несколько АСУ создают общее хранилище данных, объединяющее в единой базе данных всю межведомственную информацию. При этом все АСУ передают в единую базу данных свои базы данных или их фрагменты.

Возможно также комбинированное использование этих подходов, когда для вновь создаваемой интегрированной АСУ используется единая база данных, а для уже существующих АСУ, взаимодействующих с данной системой, обмен информацией осуществляется в соответствии с двусторонними протоколами [1].

Анализ приведенных подходов, связанных с физическим перемещением и преобразованием ИР, выявляет следующие их слабые стороны:

- сложность согласования интересов владельцев ИР и потребителей информации;

- дублирование информации, возникающее при передаче данных из одной АСУ в другую;

- недостаточная актуальность данных, представленных из одной АСУ в другую. Например, если протоколом информационного обмена предусмотрена передача информации ежемесячно, то данные отражают состояние не на текущий, а на предыдущий месяц;

- технологическая сложность процессов слияния данных, связанная с необходимостью написания процедур экспорта-импорта, а также с объединением и хранением больших массивов данных, созданных в разное время различными разработчиками на разнородных технологических платформах;

- рассредоточение правил информационного обмена по различным межведомственным протоколам и отражение по частям в алгоритмах процедур, обеспечивающих обмен данными.

1 Постановка задачи

Перечисленные недостатки описанных выше подходов объединения АСУ для совместного решения задач стимулируют поиск иных решений, обеспечивающих совместимость управленческих и технологических процессов, выполняемых в различных АСУ. В качестве такого решения можно использовать процедуру интеграции данных, в соответствии с которой осуществляется обмен ИР в рамках единого адресного пространства и обеспечивается прозрачный доступ должностных лиц к этим ресурсам в соответствии с их полномочиями.

Интеграция данных — это процесс комбинирования данных из разнообразных источников и предоставление их пользователю в унифицированном виде.

Необходимость в интеграции данных возникает тогда, когда данные из различных источников, имеющие различные формы представления, используются в рамках единого процесса.

Интеграция данных позволяет реализовать эффективный доступ к внешним гетерогенным источникам данных посредством единого интерфейса в рамках единой модели данных. При этом с точки зрения пользователя обращение к внешним источникам ничем не отличается от обращений к единому источнику данных.

Весь ход исторического развития информационных технологий можно рассматривать как эволюцию способов интеграции данных на различных уровнях. Можно выделить следующие этапы интеграции данных:

- на первом этапе в ходе формирования файловой системы (ФС) была реализована интеграция данных, хранящихся на различных типах устройств внешней памяти в рамках одной ЭВМ;

- на втором этапе при разработке концепции баз данных была осуществлена интеграция данных в рамках локальной сети;

- настоящий этап развития информационных технологий можно рассматривать как этап интеграции гетерогенных данных в рамках глобальной сети путем разработки интегрированных АСУ.

На сегодняшний день общепринятой методологии интеграции данных на уровне глобальной сети не существует. Поэтому целью данной статьи является разработка формализованной модели процесса интеграции, которую можно использовать при разработке унифицированных средств взаимодействия АСУ интегрированных систем.

2 Этапы интеграции данных

2.1 Интеграция данных с помощью файловой системы

В первые десятилетия развития вычислительной техники использовались два вида устройств внешней памяти: магнитные ленты и магнитные барабаны, затем появились магнитные диски (гибкие и жесткие). В то время файловая система еще не была разработана и каждая прикладная программа, которой требовалось хранить данные во внешней памяти, сама определяла расположение каждой порции данных на магнитной ленте или барабане и выполняла обмен между оперативной и внешней памятью. При этом программы обмена данными магнитного барабана, ленты и диска, реализуемые на уровне программно-аппаратных средств низкого уровня, существенно различались и требовали больших затрат на разработку (см. рис. 1).

С осознанием этой проблемы возникла необходимость в унификации процедур обмена данными прикладных программ с различными типами внешней памяти. Эта работа была проведена в ходе создания ФС и описывается следующим образом.

Для решения поставленной задачи, прежде всего, необходимо решить проблему различия представления данных в прикладных программах и в физических устройствах внешней памяти. С этой целью были определены:

- структура физической записи для каждого типа внешней памяти;

- дескриптор файла - обобщенная структура хранения данных в виде последовательности логических записей, связанная со структурой физической записи для каждого типа внешней памяти;

- стандартная спецификация процедур записи/считывания данных в терминах языка программирования высокого уровня.

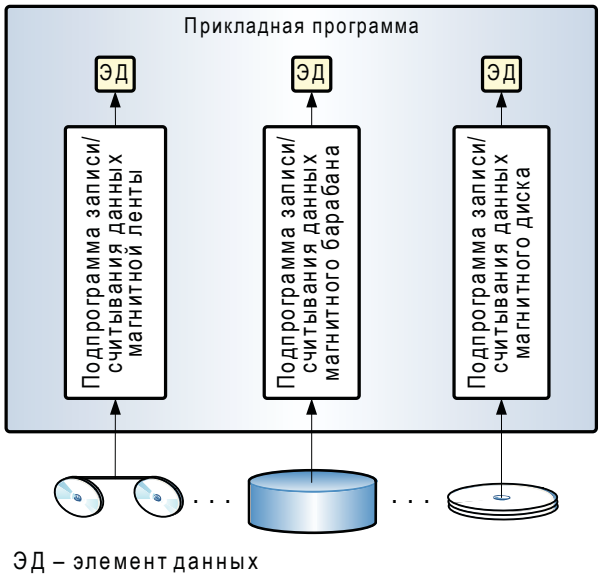


Рис. 1. Схема обмена данными между прикладной программой и внешней памятью без использования ФС

