УДК 681.3
А.А. Брюховецкий, доцент, канд. техн. наук,
С.К. Радзишевский, магистр
Севастопольский национальный технический университет
ул. Университетская 33, г. Севастополь, Украина, 99053
E-mail: root@sevgtu.sebastopol.ua

ПРОГРАММНАЯ МОДЕЛЬ ПРЕОБРАЗОВАНИЯ РЕЛЯЦИОННЫХ ДАННЫХ

Рассматривается программная модель, которая преобразует реляционные структуры данных в нереляционные массивы. Приводятся описание программной реализации и некоторые характеристики предлагаемого метода.

Ключевые слова: *базы данных, реляционные структуры данных, нереляционные массивы данных, UML-инструментарий, программная модель.*

Почти все продукты баз данных, созданные с конца 70-х годов, основаны на подходе, который называется реляционным. Реляционная модель имеет ряд недостатков [1]. В частности, в большинстве современных систем управления базами данных (СУБД) применяется такой способ преобразования, который может рассматриваться как взаимно однозначный, предусматривающий отображение отношений на хранимые файлы, а кортежей таких отношений — на хранимые записи в таких файлах. В подобных системах все данные, которые хранятся на физическом уровне, можно рассматривать в качестве непосредственного отображения того, что пользователь видит на логическом уровне. Поэтому из указанного факта следует, что подобные системы в действительности не обеспечивают настолько уж значительную независимость от данных. Еще одним недостатком является то, что обязательно приходится располагать данные в памяти только в одной физической последовательности. В результате, возникает необходимость в использовании индексов и других избыточных структур для поддержки доступа к данным, расположенным в той последовательности, которая отличается от требуемой. В связи с этим в последнее время особое внимание уделяется технологии обработки и хранения данных [2] с использованием нереляционной модели. В данной работе решается задача, связанная с реализацией программной модели использующей нереляционные данные, которая может эффективно применяться при обработке реляционных структур.

Система, реализованная с использованием модели преобразования реляционных данных (ПРД), может рассматриваться как охватывающая три уровня абстракции: реляционный (пользовательский) уровень, файловый уровень и уровень модели ПРД. На верхнем уровне данные представлены в виде отношений, которые обычным образом составлены из кортежей и атрибутов. На нижнем уровне данные представлены с помощью различных внутренних структур модели ПРД, называемых таблицами, а сами таблицы состоят из строк и столбцов. Отметим, что указанные таблицы не соответствует отношениям, кортежам или атрибутам на пользовательском уровне. При этом хранимая форма записи r состоит из двух логически различимых частей — множества значений полей и множества "связующих" данных, позволяюших связать друг с другом эти значения полей, причем для физического хранения каждой из этих частей можно воспользоваться широким спектром возможностей.

В системах с непосредственным отображением эти две части хранятся вместе. В отличие от этого, в модели ПРД эти две части хранятся отдельно — значения полей находятся в таблице значений полей, а связующая информация — в таблице реконструкции записей. Именно такое разделение является основным источником преимуществ, которые способна предоставить рассматриваемая модель.

Таблица значений полей фактически представляет множество различных вариантов сортировки одновременно (по существу, в этой таблице каждый отдельный атрибут отсортирован сразу в обоих направлениях — возрастающем и убывающем). В частности, таблица значений полей — это единственная таблица ПРД, которая содержит пользовательские данные как таковые. Все остальные таблицы содержат внутреннюю информацию (как правило, указатели), а эта информация имеет смысл только для рассматриваемой модели, но непосредственно не касается пользователя и ему не предъявляется.

Данные в таблице реконструкции записей представляют собой номера строк, а эти строки могут рассматриваться как указатели на строки таблицы значений полей. Указатели в таблице реконструкции записей образуют кольца. В эти кольца можно входить в любой точке. Поэтому процесс применения алгоритма реконструкции можно начинать с любой требуемой ячейки. При совместном использовании таблица реконструкции записей и таблица значений полей служат одновременно представителями всех способов упорядочения.

Программная система преобразования реляционных данных содержит подсистемы, изображенные на рисунке 1.

