Технологии обработки больших данных средствами СУБД ADABAS

Екатеринбург
2022

Министерство науки и высшего образования Российской Федерации
Уральский государственный экономический университет

Технологии обработки больших данных средствами СУБД ADABAS

Рекомендовано
Советом по учебно-методическим вопросам и качеству образования
Уральского государственного экономического университета
в качестве учебного пособия

Екатеринбург
2022

УДК 004(075)
ББК 32.81я73
Т38

Рецензенты:

Ученый совет Института радиоэлектроники и информационных технологий — РТФ Уральского федерального университета имени первого Президента России Б. Н. Ельцина (протокол № 2 от 24 февраля 2022 г.)

генеральный директор ООО «Октоника»
К. Г. Ведьманов

Авторский коллектив:
В. П. Часовских, *В. Г. Лабунец*, *Е. Н. Стариков*,
Е. В. Кох, М. П. Воронов

Т38 **Технологии обработки больших данных средствами СУБД ADABAS** : учебное пособие / В. П. Часовских, В. Г. Лабунец, Е. Н. Стариков [и др.] ; Министерство науки и высшего образования Российской Федерации, Уральский государственный экономический университет. — Екатеринбург : УрГЭУ, 2022. — 170 с.

ISBN 978-5-9656-0325-1

Цель учебного пособия — формирование системы теоретических и практических знаний о современных информационных технологиях хранения и обработки больших данных (big data) в среде СУБД ADABAS. Рассматриваются роль аппаратных и программных средств в управлении, основные положения построения информационных систем организации и использования СУБД при их проектировании и эксплуатации; функциональные возможности, структура, базовая модель данных и структура хранения данных СУБД ADABAS; операции создания (определения) и управления элементами физической и логической структуры БД ADABAS; основные операторы языка Natural для работы с БД ADABAS. Пособие может использоваться при изучении дисциплин «Базы данных», «Формализация информации и big data», «Администрирование информационных систем».

Для студентов очной и заочной форм обучения направлений бакалавриата 02.03.03 «Математическое обеспечение и администрирование информационных систем», 09.03.03 «Прикладная информатика», 09.03.01 «Информатика и вычислительная техника» и магистратуры 38.04.05 «Бизнес-информатика», 09.04.03 «Прикладная информатика».

УДК 004(075)
ББК 32.81я73

ISBN 978-5-9656-0325-1

Введение

Развитие информационных технологий и внедрение автоматизированных систем управления на предприятиях и в организациях приводят к росту объемов данных, хранимых организацией и служащих основой для осуществления различных видов деятельности. В условиях конкуренции возможность быстрого реагирования на изменения условий внешней среды, проведения оперативного анализа и своевременного принятия решений представляет для организации все больший интерес. В условиях постоянного повышения качества принимаемых решений автоматизированные системы анализа в организациях базируются на все более совершенных математических и информационных моделях, и все большее количество факторов влияния используется в качестве исходных данных. Кроме того, если речь идет об интеллектуальных системах, выбирающих наиболее эффективное решение с учетом решений, принятых в прошлом, и эффекта от принятия решений, объемы хранимых данных еще более возрастают.

Таким образом, существует необходимость постоянного повышения эффективности использования информационных систем, обрабатывающих большие объемы данных. Одним из способов увеличения производительности является оптимизация структуры хранения данных и запросов к данным на этапах проектирования и внедрения информационной системы. При таком подходе большое значение имеет наличие у проектировщиков системно-ориентированных программных средств, пакета концептуально единых и архитектурно отработанных программных продуктов с широкими функциональными возможностями. Важной составляющей таких пакетов является система управления базами данных (СУБД).

Сегодня мы имеем больше информации, чем можем обработать традиционными методами: от деловых операций и научных данных до спутниковых снимков, текстовых отчетов и данных военной разведки. Обычного информационного поиска уже недостаточно для принятия решений. Столкнувшись с огромными массивами данных, мы создали новые потребности, которые помогут нам сделать лучший управленческий выбор. Эти потребности заключаются в автоматическом обобщении данных, извлечении «сущности» хранящейся информации и обнаружении закономерностей в необработанных данных.

Таким образом, остро актуально постоянно повышать эффективность использования информационных систем, обрабатывающих большие объемы данных. Предлагаемое учебное пособие посвящено данной проблематике.

ADABAS (Adaptable database system), разработка Software AG, давно известна в России как высоконадежная и производительная СУБД. Это многофункциональная СУБД, с успехом применяемая в таких областях деятельности, как управление организацией, обработка научно-технической и библиографической информации, автоматизация проектных работ, обработка экономической информации. Она обеспечивает высокую производительность при работе с большими и сверхбольшими базами данных, обладает развитыми средствами контроля, поддержания и восстановления целостности баз данных.

К сегодняшнему дню ADABAS сильно изменилась и представляет собой систему управления сверхбольшими базами данных, которая, в совокупности с рядом дополнительных продуктов, существенно расширяющих ее возможности, позволяет строить не только традиционные системы обработки структурированных данных, но и текстовые информационно-поисковые системы, географические и экспертные системы, системы обработки изображений и т. д.

Компания Software опубликовала детали новой стратегии по поддержке и дальнейшему развитию линейки продуктов ADABAS до 2050 г. Планируется, что заказчики получат ресурсы необходимые для подготовки к смене поколения в индустрии программного обеспечения. Потребуются специалисты, имеющие навыки для поддержки и развития корпоративных приложений

в будущем. Релиз новых продуктов и возможностей ADABAS подтверждает актуальность выбранного курса.

Особый интерес представляют технологии больших данных в реальном времени для масштабирования сервисов и приложений в Интернете, разработки аналитических решений с формированием отчетов и поддержкой принятия решений.

Концепцию и возможности ADABAS можно оценить в следующих решениях для России.

Сферы деятельности: PublisherNews — портал системы продвижения публикаций; Информтехнологии, связь, Интернет; Консалтинг; Стандартизация

Сайты субъектов РФ: Москва

Сайты федеральных округов РФ: Центральный федеральный округ

Сайты стран: Германия; Россия

Сайты регионов мира: Центральная Европа

Цель данного учебного пособия — формирование системы теоретических и практических знаний о современных информационных технологиях хранения и обработки больших данных (big data) в среде СУБД ADABAS.

Для создания конкретных информационных систем в среде ADABAS предназначена система Natural — платформа Software AG. Это высокоуровневый язык программирования, позволяющий существенно сократить сроки и стоимость разработки бизнес-приложений, ограждая разработчиков от излишних сложностей. Natural поддерживает все типы пользовательских интерфейсов, включая Web, графический интерфейс Windows, текстовые терминалы. Приложения, написанные на Natural, могут быть легко интегрированы с любыми внешними сервисами, будь то XML/Web-сервисы, DCOM, CORBA и др. Приложения на Natural могут работать с большинством реляционных и постреляционных СУБД. Среда разработки и исполнения Natural-приложений существует для всех основных операционных систем и аппаратных платформ, включая мейнфреймы, Unix, Linux и OpenVMS.

Пособие включает общетеоретические аспекты применения информационных технологий в управлении, описание основных существующих структур баз данных, описание СУБД ADABAS (включая функциональные возможности, архитектуру,

модель данных), вопросы проектирования и эксплуатации информационных систем, построенных с помощью СУБД ADABAS, вопросы обеспечения целостности данных и организации вычислительного процесса, а также основные аспекты проектирования пользовательских приложений в среде Natural.

Первая глава пособия посвящена общим вопросам информационных технологий в управленческой деятельности. Рассматриваются роль аппаратных и программных средств в управлении, основные положения построения информационных систем организации и использования СУБД при проектировании и эксплуатации информационных систем.

Во второй главе рассматриваются особенности ADABAS как СУБД: функциональные возможности, структура системы, базовая модель данных, структура хранения данных.

Третья глава посвящена вопросам создания (определения) элементов физической и логической структуры БД ADABAS.

В рамках четвертой главы представлены некоторые возможности автоматизации проектирования баз данных. Затрагиваются создание и восстановление таблиц определения данных, реорганизация структур данных, изменение размера физических структур баз данных.

В пятой главе дается описание основных операторов языка Natural необходимых для работы с БД ADABAS.

Глава 1

Состав и сущность информационных технологий в управлении

1.1. Аппаратное и программное обеспечение информационных технологий

Классификация программного обеспечения

В основу работы компьютеров положен *программный принцип управления*, состоящий в том, что компьютер выполняет действия по заранее заданной программе. Этот принцип обеспечивает универсальность использования компьютера: в определенный момент времени решается задача соответственно выбранной программе. После ее завершения в память загружается другая программа и т. д.

Для нормального решения задач на компьютере нужно, чтобы программа была отлажена, не требовала доработок и имела соответствующую документацию. Поэтому относительно работы на компьютере часто используют термин *программное обеспечение* (ПО, software), под которым понимают совокупность программ, процедур и правил, касающихся функционирования программной системы для решения поставленной задачи.

Система управления базами данных (СУБД) — это сложная программная система накопления и последующего манипулирования данными. Каждая СУБД предоставляет интерфейс с базой

данных и может располагать средствами непосредственного доступа к последней ее пользователей.

С помощью языка описания данных создаются описания элемента и записей данных, а также взаимосвязей между ними. Для выполнения операции с базой данных (например, выборки или обновления) в прикладных программах используется язык манипулирования данными. Фактическая структура физического хранения данных известна только СУБД.

СУБД позволяют управлять большими информационными массивами — базами данных. Программные системы этого вида позволяют обрабатывать на компьютере массивы информации, обеспечивают ввод, поиск, сортировку и выборку записей, составление отчетов и т. д. Представители данного класса программ — Microsoft Access, Clipper, Paradox, Oracle, ADABAS, DB2 и пр.

Интегрированные системы часто сочетают в себе возможность СУБД, табличного процессора, текстового редактора, системы деловой графики и пр. Как правило, все компоненты интегрированной системы имеют схожий интерфейс, что облегчает обучение работе с ними. Представители интегрированных систем — пакет Microsoft Office и его бесплатный аналог Open Office.

Бухгалтерские программы предназначены для ведения бухгалтерского учета, подготовки финансовой отчетности и финансового анализа деятельности предприятий. Из-за несовместимости отечественного бухгалтерского учета с зарубежным в нашей стране используются почти исключительно отечественные бухгалтерские программы.

Классификация ЭВМ по различным признакам

Компьютеры могут быть классифицированы по ряду признаков: по принципу действия, назначению, способам организации вычислительного процесса, размерам и вычислительной мощности, функциональным возможностям, способности к параллельному выполнению программ и др.

Классификация ЭВМ по принципу действия

Электронная вычислительная машина, компьютер — комплекс технических средств, предназначенных для автоматической

обработки информации в процессе решения вычислительных и информационных задач.

По принципу действия вычислительные машины делятся на три больших класса:

– аналоговые (АВМ);

– цифровые (ЦВМ);

– гибридные (ГВМ).

Цифровые вычислительные машины (ЦВМ) — вычислительные машины дискретного действия, работают с информацией, представленной в дискретной, а точнее, в цифровой форме.

Аналоговые вычислительные машины (АВМ) — вычислительные машины непрерывного действия, работают с информацией, представленной в непрерывной (аналоговой) форме, т. е. в виде непрерывного ряда значений какой-либо физической величины (чаще всего электрического напряжения).

АВМ весьма просты и удобны в эксплуатации; программирование задач для решения на них, как правило, нетрудоемкое; скорость решения задач изменяется по желанию оператора и может быть сделана сколь угодно большой (больше, чем у ЦВМ), но точность решения задач очень низкая (относительная погрешность 2−5 %). На АВМ наиболее эффективно решать математические задачи, содержащие дифференциальные уравнения, не требующие сложной логики.

Гибридные вычислительные машины (ГВМ) — вычислительные машины комбинированного действия, работают с информацией, представленной и в цифровой, и в аналоговой форме; они совмещают в себе достоинства АВМ и ЦВМ. ГВМ целесообразно использовать для решения задач управления сложными быстродействующими техническими комплексами.

Наиболее широкое применение получили ЦВМ с электрическим представлением дискретной информации — электронные цифровые вычислительные машины, обычно называемые просто электронными вычислительными машинами (ЭВМ), без упоминания об их цифровом характере.

Классификация ЭВМ по назначению

По назначению ЭВМ можно разделить на три группы:

– универсальные (общего назначения);

– проблемно-ориентированные;

– специализированные.

Универсальные ЭВМ предназначены для решения самых различных технических задач: экономических, математических, информационных и др., отличающихся сложностью алгоритмов и большим объемом обрабатываемых данных. Они широко используются в вычислительных центрах коллективного пользования и в других мощных вычислительных комплексах.

Проблемно-ориентированные ЭВМ служат для решения более узкого круга задач, связанных, как правило, с управлением технологическими объектами, с регистрацией, накоплением и обработкой относительно небольших объемов данных, с выполнением расчетов по относительно несложным алгоритмам. Они обладают ограниченными по сравнению с универсальными ЭВМ аппаратными и программными ресурсами.

Специализированные ЭВМ используются для решения узкого круга задач или реализации строго определенной группы функций. Такая узкая ориентация ЭВМ позволяет четко специализировать их структуру, существенно снизить их сложность и стоимость при сохранении высокой производительности и надежности.

Классификация ЭВМ по размерам и функциональным возможностям

По размерам и функциональным возможностям ЭВМ можно разделить на:

– сверхбольшие (суперЭВМ);

– большие ЭВМ;

– малые ЭВМ;

– сверхмалые (микроЭВМ).

Выпуск в 1971 г. первого микропроцессора (МП) привел к появлению в 1970-х гг. нового класса — микроЭВМ. Именно наличие МП служило первоначально определяющим признаком микроЭВМ. Сейчас микропроцессоры используются во всех без исключения классах ЭВМ.

Классификация микроЭВМ:

1) универсальные:

– многопользовательские;

– однопользовательские (персональные);

2) специализированные:

– многопользовательские (серверы);

– однопользовательские (рабочие станции).

Многопользовательские микроЭВМ — это мощные микроЭВМ, оборудованные несколькими видеотерминалами и функционирующие в режиме разделения времени, что позволяет эффективно работать на них сразу нескольким пользователям.

Персональные компьютеры (ПК) — однопользовательские микроЭВМ, удовлетворяющие требованиям общедоступности и универсальности применения.

Рабочие станции (work station) представляют собой однопользовательские мощные микроЭВМ, специализированные для выполнения определенного вида работ (графических, инженерных, издательских и др.).

Серверы (server) — многопользовательские мощные микроЭВМ в вычислительных сетях, выделенные для обработки запросов от всех станций сети.

Конечно, вышеприведенная классификация весьма условна, ибо мощная современная ПК, оснащенная проблемно-ориентированным программным и аппаратным обеспечением, может использоваться и как полноправная рабочая станция, и как многопользовательская микроЭВМ, и как хороший сервер, по своим характеристикам почти не уступающий малым ЭВМ.

Классификация ПК по конструктивным особенностям

По конструктивным особенностям персональные компьютеры подразделяются на:

1) стационарные (настольные);

2) переносные:

– портативные;

– блокнот;

– карманные;

– электронные секретари;

– электронные записные книжки.

СуперЭВМ

К суперЭВМ относятся мощные многопроцессорные вычислительные машины с быстродействием сотни миллионов — десятки миллиардов операций в секунду (оп./с).

Создать высокопроизводительную ЭВМ по современной технологии на одном микропроцессоре не представляется возможным ввиду ограничения, обусловленного конечным значением скорости распространения электромагнитных волн (300 000 км/с), ибо время распространения сигнала на расстояние несколько миллиметров (линейный размер стороны МП) при быстродействии 100 млрд оп./с становится соизмеримым с временем выполнения одной операции. Поэтому суперЭВМ создаются в виде *высокопараллельных многопроцессорных вычислительных систем* (МПВС).

Высокопараллельные МПВС имеют несколько разновидностей:

– *магистральные (конвейерные) МПВС*, в которых процессоры одновременно выполняют разные операции над последовательным потоком обрабатываемых данных; по принятой классификации такие МПВС относятся к системам с многократным потоком команд и однократным потоком данных (МКОД или MISD — Multiple Instruction Single Data);

– *векторные МПВС*, в которых все процессоры одновременно выполняют одну команду над различными данными — однократный поток команд с многократным потоком данных (ОКМД или SIMD — Single Instruction Multiple Data);

– *матричные МПВС*, в которых МП одновременно выполняют разные операции над несколькими последовательными потоками обрабатываемых данных — многократный поток команд с многократным потоком данных (МКМД или MIMD — Multiple Instruction Multiple Data).

1.2. Информационная система управления организацией

Управление организацией 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.

Основу любой интегрированной системы управления составляют ее информационные модели и средства их обработки, представленные в совокупности баз данных (БД). Интегрированная система также подразумевает автоматизацию следующих процессов:

– планово-аналитической деятельности организации;

– учетной деятельности;

– финансовой деятельности;

– документооборота;

– регламентирования полномочий и обязанностей, а также их делегирования;

– принятия решений;

– системы учета заработной платы, а также мотивации персонала.

Определение и основные понятия информационных систем управления и информационных технологий

Информационная система управления — это совокупность информации, экономико-математических методов и моделей, технических, программных, других технологических средств и специалистов, предназначенная для обработки информации и принятия управленческих решений.

По *видам процессов управления* информационные системы (ИС) делятся на:

– *ИС управления технологическими процессами* — предназначены для автоматизации различных технологических процессов (гибкие технологические процессы, энергетика и т. д.);

– *ИС управления организационно-технологическими процессами* — многоуровневые, иерархические системы, которые сочетают в себе ИС управления технологическими процессами и ИС управления предприятиями;

– *ИС организационного управления* — предназначены для автоматизации функций управленческого персонала. К этому классу относятся ИС управления как промышленными фирмами, так и непромышленными экономическими объектами — предприятиями сферы обслуживания. Основными функциями таких систем являются оперативный контроль и регулирование, оперативный учет и анализ, перспективное и оперативное планирование,

бухгалтерский учет, управление сбытом и снабжением и решение других экономических и организационных задач;

– *интегрированные ИС* — предназначены для автоматизации всех функций управления фирмой и охватывают весь цикл функционирования экономического объекта, начиная от научно-исследовательских работ, проектирования, изготовления, выпуска и сбыта продукции до анализа эксплуатации изделия;

– *корпоративные ИС* — используются для автоматизации всех функций управления фирмой или корпорацией, имеющей территориально разобщенные подразделения, филиалы, отделения, офисы и т. д.;

– *ИС научных исследований* — обеспечивают решение научно-исследовательских задач на базе экономико-математических методов и моделей;

– *обучающие ИС* — используются для подготовки специалистов в системе образования, при переподготовке и повышении квалификации работников различных отраслей экономики.

В зависимости от размера организации существует несколько подходов к использованию информационных технологий.

1. *На малых предприятиях* различных сфер деятельности информационные технологии, как правило, связаны с решением задач бухгалтерского учета, накоплением информации по отдельным видам бизнес-процессов, созданием информационных баз данных по направленности деятельности фирмы и организации телекоммуникационной среды для связи пользователей между собой и с другими предприятиями и организациями.

2. *В средних организациях (предприятиях)* большое значение для управленческого звена имеет функционирование электронного документооборота и привязка его к конкретным бизнес-процессам. Для таких организаций (предприятий, фирм) характерны расширение круга решаемых функциональных задач, связанных с деятельностью фирмы, организация автоматизированных хранилищ и архивов информации, которые позволяют накапливать документы в различных форматах, предполагают наличие их структуризации, возможностей поиска, защиты информации от несанкционированного доступа и т. д.

3. *В крупных организациях (предприятиях)* информационная технология строится на базе современного программно-

аппаратного комплекса, включающего телекоммуникационные средства связи, многомашинные комплексы, развитую архитектуру «клиент-сервер», применение высокоскоростных корпоративных вычислительных сетей.

В крупных организациях сложились две формы управления — *централизованная* и *децентрализованная*.

Организации с централизованным управлением характеризуются распределением функций и полномочий среди структурных подразделений с жесткой координацией производственно-хозяйственной деятельности в аппарате управления.

Децентрализованная форма характеризуется выделением внутри организации стратегических единиц бизнеса или центров прибыли, деятельность которых поддается самостоятельному планированию и имеет свой бюджет.

Корпоративная вычислительная сеть — это интегрированная, многомашинная, распределенная система одного предприятия, имеющего территориальную рассредоточенность, состоящая из взаимодействующих локальных вычислительных сетей структурных подразделений и подсистемы связи для передачи информации.

Определяющим фактором при организации корпоративных вычислительных сетей является *простота доступа к информационным ресурсам*. В этой связи основой современного подхода технических решений в построении информационной технологии в корпоративных системах является архитектура «клиент-сервер». Реальное распространение архитектуры «клиент-сервер» стало возможным благодаря развитию и широкому внедрению в практику концепции *открытых систем*. Основным смыслом подхода открытых систем является упрощение процесса организации совместимости вычислительных сетей за счет международной и национальной стандартизации аппаратных и программных интерфейсов.

Классификация информационных систем по функциональному признаку и уровням управления

В организации типовые виды деятельности дифференцируются по функциональному признаку: производственная, марке-

тинговая, финансовая, кадровая. Каждое из направлений деятельности характеризуется определенными целями, задачами и функциями (табл. 1).

Таблица 1

Функции информационных систем

Вид системы	Функции
Информационная система маркетинга	Исследование рынка и прогнозирование продаж. Управление продажами. Рекомендации по производству новой продукции. Анализ и установление цены. Учет заказов
Производственные информационные системы	Планирование объемов работ и разработка календарных планов. Оперативный контроль и управление производством. Анализ работы оборудования. Участие в формировании заказов поставщиков. Управление запасами
Финансовые и учетные информационные системы	Управление портфелем заказов. Управление кредитной политикой. Разработка финансового плана. Финансовый анализ и прогнозирование. Контроль бюджета. Бухгалтерский учет и расчеты зарплаты
Кадровые (человеческих ресурсов) информационные системы	Анализ и прогнозирование потребности в трудовых ресурсах. Ведение архивов записей о персонале. Анализ и планирование подготовки кадров
Прочие системы, например, информационная система руководства	Контроль деятельности фирмы. Выявление оперативных проблем. Анализ управленческих и стратегических ситуаций. Обеспечение процесса выработки стратегических решений

Взаимосвязь типов информационных систем по функциональному признаку с учетом уровней управления и квалификации менеджеров может быть представлена в виде пирамиды. Основание пирамиды составляют информационные системы, с помощью которых специалисты решают структурированные задачи операционной обработки данных, а менеджеры низшего звена —

задачи оперативного управления. Наверху пирамиды на уровне стратегического управления информационные системы изменяют свою роль и становятся стратегическими, поддерживающими деятельность менеджеров высшего звена по принятию решений в условиях частично структурированных задач. На любом уровне управления необходима информация из всех функциональных систем, но в разных объемах и с разной степенью обобщения.

Основные понятия информационной системы организации

Организация — это открытая целевая система, которая производит товары или услуги, используя определенные циклы, включающие ресурсы (материальные и нематериальные активы), процессы и результаты. Как открытая система, организация взаимодействует с окружающей средой и адаптируется к изменениям в ней. Как система, организация состоит из набора взаимосвязанных элементов (людей, сырья, оборудования, финансов, информации, знаний и др.).

Организация — это механизм, с помощью которого люди объединяют усилия и работают вместе (группами и командами), чтобы сделать больше, чем в сумме могли бы сделать, работая независимо. Организационное поведение фокусируется на людях, на обмене информацией между ними.

Информационный контур — воздействие в системе управления организации управляющей и управляемой частей друг на друга в виде передачи информации (о целях управления, о состоянии управляемого процесса, об управляющих воздействиях).

Информационная система организации образуется на основе информационного контура вместе со средствами сбора, передачи, обработки и хранения информации, а также с персоналом, осуществляющим эти действия с информацией. При более сложном строении управляемого процесса управляющая часть может иметь многоуровневую структуру. Для управления организацией требуется систематизированная, предварительно подготовленная информация.

Миссия информационной системы — предоставление нужной информации в нужное время и в нужном месте для обеспечения эффективного управления всеми ресурсами организации,

создание информационно-вычислительной среды для осуществления управления.

Принятие решений — акт целенаправленного воздействия на управляемый процесс, основанный на информации о нем, определенной ранее цели и разработанной программе достижения этой цели.

Процесс принятия решений — это по сути процесс формирования решения. Под термином «принятие решений» понимают множество видов деятельности. Принятие решений — одна из важнейших функций менеджмента, лежащая в основе всех функциональных областей бизнеса. Характер решений, процесс их принятия с определенной дискретностью во времени оказывают существенное влияние на функционирование информационной системы организации, выбор информационных технологий.

Процесс принятия решения

В *модели процессов принятия решений, планирования и управления* непосредственно сам процесс принятия решений (процесс планирования) включает первые пять этапов. Последние два этапа отражают *процесс управления*. Рассмотрим каждый из этапов.

1. *Определение целей организации* позволяет оценить преимущества выбранного решения по сравнению с другими. Обеспечение максимально возможной прибыли (более точно, максимизации приведенной стоимости будущих денежных поступлений наличных средств) в качестве основной цели деятельности компании обусловлено следующим:

– управление компанией должно осуществляться менеджерами прежде всего в интересах ее владельцев или акционеров;

– максимизация прибыли приводит к улучшению общего экономического благосостояния всего общества;

– максимизация прибыли позволяет достаточно полно и всесторонне оценить возможность организации продолжить свою деятельность.

2. *Поиск альтернативных вариантов действий (стратегий)*, при помощи которых можно добиться осуществления поставленных целей. Получение информации об ожидаемой конъ-

юнктуре и перспективах изменения окружающей среды, в которой действует компания, — это наиболее сложный и важный этап процесса принятия решения. Руководство компании должно своевременно выявлять потенциальные возможности и угрозы, возникающие в окружающей среде. Компании расширяют масштабы поиска (разработка новых товаров и рынков для их продажи).

3. *Сбор данных об альтернативных действиях*. После того как потенциальные сферы деятельности компании определены, менеджеры должны оценить возможные темпы роста этих видов деятельности в будущем, способность компании выйти на заданный показатель рыночной доли и поступления денежных средств для каждого вида альтернативной деятельности при различных внешних условиях. *Внешними условиями* называют факторы, которые не могут контролироваться лицом, принимающим решения (экономический бум в стране, высокий уровень инфляции, снижение деловой активности и др.). Выбор цели определяет *стратегические перспективы* и, как следствие, тип решений, которые будут оказывать влияние на положение компании. Поэтому важно получить информацию о возможностях компании и о той окружающей среде, в которой она будет действовать. Принятие стратегических решений является прерогативой менеджеров высшего уровня.

Менеджеры также принимают решения, которые не требуют длительного привлечения ресурсов компании — *краткосрочные* или *оперативные*. Основу для принятия таких решений дает текущая внешняя ситуация, а также имеющиеся в данный момент в распоряжении менеджера оперативного уровня или компании в целом материальные, кадровые и финансовые ресурсы. Оперативные решения направлены на реализацию стратегических решений компании.

Примеры краткосрочных решений:

– Какие цены должны быть установлены на продукцию компании?

– Сколько продукции разного вида необходимо произвести?

– Какие СМИ необходимо использовать для рекламирования продукции?

– Какой уровень обслуживания будет предложен потребителям?

Для принятия краткосрочных решений менеджерам необходимо собрать соответствующую информацию, например, о ценах реализации продукции конкурентов, ожидаемом спросе на товары, прогнозируемых затратах при различных вариантах производства и альтернативных ценах продаж.

4. *Выбор оптимального варианта действий из возможных альтернатив*. На практике принятие решения включает сравнение конкурирующих между собой альтернативных вариантов действий и выбор из них того, который наилучшим образом отвечает потребностям организации. Если исходить из предположения, что цель — получение в будущем максимальных поступлений наличных средств, то выбор варианта может быть сведен к инкрементному анализу чистых поступлений денежных средств и ранжированию альтернативных вариантов.

5. *Реализация принятых решений.* Альтернативные варианты действий необходимо включить в финансовый план, составляемый для осуществления различных решений, принятых менеджерами. Посредством финансовых планов учитываются поступления в организацию наличных денежных средств и платежи организации, а также поступления от реализации продукции и затраты. Составление планов предназначено для того, чтобы каждый сотрудник организации знал о своей роли, которую он должен играть при реализации решений, принятых высшим руководством.

6. *Сравнение фактических и запланированных результатов*. На данном этапе реализуется *управленческая функция контроля и регулирования* процесса принятия решения, которая включает:

– измерение результатов деятельности;

– предоставление сведений о результатах деятельности;

– выработку корректирующих мер, направленных на то, чтобы установленные цели были достигнуты, а планы реализованы.

Мониторинг показателей функционирования осуществляется менеджером, ответственным за реализацию решения, на основе *отчета об исполнении сметы*, который готовит и пре-

доставляет ему бухгалтер. В отчетах, которые обеспечивают *обратную связь*, акцент должен делаться на тех видах деятельности, которые расходятся с планом, чтобы менеджеры уделили этим отклонениям должное внимание (*управление по отклонениям*).

7. *Корректирование выявленных отклонений от плана*. Корректирующие действия по приведению фактических результатов в соответствие с запланированными или корректировка планов в целом реализуются с помощью *контуров обратной связи*. Они показывают, что процесс принятия решения итерационный, и подчеркивают взаимосвязи между его различными этапами. Обратная связь *между этапами 7 и 2* свидетельствует о том, что ход выполнения плана должен постоянно анализироваться. И если окажется, что принятый план далее не может быть реализован, необходимо рассмотреть альтернативные варианты действий, обеспечивающие достижение организацией поставленных целей. Второй контур обратной связи (*между этапами 7 и 5*) показывает корректирующие действия, предпринимаемые с целью приведения фактических результатов в соответствие с запланированными.

Технологии поддержки принятия решений

Приведем описание инструментов, которые наиболее часто используются в информационных системах, и основные типы решений, при принятии которых их применение будет эффективно.

1. *Операционные структурированные решения*

– Системы обработки транзакций (Transaction Processing Systems).

– Технология планирования ресурсов предприятия (Enterprise Resource Planning).

– Технология хранилища данных (Data Warehouse).

2. *Неструктурированные решения на уровне специалистов*

– Интеллектуальный анализ данных (Data Mining).

– Искусственный интеллект (Artificial Intelligence):

  − Экспертные системы (Expert Systems);

  − Нейронные сети (Neural Networks).

– Системы поддержки групп (Group Support Systems) — Технология электронных совещаний.

