УДК 510 876. 5

О ТЕХНОЛОГИИ РАЗРАБОТКИ БАЗ ДАННЫХ НА ОСНОВЕ ИНСТРУМЕНТАЛЬНОЙ СИСТЕМЫ MISS

Ю. И. Бродский, В. Ю. .Лебедев

Введение

В настоящей статье предлагаются общие приемы и некоторые частные технологические аспекты программирования на IBM-совместимых ГОШ объемных баз данных (БД) с использованием инструментальной среды системы MISS [БродскийЮ.И, Лебедев В.Ю. Инструментальная система имитации MISS. М.: ВЦ РАН, 1991- 179 с. 3.] Эта система создавалась для разработчиков сложных имитационных моделей, и поскольку грамотная организация данных таких моделей - одна из основных проблем их машинной реализации, в MISS были включены достаточно мощные средства поддержки данных. Конкретнее говоря, MISS предлагает набор процедур виртуализации памяти, составляющих полную элементную основу для программирования иерархических и сетевых БД. К настоящему времени накоплен существенный положительный опыт разработки различных БД (ив связи с имитацией и безотносительно ее) с использованием этих средств, и здесь мы хотели бы поделиться им.

На сегодняшний день существует немало высококачественных коммерческих СУБД, и если говорить о разработке небольших БД "конторской ориентации", то надо, конечно, пользоваться именно этими средствами: так будет и проще и надежней. Однако, когда *нужно* запрограммировать БД с мегабайтами полезной информации и сложными процедурами обработки, стандартные СУБД типа CLIPPER оказываются удручающе неэффективными. Чаще всего в таких случаях берут какие-то элементы коммерческих СУБД, а что-то дописывают сами, как правило - на языке "С". В этой струе лежит и технология программирования БД на основе элементной базы системы виртуализации памяти MISS.

1. Макроструктура программ

Программный комплекс реализации БД в рамках предлагаемой технологии распадается на три больших блока: РЕДАКТОР, КОНВЕРТЕР и ПРОЦЕССОР. Назначение РЕДАКТОРА - обеспечить возможности наполнения БД информацией и модификации ранее заведенных данных. Выход РЕДАКТОРА служит входом КОНВЕРТЕРА, а сам КОНВЕРТЕР нужен для преобразования данных в формат, которым оперирует ПРОЦЕССОР. Содержание ПРОЦЕССОРА - средства для просмотра содержимого базы, осуществления выборок и запуска расчетных задач.

Всего предусмотрены три формата хранения данных - по одному на каждый принципиальный блок системы. Для ПРОЦЕССОРА данные предлагается оформлять в виде совокупностей списковых и ссылочных структур, реализуемых инструментальными средства-

ми MISS. «Формат КОНВЕРТЕРА - стандартные DBF-файлы. Форматом РЕДАКТОРА может быть либо первое, либо второе, в зависимости от того программируется ли РЕДАКТОР средствами MISS, или - средствами стандартной СУБД, Следует подчеркнуть, что далее если РЕДАКТОР будет использовать средства MISS, конкретные форматы данных в нем будут совсем не те, что у ПРОЦЕССОРА, поскольку такой организации данных, которая была бы удобна одновременно и для их обработки и для их редактирования, не существует.

Из трех, упомянутых выше хранилищ данных, два являются "постоянными", т. е . - сохраняемыми между сеансами работы с БД, а одно - временным, существующем только во время работы РЕДАКТОРА. Временное, внутреннее хранилище данных РЕДАКТОРА генерируется во время его запуска; это делает "обратный конвертер" - процедура, читающая информацию текущей версии базы из ее DBF-представления (т.е. - из соответствующих DBF-файлов) и формирующая списковое сетевое представление базы, удобное для редактирования. Конвертирование базы во внутренний формат редактора, в частности, предполагает лексикографические упорядочения ряда списковых блоков по содержащимся в записях именам.

Процедуры собственно редактирования должны позволять, во-первых, заводить в базу новые записи, а во-вторых, редактировать или исключать старые, обеспечивая возможности быстрого поиска записей по ключевым именам. При завершении сеанса работы с РЕДАКТОРОМ будет вызываться процедура "прямо-

го конвертирования", которая преобразует обновленную версию базы из внутреннего представления РЕДАКТОРА в набор новых DBF-файлов, после чего внутреннее представление уничтожается.