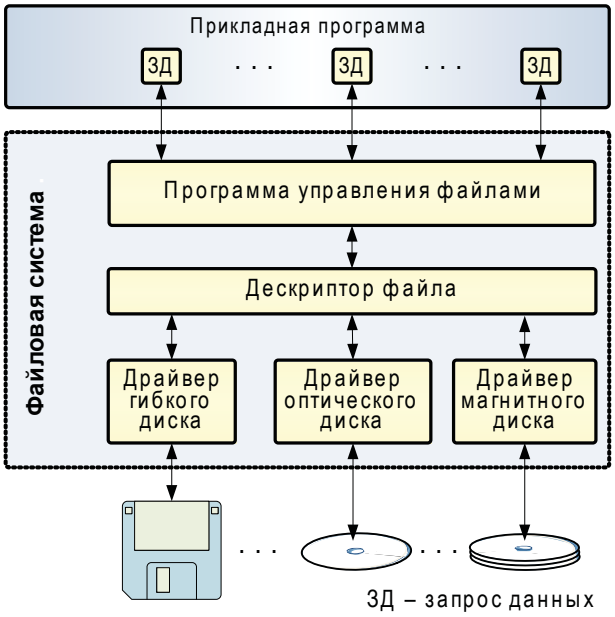


Рис. 2. Схема обмена данными между прикладной программой и внешней памятью с использованием ФС

После определения статических компонентов ФС требуется связать их между собой. Для этого были разработаны:

- набор драйверов устройств, представляющих собой программы, взаимодействующие с устройствами при выполнении различных заданий. Драйверы осуществляют преобразование логического и физического представлений данных, абстрагируя физические блоки данных в записи файлов;

- программа доступа к файлу из прикладной программы, называемая программой управления файлами. Эта программа обеспечивает связь запросов данных, сформулированных в терминах стандартной спецификации процедур записи/считывания, с логической структурой файла.

Процедура доступа к данным с помощью ФС начинается с того, что у программы управления файлами запрашивается доступ к файлу. При этом осуществляется процесс, называемый открытием файла, т.е. обеспечивается доступ к данным о типе устройства внешней памяти, пути и имени файла и т.д. Данная информация хранится в дескрипторе файла, размещаемом при работе с файлом в оперативной памяти.

После открытия файла активизируется драйвер для данного типа внешней памяти, управляющий записью и чтением данных в физическом устройстве (см. рис. 2).

Драйверы каждого типа внешней памяти различны, но программы, написанные на языке высокого уровня, обращаются к ним со стандартными запросами. Поэтому прикладному программисту не нужно обладать знаниями технических деталей, чтобы считать или записать данные на различные типы запоминающих устройств. В программах формируется стандартный запрос на чтение или запись, после чего ФС определяет требуемый драйвер, который выполняет эти операции на физическом устройстве с учетом технических деталей того или иного типа внешней памяти [2].

2.2 Интеграция данных с помощью СУБД

С расширением сферы применения ЭВМ в конце 70-х годов начали создаваться АСУ, осуществляющие комплексную автоматизацию деятельности предприятий или организаций. На первых этапах развития технологии АСУ вопросы описания данных решались индивидуально в каждом программном модуле. Но это приводило к рассогласованию и дублированию данных в разных программных модулях. Поэтому в дальнейшем для хранения данных АСУ создавались файлы коллективного доступа, разрабатываемые в централизованном порядке.

Но у файлов коллективного доступа имелись серьезные недостатки. Структура записей в файлах и наборы отношений, реализуемые в

прикладных программах, были «жесткими», и их изменение означало перестройку отношений в прикладных программах.

Сложные структуры данных, используемые при решении задачи комплексной автоматизации, связаны между собой не менее сложными взаимосвязями. Но ФС не поддерживает отношения между данными, содержащимися в файлах коллективного доступа. Поэтому поддержку связей между данными приходилось реализовывать в прикладных программах. Эти дополнительные средства управления данными составляли существенную часть прикладных программ и практически повторялись от одной системы к другой.

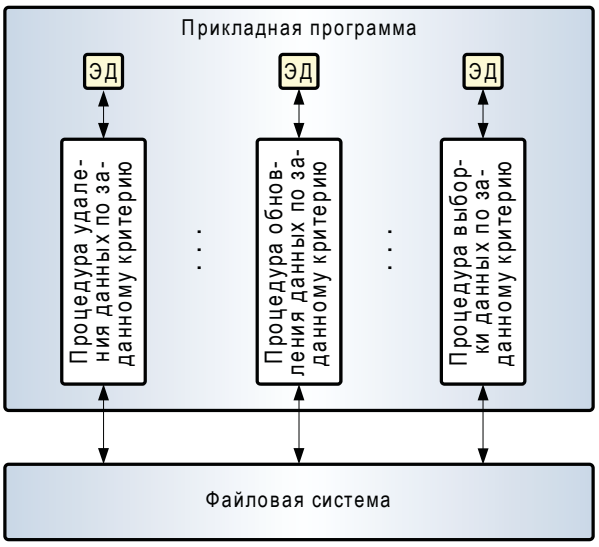
Стремление выделить и обобщить повторяющиеся фрагменты прикладных программ, ответственные за управление сложно структурированными данными коллективного доступа, явилось первой побудительной причиной создания систем управления базами данных (СУБД).

Ранние СУБД использовали иерархические и сетевые связи между элементами данных. В иерархических и сетевых базах данных (БД) для установления связей между сущностями использовались низкоуровневые физические указатели. Это позволяло извлекать данные, связанные определенными отношениями. Однако отношения, определенные на физическом уровне, не были описаны на логическом уровне, в результате запросы к базам данных приходилось программировать при возникновении необходимости, на что уходили дни и недели.

