УДК 004.652.4

Теоретические аспекты применения темпоральных реляционных баз данных для хранения социально-экономической информации

В статье рассматриваются теоретические аспекты применения темпоральных баз данных. В первой её части даётся одно из определений темпоральных БД, наиболее точно и полно отражающее их сущность; описываются принятые подходы к их организации и выделяются три подхода к созданию таких баз данных. Предлагается способ реализации темпорального расширения для существующих реляционных баз данных. Для этого предлагаются новые типы данных и операции над ними. Все описания приводятся с помощью абстрактных типов данных. В заключительной части статьи рассматривается вопрос применения темпоральных баз данных для хранения социально-экономической информации и возможность создания подобных баз данных для хранения результатов мониторинговых исследований.

Реляционная база дных, темпоральная база данных, АТД, PostgreSQL, операции.

Вадим Сергеевич
САФОНОВ
младший научный сотрудник
Института социально-экономического развития территорий РАН
VadimSafonov@gmail.com

Вся информация, с которой приходится работать, в той или иной степени имеет временную составляющую. Различие состоит лишь в том, что временная составляющая может как сохраняться в информационной системе, так и нет. Ещё несколько лет назад время хранилось только в тех системах, в которых неиспользование его приводило к искажению информации или к невозможности правильного восстановления последовательности событий. Такими системами являлись финансовые базы данных, системы хранения данных о научных экспериментах, т. е. все системы, в которых время выступает неотъемлемой частью информации, например, о состоянии исследуемой системы или объекта в некоторый момент времени или о времени совершения финансовой операции.

Малое распространение систем, накапливающих историческую информацию, вызвано, в первую очередь, тем фактом, что в большинстве случаев поставленные задачи успешно решались без сохране-

ния времени. И кроме того, такие системы являются требовательными к аппаратным ресурсам, прежде всего ёмкости носителей информации, что ещё совсем недавно было весьма актуальным.

Поэтому даже в системах, накапливающих информацию с временными атрибутами, темпоральной маркировкой снабжались только те объекты, которые без временных свойств утрачивали свои свойства. Например, если говорить о бухгалтерских системах, то информация о финансовых операциях будет иметь временную маркировку, тогда как информация о лице, совершившем операцию, будет храниться лишь в её текущем состоянии.

В последнее время интерес к системам, хранящим информацию, привязанную ко времени, усилился. Эти системы принято называть темпоральными. Темпоральные базы данных – это базы, хранящие данные, привязанные ко времени и имеющие средства управления такой информацией.

Главное отличие темпоральных систем управления базами данных (СУБД) от обычных реляционных СУБД заключается в том, что для любого объекта, который был создан в момент времени t1 и был удален в момент времени t2, сохраняются все его состояния в этом временном интервале [t1, t2], тогда как в обычной СУБД существует только текущее на конкретный момент времени состояние объекта. Таким образом, в темпоральной БД хранится история изменений состояний объекта и пользователь может получить информацию о состоянии записи в БД в любой момент времени из указанного промежутка.

В настоящее время можно выделить несколько подходов к реализации темпоральных свойств СУБД. Первый подход – это создание темпоральной СУБД с нуля, в этом случае все темпоральные свойства будут заложены в самом ядре системы управления базами данных. Хотя такие разработки и имеются, но говорить об их массовом использовании не приходится, т.к. на современном рынке господствуют нетемпоральные реляционные СУБД, а существующие темпоральные БД значительно уступают им по функциональности.

Есть также работы, в которых предлагается выполнять запросы к темпоральным БД на естественном языке, но и такой подход в ближайшее время не станет перспективным, поскольку лексические анализаторы далеки от совершенства и способны распознавать только ограниченное количество лексических конструкций. Тогда как запрос на естественном языке может быть сформулирован большим количеством способов.

Наиболее рационален третий подход – это расширение функциональности существующих СУБД за счёт создания некоторого дополнительного функционального блока, отвечающего за преобразование темпоральных запросов в реляционную форму и преобразование данных из формата хранения в структуру, удобную пользователю. При таком подходе основной проблемой является определение места размещения этого функционального блока.