3. *Поддержка решений на управленческом уровне*

– Менеджерские информационные системы (Management Information Systems).

– Системы поддержки принятия решений (Decision Support Systems).

4. *Неструктурированные решения на стратегическом уровне*

– Системы поддержки принятия решений (DSS).

– Системы поддержки для высшего руководства (Executive Support Systems).

Уровни целей и планов организации

Цель управления — определенное желаемое состояние будущего (конечный результат), которого пытается достичь организация, информация о котором доведена до сотрудников и общества.

Планирование — процесс определения целей организации и средств их достижения. Он начинается с изложения миссии, определяющей главную цель деятельности предприятия, прежде всего с позиций внешней среды.

Заявление о миссии — это общее определение целей коммерческой деятельности, а также корпоративных ценностей, которыми руководствуется компания. Для внешнего окружения (инвесторов, потребителей, поставщиков) заявление о миссии несет информацию об основных направлениях бизнеса, отличающих ее от других компаний, обоснованности ее действий, сети филиалов. Сотрудники компании из заявления о миссии ясно видят цели, к которым стремится организация, ее отношение к сотрудникам.

Стратегический уровень управления (корпоративная стратегия). Стратегические цели управления формулируют намерения организации в следующих областях: рыночные позиции, инновации, производительность, распределение ресурсов (материальных, финансовых и информационных), прибыльность, управленческая деятельность и ее развитие, трудовая деятельность и установки сотрудников, обязательства перед обществом. В стратегических планах определяются действия, которые компания намеревается предпринять для достижения стратегических

целей (бизнес-планы, планы производства и маркетинговых исследований). Ответственность за достижение и реализацию стратегических планов организации несут менеджеры высшего звена управления.

Тактический уровень управления. Тактические цели являются одним из элементов достижения стратегических целей организации и определяют результаты, которых должны достичь подразделения. Тактические планы определяют конкретные действия подразделений, направленные на достижение стратегических целей, на короткий период планирования, обычно — ближайший год. Разработка тактических планов в соответствии с корпоративной стратегией и осуществление контроля их выполнения является обязанностью менеджеров среднего звена.

Оперативный (операционный) уровень управления. Оперативные цели — это конкретные результаты, которых должны достичь подразделения, рабочие группы, отдельные работники. Эти цели точны и измеримы. Оперативные планы указывают на последовательность действий (операционных задач) по достижению операционных целей и обеспечивают выполнение тактических планов. На их основе строится деятельность менеджеров низшего звена. Операционное планирование предполагает выделение соответствующих ресурсов по каждому пункту плана, что должно быть согласовано с бюджетом организации. Для решения операционных задач составляются планы-графики, где определяются временные рамки их решения менеджерами и рабочими группами, согласованные с тактическими и стратегическими планами организации.

1.3. Системы управления базами данных

Понятие базы данных и системы управления базами данных

В узком смысле слова, *база данных* — это некоторый набор данных необходимых для работы (актуальные данные). Однако данные — это абстракция; никто никогда не видел «просто данные»; они не возникают и не существуют сами по себе.

Данные — это отражение объектов реального мира. Пусть, например, требуется хранить сведения о деталях, поступивших на склад. Для того чтобы ответить на вопрос, каким образом данные о деталях будут храниться в БД, необходимо знать, какие признаки или стороны детали будут актуальны, необходимы для работы. Среди них могут быть *название детали, ее вес, размеры, цвет, дата изготовления, материал*, из которого она сделана, и т. д. В традиционной терминологии объекты реального мира, сведения о которых хранятся в базе данных, называются *сущностями* — entities, а их актуальные признаки — *атрибутами* (attributes).

Каждый признак конкретного объекта есть *значение атрибута*. Так, деталь «двигатель» имеет значение атрибута «вес», равное «50». Это отражает тот факт, что данный двигатель весит 50 килограммов.

Было бы ошибкой считать, что в базе данных отражаются только *физические объекты*. Она способна вобрать в себя сведения об *абстракциях, процессах, явлениях,* т. е. обо всем, с чем сталкивается человек в своей деятельности. Например, в базе данных можно хранить информацию о заказах на поставку деталей на склад (хотя заказ — не физический объект, а процесс). Атрибутами сущности «заказ» будут название поставляемой детали, количество деталей, название поставщика, срок поставки и т. д.

Таким образом, в широком смысле слова, база данных — это совокупность описаний объектов реального мира и связей между ними актуальных для конкретной прикладной области.

База данных также может быть рассмотрена как один или несколько *файлов данных*, предназначенных для хранения, изменения и обработки больших объемов взаимосвязанной информации.

Система управления базами данных (СУБД) — это система программного обеспечения, позволяющая обрабатывать обращения к базе данных, поступающие от прикладных программ конечных пользователей.

СУБД позволяют объединять большие объемы информации и обрабатывать их, сортировать, делать выборки по определенным критериям и т. п.

Центральным компонентом любой СУБД является сервер базы данных. Его техническое качество в большой степени определяет главные характеристики системы, такие как *производительность, надежность, безопасность* и т. д. В то же время богатство и разнообразие возможностей, заложенных в механизм его функционирования, сильно сказываются на эффективности разработки прикладных программ.

Наряду с *технологией распределенного хранения данных* обсуждается активно развивающаяся *технология тиражирования данных*. В последнее время *проблема безопасности* в СУБД выдвинулась на первый план. Это обширная тема, требующая особого обсуждения.

Основные функции СУБД

К числу функций СУБД (для пользователей систем) принято относить следующие:

– непосредственное управление данными во внешней памяти;

– управление буферами оперативной памяти;

– управление транзакциями;

– журнализация;

– языки БД.

Для восстановления БД после жесткого сбоя используют *журнал* и *архивную копию БД*. Грубо говоря, архивная копия — это полная копия БД к моменту начала заполнения журнала (имеется много вариантов более гибкой трактовки смысла архивной копии). Конечно, для нормального восстановления БД необходимо, чтобы журнал не пострадал. К сохранности журнала во внешней памяти в СУБД предъявляются повышенные требования. Тогда восстановление БД состоит в том, что исходя из архивной копии по журналу воспроизводится работа всех транзакций, которые закончились к моменту сбоя. В принципе можно даже воспроизвести работу незавершенных транзакций и продолжить их по окончании восстановления. Однако в реальных системах это обычно не делается, поскольку процесс восстановления после жесткого сбоя является достаточно длительным.

Языки БД

Для работы с базами данных используются специальные языки, в целом называемые *языками баз данных*. В ранних СУБД поддерживалось несколько специализированных по своим функциям языков. Чаще всего выделялись два — *язык определения схемы БД (SDL — Schema Definition Language)* и *язык манипулирования данными (DML — Data Manipulation Language)*.

SDL служил главным образом для определения логической структуры БД, т. е. той структуры БД, какой она представляется пользователям.

DML содержал набор операторов манипулирования данными, т. е. операторов, позволяющих заносить данные в БД, удалять, модифицировать или выбирать существующие данные.

В современных СУБД обычно поддерживается *единый интегрированный язык*, содержащий все необходимые средства для работы с БД начиная от ее создания и обеспечивающий базовый пользовательский интерфейс с базами данных.

Типовая организация современной СУБД

Логически в современной реляционной СУБД можно выделить:

– внутреннюю часть — ядро СУБД (Data Base Engine);
– компилятор языка БД;
– подсистему поддержки времени выполнения;
– набор утилит.

В некоторых системах эти части выделяются явно, в других — нет, но логически такое разделение можно провести во всех СУБД.

Ядро СУБД отвечает за *управление данными* во внешней памяти, *управление буферами оперативной памяти, управление транзакциями* и *журнализацию*. Соответственно можно выделить такие компоненты ядра, как *менеджер данных, менеджер буферов, менеджер транзакций* и *менеджер журнала*. Функции этих компонентов взаимосвязаны, и для обеспечения корректной работы СУБД все эти компоненты должны взаимодействовать по тщательно продуманным и проверенным протоколам.

Основная функция *компилятора языка БД* — компиляция операторов языка БД в некоторую выполняемую программу.

Основная проблема реляционных СУБД — то, что языки этих систем (как правило, SQL) являются *непроцедурными*, т. е. в операторе такого языка специфицируется некоторое действие над БД, но эта спецификация не процедура, она лишь описывает в некоторой форме условия совершения желаемого действия. Поэтому компилятор должен решить, каким образом выполнять оператор языка, прежде чем произвести программу. *Результатом компиляции* является выполняемая программа, представляемая в некоторых системах в машинных кодах, но более часто в выполняемом внутреннем машинно-независимом коде.

В последнем случае реальное выполнение оператора производится с привлечением *подсистемы поддержки времени выполнения*, представляющей собой, по сути дела, интерпретатор внутреннего языка.

Наконец, в *отдельные утилиты БД* обычно выделяют такие процедуры, которые слишком накладно выполнять с использованием языка БД, например, *загрузка* и *разгрузка БД, сбор статистики, глобальная проверка целостности БД* и т. д. Утилиты программируются с использованием интерфейса ядра СУБД, а иногда даже с проникновением внутрь ядра.

Выводы по главе 1

Итак, базовое и прикладное программное обеспечение в целом является инструментарием для разработки и эксплуатации рабочих программ конечных пользователей и информационной системы в целом.

Кроме того, на практике встречаются оригинальные задачи, которые нельзя решить имеющимися прикладными программными продуктами либо с использованием прикладных программных продуктов (ППП). Результаты получаются в форме, не удовлетворяющей конечного пользователя. В этом случае с помощью систем программирования или алгоритмических языков разрабатываются оригинальные программы, учитывающие требования и условия решения задачи.

Основные тенденции развития прикладного программного обеспечения тесно связаны с созданием и переходом на информационные системы четвертого поколения, основанные на иерар-

хической структуре, в которых центр тяжести перенесен с локальных сетей конечных пользователей на сеть локальных серверов. В основу ИС четвертого поколения закладывается требование сокращения эксплуатационных ресурсов ИС при увеличении масштабируемости системы и расширения круга ее функциональных обязанностей.

Последнее обстоятельство особенно важно, так как существует устойчивая тенденция к практически 100 % интеграции информационных технологий различных функциональных подсистем (бизнес-приложений) в единую бизнес-модель предприятия. Это предполагает существование достаточно большого множества (часто противоречивых) требований к ИС со стороны конечных пользователей, неоднородных по своей квалификации и профессиональным задачам.

Разрабатываемые в настоящее время ППП основываются на концепции организации ИС четвертого поколения (которая сформировалась в начале 1990-х гг. на базе синтеза централизованной и распределенной обработки информации) и предполагают соблюдение следующих основных принципов:

– полного использования потенциала настольных систем и среды распределенной обработки;

– интеграции различных архитектурных решений без каких-либо ограничений, т. е. построения абсолютно открытой системы;

– обеспечения максимальной экономичности системы;

– достижения качественно нового уровня производительности, гибкости и динамичности организации системы;

– параллельной оптимизации структуры ИС, «бизнес-приложений» (ППП функциональных подсистем), поддерживаемых с помощью ресурсов ИС.

Вопросы для самоконтроля

1. Понятие вычислительной машины, ее основные структурные элементы.
2. Классификация ЭВМ.
3. Сфера применения мейнфреймов.
4. Сфера применения мини-компьютеров.

5. Сфера применения персональных компьютеров.
6. Классификация персональных компьютеров.
7. Общие характеристики карманного ПК.
8. Классификация базового программного обеспечения.
9. Назначение и классификация операционных систем.
10. Виды и назначение программ сервисного обслуживания.
11. Назначение и состав средств трансляторов языков программирования.
12. Назначение и состав программ технического обслуживания.
13. Классификация прикладного программного обеспечения.
14. Состав и функции интегрированных пакетов.
15. Назначение и выполняемые функции программ-редакторов.
16. Назначение и выполняемые функции табличных процессоров.
17. Системы управления базами данных и их назначение.
18. Экспертные системы и принципы из разработки.
19. Виды проблемно-ориентированных ППП.
20. Виды предметно-ориентированных ППП.

Глава 2

СУБД ADABAS

2.1. Функциональные возможности системы

СУБД ADABAS и ее окружение

ADABAS — постреляционная СУБД, поддерживающая различные модели данных, включающая вложенные отношения и иерархические поля. С помощью ADABAS можно строить иерархические, сетевые и реляционные SQL базы данных, а также сложные текстовые информационно-поисковые и интегрированные системы. Разумеется, ADABAS дает пользователям возможность работать с данными в рамках чисто реляционной модели.

ADABAS работает на разных платформах — Windows, Unix, Open VMS, AS/400, BS2000, MVS, VM, OS/390 и др.

Обладает возможностью выборочного страхового сохранения данных в процессе обычного сеанса, что важно при работе с большими объемами данных и большим количеством активных пользователей. Также параллельно могут выполняются административные и другие задачи обслуживания баз данных. В то же самое время использует относительно небольшое количество вычислительных ресурсов, и затраты на обслуживание баз данных невелики.

ADABAS прекрасно соответствует требованиям в области мультимедиа-данных, в области управления документами и др. (поддерживает динамические переменные). В отличие от систем, имеющих одну модель для всех типов данных, поддерживает различные модели и структуры данных, которые могут быть спе-

циально спроектированы для работы с разными видами приложений.

На рис. 1 показаны основные возможности СУБД ADABAS в виде схемы.

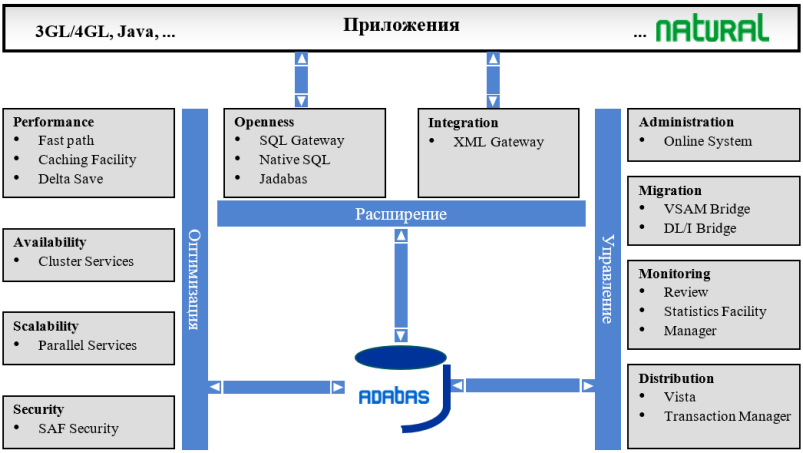

Рис. 1. СУБД ADABAS и ее окружение

Основные характеристики СУБД ADABAS:
– высокая надежность и исполнение;
– полная доступность 24×7;
– минимальная потребность в администрировании;
– доступность дополнительных расширений;
– доступность разных платформ;
– реляционные и вложенные реляционные структуры;
– различные методы доступа;
– меню-управляемые утилиты;
– администрирование, когда база данных активная.
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 отделены от особенностей конкретных физических носителей, что делает СУБД гибкой и эффективной.

Полнота функций

ADABAS относится к классу функционально развитых СУБД, которые обеспечивают пользователей большим набором функциональных возможностей. Она рассчитана на широкий класс практических применений, требующих коллективного доступа к базам данных при решении интерактивных прикладных задач приложений информационных систем.

Предназначена для накопления, интегрированного хранения, поиска, обработки и представления данных в форме, удобной для подготовки и принятия решений. Функции и технические характеристики этой системы позволяют применять ее для решения широкого класса информационных задач без разработки дополнительных прикладных программ. Различные виды поиска, широкие возможности логической обработки выбранной информации позволяют получать разнообразные типы выходных документов.

В процессе диалогового взаимодействия с системой ADABAS производятся форматированный ввод информации

в БД, модификация хранимых данных, поиск и обработка данных, а также подготовка выходных отчетов.

Средства описания сценариев диалога позволяют реализовать не только отдельные слабо связанные элементарные процессы поиска, ввода, корректировки и обработки данных, но и совокупности взаимосвязанных информационных или технологических процессов, которые могут быть каталогизированы в словаре данных системы и в дальнейшем выполнены в автоматическом режиме.

Программы ADABAS обеспечивают:

– создание и ведение больших БД сетевой или иерархической структуры;

– диалоговое программирование прикладных задач;

– обработку данных и форматирование отчетов по запросам пользователей;

– централизованное управление доступом к БД;

– сохранность информации в БД;

– поддержание целостности и непротиворечивости БД;

– ведение словаря данных;

– настройку на программно-аппаратную среду и управление функционированием системы;

– диалоговый и пакетный режимы работы.

По совокупности реализуемых функций ADABAS может быть отнесена к классу интегрированных систем, обеспечивающих ввод, обработку, корректировку и редактирование исходных данных с возможностью управления поиском и анализом информации в БД. Обладает средствами гибкой перестройки и развития структуры БД в соответствии с изменяющимися потребностями прикладных задач.

ADABAS включает в себя гибкие и простые в освоении *языковые* и *технологические* средства создания и эксплуатации БД информационных систем коллективного пользования, позволяет создавать эффективную унифицированную технологию их разработки, развертывания и ведения.

Естественность модели данных

Используемая в СУБД модель данных должна обеспечивать возможность отображения объектов реального мира в естест-

венных терминах и понятиях, не навязывающих пользователю непривычные для него представления. Модель данных ADABAS обладает одновременно простотой и значительной гибкостью. База данных состоит из совокупности связанных между собой файлов.

Логическая структура записей файлов обеспечивает естественное отображение объектов предметной области с иерархической структурой. Возможность динамического связывания различных файлов позволяет создавать иерархические или сетевые структуры, предоставляя механизм для отображения иерархических структур объектов предметной области или установления между ними более сложных взаимосвязей. При этом не требуются другие средства, кроме средств связывания для физического представления таких объектов в БД.

Указатели, поддерживающие иерархические или сетевые структуры, не хранятся в БД вместе с записями, а формируются динамически на основе связей, которые, например, могут быть созданы или переопределены после загрузки БД.

В модели данных ADABAS отсутствуют понятия «корневой» и «подчиненный» файлы. В зависимости от точки зрения пользователя один и тот же файл в разных запросах может быть как корневым, так и подчиненным.

Основной особенностью модели данных системы ADABAS является наличие механизмов для целенаправленного разбиения БД на классы ассоциаций, доступ к которым обеспечивается на основе значений атрибутов объектов предметной области. В результате создается возможность предварительной статической структуризации и выделения подмножеств записей БД на этапе ее создания. Логическое разделение БД и выделение ассоциативных подмножеств, содержащих информацию о разбиении на непересекающиеся множества, создает конструктивную основу для ассоциативного доступа к объектам предметной области и обеспечивает доступ к этим объектам только на основе информации о тех свойствах, которыми они характеризуются в рамках соответствующей предметной области.

Более того, в рамках такого представления возникает естественная возможность динамического выделения подмножества объектов (динамическая структуризация), удовлетворяющих за-

просу, без непосредственного обращения к записям БД. Пользователю достаточно задать требуемые предикаты, определяющие подмножества объектов с заданными свойствами, и связать их логическими операциями, которые необходимо выполнить над выделенными подмножествами объектов.

Такая технология работы с данными позволяет обеспечить:

– равнодоступность всех объектов и динамический характер связей между объектами БД;

– доступ к записям БД па основе значений нескольких атрибутов, определяемых в запросе и связанных логическими операциями;

– создание мощных высокоуровневых языков обработки данных;

– быструю реакцию системы на запросы пользователей за счет статического выделения подмножества объектов, ориентированных на последующую обработку данных;

– построение интегрированных документально-фактографических систем.

В модели данных ADABAS имеются два типа команд — *команды манипулирования записями* и *команды обработки ассоциаций*, обеспечивающие доступ к записям по значениям атрибутов и связям с другими записями, обладающими заданными свойствами.

Среда хранения БД ADABAS физически разделена на накопитель и ассоциатор. Накопитель содержит записи БД, а ассоциатор — ассоциативные списки внутрисистемных номеров записей. Такое построение во многих случаях обеспечивает возможность выдачи результата на основе информации, содержащейся только в ассоциаторе, без непосредственного обращения к записям БД.

ADABAS предоставляет пользователям также ряд внешних моделей, поддерживаемых соответствующими языковыми средствами. Эти модели в большей степени ориентированы на конкретные категории пользователей и обеспечивают им более удобное взаимодействие с системой. В частности, администратор БД имеет возможность с помощью средств описания интерфейсных модулей определить не только подсхему, доступную прикладной программе, но и допустимые в рамках этой подсхемы операции.

Надежность функционирования

Надежность работы СУБД может рассматриваться в двух аспектах:

– надежность организации вычислительного процесса;

– обеспечение целостности БД.

Надежность организации вычислительного процесса обеспечивается в ADABAS в первую очередь путем отделения программ, управляющих мультидоступом к БД, от прикладных программ и компонентов системы, реализующих обработку запросов пользователей. Такое построение системы полностью исключает возможность разрушения программ управления БД в случае ошибок в работе прикладных программ путем использования предоставляемых операционной системой программных и аппаратных средств защиты памяти.

ADABAS предоставляет широкие возможности для сохранения *целостности данных* в случае сбоев программного обеспечения, отказов аппаратуры и ошибок в прикладных программах. Для этой цели в системе предусмотрено:

– восстановление БД со страховой копии или на заданную контрольную точку;

– автоматическое восстановление БД на синхронную контрольную точку или транзакцию;

– удержание записей БД на время их модификации; автоматический контроль взаимной согласованности ассоциатора и накопителя;

– автономный контроль корректности ассоциатора; синтаксический и семантический контроль значений атрибутов.

Таким образом, целостность БД поддерживается на уровнях:

– всей БД;

– файлов;

– контрольных точек;

– транзакций;

– отдельных команд и атрибутов.

Механизмы поддержания целостности БД данных на уровне транзакции, команды и атрибута работают в автоматическом режиме. Этим обеспечивается быстрое восстановление БД в це-

лостное состояние и повышается эффективность функционирования системы.

Самостоятельную группу средств, поддерживающих надежное функционирование системы, составляют компоненты обеспечения сохранности информации в БД. Сохранность данных гарантируется в ADABAS на уровне доступа в систему, доступа к файлам и атрибутам записей.

Производительность системы

Высокая производительность ADABAS при выполнении операций поиска данных является *следствием* принятой организации БД, в основе которой лежит *принцип отделения механизма реализации путей доступа от самих данных*.

Информация о путях доступа к данным содержится в *ассоциативных списках*, обработка которых с учетом логических условий поиска обеспечивает *идентификацию релевантных записей* только на основе значений поисковых атрибутов без необходимости доступа к файлам накопителя БД. При этом значительно сокращается объем данных, которые требуется подвергнуть обработке, и, как следствие, уменьшается время обработки для тех приложений, где количество операций доступа и обработки данных превосходит количество операций по ведению БД.

Допускается также *рандомизированный доступ* к записям по значению атрибута, уникально идентифицирующего записи БД. В этом случае искомая запись выделяется за время, приблизительно равное одному доступу к магнитному диску.

Для *массовой обработки данных* (ввод значительных объемов данных и формирование отчетов путем последовательной обработки большого количества записей БД) предназначены специальные компоненты системы ADABAS, реализующие массовую загрузку БД и пакетный режим обработки данных, обеспечивающий одновременное формирование не менее 32 отчетов за один просмотр БД. При этом производительность системы может возрасти (в пересчете на одну запись) на порядок по сравнению с произвольным доступом к каждой записи.

При *диалоговом взаимодействии* с системой важнейшей характеристикой для пользователя является *время компиляции запроса*. Так, транслятор диалогового языка программирования

задач проверяет синтаксис запроса по мере ввода строк, что сокращает время реакции системы как в период трансляции, так и в период выполнения запроса.

С целью повышения пропускной способности системы организуется несколько параллельно работающих процессов взаимодействия с БД, причем один из них обрабатывает *команды модификации данных*, а остальные — *команды, не изменяющие состояние БД*. Такая организация вычислительного процесса наряду с повышением производительности работы системы обеспечивает возможность эффективного восстановления БД при аппаратных или программных сбоях.

Кроме того, в ADABAS организуется *параллельная работа нескольких процессов* на основе применения средств операционной системы (запуск нескольких задач в независимых разделах памяти) и мониторов управления вычислительным процессом (инициализация задачи для каждого видеотерминала). Благодаря параллельной работе на нескольких уровнях достигается эффективное использование центрального процессора.

Эффективность использования вычислительных ресурсов

Записи БД ADABAS хранятся в сжатой форме. За счет сжатия значительно сокращается объем внешней памяти, требуемой для хранения файлов БД, и уменьшается количество операций ввода-вывода. За счет сжатия записей экономится около 50 % внешней памяти. Каждой записи БД присваивается уникальный *внутрисистемный номер*, который служит ее логическим адресом на все время существования записи в БД.

Использование *логической адресации* с помощью внутрисистемных номеров позволяет сравнительно просто учитывать и распределять память, которая освобождается в процессе работы, и исключает реорганизацию БД.

Обмен информацией между внешней и оперативной памятью осуществляется через *системный буфер*. Использование системного буфера и динамического механизма управления им обеспечивает сохранение в оперативной памяти тех блоков БД, к которым наиболее часто обращались в процессе функционирования системы. Благодаря этому существенно сокращается количество необходимых обращений к БД при поиске записей и их

модификации. Эффективность использования системного буфера значительно возрастает за счет применения механизма сжатия данных.

ADABAS использует *записи переменной длины*, которые размешаются в *блоках накопителя и ассоциатора*, имеющих фиксированный размер. В каждом блоке может быть зарезервирована свободная память. *Резервирование свободной памяти* позволяет избежать частых перемещений информации при увеличении размера записей. При этом отпадает необходимость корректировки индексов или таблиц, благодаря чему не уменьшается эффективность доступа к БД по мере ее заполнения. Память, которая освобождается внутри блока, может быть немедленно использована для других записей.

Удобство эксплуатации

Структура БД, используемая в ADABAS, значительно уменьшает трудозатраты, связанные с проектированием и эксплуатацией БД, — пользователю предоставляется язык описания данных, при котором структура каждого файла может быть определена независимо от других файлов.

Для большинства приложений при моделировании объектов предметной области достаточно тех понятий, которые допустимы па уровне схемы файла: *простая группа, периодическая группа, простые* и *множественные атрибуты*.

Кроме того, для определения *ассоциативных связей между записями* в схеме БД некоторые из атрибутов можно объявить дескрипторами. Для *поисковых атрибутов (дескрипторов)* в ассоциаторе автоматически строятся *инвертированные списки*.

На любом этапе существования БД ее логическая структура может быть модифицирована без перезагрузки или реорганизации БД. Объектами модификации могут быть как логические структуры файлов, так и связи между ними. Тем самым предоставляется возможность отражать в БД изменения, которые происходят в модели предметной области.

Структура файлов может быть расширена путем включения *простых* и *множественных атрибутов* и *групп*. В схеме БД возможно добавление нового или удаление старого файла, а также определение новой или удаление старой связи.

После расширения логической структуры значения новых атрибутов могут присутствовать во вновь вводимых записях. В старые записи значения новых атрибутов добавляются с помощью *команд корректировки*.

Для каждой категории пользователей ADABAS предоставляет простые и непроцедурные языковые средства. Система располагает *диалоговым языком программирования*, который существенно сокращает трудозатраты по реализации прикладных задач.

Описания данных (длина, тип значения, информация для редактирования, имена колонок отчетов и т. д.) хранятся в словаре данных. Эти описания доступны всем пользователям ADABAS и не зависят от формата и соответствующего описания, используемого в БД, что обеспечивает определение логических объектов, отражающих специфику решаемой задачи.

Средства конструирования интерактивных технологий позволяют администратору базы данных работать с естественными понятиями его предметной области и создавать стандартные технологии сопровождения БД, обеспечивающие легкость и простоту ее эксплуатации.

Стандартные технологические процессы позволяют автоматизировать ввод, контроль и преобразование данных, загрузку и корректировку данных, организацию и ведение словаря данных и т. д.

Анализ функциональных возможностей ADABAS показывает, что эта система может служить эффективным инструментом для реализации информационных систем, существенно сокращающим трудоемкость разработки прикладных задач.

2.2. Базовая модель и структура данных системы

ADABAS разработана как высоконадежная и производительная СУБД для создания и эксплуатации больших баз данных на мейнфреймах. В настоящее время она представляет собой систему управления сверхбольшими базами данных и позволяет строить не только традиционные системы обработки структурированных данных, но и текстовые информационно-поисковые

системы, географические и экспертные системы, системы обработки изображений и т. д.

Это становится возможным благодаря тому, что ADABAS обеспечивает поддержку следующих моделей и типов данных.

1. Не первая нормальная форма (NF2 — Non-First Normal Form). Эта модель (традиционная) данных характерна для начальной версии ADABAS.

2. Традиционная реляционная модель данных. Эта модель соответствует ANSI/ISO стандарту SQL.

3. Модель данных сущность-связь (E/R модель). В ADABAS предусмотрено расширение до модели Entity-Relationship (или E/R модели) для управления сложными структурами данных с высокой степенью связности. Поддерживаются также рекурсивные структуры данных.

Объединяя эту модель с другими моделями ADABAS, можно строить мощные интегрированные базы данных и, соответственно, прикладные системы. Предпочтительные области для применения этих моделей — системы представления знаний, моделирование поведения сложных технических и биологических систем, расчеты потребностей, планирование материальных ресурсов различного вида и назначения (Bills of Materials).

4. Обработка и управление произвольными текстами. Этот тип данных (и соответствующие средства манипулирования ими) обеспечивает доступ к документам, обрабатываемым ADABAS, как к произвольным текстам. Возможно сочетать разного рода обработку текстовых данных со стандартными процедурами ADABAS для работы с форматированными данными. Например, это могут делать библиотеки, юридические конторы, службы новостей, телекомпании, газеты и журналы, правительственные организации — все, кто имеет и будет иметь данные в виде так называемых «свободных» или неструктурированных текстов.

5. Обработка и управление географическими данными. В отличие от традиционных географических (картографических) систем, которые используют структуры хранения данных (реляционные и др.) независимо от собственно географической информации, обеспечивая их совместное использование, как правило, только на уровне SQL-интерфейса, подход ADABAS характеризуется глубокой интеграцией на всех уровнях хранения и обработки.

Картографическая информация, координатные привязки объектов, форматированные данные и текстовая информация неограниченного объема хранятся в рамках единой БД. Точно так же в рамках одной программы и даже одного оператора (среда Natural) интегрирована обработка этой комплексной информации. Важно, что при этом обеспечивается не только высокая производительность геоинформсистем, но за счет поддержания логики транзакций гарантируется целостность и безопасность хранения БД и другие ее неотъемлемые характеристики.

