БАЗИ ДАНИХ, БАЗИ ЗНАНЬ ТА ІНЖЕНЕРІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

УДК 004.91
Резниченко В.А.
Институт программных систем НАН Украины

ТЕМПОРАЛЬНЫЙ SQL: 2011

Рассматривается краткая история включения темпорального SQL в стандарт ISO, а также темпоральные типы данных в SQL и возможности языка SQL:2011 по поддержанию темпоральной модели баз данных.

Розглядається коротка історія включення темпорального SQL в стандарт ISO, а також темпоральні типи даних в SQL і можливості мови SQL: 2011 по підтримці темпоральной моделі баз даних.

The article deals with a brief history of inclusion of temporal SQL in standard ISO, and temporal data types in SQL language and possibilities of SQL: 2011 language to support temporal database model.

Ключевые слова: *темпоральная модель данных, темпоральный SQL, таблица с прикладным периодом, системно-версионная таблица.*

Краткая история включения темпорального SQL в стандарт ISO

В 1995 году было принято решение о разворачивании работ по созданию нового стандарта SQL, который был назван SQL/Temporal. В связи с этим США внесло предложение по расширению соответствующих возможностей SQL, которые базировались на пионерских исследованиях коллектива под руководством Рика Снодграсса (Rick Snodgrass) из университета Аризоны. Эти предложения базировались на детально проработанных к тому времени группой Снодграсса спецификациях языка TSQL2, являющегося темпоральным расширением SQL-92. Однако некоторые члены ISO выразили сомнения по поводу этих предложений США в связи с существованием в них серьезных проблем и противоречий.

В вою очередь Великобритания внесла предложение, которое было сформулировано на основе исследований Никоса Лорентоса (Nikos Lorentzos) из университета Афины, Греция. США не согласились позицией ISO по отношению к их предложению и не поддержали предложение Великобритании. В связи с этим ANSI и ISO решили отложить дальнейшую работу по SQL/Temporal до официальной публикации версии SQL-99.

После публикации SQL-99 ни США ни Великобритания не внесли никаких новых предложений, которые бы разрешали возникшие ранее разногласия. В связи с этим в 2001 году ANSI и ISO решили прекратить деятельность по созданию стандарта SQL/Temporal.

Вторая попытка по добавлению темпоральных свойств в SQL была предпринята в 2008 году. Она началась с обсуждения, введения и принятия предложений двумя комитетами INCITS DM32.2 и ISO/IEC JTC1 SC32 WG3 по «системно-версионным таблицам» (system-versioned tables). Еще одна темпоральная черта была добавлена в SQL 2010 в году в виде «таблиц с прикладными периодами» (application-time period tables). Эти два понятия и разработанные для них соответствующие языковые средства были включены в стандарт SQL: 2011

Темпоральные типы данных в SQL

В SQL имеются следующие темпоральные типы данных (вместе с соответствующими им литералами, операторами, функциями и предикатами):

- дата (DATE)
- время (TIME)
- временная отметка (TIMESTAMP)
- интервал (INTERVAL)

Первые три описывают моменты времени, а четвертый – продолжительность времени. Первые три также получили название ***даты-времени*** (datatimes). Мы также в будем использовать этот термин, когда некоторые свойства распространяются на все эти три типа данных.

Рассмотрим все эти типы данных.

Тип данных DATE

Этот тип данных, получивший название «даты», предназначен для указания моментов времени на временной оси с использованием григорианского календаря и с дискретностью в один день. Для обозначения этого типа данных используется ключевое слово DATE, например:

```
CREATE TABLE Person (
…
Birthday DATE,
HireDate DATE,
…
)
```

Составляющими полями этого типа являются:

- YEAR – для обозначения года
- MONTH – для обозначения месяца
- DAY – для обозначения дня.

Литерал даты состоит из ключевого слова DATE и 10-символьной строки формата 'YYYY-MM-DD', где:

- YYYY – четыре цифры для представления года (значение поля YEAR)
- MM – две цифры для представления месяца (значение поля MONTH)
- DD – две цифры для представления дня (значение поля DAY)

Пример литерала даты: DATE '2014-04-21'.

Внутри литерала даты соответствующие ему поля ограничены следующим образом:

- YEAR – от 0001 до 9999
- MONTH – от 01 до 12
- DAY – от 01 до 31

Поля литерала даты также ограничены правилами григорианского календаря, например, даты '1999-04-31' или '1990-02-29' являются неправильными.

Тип данных TIME

Этот тип данных, получивший название «времени», предназначен для указания моментов времени на временной оси в формате 24 часов в сутки и с дискретностью в одну секунду. Для обозначения этого типа данных используется ключевое слово TIME, например:

```
CREATE TABLE Lecture (
…
Begin TIME,
End TIME,
…
)
```

Составляющими полями этого типа являются:

- HOUR – для обозначения часа
- MINUTE – для обозначения минуты
- SECOND – для обозначения секунды.

Литерал времени состоит из ключевого слова TIME и 8-символьной строки формата 'HH:MI-SS', где:

- HH – две цифры для представления часа (значение поля HOUR)
- MI – две цифры для представления минуты (значение поля MINUTE)
- SS – две цифры для представления секунды (значение поля DAY)

Пример, литерала времени: TIME '14:12:37'.

Внутри литерала времени соответствующие ему поля ограничены следующим образом:

- HOUR – от 00 до 23
- MINUTE – от 00 до 59
- SECOND – от 00 до 61

Значения времени задаются в SQL относительно Всемирного координированного времени (Coordinated Universal Time -UTC). Иногда UTC следует корректировать путем добавления 1 или 2 дополнительных секунд с тем, чтобы оно соответствовало астрономическому времени. В связи с этим в SQL предполагается, что минуты могут содержать 61 или 62 секунды. Вопросы о том, поддерживаются ли такие дополнительные секунды вообще, и если да, то каким образом в этом случае поддерживаются арифметические операции, отдаются «на откуп» конкретным реализациям SQL.

Дробная часть секунд. Предоставляется возможность задавать время с более низкой дискретностью, чем в одну секунду. Для этого предоставляется возможность указывать дробную часть секунды в литерале TIME, используя символ «точка». Например, литерал TIME '14:12:37.25' соответствует прибавлению 25 сотых секунды к литералу TIME '14:12:37'.

Для указания возможности использования дробной части секунд в типе данных TIME в круглых скобках указывается количество допустимых десятичных позиций. Это количество не должно превышать 6. Таким образом, по сути, имеются следующие типы времени:

TIME, TIME(0), TIME(1), TIME(2), ..., TIME(6).

