DOI: 10.17117/ns.2014.02.060

Поступило в редакцию: 08.12.2014

<http://ucom.ru/doc/ns.2014.02.060.pdf>

vka@mil.ru

Котиков П.Е., Нечай А.А., Зацепин В.А. Темпоральные базы данных и язык запросов

Kotikov P.E., Nechay A.A., Zatsepin V.A.
Temporal database and query language

Статья посвящена проблемам теории баз данных. Представлен анализ развивающегося направления, - темпоральных (временных) баз данных. Темпоральные данные, – это сейчас уже отдельная и вполне самостоятельная область исследований. Нередко, многие темпоральные системы создаются лишь на основе общих методов проектирования и разработки приложений баз данных, без использования накопленных знаний в области создания систем управления темпоральными базами данных

Ключевые слова: база данных, темпоральная база данных, производственные данные, хранилище данных, язык запросов, неявная зависимость

The article is devoted to the theory of databases. Presents an analysis of the developing areas, and temporal (temporary) database. Temporal data is now a separate and independent area of research. Often, many temporal systems are created only on the basis of common methods for designing and developing database applications without the use of accumulated knowledge in the area of control systems, temporal databases

Key words: database, temporal database, process data, store data, the query language, the implicit dependence

Котиков Павел Егеньевич

Кандидат технических наук, доцент
Военно-космическая академия им. А.Ф. Можайского
г. Санкт-Петербург, ул. Ждановская, 13

Kotikov Pavel Egenievich

Candidate of Technical Sciences, associate Professor
Military Space Academy named A.F. Mozhaiskogo
St. Petersburg, Zhdanov st., 13

Нечай Александр Анатольевич

Преподаватель
Военно-космическая академия им. А.Ф. Можайского
г. Санкт-Петербург, ул. Ждановская, 13

Nechay Aleksandr Anatolyevich

Teacher
Military Space Academy named A.F. Mozhaiskogo
St. Petersburg, Zhdanov st., 13

Зацепин Вадим Александрович

Курсант
Военно-космическая академия им. А.Ф. Можайского
г. Санкт-Петербург, ул. Ждановская, 13

Zatsepin Vadim Aleksandrovich

Cadet
Military Space Academy named A.F. Mozhaiskogo
St. Petersburg, Zhdanov st., 13

Почти все данные, связаны с различными событиями или интервалами времени. Темпоральные данные, – это сейчас уже отдельная и вполне самостоятельная область исследований. Нередко, многие темпоральные системы создаются лишь на основе общих методов проектирования и разработки приложений баз данных, без использования накопленных знаний в области создания систем управления темпоральными базами данных.

Такой подход не только повышает стоимость разработки, но и может самым негативным образом сказаться на эффективности работы приложений и наличии в них ошибок.

Темпоральные данные, – это произвольные данные, которые явно или неявно связаны с определенными датами или промежутками времени.

Под такое определение попадают почти любые данные и информация. Например, даже если нет явной зависимости от времени для какой-нибудь аксиомы, то все равно для нее имеется неявная зависимость от времени, так как когда-то нам (или системе) стало известно, что такая аксиома существует. Кроме того, есть вероятность, что в будущем аксиома будет опровергнута, или на условия ее применимости будут наложены определенные ограничения; поэтому нельзя уже будет рассматривать ее как некоторую абсолютную истину, верную во всех ситуациях и в любой момент времени. [1,с.1]

Темпоральные базы данных – это базы данных, хранящие темпоральные данные. Однако эти базы данных и содержащиеся в них данные могут рассматриваться как темпоральные не во всех случаях. Речь идет о правилах интерпретации временных меток и интервалов для конкретной системы управления базами данных (СУБД).

Необходимо понять, можно ли отдельно выделить и специальным образом интерпретировать данные атрибута «время». В категорию темпоральных СУБД не будут попадать обычные реляционные СУБД, в которых поддерживаются связанные со временем типы данных, но интерпретацией и связью данных (или событий) между собой с учетом времени приходится заниматься разработчику. [1,с.1].

Таким образом, темпоральной СУБД учитываются специфическая природа времени и изменчивость данных с течением времени.