2.3. Структура системы

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 обеспечивает независимость данных и способность информационных систем, построенных на основе СУБД, к эволюции. Наибольшее признание в настоящее время получила трехуровневая архитектура, 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

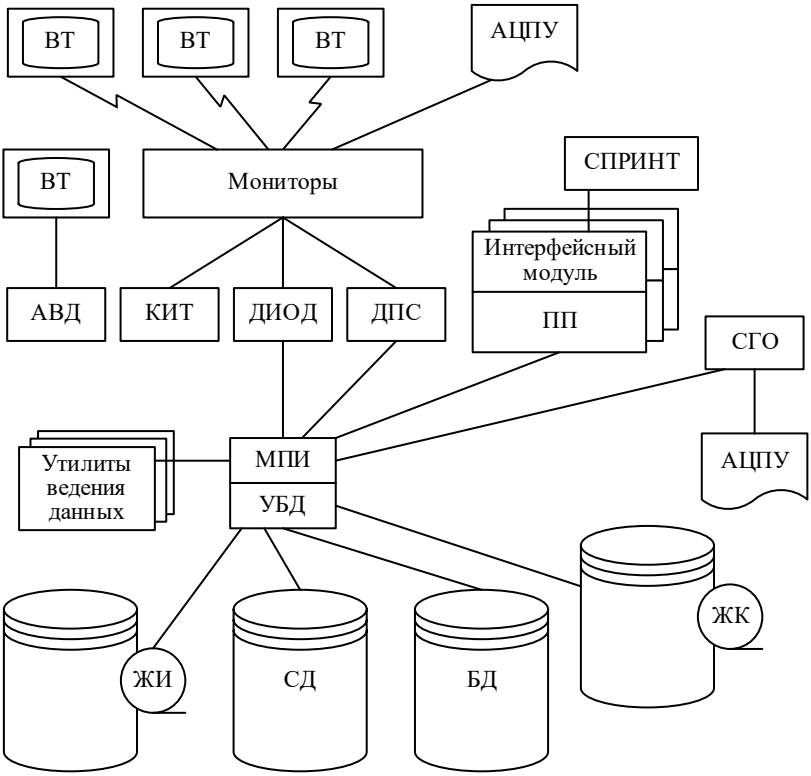

Рис. 2. Архитектура ADABAS:
ВТ — видеотерминал; АЦПУ — алфавитно-цифровое печатающее устройство; ПП — прикладные программы; ЖК — журнал команд; ЖИ — журнал изменений; СД — словарь данных; БД — база данных; АВД — автономное ведение данных; КИТ — конструирование интерактивных технологий; ДИОД — диалоговая обработка данных; ДПС — диалоговая подготовка справок; МПИ — мультипрограммный интерфейс; УБД — управление базой данных; СГО — средства генерации отчетов; СПРИНТ — средства программирования интерфейсных модулей

возможность независимой модификации внешних и внутренних представлений о БД благодаря существованию стабильной концептуальной схемы и действию соответствующих отображений.

ADABAS является СУБД с трехуровневой архитектурой. Она предоставляет в распоряжение пользователей концептуаль-

ный и внутренний уровни, ориентированные на АВД, 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 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. 2.

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.

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. В интерактивном режиме они получают параметры от средств АВД и КИТ. 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 COBOL, FORTRAN и др. 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.

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 поддерживает некоторый уровень абстракции БД, ориентированный на соответствующую категорию пользователей системы, и может рассматриваться как механизм отображения внешнего уровня на 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, на 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 элементы 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 используют 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 выдается 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 администратором базы данных (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 администратора базы данных 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, 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 для реальной вычислительной среды в соответствии с имеющимися ресурсами.

2.4. Структура хранения данных системы

СУБД ADABAS — масштабируемый полнофункциональный сервер баз данных, который представлен на всех основных серверных платформах и естественным образом функционирует в локальных и глобальных гетерогенных сетях.

ADABAS обеспечивает разработчику прикладных систем возможность выбора самых различных моделей данных.

Структура хранения данных достаточно традиционна и образует следующую иерархию.

1. **Поле (Реквизит)** — единица данных определенного типа.
2. **Запись (Строка)** — объект хранения, сведения о котором представлены в виде набора полей. Порядок следования полей несущественен.
3. **Файл (Таблица)** — набор однородных (имеющих одну структуру) записей.
4. **База данных** — физически (и технологически) обособленное хранилище файлов.

На уровне иерархии «Поле» ADABAS поддерживает все основные типы данных:

– символьные строки;

– битовые строки;

– числовые данные (двоичные и десятичные, фиксированной и переменной разрядности) и т. д.

К важнейшим особенностям ADABAS по представлению данных можно отнести поддержку следующих структурных элементов:

– **подполе** — вторичное значение, образованное частью значения, данного в элементарном (первичном) поле, имеющее уникальный идентификатор и позволяющее запросить значение из подполя без обращения ко всей информации первичного поля;

– **суперполе** — вторичное значение, образованное слиянием значений нескольких первичных полей (и подполей) записи;

– **множественное поле** — массив данных одного типа с одноуровневой индексацией. Единица данных в множественном поле называется реализацией;

– **периодическая группа** — одноуровневый массив данных определенной структуры. Каждый элемент (реализация) периодической группы представляет, в свою очередь, некоторую последовательность элементарных полей. Аналогом периодической группы являются понятия «вложенная структура» и «массив структур» в языках программирования.

Подполя, множественные поля и, особенно, периодические группы предоставляют необходимые уровни определения структурированных данных в пределах объекта хранения (записи).

Из особенностей физической организации хранения данных в ADABAS можно отметить следующие:

– сжатие данных — удаление незначащих нулей и пробелов из полей данных. Хранимое поле состоит из значения длины и строки значащих символов. Перед выдачей запрошенных данных они разжимаются в стандартный вид;

– хранение только значащих (непустых) полей. Вместо полей с пустыми значениями хранятся отметки о количестве пропущенных полей;

– хранение только значащих реализаций множественных полей и периодических групп;

– возможность отмены всех предыдущих режимов хранения, если это необходимо, назначением специальных атрибутов полям данных.

Из перечисленных свойств следует, что СУБД ADABAS естественным образом поддерживает плоские таблицы — наиболее простые и наглядные структуры данных, — являющиеся основой классической реляционной модели данных.

В то же время наличие дополнительных возможностей структурирования данных позволяет поддерживать средствами СУБД ADABAS класс моделей данных, получивших название постреляционных. Естественная поддержка постреляционной модели позволяет не только хранить в записях БД данные сложной структуры (этого можно достичь, например, моделируя вторич-

ные данные внутри первичной строки — символьной или битовой), но и различать элементы таких сложноорганизованных данных в запросах, индексах и т. п., что принципиально невозможно при моделировании.

Выводы по главе 2

Одним из первых средств, обладающих инструментарием для реализации основных моделей данных нереляционных СУБД, можно считать промышленную СУБД ADABAS.

База данных ADABAS определяется как совокупность взаимосвязанных файлов и ассоциаций записей файлов. Файл представляет собой именованную совокупность записей, имеющих одинаковую структуру. Ассоциация записей файлов представляет совокупность записей файла, обладающих общим свойством.

База данных ADABAS размещается на устройствах прямого доступа. Записи БД запоминаются в блоках этих устройств. Размер блока выбирается с учетом условия эффективного размещения целого числа блоков на дорожке используемых типов устройств.

Структурными элементами базы данных ADABAS на внутреннем уровне являются: накопитель, ассоциатор, рабочий набор и вспомогательные наборы данных. Ассоциатор используется для получения системных таблиц, преобразователя адреса и инвертированных списков. Рабочий набор данных предназначен для временного размещения промежуточной информации необходимой в процессе работы с БД.

Процедура определения базы данных включает три операции: физическое размещение, создание и редактирование файлов-контейнеров базы данных.

Вопросы для самоконтроля

1. Раскройте понятия «файл», «ассоциация», «атрибут», «группа», «периодическая группа».
2. Опишите структуру данных СУБД ADABAS и организацию связей между записями файлов БД. Каким образом хранятся записи файла и как строятся инвертированные списки?

3. Как в рамках СУБД ADABAS реализуется связь типа «многие-ко-многим»?
4. Назовите операции модификации схемы БД и операции модификации.
5. Охарактеризуйте операции чтения, поиска, селекции. В чем их различие? Назовите и опишите разновидности селекции.
6. Воспроизведите схему и алгоритм ассоциативного поиска с использованием инвертированных списков и алгоритм ассоциативного поиска записей в связанных файлах.
7. Назовите и опишите структурные элементы базы данных СУБД ADABAS.
8. Какова структура накопителя? Как задается объем накопителя и можно ли его увеличить в процессе эксплуатации?
9. Опишите структуру ассоциатора СУБД ADABAS. Что такое таблица FDT и в каком структурном элементе она хранится?
10. В каком виде хранятся инвертированные списки в ассоциаторе СУБД ADABAS? Как осуществляется доступ к требуемому инвертированному списку?
11. Как происходит преобразование BCH в адрес?
12. Опишите структуру рабочего и вспомогательных наборов данных.

Глава 3

Определение логической и физической структуры БД ADABAS

3.1. Определение БД

В рамках создания информационной системы средствами ADABAS и Natural первым необходимым шагом является определение структуры базы данных (хранилища данных). Процедура определения базы данных включает три операции: физическое размещение, создание и редактирование файлов-контейнеров базы данных (файлы с названиями ASSO, DATA, WORK). Для создания новой базы данных следует в главном контекстном окне (DBA Workbench) выбрать меню Database → New (рис. 3).

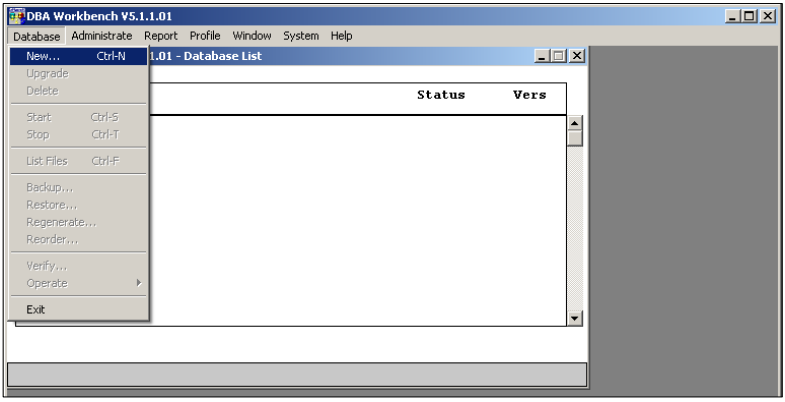

Рис. 3. Создание новой БД

В последующем окне (Create Database) для изменения будет предложен ряд параметров, определяющих размер и свойства файлов-контейнеров и БД (рис. 4). Все значения параметров и БД могут быть сохранены по умолчанию.

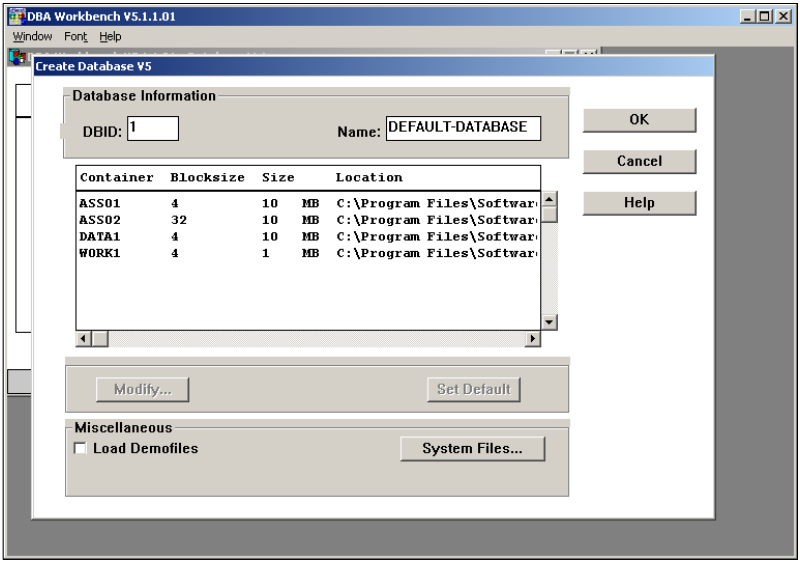

Рис. 4. Параметры БД

Информация о базе данных (Database Information)

Идентификационный номер БД (DBID). При открытии окна курсор мыши автоматически устанавливается на поле со списком ‘DBID’ (идентификационный номер БД). Следует оставить неизменным номер, предложенный системой, или ввести иной номер при помощи клавиатуры. Также можно воспользоваться списком данного поля и выбрать номер из списка. Следует учитывать, что системой в качестве идентификационного номера БД (‘DBID’) принимается любой номер от 1 до 255, который еще не был использован для создания другой БД.

Название БД (Name). При определении БД в поле ‘Name’ необходимо ввести название БД. Название не должно превышать 16 символов. Также можно сохранить название, предлагаемое по умолчанию.

Параметры файлов структуры (ASSO, DATA, WORK)

Изменение параметров файлов структуры, предложенных системой, следует производить посредством выбора соответствующего файла (ASSO1, ASSO2, DATA1, WORK1 и пр.) при помощи левой кнопки мыши и последующего выбора опции «Редактировать» («Modify…»). При этом системой будет предложено редактирование следующих параметров файлов (рис. 5).

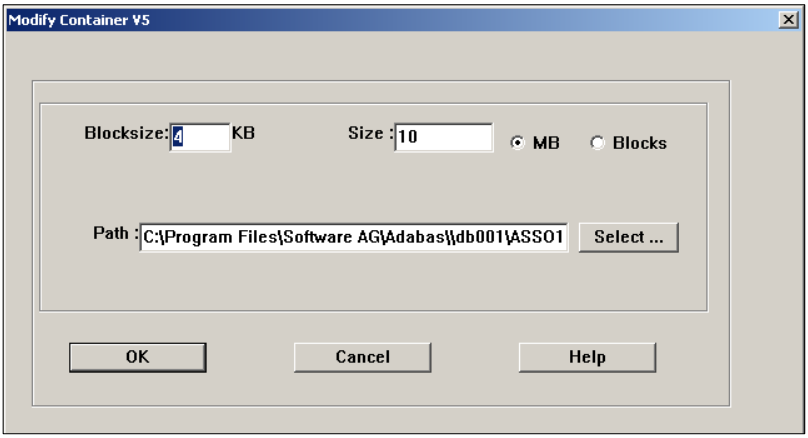

Рис. 5. Изменение параметров файлов структуры

Размер блока файла (Blocksize). В данном поле (‘Blocksize’) вводится размер блока для файла-контейнера ассоциатора БД (единица измерения — килобайт). Если значение данного поля не определено, то системой будет использоваться значение, предложенное по умолчанию.

При определении значения данного параметра для файла ассоциатора (‘ASSO Blocksize’) необходимо учитывать, что размер блока файла ассоциатора (ASSO) должен быть меньше размера блока рабочего файла (WORK). В противном случае значение данного поля не будет принято системой, о чем проектировщик БД будет информирован сообщением (рис. 6).

Размер файла (Size). Посредством ввода значения в поле «Size» определяется размер дискового пространства, используемого для изменяемого файла. Размер дискового пространства

можно определить в мегабайтах (опция «MB») и в блоках (опция «Blocks»). Значение данного параметра, предлагаемое по умолчанию — 1 Мб. Для увеличения дискового пространства для файла ассоциатора необходимо ввести соответствующее значение.

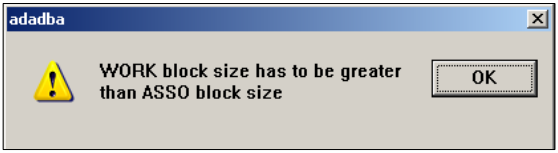

Рис. 6. Информационное сообщение, получаемое при превышении размера файла ассоциатора над размером рабочего файла

Расположение файла (Path). В текстовом поле «Path» отображается полная спецификация локализации файла ассоциатора. Можно оставить спецификацию неизменной (как предложено системой по умолчанию) либо установить иную область локализации файла при помощи вызова обзорного окна (кнопка «Select», как показано на рис. 7).

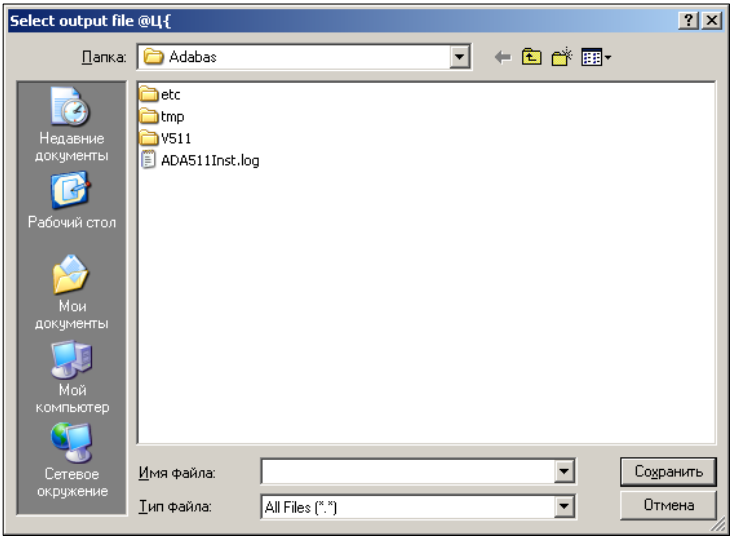

Рис. 7. Определение локализации файла

Для установки параметров файлов структуры на уровне значений, предлагаемых системой по умолчанию, следует воспользоваться опцией «Установить по умолчанию» («Set Default») (см. рис. 4).

Прочие параметры (Miscellaneous)

Номера системных файлов (System Files). В рамках данного подраздела должны быть определены идентификационные номера системных файлов БД (файлы Checkpoint File, Security File, User Data File), которые автоматически создаются и загружаются при создании БД. При нажатии на командную кнопку «Miscellaneous System Files…» появляется следующее окно (рис. 8).

По умолчанию системой предлагаются следующие номера системных файлов: checkpoint — 1, security — 2, user data — 3. При необходимости можно изменить номера системных файлов. Для сохранения изменений нажмите OK.

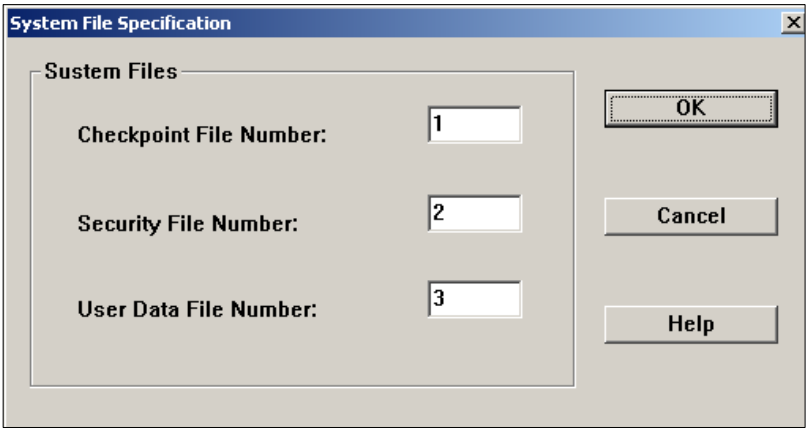

Рис. 8. Прочие параметры

Номера системных файлов не должны превышать максимальное число файлов БД ('Miscellaneous Maximum No. of Files'). В противном случае произведенные изменения не будут сохранены системой, о чем проектировщик БД будет проинформирован сообщением (рис. 9).

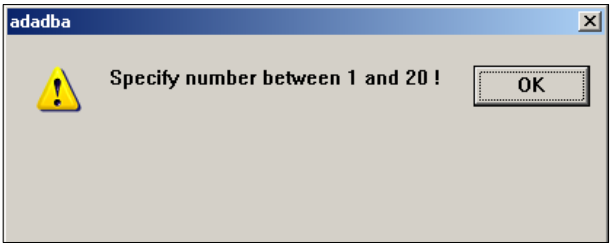

Рис. 9. Информационное сообщение,
получаемое при превышении максимального числа файлов БД

Загрузка демофайлов (Load Demofiles). Для загрузки демонстрационных файлов, предлагаемых системой, следует поставить флажок на опции «Загрузка демофайлов» («Load Demo-files»). При этом после завершения процедуры создания БД в структуру БД будут загружены соответствующие файлы.

После ввода и редактирования указанных параметров (информация о базе данных (Database Information), параметры файлов структуры (ASOO, DATA, WORK), прочие параметры (Miscellaneous)) для определения новой БД следует в окне «Create Database» выбрать ‘OK’.

Если вышеописанные параметры были заданы корректным образом, проектировщик будет проинформирован о создании новой БД соответствующим сообщением (рис. 10).

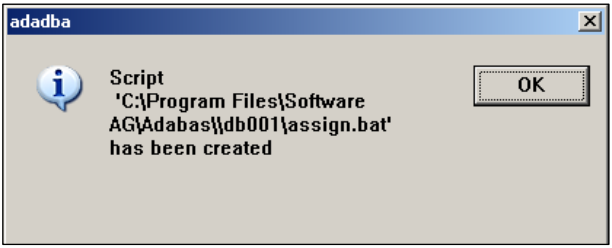

Рис. 10. Информационное сообщение об успешном создании БД

Вид главного управляющего окна («DBA Workbench») поменяется следующим образом (рис. 11).

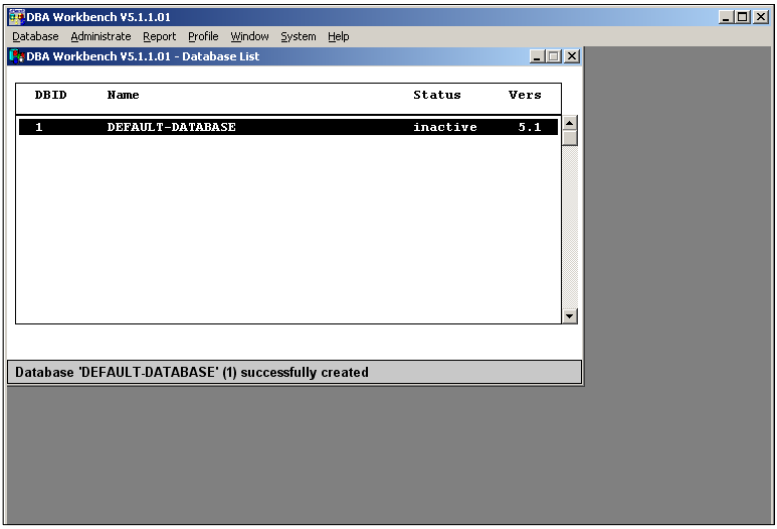

Рис. 11. Вид главного управляющего окна после создания новой БД

3.2. Определение файла БД

После определения БД для обеспечения возможности осуществления транзакций (операции внесения, удаления, редактирования данных) необходимо определение структуры файла в рамках созданной БД.

В системе ADABAS определение файла подразумевает создание таблицы определения данных (***Field Definition Table — FDT***) с определением свойств файла (локализация дискового пространства файла и пр.). После создания FDT и определения параметров файл создается в рамках выбранной БД и отображается в списке файлов БД.

Таблица FDT создается при помощи редактора FDT, а также может быть прочитана из уже существующего файла FDT или файла модуля определения данных (***Data Definition Module — DDM***) конструктора Natural. Для создания нового файла в рамках существующей БД необходимо в главном управляющем окне (DBA Workbench) выбрать (щелчком мыши) БД, в список файлов которой необходимо включить определяемый файл (рис. 12).

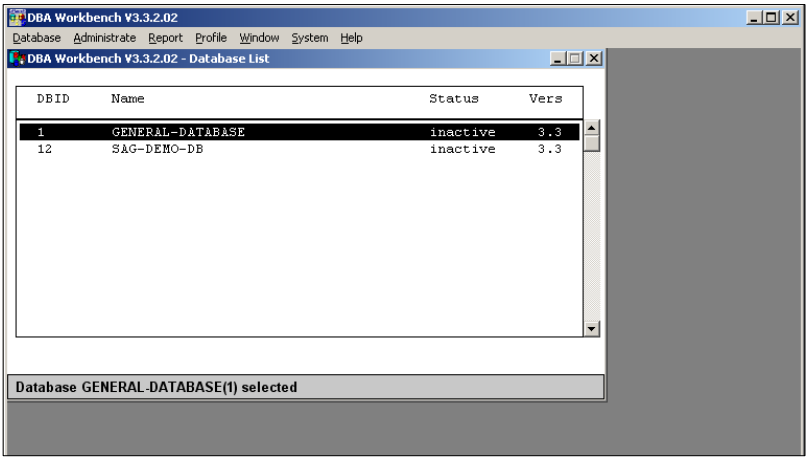

Рис. 12. Выбор БД для создания файла БД

Далее необходимо открыть окно списка существующих файлов данной БД. Окно списка открывается либо двойным щелчком мыши по записи выбранной БД, либо посредством меню Database → List Files (рис. 13).

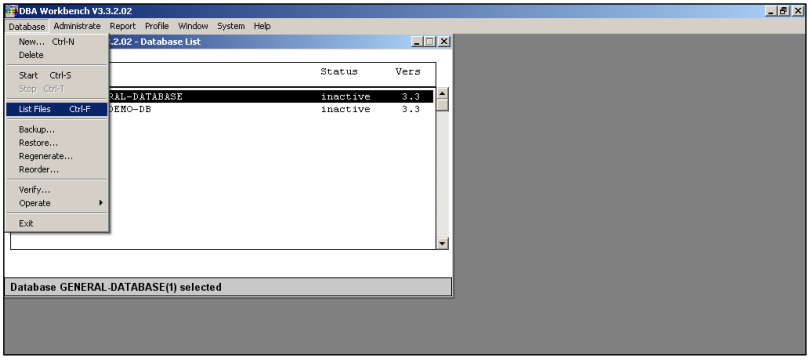

Рис. 13. Просмотр списка уже существующих файлов в БД

В появившемся окне списка файлов, даже если ни одного файла не было создано, будут отображаться 3 системных файла (рис. 14). Данные файлы создаются автоматически при определении БД.

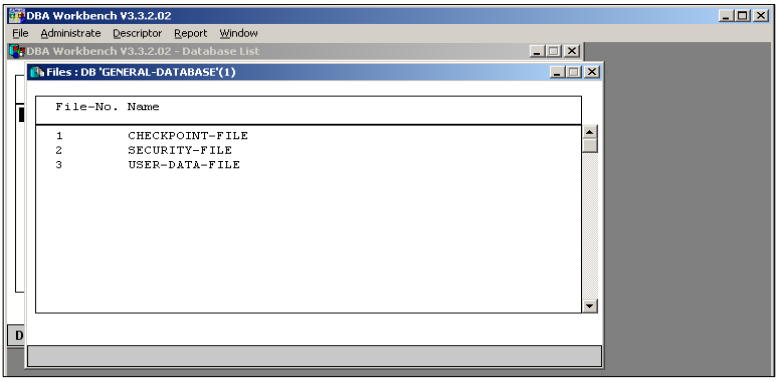

Рис. 14. Список уже существующих файлов БД

Как показано на рис. 14, системные файлы имеют первые три номера. Эти номера присваиваются по умолчанию при определении БД и могут быть изменены на другие посредством «Miscellaneous System Files…».

Далее следует выбрать в контекстном меню File → New (рис. 15).

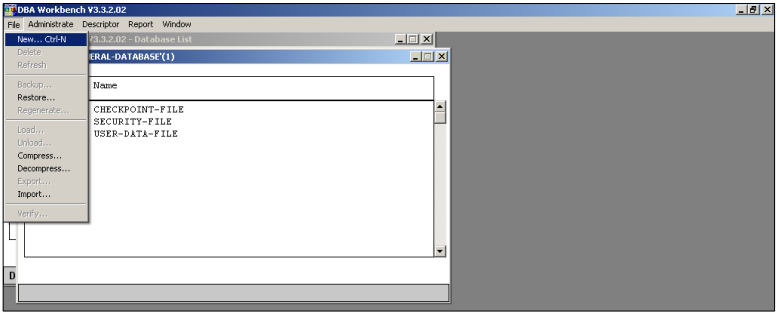

Рис. 15. Создание нового файла БД

После проведения данной процедуры открывается окно редактора FDT (рис. 16). При помощи данного редактора осуществляется определение составляющих файла (записи и элементов) посредством ввода следующей информации в текстовые поля и поля со списком: тип элемента, уровень элемента, сокращенное имя элемента, длина элемента (в байтах), формат данных элемента

и опции ограничения элемента записи. Кроме того, любой из элементов, определяемых в рамках файла, может быть объявлен дескриптором (простым дескриптором, фонетическим дескриптором, субдескриптором, супердескриптором или гипердескриптором).

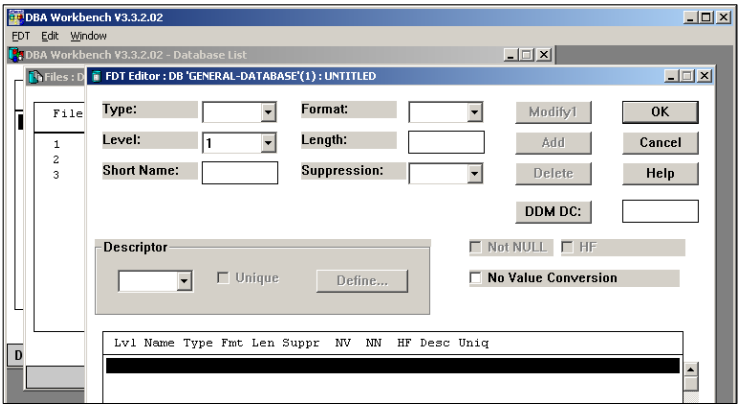

Рис. 16. Окно редактора БД

После определения составляющих файла для сохранения всех введенных значений элементов в структуре файла необходимо нажать ‘OK’. При этом открывается окно ввода параметров необходимых для создания файла (Create File) (рис. 17).

