К. С. Зайцев, С. С. Ковалевский, А. Н. Малярский, Система управления базами данных HyTech и ее использование в АСУ, *Автомат. и телемех.*, 1993, выпуск 11, 161–167

УДК 519.68:519.256

© 1993 г. К.С. ЗАЙЦЕВ, канд. техн. наук,
С.С. КОВАЛЕВСКИЙ, канд. техн. наук,
А.Н. МАЛЯРСКИЙ
(Московский инженерно-физический институт)

СИСТЕМА УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ HyTech и ЕЕ ИСПОЛЬЗОВАНИЕ В АСУ

В данной статье проводится анализ эффективности СУБД для персональных компьютеров при работе с большими базами данных. Рассматриваются особенности использования отечественной СУБД HyTech при решении задач, характерных для АСУ.

1. Введение

Наиболее перспективное направление развития автоматизированных информационно-управляющих систем связано с разработкой и использованием интегрированных информационных систем, основанных на концепциях автоматизированного банка данных. Интегрированные информационные системы получили в настоящее время широкое распространение. Для них создано большое число систем управления базами данных (СУБД), обеспечивающих создание, поддержку, ведение и развитие специальным образом организованных баз данных (БД) [1].

В связи с повсеместным внедрением персональных компьютеров проблемы создания высокоэффективных АСУ, САПР, систем искусственного интеллекта и др. стали особенно актуальными.

Это связано прежде всего с тем, что по ряду причин (в том числе и экономических) персональные компьютеры типа IBM PC перестали соответствовать своему исходному назначению. Теперь по своим техническим возможностям персональные компьютеры могут использоваться для решения сложных задач в реальном масштабе времени с большими объемами данных.

Пока объемы БД были невелики, методы доступа, реализованные в известных СУБД (Dbase, Paradox, FoxPro, FoxBase, DataFlex, Clipper, Clarion и др.), удовлетворяли пользователей и разработчиков. С ростом объемов БД стали появляться нелестные отзывы о низкой производительности перечисленных СУБД [2,3]:

при работе с большими БД,
при отработке многоключевых запросов,
при отработке межтабличных запросов,
при работе в сетевых средах.

Настоящая статья посвящена формализованному анализу эффективности методов доступа к БД и обоснованию технологии использования отечественной сверхбыстродействующей СУБД HyTech при решении задач АСУ.

2. Оценка эффективности СУБД

В настоящее время на программном рынке существуют сотни СУБД различного класса, около двух десятков из которых получили наибольшее распространение. Для оценки СУБД предложено большое число методик тестирования их эффективности,

каждая из которых имеет свои особенности. По этой причине проблема эффективной оценки и оптимального выбора СУБД является достаточно сложной.

Существуют методы оценивания СУБД с позиции разработчика и с позиции конечного пользователя. При этом предлагается оценивать инструменты программирования, языки, возможности работы с прикладными программами, хотя известно, что у каждого разработчика есть свое собственное мнение о том, как должна быть устроена идеальная среда разработки программ.

В качестве оцениваемых показателей обычно используются: скорость ввода и редактирования данных, степень автоматизации создания входных и выходных форм, возможности реляционного поиска и реляционного вывода, мощность языка программирования, скорость реляционных операций, скорость поисковых операций, скорость экспорта, импорта и создания индексов, простота изучения и использования, возможность диагностики и обработки ошибок, наличие исполнимого EXE-файла, возможность работы в сети, наличие "горячей линии" с разработчиками СУБД, стоимость СУБД и ее утилит и так далее.

Особенностью предлагаемого подхода является требование проведения анализа "работопригодности" СУБД на критических или предельных режимах работы информационной системы, характеризуемых определенным числом записей в базах данных и их размерами, числом рабочих станций в сети, особыми условиями работы системы.

Только положительные результаты предлагаемого тестирования могут дать убедительных ответ на вопрос: "Не пойдет ли вся проделанная работа "в корзину"?".

Как отмечает журнал InfoWorld: "Программы, отлично справляющиеся с импортом или с поиском на малых базах, могут оказаться совершенно неработоспособными, как только размер базы превысит определенный предел. Причина этого – не ошибки в системе, а различия с способах работы некоторых функций с базами разного размера".

В качестве средства описания методов доступа к данным, тестируемых СУБД, используется способ оценки путей доступа при выполнении операций выборки и обновления данных.

В качестве критериев эффективности выполнения поисковых операций выбираются оценка количества обращений к логическим записям БД, приведенная к единицам времени, и объем передаваемой по сети информации.

Напомним, что "метод доступа" – это совокупность технических и программных средств, обеспечивающих возможность хранения и выборки данных, расположенных на физических устройствах (обычно это внешняя память). "Механизм поиска" – это алгоритм, определяющий специфический путь доступа, который возможен в рамках заданной структуры памяти, и количество шагов вдоль этого пути для нахождения искомых данных [3].