Публикаций на русском языке, посвященные тематике темпоральных баз данных, очень мало. В содержательной, и обстоятельной статье по этому вопросу: Б.Б.Костенко и С.Д.Кузнецова «История и актуальные проблемы темпоральных баз данных» [1], подробно рассмотрены многие аспекты этой области исследования. В разделе 7 этого материала [1,с.3] дана характеристика предложений от производителей коммерческих СУБД. Это прототипы темпоральных СУБД, а также различные расширения и дополнения, позволяющие работать с темпоральными данными. Ценность этого материала совершенно очевидна в условиях ограниченной информации.[1,с.1-4].

Движение по пути создания, главным образом, расширений и дополнений подтверждает что, реляционная модель данных оказалась столь подходящей для хранения данных, обработки и представления результатов запросов, что и темпоральные базы данных с самого начала предполагалось основывать на существующей реляционной модели. При этом, темпоральное расширение остаётся лишь одним из дополнительных признаков хранимых данных.

Переходя от привычных решений к темпоральным больше всего интересуют два аспекта:

1. Как создавать?
2. Как изменится язык запросов?

В [1] отмечается, что почти все такие способы сводятся к созданию некоторого функционального блока, отвечающего за разбор темпоральных запросов, подмену их некоторыми реляционными вычислениями, а потом обратному преобразованию в требуемое темпоральное представление для возвращения результатов пользователю.

Основным отличием разных решений был уровень «вмешательства» в реляционную СУБД, а также степень интеллектуальности данного блока темпоральных преобразований (рис. 1). Сравнение различных вариантов реализации промежуточного слоя представлено, в [2].

Рис. 1. Общий вид многоуровневой архитектуры темпоральной СУБД

Рассмотрим три крайних случая:

- преобразование на уровне ядра реляционной СУБД,
- преобразование на уровне пользовательского приложения,
- использование независимого промежуточного прогоу-уровня.

Первый способ предоставляет максимум возможностей по расширению синтаксиса языка баз данных, обеспечению различных проверок, а также оптимизации. Он также обеспечивает полную прозрачность для всех пользовательских приложений, а возможно, и для администратора БД. Однако он доступен только разработчикам СУБД.

Второй способ. В отличие от простого использования приложением реляционной СУБД для работы с темпоральными данными второй способ предполагает выделение некоторых модулей или библиотеки (в пользовательском приложении), отвечающих за преобразование именно темпоральных запросов. То есть основное приложение использует некоторую темпоральную абстракцию, а не реляционную БД в чистом виде, а потом интерпретирует результаты запросов. Подобная абстракция (пусть и на уровне приложения) позволяет сократить число ошибок и отделить бизнес-логику приложения от технической составляющей хранения данных.

Третий способ подразумевает создание между пользовательским приложением и СУБД некоторого промежуточного «черного ящика» (будем называть его ргоху-уровнем), который может быть реализован в виде драйвера, сервиса, внешней библиотеки и т.п. Для пользовательского приложения этот ргоху-уровень должен работать как темпоральная СУБД. С другой стороны, для СУБД этот ргоху-уровень является обычным реляционным приложением. Поэтому ргоху-уровень оказывается прозрачным как для пользовательского приложения, так и для СУБД, а также не требует внесения изменений в их исходный код. Исходя из того, что между ргоху-уровнем и СУБД интенсивность обмена данными может оказаться на порядок выше, чем между ргоху-уровнем и приложением (с учетом различных дополнительных оптимизаций), предпочтительно размещать его как можно ближе к СУБД. [1, с. 2]

Преимущества и недостатки этих способов

1. Основным недостатком первых двух подходов является необходимость в изменении кода СУБД или приложения, и поэтому они доступны, в первую очередь, самим разработчикам.

2. Одним из недостатков второго и третьего способа является невозможность сильно влиять на внутреннее представление и хранение информации в СУБД, оптимизацию доступа к ней и т.п.

3. Одним из значительных преимуществ всех трех способов является возможность использовать уже существующие в СУБД модули интерпретации и обработки, лишь дополняя их разбором и преобразованием только темпоральных конструкций и элементов.

4. Дополнительным преимуществом третьего способа можно назвать относительную независимость от конкретной СУБД, если не используются какие-либо специфические особенности и конструкции. В большинстве случаев на работоспособность ргоху-уровня не влияет обновление версии СУБД, так как синтаксис SQL-запросов останется прежним.

В работе [1,с.2], (см. п.5.2.) есть ответ и по вопросу облика языка запросов для темпоральных СУБД.

Для работы с темпоральными базами данных необходимо было разработать язык запросов к ним, который должен был бы стать расширением языка запросов SQL к реляционным базам данных. Конструкции выборки были предложены авторами языка TSQL2 (SQL/Temporal).

