Вопросы преподавания и приложения

УДК 517.9

Управление данными в корпоративных информационных системах

© С. Н. Лизин[1],С. А. Федосин[2]

Аннотация. Производители СУБД и корпоративных платформ активно соревнуются в разработке новых функциональных возможностей своих продуктов. Однако многие проблемы, такие как совместная обработка данных или управление их жизненным циклом, могут быть решены проще и эффективнее посредством технологий управления данными.

Ключевые слова: базы данных, данные, темпоральные, транзакционные, СУБД, информационные системы.

1. Проблема отслеживания изменений

Увеличение объема информации и развитие информационных систем – два взаимосвязанных процесса: рост объемов данных требует развития средств их обработки, совершенствование которых стимулирует обработку все больших массивов данных. Сегодня практически в каждой более-менее крупной организации деятельность строится вокруг корпоративной информационной системы (а зачастую и нескольких), большинство из которых построены на основе реляционных систем управления базами данных по трехзвенной архитектуре: клиентские приложения – серверы приложений – серверы баз данных. При разработке каждой такой системы приходится решать ряд проблем, связанных с управлением данными.

Одной из наиболее сложных сегодня остается задача управления конкурентным доступом к данным. Обычно под решением данной задачи подразумевают защиту от проблем «грязного» чтения (dirty read), неповторяющегося чтения (non-repeatable read) и потерянного обновления (lost update). Для решения проблемы «грязного» чтения, заключающейся в чтении данных, записанных отмененной операцией, вполне может быть достаточно средств сервера баз данных, реализующих стандартные транзакции. Решение же проблемы неповторяющегося чтения (состоящей в отличии результатов первичного и последующего чтения данных одним клиентом, не изменявшим их), а также проблемы потерянного обновления (возникающей при одновременном изменении различными клиентами одного и того же значения элемента, при котором более раннее изменение теряется) средствами транзакций на уровне только сервера баз данных и/или сервера приложений эффективно лишь в отношении серверных задач обработки данных, которые выполняются на сервере приложений или сервере баз данных в фоновом режиме по заранее заданному алгоритму и без активного участия пользователя.

Однако защиту от аналогичных проблем требуется обеспечивать и при ручной обработке данных пользователями с использованием их клиентских приложений. Основное отличие здесь заключается в том, что каждый пользователь работает не напрямую с данными содержащимися в СУБД, а с некоторой их локальной копией, загруженной в клиентское приложение. Зачастую это приводит к тому, что когда два пользователя сохраняют изменения в одном и том же элементе (причем их исправления

[1]Аспирант кафедры АСОИУ, МГУ им. Н. П. Огарева, г. Саранск; sergey.lizin@gmail.com.
[2]Заведующий кафедры АСОИУ, МГУ им. Н. П. Огарева, г. Саранск; sergey.lizin@gmail.com.

могли касаться разных его свойств), более раннее изменение теряется, причем клиент, чье изменение потеряно, может некоторое время этого даже не видеть.

Считается, что защита от подобного рода проблем должна обеспечиваться логикой приложения за счет более активного использования транзакций. Но не все проблемы конкурентного доступа можно обойти программно за счет дополнительных проверок и транзакций. И ключевой фактор здесь – человек. Невозможно выделить программно последовательность чтений и записей пользователем данных, составляющих атомарную операцию. Следовательно, нельзя и откатить такую транзакцию при изменении данных чтения – система просто не в состоянии определить, какое количество последних действий пользователя образуют единое целое, да и какие из прочитанных им данных учел пользователь при принятии решения. Что же остается делать?

Как и во всех случаях, когда предотвратить проблему невозможно, все усилия должны быть сконцентрированы на ее отслеживании. Причем в корпоративных системах управление конкурентным доступом – далеко не единственное направление, где требуется решение задачи отслеживания изменений. Другой очень важной проблемой является сложность выявления данных, на основе которых сформированы другие, как правило, более обобщенные данные. В качестве характерного примера можно привести различного рода отчеты, формируемые на основании данных из системы, не сохраняющиеся в ней. Определить в последующем источники формирования каждого сводного показателя отчета можно лишь при условии неизменности исходных данных.

В качестве одного из решений здесь возможно введение административного запрета редактирования данных, по которым сформированы отчеты, однако, это далеко не всегда возможно, да и не слишком удобно. А с другой стороны, вполне обычным является процесс, когда после сдачи первичного отчета в последующем, при выявлении неточностей, эти неточности исправляются, а на основании исправленных данных формируется уточненный отчет (таков, к примеру, порядок в отношении налоговой отчетности).