Иерархические и сетевые СУБД поддерживали связи между структурами данных, но не смогли решить проблему повторяемости, и прикладные программы по-прежнему содержали практически одинаковые фрагменты кода, связанные с обеспечением доступа к элементам данных, хранящихся в базах данных (см. рис. 3).

В 1970 г. Кодд опубликовал революционную по содержанию статью, которая всерьез поколебала устоявшееся представление о БД. Он выдвинул идею, что данные нужно связывать в соответствии с их внутренними логическими взаимоотношениями, а не физическими указателями. Для этого было выбрано табличное (реляционное) представление информации, являющееся естественной формой представления логических отношений.

Данный подход позволил комбинировать данные из разных источников (таблиц), если логическая информация, необходимая для такого комбинирования, присутствует в исходных данных. Это открыло новые возможности для информационных систем, поскольку запросы к БД теперь не были ограничены физическими указателями. При этом было использовано еще одно новшество: логическая информация о взаимосвязях элементов данных была выделена и хранилась в системном каталоге. Формат этой информации был стандартизирован, что позво-



ЭД – элемент данных

Рис. 3. Схема обмена данными между прикладной программой и файлами коллективного доступа

лило разработать унифицированный язык запросов SQL, обеспечивающий декларативную формулировку запросов.

Таким образом, в ходе разработки реляционных СУБД были унифицированы процедуры доступа к данным со сложными взаимосвязями. При этом была решена проблема различия представления данных в файловой системе, связанных с физическими указателями, и в СУБД, оперирующей логическими взаимоотношениями, легко воспринимаемыми на уровне сознания. С этой целью были определены:

- табличные структуры данных на уровне файловой системы, представленные последовательностями записей данных a[1], a[2], ..., a[m];

- организация элементов данных в табличную форму с учетом связей между таблицами, которые на логическом уровне описываются в фиксированных структурах, называемых метаданными. Такое описание низкоуровневых данных называется моделью данных. Модель определяет единицы данных, а также специфицирует связи каждой единицы данных с другими единицами данных;

- спецификация языка запросов, с помощью которого доступ к данным, представленным в табличной форме, формулируется в терминах наименований таблиц, их столбцов и логических взаимосвязей между значениями атрибутов.

Связи между перечисленными уровнями описания данных реализованы в рамках реляционных СУБД в виде следующих программных компонентов:

- унифицированных процедур обработки произвольных табличных структур (напри-

мер, процедуры, реализующие операторы INSERT, UPDATE, SELECT и т.д.). Эти процедуры обеспечивают выполнение запросов, сформулированных в терминах табличных структур, на уровне последовательностей записей файлов. Унификация процедур обеспечивается наличием метаданных фиксированной структуры;

- интерфейсного модуля, который преобразует запросы в терминах спецификации языка запросов в вызовы соответствующих унифицированных процедур. При этом интерфейсный модуль, используя модель данных, определяет и входные атрибуты унифицированной процедуры в виде наименований таблиц, атрибутов и условий выборки.

Использование модели данных и унифицированного языка для доступа к базе данных обеспечивает выполнение так называемых «незапланированных» запросов. Реляционная СУБД, обладая достаточными знаниями о предметной области, которыми можно воспользоваться в любой момент времени, обеспечивает формулировку произвольного запроса на унифицированном языке запросов (обычно на языке SQL). Такой запрос может быть подан с терминала или встроен в прикладную программу, входящую в информационную систему.

Основное достоинство реляционных СУБД заключается в том, что они позволили абстрагировать данные, поскольку освободили разработчиков от необходимости программировать операции с данными и позволили выйти на декларативный уровень запросов к базе данных [3].

Запрос на языке SQL поступает на вход интерфейсного модуля, который осуществляет его лексический, синтаксический и семантический анализ путем сопоставления элементов языковой конструкции с данными, хранящимися в системном каталоге. Далее СУБД трансформирует логическую структуру данных из системного каталога в физическое представление, т.е. вызывает соответствующую процедуру, реализующую один из операторов языка SQL (SELECT, INSERT, UPDATE, DELETE, ...).

Именно эти процедуры осуществляют преобразование физической модели данных в логическую и обратно, абстрагируя физические блоки данных в терминах записей файлов в записи баз данных в терминах табличного представления информации.

Таким образом, СУБД находится между «логикой» и «физикой» так же, как и файловая система (см. рис. 4). Но если в первом случае на логическом уровне были унифицированы запись и считывание данных, то во втором случае унифицированы операции с данными, связанные с содержанием данных.

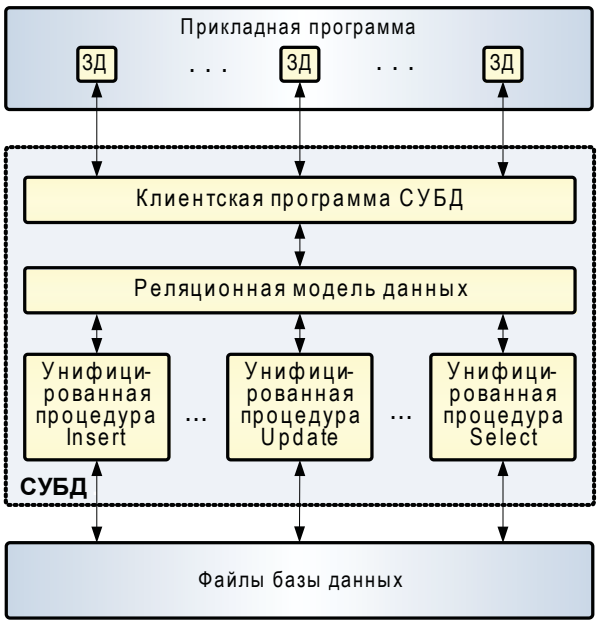
Суть интеграции данных, реализуемой СУБД, заключается в следующем. Вместо файлов коллективного доступа, используемых в ранних АСУ, были реализованы БД и СУБД. В БД размещались структурированные данные со сложными взаимосвязями, а СУБД поддерживала типовые операции над этими данными, используя сведения об их структуре и взаимосвязях. Это позволило разместить общие данные в БД на серверной машине и обеспечить к ним многопользовательский доступ из приложений, размещенных на различных машинах в пределах локальной сети. При этом прикладные программы получали доступ к данным любых других прикладных программ, используя декларативный язык запросов SQL.