Реляционная модель предполагает, что данные хранятся в таблицах, и база данных состоит из набора таких таблиц. Характерно, что любая база данных может быть названа темпоральной, поскольку неизбежно содержит данные, связанные с промежутками времени. СУБД будет темпоральной СУБД лишь в том случае, когда именно она непосредственно занимается интерпретацией темпоральных данных, то есть данных, связанных со временем.

В [1] отмечается, что под управлением темпоральной СУБД может быть как темпоральная база данных, так и обычная, и более того, для части таблиц темпоральная поддержка может быть включена, а для других таблиц – нет.

При построении модели языка запросов к темпоральной базе данных разработчики приследовали несколько целей.

Во-первых, при переходе к темпоральной модели желательно расширить все три компонента модели данных: структуру данных, операции и ограничения целостности.

Во-вторых, любая корректная конструкция в исходном языке запросов должна остаться корректной в новом языке, и семантика этих конструкций должна остаться прежней; например, результат, возвращаемый запросом, должен быть таким же. То есть необходимо обеспечить восходящую совместимость языка запросов и базы данных.

Кроме того, желательно обеспечить темпоральную восходящую совместимость: необходимо, чтобы после добавления в систему темпоральной поддержки все унаследованные конструкции продолжали работать так же, как и раньше.

Из этого следует, что все существующие приложения не должны почувствовать переход от обычной реляционной СУБД к темпоральной СУБД [3]. Более того, последующее добавление темпоральной поддержки в какую-нибудь конкретную таблицу не должно отразиться на корректности выполнения запросов. С другой стороны, подобное добавление должно позволить использовать темпоральные запросы в новых программах, заметно облегчая работу программистам и администраторам системы.

Рис. 2. Работа с таблицей без темпоральной поддержки

Рассмотрим таблицу, содержащую информацию о заработной плате сотрудников.

До добавления темпоральной поддержки таблица содержала только актуальную информацию (на текущий момент времени). В таком случае у приложения имеется возможность узнать текущее значение любых элементов данных, но предыдущее значение будет потеряно после любого изменения. После удаления записи вообще невозможно будет сказать, например, сколько получал данный сотрудник перед увольнением. Такая ситуация существует при использовании традиционной реляционной модели (рис. 2).

Предположим, что позже было принято решение о необходимости хранить историю значений заработной платы сотрудников.

Решить это можно было бы несколькими путями

1. *Первый способ.* Можно создать специальную таблицу с историей зарплаты, но это могло бы быть достаточно странным, так как уже есть таблица «заработных плат». Более того, пришлось бы модифицировать код приложения, чтобы при удалении и изменении значений происходило автоматическое обновление данных и во второй таблице.

2. *Второй способ* – добавление специальных столбцов, отражающих интервал актуальности конкретной записи, но при этом также потребуются доработка приложения, и логика многих запросов может стать не слишком очевидной.

3. *Наиболее естественным* было бы добавить для данной конкретной таблицы темпоральную поддержку. В этом случае все ранее разработанные запросы будут корректно работать (с текущим состоянием), но можно сформулировать новые темпоральные запросы, которые позволят узнать историю изменений (рис. 3).

Рис. 3. Работа реляционных приложений с темпоральной таблицей

Какие же запросы могут быть сформулированы к таблице с темпоральной поддержкой?

1. Это запрос значений на какой-нибудь момент времени 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.

2. Вполне естественным оказывается запрос «когда и какую зарплату получал каждый из сотрудников?». Здесь уже в результатах запроса появляется линия времени. Один из традиционных способов представления подобных результатов опирается на интервальное действительное время, то есть к обыч-

ному реляционному представлению результатов добавляется еще один столбец, содержащий интервалы истинности фактов.

Алгоритм формирования результатов подобных запросов можно упрощенно представить следующим образом: для каждого момента времени 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* (*sequenced valid semantics*), (см. рис. 4.)

Рис. 4. Обработка "последовательных" запросов к темпоральной таблице

Однако на основе этой семантики невозможно сформировать запрос, который требует «сравнения» нескольких последовательных моментов времени.