По умолчанию предполагается, что TIME равно TIME(0).

Тип данных TIMESTAMP

Этот тип данных, получивший название «временной отметки», предназначен для указания моментов времени на временной оси с использованием григорианского календаря и с дискретностью в одну секунду. Для обозначения этого типа данных используется ключевое слово TIMESTAMP, например:

```
CREATE TABLE Process (
…
Begin TIMESTAMP,
End TIMESTAMP,
…
)
```

Составляющими полями этого типа являются:

- YEAR – для обозначения года
- MONTH – для обозначения месяца
- DAY – для обозначения дня
- HOUR – для обозначения часа
- MINUTE – для обозначения минуты
- SECOND – для обозначения секунды.

Литерал времени состоит из ключевого слова TIMESTAMP и 19-символьной строки формата: 'YYYY-MM-DD HH:MI-SS', который является комбинацией форматов даты и времени, разделенных пробелом.

Пример литерала временной отметки : TIMESTAMP '2014-04-21 14:12:37'

Дробная часть секунд. Как и в типе данных TIME, здесь допускается возможность уменьшить дискретность представления временной отметки включением дробной части секунд. Соответствующим образом, как и в TIME, в самом типе данных в круглых скобках указывается количество используемых десятичных знаков. Однако, в отличие от типа TIME, в типе временной отметке по умолчанию предполагается, что TIMESTAMP равно TIMESTAMP(6).

Учет часового пояса. Для типов TIME и TIMESTAMP можно дополнительно указывать часовой пояс. Для этого существуют следующие разновидности их типов:

TIME WITH TIME ZONE – время с часовым поясом

TIMESTAMP WITH TIME ZONE – временная отметка с часовым поясом

Литерал для TIME или TIMESTAMP с часовым поясом содержит дополнительную составляющую смещения часового пояса, содержащую часы и минуты, разделенные двоеточием, перед которыми стоит знак «плюс». Например:

TIME '14:12:37+2:00'

TIMESTAMP '2014-04-21 14:12:37+4:00'

Текущее и местное время. Для типов даты-времени существуют следующие функции определения текущей/локальной даты-времени:

- CURRENT_DATE – возвращает текущую дату
- CURRENT_TIME – возвращает текущее время, которое равно времени UTC со смещением часового пояса, равным +00:00
- CURRENT_TIMESTAMP – то же самое, что и в CURRENT_TIME
- LOCALTIME – возвращает текущее время без временного пояса
- LOCALTIMESTAMP - возвращает текущие дату и время без временного пояса

Примечания. Земля разделена на часовые пояса, в которых часы указывают одно и то же время, называемое местным временем. Местное время, UTC-время и смещение часового пояса связаны следующим образом:

UTC-время = местное время – смещение часового пояса

Если часовой пояс не указывается, то неявно предполагается, что задается местное время, если же часовой пояс указывается – то UTC-время.

В сеансе работы SQL по команде SET TIME ZONE можно указывать временной пояс, который может принимать значение LOCAL (по умолчанию) или значение конкретного временного смещения, например, +01:00.

Тип данных INTERVAL

Интервал – это временной промежуток. Например, «три с половиной года», «90 дней и 13 часов», «5 минут и 30 секунд». Интервал – это непрерывный отрезок времени на временной оси. Он не привязан к конкретному расположению на временной оси и в этом смысле является относительным, в отличие от моментов времени, которые являются абсолютными. Интервал – это расстояние между двумя моментами времени и в связи с этим интервалы, в отличие от моментов времени, имеют направление.

В SQL предоставляется две синтаксических формы представления литералов интервалов, которые называются:

- форма представления в виде год-месяц
- форма представления в виде день-время

Причина введения двух форм представления интервалов заключается в том, что в используемом григорианском календаре продолжительность месяцев меняется (от 28 до 31 дня), в то время как дни, часы, минуты и секунды – постоянные

форма представления литерала интервала в виде год-месяц имеет следующие три разновидности:

- INTERVAL <литерал года> YEAR – интервал задается в годах. <литерал года> - это строка, содержащая последовательность цифр, которым может предшествовать знак «+» или «-», например, '127', '+47', '-317', и указывает неограниченное количество лет.
- INTERVAL <литерал месяца> MONTH – интервал задается в месяцах. Литерал месяца задается точно так же, как и литерал года, например, , '44', '+37', '-113' и указывает неограниченное количество месяцев.
- INTERVAL <литерал года-месяца> YEAR TO MONTH – интервал задается в годах и месяцах. <литерал года-месяца> - это строка, содержащая указатель года (неограниченное целое число с возможным знаком), знак минус (-), и затем указатель месяца (целое число без знака в диапазоне 0 – 11). Например, INTERVAL '124-11' YEAR TO MONTH – интервал продолжительностью в 124 года и 11 месяцев.

Форма представления литерала интервала в виде день-время имеет намного больше разновидностей:

- INTERVAL <литерал дня > DAY – интервал задается в днях
- INTERVAL <литерал часа> HOUR – интервал задается в часах
- INTERVAL <литерал минуты> MINUTE – интервал задается в минутах
- INTERVAL <литерал секунды> SECOND – интервал задается в секундах

Здесь литералы дня, часа, минуты и секунды задаются точно так же, как и литерал года.
Примеры:
INTERVAL '337' DAY – интервал в 337 дней
INTERVAL '-140' MINUTE – интервал в -140 минут
INTERVAL -'333' SECOND – интервал в -333 секунды

- INTERVAL<литерал дня-часа > DAY TO HOUR – интервал задается в днях и часах.
- INTERVAL <литерал дня-часа-минуты> DAY TO MINUTE – интервал задается в днях, часах и минутах.
- INTERVAL <литерал дня-часа-минуты-секунды > DAY TO SECOND – интервал задается в днях, часах, минутах и секундах.
- INTERVAL <литерал часа-минуты > HOUR TO MINUTE – интервал задается в часах и минутах.
- INTERVAL <литерал часа-минуты-секунды > HOUR TO SECOND – интервал задается в часах, минутах и секундах
- INTERVAL <литерал минуты-секунды > MINUTE TO SECOND – интервал задается в минутах и секундах.

При задании приведенных выше литералов день отделяется символом пробела, а составляющие часов – двоеточием.