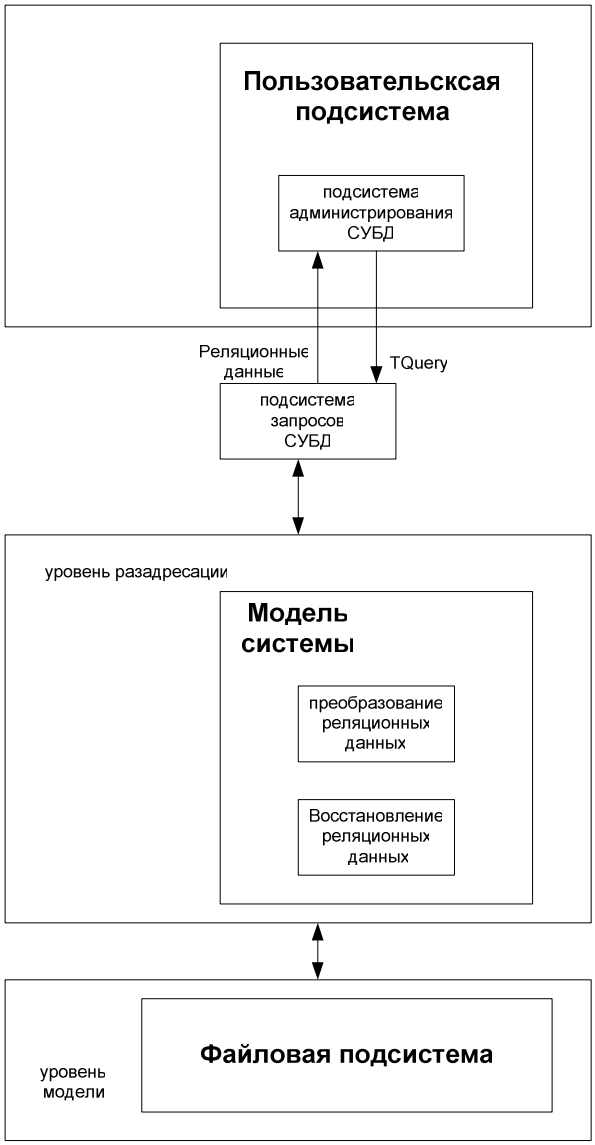

Рисунок 1 – Подсистемы программной системы и взаимодействие между ними

Такая организация архитектуры обеспечивает изоляцию верхнего и нижнего уровней. В качестве промежуточного подуровня между пользовательским и уровнем разадресации выделяется подсистема запросов – интерфейс взаимодействия между системами модели. Подсистема запросов обрабатывает объект запроса, созданный пользовательской системой, и возвращает объект результата запроса. Это позволяет добавлять и модифицировать разные типы запросов с минимальным изменением кода программы.

Проектирование программной системы выполнялось в среде *UML*-инструментария *Sparx Systems Enterprise Architect Corporate Edition 7*, которая в настоящее время успешно применяется при разработке и тестировании программных систем. В результате проведения объектно-ориентированного анализа были выделены основные классы программной системы, изображенные в виде диаграммы классов на унифицированном языке моделирования *UML* [3] (рисунок 2). Предлагаемая модель преобразования реляционных данных реализована средствами языка программирования *C++*.

Ключевым классом программной системы является класс базы данных – класс *TBD*. Он инкапсулирует свойства базы данных. База данных содержит таблицы – класс *TTable*. У каждой таблицы есть поля – класс *TField*. Кроме того, каждая таблица для получения своего реляционного (т.е. исходного) представления использует таблицу значения полей – *TValueTable* и таблицу реконструкции записей – *TReconstructionTable*. Все вышеперечисленные классы можно отнести к среднему уровню (разадресации) модели. Именно эти классы отвечают за восстановление и преобразование реляционных данных.

