Темпоральные расширения в стандарте SQL

С.Д. Кузнецов <kuzloc@ispras.ru>
Институт системного программирования им. В.П. Иванникова РАН,
109004, Россия, г. Москва, ул. А. Солженицына, дом 25
Московский государственный университет имени М.В. Ломоносова,
119991 ГСП-1 Москва, Ленинские горы,
МГУ имени М.В. Ломоносова, 2-й учебный корпус, факультет ВМК
Московский физико-технический институт,
141700, Московская область, г. Долгопрудный, Институтский пер., 9
Национальный исследовательский университет «Высшая школа экономики»
101000, Россия, г. Москва, ул. Мясницкая, д. 20

Аннотация. Рассматриваются основные черты, достоинства и недостатки темпоральных расширений языка SQL, введенных в стандарте SQL:2011. Вводятся основные понятия битемпоральной модели SQL: период времени, период прикладного времени, период системного времени, таблица с периодом прикладного времени, системно-версионная таблица. Обсуждаются способы определения таблиц с периодом прикладного времени, особенности выполнения над такими таблицами операций UPDATE и DELETE, специфика первичных и внешних ключей в таблицах с периодом прикладного времени. Описывается набор дополнительных предикатов для работы с таблицами с периодом прикладного времени и приводятся примеры запросов к таким таблицам. Рассматриваются правила определения системно-версионных таблиц, обсуждаются особенности запросов к таким таблицам. Наконец, обсуждается возможность определения битемпоральных таблиц, содержащих периоды и прикладного, и системного времени, приводятся примеры запросов. По ходу изложения отмечаются достоинства и недостатки битемпоральной модели SQL.

Ключевые слова: SQL:2011, период времени, период прикладного времени, период системного времени, таблица с периодом прикладного времени, системно-версионная таблица, битемпоральная таблица

1. Введение

Данные в базах данных почти всегда неразрывно связаны со временем. Это верно и для транзакционных баз данных (дата и время авиарейсов, поступления запасов на склады, начала обработки заказов в Интернет-

магазинах и т.д.), и тем более для аналитических баз данных (в частности, время является важнейшим измерением OLAP-кубов).

До начала 1990-х гг. представление в базах данных даты-времени и обработка соответствующей информации возлагались на приложения, что было нехорошо во многих отношениях и, в частности, привело к возникновению печально известной проблемы 2000-го года [1], которая, хотя и была преувеличена, вызвала во всем мире немало хлопот.

Серьезный шаг вперед был сделан в 1992 г. в стандарте SQL/92 (SQL:1992) [2], в котором были определены специальные «темпоральные» типы данных (DATE, TIME, TIMESTAMP и INTERVAL). Эти типы данных не только обеспечивают стандартное представление дат, значений времени и временны*х* интервалов, но и набор полезных операций над темпоральными значениями, в том числе, логическую операцию OVERLAP, вырабатывающую истинное значение, если два заданных временных диапазона имеют непустое пересечение.

Однако, несмотря на введение в SQL темпоральных типов данных, SQL-ориентированные базы данных позволяют сохранять только статические мгновенные снимки данных предметных областей. Например, если в базе данных служащих Иванову повышают зарплату, то без ухищрений на уровне приложения в базе данных невозможно сохранить данные о его предыдущей зарплате и, тем более, о размере его зарплаты год назад.

В истинно темпоральных базах данных для каждого объекта базы данных автоматически поддерживаются все его состояния от момента создания до момента уничтожения. Системы управления базами данных (СУБД), поддерживающие темпоральные базы данных, и сами называются темпоральными СУБД. Начиная с 1980-х вокруг темпоральных баз данных велась активная исследовательская работа, связанная как с теоретическими аспектами поддержки времени в базах данных, так и с проблемами реализации темпоральных СУБД [3].

В 1995 г. была завершена работа специально созданного комитета под руководством Ричарда Снодграсса по выработке спецификаций языка SQL, поддерживающего темпоральные базы данных (TSQL2) [4]. В том же году в комитете по стандартизации SQL Международной организации по стандартизации (ISO) был начат проект по разработке расширений стандарта SQL (SQL:1992) на основе идей TSQL2. Однако анализ экспертов показал наличие в TSQL2 неустранимых противоречий [5]. Кроме того, для реализации TSQL2 требовалась очень значительная переделка существующих SQL-ориентированных СУБД, а основные производители коммерческих СУБД соответствующего желания не испытывали. В результате в 2001 г. проект был закрыт.

Несколькими годами позже в комитет по стандартизации SQL поступили новые предложения расширений SQL, которые были одобрены комитетом. Это привело к принятию в конце 2011 г. нового стандарта SQL:2011, во

второй части которого (Foundation) содержатся спецификации темпоральных расширений [6]. Бесплатно доступен предельно близкий к официальному стандарту SQL:2011 драфт [7]. Темпоральные средства SQL подробно обсуждаются в книге [8], но, как свойственно Дейту, это обсуждение ведется в виде критического сравнения с собственной темпоральной моделью авторов, которой посвящена основная и весьма объемная часть книги. Это несколько затрудняет понимание сути темпоральных расширений в SQL:2011. Неформально и понятно темпоральные возможности SQL описаны в [9].

В стандарте SQL:2011 используется так называемая битемпоральная модель. Это означает, что в SQL-ориентированных базах данных могут поддерживаться два вида времени: транзакционное (transactional time, автоматически обеспечиваемое системой) и истинное (valid time, задаваемое приложениями базы данных). В стандарте SQL транзакционное время называется системным (system time), а истинное время — прикладным (application time). Подробнее об этом говорится в основной части статьи.

К середине 2017 г. полнее всего темпоральная часть SQL:2011 поддерживается в IBM DB2 [10]. Оба вида времени поддерживаются и в Oracle, но, как отмечается в [11], реализацию темпоральных средств пока нельзя считать полной (кроме того, похоже, что синтаксис темпоральных конструкций в Oracle SQL отличен от стандартного). Полная поддержка времени еще с 2010 г. имеется в Teradata [12], но реализациия опирается на TSQL2, а не на стандарт SQL. В ряде СУБД (в том числе, в Microsoft SQL Server [13] и PostgreSQL [14]) имеется частичная поддержка времени с некоторой опорой на SQL:2011.

В этой статье приводится обзор темпоральных расширений, введенных в SQL:2011.[1] Обзор опирается на материалы [6-9]. Изложение следует плану доклада на семинаре Московской секции ACM SIGMOD [15]. Во втором разделе рассматриваются основные понятия SQL:2011, связанные с темпоральными средствами. Третий раздел посвящен обсуждению средств поддержки прикладного времени, четвертый — средств поддержки системного времени. В пятом разделе описываются возможности определения битемпоральных таблиц. Шестой раздел заключает статью.

2. Основные понятия

В основе битемпоральной модели SQL лежит понятие периода времени (time period) — полуинтервала оси времени, начинающегося в точке начального времени (start time) и заканчивающегося в точке конечного времени (end time). Почти во всех случаях применяется замкнуто-открытая (closed-open) семантика периодов времени: период включает точку начального времени и не

[1] На русском языке темпоральные расширения SQL описаны в [16], однако в этой статье всего лишь честно пересказывается соответствующий материал стандарта SQL:2011 и не предпринимаются попытки его критического анализа.

включает точку конечного времени.[2] Периоды времени можно определять и связывать со строками таблиц.

