операторе, на который ссылается оператор STORE (FIND, GET, READ).

Это предложение не допускается, если используется оператор DEFINE DATA, т.к. в этом случае оператор STORE всегда ссылается на полное представление (*view*), определенное в операторе DEFINE DATA.

Системная переменная *ISN

Системная переменная Natural *ISN содержит номер ISN СУБД ADABAS или номер RBA/RRN VSAM соответственно, назначенный новой записи в результате выполнения оператора STORE. Последующая ссылка на *ISN должна включать номер соответствующего оператора STORE.

Для баз данных VSAM *ISN доступен только для файлов ESDS и RRDS. * ISN не доступен для баз данных DL/I и SQL.

Пример

Предложенный ниже программный код обеспечивает внесение пользователем данных (посредством формы ввода) и последующее внесение записей в БД.

** как и при использовании любого оператора в рамках программы Natural, первым шагом является описание области данных:*

```
0010DEFINE DATA LOCAL
0020 01 APPLYER-VIEW VIEW OF APPLYER
0030 02 LN
…
1620END-DEFINE
1630
1640
```

** далее следует запрос ввода данных (пользователь при вводе данных имеет возможность покинуть окно ввода посредством кнопки ‘МЕНЮ’, которая определяется в рамках оператора SET KEY):*

```
 1650SET KEY PF1 NAMED 'МЕНЮ'
 1660INPUT 'ЛИЧНЫЕ ДАННЫЕ АБИТУРИЕНТА' //
 1670'НОМЕР КАРТОЧКИ СТУДЕНТА ' N1 (AD=M)/
 1680'ФАМИЛИЯ              ' LN /
 1690'ИМЯ                  ' FN /
 1700'ОТЧЕСТВО             ' PA /
 1710'ПОЛ (мужской-1, женский-2)'SE /
 1720'ДАТА РОЖДЕНИЯ (ДДММГГГГ)' DB /
 1730'НАЦИОНАЛЬНОСТЬ       ' NA /
 1740'ФОРМА ОБУЧЕНИЯ (очная-1, заочная-2, вечерняя-3)'FL /
 1750'ЭКЗАМЕН ПО ВЫБОРУ (математика-1, информатика-2)'AE /
 1760'МЕДАЛЬ (есть-1, нет-0)  ' AW /
 1770'МЕДЕЦИНСКАЯ СПРАВКА (есть-1, нет-0)' MN /
 1780'КОПИЯ/ОРИГИНАЛ МЕДИЦИНСКОЙ СПРАВКИ (оригинал-1, копия-2)' ZA /
```

```
1790'КОЛИЧЕСТВО ФОТОГРАФИЙ  ' QP /
1800
1810IF *PF-KEY = 'PF1' RUN 'MAINA'
1820END-IF
1830
```

** проверка данных на соответствие и возвращение пользователя к полям ввода, данные в которых были введены некорректно:*

```
1840IF NOT (N1 = MASK (00000001-99999999))
1850REINPUT TEXT 'ВВЕДИТЕ КОРРЕКТНЫЙ ИНДЕВИДУВЛЬНЫЙ НОМЕР АБИТУРИЕНТА (8 цифр)' (CD=RE) MARK 1
1860END-IF
1870IF FN = ' '
1880REINPUT TEXT 'ВВЕДИТЕ ИМЯ АБИТУРИЕНТА' (CD=RE) MARK 3
1890END-IF
1900IF LN = ' '
1910REINPUT TEXT 'ВВЕДИТЕ ФАМИЛИЮ АБИТУРИЕНТА' (CD=RE) MARK 2
1920END-IF
1930IF PA = ' '
1940REINPUT TEXT 'ВВЕДИТЕ ОТЧЕСТВО АБИТУРИЕНТА' (CD=RE) MARK 4
1950END-IF
1960IF NOT (SE = '1' OR = '2')
1970REINPUT TEXT 'ВВЕДИТЕ ПОЛ АБИТУРИЕНТА (мужской-1, женский-2)' (CD=RE) MARK 5
1980END-IF
1990IF NOT (DB = MASK (DDMMYYYY))
2000REINPUT TEXT 'ВВЕДИТЕ ДАТУ РОЖДЕНИЯ КОРРЕКТНО (ДДММГГГГ)' (CD=RE) MARK 6
2010END-IF
2020IF NA = ' '
2030REINPUT TEXT 'ВВЕДИТЕ НАЦИОНАЛЬНОСТЬ АБИТУРИЕНТА' (CD=RE) MARK 7
2040END-IF
2050IF NOT (FL = '1' OR = '2' OR = '3')
2060REINPUT TEXT 'ВВЕДИТЕ ФОРМУ ОБУЧЕНИЯ АБИТУРИЕНТА КОРРЕКТНО (очная-1, заочная-2, вечерняя-3)' (CD=RE) MARK 8
2070END-IF
2080IF NOT (AE = '1' OR = '2')
```

```
2090REINPUT TEXT 'ВВЕДИТЕ НАИМЕНОВАНИЕ ЭКЗАМЕНА ПО ВЫБОРУ КОРРЕКТНО (математика-1, информатика-2)' (CD=RE) MARK 9
2100END-IF
2110IF NOT (AW = '1' OR = '0')
2120REINPUT TEXT 'ВВЕДИТЕ ДАННЫЕ О НАЛИЧИЕ МЕДАЛИ КОРРЕКТНО (есть-1, нет-0)' (CD=RE) MARK 10
2130END-IF
2140IF NOT (MN = '1' OR = '0')
2150REINPUT TEXT 'ВВЕДИТЕ ДАННЫЕ О МЕДИЦИНСКОЙ СПРАВКЕ КОРРЕКТНО (есть-1, нет-0)' (CD=RE) MARK 11
2160END-IF
2170IF NOT (ZA = '1' OR = '2')
2180REINPUT TEXT 'ВВЕДИТЕ ДАННЫЕ О КОПИИ/ОРИГИНАЛЕ МЕДИЦИНСКОЙ СПРАВКИ (оригинал-1, копия-2)' (CD=RE) MARK 12
2190END-IF
2200IF QP = ' '
2210REINPUT TEXT 'ВВЕДИТЕ ДАННЫЕ О КОЛИЧЕСТВЕ ФОТОГРАФИЙ, СДАННЫХ В ПРИЕМНУЮ КОМИССИЮ' (CD=RE) MARK 13
2220END-IF
```

** проверка, не соответствует ли индивидуальный номер абитуриента, данные которого заносятся в БД, индивидуальному номеру какого-либо абитуриента, данные которого были внесены в БД ранее:*

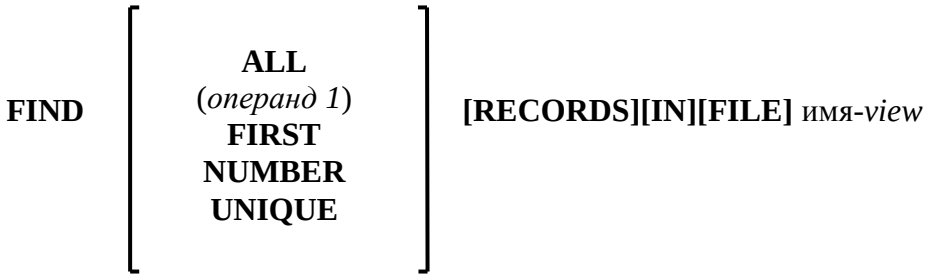
```
2250FIND NUMBER APPLYER-VIEW WITH N1 = N1
2260IF *NUMBER > 0 THEN
2270REINPUT 'АБИТУРИЕН С ТАКИМ ИНДЕВИДУАЛЬНЫМ НОМЕРОМ УЖЕ ЗАНЕСЕН В БАЗУ ДАННЫХ' (CD=RE) MARK 1
2280END-IF
2290
2300
…
6140
6150
```

** и, непосредственно, внесение данных в БД:*

```
6160STORE RECORD IN APPLYER-VIEW
6190END OF TRANSACTION
6210RUN 'MAINA'
6220END
```

9.2. Поиск записей в БД

В системе Natural поиск записей осуществляется при помощи оператора FIND. Синтаксис оператора может быть описан следующим образом (рис. 26., таблица 1):



[PASSWORD = *операнд 2***]**
[CIPHER = *операнд 3***]**
[WITH][[LIMIT] (*операнд 4*)**]** *базовый критерий поиска*
[*COUPLED-предложение***]…**4/42
[STARTING WITH ISN = *операнд 5***]**
[*SORTED-BY-предложение***]**
[*RETAIN-предложение***]**
[*WHERE-предложение***]**
[*IF-NO-RECORDS-FOUND-предложение***]**
Оператор…

END-FIND **(*структурный режим*)**
[**LOOP**] **(*режим отчета*)**

Рис. 26. Синтаксис оператора FIND

Описание операндов оператора FIND

Таблица 1

Операнд	Возможная структура					Возможные форматы											Разрешение ссылки	Динамическое определение
Операнд 1	C	S					N	P	I								да	нет
Операнд 2	C	S				A											да	нет
Операнд 3	C	S					N										да	нет
Операнд 4	C	S					N	P	I		B						да	нет
Операнд 5	C	S					N	P	I		B						да	нет

Функция оператора

Оператор FIND применяется для выборки некоторого набора записей из базы данных по критерию поиска, состоящему из полей, определенных как дескрипторы (ключи).

Для каждой выбранной записи данный оператор инициирует цикл обработки. В пределах цикла обработки можно обращаться к любому полю каждой записи. Для этого нет необходимости выдавать оператор READ после оператора FIND.

Соглашения для баз данных DL/I

В данном описание оператора FIND вместо термина "записи" нужно подставить термин "сегмент" (segment occurrence). При доступе к полю, начинающемуся после последнего байта данного сегмента, отображение этого поля в памяти заполняется в соответствие с его форматом (числами, пробелами и т.д.).

Соглашения для баз данных SQL

Предложения FIND FIRST, а также PASSWORD, CIPHER, COUPLED и RETAIN не допускаются.

Предложение FIND UNIQUE не допускается. (Исключение: на mainframe предложение FIND UNIQUE может использоваться для главных (primary) ключей; однако, это разрешается только для совместимости и не должно использоваться.)

Предложение SORTED BY соответствует в SQL предложению ORDER BY.

Базовый критерий поиска для таблицы базы данных SQL может быть задан точно так же, как и для файла ADABAS. Используемый в данном контексте термин "запись" соответствует в SQL термину "строка".

Соглашения для баз данных VSAM

Оператор FIND допустим только для key-sequenced (KSDS) и entry-sequenced (ESDS) наборов данных VSAM. Для ESDS должен быть определен дополнительный индекс для основного кластера.

Ограничения для сервера ENTIRE SYSTEM

Предложения FIND NUMBER и FIND UNIQUE , а также PASSWORD, CIPHER, COUPLED и RETAIN не допускаются.

Ограничение обработки - ALL/операнд 1

Количество подлежащих обработки записей из выбранного набора может быть ограничено с помощью операнда 1, который задается в виде числовой константы или имени числовой переменной, заключенных в круглые скобки. По желанию может быть задан ALL, который подчеркивает, что все выбранные записи должны быть обработаны.

Это ограничение распространяется на цикл, инициализируемый оператором FIND. В число обрабатываемых записей не входят записи, отвергнутые в соответствие с дополнительным критерием предложения WHERE.

```
FIND   (5)   • EMPLOYEES   WITH...
MOVE   10   TO   #CNT (N2)
FIND   (#CNT) EMPLOYEES   WITH...
```

Замечание! Если необходимо обработать 4-значное число записей, оно должно определяться с лидирующим нолем (Onnnn), так как Natural интерпретирует каждое 4-значное число, заключенное в круглые скобки, как номер строки оператора. Операнд 1 не влияет на размер набора ISN, который должен быть сохранен предложением RETAIN. Операнд 1 проверяется внутри цикла FIND. Если значение операнда 1 изменяется в пределах цикла FIND, это не отражается на количестве обработанных записей.

FIND FIRST, FIND NUMBER, FIND UNIQUE

Эти операторы используются для выборки первой записи выбранного набора (FIND FIRST), для определения количества записей в выбранном

наборе (FIND NUMBER) или для проверки того, что только одна запись удовлетворяет критерию поиска (FIND UNIQUE) .

имя view

Имя представления пользователя определяется либо в блоке DEFINE DATA, либо отдельно в глобальной или локальной области данных. В режиме отчета имя view может также быть именем DDM.

***Предложение* PASSWORD**

Замечание! Предложение PASSWORD применимо только для баз данных ADABAS или VSAM. Это предложение не допускается для сервера ENTIRE SYSTEM.

Предложение PASSWORD используется для указания пароля (операнд2) при поиске данных в защищенном паролем файле СУБД ADABAS. Если требуется доступ к защищенному паролем файлу, необходимо обратиться к сотруднику, ответственному за вопросы, связанные с использованием/назначением пароля.

Если пароль определен как константа, предложение PASSWORD должно всегда указываться в самом начале строки исходного текста; это является гарантией того, что пароль не будет печататься в исходном тексте программы. Для того чтобы данное предложение PASSWORD не высвечивалось в режиме телеобработки, пользователи должны предварительно ввести терминальную команду "%*".

Если предложение PASSWORD опущено, то применяется пароль, указанный в операторе PASSW.

Значение пароля не должно изменяться внутри цикла обработки.

Ниже показан пример использования предложения PASSWORD:

* *определение области данных:*

```
DEFINE DATA LOCAL
1 EMPLOY-VIEW VIEW OF EMPLOYEES
2 NAME
2 PERSONNEL-ID
1 #PASS (A8)
END-DEFINE
```

* *запрос ввода пароля пользователем:*

```
INPUT 'ENTER PASSWORD FOR EMPLOYEE FILE:' #PASS (AD=N)
LIMIT 2
```

** поиск в файле EMPLOYEE и вывод на экран фамилий и индивидуальных номеров сотрудников с фамилией "SMITH", при условии, что пользователь ввел верный пароль:*

```
FIND EMPLOY-VIEW
PASSWORD = #PASS
WITH NAME = 'SMITH'
DISPLAY NOTITLE NAME PERSONNEL-ID
END-FIND
```

** завершение программы:*

```
END
```

***Предложение* CIPHER**

Замечание! Предложение CIPHER применимо только для баз данных ADABAS. Это предложение не допускается для сервера ENTIRE SYSTEM.

Предложение CIPHER используется для указания шифра (операнд 3) при поиске данных в зашифрованном файле СУБД ADABAS. Если требуется доступ к зашифрованному файлу, необходимо обратиться к сотруднику, ответственному за вопросы, связанные с использованием/назначением шифра.

Шифр может быть определен как числовая константа (8 цифр) или как определенная пользователем переменная формата/длины N8.

Если шифр определен как константа, предложение CIPHER должно всегда указываться в самом начале строки исходного текста; это является гарантией того, что шифр не будет печататься в исходном тексте программы. Для того чтобы данное предложение CIPHER не высвечивалось в режиме телеобработки, пользователи должны предварительно ввести терминальную команду "%*".

Значение шифра не должно изменяться в пределах цикла обработки оператора FIND.

Ниже показан пример использования предложения CIPHER:

** определение области данных:*

```
DEFINE DATA LOCAL
1 EMPLOY-VIEW VIEW OF EMPLOYEES
2 NAME
2 PERSONNEL-ID
1 #PASS (A8)
1 #CIPHER (N8)
END-DEFINE
```

** запрос ввода пароля и ввода ключа шрифта пользователем:*

```
LIMIT   2
INPUT 'ENTER PASSWORD FOR EMPLOYEE PILE:' #PASS (AD=N)
/ 'ENTER CIPHER KEY FOR EMPLOYEE FILE:' #CIPHER (AD=N)
```

** осуществление поиска:*

```
FIND EMPLOY-VIEW
PASSWORD = #PASS
CIPHER = #CIPHER
WITH NAME = 'SMITH'
DISPLAY NOTITLE   NAME   PERSONNEL-ID
END-FIND
```

** конец программы:*

```
END
```

***Предложение* WITH**

Предложение WITH является обязательным и используется для задания базового критерия поиска, состоящего из ключевых полей (дескрипторов), определенных в базе данных.

Для файлов ADABAS в предложение WITH можно использовать дескрипторы, субдескрипторы, супердескрипторы, гипердескрипторы и фонетические дескрипторы. На мэйнфрейме можно также определить недескриптор (то есть поле, определенное в DDM с опцией "N").

Для файлов DL/I можно использовать только ключевые поля, определенные в DDM с опцией "D".

Для файлов VSAM можно использовать только ключевые поля VSAM.

Количество записей, которые будут выбраны на базе критерия предложения WITH, может быть ограничено с помощью ключевого слова LIMIT в сочетание с числовой константой или именем пользовательской переменной, заключенными в круглые скобки, которые содержат величину ограничения (операнд4). Если число выбранных записей превышает эту величину, программа будет завершаться с сообщением об ошибке.

Замечание! Если ограничение выражается 4-значным числом, оно должно определяться с лидирующим нолем (Onnnn), так как Natural интерпретирует каждое 4-значное число, заключенное в круглые скобки, как номер строки оператора.

Критерий поиска для файлов ADABAS - базовый критерий поиска

Синтаксис показан на рис. 27. и в таблице 2.

Описание операндов критерия поиска для файлов ADABAS

Таблица 2

Операнд	Возможная структура					Возможные форматы											Разрешение ссылки	Динамическое определение
Дескриптор		S	A			A	N	P	I	F	B	D	T	L			нет	нет
Значение	C	S				A	N	P	I	F	B	D	T	L			да	нет
Имя набора	C	S				A											нет	нет

дескриптор

Дескриптор, субдескриптор, супердескриптор, гипердескриптор или фонетический дескриптор ADABAS. Поле, определенное в DDM как недескриптор, может также быть определено.

Дескриптор, входящий в состав периодической группы, может быть задан с индексом или без него. Если индекс не задан, запись будет выбираться, если указанное значение найдется в любой реализации. Если индекс задан, запись будет выбираться, если указанное значение найдется в реализации с заданным индексом.

Индекс должен задаваться в виде константы. Диапазон индексов задавать нельзя.

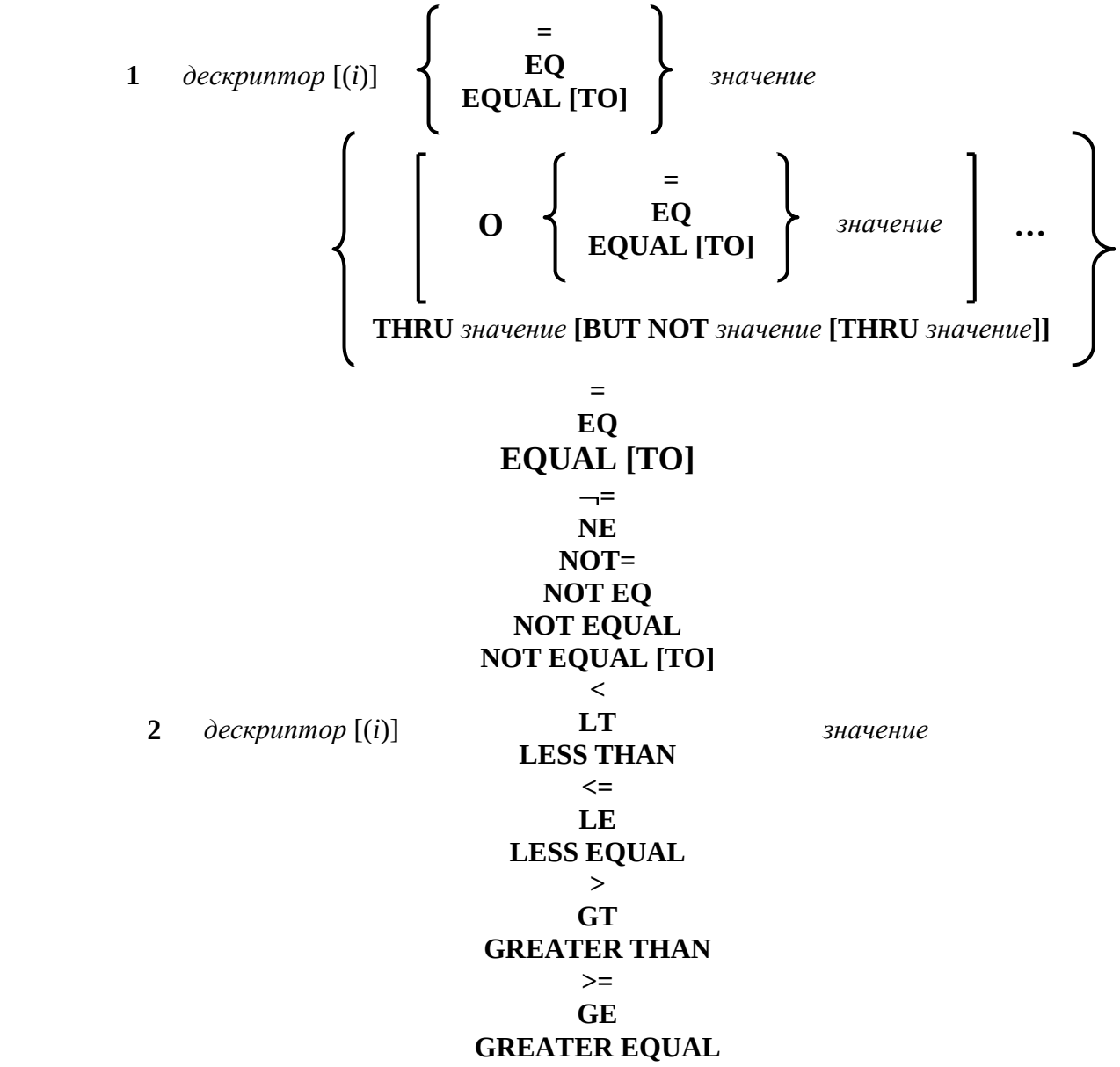


Рис. 27. Критерий поиска для файлов ADABAS

Для дескриптора, являющегося множественным полем, индекс указываться не должен. Запись будет выбираться, если значение обнаружится в любой реализации записи.

значение

Значение поиска. Форматы дескриптора и значения поиска должны быть совместимыми.

имя набора

Идентифицирует набор записей, предварительно выбранных оператором FIND, в котором было указано предложение RETAIN. Набор записей, на который ссылается оператор FIND, должен быть создан из того же самого физического файла ADABAS. Имя набора может быть определено как текстовая константа (максимум 32 символа) или как содержимое алфавитно-цифровой переменной. Имя набора не может использоваться в среде ENTIRE SYSTEM SERVER .

Примеры базового критерия поиска в предложение WITH:

FIND STAFF WITH NAME = 'SMITH’
FIND STAFF WITH CITY NE 'BOSTON'
FIND STAFF WITH BIRTH = 610803
FIND STAFF WITH BIRTH = 610803 THRU 610811
FIND STAFF WITH NAME = 'O HARA' OR = 'JONES' OR = 'JACKSON'
FIND STAFF WITH PERSONNEL-ID = 100082 THRU 100100 BUT NOT 100087 THRU 100095

Примеры базового критерия поиска с множественным полем:

Когда в базовом критерии поиска используется множественное поле, могут быть получены четыре различных варианта результата (поле MU-FIELD предполагается множественным):

1. FIND XYZ-VIEW WITH **MU-FIELD** = **'A'**

Этот оператор выбирает записи, в которых, по крайней мере, одна реализация поля MU-FIELD имеет значение "A".

2. FIND XYZ-VIEW WITH **MU-FIELD NOT EQUAL 'A'**

Этот оператор выбирает записи, в которых, по крайней мере, одна реализация поля MU-FIELD не имеет значение "A".

3. FIND XYZ-VIEW WITH **NOT MU-FIELD NOT EQUAL 'A'**

Этот оператор выбирает записи, в которых каждая реализация поля MU-FIELD имеет значение "A".

4. FIND XYZ-VIEW WITH **NOT MU-FIELD** = **'A'**

Этот оператор выбирает записи, в которых ни одна реализация поля MU-FIELD не имеет значение "A".

Критерий поиска с нулевым индикатором* (Null-indicator) - *базовый критерий поиска

Синтаксис показан на рис. 28. и в таблице 3.

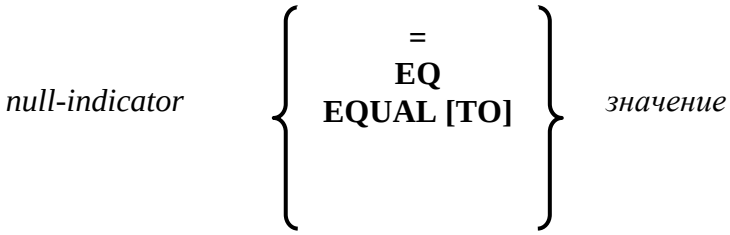


Рис. 28. Критерий поиска с нулевым индикатором

Описание операндов критерия поиска с нулевым индикатором (Null-indicator)

Таблица 3

Операнд	Возможная структура					Возможные форматы										Разрешение ссылки	Динамическое определение
null-indicator		S							I							нет	нет
Значение	C	S					N	P	I	F	B					да	нет

Комбинирование критериев поиска (для файлов ADABAS)

Базовые критерии поиска могут комбинироваться с помощью булевых операторов AND, OR и NOT. Для управления порядком обработки критериев поиска используются круглые скобки. Порядок обработки операторов следующий:

1. **( )** Круглые скобки
2. **NOT** Отрицание (только для базового критерия поиска).
3. **AND** И
4. **OR** ИЛИ

Базовые критерии поиска могут быть соединены логическими операторами для формирования комплексного выражения поиска. Синтаксис для такого комплексного выражения поиска следующий (рис. 29.):

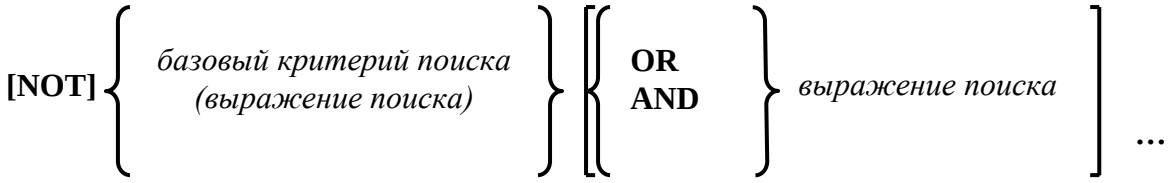


Рис. 29. Базовые критерии поиска

Примеры комплексного выражения поиска в предложение WITH:

FIND STAFF WITH BIRTH LT 19770101 AND DEPT = 'DEPT06'
FIND STAFF WITH JOB-TITLE = ‘CLERK TYRIST’ AND (BIRTH GT 19560101 OR LANG = 'SPANISH')
FIND STAFF WITH JOB-TITLE = 'CLERK TYRIST' AND NOT (BIRTH GT 19560101 OR LANG = 'SPANISH')
FIND STAFF WITH DEPT = 'ABC' THRU 'DEF' AND CITY = 'WASHINGTON1 OR = 'LOS ANGELES' AND BIRTH GT 19360101
FIND CAR WITH MAKE = 'ФОЛЬКСВАГЕН1 AND COLOR = 'RED' OR = 'BLUE' OR = 'BLACK'

Дескриптор - ключ - использование

При формировании базового критерия поиска пользователи СУБД Adabas могут использовать поля базы данных, которые определены как дескрипторы.

Субдескриптор, супердескриптор, гипердескриптор и фонетический дескриптор

При формировании критерия поиска в СУБД ADABAS могут быть использованы субдескрипторы, суп ер дескрипторы, гипердескрипторы и фонетические дескрипторы.

–Субдескриптор - это дескриптор, сформированный из части поля.

–Супердескриптор - это дескриптор, значение которого сформировано из одного или нескольких полей или частей полей.

–Гипердескриптор - это дескриптор, который сформирован по определенному алгоритму пользователя.

–Фонетический дескриптор - это дескриптор, который дает возможность пользователю осуществить фонетический поиск значений поля (например, имя сотрудника). Результатом фонетического поиска являются все значения, которые по произношению подобны значению, заданному для поиска.

Значения для субдескрипторов, супердескрипторов, фонетических дескрипторов

Значения, используемые для этих типов дескрипторов, должны быть совместимы с внутренним форматом дескриптора, Внутренний формат субдескриптора совпадает с форматом поля, из которого субдескриптор получен. Внутренний формат супердескриптора двоичный, если все поля, из которых он получен, определены с числовым форматом; иначе, супердескриптор имеет алфавитно-цифровой формат. Фонетические дескрипторы всегда имеют алфавитно-цифровой формат.

Значения для субдескрипторов и супердескрипторов могут быть определены следующим образом:

– Могут быть заданы числовые или шестнадцатеричные константы. Шестнадцатеричная константа должна использоваться для значения супердескриптора, который имеет двоичный формат (смотри выше).

– Значения переменных пользователя могут быть заданы с использованием оператора REDEFINE для выделения частей полей, из которых формируется значение субдескриптора или су пер дескриптора.

Использование дескрипторов, содержащихся в массивах базы данных

Дескриптор, который содержится в массиве полей базы данных, также может использоваться при формировании базового критерия поиска. Для баз данных ADABAS таким дескриптором может быть множественное поле или поле, входящее в состав периодической группы.

Дескриптор, входящий в состав периодической группы, может быть задан с индексом или без него. Если индекс не задан, запись будет выбираться, если указанное значение найдется в любой реализации. Если индекс задан, запись будет выбираться, если указанное значение найдется в реализации с заданным индексом. Индекс должен задаваться в виде константы. Диапазон индексов задавать нельзя. Для дескриптора, являющегося множественным полем, индекс указываться не должен. Запись будет выбираться, если значение обнаружится в любой реализации поля.

Примеры использования массивов базы данных:

В следующих примерах предполагается, что поле SALARY является дескриптором, входящим в состав периодической группы, а поле LANG является множественным полем.

1. FIND EMPLOYEES WITH SALARY LT 20000 (вызывает поиск по всем реализациям поля SALARY)

2. FIND EMPLOYEES WITH SALARY (1) LT 20000 (вызывает поиск только по первой реализации)

3. FIND EMPLOYEES WITH SALARY (1:4) LT 20000 /* недопустимо (для поля периодической группы в критерии поиска не должен задаваться; диапазон значений индекса)

4. FIND EMPLOYEES WITH LANG = 'FRENCH' (вызывает поиск по всем значениям поля LANG)

5. FIND EMPLOYEES WITH LANG f 1) = 'FRENCH' /* недопустимо (Для множественного поля в критерии поиска не должен задаваться индекс)

COUPLED-*предложение*

Это предложение применимо только для баз данных ADABAS. Предложение не разрешается в среде ENTIRE SYSTEM SERVER. Синтаксис данного предложения показан на рис. 30. и в таблице 4.

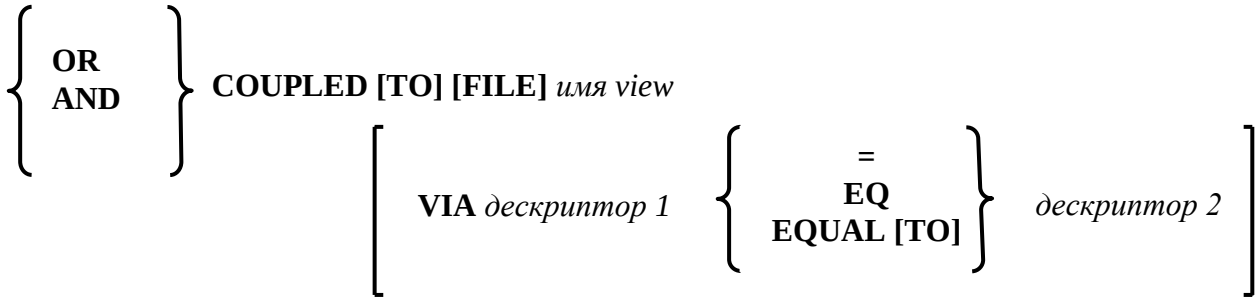


Рис. 30. Синтаксис COUPLED-предложения

Описание операндов COUPLED-предложения

Таблица 4

Операнд	Возможная структура	Возможные форматы	Разрешение ссылки	Динамическое определение
Дескриптор1	S A	A N P B	нет	нет
Дескриптор2	S A	A P B	нет	нет

Замечание! Предложение COUPLED может быть определено максимум 4 раза без предложения VIA; а с предложением VIA - до 42 раз.

Предложение COUPLED используется для организации поиска по связанным файлам СУБД ADABAS. Связанные файлы позволяют определять дескрипторы из различных файлов базы данных в критерии поиска одного оператора FIND.

В одном и том же операторе FIND в двух разных предложениях COUPLED не должен использоваться один и тот же файл ADABAS.

Имя набора (см. предложение RETAIN) нельзя определять в базовом критерии поиска.

Поля файла базы данных, указанного в предложение COUPLED, не доступны для последующего использования в программе до тех пор, пока для связанного файла не будет выдан другой оператор FIND или READ.

Замечание! Если используется предложение COUPLED, главное предложение WITH может быть опущено. Если опущено главное предложение WITH, в предложение COUPLED не могут быть определены ключевые слова AND/OR.

***Физическая связь файлов без предложения* VIA**

Файлы, используемые в предложении COUPLED без VIA, должны быть физически связаны с помощью соответствующей утилиты ADABAS.

Пример использования физически связанных файлов:

```
    * определение области данных:
DEFINE DATA LOCAL
1   EMPLOY-VIEW VIEW OF   EMPLOYEES
  2   NAME
1   VEHIC-VIEW VIEW OF   VEHICLES
  2   MAKE
END-DEFINE
    * поиск в физически связанных файлах:
FIND   EMPLOY-VIEW WITH   CITY = 'FRANKFURT'
              AND   COOPLED   TO VEHIC-VIEW WITH MAKE = 'VW'
    * вывод результатов на экран монитора:
DISPLAY NOTITLE   NAME
END-FIND
    * конец программы:
```

```
END
```

В приведенном выше примере использование поля NAME в операторе DISPLAY является допустимым, так как это поле содержится в файле EMPLOYEES, тогда как обращение к полю MAKE было бы недопустимым, так как поле MAKE содержится в файле VEHICLES, который был определен в предложении COUPLED.

В данном примере записи будут найдены только в том случае, если файлы EMPLOYEES и VEHICLES физически связаны.

***Логическая связь файлов - предложение* VIA**

Опция "VIA дескриптор 1 = дескриптор 2" позволяет логически связать несколько файлов ADABAS в запросе поиска. Дескриптор 1 является полем из первого представления (*view*), а дескриптор 2 - поле из второго представления (*view*). Два файла физически не связаны в ADABAS.

Пример использования предложения VIA:

```
    * определение области данных:
DEFINE   DATA   LOCAL
1   EMPLOY-VIEW VIEW OF   EMPLOYEES
    2   PERSONNEL-ID
    2   NAME
    2   FIRST-NAME
1 VEHIC-VIEW VIEW OF VEHICLES
    2   PERSONNEL-ID
END-DEFINE
    * поиск данных:
FIND   EMPLOY-VIEW WITH NAME  =  'ADKINSON'
     AND   COUPLED   TO VEHIC-VIEW
              VIA   PERSONNEL-ID = PERSONNEL-ID
              WITH MAKE   =   'VOLVO'
DISPLAY   PERSONNEL-ID NAME   FIRST-NAME
END-FIND
    * конец программы:
END
```

STARTING WITH ISN=операнд 5

Замечание. Это предложение применимо только для баз данных ADABAS и VSAM; для VSAM допустимо только для ESDS.

