ста и редактор контекста многокомпонентности, используемый администратором МИС.

Кроме системных модулей, были модифицированы модули, используемые пользователями для работы с данными: редактор титульных листов медицинских карт, форма регистрации переводов пациентов по отделениям.

Для подключения новой функциональности были откорректированы статистические отчеты, позволяющие получать данные о работе конкретной компоненты комплексного ЛПУ и комплексного ЛПУ в целом.

Возможности механизма поддержки многокомпонентности

Созданный механизм поддержки совместной работы МИС в мультипликативных структурах ЛПУ характеризуется следующими отличиями.

Имеется контекст механизма многокомпонентности. Для каждого пользователя хранится набор переменных, используемых механизмом поддержки многокомпонентности. Каждая переменная хранится в двух экземплярах: значение, проставленное администратором, и значение, заданное пользователем в процессе работы. Благодаря этому у администратора всегда есть возможность восстановить значение любой переменной контекста.

Имеется системный механизм поддержки многокомпонентности. На сервере БД реализован механизм фильтрации данных, выдаваемых пользователю по его запросу. Параметры ограничения данных хранятся в контексте каждого пользователя. При вычитывании параметров проверяется возможность смены значения переменной пользователем. Если пользователю не разрешено менять значение, берется значение переменной, заданное администратором системы.

Имеется редактор переменных контекстов пользователей. У администратора есть возможность управления контекстом пользователей.

В клиентских модулях предоставлена возможность динамического управления настройками пользовательского контекста механизма многокомпонентности. При задании значения переменной контекста в редакторе эти изменения сразу вступают в силу, и пользователь без дополнительных действий видит другой набор данных, ограниченный новыми настройками системы.

Литература

1. Гулиев Я.И., Комаров С.И. Интегрированная распределенная информационная система крупного лечебно-диагностического учреждения // Стратегии здоровья: информационные технологии и интеллектуальное обеспечение медицины-97: тез. докл. IV междунар. форума. М., 1997.

2. Никитина Галина. Механизм виртуальных частных баз данных в СУБД Oracle. URL: http://www.oracle.com/ru/oramag/octnov2002/easy_vpd.html (дата обращения: 01.12.2008).

3. The Virtual Private Database in Oracle9iR2. Understanding Oracle9i Security for Service Providers. An Oracle Technical White Paper. January 2002.

4. Сайт Исследовательского центра медицинской информатики Института программных систем РАН. URL: http://www.interin.ru (дата обращения: 01.12.2008).

5. Сайт корпорации Oracle, разработчика СУБД Oracle. URL: http://www.oracle.com (дата обращения: 01.12.2008).

РАЗРАБОТКА ТЕМПОРАЛЬНОЙ МОДЕЛИ ДАННЫХ В МЕДИЦИНСКОЙ ИНФОРМАЦИОННОЙ СИСТЕМЕ

А.Н. Базаркин (ИПС им. А.К. Айламазяна РАН, г. Переславль-Залесский, bugs@interin.ru)

В статье исследуются основные методы построения темпоральных моделей данных в реляционных СУБД. Приводится несколько критериев классификации методов их построения, а также обобщается опыт реализации одной из темпоральных моделей в интегрированной подсистеме МИС Интерин PROMIS.

Ключевые слова: *темпоральность, историчность, темпоральная модель данных, темпоральные БД, здравоохранение, медицинские информационные системы, лечебно-профилактические учреждения.*

В отличие от традиционных моделей данных, обеспечивающих хранение лишь мгновенного снимка объектов предметной области, темпоральные модели данных позволяют хранить информацию об эволюции объектов: для любого объекта, который был создан в момент времени **T1** и закончил свое существование в момент времени **T2**, в БД будут сохранены все его состояния на временном интервале **[T1, T2]**.

Под *темпоральностью* объекта следует понимать явную или неявную связь объекта с определенными датами или промежутками времени. В самом широком смысле темпоральные данные – это данные, которые могут изменяться с течением времени.

На рынке коммерческих БД отсутствуют СУБД, обладающие полноценными темпоральными возможностями.

