УДК 004.451.5

РЕАЛИЗАЦИЯ ПРЕИМУЩЕСТВ СЕРВЕРА СУБД ORACLE ДЛЯ ЭФФЕКТИВНОГО УПРАВЛЕНИЯ ИНФОРМАЦИОННОЙ СИСТЕМОЙ ВГУ

А. П. Толстобров, В. В. Фертиков

Воронежский государственный университет

Описаны технология разработки, инструменты создания и алгоритмическая реализация специализированных компонентов на сервере СУБД Oracle, организующих автоматическое и прозрачное для клиентских приложений сохранение истории модифицируемых данных, сопровождающей относительно быструю эволюцию предметной области во времени.

ВВЕДЕНИЕ

Проектирование корпоративной информационной системы (КИС) вуза требует учета специфики реализуемых системой функциональных процессов управления вузом. Важной задачей, решаемой распределенным программным комплексом информационной поддержки управления учебным процессом, является хранение и сопровождение эволюции предметной области. В рамках реляционной модели это означает сохранение временной последовательности всех состояний формализованных сущностей и связей между ними. Реализация механизмов, обеспечивающих автоматическое сохранение истории модифицируемых данных, разработана в Воронежском госуниверситете (ВГУ) с применением средств системы управления базой данных (СУБД) Oracle.

1. ОСОБЕННОСТИ ПОСТРОЕНИЯ КИС ДЛЯ ВГУ

Анализ показывает наличие как минимум двух видов эволюционирующих сущностей, различающихся временным масштабом или скоростью (частотой) модификации. Например, отношения схемы базы данных (БД), сохраняющие информацию о состоянии учебной деятельности отдельных студентов, их персональные анкетные данные, атрибуты договоров на обучение, платежей по ним, характеристики предметов договоров и т.п., можно считать быстро меняющимися данными, в темпе OLTP-транзакций. Каждое из таких отношений, как правило, сопровождается определенной прикладной подсистемой КИС, призванной решать конкретные задачи информационного обеспечения и автоматизации деятельности управленческих служб и подразделений (бухгалтерские и планово-финансовые задачи, учет кадров, управление учебным процессом, информационная поддержка руководства и др.). В то же время крупная КИС нуждается в представлении и обслуживании общих массивов данных справочного характера, совместно используемых не одной, а многими или всеми подсистемами. Это может быть, например, информация о структуре подразделений вуза, аудиторном фонде, номенклатуре образовательных программ, о различных нормативах, об ученых и степенных советах и т.п. Подобная информация модифицируется централизованно и, как правило, относительно редко (в темпе, сравнимом с DSS-транзакциями). КИС ВГУ обеспечивает механизмы сохранения эволюции обоих масштабов времени. При этом, несмотря на принципиальные различия в их реализации, оба они выполнены в виде специализированных надстроек над реляционной моделью средствами сервера СУБД Oracle.

Центральной частью университетской КИС является описанное авторами в [1] информационное хранилище базы данных (БД), которое обеспечивает унифицированное представление, централизованное хранение и обслуживание разнообразных данных, используемых разными специализированными подсистемами. Решение о централизации такого рода информационных служб приводит к необходимости отказа от учета специфики конкретных, использующих эти данные, задач и влечет за собой определенные потери эффективности. Поэтому основным аргументом в его пользу должен служить относительно невысокий темп эволюции централизо-

ванно хранимой информационной структуры. Унифицированное представление разнотипных данных обеспечивается реализацией хранилищем модели «сущность-связь» (совокупность структурных единиц, составляющих предметную область, и отношений между ними). Под структурными единицами подразумеваются самые разнообразные сущности: подразделения (университет, кафедры, факультеты, отделы...), должностные лица (ректор, проректор, декан факультета...), помещения (учебные корпуса, аудитории, лаборатории...) и т.д. Понятие «связь» инкапсулирует информацию о всевозможных бинарных отношениях между структурными единицами, например, «кафедра *входит в состав* факультета», «проректор *подчиняется* ректору», «аудитория *размещается* в корпусе» и т.д.