Для успешного определения файла необходимо ввести ряд следующих параметров в данном окне:

– номер файла (File Number). В данном поле отображается следующий свободный порядковый номер файла. Можно оставить предлагаемое значение либо ввести иной номер. При этом следует учитывать, что номер не должен превышать максимальное число файлов, определенных для данной БД, а также должен быть отличным от номеров уже существующих файлов и номеров системных файлов, отображаемых в списке файлов БД;

– название файла (File Name). Название файла, введенное в данное поле, будет отображаться в списке файлов БД и отчетах;

– максимальный ISN (Maximum ISN). Для дальнейшего определения файла необходимо указать предполагаемый наибольший внутрисистемный номер (Internal Sequence Number —

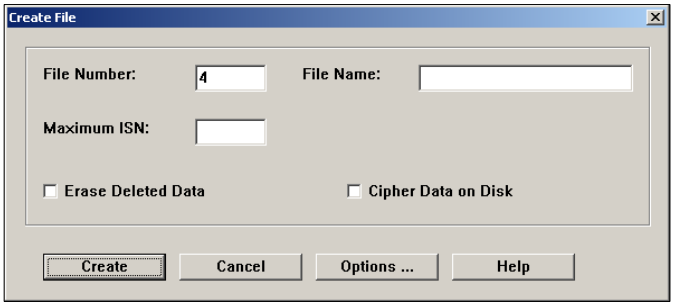

Рис. 17. Сохранение значений элементов в структуре файлов

ISN) в текстовом поле ‘Maximum ISN’. Следует учитывать, что от значения данного параметра напрямую зависит размер дискового пространства, которое будет отведено для адресного преобразователя (Address Converter) при создании файла;

– опция «Стереть удаленные данные» (Erase Deleted Data). Данная опция используется для замещения индексов и блоков хранения данных нулевыми значениями при их возвращении к свободному табличному пространству (удалении);

– опция «Шифровать данные на диске» (Cipher Data on Disk). Опция используется для включения функции шифрования данных.

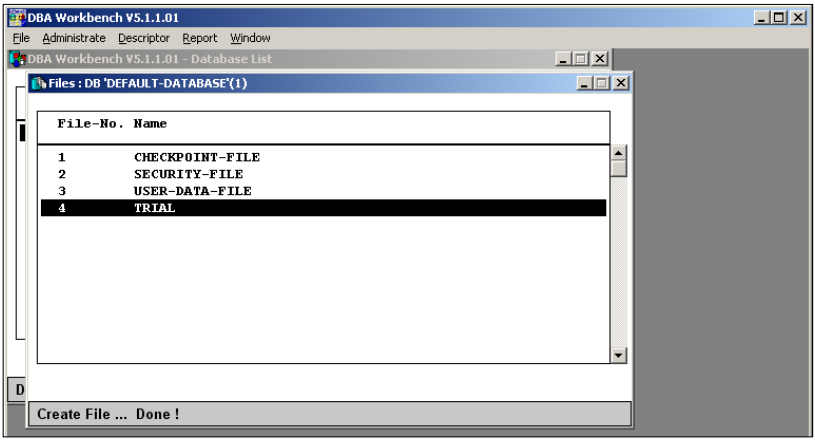

Рис. 18. Список файлов БД после создания нового файла БД

Для создания файла после ввода перечисленных параметров следует выбрать ‘Create’. При этом параметры элементов файла и характеристики структуры данных создаваемого файла будут сохранены установленными по умолчанию. Если все параметры были введены корректно, в списке файлов БД будут отображены номер и имя созданного файла (рис. 18).

3.3. Определение записи

Под записью в данном случае понимается совокупность всех полей, определенных в рамках файла. Под определением записи понимается последовательное определение каждого из полей (атрибутов) посредством редактора FDT при определении файла. Каждое из полей (атрибут) определяется рядом обязательных параметров, в то время как часть параметров назначается по выбору. При определении файла в рамках существующей БД проектировщик БД вызывает редактор FDT (рис. 19).

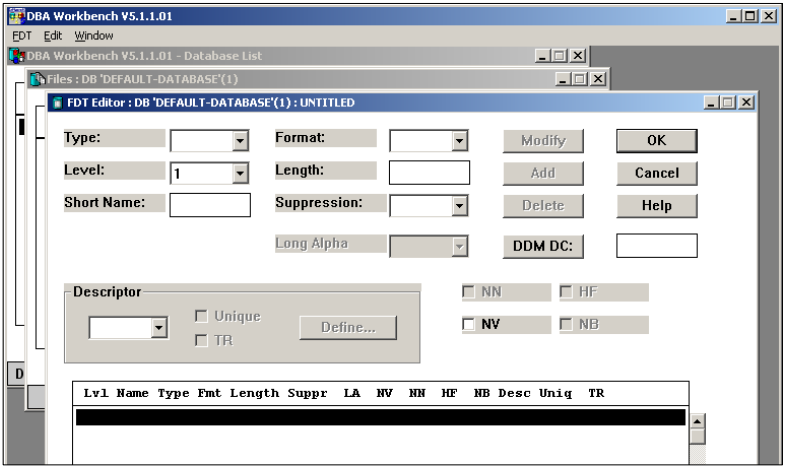

Рис. 19. Создание записи при помощи редактора FDT

Далее для каждого поля (атрибута) при помощи текстовых полей и полей со списком вводятся значения следующих параметров.

Тип (Type)

Допускается пустое значение. Следует из списка поля ‘Type’ выбрать необходимый тип определяемого атрибута. По умолчанию системой предлагается пустое значение. Обозначениям данного параметра соответствуют следующие значения (рис. 20):

– пустое значение — простое (элементарное) поле;
– GR — группа;
– MU — множественное поле;
– PE — периодическая группа.

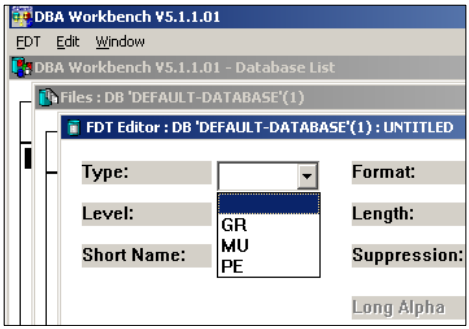

Рис. 20. Тип определяемого атрибута

Опция MU указывает, что поле может содержать больше одного значения в отдельной записи. Фактическое количество значений, присутствующих в каждой записи, может меняться от 0 до 191, но по крайней мере одно значение (даже пустое) должно присутствовать в каждой входной записи.

Значения хранятся согласно другим опциям, указанным для поля. Первое значение — это предшествующий полю счетчик, который указывает количество значений, в настоящее время присутствующих в поле. Количество хранимых значений, равное количеству значений, представленных во входной записи, плюс любые значения, добавленные во время обновления поля, меньше любых подавленных значений (это применяется, только если поле определено с опцией NU).

Если количество значений, содержащихся в каждой входной записи, постоянно, количество может быть определено в описании оператора MU в формате MU(n), где n равняется коли-

честву значений, представленных в каждой входной записи. Например:

FNDEF='01,AA,5,A,MU(3)'

показывает, что три значения множественного поля AA присутствуют в каждой входной записи. Значение ноль (0) указывает, что во входной записи никакие значения для множественного поля не присутствуют.

Если количество значений не постоянно для всех входных записей, то однобайтовый двоичный счетчик поля должен предшествовать первому значению каждой входной записи, указывая количество существующих в ней значений.

Все значения, задаваемые во время входа или обновления, будут сжаты (если только не была указана опция FI). Необходимо проявить осторожность при совместном использовании опций FI и MU: если присутствует большое количество сжимаемых значений, то можно потратить впустую большое количество памяти на диске.

Если опция NU определена с опцией MU, то нулевые значения будут подавлены и логически, и физически. Позиционность всех значений (включая пустые значения) поддерживается в MU реализации, если не определена опция NU. Если большое количество нулевых значений присутствует в группе поля MU, опция NU может уменьшать требования дисковой памяти для поля, но не должна использоваться, если должны поддерживаться относительные позиции значений.

Опция NC (или NC/NN) не может быть определена для поля с опцией MU.

Опция PE указывает, что должна быть определена периодическая группа. Периодическая группа может состоять из одного или более полей, и в пределах данной записи могут встречаться от 0 до 99 реализаций (или 191, если определен параметр утилиты ADACMP MAXPE191), но по крайней мере одна реализация (даже если она содержит все нулевые значения) должна присутствовать в каждой входной записи.

Периодическая группа должна быть описана на уровне 01. Все поля, которые должны содержаться в периодической группе, должны следовать тотчас после и должны быть описаны на уровне 02 или выше (с приращением 1 до максимального 7 уровня).

Следующее описание уровня 01 указывает на конец текущей периодической группы.

Опция PE может быть описана только с именем группы. Параметры длины и формата не могут быть определены с именем группы. Периодическая группа может содержать дескрипторы и/или множественные поля и другие группы, но не может содержать другую периодическую группу. Далее следуют примеры:

```
FNDEF='01,GA,PE'              периодическая группа GA
FNDEF='02,A1,6,A,NU'
FNDEF='02,A2,2,B,NU'
FNDEF='02,A3,4,P,NU'
FNDEF='01,GB,PE(3)'           периодическая группа GB
FNOEF='02,B1,4,A,DE,NU'
FNDEF='02,B2,5,A,MU(2),NU'
FNDEF='02,B3'
FNDEF='03,B4,20,A,NU'
FNDEF='03,B5,7,U,NU'
```

Номер уровня (Level)

Не допускается пустое значение. Значение, устанавливаемое по умолчанию — 1. Область допустимых значений — 1−7. Ввести необходимое значение уровня можно либо с помощью клавиатуры, либо выбрав соответствующий уровень из списка. Номера уровней нельзя пропускать, когда назначаются номера уровней для вложенных групп.

Номер уровня — это число, состоящее из одной или двух цифр в диапазоне 01−07 (лидирующие нули необязательны), используемое для построения группы полей. Поля, имеющие номер уровня 02 или больше, входят в состав предшествующей группы, которой назначен более низкий номер уровня.

Описание группы позволяет ссылку на ряд полей (может быть только 1 поле), используя имя группы. Это обеспечивает удобный и эффективный метод ссылки на ряд логических полей.

Для определения группы могут использоваться номера уровней 01−06. Группа может состоять из других групп. Номера уровней нельзя пропускать, когда назначаются номера уровней для вложенных групп.

FNDEF='01,GA' группа
FNDEF='02,A1,...' элементарное или множественное поле
FNDEF='02,A2,...' элементарное или множественное поле
FNDEF='01,GB' группа
FNDEF='02,B1,...' элементарное или множественное поле
FNDEF='02,GC' группа (вложенная)
FNDEF='03,C1,...' элементарное или множественное поле
FNDEF='03,C2,...' элементарное или множественное поле

Поля A1 и A2 находятся в группе GA. Поле B1 и группа GC (состоящая из полей C1 и C2) находятся в группе GB.

Сокращенное имя (Short Name)

Не допускается пустое значение. По умолчанию значение не устанавливается. Имя должно быть уникальным в пределах файла, а также должно быть двухсимвольным. Первый символ обязательно алфавитный, второй символ может быть как символьным, так и числовым. Специальные символы не допускаются. Также нельзя использовать значения E0−E9, так как они зарезервированы системой в качестве редактирующих масок.

Правильные имена: AA, B4, S3, WM

Неверные имена:

A (один символ)

E3 (маска редактирования)

F* (присутствует специальный символ)

6M (первый символ цифра)

Формат (Format)

Не допускается пустое значение. По умолчанию значение не устанавливается. В поле со списком (рис. 21) предлагаются следующие значения параметра «Формат»:

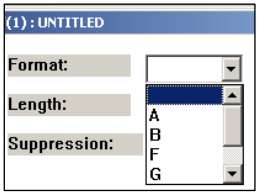

Рис. 21. Формат определяемого элемента

A — алфавитно-цифровой (выровненный влево);

B — двоичный (выровненный вправо, без знака / положительный);

F — с фиксированной точкой (выровненный вправо, со знаком, с использованием двойного дополнения для отрицательных чисел);

G — с плавающей точкой (нормализованная форма, со знаком);

P — упакованный десятичный (выровненный вправо, со знаком);

U — распакованный десятичный (выровненный вправо, со знаком).

Стандартный формат (по умолчанию) используется ADABAS во время обработки команды. Он вводится в таблицу описания полей и используется при чтении/обновлении поля до тех пор, пока пользователь не определит значение, перекрывающее стандартный формат.

Стандартный формат определяется для поля. Он не может быть определен для имени группы. Когда группа читается (записывается), поля в пределах группы всегда возвращаются (должно быть обеспечено) согласно стандартному формату каждого индивидуального поля. Указанный формат определяет тип сжатия, которое будет выполнено для данного поля.

Поле с фиксированной точкой имеет длину два или четыре байта. Положительное значение находится в нормальной форме, а отрицательное значение в форме двойного дополнения.

Формат поля с плавающей точкой может быть длиной четыре байта (одноразрядное) или восемь байт (двухразрядное). Поддерживается преобразование значения поля, определенного с плавающей точкой, в другой формат.

Если двоичное поле должно быть дескриптором и поле может содержать положительные и отрицательные числа, то вместо формата B должен использоваться формат F, потому что формат B принимает эти значения без знака (как положительные).

Длина (Length)

Допускается пустое значение для имени группы. При определении простого поля указание длины обязательно. По умолчанию значение не устанавливается.

Значение длины используется для того, чтобы определить стандартную длину (по умолчанию), которую использует ADABAS во время обработки команды. Указанная стандартная длина вводится в таблицу описания полей (FDT) и используется при чтении/обновлении поля до тех пор, пока пользователь не определит значение, перекрывающее ее.

Максимальные длины полей, которые могут быть определены, зависят от формата (табл. 2).

Таблица 2

Максимальная длина полей в зависимости от формата

Формат	Максимальная длина
Алфавитно-цифровой (A)	253 байта
Двоичный (B)	126 байтов
С фиксированной точкой (F)	4 байта (всегда 2 или 4 байта)
С плавающей точкой (G)	8 байтов (всегда 4 или 8 байтов)
Упакованный десятичный (P)	15 байтов
Распакованный десятичный (U)	29 байтов

Для имени группы не может быть определена стандартная длина. Стандартная длина не ограничивает размер любого данного значения поля, если только не используется опция FI. Команды чтения или добавления могут перекрывать стандартную длину поля до максимальной длины, разрешенной для этого формата. Если стандартная длина поля нулевая, то поле принимается как поле переменной длины. Поля переменной длины не имеют стандартной (по умолчанию) длины. В этом случае длина поля будет принимать значения: для полей с фиксированной точкой (F) — только два или четыре байта; для полей с плавающей точкой (G) — только четыре или восемь байтов.

Если во время выполнения команды ADABAS производится обращение к полю переменной длины без указания длины, то будет возвращено значение этого поля, которому предшествует однобайтовая двоичная длина поля (включая байт длины непосредственно). Это значение длины должно быть определено при обновлении поля и также во входных записях. Если поле определено с опцией длинное алфавитно-цифровое (long alpha — LA), значению предшествует двухбайтная двоичная длина поля (включая два байта длины).

Подавление (Suppression)

Допускается и устанавливается по умолчанию пустое значение. Параметры данного поля определяют системные функции

подавления нулевых и прочих значений при вводе записи в созданную БД. В поле со списком (рис. 22) предлагаются следующие значения параметра:

– пустое значение — подавление по умолчанию;

– FI — фиксированная длина пространства памяти;

– NU — подавление нулевого значения;

– NC — SQL опция нулевого значения.

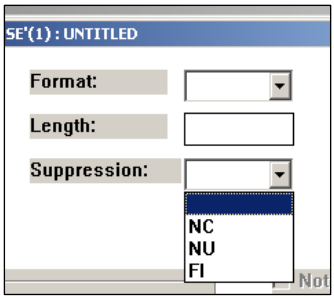

Рис. 22. Функция подавления

FI — фиксированная длина пространства памяти. Опция указывает, что поле должно иметь фиксированную длину пространства памяти. Значения в поле хранятся без внутреннего байта длины, не сжаты и не могут быть длиннее, чем заданная длина поля.

Опция FI рекомендуется для полей длиной в один или два байта, которые имеют низкую вероятность содержания пустого значения (количество персонала, пол и т. д.), и для полей, содержащих значения, которые не могут быть сжаты. Опция не рекомендуется для множественных полей или для полей в пределах периодической группы. Любые пустые значения для таких полей не подавляются (или сжимаются), что может вызвать значительные затраты дисковой памяти и увеличение времени обработки.

Опция FI не может быть определена для:

– полей U-формата;

– полей с опций NC, NN или NU;

– полей переменной длины, определенных с нулевой длиной (0);

– дескрипторов внутри периодической (PE) группы.

Поле, определенное с опцией FI, не может быть обновлено значением, которое превышает стандартную длину поля.

Пример использования опции FI показан в табл. 3.

Таблица 3

Пример использования опции FI

Вариант использования	Описание	Данные пользователя	Внутреннее представление
Без опции FI	FNDEF='01,AA,3,P'	33104C 00003C	0433104F (4 байта) 023F (2 байта)
С опцией FI	FNDEF='01,AA,3,P,FI'	33104C 00003C	33104F (3 байта) 00003F (3 байта)

NU — подавление нулевого значения. Опция подавляет нулевые значения, встречающиеся в поле.

Нормальное сжатие (опции NU или FI не определены) приводит к представлению нулевого значения двумя байтами (один байт для значения длины, второй — непосредственно для значения, в этом случае нулевого). Применение опции подавления нулевой величины приводит к внутреннему представлению нулевой величины индикатором однобайтного «нулевого поля». Само нулевое значение не хранится.

Ряд последовательных полей, содержащих нулевые значения и определенных с опцией NU, представляются однобайтовым «пустым полем» индикатором (двоичное представление 11nnnnnn), где nnnnnn — количество соседних байтов внутри поля, содержащих нулевые значения, общим числом до 63. По этой причине поля, определенные с опцией NU, должны быть по возможности сгруппированы вместе.

Если опция NU определена для дескриптора, то все нулевые значения для этого дескриптора не хранятся в инвертированном списке. Поэтому результатом команды FIND (редактор Natural), в которой используется этот дескриптор и для которой нулевое значение используется для поиска, всегда будет отсутствие выбранных записей. Даже в том случае, если в памяти данных имеются записи, которые содержат нулевое значение для дескриптора. Если дескриптор, определенный с опцией NU, используется для управления логической последовательностью в команде чтения в логической последовательности (L3/L6), записи, которые содержат нулевое значение для дескриптора, не будут читаться.

Дескриптор, который нужно использовать в качестве основы для связи файла и для которого существует большое коли-

чество пустых значений, должен быть определен с опцией NU, чтобы уменьшить общий размер списков связи.

Опция NU не может быть определена для полей, определенных с объединенными опциями NC/NN или с опцией FI.

Пример использования опции NU (табл. 4) — C1 указывает на 1 пустое поле.

Таблица 4

Пример использования опции NU

Вариант использования	Описание	Данные пользователя	Внутреннее представление
Нормальное сжатие	FNDEF='01,AA,2,B'	0000	0200 (2 байта)
С опцией FI	FNDEF='01,AA,2,B,FF	0000	0000 (2 байта)
С опцией NU	FNDEF='01,AA,2,B,NU'	0000	C1 (1 байт)

NC — SQL опция нулевого значения. В поле, которое не определено с опцией NC (не подсчитывается), нулевое значение является или нулем, или пробелом, в зависимости от формата поля.

В поле, которое определено с опцией NC, нули или пробелы, определенные в буфере записи, интерпретируются по-разному в зависимости от значения «нулевого индикатора»: или как истинные нули или пробелы (как «значащие» нули), или как неопределенные значения (как истинный SQL или «незначащие» нули).

Если поле, определенное с опцией NC, не имеет никакого значения, указанного в буфере записи, значение поля всегда обрабатывается как SQL нулевое значение.

Когда интерпретируется как истинный SQL нуль, нулевое значение удовлетворяет SQL интерпретацию поля, не имеющего никакого значения. Это означает, что значение поля не было введено; т. е. значение поля не определено.

Значение нулевого индикатора ответственно за внутреннее ADABAS представление нуля. Это значение всегда имеет длину два байта и формат с фиксированной точкой, независимо от формата данных. Он определяется в буфере записи, когда значение поля добавляется или обновляется, и возвращается в буфер записи, когда значение поля читается.

Для команд Update (обновления) (Ax) или Add (добавления) (Nx) значение нулевого индикатора должно быть установлено в буфере записи в положение, которое соответствует обозначению полей в форматном буфере. Установка (шестнадцатеричное значение) должна быть одной из следующих:

FFFF — значение поля установлено в «не определено», незначащий нуль; различия между «нет значения», двоичные нули или пробелы для поля в буфере записи игнорируются; все интерпретируются одинаково как «нет значения»;

0000 — нет значения, двоичные нули или пробелы для поля в буфере записи интерпретируются как незначащие нулевые значения.

Для команд Read (чтения) (Lx) или Find с Read (поиска с чтением) (Sx с форматным буфером) ваша программа должна проверять значение нулевого индикатора, возвращенного в позицию буфера записи, соответствующую позиции поля в форматном буфере. Нулевой индикатор может принимать следующие значения:

FFFF — ноль или пробел в поле не важны;

0000 — ноль или пробел в поле — значащее значение, т. е. истинный ноль или пробел;

xxxx — поле усечено, значение нулевого индикатора содержит длину (xxxx) полного значения, которое хранится в записи базы данных.

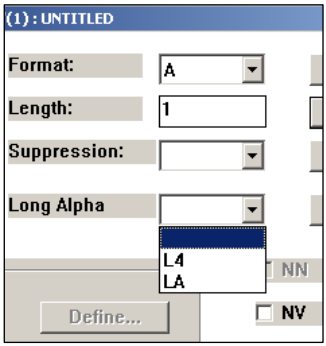

Рис. 23. Функция Long Alpha

«Длинные значения» (Long Alpha)

Предлагаются следующие опции для данного параметра (рис. 23):

L4 — длинное алфавитно-цифровое поле (4 байта);

LA — длинное алфавитно-цифровое поле (2 байта).

Опции L4 и LA могут использоваться для полей алфавитно-цифрового формата. При этом такое поле может содержать значение длиной до 16,381 байт.

Дескриптор (Descriptor)

Допускается и устанавливается по умолчанию пустое значение. Любое поле при определении FDT может быть определено дескриптором. Для поля, определенного дескриптором, будет сделан вход в инвертированном списке ассоциатора, который позволит этому полю: использоваться в выражении поиска и в качестве ключа сортировки в команде FIND, управлять последовательным логическим чтением или использоваться для связи файла.

Если дескрипторное поле не инициализировано и логически падает мимо конца физической записи, вход инвертированного списка для той записи не формируется по причине производительности. Опция дескриптора должна использоваться осторожно, особенно если файл большой и поле, которое рассматривается как дескриптор, часто обновляется.

В процессе определения записи предлагаются следующие опции для параметра «Дескриптор» (рис. 24):

DE — дескриптор;

SB — субдескриптор;

SP — супердескриптор;

HY — гипердескриптор;

PH — фонетический дескриптор.

При определении всех типов дескриптора (кроме фонетического) может быть использована опция уникального дескриптора (Unique), а для определения субдескрипора, супердескриптора, гипердескриптора и фонетического дескриптора используется альтернатива Define…

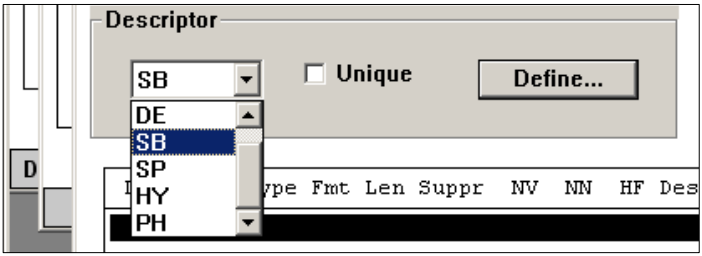

Рис. 24. Определение дескриптора

SB — описание субдескриптора

Субдескриптор — это дескриптор, созданный из части элементарного поля. Элементарное поле может быть или не быть дескриптором непосредственно. Субдескриптор может также использоваться как субполе, т. е. он может быть определен в форматном буфере для определения формата выходных записей.

Субдескриптор должен быть определен со следующими параметрами:

– имя — правила присвоения имени субдескриптору идентичны тем, которые используются для имени поля ADABAS;

– UQ (опция Unique) — указывает, что субдескриптор должен быть определен как уникальный;

– исходное поле — имя исходного поля, из которого будет получен субдескриптор;

– начало — относительная позиция байта в пределах исходного поля, где начинается описание субдескриптора;

– конец — относительная позиция байта в пределах исходного поля, где кончается описание субдескриптора.

Подсчет параметров «начало» и «конец» делается слева направо начиная с 1 для алфавитно-цифровых полей и справа налево начиная с 1 для цифровых и двоичных полей. Если исходное поле определено с форматом P, знак результирующего значения субдескриптора будет взят из 4 битов младшего разряда байта младшего разряда (1 байт).

Исходное поле может быть:

– дескриптором;

– элементарным полем;

– множественным полем (но не конкретная реализация множественного поля);

– входить в состав периодической группы (но не конкретная реализация периодической труппы).

Исходное поле не может быть:

– суб/суперполем, субдескриптором, супердескриптором или фонетическим дескриптором;

– G формата (с плавающей точкой).

Если поле определено с опцией NU, никакие входы не генерируются в инвертированном списке субдескриптора для тех записей, которые содержат нулевое значение (в пределах опреде-

ленных байт позиций) для поля. Формат тот же самый, что и для исходного поля.

Если исходное поле не инициализировано и логически падает мимо конца физической записи, вход инвертированного списка для этой записи не формируется по причине производительности. В этом случае для создания входа инвертированного списка необходимо файл разгрузить с опцией SHORT, разуплотнить и загрузить повторно или использовать прикладную программу для инициализации поля каждой записи файла.

Синтаксис субдескриптора должен быть описан посредством использования альтернативы ‘Define’ (рис. 25).

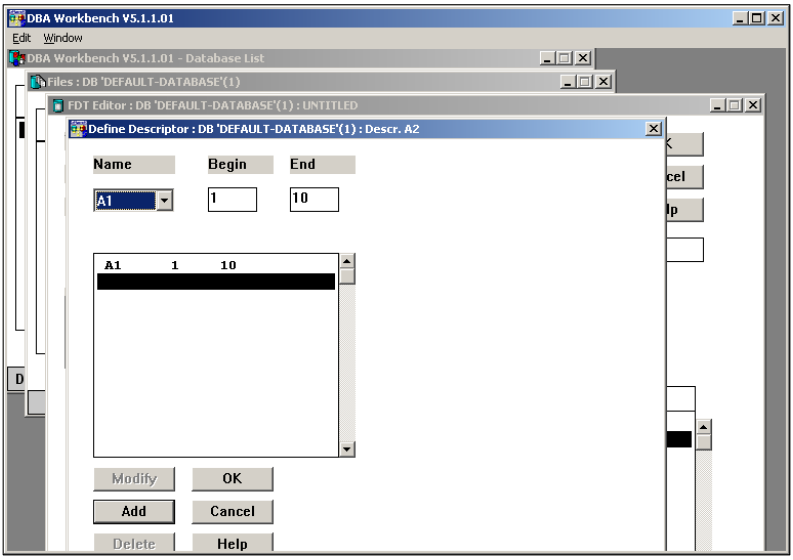

Рис. 25. Определение синтаксиса субдескриптора

В окне «Define Descriptor» необходимо определить исходное поле, начало и конец субдескриптора:

– имя исходного поля (Name). Посредством поля со списком ‘Name’ (отображаются уже определенные поля) необходимо выбрать исходное поле для субдескриптора;

– начало (Begin). В поле «Begin» указывается параметр «начало субдескриптора»;

– конец (End). В поле «End» указывается параметр «конец субдескриптора»;

– альтернатива «добавить» (Add). После определения параметров имени исходного поля, начала и конца для определения субдескриптора необходимо выбрать альтернативу «Add». При этом запись определенного субдескриптора отображается в соответствующем поле;

– альтернатива «изменить» (Modify). Для изменения указанных параметров субдескриптора следует использовать опцию «Modify»;

– альтернатива «удалить» (Delete). Для удаления ранее указанных параметров субдескриптора следует использовать опцию «Delete».

Для сохранения параметров субдескриптора нужно воспользоваться системной кнопкой «OK».

При определении субдескриптора следует учитывать, что в качестве параметра «исходное поле», как следует из свойств субдескриптора, может быть выбрано только одно поле из уже определенных. В противном случае значение данного параметра не будет принято системой, о чем проектировщик БД будет уведомлен сообщением (рис. 26).

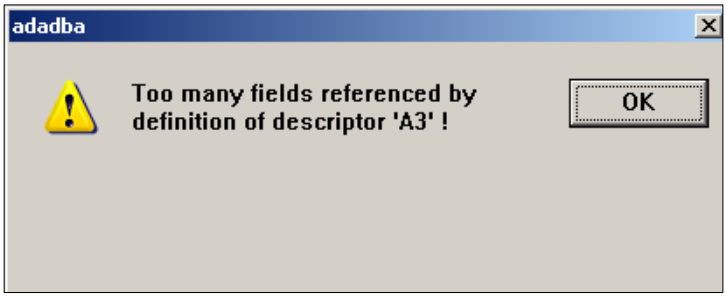

Рис. 26. Информационное сообщение,
получаемое при выборе нескольких исходных полей

Определение параметров «имя» и UQ (Unique) субдескриптора осуществляется при помощи основного окна редактора FDT.

SP — описание супердескриптора

Супердескриптор — это дескриптор, созданный из нескольких полей, частей полей или их комбинации. Каждое полепервоисточник (или часть поля) используемое для определения супердескриптора, называется исходным. Супердескриптор может быть определен с использованием от 2 до 20 исходных полей или частей поля. Супердескриптор может быть определен как уникальный дескриптор. Супердескриптор может также использоваться как суперполе, т. е. он может быть определен в форматном буфере для определения формата выходной записи.

Суперскриптор должен быть определен со следующими параметрами:

– имя — правила присвоения имени супердескриптору идентичны тем, которые используются для имени поля ADABAS;

– UQ (опция Unique) — указывает, что супердескриптор должен быть определен как уникальный;

– исходное поле — имя исходного поля, из которого будет получен элемент супердескриптора; может быть определено до 20 исходных полей;

– начало — относительная позиция байта в пределах исходного поля, где начинается описание супердескриптора;

– конец — относительная позиция байта в пределах исходного поля, где заканчивается описание супердескриптора.