Можно выделить три основных варианта размещения: в ядре СУБД, в пользовательском приложении и в виде некоторого промежуточного модуля.

Первый вариант плох тем, что он доступен только для разработчиков СУБД, хотя и предоставляет максимальную прозрачность данных для приложений и большие возможности по оптимизации запросов.

Второй способ более доступен для разработчиков приложений; темпоральные свойства доступны для приложения независимо от выбранной СУБД, в то же время такие свойства могут быть заложены только в новых версиях программ, но не в уже существующих. В настоящее время этот способ применяется чаще других.

При использовании третьего варианта созданное расширение прозрачно как для клиентской части, так и для серверной. Однако интенсивность обмена данными

между модулем и приложением значительно меньше, чем между модулем и СУБД, поэтому наиболее логично размещать данный промежуточный модуль как можно ближе к ядру СУБД.

На наш взгляд, наиболее рациональным является использование не одного какого-либо подхода, а объединение первого и третьего подходов, т.е. внедрение части модуля в ядро СУБД и применение промежуточного уровня. При этом можно получить поддержку целостности временных данных на уровне СУБД, а все преобразования запросов выполнять в независимом модуле. Поясним более подробно.

Так, темпоральные таблицы, в отличие от обычных реляционных таблиц, требуют использования двух дополнительных полей для хранения меток начала и конца периода. Это приводит к необходимости использования составных первичных ключей, т.е. выполнять связь таблиц приходится по трём и более полям.

По нашему мнению, рациональнее определить новый составной тип данных, содержащий некоторый уникальный идентификатор и две временные метки. В виде абстрактного типа данных (АТД) [2] новый темпоральный тип для работы со временем актуальности может быть описан следующим образом:

```
DATA_TYPE ValidTimeKey IS
OPERATIONS:
CONS: → ValidTimeKey(Key, Time_Begin, Time_End=NULL)
CLOSE: ValidTimeKey(Key,Time_Begin,NULL) →
ValidTimeKey(Key,Time_Begin, Time_End)
IS_OPEN: ValidTimeKey → bool
IS_LINKED: ValidTimeKey → bool
IS_INTERSECT: ValidTimeKey → bool
LESS_THAN: ValidTimeKey < ValidTimeKey → bool
LESS_THAN_EQ: ValidTimeKey <= ValidTimeKey → bool
LESS_EQ: ValidTimeKey == ValidTimeKey → bool
GREATER_THEN_EQ: ValidTimeKey >= ValidTimeKey → bool
GREATER_THEN: ValidTimeKey > ValidTimeKey → bool
GET_KEY: ValidTimeKey (Key,Time_Begin, Time_End) → Key
END ValidTimeKey
```

Для данного типа определены следующие операции: IS_OPEN – позволяет определить, открыт ли период; IS_LINKED – определяет наличие ссылок на данный столбец из других таблиц; CONS – конструктор типа; CLOSE – закрывает открытый период; IS_INTERSECT – проверяет пересекаемость периодов актуальности одного объекта; GET_KEY – получает числовую часть типа.

Следует заметить, что возможность определения новых типов заложена в текущей версии стандарта SQL, а применение для описания нового типа АТД позволяет легко реализовать новый тип данных применительно к любой современной реляционной СУБД, поддерживающей добавление новых типов.

Для экспериментов в текущей работе использовалась СУБД PostgreSQL версии 8.2 [3].

Эта система была выбрана по нескольким причинам. К ним относятся, во-первых, свободное распространение и открытость исходного кода. Во-вторых, широкие функциональные возможности системы. В-третьих, наличие подробной документации. В данной статье вся реализация рассматривается применительно к данной СУБД.

