Е.В. ПУГИН

Реализация обработки временных типов данных в СУБД SciDB

Муромский институт (филиал) ФГБОУ ВПО «Владимирский государственный университет имени А.Г. и Н.Г. Столетовых», г.Муром

УДК 004.047

В статье рассматриваются проблемы обработки временных типов данных. Предлагаются примеры реализации, возможные решения и альтернативы.

В информационных системах повсеместно используются программы и операторы обработки временных типов данных. Примерами таких систем могут являться различные СУБД [9, 13], финансовые приложения, навигационные системы, системы автоматического и автоматизированного управления, программы для станков с ЧПУ. В большинстве систем этому вопросу не уделяется должного внимания, что в конечном итоге приводит к ошибкам и некорректному поведению. Это может повлечь за собой большие материальные потери. Например, при организации банковских и биржевых транзакций. Данная статья посвящена описанию проблем реализации обработки временных типов данных в информационных системах и способов их преодоления.

Проблемы, возникающие при обработке

Сложность работы с такими данными обусловлена в первую очередь определёнными естественными причинами:

1. Неравномерное вращение Земли вокруг Солнца. Из-за этого календарный год может длиться 365 или 366 дней.

2. Введение високосной секунды или секунды координации [5, 6]. Они вносятся по астрономическим наблюдениям в конце суток по всемирному времени 30 июля или 31 декабря. В такие дни после времени 23:59:59 идёт 23:59:60. Это связано с замедлением вращения Земли вокруг своей оси. Известны случаи зависания некоторых

серверов. При этом может наблюдаться высокая нагрузка на процессор и полная потеря связи с сервером.

3. Различное представление календарей. Наиболее известными являются Юлианский (до 1400 года) и Грегорианский календари [3, 7, 14]. Но также существует и большое число других календарей, которыми пользовались древние цивилизации и народы. Их использование может быть востребовано в различных архивных ИС, которые хранят ценные исторические сведения.

4 Существование часовых поясов и переходов на летнее или зимнее время. Накладывает свои ограничения в системах, где требуется представлять текущее время в виде местного времени для разных стран. Также возможна проблема при использование ИС во время или после перемещения из одного часового пояса в другой.

Также существует ряд чисто технических проблем, связанных с конкретной реализацией обработки даты и времени:

1. Проблема 2000 года [1]. Связана с тем, что во многих приложениях, разработанных в 80-90 годы 20 века, для представления года использовались только два десятичных знака – 12.03.85, 05.12.98. В таком представлении после 99 года наступал 00 год. Это приводило к массе ошибок в различных финансовых приложениях, что привело бы к огромным потерям. Решение данной проблемы также стоило организациям и предприятиям немалых затрат.

2. Проблема 2038 года [2]. В системах, использующих стандарт POSIX (UNIX-время), время представляется количеством секунд, прошедших с 1 января 1970 года. На большинстве 32-битных систем для хранения секунд используется знаковое целое число, занимающее 4 байта. Таким образом, последней датой в таком формате будет 19.01.2038 03:14:07. После этого такое поле данных будет представлять отрицательное число, которое может по-разному трактоваться различными системами (1901 или 1970 год).

3. Проблема 10000 года [8]. Аналогична проблеме 2000 года. Некорректная дата может возникнуть в системах, использующих четыре цифры для представления года. Данная проблема больше представляется теоретической, нежели практической, но на самом деле обработка дат за 10000 годом может быть важна для задач и симуляций астрономических масштабов или в системах, оцениваю-

щих и моделирующих проекты долгосрочных захоронений ядерных отходов.

Описание типов даты и времени, используемых в ИС

Подробное описание типов даты и времени, их представлений можно найти в международном стандарте ISO-8601 [4]. Ограничимся кратким их описанием. Согласно стандарту ISO-8601 можно выделить следующие типы даты и времени:

1. Дата. Формат – DD.MM.YYYY. Также, помимо dmy-представления в различных странах могут применяться mdy- и ydm-представления с разными разделителями, такими как ‘/’, ‘:’, ‘-‘, ‘.’.

2. Время (от 00:00:00 до 24:00:00). Формат – HH.MM.SS. Допускается введение дробных долей секунды. Обычно это 3, 6 или 9 знаков после запятой.

3. Временная метка (от англ. timestamp) – комбинация даты и времени. Находит огромное число применений в таких сферах, как экономика и финансы (метки транзакций), информационная безопасность (создание цифровых подписей, электронных ключей и т.д.), СУБД, сетевые протоколы, управление временем в протоколах навигации систем GPS и ГЛОНАСС. Также во временных метках используются дробные доли секунд.