В отличие от других подходов (в частности, TSQL2 [4]), в модели SQL для периодов времени не вводится специальный тип данных. В [9] отказ от введения нового типа данных обосновывается следующими соображениями. Во-первых, добавление нового типа данных в стандарт SQL потребовало бы не только серьезных переделок в SQL-ориентированных СУБД, но и оснащения соответствующими средствами всех языков серверного программирования, расширения всех API для клиентского программирования (ODBC, JDBC, .NET и т.д.), соответствующих переделок средств ETL и т.д. Все это (в лучшем случае) значительно задержало бы реализацию темпоральных расширений.

Во-вторых, поскольку темпоральные возможности реально требуются во многих приложениях, при отсутствии соответствующей поддержки со стороны СУБД многие разработчики приложений реализовывали нужные им темпоральные возможности внутри приложений. Для этого обычно в таблицы баз данных добавлялись два столбца — начальное время и конечное время периода. Подобные приложения проще адаптировать к системной поддержке времени, если периоду времени по-прежнему будут соответствовать два столбца каждой темпоральной таблицы.

```
<table period definition> ::=
<system or application time period specification>
<left paren> <period begin column name> <comma> <period
end column name> <right paren>

<system or application time period specification> ::=
<system time period specification>
| <application time period specification>

<system time period specification> ::=
PERIOD FOR SYSTEM_TIME

<application time period specification> ::=
PERIOD FOR <application time period name>
```

Рис. 1. Определение периода времени в определении таблицы

Синтаксис определения периода времени в определении таблицы показан на рис. 1. В стандарте SQL определение периода является частью определения таблицы, расширяя набор метаданных, связанных с этой таблицей. Определение периода — это именованный компонент определения таблицы,

[2] Не следует путать периоды времени со значениями типов INTERVAL, которые задают лишь продолжительность какого-либо события, не привязанную к точкам оси времени.

задающий пару столбцов, в которых сохраняются точки начального и конечного времени периода. Эти столбцы являются обычными столбцами таблицы со своими собственными именами. Имя периода времени находится в том же пространстве имен, что и имена столбцов, поэтому оно должно отличаться от имени любого столбца той же таблицы. В операции ALTER TABLE имеется возможность отмены определения периодов.

Битемпоральность модели SQL означает, что в ней поддерживаются две разновидности времени, традиционно называемые истинным и транзакционным временем. Период истинного времени — это период, в течение которого строка считается правильно отражающей реальность пользователей базы данных. Период транзакционного времени — это период между двумя последовательными изменениями данной строки (время жизни строки таблицы начинает с ее вставки в таблицу, включает ноль или большее число изменений и завершается при удалении строки).

Для любой строки ее период транзакционного времени может произвольным образом отличаться от периода истинного времени. Например, если 26 мая 2017 г. в таблицу USA_Presidents вставляется строка с данными Барака Обамы, то в соответствии с реальностью она корректна в периоде истинного времени с 20 января 2009 года по 20 января 2017 года. В периоде же транзакционного времени данная строка действительна с 26 мая 2017 г. до ближайшей операции изменения.

В SQL:2011 транзакционное время поддерживается на основе механизма системно-версионных таблиц (system-versioned table), в определение которых входит определение периода системного времени (system-time period), а истинное время — на основе таблиц, в определение которых входит определение периода прикладного времени (application-time period). Имя периода системного времени в стандарте предопределено (SYSTEM_TIME), а имя периода прикладного времени задается пользователями. Для каждой таблицы можно определить не более одного периода системного времени и не более одного периода прикладного времени.

3. Таблицы с периодом прикладного времени

Таблицы с периодом прикладного времени предназначены для удовлетворения потребностей приложений, которым нужно сохранять периоды времени, в которые данные полагаются истинными в реальном мире. Примером может служить страховое приложение, в котором необходимо отслеживать условия страхования клиента в любой заданной точке времени.

Для таких приложений требуется, чтобы их пользователи отвечали за установку начальной и конечной точек периодов истинности строк, и чтобы пользователи могли выбирать любые значения времени (в прошлом, настоящем или будущем) в качестве начальной и конечной точек периодов. Кроме того, пользователям должна быть предоставлена возможность изменять

периоды истинности существующих строк при обнаружении ошибок или поступлении новой информации.

```
CREATE TABLE EMP (
EMP_NO EMP_NO,
EMP_DEPT_NO DEPT_NO,
EMPStart DATE NOT NULL,
EMPEnd DATE NOT NULL,
PERIOD FOR EMPPeriod (EMPStart, EMPEnd));
```

Рис. 2. Определение таблицы с периодом прикладного времени

На рис. 2 приведено определение таблицы служащих с периодом прикладного времени. Здесь типом данных столбцов начала и конца периода является DATE, но можно использовать и тип TIMESTAMP, лишь бы у обоих столбцов типы данных были одинаковыми. В обоих столбцах запрещаются неопределенные значения.

Начальные значения начальной и конечной точек периода прикладного времени для данной строки устанавливаются при вставке этой строки в таблицу с помощью обычного оператора INSERT. На рис. 3 показан оператор INSERT, вставляющий одну строку в таблицу EMP (данные о служащем с номером 22217, который работал в третьем отделе в период с 1 января 2010 г. до 12 ноября 2011 г.), и состояние этой таблицы после выполнения операции вставки (предполагается, что до этого таблица была пустой)[3,4]. Для изменения (включая столбцы начальной и конечной точек периода) и удаления строк таблиц с периодом прикладного времени можно использовать обычные операторы UPDATE и DELETE соответственно.

```
INSERT INTO EMP VALUES (22217,
DATE '01.01.2010',
DATE '12.11.2011', 3)
```

EMP

EMP_NO	EMPStart	EMPEnd	EMP_DEPT_NO
22217	01.01.2010	12.11.2011	3

Рис. 3. Вставка строки в таблицу EMP

3.1 Операции UPDATE и DELETE с разделом FOR PORTION OF

Для таблиц с периодом прикладного времени в SQL:2011 обеспечивается возможность изменений, которые действуют в течение заданного периода

[3] В статье используются примеры из [9].

[4] Во всех примерах используется европейский формат представления литералов типа DATE.

времени. Для этого в операторах UPDATE и DELETE нужно использовать новый раздел FOR PORTION OF. Расширенный синтаксис операторов UPDATE и DELETE показан на рис. 4.

```
<update statement: searched> ::=
UPDATE <target table>
[FOR PORTION OF <application time period name>
FROM <point in time 1> TO <point in time 2>]
[[AS] <correlation name>]
SET <set clause list>
[WHERE <search condition>]

<delete statement: searched> ::=
DELETE FROM <target table>
[FOR PORTION OF <application time period name>
FROM <point in time 1> TO <point in time 2>]
[[AS] <correlation name>]
[WHERE <search condition>]
```

Рис. 4. Синтаксис «поисковых» операторов UPDATE и DELETE в SQL:2011

Предположим, что про служащего с номером 22217 стало известно, что в период с 3 февраля 2011 г. по 10 сентября 2011 г. он числился в отделе номер 4. Тогда над таблицей EMP можно выполнить операцию UPDATE, текст которой и вид результирующей таблицы показаны на рис. 5 (предполагается, что изменение применяется к таблице EMP с рис. 3).

```
UPDATE EMP
FOR PORTION OF EMPPeriod
FROM DATE '03.02.2011'
TO DATE '09.10.2011'
SET EMP_DEPT_NO = 4
WHERE EMP_NO = 22217
```

EMP

EMP_NO	EMPStart	EMPEnd	EMP_DEPT_NO
22217	01.01.2010	03.02.2011	3
22217	03.02.2011	09.10.2011	4
22217	09.10.2011	12.11.2011	3

Рис. 5. Оператор UPDATE с разделом FOR PORTION OF

При выполнении этого оператора СУБД должна найти в таблице EMP все строки, у которых период прикладного времени имеет общую часть с периодом *P* с 3 февраля по 10 сентября 2011 г. (в соответствии с принятой замкнуто-открытой семантикой 3 февраля входит в *P*, а 10 сентября не входит). Для тех строк, для которых период прикладного времени входит в *P*,

выполняется оставшаяся часть оператора (т.е. в строках, удовлетворяющих условию, меняется значение EMP_DEPT_NO).

