УДК 681. 324

СПОСОБ ПОВЫШЕНИЯ ЭФФЕКТИВНОСТИ ОБНОВЛЕНИЯ ИНФОРМАЦИИ В РЕЛЯЦИОННЫХ БАЗАХ ДАННЫХ С ИЕРАРХИЧЕСКОЙ СТРУКТУРОЙ

С.В. Дуденко
(Харьковский университет Воздушных Сил им. И.Кожедуба)

С целью повышения эффективности обновления информации в реляционных базах данных с иерархической структурой предложен способ, основанный на использовании триггеров, который позволяет существенно сократить время обновления информации.

реляционная база данных, обновление информации, триггер

Постановка задачи. Реляционная база данных (БД) имеет набор ограничений целостности, которые гарантируют корректность данных. Существует два важных правила целостности, которые называются целостностью сущностей и ссылочной целостностью. Правило ссылочной целостности основано на согласованности внешних ключей, т.е. для каждого значения внешнего ключа должно существовать соответствующее значение первичного ключа в родительском отношении [1]. Указанное правило достигается путем каскадирования операций удаления и обновления данных.

При обновлении информации, система управления базой данных (СУБД), как правило, осуществляет последовательно операции удаления старого и вставки нового значения. Данные операции скрыты от конечного пользователя, и он не может влиять на ход их выполнения. В случае, когда реляционная БД имеет иерархическую структуру (не следует путать с иерархической моделью) возникает лавинообразное каскадирование операции обновления всех дочерних таблиц, что существенно повышает время обновления базы данных в целом.

Предположим, указанная система разнесена на нескольких серверах в компьютерной сети, тогда задача загрузки и синхронизации серверов решается с помощью репликации данных. В случае отсутствия постоянной связи эта задача усложняется, особенно в условиях необходимости частого обновления информации. Как вариант решения указанной задачи в современных источниках предлагают использовать формат данных xml для переноса информации между базами данных. Однако при таком

© С.В. Дуденко

подходе к загрузке больших массивов информации, в отсутствии постоянного соединения, время обновления базы данных в целом будет велико, откуда формируется задача минимизации указанного времени.

Анализ литературы. Вопрос обеспечения целостности баз данных с реляционной моделью глубоко раскрыт во многих современных источниках, при этом наиболее полно эта тема раскрывается в [2, 7]. Однако вопрос повышения эффективности обновления информации в реляционных базах данных с иерархической структурой не рассматривается в полном объеме. В [4] как вариант предложен способ, основанный на частичной загрузке данных в её локальную копию, что не всегда возможно.

Целью статьи является рассмотрение механизмов обеспечения целостности в реляционных БД с разработкой способа повышения эффективности обновления информации в БД с иерархической структурой, основанном на использовании триггеров, который позволит существенно сократить время обновления информации.

В распределенных БД [8] возникает большое количество дополнительных задач, которые не являются типовыми для локальных: загрузка больших массивов информации, отслеживание изменений, вносимых разными пользователями и соответственно повышение эффективности обновления информации. Последовательно рассмотрим каждую из задач.

Загрузка больших массивов информации. В конце 60-х годов появились работы, в которых обсуждались возможности применения различных табличных даталогических моделей данных, т.е. возможности использования привычных способов представления данных. Наиболее значительной из них была статья сотрудника фирмы IBM Э.Кодда [1], где впервые был применен термин "реляционная модель данных". Будучи математиком по образованию Э.Кодд предложил использовать для обработки данных аппарат теории множеств (объединение, пересечение, разность, декартово произведение). Он показал, что любое представление данных сводится к совокупности двумерных таблиц особого вида, известного в математике как отношение – relation. Эта работа послужила стимулом для большого количества статей и книг, в которых реляционная модель получила дальнейшее развитие.

Иерархическая структура предполагает построение в виде, представленном на рис. 1.

После семантического моделировании предметной области для автоматизированной информационной системы на основе базы данных, как правило, используется реляционная модель с иерархической структурой, когда в качестве вершины 1 (рис. 1) используют центральную

сущность предметной области. Например, для информационно-аналитической системы «Кадры» – это будет общая информация о человеке (табл. 1).