4. Продолжительность (от англ. time duration) или интервал времени, не привязанный ни к какой точке во времени.

5. Временной интервал (от англ. Time interval) – интервал времени, привязанный к точке во времени. Его можно представить в виде двух точек во времени или в виде точки начала и продолжительности.

Особенности реализации временных типов данных

В большинстве языков программирования есть свои реализации обработки таких типов данных, но как можно видеть из описанных проблем, они зачастую не справляются со сложными ситуациями наподобие високосных секунд, представления дат до 1900 или 1970 года и после 3000 года. Например, данные проблемы можно наблюдать в языке C++.

Многие программисты не учитывают недостатки языков программирования и пользуются стандартными библиотеками. Отсюда

вытекает причина малого числа сторонних открытых библиотек по работе с датой и временем. Самой известной является open source библиотека boost [10, 11] и её подсистема boost::date_time. Она представляет собой богатейший набор способов обработки временных типов данных, различные конверторы времени и дат из строковых представлений и обратно, способы работы с часовыми поясами и т.п. Её преимуществом являются своевременные обновления, сочетающиеся с обратной совместимостью поддержки кода. Программистам не требуется переписывать существующий код, а только лишь необходимо обновить библиотеку и скомпилировать новую версию своего приложения.

В некоторых крупных проектах подсистема обработки времени реализована собственными силами с учётом всех нюансов и проблем, описанных выше. Примером может служить реализация временных типов данных в открытой СУБД PostgreSQL.

Рассмотрим возможные реализации каждого из типов.

1. Дата. Наилучшем хранилищем для типа данных «дата» будет являться знаковое 4-х байтное целое, хранящее число дней, начиная с определённого момента. Например, число дней от 1 января 1 года н.э. Максимальная дата – около 5883516 лет. Перевод дат в такой формат может быть осуществлён следующим образом:

$$days = year * 365 + (year / 400) * 97 + (year \% 400) / 4 + yday$$

где days – число дней с 1 января 1 г. н.э.,

year – год, предыдущий текущему,

yday – номер дня в текущем году, который элементарно рассчитывается, используя текущий месяц и день месяца,

% - деление с остатком,

/ - операция получения целой части от частного.

2. Время. Под временем здесь подразумевается суточное время от 00:00:00 до 24:00:00. Для представления такого промежутка должно хватить всего лишь 3 байт (или 4 для выравнивания по аппаратной границе), но зачастую возникает необходимость отсчитывать доли секунд. Обычно ограничиваются точность до 6 знаков после запятой, то есть максимальная разрешающая способность – 1 микросекунда. Для хранения такого количества микросекунд необходимо использование 8 байтного целого числа. Преобразования

времени в хранимые микросекунды и обратно выполняются по следующим формулам:

$$r = mcs + (s + (m + h * 60) * 60)) * 1000000$$

$$mcs = r\%1000000$$

$$s = (r/1000000)\%60$$

$$m = (r/1000000/60)\%60$$

$$h = ms/1000000/60/60$$

где mcs – число микросекунд,

s – число секунд,

m – число минут,

h – число часов,

r – число, хранимое в памяти компьютера

% - деление с остатком,

/ - операция получения целой части от частного.

3. Временная метка. Реализуется на основе типа данных «время». Размер типа в 8 байт позволяет включить в него данные о дате. Преобразования временных меток во внутреннее представление и обратно выполняются на основе формул для времени с добавлением переменных дней, месяцев и лет. Максимальная временная метка, которая может быть представлена таким способом – около 291672 лет.

4. Продолжительность. Реализуется аналогично временной метке. В результате получаем хранение некоего интервала времени от в микросекундах от 0. Продолжительность используется в различных операциях с датой и временем и может быть отрицательной величиной.

5. Временной интервал. Такой интервал реализуется посредством создания структуры данных, содержащей в себе метку начала (timestamp) и продолжительность (duration). Возможен вариант хранения двух временных меток. Размер структуры – от 12 до 16 байт в зависимости от реализации.

Помимо типов данных для нормальной работы необходима некоторая библиотека базовых функций для работы со временем. К примеру, функции определения дня недели по дате, выборка отдельных элементов даты (год, месяц, день) или времени (часы, минуты, секунды, микросекунды).