Подсчет параметров «начало» и «конец» делается слева направо начиная с 1 для алфавитно-цифровых полей и справа налево начиная с 1 для цифровых и двоичных полей. Для любого исходного поля, кроме тех, которые определены как FI, значения начала и конца имеют силу в пределах разрешенного диапазона для типа данных исходного поля.

Исходное поле может быть:

– элементарным или максимумом одним множественным полем (но не определенного значения множественного поля);

– в пределах периодической группы (но не определенная реализация);

– дескриптором.

Исходное поле не может быть:

– супер-, суб- или фонетическим дескриптором;

– G формата (с плавающей точкой);

– поле опции NC, если другое исходное поле — поле опции NU;

– длинным алфавитно-цифровым (LA) полем.

Если исходное поле определено с опцией NU, никакие входы не генерируются в инвертированном списке супердескриптора для записей, содержащих нулевые значения для поля. Это действительно независимо от присутствия или отсутствия значений для других элементов супердескриптора.

Если исходное поле не инициализировано и логически падает мимо конца физической записи, вход инвертированного списка для этой записи не формируется по причине производительности. В этом случае для создания входа инвертированного списка необходимо файл разгрузить с опцией SHORT, разуплотнить и загрузить повторно или использовать прикладную программу для инициализации поля каждой записи файла.

Полная длина любого значения супердескриптора не может превышать 253 байта (алфавитно-цифровые) или 126 (двоичных) байтов. Супердескриптор имеет алфавитно-цифровой формат, если какой-нибудь элемент супердескриптора получен из алфавитно-цифрового исходного поля; иначе супердескриптор имеет двоичный формат.

Все супердескрипторы двоичного формата обрабатываются как числа без знака. Синтаксис супердескриптора должен быть описан посредством использования альтернативы «Define» аналогично описанию синтаксиса субдескриптора.

При определении супердескриптора, в отличие от субдескриптора, в качестве исходных могут использоваться от 2 до 20 уже определенных в таблице FDT полей.

Определение параметров «имя» и UQ (Unique) супердескриптора осуществляется при помощи основного окна редактора FDT.

HY — описание гипердескриптора

Опция гипердескриптора позволяет сгенерировать значения дескриптора на основании алгоритма, обеспеченного пользователем.

Значения основаны на алгоритмах, закодированных в специальных пользовательских выходах гипердескриптора (от HEX01

до HEX31). Каждому гипердескриптору должен быть назначен пользовательский выход, и отдельный пользовательский выход может обрабатывать множество гипердескрипторов (рис. 27).

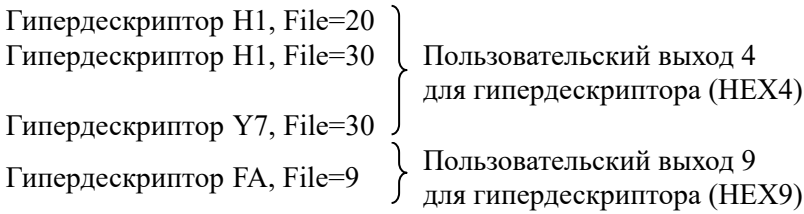

Рис. 27. Гипердескриптор и пользовательские выходы

Входные параметры необходимые для пользовательского выхода:

– имя гипердескриптора;

– номер файла;

– адреса полей, взятых из записи памяти данных, вместе с именем поля и PE индексом (если применяется). Эти адреса указывают на сжатые значения полей.

Пользовательский выход должен возвратить значение дескриптора (DVT) в сжатом формате. Может быть не возвращено никакого значения или одно или большее количество значений в зависимости от опций (PE, MU), определенных для гипердескриптора.

Исходный ISN, назначенный для входного значения, может быть изменен.

При использовании гипердескрипторов нужно учитывать следующие примечания:

– проектировщик определяет формат, длину и опции гипердескриптора. Они не могут быть взяты от исходного поля, определенного в параметре HY;

– поиск, использующий значение гипердескриптора, выполняется тем же способом, что и для стандартных дескрипторов;

– проектировщик ответствен за создание корректных значений гипердескриптора. Нет никакого стандартного пути проверки значений гипердескриптора на целостность в памяти данных. Пользователь должен установить правила описания каждого значения и проверки корректности значения;

– если формат гипердескриптора упакованный или распакованный, ADABAS проверяет возвращенные значения на правильность. Знак полбайта для упакованного значения может содержать A, C, E, F (положительный) или B, D (отрицательный). ADABAS преобразует знак в F или D;

– если файл содержит больше одного гипердескриптора, назначенные выходы вызываются в алфавитном порядке имен гипердескрипторов.

Гипердескриптор должен быть определен со следующими параметрами:

– номер — ядро ADABAS использует этот номер для определения пользовательского выхода гипердескриптора, который будет вызван;

– имя — правила присвоения имени гипердескриптору идентичны тем, которые используются для имени поля ADABAS;

– длина — по умолчанию.

– формат — форматы идентичны тем, которые используются для имени поля ADABAS;

– опция — вместе с гипердескриптором могут использоваться следующие опции: MU — множественное поле; NU — подавление нулевого значения; PE — поле периодической группы; UQ — уникальный дескриптор; SV — создание поиска значений;

– исходное поле — гипердескриптор может иметь от 1 до 20 исходных полей. Имена полей и адреса передаются пользовательскому выходу. Если исходное поле определено с опцией NU, никакие входы не генерируются в инвертированном списке гипердескриптора для записей, содержащих нулевые значения поля. Это не зависит от присутствия или отсутствия значений для других элементов гипердескриптора.

Если исходное поле не инициализировано и логически оно попадает мимо конца физической записи, то вход инвертированного списка для этой записи не формируется по причине производительности. В этом случае для создания входа инвертированного списка необходимо файл разгрузить с опцией SHORT, разуплотнить и загрузить повторно или использовать прикладную программу для инициализации поля каждой записи файла.

Синтаксис гипердескриптора должен быть описан посредством использования альтернативы «Define» (рис. 28).

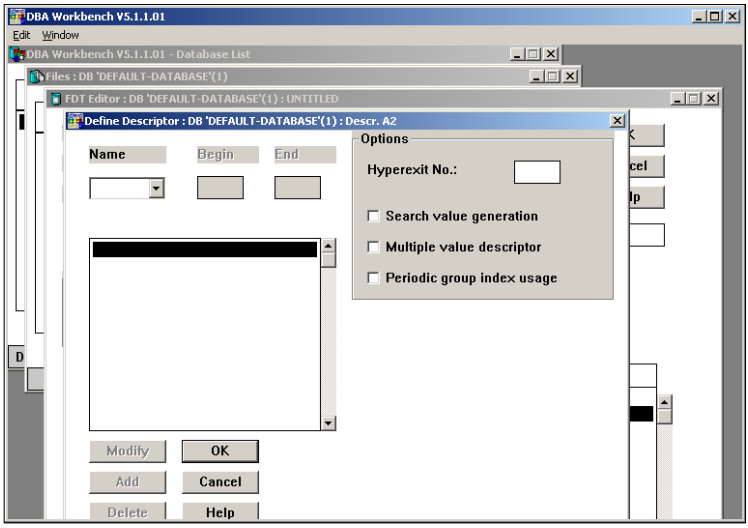

Рис. 28. Определение синтаксиса гипердескриптора

В окне «Define Descriptor» определяются следующие параметры гипердескриптора:

– имя исходного поля (Name). Посредством поля со списком ‘Name’ (отображаются уже определенные поля) необходимо выбрать исходное поле (исходные поля) для гипердескриптора;

– начало (Begin) и конец (End). В полях «Begin» и «End» автоматически устанавливается полная длина исходных полей гипердескриптора. Для проектировщика эти поля являются неактивными и не подлежат изменению;

– номер пользовательского выхода (Hyperexit No.). Посредством данного поля необходимо установить параметр «номер гипердескриптора»;

– опции (Options). Здесь определяются такие составляющие параметра «опция» как MU — альтернатива «Multiple value descriptor», PE — альтернатива «Periodic group index usage» и SV — альтернатива «Search value generation»;

– альтернатива «добавить» (Add). После определения параметров гипердескриптора, необходимо выбрать альтернативу «Add». При этом запись определенного гипердескриптора отображается в соответствующем поле;

– альтернатива «изменить» (Modify). Для изменения указанных параметров гипердескриптора следует использовать опцию «Modify»;

– альтернатива «удалить» (Delete). Для удаления ранее указанных параметров гипердескриптора следует использовать опцию «Delete».

Для сохранения параметров гипердескриптора нужно воспользоваться системной кнопкой «OK».

Определение параметров «имя», «длина», «формат» гипердескриптора, а также таких составляющих параметра «опция», как UQ и NU, осуществляется при помощи основного окна редактора FDT, аналогично определению параметров «имя», UQ (Unique) суб- и супердескриптора.

Для лучшего понимания процедуры определения гипердескриптора ниже приведен пример его описания (табл. 5).

Описание гипердескриптора:

HYPDE=’2,NH,60,A,MU,NU=LN,FN,FR’

Гипердескриптору назначен пользовательский выход 2 и имя гипердескриптора — HN.

Длина гипердескриптора — 60, формат — алфавитно-цифровой и множественное (MU) поле с подавлением нулей (NU).

Значения для гипердескриптора должны быть получены из полей LN, FN и FR.

Таблица 5

Пример описания гипердескриптора

Описание определяемых полей	Значения полей
FNDEF='01,LN,20,A,DE,NU'	Фамилия
FNDEF='01,FN,20,A,MU,NU'	Имя
FNDEF='01,ID,4,B,NU'	Идентификатор
FNDEF='01,AG,3,U'	Возраст
FNDEF='01,AD,PE'	Адрес
FNDEF='02,CI,20,A,NU'	Город
FNDEF='02,ST,20,A,NU'	Улица
FNDEF='01,FA,PE'	Родственники
FNDEF='02,NR,20,A,NU	Фамилия родственника
FNDEF='02,FR,20,A,MU,NU'	Имя родственника

PH — фонетический дескриптор

В результате выполнения команды FIND (редактор Natural) с использованием фонетического дескриптора все возвращенные записи будут содержать подобные фонетические значения (иметь аналогичное произношение). Фонетическое значение дескриптора основано на первых 20 байтах значения поля. Рассматриваются только алфавитные значения; числовые значения, специальные символы и пробелы игнорируются. Нижний и верхний регистры алфавитно-цифровых символов внутренне идентичны.

Фонетический дескриптор должен быть определен со следующими параметрами:

– имя — правила присвоения имени фонетическому дескриптору идентичны тем, которые используются для имени поля ADABAS;

– поле — имя поля, которое будет транскрибировано фонетически.

Поле должно быть:

– элементарным или множественным;

– определено в алфавитно-цифровом формате.

Поле может быть дескриптором.

Поле не может:

– быть суб-, супер- или гипердескриптором;

– входить в состав периодической группы;

– использоваться как исходное поле для более чем одного фонетического дескриптора.

Если поле определено с опцией NU, никакие входы не генерируются в инвертированном списке фонетического дескриптора для тех записей, которые содержат нулевое значение (в пределах определенных байтов позиций) для поля. Формат тот же самый, что и для поля.

Если поле не инициализировано и логически падает мимо конца физической записи, вход инвертированного списка для этой записи не формируется по причине производительности. В этом случае для создания входа инвертированного списка необходимо файл разгрузить с опцией SHORT, разуплотнить и загрузить повторно или использовать прикладную программу для инициализации поля каждой записи файла.

Синтаксис фонетического дескриптора должен быть описан посредством использования альтернативы «Define» (рис. 29).

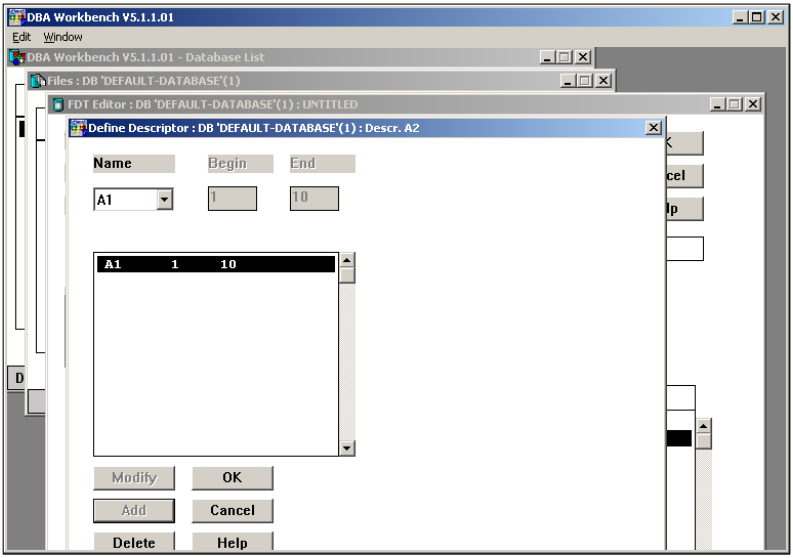

Рис. 29. Определение синтаксиса фонетического дескриптора

В окне «Define Descriptor» определяются следующие параметры фонетического дескриптора:

– имя исходного поля (Name). Посредством поля со списком «Name» (отображаются уже определенные поля) необходимо выбрать поле фонетического дескриптора;

– начало (Begin) и конец (End). В полях «Begin» и «End» автоматически устанавливается полная длина поля фонетического дескриптора. Для проектировщика эти поля являются неактивными и не подлежат изменению;

– альтернатива «добавить» (Add). После определения параметров фонетического дескриптора необходимо выбрать альтернативу «Add». При этом запись определенного фонетического дескриптора отображается в соответствующем поле;

– альтернатива «изменить» (Modify). Для изменения указанных параметров фонетического дескриптора следует использовать опцию «Modify»;

– альтернатива «удалить» (Delete). Для удаления ранее указанных параметров фонетического дескриптора следует использовать опцию «Delete».

Для сохранения параметров фонетического дескриптора нужно воспользоваться системной кнопкой «OK».

Определение параметра «имя» фонетического дескриптора осуществляется при помощи основного окна редактора FDT аналогично определению параметров «имя», UQ (Unique) суб- и супердескриптора.

Ниже приведен пример описания фонетического дескриптора:

Описание поля: FNDEF='01,AA,20,A,DE,NU'
Фонетическое описание: PHONDE='PA(AA)'

NN — SQL опция не нулевого значения

При помощи редактора FDT для каждого элементарного поля может быть задана функция NN (Not NULL).

Опция NN («не ноль» или «нулевое значение не разрешено») может быть определена, только когда опция NC также определена для поля. Опция NN указывает, что NC поле должно всегда иметь определенное значение (включая ноль или пробел); оно не может содержать «нет значения».

Опция NN гарантирует, что поле не будет оставлено неопределенным, когда запись добавлена или обновлена; поле всегда должно устанавливаться в значащее значение.

Следующий пример показывает, как незначащий ноль будет обработан в двухбайтном алфавитно-цифровом поле AA, когда оно определено с и без опции NN (табл. 6).

Таблица 6

Пример представления нулей с опцией NN

Опция	Описание поля	Значение нулевого индикатора	Внутреннее представление ADABAS
С опцией NN	01,AA,2,A,NC,NN	FFFF (незначащие нули)	Нет, ошибка
Без функции NN	01,AA,2,A,NC	FFFF (незначащие нули)	C1

Ниже следует пример корректного описания поля AA с назначением опций NC и NN:

FNDEF='01,AA,4,A,NN,NC,DE'

Далее следует пример неправильного использования опций NC/NN. Эти описания приведут к тому, что в результате будет получена ошибка:

```
FNDEF='01,AA,4,A,NC,NU'   (опции NU и NC несовместимы)
FNDEF='01,AB,4,A,NC,FI'   (опции NC и FI не совместимы)
FNDEF='01,PG,PE'          (не разрешается использовать
FNDEF='02,P1,4,A,NC'      опцию NC внутри PE группы)
```

Альтернатива «добавить» (Add). После определения каждого поля следует выбрать альтернативу «Add» в основном окне редактора FDT. При этом системой осуществляется проверка синтаксиса каждого поля. Если системой будут выявлены какие-либо ошибки, проектировщик будет осведомлен посредством сообщений. Для продолжения работы с редактором необходимо исправить все ошибки, выявленные системой.

Альтернатива «изменить» (Modify). Для внесения изменений в уже введенные параметры необходимо выделить (с помощью курсора мыши) нужное поле и воспользоваться альтернативой «Modify».

Альтернатива «удалить» (Delete). В случае когда необходимо удалить какое-либо из определенных полей, следует выделить (с помощью курсора мыши) нужное поле и воспользоваться альтернативой «Delete».

После определения параметров атрибутов следует воспользоваться системной кнопкой «OK» основного окна редактора FDT и перейти к процедуре сохранения файла.

3.4. Определение параметров ассоциатора и обработки данных

Под определением элементов файла в рамках создания БД подразумевается определение экстентов таких параметров, как хранилище данных (Data Storage, DATA), адресный конвертор

(Address Converter, AC), стандартный индекс (Normal Index, NI), верхний индекс (Upper Index, UI). Перечисленные параметры используются для указания типа и размера экстента, который будет определен. Экстенты этих параметров вводятся и корректируются при помощи опций при сохранении файла БД.

Для определения экстентов данных параметров необходимо создать файл в рамках существующей БД и определить все поля для внесения пользовательских данных. При сохранении определенного файла (при помощи окна «Create File») необходимо воспользоваться системной кнопкой «Опции» («Options…») (рис. 29).

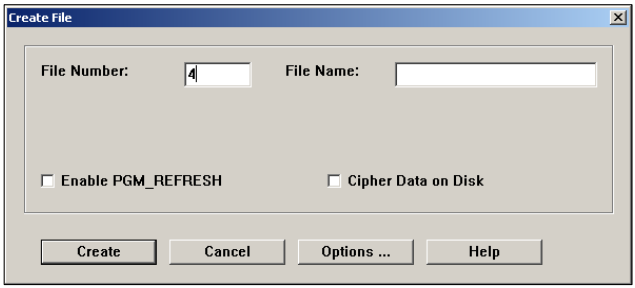

Рис. 29. Определение экстентов параметров

Далее проектировщику будет предложено окно для определения параметров (рис. 30).

В появившемся окне предлагаются следующие группы для ввода параметров:

– непрерывные (Contiguous). Для внесения значений любого из параметров — хранилище данных, адресный конвертор, стандартный индекс, верхний индекс — необходимо поставить флажок напротив соответствующего параметра;

– размер (Size). В данной группе значений указывается размер (в блоках — опция «Blocks» или мегабайтах — опция «MB») каждого из параметров — хранилище данных, стандартный индекс, верхний индекс. Размер адресного конвертора фиксирован;

– размер блока (Block Size). Для определения размера блоков каждого из параметров необходимо ввести значение (в килобайтах) в соответствующее параметру поле данной группы;

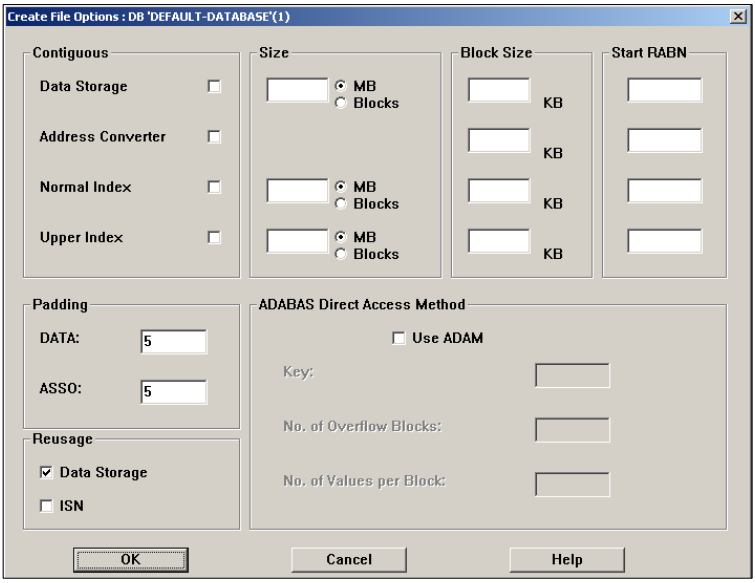

Рис. 30. Определение параметров

– начальный RABN (Start RABN). Если необходимо разместить область какого-либо из параметров на определенном дисковом сегменте или в специальном экстенте, следует ввести значение начального RABN в соответствующее параметру поле. В случае когда начальный RABN не определен, система размещает нижние ряды последовательных свободных RABN так, чтобы обеспечить достаточно свободного дискового пространства для каждого из элементов.

Если размеры и начальный RABN не определены проектировщиком, они будут автоматически назначены для каждого из вышеперечисленных параметров.

– дополнительное пространство (Padding). При определении значений дополнительного пространства для файла ассоциатора (ASSO) и файла данных (DATA) системой резервируется область дополнительного дискового пространства для записей, которые могут потребовать дополнительное место при обновлении. Это позволяет избежать перемещения записей в иные блоки. Если

ожидается незначительное (или отсутствие) обновление дескрипторов (файл ASSO) и незначительное (или отсутствие) расширение записей (файл DATA), следует установить значение данного параметра в диапазоне от 0 до 10. Если же ожидается широкое обновление дескрипторов и большое расширение записей, значение параметра необходимо определить в диапазоне от 10 до 50;

– повторное использование (Reusage). Данная опция используется для повторного использования блоков данных и номеров ISN. При использовании опции «хранение данных» (Data Storage) при добавлении новой записи или перемещении записей в результате осуществления обновлений системой используется первый найденный блок, в котором имеется достаточно пространства для размещения записи. При использовании опции «номер ISN» (ISN) номера ISN тех записей, которые были удалены, перемещаются (присваиваются иным записям), в противном случае (когда данная опция не используется) при добавлении новой записи ей присваивается следующий (наибольший) номер ISN. Особенность использования опции можно рассмотреть на примере (табл. 7).

Таблица 7

Пример использования опции «номер ISN»

Последовательность ISN		
до удаления записей и внесения новой записи	при использовании опции *Reusage номер ISN*	без использования опции *Reusage номер ISN*
…	…	…
149	149	149
150	150	152
151	151	153
152		

В первоначальном состоянии имелось несколько записей, и их номера ISN были расположены по порядку (первая колонка табл. 26). Допустим, что было удалено две записи (соответствующих номерам ISN 150 и 151) и одна запись была добавлена. Результат присвоения номеров ISN в данном случае при и без использования опции «Reusage номер ISN» показан во второй и третьей колонках табл. 7.

Метод направленного запроса ADABAS (ADABAS Direct Access Method, ADAM)

Метод направленного запроса ADABAS (ADAM) позволяет поиск записей прямо из хранилища данных без обращения к инвертированным спискам. Номер блока хранилища данных, в котором располагается запись, высчитывается на основе рандомизированного алгоритма — ADAM-ключа записи. ADAM-ключ каждой записи должен быть уникальным. Номер ISN записи также может использоваться в качестве ADAM-ключа. Для использования опции ADAM должны быть определены следующие параметры:

– использовать ADAM (Use ADAM). При введении параметров для использования в рамках метода ADAM необходимо выбрать данную опцию. Это делает поля для ввода параметров активными для проектировщика;

– ADAM-ключ (Key). В данном поле указывается сокращенное имя (Short Name) определенного в файле дескриптора либо номер ISN. В качестве ADAM-ключа не может использоваться суб-, супер-, фонетический или гипердескриптор, а также множественное поле или поле в рамках заданной периодической группы и поля, определенные с опцией NU/NC.

В качестве ADAM-ключа необходимо указывать какой-либо из уникальных дескрипторов, определенных в рамках создаваемого файла, либо номер ISN. В случае когда объявленный ADAM-ключ не является уникальным дескриптором, проектировщик будет проинформирован сообщением (рис. 31).

– количество дополнительных блоков (No. Of Overflow Blocks). При использовании опции ADAM необходимо определить количество блоков, отводимых дополнительно для осуществления операций. Требуется как минимум один блок. Количество блоков может быть увеличено в дальнейшем. Дополнительные блоки используются, когда недостаточно дискового пространства для посчитанного методом ADAM блока новой записи. Преимущество при выполнении операций с использованием метода ADAM повышается с ростом количества дополнительных блоков;

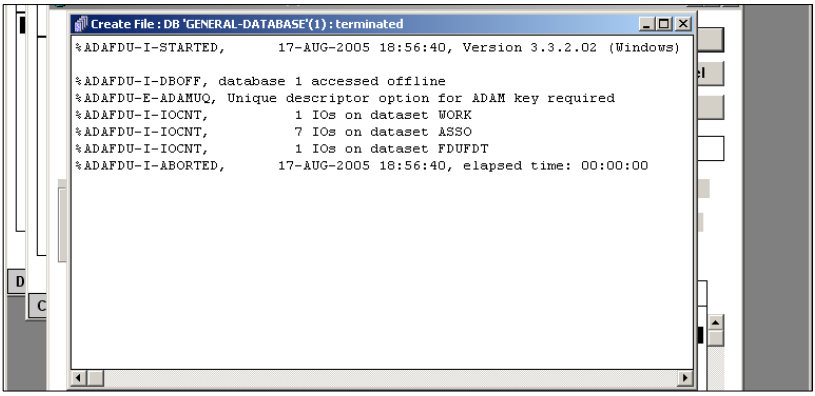

Рис. 31. Информационное сообщение, получаемое если ADAM-ключ не является дескриптором

– количество значений в блоке (No. of Values per Block). Данный параметр используется для распределения записей данных. Если ADAM-ключ определен как ISN, или дескриптор с фиксированной точкой, данный параметр определяет количество последовательных значений, которые будут храниться в одном блоке. Если ADAM-ключ является дескриптором формата алфавитно-цифрового, двоичного или с плавающей точкой, данный параметр определяет смещение с конца значения записи на 4 байта (значение, используемое как ключ).

Следующий пример наглядно показывает действие данного параметра (рис. 32).

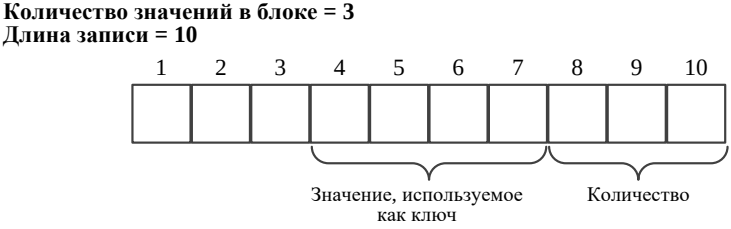

Рис. 32. Действие параметра ADAM (формата алфавитно-цифрового, двоичного или с плавающей точкой)

Если длина значения для определения смещения оказывается меньше чем 4 байта, то это значение будет расширено до нужной длины двоичными нулями.

Если формат ADAM-ключа запакованный или не запакованный, количество значений в блоке определяет смещение (в байтах) с конца значения записи до той позиции, в которой значение может быть рассмотрено в качестве значения ADAM. Это значение ADAM будет преобразовано в 4-байтовое целочисленное значение. Дальнейший пример иллюстрирует действие данного параметра для запакованного и распакованного формата ADAM-ключа (рис. 33).

Рис. 33. Действие параметра ADAM (формата запакованный или не запакованный)

Главное преимущество использования дескрипторов в качестве ADAM-ключа — понижение количества запросов к списку номеров ISN. Главное преимущество использования ISN в качестве ADAM-ключа — уменьшение количества запросов к адресному конвертору.

После определения всех параметров следует воспользоваться системной кнопкой «OK» и перейти к сохранению файла.

Выводы по главе 3

Процедура определения базы данных включает три операции: физическое размещение, создание и редактирование файлов-контейнеров базы данных (файлы с названиями ASSO, DATA, WORK). При определении БД в поле ‘Name’ необходимо ввести

название БД. Название не должно превышать 16 символов. Можно сохранить название, предлагаемое по умолчанию. Должны быть также определены идентификационные номера системных файлов БД (файлы Checkpoint File, Security File, User Data File), которые автоматически создаются и загружаются при создании БД. После определения БД для обеспечения возможности осуществления транзакций (операции внесения, удаления, редактирования данных) необходимо определение структуры файла в рамках созданной БД.

В системе ADABAS определение файла подразумевает создание таблицы определения данных (Field Definition Table — FDT) с последующим определением свойств файла (локализация дискового пространства файла и пр.). После создания FDT и определения параметров файл создается в рамках выбранной БД и отображается в списке файлов БД.

Под определением записи понимается последовательное определение каждого из полей (атрибутов) посредством редактора FDT при определении файла. Каждое из полей (атрибут) определяется рядом обязательных параметров, в то время как часть параметров назначается по выбору.

Под определением элементов файла в рамках создания БД подразумевается определение экстентов таких параметров, как хранилище данных (Data Storage или DATA), адресный конвертор (Address Converter или AC), стандартный индекс (Normal Index или NI), верхний индекс (Upper Index или UI).

Вопросы для самоконтроля

1. Опишите модели данных СУБД ADABAS.
2. Раскройте понятие применительно к СУБД ADABAS: база данных.
3. Раскройте понятие применительно к СУБД ADABAS: файл.
4. Раскройте понятие применительно к СУБД ADABAS: ассоциация.
5. Раскройте понятие применительно к СУБД ADABAS: атрибут.
6. Раскройте понятие применительно к СУБД ADABAS: множественный атрибут.
7. Раскройте понятие применительно к СУБД ADABAS: поисковый атрибут.

8. Раскройте понятие применительно к СУБД ADABAS: простая, составная и периодическая группа.
9. Какие типы связей существуют между записями файлов БД ADABAS?
10. Перечень и описание основных операций базовой модели данных СУБД ADABAS.
11. Понятие и сущность ассоциативного поиска записей файлов БД ADABAS.
12. Понятие и сущность ассоциативного поиска записей в связанных файлах и рандомизированного доступа к записям файлов БД ADABAS.
13. Архитектура СУБД ADABAS как совокупность программных компонентов.
14. Многоуровневая архитектура СУБД ADABAS: внешний, концептуальный и внутренний уровни.
15. Среда хранения данных СУБД ADABAS и структурные элементы БД ADABAS.
16. Структура элементов БД ADABAS: накопитель, ассоциатор, рабочий набор и вспомогательные наборы данных.

Глава 4

Средства автоматизации проектирования БД ADABAS

4.1. Редактирование и сохранение таблицы определения данных

Как уже оговаривалось, для создания структур данных ADABAS (обеспечивающих хранение пользовательских данных и дающих проектировщику возможность управления данными) необходимо определить БД (в которой располагаются файлы с индивидуальной структурой хранения), в рамках созданной БД — определить файл (в котором задается структура хранения данных), в рамках файла — определить записи и параметры ассоциатора.