Все приведенные выше интервальные литералы имеют соответствующие им типы, наименования котрых состоят используемых в написании литералов ключевых слов. Например, литералу INTERVAL <литерал минуты-секунды > MINUTE TO SECOND соответствует интервальный тип INTERVAL MINUTE TO SECOND.

К именам типов могут добавляться спецификации точности представления соответствующих компонент интервала. Например, INTERVAL DAY (3), INTERVAL SECOND (6,5), INTERVAL DAY (3) TO SECOND (6). Для всех составляющих интервала кроме секунд указывается количество цифр, которые могут использоваться для их представления, а для секунд может дополнительно указываться количество цифр дробной части.

Сравнение литералов даты-времени и интервалов

Литералы типа даты-времени могут сравниваться между собой как и обычные числовые литералы. То же самое относится и к интервалам, однако при сравнении нельзя смешивать формы представления год-месяц и день-время.

Арифметические операции со временем

Обычные арифметические операции (+, - , *, /) над датами-временем и интервалами предполагают поддержание естественных правил оперирования датами, временем и интервалами согласно григорианскому календарю. Это предполагает наложение определенных естественных ограничений на использование арифметических операций. Операции над датами-временем предполагают, что их операнды являются сравнимыми, то же самое относится и к интервалам.

К первой группе операций относятся такие, которые позволяют перемещать моменты времени по временной оси в обоих направлениях. Для этого к дате-времени можно прибавлять или отнимать .интервалы, а также к интервалу прибавлять дату-время. При этом предполагается, что вариант используемого интервала совпадает с вариантом используемой даты-времени. Например:

```
TIME '12:45:00' + INTERVAL '90'
MINUTES
CURRENT_TIMESTAMP – INTERVAL
'1' DAY
DATE '2014-04-21' + INTERVAL '12-11'
YEAR TO MONTH
```

Результат имеет ту же дискретность представления даты-времени, что и операнд даты-времени. От интервала нельзя отнимать дату-время.

Ко второй группе относится единственная операция, которая дает расстояние между двумя моментами времени на временной оси – это разность между датами-временем, в результате которой получаем интервал. Например:

```
TIME '17:45:00' - TIME '10:37:00'
```

Еще одна группа операций позволяет складывать и вычитать интервалы, получая при этом интервал. При этом нельзя смешивать интервалы год-месяц с интервалами день-время. Пример:

```
(TIME '14:19:37' – TIME '12:44:50') +
INTERVAL '920' MINUTES
```

Наконец, последняя группа операций позволяет умножать или делить интервал на число, получая при этом интервал. Например:

```
INTERVAL '73-11' YEAR TO MONTH *
3
INTERVAL '12:43:18'  HOUR TO
SECOND / 6
```

В приводимой ниже таблице приводится список всех допустимых арифметических операций над датами-временем и интервалами.

1-й операнд	Операция	2-й операнд	Результат
дата-время	+ или -	интервал	дата-время
интервал	+	дата-время	дата-время
дата-время	-	дата-время	интервал
интервал	+ или -	интервал	интервал
интервал	* или /	число	интервал
число	*	интервал	интервал

Преобразование типов данных

В SQL имеется функция CAST…TO, которая позволяет производить явное преобразование типов данных литералов. В таблице ниже приводятся возможные варианты преобразования темпоральных литералов в другие типы данных и наоборот.

ИЗ\В	Число	Строка	Дата	Время	Временная отметка	Интервал год-месяц	Интервал день-время
Число			-	-	-	ДА	ДА
Строка			ДА	ДА	ДА	ДА	ДА
Дата	-	ДА	ДА	-	ДА	-	-
Время	-	ДА	-	ДА	Да	-	-
Временная отметка	-	ДА	ДА	ДА	ДА	-	-
Интервал год-месяц	ДА	ДА	-	-	-	ДА	-
Интервал день-время	ДА	ДА	-	-	-	-	ДА

Приведем ряд примеров такого преобразования:

CAST ('10:00:00' AS TIME) - строка во время
CAST ('2014-04-24' AS DATE) - строка в дату
CAST ('3' AS INTERVAL YEAR) - строка в интервал
CAST (INTERVAL '7' MINUTES AS NUMBER) - интервал в число

Извлечение полей из темпоральных данных

В SQL имеется полезная функция, EXTRACT ... FROM, которая позволяет извлекать любое поле темпорального данного (даты-времени и интервала) и представлять его в виде числа. Напомним, что полями темпорального данного являются: YEAR, MONTH, DAY, HOUR, MINUTE, SECOND, TIMEZONE_HOUR, TIMEZONE_MINUTE

Приведем примеры:

EXTRACT (YEAR FROM DATE '2014-04-24')
EXTRACT (SECOND FROM TIME '12:45:17')
EXTRACT (MINUTE FROM INTERVAL '12:43:18' HOUR TO SECOND)

Темпоральное расширение SQL: 2011

В этом разделе рассмотрим возможности SQL: 2011 по работе с темпоральными таблицами.

Периоды

Основная идея темпоральности в SQL заключается в том, что со строками таблиц могут ассоциироваться временные промежутки, которые получили название ***периодов*** (periods). Периоды напоминают интервалы в том смысле, что и те и другие описывают промежутки времени. Однако их существенное различие заключается в том, что интервалы являются относительными промежуткам времени, то есть они не привязаны к конкретному расположению на временной оси, а периоды – абсолютные временные промежутки - их начало и конец определяются конкретными моментами времени на временной оси.

Итак, период определяется двумя упорядоченными моментами времени, являющимися датами-временем, которые называются началом и концом периода. Естественно, что начало должно быть меньше конца. В SQL: 2011 период имеет так называемую замкнутую-открытую интерпретацию. То есть период представляет собой все моменты времени, между начальным и конечным, причем включая начальный момент времени, но исключая конечный.

В SQL: 2011 период – это не тип данных, а описательная составляющая таблицы.

Два момента времени, имеющие тип даты-времени и выступающие в качестве начала и конца периода, являются обычными столбцами таблицы, а период – это поименованная составляющая таблицы, которая ассоциируется с этими двумя столбцами. Предложения CREATE TABLE и ALTER TABLE обладают синтаксисом для создания, изменения и удаления определения периода. Имя периода относится к пространству имен столбцов, то есть в пределах таблицы имя периода не может совпадать с именами столбцов. Ниже приводится фрагмент определения периода в контексте определения таблицы:

EmpStart DATE,
EmpEnd DATE,
PERIOD FOR EmpPeriod (EmpStart, EmpEnd)

Здесь EmpStart и EmpEnd обычные столбцы типа DATE, а EmpPeriod – имя периода, который ассоциируется с этими датами как с началом и концом периода.