Для их реализации в рамках *информационной системы* (ИС) программистам, как правило, приходится разрабатывать специальные средства, расширяющие и дополняющие существующие реляционные модели. Весьма распространенной проблемой разработки таких приложений является

отсутствие полного понимания того, каким образом и на каком уровне должна поддерживаться темпоральность в БД.

В соответствии с устоявшимися понятиями *темпоральная модель данных* (ТМД) – это модель данных, ориентированная на хранение темпоральных данных, все аспекты которой также должны быть темпоральными. Традиционная модель данных – **M=(DS, OP, C)** – состоит из трех компонент: структура данных **DS**, операции **OP** и ограничения целостности **C**, а темпоральная – **MT=(DST, OPT, CT)** – должна поддерживать все понятия, входящие в каждый из трех компонент, с учетом изменений данных во времени [1]. Структура данных должна быть адаптирована таким образом, чтобы имелась возможность хранить темпоральные данные. Алгебру и операции модификации следует переопределить, используя темпоральную семантику. Дополнительно для каждого ограничения целостности в нетемпоральной модели данных **M** темпоральная – **MT** – должна поддерживать темпоральный аналог нетемпорального ограничения. Семантика темпоральных ограничений целостности также должна быть переопределена.

Таким образом, разработка ТМД предполагает развитие следующих темпоральных составляющих: структура данных, ограничения целостности и ключи, запросы и модификации, алгебра.

Если рассматривать данные, представленные в БД, как некое отражение текущего состояния действительности для моделируемого мира, каждая запись может быть воспринята как факт, являющийся истинным в определенный момент или интервал времени. При переходе к темпоральной БД для каждого факта можно указать тот промежуток времени, в который этот факт являлся истинным в моделируемом мире, представленном в БД. Подобное представление времени, когда с данными связывается промежуток времени их актуальности (с точки зрения моделируемого мира), называется модельным, или действительным (*valid*) временем.

Другим типом линии времени, который рассматривается исследователями темпоральных БД, является транзакционное время. В любой СУБД каждой записи БД можно сопоставить тот промежуток времени, когда данная запись была представлена в БД, то есть промежуток времени между моментами добавления записи и ее удаления из БД.

Исследователями выделяются три фундаментальных типа темпоральных данных [1]:

- момент времени (*instant*) (событие, которое произошло или произойдет в определенный момент, например, сейчас или 1 августа 2009 года в 13.40);
- интервал времени (*interval*) (длительность временного отрезка, например 2 года);
- период времени (*period*) (конкретный отрезок времени, допустим, с 23 апреля 2007 года по 1 августа 2009 года).

Битемпоральная модель данных оперирует как модельным, так и транзакционным временем. Именно битемпоральная модель наиболее востребована в большинстве ИС.

Методы представления данных

Модифицирование реляционной модели данных для обеспечения поддержки работы с темпоральными данными предполагает изменения модели на уровне СУБД. Однако устройство большинства СУБД – это *черный ящик*, изменения в котором не представляются возможными. Поэтому основные способы обеспечения поддержки темпоральных данных заключаются в поддержке темпоральной функциональности на уровне приложения либо в расширении реляционной модели данных до темпоральной.

На практике существуют два принципиальных подхода к реализации ТМД: реализация темпоральной поддержки на уровне приложения и расширение нетемпоральной модели данных до темпоральной.

В литературе встречаются и другие способы, такие как генерализация модели данных до темпоральной и использование абстрактных типов, но на практике использование этих подходов сопряжено с определенными сложностями [2]. Метод реализации темпоральности на уровне приложения предполагает разработку специальных средств поддержки темпоральности на уровне приложения. Однако на практике данный подход приводит к существенным проблемам, например, когда требуется изменить или заменить часть кода в приложении. Темпоральная семантика в таком случае проектируется каждым разработчиком заново. Темпоральная логика, реализованная на уровне приложения, может быть удобным сиюминутным решением, но не дальновидной стратегией проектирования ИС.

Расширение нетемпоральной модели данных до ТМД означает, что для спецификации темпоральных понятий используются основные концепции, поддерживаемые нетемпоральной моделью данных. Язык запросов и алгебра расширяются дополнительными операциями для того, чтобы иметь возможность описывать темпоральные операции с данными.