2.3 Интеграция данных в ИАСУ

На сегодняшний день интеграция данных из гетерогенных источников, распределенных в глобальной сети, является одним из магистральных направлений исследований по разработке информационного обеспечения.

Отмеченные недостатки подходов объединения АСУ связаны с тем, что совместимость данных при решении задач обеспечивается физической передачей данных, сопровождаемой сложными, дорогостоящими процедурами импорта-экспорта и преобразования информации на приемном пункте.

Сложность совмещения данных из гетерогенных источников, распределенных в глобальной сети, объясняется наличием разнородных ИР, имеющих формализованную и неформализованную структуру контента. Это усложняет унифицированный доступ к данным в АСУ, так как каждый вид ИР создается с помощью соответствующего программного средства, что требует соответствующего количества интерфейсов для работы со всеми видами ИР (см. рис. 5).



ЗД – запрос к данным в соответсвии со спецификацией SQL

Рис. 4. Схема обмена данными между прикладной программой и базой данных

В отличие от двух предыдущих этапов, данный этап интеграции находится еще в процессе становления и представлен в виде различных подходов, описанных во введении. Решения, предлагаемые в этих подходах, не отличаются универсальностью, не реализуют всех ключевых аспектов интеграции данных и не могут рассматриваться как полное решение проблемы интеграции гетерогенных ИР в АСУ.

В наибольшей степени принципу интеграции соответствует способ согласования ИР, называемый федерализацией данных и используемый в таких системах, как IBM Information Server, IBS Catalog и т.д. [4]. *Федеративное объединение данных* - это объединенное представление данных из множества различных источников. Данный подход к интеграции приобретает все большую популярность в качестве эффективного средства виртуального объединения данных из разных источников в реальном времени, то есть без физического их перемещения.

Для решения задачи унификации процедур доступа к данным из гетерогенных источников путем федерализации необходимо определить:

- типы и форматы разнородных ИР (формализованных и неформализованных), циркулирующих в системе. В качестве типов и форматов ИР могут рассматриваться: текстовые данные (формат rtf), табличные данные (формат Excel), презентационные данные (формат PowerPoint), фактографические данные, картографическая информация, Web-страницы и т.д.;

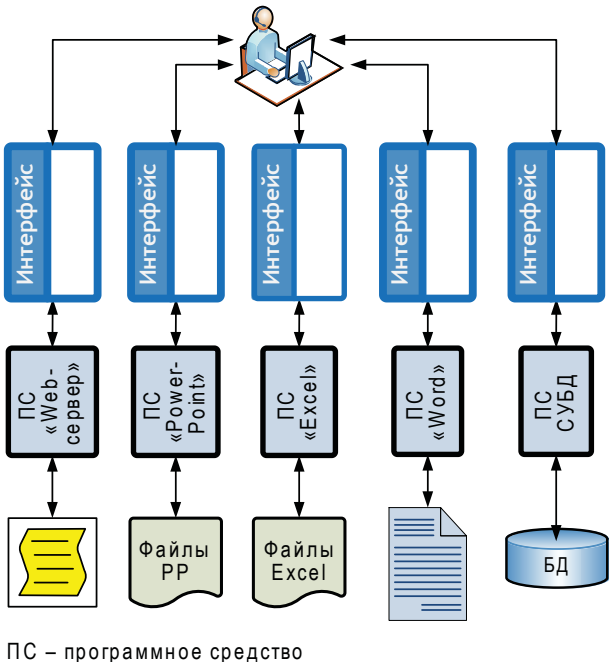
- базу метаданных, обеспечивающую унифицированное описание разнородных ИР (формализованных и неформализованных). В базе метаданных содержится информация о типах и форматах данных (т.е. о программном средстве, поддерживающем данный тип ИР), о путях доступа к ИР, об атрибутах формализованных ИР и т.д.;

- графический интерфейс пользователя, обеспечивающий унифицированную спецификацию для доступа к разнородным ИР (формализованным и неформализованным).

Для объединения перечисленных компонентов в единую систему необходимо разработать процессор федерализации данных, который кроме базы метаданных и графического интерфейса пользователя содержит:

- унифицированный модуль вызова программных средств для доступа к разнородным ИР (формализованным и неформализованным). Данный модуль для физической обработки ИР загружает соответствующее программное средство, используя сведения из базы метаданных о типе ИР, пути доступа и атрибутах объекта (для формализованных ИР);

- программный модуль, включающий в свой состав базовые сервисы доступа к базе метаданных, используемые для преобразования запросов, сформулированных с помощью графического интерфейса пользователя, в вызовы программных средств для доступа к ИР (формализованным и неформализованным).



ПС – программное средство

Рис. 5. Схема доступа к гетерогенным данным через интерфейсы программных систем

Федерализация данных обеспечивает единую виртуальную картину по всем источникам данных. Пользователь или приложение направляет запрос к этой виртуальной картине, а процессор федерализации данных извлекает данные из соответствующих источников и отправляет результаты пользователю или приложению, от которого пришел запрос.