В общем виде целью отслеживания изменений является возможность выяснения данных, существовавших в системе (локальной копии данных) на момент принятия пользователем решения об изменении данных, формирования отчета и т.п. Кроме того, зачастую также требуется наличие информации об авторстве изменений и исправлений. Причем сведения нужны относительно всех модификаций, а не только последних, так как ошибки в данных (умышленные или неумышленные) могли быть внесены ранее, а другие пользователи в последующем основывались на них.

Особенно актуальной данная задача выглядит в свете повсеместного перехода от бумажных информационных ресурсов к электронным, в которых требуется обеспечение придания юридической значимости данным (посредством ЭЦП). Отдельно следует отметить, что речь здесь идет не о придании юридической значимости отдельным документам, передаваемым между организациями, а о формировании полностью юридически значимого массива данных, любая выборка из которого также имеет официальный статус. Естественно, что при этом должна обеспечиваться персональная ответственность лиц, формирующих данный информационный массив. Особенно это важно для государственных систем, где каждый служащий имеет свои полномочия и несет соответствующую ответственность.

Также задача отслеживания модификаций данных крайне важна для обеспечения их синхронизации при совместной обработке данных в нескольких информационных системах. Если системы работают последовательно (то есть данные из первой служат сырьем для второй) то все относительно просто. Сложнее дело обстоит с организацией параллельной обработки данных, например, при управлении мастер-данными. Здесь широко применяется централизованная модель, при которой все мастер-данные хранятся

в выделенной системе, однако, это не всегда удобно и возможно, особенно при взаимодействии информационных систем различных организаций. Кроме того, данная модель не основана на технологии управления данными, а представляет собой лишь архитектурное решение.

Если вернуться к государственным информационным системам, то организация параллельной обработки данных в электронных информационных ресурсах, каждый из которых содержит юридически значимую информацию, является одним из основных инфраструктурных элементов к построению электронного правительства.

2. Транзакционные и нетранзакционные

Для решения задачи отслеживания всех производимых модификаций разумно использовать технологии темпоральных баз данных. Но на сегодняшний день полноценные промышленные реализации темпоральных баз данных, по сути, отсутствуют. Что любопытно, они вполне могли появиться лет пять-десять назад, но в силу модных тенденций в сфере управления данными, сместивших акценты в сторону поддержки XML и решения задач интеграции, темпоральные технологии остались в стороне. Некоторые современные СУБД содержат специализированные механизмы, которые позволяют использовать фоновую версионность значений атрибутов. Однако ее далеко не всегда удобно использовать. Но важнее другое – реляционная модель данных предоставляет весьма широкие средства, которые вполне могут быть использованы для решения перечисленных проблем. Может быть и не стоит ее искусственно расширять дополнительной временной размерностью?

Данные, обрабатываемые в каждой информационной системе можно разделить на транзакционные и нетранзакционные. Транзакционные данные – это данные, каждая запись которых относится к фиксированному моменту времени и содержит сведения, фиксированные на данный момент времени, не изменяющиеся в последующем. Соответственно нетранзакционные данные – все остальные.

Транзакционные данные обычно представляют собой повторяющиеся примеры событий, явлений, происшествий одного и того же типа. Сюда относятся все заявки, счета, накладные – да вообще все документы (как сущности), так как все они фиксированы (так как это уже оформленные документы) и привязаны к некоторому моменту времени (времени их составления или регистрации). Нетранзакционные данные, которые часто называют справочными или базовыми, представляют собой списки однотипных объектов: сущностей, предметов и абстрактных категорий. Как правило, это разнообразного рода справочники и классификаторы, другими словами – мастер-данные или, как частный случай, нормативно-справочная информация.

Нетрудно заметить, что связь между этими двумя категориями, как правило, односторонняя: при описании транзакционных схем могут использоваться как транзакционные, так и нетранзакционные данные, а при описании же нетранзакционных данных, как правило, используются только нетранзакционные данные. Например, при составлении накладной используются несколько справочников (контрагентов, номенклатуры), а основанием для накладной может служить, например, счет.

С другой стороны, если не принимать во внимание конкретное информационное представление данных и процедуры их архивации и утилизации, то можно утверждать, что количество информационных объектов транзакционных данных постоянно увеличивается, так как регистрация каждого нового события вызывает появление новой записи. В то же время, количество информационных объектов нетранзакционных

данных относительно статично.

Значение каждой записи транзакционных данных, как правило, остается неизменным с момента ее фиксации. Исключения составляют случаи коррекции записи ввиду неточностей или ошибок, что, кстати, обычно, в информационной системе рассматривается как не совсем корректное действие – коррекция должна проводится отдельной операцией. Нетранзакционные данные не привязаны к конкретному моменту времени, но могут быть определены в течение их жизненного цикла. Значения атрибутов каждого элемента таких данных в течение его жизненного цикла могут меняться.

