ISSN 2079-3316 ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ № 2(11), 2012, с. 77–82

УДК 004.051+004.623

С. В. Знаменский

Глобальная идентификация данных в долговременной перспективе

Аннотация.

Ключевые слова и фразы: система версионирования, хранилище данных, структура B+tree, метка времени, история изменений, долгоживущие информационные системы.

Введение

[1]

Целостная неподдельная история изменений незаменима и широко используется

- для эффективного согласования распределённых данных,
- для ускорения отладки нового кода,
- для быстрого восстановления при поломках,
- для упрощения расследования вторжений и других происшествий ,
- для повышения ответственности персонала.

Растёт потребность в организации более эффективного доступа к истории изменений перестраивающихся структур данных [1–3].

При этом накопление структурных изменений в данных способно экспоненциально увеличивать объём хранимой информации, существенно нелинейно увеличивая затраты на обработку запросов.

Мы сосредоточим внимание на истории изменений иерархических структур данных, занимающих особое место в информатике. Любые существующие структуры данных общепринято описывать в формате XML. Такое описание имеет иерархическую структуру, однозначно и полностью фиксирующую исходную структуру данных.

Работа проводилась при финансовой поддержке Министерства образования и науки Российской Федерации.

Если исходная структура перестраивается, то изменяется и иерархия XML-описания. Таким образом, любые перестройки произвольных структур данных могут качественно сохраняться в виде перестроек иерархической структуры.

Рассмотрим задачу эффективного представления истории изменений иерархической структуры данных. Иерархическая структура данных понимается как дерево,с узлами которого ассоциируются данные, сериализованные строками произвольной длины, а порядок следования детских узлов любого родителя фиксирован.

1. Базовые запросы на чтение и средства их реализации

Оптимизировать будем только базовые вида запросов, возвращающие на непосредственно предшествующий заданному момент времени для заданного в запросе узла:

(1) ассоциированные данные и время их последнего изменения;
(2) последний узел-ребёнок (включая удалённые);
(3) предыдущий узел того же родителя (возможно удалённый);
(4) вся информация ветки, выходящей из узла.

Трёх первых видов запросов достаточно для получения полной информации об истории изменений данных например, на языке SPARQL [4][1]. Четвёртый вид предназначен для организации удалённого дублирования данных.

Мы предполагаем, что родитель заданного узла определяется по идентификатору узла. Это означает, что перемещение веток осуществляется копированием и удалением[2].

Не секрет, что первый тип запросов быстрее всего обрабатывается организацией упорядоченного списка через бинарное дерево. Второй тип реализуется созданием для каждой ветки, особого узла содержащего время её модификации.

---

[1]Для ускорения обработки полезно ввести служебные узлы, индексирующие информацию; например, узел, содержащий время последнего изменения ветки, растущей из родительского узла.

[2]Эффективная поддержка перемещения веток может быть в дальнейшем реализована на описываемой основе через создание символической ссылки у нового родителя на состояние перемещённой ветки и пометки об удалении у старого. Обработка ссылок и пометок замедлит выполнение основных видов запросов.

Заменить дополнительную дублирующую структуру может семантически насыщенная система идентификации узлов. При лексикографическом упорядочении таких идентификаторов родитель предшествует ребёнку и порядок детей одного родителя сохраняется, а идентификаторы узлов любой ветки заполняют интервал упорядоченного списка. Операция чтения дополнительной структуры на диске заменяется вычислением функции, задающей границы этого интервала, и чтением основной структуры.

2. Ветвления и их идентификация

2.1. Семантическая идентификация узлов

Система идентификации узлов, обеспечивающая связность множества идентификаторов узлов любой ветки, не является чем-то доселе неизвестным. Знакомой всем системой таких идентификаторов является идентификация файлов полным путём в файловой системе ISO 9660. Схема проста: имена содержащих файл директорий и имя самого файла конкатенируются с разделителем '/', обладающим важным свойством: этот разделитель не должен при лексикографическом упорядочении оказаться между возможными продолжениями имени любого имени файла или директории. Именно это свойство обеспечивает связность множества идентификаторов, а его нарушение угрожает нарушением этой связности.

Возможны и альтернативные варианты без разделителей. Можно, например, условиться, что имя файла или директории всегда начинается с заглавной латинской буквы, а продолжается только строчными латинскими буквами.

Эти варианты нуждаются в поправке в нашей ситуации, когда начало ветки также является узлом, с которым могут ассоциироваться данные. Без поправки начало отрывается от ветки. Поправить ситуацию можно всегда добавляя в конец пути лексикографически большую строку, чем возможные продолжения имени. Наиболее экономный вариант поправки в первом варианте — использование в качестве разделителя большего байта, чем возможные символы продолжения имени, а во втором варианте — такое изменение кодировки, при котором заглавные латинские буквы займут позиции в конце кодовой таблицы.

Возможны и иные варианты. Постепенно выявляются ограничения, требующие учёта при выборе варианта. Одно из них — необходимость использовать буквы национальных алфавитов в идентификаторах с соблюдением алфавитного порядка [8].