Уже отмечалась важность FDT редактора при определении записи в рамках файла. В данном разделе рассматриваются дополнительные возможности FDT редактора, которые можно использовать при проектировании информационных систем.

Редактирование таблицы определения данных (таблицы FDT)

Таблица определения данных создается посредством последовательного определения полей в рамках определения файла БД. В процессе эксплуатации созданной БД может возникнуть необходимость несколько изменить параметры каких-либо из полей в структуре хранения файла. Для внесения изменений в структуру необходимо придерживаться определенной последовательности.

1. Следует убедиться, что БД, в рамках которой находится файл, структуру которого необходимо изменить, находится в активном режиме (*active*). Если же БД находится в неактивном режиме (в главном контекстном окне БД отображается со статусом *inactive*), следует выбрать в главном контекстном окне (DBA Workbench) необходимую БД и воспользоваться альтернативой контекстного меню Database → Start (рис. 34).

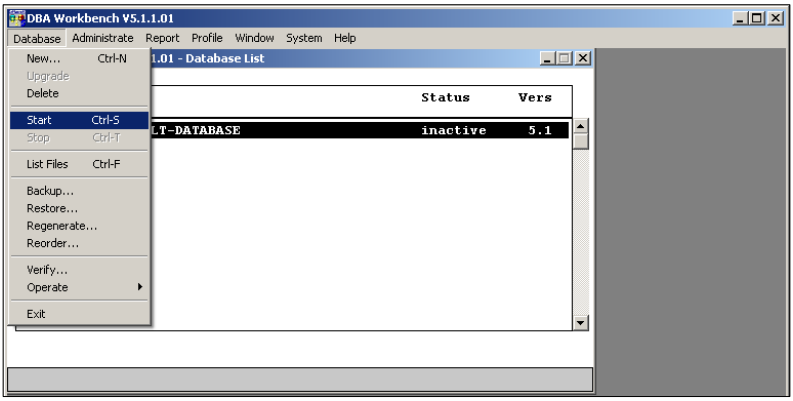

Рис. 34. Запуск БД (перевод в активное состояние)

При этом проектировщик будет уведомлен о включении активного режима БД следующим сообщением (рис. 35).

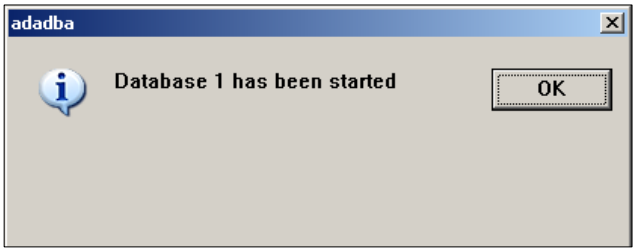

Рис. 35. Информационное сообщение
о включении активного режима БД

После этого выбранная БД будет отображаться со статусом *active* (рис. 36).

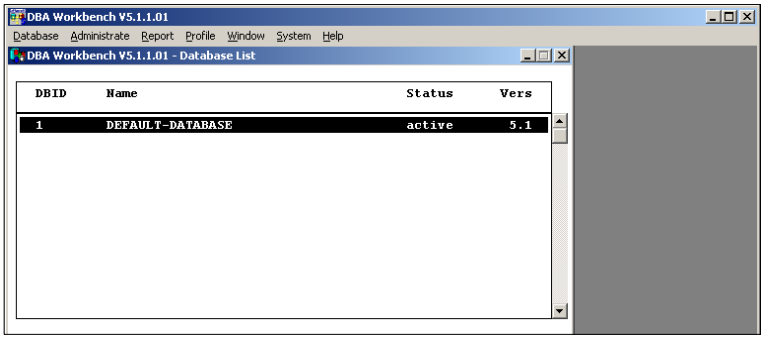

Рис. 36. Вид управляющего окна при включении активного режима БД

2. Из списка файлов соответствующей БД следует выбрать файл, в структуру которого должны быть внесены изменения (в данном примере — employee), и в контекстном меню выбрать альтернативу Administrate → Modify FDT… (рис. 37).

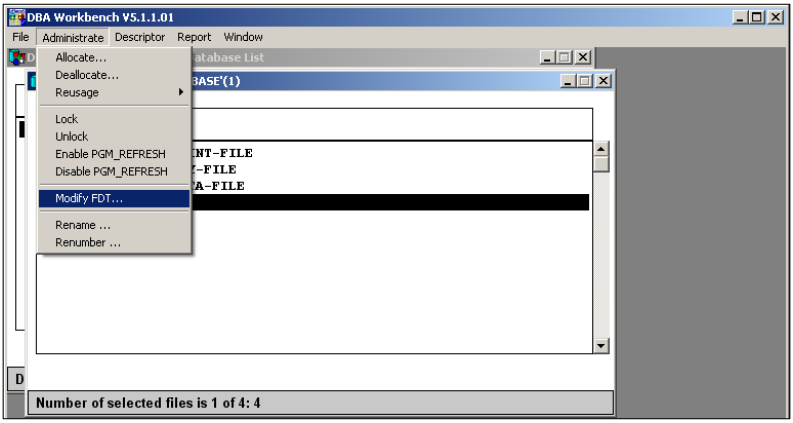

Рис. 37. Изменение структуры файла БД

3. В следующем окне будет отображена структура FDT выбранного файла. Проектировщик может изменить длины уже заданных полей (остальные их параметры недоступны для редактирования), а также определить параметры новых полей необходимых для дальнейшего проектирования информационной системы при помощи средств редактора FDT.

При увеличении длин уже определенных полей следует учитывать, что предел расширения записей определен параметром *Padding*. При необходимости следует расширить область данного параметра.

4. После завершения ввода параметров новых полей и изменения длин уже определенных полей необходимо воспользоваться системной кнопкой «OK».

Сохранение таблицы определения данных (таблицы FDT)

Разработчиками ADABAS и авторами данного пособия настоятельно рекомендуется при процедуре определения файла БД и определении параметров полей, перед тем как перейти к процедуре сохранения файла, сохранить полученную таблицу определения данных FDT (для последующего использования и восстановления файлов). Эти рекомендации также относятся к процедуре редактирования таблицы определения данных. Для этого разработана последовательность операций.

1. После окончательного определения (редактирования) полей таблицы в окне FDT редактора следует воспользоваться альтернативой FDT → Save FDT As… либо альтернативой FDT → Save FDT (рис. 38).

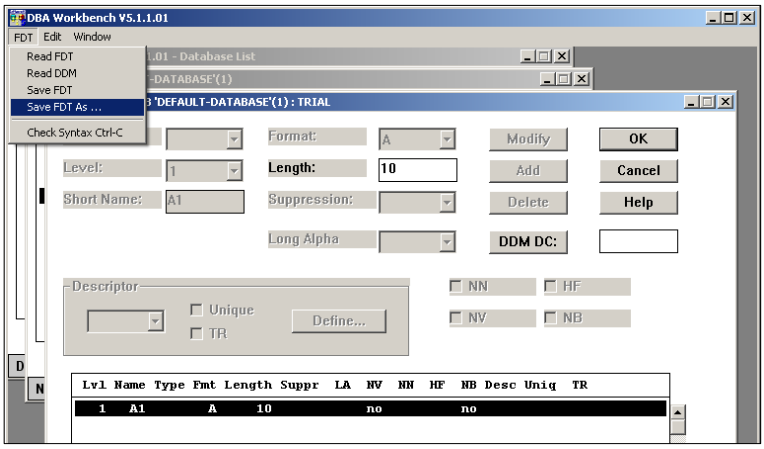

Рис. 38. Сохранение таблицы FDT в файл

Разница в использовании альтернатив Save FDT As... и Save FDT заключается в том, что при редактировании таблицы определения данных альтернатива Save FDT позволит вносить произведенные изменения в уже существующий файл FDT (который является основой для созданного файла). В свою очередь, альтернатива Save FDT As... предлагает проектировщику указать имя иного файла для сохранения произведенных изменений. При осуществлении процедуры определения файла функции данных альтернатив идентичны.

2. В появившемся окне (рис. 39) следует указать путь к сохраняемому файлу, указать имя сохраняемого файла FDT (например, EMPLOYEE1.FDT) и выбрать системную кнопку «Сохранить».

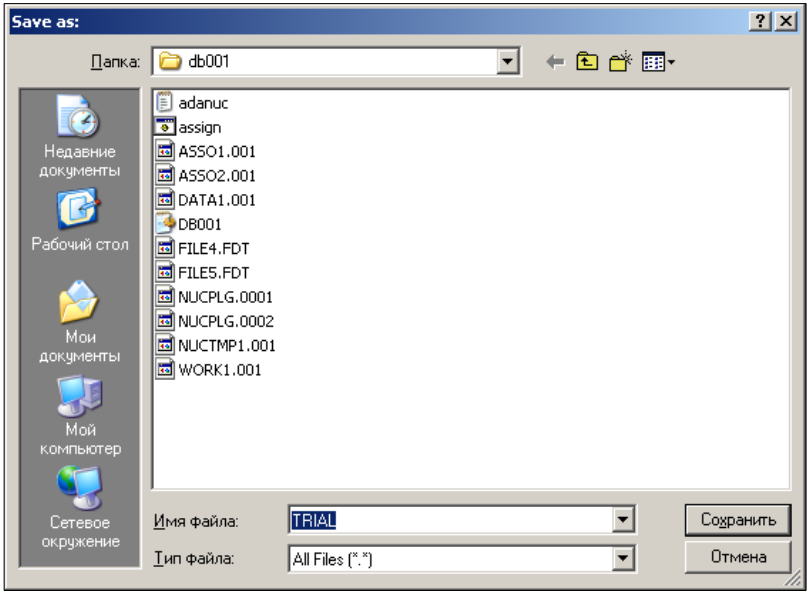

Рис. 39. Определение локализации сохраняемого файла

Далее, если указанное имя для сохранения файла FDT было корректным, проектировщик БД будет проинформирован сообщением (рис. 40), после чего управление будет возвращено к окну редактора FDT.

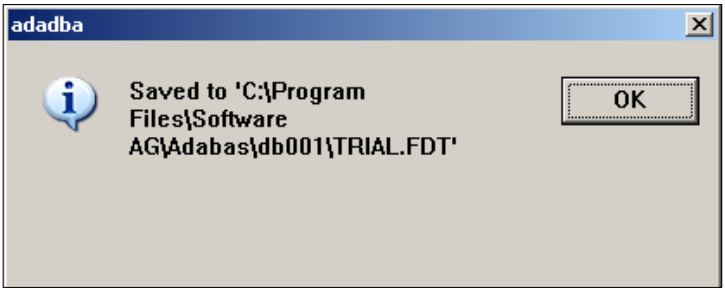

Рис. 40. Информационное сообщение об успешном сохранении таблицы FDT

4.2. Способы создания и восстановления таблицы определения данных

В целях подстраховки от случайного удаления файлов, определенных в рамках БД, следует предусмотреть процедуру восстановления таблиц определения данных, которая является основой для определения файлов.

Восстановление таблицы определения данных (таблицы FDT)

Для восстановления таблицы определения данных существует последовательность операций (при условии, что была сохранена копия файла FDT требуемой структуры либо существует DDM, созданный в редакторе Natural, соответствующий файлу БД с соответствующей структурой — файл с расширением *.NSD):

1. Необходимо воспользоваться альтернативой File → New контекстного меню окна списка файлов БД (рис. 41).

2. В появившемся окне редактора FDT следует выбрать альтернативу FDT → Read FDT или FDT → Read DDM (рис. 42).

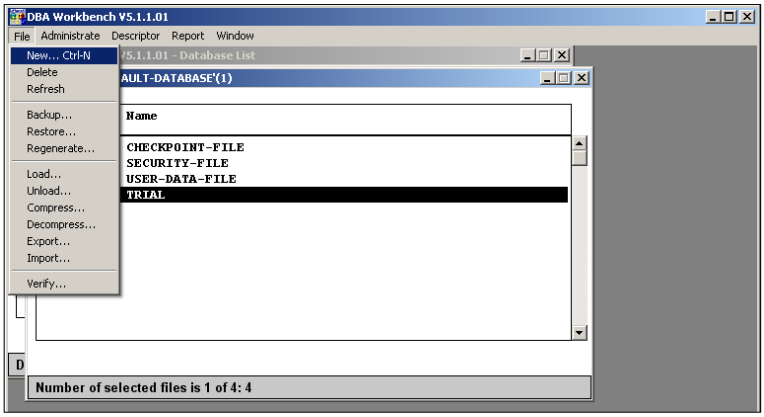

Рис. 41. Создание нового файла (восстановление таблицы FDT)

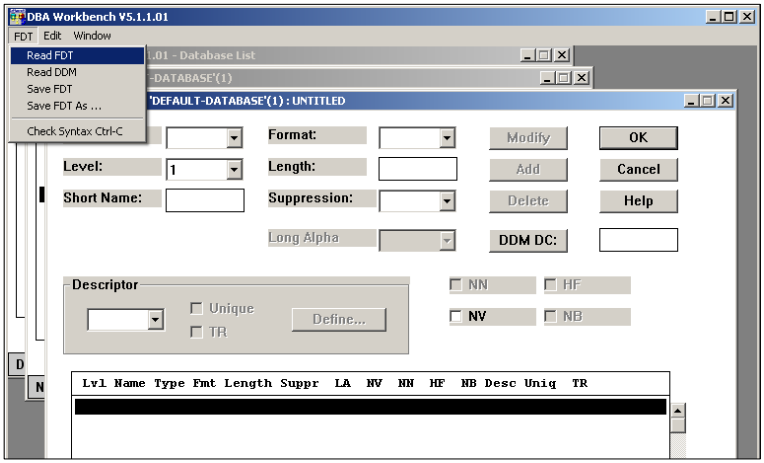

Рис. 42. Чтение таблицы FDT

При восстановлении таблицы FDT при помощи файла DDM необходимо, чтобы этот файл был создан и физически располагался на жестком диске. Следует обратить внимание, что при создании файла DDM (в редакторе Natural) файлы сохраняются в файловой структуре Natural. Таким образом, необходимо либо четко знать расположение требуемого файла DDM и указывать его в последующем окне, либо создать отдельную папку и скопировать в нее нужный файл DDM.

3. В последующем окне (рис. 43) необходимо указать путь и название файла FDT (DDM), на основе которого восстанавливается структура таблицы FDT.

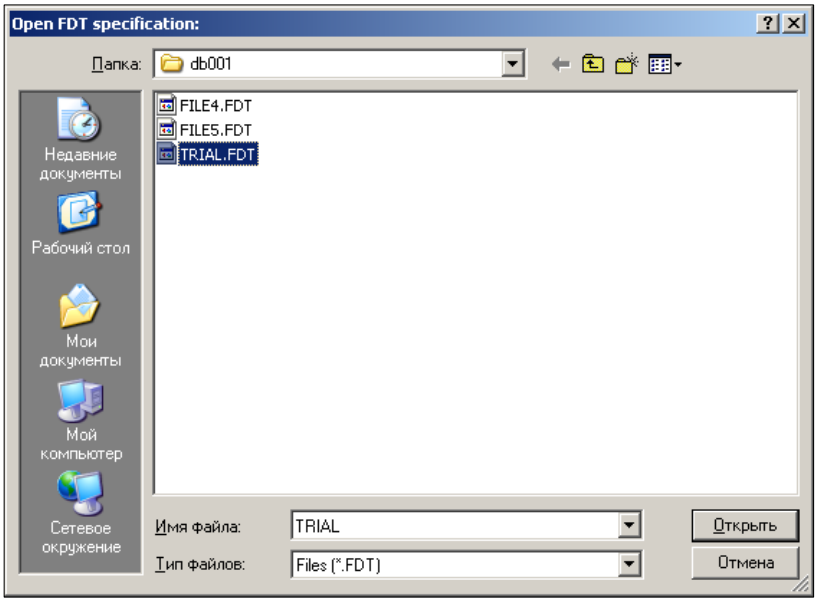

Рис. 43. Определение локализации файла таблицы FDT

4. В окне редактора FDT будет отображена структура FDT (DDM) файла, которую (при необходимости дополнив новыми элементами или удалив лишние элементы) необходимо сохранить.

Дополнительные функции редактора таблиц определения данных (таблиц FDT)

В редакторе FDT предусмотрены дополнительные возможности, которые помогут проектировщику БД сэкономить время на создание и сохранение таблицы FDT.

В случае, когда проектировщик БД определяет серию схожих полей в рамках таблицы FDT, весьма полезными могут

оказаться функции копирования, вырезания и вставки полей FDT редактора. Чтобы воспользоваться данными функциями, необходимо выбрать (установив курсор мыши) уже определенное в рамках таблицы FDT поле (в данном примере — поле с сокращенным именем AX), и воспользоваться альтернативой Edit → Copy контекстного меню (рис. 44).

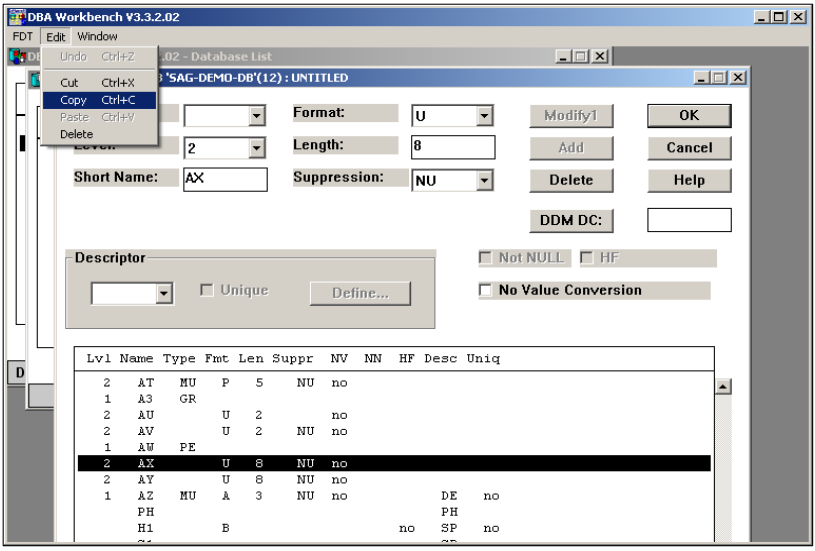

Рис. 44. Функция копирования редактора FDT

Далее необходимо выбрать требуемое для нового элемента место (используя курсор мыши) и вставить скопированное поле, используя функцию вставки Edit → Paste (см. рис. 44).

После проведения данной операции в редактируемой FDT таблице будут отображены два совершенно идентичных поля (в данном примере — AX) (рис. 45).

Далее при необходимости следует изменить параметры скопированного поля (***Level***, ***Short name***, ***Format*** и пр.), воспользовавшись альтернативой Modify.

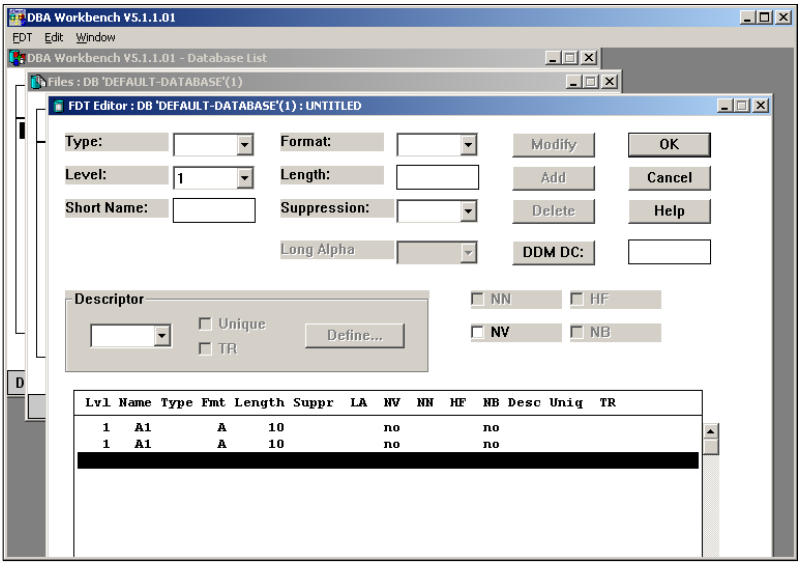

Рис. 45. Функция вставки редактора FDT

При копировании поля в рамках одной таблицы FDT следует учитывать, что одна таблица FDT не может содержать два или несколько полей с одинаковыми именами. Поэтому необходимо изменять параметр *Short name* копируемых полей (в данном примере — AX). В противном случае при переходе к сохранению файла система, проводя проверку синтаксиса, не примет произведенные изменения, о чем проектировщик будет уведомлен сообщением (рис. 46).

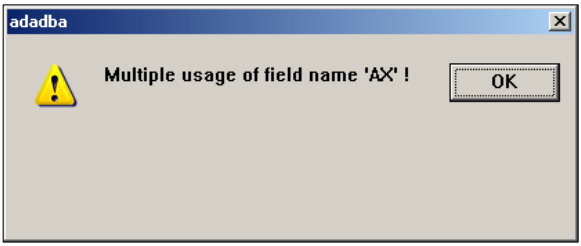

Рис. 46. Информационное сообщение, получаемое при использовании нескольких идентичных имен в одной таблице FDT

В том случае, когда проектировщику БД необходимо переместить какое-либо поле (например, из одной периодической группы в другую), следует воспользоваться функцией вырезания, придерживаясь той же последовательности, что и при осуществлении функции копирования полей (альтернативы Edit → Cut, Edit → Paste контекстного меню) (см. рис. 44).

Функции копирования, вырезания и вставки могут использоваться как при редактировании полей в рамках одной таблицы определения данных (таблицы FDT), так и при перемещении параметров полей между различными таблицами FDT при одновременном их редактировании.

Для корректного редактирования таблиц FDT предусмотрена функция проверки синтаксиса. Автоматическая проверка синтаксиса производится при переходе к сохранению определяемого файла. Однако проектировщик БД может воспользоваться данной функцией в любое время.

Чтобы воспользоваться функцией проверки синтаксиса, следует выбрать альтернативу FDT → Check Syntax контекстного меню редактора FDT (рис. 47).

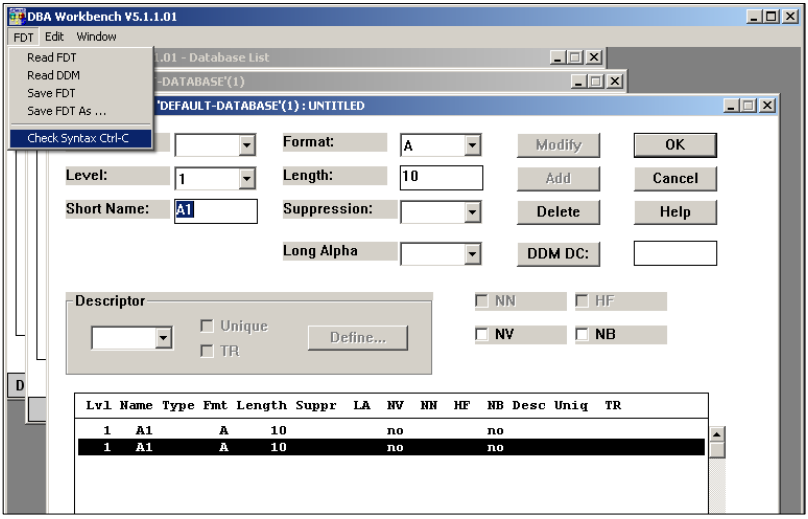

Рис. 47. Функция проверки синтаксиса

Далее проектировщик будет уведомлен последовательными информационными сообщениями обо всех ошибках, допущенных при определении полей. Функция проверки синтаксиса включает проверку периодических групп, проверку структуры уровней полей и т. д.

В случае необходимости проектировщик БД может совмещать алгоритм восстановления таблицы определения данных с функциями копирования, вырезания и вставки редактора FDT. Так, при создании новой таблицы FDT или определении нового файла проектировщик может руководствоваться такой последовательностью операций.

1. Создать новый файл и восстановить требуемую таблицу определения данных (как описано в разделе «Восстановление таблицы определения данных» данного пункта). Можно создать несколько новых файлов и восстановить несколько таблиц FDT; поля, которые будут использоваться для создания новой таблицы, содержатся в нескольких уже созданных таблицах FDT.
2. Скопировать необходимые для новой таблицы FDT поля из восстановленной таблицы (таблиц) FDT.
3. Вставить скопированные поля в новую таблицу FDT и произвести необходимые изменения параметров полей.
4. Проверить синтаксис новой таблицы FDT.
5. Сохранить таблицу определения данных.
6. Сохранить определяемый файл.

Также проектировщик может совмещать воссоздаваемые таблицы FDT с таблицами FDT уже определенных и сохраненных файлов (при использовании их в качестве источников для создания новой таблицы FDT или определении файла в рамках существующей БД).

4.3. Реорганизация структур данных ADABAS

В процессе администрирования информационной системы часто возникает потребность расширить или уменьшить количество файлов БД для обеспечения большего объема дискового пространства для хранения пользовательских данных и т. д. Изменения структуры информационной системы, а следовательно,

и структуры данных неизбежны, так как любая организация постоянно изменяется в условиях воздействия на нее внешней среды. В организации появляются новые отделы, новые сотрудники, а также происходят реструктуризации ее подразделений. Это влечет необходимость изменения как структуры информационной системы организации, так и средств управления информационной системой.

DBA Workbench предлагает достаточно широкий спектр инструментов, позволяющих изменить созданную структуру данных, не нарушая при этом целостность баз данных и обеспечивая сохранение в новых структурах уже введенных в базы пользовательских данных.

Реорганизация БД подразумевает изменение структуры размещения всей БД, а также смещает глобальные области БД и устраняет фрагментацию экстентов ***адресного конвертора***, ***хранилища данных*** и ***индекса файлов*** БД посредством физического изменения их местонахождения.

Для того чтобы воспользоваться функцией реорганизации БД, следует убедиться, что реорганизуемая БД находится в неактивном режиме (*inactive*). Если же БД находится в активном режиме (отображается со статусом *active*), следует в главном контекстном окне (DBA Workbench) выбрать реорганизуемую БД и воспользоваться альтернативой Database → Stop (рис. 48).

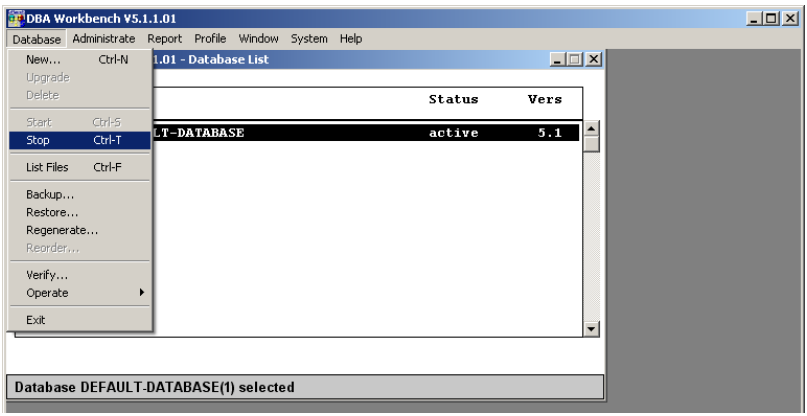

Рис. 48. Остановка БД (перевод в неактивный режим)

Далее в появившемся окне следует указать режим остановки БД (рис. 49).

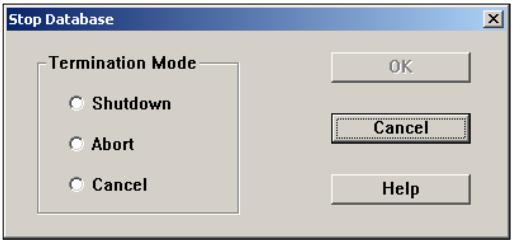

Рис. 49. Режимы остановки БД

Нормальное завершение работы БД (Shutdown)

Данная опция логически останавливает сессию БД. Это значит, что во время остановки транзакции новых пользователей игнорируются, а обновления текущих пользователей (осуществлявших транзакции обновления в момент остановки БД) продолжают выполняться до завершения текущих транзакций каждого из пользователей или до истечения времени, определенного в параметре TT (*nucleus parameter*). После того как активность всех обновлений БД завершается, осуществляется остановка сессии БД.

Аварийное завершение работы БД (Abort)

Данная опция завершает сессию БД немедленно. Все командные процессы, включая текущие обновления БД, мгновенно завершаются. Работа БД останавливается аварийно.

Функция реорганизации не может быть осуществлена с БД, работа которой была завершена аварийно (с опцией *Abort*).

Быстрое завершение работы БД (Cancel)

Данная опция мгновенно завершает сессию БД. При этом завершаются транзакции всех пользователей, выполняющих запросы к данным БД, и работа БД останавливается.

Выбрав необходимый способ остановки БД, следует воспользоваться системной кнопкой «OK».

После осуществления вышеописанной последовательности операций пользователь будет проинформирован об остановке

БД информационным сообщением (рис. 50), и реорганизуемая БД будет отображаться в списке баз данных со статусом неактивного режима (*inactive*) (рис. 51).

Рис. 50. Информационное сообщение
об успешном переводе БД в неактивный режим

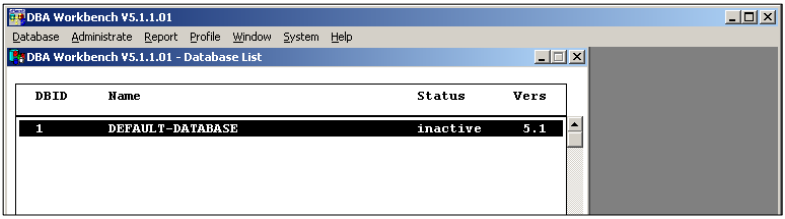

Рис. 51. Вид управляющего окна
после перевода БД в неактивный режим

Для реорганизации БД необходимо воспользоваться альтернативой Database → Reorder… контекстного меню (рис. 52).