По определению процесс федерализации данных всегда заключается в *извлечении* данных из первичных источников на основании внешних требований. Все необходимые преобразования данных осуществляются при их извлечении из первичных файлов.

Один из ключевых элементов федеративной системы - это метаданные, которые используются процессором федерализации данных для доступа к первичным данным (см. рис. 6).

Основная проблема, требующая своего решения на данном этапе, — это наличие большого объема неформализованных документов, реализованных в виде текстовых файлов и Web-страниц. Данные, содержащиеся в этих ИР, не могут быть подвержены автоматизированной обработке, что существенно ограничивает возможности сегодняшних АСУ по интеграции данных.

Поэтому необходимо сократить долю неформализованных данных путем разработки новой модели данных, охватывающей более широкий спектр ИР. Проблематика моделирования данных связана с таким представлением данных, которое наиболее естественно отра-

жает описываемые объекты. Основной единицей обработки в АСУ является документ, поэтому необходима модель, ориентированная на эффективное представление его статических и динамических компонентов.

В настоящее время строго формализованной теории моделей данных не существует. Но в нашем распоряжении имеется опыт разработки моделей данных при разработке ФС, СУБД, процессора федерализации данных. В соответствии с целью статьи в последующих разделах этот опыт будет обобщен и описаны механизм и формальная модель интеграции данных.

3 Механизмы процесса интеграции

3.1 Предпосылки интеграции

На первый взгляд может показаться, что при формировании ФС, концепции БД и решении задачи обеспечения совместимости разнотипных ИР в глобальной сети решались совершенно разные задачи.

Но на каждом из описанных этапов развития информационных технологий разработчики сталкивались с одними и теми же проблемами. Это:

- разнообразие способов представления данных: на магнитных барабанах, лентах и дисках при формировании ФС; фрагментарное описание данных предметной области в различных прикладных программах при формировании концепции БД; текстовые, фактографические, мультимедийные ИР при интеграции гетерогенных АСУ;

ПС – программное средство

Рис. 6. Схема доступа к гетерогенным данным через унифицированный интерфейс при федерализации данных

- низкий уровень абстракции данных относительно решаемых на данном этапе задач, вследствие чего в прикладных программах приходится иметь дело с низкоуровневым программированием, повышающим трудоемкость разработки;

- малая адекватность отражения семантики предметной области существующими способами представления данных, вследствие чего логика представления для пользователя и в программах существенно различается, что приводит к усложнению взаимодействия пользователя с прикладными программами.

Если проанализировать способы решения этих проблем на описанных этапах развития информационных технологий, то оказывается, что при этом были использованы одни и те же механизмы, обеспечивающие эффективную организацию доступа к внешним, гетерогенным источникам данных посредством единого интерфейса в рамках единой модели данных.

3.2 Многоплановое представление данных

Обобщив опыт решения задач по доступу к информационным ресурсам, можно констатировать, что организация информации в машине связана с ее многоплановым представлением на следующих уровнях.

Низкоуровневые данные, как правило, фрагментированы и рассредоточены по различным источникам. Данные на этом уровне представлены с учетом особенностей формата и физической среды хранения для каждого источника данных.

Например, таблица на физическом уровне представлена последовательностью элементов данных a[1], a[2], ..., a[m], размещенных на поверхностях, цилиндрах и секторах магнитного диска. Операции на этом уровне управляют работой устройства хранения данных по определению поверхностей, цилиндров и секторов.

Модель данных – атрибутивная форма представления свойств и связей объектов предметной области, ориентированная для описания средствами формальных языков.

Например, таблица на уровне языка программирования высокого уровня описывается выражением Array[i, j], а операции реализованы в виде функций insert (Array, i), delete (Array, i) и т.д.

На уровне модели данных появляется целостная картина имеющейся фрагментированной по различным источникам информации. Информационная модель должна быть опубликована и доступна пользователям и прикладным программам.

Концептуальное представление – это представление объектов на уровне человеческого восприятия. Например, таблица Array[i, j] на концептуальном уровне описывается как графическое изображение

таблицы с заголовком, столбцами, строками и содержимым полей. А операции с таблицей insert (Array, i), delete (Array, i) и т.д. представлены в виде командных строк «Вставить строку», «Удалить строку» и т.д.

Элементы данного уровня по существу представляют собой перечень информационных услуг, оказываемых потребителям информации. Этот перечень должен быть опубликован и доступен абонентам. Именно через эту публикацию осуществляется доступ к элементам информационной модели, так как она воспринимается только через элементы концептуального представления.

Перечисленные планы представления данных объективно присутствуют в любой автоматизированной системе, где в процессе решения задач совместно с машиной принимает участие и человек. И именно несоответствие способов представления данных в человеческой и машинной памяти является основным движущим фактором развития информационных технологий.

Сферы применения информационных технологий постоянно расширяются, и со временем появляются такие задачи, решение которых требует более тесного взаимодействия с человеком. В результате использование текущих моделей данных усложняет взаимодействие с человеком и становится нецелесообразным. С целью уменьшения несоответствия между имеющейся моделью данных и классом решаемых задач формируется новая модель данных, представляющая исходные данные и операции над ними в более привычном для человека виде.

На последующих этапах модели данных, используемые на ранних этапах, рассматриваются как низкоуровневое представление, а в качестве модели данных рассматривается новое высокоуровневое представление, семантически более близкое к человеческим представлениям о мире. Таким образом, на каждом этапе эволюции средств доступа к данным происходит уменьшение семантического разрыва между представлением данных в человеческой и машинной памяти.