Данное предложение предназначено для определения значения, с которого начинается отбор записей – операнд 5. В операнде5 задается внутренний порядковый номер (ISN) ADABAS или относительный байт адреса (RBA) VSAM соответственно.

Это предложение может использоваться для повторного запуска цикла FIND, обработка которого была прервана, для того чтобы определить следующую запись, с которой обработка будет продолжена. Это особенно полезно тогда, когда следующая запись не может быть идентифицирована уникально каким-либо значением ее дескриптора. Это также полезно в распределенном приложение клиент/сервер, где чтение записей выполняется серверной программой, в то время как дальнейшая обработка записей выполняется клиентской программой, а записи обрабатываются не все сразу, а порциями (группами).

Замечание. Фактически начальным значением будет не значение, указанное в операнде5, а следующее за ним значение.

предложение **SORTED BY**

Замечание. Это предложение применимо только для баз данных ADABAS и SQL. Предложение не разрешается в среде ENTIRE SYSTEM SERVER.

Синтаксис предложения показан на рис. 31.

SORTED [BY] *дескриптор…***3 [DESCENDING]**

Рис. 31. Предложение SORTED BY

Предложение SORTED BY используется для сортировки отобранных записей по одному, двум или трем дескрипторам. Дескрипторы, по которым сортируются записи, могут отличаться от дескрипторов, по которым осуществляется их отбор.

По умолчанию записи сортируются по возрастанию значения дескриптора; для сортировки записей по убыванию необходимо определить ключевое слово DESCENDING. Сортировка выполняется с помощью инвертированных списков ADABAS и не приводит к чтению записей.

Замечание. Использование этого предложения может привести к уменьшению производительности, если дескриптор, используемый для

управления сортировкой, содержит большое количество значений. Это связано с тем, что для определения каждой выбранной записи в последовательности сортировки необходимо просматривать весь список значений дескриптора. Для сортировки большого количества записей желательно использовать оператор SORT.

Дескриптор, входящий в состав периодической группы, не может быть определен в данном предложении. Множественное поле (без индекса) может быть определено в данном предложении.

В предложении SORTED BY также могут быть определены не дескрипторы, за исключением системы OpenVMS и мэйнфреймов.

Предложения SORTED BY и RETAIN не могут использоваться вместе.

Пример использования предложения SORTED BY:

```
    * определение области данных:
DEFINE DATA LOCAL
1       EMPLOY-VIEW VIEW OF EMPLOYEES
   2    CITY
   2    NAME
   2    FIRST-NAME
   2            PERSONNEL-ID
END-DEFINE
    * поиск данных с использованием предложения SORTED BY:
LIMIT   10
FIND EMPLOY-VIEW WITH CITY = 'FRANKFURT'
        SORTED BY NAME PERSONNEL-ID
    * вывод результатов на экран монитора:
DISPLAY NOTITLE NAME  (IS=ON) FIRST-NAME   PERSONNEL-ID
END-FIND
    * конец программы
END
```

***Предложение* RETAIN**

Замечание. Это предложение применимо только для баз данных ADABAS. Предложение не разрешается с ENTIRE SYSTEM SERVER.

Синтаксис предложения показан на рис. 32. и в таблице 5.

RETAIN AS *операнд 6*

Рис. 32. Синтаксис предложения RETAIN

Описание операндов предложения RETAIN

Таблица 5

Операнд	Возможная структура					Возможные форматы											Разрешение ссылки	Динамическое определение
Операнд 6	C	S				A											да	нет

Предложение RETAIN дает возможность сохранить результаты поиска в больших файлах для последующей обработки. Результаты поиска запоминаются как "набор ISN" в рабочем наборе ADABAS. Этот набор используется в последующих операторах FIND в качестве базового критерия поиска для дальнейшей детализации (уточнения) набора или для обработки записей. Созданный набор связан с файлом (особенность файла) и может использоваться в другом операторе FIND только в том случае, когда обрабатывается тот же самый файл. К набору может обращаться любая программа Natural.

Имя набора – операнд 6

Имя набора используется для идентификации набора записей. Имя может быть определено как алфавитно-цифровая константа или как содержимое алфавитно-цифровой пользовательской переменной. Дубликат имени не проверяется; следовательно, если будет задано дублирующее имя набора, новый набор заменяет старый набор.

Освобождение наборов

Для количества сохраняемых наборов или количества ISN в наборе не существует определенных ограничений. Рекомендуется поддерживать минимальное количество наборов ISN, определенных в одно время. Наборы, которые больше не нужны, рекомендуется удалять оператором RELEASE SETS.

Сохраненные наборы, если они не удалены оператором RELEASE, существуют до конца сеанса Natural или до перехода в другую библиотеку, когда они автоматически удаляются. К набору, созданному одной программой, может обращаться другая программа для обработки или дальнейшей детализации с помощью дополнительных критериев поиска.

Модификации другими пользователями

Записи, ISN которых сохранены в наборе, не защищены от доступа и/или модификации другими пользователями. Следовательно, перед обработкой записей из набора полезно проверить соответствие набора исходному

критерию поиска, который использовался при создании набора. Это выполняется другим оператором FIND, использующим имя набора в предложении WITH в качестве базового критерия поиска и задающим исходный критерий поиска в предложении WHERE (то есть тот базовый критерий поиска, который был использован при создании набора в предложении WITH оператора FIND).

Ограничение

Предложения RETAIN и SORTED BY не могут использоваться вместе.

Пример предложения RETAIN:

```
    * определение области данных:
DEFINE DATA LOCAL
1 EMPLOY-VIBW VIEW OF EMPLOYEES
    2 CITY
    2 BIRTH
    2 NAME
1 #BIRTH (D)
END-DEFINE
    * пересылка значения '19400101' значению переменной #BIRTH:
MOVE EDITED '19400101' TO #BIRTH (EM =YYYYMMDD)
    * поиск данных по пересланному значению и сохранение результатов в
файл 'AGESET1', остановка работы программы в случае, если ни одного
значения не найдено:
FIND NUMBER EMPLOY-VIEW WITH BIRTH GT #BIRTH
            RETAIN AS 'AGESET1'
IF *NUMBER = 0
STOP
END-IF
    * новый поиск данных с учетом значений, хранящихся в файле
'AGESET1', вывод результатов на экран монитора:
FIND EMPLOY-VIEW WITH 'AGESET1' AND CITY = 'NEW YORK'
DISPLAY NOTITLE NAME CITY BIRTH (EM = YYYY-MM-DD)
END-FIND
    * освобождение набора 'AGESET1':
RELEASE SET 'AGESET1'
    * конец программы:
END
```

***Предложение* WHERE**

Предложение WHERE используется для определения дополнительного критерия отбора, который проверяется после чтения записи (выбранной в

соответствие с критерием предложения WITH) и перед ее обработкой (в том числе и перед проверкой ситуации AT BREAK).

Синтаксис предложения WHERE показан на рис. 33.

WHERE *логическое условие*

Рис. 33. Синтаксис предложения WHERE

Если в операторе FIND, содержащем предложение WHERE, определено ограничение обработки, записи, отвергнутые в соответствие с критерием предложения WHERE, не учитываются при проверке этого ограничения. Однако, при проверке любого глобального ограничения, заданного в параметре сеанса Natural, команде GLOBALS или в операторе LIMIT, эти записи учитываются.

Пример предложения WHERE:

```
    * определение области данных:
DEFINE DATA LOCAL
1 EMPLOY-VIEW VIEW OF EMPLOYEES
  2 PERSONNEL-ID
  2 NAME
  2 JOB-TITLE
  2 CITY
END-DEFINE
    * поиск данных с использованием предложения WHERE:
FIND EMPLOY-VIEW WITH CITY = 'PARIS'
            WHERE JOB-TITLE = 'INGENIEUR COMMERCIAL'
    * вывод результатов:
DISPLAY NOTITLE CITY JOB-TITLE PERSONNEL-ID NAME
END-FIND
    * конец программы
END
```

***Предложение* IF NO RECORDS FOUND**

Синтаксис предложения показан на рис. 34. и рис. 35.

IF NO [RECORDS] [FOUND]

END-NOREC

Рис. 34. Синтаксис предложения IF NO RECORDS FOUND
(структурный режим)

Предложение IF NO RECORDS FOUND используется для инициализации цикла обработки в том случае, когда никакие записи не удовлетворяют критериям отбора, указанным в предложениях WITH и WHERE.

IF NO [RECORDS] [FOUND]

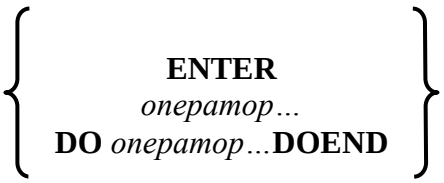


Рис. 35. Синтаксис предложения IF NO RECORDS FOUND
(режим отчета)

Если никакие записи не удовлетворяют критериям отбора, указанным в предложениях WITH и WHERE, предложение IF NO RECORDS FOUND вызывает однократное выполнение цикла обработки с "пустой" записью. Если это не желательно, необходимо задать оператор ESCAPE BOTTOM в предложение IF NO RECORDS FOUND.

Если в предложение IF NO RECORDS FOUND заданы один или несколько операторов, они будут выполняться непосредственно перед началом цикла обработки. Если перед началом цикла не должны выполняться никакие операторы, необходимо использовать ключевое слово ENTER.

Значения полей базы данных

Если в операторах, входящих в состав предложения IF NO RECORDS FOUND, не присвоены другие значения, то всем полям файла, определенного

в текущем цикле, к которым производится обращение, Natural присваивает нулевые значения.

Вычисление системных функций

Системные функции вычисляются один раз для пустой записи, которая создается для обработки в результате предложения IF NO RECORDS FOUND.

Ограничение

Это предложение не используется с FIND FIRST, FIND NUMBER и FIND UNIQUE.

Пример предложения IF NO RECORDS FOUND:

```
    * определение области данных:
DEFINE DATA LOCAL
1 EMPLOY-VIEW VIEW OF EMPLOYEES
    2 PERSONNEL-ID
    2 NAME
    2 FIRST-NAME
1 VEHIC-VIEW VIEW OF VEHICLES
    2 PERSONNEL-ID
    2 MAKE
END-DEFINE
    * чтение первых 15 записей из файла 'EMPLOYEES', начиная с фамилии
'JONES':
LIMIT 15
EMP. READ EMPLOY-VIEW BY NAME STARTING FROM 'JONES'
    * поиск записей в файле 'VEHICLES' на основе индивидуальных номеров
сотрудников из списка полученного при помощи чтения записей в файле
'EMPLOYEES'; если в файле 'VEHICLES' запись с соответствующим
номером не найдена – пересылка значения '*** NO CAR ***' в поле 'MAKE':
VEH.  FIND VEHIC-VIEW WITH PERSONNEL-ID = PERSONNEL-ID (EMP.)
IF NO RECORDS FOUND
MOVE '*** NO CAR ***' TO MAKE
END-NOREC
    * вывод результатов поиска на экран (результатом является список 15
сотрудников с указанием фирм-производителей автомобилей сотрудников;
если информация о производителе автомобиля сотрудника отсутствует,
запись о сотруднике выводится на экран со статусом '*** NO CAR ***'):
DISPLAY NOTITLE NAME (EMP.)  (IS=ON) FIRST-NAME (EMP.)  (IS=ON)
MAKE (VEH.)
    * завершение работы операторов чтения и поиска и конец программы:
```

```
END-FIND
END-READ
END
```

***Системные переменные в операторе* FIND**

Для каждого выполняемого оператора FIND автоматически создаются системные переменные *ISN, *NUMBER и *COUNTER. Если системная переменная находится за пределами текущего цикла обработки или относится к операторам FIND UNIQUE, FIND FIRST или FIND NUMBER, должен быть указан номер оператора. Формат/длина каждой из системных переменных - P10; этот формат/длина не может быть изменен.

***ISN**

Для баз данных ADABAS системная переменная *ISN содержит внутренний последовательный номер записи (ISN), обрабатываемой в настоящее время. Для оператора FIND NUMBER переменная *ISN не доступна.

Для баз данных VSAM системная переменная *ISN содержит относительный байтовый адрес (RBA) записи, обрабатываемой в настоящее время (только для файлов ESDS).

Для баз данных DL/I и SQL и в среде ENTIRE SYSTEM SERVER переменная *ISN не доступна.

***NUMBER**

Содержит число записей, удовлетворяющих базовому критерию поиска, указанному в предложении WITH.

Для баз данных DL/I системная переменная *NUMBER содержит "0", если ни одна реализация сегмента не удовлетворяет критерию поиска, и значение "9999", если, по крайней мере, одна реализация сегмента удовлетворяет критерию поиска.

Для баз данных VSAM системная переменная *NUMBER содержит значащее значение только в том случае, когда в критерии поиска используется операции EQUAL TO. С любой другой операцией переменная *NUMBER будет равна "О" если никакие записи не были найдены; любой другое значение указывает, что записи были найдены, но данное значение не имеет никакого отношения к количеству фактически найденных записей.

В среде ENTIRE SYSTEM SERVER системная переменная *NUMBER не доступна.

***COUNTER**

Содержит число вхождений в цикл обработки.

Пример использования системных переменных:

```
*   определение области данных:
DEFINE   DATA   LOCAL
1   EMPLOY-VIEW VIEW   OF   EMPLOYEES
    2    PERSONNEL-ID
    2   NAME
    2   CITY
END-DEFINE
    * поиск записей о сотрудниках, находящихся в городе Мадрид, в файле
'EMPLOYEES':
LIMIT 3
FIND   EMPLOY-VIBW WITH CITY = 'MADRID'
    * вывод результатов поиска на экран монитора:
DISPLAY NOTITLE   PERSONNEL-ID   NAME
      *ISN  *NUMBER  *COUNTER
    * завершение работы оператора поиска и конец программы:
END-FIND
END
```

***Использование нескольких операторов* FIND**

Использование нескольких операторов FIND создает вложенность циклов, посредством чего для каждой выбранной во внешнем цикле записи выполняется внутренний цикл.

Пример использования нескольких операторов FIND:

В приведенном ниже примере сначала из файла EMPLOYEES выбираются записи с фамилией SMITH. Затем значения поля PERSONNEL-ID из файла EMPLOYEES используются как ключ поиска для доступа к файлу VEHICLES. Результирующий отчет содержит поля NAME и FIRST-NAME (полученные из файла EMPLOYEES) всех сотрудников с фамилией SMITH, а также модели автомобилей MAKE (полученные из файла VEHICLES), принадлежащих этим сотрудникам:

```
    * определение области данных:
DEFINE DATA LOCAL
1 EMPLOY-VIEW VIEW OF EMPLOYEES
    2 PERSONNEL-ID
    2 NAME
    2 FIRST-NAME
```

```
1 VEHIC-VIEW VIEW OF VEHICLES
    2 PERSONNEL-ID
    2 MAKE
END-DEFINE
```

** поиск в файле 'EMPLOYEES' записей сотрудников с фамилией 'SMITH', и далее – поиск в файле 'VEHICLES' записей сотрудников с индивидуальными номерами, соответствующими индивидуальным номерам записей сотрудников, найденных в файле 'EMPLOYEES':*

```
LIMIT 15
EMP. FIND EMPLOY-VIEW WITH NAME = 'SMITH'
VEH. FIND VEHIC-VIEW WITH PERSONNEL-ID = EMP.PERSONNEL-ID
```

** пересылка значения '*** NO CAR ***' в поле фирмы-производителя автомобиля сотрудника, если в файле 'VEHICLES' не найдена запись с индивидуальным номером, соответствующим индивидуальному номеру сотрудника, найденного в файле "EMPLOYEES':*

```
IF NO RECORDS FOUND
MOVE '* * * NO CAR * * *' TO MAKE
END-NOREC
```

** вывод результатов поиска на экран монитора:*

```
DISPLAY NOTITLE EMP.NAME (IS=ON)
                EMP.FIRST-NAME (IS=ON)
                VEH.MAKE
```

** завершение работы операторов поиска:*

```
END-FIND
END-FIND
END
```

FIND FIRST

Оператор FIND FIRST используется для выбора и обработки первой записи, которая удовлетворяет критериям WITH и WHERE.

Для баз данных ADABAS обрабатываемая запись будет записью из выбранного набора, имеющей самый меньший ISN.

Этот оператор не инициирует цикл обработки.

Ограничения. FIND FIRST используется только в режиме отчета.

FIND FIRST не доступно для баз данных DL/I и SQL.

Предложение IF NO RECORDS FOUND не может использоваться с оператором FIND FIRST.

Системные переменные с оператором **FIND FIRST**

С оператором FIND FIRST доступны следующие системные переменные Natural:

***ISN**

Содержит ISN выбранной записи. Если никакая запись не удовлетворяет критериям поиска WITH и WHERE, *ISN будет равен "0".

Переменная *ISN не доступна для баз данных VSAM или в среде ENTIRE SYSTEM SERVER.

***NUMBER**

Содержит число записей, найденных после проверки критерия WITH, но перед проверкой критерия WHERE. Если ни одна запись не удовлетворяет критерию WITH, *NUMBER будет равна "0".

Переменная *NUMBER не доступна в среде ENTIRE SYSTEM SERVER .

***COUNTER**

Содержит "1", если запись была найдена; содержит "0", если никакая запись не найдена.

FIND NUMBER

Оператор FIND NUMBER используется для определения количества записей, удовлетворяющих критериям WITH/WHERE. Данный оператор не инициирует цикла обработки и ему не доступны поля базы данных.

Замечание. Использование предложения WHERE может привести к уменьшению производительности.

Ограничения. В операторе FIND NUMBER нельзя использовать предложения SORTED BY и IF NO RECORDS FOUND.

Предложение WHERE нельзя использовать в структурном режиме.

Оператор FIND NUMBER не доступен для баз данных DL/I.

Оператор FIND NUMBER не доступен в среде ENTIRE SYSTEM SERVER.

Системные переменные с оператором **FIND NUMBER**

С оператором FIND NUMBER доступны следующие системные переменные Natural:

***NUMBER**

Содержит число записей, удовлетворяющих критерию поиска WITH.

***COUNTER**

Содержит число записей, найденных после проверки критерия WHERE.

Системная переменная *COUNTER доступна только в том случае, когда оператор FIND NUMBER содержит предложение WHERE.

Пример использования оператора FIND NUMBER:

```
*   описание области данных:
DEFINE DATA   LOCAL
1   EMPLOY-VIEW VIEW OF   EMPLOYEES
    2    CITY
    2   BIRTH
1   #BIRTH   (D)
END-DEFINE
    * пересылка значения ‘19500101’ значению переменной #BIRTH:
MOVE   EDITED   '19500101'   TO   #BIRTH    (EM=YYYYMMDD)
    * поиск количества сотрудников в городе Мадрид, дата рождения
которых до 01 января 1950 года:
FIND NUMBER EMPLOY-VIEW WITH   CITY   =   'MADRID'
        WHERE BIRTH   LT   #BIRTH
    * вывод результатов на экран монитора:
WRITE NOTITLE   'TOTAL   RECORDS   SELECTED:                    '
*NUMBER
                  /   'TOTAL   BORN   BEFORE   1   JAN   1950:      '
              *COUNTER
    * конец программы:
END
```

FIND UNIQUE

Оператор FIND UNIQUE используется для проверки того, что только одна запись удовлетворяет критерию поиска. Данный оператор •• инициирует цикла обработки. Если определено предложение WHERE, то для его обработки автоматически создается внутренний цикл.

Если ни одна запись не удовлетворяет заданному критерию или несколько записей удовлетворяют данному критерию, выдается сообщение об ошибке. Эта ситуация может проверяться в операторе ON ERROR.

Ограничения. FIND UNIQUE используется только в режиме отчета.

FIND UNIQUE не доступно для баз данных DL/I или в среде Entire System Server.

FIND UNIQUE нельзя использовать для баз данных SQL. (Исключение: на mainframe предложение FIND UNIQUE может использоваться для главных

(primary) ключей; однако, это разрешается только для совместимости и не должно использоваться.)

В операторе FIND UNIQUE нельзя использовать предложения SORTED BY и IF NO RECORDS FOUND.

9.3. Организация выдачи данных на экран и печати

Синтаксис оператора DISPLAY показан на рис. 36, рис. 37, рис. 38, рис. 39 и в таблице 6.

DISPLAY [(*rep***)] [***режимы***] {[/…] [***формат-вывода***] *элемент-вывода*}…**

Рис. 36. Синтаксис оператора DISPLAY

[NOTITLE] [NOHDR] [[AND] [GIVE] [SYSTEM] FUNCTIONS] [(*параметры***)]**

Рис. 37. Режимы оператора DISPLAY

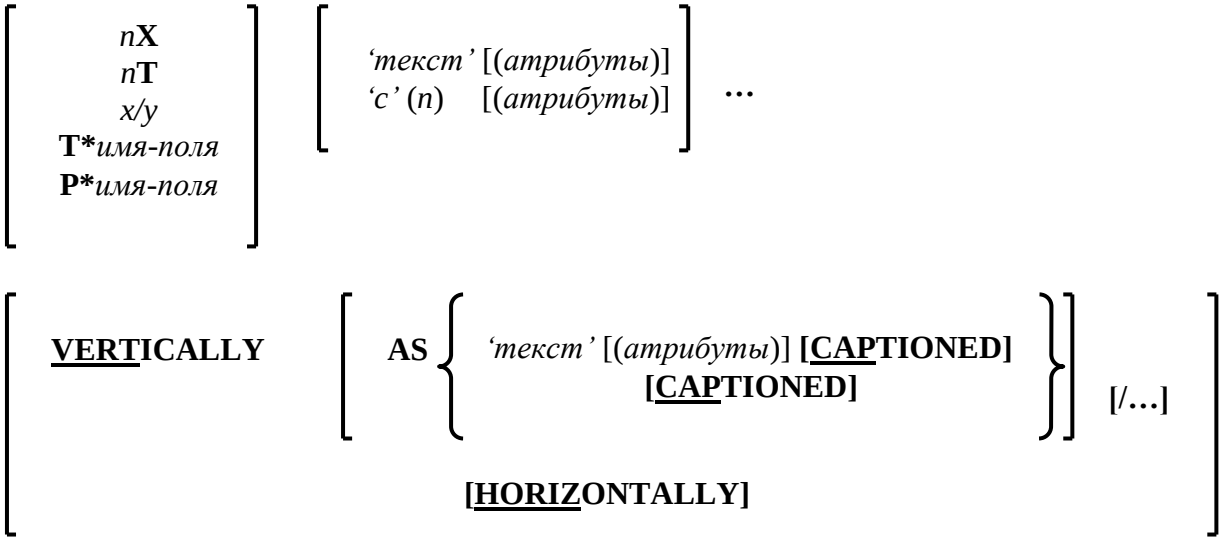


Рис. 38. Формат ввода оператора DISPLAY

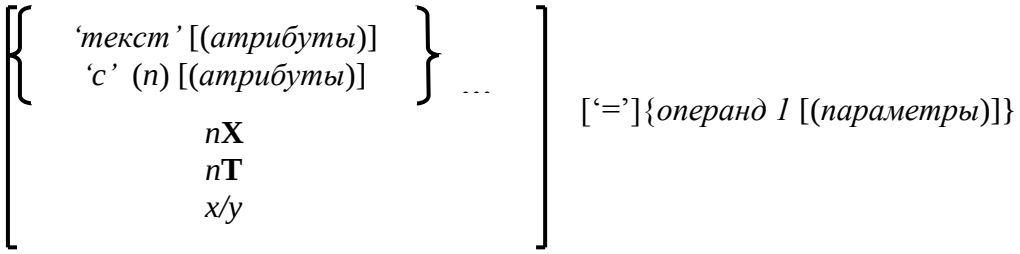


Рис. 39. Элемент ввода оператора DISPLAY

Описание операндов элемента вывода оператора DISPLAY

Таблица 6

Операнд	Возможная структура					Возможные форматы											Разрешение ссылки	Динамическое определение
Операнд 1		S	A	G	N	A	N	P	I	F	B	D	T	L		G	O	да	нет

***Функция оператора* DISPLAY**

Оператор DISPLAY используется для определения полей, подлежащих выводу в отчет в форме столбцов. Столбец создается для каждого поля, и над столбцом помещается заголовок поля.

***Идентификатор отче*та - rep**

Обозначение (rep) может использоваться для указания идентификатора отчета, формируемого оператором DISPLAY. В качестве идентификатора могут быть заданы либо номер от 0 до 31, либо логическое имя, присвоенное с помощью оператора DEFINE PRINTER. Если обозначение (rep) не задано, оператор DISPLAY относится к первому отчету.

***Заголовок страницы*/NOTITLE**

Для каждой страницы отчета, созданной оператором DISPLAY, система Natural по умолчанию генерирует одну строку заголовка. Это заголовок содержит номер страницы, время дня и дату. Время дня устанавливается в начале выполнения программы (режим TP) или в момент старта задания (пакетный режим).

Строка заголовка, сформированная по умолчанию, может быть изменена оператором WRITE TITLE или подавлена ключевым словом NOTITLE в операторе DISPLAY.

Формируется заголовок по умолчанию:

DISPLAY NAME

Формируется заголовок пользователя:

DISPLAY NAME WRITE TITLE 'USER TITLE'

Никакой заголовок не формируется:

DISPLAY NOTITLE NAME

Если используется опция NOTITLE, ее действие распространяется на все операторы DISPLAY, PRINT и WRITE в пределах объекта, который формирует отчет.

***Заголовки столбцов*/NOHDR**

Для заголовков столбцов, генерируемых для каждого поля, указанного в операторе DISPLAY, действуют следующие правила:

Текст заголовка можно явно задать в операторе DISPLAY перед именем поля. Например:

DISPLAY ‘EMPLOYEE’ NAME 'SALARY' SALARY

Если текст заголовка для поля не указан явно, будет использоваться заголовок, определенный в операторе DEFINE DATA, Если для поля базы данных текст заголовка не указан в операторе DEFINE DATA, по умолчанию будет использоваться заголовок из DDM; если в DDM заголовок не определен, в качестве заголовка будет использоваться имя поля. Если текст заголовка не указан для переменной пользователя в операторе DEFINE DATA, в качестве заголовка будет использоваться имя переменной. Смотри также оператор DEFINE DATA для описания заголовка.

DISPLAY NAME SALARY #NEW-SALARY

Natural всегда подчеркивает заголовки столбца и генерирует одну пустую строку между линией подчеркивания и данными.

Если в программе имеется несколько операторов DISPLAY, то используются заголовки столбца(ов), определенные первым оператором DISPLAY; все это вычисляется во время трансляции.

Подавление заголовков столбца

Чтобы подавить заголовок для поля, перед его именем необходимо определить символы '/' (апостроф-слэш-апостроф). Например:

DISPLAY '/' NAME 'SALARY' SALARY

Чтобы подавить все заголовки столбца, необходимо определить ключевое слово NOHDR:

DISPLAY NOHDR NAME SALARY

Опция NOHDR эффективна только для первого оператора DISPLAY, поскольку все последующие операторы DISPLAY не могут создавать заголовки столбца.

Если используются и NOTITLE, и NOHDR, они должны быть определены в следующем порядке:

DISPLAY NOTITLE NOHDR MAME SALARY

Параметры

Один или несколько параметров, заключенных в круглые скобки, могут быть заданы. В таблицах 7 и 8 представлено описание параметров формата вывода и заголовков столбцов.

Каждый заданный параметр будет перекрывать соответствующий параметр, ранее определенный в команде GLOBALS или операторах SET GLOBALS или FORMAT. Если задается несколько параметров, они должны отделяться друг от друга пробелами. Описание каждого параметра не должно переходить на другую строку.

Система позиционирования поля формата вывода

Таблица 7

*n***X**	Пример: DISPLAY NAME **5X** SALARY Это обозначение используется для вставки *n* пробелов между столбцами, *n* не должен быть равно 0.
*n***T**	Это обозначение вызывает позиционирование (табуляцию) на позицию п. Обратное позиционирование не допускается. В следующем примере поле NAME выводится, начиная с позиции 25, а SALARY- начиная с позиции 50: DISPLAY **25T** NAME **50T** SALARY
x/y	Это обозначений вызывает размещение следующего элемента в строке *x* и в столбце *y*. *Y* не должен быть равен 0. Обратное позиционирование не допускается.
T * *имя-поля*	Это обозначение вызывает позиционирование на ту же колонку, где поле *'имя-поля'* было в предыдущем операторе DISPLAY. Обратное позиционирование не допускается.
P * *имя-поля*	Это обозначение вызывает позиционирование на ту же колонку и *строку поля,* где поле *'имя-поля'* было в предыдущем операторе DISPLAY. Чаще всего используется в вертикальном режиме печати. Обратное позиционирование не допускается.

Регулирование заголовка столбца формата вывода

Таблица 8

‘=’	Помещенный непосредственно перед полем знак '=' указывает на то, что в качестве заголовка используется заголовок по умолчанию, заданный для данного поля в DDM или, при его отсутствии, имя поля.

'текст'	Помещенный непосредственно перед полем текст перекрывает заголовок столбца. Символ V, помещенный перед полем, подавляет заголовок для поля. DISPLAY **'EMPLOYEE'** NAME 'MARITAL/STATUS' MAR-STAT Если перед именем поля задаются несколько текстовых элементов, последний из них будет использоваться в качестве заголовка столбца, а другие текстовые элементы будут помещены перед значением поля в пределах столбца.
'c' (n)	Непосредственно перед значением поля *n* раз высвечивается символ, заключенный в апострофы. DISPLAY '*' **(5)** '=' NAME

Атрибуты

Используется два вида атрибутов:

1. Атрибуты изображения (таблица 9).
2. Атрибуты цвета (таблица 10).

Атрибуты изображения

Таблица 9

Атрибуты	Значение	Расшифровка
B	мигание	Значение поля показывается в мигающем режиме.
C	курсив	Значение поля печатается курсивом.
D	интенсивность цвета по умолчанию	Значение поля показывается с нормальной интенсивностью (без каких-либо выделений). Нормальная интенсивность цвета является параметром, устанавливаемым по умолчанию.
I	повышенная интенсивность цвета	Значение поля показывается с повышенной интенсивностью.
N	скрытый текст	Значение поля не показывается на экране.
U	подчеркнутый	Значение поля печатается подчеркнутым.
V	инверсия	Печатается инверсионное изображение значения поля

Атрибуты цвета

Таблица 10

BL	Синий
GR	Зеленый
NE	Нейтральный

PI	Розовый
RE	Красный
TU	Бирюзовый
YE	Желтый

Вертикальный/горизонтальный режимы отображения

Предложение VERT применяется для отображения нескольких значений поля друг под другом в одном и том же столбце. В вертикальном режиме новый столбец описывается путем задания ключевых слов VERT или HORIZ.

В вертикальном режиме заголовком столбца управляют с помощью предложения AS, как описано ниже.

–Заголовок столбца не генерируется, если предложение AS не используется.

DISPLAY **VERT** NAME SALARY

–Если задано AS 'текст', то текст, указанный в апострофах, используется в качестве заголовка столбца. Символ "/" в тексте заголовка вызовет печать заголовка столбца в несколько строк.

DISPLAY **VERT AS 'LAST/NAME'** NAME

–Если задано AS 'текст' CAPTIONED, то текст, указанный в апострофах, используется в качестве заголовка столбца, а стандартный заголовок столбца или имя поля вставляются непосредственно перед значением поля в каждой выводимой строке.

DISPLAY **VERT AS 'PERSONS/SELECTED' CAPTIONED** NAME FIRST-NAME

–Если задано AS CAPTIONED, в качестве заголовка столбца будет использоваться стандартный заголовок для поля (текст заголовка или имя поля).

DISPLAY **VERT AS CAPTIONED** NAME FIRST-NAME

Вертикальная и горизонтальная ориентация столбцов может смешиваться, используя соответствующее ключевое слово.

Чтобы подавить вертикальную печать для какого-либо одного элемента, можно перед этим элементом поместить символ "-". Например:

DISPLAY VERT NAME - FIRST-NAME SALARY

В приведенном выше примере поле FIRST-NAME будет выведено горизонтально следом за полем NAME, в то время как поле SALARY будет выведено вертикально, то есть ниже NAME.

Стандартным режимом отображения является горизонтальный режим. Для каждого выводимого поля строится столбец.

Заголовки столбца получаются и используются системой Natural в соответствие со следующими приоритетами:

– заголовок 'текст', указанный в операторе DISPLAY;

– заголовок по умолчанию, определенный в DDM (поля базы данных), или имя пользовательской переменной;

– имя поля как определено в DDM (если для поля базы данных не задан заголовок по умолчанию).

Для имен групп заголовок формируется для всей группы. При работе с группой заданный пользователем заголовок перекрывает полностью заголовок для всей группы.

Максимальное число строк заголовка столбца - 15.

Для оператора DISPLAY не допускается переполнение выходной строки, в этом случае выдается сообщение об ошибке.

Пример использования оператора DISPLAY

```
    * чтение первых 4-х записей из файла 'EMPLOYEES':
LIMIT   4
READ EMPLOYEES BY NAME
    * вывод результатов чтения на экран
DISPLAY NOTITLE 5X NAME 50T JOB-TITLE
    * завершение работы оператора чтения и конец программы
END-READ
END
    * результаты выводятся на экран монитора в следующем виде:

NAME                                CURRENT
                                    POSITION
- - - - - - - - - - - - - - -      - - - - - - - - - - - - - - - - - - - -
ABELLAN                             MAQUINISTA
ACHIESON                            DATA BASE ADMINISTRATOR
ADAM                                CHEF DE SERVICE
ADKINSON                            SALES PERSON
```

Вывод отчетов на печать осуществляется с помощью оператора DEFINE PRINTER. Синтаксис оператора DEFINE PRINTER показан на рис. 40.

Оператор DEFINE PRINTER используется для назначения символического имени номеру отчета и управления размещением отчета на логическом устройстве. Это обеспечивает дополнительную гибкость системы при выводе отчетов на различные логические устройства. Если во время выполнения оператора принтер открыт, оператор неявно закроет данный принтер.

