УЧЕНЫЕ ЗАПИСКИ ВЫСШИХ УЧЕБНЫХ ЗАВЕДЕНИЙ ЛИТОВСКОЙ ССР
ЭКОНОМИКА, 27, 1984

ТИПИЗАЦИЯ КОМПОНЕНТ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ И ПРЕДПОСЫЛКИ СОЗДАНИЯ АВТОМАТИЗИРОВАННЫХ БАНКОВ — АЛГОРИТМОВ И ПРОГРАММ

З. БРАЗАЙТИС

Организация данных и их использование в системах машинной обработки (СМО) претерпели значительные изменения с появлением баз и банков данных. До этого, как известно, различные программисты поразному представляли одни и те же данные и должны были их каким-то образом модифицировать практически при каждом изменении и расширении круга решаемых задач. Иначе говоря, из-за жесткой зависимости данных и программ отсутствовала гибкость их использования, что требовало значительных затрат на переработку существующих программ и реорганизацию данных. Концепцией баз данных в качестве одного из основных принципов предусматривается независимость данных от программ. Его реализация связана с разделением данных на логическом и физическом уровнях и созданием специальных программных средств привязки физических данных к программам их обработки на логическом уровне. Одновременно в базах данных должна быть обеспечена возможность обращения к данным на языках высокого уровня, позволяющих пользователям формулировать цели обращения в привычных для них терминах конкретной предметной области. Построение структур данных различных уровней и процедур их отображения (перехода с одного уровня в другой) обеспечивается системой управления базой данных (СУБД), которую можно считать программной надстройкой базы данных. Таким образом, программное обеспечение (ПО) современных СМО функционально можно разделить на три компоненты, представленные на рис. 1 в их связи с различными уровнями представления данных согласно (1, с. 75).

Состав каждой компоненты неоднороден. Прикладные программы и средства их генерации, например, включают в себя модули анализа (синтаксического, лексического и семантического) обращений пользователей и формирования стандартизованных запросов, на основе которых могут быть сгенерированы, настроены или загружены для выполнения рабочие программы, реализующие процедуры поиска входных данных, их преобразования, формирования и вывода результатов. С точки зрения независимости прикладных программ от данных главным здесь является построение логической структуры данных по их описанию на языке обращения. Для этого используются специальные тезаурусы и идентификаторы логических элементов базы данных. Характерной особенностью описания данных в прикладных программах является его независимость от языка программирования (в смысле его содержания, а не формы) и конкретной физической ЭВМ, ее конфигурации. К данной компоненте ПО также относятся средства используемой операционной системы ЭВМ, которые организуют трансляцию, редактирование и управление выполнением программных модулей. Сюда же включаются программные средства обеспечения достоверности данных, надежности функционирования системы машинной обработки данных.

Системные программы логической организации данных обеспечивают согласование логической структуры данных, представляемой в прикладных программах, с глобальной структурой базы данных, в которой заложена общая модель организации данных и по которой определяется конкретная схема формирования требуемых логических записей согласно логическому представлению их элементов в базе данных. Они выполняют роль промежуточного звена перехода от логического уровня

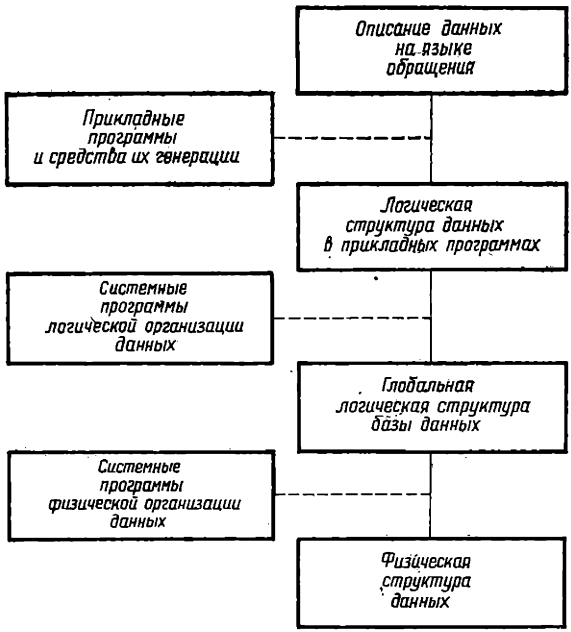

Рис. 1. Компоненты программного обеспечения и уровни представления данных

представления данных пользователям системы к их физическому отображению в базе данных. В некоторых СУБД автономное звено может отсутствовать, если логическая и физическая структуры данных представлены одной общей схемой, что характерно для сравнительно простых баз данных, использующих какой-либо один способ их организации (иерархический, сетевой, реляционный и т. д.). В таких случаях выделенные нами вторая и третья компоненты ПО представляют собой единое целое, оперирующее лишь двумя отображениями структуры данных.