На практике этот подход расширения схемы данных наиболее широко используется для построения ТМД. Его преимущество состоит в том, что данный метод предполагает изменение лишь отдельных частей модели, например, языка запросов или ограничения целостности. Метод доступа к информации и структура данных остаются неизменными.

В рамках данного подхода предложены различные ТМД. Критерии принципиальных отличий этих моделей друг от друга следующие:

- тип темпоральных данных (дискретное или интервальное представление времени);
- обеспечение темпоральности на уровне отдельных атрибутов или на уровне кортежа.

Темпоральные данные могут быть связаны как с дискретным представлением времени – моментом, так и с интервальным. Преимущество модели, основанной на дискретном представлении, заключается в ее простоте с точки зрения поддержки стандарта *SQL-92*. Однако связь темпоральных объектов с одним атрибутом времени может усложнить и без того непростые темпоральные запросы и операции. В этом плане проще в реализации модель с интервальным представлением времени.

Одним из недостатков этого подхода является отсутствие поддержки понятия интервала в стандарте *SQL-92*, оно может быть смоделировано использованием двух моментов времени.

Второй критерий построения ТМД определяет следующие пять подходов к представлению темпоральных данных [2].

1. Модель представления темпоральных данных, предложенная Р. Снодграсом.

Пусть битемпоральное отношение **R** имеет набор атрибутов **(A1, ..., An, T)**, где **T** – битемпоральный атрибут, определенный на множестве битемпоральных элементов. Тогда **R** можно записать в следующем виде: **R=(A1, …, An, Ts, Te, Vs, Ve)**.

Дополнительные атрибуты **Ts**, **Te**, **Vs**, **Ve** – это атомарные темпоральные атрибуты времени, содержащие дату начала и окончания транзакционного и модельного времени. Данное представление отношений является самым естественным и наиболее часто используемым способом представления битемпоральных отношений.

2. Модель представления темпоральных данных, предложенная К. Дженсеном.

Особенность данного представления в том, что историчные кортежи никогда не обновляются, то есть доступны только для чтения. Таким образом, это представление данных хорошо подходит для основанного на архивах хранения битемпоральных отношений. Битемпоральное отношение **R** с набором атрибутов **A1, …, An** может быть представлено в следующем виде: **R=(A1, …, An, Vs, Ve, T, Op)**.

Как и в предыдущей схеме представления данных, атрибуты **Vs** и **Ve** хранят даты начала и окончания актуальности факта в моделируемой реальности соответственно. Атрибут **T** хранит информацию о времени внесения кортежа в журнал изменений. Запросы на создание и удаление кортежей обозначаются в атрибуте **Op** соответствующими символами – **I** (*Insert*) и **D** (*Delete*). Модификация данных представляет собой пару запросов – удаление и создание записи – с одинаковым атрибутом времени **T**.

3. Модель представления темпоральных данных, предложенная С. Гадией.

Данный подход предполагает наличие битемпоральных меток у каждого из атрибутов кортежа, что обеспечивает возможность более гибкого моделирования реальности. Пусть битемпоральное отношение **R** имеет атрибуты **(A1, ..., An, T)**, где **T** – темпоральный атрибут, определенный на множестве битемпоральных элементов. Тогда битемпоральное отношение **R** может быть представлено в виде отношений, где каждый из атрибутов имеет свою темпоральную метку: **R=({([Ts, Te] [Vs, Ve] A1)},…,{ ([Ts, Te] [Vs, Ve] An)})**.

Кортеж состоит из **n** элементов. Каждый элемент представляет собой тройку значений: транзакционное время **[Ts, Te]**, модельное время **[Vs, Ve]** и значение атрибута **Ai**.

4. Модель представления темпоральных данных, предложенная Е. МакКензи.

В данной модели битемпоральное отношение – это последовательность состояний в модельном времени, проиндексированная транзакционным временем. В кортежах с модельным временем атрибуты имеют свои темпоральные метки. Битемпоральное отношение **R** с набором атрибутов **A1, …, An** может быть представлено в виде отношения, в котором каждый атрибут помечается временной меткой**: R=(VR, T)**, где **VR** – отношение в модельном времени; **T** – транзакционное время. Схема состояний отношения модельного времени имеет вид: **VR=(A1V1, ..., AnVn)**, где **A1, …, An** – набор атрибутов; **Vi** – атрибут модельного времени, связанный с каждым атрибутом **Ai** и обозначающий время актуальности значения атрибута **Ai** в моделируемой реальности.