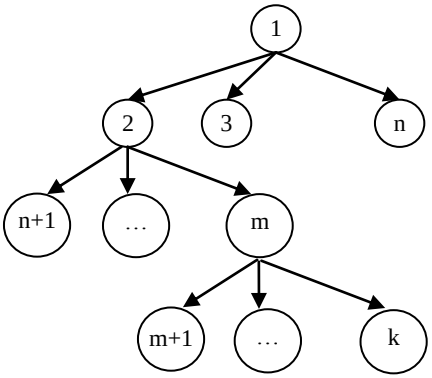

Рис. 1. Иерархическая структура

Таблица 1

Общая информация о человеке

Название	Домен	Назначение
codsotrudnica	bigint	Идентификационный код человека
sex	bit	Пол
surname	nvarchar(50)	Фамилия
nam	nvarchar(40)	Имя
father	nvarchar(40)	Отчество
foto	image	Фотография
datainput	timestamp	Для отслеживания изменений
dataedit	smalldatetime	Дата внесения изменений
whoedit	nvarchar(30)	Кто внес последние изменения

К центральной таблице подключают с помощью связей остальные, выстраивая систему иерархически и группируя по функциональному признаку. Таким образом, получаем реляционную базу данных с иерархической структурой. Развивая систему (рис. 2) получим базу данных, которая будет содержать несколько сотен таблиц, при этом только иерархическая структура содержит порядка 35 таблиц.

Для разработки прикладных алгоритмов управления базами данных в последнее время широко используют платформу dot.Net и разработанный для неё язык программирования C#, в пространстве которого существуют методы работы с форматом данных xml [4, 5]. Для записи это:

myDataSet.WriteXml (myMemStream, XmlWriteMode.WriteSchema),

а для чтения:

myDataSet.ReadXml (myMemStream, XmlReadMode.ReadSchema),

где DataSet – набор таблиц базы данных.

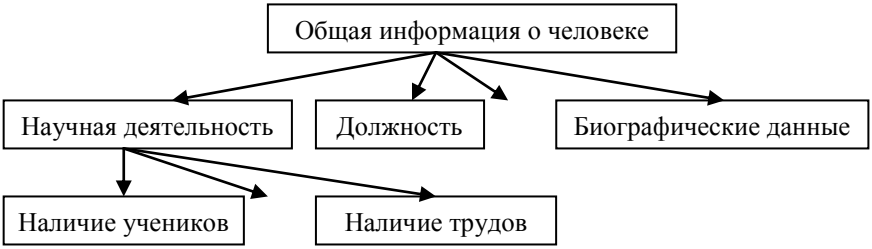

Рис. 2. Иерархическая структура
для информационно-аналитической системы «Кадры»

Формирование набора DataSet, осуществляется с использованием метода Merge() [5] объекта DataSet, который может объединить массив объектов DataRow, объект DataTable или объект DataSet с существующим объектом DataSet. Если у объекта DataSet, служащего приемником, определен первичный ключ, то данные из источника сопоставляются строкам приемника, имеющим то же значение первичного ключа. Там, где обнаруживаются совпадения, в существующую строку приемника записываются обновленные значения. В противном случае в приемнике создается новая строка. Такой подход к объединению наборов данных приводит к тому, что при записи изменений в базу данных возникает необходимость обновления всей иерархической структуры.

Отслеживание изменений, вносимых пользователями. Для отслеживания изменений, вносимых пользователями достаточно написать простой триггер для центральной таблицы, например, в случае реализации системы на СУБД SQL-server [6], с использованием языка SQL [3] для табл. 1:

```
CREATE TRIGGER CorrDataEdit ON dbo.prsn_multiplexedinformation
OF UPDATE
AS BEGIN
    UPDATE [dbo].[prsn_multiplexedinformation]
    SET
            dataedit = GETDATE(),
            whoedit = SYSTEM_USER
    FROM inserted
            WHERE [dbo].[prsn_multiplexedinformation].[codsotrudnica]
                    IN (SELECT codsotrudnica FROM inserted)
```

END

Гарантом выполнения триггера будет наличие временного штампа в таблице 1 – datainput, определенного в домене timestamp. Однако, при таком подходе к загрузке больших массивов информации, которые планируется переносить с использованием формата xml в отсутствии постоянного соединения и определенным выше механизмом отслеживания изменений, время обновления БД в целом будет велико, так для обновления 1000 записей в центральной таблице потребуется порядка 3 минут.