3.3 Абстрагирование и унифицированная среда взаимодействия как способы интеграции данных

Отображения как связующие звенья между разноплановыми представлениями данных.

Описанные уровни представления необходимо объединить в единую систему путем реализации отображений между:

- низкоуровневыми данными и моделью данных;
- моделью данных и концептуальным представлением.

Эти отображения реализуются в виде программных модулей и выполняют следующие функции:

- отображение между низкоуровневым представлением и моделью данных осуществляется с помощью унифицированных процедур, которые реализованы в низкоуровневой среде, но сведения о них содержатся в модели данных. Другими словами, интерфейс типовых процедур принадлежит модели данных, а их реализации функционируют в среде низкоуровневых данных. Например, интерфейс процедуры вставки строки в массив Insert (Array, i) относится к модели данных, а реализация этой процедуры, оперирующая последовательностью данных массива a[1],...,a[m], производит соответствующие изменения в физической памяти;
- отображение между концептуальным представлением и моделью данных - также реализуется с помощью программного модуля, который преобразует чувственно воспринимаемые образы ИР и операции над ними в вызовы унифицированных процедур с атрибутами. Например, если на экране с помощью интерфейсного модуля выбрана некоторая таблица и выделена первая свободная строка, а затем в контекстном меню выбрана команда «Вставить строку», то на уровне модели данных активизируется функция Insert (Array, i), которая реализует вставку строки в физической памяти машины.

Абстрагирование данных.

Для процесса отображения между низкоуровневым представлением и моделью данных используется термин «абстракция», характеризующий различие между внешними свойствами данных, отраженными в модели данных, и деталями низкоуровневого представления.

Таким образом, для манипулирования низкоуровневыми данными в терминах модели данных используются высокоуровневые обозначения типовых процедур доступа, которые при выполнении реализуют детальные операции над низкоуровневыми данными.

Введение промежуточного слоя между низкоуровневыми данными и моделью данных, составленного из набора унифицированных процедур, позволяет ограждать пользователя или программиста от деталей фактической организации низкоуровневых данных, т.е. абстрагироваться от этих деталей, чтобы он мог обращаться с данными так, как если бы они были организованы в наиболее удобной для него форме.

В ФС данный слой представлен драйверами устройств внешней памяти, в СУБД — набором процедур, реализующих операторы SELECT, INSERT, UPDATE, DELETE, .., а при федеративном объединении данных — набором программных средств, выполняющих реальный доступ к гетерогенным ИР.

В дальнейшем этот слой будем называть адаптером доступа, представленным набором базовых сервисов, которые воспринимают высокоуровневые конструкции модели данных в качестве входных данных и в ходе выполнения

реализуют детальные операции над низкоуровневыми данными.

Унифицированный интерфейс.

Процесс отображения между концептуальным представлением и моделью данных по существу реализует унифицированный интерфейс, так как при этом обеспечивается доступ к низкоуровневым гетерогенным источникам данных через общую единообразную среду взаимодействия. Как запросы, так и ответы на них при этом должны формулироваться в абстрактных терминах модели данных.

Введение промежуточного уровня между концептуальным представлением и моделью данных обеспечивает доступ к унифицированным процедурам путем манипуляции чувственно воспринимаемыми образами модели данных.

Итак, мы выявили механизм интеграции данных, реализуемый путем:

- разнопланового представления данных в виде низкоуровневых (физических) данных, модели этих данных и концептуальных образов элементов модели данных;
- абстракции низкоуровневых данных с помощью модели данных;
- реализации единой среды доступа к гетерогенным источниками низкоуровневых данных с помощью концептуальных образов элементов модели данных.

4 Формальная модель процесса интеграции

Выше было отмечено: полному решению интеграции данных, используемых в современных АСУ, мешает наличие большого объема неформализованных ИР. Поэтому было предложено разработать модель данных, которая обеспечивала бы формализацию более широкого спектра ИР и включала перечисленные форматы неформализованных ИР в качестве составных элементов.

Но прежде чем приступить к выполнению этой задачи, необходимо решить важную проблему: формализовать процесс интеграции, т.е. перевести вербальное описание механизмов интеграции на язык формальных конструкций с тем, чтобы реализовать их впоследствии в виде структур и алгоритмов обработки данных.

С этой целью предложено формальное описание:

- представления данных на разных уровнях;
- процесса абстракции данных в виде отображений множеств источников данных на множество метаданных;
- процесса доступа к данным в виде одного единственного отображения множества концептуальных образов на множество метаданных.

4.1 Гетерогенные низкоуровневые данные

Формально гетерогенные источники низкоуровневых данных можно представить как совокупность множеств:

$$D = (D_1, ..., D_i, ..., D_N),$$

где D - множество всех типов ИР системы;

$D_i = \{d^i_1, ..., d^i_{j(i)}, ..., d^i_{n(i)}\}$ - множество ИР i-го типа;

$d^i_{j(i)}$ - j-й ИР в множестве D_i;

$n(i)$ - число ИР в множестве D_i.

Мощность множества D определяется как сумма

$$V = n(1) + ... + n(i) + .. + n(N).$$

ИР $d^i_{j(i)}$ каждого типа D_i описываются специфическим набором характеристик. Специфичность описания заключается в том, что различаются как число, так и состав характеристик ИР различных типов, как это показано в выражении:

$$\pi^i : \pi^i_1, ..., \pi^i_{\mu(i)}, ..., \pi^i_{m(i)},$$

где $\pi^i_{\mu(i)}$ - $\mu(i)$-й параметр, описывающий ИР $d^i_{j(i)}$;

$m(i)$ - число параметров, описывающих ИР множества D_i.

