Информатика и системы управления

УДК 004.056

С.Г. Самохвалова, А.О. Вороненко

ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ ДОКУМЕНТНО-ОРИЕНТИРОВАННОЙ СИСТЕМЫ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ

В статье рассматривается разработка оптимизированного механизма поиска информации в документно-ориентированных системах управления базами данных, обеспечивающих высокие скоростные характеристики при работе с документами различной структуры.

Ключевые слова: документно-ориентированные системы управления базами данных, документ, система управления базами данных.

DESIGN AND IMPLEMENTATION OF DOCUMENTALLY-ORIENTED DATA BASE CONTROL SYSTEM

The article considers the development of an optimized mechanism for searching information of document-oriented database management systems, which provides high speed characteristics when working with documents of different structures.

Key words: document-oriented database management systems, document, database management system.

Одна из важных задач компьютерных систем – хранение и обработка данных. Опыт применения ЭВМ для построения прикладных систем обработки данных показывает, что самым эффективным инструментом здесь являются не универсальные алгоритмические языки высокого уровня, а специализированные языки. Обычно они включаются в состав СУБД. Характеристики созданных прикладных пакетов определяются прежде всего принятой в СУБД организацией данных и типом используемого транслятора. СУБД позволяют структурировать, систематизировать и организовать данные для их компьютерного хранения и обработки.

Каждую систему управления базами данных можно охарактеризовать следующими параметрами:

1) качественные характеристики – возможность или невозможность хранения в БД информации той или иной структуры;

2) количественные характеристики – такие как максимальное количество хранимых в БД документов, максимальный объем одного документа;

3) скоростные – скорость выборки нужных документов из БД по произвольному набору поисковых элементов;

4) простота реализации на основе данной СУБД конкретных приложений;

5) рыночная стоимость СУБД.

На сегодняшний день на рынке программного обеспечения для персональных компьютеров существует большой выбор систем управления базами данных. Условно их можно разделить на две основные группы – «дешевые» и «дорогие».

К первой группе относятся такие популярные СУБД как FoxPro, Paradox, Microsoft Access, DBase и др. Они характеризуются средними количественными и качественными параметрами, простотой реализации на их основе конкретных приложений, а также очень малой рыночной стоимостью. Однако их качественные характеристики довольно низки и часто не удовлетворяют все более растущие потребности конечных пользователей. Основные из них – жесткость структуры хранимой информации, ограниченность объема документа, невозможность хранить или индексировать в этих СУБД объекты произвольного размера (тексты произвольной длины, графические изображения).

Ко второй группе относятся, условно говоря, «большие» СУБД, – например, Oracle, Informix, Sybase и др. Эта группа характеризуется высокими количественными, качественными и скоростными характеристиками. Разработка же конкретных приложений с использованием вышеупомянутых СУБД сопряжена с определенными трудностями – для них требуются специалисты высокой квалификации. Но самый главный их недостаток − очень высокая рыночная стоимость.

Для конечного пользователя привычнее получать информацию в виде электронных документов, которые представляют аналоги бумажных. Многие существующие СУБД предоставляют такие возможности, однако в большинстве случаев это сопряжено с появлением дополнительных затрат, так как существует необходимость наложения достаточно произвольной структуры документа на довольно жесткую структуру представления данных в СУБД. Эту трудность сейчас пытаются обойти, появились СУБД нового типа, которые позволяют хранить данные непосредственно в виде электронных документов. Такие системы получили название документно-ориентированных СУБД.

Документно-ориентированная СУБД – система управления базами данных, предназначенная для хранения иерархических структур документов (данных), в основе которой лежат документные хранилища, имеющие структуру дерева.

В основе метода лежит алгоритм отыскания дерева наименьшего веса для неориентированного графа с взвешенными ребрами. В действительности будет отыскиваться остовное дерево (минимальное покрывающее дерево) наибольшего веса, но поскольку все остовные деревья данного графа имеют одно и то же количество вершин, то нужный алгоритм может быть получен из алгоритма нахождения остовного дерева наименьшего веса с помощью перемены знака у весов ребер на противоположный.

Положим, что $R = \{R_1, R_2,\ldots,R_p\}$ – схема базы данных над U, G – некоторый граф пересечений для R. Классом атрибута $A \in U$ является множество $\{R|\ A \in R_i$ и $R_i \in R\}$ (Class (A)). Вес атрибута есть число WT(A) = |Class (A)| - 1. Вес схемы R, обозначаемый WT(R), по определению равен величине:

$$\sum_{A \in U} WT(A).$$

Вес A в G, обозначаемый $WT(A,G)$, – это число ребер в G, метки которых содержат A в качестве элемента. Весом G является величина:

$$WT(G) = \sum_{A \in U} WT(A,G).$$

Весом $WT(e)$ ребра e графа G является число |L(e)|. Непосредственно видно, что WT(G) можно также вычислить по формуле:

$$\sum_{e \in G} WT(e).$$

Рассмотрим граф соединений G для схемы базы данных R - {ABC, BD, CDE, DEI, UK}, изображенный на рис. 1.