Вопрос об обоснованном выделении схемы семантической идентификации узлов дерева требует таким образом отдельного исследования.

2.2. Уровни версионирования

Иерархическая структура не является единственной основой ветвления данных. Возможны дополнительные ветвления, связанные с различными версиями данных в системе

2.2.1. *Административные версии системы*

При анализе инцидентов и подготовке обновлений требуется проведение экспериментов на копии системы или её распределённой части. Многие такие эксперименты могут проводиться на имеющейся физической основе на виртуальной копии системы. При этом принципиальное отсутствие возможности изменить старые версии открывает безопасный путь создания виртуальной копии без фактического копирования данных. Речь идёт об одновременном поддержании нескольких версий состояний системы, отличающихся поздними изменениями.

2.2.2. *Управляемое версионирование веток*

Ветка может содержать информацию, подлежащую контролю версий, например, сложный программный код на C++. В этом случае разработчики выделяют основную и иные версии, которые могут иметь значительные общие части, но изменяться изолированно.

2.2.3. *Автоматическое версионирование*

При обработке транзакций и других сложных вычислениях возникают версии данных, контролируемые не людьми, а вычислительными процессами. Они тоже способны ветвиться[?].

2.3. Схемы версионирования

В зависимости от потребностей организаторов маркировка версий может отличаться[6]. При этом задача семантической идентификации узлов дерева версий по сути ничем не отличается от аналогичной задачи для дерева иерархии данных.

Единственный подводный камень, который здесь просматривается, — это общепринятая нелексикографическая маркировка версий одного уровня. К примеру, версия `10` лексикографически меньше, чем версия `2`. Общепринятый путь решения этой проблемы состоит в приписывании достаточного числа нулей, например, `010` уже больше, чем `002`.

Этот путь рано или поздно заведёт в тупик исчерпания доступных номеров версий. В этой ситуации обычно в пожарном порядке меняют схему именования версий вместо того, чтобы использовать простой выход из этого тупика — добавление разряда вместе с буквой. Например, `a1001` лексикографически больше, чем `999`.

3. Схема реализации и априорные оценки

[7]

Каждое изменение любого элемента данных помечается для хранения временной меткой (timestamp). Список изменений упорядочивается по времени и быстрый доступ к актуальной версии данного осуществляется делением всего упорядоченного списка изменений пополам до тех пор, пока не будет обнаружено изменение, актуальное на запрошенный момент времени.

4. Выводы

Список литературы

[1] Nierstrasz O., Denker M., G îrba T., Lienhard A., Röthlisberger D. *Change-Enabled Software Systems* // Challenges for Software-Intensive Systems and New Computing Paradigms. LNCS 5380, 2008, p. 64–79 ↑[]

[2] Miles S., Groth P., Munroe S., Moreau L. *PrIMe: A Methodology for Developing Provenance-Aware Applications* // ACM Transactions on Software Engineering and Methodology, 2009, p. 1-42 ↑

[3] Kieseberg P., Schrittwieser S., Mulazzani M., Huber M., Weippl E. *Trees Cannot Lie: Using Data Structures for Forensics Purposes* // Intelligence and Security Informatics Conference (EISIC), 2011, p. 282–285 ↑[]

[4] Préz J., Arenas M., Gutierrez C. *nSPARQL: A navigational language for RDF.e* // Web Semantics: Science, Services and Agents on the World Wide Web, 2009, p. 255–270 ↑1

[5] Jiang L., Salzberg B., Lomet D., Barrena M. *The BT-Tree: A Branched and Temporal Access Method* // VLDB '00 Proceedings, 2000, p. 451–460 ↑

[6] Zhu N. *Data Versioning Systems*, Stony Brook University, 2003 ↑2.3

[7] Знаменский С. В. *Ретроспективная основа совместной реорганизации сложных информационных ресурсов* // Электронные библиотеки: перспективные методы и технологии, электронные коллекции // Труды XIII Всероссийской научной конференции RCDL-2011. — Воронеж : Воронежский госуниверситет, 2011, с. 93–101 ↑3

[8] Знаменский С. В. *Процессный подход к эволюционированию информационной системы. Ретроспективное индексирование* // Программные системы: теория и приложения, 2011. **Т. 2**, № 4(8), с. 115–125 ↑2.1

Об авторе:

Сергей Витальевич Знаменский

Автор критерия разрешимости уравнений свёртки в пространстве функций, голоморфных на множестве, понятий выпуклости в направлении и $\mathbb{C}$-выпуклости, русификации TEX для журналов Отделения математики РАН, графического пакета `mfpic3d` и нового подхода к построению информационных систем.

e-mail: svz@latex.pereslavl.ru

Образец ссылки на эту публикацию:

С. В. Знаменский. *Глобальная идентификация данных в долговременной перспективе* // Программные системы: теория и приложения : электрон. научн. журн. 2012. Т. 3, № 2(11), с. 77–82.

URL: http://psta.psiras.ru/read/psta2012_2_77-82.pdf

S. V. Znamenskij. *Global data identification in a long term perspective.*

Abstract. (in Russian)

Key Words and Phrases: versioning, data storage, B+tree structure, timestamp, changes history, long term information system.