Вторая компонента является чисто внутренним программным обеспечением управления базой данных. Она построена на принятых принципах логической организации базы данных и не зависит ни от функциональных обращений пользователей (т. е. решаемых в системе задач), ни от физической (аппаратурной) реализации СМО. Такие системные программы недоступны не только пользователям базы данных, но и прикладным программистам. Их создание и сопровождение целиком ложатся на системных программистов и администраторов баз данных.

Системные программы физической организации данных осуществляют непосредственный поиск данных в физической базе (в памяти системы) и их передачу прикладным программам согласно описанию на логическом уровне. Именно они обеспечивают достижение конечной цели обращения к базе данных. Опираясь на определенную схему логического представления данных, эти программы делают доступными требуемые элементы базы данных, физически расположенные в определенных файлах, хранящихся во внешней памяти конкретной ЭВМ. В связи с этим к третьей компоненте ПО относятся не только программные средства управления переходом от логического уровня представления данных к физическому, но и модули используемой операционной системы ЭВМ, связанные с реализацией процедур ввода—вывода данных на различных машинных носителях в условиях применения различных методов и способов их размещения, перекомпоновки элементов и переблокирования записей файлов данных.

Все вышесказанное целиком и полностью относится лишь к использованию баз данных, причем переходы с одного уровня представления данных на другой в основном рассмотрены нами только в одном направлении — от пользователей к физическим данным. Обратное отражение данных обеспечивается теми же компонентами ПО СМО. Из изложенного можно сделать вывод, что программная надстройка баз данных является сложной системой, а точки входа пользователей в нее в основном сосредоточены в первой компоненте, сложность (или простота) построения которой главным образом зависит от возможностей двух остальных компонент. Последние же в решающей степени определяются глубиной разрыва между логическим и физическим уровнями представления данных. Иными словами, чем больше такой разрыв, тем более громоздкой становится надстройка базы данных. В связи с этим нельзя не заметить, что с развитием концепций баз данных на современном этапе проявляется тенденция углубления такого разрыва, главным образом, из-за желания создать беспредельно универсальные базы данных (2, с. 173—174). Поэтому возникает опасность чрезмерного усложнения СУБД, вообще-то призванных упростить организацию и использование данных, и появления «интеллектуального» несоответствия между целями конкретных СМО и средствами их достижения, что может привести к нерациональному и неэффективному использованию ресурсов ЭВМ. Проблема становится еще острее, если учесть сложности, связанные с организацией (формированием и ведением) баз данных, т. е. теми вопросами, которые в данной статье нами преднамеренно не рассматривались.

Для преодоления возникающих противоречий между логической и физической структурой баз данных наиболее перспективным, на наш взгляд, является все более широкое использование семантики данных при их описании на логическом уровне. Реализация такого направления связана с разработкой типовых семантических структур и типовых процедур обращения к ним, ориентированных на определенную предметную область. При этом типовость составляющих ПО СМО как результат типизации в разрезе выделенных нами трех компонент должна сочетаться с разумной универсальностью, достигаемой за счет некоторой функциональной избыточности, с управляемостью по принципу самоконтроля (4, с. 83—90) и простотой общения пользователей с базой данных. Удачное сочетание этих свойств представляется надежной предпосылкой эффективного функционирования современных СМО.

Рассмотрим возможный подход к типизации семантических структур. Каждый файл в базе данных может содержать не только данные, но и некоторую адресную информацию, с использованием которой может быть реализована контекстная защита данных. Так, например, специальные средства контроля позволяют строго соблюдать семантически

правильные действия с адресной информацией, представляющей собой межфайловые ссылки. Все ссылки целесообразно сосредоточить в одном общем файле (файле заголовков), а в заголовке каждого файла данных записать межфайловые ссылки, каждая из которых указывает относительный адрес заголовка файла в файле заголовков. Таким образом, любой объект в базе данных может быть достигнут только через ссылки. Однако обращение к данным производится по их идентификаторам. В связи с этим в базу данных необходимо включить контекстные справочники, которые могут быть объединены в один общий файл. Они представляют собой иерархические индексные таблицы с переменным числом уровней. Каждый справочник содержит строки, состоящие из двух элементов: идентификатора объекта и межфайловой ссылки на файл или другой справочник. Кроме того, в справочниках можно хранить информацию, характеризующую конкретный доступ к определенному объекту базы данных через определенный вход, т. е. для пользователей с различными входами один и тот же объект контекстно может представляться по-разному. Если в обращении пользователя встречается составной идентификатор или какой-либо идентификатор в первом справочнике не найден, просматриваются косвенные контексты по ссылкам на другие справочники (этот процесс может носить рекурсивный характер). Кроме рассмотренных, в базу данных целесообразно включить файл пользователей, определяющий их файловый контекст, пароли обращения, способы проверки доступа к данным и пр.