Если у периода прикладного времени строки, у которой период имеет общую часть с периодом *P*, есть также и часть, строго предшествующая *P* или строго следующая за *P*, то строка расщепляется на две или три последовательные (во времени) строки в зависимости от вида перекрытия, и у той строки, у которой период прикладного времени содержится в *P*, изменяется столбец EMP_DEPT_NO. Конечно, во всех строках соответствующим образом изменяются столбцы EMPStart и/или EMPEnd. На рис. 5 мы имеем дело с расщеплением исходной строки на три. Первая и третья строки считаются заново вставленными (и для них могут сработать триггеры по INSERT), а вторая — обновленной (и для нее могут сработать триггеры по UPDATE).

Аналогично работает оператор DELETE с разделом FOR PORTION OF. Предположим, про служащего с номером 22217 стало известно, что в период с 3 февраля 2011 г. по 10 сентября 2011 г. он не работал. Тогда над таблицей EMP можно выполнить оператор DELETE, текст которого и вид результирующей таблицы показаны на рис. 6 (предполагается, что изменение применяется к таблице EMP с рис. 3).

```
DELETE EMP
FOR PORTION OF EMPPeriod
FROM DATE '03.02.2011'
TO DATE '09.10.2011'
WHERE EMP_NO = 22217
```

EMP

EMP_NO	EMPStart	EMPEnd	EMP_DEPT_NO
22217	01.01.2010	03.02.2011	3
22217	09.10.2011	12.11.2011	3

Рис. 6. Оператор DELETE с разделом FOR PORTION OF

Для тех строк, для которых период прикладного времени входит в *P*, выполняется оставшаяся часть оператора (т.е. строки, удовлетворяющие условию, удаляются). Если у периода прикладного времени строки, у которой период имеет общую часть с периодом *P*, есть также часть, строго предшествующая *P* или строго следующая за *P*, то строка расщепляется на две или три последовательные (во времени) строки в зависимости от вида перекрытия, и та строка, у которой период прикладного времени содержится в *P*, удаляется. Конечно, в остающихся строках соответствующим образом изменяются столбцы EMPStart или EMPEnd.

На рис. 6 мы имеем дело с расщеплением исходной строки на три. Для удаленной строки может сработать триггер по DELETE, а для оставшихся строк — по INSERT.

3.2 Первичный ключ и ссылочные ограничения целостности

Как показывают примеры предыдущего подраздела, если таблица EMP является таблицей с периодом прикладного времени, то в отличие от привычных таблиц служащих столбец EMP_NO не является первичным ключом. Похоже, что в этом случае первичным ключом должна являться комбинация столбцов {EMP_NO, EMPStart, EMPEnd} (точнее, комбинация столбца EMP_NO и периода EMPPeriod, потому что у служащего должен иметься один отдел в каждой точке каждого периода прикладного времени). Однако этого не всегда достаточно.

EMP (a)

EMP_NO	EMPStart	EMPEnd	EMP_DEPT_NO
22217	01.01.2010	03.02.2011	3
22217	01.01.2010	03.02.2011	4

EMP (b)

EMP_NO	EMPStart	EMPEnd	EMP_DEPT_NO
22217	01.01.2010	03.02.2011	3
22217	10.09.2010	03.02.2011	4

Рис. 7. Строки с перекрывающимися периодами прикладного времени

Если объявить первичным ключом таблицы EMP составной столбец {EMP_NO, EMPPeriod}, то наполнение EMP с рис. 7 (a) недопустимо, поскольку нарушает ограничение первичного ключа. Однако таблица EMP вполне может содержать строки с рис. 7 (b). Содержательно это означает, что служащий 22217 не мог работать сразу в двух отделах в периоде с 1 января 2010 г. до 3 февраля 2011 г., но ему было позволено это с 10 сентября 2010 г.

Это ограничение странно само по себе, трудно объяснить его смысл. Но главное, что недопустимость таблицы с рис. 7 (a) согласуется с тем, что в таблице EMP без периода прикладного времени столбец EMP_NO являлся бы первичным ключом. В частности, это означает, что ни один служащий ни в какой момент времени не может работать сразу в двух отделах, а допустимость EMP с рис. 7 (b) смыслу EMP_NO противоречит.

Поэтому ситуации, подобные той, которая показана на рис. 7 (b), в стандарте SQL следовало бы запретить, уточнив семантику первичного ключа, включающего период прикладного времени. Однако разработчики стандарта на это не пошли, обеспечив вместо этого спецификацию WITHOUT OVERLAPS в определении первичного ключа. Для нашей таблицы EMP первичный ключ следовало бы определить следующим образом:

```
PRIMARY KEY (ENo, EMPPeriod, WITHOUT OVERLAPS).
```

Спецификация первичного (и возможного) ключа WITHOUT OVERLAPS в SQL:2011 является опциональной, так что по умолчанию перекрытия периодов прикладного времени у строк, соответствующих одной и той же

сущности предметной области, допускаются. Как мы видели выше, это приводит к потенциальной возможности возникновения ситуаций, противоречащих здравому смыслу. Так что, как минимум, стоило бы запретить такие перекрытия по умолчанию (если не запретить вовсе).

```
(a)
CREATE TABLE DEPT (
DEPT_NO DEPT_NO,
DEPT_NAME VARCHAR(30),
DEPTStart DATE NOT NULL,
DEPTEnd DATE NOT NULL,
PERIOD FOR DEPTPeriod (DEPTStart, DEPTEnd),
PRIMARY KEY (DEPT_NO, DEPTPeriod WITHOUT OVERLAPS))
```

(b) EMP

EMP_NO	EMPStart	EMPEnd	EMP_DEPT_NO
22217	01.01.2010	03.02.2011	3
22217	03.02.2011	12.11.2011	4

DEPT

DEPT_NO	DEPTStart	DEPTEnd	DEPT_NAME
3	01.01.2009	31.12.2011	Test
4	01.06.2011	31.12.2011	QA

(c) EMP

EMP_NO	EMPStart	EMPEnd	EMP_DEPT_NO
22217	01.01.2010	03.02.2011	3
22217	03.02.2011	12.11.2011	4

DEPT

DEPT_NO	DEPTStart	DEPTEnd	DEPT_NAME
3	01.01.2009	31.12.2011	Test
4	01.02.2011	31.12.2011	QA

(d) EMP

EMP_NO	EMPStart	EMPEnd	EMP_DEPT_NO
22217	01.01.2010	03.02.2011	3
22217	03.02.2011	12.11.2011	4

DEPT

DEPT_NO	DEPTStart	DEPTEnd	DEPT_NAME
3	01.01.2009	31.12.2011	Test
4	01.02.2011	30.05.2011	QA
4	01.06.2011	31.12.2011	Cross-Check

Рис. 8. Недопустимые и допустимые значения внешнего ключа

На рис. 8 показано определение таблицы DEPT с периодом прикладного времени (a) и три теоретически возможных наполнения таблиц EMP и DEPT ((b), (c) и (d)). В соответствии со здравым смыслом и традиционной семантикой базы данных EMP-DEPT столбец EMP_DEPT_NO таблицы EMP должен содержать «ссылки» на строки таблицы DEPT, и каждому отличного от NULL значению столбца EMP_DEPT_NO должна соответствовать в точности одна строка таблицы DEPT (отдел, в котором работает каждый служащий, должен существовать и быть единственным). Другими словами, в традиционном случае столбец EMP_DEPT_NO является внешним ключом, которому соответствует первичный ключ DEPT_NO таблицы DEPT.