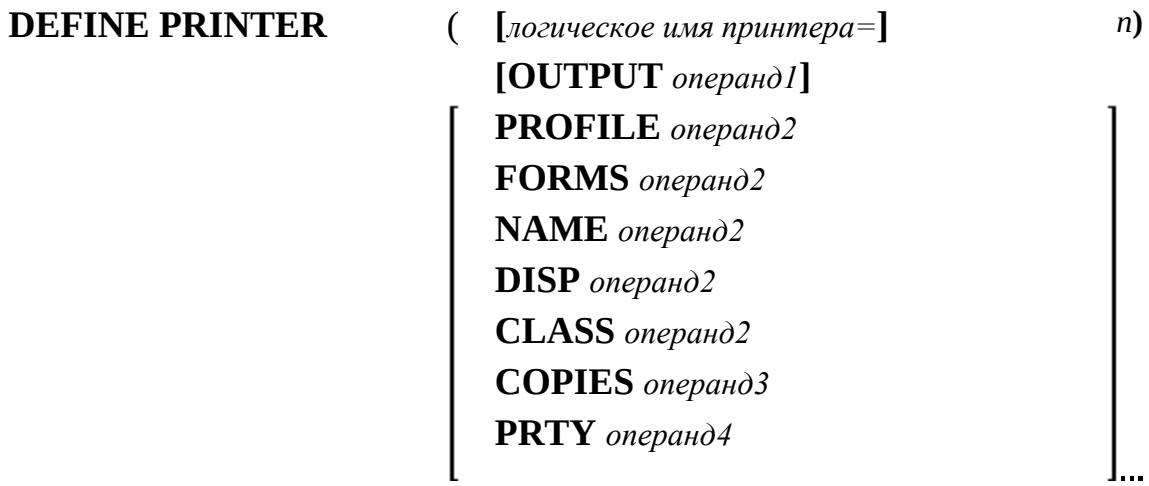


Рис. 40. Синтаксис оператора DEFINE PRINTER

Логическое имя принтера - имя, назначаемое принтеру с номером n (номер принтера имеет значение от 1 до 31).

OUTPUT *операнд1* - определяет пункт назначения в системе. Например ‘*LPTnn*’ для Win, Unix; ‘*CMPRT01*’ для DDNAME OS/390

FORMS/NAME/DISP/CLASS/COPIES/PRTY - опции для управления печатью, обрабатываемые ОС (используются только для мэйнфрейм).

Закрытие принтера осуществляется при помощи оператора CLOSE PRINTER. Синтаксис оператора CLOSE PRINTER показан на рис. 41. Принтер закрывается также при повторном использовании оператора DEFINE PRINTER и при завершении программы.

CLOSE PRINTER *(логическое имя)*

Рис. 41. Синтаксис оператора CLOSE PRINTER

Пример использования операторов DEFINE PRINTER и CLOSE PRINTER

```
    * определение принтера для печати
DEFINE PRINTER (1) OUTPUT 'TID100'
    * печать
WRITE (1) 'PRINTED ON PRINTER TID100'
    * закрытие принтера и конец программы
CLOSE PRINTER (1)
END
```

9.4. Операторы доступа и манипулирования данными

Перечень операторов доступа к базе данных и манипулирования данными представлен в таблице 11.

Операторы доступа и манипулирования данными

Таблица 11

READ	Считывает записи файла БД в физической или логической последовательности
HISTOGRAM	Считывает значения поля БД
GET	Читает запись с заданным ISN (последовательным внутренним номером записи в файле)
STORE	Добавляет новую запись в файл БД
UPDATE	Обновляет запись в файле БД
DELETE	Удаляет запись из файла БД
END TRANSACTION	Конец логической транзакции
BACKOUT TRANSACTION	Откат на предыдущую транзакцию
GET TRANSACTION DATA	Чтение данных транзакции, сохраненных предыдущим оператором END TRANSACTION
ACCEPT/REJECT	Принимает / отвергает запись по указанному пользователем критерию
LIMIT	Ограничивает цикл обработки операторов READ, FIND и HISTOGRAM
GET SAME	Повторно читает запись обрабатываемый в настоящее время
PASSW	Задает пароль доступа пользователя к защищенному файлу
RETRY	Повторная попытка чтения записи, задержанной другим пользователем
AT START OF DATA	Выполняет заданные операторы после чтения первой записи в цикле обработки
AT END OF DATA	Выполняет заданные операторы после чтения последней записи в цикле обработки

AT BREAK	Выполняет заданные операторы при изменении значения поля (обработка прерывания)
BEFORE BREAK PROCESSING	Выполняет заданные операторы перед обработкой прерывания
PERFORM BREAK PROCESSING	Выполняет обработку прерывания

***Синтаксис оператора* READ**

Синтаксис оператора READ показан на рис. 42.

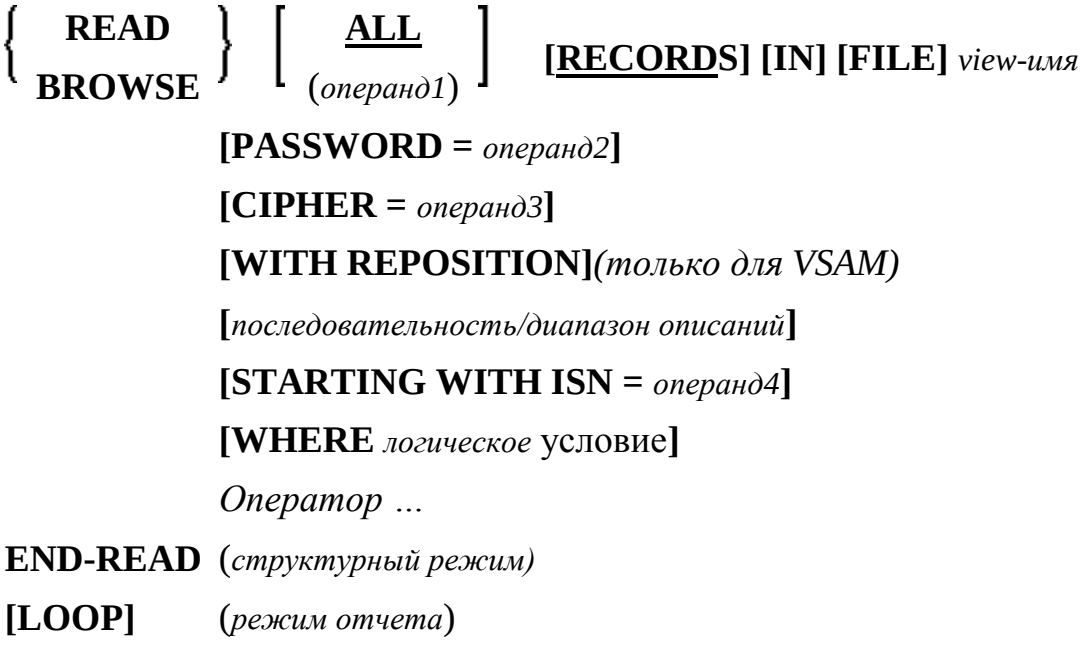


Рис. 42. Синтаксис оператора READ

Последовательность/диапазон описаний для чтения записей в физической последовательности (в которой хранятся в БД) показан на рис. 43.

Рис. 43. Последовательность/диапазон для чтения в физической последовательности

Последовательность/диапазон описаний для чтения записей в порядке возрастания ISN показан на рис. 44.

Рис. 44. Последовательность/диапазон для чтения
в порядке возрастания ISN

Последовательность/диапазон описаний для чтения записей в последовательности значений дескриптора показан на рис. 45.

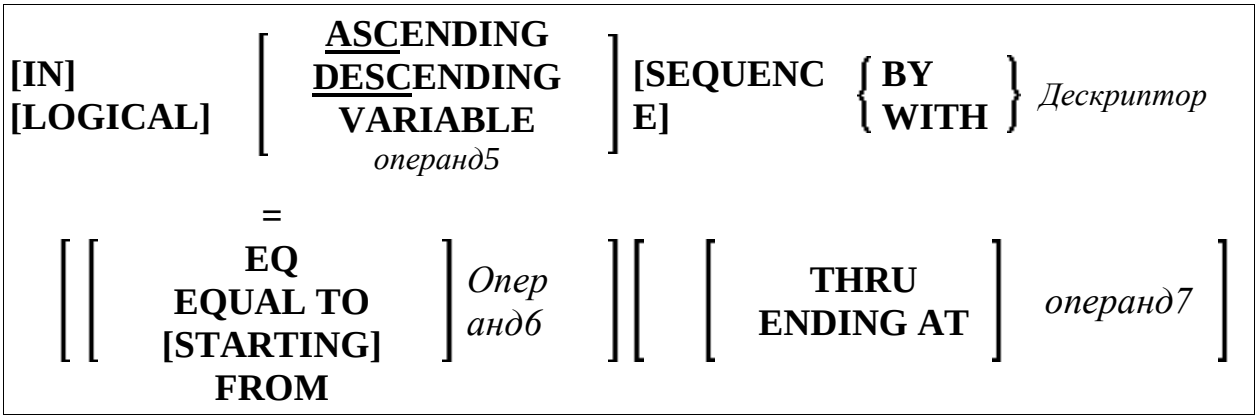


Рис. 45. Последовательность/диапазон для чтения в последовательности
значений дескриптора

STARTING WITH ISN = *операнд4* - определение значения ISN, с которого начинается отбор записей

WHERE *логическое условие* - дополнительный критерий отбора, проверяется после чтения записи и перед началом обработки

Пример программ с оператором READ

** определение локальной области данных*

```
DEFINE DATA LOCAL
1 EMPLOY-VIEW VIEW OF EMPLOYEES
  2 PERSONNEL-ID
  2 NAME
END-DEFINE
```

** чтение первых трех записей в физической последовательности и вывод результатов на экран*

```
LIMIT 3
```

```
WRITE 'READ IN PHYSICAL SEQUENCE'
READ EMPLOY-VIEW IN PHYSICAL SEQUENCE
DISPLAY NOTITLE PERSONNEL-ID NAME *ISN *COUNTER
END-READ
```

** чтение первых трех записей в порядке возрастания ISN и вывод результатов на экран*

```
WRITE / 'READ IN ISN SEQUENCE'
READ EMPLOY-VIEW BY ISN
STARTING FROM 1 ENDING AT 3
DISPLAY PERSONNEL-ID NAME *ISN *COUNTER
END-READ
```

** чтение первых трех записей алфавитном порядке и вывод результатов на экран*

```
WRITE / 'READ IN NAME SEQUENCE'
READ EMPLOY-VIEW BY NAME
DISPLAY PERSONNEL-ID NAME *ISN *COUNTER
END-READ
```

** чтение первых трех записей алфавитном порядке, начиная с буквы 'M' и вывод результатов на экран*

```
WRITE / 'READ IN NAME SEQUENCE STARTING FROM "M"'
READ EMPLOY-VIEW BY NAME
STARTING FROM 'M'
DISPLAY PERSONNEL-ID NAME *ISN *COUNTER
END-READ
```

** конец программы*

```
END
```

** результаты выводятся на экран монитора в следующем виде:*

```
PERSONNEL       NAME          ISN      CNT
    ID
  ---------     -------------------- --------- ---------

READ IN PHYSICAL SEQUENCE
50005600  MORENO                 2        1
50005500  BLOND                  3        2
50005300  MAIZIERE                4        3

READ IN ISN SEQUENCE
50005800  ADAM                   1        1
50005600  MORENO                 2        2
```

```
50005500  BLOND                3       3

READ IN NAME SEQUENCE
60008339  ABELLAN              479     1
30000231  ACHIESON             884     2
50005800  ADAM                 1       3

READ IN NAME SEQUENCE STARTING FROM 'M'
30008125  MACDONALD            929     1
20028700  MACKARNESS           780     2
40000045  MADSEN               509     3
```

Синтаксис оператора HISTOGRAM

Синтаксис оператора HISTOGRAM показан на рис. 46.

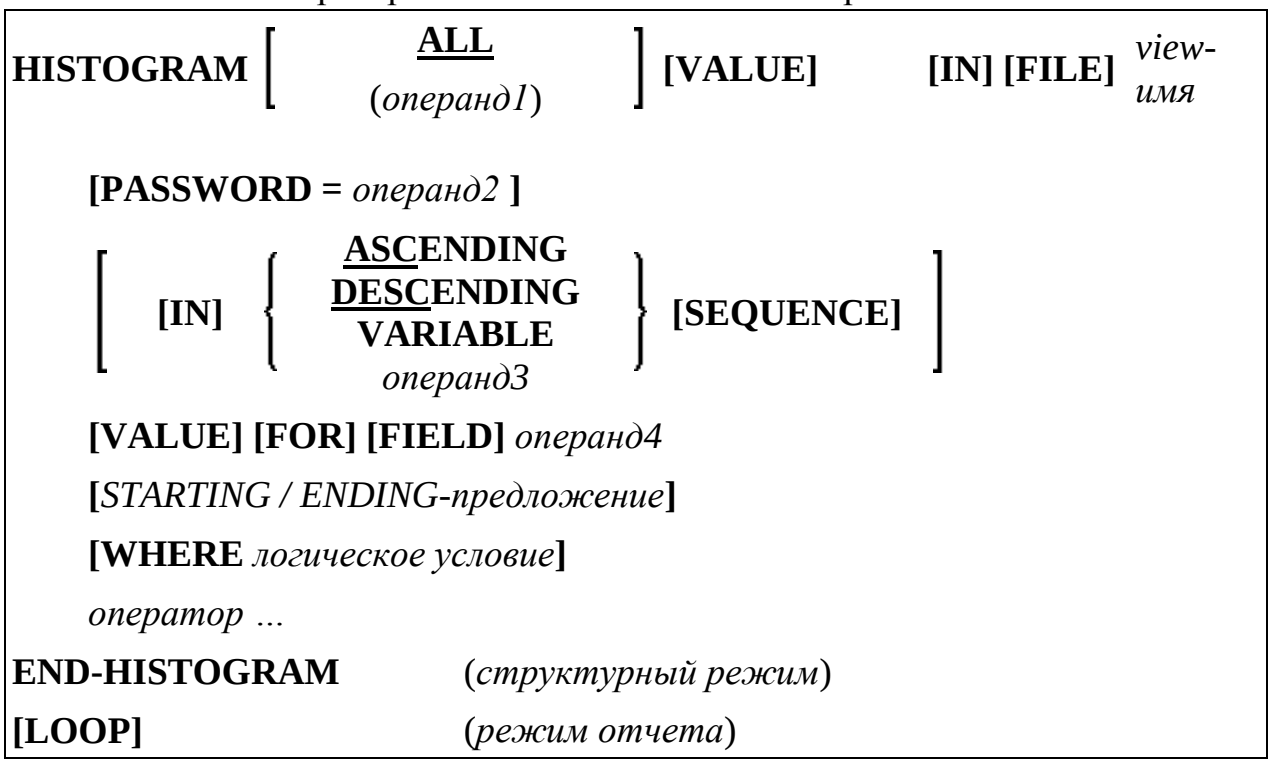


Рис. 46. Синтаксис оператора HISTOGRAM

VARIABLE *операнд3* - операнд с форматом A1 и одним из значений (A – возрастание, D – убывание).

[VALUE] [FOR] [FIELD] *операнд4* – может быть дескриптором, субдескриптором, супердескриптором или гипердескриптором.

Используя STARTING / ENDING-предложение можно определить начальное и конечное значение дескриптора. Синтаксис предложения показан на рис. 47.

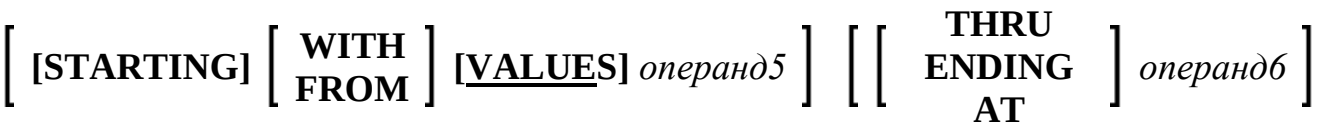


Рис. 47. Синтаксис предложения STARTING / ENDING

WHERE *логическое условие* – определение дополнительного критерия отбора проверяется после чтения записи и перед началом обработки. Дескриптор должен быть тот же, что и в операторе HISTOGRAM. Никакие другие поля из выбранного файла недоступны для обработки в этом операторе HISTOGRAM.

Пример программ с оператором HISTOGRAM

```
* определение локальной области данных
DEFINE DATA LOCAL
1 EMPLOY-VIEW VIEW OF EMPLOYEES
  2 CITY
END-DEFINE
* считывание первых восьми записей, начиная с буквы ‘M’ и вывод результатов на экран
LIMIT 8
HISTOGRAM EMPLOY-VIEW CITY STARTING FROM 'M'
  DISPLAY NOTITLE CITY
'NUMBER OF/PERSONS' *NUMBER *COUNTER
END-HISTOGRAM
* конец программы
END
```

** результаты выводятся на экран монитора в следующем виде:*

```
        CITY              NUMBER OF CNT
                            PERSONS
 -------------------- -    -------- ---------
 MADISON                        31
 MADRID                        412
 MAILLY LE CAMP                13
 MAMERS                        14
 MANSFIELD                     45
```

```
MARSEILLE          26
MATLOCK            17
MELBOURNE          28
```

Синтаксис оператора GET

Оператор используется только для ADABAS. Синтаксис оператора GET показан на рис. 48.

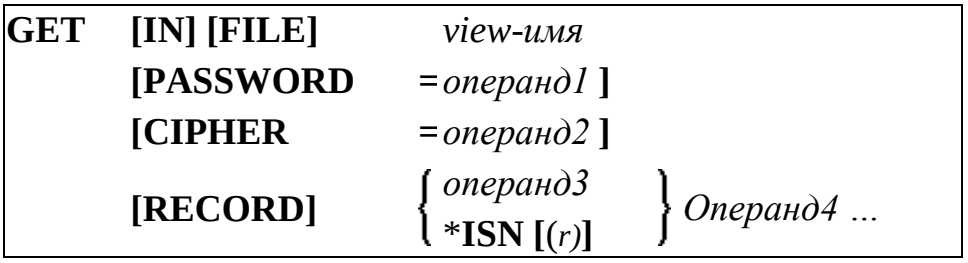


Рис. 48. Синтаксис оператора GET

ISN - должен быть представлен в виде:

– числовой константы;

– переменной в *операнд3*;

– системной переменной *ISN (R).

Операнд 4 недоступен в структурном режиме.

Пример программ с оператором GET

```
* определение локальной области данных
DEFINE DATA LOCAL
1 PERSONS VIEW OF EMPLOYEES
  2 PERSONNEL-ID
  2 NAME
  2 FIRST-NAME
1 SALARY-INFO VIEW OF EMPLOYEES
  2 NAME
  2 CURR-CODE (1:1)
  2 SALARY    (1:1)
1 #ISN-ARRAY  (B4/1:10)
1 #LINE-NR    (N2)
END-DEFINE
* чтение первых десяти записей в алфавитном порядке, запись значений
системных переменных *COUNTER и *ISN, и вывод результатов на экран
FORMAT PS=16
LIMIT 10
READ PERSONS BY NAME
```

```
  MOVE *COUNTER TO #LINE-NR
  MOVE *ISN TO #ISN-ARRAY (#LINE-NR)
  DISPLAY #LINE-NR PERSONNEL-ID NAME FIRST-NAME
```

** по завершении – запрос ввода номера строки для получения информации о заработной плате, получение записей по полученному критерию и вывод результатов на экран*

```
AT END OF PAGE
  INPUT 'PLEASE SELECT LINE-NR FOR SALARY INFORMATION:' #LINE-
NR
  IF #LINE-NR = 1 THRU 10
    GET SALARY-INFO #ISN-ARRAY (#LINE-NR)
    WRITE / SALARY-INFO.NAME
      SALARY-INFO.SALARY (1)
      SALARY-INFO.CURR-CODE (1)
  END-IF
END-ENDPAGE
```

** завершение операции чтения записей и конец программы*

```
END*ISN -READ
END
```

Доступ к файлам СУБД ADABAS

Схема доступа к файлам СУБД ADABAS показана на рис. 49.

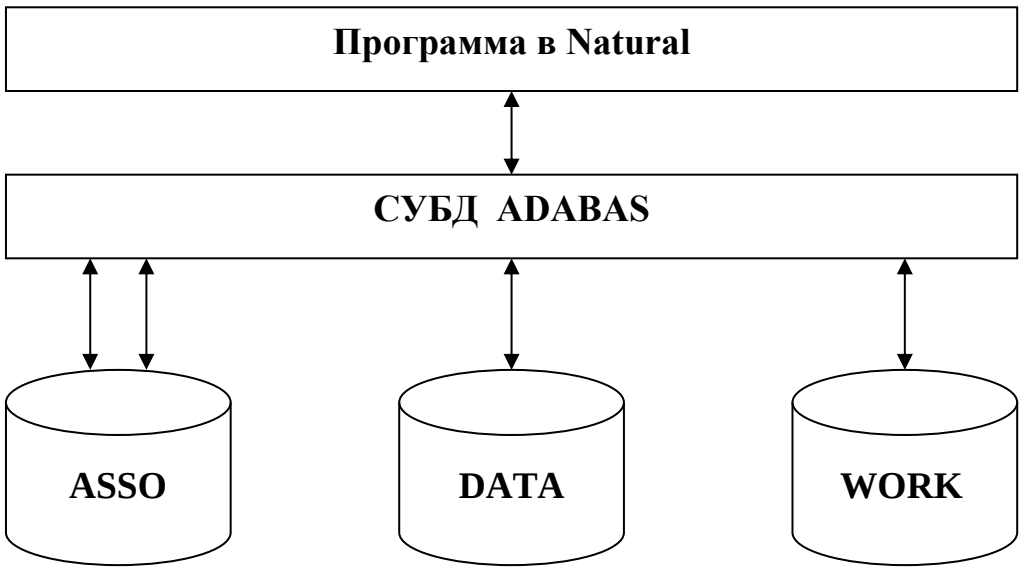


Рис. 49. Доступ к файлам СУБД ADABAS

Быстродействие осуществления операций с файлами СУБД ADABAS различными операторами чтения Natural показано в таблице 12 (ИС – инвертированный список, ПА – преобразователь адреса (NI,UI)).

Быстродействие осуществления операций

Таблица 12

Опера-торы	Назначение	ASSO	DATA	WORK	Результат / Скорость
Read … in Physical	Чтение в физической последовательности		Чтение		Данные / Самая большая
Read … by ISN	Чтение в последовательности ISN	Чтение ПА	Чтение		Данные / Медленнее чем in Physical
Read … in Logical	Чтение в логической последовательности	Чтение ИС Чтение ПА	Чтение		Данные / Медленнее чем by ISN
Histogram	Чтение данных дескриптора	Чтение ИС			Значение дескриптора с количеством значений
Find … with	Поиск по условию поиска	Чтение ИС		Запись	Список ISN
(Read by ISN)		Чтение ПА	Чтение	Чтение	Данные полей
Find … with … sorted …	Поиск по условию поиска с сортировкой	Чтение ИС Чтение ПА	Чтение	Запись	Список ISN / Медленнее, чем без доступа к DATA
(Read by ISN)		Чтение ПА	Чтение	Чтение	Данные полей
Find … with … where …	Поиск по условию поиска с доступом к данным	Чтение ИС Чтение ПА	Чтение	Запись	Список ISN / Медленнее, чем без доступа к DATA
(Read by ISN)		Чтение ПА	Чтение	Чтение	Данные полей
Get ISN	Прямой доступ	Чтение ПА	Чтение		Данные полей / Большая

Синтаксис оператора UPDATE

Синтаксис оператора UPDATE показан на рис. 50.

UPDATE [RECORD] [IN] [STATEMENT] [(*r*)]

Для отчетного режима дополнительно:

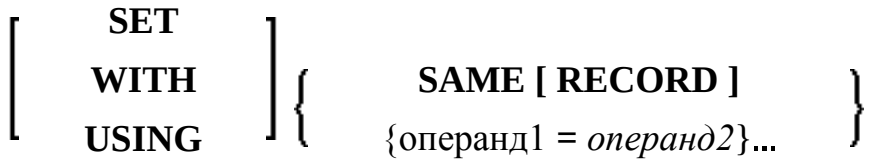


Рис. 50. Синтаксис оператора UPDATE

Модифицируемая запись должна быть прочитана операторами: READ, FIND, GET, STORE с меткой (r).

Использование оператора UPDATE ставит в состояние удержания записи, прочитанные для модификации операторами READ и FIND.

Синтаксис оператора DELETE

Синтаксис оператора DELETE показан на рис. 51.

DELETE [RECORD] [IN] [STATEMENT] [(*r*)]

Рис. 51. Синтаксис оператора DELETE

Удаляемая запись должна быть прочитана операторами: READ, FIND, GET, STORE с меткой (r).

Использование оператора DELETE ставит в *состояние удержания* записи, прочитанные для удаления операторами READ и FIND.

***Пример использования оператора UPDATE*:**

```
* определение локальной области данных
DEFINE DATA LOCAL
1 EMPLOY-VIEW VIEW OF EMPLOYEES
  2 NAME
  2 FIRST-NAME
  2 CITY
1 #NAME (A20)
END-DEFINE
* запрос ввода имени и проверка на соответствии введенного значения ненулевому значению
INPUT 'ENTER A NAME:' #NAME (AD=M)
IF #NAME = ' '
STOP
END-IF
* поиск по заданному имени, запрос ввода новых значений полного имени и внесение изменений в БД
FIND EMPLOY-VIEW WITH NAME = #NAME
```

```
IF NO RECORDS FOUND
  REINPUT WITH 'NO RECORDS FOUND'  MARK 1
  END-NOREC
INPUT 'NAME:' NAME (AD=O) /
      'FIRST NAME:' FIRST-NAME (AD=M) /
      'CITY:' CITY (AD=M)
UPDATE
END TRANSACTION
END-FIND
      * конец программы
END
```

Пример использования оператора DELETE:

```
      * определение локальной области данных
DEFINE DATA LOCAL
1 EMPLOY-VIEW VIEW OF EMPLOYEES
  2 NAME
END-DEFINE
      * поиск записей по имени 'ALDEN' и последующее их удаление
FIND EMPLOY-VIEW WITH NAME = 'ALDEN'
  DELETE
END TRANSACTION
      * вывод сообщения об успешном удалении записи на экран
AT END OF DATA
   WRITE NOTITLE *NUMBER 'RECORDS DELETED'
END-ENDDATA
      * завершение работы оператора поиска и конец программы
END-FIND
END
```

Логическая транзакция

Схема осуществления логических транзакций с различными файлами СУБД ADABAS, показана на рис. 52.

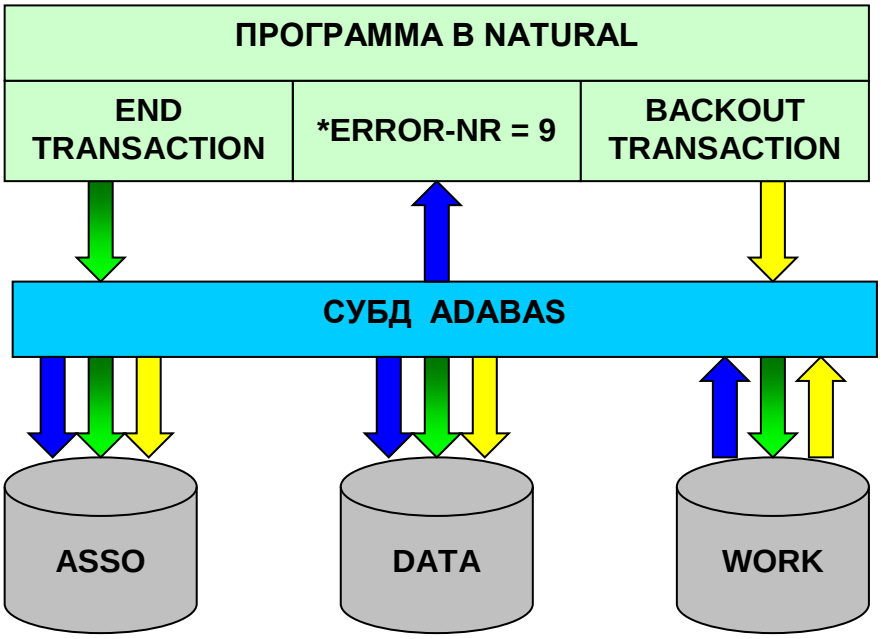


Рис. 52. Схема осуществления логической трансакции с файлами СУБД ADABAS

END [OF] TRANSACTION [*операнд1…***]**

BACKOUT [TRANSACTION]
GET TRANSACTION [DATA] [*операнд1…***]**

Автоматический откат транзакции

Транзакция - является наименьшей логической единицей работы, которая должна быть выполнена полностью для обеспечения логической целостности БД. *Логика транзакции* определяется разработчиком и обеспечивается прикладной программой. Например, изменения значения одного и того же реквизита должны быть выполнены в двух и более файлах для обеспечения *логической целостности*. Операции *модификации* выполняются сразу с БД и протоколируется в рабочей области (WORK). Прикладная программа должна подтвердить завершение транзакции с помощью команды "завершения логической транзакции".

Удержание и блокировка записей. При модификации записей пользователь должен обеспечить логическую целостность проводимых изменений. Для этого программа пользователя должна заблокировать доступ к записи других пользователей, используя опцию удержания HOLD. С этого

момента другие пользователи не могут изменить удержанную запись, а могут только прочитать не для изменения. Запись может быть изменена только тем пользователем, который ее удерживает. Запись, для которой установлен режим удержания, ADABAS блокирует, используя внутреннюю таблицу. Если другой пользователь выдаст команду чтения с опцией удержания для заблокированной записи, то ADABAS отвергнет данный запрос, выдав код возврата 145. Программа, получив такой код возврата должна ожидать освобождения записи или обработать как ошибку.

Операторы обработки транзакций

END [OF] TRANSACTION [*операнд1…***] -** указывает конец логической транзакции, сохраняет все изменения в БД и приводит к освобождению всех записей, удержанных во время транзакции. Если заданы *операнд1*, то эти операнды сохранятся в системном файле Adabas (*файл контрольных точек*) как данные транзакции (максимально до 2 КБ). Эти данные транзакции можно прочитать, используя, оператор GET TRANSACTION DATA.

BACKOUT [TRANSACTION] - отменяет все изменения, произведенные в базе данных во время логической транзакции. Оператор освобождает все записи, удержанные во время выполнения этой транзакции. Команда выдается системой Natural при нажатии клавиши Clear и ввода терминальной команды (%%).

GET TRANSACTION [DATA] [*операнд1…***] -** чтение данных транзакции, сохраненных предыдущим оператором END TRANSACTION. Последовательность, длины и форматы полей оператора GET TRANSACTION должны быть идентичны последовательности, длинам и форматам, указанным в операторе END TRANSACTION.

СУБД ADABAS проводит *автоматический откат транзакции*, если программа не завершила транзакцию в пределах выделенного времени.

Откат транзакции ADABAS может выдавать в следующих случаях:

– Превышение лимита времени удержания записей и выполнения транзакции (ожидание ответа пользователя).

– При аварийном завершении программы.

– При больших циклах обработки вследствие неправильно спроектированной программы.

При автоматическом откате все проведенные модификации откатываются, записи освобождаются, и пользователи получают доступ на модификацию к освобожденным записям.

При аварийных отключеньях ЭВМ и сбоях, приведших к непредвиденным остановкам СУБД, при повторном запуске СУБД ADABAS происходит автоматический откат незавершенных транзакций, информация о которых сохраняется в рабочей области WORK.

Вопросы для самопроверки

1. Основные средства системы SPoD.
2. Функции среды SPoD.
3. Свойства среды SPoD.
4. Основные компоненты системы SPoD и их назначение.
5. Модели данных СУБД ADABAS.
6. Понятия, применительно к СУБД ADABAS: база данных, файл, ассоциация, атрибут, множественный атрибут, поисковый атрибут, простая, составная и периодическая группа.
7. Организация и типы связей между записями файлов БД ADABAS.
8. Перечень и описание основных операций базовой модели данных СУБД ADABAS.
9. Понятие и сущность ассоциативного поиска записей файлов БД ADABAS.
10. Понятие и сущность ассоциативного поиска записей в связанных файлах и рандомизированного доступа к записям файлов БД ADABAS.
11. Архитектура СУБД ADABAS, как совокупность программных компонентов.
12. Многоуровневая архитектура СУБД ADABAS: внешний, концептуальный и внутренний уровни.
13. Среда хранения данных СУБД ADABAS и структурные элементы БД ADABAS.
14. Структура элементов БД ADABAS: накопитель, ассоциатор, рабочий набор и вспомогательные наборы данных.
15. Внесение записей в БД: синтаксис оператора STORE.
16. Поиск записей в БД: синтаксис оператора FIND.

17. Организация выдачи данных на экран и печати: синтаксис операторов DISPLAY и DEFINE PRINTER.
18. Операторы доступа и манипулирования данными: синтаксис операторов READ, HISTOGRAM, GET, UPDATE, DELETE.

Литература

1. Желваков Б.Б. Архитектура корпоративных информационных систем: Учебное пособие. - СПб.: СПбГИЭУ, 2012. - 622 с.
2. Часовских В.П., Воронов М.П. Исследование системных связей и закономерностей функционирования корпоративной информационной системы лесопромышленного предприятия в среде ADABAS и Natural: Монография, электронное издание. 2 изд. испр. и доп. – Екатеринбург: Урал. гос. лесотехн. ун-т, 2012. 180 с.
3. Воронов М.П., Часовских В.П. Системные связи и компоненты ядра КИС лесопромышленного предприятия в среде ADABAS и Natural // Фундаментальные исследования. - Пенза: Типография ИД «Академия Естествознания», 2011. - №8 – с. 333-337.
4. Бураков П.В., Петров В.Ю. Введение в системы баз данных: Учебное пособие. - СПб: СПбГУ ИТМО, 2010. - 128 с.
5. Токмаков, Г. П. Базы данных. Концепция баз данных, реляционная модель данных, языки SQL и XML: учебное пособие / Г. П. Токмаков. - Ульяновск: УлГТУ, 2010. - 192 с.
6. Системы управления базами данных ДИСОД / Е. С. Броневщук, В. И. Бурдаков, Л. И. Гуков и др.; Под общ. рук. В. И. Дракина. – М.; Финансы и статистика, 1987.-263 с.
7. ADABAS: описание утилит. Руководство пользователей.
8. NATURAL: операторы и синтаксис языка. Руководство пользователей.
9. Автоматизация систем управления предприятиями стандарта ERP-MRPII. / Обухов И.А., Гайфуллин Б.Н. - Интерфейс-Пресс, 2002.
10. Верников Г. Г. Стандарт MRPII. Структура и основные принципы работы MRPII-систем. – М: 2002.
11. Воронов М.П. Методы модификации модульных структур учета продукции в корпоративных информационных системах лесопромышленных предприятий. Диссертация на соискание ученой степени кандидата технических наук. - Екатеринбург, 2006.

12. Воронов М.П. Создание информационных систем управления лесопромышленным предприятием в среде ADABAS и Natural / М.П. Воронов, В.П. Часовских // Лесной Журнал, 2006. - №1. - С. 112-119.
13. Воронов М.П., Фатеркин А.С., Часовских В.П. Информационные технологии в управлении: СУБД ADABAS и проектирование приложений средствами NATURAL. - Екатеринбург: Уральский государственный лесотехнический университет, 2006. - 477 с.

Состав, инструментарий и основные приемы конфигурирования в среде разработки приложений 1С:Предприятие

1С:Предприятие является универсальной системой автоматизации экономической и организационной деятельности предприятия. Поскольку такая деятельность может быть довольно разнообразной, система 1С:Предприятие может «приспосабливаться» к особенностям конкретной области деятельности, в которой она применяется. Для обозначения такой способности используется термин *конфигурируемость*, т.е. возможность настройки системы на особенности конкретного предприятия и класса решаемых задач.