К основным внутренним процедурам (операциям) над файлами рассмотренной структуры можно отнести их генерацию, уничтожение, открытие и закрытие, считывание и модификацию данных, а также различные виды обмена (непосредственный между данными файлов и массивами пользователей, буферный и др.). Модули их реализации расширяют возможности существующих операционных систем ЭВМ.

Типизация отдельных составляющих компонент ПО СМО требует тщательного анализа алгоритмов и процедур обработки данных. Для прикладных программ, реализующих обращения пользователей к базе данных, целесообразно выделить ведущую и исполнительные процедуры. Ведущая процедура должна обеспечить вызов исполнительных в определенной последовательности, зависящей от содержания конкретного обращения. Исполнительные процедуры функционально можно разбить на следующие группы: информационный поиск в базе данных, адаптация поиска и подключение оперативных данных; обновление данных и его адаптация; обработка данных и адаптация обработки, связанная с изменениями в предметной области; редактирование и корректировка результатов (ответов на запросы); адаптация корректировки результатов; прагматические процедуры, обеспечивающие определенные удобства для пользователей.

Особое внимание следует уделить процедурам обработки данных и ее адаптации. Наличие множества возможных вариантов построения различных процедур, их пересечений требует формализованного описания алгоритмов, которое позволило бы проанализировать их семантику с целью выделения в них эквивалентных схем, удобных для математического исследования. В качестве инструмента построения и анализа семантических моделей программ обработки данных можно использовать теорию графов (3). Это позволяет автоматизировать процесс типизации программных модулей с помощью ЭВМ.

Вопросы типизации ПО тесно связаны с внедрением готовых пакетов прикладных программ (ППП). Использование ППП при проектировании СМО дает возможность снизить затраты, сократить сроки и повысить качество работ. Исходным пунктом при выборе тех или иных готовых программных средств должны быть экономическая эффективность их применения, определение путей реализации экономического потен-

циала ППП. При этом следует учитывать затраты на их приобретение, поддержание в работоспособном состоянии (доработку, модернизацию), сопровождение и внедрение, а также разработку межпакетного интерфейса.

Рассмотренный нами подход к типизации компонент программного обеспечения СМО создает предпосылки для построения автоматизированных банков алгоритмов и программ (АБАП). Целью создания такого банка для конкретной СМО является автоматизация процессов разработки и эксплуатации прикладных программ. АБАП представляет собой набор системных средств обработки данных, обеспечивающий формирование, хранение, корректировку и интегрированное использование прикладных программ и алгоритмов, а также их составляющих (модулей). Рассмотрим некоторые проблемы, возникающие в связи с организацией АБАП.

Прежде всего уточним понятия программы и алгоритма. Программа — это синтаксический ориентированный граф, представленный в виде конструкций машинного языка, который может быть непосредственно реализован на ЭВМ путем выполнения действий согласно семантике отдельных элементов (операторов). Формализованное описание программы не на машинном языке представляет собой ее алгоритм. Таким образом, он является моделью программы и отличается от нее своим базисом (алфавитом). Так как практически невозможно разработать программу без ее алгоритма, мы сталкиваемся с вопросом моделирования ПО. С другой стороны, модели нужны для исследования системы ПО, когда они используются в качестве инструмента для рационализации ее структуры, определения функциональной эквивалентности отдельных составляющих, их формальных связей и т. п. Для организации АБАП в процессе моделирования основная роль отводится структуре программ, а не средствам их формализации и технологии конструирования; в данном случае модели должны быть независимыми от используемых систем программирования и механизма генерации рабочих программ.

В условиях использования современных операционных систем ЭВМ данные в программах описываются, как уже отмечалось, на логическом уровне, а физическое обращение к ним обеспечивают системные средства ввода—вывода, для которых определенные параметры передаются не из программы. В связи с этим чисто машинный алгоритм, состоящий из спецификаций данных и процедур их обработки (преобразования), всегда необходимо дополнить блоком параметризации ввода—вывода на физическом (машинном) уровне с тем, чтобы он стал программой согласно нашему определению. Таким образом, алгоритм, или модель программы, содержит три компоненты: логическое описание данных (ЛОД), процедуры ввода—вывода (ПВВ) и процедуры преобразования данных (ППД). Такое представление алгоритмов позволяет компилировать программы сложной структуры с использованием различных комбинаций перечисленных компонент.