В таблице DEPT с периодом прикладного времени DEPT_NO не является первичным ключом. Тем не менее, для таблиц EMP и DEPT с рис. 8 (b) и 8 (c) ограничение внешнего ключа, на первый взгляд, выполняется: служащий 22217 в разные периоды времени работал в отделах 3 и 4, и в таблице DEPT содержатся две строки, соответствующие обоим этим отделам.

Однако DEPT, как и EMP, является таблицей с периодом прикладного времени! Каждая строка каждой из этих таблиц связана с периодом ее корректности. Содержимое таблицы EMP с рис. 8 (b) показывает, что служащий 22217 работал в 3-м отделе с 1 февраля 2010 г. по 3 февраля 2011 г., а с 3 февраля 2011 г. по 12 ноября 2011 г. работал в 4-м отделе. Из содержимого же таблицы DEPT следует, что отдел номер 3 существовал с 1 января 2009 г. по 31 декабря 2011 г., а 4-й отдел — с 1 июня по 31 декабря 2011 г. Следовательно, служащий 22217 мог работать в отделе 3 с 1 февраля 2010 г. по 3 февраля 2011 г., но не мог работать в отделе 4 с 3 февраля 2011 г. по 12 ноября 2011 г., и в действительности ограничение внешнего ключа для таблиц EMP и DEPT с рис. 8 (b) не выполняется.

С другой стороны, содержимое таблицы DEPT на рис. 8 (c) показывает, что отдел 4 существовал с 1 февраля по 31 декабря 2011 г. Поэтому служащий 22217 в период 3 февраля 2011 г. по 12 ноября 2011 г. мог работать в этом отделе, и ограничение внешнего ключа для таблиц EMP и DEPT с рис. 8 (c) выполняется.

Более того, по смыслу ссылочное ограничение выполняется и для таблиц EMP и DEPT с рис. 8 (d), где в DEPT присутствуют две строки с DEPT_NO, равным 4. Действительно, в этом варианте DEPT отдел 4 существовал с 1 февраля по 31 мая 2011 г. по названием QA и с 1 июня по 31 декабря 2011 г. под названием Cross-Check. Поэтому служащий 22217 в период 3 февраля 2011 г. по 12 ноября 2011 г. мог работать в отделе номер 4.

Для обеспечения должного поведения внешнего ключа в стандарте SQL:2011 расширяется синтаксис его определения для таблиц с периодом прикладного времени и соответствующим образом изменяется определение ограничения ссылочной целостности для таких таблиц. На рис. 9 представлен фрагмент набора синтаксических правил расширенного определения внешнего ключа.

```
<referential constraint definition> ::=
FOREIGN KEY <left paren> <referencing column list>
[<comma> <referencing period specification>] <right
paren>
<references specification>

<references specification> ::=
REFERENCES <referenced table and columns>
[MATCH <match type>] [<referential triggered action>]

<match type> ::=
FULL
| PARTIAL
| SIMPLE

<referencing column list> ::=
<column name list>

<referencing period specification> ::=
PERIOD <application time period name>

<referenced table and columns> ::=
<table name> [<left paren> <referenced column list>
[<comma> <referenced period specification> ] <right
paren>]

<referenced column list> ::=
<column name list>

<referenced period specification> ::=
PERIOD <application time period name>
```

Рис. 9. Синтаксис определения внешнего ключа для таблиц с периодом прикладного времени

Как показывают синтаксические правила, внешний ключ таблицы с периодом прикладного времени должен включать этот период и может ссылаться только на таблицы с периодом прикладного времени. Для таблицы EMP с рис. 8 (а) определение внешнего ключа могло бы выглядеть следующим образом:

```
FOREIGN KEY (EMP_DEPT_NO, Period EMPPeriod)
REFERENCES DEPT (DEPT_NO, Period DEPTPeriod).
```

Ссылочное ограничение таблицы с периодом прикладного времени удовлетворяется в том и только в том случае, когда для каждой ее строки либо значения всех ссылающихся столбцов (referencing column на рис. 9) являются неопределенными, либо значения всех ссылающихся столбцов отличны от NULL и в таблице, на которую ведет ссылка, имеется непустой набор строк, значения соответствующих столбцов которых совпадают со значениями ссылающихся столбцов данной строки и объединение значений периодов прикладного времени которых включает значение периода прикладного времени данной строки.[5]

3.3 Предикаты с периодом прикладного времени

В стандарте SQL:2011 специфицированы новые предикаты, позволяющие формулировать условия выборки, включающие периоды прикладного времени: OVERLAPS, EQUALS, CONTAINS, PRECEDES, SUCCEEDS, IMMEDIATELY PRECEDES и IMMEDIATELY SUCCEEDS. Неформальный фрагмент синтаксических правил приведен на рис. 10.

```
<period predicate> ::=
<period predicand> predicate_name <period predicand>

<period predicand> ::=
<period reference>
| PERIOD <left paren> <period start value>
<comma> <period end value> <right paren>

<period contains predicate> ::=
<period predicand> CONTAINS
{<period predicand> | <datetime value expression>}
```

Рис. 10. Неформальный синтаксис предикатов с периодом прикладного времени

Синтаксические правила для всех предикатов, кроме CONTAINS, практически совпадают. Поэтому в первом «мета»-правиле вместо predicate_name можно подставить OVERLAPS, EQUALS, PRECEDES, SUCCEEDS, IMMEDIATELY PRECEDES или IMMEDIATELY SUCCEEDS. Периоды-операнды могут задаваться путем указания имени существующего периода прикладного

[5] Приводится определение для спецификации соответствия FULL. Для случаев SIMPLE и PARTIAL определение изменяется очевидным образом.

времени, либо путем явного задания начальной и конечной точек периода. Для предиката CONTAINS в качестве второго операнда, кроме периода, можно указывать точку временной оси.

- Предикат OVERLAPS принимает значение true в том и только в том случае, когда два заданных периода прикладного времени перекрываются, то есть у них имеется хотя бы одна общая точка временной оси.
- Предикат EQUALS принимает значение true в том и только в том случае, когда у двух заданных периодов прикладного времени совпадают начальные и конечные точки.
- Предикат CONTAINS принимает значение true в том и только в том случае, когда первый период включает второй период или заданную точку временной оси.
- Предикат PRECEDES принимает значение true в том и только в том случае, когда первый период предшествует второму, то есть все значения времени первого периода меньше начальной точки второго периода.
- Предикат SUCCEEDS принимает значение true в том и только в том случае, когда первый период следует за вторым периодом, то есть все временны*е* значения, входящие в первый период, больше или равны конечному значению второго периода.
- Предикат IMMEDIATELY PRECEDES принимает значение true в том и только в том случае, когда первый период непосредственно предшествует второму периоду, то есть конечная точка первого периода совпадает с начальной точкой второго периода.
- Наконец, предикат IMMEDIATELY SUCCEEDS принимает значение true в том и только в том случае, когда первый период непосредственно следует за первым периодом, то есть начальная точка первого периода совпадает с конечной точкой второго периода.
- Если какое-либо выражение даты-времени, вычисляемое во время оценки предиката с периодом прикладного времени, приводит к неопределенному значению, то значением предиката является unknown.

Смысл предикатов достаточно очевиден. Для порядка приведем две формулировки запросов над таблицами EMP и DEPT с периодами прикладного времени.