Формула определения дня недели по дате называется формулой Зеллера [12]. Для Грегорианского календаря она выглядит следующим образом:

$$h = (q + \frac{13 * (m + 1)}{5} + K + \frac{K}{4} + \frac{J}{4} - 2J)\%7$$

где h – день недели (0 – воскресенье, 1 – понедельник, 2 – вторник и т.д.)

q – день месяца,

m – номер месяца (3 – март, 4 – апрель, 5- май, …, 13 – январь, 14 – февраль),

K – год века (year % 100),

J – целая часть деления века на 100 (year / 100). Например, 19 для 1955 года.

Для хранения информации, связанной с часовыми поясами требуется введение дополнительных полей и соответствующая их обработка. Также при работе с часовыми поясами удобно хранить их символьные трёхбуквенные названия для распознавания и преобразования из текстовых представлений.

Чтобы фиксировать добавления секунд координации, необходимо постоянно следить за реализованной системой и вносить требуемые изменения.

Ещё одним из способов реализации больших дат, является их представление в виде обычных целых чисел, которые представляют собой число лет. Такая простейшая реализация может использоваться в астрономических симуляциях Вселенной, где разрешающая способность может ограничиваться не только одним годом, но и целыми тысячами лет.

Реализация обработки даты и времени в SciDB

В качестве конкретного примера рассмотрим реализацию временных типов данных на основе библиотеки boost::date_time в распределённой СУБД научного назначения SciDB [9], которая служит для обработки и хранения сверхбольших объёмов данных.

Внутри движка системы потребность в сложных манипуляциях с датой и временем не возникает. Основная задача в данном случае заключается в создании видимых пользователю СУБД типов данных и предоставлении набора функций по их обработке. Исходя из это-

го, было принято решение о вынесении реализации в отдельную библиотеку.

Разработанный плагин предоставляет пользователю 4 типа данных: время (time), дату (date), временные метки (timestamp) и продолжительность (interval). Они представляют собой следующие типы из библиотеки boost::date_time:

1. Дата – boost::gregorian::date. Данный тип предоставляет пользователю возможности по работе с Грегорианским календарём, начиная от 1400 года и заканчивая 9999. Размер типа – 4 байта. Внутреннее представление хранится в виде целого числа дней с определённого промежутка во времени. Пример объявления:

```
boost::gregorian::date d(2012, 9, 13);
```

2. Время и продолжительность основаны на типе boost::posix_time::time_duration. При работе со временем на этот тип вручную накладываются ограничения диапазона от 00:00:00 до 24:00:00. Для продолжительностей таких ограничений нет. Также они могут быть отрицательными числами. Точность данного типа составляет 1 микро- или наносекунду в зависимости от опций компиляции библиотеки boost. Размер типов – 8 байт. Внутреннее представление – число долей секунд, начиная от 0. Пример объявления:

```
//12:23:59
boost::posix_time::time_interval time(12, 23, 59);
//00:00:01.123456
boost::posix_time::time_interval interval(0, 0, 1,
123456);
```

3. Временные метки реализованы посредством типа boost::posix_time::ptime. Диапазон возможных значений такой же, как у типа date, а точность и размер, как у типа time_duration. Данный тип может предоставляет доступ, как к хранимой дате, так и ко времени по отдельности. Пример объявления:

```
//13.09.2012 12:23:59
boost::posix_time::ptime timestamp(d, time);
```

Помимо временных типов данных пользователю предоставляется набор функций для работы с этими типами. Рассмотрим некоторые из них:

1. SQL-совместимые функции получения текущей даты, времени и временной метки: current_date(), current_time(), current_timestamp(), localtime(), localtimestamp(). Пример реализации:

```
date d = day_clock::local_day();
ptime p = microsec_clock::local_time();
time_duration td = p.time_of_day();
```

2. Функции, возвращающие составляющие даты (год, месяц, день) или времени (часы, минуты, секунд, доли секунд), а также функции возвращающие дополнительные сведения. Например, номер дня в году (от 1 до 365/366), день недели. Данные функции являются обёртками (от англ. wrapper) соответствующих функций библиотеки boost. Пример:

```
date d(2012, 9, 13);
int y = d.year(); //2012
int M = d.month(); //9
time_duration td(5, 10, 15, 999999);
int m = td.minutes(); //10
int h = td.hours(); //5
int fr = td.fractional_seconds(); //999999
```

3. Функции конвертации строк во внутреннее представление и обратно. Необходимы для преобразования пользовательских данных в виде текста, полученных из других источников, и загрузки их в базу данных, а также для удобного пользователю представления дат и времени.