К таким запросам относится большинство запросов, включающих агрегационные функции «во времени», например, «вывести среднюю заработную плату сотрудника за все периоды времени» (отметим, что результат будет представлен обычной реляционной таблицей) или «вывести периоды, когда на оплату труда сотрудников уходила максимальная сумма». Предусмотреть все подобные типы запросов невозможно, и также нельзя ограничивать выразительные возможности языка, поэтому была предложена конструкция запросов произвольного доступа к темпоральным данным (*Non-Sequenced Queries*), которые предоставляют возможность самостоятельно сформулировать необходимый запрос, накладывая ограничения на системный темпоральный столбец. (См. рис. 5.).

Обстоятельное рассмотрение всех случаев в [1] показывает, что при наличии темпоральной поддержки в системе можно выделить четыре уровня использования запросов (для конкретной таблицы):

- 1) темпоральная поддержка для таблицы отсутствует;
- 2) использование реляционных запросов к таблице с темпоральной поддержкой;
- 3 использование последовательных запросов;
- 4 использование произвольных темпоральных запросов.

Рис. 5. Обработка "произвольных" запросов к темпоральной таблице

Отмечается, что подобная классификация верна как для таблиц с поддержкой действительного времени, так и для тех, для которых поддерживается транзакционное время.

Поскольку эти линии времени являются ортогональными и независимыми, в системе с темпоральной поддержкой **получается шестнадцать вариантов использования запросов на выборку.**

На основе запросов на выборку можно формулировать темпоральные ограничения целостности, которые будут проверяться при операциях обновления базы данных. Для операций модификации и удаления записей можно накладывать аналогичные ограничения в их разделах WHERE.

Обработка запросов на обновление для транзакционного и действительного времени будет принципиально отличаться. Например, в разделе SET невозможно задавать системные поля транзакционного времени. Поэтому полностью корректная реализация поддержки транзакционного времени возможна только при реализации темпоральных операций внутри ядра СУБД.

Список литературы:

1. Костенко Б.Б., Кузнецов С.Д. История и актуальные проблемы темпоральных баз данных. URL: <http://citforum.oldbank.com/database/articles/temporal/index.shtml>
2. Torp K., Jensen C.S., Snodgrass R.T. Stratum Approaches to Temporal DBMS Implementation. In Proceedings of the 1998 International Database Engineering and Applications Symposium, Cardiff, Wales, UK, July 1998.
3. Bair J., Böhlen M.H., Jensen C.S., Snodgrass R.T. Notions of Upward Compatibility of Temporal Query Languages. *Wirtschaftsinformatik*, 39 (1): 25–34, February 1997.
4. Нечай А.А. Выявление недеklarированных возможностей аппаратно-программного обеспечения // Экономика и социум. 2014. № 1-2 (10). С. 457-460.
5. Лохвицкий В.А., Калиниченко С.В., Нечай А.А. Подход к построению системы автоматизированной интеграции информации в базу данных для её своевременной актуализации // Мир современной науки. 2014. № 2 (24). С. 8-12.
6. Нечай А.А., Матвеев С.В., Сафонов В.М. Выбор и обоснование показателей эффективности решения задачи распределения объектов по средствам поражения // Мир современной науки. 2014. №2 (24). С. 13-16.

References:

1. Kostenko B.B., Kuznetsov, S.D. *History and actual problems of tempo Federal databases.*
URL: <http://citforum.oldbank.com/database/articles/temporal/index.shtml>
2. Torp K., Jensen C.S., Snodgrass R.T. *Stratum Approaches to Temporal DBMS Implementation. In Proceedings of the 1998 International Database Engineering and Applications Symposium, Cardiff, Wales, UK, July 1998.*
3. Bair J., Böhlen M.H., Jensen C.S., Snodgrass R.T. *Notions of Upward Compatibility of Temporal Query Languages. Wirtschaftsinformatik, 39 (1): 25-34, February 1997.*
4. Nechay A.A. *Detection of undeclared capabilities of the hardware-software // Economy and society. 2014. № 1-2 (10). Pp. 457-460.*
5. Lokhvitski B.A., Kalinichenko S.C., Nechay A.A. *Approach to building a system for automated integration of data into a database for its timely update of // the World of modern science. 2014. № 2 (24). Pp. 8-12.*
6. Nechay A.A., Matveev S.C., Safonov V.M. *Selection and justification of the efficiency of solving the problem of distribution of objects by means of destruction // the World of modern science. 2014. № 2 (24). Pp. 13-16.*

© 2014, Котиков П.Е., Нечай А.А., Зацепин В.А.
Темпоральные базы данных и язык запросов

© 2014, Kotikov P.E., Nechay A.A., Zatsepin V.A.
Temporal database and query language