Выделяются три основные группы методов доступа по приложениям пользователей:

1) получить все, получить многие;

2) получить уникальную;

3) получить некоторые.

Перед тем как перейти к детальному анализу методов доступа, введем необходимые переменные:

NR – длина последовательного файла в блоках,

EBF – коэффициент полезного блокирования,

$NDLK$ – [NR/EBF] число блоков в файле,

SBA – время доступа к блоку при последовательном доступе,

RBA – время доступа к блоку при произвольном доступе,

NM – число записей в БД,

NTR –число записей-целей запроса.

3. B-деревья

Рассмотрим широко используемую в большинстве современных СУБД организацию индексов в виде B-деревьев [2]. B-дерево представляет собой многоуровневый иерархически организованный индексный файл и позволяет осуществлять эффективный поиск по главному ключу первой записи, удовлетворяющей поисковому предписанию. Индексный файл содержит специальные точки входа, называемые "ключевыми", которые связывают номер записи информационного файла с соответствующим значением ключа в индексном файле, т.е. главный ключ служит указателем на записи-цели.

Отбор всех записей-целей, удовлетворяющих критерию поиска, производится путем обхода всего индексного дерева и считыванием записей-целей в ОП для дальнейшей обработки. Очевидно, что время поиска линейно зависит от числа записей, соответствующих значению главного ключа, и длины записи в БД.

Таблица 1

Тип приложения	Время поиска (B-деревья)	
	нижняя оценка	верхняя оценка
Получить уникальную	$2RBA$	$\log_{k+1}((NR+1)/2)$
Получить ВСЕ	$NR(1+1/2*k)RBA$	$NR(1+1/k)RBA$

Из приведенных в табл.1 аналитических соотношений, где K – коэффициент блокирования, видно, что время поиска всех записей, удовлетворяющих критерию поиска, может быть чрезвычайно большим и будет измеряться десятками минут и часами.

4. Инвертированные списки

Альтернативным методом организации данных являются инвертированные списки, использующие ассоциатор, связывающий значения ключей со списками внутрисистемных номеров записей информационного файла. соотвествующих этим значениям.

Поиск в БД с использованием инвертированных списков связан с поиском значений ключа в ассоциаторе и формированием полного списка записей-целей.

К преимуществам использования инвертированных списков в СУБД относятся следующие факторы: поиск ведется в ассоциаторе без обращения к самим данным, время поиска не зависит от длины записи и от длины ключа, время поиска практически не зависит от числа записей-целей и от числа записей в БД, результатом поиска являются число записей-целей и список их внутрисистемных номеров.

Следует обратить внимание на тот факт, что приведенные аналитические соотношения для инвертированных списков в [5] верны только для предложенного в этой работе алгоритма слияния результатов поиска по каждому элементу условий. Предполагается, что все индексы и списки указателей доспупа (внутрисистемных номеров) упорядочены. Вопрос: "Как оценить время выполнения запроса в противном случае (любой знак соотношения, кроме "=")" остается, к сожалению, открытым.

В силу того, что для нескольких условий множество записей-целей получается в результате объединения множеств записей-целей отдельных условий записи, оценка количества обращений для такого вида запроса равна сумме оценок для запросов с одним условием записи:

$$P = \mathrm{SUM}(p(i)) + \mathrm{SUM}(\mathrm{Sort}(NTR(i)),$$

где i – общее число атомарных запросов; $p(i) = P$(поиск в индексе 1) $+P$(поиск в индексе 2) $+P$(поиск в списке указателей доступа); индекс 1 – индекс типов элемента, индекс 2 – индекс значения элементов.

Если представить ассоциатор в виде матрицы и оглавления, а промежуточные результаты поиска хранить в специально подготовленном буфере, то оценка количества обращений примет следующий вид:

$$P = \mathrm{SUM}(p(i)),$$

где i – общее число атомарных запросов;

$$p(i) = \left\lceil \frac{NM * NR}{NTR * EBF} \right\rceil .$$

Если воспользоваться одним из преимуществ инвертированных списков – быстрое исполнение логического отрицания (в B-деревьях для этого потребуется совершить обход дерева в противоположном направлении), то в случае, если $p(i) \bigcup NM/2$, то можно легко оптимизировать алгоритм обработки запросов. При этом число обращений в БД можно определить следующим образом:

$$p(i) = 0,5 * [p(i) * [\mathrm{Sign}[NM/2 - p(i) + 1] + \\ + (NM - p(i)) * [\mathrm{Sign}[p(i) - NM/2] + 1]] .$$

Таким образом с помощью данного приема удается избавиться от самого "длительного" слагаемого – сортировки промежуточных результатов, которая занимает львиную долю времени выполнения запроса.