Запрос 3.3.1. Найти номера служащих, начавших работать в отделе номер 3 во время существования отдела номер 4.

```
SELECT EMP_NO
FROM EMP, DEPT
WHERE EMP_DEPT_NO = 3 AND DEPT_NO = 4 AND
DEPTPeriod CONTAINS EMPStart
```

Запрос 3.3.2. Найти номера служащих, которые завершили работу в отделе номер 3 до образования отдела номер 4.

```
SELECT EMP_NO
FROM EMP, DEPT
WHERE EMP_DEPT_NO = 3 AND DEPT_NO = 4 AND
EMPPeriod PRECEDES DEPTPeriod
```

4. Системно-версионные таблицы

Системно-версионные таблицы предназначены для удовлетворения потребностей приложений, нуждающихся в поддержке точной истории изменения данных. Характерным классом таких приложений являются банковские приложения, в которых требуется сохранение предыдущих состояний клиентских счетов, чтобы каждому клиенту можно было предоставить детальную историю состояния его счета.

Для поддержки подобных приложений нужно, чтобы перед выполнением любой операции обновления или удаления строки автоматически сохранялось предыдущее состояние этой строки вместе со значением периода времени, в течение которого эта строка являлась текущей. Другим важным требованием является то, чтобы система (а не пользователь) поддерживала начальные и конечные точки периодов строк, и чтобы пользователи не могли изменять содержимое исторических строк, а также связанные с ними периоды.

```
CREATE TABLE EMP (
EMP_NO EMP_NO,
EMP_NAME VARCHAR(30),
EMP_Sys_start TIMESTAMP(12)
GENERATED ALWAYS AS ROW START,
EMP_Sys_end TIMESTAMP(12)
GENERATED ALWAYS AS ROW END,
PERIOD FOR SYSTEM_TIME
(EMP_Sys_start, EMP_Sys_end)
) WITH SYSTEM VERSIONING
```

Рис. 11. Определение системно-версионной таблицы

На рис. 11 приведено возможное определение системно-версионного варианта таблицы EMP. В определении системно-версионной таблицы должна

содержаться спецификация периода с именем SYSTEM_TINE и ключевые слова WITH SYSTEM VERSIONING. Как и для таблиц с периодами прикладного времени, в системно-версионной таблице должны быть определены два столбца с произвольными именами, сохраняющие начальное и конечное время периодов системного времени (на рис. 11 это столбцы EMP_Sys_start и EMP_Sys_end). Стандарт допускает использование для этих столбцов типов данных DATE и TIMESTAMP, однако в реализациях следует ожидать использования только типа TIMESTAMP с максимально допустимой точностью в долях секунды.

Как и для периодов прикладного времени, для периодов системного времени используется замкнуто-открытая модель. В любой момент времени строка системно-версионной таблицы считается *текущей системной строкой* (*current system row*), если ее период системного времени включает текущее время. Строка системно-версионной таблицы, не являющаяся текущей, считается *исторической системной строкой* (*historical system row*).

4.1 Отличия системно-версионных таблиц от таблиц с периодами прикладного времени

Вот основные отличия системно-версионных таблиц от таблиц с периодами прикладного времени.

- В отличие от таблиц с периодами прикладного времени, пользователи не могут заносить значения в столбцы начала и конца периода системного времени (или изменять значения этих столбцов). Эти столбцы устанавливаются и изменяются системой автоматически. Поэтому определения таких столбцов должны включать ключевые слова GENERATED ALWAYS.
- При вставке строки в системно-версионную таблицу в столбец начала периода системного времени этой строки заносится *временная метка транзакции* (*transaction timestamp*) — специальное значение, ассоциированное с каждой транзакцией[6], а в столбец конца периода — наибольшее значение типа данных этого столбца. На рис. 12 показан пример вставки строки в пустую системно-версионную таблицу EMP (считается, что значением временной метки соответствующей транзакции является 01.01.2012 09:00:00).

[6] Хотя естественно считать, что временная метка транзакции содержит полные дату и время ее начала, стандарт SQL:2011 позволяет в каждой реализации выбрать собственный вариант поддержки временных меток; требуется лишь неизменность временной метки транзакции после начала этой транзакции.

```
INSERT INTO EMP (EMP_NO, EMP_NAME)
VALUES (22217, 'Joe');
```

EMP

EMP_NO	EMP_NAME	EMP_Sys_Start	EMP_Sys_End
22217	Joe	01.01.2012 09:00:00	31.12.9999 23:59:59

Рис. 12. Вставка строки в системно-версионную таблицу EMP

- Операции UPDATE и DELETE над системно-версионной таблицей воздействуют только на текущие системные строки. Пользователи не могут обновлять или удалять исторические системные строки. Они также не могут изменять значения начального и конечного столбцов периода системного времени ни в текущих, ни в исторических системных строках.
- При выполнении операций UPDATE и DELETE над системно-версионной таблицей происходит автоматическая вставка исторической системной строки для каждой обновляемой или удаляемой текущей системной строки.

4.2 Операции UPDATE и DELETE над системно-версионными таблицами

Операция UPDATE над строкой системно-версионной таблицы сначала заносит в таблицу копию обновляемой строки, к которой в столбец конечной точки периода системного времени устанавливается значение временной метки транзакции, в которой выполняется UPDATE (строка перестала быть текущей во время выполнения этой транзакции). Затем строка соответствующим образом обновляется, причем значением ее столбца начальной точки периода системного времени становится временная метка транзакции. На рис. 13 показан пример операции UPDATE над таблицей EMP с рис. 12 (считается, что значением временной метки соответствующей транзакции является 03.02.2012 10:00:00).


```
UPDATE Emp
SET EName = 'Tom'
WHERE ENo = 22217
```

EMP

EMP_NO	EMP_NAME	EMP_Sys_Start	EMP_Sys_End
22217	Joe	01.01.2012 09:00:00	03.02.2012 10:00:00
22217	Tom	03.02.2012 10:00:00	31.12.9999 23:59:59

Рис. 13. Обновление строки в системно-версионной таблице

При выполнении операции UPDATE возбуждаются триггеры по обновлению для новой текущей системной строки и не возбуждаются триггеры по вставке для новой исторической системной строки. Исторические строки, образуемые при обновлении одной и той же строки таблицы, образуют последовательную цепочку без разрывов между их периодами системного времени.

Замечание 4.3.1. Если в одной транзакции некоторая строка таблицы с периодом системного времени обновляется более одного раза, то историческая системная строка заносится в таблицу только один раз при выполнении первой операции обновления. Как указано в стандарте SQL:2011, если значение столбца начала периода системного времени обновляемой строки совпадает со значением временной метки транзакции, то эта строка обновляется без вставки соответствующей исторической системной строки.

Замечание 4.3.2. Если при выполнении операции обновления транзакции некоторой строки таблицы с периодом системного времени оказывается, что значение столбца начала периода системного времени обновляемой строки меньше значения временной метки транзакции, то операция не выполняется, и возникает исключительная ситуация «invalid row version». Другими словами, по естественным причинам не допускается обновление строки таблицы с периодом системного времени, которая до этого обновлялась в транзакции, более «молодой», чем та, в которой выполняется текущая операция обновления (не удается сформировать допустимый период системного времени в новой исторической системной строке). Понятно, что в возникновении такой ситуации никак не повинно приложение, от имени которого выполняется пострадавшая транзакция, и поэтому возможность возникновения исключительной ситуации «invalid row version» можно считать серьезным недостатком подхода к работе с системным временем, предлагаемого в стандарте. Если ничего не менять, то единственным выходом из положения (по крайней мере, на мой взгляд; в стандарте и у его комментаторов на этот счет ничего не говорится) является полный откат пострадавшей транзакции и повторное ее выполнение с новой временной

меткой. При повторном выполнении неприятная ситуация, скорее всего, не повторится.