Кроме этого, для каждого множества D_i определяется набор операций:

$$\omega^i : \omega^i_1, ..., \omega^i_{\sigma(i)}, ..., \omega^i_{s(i)},$$

где $\omega^i_{\sigma(i)}$ - $\sigma(i)$-я операция над ИР множества D_i;

$s(i)$ - число операций над ИР множества D_i.

Таким образом, ИР $d^i_{j(i)}$ каждого типа D_i описываются совокупностью характеристик π^i и операций ω^i, т.е. $d^i_{j(i)} : d^i_{j(i)}(\pi^i, \omega^i)$.

Следует также отметить, что совокупность низкоуровневых операций для всех множеств D_i составляет множество $\omega = \omega_1, ..., \omega_S$, где $S = s(1) + ... + s(i) + ... + s(N)$.

4.2 Логическая модель данных

Основная цель разработки модели данных — обеспечение унифицированного описания гетерогенных низкоуровневых источников данных. При этом формируется множество M унифицированных описаний, мощность которого соответствует мощности множества D:

$$M = \{m_1, ..., m_v, ..., m_V\},$$

где m_v – v -й элемент множества M, представляющий собой унифицированное описание элементов множеств $D_1, ..., D_i, ..., D_N$.

Каждый элемент m_v множества M представлен унифицированным набором параметров $\tilde{\pi} : \tilde{\pi}_1, ..., \tilde{\pi}_p$ для описания данных множеств $D_1, ..., D_i, ..., D_N$.

Для элементов m_v определен набор описаний операций $\tilde{\omega} : \tilde{\omega}_1, ..., \tilde{\omega}_T$, обеспечивающий доступ к низкоуровневым операциям множества Ω. Необходимо отметить, что число операций T на уровне модели данных существенно меньше числа S низкоуровневых операций. Объясняется это тем, что одноименные операции для различных типов внешней памяти, такие, как «записать на магнитный барабан», «записать на магнитный диск» и т.д., на уровне модели данных представлены одной операцией «записать во внешнюю память» с указанием типа внешней памяти.

Кроме этого, каждый элемент $m_v \in M$ должен содержать ссылку $p_i^v : m_v \leftrightarrow d^i_{j(i)}$, связывающую данный элемент с описываемым ИР $d^i_{j(i)} \in D_i$.

Таким образом, элементы $m_v \in M$ представляются как

$$m_v : m_v(\tilde{\pi}_1, ..., \tilde{\pi}_p; \tilde{\omega}_1, ..., \tilde{\omega}_T; p_i^v) = m_v(\tilde{\pi}, \tilde{\omega}, p_i^v)$$

и содержат унифицированное описание элементов всех множеств $D_1, ..., D_i, ..., D_N$.

Следует иметь в виду, что в множестве M отображаются не сами ИР из множеств D_i, а их описания, сведенные к определенному стандарту. Другими словами, в множестве M представлены данные о данных или метаданные.

4.3 Концептуальное представление модели данных

Основная цель интеграции - создание единого высокоуровневого интерфейса пользователя, представляющего общую единообразную среду взаимодействия для различного оборудования, программных средств и ИР.

Поэтому требуется сформировать еще один уровень представления, выражающий модель данных через чувственно воспринимаемые элементы.

Эти элементы образуют множество

$$C = \{c_1, ..., c_v, ..., c_V\},$$

мощность которого равна мощности множества M. Каждому элементу множества C сопоставляются наименования характеристик и операций множества M:

$$c_v : c_v(\overline{\pi}_1, ..., \overline{\pi}_p; \overline{\omega}_1, ..., \overline{\omega}_T) = c_v(\overline{\pi}, \overline{\omega}),$$

где $\overline{\pi} : \overline{\pi}_1, ..., \overline{\pi}_p$ и $\overline{\omega} : \overline{\omega}_1, ..., \overline{\omega}_T$ - унифицированные наборы характеристик и операций модели данных, представленные в чувственно воспринимаемом виде.

4.4 Абстрагирование низкоуровневых данных как процесс отображения низкоуровневых данных на логическую модель

Выше процесс отображения между низкоуровневым представлением и моделью данных был обозначен термином «абстракция», характеризующим различие между унифицированными внешними свойствами данных $m_v(\tilde{\pi}, \tilde{\omega}, p_i^v)$, отраженными в модели данных, и деталями низкоуровневого представления $d^i_{j(i)}(\pi^i, \omega^i)$.

Для связи этих двух уровней представления данных используются адаптеры доступа к информационным ресурсам. При этом адаптеры доступа воспринимают высокоуровневые конструкции модели данных в качестве входных данных и реализуют детальные операции над низкоуровневыми данными.

Формально эту связь можно описать как изоморфное отображение множеств M и D, представляемое выражением

$$T_i^M : m_v(\tilde{\pi}, \tilde{\omega}, p_i^v) \leftrightarrow d^i_{j(i)}(\pi^i, \omega^i),$$

где $i=1, ..., N$,

$$j(i) = 1, ..., n(i),$$

$$V = n(1) + ... + n(i) + .. + n(N),$$

$$v = 1, ..., V.$$

Таким образом, мы получаем совокупность отображений (адаптеров доступа) $T_1^M, ..., T_i^M, ..., T_N^M$, реализующих операции ω^i над характеристиками π^i низкоуровневых данных $d^i_{j(i)}$. В качестве входных параметров для этих отображений выступают высокоуровневые атрибуты $\tilde{\pi}, \tilde{\omega}, p_i^v$ модели данных, а результатом выполнения – реализация операций ω^i

по доступу к характеристикам $\pi^i_1,...,\pi^i_{m(i)}$ ИР $d^i_{j(i)}$ из множества D_i.