Типами данных столбцов начала и конца периода могут быть любой из типов даты-времени, причем типы обоих столбцов должны совпадать.

Литерал периода задается с использованием следующего синтаксиса:

PERIOD (<литерал даты времени 1>, <литерал даты времени 2>)
PERIOD (<литерал даты времени>, <литерал интервала>)

Например: PERIOD (DATE '2010-01-01', DATE '2011-01-01')

Для периодов определены следующие бинарные предикаты:

- OVERLAPS – два периода имеют хотя бы одну общую временную точку
- CONTAINS – первый период содержит в себе второй период
- EQUALS – оба периода равны между собой
- PRECEDES – конечная точка первого периода меньше или равна начальной точке второго периода
- SUCCEEDS – начальная точка первого периода больше или равна конечной точке первого периода
- IMMEDIATELY PRECEDES – конечная точка первого периода совпадает с начальной точкой второго период
- IMMEDIATELY SUCCEEDS – начальная точка первого периода совпадает с конечной точкой второго периода

Типы таблиц

В литературе по темпоральным базам данных рассматриваются следующие два измерения времени:

- ***допустимое время*** (valid time) – период времени, в течение которого считается, что строка таблицы корректно отражает ситуацию реального мира;
- ***время транзакции*** (transaction time) – период времени, в течение которого строка таблицы является зарегистрированной в базе данных.

Для любой строки эти два времени могут существенно отличаться. Например, информация о запланированных сроках выполнения проекта может появиться в базе данных намного раньше, чем информация о реальных сроках его выполнения.

В SQL:2011 время транзакции поддерживается посредством системно-версионных таблиц (system-versioned tables), а действительное время – с помощью таблиц с прикладными периодами (application-time period tables).Системный временной период в стандарте имеет специально выделенное имя SYSTEM_TIME, а имя прикладного временного периода определяет сам пользователь. Относительно одной таблицы можно определить не более одного системного периода и не более одного прикладного периода. Таким образом, с точки зрения темпоральности выделяются следующие четыре разновидности таблиц:

- **Нетемпоральная таблица**. Обычная таблица, в которой сохраняется самая последняя актуальная информация. Какая-либо предыстория отсутствует
- **Таблица с прикладным периодом**. В этой таблице сохраняется предыстория изменения данных

согласно их реальным изменениям в предметной области.

- **Системно-версионная таблица**. В этой таблице фиксируется история изменения текущих данных предметной области самой системой (базой данных).
- **Системно-версионная таблица с прикладным периодом (битемпоральная таблица)**. Фиксируются обе предыстории изменения данных. – прикладная и системная.

Далее рассмотримспособы опредлелния и манипулирования всеми тремя разновидностями темпоральных таблиц.

Таблицы с прикладными периодами

Итак, таблица с прикладными периодами – это такая таблица, которая с помощью фразы PERIOD содержит определение единственного в таблице периода с заданным пользователем именем периода. Кроме того, таблица содержит два дополнительных столбца, выступающих в роли начала и конца периода. Значения обоих столбцов определяются пользователем.

Приведем пример определения таблицы с прикладным периодом:

```
CREATE TABLE Employees (
    EmpNo       INTEGER,
    EmpStart    DATE,
    EmpEnd      DATE,
    EmpDept     INTEGER,
    PERIOD FOR EmpPeriod (EmpStart, EmpEnd)
```

Рассмотрим, как действуют команды вставки, обновления и удаления в таблицах с прикладными периодами.

Вставка строк в таблицу с прикладным периодом

С помощью традиционной команды INSERT можно вставлять любые строки в таблицу. Столбцы начала и конца периода трактуются как обычные столбцы с дополнительными ограничениями целостности:

- начало и конец – NOT NULL,
- начало должно быть меньше конца.

Например, следующая команда INSERT вставляет строки в таблицу Employees:

```
INSERT INTO Employees (EmpNo,
EmpStart, EmpEnd, EmpDept)
    VALUES (15, DATE '2014-01-01',
DATE '2014-04-12', 17),
                (27, DATE '2014-02-15',
DATE '2014-05-17', 5)
```

В результате получаем следующую таблицу:

EmpNo	EmpStart	EmpEnd	EmpDept
15	2014-01-01	2014-04-12	17
27	2014-02-15	2014-05-17	5

Обновление строк в таблице с прикладным периодом

Традиционная команда UPDATE может использоваться для обычного обновления строк таблицы с прикладными периодами, включая и столбцы периода. Приведем пример:

```
UPDATE Employees
  SET     EmpDept = 3
  WHERE         EmpNo = 15
```

EmpNo	EmpStart	EmpEnd	EmpDept
15	2014-01-01	2014-04-12	3
27	2014-02-15	2014-05-17	5

Кроме того, в SQL:2011 в UPDATE введена новая синтаксическая конструкция FOR PORTION для указания временного периода, на протяжении которого применяется обновление (период обновления). Такое обновление строки может приводить к вставке до двух новых строк, чтобы сохранить информацию на протяжении тех периодов, которые находятся за пределами периода обновления. При использовании этого варианта обновления пользователю не разрешается изменять начальный и конечный столбцы периода.

Выполнение этой команды производится следующим образом:

- Отыскиваются все строки, которые удовлетворяют условию фразы WHERE.
- Затем среди них отыскиваются все строки, периоды которых пересекаются с указанным во фразе PORTION. При этом соблюдается соглашение, что период содержит начальную точку, но не содержит конечную.
- Если период найденной строки полностью содержится в периоде, заданном во фразе PORTION, то строка обновляется согласно заданному условию.
- Если период найденной строки в начале и/или в конце выходит за пределы периода во фразе PORTION, то тогда строка разбивается на две или три непрерывных с точки зрения их периодов строки. Та строка, которая содержится в периоде фразы PORTION, обновляется, а одна или две других рассматриваются как вновь вставляемые.

На приводимых ниже рисунках дается графическое представление результатов выполнения команды UPDATE в зависимости от возможных вариантов пересечения исходного периода и периода обновления.