Операция DELETE над системно-версионной таблицей изменяет значение столбцов конца периода системного времени «удаляемых» текущих системных строк, делая их тем самым историческими системными строками. На рис. 14 показан пример операции DELETE над системно-версионной таблицей EMP с рис. 12 (предполагается, что значением временной метки соответствующей транзакции является 01.06.2012 00:00:00).

```
DELETE FROM EMP
WHERE EMP_NO = 22217;
```

EMP

EMP_NO	EMP_NAME	EMP_Sys_Start	EMP_Sys_End
22217	Joe	01.01.2012 09:00:0	01.06.2012 00:00:00

Рис. 14. Удаление строки из системно-версионной таблицы EMP

При таком «удалении» строк возбуждаются триггеры по удалению. Как и операция обновления, операция удаления строки из системно-версионной таблицы может не удасться (см. замечание 4.3.2).

4.3 Первичный ключ и ссылочное ограничение целостности

Исторические системные строки в системно-версионных таблицах не изменяются, и каждая из них происходит из некоторой текущей системной строки. Поэтому, если текущая системная строка удовлетворяла ограничениям первичного и внешнего ключа до того, как превратилась в историческую системную строку, то и историческая строка ограничениям удовлетворяет, и проверять их для нее никогда не требуется.

Поэтому для системно-версионных таблиц не требуется включать в состав первичного или внешнего ключа период системного времени. В частности, для системно-версионной таблицы EMP первичным ключом может быть объявлен столбец EMP_NO, а внешним ключом, ссылающимся на таблицу DEPT — столбец EMP_DEPT_NO.

Замечание 4.4.1. Приведенные выше утверждения заведомо справедливы, если *все* таблицы базы данных являются системно-версионными. Тогда можно утверждать, что для любой временной метки в прошлом все исторические и текущие строки всех таблиц, период системного времени которых включает данную временную метку, образуют целостную базу данных в соответствии с набором ограничений целостности, существовавшим в соответствующий момент времени. В противном случае целостность версий обеспечивается не обязательно. Действительно, предположим, что таблица EMP является системно-версионной, а таблица DEPT — нет. Пусть некоторая строка

emp_row таблицы EMP стала исторической (в результате обновления) в момент времени t_0. Пусть эта строка ссылается на некоторую строку dept_row таблицы DEPT. В момент t_0 ссылочная целостность поддерживается. Затем в момент t_1 ($t_1 > t_0$) строка dept_row из таблицы DEPT удаляется. После этого ссылка на dept_row из emp_row становится ложной (в базе данных вообще нет строки, на которую ссылается emp_row). Это замечание относится не только к ограничениям ссылочной целостности, но и ко всем ограничениям, затрагивающим более чем одну таблицу.

4.4 Запросы в системно-версионных таблицах

Имя системно-версионной таблицы можно просто указывать в разделе FROM запроса, и тогда в расчет принимаются только текущие системные строки этой таблицы. Чтобы можно было выбрать исторические системные строки, в стандарте SQL:2011 предлагаются синтаксические расширения конструкции table reference. Фрагмент набора синтаксических правил приведен на рис. 15.

```
<table primary> ::=
<table or query name> [<query system time period
specification>] ...

<query system time period specification> ::=
FOR SYSTEM_TIME AS OF <datetime value expression>
| FOR SYSTEM_TIME BETWEEN [ASYMMETRIC | SYMMETRIC]
<datetime value expression> AND <datetime value
expression>
| FOR SYSTEM_TIME FROM
<datetime value expression> TO <datetime value
expression>
```

Рис. 15. Синтаксис ссылки на системно-версионную таблицу в разделе FROM запроса

```
(a)
SELECT ENo, EName, EMP_Sys_start, EMP_Sys_end
FROM EMP FOR SYSTEM_TIME AS OF
TIMESTAMP '02.01.2011 00:00:00'

(b)
SELECT ENo, EName, EMP_Sys_start, EMP_Sys_end
FROM EMP FOR SYSTEM_TIME FROM
TIMESTAMP '02.01.2011 00:00:00' TO
TIMESTAMP '31.12.2011 00:00:00'

(c)
SELECT ENo, EName, EMP_Sys_start, EMP_Sys_end
FROM EMP FOR SYSTEM_TIME BETWEEN
TIMESTAMP '02.01.2011 00:00:00' AND
TIMESTAMP '31.12.2011 00:00:00'

(d)
SELECT ENo, EName, EMP_Sys_start, EMP_Sys_end
FROM EMP FOR SYSTEM_TIME FROM
TIMESTAMP '0001-01-01 00:00:00' TO
TIMESTAMP '9999-12-31 23:59:59'
```

Рис. 16. Примеры запросов к системно-версионной таблице

Как показывает синтаксис на рис. 15, имеются три основные возможности указания временных характеристик исторических системных строк при указании ссылки на системно-версионную таблицу в разделе FROM запроса. Спецификация FOR SYSTEM TIME AS OF инструктирует систему о потребности выбрать все строки таблицы, период системного времени которых включает заданную временную метку (ее значение должно быть не меньше значения начального столбца и меньше значения конечного столбца периода системного времени строк-кандидатов на выборку). Пример операции выборки со спецификацией FOR SYSTEM TIME AS OF показан на рис. 16 (a).

Спецификация FOR SYSTEM_TIME FROM … TO … означает, что кандидатами на выборку должны являться все строки указанной системно-версионной таблицы, у которых значение столбца начала периода системного времени меньше значения второй заданной временной метки, а значение столбца конца периода больше значения первой временной метки. Другими словами, кандидатами на выборку являются все строки, которые являлись текущими системными строками в период системного времени, указанный в запросе. Пример операции выборки со спецификацией FOR SYSTEM_TIME FROM … TO … показан на рис. 16 (b).

Наконец, спецификация FROM EMP FOR SYSTEM_TIME BETWEEN … AND … говорит о том, что кандидатами на выборку должны являться все строки

указанной системно-версионной таблицы, у которых значение столбца начала периода системного времени не больше значения второй заданной временной метки, а значение столбца конца периода больше значения первой временной метки. При этом допускается, чтобы у обоих заданных в запросе временных меток было одно и то же значение. По сути, запрос с BETWEEN отличается от запроса с FROM тем, что в число строк-кандидатов попадают те, для которых значение столбца конца периода системного времени совпадает со значением первой временной метки (концом заданного в запросе периода). Пример операции выборки со спецификацией FROM EMP FOR SYSTEM_TIME BETWEEN … AND … показан на рис. 16 (c).

Как видно из синтаксиса на рис. 15, спецификация с BETWEEN может включать ключевые слова SYMMETRIC или ASYMMETRIC (по умолчанию полагается задание ASYMMETRIC). При использовании варианта ASYMMETRIC требуется, чтобы значение второй временной метки было не меньше значения первой временной метки. Если же задано ASYMMETRIC, и значение второй временной метки оказывается меньше значения первой временной метки (в общем случае эти значения вычисляются на стадии выполнения запроса), то эти временные метки меняются местами — вторая метка становится первой, а первая — второй.

Наконец, как показывает рис. 16 (d), из системно-версионной таблицы можно выбрать все строки, и исторические, и текущие.