5. Модель представления темпоральных данных, предложенная Дж. Бен-Зви.

Пусть битемпоральное отношение **R** состоит из набора атрибутов **(A1, ..., An, T)**, где **T** – темпоральный атрибут, определенный на множестве битемпоральных элементов. Тогда **R** может быть представлено в модели Бен-Зви следующим образом: **R=(A1, …, An, Tes, Trs, Tee, Tre, Td)**.

В кортеже значение атрибута **Tes** (*effective start*) – это время, когда значение атрибута кортежа становится актуальным. Атрибут **Trs** хранит информацию о том, когда **Tes** было сохранено в БД. Аналогично **Tre** хранит информацию о том, когда факт перестает быть актуальным в моделируемой реальности, а **Tee** – когда **Tre** было зафиксировано в БД. Атрибут **Td** указывает на время, когда запись была логически удалена из БД.

Кроме этого, темпоральные модели данных могут отличаться дополнительными критериями, такими как возможность работы с ошибочно введенными данными. Существует ряд методов, предложенных отечественными авторами, суть

которых сводится к расширению традиционной модели до темпоральной посредством введения дополнительных таблиц-связей [3]. Авторы доказывают жизнеспособность и пригодность данных методов на примере разработки прикладного программного обеспечения, однако реализованные ими темпоральные возможности не являются полноценными с точки зрения определения ТМД.

Реализация темпоральной модели

Опишем реализацию ТМД в *медицинской информационной системе* (МИС) Интерин *PROMIS*. Система представляет собой информационную и функциональную среду, объединяющую элементы различных классов. МИС обеспечивает комплексную автоматизацию и информационную поддержку всех служб медицинского учреждения. Важное место в ней занимает подсистема управления персоналом, эффективная работа которой является необходимым условием нормального функционирования учреждения [4]. Подсистема управления персоналом предназначена для автоматизации работы с кадровым составом учреждения, в ее функции входят сквозное ведение штатного расписания, а также оформление и проведение в системы приказов по кадрам на дату как в прошлом, так и в будущем [5]. Реализация темпоральности в данной подсистеме имеет ряд системотехнических сложностей, для решения которых потребовалось принятие научно-обоснованных решений архитектурного и методологического характера.

Опишем некоторые моменты реализации ТМД за счет расширения существующей реляционной модели. Автором сознательно выбраны наиболее важные аспекты реализации темпоральности, которые, по его мнению, заслуживают наибольшего внимания.

Выбор метода. В качестве основной модели структуры темпоральных данных выбрана модель, предложенная Р. Снодграсом и реализующая темпоральные характеристики на уровне кортежа. Данный метод выбран из-за наибольшей естественности и простой реализации по сравнению с другими [2].

Модель данных подсистемы управления кадрами представляет собой набор более чем из тридцати таблиц, двенадцать из которых темпоральные. Рассмотрим ТМД на примере четырех таблиц: K_PERSONS, K_ISPOLS, K_DOLS и K_DOL_DICT. Таблица K_PERSONS содержит тридцать четыре поля, для примера возьмем только некоторые из них: таблицу K_ISPOLS, состоящую из двадцати восьми полей, включая PERSON_ID и DOL_ID; таблицу K_DOLS, содержащую восемнадцать полей, включая DOL_ID и DOL_DICT_ID; таблицу K_DOL_DICT, в которой два поля.

Таким образом, рассмотрим следующие таблицы и атрибуты:

- K_PERSONS (PERSON_ID, FAMILY, NAME);
- K_ISPOLS (PERSON_ID, DOL_ID, SALARY);
- K_DOLS (DOL_ID, DOL_DICT_ID);
- K_DOL_DICT (DOL_DICT_ID, DOL_NAME).

Таблица K_PERSONS содержит информацию о сотрудниках организации; K_DOLS – список должностей организации; K_ISPOLS – информацию о том, какую должность занимает сотрудник; K_DOL_DICT – это справочник должностей. PERSON_ID – *первичный ключ* (ПК) таблицы K_PERSONS, ПК таблицы K_ISPOLS состоит из пары атрибутов (PERSON_ID, DOL_ID), ПК таблицы K_DOLS – DOL_ID.