1С:Предприятие – это не просто программа, существующая в виде набора неизменяемых файлов, а совокупность различных программных инструментов, с которыми работают разработчики и пользователи. Приложение разработано на технологической платформе, которая включает средства работы с базой данных, встроенные язык программирования, редактор диалоговых форм и текстовый редактор, единый механизм обработки запросов.

10. Архитектура среды и особенности функционирования системы

Логически всю систему 1С:Предприятие можно разделить на две большие части, которые тесно взаимосвязаны друг с другом:

- *конфигурация* - прикладные решения предметной области, которые создаются средствами технологической платформы;
- *технологическая платформа* - управляет работой конфигурации, позволяет вносить в нее изменения или создавать собственную конфигурацию.

Существует одна платформа (1С:Предприятие) и множество конфигураций. Сама по себе платформа не может выполнить никаких задач автоматизации, так как она создана для обеспечения работы какой-либо конфигурации (рис. 53).

Рис. 53. Конфигураций много, а платформа одна

В *технологической платформе* выделяют две составляющие:

- среда разработки (режим Конфигуратор);
- среда исполнения (режим 1С:Предприятие).

Среда разработки

При создании прикладных решений применяются как визуальные средства разработки, так и программирование на встроенном языке. В режиме конфигуратора создаются все необходимые для решения прикладной задачи объекты, отражающие модель предметной области. В режиме конфигурирования формируется структура информационной базы, алгоритмы обработки, формы диалогов и выходных документов. На этапе конфигурирования система оперирует универсальными понятиями (объектами) - «Справочник», «Документ», «Журнал документов», «Реквизит», «Регистр» и др. Совокупность этих понятий и определяет концепцию системы. Работа программиста в среде разработки приводит к построению конкретной конфигурации.

Среда исполнения

Работа пользователя с информационной базой осуществляется при запуске системы в режиме «1С:Предприятие». В режиме исполнения система оперирует конкретными понятиями, описанными на этапе конфигурирования. При этом выполняется функционирование системы в предметной области: заполнение справочников, ввод документов, выполнение регламентных расчетов, формирование отчетов и т.д. В качестве примера прикладных решений можно перечислить:

- 1С:Управление небольшой фирмой;

- 1С:Бухгалтерия;
- 1С:Предприятие. Управление торговлей;
- 1С:Зарплата и Управление Персоналом;
- 1С:Предприятие. Управление производственным предприятием;
- 1С:Консолидания и т.д.

О других типовых прикладных решениях более подобно можно узнать на http://1c.ru/.

11. Основные понятия системы 1С:Предприятие

Основу системы 1С:Предприятие составляет понятие ***метаданные*** («данные о данных») - это совокупность объектов метаданных, настроенных на хранение и обработку информации.

Под ***объектом метаданных*** в системе 1С понимается формальное описание группы понятий предметной области со сходными характеристиками и одинаковым предназначением. Все объекты метаданных, которые существуют в системе 1С:Предприятие, образуют несколько основных видов. Каждый вид объектов метаданных представляет собой те «строительные элементы», из которых будет создана конфигурация. Названия видов объектов разработчик видит на первом уровне ***дерева метаданных*** в окне Кофигурация (рис. 54).

Ниже приведена краткая характеристика основных объектов метаданных.

Константы - постоянные и условно-постоянные величины, хранят информацию, которая не изменяется или изменяется достаточно редко: название организации, ее почтовый адрес и т.д.

Справочники - списки однородных элементов, используются для хранения нормативно-справочной информации.

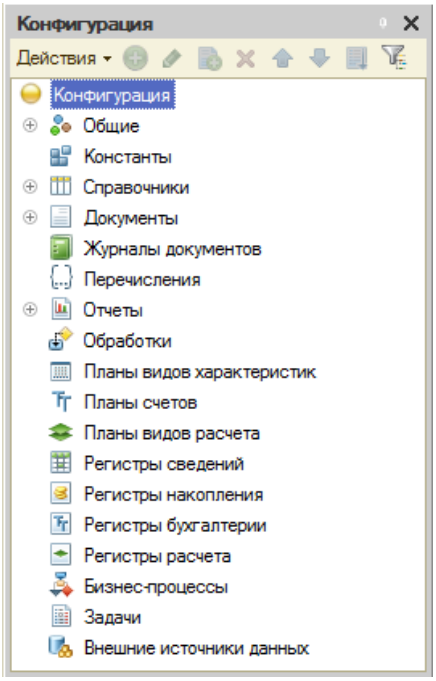


Рис. 54. Дерево метаданных

Документы - для ввода информации о совершенных хозяйственных операциях (приход, расход и т.д.).

Журналы документов - списки объектов данных типа «Документ», служат для работы с документами.

Перечисления - для хранения неизменяемых списков значений в конфигурации.

Отчеты - средства получения выходной информации; источником данных для посторения отчетов служат докуметы, справочники и регистры, а также информация, хранящаяся в константах.

Обработки - используются для выполнения различных действий над информационной базой.

Планы видов характеристик - предназначены для описания структуры хранения информации о характеристиках, создаваемых пользователем.

Планы счетов - списки объектов (счетов), по которым выполняется группировка средств бухгалтерского учета.

Регистры - средство накопления оперативной информации о наличии и движении средств.

Данные объекты являются условными «заготовками», на основании которых строятся прикладные объекты конкретной конфигурации.

Каждый объект метаданных обладает уникальным набором свойств. Этот набор описан на уровне системы и не может быть изменен, поскольку определяется назначением объекта в системе.

Главным свойством любого объекта метаданных является ***идентификатор*** – краткое наименование объекта. Имена объектов метаданных не могут содержать пробелов и начинаться с цифры.

Реквизиты объектов поддерживают следующие ***примитивные типы данных***:

- *числовой* – любое десятичное число; над данными числового типа определены основные арифметические операции;
- *строковый* – любая последовательность символов, в том числе и пустая;
- *дата* – представление любой корректной даты;
- *булево* – логическое «да» или «нет».

Помимо примитивных типов данных, изначально определенных системой, могут использоваться ***агрегатные типы данных***, определяемые только конкретной конфигурацией. Такие типы не присутствуют в конфигурации постоянно, а появляются в результате того, что добавлены некоторые объекты конфигурации (например, справочники, перечисления).

Например, после создания справочника «Города» появится тип данных СправочникСсылка.Города. И если указать какому-либо реквизиту этот тип данных, то в нем сохранится ссылка на конкретный объект справочника «Города» и в режиме 1С:Предприятие будет доступен список значений справочника «Города» (поле со списком).

12. Знакомство с платформой 1С:Предприятие

Все технические моменты, связанные с работой в Конфигураторе, будут описаны на конкретной задаче: организация ведет торговую деятельность, которая заключается в закупке товаров у поставщиков и производителей по документам «Приходная накладная». Дальнейшие действия организации сводятся к продаже этих товаров через торговую сеть, состоящую из нескольких филиалов по документам «Расходная накладная». Используя возможности 1С:Предприятие 8.2, необходимо разработать систему, координирующую деятельность организации.

Из условия поставленной задачи можно выделить следующие информационные объекты (справочники):

1) «Товар» – хранится информация о номенклатуре товаров;
2) «Фирмы» – наименования фирм-поставщиков и фирм-покупателей товаров;
3) «Филиалы» – названия филиалов фирм;
4) «Города» - названия городов, в которых находятся фирмы;
5) «Менеджеры» - сведения о менеджерах фирмы, торгующей товарами.

На основе справочников создаются документы «Приходная накладная» – поступление товаров и «Расходная накладная» – продажа товаров.

В общем виде схема потока информации от справочников в документы следующая представлена на рис. 55.

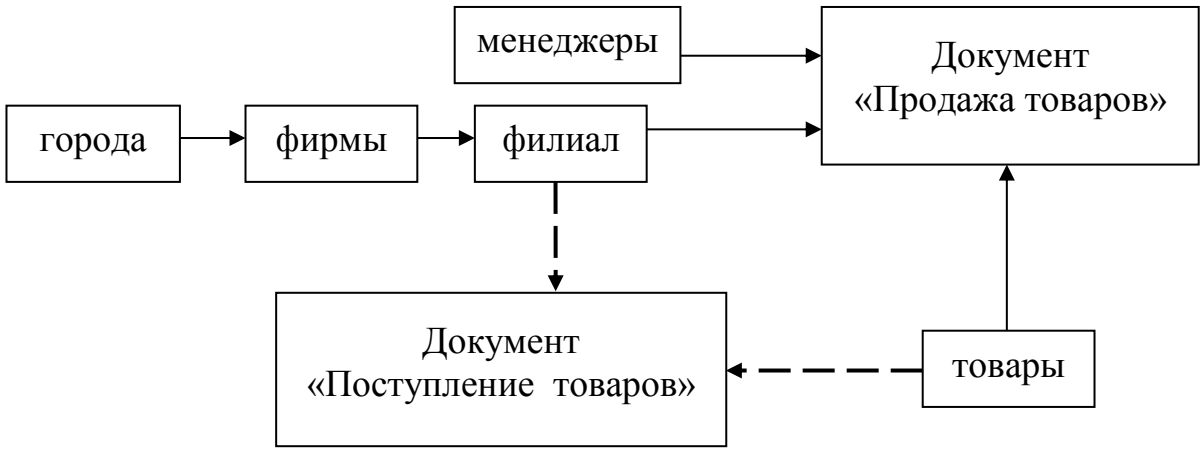


Рис. 55. Схема потока информации

Создание новой информационной базы

Окно запуска 1С позволяет начать новую разработку или совершенствовать уже имеющуюся. Для создания новой информационной базы необходимо щелкнуть по кнопке «Добавить» (рис. 56).

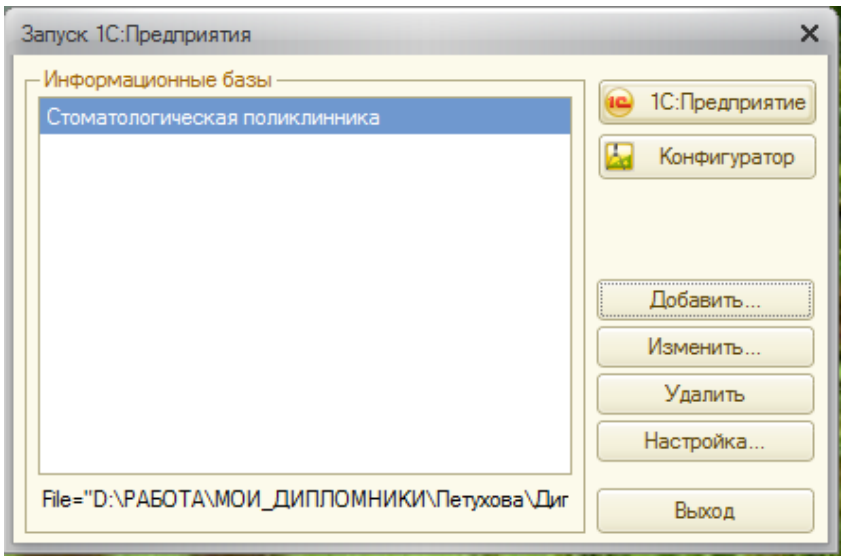


Рис. 56. Окно запуска системы 1С

Это приводит к появлению окна «Добавление информационной базы/группы», в котором необходимо выбрать сценарий – «Создание новой информационной базы». Щелкнуть по кнопке «Далее» (рис. 57).

На следующем этапе требуется определиться со способом создания будущей разработки. Необходимо выбрать нижний переключатель – «Создание информационной базы без конфигурации для разработки новой конфигурации или загрузки выгруженной ранее информационной базы». Щелкнуть по кнопке «Далее» (рис. 58).

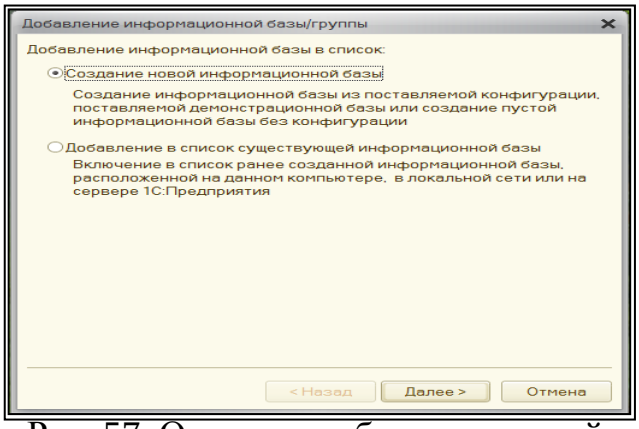


Рис. 57. Окно для добавления новой (выбора существующей) информационной базы

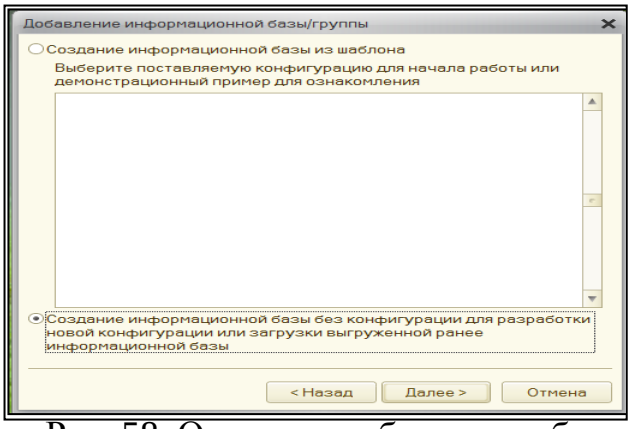


Рис. 58. Окно для выбора способа создания информационной базы

После выполненных действий на экране появляется окно, в котором необходимо задать имя создаваемой информационной базы (рис. 59). Щелчок

по кнопке «Далее» открывает диалоговое окно (рис. 60), в котором указать месторасположение создаваемого приложения.

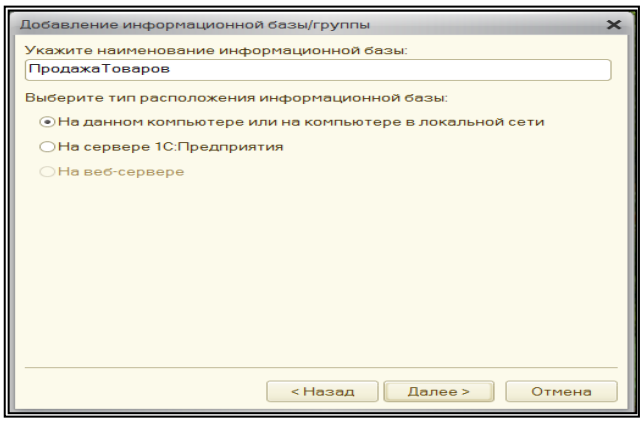


Рис. 59. Окно для наименования новой информационной базы

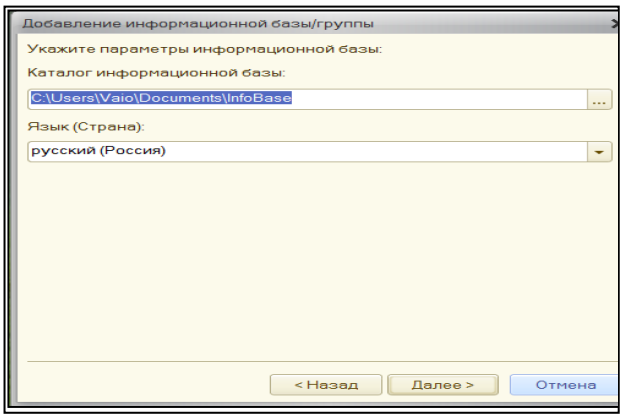


Рис. 60. Окно для указания каталога информационной базы

После щелчка по кнопке «Далее» открывается окно, предназначенное для установки параметров запуска системы. В нашем случае никакие параметры изменять не надо, достаточно щелкнуть по кнопке «Готово» (рис. 61).

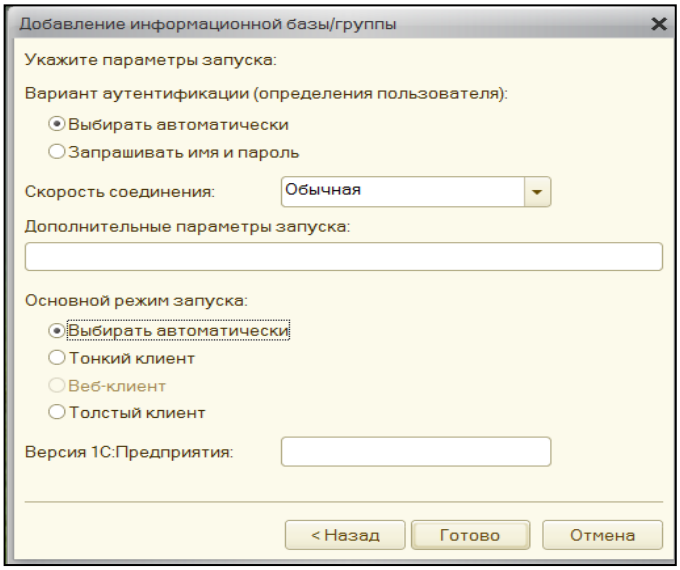


Рис. 61. Окно для укзания параметров запуска системы

Для начала работы по разработке своей информационной базы необходимо выделить ее имя в стартовом окне системы 1С:Предприятие и щелкнуть по кнопке «Конфигуратор» (рис. 62).

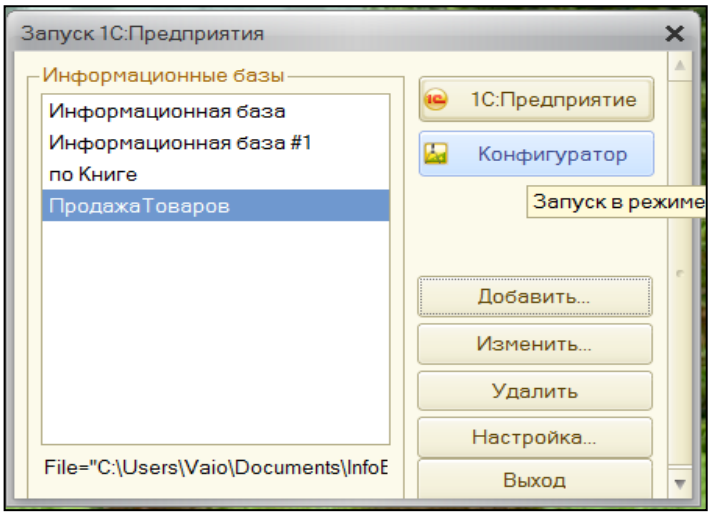


Рис. 62. Запуск Конфигуратора информационной базы «Продажа Товаров»

Окно разработки конфигуратора представляет собой среду для разработки прикладных решений. В меню Конфигурация выбрать раздел «Открыть конфигурацию». Создаваемые объекты автоматически отображаются в этом дереве.

Создание подсистем

Подсистема представляет собой основной элемент построения интерфейса системы. Любое прикладное решение подразумевает ориентацию интерфейса на конкретного пользователя (менеджер, бухгалтер и т.д.). В интерфейс включаются только те позиции которые требуются конкретному пользователю.

Подсистемы позволяют выделить в конфигурации функциональные части, на которые логически разбивается создаваемое прикладное решение.

Эти объекты располагаются в ветке «Общие» окна Конфигуратора и позволяют строить древовидную структуру, состоящую из подсистем и подчиненных подсистем.

Для создания новой подсистемы в окне Конфигуратора необходимо открыть ветку «Общие», выделить ветвь «Подсистемы» и щелкнуть правой кнопкой мыши для вызова контекстного меню. Выбрать пункт «Добавить» (рис. 63). В появившемся окне редактирования объекта указать имя новой подсистемы – «Управление» (рис. 64).

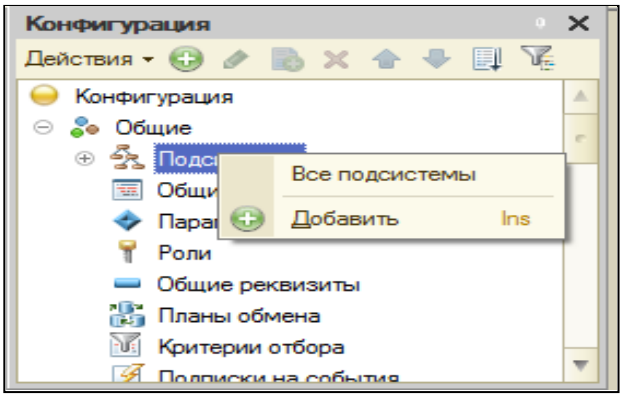


Рис. 63. Добавление новой подсистемы

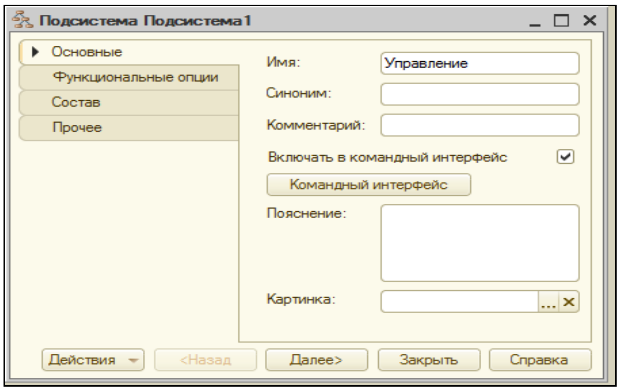


Рис. 64. Формирование подсистемы «Управление»

Аналогичным образом формируются подсистемы «Поступление», «Продажа», «Аналитика», «Регистры», «Отчеты».

Для запуска режима исполнения приложения необходимо в меню «Отладка» выбрать «Начать отладку», что позволит перейти в режим 1С:Предприятие.

Чтобы придать индивидуальность каждой подсистеме можно добавить уникальный значок. Для редактирования подсистемы «Управление» открыть ее в Конфигураторе. На вкладке «Основные» в поле «Картинка» щелкнуть по кнопке с изображением трех точек. В появившемся окне «Выбор картинки» в разделе «Стандартные» выбрать подходящий рисунок. Нажать «Ок».

Создание справочников

Справочник представляет собой информационный объект предметной области. В Конфигураторе создается структура любого справочника. Справочники являются источником информации, на основании которых разработчик может создавать свои объекты конфигурации.

Имя справочника (как и любого объекта метаданных) не может начинаться с цифры, содержать пробелы и различные специальные символы (кроме подчеркивания). Если имя состоит из двух и более слов, то они пишутся в одно слово без пробела, при этом можно использовать сокращения. Если введенное имя совпадает с уже существующим, программа сама предупредит об этом.

Справочник имеет вид таблицы. В системе 1С столбцы справочника называются реквизитами. В любом справочнике по умолчанию присутствуют обязательные (стандартные) реквизиты – *Код* и *Наименование*.

Код справочника используется для идентификации элементов справочника и содержит уникальные для каждого элемента справочника значения. Платформа может сама контролировать уникальность кодов и поддерживать автоматическую нумерацию элементов справочника. Длина кода – 9 символов.

Длина наименования справочника – 25 символов. Изменить длину наименования и кода можно на вкладке «Данные» при добавлении нового справочника в Конфигураторе.

Справочники могут содержать табличные части, которые также являются объектами конфигурации, только не основными, а вспомогательными.

Форма представления справочника ориентирована исключительно на пользователя – она предоставляет ему удобный интерфейс для работы. В зависимости от того, какие действия будут выполняться со справочником в 1С:Предприятие существует пять форм представления справочников:

- Форма элемента – для редактирования или создания элемента справочника;
- Форма группы – для редактирования или создания группы справочника;
- Форма списка – для отображения списка элементов справочника;
- Форма для выбора используется для того, чтобы в поле некоторой формы выбрать один из элементов справочника;
- Форма для выбора группы используется, когда в поле некоторой формы нужно выбрать не просто элемент справочника, а одну из его групп.

Справочник «Города»

1. Щелкнуть правой кнопкой мыши по ветви дерева конфигуратора «Справочники», в появившемся контекстном меню выбрать «Добавить». Окно редактирования объекта конфигурации «Справочники» имеет вид совокупности тематических вкладок (рис. 65).

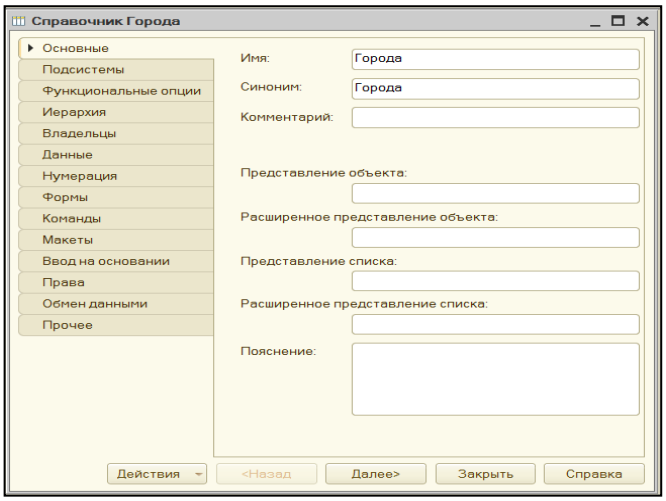


Рис. 65. Окно редактирования объекта конфигурации «Справочники»

2. На вкладке «Основные» задается Имя справочника – «Города».

3. На вкладке «Подсистемы» указывается принадлежность справочника к подсистеме. Все создаваемые по нашему примеру справочники будут относиться к подсистеме «Управление».

Справочник «Фирмы» будет иметь «шапку» с общими сведениями и табличную часть, в которой будет храниться информация о сотрудниках.

На вкладке «Данные», которая предназначена для определения реквизитов справочника, установить «Длину наименования» – 30. Значение, устанавливаемое для «Длины кода» по умолчанию, оставить без изменения (рис. 66).

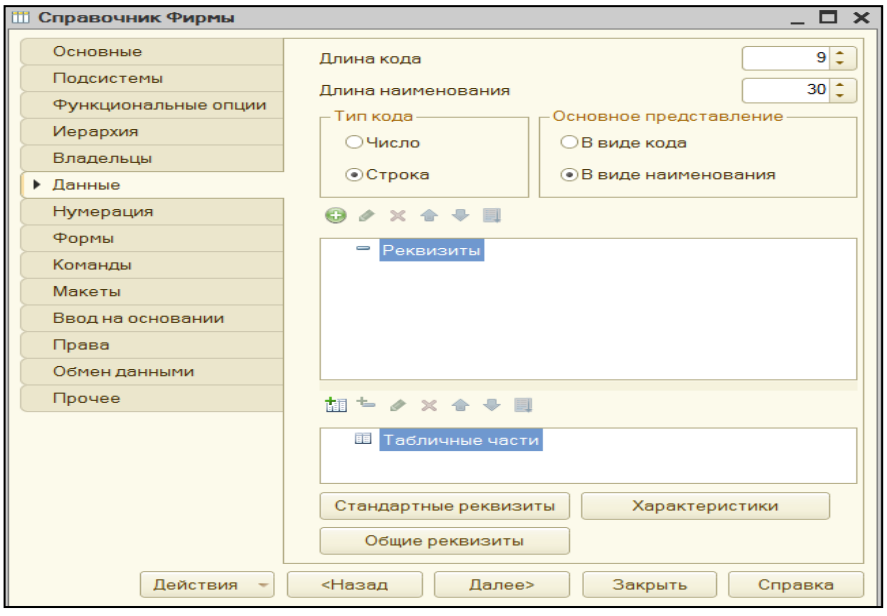


Рис. 66. Вкладка «Данные» окна редактирования справочника «Фирмы»

В центральной части окна вкладки «Данные» располагается поле «Реквизиты», предназначенное для добавления необходимых дополнительных реквизитов в справочник.

Для хранения информации о месторасположении фирмы, с которой ведется коммерческая деятельность, добавим реквизит – Город. Для этого щелкнуть по пиктограмме со знаком «+» и в появившемся окне свойств (рис. 67) указать:

- имя реквизита– Город;
- тип - «СправочникСсылка.Города».

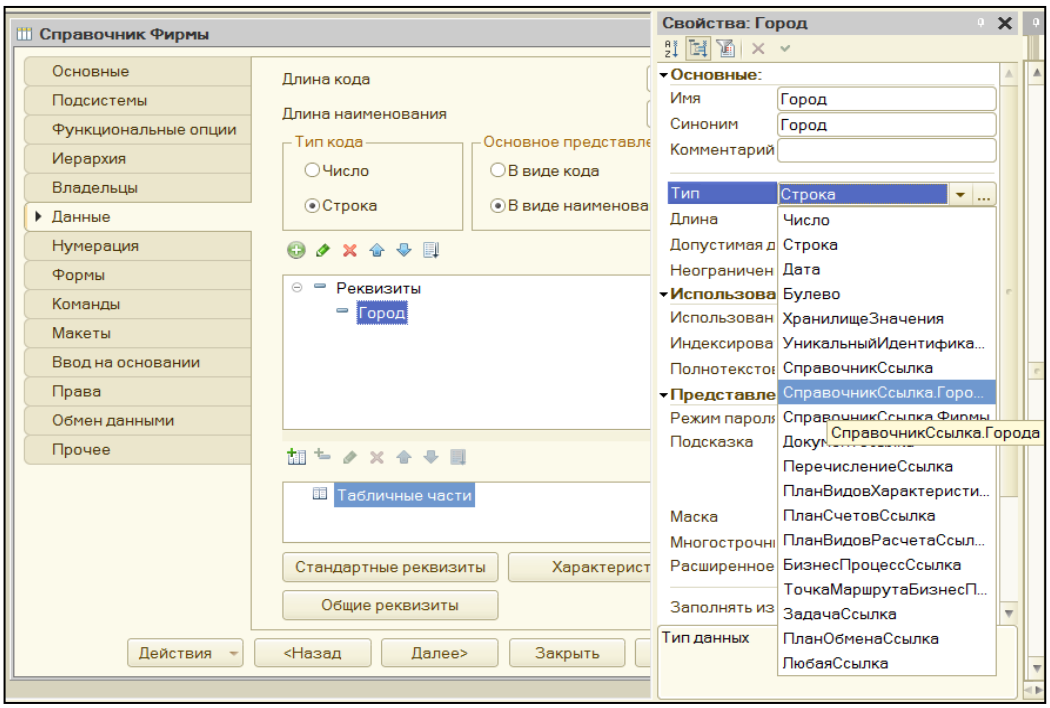


Рис. 67. Окно изменения свойств реквизита «Город» справочника «Фирмы»

Для добавления в справочник «Фирмы» табличной части «КонтактныеЛица» необходимо:

1) щелкнуть по пиктограмме «Добавить табличную часть» панели окна «Табличная часть» и указать имя табличной части – «КонтактныеЛица» (рис. 68):

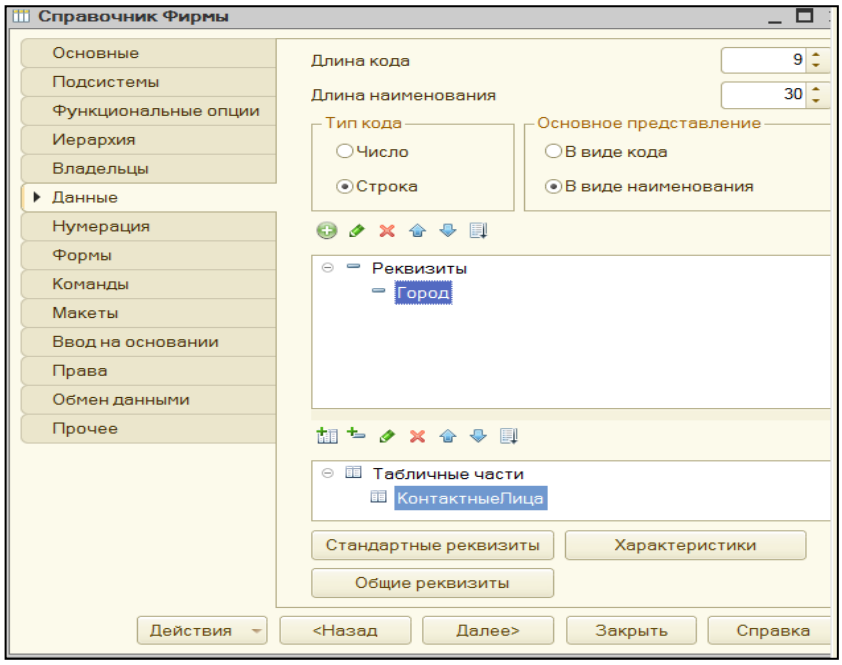


Рис. 68. Создание табличной части «КонтактныеЛица» справочника «Фирмы»

2) для добавления в табличную часть реквизитов (рис. 69) используется кнопка «Добавить реквизит» панели «Табличной части»:

- Сотрудник – для внесения ФИО (тип – строка, 50 символов);
- Должность – для определения должности специалиста (строка, 25 символов);
- Телефон – строка, 25 символов.

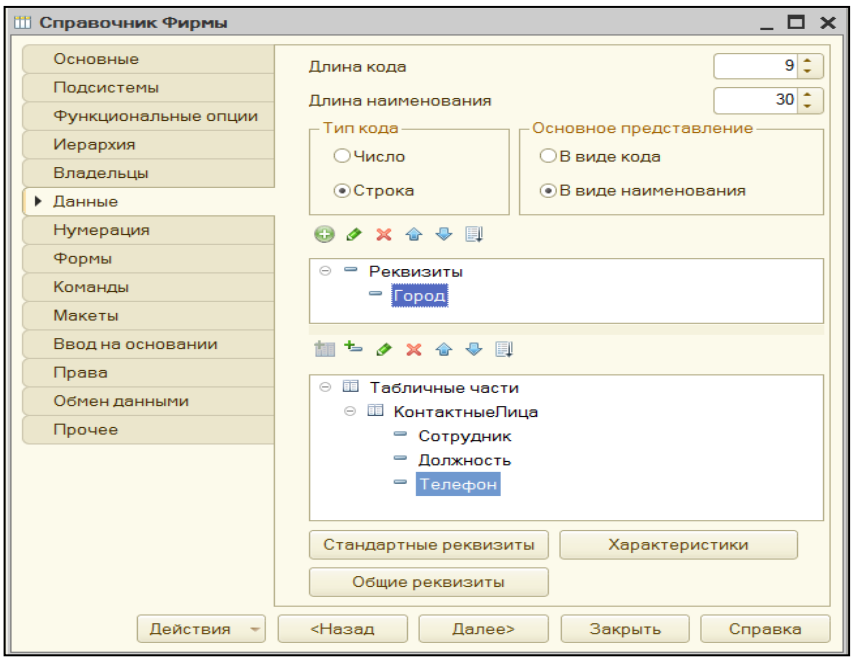


Рис. 69. Внесение реквизитов в табличную часть справочника «Фирмы»

Справочник «Товары» не имеет дополнительных реквизитов.