При моделировании программ особое внимание следует уделить ППД. Прежде всего необходимо провести их функциональную дифференциацию. Функцию процедуры или программы можно интерпретировать как логический аналог какой-либо математической функции или вычисления определенного значения. Здесь очень важно построить «чистые» функции, не имеющие посторонних эффектов. Конкретные же процедуры могут компилироваться из отдельных функций с добавлением таких эффектов, как, например, средств информационной связи. При разработке АБАП возникает еще один важный вопрос моделирования программ, связанный с организацией параллельных вычислений. Мы имеем в виду одновременное выполнение двух или нескольких процедур одной и той же программы на нескольких процессорах ЭВМ или на одном процессоре в режиме прерывания. В связи с этим в некоторых алгоритмах появ-

ляется четвертая компонента — описание параллельных вычислений (ОПВ), на основе которого строится механизм динамического распределения ресурсов ЭВМ и вызова процедур для их выполнения. Здесь должны быть предусмотрены особые ситуации, при которых без определенной реакции пользователя выполнение программы по предусмотренной схеме не может быть продолжено.

Таким образом, в общем случае модель любой программы можно представить в следующем виде (компоненты ПВВ и ОПВ могут отсутствовать):

$$М = \{ЛОД\ [, ПВВ\ ], ППД\ [, ОПВ]\}.$$

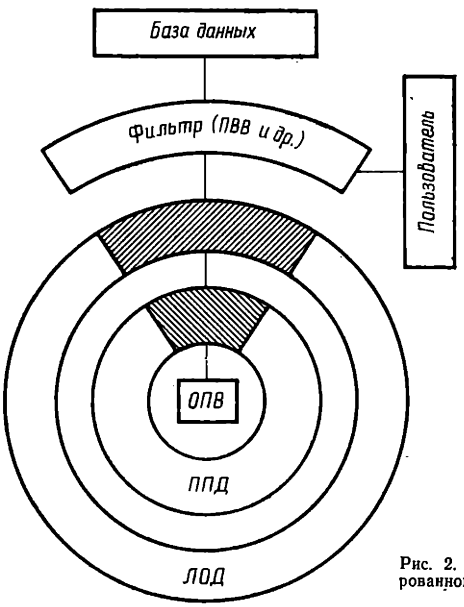

Рис. 2. Структура автоматизированного банка алгоритмов и программ

Фонд АБАП можно организовать по такому принципу: все компоненты описания данных и отдельно процедурные компоненты объединить в кольцо таким образом, чтобы соседние компоненты имели точки сопряжения, т. е. некоторые общие элементы. При таком построении фонда, которое может быть выполнено на основе типизации компонент, для любой задачи обработки данных можно определить структурный спектр компонент, при этом для генерации конкретной программы оба кольца должны занимать такое положение, чтобы их спектры оказались рядом. Это должна обеспечить система управления автоматизированным банком алгоритмов и программ. Схема такого АБАП представлена на рис. 2. Внутреннее кольцо объединяет процедурные компоненты, а внешнее — описательные. Заштрихованные участки представляют собой спектры конкретной задачи. В блок фильтра, обеспечивающего связь программы с базой данных и пользователем, включаются ПВВ с необходимыми параметрами физического обращения к данным.

В заключение необходимо отметить, что спектральное построение прикладных программ в условиях АБАП позволяет автоматически ком-

плектовать задачи обработки данных, определять последовательность их решения, заполнять пробелы в кольцах при постановке таких задач, для которых не могут быть определены спектры по имеющимся описательным и процедурным компонентам, включенным в автоматизированный банк алгоритмов и программ.

Вильнюсский университет
им. В. Капсукаса
Кафедра экономической
информации

Редколлегии вручено
в октябре 1982 г.

ЛИТЕРАТУРА

1. Мартин Дж. Организация баз данных в вычислительных системах. 2-е изд., доп.— М.: Наука, 1980.
2. Модели данных и системы баз данных.— В кн.: Труды совместного советско-американского семинара, сост. 14—23 ноября 1977. М.: Наука, 1979.
3. Подловченко Р. И. Иерархия моделей программ.— Программирование, 1981, № 2.
4. Шураков В. В. Надежность программного обеспечения систем обработки данных.— М.: Статистика, 1981.