Основным преимуществом реализованного подхода является возможность модификации не только хранимых данных, но и самой их семантической структуры, логической схемы без вмешательства в организацию на уровне реляционной модели, т.е. без каких-либо изменений схемы базы данных. Понятно, что накладные расходы на локальное сопровождение таких ресурсов каждой отдельной подсистемой-потребителем оказались бы существенно выше по сравнению с их централизованным обслуживанием. В то же время очевидны преимущества, предлагаемые централизованным хранилищем, в котором процессы пополнения, использования и модификации данных непрерывны и совмещены во времени с процессами разработки структуры их хранения. В данном случае отказ от учета на уровне реляционной схемы специфических потребностей конкретных потребителей информации не приводит к упомянутой потере эффективности. Точнее, управление этой эффективностью становится доступным разработчику структуры путем соответствующей ее адаптации. Например, выявив узкие места структуры, возникшие из-за появления определенных запросов информации из конкретной подсистемы-потребителя, можно путем соответствующей доработки логической схемы (добавлением новых «абстрактных» сущностей и связей с ними) повысить эффективность выполнения запросов. Таким образом, централизация хранения и обслуживания медленно меняющейся информации сопровождается в КИС унификацией представления данных, гибкостью информационной структуры и интерфейса пользователя, а также повышенными требованиями к квалификации персонала.

В отличие от вышеизложенного, реализация механизмов, сопровождающих масштаб времени быстро меняющихся данных, должна обладать противоположными качествами. В данном случае отказ от учета специфики конкретных прикладных подсистем не представляется возможным. В то же время автоматизация сохранения последовательности состояний специфических формализованных сущностей выступает на первый план. Опыт долговременной эксплуатации КИС ВГУ говорит в пользу реализации специализированных серверных компонентов, организующих автоматическое сохранение истории модифицируемых данных, сопровождающей быструю эволюцию предметной области во времени. Одновременно с этим описываемые компоненты решают задачу аудита работы системы в дополнение к стандартным средствам СУБД, что повышает надежность КИС в целом.

Основная идея реализации механизма — хранение в одних и тех же таблицах БД вместе с данными, актуальными на текущий момент времени, всей последовательности состояний, описывающих сущности данных. Это позволило потаблично структурировать не только сами данные, но также и программные компоненты, обеспечивающие механизмы их эволюции: каждая подобная таблица обеспечивается типовым набором связанных между собой программных объектов БД (триггеров, хранимых процедур и пакетов PL/SQL Oracle). Важным достижением можно считать специальную технологию полуавтоматической генерации этих серверных компонентов, основанную на разработке универсальных шаблонов SQL-сценариев и применении CASE-средств для получения окончательных сценариев DDL, готовых для запуска на сервере Oracle с целью создания компонентов обслуживания конкретных таблиц БД. Невозможность в быстром масштабе времени унификации представления данных и применения высокоинтеллектуальных решений с участием человека успешно восполняется в данном случае автоматизацией разработки унифицированной программной надстройки.

Структурирование серверных компонентов дает еще одно преимущество: инструментарий надстройки устанавливается над определенным

набором таблиц, который определяется на этапе проектирования схемы БД. В этот набор входят либо наиболее ответственные таблицы, нуждающиеся в обеспечении дополнительными механизмами аудита за изменением их состояния (например, таблица платежей по договорам на обучение), либо реляционные отношения, соответствующие эволюционирующим сущностям предметной области. Например, установив компоненты надстройки над таблицей, сохраняющей информацию о состоянии учебной деятельности студентов, мы автоматически решаем задачу ведения электронного архива личных дел с учетом возможного прерывания процесса обучения, его завершения и последующего возобновления. Соответственно предусмотрены два режима обслуживания надстройкой таблиц, причем переключение режимов может осуществляться динамически под управлением клиентского приложения и независимо для каждой отдельно взятой таблицы. Режим аудита автоматически сохраняет всю последовательность редактирования данных вместе с некоторой служебной информацией (дата и время модификации записи, идентификатор пользователя БД-оператора, произведшего изменения, и т.д.). Режим сохранения истории модификации записей (т.е. «официальной» последовательности состояний), помимо перечисленного, требует внесения оператором информации о документальном обосновании такого редактирования (например, атрибутов приказа об изменении состояния учебной деятельности студента или другого документа, подтверждающего необходимость внесения изменений в личное дело).

Реализация описываемого механизма хранения последовательности состояний потребовала добавления в каждую из соответствующих таблиц четырех служебных полей:

1. Идентификатор кортежа (назовем его для краткости дальнейшего изложения ид_корт) — числовой первичный ключ таблицы.

2. Прежний идентификатор кортежа (далее — пр_ид_корт) содержит нуль для актуальной на текущий момент записи (последнего состояния или единственного состояния, если запись не редактировалась) или не равную нулю ссылку на актуальную запись для всех предыдущих состояний последовательности. Эта ссылка по абсолютной величине равна ид_корт актуального кортежа, при этом положительное значение она принимает для кортежей, сохраняющих историю модификации данных, отрицательное соответствует кортежам, сформированным в режиме аудита.