Добавление темпоральности. Для того чтобы ТМД стала битемпоральной, в таблицы были добавлены по четыре темпоральных атрибута. Первая пара атрибутов (ACTUAL_FROM, ACTUAL_TO) отражает период актуальности информации в моделируемой реальности (модельное время), а вторая пара (IN_DATE, OUT_DATE) – время фактической регистрации факта в БД и время его логического удаления (транзакционное время).

Темпоральные ключи. Для таблиц с темпоральной поддержкой требуется изменить состав первичных ключей – в них необходимо включить темпоральные атрибуты. Значение первичного ключа таблицы должно быть уникальным. Для оригинальной таблицы K_ISPOLS значение пары (PERSON_ID, DOL_ID) уникально в любой момент. Это означает, что ни один из сотрудников не может числиться более чем на одной должности (речь идет об основных видах исполнения). С добавлением темпоральной поддержки в данной таблице пары значений (PERSON_ID, DOL_ID) могут повторяться. Добавление темпорального атрибута ACTUAL_FROM или ACTUAL_TO в состав ключа не решает проблемы темпоральных ключей.

Проблема остается в том случае, когда даты модельного времени отличаются, например, на один день, то есть периоды актуальности данных пересекаются, что приводит к неуникальности значений первичного ключа на некоторых промежутках времени. Для решения этой задачи была разработана более сложная конструкция – последовательное (*sequenced*) условие уникальности в каждый момент времени:

```
WHERE NOT EXISTS (SELECT *
        FROM K_ISPOLS I1,
             K_ISPOL I2
          WHERE I1.PERSON_ID = I2.PERSON_ID
          AND I1.DOL_ID = I2.DOL_ID
          AND I1.ACTUAL_FROM < I2.ACTUAL_TO
          AND I2.ACTUAL_FROM < I1.ACTUAL_TO
          AND I1.rowid <> I2.rowid )
          AND NOT EXISTS (
                SELECT *
                    FROM K_ISPOLS I1
                 WHERE I1.K_PERSON_ID IS NULL OR
                           I1.DOL_ID IS NULL)
```

Темпоральная уникальность. Для оригинальной таблицы K_ISPOLS условие уникально-

сти может быть записано в виде UNIQUE (PERSON_ID, DOL_ID).

Однако с добавлением темпоральной поддержки в таблицы этого условия оказывается недостаточно. Недостаточным является и добавление одного темпорального атрибута или пары атрибутов UNIQUE (PERSON_ID, DOL_ID, ACTUAL_FROM, ACTUAL_TO), поскольку пара тех же значений (PERSON_ID, DOL_ID) может быть добавлена с датами, отличающимися, например, на один день. В этом случае условие уникальности выполняться не будет.

Для решения этих проблем были разработаны более сложные условия уникальности следующего вида:

```
WHERE NOT EXISTS (SELECT *
          FROM K_ISPOLS AS I1
         WHERE 1 < (SELECT COUNT(PERSON_ID)
                      FROM K_ISPOLS AS I2
                     WHERE I1.PERSON_ID = I2.PERSON_ID
                       AND I1.DOL_ID = I2.DOL_ID
                       AND I1.ACTUAL_FROM <= CURRENT_DATE
                       AND CURRENT_DATE < I1.ACTUAL_TO
                       AND I2.ACTUAL_FROM <= CURRENT_DATE
                       AND CURRENT_DATE < I2.ACTUAL_TO))
```

Темпоральные ограничения целостности. Добавление темпоральной поддержки порождает три возможных случая ограничения ссылочной целостности: ни одна из двух таблиц не темпоральна, одна из двух таблиц темпоральна, обе таблицы темпоральны.