Сравнение верхних и нижних оценок для B-деревьев и для инвертированных списков показывает, что разница во времени поиска составляет для реальных больших БД сотни и тысячи раз.

Инвертированные списки обеспечивают самый быстрый метод доступа к записям-целям как при простом, так и при многоключевом поиске. Поэтому СУБД, построенная на инвертированных списках, будет особенно эффективна, если используется БД больших объемов (десятки, сотни тысяч и миллионы записей), занимающие десятки и сотни Мбайт.

Примером такой СУБД на IBM PC в среде MS DOS и сетевой среде NetWare и является СУБД HyTech (сокращение от Hyper Technology).

5. СУБД HyTech

Новые возможности технических средств, позволяющих хранить БД больших объемов, привели к необходимости использовать иные программные системы для их обработки. Одной из таких систем является СУБД HyTech.

СУБД HyTech реализована в виде резидентной программы (ядра) размером 80 Кбайт [2]. Кроме ядра в нее входят: утилита администратора БД, библиотека HyTech Vision, инструментальные средства для терминального доступа к БД(ИСБД), средства поддержки Call – интерфейса из программ, написанных на языках C, Pascal, Assembler.

Утилита администратора БД выполняет функции: создания и редактирования описаний схем и подсхем БД; импорта и экспорта данных; пакетного удаления, добавления и модификации записей в БД; загрузки и создания БД и ассоциатора; контроля соответствия ассоциатора области данных.

Комплекс ИСБД выполняет функции: настройки на конкретную схему БД; просмотра, добавления, удаления и модификации записей БД.

Комплекс ИСБД реализует операторы языка запросов к БД и межтабличные запросы.

Комплекс ИСБД создает форматы табличного вывода результатов поиска.

Комплекс ИСБД обеспечивает получение справочной информации о состоянии БД; обработку результатов запросов к БД средствами деловой графики.

Взаимодействие с СУБД из прикладных программ осуществляется посредством функций доступа к ядру. Каждая функция ядра оформлена в виде процедуры. Процедуры откомпилированы и собраны в библиотеки, используемые для разных моделей памяти (C, Assembler), или оформлены в виде .TPU файлов (Pascal).

Для вывода краткой справочной информации о возможных действиях пользователя и назначении клавиш используется нижняя строка экрана. Эта информация меняется при перемещении по пунктам основного или пунктам подчиненных меню и представлена в сокращенной форме. Получение подробных инструкций по текущему пункту меню или объяснения возможных действий выполняется посредством онлайнового справочника.

Системы меню программ администрирования и ИСБД построены по иерархическому принципу и содержат несколько уровней.

СУБД HyTech обеспечивает следующие характеристики: количество связанных файлов БД – определяется техническими характеристиками ПЭВМ; количество записей в одном файле – до 16 777 214; длина записи – до 65 520 байт; количество полей в записи – до 8192; длина ключевого поля – до 255 байт; длина простого поля – до 65 520 байт; количество ключей (включая уникальные) – до 8192.

Поддерживаемые СУБД HyTech типы данных: ASC – символьная строка (массив байтов); BYT – целое число без знака, занимающее один байт, с диапазоном значений (0 – 254); INT – целое число со знаком, занимающее два байта, с диапазоном значений (−32737 – +32737); LNG – длинное целое число со знаком, занимающее четыре байта, с диапазоном значений (−2147483647 – +2147483647); FLT – действительное число со знаком, занимающее четыре байта, с диапазоном порядка (3,4E−38 – 3,4E+38); DUB – действительное длинное число со знаком, занимающее восемь байтов, с диапазоном порядка (1,7E-308 – 1,7E+308); DAT – тип дата (2 байта).

Численные данные хранятся в машинном виде.

6. Фрагменты результатов тестирования СУБД HyTech

Фрагменты результатов тестирования СУБД HyTech на IBM PC AT/21 Мгц, ОЗУ – 640 Кбайт, диск – 112 Мбайт, среднее время доступа – 24 мс приведены в табл.2, а в работе [6] дано сравнение с известными СУБД.

Объем БД – 500 000 записей.

Длина записи – 97 байт (запись содержит один первичный и два возможных ключа).

Тестирование СУБД HyTech в многопользовательском режиме в сети NetWare и сравнение ее характеристик с Dbase-подобными СУБД подтвердили утверждения о том, что предложенная Hyper-технология обработки данных имеет неоспоримые преимущества не только в среде MS DOS, но и еще более наглядно проявляется при работе в сети.