3. Время создания кортежа (обозначим вр_корт) фиксирует момент редактирования данных (дату и время на сервере базы данных).

4. Пользователь, создавший кортеж (польз_корт), служит для хранения серверного идентификатора пользователя базы данных, внесшего изменения.

Таким образом, процедура добавления нового состояния последовательности упрощенно состоит из следующих шагов:

1. Создать новый кортеж добавлением записи в таблицу. При этом ид_корт генерируется соответствующей последовательностью Oracle, запоминаются текущие значения в полях вр_корт и польз_корт, а в поле пр_ид_корт сохраняется ссылка на первичный ключ актуальной до данного момента записи (знак соответствует режиму обновления последовательности «история/аудит»).

2. Скопировать все поля данных актуального состояния (за исключением перечисленных служебных) во вновь созданную запись таблицы.

3. Произвести обычное обновление полей данных записи для получения кортежа нового актуального состояния.

Табл. 1 иллюстрирует описанную процедуру примером модификации некоторой последовательности состояний при помощи операции редактирования актуального кортежа: состояние <d> заменяется состоянием <e>. Следует обратить внимание на важную деталь такого способа хранения, решающую проблему ссылочной целостности данных: ид_корт актуального состояния в любом случае не меняется. Таким образом, значение этого поля можно считать уникальным идентификатором всей последовательности состояний (т.е. формализованной сущности в целом) и предусматривать реализацию ссылок на нее при помощи внешних ключей реляционной схемы базы данных. Заметим, что временной интервал актуальности в таблице БД не хранится, поскольку может быть получен несложным анализом поля вр_корт последовательности кортежей.

Описанная процедура реализована на сервере базы данных и активируется триггерами, связанными с операцией модификации соответствующей таблицы. Таким образом, она

Таблица 1

Пример операции пополнения последовательности состояний

ид_корт	пр_ид_корт	вр_корт	Интервал актуальности	Состояние
До операции редактирования				
1	0	понедельник	актуально с четверга	d
2	1	вторник	[понедельник — вторник)	a
3	–1	среда	[вторник — среда)	b
4	1	четверг	[среда — четверг)	c
После операции				
1	0	понедельник	актуально с пятницы	e
2	1	вторник	[понедельник — вторник)	a
3	–1	среда	[вторник — среда)	b
4	1	четверг	[среда — четверг)	c
5	1	пятница	[четверг — пятница)	d

становится прозрачной для клиентских приложений, оперирующих стандартными UPDATE SQL-запросами. Техническая реализация процедуры потребовала учета специфики средств СУБД Oracle (или стала возможной благодаря последним). Основным ограничением на подобные алгоритмы, как известно, выступает запрет обращения к изменяющимся (mutating) таблицам, в том случае, если требовать их выполнения в рамках одной транзакции. В данном случае это требование является абсолютно необходимым, поэтому при разработке использовалась известная технология преодоления означенной трудности: создание триггеров различных уровней (строкового и операторного) с организацией обмена данными между ними через переменные пакета PL/SQL. В результате шаг 1 становится последним во времени шагом описываемой процедуры: триггер уровня строки копирует поля актуального состояния в глобальную область памяти пользовательской сессии, после чего запись обновляется данными нового состояния (шаг 3) и, наконец, в операторном триггере создается новый кортеж с данными прежнего состояния, выбираемыми из памяти (шаг 1).

Все таблицы, сопровождаемые надстройкой, секционированы средствами СУБД Oracle таким образом, что данные аудита, история и актуальные на текущий момент данные хранятся в разных секциях. Такая структура существенно повышает эффективность большинства выборок информации, поскольку они, как правило, ограничиваются лишь актуальными данными, и остальные секции при их исполнении не задействуются. В то же время запросы на выборку данных всех состояний одновременно формулируются еще более компактно, поскольку они по-прежнему адресуются к одной таблице, а дополнительных ограничений (пр_ид_корт = 0) не имеют. Примером подобного запроса к единственной таблице персональных данных является поиск студента, сменившего фамилию, независимо от того, старая или новая фамилия служит ключевым словом поиска. Помимо этого, хранение каждой из секций производится в различных табличных пространствах Oracle и на различных физических носителях, что способствует повышению надежности и эффективности КИС.