Замечание 4.5.1. Описанные выше конструкции, безусловно, работают в запросах над одной системно-версионной таблицей. Однако спецификация требуемого от выбираемых строк периода системного времени при указании ссылки на таблицу может привести к получению бессмысленных результатов, если в разделе FROM запроса указывается несколько ссылок на таблицы и хотя бы в одной из этих ссылок содержатся обсуждавшиеся выше спецификации. Например, может ли быть осмысленным результат естественного соединения исторических записей из EMP и DEPT, если для этих системно-версионных таблиц в разделе FROM заданы разные периоды? Еще интереснее (и бессмысленнее) соединять исторические строки с текущими. Мне кажется, что здесь видна явная существенная недоработка стандарта SQL:2011. Интересно, что аналогичные проблемы имеются и в реализациях. Например, в Microsoft SQL Server 2016 реализована близкая к стандарту поддержка системно-версионных таблиц, и для обхода (хотя бы частичного) отмеченных проблем рекомендуется накрывать используемые в запросах системно-версионные таблицы представлениями, чтобы в запросе использовалась только одна ссылка на таблицу [17]. Понятно, что в целом это проблему не решает.

5. Битемпоральные таблицы

Понятно, что понятия прикладного и системного времени являются ортогональными, и никто не мешает определить таблицу, которая

одновременно являлась бы и таблицей с периодом прикладного времени, и таблицей с периодом системного времени. Такие таблицы уместно называть *битемпоральными*, хотя в стандарте SQL:2011 этот термин не используется (как и термин *темпоральный* вообще). На рис. 17 показано возможное определение битемпоральной таблицы EMP.

```
CREATE TABLE EMP (
EMP_NO EMP_NO,
EMP_DEPT_NO DEPT_NO,
EMP_NAME VARCHAR(30),
EMPStart DATE NOT NULL,
EMPEnd DATE NOT NULL,
PERIOD FOR EMPPeriod (EMPStart, EMPEnd),
EMP_Sys_start TIMESTAMP(12)
GENERATED ALWAYS AS ROW START,
EMP_Sys_end TIMESTAMP(12)
GENERATED ALWAYS AS ROW END,
PERIOD FOR SYSTEM_TIME
(ENP_Sys_start, EMP_Sys_end),
PRIMARY KEY (EMP_NO, EMPPeriod
WITHOUT OVERLAPS),
FOREIGN KEY (EMP_DEPT_NO, PERIOD EMPPeriod)
REFERENCES (DEPT_NO, PERIOD DEPTPeriod)
) WITH SYSTEM VERSIONING
```

Рис. 17. Битемпоральная таблица EMP

Битемпоральные таблицы полезны в тех случаях, когда требуется знать системное время изменений состояния базы данных, отражающие изменения состояния моделируемых в ней объектов в прикладном времени. Например, служащая с фамилией Smith может изменить свою фамилию на Brown 25 мая 2017 г. (например, выйти замуж). 25.05.2017 это официально фиксируется в паспорте, но соответствующее изменение строки таблицы EMP происходит только через месяц — 26 июня 2017 г. (по окончанию медового месяца) Тогда период прикладного времени для новой строки будет начинаться с 25.05.2017, а период системного времени, в котором эта строка является текущей, — с 26.06.2017. Эта информация, например, позволяет исправить внешние документы, в которых служащая именовалась девичей фамилией и которые были произведены с 25 мая по 25 июня 2017 г.

Как и в случае таблиц с периодом прикладного времени, над битемпоральными таблицами можно выполнять как традиционные операции UPDATE и DELETE, так и операции обновления и удаления с ключевыми словами FOR PORTION OF. Как и в случае таблиц с периодом системного времени, обновлять и удалять можно только текущие системные строки, и в результате выполнения этих операций в таблицу вставляются соответствующие исторические системные строки.

Запросы над битемпоральными таблицами могут включать и предикаты прикладного времени, и спецификации периодов системного времени. Пример запроса показан на рис. 18 (найти номера служащих, у которых строка имени равнялась ‘Brown' 25 мая 2017 г. прикладного времени и являлась текущей системной строкой 25 июня 2017 г.).

```
SELECT EMP_NO
FROM EMP FOR SYSTEM_TIME AS OF
   TIMESTAMP '25.06.2017 23:59:59'
WHERE EMP_NAME = ‘Brown' AND
EMPPeriod CONTAINS DATE '25.05.2017'
```

Рис. 18. Пример запроса к битемпоральной таблице

6. Заключение

14 лет тому назад в Москву приезжал бессменный редактор стандартов SQL Джим Мелтон [18]. Один из моих вопросов, не вошедших в опубликованное интервью, касался темпоральных расширений SQL. Меня удивляло, что при наличии хорошо разработанных предложений по расширению SQL темпоральными средствами [4] комитет по стандартизации SQL затягивает принятие соответствующего стандарта. Тогда Джим ничего не сказал насчет того, что TSQL2 не удовлетворил стандартизаторов, а лишь заметил, что комитет руководствуется пожеланиями основных вендоров SQL-ориентированных СУБД, а те, в свою очередь, стараются удовлетворять пожелания своих пользователей.

В 2011-м году время пришло, и в стандарте SQL появились темпоральные расширения. Они основаны вовсе не на TSQL2, как можно было ожидать в конце прошлого века. По мнению Дейта [8] основным источником принятых приложений является IBM (некоторые подтверждения этой гипотезы можно найти в [10]). И у принятых предложений имеется ряд достоинств:

- разумное использование базовой инфраструктуры SQL;
- простота и понятность;
- понятная и сравнительно простая возможная реализация в СУБД, поддерживающих предыдущие стандарты SQL.

Наверное, Дейт, Дарвен и Лоренцос в [8] правильно указывают на неполноту и ограниченность решения, внедренного в стандарт SQL, но на мой взгляд даже одно достоинство простоты перевешивает концептуальные недостатки этого решения. Такой простотой не отличается собственное решение проблемы темпоральных баз данных, предлагаемое Дейтом и др. в [8], а сложность отпугивает потенциальных пользователей.

А вот добиться того, чтобы с использованием новых конструкций SQL пользователи и разработчики приложений не могли получать бессмысленные результаты, упомянутые в данной статье, просто необходимо. Иначе пользователи не будут доверять темпоральным средствам. Пока это

стандартизаторами SQL не сделано, хотя вышла уже следующая версия стандарта [19]. Придется еще подождать…

Список литературы

[1]. Константин Пьянзин. Проблема Y2K. Журнал сетевых решений/LAN, 1998, № 05

[2]. С.Д. Кузнецов. Основы баз данных. 2-е изд. Бином. Лаборатория знаний, Интернет-университет информационных технологий, Москва, 2007, 488 стр.; лекции 11 и 14

[3]. С.Д. Кузнецов, Б.Б.Костенко. История и актуальные проблемы темпоральных баз данных. Труды Института системного программирования, т. 13, часть 2, М., ИСП РАН, 2007, стр. 77-114

[4]. Richard T. Snodgrass, editor. The TSQL2 Temporal Query Language. Kluwer Academic Publishers, 1995, 674+xxiv p. Спецификация языка доступна online на ftp://ftp.cs.arizona.edu/tsql/tsql2/bookspec.pdf

[5]. Hugh Darwen, C.J. Date. An overview and Analysis of Proposals Based on the TSQL2 Approach. In Date on Database: Writings 2000-2006, C.J. Date, Apress, 2006. Драфт статьи доступен online на http://www.dcs.warwick.ac.uk/~hugh/TTM/OnTSQL2.pdf

[6]. ISO/IEC 9075-2:2011, Information technology—Database languages—SQL—Part 2: Foundation (SQL/Foundation), 2011

[7]. SQL:20nn Working Draft Documents. http://www.wiscorp.com/sql20nn.zip., 2011-12-21