Для справочника «Товары» создадим форму списка:

1) перейти на вкладку «Формы» и щелкнуть по значку лупы, расположенному в строке «Списка» раздела «Формы» (рис. 5.70):

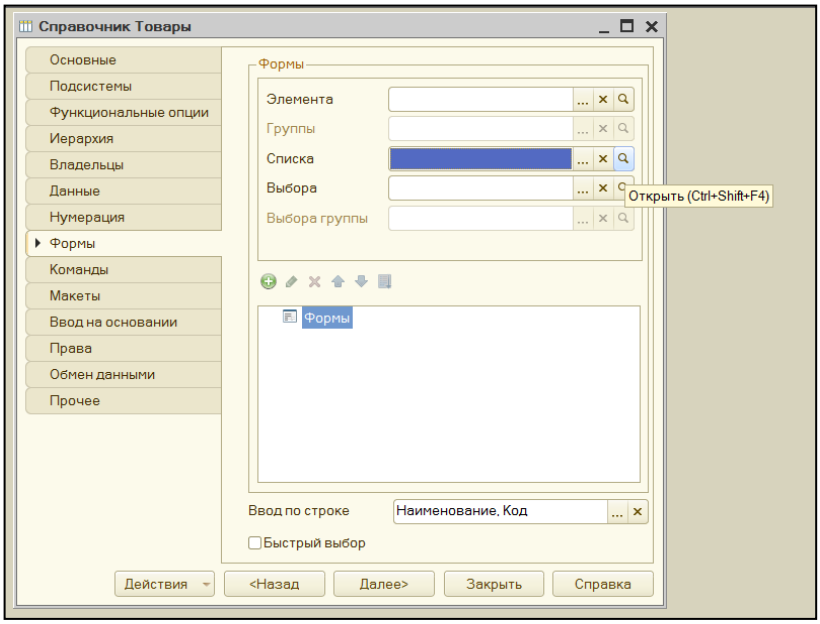


Рис. 70. Вкладка «Формы» справочника «Товары»

2) в появившемся окне диалога конструктора оставить параметры по умолчанию;

3) после щелчка по кнопке «Далее» в появившемся окне включить переключатели реквизитов «Наименование» и «Код» для добавления в создаваемую форму справочника (рис. 71):

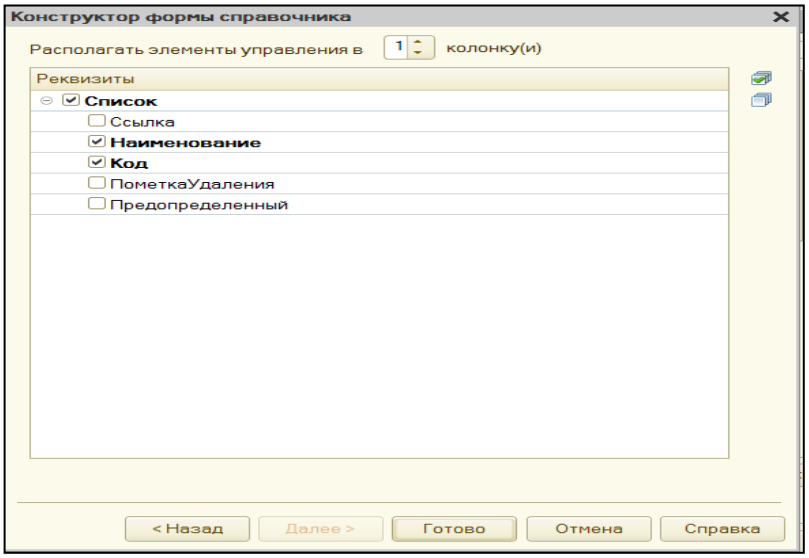


Рис. 71. Окно конструктора, предназначенное для включения реквизитов в форму

4) щелкнув по кнопке «Готово», на экране откроется окно редактора форм (рис. 72):

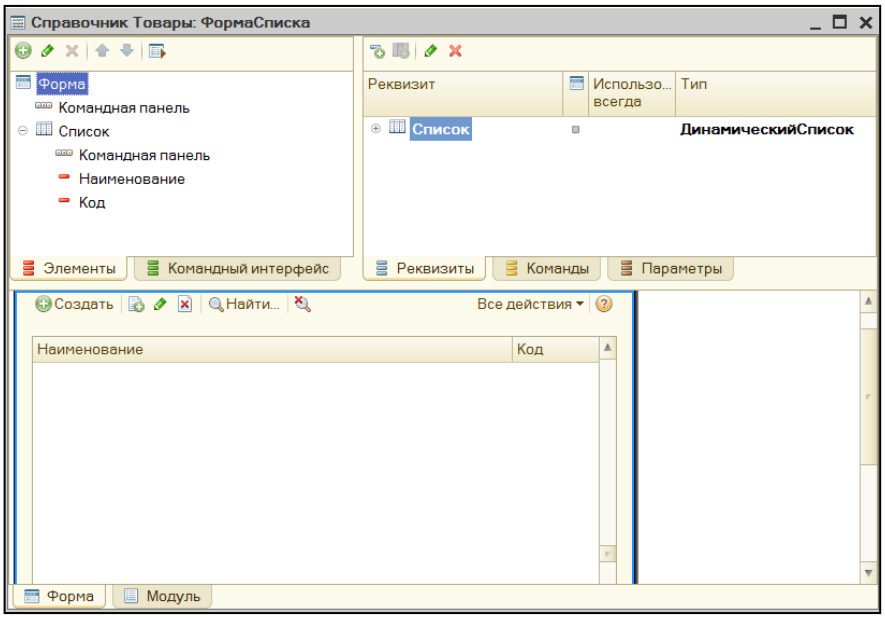


Рис. 72. Форма списка справочника «Товары» в редакторе форм

Окно редактора форм состоит из трех частей:

- в верхней левой части окна перечислены элементы, входящие в состав формы списка справочника «Товары»;
- в верхней правой части окна определяется список реквизитов формы («Данные формы»);
- в нижней части окна находится область предварительного просмотра (какой внешний вид имеет форма в результате выполненных действий).

Для удобства работы со справочником «Товары» поменяем подпись столбца «Наименование» на «Товар».

Для этого:

1) вызвать в контекстном меню окно свойств поля «Наименование»;
2) установить в разделе «Заголовок» новый вариант подписи – «Товар».

Самостоятельно

1. Создать справочники «Филиалы» и «Менеджеры» без дополнительных реквизитов.

2. Заполнить все созданные справочники произвольными данными в режиме запуска 1С:Предприятие.

Создание документов

Документы относятся к наиболее популярным объектам конфигурации. Они предназначены для фиксирования информации о происходящих событиях в подразделении или организации в целом. На основании объекта конфигурации Документ разработчик создает свои объекты конфигурации – документы конкретной прикладной направленности.

Документ обладает способностью проведения. Это означает, что событие, которое он отражает привело к состоянию изменения учета. До тех пор, пока документ не проведен, состояние учета неизменно и документ – не более чем черновик, заготовка.

Поскольку документ обладает способностью вносить изменения в состояние учета, он всегда привязан к конкретному моменту времени. В любом документе автоматически создаются два реквизита:

- *Дата* – для отображения даты документа;
- *Номер* – для последовательности нумерации документов одного типа.

В процессе работы пользователь может самостоятельно создавать новые документы – приходные и расходные накладные, счета и т.д. В базе данных каждый документ представляет собой отдельную запись в основной таблице, хранящей информацию об этом виде документов.

Для описания подробной информации по документу (например, список приходуемых товаров документа Приходная накладная) служат табличные части объекта конфигурации.

Для визуализации документа существует несколько основных форм:

- Форма документа – для редактирования или создания элемента документа;
- Форма списка – для отображения списка элементов документа;
- Форма выбора используется для того, чтобы в поле некоторой формы выбрать один из элементов документа.

Документ «ПоступлениеТоваров» фиксирует информацию о поступающих товарах. В качестве имени на вкладке «Основные» указать «ПоступлениеТоваров». Отнести документ к подсистеме «Поступление».

На вкладке «Данные» добавить реквизиты в Заголовочную часть («шапку») и Табличную часть для перечня однотипных данных (рис. 73).

Для создаваемого документа «ПоступлениеТоваров» в область «шапки» разместить реквизиты:

- Фирма (тип – СправочникСсылка.Фирмы);
- Филиал (тип – СправочникСсылка.Филиалы).

При создании табличной части «ПереченьТоваров» добавить реквизиты:

- Товар (тип – СправочникСсылка.Товары);
- Цена (тип – число, точность – 2);
- Количество (тип – число, точность - 2);
- Сумма (тип – число, точность - 2).

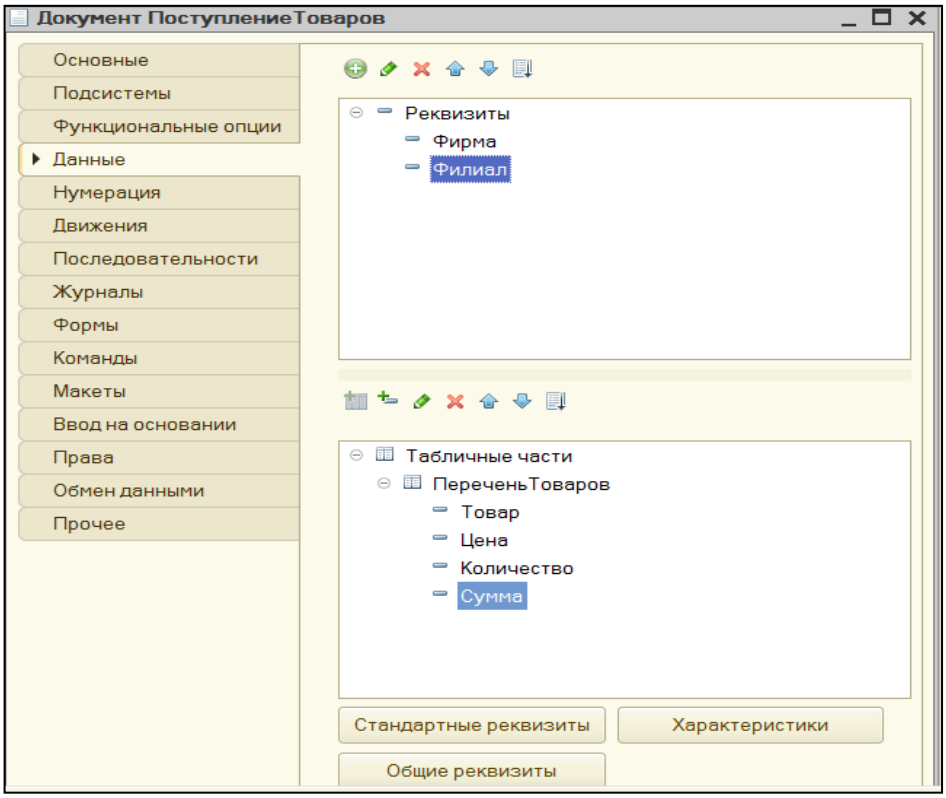


Рис. 73. Вкладка «Данные» окна редактирования документа «ПоступлениеТоваров»

После определения структуры документа создается его экранная форма. Для этого в окне редактирования объекта «ПоступлениеТоваров» выбрать вкладку «Формы» и щелкнуть на значке лупы в строке «Документа» раздела «Формы».

В появившемся окне все установки, предложенные системой, оставить без изменений. После щелчка по кнопке «Далее» откроется окно конструктора (рис. 74), в котором установлены флажки для отображения на

форме ранее созданных реквизитов документа. Работа в режиме диалога с конструктором формы документа завершается щелчком по кнопке «Готово».

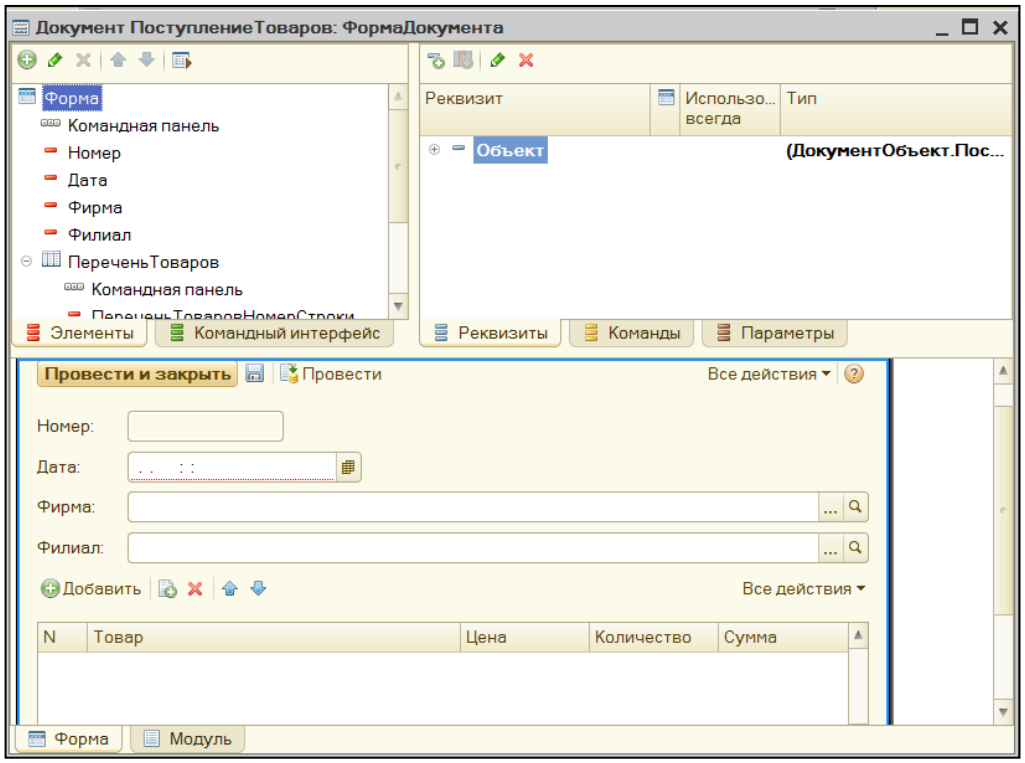


Рис. 74. Окно конструктора формы документа «ПоступлениеТоваров»

В нижней части экрана, в окне предварительного просмотра видно, что реквизиты Номер и Дата расположены друг под другом. Обычно в типовых документах они располагаются на одном уровне. Для выделения этих элементов создадим Обычную группу.

Для этого в верхней левой части экрана щелкнуть правой кнопкой мыши по ветке «Форма». В появившемся меню выбрать «Добавить» → «Группа – Обычная группа» и нажать «Ок».

После выполненных действий справа на экране появится окно свойств «Группы1». Изменить имя на – «Группа».

В окне свойств «Группы» (рис. 75) очистить поле «Заголовок», установить свойства:

- отображение – Рамка группы;
- группировка – Горизонтальная.

Используя кнопки перемещения элементов «Вверх» или «Вниз» в левом верхнем окне «Элементы», установить папку «Группа» после позиции «Командная панель».

Левой кнопкой мыши зацепить реквизит Номер и перетащить в папку «Группа». Аналогичные действия произвести с реквизитом Дата (рис. 75).

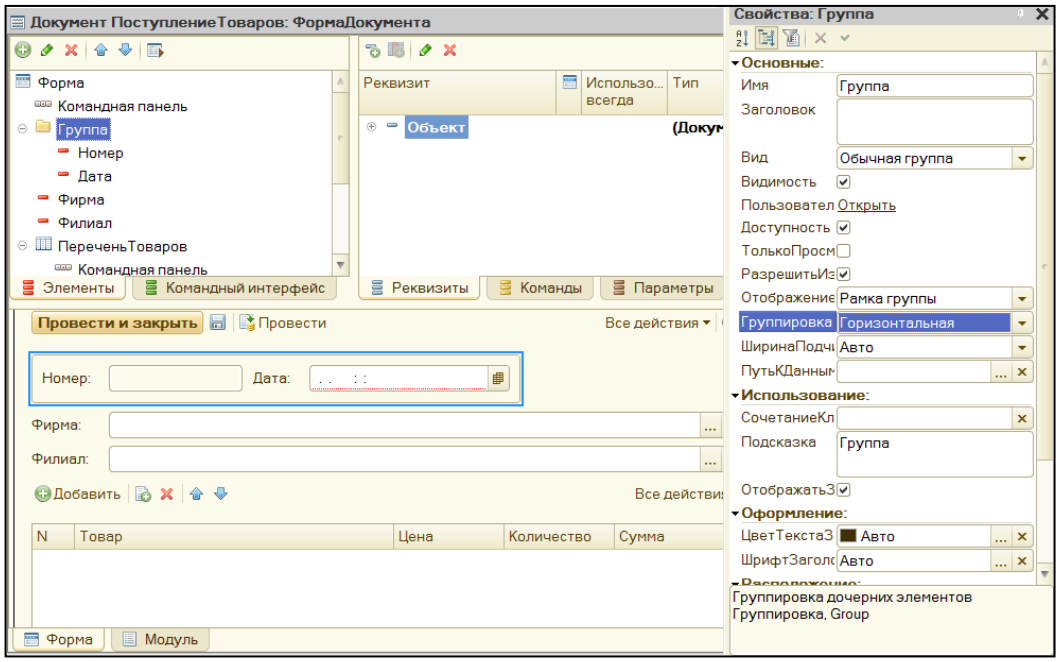


Рис. 75. Окно свойств «Группы»

Документ «ПродажаТоваров» фиксирует информацию о проданных товарах. В качестве имени на вкладке «Основные» указать «ПродажаТоваров». Отнести документ к подсистеме «Продажа».

В область «шапки» разместить реквизиты:

- Фирма (тип – СправочникСсылка.Фирмы);
- Филиал (тип – СправочникСсылка.Филиалы);
- Менеджер (тип – СправочникСсылка.Менеджеры).

При создании табличной части «ПереченьТоваров» добавим реквизиты:

- Товар (тип – СправочникСсылка.Товары);
- Цена (тип – число, точность – 2);
- Количество (тип – число, точность - 2);
- Сумма (тип – число, точность - 2).

Самостоятельно

1. Для документа «ПродажаТоваров» создать «ФормуДокумента», выделить реквизиты «Номер» и «Дата» в группу с горизонтальным расположением элементов.

2. В режиме 1С:Предприятие заполнить несколько документов «ПоступлениеТоваров» и «ПродажаТоваров».

13. Программирование формы документа

В платформе 1С:Предприятие 8.2 можно выделить два программных компонента:

a) *клиентское приложение* – программа, которая обеспечивает интерактивное взаимодействие системы с пользователем;
b) *сервер 1С:Предприятия* обеспечивает взаимодействие клиентского приложения с хранилищем данных (базой данных), исполняет программный код.

Начало работы пользователя всегда связано с клиентской частью – пользователь запускает программное приложение. На следующем этапе производится подключение к базе данных – в работу включается сервер 1С:Предприятие. После этого управление опять передается клиенту. Подобная передача управления от клиента северу и обратно производится постоянно на протяжении всего сеанса работы пользователя.

На сервере и на клиенте доступны разные свойства, методы и объекты встроенного языка. В связи с этим при создании программной процедуры или функции необходимо указать системе, для какого из двух компонентов она предназначается. Для этого в программном коде всех процедур имеются директивы компиляции:

&НаКлиенте – выполнение процедуры или функции производится в контексте клиентского приложения.

&НаСервере – выполнение программного кода происходит на сервере для обработки серверных событий формы.

&НаСервереБезКонтекста – выполнение программного кода происходит на сервере, однако реквизиты формы недоступны.

&НаКлиентеНаСервереБезКонтекста может выполняться как на сервере, так и в клиентском приложении. Используется очень редко.

Программный модуль – это «хранилище» для текста процедур или функций, вызываемых системой во время исполнения задачи в определенные моменты работы. Программные модули создаются для описания специфических алгоритмов функционирования.

Встроенный язык представляет собой предметно-ориентированный язык программирования. Все операторы языка имеют русское написание.

Исходный текст программного модуля может состоять из операторов и комментариев.

Комментарии используются для размещения в тексте программного модуля всякого рода пояснений к работе модуля. В тексте программного модуля комментарий начинается парой символов “//”.

Операторы имеют вид стандартного обращения к процедуре. Между собой операторы обязательно следует разделять символом “;”. Можно располагать произвольное число операторов на одной строке, разделяя их символом “;”.

Именем переменной, объявленной процедуры или функции, может быть любая последовательность букв, цифр и знаков подчеркивания “_”. Вновь создаваемые имена не должны совпадать с зарезервированными словами языка или именами существующих процедур и функций, доступных на момент исполнения. Распознавание имен переменных, процедур и функций ведется без учета регистра букв.

Автоматизация расчета суммы

При заполнении документа «ПоступлениеТоваров» серьезным недостатком является необходимость ручного заполнения поля Сумма, что препятствует автоматизации работы пользователей при внесении данных в документ.

Цель:

- ✓ после установки количества единиц товара должна автоматически рассчитываться сумма (произведение количества на цену);
- ✓ после установки цены товара также должна рассчитываться сумма (произведение цены на количество).

В Конфигураторе в свойствах документа «ПоступлениеТоваров» на вкладке «Форма» вызвать «ФормуДокумента» двойным щелчком левой кнопки мыши по ее имени.

В окне конструктора «ФормыДокумента» в списке элементов (верхнее левое окно) дважды щелкнуть левой кнопкой мыши по имени поля «ПереченьТоваровКоличество» для вызова окна свойств (рис. 76).

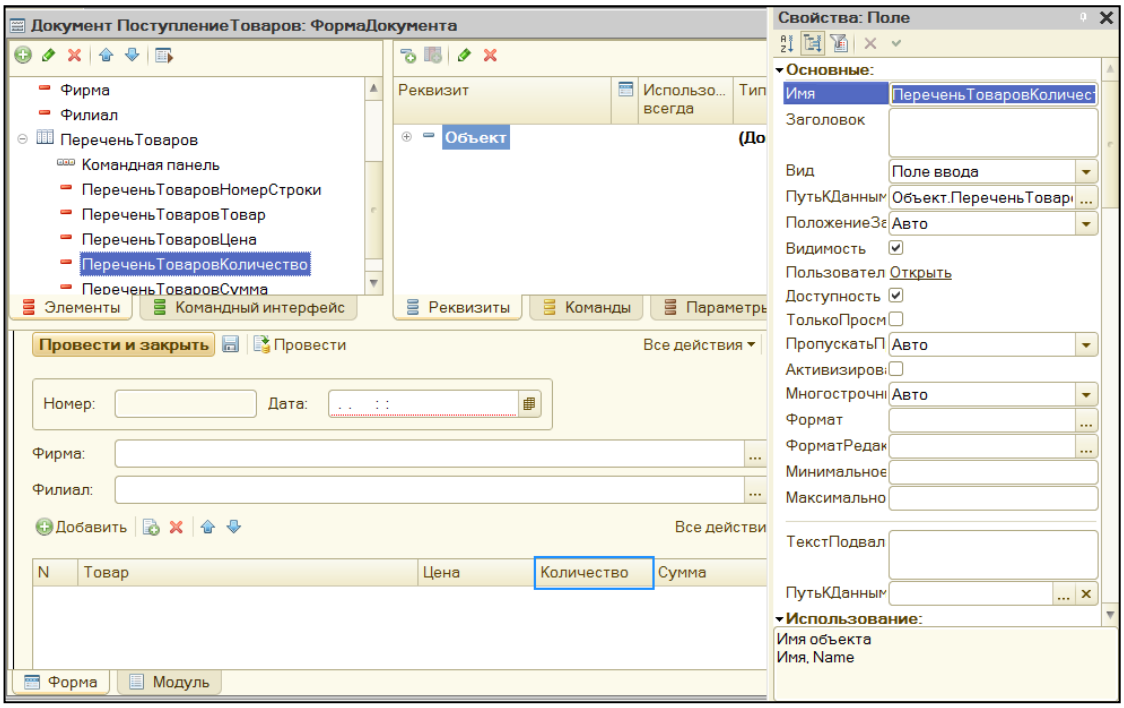


Рис. 76. Окно свойств поля ПереченьТоваровКоличество

В правой части экрана откроется «Список свойств». В строке события «ПриИзменении» окна свойств поля «ПереченьТоваровКоличество» щелкнуть по значку лупы, в результате попадем в процедуру, которая автоматически выполняется при указании Количества товара в табличной части документа «ПоступлениеТоваров» в режиме работы 1С.

Раскрыть процедуру и написать листинг для решения поставленной задачи:

```
&НаКлиенте
Процедура ПереченьТоваровКоличествоПриИзменении(Элемент)
СтрТабЧасти=Элементы.ПереченьТоваров.ТекущиеДанные
 СтрТабЧасти.Сумма=СтрТабЧасти.Цена*СтрТабЧасти.Количество
КонецПроцедуры
```

Данная конструкция производит заполнение поля Сумма табличной части текущей строки документа.

Ключевые слова `Процедура` и `КонецПроцедуры` определяют соответственно начало и окончание.

Переменная `СтрТабЧасти` определяет редактируемые в строке данные. `Элементы` – это элементы управления, расположенные на форме документа в табличной части с именем «ПереченьТоваров».

Аналогичная процедура «ПриИзменении» составляется для реквизита Цена табличной части «ПереченьТоваров».

Автоматическое заполнение полей

Когда пользователь в режиме 1С:Предприятие выбирает Товар, то данные в поля ЕдиницаИзмерения и Цена должны подставляться автоматически из информации, хранящейся в справочнике «Товары».

Для этого:

1. Создать справочник «ЕдиницаИзмерения» с дополнительным реквизитом Расшифровка (тип – Строка, длина – 100), подсистема - «Управление» (см. Создание справочника).
2. В справочник «Товары» добавить элементы ЕдИзм (тип данных – СправочникСсылка.ЕдиницаИзмерения) и Цена (тип данных – число, точность – 2).
3. В режиме Конфигуратор в табличную часть документа «ПоступлениеТоваров» добавить реквизит ЕдИзм (тип – СправочникСсылка.ЕдиницаИзмерения). Используя кнопки перемещения, установить последовательность реквизитов - Товар, ЕдИзм, Цена, Количество, Сумма.
4. В разделе «Формы» документа «ПоступлениеТоваров» открыть имеющуюся форму документа. В правом верхнем окне «Реквизит» конструктора открыть список «Объект» и из табличной части «ПереченьТоваров» левой кнопкой мыши перетащить реквизит ЕдИзм в табличную часть «ПереченьТоваров» левого верхнего окна «Форма», установив позицию после реквизита Товар.
5. В окне свойств реквизита Товар окна «Форма» вызвать событие «ПриИзменении» и написать текст процедуры:

```
&НаКлиенте
Процедура ПереченьТоваровТоварПриИзменении(Элемент)
Стр = Элементы.ПереченьТоваров.ТекущиеДанные;
Стр.ЕдИзм = ИзвлечьЕдИзм(Стр.Товар);
Стр.Цена = ИзвлечьЦену(Стр.Товар);
КонецПроцедуры
```

В тексте процедуры вызываются функции ИзвлечьЕдИзм и ИзвлечьЦену. Это пользовательские функции, которые необходимо

создать. Параметр передается выбранным пользователем значением элемента «Товар» табличной части «ПереченьТоваров» документа «ПоступлениеТоваров», а результат действия этих функций извлекается из справочника «Товары».

Тексты функций ИзвлечьЕдИзм и ИзвлечьЦену располагаются на вкладке «Модуль» окна формы документа. Функции выполняются &НаСервере.

```
&НаСервере
Функция ИзвлечьЕдИзм(Товар)
 Возврат Товар.ЕдИзм;
КонецФункции

&НаСервере
Функция ИзвлечьЦену(Товар)
 Возврат Товар.Цена;
КонецФункции
```

6. Проверить работу модуля в режиме 1С:Предприятие, предварительно заполнив справочник «ЕдиницаИзмерения», и добавить данные в справочник «Товары».

Заметьте, что при выборе товара поля ЕдИзм и Цена заполняются автоматически, но разрешается редактирование. Для исключения такой возможности в режиме Конфигуратор для реквизитов ЕдИзм и Цена табличной части «ПереченьТоваров» документа «ПоступлениеТоваров» вкладки «Формы» в палитре свойств отключить флажок «Доступность».

Формирование сообщения пользователю

Необходимо доработать документ «ПоступлениеТоваров» таким образом, чтобы прямо из формы при щелчке по кнопке в окне сообщений обеспечить просмотр номера телефона поставщика, указанного в поле Фирма.

Для этого:

1. В справочнике «Фирмы» добавить реквизит КонтактныйТел (тип – Текст) и в режиме 1С:Предприятие заполнить данными.

2. В окне редактирования формы документа «ПоступлениеТоваров» в верхнем правом окне на вкладке «Команды» нажать кнопку «Добавление новой команды». Указать имя – «КонтактныйТелефон» (рис. 77).

3. В разделе «Действия» окна свойств команды «КонтактныйТелефон» (рис. 77) щелкнуть по значку с изображением лупы и написать следующую процедуру:

```
&НаКлиенте
Процедура КонтактныйТелефон(Команда)
Сообщение = Новый СообщениеПользователю;
Сообщение.Текст = ПолучитьТел(Объект.Фирма);
Сообщение.Сообщить();
КонецПроцедуры
```

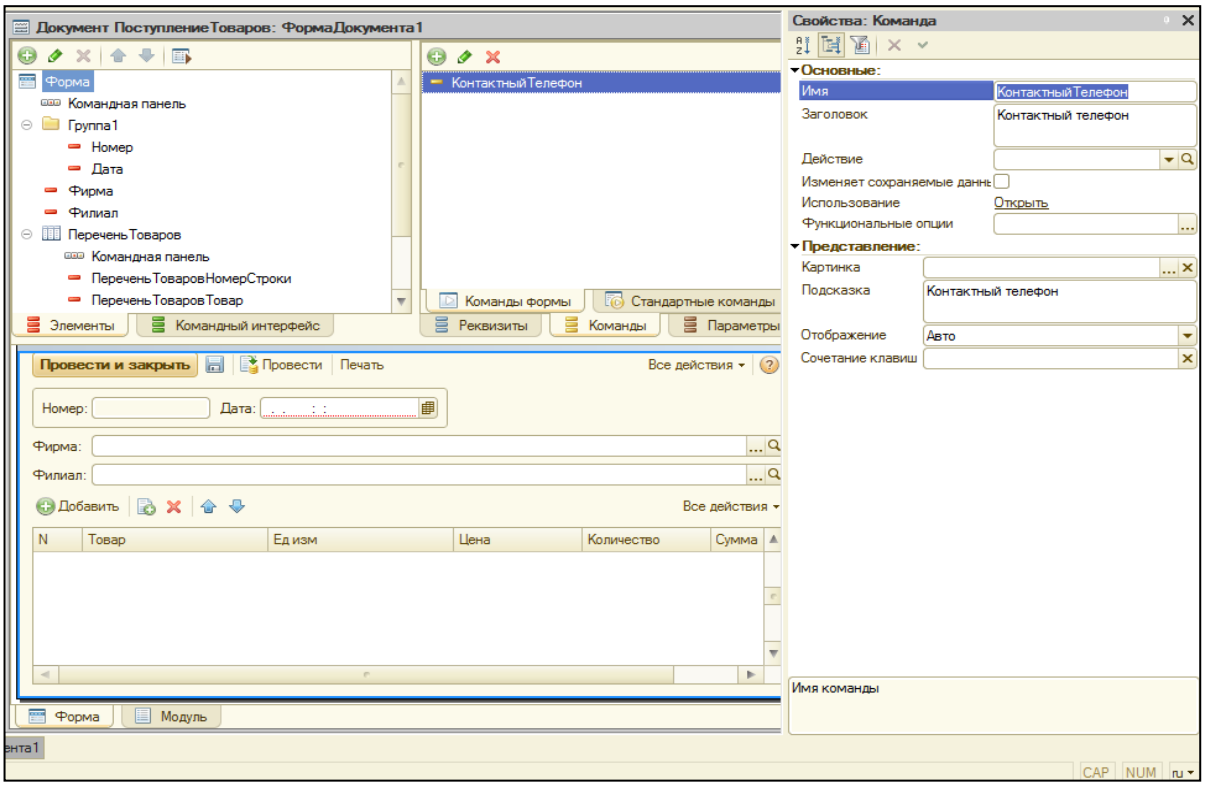


Рис. 77. Окно свойств добавления команды «КонтактныйТелефон»

В тексте программы «КонтактныйТелефон» необходимо воспользоваться серверной функцией ПолучитьТел для возврата значения с номером телефона:

```
&НаСервере
Функция ПолучитьТел(Фирма)
Возврат Фирма.КонтактныйТел
```

КонецФункции

4. Для добавления кнопки «Контактный телефон» в интерфейс формы документа «ПоступлениеТоваров» необходимо перетащить мышкой имя команды «КонтактныйТелефон» правого верхнего окна «Команды формы» на элемент «Командная панель» левого верхнего окна «Элементы» (рис. 78). Оцените результат работы в режиме запуска 1С.

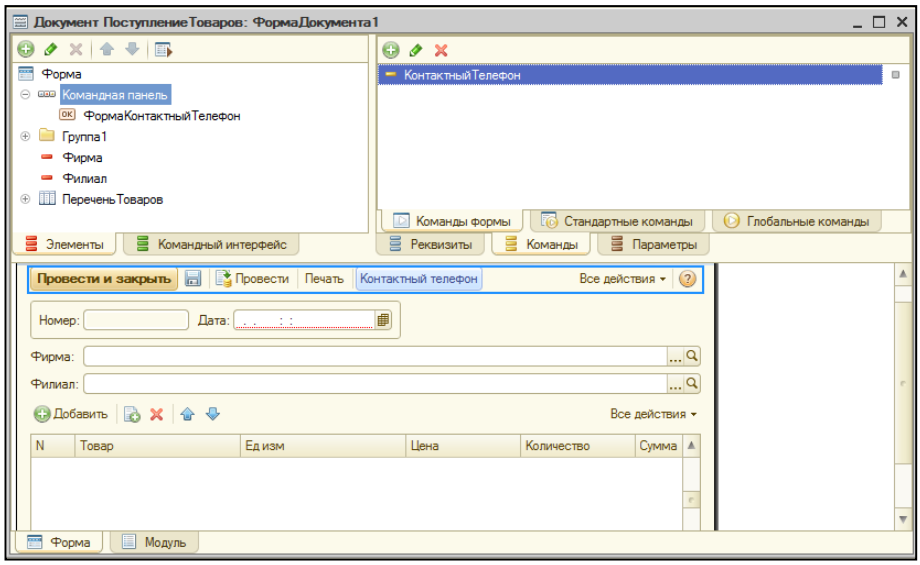


Рис. 78. Добавление кнопки «КонтактныйТелефон» в форму

Расчет итогов по документу

При работе с документом «ПродажаТоваров» для полноты картины менеджеру, организующему сделку, важно видеть итоговую сумму продажи по всем товарам составляемого документа.

Функция **Итог(<колонка>)** возвращает сумму значений всех строк в указанной колонке. Итог подсчитывается только для колонок, у которых указан тип число.

Для автоматизации расчета суммы необходимо:

1. В документ «ПродажаТоваров» добавить реквизит – СумДок (тип – число, точность – 2);

2. На вкладке «Формы» внести СумДок в форму документа, используя верхнее правое окно «Реквизит» (рис. 79). В окне свойств отключить флажок доступности поля.