Таким образом, проблема отслеживания модификаций актуальна по большей мере для нетранзакционных данных. Для транзакционных данных она имеет смысл лишь в части отслеживания исправлений.

Одной из основных причин появления проблем, связанных с представлением нетранзакционных данных, является то, что нетранзакционные данные часто воспринимаются как условно-постоянные и, как следствие, имеющие статичное представление в базах данных. Для записи данных в SQL используются операторы вставки (insert), изменения (update) и удаления (delete). Как правило, данные операторы используются весьма прямолинейно: появился новый объект – вставили новую запись, изменились его характеристики – обновили их, исчез объект – удалили запись. Справедливости ради следует отметить, что для повышения ссылочной целостности вместо операции удаления в настоящее время все больше используется обновление дополнительного атрибута (статуса) записи.

Однако, операция обновления также не безобидна. Поступление информации о прекращении существования объекта – это новая информация, однако результатом поступления информации не должно быть общее сокращение информации в базе данных, наоборот, должно происходить ее увеличение. Аналогично поступление информации об изменении характеристик одного из объектов является поступлением новой информации, и общий объем информации в базе данных также должен увеличиваться. То же самое можно сказать и про исправления. Другими словами, информационные элементы должны хранить информацию о жизненном цикле реальных объектов, а не повторять его.

Используя термины SQL, можно говорить, что использования для отражения прекращения существования объектов и изменения их характеристик недопустимо использование операторов delete и update. Данные операторы являются служебными и должны использоваться исключительно в служебных целях: для перемещения, архивации и утилизации массивов данных. Таким образом, можно сделать вывод, что все данные, циркулирующие в базе данных должны быть транзакционными; нетранзакционные данные должны быть представлены в виде цепочки транзакционных данных.

3. Темпоральность в реляционной СУБД

Транзакционные данные, как правило, ассоциированы только с одним значением времени. Для обеспечения же управления конкурентным доступом и реализации раздельного отражения операций исправления, требуется использование битемпоральной модели, включающей действительное и транзакционное время. Действительное время – это время, указывающее на время актуальности существования и значений атрибутов реальных объектов. Транзакционное время – время внесения новых сведений об объекте в информационную систему.

Двойная темпоральность позволяет не просто более точно описать модель системы, но и определить операцию исправления. Для этого в дополнение к существующей записи

с неправильным значением вносится корректирующая запись с тем же действительным временем, текущим транзакционным временем и исправленными значениями.

Кроме расширения возможностей представления данных в информационной системе, использование битемпоральной модели позволяет обеспечить управление конкурентным доступом в части защиты от проблем неповторяющегося чтения. Это достигается за счет того, что в рамках одной операции (транзакции) при обработке данных используется ограничение на выборку данных: используются только записи, внесенные в информационную систему до начала транзакции.

Двойная темпоральность данных в сочетании с управлением данных в режиме «только вставка», позволяют обеспечивать хранение вместе с данными сведения о пользователе, внесшем изменения, а также его электронную цифровую подпись, рассчитанную на основании внесенных данных. Это позволяет организовать юридически значимое хранилище данных с разделением персональной ответственности за их содержание. Кроме того, данная технология позволяет обеспечить возможность отслеживания данных, которые были в системе до внесения пользователем изменений. Аналогичным же образом можно отследить первичные данные, находившиеся в системе при формировании вторичных данных, при условии сохранения с вторичными данными штампа времени, по состоянию на которое они сформированы.

При этом следует отметить, что в трехзвенной архитектуре здесь возможны конфликтные ситуации, связанные с тем, что между временем чтения данных и моментом внесения изменений другим пользователем в данные могут быть внесены изменения. Для того чтобы защититься от этого, имеет смысл использовать два транзакционных времени: основное транзакционное время – транзакционное время внесения изменений; и транзакционное время чтения данных – для отслеживания основного транзакционного времени данных, на основании каких данных производилась их обработка.

Однако в данной модели в чистом виде невозможно отобразить исчезновение (удаление) объекта. Однако, кроме операций появления, исчезновения элементов и изменения их атрибутов, с ними могут происходить также такие события как объединение, присоединение, разделение, выделение, реорганизация. Для их реализации необходимо использовать дополнительную таблицу, причем отражение должно осуществляться не точкой, как обычно, а переход (вектор), с указанием предшествующего идентификатора (null при появлении элемента) и нового идентификатора (null при исчезновении элемента). Используя данный подход несколькими взаимосвязанными записями можно отразить в базе данных любую из перечисленных выше операций. Немаловажной проблемой является обеспечение мягкой модернизации информационных систем вместе с соответствующими взаимосвязями между ними. В первую очередь сложность составляет необходимость модернизации метаданных и обеспечение работы с данными прошлых периодов в соответствующей схеме данных. Немаловажное значение имеет также сохранение связанности систем при изменении структур данных без дополнительной разработки, а лишь за счет соответствующей перенастройки связей. По сути, кроме жизненного цикла элемента данных и жизненного цикла его атрибутов, имеет смысл рассматривать еще два жизненных цикла метаданных: жизненный цикл классов (таблиц) и жизненный цикл атрибутов классов (столбцов таблиц). Для их описания требуются дополнительные таблицы, в которых также как и для отражения операций с элементами используются переходы.