Новый тип данных для СУБД PostgreSQL был определён следующим образом:

```
CREATE TYPE validtime (
INPUT = valid_time_in,
OUTPUT = valid_time_out,
internallength = 24,
alignment = double
);
```

В данном примере функция valid_time_in() преобразует данные, передаваемые на её вход, во внутреннюю структуру следующего вида:

```
typedef struct ValidTime {
int ValidBegin;
int ValidEnd;
int key;
}
```

Данная структура хранит информацию метки времени начала и конца периодов и уникальный идентификатор, комбинация этих трёх значений обеспечивает уникальность каждого кортежа. А функция valid_time_out() выполняет обратные действия. Данные функции были реализованы на языке Си.

Для пользователя БД тип представляется в виде строки вида:

```
(key, validbegin, validend)
```

где key – некоторый уникальный числовой идентификатор;

validbegin и validend – метки начала и конца периода.

Элемент типа key должен быть целым числом, в качестве меток времени может быть указана метка времени в формате Unix Timestamp либо дата в формате ГГГГ-ММ-ДД или ГГГГ-ММ-ДД ЧЧ:ММ:СС. Функция valid_time_in() автоматически преобразует указанные даты в метку времени. Все даты сохраняются в виде временных меток независимо от формата, указанного пользователем.

Так как одной из целей, преследуемых при создании нового типа данных, является отказ от использования составных первичных ключей, то создаваемый тип должен иметь возможность использоваться в качестве ключа; для этого применительно к СУБД PostgreSQL необходимо определить класс операторов, отвечающий за построение индексного дерева. В качестве индекса использовалось B-дерево.

Для создания B-дерева были определены пять операторов: меньше; меньше или равно; равно; больше или равно и больше. Каждый оператор был реализован в виде функции на языке Си, производящей сравнение двух значений нового типа и возвращающей значение типа Boolean. Ниже представлен пример внедрения в БД оператора “меньше”.

```
CREATE OPERATOR < (
leftarg = validtime, rightarg = validtime,
procedure = valid_time_less_than,
commutator = >, negator = >=,
restrict = scalarltsel, join = scalarltjoinsel
);
```

Затем, с использованием данных операторов, был определён операторный класс, позволяющий применять новый тип в первичном ключе.

```
CREATE OPERATOR CLASS valid_
time_abs_ops
DEFAULT FOR TYPE validtime
USING btree AS
OPERATOR 1 < ,
OPERATOR 2 <= ,
OPERATOR 3 = ,
OPERATOR 4 >= ,
OPERATOR 5 > ,
FUNCTION 1 valid_time_abs_
cmp(validtime, validtime);
```

Описанные выше действия позволили использовать в качестве первичных ключей не группу из трёх полей, а одно поле нового типа, что упростило написание запросов.

Однако создание нового типа данных не преобразует реляционную БД в темпоральную, а является лишь первым шагом на пути к такому преобразованию.

Кроме создания нового типа данных необходимы преобразования запросов из темпоральной формы в форму, оговорённую стандартом SQL. Рациональнее всего эти преобразования выполнять в некотором промежуточном модуле, причём для пользователя запросы к темпоральной базе данных, использующей новые типы данных, не должны сильно отличаться от запросов на классическом языке, а простейшие запросы к темпоральной БД не должны отличаться от запросов к обычной реляционной базе данных. Иными словами, запрос «SELECT * FROM table1;» в обоих случаях должен вернуть одинаковый результат (применительно к темпоральной СУБД – все актуальные на данный момент записи).

Рассмотрим задачу формирования единой БД социологических опросов. Хотя сами по себе такие данные вернее будет отнести к информации, содержащейся во временных рядах, а не к темпоральной, добавление временных атрибутов будет полезным для некоторых аспектов этих БД, поскольку с течением времени может происходить изменение вопросов или вариантов ответов. Следует обратить внимание на тот факт, что некоторые части анкет совпадают, хотя опросы могут преследовать разные цели или быть ориентированы на различные группы населения (например, различающиеся по месту жительства или социальному положению и т.д.). Сведение данных таких опросов в единую базу позволит значительно расширить исходную информацию для анализа.

Таким образом, создание единой нормализованной реляционной базы данных является важной задачей, а добавление темпоральных атрибутов в те таблицы, где это необходимо, позволит расширить её возможности.