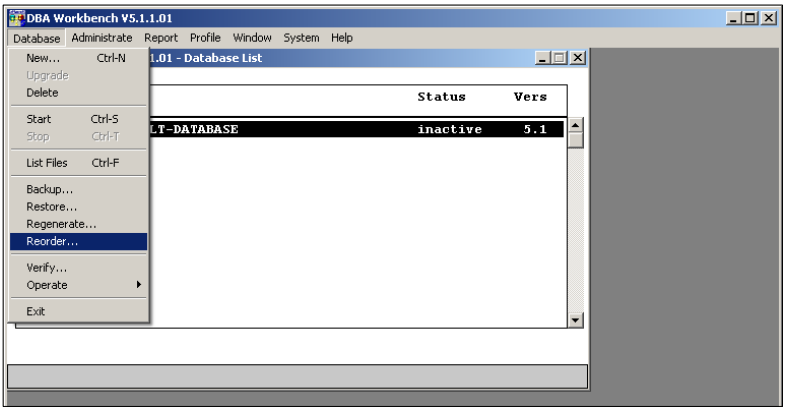

Рис. 52. Реорганизация БД

Далее в появившемся окне (рис. 53) следует указать параметры реорганизации БД.

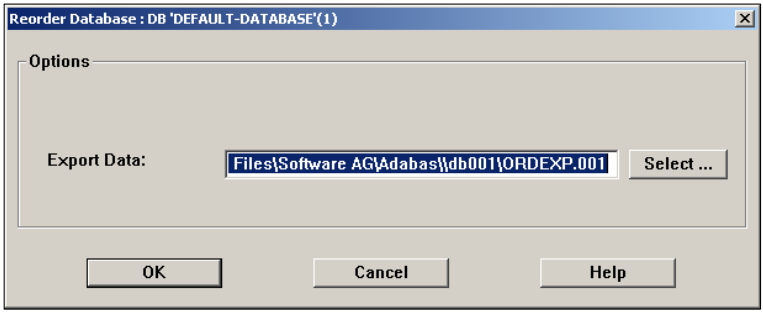

Рис. 53. Параметры реорганизации

Параметры реорганизации БД следующие.

Путь к файлу экспорта данных (Export Data)

В данном текстовом поле отображается путь к временному файлу данных (ORDEXP). Можно оставить указанный в данном поле файл в качестве временного файла данных либо указать (посредством системной кнопки ‘Select’) путь к иному файлу.

После редактирования параметров реорганизации БД необходимо выбрать кнопку «OK».

Далее проектировщик будет уведомлен сообщением (рис. 54), в котором системой предлагается удалить временный файл данных (ORDEXP).

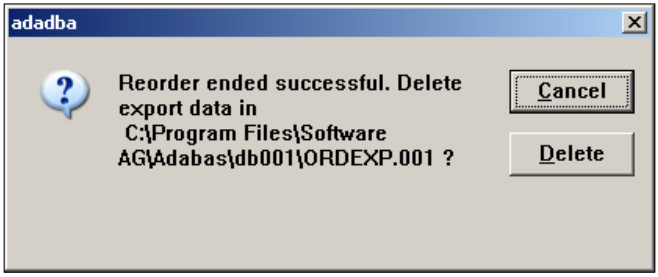

Рис. 54. Информационное сообщение (удаление временного файла данных)

Для удаления временного файла данных следует выбрать альтернативу Delete.

Обо всех операциях, проведенных системой при реорганизации БД, проектировщик будет проинформирован последующим текстовым сообщением (рис. 55).

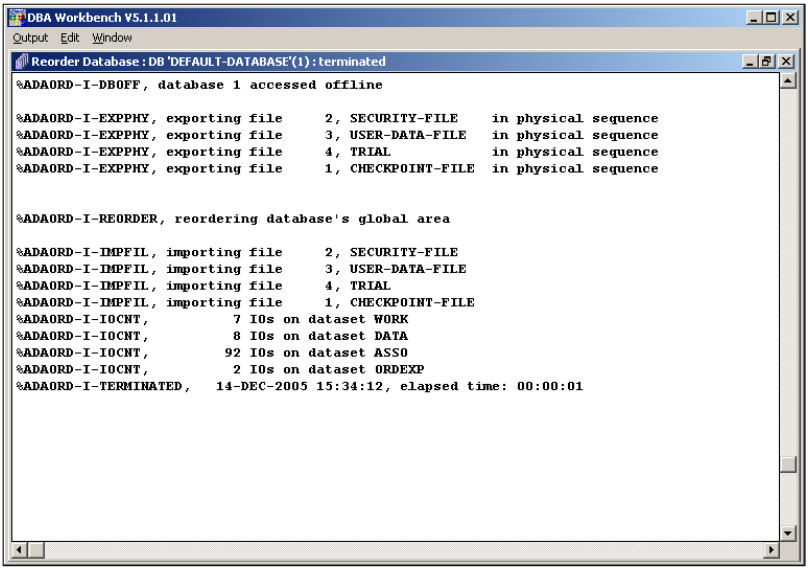

Рис. 55. Отчет о проведенных операциях при реорганизации

Процесс реорганизации БД происходит в определенной последовательности.

Шаг 1. Все уже определенные в рамках БД файлы экспортируются в указанный в параметре ‘Export Data’ файл.

Шаг 2. Происходит смещение контрольных блоков БД и области таблицы определения файлов БД с учетом нового максимального количества файлов в БД.

Шаг 3. Файлы импортируются обратно в БД. При этом каждый из файлов перемещен, множественные логические экстенты сжаты в единый логический экстент и факторы дополнительного пространства (Padding) упразднены.

4.4. Изменение размера физических структур БД

В случае когда размер структур данных (ASSO, DATA и пр.) становится недостаточным для дальнейшего хранения пользовательских данных (о чем администратор информационной системы будет уведомлен системными информационными сообщениями), одним из выходов может стать создание дополнительных контейнеров данных. Контейнеры данных могут быть также удалены (если размер структур данных, предусмотренный проектировщиком БД, значительно превышает объем хранимых пользовательских данных и есть необходимость использовать зарезервированный объем дискового пространства для создания иных БД).

В рамках системы ADABAS функция управления контейнерами служит для создания или удаления контейнеров в существующих наборах ***ассоциатора (ASSO)*** или ***хранилища данных (Data Storage)***, а также для создания контейнеров SORT, TEMP и WORK.

Для создания дополнительного контейнера ассоциатора (ASSO) или хранилища данных (Data Storage) следует выбрать в главном контекстном окне (DBA Workbench) БД, к составляющим которой необходимо добавить контейнер, и воспользоваться альтернативой контекстного меню Administrate → Containers → Add… (рис. 56).

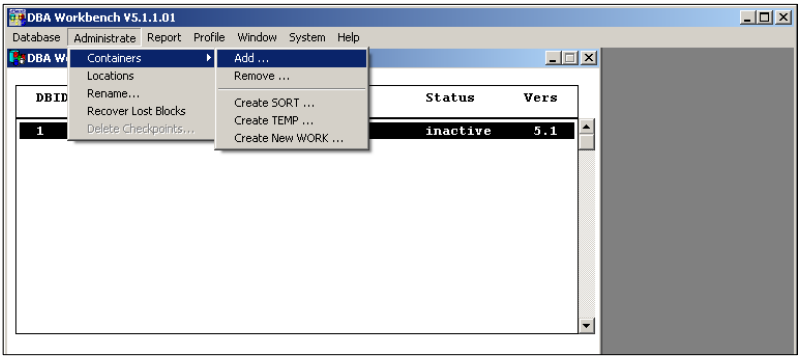

Рис. 56. Создание дополнительного контейнера данных

В последующем окне (рис. 57) необходимо указать параметры для создания контейнера.

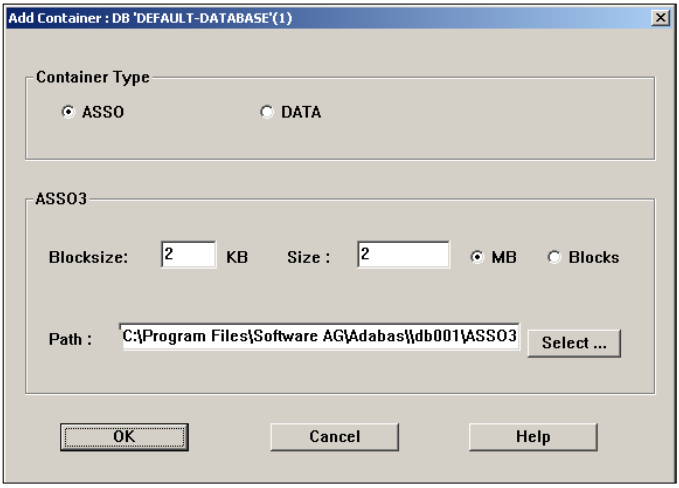

Рис. 57. Параметры для создания контейнера

Тип контейнера (Container Type)

Посредством задания значения данного параметра проектировщик определяет тип контейнера, который необходимо добавить в существующую структуру данных БД. Предлагается выбрать один из следующих типов:

ASSO — контейнер ассоциатора;

DATA — контейнер хранилища данных.

Параметры дополнительного контейнера

Все параметры дополнительного контейнера сгруппированы блоком ввода, которому присваивается имя, аналогичное имени дополнительного контейнера (на рис. 57 — ASSO3). Дополнительному контейнеру присваивается имя соответствующего типа контейнера (ASSO, DATA) с номером, следующим по порядку за предыдущим номером контейнера. При определении БД создаются контейнеры со следующими именами: ASSO1, ASSO2, DATA1, WORK. Таким образом, при создании дополнительных контейнеров автоматически будут отображаться следующие имена файлов контейнера: ASSO3 — при создании допол-

нительного контейнера типа ASSO; DATA2 — при создании дополнительного контейнера типа DATA.

В рамках данной группы рассматриваются следующие параметры.

Размер блока (Blocksize)

В текстовом поле ‘Blocksize’ отображается размер блока добавляемого типа контейнера. По умолчанию системой предлагается использовать 2 Кбайта для контейнеров ASSO и 4 Кбайта — для контейнера DATA. В случае необходимости следует сменить значения.

Размер контейнера (Size)

Размер дискового пространства, выделяемый для использования в рамках дополнительного контейнера, указывается в текстовом окне ‘Size’. Для смены значения, предлагаемого по умолчанию, следует ввести в данном окне соответствующее значение. Размер контейнера можно задать как в блоках (опция Blocks), так и в мегабайтах (опция MB).

Область нахождения файла контейнера (Path)

В текстовом окне ‘Path’ отображается полный путь к файлу дополнительного контейнера. При необходимости нужно указать желаемый путь к файлу, воспользовавшись окном обзора, вызываемым кнопкой ‘Select…’.

Создание дополнительного контейнера WORK возможно только при неактивном ядре системы (статус БД — ‘inactive’). Также следует учитывать, что максимальное количество контейнеров типа WORK в системе — 2 (соответственно, имя последнего — WORK2).

После определения параметров дополнительного контейнера необходимо воспользоваться системной кнопкой «OK».

Дополнительный контейнер будет создан системой, о чем проектировщик БД будет проинформирован сообщением (рис. 58).

Для удаления контейнера из структуры данных необходимо выбрать в главном контекстном окне (DBA Workbench) БД, из которой необходимо удалить контейнер, и воспользоваться альтернативой контекстного меню Administrate → Containers → Remove … (см. рис. 56).

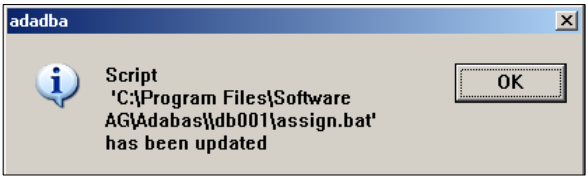

Рис. 58. Информационное сообщение
об успешном создании дополнительного контейнера

В последующем окне (рис. 59) вводится тип контейнера, который необходимо удалить. Системой удаляется последний контейнер указанного типа в существующем наборе ассоциатора или наборе данных.

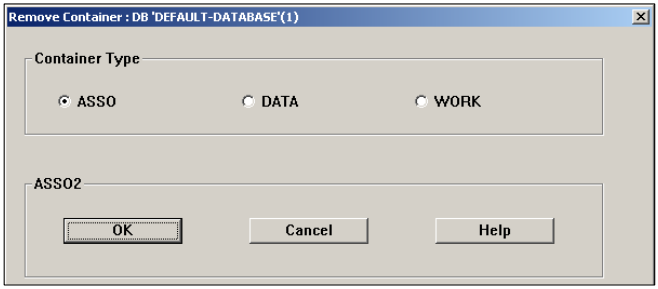

Рис. 59. Удаление контейнера

Функция удаления контейнеров доступна для проектировщика только при неактивном ядре системы (статус БД — 'inactive'). Могут быть удалены только те контейнеры, которые были определены проектировщиком дополнительно. Контейнеры, определяемые в процессе определения БД, не могут быть удалены (ASSO1, DATA1, WORK1).

Для удаления контейнера следует воспользоваться кнопкой «OK».

Для создания дополнительного контейнера типа SORT следует выбрать в главном контекстном окне (DBA Workbench) БД, к составляющим которой необходимо добавить контейнер, и воспользоваться альтернативой контекстного меню Administrate → Containers → Create SORT… (см. рис. 56).

Для успешного создания контейнера SORT нужно в последующем окне (рис. 60) ввести соответствующие параметры.

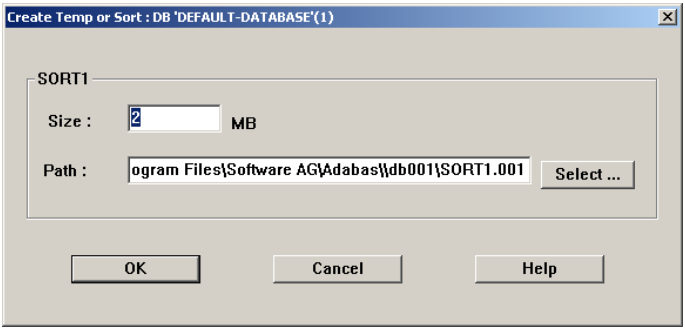

Рис. 60. Создание контейнера SORT

Размер контейнера (Size)

Для смены значения, предлагаемого по умолчанию, следует ввести в данном окне соответствующее значение. Размер контейнера задается в мегабайтах.

Область нахождения файла контейнера (Path)

В текстовом окне ‘Path’ отображается полный путь к файлу дополнительного контейнера. При необходимости нужно указать желаемый путь к файлу, воспользовавшись окном обзора, вызываемым кнопкой ‘Select…’

После ввода всех параметров следует воспользоваться кнопкой «OK». При этом проектировщик будет проинформирован о создании дополнительного контейнера сообщением (рис. 61).

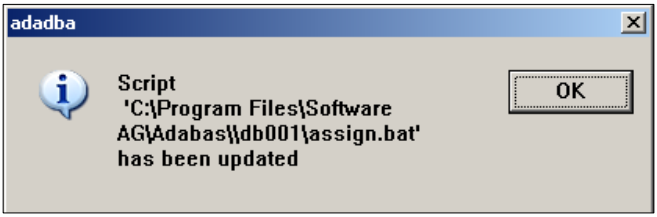

Рис. 61. Информационное сообщение
об успешном создании контейнера

Для создания дополнительного контейнера типа TEMP следует выбрать в главном контекстном окне (DBA Workbench) БД, к составляющим которой необходимо добавить контейнер, и воспользоваться альтернативой контекстного меню Administrate → Containers → Create TEMP… (см. рис. 56).

Для успешного создания контейнера TEMP нужно в последующем окне (рис. 62) ввести соответствующие параметры.

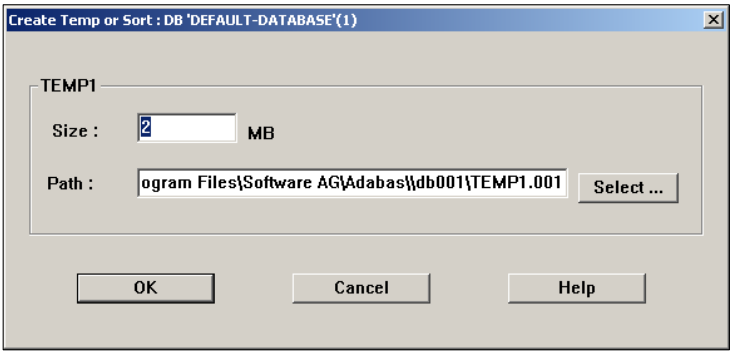

Рис. 62. Создание контейнера TEMP

Размер контейнера (Size)

Для смены значения, предлагаемого по умолчанию, следует ввести в данном окне соответствующее значение. Размер контейнера задается в мегабайтах.

Область нахождения файла контейнера (Path)

В текстовом окне ‘Path’ отображается полный путь к файлу дополнительного контейнера. При необходимости нужно указать желаемый путь к файлу, воспользовавшись окном обзора, вызываемым кнопкой ‘Select…’

После ввода всех параметров следует воспользоваться кнопкой «OK». При этом проектировщик будет проинформирован о создании дополнительного контейнера.

Для создания нового контейнера типа WORK следует выбрать в главном контекстном окне (DBA Workbench) БД, к составляющим которой необходимо добавить контейнер, и воспользоваться альтернативой контекстного меню Administrate → Containers → Create New WORK… (см. рис. 56).

Для успешного создания нового контейнера WORK нужно в последующем окне (рис. 63) ввести соответствующие параметры.

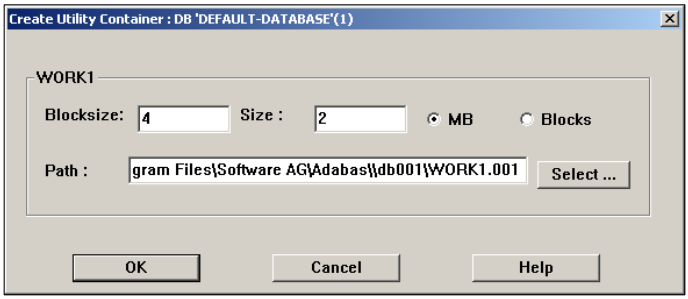

Рис. 63. Создание контейнера WORK

Размер блока (Blocksize)

В текстовом поле ‘Blocksize’ отображается размер блока контейнера. По умолчанию системой предлагается использовать 4 Кбайта для новых контейнеров WORK. В случае необходимости следует сменить значения.

Размер контейнера (Size)

Размер дискового пространства, выделяемый для использования в рамках нового контейнера WORK, указывается в текстовом окне ‘Size’. Для смены значения, предлагаемого по умолчанию, следует ввести в данном окне соответствующее значение. Размер контейнера можно задать как в блоках (опция Blocks), так и в мегабайтах (опция MB).

Область нахождения файла контейнера (Path)

В текстовом окне ‘Path’ отображается полный путь к файлу нового контейнера WORK. При необходимости нужно указать желаемый путь к файлу, воспользовавшись окном обзора, вызываемым кнопкой ‘Select…’.

Имя файла нового контейнера WORK (отображаемое в строке ‘Path’) не должно иметь имя, идентичное старому контейнеру WORK. В противном случае система не сможет создать новый контейнер, о чем проектировщик будет проинформирован сообщением (рис. 64).

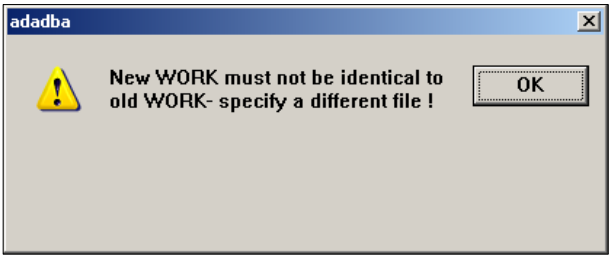

Рис. 64. Информационное сообщение,
получаемое при двойном использовании имени контейнера WORK

После ввода всех параметров следует воспользоваться кнопкой «OK». При этом проектировщик будет проинформирован о создании нового контейнера WORK.

Для изменения модификации создаваемых дополнительных контейнеров можно воспользоваться функцией контроля контейнеров. Данная функция служит для изменения местонахождения файлов контейнеров.

Чтобы воспользоваться данной функцией, необходимо выбрать в главном контекстном окне (DBA Workbench) БД, контейнеры которой нужно модифицировать, и воспользоваться альтернативой контекстного меню Profile → Containers… (рис. 65). При использовании данной функции БД может находиться как в активном, так и в неактивном режиме.

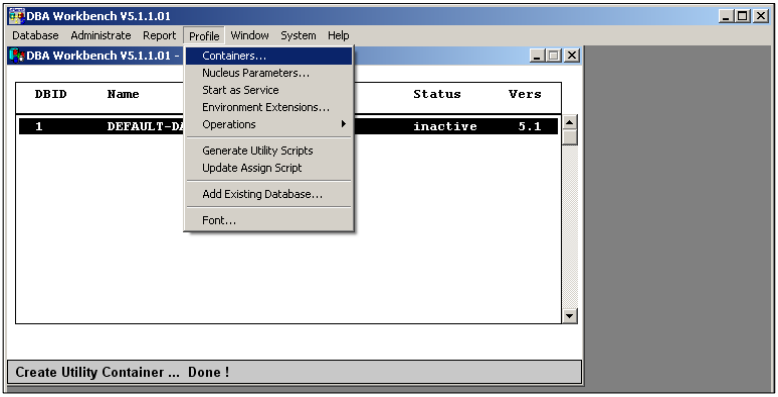

Рис. 65. Изменение модификации дополнительных контейнеров

В последующем окне (рис. 66) отображаются все действия, произведенные с контейнерами (добавление, удаление).

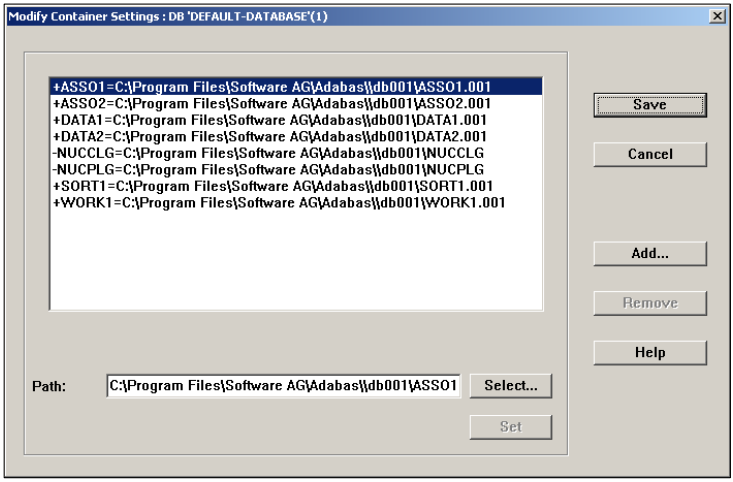

Рис. 66. Отчет об изменении модификации контейнеров

При помощи изменения данных в появившемся окне проектировщик может изменять местонахождение файлов контейнеров, изменять их и добавлять новые. Это может потребоваться, если контейнеры не были созданы посредством главного окна управления DBA Workbench или когда необходимо указать файлы контейнеров SORT и WORK, принадлежащие иной БД, в качестве используемых в рамках данной БД под заданными именами.

Область нахождения файла контейнера (Path)

Для модификации области местонахождения файлов контейнеров следует воспользоваться текстовым окном ‘Path’. Может быть использована опция ‘Select…’. Необходимо указать полный путь к файлам контейнера, созданным в рамках структуры данных этой БД либо иной БД.

Добавление нового контейнера (Add)

Для создания нового контейнера в рамках структуры данных БД можно использовать опцию ‘Add’ (см. рис. 66). При выборе данной опции появляется следующее управляющее окно (рис. 67).

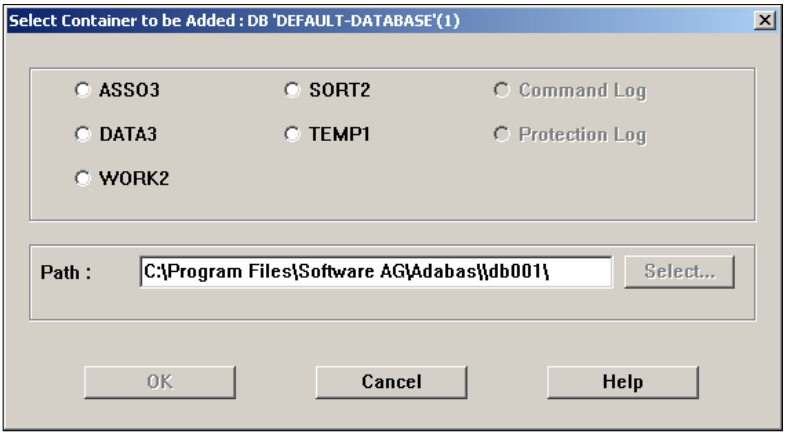

Рис. 67. Добавление нового контейнера

Посредством управляющего окна предлагается выбрать имя соответствующего типа контейнера и область местонахождения. В качестве активных для проектировщика опций отображаются только «свободные» контейнеры (следующие по порядку). Например, если файлы контейнеров ASSO1 и ASSO2 существуют, то в качестве следующего контейнера, который может быть добавлен к структуре, будет отображаться ASSO3.

Файл дополнительного контейнера должен быть создан до того, как он сможет быть добавлен в структуру данных посредством использования функции модификации.

Удаление контейнера (Remove)

Для удаления контейнеров необходимо выбрать запись, соответствующую контейнеру, который должен быть удален, и воспользоваться кнопкой ‘Remove’ (см. рис. 66).

После модифицирования всех параметров контейнеров для сохранения изменений необходимо воспользоваться кнопкой ‘Save’ (см. рис. 66). Проектировщик будет уведомлен сообщением о проведении модификаций структуры данных.

4.5. Способы моделирования структур организации посредством структуры таблицы определения данных

При создании автоматизированной системы управления организацией необходимо учитывать имеющиеся связи между структурными подразделениями, иерархию полномочий и их делегирование, а также значимость этих связей при осуществлении различных процессов деятельности.

В рамках теории управления рассматривается несколько возможных форм представления структуры организации. Все они, как правило, сводятся к графическим отображениям, когда при помощи символов, обозначающих различные подразделения, а также схематических связей, обозначающих отношения между подразделениями (количественных и качественных), строится адекватная модель структуры организации. Любая организация может быть представлена с разной степенью подробности (вплоть до отображения каждой из должностей).

Графически отображенная модель необходима при принятии стратегических и оперативных решений по реструктуризации организации с целью более эффективного управления, по созданию дополнительных подразделений, по реорганизации отношений (например, финансовых) при желаемом увеличении прибыли и т. д.

При построении интегрированной системы управления организацией создание модели структуры также необходимо. Способы задания модели структуры могут быть различными.

В среде ADABAS и Natural все связи между переменными могут быть представлены в следующем виде.

1. Создание структуры при помощи задания свойств полей в ядре БД, фиксированные связи между которыми могут отражаться в виде так называемых «периодических групп». Вся последующая обработка данных, а также интерфейс предоставления конечных результатов (результаты анализа, учета и пр.) осуществляется при помощи средств Natural. Сама структура в понятной форме будет отражать действительную структуру

организации. Ядро БД, таким образом, будет являться моделью структуры.

2. Создание структуры путем задания параметров переменных в ядре ADABAS без задания структуры с помощью «периодических групп». Структура при этом будет вырисовываться только при обработке данных средствами Natural и при последующем графическом отображении (также с помощью средств Natural). В ядре БД структура прослеживаться не будет.

В зависимости от уровня гибкости структуры организации возможны следующие варианты.

Жестко фиксированная модель структуры

При построении средствами ADABAS и Natural жесткой модели структуры организации посредством фиксирования свойств и признаков должностей и подразделений на всех уровнях иерархия подчиненностей отображается в виде последовательности групп данных («периодических групп»). Иными словами, подчиненности моделируются при помощи включения групп в состав вышестоящей группы при создании ядра БД (таблица FDT). Модель структуры в данном случае можно представить в следующем виде:

Ядро БД (таблица FDT):
− group 1 (организация):
  − group 2_a (отдел 1):
    − field 1_{2a} (свойство 1 отдела 1)
    − field 2_{2a} (свойство 2 отдела 1)
    − …
    − field n_{2a} (свойство n отдела 1)
    − group 3_a (отдел 1.1 подчинен отделу 1):
      − field 1_{3a} (свойство 1 отдела 1.1)
      − field 2_{3a} (свойство 2 отдела 1.1)
      − …
      − field m_{3a} (свойство m отдела 1.1)
      − group 4_a (отдел 1.1.1 подчинен отделу 1.1)
      − group 4_b (отдел 1.1.2 подчинен отделу 1.1)
      − …
      − group 4_k (отдел 1.1.k подчинен отделу 1.1)
    − group 3_b (отдел 1.2 подчинен отделу 1)
    − …

− group 3_p (отдел 1.p подчинен отделу 1)
− group 2_b (отдел 2)
− …
− group 2_h (отдел h)

Данный способ задания модели структуры может быть успешно применим, если структура организации является регламентированной и неизменяемой, с четко определенным количеством должностей на всех уровнях во всех подразделениях. При возникновении необходимости изменить структуру организации неизбежны коренные изменения структуры БД и реорганизация данных, что при больших размерах БД может оказаться трудоемким процессом.

Гибкая модель структуры

Очень часто в процессе деятельности руководители поддерживают гибкую структуру управляемой организации. Применение такой структуры позволяет быстро реагировать на изменения во внешней и внутренней среде, а также в короткие сроки осуществлять инновационные, производственные и прочие проекты.

При создании гибкой модели структуры подразумевается, что свойства и признаки подразделений и сотрудников жестко не фиксируются при помощи «периодических групп»; иерархия подчиненностей никак не обозначается в структуре ядра БД (таблица FDT). Подразумевается задание ряда признаков для каждого из сотрудников; сама структура организации представляется и в последующем отображается при помощи обработки и группировки данных по указанным признакам с помощью средств Natural. Ряд используемых признаков для каждого из сотрудников организации в данном случае должен быть идентичным. Для каждого сотрудника заполняется определенная форма, включающая следующие данные:

− Фамилия, имя, отчество
− Стаж
− Образование
− Домашний адрес
− …

} Личные данные сотрудника

− Структурное подразделение
− Регламентированные обязанности
− Размер заработной платы
− Индивидуальные задания
− …
−
− Прочие данные

} Организационные данные

В этом случае при изменениях в структуре организации изменения в ядре БД не требуются. Изменяются лишь значения определенных признаков сотрудников организации. Структура ядра меняется только при добавлении новых, необходимых для определенного вида анализа или представления структуры, параметров.

Все данные обо всех сотрудниках записываются в единый файл БД, а данные об изменениях в структуре организации (реорганизация подразделений, движение сотрудников, изменение спектра и делегирование полномочий и т. д.) могут быть отражены в отдельно созданном файле ADABAS. При такой организации данных динамика изменения структуры является доступной и дискретные состояния структуры организации за любой период могут быть получены для дальнейшего анализа.