В приводимом ниже примере производится изменение номера отдела на 4 служащего с номером 15 на период с 2014-02-10 и по 2014-03-15

```
UPDATE Employees
  FOR PORTION OF EmpPeriod
  FROM DATE '2014-02-10'TO DATE '2014-03-15'
  SET EmpDept = 4
  WHERE EmpNo = 15
```

В результате получим:

EmpNo	EmpStart	EmpEnd	EmpDept
15	2014-01-01	2014-02-10	3
15	2014-02-10	2014-03-15	4
15	2014-03-15	2014-04-12	3
27	2014-02-15	2014-05-17	5

Удаление строк в таблице с прикладным периодом

Традиционная команда DELETE может использоваться для обычного удаления строк таблицы с прикладными периодами. Приведем пример:

```
DELETE FROM Employees
WHERE EmpNo = 27
```

EmpNo	EmpStart	EmpEnd	EmpDept
15	2014-01-01	2014-02-10	3
15	2014-02-10	2014-03-15	4
15	2014-03-15	2014-04-12	3

Кроме того, в SQL:2011 в DELETE, как и в UPDATE, введена новая синтаксическая конструкция FOR PORTION для указания временного периода, на протяжении которого применяется удаление (период удаления). Такое удаление всегда приводит к удалению текущей строки и к созданию одной или двух новых.

Выполнение этой команды производится следующим образом:

- Отыскиваются все строки, которые удовлетворяют условию фразы WHERE.
- Затем среди них отыскиваются все строки, периоды которых пересекаются с указанным во фразе PORTION. При этом соблюдается соглашение, что период содержит начальную точку, не содержит конечную.
- Если период найденной строки полностью содержится в периоде, заданном во фразе PORTION, то строка удаляется.
- Если период найденной строки в начале и/или в конце выходит за пределы периода во фразе PORTION, то тогда строка разбивается на две или три непрерывных с точки зрения их периодов строки. Та строка, которая содержится в периоде фразы PORTION, удаляется.

На приводимых ниже рисунках дается графическое представление результатов выполнения команды DELETE в зависимости от возможных вариантов пересечения исходного периода и периода удаления.

В следующем примере удаляется информация о служащем с номером 15 на протяжении периода с 2014-02-15 и по 2014-02-25:

```
DELETE Employees
FOR PORTION OF EmpPeriod
FROM DATE '2014-02-15'TO DATE '2014-02-25'
WHERE EmpNo = 15
```

EmpNo	EmpStart	EmpEnd	EmpDept
15	2014-01-01	2014-02-10	3
15	2014-02-10	2014-02-15	4
15	2014-02-25	2014-03-15	4
15	2014-03-15	2014-04-12	3

Первичные ключи в таблицах с прикладными периодами

В используемой нами таблице Employees имеется столбец EmpNo, который с точки зрения традиционного понимания таблицы является полем первичного ключа. То есть он должен был бы обладать ограничениями целостности UNIQUE и NOT NULL. Но, как следует из наших примеров, наличие периода EmpPeriod приводит к тому, что EmpNo не является первичным ключом, так как с одним сотрудником может быть связано множество временных периодов, каждый из которых отражает специфическое для сотрудника состояние, а именно, пребывание в том или ином отделе. В этом случае можно было бы включить в состав первичного ключа, кроме EmpNo, еще EmpStrat и/или EmpEnd, предполагая, что пара/тройка значений этих столбцов точно не повторится.

Однако, в этом случае возможны ситуации, когда периоды одного сотрудника пересекаются, что означает возможность одновременной работы в двух и более отделах. Если это действительно допускается в предметной области – то все в порядке. Если же нет – то с помощью такого первичного ключа описать это ограничение невозможно. В связи с этим в SQL:2011 была введена возможность включать в состав первичного ключа весь период с возможным указанием непересекаемости экземпляров периода в составе первичного ключа. Так для нашего примера можно определить:

PRIMARY KEY (EmpNo, EmpPeriod WITHOUT OVERLAPS)

Теперь пара (EmpNo, EmpPeriod) становится UNIQUE и NOT NULL, причем множество периодов каждого сотрудника должно быть непересекающимся. Теперь при вставке любой строки она проверяется на ее соответствие этим трем ограничениям. (UNIQUE, NOT NULL, NOT OVERLAPS)

Внешние ключи в таблицах с прикладными периодами

Для обсуждения внешних ключей и ссылочной целостности в таблицах с прикладными периодами приведем определение еще одной таблицы – таблицы отделов, в которых работают сотрудники

```
CREATE TABLE Department (
    DeptNo          INTEGER,
```

```
    DeptStart       DATE,
    DeptEnd         DATE,
    PERIOD FOR DeptPeriod (DeptStart,
DeptEnd),
    PRIMARY KEY (DeptNo, DeptPeriod
        WITHOUT OVERLAPS)
    )
```

Предположим, что период в этой таблице указывает промежутки времени, когда отдел действительно существовал.

Пусть имеются следующие две таблицы

DeptNo	DeptStart	DeptEnd
5	2014-03-01	2014-07-30
17	2010-01-01	2014-05-01

EmpNo	EmpStart	EmpEnd	EmpDept
15	2014-01-01	2014-04-12	17
27	2014-02-15	2014-05-17	5

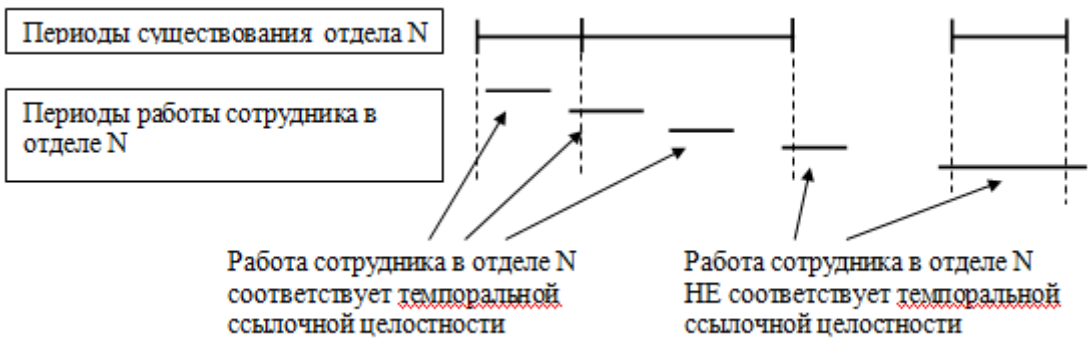
На первый взгляд ссылочная целостность у нас присутствует в том смысле, что отделы, имеющиеся в таблице сотрудников, также имеются в таблице отделов. Но мы хотели бы вкладывать более глубокий смысл в ссылочную целостность в темпоральных таблицах, а именно, выражаясь в терминах этих двух таблиц: чтобы сотрудник в любой момент времени работал в отделе, который в этот момент времени действительно существовал. Если обратиться к нашим таблицам, то для сотрудника 15 и отдела 17 это правило выполняется, однако 27-й сотрудник поступил на работу в 5-й отдел до того, как этот отдел был создан.