4.5 Унифицированный интерфейс как процесс отображения между логической моделью и концептуальным представлением

Нам осталось связать концептуальное представление с моделью данных. Формально связь между концептуальным представлением и моделью данных описывается с помощью отображения:

$$T^C_M : c_v(\overline{\pi},\overline{\omega}) \leftrightarrow m_v(\tilde{\pi},\tilde{\omega}, p^v_i),$$

где $v=1, ..., V$.

Как следует из вышеприведенного выражения, связь между концептуальным представлением и моделью данных реализуется с помощью одного единственного отображения T^C_M. Данное отображение по существу представляет собой интерфейсный модуль, обеспечивающий доступ к данным и операциям над ними в терминах модели данных.

4.6 Доступ к гетерогенным низкоуровневым источникам данных посредством единого интерфейса в рамках единой модели данных

Схема доступа к гетерогенным низкоуровневым данным через унифицированный интерфейс в рамках единой модели данных с использованием описанной формальной модели приведена на рис. 7.

Манипуляции низкоуровневыми данными с применением высокоуровневого пользовательского интерфейса, описываемого отображением T^C_M, реализуется следующим образом.

1. Пользователь выбирает концептуальный образ c_v ИР из множества C, выбирает операцию $\overline{\omega}$ и инициирует отображение T^C_M.
2. Данное отображение определяет элемент m_v из множества M, ставит в соответствие образу операции $\overline{\omega}$ операцию $\tilde{\omega}$ в терминах модели данных и активизирует ссылку p^v_i, определяющую набор параметров низкоуровневого представления данных в множестве D.
3. Это, в свою очередь, активизирует отображение T^M_i, реализующее низкоуровневые операции над ИР $d^i_{j(i)}$ на уровне физической памяти машины.

В результате выполнения описанных шагов инициируется цепочка преобразований:

$$c_v(\overline{\pi},\overline{\omega}) \leftrightarrow m_v(\tilde{\pi},\tilde{\omega}, p^v_i) \leftrightarrow d^i_{j(i)}(\pi^i,\omega^i).$$

На уровне модели данных семейство отображений T^M_i $(i=1,...,N)$ заменяется одним отображением T^C_M, и именно с этим отображением имеет дело пользователь при работе с программным приложением.

Этот факт отражает основное преимущество, которое мы получаем в результате интеграции гетерогенных данных. Стандартизация статических характеристик данных с помощью модели данных определяет в описании фиксированные позиции, которые можно легко интерпретировать, приписав им адаптеры доступа (базовые сервисы), использующие эти значения должным образом.

Это придает метаданным свойство внутренней интерпретируемости, что существенно снижает объем программного кода при разработке прикладных программ.

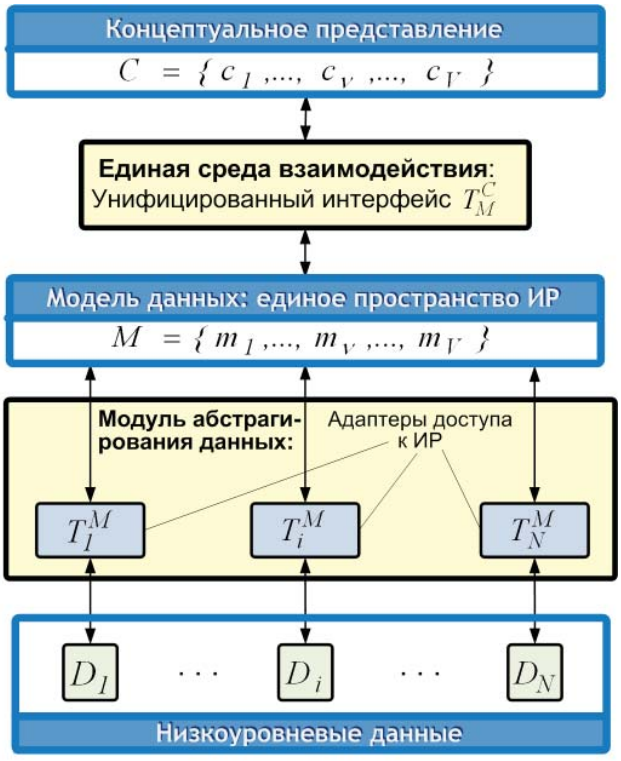


Рис. 7. Формальная модель схемы доступа к низкоуровневым данным через унифицированный интерфейс

Заключение

В статье рассмотрена проблематика совместимости ИР при создании интегрированных АСУ. На основе анализа существующих подходов объединения гетерогенных АСУ для совместного решения управленческих и технологических задач выявлены их недостатки и поставлена задача разработки формализованной

модели процесса интеграции гетерогенных ИР.

С этой целью проведен анализ технических решений, связанных с интеграцией данных на разных этапах развития информационных технологий. В результате этого анализа выявлен механизм интеграции и определены его компоненты.

В последнем разделе статьи предложена формальная модель описанного механизма интеграции данных, которую можно использовать при разработке унифицированных средств взаимодействия АСУ интегрированных систем.

СПИСОК ЛИТЕРАТУРЫ

1. Единое информационно-функциональное пространство ВМФ: от идеи до реализации / под общ. ред. В.И. Кидалова. СПб.: Ника, 2003.

2. Дейел Г. Введение в операционные системы: В 2 т. Т.1 / Пер. с англ. — М.: Мир, 1987. — 480 с.

3. Л. Черняк. Когда СУБД не были реляционными // Computerworld. 2006. №27. С. 44 — 45.

4. Управление контентом предприятия от IBM // Инновации в технологиях и бизнесе. — 2008. — №1. - С. 11 — 13.