Способ повышения эффективности обновления информации в реляционных базах данных с иерархической структурой. В качестве показателя эффективности обновления информации выберем время выполнения операции обновления, критерием эффективности будет выступать правило минимизации времени.

В чем причина такого увеличения времени? Прежде всего, при объединении наборов данных все строки центральной таблицы переходят в состояние modified, а следовательно требуют обновления всей центральной таблицы, что в свою очередь вызывает лавинный эффект каскадирования операции обновления всех дочерних таблиц.

Суть предложенного способа состоит в использовании специального триггера, который отслеживал бы признаки обновления столбцов. Триггер создается с целью определения номера столбца (ов), которые требуется обновить. Он допускает обновление либо столбца (ов), которые описывают первичный ключ центральной таблицы, либо другие изменения, что позволяет предотвратить лавинный эффект в случае загрузки больших объемов информации. Для таблицы 1 триггер имеет вид:

```
CREATE TRIGGER CorrDataEdit ON dbo.prsn_multiplexedinformation
INSTEAD OF UPDATE
AS IF (COLUMNS_UPDATED()  & 1) = 1
BEGIN
    UPDATE [dbo].[prsn_multiplexedinformation]
    SET     codsotrudnica = inserted.codsotrudnica,
            dataedit = GETDATE(), whoedit = SYSTEM_USER
    FROM inserted
            WHERE [dbo].[prsn_multiplexedinformation].[codsotrudnica]
                    IN (SELECT codsotrudnica FROM inserted)
END
ELSE BEGIN
    UPDATE [dbo].[prsn_multiplexedinformation]
    SET     sex = inserted.sex,  surname = inserted.surname, nam = inserted.nam,
            father = inserted.father,  foto = inserted.foto,
            dataedit = GETDATE(), whoedit = SYSTEM_USER
```

```
    FROM inserted
        WHERE [dbo].[prsn_multiplexedinformation].[codsotrudnica]
            IN (SELECT codsotrudnica FROM inserted)
END
```

Как видно из триггера, причина лавинного эффекта – эта столбец codsotrudnica, обновление которого и служит основой резкого увеличения времени. После записи триггера в таблицу 1, для обновления 1000 записей в центральной таблице потребуется всего 7 секунд,

Вывод. Таким образом, можно сделать вывод, что использование предложенного способа повышения эффективности обновления информации в реляционных базах данных с иерархической структурой, основанного на использовании триггеров, позволяет уменьшить время обновления 1000 записей с 3 минут до 7 секунд, что составляет суммарный выигрыш ≈ 25 раз.

ЛИТЕРАТУРА

1. *Codd E.F. A Relational Model of Data for Large Shared Data Banks. CACM 13: 6, June 1970.*
2. *Конноли Томас.Базы данных:проектирование, реализация и сопровождение. Теория и практика. Пер. с англ.: Уч. пос. – М.: Издательский дом "Вильямс", 2000. – 1120 с.*
3. *Джеймс Р. Гробф, Пол Н. Вайнберг. SQL: полное руководство. Пер. с англ. – К.: Издательская группа BHV, 2000. – 608 с.*
4. *Гамильтон Б. ADO.NET Сборник рецептов. Для профессионалов. – СПб.: Питер, 2005. – 576 с.*
5. *Темплман Джулиан, Виттер Дэвид. Net Framework: Библиотека классов. Пер. с англ. – М.: КУДИЦ-ОБРАЗ, 2003. – 672 с.*
6. *Мамаев Е.В. Microsoft SQL Server 2000. – СПб.: БХВ-Петербург, 2004. – 1280 с.*
7. *Райордан Ребекка. Основы реляционных баз данных. Пер. с англ. — М.: Издательско-торговый дом «Русская Редакция», 2001. – 384 с.*
8. *Пономаренко В.С., Павленко Л.А. Організація даних у розподілених інформаційних системах: Навчальний посібник. – Х.: РІО ХДЕУ, 2000. – 104 с.*

Поступила 23.05.2006

Рецензент: доктор физико-математических наук, профессор С.В. Смеляков, Харьковский университет Воздушных Сил им. И. Кожедуба