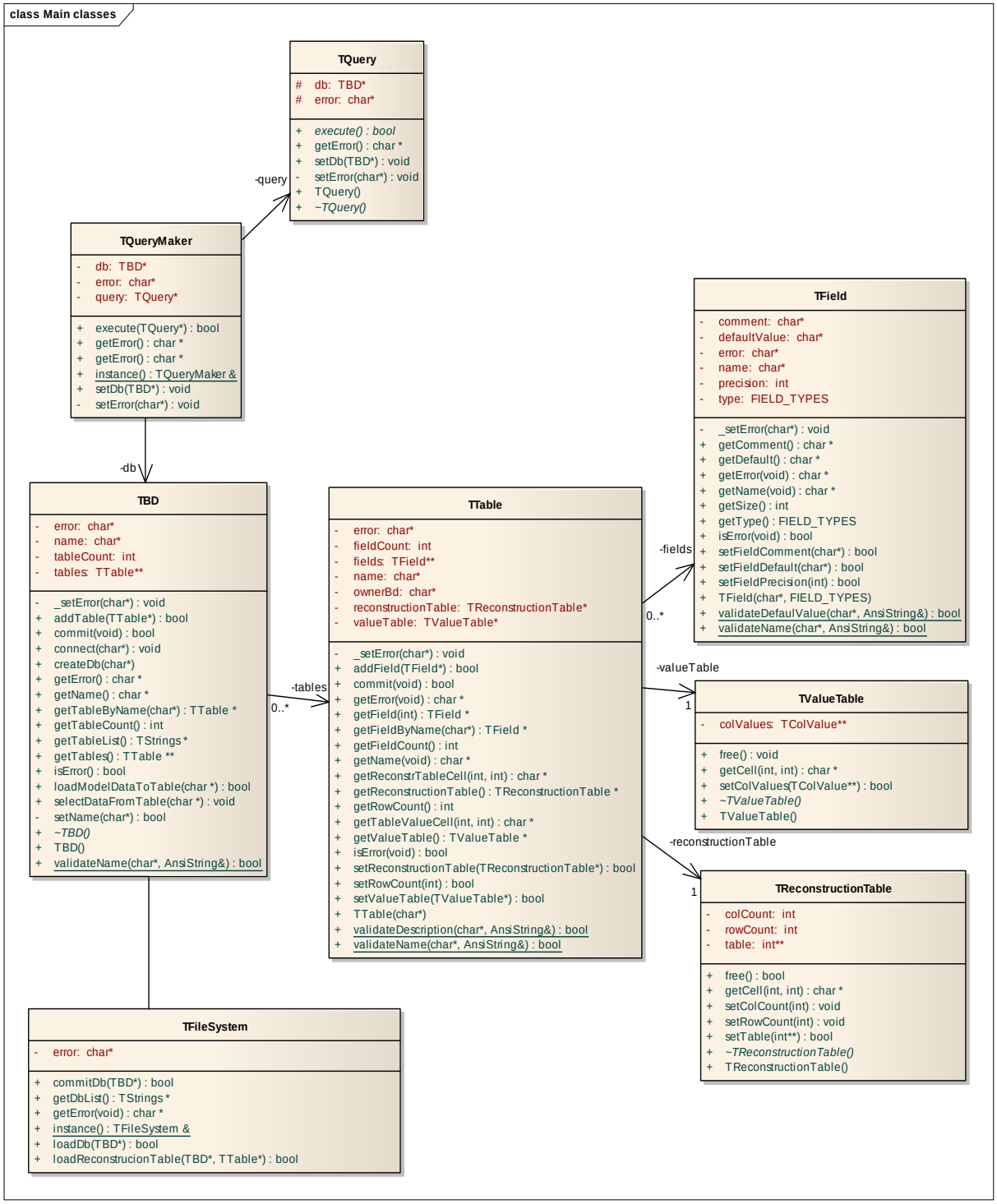

Рисунок 2 — Диаграмма классов программной системы на языке UML

Классами пользовательского уровня являются классы визуальных компонентов, которые на данной диаграмме опущены. Заметим, что объекты данных классов создают объекты классов запросов. Объекты классов запросов наследуются от базового абстрактного класса *Tquery*. Интерфейс данного класса содержит виртуальный метод *execute()*. Именно в этом методе переопределяются действия необходимые для данного класса запросов путем внесения минимальных изменений в программный код, связанных с применяемым методом.

Объект класса запроса передается в подсистему обработки запросов, представленной синглтон-классом *TQueryMaker*. Подсистема обработки запросов выполняет полученный запрос. Если запрос

выполняется удачно, то в подсистему выполнения запросов возвращается объект результата запроса, если случается ошибка, в пользовательскую систему передается сообщение об ошибке.

Исследуем оценки быстродействия операции соединения в реляционной и предлагаемой моделях. Ограничимся только анализом затрат на выполнение операций ввода-вывода одного кортежа без учета распределения кортежей по страницам. Пусть имеется два отношения R и S, содержащие соответственно m и n кортежей. В [1] приводятся оценки быстродействия операции соединения в реляционной СУБД. Методом последовательного просмотра потребуется всего $m + (n * m)$ операций чтения кортежей. В описанном подходе предполагается, что для отношения S не предусмотрен индекс и не применяется хэш-функция для ускорения доступа к данным.

Метод, использующий поиск по индексу, требует $m + (m * n)/dCS$ операций чтения , где dCS — количество различных значений атрибута C отношения R.

Рассмотрим операцию соединения в модели преобразования реляционных данных. Ниже представлен фрагмент псевдокода.

```
j = 0;
//j = binarSearch(R.C, S.C[0])
i = 0;
while(i < m) {
        if (R.C[j] == S.C[i]) {
        // добавить соединяемый кортеж R[i]*S[j] к результату
        } elseif (R.C[j] > S.C[i]) {
                i++;
       }  elseif (R.C[j] < S.C[i]){
                j++;
        } else { break;    }
}
```

Модель позволяет выполнять операцию соединения по методу сортировки-слияния без необходимости выполнения самой сортировки, поскольку сортировка осуществляется при формировании таблиц значений полей и реконструкции записей. В данном случае количество операций чтения составит $m + n$.

Таким образом, модель преобразования реляционных данных обеспечивает гораздо большую независимость от данных, по сравнению с той, которая может быть достигнута в системах с непосредственным отображением. Также отсутствует необходимость использования избыточных структур, таких как индексы. Кроме того, сжатие данных которое может использоваться в модели предусматривает что, столбцы таблиц содержат только соответствующие уникальные значения. При этом модель сохраняет многие положительные стороны реляционной модели, такие как относительная простота работы разработчика (пользователь по прежнему работает с обычными таблицами).

Дальнейшие исследования предполагается провести с целью получения оценок быстродействия выполнения основных реляционных операторов при использовании усовершенствованного представления таблиц модели ПРД.

Библиографический список

1. Дейт К.Дж. Введение в системы баз данных / К.Дж. Дейт. — М.: «Вильямс», 2004. — 1316 с.

2. U.S. Patent and Trademark Office: Value–Instance–Connectivity Computer– Implemented Database. U.S. Patent No. 6,009,432. — December 28, 1999.

3. Нейбург Э.Дж. Проектирование баз данных с помощью UML / Э.Дж. Нейбург, Р.А. Максимчук. — М.: Вильямс, 2002. — 88 с.

Поступила в редакцию 30.07.2009 г.