Наиболее сложным случаем реализации условия ограничения целостности данных является третий случай, когда в обеих таблицах реализована темпоральная поддержка:

```
NOT EXISTS ( SELECT *
       FROM K_ISPOLS AS I
      WHERE NOT EXISTS (
             SELECT *
               FROM K_DOLS AS P
              WHERE I.DOL_ID = P.DOL_ID
                AND P.ACTUAL_FROM <= I.ACTUAL_FROM
                AND I.ACTUAL_FROM < P.ACTUAL_TO)
         OR NOT EXISTS (
             SELECT *
              FROM K_DOLS AS P
             WHERE I.DOL_ID = P.DOL_ID
               AND P.ACTUAL_FROM < I.ACTUAL_TO
               AND I.ACTUAL_TO <= P.ACTUAL_TO))
```

Темпоральные запросы. В теории темпоральных БД выделяют три фундаментальных типа запросов и модификаций: текущие (*current*), последовательные (*sequenced*), произвольные (*non sequenced*) [1].

Запросы к оригинальным таблицам до добавления темпоральной поддержки соответствуют текущему состоянию моделируемой реальности. Запросы к битемпоральным таблицам приобретают специфику. Рассмотрим более подробно различные типы запросов.

Текущий запрос представляет собой запрос значений на какой-нибудь момент в прошлом, то есть создание среза истинности фактов на произвольную дату.

Например, для обычного реляционного запроса «какую зарплату сейчас получает каждый из сотрудников?» можно легко сформулировать его темпорального двойника «какую зарплату получал каждый из сотрудников в указанную дату?». В этом случае результат запроса останется в рамках реляционного представления.

Более сложным случаем являются ***последовательные запросы***. Вполне естественным оказывается запрос «когда и какую зарплату получал каждый из сотрудников?». Здесь уже в результатах запроса появляется линия времени.

Алгоритм формирования результатов подобных запросов можно упрощенно представить следующим образом: для каждого момента вычисляется реляционный подзапрос «какую зарплату получает каждый из сотрудников?», после чего к общему результату добавляются результаты этих подзапросов с учетом интервалов истинности. Подобная семантика последовательной интерпретации реляционных запросов называется последовательной.

Для случая последовательных запросов рассмотрим более подробно особенности операций выборки и связывания.

Последовательная выборка данных не требует особых средств поддержки темпоральности и достаточно просто реализуется.

Более сложной задачей является связывание двух темпоральных таблиц. Например, для вычисления зарплаты и должности для каждого сотрудника нужно вычислить значение заработной платы для каждого промежутка времени.

История заработной платы хранится в таблице K_ISPOLS. Прямое связывание двух таблиц K_ISPOLS и K_PERSONS для каждого момента неэффективно, потому что значения заработных плат и информация о сотрудниках остаются неизменными в течение длительных периодов. Поэтому данные таблицы следует связывать с использованием их темпоральных периодов. Возможны два случая:

- период в первой из двух связываемых таблиц является более продолжительным, чем период во второй таблице;
- период во второй из двух связываемых таблиц является более продолжительным, чем период в первой таблице.

Таким образом, выборки темпоральных данных в случае, когда все таблицы имеют темпоральную поддержку, становятся более сложными и состоят, как правило, из нескольких подзапросов в зависимости от пересечений и наложений периодов темпоральности.

Так, например, запрос на выборку истории зарплат всех сотрудников потребует рассмотрения двух общих случаев наложения периодов таблиц в одном запросе.

Темпоральные модификации

Текущие модификации в общем случае предполагают обновление записи и изменение периода

актуальности с некоторого момента в прошлом и по настоящее время [1].

Операции создания записи в этом случае требуют дополнительных явных условий проверки в блоке WHERE на ограничения ссылочной целостности и уникальности.

В общем случае текущее удаление записи представляет собой обновление даты окончания периодов модельного и транзакционного времени.

Текущее обновление записи в общем случае состоит из следующей последовательности действий:

- создание новой записи с соответствующими битемпоральными атрибутами;
- обновление всех записей, дата актуальности которых больше даты начала действия созданной записи.

Последовательные модификации предполагают обновление записи в некоторый период в прошлом, то есть границы актуальности этого периода на оси модельного времени располагаются до настоящего момента [1].

Новая запись в случае последовательных модификаций создается при выполнении следующих условий:

- в этом периоде актуальности нет дубликатов записи;
- для этого периода есть актуальное значение в таблице, на которую ссылается новая запись;
- нет разрывов во временной оси модельного времени.