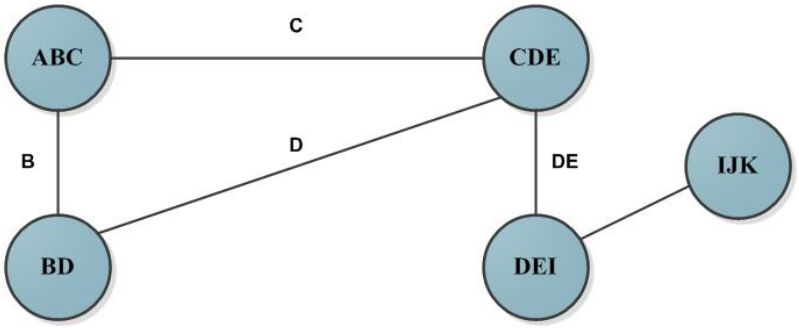

Рис. 1. Граф соединений базы данных R.

Для схемы R имеем:

WT (A) = 0, WT (E) = 1,
WT (B) = 1, WT (I) = 1,
WT (C) = 1, WT (J) = 0,
WT (D) = 2, WT (K) = 0

И поэтому WT (R) = 6. Для графа G:

WT (A, G) = 0, WT (E, G) = 1,
WT (B, G) = 1, WT (I, G) = 1,
WT (C, G) = 1, WT (J, G) = 0,
WT (D, G) = 2, WT (K, G) = 0

И, таким образом, WT (G) = 6.

Докажем, что если схема базы данных $R = \{R_1, R_2,\ldots,R_p\}$ обладает некоторым деревом соединений G, то любое остовное дерево наибольшего веса для графа I_R (относительно весов, приписанных его ребрам) – это дерево соединений. Более того, G является остовным деревом наибольшего веса для I_R и WT(G) =WT(R).

Сначала покажем, что WT(A) равняется WT(A,G) для любого атрибута A. Действительно, в G имеется WT (A)+1 вершин, содержащих A, и поэтому требуется не менее WT(A) ребер, чтобы построить A-пути между каждой парой этих вершин. Следовательно, WT(A,G)/WT(A). Любое ребро e, для которого $A \in L(e)$, соединяет некоторые вершины из Class(A). Если число ребер графа G, которые содержат A в своих метках, превышает WT(A), то найдется составленный из этих ребер цикл, проходящий через некоторое множество вершин из Class(A). Следовательно, $WT(A,G) \leq WT(A)$, так что WT(A,G) = WT(A). Отсюда следует, что WT(G)= WT(R) .

По определению G – остовное дерево для I_R. Предположим, что существует другое остовное дерево G для I_R, вес которого больше, чем вес G. Тогда найдется атрибут A, для которого WT(A,G) < WT(A,G). Согласно предыдущим рассуждениям, G должен содержать цикл, проходящий через некоторые вершины из Class(A), что противоречит выбору G. Поэтому G является остовным деревом наибольшего веса.

Пусть, наконец, G – любое остовное дерево наибольшего веса для I_R. Согласно предыдущей информации, WT(A) = WT(A,G) = WT(A, G) для любого атрибута A. Поскольку G – дерево и WT(A) ребер, содержащих G, которые содержат A в своих метках, то любые два элемента из Class(A) должны быть соединены некоторым A-путем. Следовательно, G – дерево соединения.

Вышеизложенное доказательство предоставляет достаточно эффективный способ проверки существования дерева соединений для произвольной схемы базы данных R. Найдем I_R (необходимо учитывать лишь ребра с непустыми метками), а потом найдем для него остовное дерево G наибольшего веса. Если G – дерево соединений, то, очевидно, схема R обладает деревом соединений. Если G не дерево соединений, то таких деревьев для R нет. Например, дерево G, изображенное на рис. 1, –

остовное дерево наибольшего веса для графа I_R, где R= {ABC, BD, ODE, DEI, IJK}. Дерево G не является деревом соединений и поэтому дерева соединений для R не существует.

На рис. 1 изображено остовное дерево G наибольшего веса для графа I_{Ra}, где R_a = {ABC, BCD, CDE}. Как уже отмечалось, G – дерево соединений для R_a.