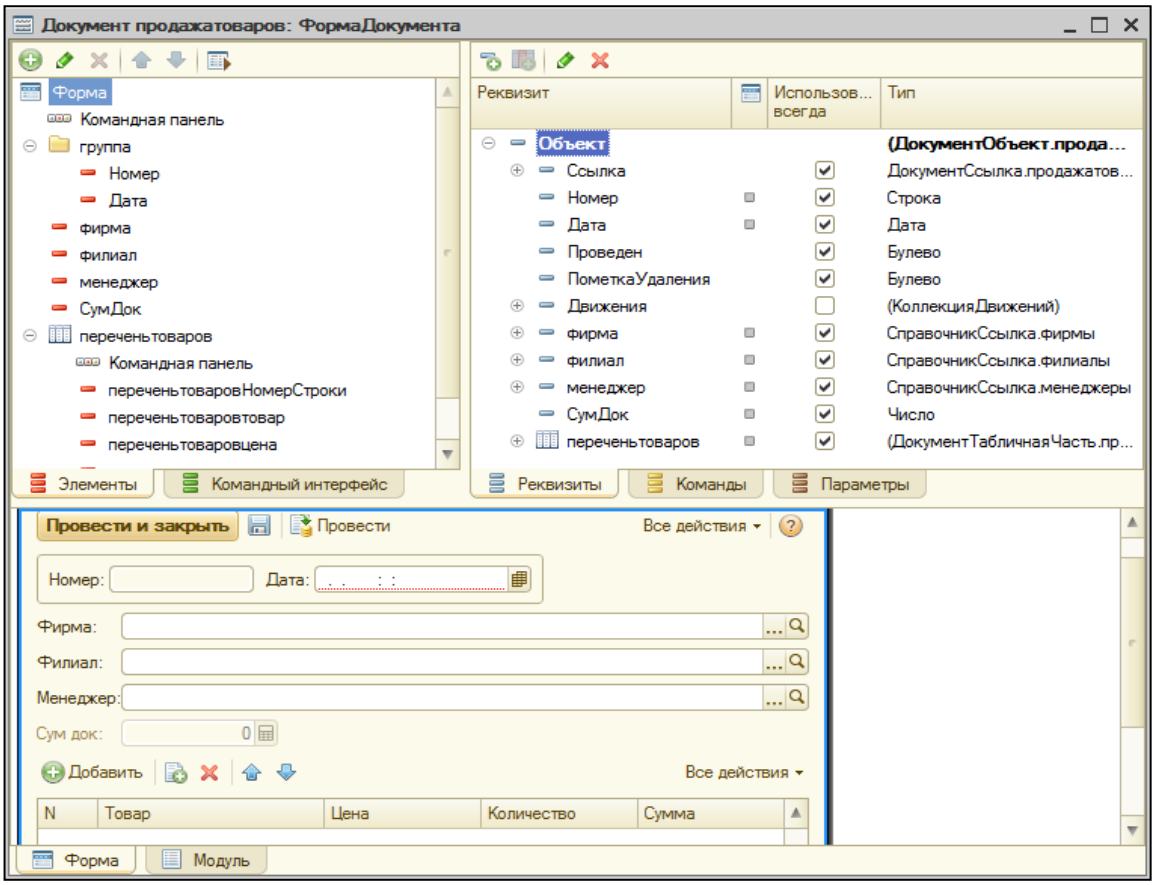


Рис. 79. Добавление реквизита СумДок в форму документа «ПродажаТоваров»

3. Добавить процедуру для события «ПриИзменении» табличной части «ПереченьТоваров»:

```
&НаКлиенте
Процедура ПереченьтоваровПриИзменении(Элемент)
Объект.СумДок=Объект.Переченьтоваров.Итог("Сумма");
КонецПроцедуры
```

Самостоятельно создать документ «Продажа товаров», в котором обеспечить автоматический расчет суммы по строке табличной части и документа в целом, автоматическое заполнение полей «Цена» и «ЕдИзм» табличной части «ПереченьТоваров» при выборе наименования товара.

Создание печатной формы документа

В процессе работы пользователям помимо электронной формы документов требуются их печатные варианты. Для создания печатной формы

документа «ПоступлениеТоваров» необходимо открыть окно этого объекта в режиме конфигурации:

1. На вкладке «Макеты» щелкнуть по кнопке «Конструкторы» в нижней части окна. Выбрать «Конструктор печати» (рис. 80).

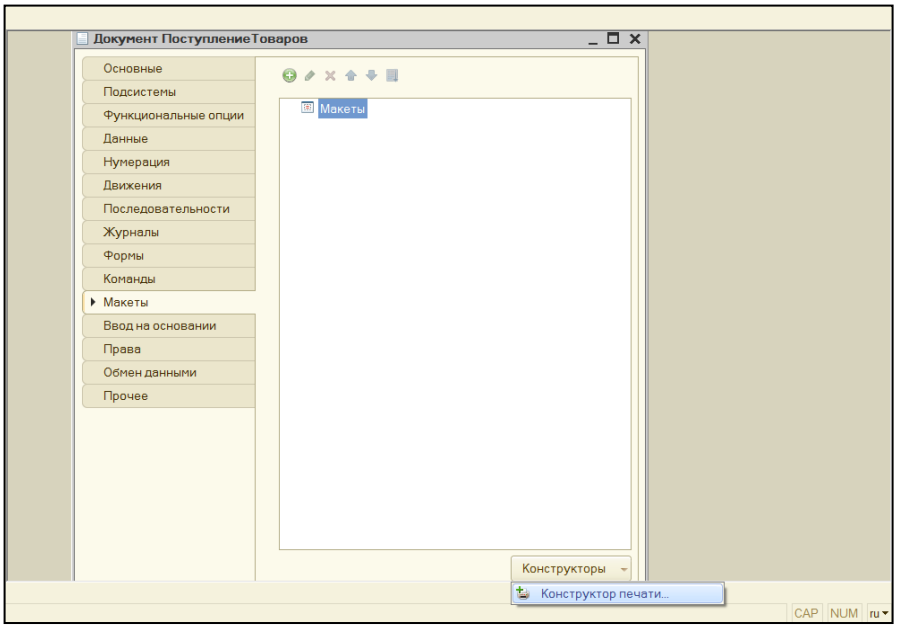


Рис. 80. Вкладка «Макеты» документа «ПоступлениеТоваров»

В появившемся окне согласиться с предложением создать команду для печати с именем «Печать», щелкнуть по кнопке «Далее».

В следующем окне «Конструктора печати документов» следует задать реквизиты «шапки» документа (рис. 81), выбрать все, кроме реквизита СумДок.

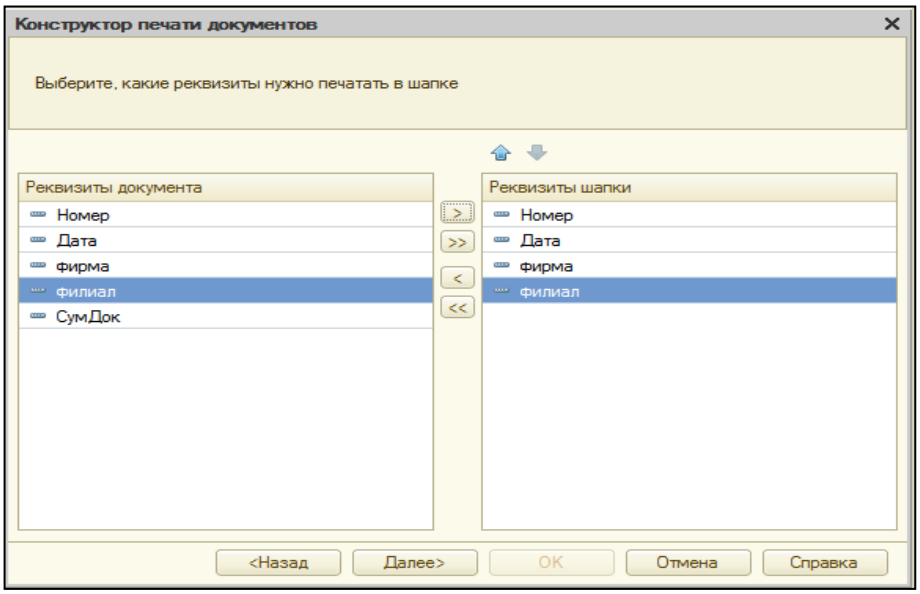


Рис. 81. Добавление реквизитов «шапки» в печатную форму

После этого в новом диалоговом окне задать содержание реквизитов табличной части (рис. 82) печатной формы (включить все поля).

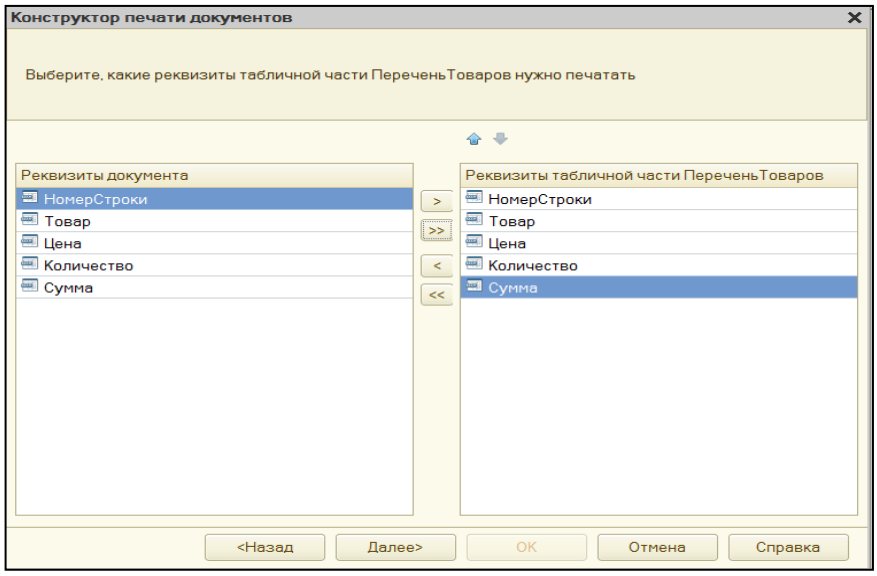


Рис. 82. Добавление реквизитов табличной части в печатную форму

В «подвале» будет отражаться итоговая сумма по документу, поэтому следует выбрать реквизит СумДок (рис. 83). Для продолжения нажать кнопку «Далее».

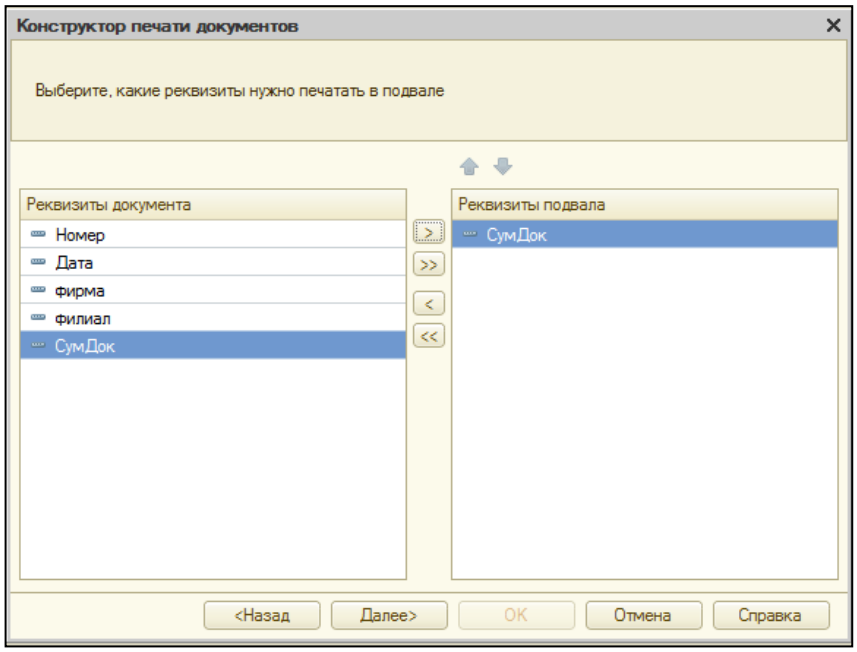


Рис. 83. Добавление реквизитов «подвала» в печатную форму

В новом появившемся окне указывается группа, куда будет входить команда формирования печатной формы (рис. 84).

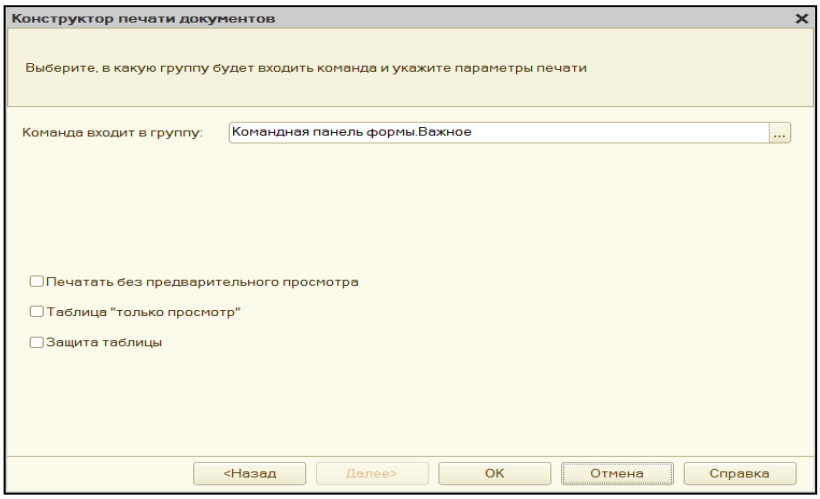


Рис. 84. Определение группы команды Печать

В завершении работы Конструктора на экране отобразится макет печатной формы документа «ПоступлениеТовара» (рис. 86) и набор автоматически составленных программных процедур.

Обычно, в печатных формах всегда присутствуют подписи руководителей. Для внесения изменений в печатную форму следует в конфигураторе открыть макет «Печать» документа «Поступление товаров» и создать новую секцию табличного документа – Подписи.

Для этого:

1. Выделить 3-4 строки в нижней части макета.

2. В главном меню «Таблица» выбрать раздел «Имена» → «Назначить имя». В результате на экране откроется окно, в котором следует внести имя – «Подписи». В созданной области ввести текст в соответствии с рис. 85.

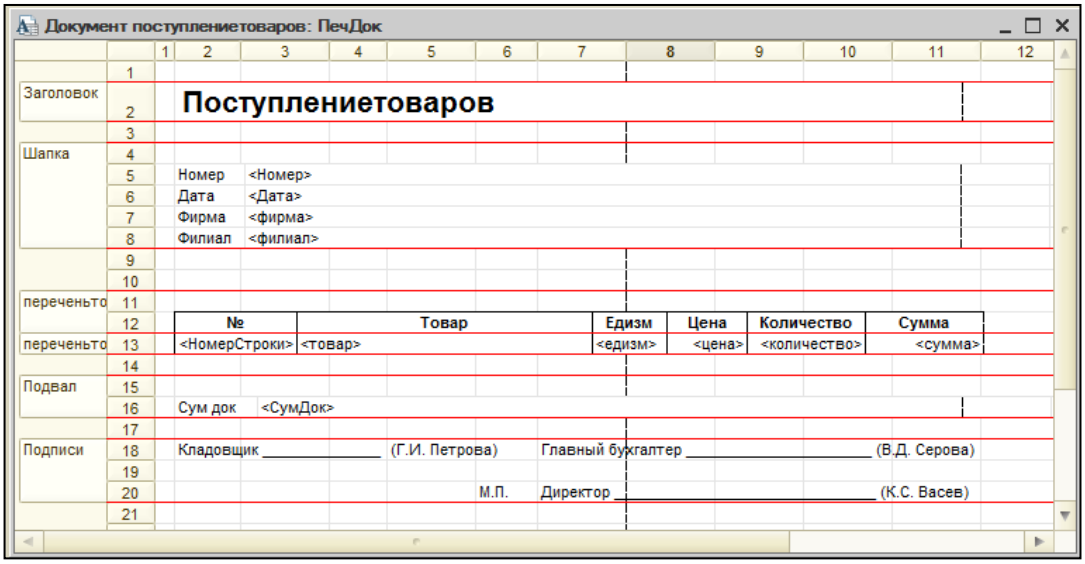


Рис. 85. Выделение области «Подписи» в печатной форме

3. На вкладке «Прочее» документа «ПоступлениеТоваров» щелкнуть по кнопке «Модуль менеджера» для внесения изменений в текст процедуры «Печать».

Добавить в текст процедуры строки, выделенные жирным шрифтом:

```
…
ОбластьЗаголовок = Макет.ПолучитьОбласть("Заголовок");
Шапка = Макет.ПолучитьОбласть("Шапка");
ОбластьПереченьТоваровШапка = Макет.ПолучитьОбласть("ПереченьТоваровШапка");
ОбластьПереченьТоваров = Макет.ПолучитьОбласть("ПереченьТоваров");
Подвал = Макет.ПолучитьОбласть("Подвал");
ОбластьПодписи = Макет.ПолучитьОбласть("Подписи");
ТабДок.Очистить();
…
КонецЦикла;
Подвал.Параметры.Заполнить(Выборка);
ТабДок.Вывести(Подвал);
ТабДок.Вывести(ОбластьПодписи);
            ВставлятьРазделительСтраниц = Истина;
    КонецЦикла;
    //}}
КонецПроцедуры
```

Самостоятельно создайте печатную форму для документа «ПродажаТоваров». В печатной форме документа должны быть подписи Менеджера, Кладовщика, Начальника отдела продаж фирмы.

14. Язык запросов 1С:Предприятие

Запросы в системе 1С предназначены для выборки информации из базы данных. Запросы в 1С способны выполнять много полезных функций: группировка отобранных записей, вычисление итогов и т.д.

Запрос в системе 1С представляет собой текст на специальном языке запросов. В этом тексте описывается, что является источником информации для запроса, а также указываются условия для построения запроса. В качестве источника информации используются таблицы информационной базы.

Таблицы, участвующие в запросе, делятся на два основных класса *реальные* и *виртуальные*. Реальные таблицы хранятся в базе данных, а виртуальных таблиц в базе данных нет. При обращении к информации

виртуальных таблиц система самостоятельно автоматически собирает информацию из реальных таблиц базы данных для выполнения запроса.

Текст запроса может состоять из нескольких частей:

- *описание запроса* – определяет источники данных, поля выборки, группировки и т.д.;
- *объединение запросов* – определяет, как будут объединены результаты выполнения нескольких запросов;
- *упорядочивание результатов* – определяет условия упорядочивания строк результата запроса;
- *автоупорядочивание* – позволяет включить режим автоматического упорядочивания строк результата запроса;
- *описание итогов* – определяет, какие итоги необходимо рассчитать в запросе и каким образом группировать результат.

Детальнее с языком запросов удобнее знакомиться в процессе решения конкретных задач.

Для работы с запросами будем использовать подсистему «Аналитика».

1.Выборка предназначена для отбора данных, хранящихся в таблицах.

Пример. Создать список всех работающих менеджеров из справочника «Фирмы», напротив фамилии каждого сотрудника должен быть указан номер его телефона и организация, в которой он работает. Список отсортировать по алфавиту.

Для этого:

1) перейти на ветвь конфигурации «Отчеты» → выбрать «Добавить» в контекстном меню. На вкладке «Основные» (рис. 86) указать имя нового запроса – «Выборка» и подсистему;

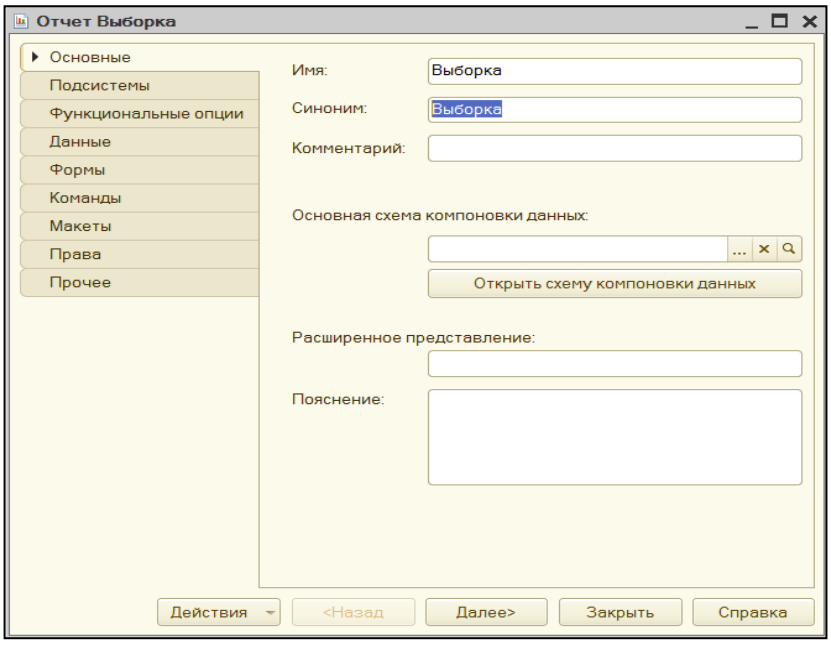


Рис. 86. Окно редактирования объекта конфигурации «Выборка»

2) после нажатия кнопки «Открыть схему компоновки данных» на вкладке «Основные», появится окно конструктора макета (рис. 87), щелкнуть по кнопке «Готово», оставив без изменения все настройки системы:

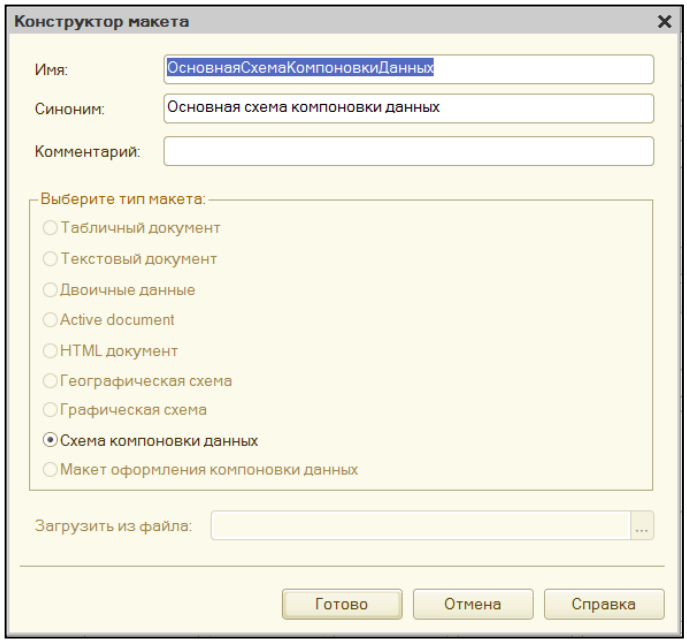


Рис. 87. Окно «Конструктор макета»

В результате выполненных действий на экране откроется окно конструктора схемы компоновки данных,выбрать вкладку «Наборы данных» (рис. 88).

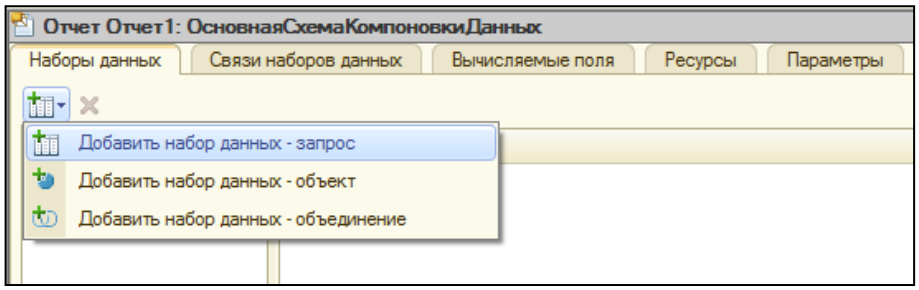


Рис. 88. Окно конструктора схемы компоновки данных

3) с помощью кнопки «Добавить набор данных» выбрать вариант – «Добавить набор данных – Запрос» (рис. 88).

В окно «Запрос», расположенному в нижней части, добавить текст:

```
ВЫБРАТЬ Сотрудник, Телефон,
Ссылка.Наименование КАК Организация
ИЗ Справочник.Фирмы.КонтактныеЛица
УПОРЯДОЧИТЬ ПО Сотрудник
```

В запросе используется ключевое слово **ВЫБРАТЬ**, позволяющее указать список полей для данного запроса.

При запросе к табличной части обращение к обычным (вне таблицы) реквизитам справочника производится через поле **ССЫЛКА**.

Например, `Ссылка.Наименование как Организация`

Ключевое слово **КАК** позволяет определить синоним вместо длинной конструкции `Ссылка.Наименование` использовать короткую - `Организация`.

Ключевое слово **ИЗ** позволяет определить таблицы, участвующие в запросе.

Для упорядочивания данных запроса используется конструкция **УПОРЯДОЧИТЬ ПО**, в которой указываются поля, по которым производится сортировка.

4) перейти на вкладку «Настройки», где в разделе «Выбранные поля» выбрать все интересующие поля (рис. 89), участвующие в запросе (двойной щелчок по имени поля в левом нижнем окне).

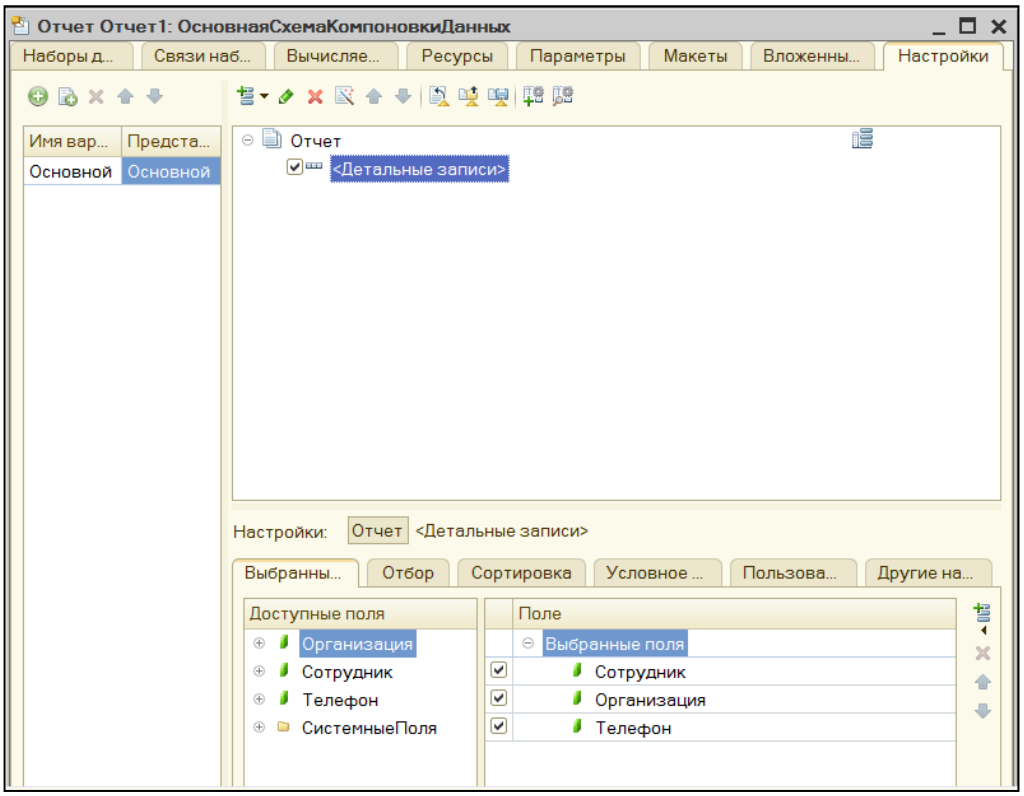


Рис. 89. Формирование полей для схемы компоновки данных

В контекстном меню «Отчет» правой верхней части окна выбрать пиктограмму «Новая группировка», указать тип – «без иерархии».

5) посмотреть результат выполнения запроса в режиме 1С:Предприятие, нажав кнопку «Сформировать» (рис. 90).

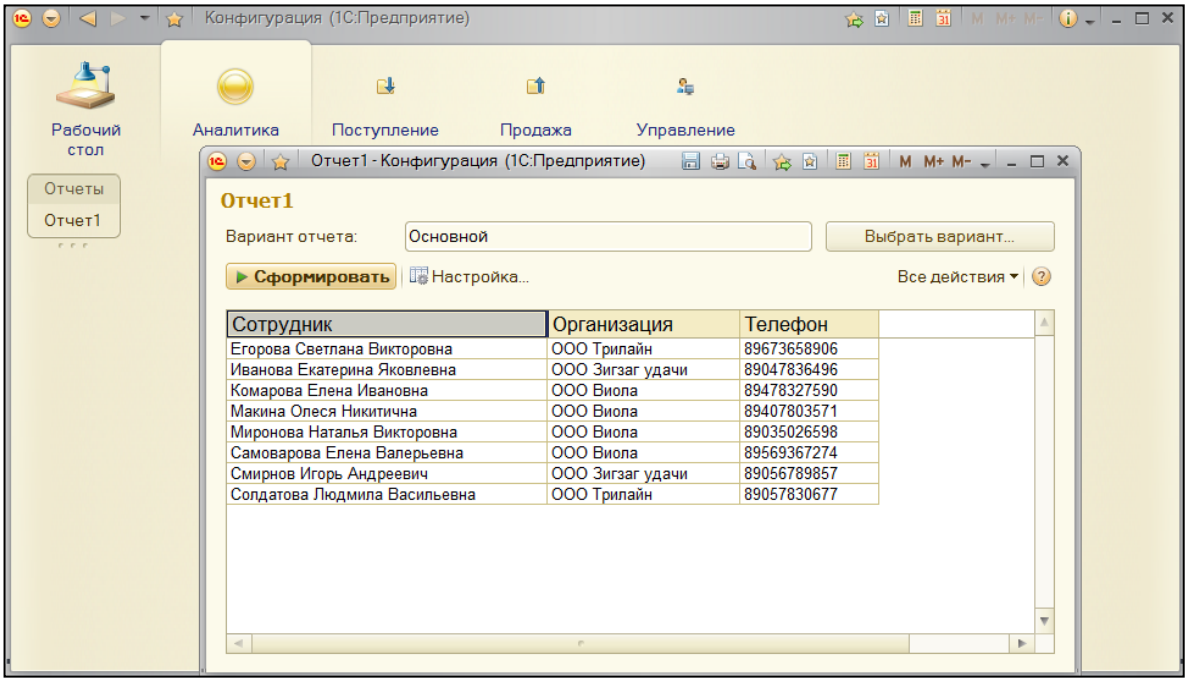


Рис. 90. Отчет «Выборка» в режиме 1С:Предприятие

2.Сортировка упорядочивает записи результата запроса.

Ключевое слово **УБЫВ** в конструкции **УПОРЯДОЧИТЬ ПО** говорит о том, что отобранная информация должна быть отсортирована в порядке убывания значений указанного поля (в нашем примере – номер телефона).

Пример. Создать запрос, который выводит список сотрудников каждой организации с сортировкой по убыванию номера телефона.

Текст запроса:

```
ВЫБРАТЬ Телефон,Сотрудник,
Ссылка.Наименование КАК Организация
ИЗ Справочник.Фирмы.КонтактныеЛица
УПОРЯДОЧИТЬ ПО Телефон УБЫВ
```

На вкладке «Настройки» выбрать поля, отражаемые в запросе, и указать «Детальные записи». Проверить результат работы.

3.Группировка записей отбирает в результат запроса только уникальные комбинации значений.

Ключевое слово **РАЗЛИЧНЫЕ** позволяет отобрать только различающиеся строки.

Пример. Необходимо сформировать перечень товаров, которые поступали от каждого поставщика по документам «ПоступлениеТоваров».

Создать новый запрос, в котором набрать следующий текст:

```
ВЫБРАТЬ РАЗЛИЧНЫЕ Товар.Наименование КАК Товар,
Ссылка.Фирма.Наименование КАК ФирмаПоставщик
ИЗ Документ.ПоступлениеТоваров.ПереченьТоваров КАК
Перечень
УПОРЯДОЧИТЬ ПО Товар ВОЗР
```

На вкладке «Настройки» выбрать поля, отражаемые в запросе, и указать «Детальные записи». Проверить результат работы.

4.Логические операторы в запросе используются для отбора записей по заданным условиям.

Ключевое слово **ГДЕ** позволяет задать условие отбора данных из исходных таблиц.

Пример. Получить список товаров по документам «ПоступлениеТоваров», которых поступило в количестве более 1000 и на сумму более 20000 руб.

Текст запроса:

```
ВЫБРАТЬ Товар, Количество, Сумма,
Ссылка.Филиал КАК Подразделение
ИЗ Документ.ПоступлениеТоваров.ПереченьТоваров
ГДЕ Количество > 1000 И Сумма > 20000
```

На вкладке «Настройки» выбрать поля, отражаемые в запросе, и указать «Детальные записи». Проверить результат работы.

5.Параметрический: позволяет настраивать результат выполнения запроса на параметр, задаваемый пользователем.

Пример. Необходимо получить информацию о документах по поступлениям товаров в определенный филиал не позднее указанной даты.

1) создать запрос и набрать следующий текст:

```
ВЫБРАТЬ Дата, Номер
ИЗ Документ.ПоступлениеТоваров
ГДЕ Дата < &ГраницаДаты И Филиал = &УказанныйФилиал
```

Символ **&** в запросе обозначает параметр. По нашему примеру параметров два – Дата и Филиал.

2) перейти на вкладку «Параметры» и отключить флажки «Ограничения доступности» (рис. 91) для полей, являющихся параметром («ГраницаДаты» и «УказанныйФилиал»).

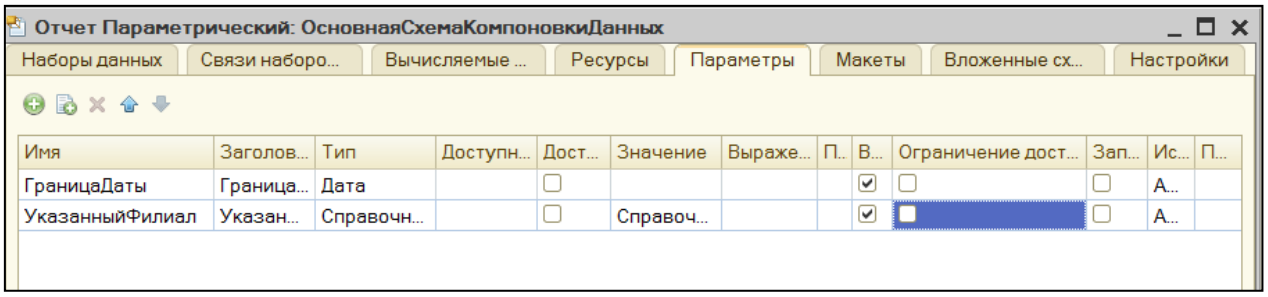


Рис. 91. Вкладка «Параметры» окна конструктора запроса «Параметрический»

3) на вкладке «Настройки» добавить поля, участвующие в запросе и указать «Детальные записи в отчете», чтобы предоставить пользователю возможность устанавливать параметры Дата и Филиал, необходимо перейти на вкладку «Параметры» окна «Настройки» (рис. 92), выделить строку «Границы даты» и в контекстном меню выбрать пункт «Свойства элементов пользовательских настроек», указать вариант «Включать в пользовательские

настройки». Аналогичным образом поступить с параметром «Указанный филиал»;

4) установить флажки «Отображения параметров на экране вовремя запуска запроса» (или нажать пиктограмму «Отметить все элементы»). Проверить работу запроса в режиме 1С.