Обобщенный вариант поддержания ссылочной целостности в таблицах с прикладным периодом на примере отделов-сотрудников приведен на рисунке ниже:

Итак, ссылочная целостность для таблиц с прикладными периодами означает следующее: для любой строки дочерней таблицы найдется такое объединение непрерывных периодов соответствующих строк родительской таблицы, что этот объединенный период содержит в себе период рассматриваемой строки дочерней таблицы. Заметим, что объединенный период может состоять из одного периода.

Именно такую интерпретацию имеет ссылочная целостность для таблиц с прикладными периодами, выражающаяся следующей фразой внешнего ключа, которая вставляется в определение таблицы Employees:

```
    FOREIGN KEY (EmpDept, PERIOD
EmpPeriod)
    REFERENCES Department (DeptNo,
PERIOD DeptPeriod)
```



Запросы к таблицам с прикладными периодами

К этим таблицам в полной мере применим синтаксис запросов для обычных таблиц. Приведем ряд примеров.

Запрос: кто работал в отделе 5 1 марта 2014 года:

```
SELECT EmpNo
  FROM Employees
  WHERE EmpStart <= DATE '2014-03-01'
AND
      EmpEnd > DATE '2014-03-01'
```

Запрос: в каком отделе сейчас работает сотрудник 15:

```
SELECT EmpDept
  FROM Employees
  WHERE EmpNo = 15 AND  EmpStart <=
CURRENT_DATE AND
      EmpEnd > CURRENT_DATE
```

Запрос: в скольких отделах работал сотрудник 15 после 25 июня 2013 года:

```
SELECT Count (Distinct EmpDept)
  FROM Employees
  WHERE EmpNo = 15  AND EmpStart >
DATE '2013-06-15' AND
                  EmpEnd  <
CURRENT_DATE
```

Можно также использовать предикаты над периодами, например:

Запрос: вывести сотрудников, которые проработали в отделе 5 хотя бы в один из дней следующего периода 01-07-2012 – 30-08-2013:

```
SELECT EmpNo
  FROM Employees
  WHERE EmpDept = 5 AND
      EmpPeriod OVERLAPS
      PERIOD(DATE '2012-07-01', DATE
'2013-08-30')
```

Запрос: вывести сотрудников, которые проработали в отделе 5 как минимум на протяжении следующего периода 01-07-2012 – 30-08-2013:

```
SELECT EmpNo
  FROM Employees
  WHERE EmpDept = 5 AND
      EmpPeriod CONTAINS
      PERIOD(DATE '2012-07-01', DATE
'2013-08-30')
```

Системно-версионные таблицы

Системно-версионная таблица – это такая таблица, в которой помимо ее текущего актуального состояния хранится вся ее предыстория, имеется в виду предыстория существования каждой строки таблицы. При создании строки фиксируется момент времени ее создания. При любом изменении строки запоминается ее предыдущее значение вместе с датой ее создания и датой, до которой она оставалась такой, а также запоминается новое измененное значение вместе с датой, когда произошло изменение. Эта строка становится текущей актуальной строкой. Если строка удаляется, то она также запоминается вместе с датой ее удаления, и так далее. Таким образом, любое изменение или удаление строки приводит к автоматическом запоминанию ее старого варианта.

Для запоминания временных характеристик с каждой строкой, как и в таблице с прикладным периодом, ассоциируется временной период, однако пользователь НЕ может изменять его значения. Содержимым такого периода управляет система.

Строки, периоды которых содержат текущую дату, называются ***текущими системными строками***. Все остальные называются ***историческими системными строками***. Только текущие системные строки можно обновлять и удалять. Все накладываемые на таблицу ограничения целостности имеют отношение только к текущим системным строкам.

Итак, с точки зрения синтаксиса определения, таблица, которая содержит определение периода с помощью фразы PERIOD со стандартным предварительно определенным именем SYSTEM_TIME, а также в своем определении имеет спецификацию WITH SYSTEM VERSIONING, является системно-версионной таблицей.

Приведем пример определения системно-версионной таблицы:

```
CREATE TABLE Employees (
  EmpNo INTEGER,
  System_start TIMESTAMP(12)
GENERATED ALWAYS AS ROW
START,
  System_end TIMESTAMP(12)
GENERATED ALWAYS AS ROW END,
  EmpName VARCHAR(30),
  DepNo INTEGER
  PERIOD FOR SYSTEM_TIME
(System_start, System_end)
) WITH SYSTEM VERSIONING
```

Прокомментируем это описание:

System_start, System_end	–	два столбца начала и конца периода. Имена столбцов определяются пользователем.
GENERATED ALWAYS…	–	эта фраза указывает, что значение соответствующего столбца автоматически генерируется системой, делается это всегда.
AS ROW START, AS ROW END	–	в каком качестве выступает соответствующий столбец – в качестве начальной точки периода или конечной точки периода.
SYSTEM_TIME	–	предварительно определенное имя периода, предназначенное только для системно-версионной таблицы.
WITH SYSTEM VERSIONING	–	указание того, что это определение системно-версионной таблицы

Рассмотрим, как работает механизм версионности при манипулировании таблицей.

Вставка строки. При вставке строки в системно-версионную таблицу начальная точка интервала принимает значение времени выполнения соответствующей транзакции, а конечная точка принимает значение максимально допустимое для значений данного типа. Все строки, в ставленые в одной транзакции, принимают одно и то же значение их периодов.