Результаты измерений за все периоды сформированы в виде файлов программы SPSS и по сути дела представляют таблицу, в каждой строке которой хранятся данные одной анкеты. Сведение в единое целое данных опросов, проведённых в различные периоды, в такой форме является весьма затруднительным, поскольку этот вид представления удобен для анализа отдельно взятых результатов.

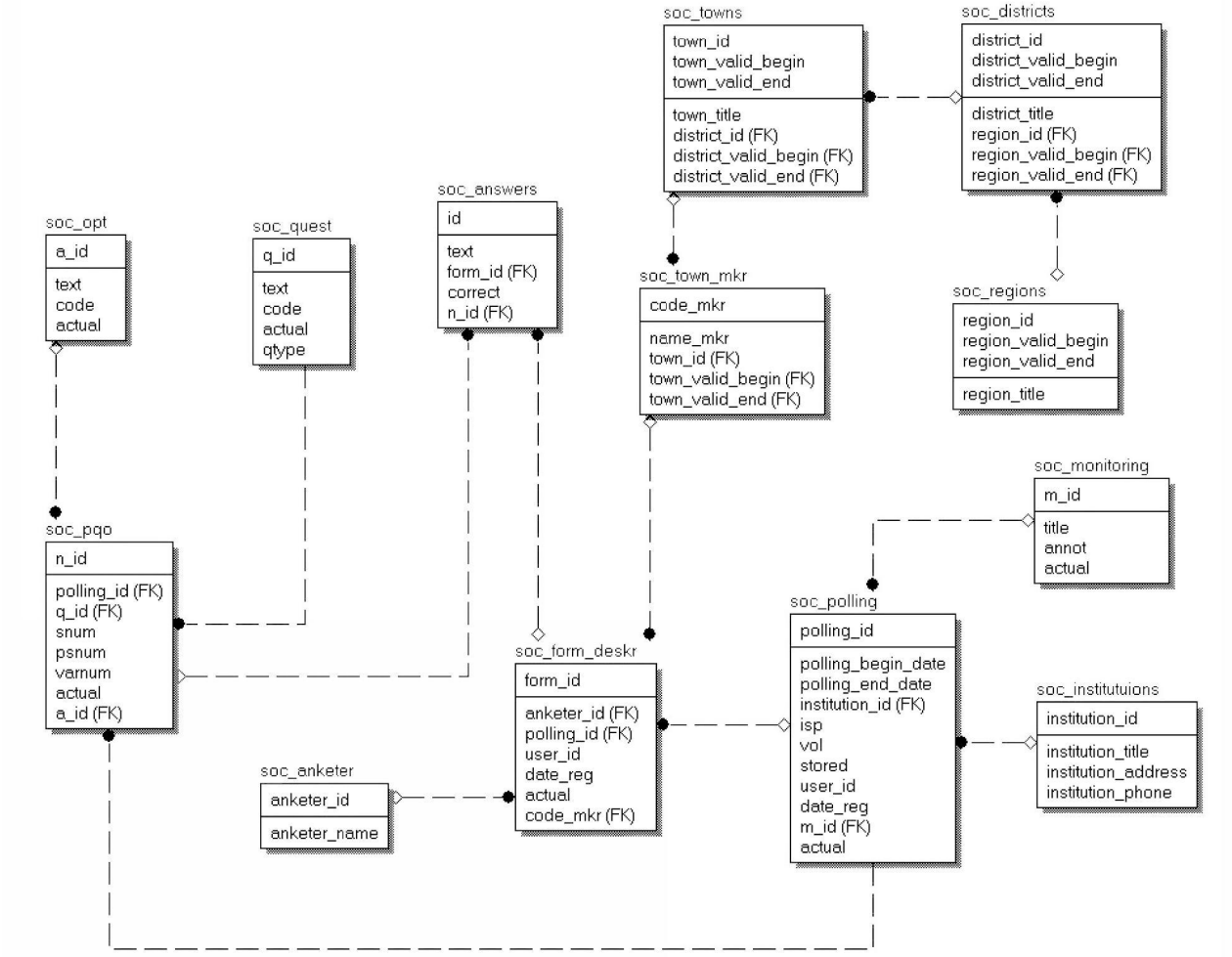
Хотя сами опросы не являются временными данными, но из логической схемы видно, что для некоторых таблиц может быть добавлен временной аспект, например для группы таблиц, описывающих место заполнения анкеты. Несмотря на достаточно постоянное административно-территориальное деление, всё же, в силу сложившейся в последнее время тенденции укрупнения административно-территориальных единиц, такие изменения необходимо учитывать при разработке БД. Без введения временной составляющей поддерживать целостность данных будет сложно.