4. Функции создания временных типов данных из составных частей. Являются обёртками для конструкторов типов, показанных в п.2. Данные функции принимают на вход соответствующие значения и конструируют из них внутреннее представление значения, хранящееся в БД. Пример AFL-запросов:

```
create array test1 <val1:timestamp> [i=0:0,1,0];
store(build(test1, make_ts(2010, 12, 1, 5, 14, 23, 123456)), test1);
[(«2010-Dec-01 05:14:23.123456»)]
```

5. Функции сканирования похожи на функции преобразования, описанные в п.3., но предоставляют пользователю гибкость при загрузке данных, оформленных нестандартным образом. Данные функции принимают на вход шаблон с определёнными специфика-

торами и выполняют соответствующее сканирование в поисках значений.

6. Функции сравнения. Для всех типов поддерживаются любые операции сравнения друг с другом: <, <=, >, >=, =, <>.

7. Арифметические операции. Также некоторые типы поддерживают арифметические операции. Например, добавление к временной метке некоторой продолжительности и т.п.

Такая реализация основных временных типов данных и широкого набора функций предоставляют пользователю большие возможности по работе, преобразованию и обработке таких данных.

Заключение

В данной статье были проанализированы проблемы, возникающие при программирование систем, использующих временные типы данных, а также рассмотрены возможные способы их решения при построении собственной реализации. Приведены формулы нахождения некоторых из параметров. Рассмотрена реализация обработки дат и времени в СУБД SciDB.

Из-за существенного числа проблем и сложностей при их реализации настоятельно рекомендуется использовать существующие и развивающиеся открытые библиотеки обработки даты и времени, такие как boost::date_time. Необходимо отметить, что даже они не в состоянии обрабатывать большие диапазоны дат на текущий момент.

Также следует учитывать потребности информационной системы в данных типах. Если в системе они используются очень редко, и для результата не требуется высокой точности, то возможно использование стандартных средств конкретного языка программирования.

Литература

1. A. van Deursen, "The Leap Year Problem" The Year/2000 Journal 2(4):65–70, July/August, 1998

2. Diomidis Spinellis (2006). Code quality: the open source perspective.. Effective software development serie in Safari Books Online (illustrated ed.). Adobe Press. ISBN 0-321-16607-8.

3. Duncan, D. E. (1999). Calendar: Humanity's Epic Struggle To Determine A True And Accurate Year. Harper Perennial. ISBN 0-380-79324-5.

4. ISO 8601:2004(E). ISO. 2004-12-01.

5. Meeus, J. & Savoie, D. (1992). The history of the tropical year. Journal of the British Astronomical Association, 102(1), 40–42.

6. Morrison, L. and Stephenson, F. R., «Historical Values of the Earth's Clock Error ΔT and the Calculation of Eclipses», J. Hist. Astron., Vol. 35 Part 3, August 2004, No. 120, pp 327—336 (2004)

7. Moyer, G. (May 1982). "The Gregorian Calendar". Scientific American", pp. 144–152.

8. Roger Smith. "Lessons from the Long Now". Software Development Magazine.

9. Overview of SCIDB: Large scale array storage, processing and analysis. Rogers J., Zdonik S., Simakov R., Velikhov P., Smirnov A., Knizhnik K., Soroush E., Balazinska M., DeWitt D., Heath B., Maier D., Madden S., Stonebraker M., Patel J., Brown P.G. Proceedings of the ACM SIGMOD International Conference on Management of Data 2010 International Conference on Management of Data, SIGMOD '10. Сер. "Proceedings of the 2010 International Conference on Management of Data, SIGMOD '10" sponsors: ACM SIGMOD. Indianapolis, IN, 2010. С. 963-968.

10. Schäling, Boris (2011). The Boost C++ Libraries. XML Press. ISBN 978-0-9822191-9-5.

11. Siek, Jeremy G.; Lee, Lie-Quan & Lumsdaine, Andrew (2001). The Boost Graph Library: User Guide and Reference Manual. Addison-Wesley. ISBN 978-0-201-72914-6.

12. Zeller's congruence // Wikipedia, the free encyclopedia. — 2012. — 31 мая [Электронный ресурс]. URL: http://en.wikipedia.org/wiki/Zeller's_congruence (дата обращения: 06.09.2012).

13. *Пугин Е.В., Симаков Р.А.* Распараллеливание математических вычислений на примере операторов линейной алгебры / Алгоритмы, методы и системы обработки данных. 2011. № 17. С. 12-12.

14. Федеральный закон Российской Федерации от 3 июня 2011 г. N 107-ФЗ «Об исчислении времени», ст. 2, п. 2

ПУГИН Е.В.

E-MAIL: EGOR.PUGIN@GMAIL.COM