[8]. C. J. Date, Hugh Darwen, Nikos A. Lorentzos. Time and Relational Theory. Temporal Databases in the Relational Model and SQL. Morgan Kaufmann; 2 edition, July 30, 2014

[9]. Krishna Kulkarni, Jan-Eike Michels. Temporal features in SQL:2011. ACM SIGMOD Record, Volume 41, Issue 3, September 2012, pp. 34-43

[10]. Cynthia Saracco, Matthias Nicola, and Lenisha Gandhi. A matter of time: Temporal data management in DB2 10. IBM developerWorks, https://www.ibm.com/developerworks/data/library/techarticle/dm-1204db2temporaldata/

[11]. Philipp Salvisberg. Multi-temporal Features in Oracle 12c. https://www.salvis.com/blog/2014/01/04/multi-temporal-database-features-in-oracle-12c/

[12]. Teradata. Temporal Table Support. http://www.info.teradata.com/HTMLPubs/DB_TTU_13_10/index.html#page/SQL_Reference/B035_1182_109A/title.01.2.html

[13]. Koen Verbeeck. Introduction to SQL Server 2016 Temporal Tables. https://www.mssqltips.com/sqlservertip/3680/introduction-to-sql-server-2016-temporal-tables/

[14]. Clark Dave. Historical records with PostgreSQL, temporal tables and SQL:2011. http://clarkdave.net/2015/02/historical-records-with-postgresql-and-temporal-tables-and-sql-2011/

[15]. С. Д. Кузнецов. Стандарт SQL во втором десятилетии XXI века. 181 заседание семинара Московской секции ACM SIGMOD. 21 января 2016 г. http://synthesis.ipi.ac.ru/sigmod/seminar/s20161229

[16]. Резниченко В.А. Темпоральный SQL:2011. Інженерія програмного забезпечення, № 3 – 4 (15 – 16), 2013, стр. 48-65

[17]. Carl Rabeler, Craig Guyer. Querying Data in a System-Versioned Temporal Table. https://docs.microsoft.com/en-us/sql/relational-databases/tables/querying-data-in-a-system-versioned-temporal-table

[18]. Леонид Черняк. Джим Мелтон о судьбе стандарта SQL. Открытые системы. СУБД, № 6, 2003. https://www.osp.ru/os/2003/06/183149/

[19]. New Edition of Database Language SQL Standard Published. https://share.ansi.org/Shared%20Documents/News%20and%20Publications/Links%20Within%20Stories/SQL%20standard%20published_POST.pdf

Temporal Extensions in the SQL Standard

S.D. Kuznetsov <kuzloc@ispras.ru>
Ivannikov Institute for System Programming of the RAS,
25, Alexander Solzhenitsyn Str., Moscow, 109004, Russia.
Lomonosov Moscow State University,
GSP-1, Leninskie Gory, Moscow, 119991, Russia.
Moscow Institute of Physics and Technology (State University),
9 Institutskiy per., Dolgoprudny, Moscow Region, 141700, Russia
National Research University Higher School of Economics (HSE)
11 Myasnitskaya Ulitsa, Moscow, 101000, Russia

Abstract. The paper considers main features, advantages and disadvantages of temporal extensions of the SQL language introduced in the SQL:2011. The basic concepts of the bitemporal model of SQL are introduced: time period, application-time period, system-time period, application-time period table, system-version table. Methods for defining tables with an application-time period, the specifics of executing UPDATE and DELETE operations over such tables, and the specifics of primary and foreign keys in tables with an application time period are discussed. A set of additional predicates for working with tables with a application-time period is described and examples of queries to such tables are given. The rules for determining system-version tables are considered, the features of queries to such tables are discussed. Finally, the possibility of determining the bitemporal tables containing periods of both application and system time is discussed, and examples of queries are given. In the course of the presentation, the advantages and disadvantages of the bitemporal model of SQL are noted.

Keywords: SQL:2011; time period; application-time period; system-time period; application-time period table; system-version table; bitemporal table

References

[1]. Konstantin Pianzin. The Y2K problem. LAN Magazine, 1998, issue 05 (in Russian)
[2]. S.D. Kuznetsov. Database essentials. 2nd edition. Binom. Laboratory of Knowledge, Internet-University for Information Technology, Moscow, 2007, 488 pp.; lectures 11 and 14 (in Russian)
[3]. S.D. Kuznetsov, B.B. Kostenko. History and current problems of temporal databases. Proc. ISP RAS, vol. 13, part 2, M., ISP RAS, 2007, pp. 77-114 (in Russian)
[4]. Richard T. Snodgrass, editor. The TSQL2 Temporal Query Language. Kluwer Academic Publishers, 1995, 674+xxiv p. Language specification is available online at ftp://ftp.cs.arizona.edu/tsql/tsql2/bookspec.pdf
[5]. Hugh Darwen, C.J. Date. An overview and Analysis of Proposals Based on the TSQL2 Approach. In Date on Database: Writings 2000-2006, C.J. Date, Apress, 2006. A draft is available online at http://www.dcs.warwick.ac.uk/~hugh/TTM/OnTSQL2.pdf
[6]. ISO/IEC 9075-2:2011, Information technology—Database languages—SQL—Part 2: Foundation (SQL/Foundation), 2011

[7]. SQL:20nn Working Draft Documents. http://www.wiscorp.com/sql20nn.zip., 2011-12-21
[8]. C. J. Date, Hugh Darwen, Nikos A. Lorentzos. Time and Relational Theory. Temporal Databases in the Relational Model and SQL. Morgan Kaufmann; 2 edition, July 30, 2014
[9]. Krishna Kulkarni, Jan-Eike Michels. Temporal features in SQL:2011. ACM SIGMOD Record, Volume 41, Issue 3, September 2012, pp. 34-43
[10]. Cynthia Saracco, Matthias Nicola, and Lenisha Gandhi. A matter of time: Temporal data management in DB2 10. IBM developerWorks, https://www.ibm.com/developerworks/data/library/techarticle/dm-1204db2temporaldata/
[11]. Philipp Salvisberg. Multi-temporal Features in Oracle 12c. https://www.salvis.com/blog/2014/01/04/multi-temporal-database-features-in-oracle-12c/
[12]. Teradata. Temporal Table Support. http://www.info.teradata.com/HTMLPubs/DB_TTU_13_10/index.html#page/SQL_Reference/B035_1182_109A/title.01.2.html
[13]. Koen Verbeeck. Introduction to SQL Server 2016 Temporal Tables. https://www.mssqltips.com/sqlservertip/3680/introduction-to-sql-server-2016-temporal-tables/
[14]. Clark Dave. Historical records with PostgreSQL, temporal tables and SQL:2011. http://clarkdave.net/2015/02/historical-records-with-postgresql-and-temporal-tables-and-sql-2011/
[15]. S.D. Kuznetsov. The SQL standard in the second decade of the XXI century. 181 session of seminar of the Moscow section of ACM SIGMOD. January 21, 2016 http://synthesis.ipi.ac.ru/sigmod/seminar/s20161229 (in Russian)
[16]. Carl Rabeler, Craig Guyer. Querying Data in a System-Versioned Temporal Table. https://docs.microsoft.com/en-us/sql/relational-databases/tables/querying-data-in-a-system-versioned-temporal-table
[17]. Leonid Cherniak. Jim Melton on the fate of SQL standard. Open Systems. DBMS, issue 6, 2003. https://www.osp.ru/os/2003/06/183149/ (in Russian)
[18]. New Edition of Database Language SQL Standard Published. https://share.ansi.org/Shared%20Documents/News%20and%20Publications/Links%20Within%20Stories/SQL%20standard%20published_POST.pdf