Кроме того, временной аспект необходим в таблицах, содержащих вопросы и варианты ответов на них, так как с течением времени разработчики анкет нередко вносят в них изменения.

В приведённой на рисунке схеме базы данных первичные ключи всех таблиц, которые предлагается преобразовать в темпоральную форму, являются простыми, представляя собой некоторый уникальный идентификатор, удовлетворяющий нормальным формам и обеспечивающий неповторяемость кортежа.

При преобразовании базы в темпоральный вид невозможно применение простых первичных ключей для этих таблиц, поскольку не будет обеспечена уникальность записей, вот почему необходимо использовать составные первичные ключи вида [идентификатор записи, начало периода актуальности, конец периода актуальности]. И тогда схема базы данных будет иметь следующий вид *(рисунок)*.

Предлагаемая структура СУБД с учётом времени



Данная схема может быть легко представлена в нетемпоральной форме, для этого достаточно лишь убрать поля, отвечающие за временную составляющую, и поэтому в статье не приводится.

Хотя преобразование в темпоральный вид расширяет возможности БД, в то же время может происходить и некоторое усложнение запросов. Причём информация для темпоральной базы будет более подробной и достоверной. Так, если в нетемпоральной БД изменить варианты ответа на какой-либо закрытый вопрос (например, добавить новый вариант ответа), то изменения коснутся всех существующих данных.

Если же обратиться к темпоральной БД, то добавленные временные поля как раз и помогают решить эту задачу, указывая, в какой период существовал тот или иной вариант ответа, хотя сам запрос не изменяется.

Таким образом, можно сделать ряд выводов.

Во-первых, определение периодов актуальности для всех элементов анкеты устра-

няет неоднозначности, касающиеся формулировок вопросов, мест проведения анкетирований и т.п.

Во-вторых, добавление в БД дополнительной информации расширяет возможности анализа, позволяя извлекать и анализировать данные в динамике.

В-третьих, темпоральную БД значительно проще связать с событиями, непосредственно не отраженными в ней, но привязанными к определённым временным интервалам.

В целом можно сказать, что в настоящий момент темпоральные базы являются перспективным направлением исследований. И наиболее оптимально, по нашему мнению, создание специальных расширений управления темпоральными данными для существующих СУБД.

Литература

1. Snodgrass, R.T. Developing time-oriented database application in SQL / R.T. Snodgrass. – San Francisco: Morgan Kaufmann Publishers, 2000.

2. Фути, К. Языки программирования и схемотехника СБИС: пер. с япон. / К. Фути, Н. Судзуки. – М.: Мир, 1988. – 224 с.

3. PostgreSQL 8.2.0 Documentation [Электронный ресурс] // PostgreSQL. – Режим доступа: http://www.postgresql.org/files/ documentation/pdf/8.2/ postgresql-8.2-A4.pdf

4. Образование, наука, бизнес: особенности регионального развития и интеграции: сб. трудов Всероссийской научно-практической конференции. – Череповец: ИМИТ СПбГПУ, 2005. – 344 с.

5. Порай, Д.С. Реализация концепции темпоральной базы данных средствами реляционной СУБД [Электронный ресурс] / Д.С. Порай, А.В. Соловьев, Г.В. Корольков.– Режим доступа: ftp://ftp.dol.ru/pub/users/ cgntv/download/sbornic/sbornic5/Doc8.doc