Заметим, что использование специфики возможностей СУБД Oracle, таких как секционирование таблиц, обмен данными через переменные пакета PL/SQL между триггерами различных уровней и некоторых других, призвано оптимизировать прежде всего автономную работу серверной надстройки. Однако эти же особенности вызывают определенные трудности при реализации компонентов КИС, дополняющих возможности автономного режима, например, клиентских программ для решения задачи просмотра и редактирования последовательности автоматически зафиксированных состояний формализуемых сущностей. Возникновение необходимости каких-либо дополнительных манипуляций состояниями можно рассматривать в качестве исключительной ситуации в КИС. Например, такого рода ситуациии, когда приказы об изменении состояния учебной деятельности студента (перевод с курса на курс, на другую специальность, о выдаче диплома и т.п.) из-за задержки их рассылки требуется обрабатывать не в

надлежащем порядке. Соответственно реализован ряд серверных компонентов поддержки клиентских приложений для манипуляции последовательностями состояний: операции редактирования, либо актуализации предыдущих состояний, изменения порядка кортежей в последовательности, вставки «задним числом» новых состояний внутрь существующей последовательности и т.п. Однако наиболее серьезные проблемы связаны, как правило, с неизбежными для любой КИС процедурами импорта данных. Например, персональные анкетные данные, наполняющие централизованное отношение БД из различных источников КИС (конвертирование из подсистем «Абитуриент», «Аспирантура», ручной ввод данных и т.д.), могут быть не синхронизированы, в результате чего в базе данных образуется сразу несколько сущностей «Персона», отвечающих одной реальной личности, каждая из которых, в свою очередь, сопровождается своей историей развития в виде сохраненной последовательности состояний. В связи с этим наиболее важной дополнительной операцией манипулирования данными о состояниях можно считать слияние нескольких эволюционирующих сущностей в одну с образованием единой результирующей последовательности состояний. Реализованный для этих целей алгоритм позволяет учесть как требования логического уровня схемы БД, так и тонкости ее технической реализации на сервере СУБД Oracle. То есть, с одной стороны, формируемая последовательность сохраняет хронологию и данные об интервалах актуальности состояний, одновременно предоставляя возможность выборки информации аудита. С другой стороны, технический уровень диктует специфику алгоритма, в частности связанную с необходимостью переноса записей между секциями таблиц (например, актуальное состояние одной из объединяемых последовательностей должно стать историей, что влечет за собой упомянутый межсекционный перенос соответствующей записи). Предлагаемый алгоритм минимизирует количество таких переносов. Помимо прочего, операция слияния истории инициирует серию каскадного обновления дочерних таблиц по автоматически формируемым сценариям, что позволяет избежать большинства исключительных ситуаций на сервере СУБД.

Многие детали разработанного алгоритма демонстрируют пример операции объединения двух последовательностей, показанный в табл. 2. Здесь римские цифры обозначают определенные поля таблицы базы данных или вычисляемые параметры кортежей: I — идентификатор кортежа (поле ид_корт), либо время создания кортежа (поле вр_корт) (для целей иллюстрации не имеет значения выбор конкретного из этих двух полей, поскольку оба они содержат уникальные значения и упорядочены, первичный ключ ид_корт генерируется последовательностью Oracle); II — прежний идентификатор кортежа (поле пр_ид_корт); III — вычисляемый интервал актуальности состояния (в БД не хранится); IV — совокупность полей данных, характеризующих состояние; VII — поле польз_корт.

Для выявления очередности состояний объединяемой последовательности производятся шаги 1—3, не требующие манипуляций с таблицей БД: вся процедура реализована в пакете PL/SQL. Лишь на шаге 4 формируемая последовательность сохраняется в таблице. Заметим, что эта операция не требует генерации новых значений первичного ключа ид_корт, благодаря тому, что общее число состояний (а, следовательно, и кортежей) не изменилось. При этом для расстановки записей в соответствии с очередностью делается серия SQL-запросов UPDATE для модификации полей ид_корт и вр_корт: столбец V для иллюстрации содержит прежнее значение этих полей, заменяемое значением в столбце I, столбец VI помечает записи, действительно требующие такой замены. Исключение из данной экономной процедуры составляют актуальные кортежи исходных последовательностей. Младший (более поздний) из них <e> может быть помещен в кортеж с ид_корт = 1 только UPDATE-запросом всех полей, кроме служебных (запись с ид_корт = 1 таблицы БД удалять нельзя из-за необходимости поддержки ссылочной целостности). Старший (ранний) из актуальных кортежей <a> должен быть перенесен в историю, для чего делается его вставка (INSERT-запрос) с измененными служебными полями (ид_корт = 7, пр_ид_корт = 1). Для обеспечения работоспособности алгоритма необходимо также предварительно удалить последнюю актуальную запись с ид_корт = 3. Заметим, что последовательность операций можно осуществить в порядке,