При удалении записи возможны четыре варианта пересечения периодов актуальности – *оригинального* (ОП) и *удаляемого* (УП): УП целиком входит в ОП, УП начинается в рамках ОП, УП заканчивается в рамках ОП и ОП целиком входит в УП. В общем случае удаление записи при последовательных модификациях состоит из следующих действий:

- копируется запись ОП, атрибут ACTUAL_FROM задается равным дате окончания УП;
- для этой записи атрибут ACTUAL_TO задается равным дате начала УП;
- для этой записи атрибут ACTUAL_FROM задается равным дате окончания УП;
- удаляются записи ОП, которые целиком входят в период УП.

В случае обновления записи также возможны четыре варианта взаимного расположения двух периодов – *оригинального* (ОП) и *модифицируемого* (МП). Обновление записи при последовательных модификациях представляет собой следующий набор операторов:

- копируется запись ОП, атрибут ACTUAL_TO задается равным дате начала МП;
- копируется запись ОП, атрибут ACTUAL_FROM задается равным дате окончания МП;
- обновляются необходимые атрибуты у тех записей, период актуальности которых пересекается с МП;
- атрибут ACTUAL_FROM задается равным дате начала МП для тех записей, период актуальности которых пересекает МП;
- атрибут ACTUAL_TO задается равным дате окончания МП для тех записей, период актуальности которых пересекает МП.

Запросы и модификации произвольного типа оперируют темпоральными данными произвольно, являются достаточно редкими и должны рассматриваться отдельно в конкретном случае. Например, к таким запросам можно отнести запросы, требующие сравнения нескольких последовательных моментов времени, обычно включающие агрегационные функции «во времени», например, «вывести среднюю заработную плату сотрудника за все периоды времени» [1]. Запросы данного типа при реализации ТМД в рамках системы управления кадрами МИС Интерин *PROMIS* не рассматривались.

В заключение следует отметить, что исследования в области темпоральных БД ведутся уже более трех десятилетий и до сих пор остаются актуальными. За это время было сформулировано множество методик построения темпоральных БД, предложено множество различных способов построения ТМД. Функциональные возможности ИС, разработанной на базе темпоральной БД, поднимаются на качественно новый уровень. Практически все данные, которыми оперируют ИС, являются темпоральными, то есть в той или иной мере связаны с динамикой изменения во времени. Корректная и оперативная работа с темпоральными данными приобретает особое значение в сфере медицинских информационных технологий, где качество информации в конечном счете может оказывать влияние на здоровье человека.

В данной работе сделан обзор основных темпоральных структур данных. На основе наиболее естественного и простого из них реализована ТМД в рамках подсистемы управления персоналом МИС Интерин *PROMIS*. Подсистема управления персоналом является интегрированной частью МИС Интерин *PROMIS*, где особенно востребованы темпоральные возможности. Изложенный подход построения темпоральной модели успешно реализован, а также апробирован в ЦКБ РАН.

Литература

1. Snodgrass R. Developing Time-Oriented Database Applications in SQL – Morgan Kaufmann Publishers, 1999.

2. Jensen C.S., Soo M.D., Snodgrass R.T. Unifying Temporal Data Models Via a Conceptual Model, Information Systems Vol. 19, No. 7, 1994, pp. 513–547.

3. Порай Д.С., Соловьев А.В., Корольков Г.В. Реализация концепции темпоральной базы данных средствами реляционной СУБД // Документооборот. Концепции и инструментарий: сб. науч. тр. / Ин-т системного анализа РАН; под ред. В.Л. Ар-

лазарова, Н.Е. Емельянова. М.: Едиториал УРСС, 2004. С. 92–109.

4. Назаренко Г.И., Гулиев Я.И., Ермаков Д.Е. Медицинские информационные системы: теория и практика; под ред. Г.И. Назаренко, Г.С. Осипова. М.: Физматлит, 2005. 320 с.

5. Базаркин А.Н., Хаткевич М.И., Хаткевич Ю.И. Подсистема управления кадрами в интегрированных медицинских информационных системах // Программные системы: теория и приложения: тр. Междунар. конф.: ИПС РАН, Переславль-Залесский, 2006. Т. 1. С. 113–124.