Приведем пример. Пусть выполнена следующая вставка в транзакции, которая имеет следующую временную метку: и 2012-01-01 09:00:004:

```
INSERT INTO Employees (EmpNo,
EmpName)
VALUES (17, 'Bob', 5)
```

В результате получим следующую таблицу:

EmpNo	**System_start**	**System_end**	**EmpName**	**DepNo**
17	2012-01-01 09:00:004	9999-12-31 23:59:59	Bob	5

Обновление строки. Обновление производится только текущей системной строки. Пользователь НЕ может обновлять исторические системные строки. Пользователь также не может обновлять начальную и конечную точки (то есть изменять период) текущих и исторических строк. Инициирование команды UPDATE приводит к выполнению следующих действий:

- Создается копия строки, которая будет обновляться. При этом начальная точка периода остается прежней, а конечная устанавливается равной временной отметки транзакции (тем самым указывается, что эта копия перестает быть текущей строкой, так как конечная точка НЕ включается в состав периода)
- Обновляется строка согласно команде UPDATE. При этом начальная точка периода устанавливается равной временной отметки транзакции (тем самым указывается, что эта строка становится текущей системной строкой, так как начальная точка включается в состав периода).

Например, пусть выполняется следующее обновление:

```
UPDATE Employees
  SET EmpName = 'Tom'
  WHERE EmpNo = 17
```

Если предположить, что временная отметка транзакции, в которой выполнена команда, равна 2012-02-03 10:00:00, то получим следующую таблицу:

EmpNo	System_start	System_end	EmpName	DepNo
17	2012-01-01 09:00:004	2012-02-03 10:00:00	Bob	5
17	2012-02-03 10:00:00	9999-12-31 23:59:59	Tom	5

Отметим, что исторические системные строки, полученные в результате последовательных обновлений заданной строки, образуют непрерывную цепочку без временных промежутков между их периодами.

Удаление строки. Удалить можно только текущую системную строку. Пользователь НЕ может удалить исторические системные строки. Пользователь также не может удалить начальную и конечную точки текущих и исторических строк. Команда DELETE в системно-версионной таблице НЕ удаляет строку, а изменяет конечную точку периода на временную отметку транзакции, указывая тем самым, что строка перестает быть текущей. Например, если команда:

```
DELETE FROM Employees
  WHERE EmpNo = 17
```

выполнена транзакцией с временной отметкой 2012-06-01 00:00:17, то в результате наша таблица изменится следующим образом:

EmpNo	System_start	System_end	EmpName	DepNo
17	2012-01-01 09:00:004	2012-02-03 10:00:00	Bob	5
17	2012-02-03 10:00:00	2012-06-01 00:00:17	Tom	5

Первичный и внешний ключи в системно-версионной таблице

В этом случае описание и поддержание первичного и внешнего ключа намного проще, чем в таблице с прикладным периодом. Это объясняется тем, что эти ограничения целостности распространяются только на текущие строки. Исторические строки – это неизменяемые отпечатки прошлого. Ограничения целостности по отношению к историческим строкам в свое время уже были проверены, когда они были текущими, поэтому необходимость в этом для них отпадает. В связи с этим в определение первичного и внешнего ключа нет необходимости включать периоды.

Например, к нашей таблице можно добавить следующее определение первичного ключа:

```
PRIMARY KEY (EmpNo)
```

И это будет означать, что в этой таблице всегда будет присутствовать только одна текущая системная строка с заданным значением EmpNo.

Те же рассуждения имеют место и для ограничения целостности внешнего ключа.

Запросы в системно-версионной таблице

В связи с тем, что что системно-версионные таблицы хранят всю предысторию изменений строк таблицы, то в SQL: 2011 предоставляются дополнительные возможности по организации поиска, которые кратко описаны ниже. Суть этих возможностей заключается во введении дополнительной фразы FOR SYSTEM_TIME AS, во фразе FROM. Она имеет три разновидности. Первая из них имеет следующий синтаксис:

```
FOR SYSTEM_TIME AS OF <выражение даты-времени>
```

и позволяет формулировать запросы по отношению к определенному моменту времени.

Например, следующий запрос выбирает все строки таблицы Employees, которые были текущими 2 марта 2014 года

```
SELECT *
  FROM Employees
  FOR SYSTEM_TIME AS OF
  TIMESTAMP '2011-01-02 00:00:00'
```

В следующем запросе выводится номер отдела, в котором работал Tom 17 декабря 2001 года:

```
SELECT DepNo
  FROM Employees FOR SYSTEM_TIME
  AS OF DATE '2001-12-17'
  WHERE EmpName = 'Tom'
```

Два других варианта фразы FOR SYSTEM_TIME AS имеют следующий синтаксис:

```
FOR SYSTEM_TIME FROM <
выражение даты-времени 1>
        TO < выражение даты-времени
2>
FOR SYSTEM_TIME BETWEEN <
выражение даты-времени 1>
        AND < выражение даты-
времени 2>
```

Они позволяют отыскивать исторические строки (включая и текущие) меду двумя моментами времени. Их различие заключается в том, что в первом варианте конец указанного промежутка времени не включается в поиск, а во втором – включается. Например, следующий запрос отыскивает все строки, которые были текущими, начиная с TIMESTAMP '2011-01-02 00:00:00’и до (но не выключая) TIMESTAMP '2011-12-31 00:00:00':

```
SELECT
EmpNo,EmpName,System_Start,System_E
nd, DepNo
  FROM Employees  FOR SYSTEM_TIME
```

```
FROM
    TIMESTAMP '2011-01-02
00:00:00’TO
    TIMESTAMP '2011-12-31 00:00:00'
```

С другой стороны, в следующем запросе отыскиваются все строки, которые были текущими, начиная с TIMESTAMP '2011-01-02 00:00:00’ и включая по TIMESTAMP '2011-12-31 00:00:00':

```
SELECT
EmpNo,EmpName,System_Start,System_E
nd, DepNo
 FROM Employees FOR SYSTEM_TIME
BETWEEN
    TIMESTAMP '2011-01-02
00:00:00'AND
    TIMESTAMP '2011-12-31 00:00:00'
```

Запрос: в скольких отделах работал Tom, начиная с 1 января 2000 года :

```
SELECT Count(Distinct DepNo)
 FROM Employees FOR SYSTEM_TIME
BETWEEN DATE '2000-01-01'
                    AND
CURRENT_DATE
 WHERE EmpName = 'Tom'
```

Если в запросе к системно-версионной таблице фраза FOR SYSTEM_TIME AS отсутствует, то поиск ведется только среди текущих системных строк. Например, в следующем запросы находится отдел, в котором Tom работает в настоящее время.

```
SELECT DepNo
 FROM   Employees
 WHERE  EmpName = 'Tom'
CREATE TABLE Employees (
 EmpName VARCHAR(50),
 EmpDept INTEGER,
 EmpStrat DATE,
 EmpEnd DATE,
 SystemStart TIMESTAMP(6)
GENERATED ALWAYS AS ROW
START,
 SystemEnd   TIMESTAMP(6)
GENERATED ALWAYS AS ROW
END,
 PERIOD FOR EmpPeriod_(EmpStart,
EmpEnd),
 PERIOD FOR SYSTEM_TIME
(SystemStart, SystemEnd),
 PRIMARY KEY (EmpName, empPeriod
WITHOUT OVERLAPS),
 FOREIGN KEY (EmpDept, PERIOD
EmpPeriod)
    REFERENCES Departments (DeptNo,
PERIOD DeptPeriod))
 WITH SYSTEM VERSIONING
```