Таблица 2

Пример операции объединения последовательностей состояний

Исходно: 2 последовательности состояний								
Последовательность 1					Последовательность 2			
I	II	III	IV		I	II	III	IV
1	0	[6..)	a		3	0	[8..)	e
2	1	[1, 2)	b		4	3	[3, 4)	f
5	–1	[2, 5)	c		7	3	[4, 7)	g
6	1	[5, 6)	d		8	–3	[7, 8)	h
Шаг 1. «Прокрутка» кортежей для получения временной последовательности								
1	2	5	6		3	4	7	8
b	c	d	a		f	g	h	e
Шаг 2. Слияние последовательностей								
1	2	3	4	5	6	7	8	
b	c	d	g	d	a	h	e	
Шаг 3. «Прокрутка» результирующей последовательности								
1	2	3	4	5	6	7	8	
e	b	c	f	g	d	a	h	
Шаг 4. Запись в БД с использованием прежнего набора ид_корт								
I	II	III	IV	V	VI	VII		
1	0	[8..)	e	3	+	пользователь кортежа <a>		
2	1	[1, 2)	b	2		пользователь кортежа <b>		
3	–1	[2, 3)	c	5	+	пользователь кортежа <c>		
4	1	[3, 4)	f	4		пользователь кортежа <f>		
5	1	[4, 5)	g	7	+	пользователь кортежа <g>		
6	1	[5, 6)	d	6		пользователь кортежа <d>		
7	1	[6, 7)	a	1	+	текущий пользователь		
8	–1	[7, 8)	h	8		пользователь кортежа <h>		

определяемом столбцом I, т.к. новые значения ид_корт не превосходят заменяемые ($I \le V$), и кортежи могут либо оставаться на месте, либо перемещаться на место с тем ид_корт, которое уже освободилось на предыдущем шаге. Исключением является старший актуальный кортеж <a>, место для вставки которого, тем не менее, также освободится. Заметим, что все записи, за исключением соответствующей старшему из исходных актуальных состояний <a>, сохраняют свое положение в пределах секций таблиц (не изменяют своего статуса «актуально/история/аудит»). Таким образом, использование алгоритмом пары запросов DELETE, INSERT, оправдано еще и необходимостью данного межсекционного переноса записи.

Необходимо отметить особенности интерпретации результирующих служебных полей. Значения ид_корт и вр_корт конкретных кортежей могут измениться (столбец VI), что позволит как и раньше определять момент окончания их актуальности. Тем не менее, значение вр_корт не обязательно определит момент редактирования данной записи: актуальность могла завершиться из-за вступления в силу состояния из другой последовательности. Противоположное решение разработчиками алгоритма было принято относительно поля польз_корт: вовсех записях, кроме двух исходно актуальных (столбец VII табл. 2), его значение не меняется. В результате оно сохраняет идентификатор пользователя, изменившего именно данную запись. Причем, в отличие от обычного автоматического режима пополнения последовательности, этот идентификатор необязательно будет определять пользователя, создавшего следующее состояние, поскольку это состояние могло прийти из другой последовательности. Тем не менее, наиболее логичными являются следующие действия: в последнем (младшем) актуальном кортеже сохраняется значение поля польз_корт из первого (старшего) кортежа <a>, тем самым фиксируется пользователь, начавший результирующую последовательность,

образовав ее первое состояние. В свою очередь, в первом (старшем) актуальном кортеже, состояние которого стало историей, сохраняется пользователь текущей сессии для фиксации факта его работы.

ЗАКЛЮЧЕНИЕ

Таким образом, описанная технология разработки и алгоритмическая реализация специализированных компонентов на сервере СУБД Oracle позволяют организовать автоматическое и прозрачное для клиентских приложений сохранение истории модифицируемых данных, сопровождающей относительно быструю эволюцию сущностей предметной области во времени. При этом сохранение истории модифицируемых данных обеспечивается как в режиме аудита операций по модификации данных, так и в режиме фиксирования документально подтвержденных событий по изменению состояния сущностей предметной области.

СПИСОК ЛИТЕРАТУРЫ

1. *Фертиков В.В.* Структурированное хранилище эволюционирующих данных и комплекс программных средств его эксплуатации / В. В. Фертиков, А. П. Толстобров //Вестник Воронеж. гос. ун-та, Сер. : Системный анализ и информационные технологии. — 2006. — № 1. — С. 150—158.