РАСПРЕДЕЛЕННЫЙ АРХИВ ИЗОБРАЖЕНИЙ ДИСТАНЦИОННОГО ЗОНДИРОВАНИЯ ЗЕМЛИ

(Работа выполнена при поддержке Программы № 1 фундаментальных исследований Президиума РАН «Проблемы создания национальной научной распределенной информационно-вычислительной среды на основе развития GRID-технологий и современных телекоммуникационных сетей» и проекта РФФИ 07-07-12038-офи)

А.А. Московский, к.х.н.; Е.О. Тютляева; А.Ю. Первин; Е.В. Шевчук
(ИПС им. А.К. Айламазяна РАН, г. Переславль-Залесский,
moskov@phys069b-2.chem.msu.ru, xgl@pereslavl.ru)

Описывается подход к созданию распределенного архива данных дистанционного зондирования Земли, реализующий концепцию активного хранилища. Интеграция параллельной файловой системы Lustre и системы автоматического динамического распараллеливания (T-Sim) обеспечивает обработку данных ДЗЗ непосредственно на узлах хранения, позволяя сократить расходы на пересылку данных по сети и повысить эффективность обработки.

Ключевые слова: *распределенный архив, активное хранилище, дистанционное зондирование Земли, кластерная файловая система, распределенная обработка данных.*

Дистанционное зондирование Земли (ДЗЗ) – область человеческой деятельности, тесно связанная с обработкой больших объемов информации при помощи компьютерной техники. При этом накапливаются архивы данных по наблюдениям от одного или нескольких приборов, установленных на искусственных спутниках Земли. Объем архивов может составлять несколько петабайт (10^{15} байт) информации.

Эффективное управление возрастающими объемами данных до сих пор остается значительной проблемой. Рост емкости хранилищ только обостряет проблему большой стоимости перемещения данных между обрабатывающими узлами и устройствами хранения. В последние годы были разработаны *файловые системы* (ФС) для решения проблемы управления данными в контексте высокопроизводительных вычислительных систем. Некоторые из таких ФС, например *Lustre* [1] и *PVFS*, используют выпускаемые промышленные серверы в качестве серверов ввода/вывода, то есть мест для хранения данных. Общая вычислительная мощность сотен и тысяч узлов хранения может быть очень значительной, но обычно она не используется, так как этим узлам отводится только роль хранилищ.

Один из подходов по снижению требований к пропускной способности между хранилищами и вычислительными устройствами и задействованию вычислительных мощностей устройств хранения – это приближение вычисления к местам хранения данных. Такой подход позиционируется как активные хранилища в контексте параллельных ФС. Обработка данных непосредственно на узлах хранения существенно снижает объемы передачи данных по сети и, следовательно, общесетевой трафик.

Активные хранилища нацелены на приложения с интенсивной стадией ввода/вывода, входные данные для которых можно разбить на независимые наборы. Они могут использоваться для обработки результатов моделирования в различных научных областях. Задачи обработки и хранения данных ДЗЗ имеют все свойства, позволяющие говорить о том, что реализация для их решения концепции активных хранилищ является прекрасным выбором.

Использование распределенных активных хранилищ позволяет создавать масштабируемые, высокоскоростные, управляемые распределенные системы с высокой пропускной способностью.

Предлагаем один из возможных подходов к организации активного хранилища, реализованный в программной системе для распределенного хранения и обработки данных ДЗЗ. Описываемая программная система объединяет возможности параллельной ФС *Lustre* и системы автоматического динамического распараллеливания (библиотеки *T-Sim* [2]), позволяя создать прототип распределенного архива изображений – активного хранилища, в котором данные обрабатываются на тех же узлах, где хранятся, и обеспечивается автоматическая балансировка нагрузки на узлах кластера.

Технология активных хранилищ в последние годы приобрела особую популярность, что свидетельствует о ее актуальности. Существуют схожие по задачам системы, такие как *ActiveStorage*, *Cascading* [3], *Pig* [4], *Hadoop* [5]. Большинство разработок базируются на какой-либо специализированной кластерной файловой системе, удобной для организации активного хранилища. Предложенный авторами подход не является исключением – данная разработка использует ФС *Lustre*. Отличительной особенностью подхода является наличие шаблонов задач для различных стратегий