Напомним, что в SQL: 2011 имеется функция CURRENT_TIMESTAMP, которая принимает значение текущей временной отметки. Поэтому приведенный выше запрос равносилен следующему:

```
SELECT DepNo
 FROM employees FOR SYSTEM_TIME
AS OF CURRENT_TIMESTAMP
 WHERE EmpName = 'Tom'
```

Системно-версионные таблицы с прикладными периодами

Таблица может быть как системно-версионной, так и с прикладным периодом, объединяя в себе свойства обоих таблиц. Такая таблица оказывается полезной, когда относительно некоторого события следует хранить не только период, на протяжении которого он действительно имел место в предметной области, но и период, когда об этом событии «знала» машина.

Часто в литературе таблицы, объединяющие в себе свойства системно-версионной таблицы и таблицы с прикладным периодом называют битемпоральными. Мы также будем использовать это имя.

На приводимом ниже примере дается определение битемпоральной таблицы сотрудников, которая представляет собой объединение ранее использовавшихся вариантов этой таблицы. Жирным шрифтом выделены те составляющие, которые собственно относятся к темпоральным аспектам.

В битемпоральной таблице можно в полной мере использовать возможности обоих вариантов темпоральности. Мы не будем описывать детали из применения, а приведем ряд примеров, которые, как нам кажется, многое объясняют. В приводимых примерах для простоты мы будет использовать тип DATE вместо TIMESTAMP для указания начала и конца системного периода.

1) 01.05.2012 в таблицу была внесена информация, что 12.05.2012 в отделы 13 и 25 будут приняты на работу Иванов и Петров соответственно:

```
INSERT INTO Employees (EmpName,
EmpDept, EmpStart, EmpEnd)
VALUES ('Иванов', 13, DATE '2012-05-
12', DATE '9999-12-31'),
    ('Петров', 25, DATE '2012-05-12',
DATE '9999-12-31')
```

EmpName	EmpDept	EmpStart	EmpEnd	SystemStart	SystemEnd
Иванов	13	2012-05-12	9999-12-31	2012-05-01	9999-12-31
Петров	25	2012-05-12	9999-12-31	2012-05-01	9999-12-31

2) Спустя 10 дней 10.05.2012 было обнаружена ошибка, что Иванов будет принят на работу в отдел 15, а не в отдел 13. Чтобы исправить это, следует выполнить обычную команду обновления:

```
UPDATE Employees
SET EmpDept = 15
WHERE EmpName = 'Иванов'
```

EmpName	EmpDept	EmpStart	EmpEnd	SystemStart	SystemEnd
Иванов	15	2012-05-12	9999-12-31	2012-05-10	9999-12-31
Иванов	13	2012-05-12	9999-12-31	2012-05-01	2012-05-10
Петров	25	2012-05-12	9999-12-31	2012-05-01	9999-12-31

3) 15.12.2013 в таблицу была введена информация, что на период с 01.01.2014 и по 31.01.2014 Иванов переводится в отдел 12:

```
UPDATE Employees FOR PORTION OF EmpPeriod
    FROM DATE '2014-01-01' TO DATE 2014-01-31'
    SET EmpDept = 12 WHERE EmpName = 'Иванов'
```

Здесь выполняется частичное обновление периода Иванова. В связи с этим в результате обновления одна срока обновляется и добавляются две новых. Эти три строки являются текущими . Кроме того, еще одна срока попадает в архив.

EmpName	EmpDept	EmpStart	EmpEnd	SystemStart	SystemEnd
Иванов	15	2014-01-31	9999-12-31	2013-12-15	9999-12-31
Иванов	12	2014-01-01	2014-01-31	2013-12-15	9999-12-31
Иванов	15	2012-05-12	2014-01-01	2013-12-15	9999-12-31
Иванов	15	2012-05-12	9999-12-31	2012-05-10	2013-12-15
Иванов	13	2012-05-12	9999-12-31	2012-05-01	2012-05-10
Петров	25	2012-05-12	9999-12-31	2012-05-01	9999-12-31

Обратите внимание, что в наших примерах текущими системными строками являются те, у которых столбец SystemEnd равен 9999-12-31, остальные являются историческими системными строками (то есть они уже не являются актуальными)

4) Наконец, 15.04.2014 в базе данных удаляется информация об Иванове в связи с его увольнением:

```
DELETE FROM Employees
WHERE EmpName = 'Иванов'
```

EmpName	EmpDept	EmpStart	EmpEnd	SystemStart	SystemEnd
Иванов	15	2014-01-31	9999-12-31	2013-12-15	2014-04-15
Иванов	12	2014-01-01	2014-01-31	2013-12-15	2014-04-15
Иванов	15	2012-05-12	2014-01-01	2013-12-15	2014-04-15
Иванов	15	2012-05-12	9999-12-31	2012-05-10	2013-12-15
Иванов	13	2012-05-12	9999-12-31	2012-05-01	2012-05-10
Петров	25	2012-05-12	9999-12-31	2012-05-01	9999-12-31

Удаление фиксируется переводом всех текущих строк об Иванове в разряд исторических.

Список использованных источников

1. Baumunk C. An Overview of Temporal Features in SQL:2011. / Craig Baumunk // [Електронний ресурс]. – Режим доступу: http://docs.temporaldata.com/Temporal_extensions_to_SQL_2011_20120104.pdf

2. Kulkarni K., Michels J.-E. Temporal features in SQL:2011 / Krishna Kulkarni, Jan-Eike Michels // ACM SIGMOD Record. – September 2012. – Volume 41. – Issue 3. – P. 34 – 43.

Сведения об авторе:

Резниченко Валерий Анатольевич – зам. зав. отдела Института программных систем НАН Украины, канд. физ-мат. наук, с.н.с. Научные интересы: базы данных и знаний, информационные системы, семантический веб, электронные библиотеки.

E-mail: vreznichenko_47@mail.ru