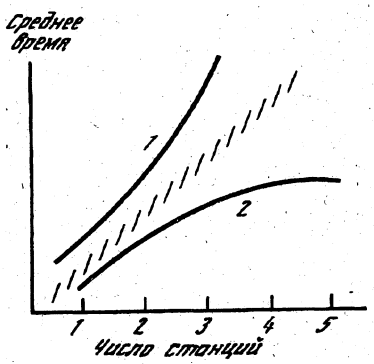
Таблица 2

Тип теста	Время, с		
	Минимальное	Среднее	Максимальное
Загрузка БД перевод во внутренний формат, создание всех индексов	1040		
Поиск по одному ключу. Допустимые знаки соотношения: $=, !=, <, >, \geqslant, \leqslant, [...], \min, \max$	0,05	0,1	0,2
Сложный поиск по двум ключам Допустимые логические соотношения: OR, AND, NOT	0,2	4,5	9,0
Сложный поиск по множеству $[N]$ ключей (включая скобочные вложения)	$0,1N$	$2,25N$	$4,5N$

Нижеприведенный фрагмент тестирования в сети NetWare на пяти станциях демонстрирует это.

Для двух баз данных объемом 100 000 записей каждая с длиной записи 101 байт каждая получены следующие результаты:

поиск по одному ключу	0,2 с,
поиск по двум ключам	7,5 с,
поиск по пяти ключам	12,0 с,
поиск по маске	32,0 с,
вставка записи	1,75 с,
замена записи	2,2 с,
удаление записи	0,65 с,
сложный поиск по двум БД	48,5 с.



1 – для Dbase-подобных,
2 – для СУБД HyTech.

Графики зависимостей среднего времени выполнения каждой из восьми описанных выше операций в зависимости от числа рабочих станций в вычислительной сети имеют вид, представленный на рисунке.

СПИСОК ЛИТЕРАТУРЫ

1. *Мамиконов А.Г., Кульба В.В., Лутковский Ю.П.* Анализ предметной области банков данных и построение оптимальных структур баз данных с учетом требований к достоверности информации. Препринт. М.: Институт проблем управления, 1988.
2. *Ковалевский С.С., Мялярский А.Н., Зайцев К.С.* СУБД НуТech – шаг к реальному масштабу времени // МИР ПК. 1991. № 8. С.56-63.
3. *Ковалевский С.С., Малярский А.Н., Зайцев К.С.* Взгляд на серверы баз данных // МИР ПК. 1992. № 4. С.59-63.
4. *Тайлер Бэнд.* СУБД на основе языка SQL // МИР ПК. 1991. № 4.
5. *Тиори Т., Фрай Дж.* Проектирование структур баз данных. Кн.2. М.: Мир, 1985.
6. *Гетманский А.Д., Топчян Б.Х.* СУБД: Так какая же быстрее // МИР ПК. 1992. № 2. С.59-62.

Поступила в редакцию 09.09.92

УДК 621.391.7

© 1993 г. О.В. КАЗАРИН,
Л.М. УХЛИНОВ, канд. техн. наук
(Москва)

ИНТЕРАКТИВНАЯ СИСТЕМА ДОКАЗАТЕЛЬСТВ ДЛЯ ИНТЕЛЛЕКТУАЛЬНЫХ СРЕДСТВ КОНТРОЛЯ ДОСТУПА К ИНФОРМАЦИОННО-ВЫЧИСЛИТЕЛЬНЫМ РЕСУРСАМ

Рассматривается протокол определения полномочий пользователя, реализуемый системой контроля доступа к информационно-вычислительным ресурсам. Основой протокола является принцип доказательства при нулевых знаниях, что позволяет реализовать его в интеллектуальных идентификационных карточках со встроенным вычислительным устройством. Приводятся алгоритмы доказательства прав пользователя и их проверки, базирующиеся на свойствах эллиптических кривых.

1. Введение

Для реализации надежного механизма защиты вычислительных систем требуется эффективная схема аутентификации (определения достоверности) пользователя. Традиционные таблицы паролей не представляют особого препятствия для опытного нарушителя. Даже если система защищена по входу с помощью достаточно стойкой односторонней функции, все еще остается возможность снятия пароля нарушителем во время его ввода с терминала или перехвата при передаче по линии связи. С появлением интеллектуальных (идентификационных) карточек (ИК) и протоколов вида "доказательства при нулевых знаниях" эта проблема понемногу находит свое решение. Безопасная ИК, способная выполнять автономные вычисления, или удаленный защищенный терминал могут гарантированно обеспечить санкционированный доступ к системе, не раскрывая информации, которой они пользуются для доказательства своих полномочий. Подобные системы можно отнести к системам высокоуровневой защиты, которые могут применяться в военных системах контроля команд, при защите кредитных карточек от подделок, для обеспечения безопасности входа в вычислительную систему и так далее. В данной работе рассматривается один из таких протоколов, который обеспечивает надежную и эффективную аутентификацию пользователей при доступе к информационно-вычислительным ресурсам.