На рисунке 3.1 изображена структура для представления данных по описанной технологии. Для связи записей и экономии места выделена дополнительная таблица транзакций, в которой хранятся транзакционное время чтения, транзакционное время записи, сведения об авторе и его электронная цифровая подпись. Это дополнительно

позволяет осуществлять ручной откат транзакций, а также в некоторой степени сократить место, требуемое для хранения информации о пользователях и ЭЦП. Все таблицы элементов объединены в единую таблицу, которая имеет поле, указывающее на класс элемента. Так же объединены и все таблицы значений в одну, которая имеет поля ссылки на таблицу элементов и таблицу атрибутов.

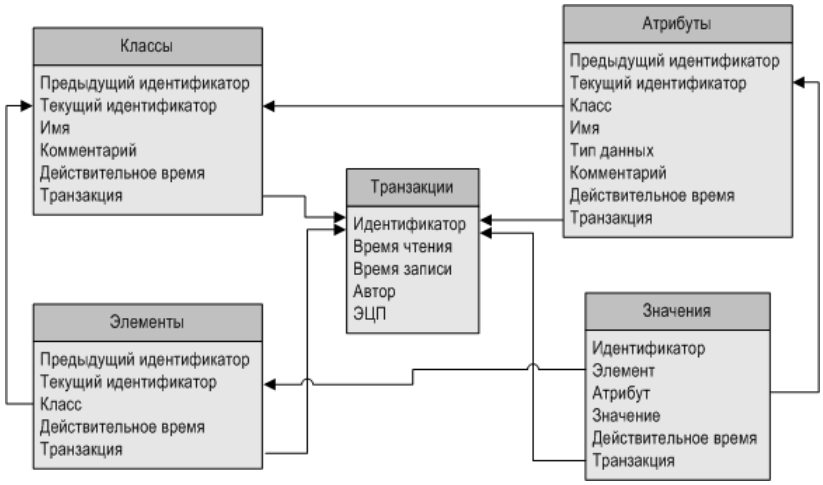

Рисунок **3.1**

4. Заключение

К преимуществам данной модели можно отнести значительное повышение ссылочной целостности, раздельное отражение операций изменения и исправления данных, возможность организации частичной защиты от проблем конкурентного доступа к данным, решение проблемы отслеживания источников вторичных данных, обеспечение отслеживания авторства и возможность формирования юридически значимого массива данных.

Темпоральная организация данных позволяет значительно упростить их совместное редактирование в нескольких информационных системах, а темпоральная организация метаданных – предоставить возможность мягкой модернизации структур данных и обеспечение работы с данными прошлых периодов в соответствующей схеме данных.

Описанный подход может быть применим при построении систем самого различного назначения. В первую очередь систем (подсистем) управления мастер-данными или нормативно-справочной информацией. В качестве частного примера, использование описанной технологии может решить множество проблем при построении одного из основных компонентов электронного правительства – системы реестров государственных услуг.

Список литературы

1. Snodgrass, Richard T. Developing Time-Oriented Database Applications in SQL. San Francisco, California : Morgan Kaufmann Publishers, 2000.

2. Jensen, Christian S. и Snodgrass, Richard Thomas. Semantics of Time-Varying Attributes and their Use for Temporal Database Design. Lecture Notes In Computer Science. 1995 г., Т. Vol. 1021.

3. Date, C. J., Darwen, Hugh и Lorentzos, Nikos A. Temporal Data and the Relational Model. San Francisco : Morgan Kaufmann Publishers, 2003.

Data management in enterprise information systems

© **S. N. Lizin**[3], **S. A. Fedosin**[4]

Abstract. DBMS and enterprise frameworks vendors compete for developing new functionalities of their products. But many problems, such as collaborate data processing or data lifecycle management, can be solved more simply and more effective using data management technologies.
Key Words: databases, data, temporal, transactional, DBMS, informational systems.

[3]Post graduate student of chair of ASIP&M, MSU after N. P. Ogarev, Saransk; sergey.lizin@gmail.com.
[4]Professor, Head of chair of ASIP&M, MSU after N. P. Ogarev, Saransk; sergey.lizin@gmail.com.