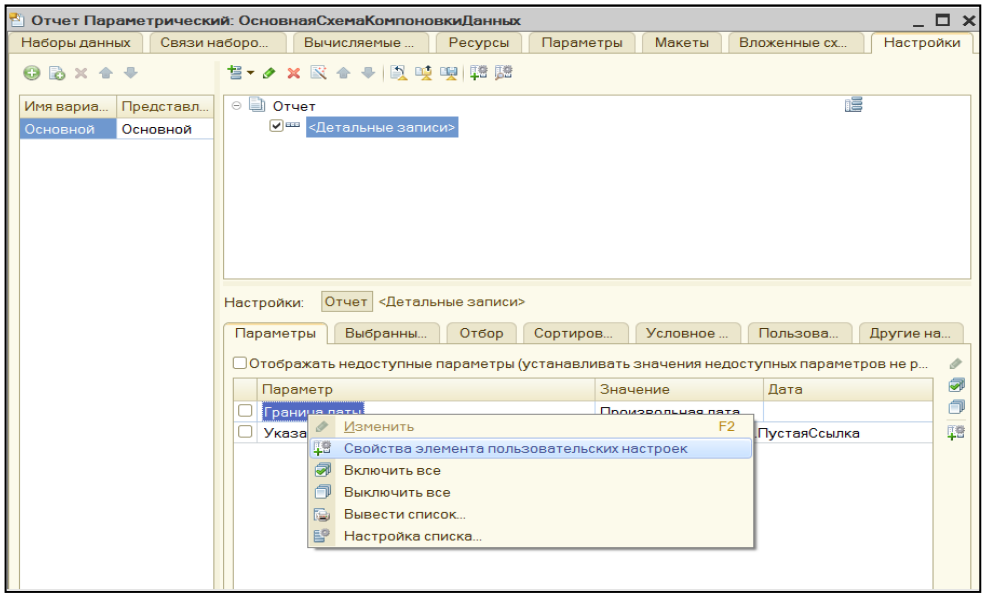


Рис. 92. Дополнительная вкладка «Параметры» окна «Настройки»

6. Интервал позволяет задать границы отбора записей.

Для создания интервалов в языке запросов существует ключевое слово **МЕЖДУ**.

Пример. Вывести все документы «ПродажаТоваров», в которых сумма сделок попадает в указанные пользователем границы.

Перед выполнением запроса следует обозначить значения параметров Сумма1 и Сумма2. В результат запроса попадут только те документы, в которых сумма от продажи товаров соответствует выбранному интервалу.

Для этого:

1) создать запрос со следующим текстом:

```
ВЫБРАТЬ Товар, Количество, Сумма,
Ссылка.Номер КАК НомерДокумента,
Ссылка.Дата КАК ДатаДокумента,
Ссылка.Филиал КАК Подразделение,
Ссылка.Менеджер КАК Специалист
ИЗ Документ.ПродажаТоваров.ПереченьТоваров
ГДЕ Документ.ПродажаТоваров.ПереченьТоваров.Сумма МЕЖДУ
&Сумма1 И &Сумма2
```

2) отключить флажки «Ограничение доступа» на вкладке «Параметры». На вкладке «Настройки» добавить «Детальные записи», «Выбранные поля» (рис. 93), установить флажки «Включать в пользовательские настройки».

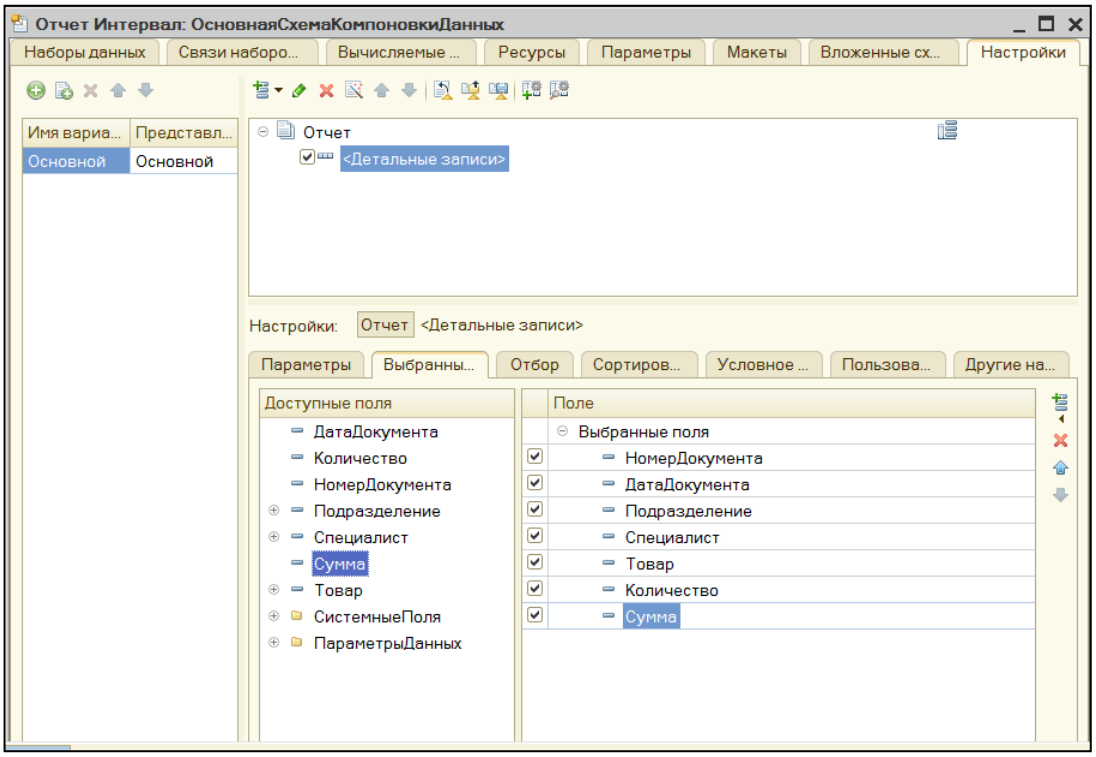


Рис. 93. Вкладка «Настройки» окна конструктора запроса «Интервал»

Сохранить все изменения конфигурации. Проверить работу запроса в режиме 1С:Предприятие.

7.Агрегатные функции используются для получения разнообразной сводной информации по объектам конфигурации.

Пример. По документам «ПоступлениеТоваров» могут быть поставлены следующие вопросы:

- какова сумма поступлений всех товаров?
- каково среднее количество поступлений каждого товара?
- какова наибольшая сумма поступлений по каждому товару?

Для реализации таких вопросов используются агрегатные функции. Любая агрегатная функция принимает в качестве аргумента какой-либо столбец табличных данных, а возвращает единственное значение. Агрегатная функция предназначена для обобщения значений указанного столбца.

Наиболее часто используются следующие функции:

СУММА() – вычисляет сумму всех значений, содержащихся в указанном столбце. В качестве параметра задаются поля с числовыми данными;
МАКСИМУМ() – находит наибольшее значение в указанном столбце;
МИНИМУМ() – находит наименьшее значение в указанном столбце;
СРЕДНЕЕ() – вычисляет среднее арифметическое в указанном столбце;
КОЛИЧЕСТВО() – подсчитывает количество значений, содержащихся в указанном столбце.

Свойства:

- подсчитываются значения, не равные *NULL*;

- для подсчета различных значений указанного поля, на равных *NULL*, используется ключевое слово *РАЗЛИЧНЫЕ*;

- для подсчета количество строк в результате запроса используется в качестве параметра функции символ *****.

Пример. Для документов «ПоступлениеТоваров» по каждому виду товара подсчитать максимальное, минимальное, среднее значение.

Создать новый запрос, набрать следующий текст:

```
ВЫБРАТЬ Товар,
Сумма(Сумма) КАК СуммаПоТовару,
МАКСИМУМ(Сумма) КАК МаксСумма,
МИНИМУМ(Сумма) КАК МинСумма,
СРЕДНЕЕ(Сумма) КАК СрСумма
ИЗ Документ.ПоступлениеТоваров.ПеречеьТоваров
СГРУППИРОВАТЬ ПО Товар
```

Конструкция **СГРУППИРОВАТЬ ПО** языка запросов группирует («накладывает») одинаковые записи столбца таблицы.

На вкладке «Настройки» выбрать поля, отражаемые в запросе, и указать «Детальные записи». Проверить результат работы.

8.Внутреннее соединение таблиц показывает имеющиеся одинаковые записи таблиц, используемых в запросе.

Ключевое слово **СОЕДИНЕНИЕ** позволяет отобрать только одинаковые записи из документов.

Пример. Выбрать из документов «ПоступлениеТоваров» и «ПродажаТоваров» одноименные товары, имеющиеся в базе.

Текст запроса:

```
ВЫБРАТЬ РАЗЛИЧНЫЕ
ПеречеьПоступлений.Товар КАК ПоступившийТовар,
ПеречеьПродаж.Товар КАК ПроданныйТовар
ИЗ Документ.ПоступлениеТоваров.ПеречеьТоваров КАК
ПеречеьПоступлений
СОЕДИНЕНИЕ
Документ.ПродажаТоваров.ПеречеьТоваров КАК
ПеречеьПродаж
    ПО ПеречеьПоступлений.Товар = ПеречеьПродаж.Товар
```

На вкладке «Настройки» выбрать поля, отражаемые в запросе, и указать «Детальные записи». Проверить результат работы.

9.Полное соединение таблиц содержит комбинации записи из обеих исходных таблиц, которые соответствуют указанному условию. Кроме того, в результат запроса включаются и те записи из обоих источников, для которых не найдено соответствий. Таким образом, в результат запроса включаются все записи из обоих источников и соединяются друг с другом при выполнении указанного условия.

Пример. Необходимо отобрать фирмы, которые выступали как в качестве продавцов, так и в качестве покупателей. При этом в результат необходимо включить и те фирмы, которые выступали только в одном качестве (или продавец, или покупатель).

Текст запроса:

```
ВЫБРАТЬ РАЗЛИЧНЫЕ
Продажа.Фирма.Наименование КАК ФПокупатель,
Поступление.Фирма.Наименование КАК ФПродавец
ИЗДокумент.ПродажаТоваров КАК Продажа
ПОЛНОЕ СОЕДИНЕНИЕ
Документ.ПоступлениеТоваров КАК Поступление
ПО
Продажа.Фирма.Наименование=Поступление.Фирма.Наименование
```

На вкладке «Настройки» выбрать поля, отражаемые в запросе, и указать «Детальные записи». Проверить результат работы.

15. Регистры накопления в 1С:Предприятие

Регистры накопления предназначены для реализации механизма количественного учета информации. Вся информация, которая вводится с помощью документов, должна быть зафиксирована в регистрах накопления.

Регистр накопления представляет собой объект конфигурации, предназначенный для хранения движений – изменений, которые в нем происходят при проведении документа. Для работы с регистром накопления в первую очередь необходимо определить его структуру – в каких разрезах следует накапливать данные с целью последующего максимального эффективного извлечения информации.

Регистры накопления могут быть двух видов – *остатков* и *оборотов*.

Для ***регистров остатков*** система 1С:Предприятие создает несколько виртуальных таблиц – Таблицу остатков, Таблицу оборотов и совместную Таблицу оборотов и остатков. У регистра остатков присутствует параметр Вид движения, который для каждой записи в регистре устанавливает параметр *Приход* или *Расход* в зависимости от операции.

Для ***регистров оборотов*** система 1С создает только одну виртуальную Таблицу оборотов.

Данные, хранящиеся в регистрах накопления, подразделяются на *измерения* и *ресурсы*. Накопление количественной информации может производиться в разрезе одного или нескольких ***измерений***, которые описываются при разработке конфигурации. Например, номенклатура товара или фирма.

Виды числовой информации, аккумулируемой регистром накопления, называются ***ресурсами***. Так регистр накопления может накапливать информацию о количестве различных товаров на фирме.

Изменение состояния регистра накопления происходит при проведении документа. В этом случае в регистр добавляется одна или несколько записей.

Кроме измерений и ресурсов, у регистра накопления может быть создан набор реквизитов (например, фамилия человека, который продал товар, и т.д.).

1. Регистр остатков «КоличествоТоваров» обеспечивает накопление информации о количестве товаров как результат их поступления и продажи.

Для этого:

1) в окне Конфигурация щелкнуть правой кнопкой мыши на значке «Регистры накопления», в контекстном меню выбрать пункт «Добавить»;

2) в окне редактирования объекта конфигурации на вкладке «Основные» указать имя создаваемого регистра – «КоличествоТоваров», вид регистра – «Остатки»;

3) на вкладке «Данные» создать:

- измерения регистра:

- Товар (тип – СправочникСсылка.Товары);
- Филиал (тип – Справочникссылка.Филиалы);

- ресурс:

- Количество (тип – число, точность – 2).

В итоге должно получиться (рис. 94):

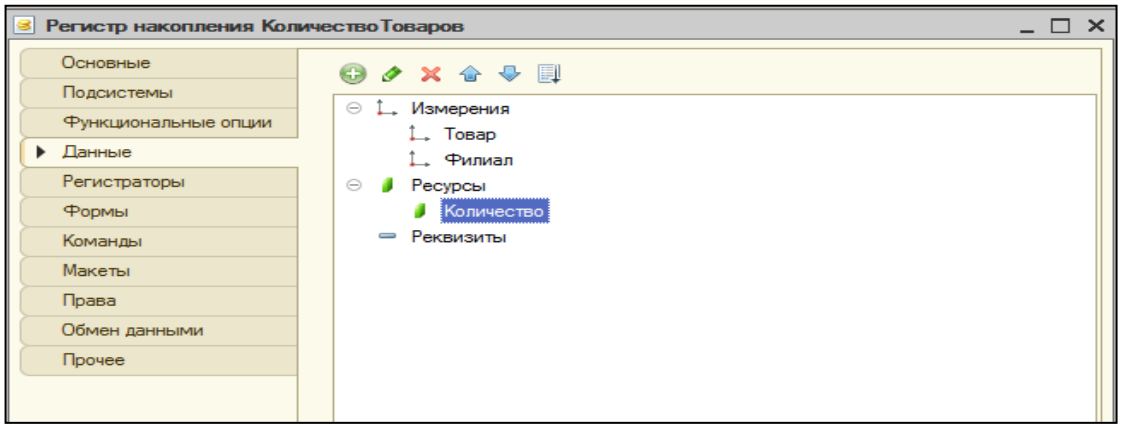


Рис. 94. Измерения и ресурсы регистра «КоличествоТоваров»

4) использование регистра накопления без документа, который бы совершал движения по данному регистру, невозможно. «Подцепим» регистр «КоличествоТоваров» к документу «ПоступлениеТоваров»:

a) у документа «ПоступлениеТоваров» на вкладке «Движение» (рис. 95) выбрать в списке регистров «КоличествоТоваров».

b) щелкнуть по кнопке «Конструктор движений».

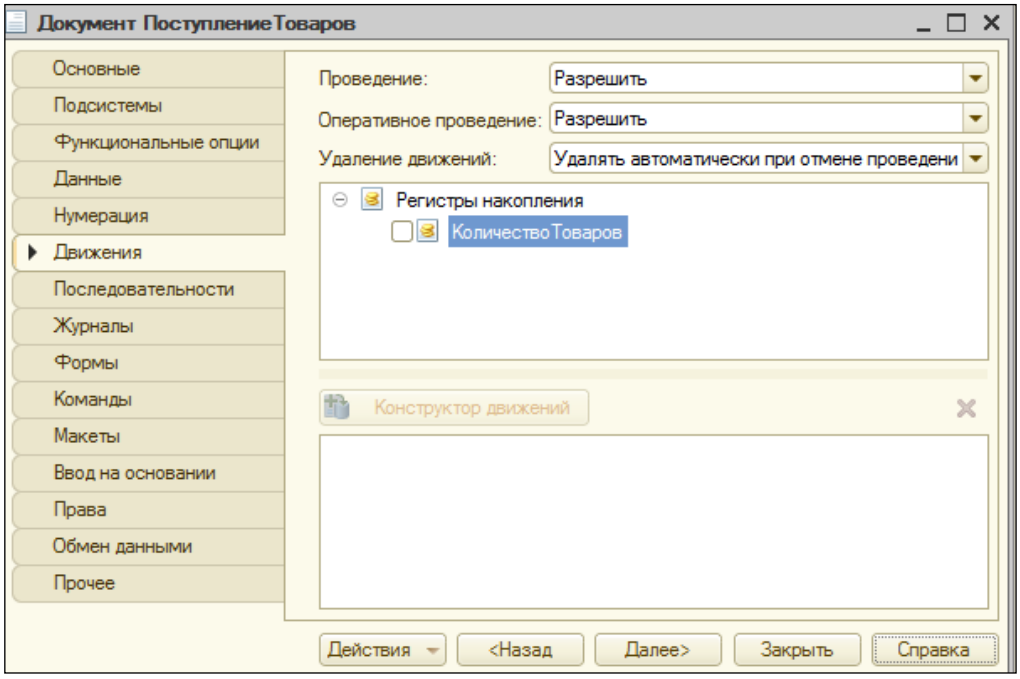


Рис. 95. Вкладка «Движение» документа «ПоступлениеТоваров»

Окно *Конструктора движения регистров* (рис. 96) состоит из трех частей:

- регистры – перечислены регистры, по которым документ может создавать движения;
- реквизиты документа – первоначально содержатся реквизиты шапки документа «ПоступлениеТоваров». Для добавления реквизитов табличной части следует в центральной части окна из раскрывающегося списка Табличная часть выбрать имя табличной части документа – «ПеречеьТоваров».
- нижняя часть – расположены поля регистра накопления «КоличествоТоваров». Для указания соответствия полей регистра и реквизитов документа необходимо щелкнуть мышкой по кнопке «Заполнить выражения». В результате, напротив названий полей регистра сформируются соответствующие выражения.

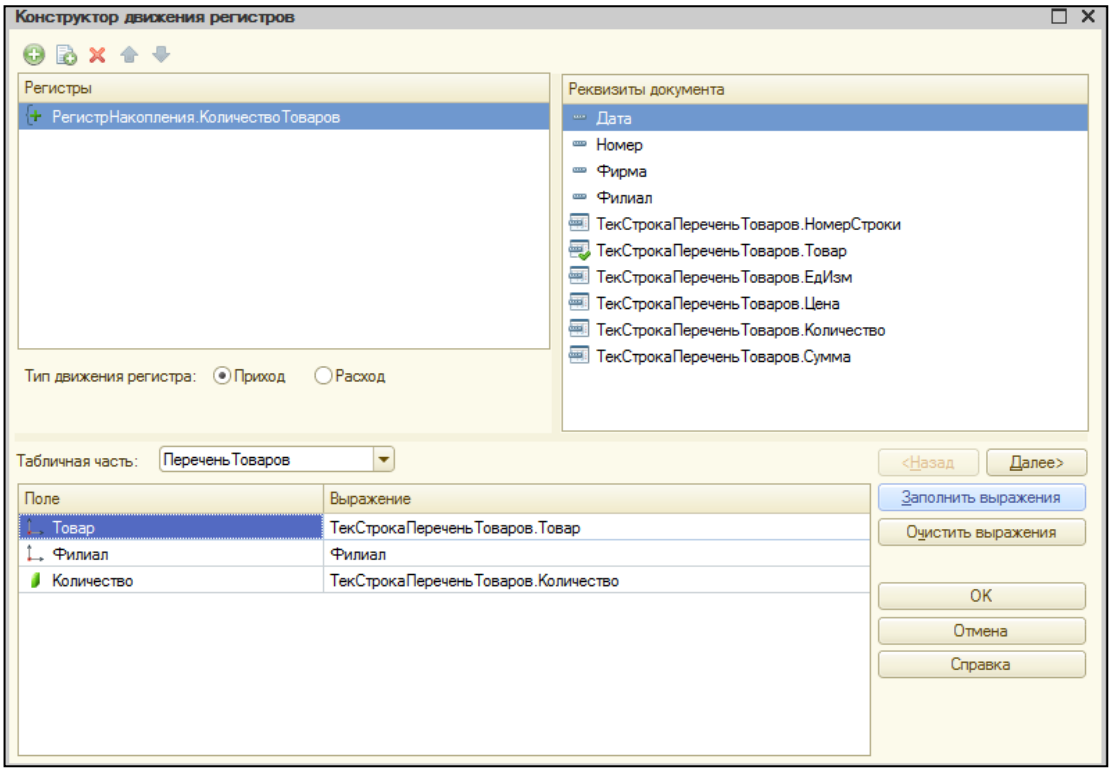


Рис. 96. Окно конструктора движений регистров

c) в окне «Конструктора движений регистров» указать «Тип движения регистра» – Приход;

d) щелкнуть по кнопке «Ок».

5) для включения регистра накопления «КоличествоТоваров» к подсистеме «Регистры» необходимо установить флажок командного интерфейса в контекстном меню подсистемы как показано на рис. 97.

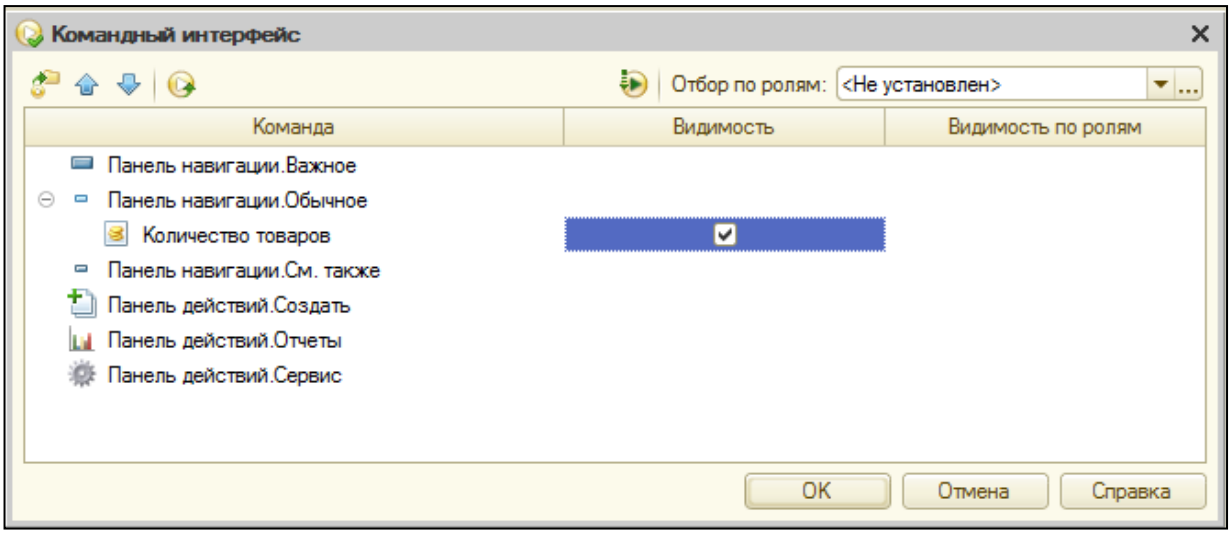


Рис. 97. Окно командного интерфейса подсистемы «Регистры»

Запустите режим работы 1С:Предприятие. Добавьте новый документ «Поступление товара». Посмотрите содержание регистра «Количество товаров».

2.Регистр оборотов ПродажиПоФилиалам позволяет настроить учет продаж по филиалам.

Для этого:

1) создать новый регистр накопления с именем «ПродажиПоФилиалам»;

2) на вкладке «Основные» окна редактирования объекта конфигурации установить имя и вид регистра накопления – «Обороты»;

3) на вкладке «Данные» ввести:

- измерения:
  - Филиал (тип – СправочникСсылка.Филиалы);
  - Менеджер (тип – СправочникСсылка.Менеджеры);
- ресурс:
  - Сумма (тип – число, точность – 2);

4) для организации движения в документе «ПродажаТоваров» сразу по двум регистрам «КоличествоТоваров» и «ПродажиПоФилиалам» на вкладке «Движение» в окне «Регистры накопления» установить флажок для второго регистра – «ПродажиПоФилиалам» (рис. 98):

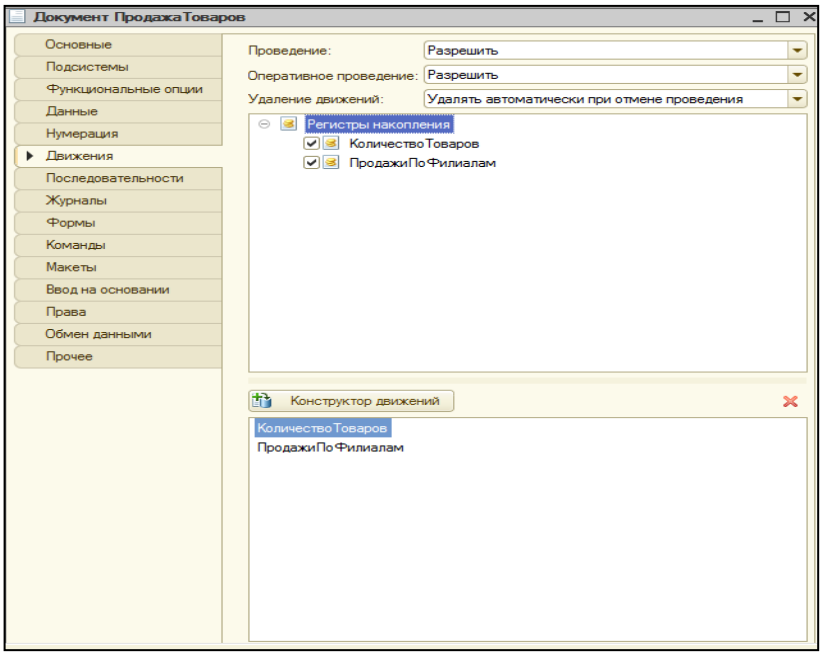


Рис. 98. Организация движения учета по двум регистрам у документа «ПродажаТоваров»

5) в окне «Конструктор движения регистров» в левом верхнем окне «Регистры» добавить регистр «ПродажиПоФилиалам», выбрать табличную часть и заполнить выражения полей для данного регистра (рис. 99):

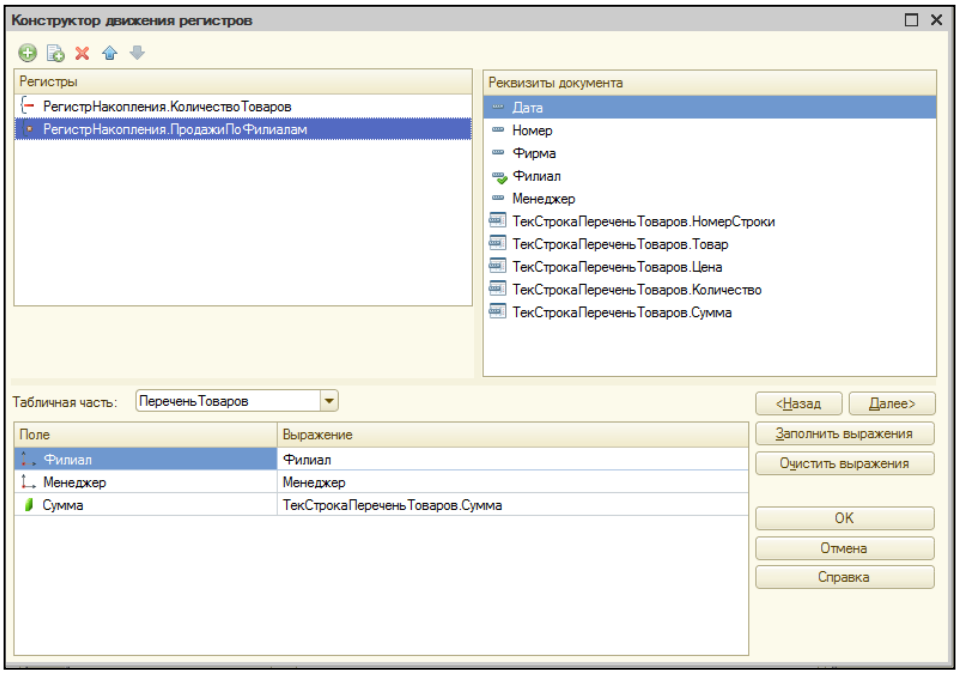


Рис. 99. Окно конструктора движения регистров документа «ПродажаТоваров»

6) отнести созданный регистр к подсистеме – «Регистры», включить флажок командного интерфейса.

Для проверки работы в режиме 1С:Предприятие добавьте новый документ «Продажа товаров» и посмотрите содержание регистра «ПродажиПоФилиалам».

Самостоятельно «прицепить» регистр накопления «КоличествоТоваров» к документу «ПродажаТоваров» («Тип движения регистра» – Расход). Проверить работу регистра.

16. Схема компоновки данных

При решении любых задач, связанных с количественным учетом, ключевым и одновременно итоговым результатом являются отчеты.

С помощью отчетов пользователь может получать необходимые ему выходные данные. Алгоритм формирования выходных данных описывается при помощи визуальных средств или с использованием встроенного языка. В

реальной жизни объектами конфигурации «Отчет» соответствуют всевозможные таблицы выходных данных, сводных данных, диаграмм и т.д.

Система компоновки данных (СКД) предназначена для создания отчетов с использованием построителя отчетов, автоматизирующего процесс создания.

Характерные черты СКД:

- создание отчета без программирования;
- использование автоматически генерируемых форм просмотра и настройки отчета;
- настройка структуры отчета;
- совмещение в отчете несколько таблиц.

Основной этап разработки отчета называется созданием схемы компоновки данных, который выполняется с использованием конструкторов. Следующие этапы процесса компоновки данных связаны с редактирование параметров компоновки данных, формирование макета компоновки данных, исполнение и вывод результата компоновки данных.

Все создаваемые отчеты будут относиться подсистеме «Отчеты».

1. Отчет по остаткам товаров

Пример. Необходимо построить отчет, включающий в себя информацию о количестве остатков каждого товара в филиалах. Для получения результата используется регистр накопления «КоличествоТоваров».

Для этого:

1) создать новый объект конфигурации типа – «Отчет», указать имя – «Остатки»;

2) нажать кнопку «Открыть схему компоновки данных» на вкладке «Основные»;

3) сохранить настройки системы, установленные по умолчанию, нажать кнопку «Готово»;

4) в окне Конструктора схемы компоновки данных с помощью кнопки «Добавить набор данных» выбрать вариант – запрос;

5) щелкнуть по кнопке «Конструктор запроса…», откроется окно «Конструктор запроса» (рис. 100):

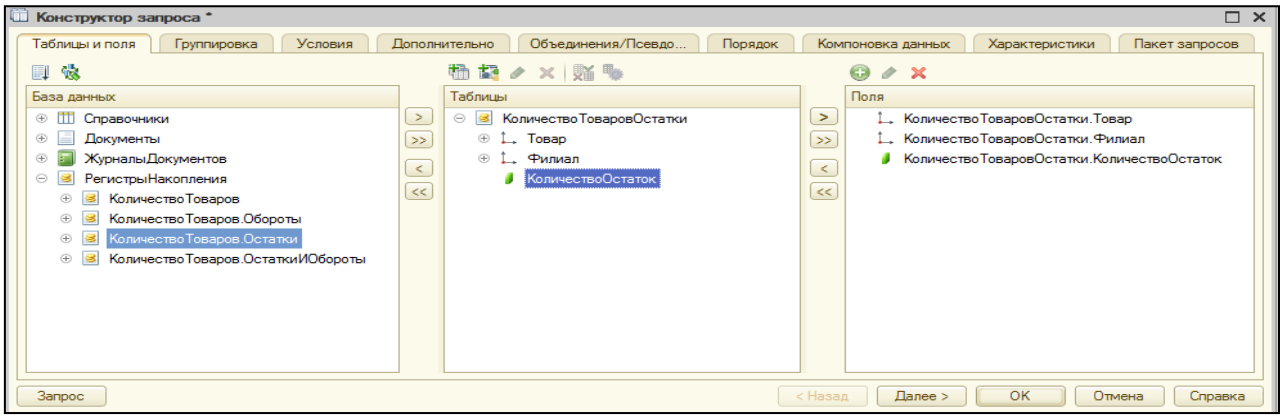


Рис. 100. Окно Конструктора запросов

В этом окне присутствуют три раздела:

- база данных – представлены имеющиеся таблицы для формирования запросов и разнообразных отчетов;
- таблицы – состав реквизитов объектов конфигурации;
- поля – реквизиты объектов, выбранные для построения.