Документно-ориентированные СУБД позволяют хранить и обрабатывать такие массивы данных, обработка которых с трудом поддается реляционным и даже объектно-ориентированным системам управления базами данных. Более того, благодаря использованию документно-ориентированной модели обработки СУБД предоставляет пользователям ряд полезных функций:

1) эффективное управление и распределение деловой информации, которая обычно представлена в виде различных типов данных: таблицы, отформатированный текст, связанные или внедренные объекты, полнотекстовый поиск информации, который позволяет пользователям индексировать документы и проводить их поиск по запросам;

2) хранение любых типов данных, начиная от простого текста, чисел, времени и дат, до форматированного текста, графических образов и произвольных данных, которые могут храниться в виде присоединенных объектов в своем родном формате;

3) достаточно гибкая функция управления версиями в документно-ориентированных базах данных, ее можно модифицировать в соответствии с потребностями любой рабочей группы.

Выделим несколько функциональных модулей.

Модуль ввода данных – удобный пользовательский интерфейс с наличием меню и пиктограмм для часто используемых функций, раскрывающихся списков ключевых слов, функций и переменных для автоматического ввода значений из данного списка. Интерфейс базы данных обеспечивает ввод новых и изменение уже хранящихся данных. Такая система необходима для ввода информации: о законах РФ, о видах социальной поддержке и признаках учета, о льготниках и членах их семьи, о выплатах, о предоставленных документах.

Модуль хранения данных – представление данных в виде физических таблиц для ведения статистики:

1) личные данные льготников и членов их семей;

2) отчеты, распоряжения, протоколы, решения;

3) выплатные ведомости, разовые поручения, почтовые переводы.

Модуль обработки данных – набор функций и процедур, необходимых для обработки данных, реализуемых посредством среды визуального программирования ReportEditor. После внесения входных данных программа начинает их обрабатывать. Процесс обработки заключается в составлении запросов, внесении изменений, формировании документов на бумажных носителях. После завершения обработки формируются выходные данные.

Модуль вывода данных – результаты проделанной работы. Выходные данные представлены в виде преобразованной и обработанной входной информации в форме отчетов.

Генератор отчетов FASTREPORTS формирует выходные файлы формата RTF, которые в дальнейшем через автоматизированную систему «Адресная социальная помощь» откроет Microsoft Word, а при его отсутствии – WordPad, умеющий разбивать на страницы, а также имеющий все стандартные возможности генераторов отчетов плюс возможности программирования через скрипты. Кроме того, удалось ввести в генератор несколько новых функций форматирования (например, вывод суммы прописью, вывод названия месяца по номеру) и реализовать работу с ключевыми словами. Через генератор удалось также получить довольно сложные отчеты, включая отчеты с вложенными таблицами (до 6 уровней).

Редактор отчетов FASTREPORTS работает непосредственно с указанным источником данных (таблица, SQL-запрос), дает возможность взять поля и расставить по отчету. При разработке шаблона используются ключевые слова, которые во время построения отчета связаны с источником данных.

На основе генератора отчетов FASTREPORTS предложен алгоритм накопления данных для документно-ориентированной СУБД, структура которой представлена в виде произвольного графа (рис. 2).

Рис. 2. Алгоритм накопления данных.

Суть алгоритма заключается в следующем: при наполнении базы данных (как при создании нового узла структуры, так и при модификации имеющегося) производится проверка существования дерева соединений для результирующей схемы базы данных. Если такое дерево существует, система производит модификацию структуры, в противном случае выдается сообщение о невозможности выполнения запрошенной операции.

Такая методика обеспечивает целостность структуры базы данных и позволяет обрабатывать документы, содержащие информацию произвольного вида.

В ходе работы был рассмотрен механизм отыскания дерева соединений, предложен алгоритм накопления данных для документно-ориентированной СУБД. В дальнейшем будет рассматриваться доработка документно-ориентированной СУБД и файла ключевых слов для обеспечения высоких скоростных характеристик при работе с документами различной структуры.

1. Алексеев, В.А. Основы проектирования и реализации баз данных. – Липецк: Липецкий гос. технический ун-т, 2014. – 14 с.
2. Бирюков, А.Н. Процессы управления информационными технологиями. – М.: ИНТУИТ, 2016. – 36 с.
3. Братченко, Н.Ю. Распределенные базы данных. – Ставрополь: Северо-Кавказский федеральный ун-т, 2015. – 58 с.
4. Суржко, С.В. Реляционные базы данных. – СПб.: Питер, 2001. – 84 с.
5. Тарасов, С.В. СУБД для программиста. Базы данных изнутри. – М.: СОЛОН-ПРЕСС, 2015. – 96 с.