Смешанная модель структуры

Данная модель структуры организации представляет собой синтез жестко фиксированной и гибкой моделей. В этом случае структура организации (а также свойства и признаки подразделений организации) отражается при помощи задания «периодических групп» в одном файле БД (таблица FDT), а информация о сотрудниках каждого из отделов — в других файлах. Обработка и предоставление данных осуществляется при помощи средств Natural.

Структуру организации, таким образом, можно представить в следующем виде:

− file 1 (основной, отражает общую структуру организации):

− group 1 (отдел 1):

− field 1_1 (свойство 1 отдела 1)

- field 2_1 (свойство 2 отдела 1)
- …
- field n_1 (свойство n отдела 1)

- group 2 (отдел 2):
  - field 1_2 (свойство 1 отдела 2)
  - field 2_2 (свойство 2 отдела 2)
  - …
  - field p_2 (свойство p отдела 2)
- …
- group m (отдел m):
  - field 1_m (свойство 1 отдела m)
  - field 2_m (свойство 2 отдела m)
  - …
  - field h_m (свойство h отдела m)

- file 2 (отражает структуру отдела 1):
  - field 1_{f2}
  - field 2_{f2}

  } Личные и организационные данные о сотрудниках отдела 1

  - …
  - field k_{f2}
- …
- file t (отражает структуру отдела t):
  - field 1_{ft}
  - field 2_{ft}

  } Личные и организационные данные о сотрудниках отдела t

  - …
  - field k_{ft}

При такой модели структура организации (подразделений) отражается жестко фиксированной, в то время как структура отделов является гибкой. При внесении данных об изменениях структуры подразделений (например, при введении или упразднении должностей) структура БД остается неизменной.

Выводы по главе 4

Таблица определения данных создается посредством последовательного определения полей в рамках определения файла БД. В процессе эксплуатации созданной БД может возникнуть необходимость несколько изменить параметры каких-либо полей в структуре хранения файла.

В целях подстраховки от случайного удаления определенных в рамках БД файлов следует предусмотреть процедуру восстановления таблиц определения данных, которая является основой для определения файлов. В редакторе FDT предусмотрены дополнительные возможности, которые помогут проектировщику БД сэкономить время на создание и сохранение таблицы FDT. Для корректного редактирования таблиц FDT предусмотрена функция проверки синтаксиса. Автоматическая проверка синтаксиса производится при переходе к сохранению определяемого файла.

В процессе администрирования информационной системы часто возникает потребность расширить или уменьшить количество файлов БД с целью обеспечения большего объема дискового пространства для хранения пользовательских данных и т. д. Изменения структуры информационной системы, а следовательно, и структуры данных неизбежны, так как любая организация постоянно изменяется в условиях воздействия на нее внешней среды. В организации появляются новые отделы, новые сотрудники, происходит реструктуризация ее подразделений. Каждое из таких изменений влечет необходимость изменения как структуры информационной системы организации, так и средств управления информационной системой.

DBA Workbench предлагает достаточно широкий спектр инструментов, позволяющих изменять созданную структуру данных, не нарушая при этом целостность баз данных и обеспечивая сохранение в новых структурах уже введенных в базы пользовательских данных.

В среде ADABAS и Natural все связи между переменными могут быть представлены в следующем виде:

– создание структуры при помощи задания свойств полей в ядре БД, фиксированные связи между которыми могут отражаться в виде так называемых «периодических групп»;

– создание структуры путем задания параметров переменных в ядре ADABAS без задания структуры с помощью «периодических групп».

В зависимости от уровня гибкости структуры организации возможны различные способы задания ее модели.

Вопросы для самоконтроля

1. Для чего служит таблица определения данных (таблица FDT)?
2. Каким образом создается таблица определения данных (таблица FDT)?
3. Каким способом осуществляется модификация таблицы определения данных (таблицы FDT)?
4. Как осуществить восстановление таблицы определения данных (таблицы FDT)?
5. Какими способами можно увеличить объем дискового пространства для хранения пользовательских данных в СУБД ADABAS?
6. Что подразумевает реорганизация БД в рамках СУБД ADABAS?
7. Для чего служит файл экспорта данных (Export Data)?
8. Каково предназначение контейнеров SORT, TEMP и WORK?
9. Каким образом можно создать дополнительный контейнер ассоциатора?
10. Каким образом можно создать дополнительный контейнер данных?
11. Сколько дополнительных контейнеров ассоциатора можно создать?
12. Сколько дополнительных контейнеров данных можно создать?
13. Назовите способы моделирования структур организации посредством структуры таблицы определения данных.
14. Каким образом в СУБД ADABAS задается жестко фиксированная модель структуры?
15. Каким образом в СУБД ADABAS задается гибкая модель структуры?
16. Каким образом в СУБД ADABAS задается смешанная модель структуры?

Глава 5

Поиск и извлечение данных в БД ADABAS и редактирование данных средствами Natural

5.1. Внесение записей

При осуществлении транзакций с БД ADABAS с использованием средств программирования Natural 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. 68, 69).

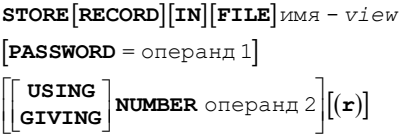

Рис. 68. Синтаксис оператора STORE (структурный режим)

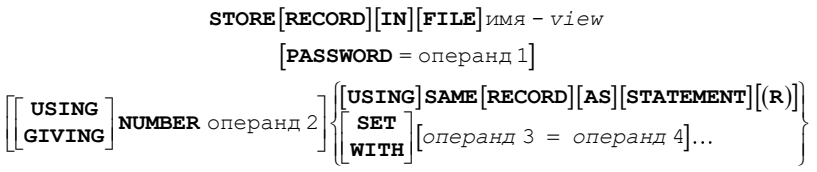

Рис. 69. Синтаксис оператора STORE (режим отчета)

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 также должно быть предусмотрено значение для поля главного ключа.

Далее приводится описание 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.

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

PASSWORD/CIPHER

Данные предложения применимы 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, так как 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 длину 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, так как 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'МЕДИЦИНСКАЯ СПРАВКА (есть-1, нет-0)' MN /
1780'КОПИЯ/ОРИГИНАЛ МЕДИЦИНСКОЙ СПРАВКИ (оригинал-1, ко-
пия-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 ИНДИВИДУАЛЬНЫЙ 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 КОР-
РЕКТНО (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 НАЛИЧИИ МЕДАЛИ КОР-
РЕКТНО (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 МЕДИ-
ЦИНСКОЙ 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
   2230
   2240
```

** 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 'АБИТУРИЕНТ С ТАКИМ ИНДИВИДУАЛЬНЫМ 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
   6170
   6180
   6190END OF TRANSACTION
   6200
   6210RUN 'MAINA'
   6220END
```

5.2. Поиск записей

В системе Natural поиск записей осуществляется при помощи оператора FIND. Синтаксис оператора может быть описан следующим образом (рис. 70, табл. 8).

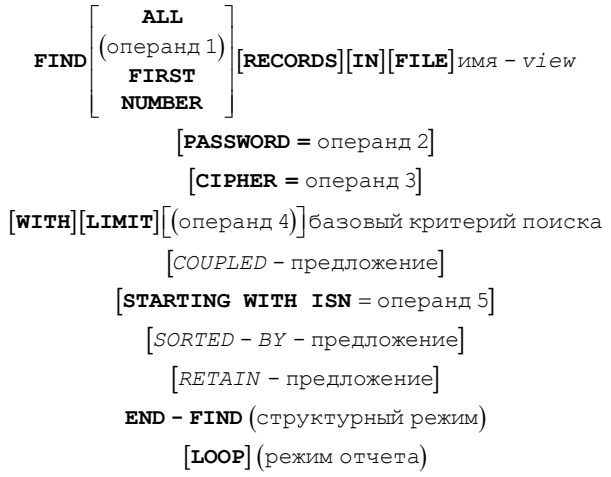

Рис. 70. Синтаксис оператора FIND (режим отчета)

Таблица 8

Описание операндов оператора FIND

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.

***Предложение* CIPHER**

Шифр может быть определен как числовая константа (8 цифр) или как заданная 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 предложении WITH можно использовать дескрипторы, субдескрипторы, супердескрипторы, гипердескрипторы и фонетические дескрипторы. На мейнфрейме можно также определить недескриптор (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 сочетании 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.

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

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. 71).

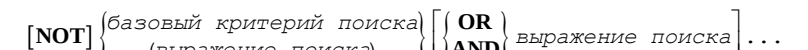

Рис. 71. Синтаксис для комплексного выражения поиска

Примеры комплексного выражения поиска в предложении 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 = 'WASH-
INGTON1 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, супердескрипторы, 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;

– значения переменных пользователя могут быть заданы с помощью оператора REDEFINE для выделения частей полей, из которых формируется значение субдескриптора или супердескриптора.

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)

STARTING WITH ISN=операнд 5

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.

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 приложении клиент-сервер, где чтение записей выполняется серверной программой, в то время как дальнейшая обработка записей — клиентской программой, а записи обрабатываются не все сразу, а порциями (группами).

Фактически начальным будет не значение, указанное в операнде 5, а следующее за ним значение.

Предложение SORTED BY

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

Синтаксис предложения показан на рис. 72.

```
SORTED [BY] дескриптор ... 3 [DESCENDING]
```

Рис. 72. 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.

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 мейнфреймов.

Предложения SORTED BY и RETAIN не могут использоваться вместе.

Пример предложения SORTED BY

** Определение области данных:*

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

** Поиск данных с использованием 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
```

Имя набора — операнд6

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 (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
```

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 доступен 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 равна "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.

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 цикл 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 доступен 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.

5.3. Организация выдачи данных на экран монитора и печать

Синтаксис оператора DISPLAY показан на рис. 73–76 и в табл. 9.

Рис. 73. Синтаксис оператора DISPLAY

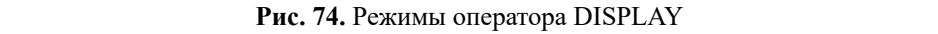

Рис. 74. Режимы оператора DISPLAY

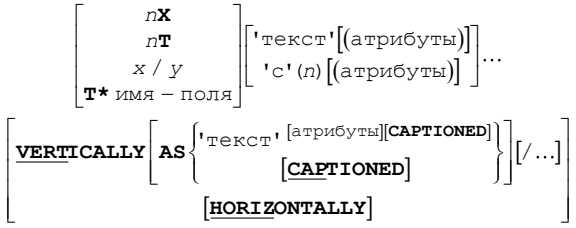

Рис. 75. Формат ввода оператора DISPLAY

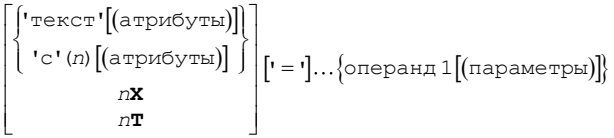

Рис. 76. Элемент ввода оператора DISPLAY

Таблица 9

Описание операндов элемента вывода оператора DISPLAY

Операнд	Возможная структура	Возможные форматы																Разрешение ссылки	Динамическое определение
		S	A	G	N	A	N	P	I	F	B	D	T	L		G	O		
Операнд 1																		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 отчета* (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. 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.

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

DISPLAY ‘EMPLOYEE’ NAME 'SALARY' SALARY

2. 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 столбцов, 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

GIVE SYSTEM FUNCTIONS

Предложение GIVE SYSTEM FUNCTIONS используется, чтобы сделать доступными системные функции AVER, COUNT, MAX, MIN, NAVER, NCOUNT, NMIN, SUM, TOTAL, которые вычисляются во время выполнения оператора DISPLAY, содержащего предложение GIVE SYSTEM FUNCTIONS.

На эти системные функции можно затем ссылаться в любом операторе, выполняемом в ситуации конца страницы (end-of-page).

В отчете может быть только один оператор DISPLAY, содержащий предложение GIVE SYSTEM FUNCTIONS. Системные функции в операторе DISPLAY вычисляются для отдельных страниц, т. е. когда инициируется новая страница, все функции (кроме TOTAL) принимают нулевое значение.

Если системные функции используются в операторе DISPLAY, расположенном в процедуре, ситуация "конец страницы" должна обрабатываться в этой же процедуре.

Параметры

Один или несколько параметров, заключенных в круглые скобки, могут быть заданы.

Каждый заданный параметр будет перекрывать соответствующий параметр, ранее определенный в команде GLOBALS либо 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.

Атрибуты позиционирования поля формата вывода и регулирования заголовка столбца форматов вывода показаны в табл. 10, 11.

Таблица 10

Система позиционирования поля формата вывода

Атрибут	Описание
*n***X**	Пример: DISPLAY NAME **5X** SALARY Обозначение используется для вставки *n* пробелов между столбцами, *n* не должно быть равно 0
*n***T**	Обозначение вызывает позиционирование (табуляцию) на позицию *n*. 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*	Обозначение вызывает размещение следующего элемента в строке *x* и в столбце *y*; *y* не должен быть равен 0. Обратное позиционирование не допускается
T * *имя-поля*	Обозначение вызывает позиционирование на ту же колонку, где поле '*имя-поля*' было в предыдущем операторе DISPLAY. Обратное позиционирование не допускается
P * *имя-поля*	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

Таблица 11

Регулирование заголовка столбца формата вывода

Атрибут	Описание
‘=’	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

Окончание табл. 11

Атрибут	Описание
'*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

Указывает атрибуты изображения (табл. 12) и цвета (табл. 13), которые будут использоваться для отображения текста.

Таблица 12

Атрибуты изображения

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

Таблица 13

Атрибуты цвета

Атрибут	Цвет	Атрибут	Цвет	Атрибут	Цвет
BL	Синий	**PI**	Розовый	**TU**	Бирюзовый
GR	Зеленый	**RE**	Красный	**YE**	Желтый
NE	Нейтральный				

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, т. е. ниже NAME.

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

Заголовки столбца получаются и используются системой Natural в соответствии 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 случав выдается сообщение об ошибке.

Перевод строки (/)

Этот символ вызывает переход на следующую строку, если он задан внутри текстового элемента.

Если этот символ задается между выходными элементами, то элемент, указанный после '/', будет размещаться по вертикали в том же самом столбце. Заголовок для этого столбца будет формироваться путем размещения заголовков выводимых элементов по вертикали над столбцом.

Действия, выполняемые по умолчанию

Для оператора DISPLAY по умолчанию применяются следующие правила.

1. Значение ширины отчета по умолчанию назначается во время установки системы Natural. Это значение равно 132 для пакетного режима или длине строки терминала для режима телеобработки; может быть отменено параметром сеанса LS. В режиме телеобработки длина строки (LS) и размер страницы (PS) устанавливаются системой Natural на основании физических характеристик используемого терминала.

2. Первой физически выводимой колонкой на экране терминала является колонка 2 (так как колонка 1 резервируется для использования в качестве атрибута на терминале типа 3270). При выводе отчета на бумагу печать начинается с крайней левой колонки (колонка 1).

3. По умолчанию между элементами пропускается одна позиция, что является минимальным зарезервированным расстоянием между столбцами для терминала. Это значение может быть отменено (изменено) параметром сеанса SF.

4. Ширина столбца отчета равна значению длины поля или заголовка, в зависимости от того, что из них больше (если не

задан параметр HW). Если длиннее поле, то заголовок будет центрироваться, если только не заданы параметры HC=L или HC=R для выравнивания заголовка влево или вправо. Если длиннее заголовок, то поле будет прижато влево под заголовком. Значения полей прижаты влево для алфавитно-цифровых полей и вправо для числовых полей. Числовые поля можно прижать влево, определив AD=L. Алфавитно-цифровые поля можно прижать вправо, определив AD=R. В вертикальном режиме ширину столбца определяет самое длинное значение поля или самый длинный заголовок среди всех выводимых полей (если не используется параметр HW).

5. При печати числового поля резервируется одна дополнительная позиция для знака. Подавить печать знака можно параметром сеанса SG.

6. Перед выполнением оператора DISPLAY проверяется переполнение страницы. Во время выполнения оператора DISPLAY не генерируется заголовок новой страницы или хвостовая информация.

Пример использования оператора DISPLAY

** Чтение первых 4 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 показан на рис. 77.

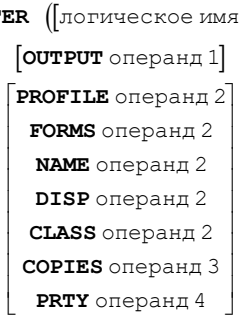

Рис. 77. Синтаксис оператора DEFINE PRINTER

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.

Логическое имя принтера — имя, назначаемое принтеру с номером *n* (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 мейнфреймов).

Закрытие принтера осуществляется при помощи оператора CLOSE PRINTER. Синтаксис оператора показан на рис. 78. Принтер закрывается также при повторном использовании оператора DEFINE PRINTER и при завершении программы.

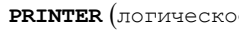

Рис. 78. Синтаксис оператора CLOSE PRINTER

Примеры 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
```

** Определение принтера для печати отчета (REPORT1):*

```
DEFINE PRINTER (REPORT1 = 1) OUTPUT 'LPT1'
```

** Печать отчета (REPORT1):*

```
WRITE (REPORT1) 'REPORT1 PRINTED ON PRINTER LPT1'
```

** Закрытие принтера и конец программы:*

```
CLOSE PRINTER (REPORT1)
END
```

Выводы по главе 5

Оператор STORE используется для добавления записи в базу данных.

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 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.

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).

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

Операторы FIND FIRST, FIND NUMBER, FIND UNIQUE 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).

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

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.

Вопросы для самоконтроля

1. Внесение записей в БД: опишите синтаксис оператора STORE.
2. Поиск записей в БД: опишите синтаксис оператора FIND.
3. Организация выдачи данных на экран и печати: опишите синтаксис оператора DISPLAY.
4. Организация выдачи данных на экран и печати: опишите синтаксис оператора DEFINE PRINTER.
5. Операторы доступа и манипулирования данными: опишите синтаксис оператора READ.
6. Операторы доступа и манипулирования данными: опишите синтаксис оператора HISTOGRAM.
7. Операторы доступа и манипулирования данными: опишите синтаксис оператора GET.
8. Операторы доступа и манипулирования данными: опишите синтаксис оператора UPDATE.
9. Операторы доступа и манипулирования данными: опишите синтаксис оператора DELETE.

Заключение

Спектр технологий обработки больших данных велик, и выбор той или иной технологии обусловлен спецификой задачи, которую требуется решить в конкретных условиях, в конкретное время с учетом определенных ограничений в ресурсах.

Отрасль больших данных развивается быстрыми темпами. Одним из наиболее актуальных направлений развития является применение облачных технологий при обработке больших данных промышленных предприятий с целью построения цифровых двойников. Также перспективно развитие машинного обучения и интеллектуального анализа больших данных.

СУБД ADABAS является одной из наиболее эффективных в современном мире. Материалы данного учебного пособия позволят обучающимся освоить основные методы обработки больших данных в СУБД ADABAS и получить необходимые для этого навыки.

Библиографический список

Использованная литература

1. *Автоматизированные* информационные технологии в экономике: учебник / М. И. Семенов, И. Т. Трубилин, В. И. Лойко, Т. П. Барановская; под общ. ред. И. Т. Трубилина. М.: Финансы и статистика, 2000. 416 с.

2. *Бердышев А. В.* Искусственный интеллект как технологическая основа развития банков // Вестник университета. 2018. № 5. С. 91–94.

3. *Вайгенд А.* BIG DATA. Вся технология в одной книге. М.: Эксмо, 2018. 384 с.

4. *Власов А. И., Лыткин С. Л., Яковлев В. Л.* Краткое практическое руководство разработчика информационных систем на базе СУБД Oracle: Библиотечка журнала «Информационные технологии». М.: Машиностроение, 2000. 120 с.

5. *Воронов А. А., Кондратьев Г. А., Чистяков Ю. В.* Теоретические основы построение автоматизированных систем управления. Разработка технического задания. М.: Наука, 1977. 232 с.

6. *Воронов М. П., Фатеркин А. С., Часовских В. П.* Информационные технологии в управлении: СУБД ADABAS и проектирование приложений средствами NATURAL. Екатеринбург: УГЛТУ, 2006. 477 с.

7. *Гараева А. Р., Миниханов Р. Н., Дагаева М. В., Кильдеева С. С., Аникин И. В.* Интеллектуальный анализ больших пространственно-временных данных для служб экстренного реагирования // Современные информационные технологии и ИТ-образование. 2018. Т. 14, № 3. С. 679–685.

8. *Гордеев А. В., Молчанов А. Ю.* Системное программное обеспечение. СПб.: Питер, 2001. 736 с.

9. *Зайцев Н. Г.* Информационное и математическое обеспечение АСУП. 2-е изд., испр. и доп. Киев: Техніка, 1974. 144 с.

10. *Зеленков Ю. А.* Введение в базы данных. 1997. URL: http://www.mstu.edu.ru/study/materials/zelenkov/ch_6_2.html.

11. *Инновационная* экономика для современного мира / В. А. Балашов, И. Я. Львович, А. Б. Почтовюк [и др.]. Одесса: КУПРИЕНКО СВ, 2018. 118 с.

12. *Информационные* технологии управления: учеб. пособие / В. П. Часовских, Г. А. Акчурина, А. В. Слободин [и др.]. 4-е изд., испр. и доп. Екатеринбург: УГЛТУ, 2015. 667 с.

13. *Кастеллани К.* Автоматизация решения задач управления: пер. с фр. М.: Мир, 1982. 472 с.

14. *Когаловский М. Р.* Энциклопедия технологий баз данных. М.: Финансы и статистика, 2002. 800 с.

15. *Коннолли Т., Бегг К., Страчан А.* Базы данных: проектирование, реализация и сопровождение. Теория и практика: учеб. пособие: пер. с англ. 2-е изд. М.: Изд. дом «Вильямс», 2000. 1120 с.

16. *Кузнецов Е. Н.* Анализ структуры сетевых взаимодействий: контекстно-зависимые меры центральности // Управление большими системами: сб. тр. 2019. № 80. С. 57–82.

17. *Майер-Шенбергер В., Кукьер К.* Большие данные. Революция, которая изменит то, как мы живем, работаем и мыслим / пер. с англ. И. Гайдюк. М.: Манн, Иванов и Фербер, 2014. 240 с.

18. *Метод* анализа и синтеза модульных информационно-управляющих систем / Н. А. Кузнецов, В. В. Кульба, С. С. Ковалевский, С. А. Косяченко. М.: Физматлит, 2002. 800 с.

19. *Модин А. А., Яковенко Е. Г., Погребной Е. П.* Справочник разработчика АСУ. 2-е изд., перераб. и доп. М.: Экономика, 1978. 583 с.

20. *Основы* систем автоматизированного проектирования / М. М. Берхеев, А. Н. Козин, И. А. Корниенко, Ю. В. Кожевников. Казань: Изд-во Казан. ун-та, 1988. 254 с.

21. *Пустоваров В. И.* Ассемблер: программирование и анализ корректности машинных программ. Киев: Изд. группа BHV, 2000. 479 с.

22. *Селезнев К.* Проблемы анализа Больших Данных // Открытые системы. СУБД. 2012. № 07. URL: https://www.osp.ru/os/2012/07/13017638.

23. *Силен Д., Мейсман А., Али М.* Основы Data Science и Big Data. Python и наука о данных. СПб.: Питер, 2017. 336 с.

24. *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.

25. *Системы* управления базами данных и знаний: справ. изд. / А. Н. Наумов, А. М. Вендров, В. К. Иванов [и др.]; под ред. А. Н. Наумова. М.: Финансы и статистика, 1991. 352 с.

26. *Справочник* проектировщика систем автоматизации управления производством / под ред. Г. Л. Смилянского. 2-е изд., перераб. и доп. М.: Машиностроение, 1976. 590 с.

27. *Тиндал С.* Большие данные: все, что вам необходимо знать // PC Week/RE. 2012. № 25 (810). URL: https://www.itweek.ru/idea/article/detail.php?ID=141962.

28. *Управление* ИТ-разработкой и внедрением: учеб. пособие / В. П. Часовских, М. П. Воронов, В. Г. Лабунец, Е. Н. Стариков. Екатеринбург: УрГЭУ, 2021. 173 с.

29. *Федотов Н. Н., Венчковский Л. Б.* Средства информационного обеспечения автоматизированных систем управления. М.: Изд-во стандартов, 1989. 192 с.

30. *Формализация* информации и big data: учеб. пособие / В. П. Часовских, М. П. Воронов, В. Г. Лабунец [и др.]. Екатеринбург: УрГЭУ, 2021. 218 с.

31. *Хансэн Г., Хансэн Дж.* Базы данных: разработка и управление: пер. с англ. М.: БИНОМ, 1999. 704 с.

32. *Черпаков И. В.* Теоретические основы информатики: учебник и практикум для акад. бакалавриата. М.: Юрайт, 2017. 353 с.

33. *Чехарин Е. Е.* Большие данные: большие проблемы // Перспективы науки и образования. 2016. № 3 (21). С. 7–11.

34. *Экономическая* информатика / под ред. П. В. Конюховского, Д. Н. Колесова. СПб.: Питер, 2000. 560 с.

35. *Экономическая* информатика: учеб. для вузов / В. В. Евдокимов, Ю. Б. Бекаревич, С. А. Бондаренко, А. М. Власовец; под ред. В. В. Евдокимова. СПб.: Питер, 1997. 592 с.

36. *Bonacich P.* Power and Centrality: A family of Measures // American Journal of Sociology. 1987. № 5 (92). P. 1170–1182.

37. *Bonacich P., Lloyd P.* Eigenvector-like measures of centrality for asymmetric relations // Social Networks. 2001. № 3 (23). P. 191–201.

38. *Eisenberg A., Melton J.* SQL Standardization: The Next Steps // ACM SIGMOD Record. 2000.Vol. 29, no. 1. P. 63–67.

39. *Gray J., Chaudhuri S., Bosworth A., Layman A., Reichart D., Venkatrao M., Pellow F., Pirahesh H.* Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab and Sub-Totals // Data Mining and Knowledge Discovery. 1997. Vol. 1, no. 1. P. 29–53.

40. *Kimball R.* The Data Warehouse Toolkit: Practical Techniques for Building Dimensional Data Warehouses. New York: John Wiley & Sons, 1996. 388 p.

41. *Lenz H.-J., Shoshani A.* Summarizability in OLAP and Statistical Data Bases // Scientific and Statistical Database Management: Proc. 9th Int'l Conf. Los Alamitos, Calif.: IEEE CS Press, 1997. P. 132–143.

42. *Osmar R. Zaiane.* Principles of Knowledge Discovery in Databases. 1999. Chapter I: Introduction to Data Mining. URL: https://webdocs.cs.ualberta.ca/~zaiane/courses/cmput690/notes/Chapter1/index.html.

43. *Pedersen T. B., Jensen C. S., Dyreson C. E.* A Foundation for Capturing and Querying Complex Multidimensional Data // Information Systems. 2001. Vol. 26, no. 5. P. 383–423.

44. *Pedersen T. B., Jensen C. S., Dyreson C. E.* Extending Practical Pre-Aggregation in On-Line Analytical Processing // Very Large Databases: Proc. 25th Int'l Conf. San Francisco, CA: Morgan Kaufmann, 1999. P. 663–674.

45. *Sexton D. G., Oh O., Kishor R.* Rules of Crowdsourcing: Models, Issues, and Systems of Control // Information Systems Management. 2013. Vol. 30, issue 1. 40 p.

46. *Thomsen E.* OLAP Solutions: Building Multidimensional Information Systems. New York: John Wiley & Sons, 1997. 576 p.

47. *Vassiliadis P., Sellis T. K.* A Survey of Logical Models for OLAP Databases // ACM SIGMOD Record. 1999. Vol. 28, no. 4. P. 64–69.

48. *Zurek T., Sinnwell M.* Datawarehousing Has More Colours Than Just Black and White // Very Large Databases: Proc. 25th Int'l Conf. San Francisco, CA: Morgan Kaufmann, 1999. P. 726–729.

Рекомендуемая литература

ЫН Анналин, Су Кеннет. Теоретический минимум по Big Data. Всё, что нужно знать о больших данных. СПб.: Питер, 2019. 208 с.

Андреас Вайгенд. BIG DATA. Вся технология в одной книге. М.: Эксмо, 2018. 384 с.

Силен Д., Мейсман А., Али М. Основы Data Science и Big Data. Python и наука о данных. СПб: Питер, 2017. 336 с.

Варнавский А. В., Бурякова А. О., Себеченко Е. В. Блокчейн на службе государства. М: КноРус, 2020. 215 с.

Информация об авторах

Часовских Виктор Петрович — профессор кафедры шахматного искусства и компьютерной математики УрГЭУ, доктор технических наук, профессор
e-mail: u2007u@ya.ru

Лабунец Валерий Григорьевич — профессор кафедры шахматного искусства и компьютерной математики УрГЭУ, доктор технических наук, профессор
e-mail: vlabunets05@yahoo.com

Стариков Евгений Николаевич — заведующий кафедрой шахматного искусства и компьютерной математики УрГЭУ, кандидат экономических наук, доцент
e-mail: starikov_en@usue.ru

Кох Елена Викторовна — доцент кафедры интеллектуальных технологий Уральского государственного лесотехнического университета, кандидат сельскохозяйственных наук, доцент
e-mail: kohev@m.usfeu.ru

Воронов Михаил Петрович — доцент кафедры шахматного искусства и компьютерной математики УрГЭУ, кандидат технических наук, доцент
e-mail: mstrk@yandex.ru

Оглавление

Учебное издание

Часовских Виктор Петрович,
Лабунец Валерий Григорьевич,
Стариков Евгений Николаевич
и др.

Технологии обработки больших данных средствами СУБД ADABAS

Учебное пособие

Редактор и корректор *Л. В. Матвеева*
Компьютерная верстка *И. В. Засухиной*

Поз. 23. Подписано в печать 15.07.2022.
Формат 60 × 84 1/16. Бумага офсетная. Печать плоская.
Уч.-изд. л. 6,6. Усл. печ. л. 10,0. Печ. л. 10,8. Заказ 442. Тираж 37 экз.
Издательство Уральского государственного экономического университета
620144, г. Екатеринбург, ул. 8 Марта/Народной Воли, 62/45

Отпечатано с готового оригинал-макета в подразделении оперативной полиграфии
Уральского государственного экономического университета

55 лет

УРАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ
ЭКОНОМИЧЕСКИЙ УНИВЕРСИТЕТ

ISBN 978-5-9656-0325-1