6) по нашему примеру необходимо выбрать регистр накопления «КоличествоТоваров.Остатки», внести все три поля этого регистра в раздел «Поля» (рис. 100);

7) на вкладке «Порядок» (рис. 101) Конструктора запросов указать вид сортировки - «Убывание» по «КолиствоОстатков». Нажать «Ок», в результате текст запроса формируется автоматически;

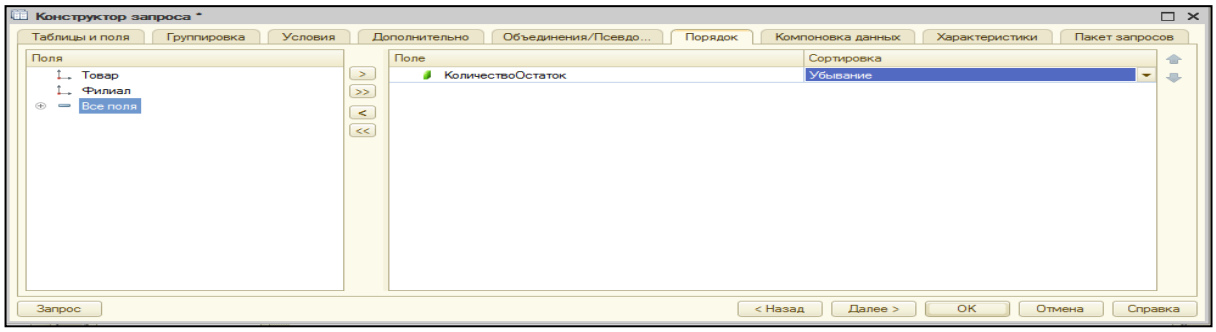


Рис. 101. Вкладка «Порядок» окна Конструктора запросов

8) на вкладке «Настройки» ОсновнойСхемыКомпоновкиДанных указать «Детальные записи» в окне «Отчет» и выбрать поля в окне «Настройки» (рис. 102):

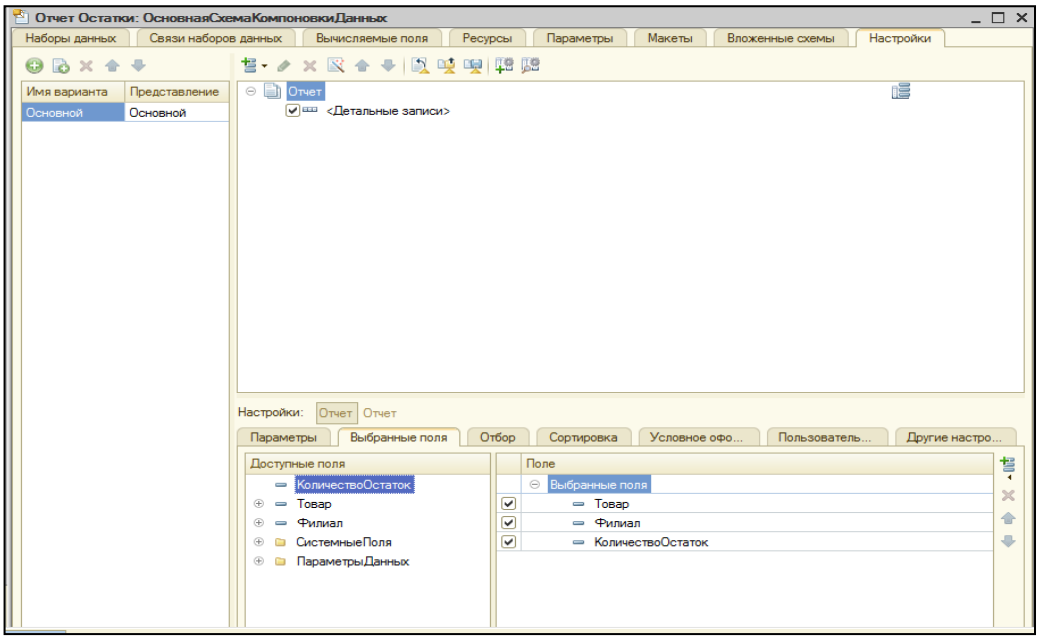


Рис. 102. Окно редактирования отчета «Остатки»

Проверить результат работы отчета в режиме 1С.

2. Отчет по работе филиалов

Пример. Создать отчет, который позволяет получить информацию об уровне продаж товаров в филиалах за указанный период.

Для этого:

1) создать схему компоновки данных с именем «РаботаФилиалов» и новый набор данных – запрос;

2) в окне Конструктора запроса выбрать таблицы и поля:

- справочник «Филиалы» (поле – «Филиалы.Ссылка»);
- виртуальная таблица «ПродажиПоФилиаламОбороты» (поле – «ПродажиПоФилиаламОбороты.СуммаОборот»);

3) на вкладке «Связи» (рис. 103) снять флажок «Все» у «ПродажиПоФилиаламОбороты» и установить флажок «Все» для таблицы справочника «Филиалы», чтобы для каждой записи справочника были поставлены в соответствии значения оборота по данному филиалу из виртуальной таблицы регистра;

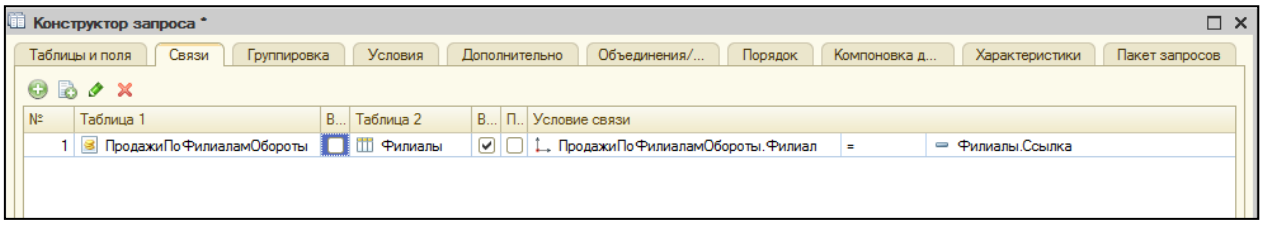


Рис. 103. Вкладка «Связи» окна конструктора

4) на вкладке «Объединения/Псевдонимы» (рис. 104) установить псевдоним «НашФилиал» вместо сочетания «Филиалы.Ссылка»:

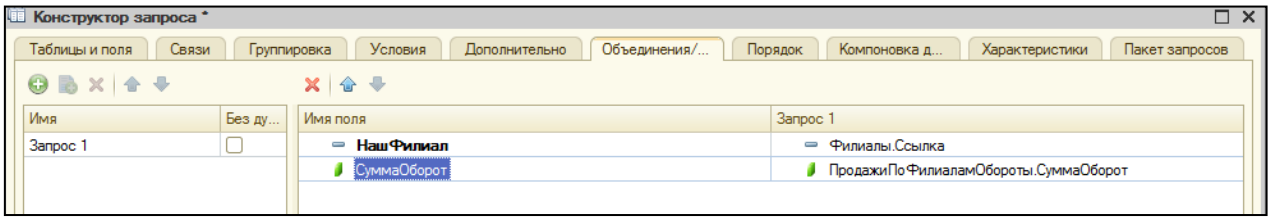


Рис. 104. Вкладка «Объединения/Псевдонимы» окна конструктора

5) на вкладке «Порядок» указать, чтобы результат был отсортирован по значению поля СуммаОборот в порядке убывания. Нажать «Ок».

6) на вкладке «Ресурсы» (рис. 105) первоначального окна «Схемы компоновки данных» включить в список ресурсов поле СуммаОборот:

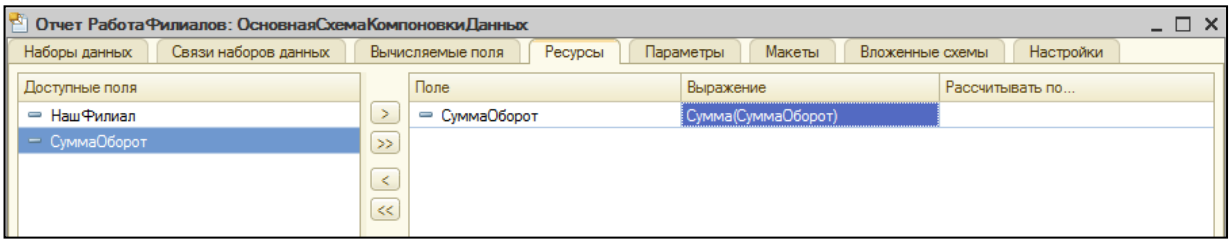


Рис. 105. Вкладка «Ресурсы» отчета «РаботаФилиалов»

7) на вкладке «Параметры» система 1С автоматически добавила параметры НачалоПериода и КонецПериода, анализируя текст запроса.

Тип «Дата», кроме непосредственной даты, содержит время. Чтобы удалить время в значении даты, необходимо дважды щелкнуть в графе «Тип» параметра НачалоПериода и с помощью кнопки с многоточием открыть окно «Редактирование типа данных» (рис. 106), где в строке «Состав даты» выбрать вариант «Дата».

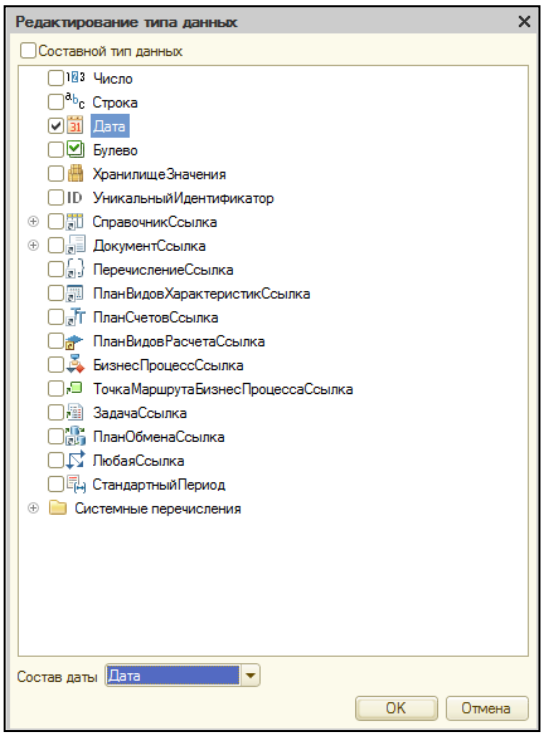


Рис. 106. Окно редактирования типа данных

Аналогичным образом скорректировать значение задаваемого параметра КонецПериода.

8) на вкладке «Настройки» окна отчета:
   - добавить «Детальные записи»;
   - выбрать поля для отображения в отчете – НашФилиал и СуммаОборот;
   - установить у параметров «НачалоПериода» и «КонецПериода» флажок «Включать в пользовательские настройки»;

Проверить результат работы.

3. Отчет по работе менеджеров (с графиком)

Пример. Создать отчет в виде графика, отражающего сравнение результатов работы менеджеров. Каждая строка будет содержать фамилию менеджера и сумму продаж товаров, которые он совершил. Если менеджер за указанный период продаж не совершил, его фамилия на диаграмме присутствовать будет, но с минимальным значением функции.

Для этого:

1) создать новый отчет с именем «РаботаМенеджеров»;

2) на вкладке конструктора «Схемы компоновки данных» выбрать вариант «Добавить набор данных - запрос»;

3) в окне Конструктора запроса выбрать следующие таблицы и поля:

- справочник «Менеджеры» (поле – Менеджеры.Ссылка);
- виртуальная таблица «ПродажиПоФилиаламОбороты» (поле – СуммаОборот);

4) на вкладке «Связи» установить флажок «Все» только для таблицы справочника «Менеджеры»;

5) на вкладке «Объединения/Псевдонимы» заменить конструкцию «Ссылка» на «МенеджерыФирмы» для понятного представления. Щелкнуть по «Ок»;

6) в окне «Схемы компоновки данных» на вкладке «Ресурсы» выбрать ресурс СуммаОборот;

7) на вкладке «Параметры» установить формат НачалоПериода и КонецПериода без указания значения времени;

8) на вкладке «Настройки» первоначального окна конструктора:

- в верхнем правом окне «Отчет» щелкнуть по пиктограмме с изображением «волшебной палочки» - «Открыть конструктор настроек»:

Шаг 1. Выбрать тип отчета - «Диаграмма». Нажать кнопку «Далее».

Шаг 2. Выбрать поля, отображаемые в отчете – НашиМенеджеры и СуммаОборот. Нажать кнопку «Далее».

Шаг 3. Выбрать поле НашиМенеджеры в качестве группировки точек диаграммы. Нажать кнопку «Далее».

Шаг 4. Порядок упорядочивания данных в отчете выберем – «По возрастанию» для поля Менеджеры.

Шаг 5. Выбрать тип диаграммы – «Гистограмма объемная». Нажать «Ок».

- в окне «Настройки» выбрать поля НашиМенеджеры и СуммаОборот для отображения в запросе, установить у параметров НачалоПериода и КонецПериода флажок «Включать в пользовательские настройки».

В итоге получим (рис. 107):

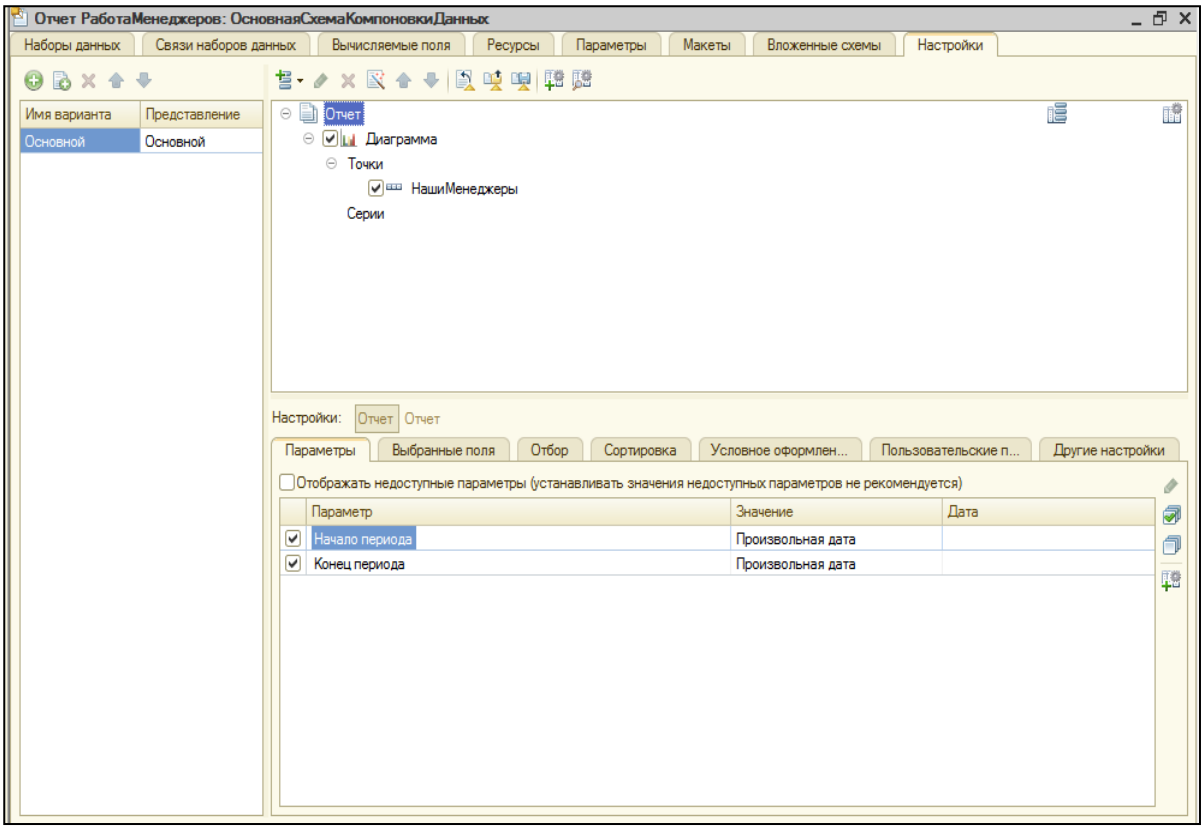


Рис. 107. Вкладка «Настройки» отчета «Работа менеджеров»

Сохранить изменения и проверить работу отчета «РаботаМенеджеров» в режиме 1С.

Таким образом, на примере поставленной задачи были рассмотрены вопросы создания конфигурации, программирования основных действий, формирования запросов и отчетов, позволяющие руководству оценить эффективность работы менеджеров. Полученные знания можно использовать для проектирования собственных задач.

Вопросы для самопроверки

1. Что такое конфигурируемость системы 1С:Предприятие.
2. Из каких основных частей состоит система.
3. Для чего используются разные режимы запуска системы 1С:Предприятие.
4. Что такое конфигурация.
5. Какие объекты конфигурации существуют.
6. Что создает система на основе объектов конфигурации.
7. Как добавить новый объект конфигурации.

8. Для чего используется объект конфигурации Подсистема.

9. Как описать логическую структуру конфигурации при помощи объектов Подсистема.

10. Для чего предназначен объект конфигурации Справочник.

11. Каковы характерные особенности справочника.

12. Для чего используются реквизиты и табличные части справочника.

13. Какие основные формы существуют у справочника.

14. Как добавить объект конфигурации Справочник.

15. Как добавить новые реквизиты в справочник.

16. Какими характерными особенностями обладает объект конфигурации Документ.

17. Какие существуют основные формы документа.

18. Для чего предназначены реквизиты и табличные части документа.

19. Что такое обработчик события и как его создать.

20. Для чего предназначен объект встроенного языка Запрос.

21. Для чего предназначена система компоновки данных.

22. Для чего предназначена настройка компоновки данных.

23. Из каких частей состоит текст запроса, какие из них являются обязательными.

24. Какие объекты могут использоваться в качестве источника данных запроса.

25. Что такое параметры запроса.

26. Как упорядочить данные в запросе.

27. Как создать отчет, содержащий диаграмму.

28. Для чего предназначены объект Регистры.

29. Какие виды регистров накопления существуют.

30. Что такое ресурсы и измерения регистров.

Литература

1. Кашаев С. М. Программирование в 1С:Предприятие:8.2. – СПб.: Питер, 2011. – 272 с.: ил.
2. Кашаев С.М. 1С:Предприятие 8.2. Программирование и визуальная разработка на примерах. – СПб.: БХВ–Петербург, 2011. – 320 с.: ил.

3. Радченко М.Г. 1С:Предприятие 8.2. Практическое пособие разработчика. Примеры и типовые приемы/ М.Г. Радченко, Е.Ю. Хрусталева. – М.:ООО «1С-Паблишинг», 2009.-872с.: ил.
4. Крайнова Т.С. Предметно-ориентированные экономические информационные системы. Методические указания по выполнению лабораторно-практического цикла студентов.- Екатеринбург: Уральский государственный лесотехнический университет, 2012. – 54 с.

Организация хранения и обработки информации

Существуют два принципиально противоположных подхода к использованию компьютеров:

1*) централизованный* - создание возможно большого компьютера, который будет накапливать, обрабатывать всю информацию и организовывать доступ пользователей к нему;

2) *децентрализованный* - организация многих банков данных и множества территориально рассредоточенных компьютеров, которые будут обрабатывать информацию, и создание системы связи между ними.

Рассмотрим два этих подхода с позиций удобства для бизнеса.

Сейчас, в эпоху информационной революции экономика становится глобальной. В бизнесе принимают участие миллионы фирм, предприятий, магазинов и т.д., которые распределены по всему миру. Так как пользователи географически рассредоточены, то с этой позиции более предпочтительным является второй подход - децентрализация баз данных и компьютеров и совершенствование связи между ними.

Тем не менее, оценим также централизованный и децентрализованный подходы с технико-экономической позиции создания компьютеров и компьютерных сетей.

Заменят ли персональные компьютеры большие универсальные вычислительные машины? Если да, то почему этого не случилось до сих пор? Где место клиентов, серверов и сетей в общей картине? Что все это означает? Это только новая технология или начало еще более фундаментальных изменений?

Исторически так сложилось, что построение систем бизнеса было простым делом: поставь большой компьютер, подключи терминалы и наращивай возможности своего большого компьютера для удовлетворения возрастающих требований. С появлением мини-компьютеров стали говорить о новом типе распределенных систем: подключайся к компьютеру с удаленного места своей работы и разделяй нагрузку между многими маленькими компьютерами вместо того, чтобы загрузить все в один громадный компьютер. Эта концепция показалась привлекательной, но к великой досаде пользователей многие попытки на протяжении последних 20

лет заставить ее работать не завершились успехом, и большинство крупных систем бизнеса сегодня по-прежнему сильно централизовано. Однако новые сверхмощные персональные компьютеры вышли на передовую. Со всеми этими персональными компьютерами, серверами и локальными сетями распределенные системы стали даже привлекательнее, чем в прошлом. Но смогут ли люди строить реально работающие распределенные системы? И если да, то когда и как?

Удивительно тяжело построить действительно большой компьютер. Большие компьютеры стоят миллионы долларов, а суперкомпьютеры могут стоить многие миллионы. Однако ни один из них не является в тысячи раз быстрее, чем персональные компьютеры. Они действительно стоят в тысячи раз дороже, чем персональный компьютер, но определенно не делают в тысячи раз больше работы. Проблема заключается в том, что цена компьютеров растет гораздо быстрее, чем мощность, заключенная в них. Другая проблема еще сложнее: даже с неограниченным бюджетом существует реальный верхний предел для мощности самого большого компьютера.

Авиалинии ежедневно сталкиваются с этой проблемой. Центральная система резервирования билетов для авиалиний выполняет всю работу на единственном компьютере. Рационально, чтобы пассажир в любом аэропорту мира мог заказать билет в любой другой аэропорт в течение не более 24 часов. Поэтому имеет смысл держать всю информацию о полетах, наличии посадочных мест и количестве багажа в одном компьютере. Как же центральный компьютер сможет справиться с потоком заявок, хлынувших на него со всего мира? - С большим трудом.

Много лет назад самые крупные авиакомпании поняли, что такой объем информации будет большой проблемой для их централизованной системы резервирования билетов. Столкнувшись с этим, они решили писать свои приложения специальным способом, который выжимает даже самую последнюю каплю мощности из их перегруженного большого компьютера.

В результате самые крупные авиалинии склонны заключать договоры с основными производителями больших компьютеров. Как только новейшие, более емкие и быстрые, большие компьютеры готовы, компания-производитель снабжает их подходящими замечаниями по эксплуатации и

сразу передает авиалиниям. Цена не является проблемой: все, что может помочь авиалиниям увеличить скорость их числовых мельниц, используется немедленно, как только появляется.

Но и авиалинии - далеко не исключение. Многие важнейшие вычислительные проблемы не имеют простого решения, так как не существует компьютера, достаточно быстрого, чтобы выполнить вычисления. Например, моделирование погоды для больших пространств будет настолько точным, насколько может быть велики совокупность данных, описывающих модель и содержащихся одновременно в компьютере. Поддержка трасс перемещений воздушных масс, температурных изменений и распределения давления по поверхности любого значительного участка Земли быстро переполняет емкость даже самых больших суперкомпьютеров.

Помимо этого существовали экономические причины развития больших компьютеров в разных областях. На протяжении многих лет в компьютерном мире фирма IBM не только доминировала, но фактически была владельцем компьютерного рынка, занимая 80% его. Херб Гросх (редактор ComputerWord) заметил, что модель ценообразования IBM была очень привлекательной для пользователей, покупавших самые большие компьютеры. В 60-70-х годах вычислительная скорость компьютеров была прямо пропорциональна квадрату их стоимости. Другими словами, израсходуете на компьютер в два раза больше и получите скорость в четыре раза больше, израсходуете в три раза больше - будете иметь скорость в девять раз больше, и т.д. Это и есть *закон Гросха*.

Но, кроме скорости работы, существует еще такой важный показатель мощности, как пропускная способность. Пропускная способность - обработка такого большого количества задач в любой заданный момент времени, какое только возможно в течение заданного промежутка времени. Однако, обеспечивая адекватную пропускную способность, большие универсальные компьютеры сталкиваются с проблемой транзакций реального времени, а также с проблемами пакетной обработки.

В деловых приложениях *транзакции реального времени* - это относительно небольшие задачи, которые выполняют запросы процессов бизнеса непосредственно вслед за их появлением. Транзакции могут передвигать товары в описи с одного места склада в другое, перемещать

фонды между банковскими счетами, бронировать билеты на самолет или инициировать установку коробки на погрузчик. Каждая из этих задач сама по себе включает очень незначительную компьютерную обработку. Однако в типичной большой организации сотни таких маленьких транзакций могут происходить каждую секунду. Таким образом, транзакции подобны синапсам (связующим окончаниям нервных клеток нашего мозга) для систем бизнеса.

Точно так же база данных (БД) подобна центральной памяти организации. В управлении системой БД транзакция определяется как серия операций, которые в случае успешного завершения всегда оставляют БД в согласованном состоянии. Почему так важно и трудно выполнять множество транзакций до полного завершения? Обычно компьютеры и программы работают без прерывания. Однако иногда может пропасть электроэнергия в сети, проявиться ошибка в программных средствах, произойти аппаратный сбой или оператор даст ошибочную команду, которая приведет к остановке компьютера. Независимо от этих причин операционная система большого универсального компьютера все-таки гарантирует правильное выполнение всех транзакций. Она достигает этого достаточно большими усилиями.

Силы, которые сделали персональные компьютеры могущественными и доступными практически каждому, фактически аннулировали (хотя и не отменили) действие закона Гросха. Это аннулирование все же ограничено в некоторых отношениях: если вы тратите больше на компьютер, вы можете получить больше вычислительной скорости, но его стоимость не будет пропорциональна увеличению мощности. В свое время затрачивали в два раза больше средств, чтобы получить в четыре раза большую мощность, что обусловливало выгодность этой сделки. Сегодня увеличение затрат в 100 раз дает увеличение мощности в 10 раз, что оставляет неприятный осадок от такой покупки. Но есть ли тут реальный выбор?

Действительно ли большой универсальный компьютер так велик? Большой универсальный компьютер - это, по существу, целая группа различных меньших компьютеров, упакованных в одну коробку и соединенных вместе с помощью внутренней, высокоскоростной сети, называемой шиной. Если большой универсальный компьютер - это связка нескольких компьютеров, исполняющих обязанности единственного большого компьютера, почему бы нам не взять несколько маленьких

компьютеров и не заставить их работать так же, как единственный большой компьютер? Действительно, почему бы не поручить группе персональных компьютеров эту работу?

Обратимся к старому процессу обработки накладных и счетов. В нем большой универсальный компьютер проверял неоплаченные счета один за другим и решал, нужно или нет выполнять следующие задачи:

- отослать письмо с напоминанием;
- поинтересоваться состояниями счетов;
- понизить кредитный лимит заказчику.

Некоторая организация выполняет это классическое приложение на громадном универсальном компьютере, самом большом из доступных сегодня. Обрабатывая более 500 000 счетов каждый месяц, приложение выполняется более 6,5 ч. Компания написала это приложение таким образом, чтобы его легко можно было преобразовать для выполнения на более мелких компьютерах.

В качестве эксперимента компания установила сеть с 8 серверами БД и 20 персональными компьютерами, функционирующими как вычислительные серверы. Вычислительные серверы имели мощность, эквивалентную мощности машины класса IBM 486 с тактовой частотой 66 МГц. После всех подсчетов полная конфигурация такой сети стоит менее 10% стоимости большого универсального компьютера.

Какое же реальное время выполнения мы получили? Тридцать минут! Несмотря на то, что такой процесс продолжался 6,5 ч работы большого универсального компьютера, он занял только полчаса работы сети маленьких компьютеров. Приложение архитектуры "клиент-сервер" выполнилось более чем в 12 раз быстрее, чем на большом универсальном компьютере. Какой вывод можно из этого сделать?

Построение сетей на основе небольших, многопроцессорных серверов привело к благоприятным возможностям в соотношении цена/производительность. Закон Гросха не только аннулировался, но вообще превращался в свою противоположность. Чем меньше вы платите за ваш компьютер, тем большую мощность получаете. Исторически складывалось так, что лучшим решением для увеличения мощности была покупка большего компьютера. Теперь самый лучший путь получить

дополнительную мощность - купить меньший компьютер. Конечно, вы должны купить множество таких более маленьких компьютеров, но в целом множество меньших компьютеров все равно произведут больше компьютерной мощности за меньшие деньги. Поэтому по отношению к конкретному компьютеру вы получите больше мощности, если потратите меньше денег на каждый компьютер, но купите большее их количество.

Эксперимент с просроченными счетами указывает на различные аспекты распределенной обработки, а не конкретно на многопроцессорные серверы. Если сеть "клиент-сервер" и так в совокупности функционирует как один огромный компьютер, то зачем еще многопроцессорные серверы? Почему бы не сделать сеть персональных компьютеров, функционирующих как серверы? Если просроченные счета подсчитываются быстрее на 20 вычислительных серверах, подсоединенных к 8 серверам БД то зачем Вам вообще держать какие-либо большие компьютеры? Таков справедливый призыв при подходе, основанном на системе "клиент-сервер".

Вам должны быть ясны все плюсы и минусы такого подхода. Использование сети маленьких компьютеров для замены ими единственной большой машины может повернуть ваши дела к удешевлению расходов, а в области производительности - к увеличению скорости. Но будет ли это решение проще? На этот вопрос трудно ответить. С одной стороны, управление единственным большим компьютером более открыто и прямолинейно и, следовательно, проще. С другой стороны, сеть маленьких компьютеров позволяет распараллеливать компьютерную мощность согласно структуре организации. Перемещение места расположения компьютера на рабочий стол также упрощает его использование для владельца. Так что пример с просроченными счетами поднимает некоторые болезненные вопросы о действительной необходимости больших компьютеров в современном мире. Наконец, компании могут решить, что им вообще нет необходимости содержать большие универсальные компьютеры; технически такая возможность осуществима. Заметим, однако, что подобные решения требуют осторожного подхода и вдумчивой оценки.

Существует еще одно неудобство больших компьютеров - недостаток персональной свободы для пользователей. Персональные компьютеры стали популярными так быстро, потому что они обещают свободу для своих

пользователей. Пока компьютеры не подешевели, никто не задумывался о том, как стесняет ограниченный доступ к компьютерной мощности. Все соглашались, что компьютеры - это большие, дорогостоящие коробки с отдельным помещением и службой управления, но ни в коем случае не персональный слуга. Следовательно, централизованная модель применения компьютеров автоматически исключала такое их использование, как предлагают нам сегодня персональные компьютеры.

Многие крупные организации превышают практические ограничения, которые характерны для их центральных больших универсальных компьютеров. Авиалинии - пример крайнего положения в этой проблеме. Громадные приложения обслуживания авиалиний чрезмерно напрягают даже самые большие компьютеры, но организации с более умеренными требованиями также встречаются с этими ограничениями. Как только загрузка приближается к пределу возможностей большого компьютера, персонал вынужден сражаться за поддержание допустимого для работы времени реакции на транзакции, беспокоиться о том, сколько времени займет выполнение того или иного пакетного задания, и предаваться тяжким раздумьям на тему о том, что новые приложения потребуют модернизации системы, а это такое дорогое удовольствие!

Опыт XX столетия продемонстрировал неопровержимые аргументы в пользу децентрализованных операций всех видов. Хотя большие универсальные компьютеры и имеют некоторые замечательные возможности, сети персональных компьютеров могут заменить большие машины. Скорее всего, не единственным персональным компьютером, но их сетью - определенно.

Конечно, системы персональных компьютеров в конфигурации "клиент-сервер" - это принудительная модель сети, используемой как один компьютер. Но фактически локальная сеть может быть лучшим "большим универсальным компьютером", чем сам большой компьютер.

Революция, вызванная появлением персональных компьютеров, продолжается. Все находятся в состоянии перехода к персональным компьютерам. Большие универсальные компьютеры стоимостью миллионы долларов заменяются сетями персональных компьютеров, которые стоят всего тысячи. Это результат компьютерного разукрупнения. Компании,

занимающиеся перепроектированием бизнеса, разукрупняются еще и организационно - приспосабливают управление среднего слоя к этим изменениям и реализуют решения, которые приближают их к передовому уровню. Выполняя больше работы с меньшим количеством людей, такие организации вынуждены серьезно относиться к разделению полномочий. В этом им помогает компьютерная система, которой может управлять напрямую как коллектив, так и каждый индивидуально. Таким образом, вместо замены больших компьютеров более дешевыми меньшими компьютерами, но по-прежнему управляемыми централизованно, революция разукрупнения призывает заменить большие компьютеры сотнями компактных систем, каждая из которых взаимодействует с остальными и обслуживает потребности локальных коллективов и конкретных индивидуумов. Это культурное разукрупнение, занимающееся перемещением управления организацией из центра этого процесса в локальные офисы и самоуправляемые коллективы. Результатом являются распределенные компьютерные системы, которые поддерживают децентрализованное принятие решений и управляются уполномоченными служащими, акцентирующими свое внимание на качестве продукции и возможности быстрого реагирования на потребности пользователя. Это революция технологии "клиент-сервер" 90-х годов.

Далее рассмотрим более подробно технологию организации локальных вычислительных сетей и глобальной сети Интернет.

Вопросы для самопроверки

1. Централизованный подход к организации хранения и обработки информации.
2. Децентрализованный подход к организации хранения и обработки информации.
3. Достоинства и недостатки централизованного и децентрализованного распределения информации
4. Сферы применения централизованной и децентрализованной обработки информации.
5. Закон Гросха.

Литература

1. Душин В.К. Теоретические основы информационных процессов и систем: Учебник. – М.: Издательство «Дашков и К», 2012. – 348 с.
2. Сухомлин В.А. Введение в анализ информационных технологий: Учебник. – М.: Горячая линия – Телеком, 2003. – 427 с.
3. Козлов В.А., Багдамян В.Е., Чернышев А.Б. Современные Интернет технологии. Ч.1: HTML и CSS. - Пятигорск: ПГТУ, 2011. - 103 с.
4. Технология построения интерактивных Web-ресурсов: учебное пособие. - Ульяновск: УлГТУ, 2009. - 100 с.
5. Втюрин В.А. Компьютерные технологии в области автоматизации и управления. Учебное пособие по направлению 220700 "Автоматизация технологических процессов". - СПб: СПбГЛТУ. 2011. - 103 с.
6. Бройдо В.Л., Ильина О.П. Вычислительные системы, сети и телекоммуникации: учеб. пособие для студентов вузов. - 4-е изд. - Санкт-Петербург: Питер, 2011. - 554 с.
7. Дунаев С. Доступ к базам данных и техника работы в сети. Практические приемы современного программирования. – М.: Диалог, МИФИ, 1999. – 416 с.
8. Костров А.В.,Александров Д.В. Уроки информационного менеджмента. Практикум: Учебное пособие. – М.: Финансы и статистика, 2005.
9. Олифер В.Г., Олифер Н.А. Компьютерные сети. Принципы, технологии, протоколы. – СПб: Питер, 2000. – 672 с.
10. Громов Г.Р. Национальные информационные ресурсы: проблемы промышленной эксплуатации. – М.: Наука, 1984. – 240 с.
11. Карр Николас Дж. Блеск и нищета информационных технологий: Почему ИТ не являются конкурентным преимуществом./ Пер. с англ. – М.: Издательский дом «Секрет фирмы», 2005. – 176 с.
12. Дж. Лодон, К. Лодон. Управление информационными системами. 7-е изд./Пер. с англ. – СПб.: Питер, 2005. – 912 с.

ПРИЛОЖЕНИЕ

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2015616260

Программное обеспечение адаптации сайта образовательного процесса к реализуемой образовательной программе вуза

Правообладатель: ***Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Уральский государственный лесотехнический университет» (RU)***

Авторы: ***Часовских Виктор Петрович (RU), Стаин Дмитрий Александрович (RU), Воронов Михаил Петрович (RU), Кох Елена Викторовна (RU)***

Заявка № **2015612667**

Дата поступления **06 апреля 2015 г.**

Дата государственной регистрации в Реестре программ для ЭВМ ***04 июня 2015 г.***

Врио руководителя Федеральной службы по интеллектуальной собственности

Л.Л. Кирий

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2015617724

Программа управления динамически настраиваемым сайтом основных сведений об образовательной организации высшего профессионального образования

Правообладатель: ***Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Уральский государственный лесотехнический университет» (RU)***

Авторы: ***см. на обороте***

Заявка № **2015612669**
Дата поступления **06 апреля 2015 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***21 июля 2015 г.***

Врио руководителя Федеральной службы по интеллектуальной собственности

Л.Л. Кирий

Учебное пособие

Виктор Петрович Часовских
Михаил Петрович Воронов
Галия Абдулазисовна Акчурина
Елена Викторовна Кох
Галина Львовна Нохрина

Построение корпоративных систем информационного обслуживания управленческой деятельности

ISBN 978-5-6041352-6-6

Компьютерная верстка - **М.П. Воронов**

Уральский государственный лесотехнический университет
620100, Екатеринбург, ул. Сибирский тракт, 37