В предлагаемой технологии DBF-формат данных по существу является промежуточным и, вообще говоря, без него можно обойтись, обеспечив прямое генерирование сохраняемой рабочей версии базы из ее внутреннего представления в РЕДАКТОРЕ и обратное конвертирование. Однако, промежуточное представление целесообразно сохранить по трем причинам:

а) ради полного распараллеливания разработки РЕДАКТОРА и ПРОЦЕССОРА;

б) для упрощения контроля корректности данных на этапе отладки программ системы;

в) для предоставления возможности работать с базой через коммерческие СУБД типа CLIPPER, FOXPRO и т. д,

Из трех составляющих системы КОНВЕРТЕР - самая неприятная для программирования- Обычно КОНВЕРТЕР довольно громоздкая программа- Разбирая содержимое DBF-файлов базы и порождая списковое представление данных, она должна оптимизировать использование памяти и обеспечивать (блокированием списков) возможность эффективного доступа к данным- КОНВЕРТЕР - это единственная недиалоговая программа системы: её работа не предполагает вмешательства оператора, поскольку и входы и выходы в данном случае предопределены-

2. Приемы организации данных

В состав инструментального набора средств виртуализации памяти MISS входят процедуры для формирования и редактирования таких структур данных как виртуальные (раэмещаемые в виртуализированной памяти)

а) одиночные записи,

б) списки,

в) массивы.

При построении РЕДАКТОРА БД полезными, возможно, окажутся все эти типы данных- Однако для ПРОЦЕССОРА, к которому предъявляются жесткие требования по эффективности доступа к данным и компактности форматов их хранения, подойдут только одиночные записи и списки: работа с элементами массивов организована в MISS не слишком эффективно.

Обычно содержимое БД распадается на несколько блоков, каждый из которых по сути является совокупностью записей относящихся к некоторому набору объектов. В реляционных базах форматом представления таких блоков являются отношения (таблицы). Если же использовать списковые представления, а мы предлагаем, именно это, совокупность однородных по структуре записей можно задать

а) как единый список с элементами типа записи нужной структуры;

б) как набор параллельных списков, формируемых по принципу: "одно поле - один список".

Эффективным является только второй способ представления дан-

ных, при котором такие операции над базой как отбор объектов по значениям ключевых полей относящихся к ним записей и визуализация частичной информации по объектам требует привлечения (подкачки в рабочую область виртуальной памяти) лишь нужных данных.

Организуя параллельные списки их элементами, можно сделать значения отдельных полей соответствующих записей, и это - прямой подход. Однако, лучше прибегать к блокировке и объединять в одном элементе сразу несколько (одно и то же число для всех параллельных списков) значений одного поля. Второй подход несколько осложняет программирование доступа к полям, но зато эффективность этого доступа резко увеличивается, так как снижаются накладные расходы на поиск нужного элемента списка. В связи с блокированием значений полей в параллельных списках возникает вопрос о разумных размерах блоков. По соображениям простоты (а соответственно и - эффективности) внутренней структуры формируемой MISS виртуальной памяти лучше, чтобы было поменьше разнобоя в длинах элементов списков. Скажем, в разработанных по рассматриваемой технологии БД данные для всех параллельных списков группировались в блоки длиной в 1Кб.

Типичной разновидностью данных в самых разных по назначению БД являются имена объектов, причем имеются ввиду их естественные и переменные по длине имена, а не какие-то кодовые сокращения с фиксированным числом букв. Эффективным способом хранения таких имен в БД является пара (сблокирован-

ных) списков- В один список заносятся сами имена (разделенные в блоках нулевым байтом), а другой составляется из трехбайтовых вспомогательных записей, имеющих смысл "адресов" имен в первом списке; в первый байт k-ой записи записывается номер того блока первого списка, в котором содержгся k-ое имя, а два вторых байта определяют номер начальной позиции k-ого имени в блоке. Этот же способ организации данных успешно применялся и в случаях, когда в информации, относящейся к некоторому набору объектов, имеются существенные "пробелы", т- е. для значительной части объектов какие-то данные отсутствуют. Тогда в первый список заносятся существующие данные, а во второй их "адреса" в первом списке, причем некий специфический адрес, например, целиком состоящий из битовых единиц, считается признаком отсутствия информации по объекту.

Весьма полезными при работе с содержимым БД являются вспомогательные данные типа битовых масок для наборов объектов. Позиции битовой маски набора используются для выделения в нем неких подмножеств, например - в результате осуществления выборки по какому-то признаку, С одной стороны, хранение битовых масок не требует много памяти, а с другой - позволяет очень эффективно выполнять операции алгебры множеств над подборами одного набора объектов а также работать с подмножествами данных базы.