OSTIS-2015
(Open Semantic Technologies for Intelligent Systems)

УДК 004.822:514

МЕТОД ХРАНЕНИЯ ИЕРАРХИЧЕСКИХ СЕМАНТИЧЕСКИХ СЕТЕЙ В РЕЛЯЦИОННОЙ БАЗЕ ДАННЫХ

Ванясин Н.В., Сидоркина И.Г.

Поволжский государственный технологической университет,
г. Йошкар-Ола, Республика Марий Эл, Россия

nikita.vanyasin@gmail.com

igs592000@mail.ru

В данной статье предложено решение задачи хранения множества семантических иерархических сетей в реляционной базе данных. Решение базируется на иерархических семантических сетях, определенных в программной системе iSpring Cloud. Особое внимание уделено оптимизации существующего метода хранения, проведенной в целях уменьшения дублирования данных. В работе скомбинированы алгоритмы операций над семантическими сетями и алгоритмы хранения иерархических данных в реляционной базе данных.
Ключевые слова: семантические сети; метаданные; хранение иерархических данных.

Введение

При разработке программной системы iSpring Cloud (компания iSpringSolutions) [iSpringCloud, 2014] возникла задача хранения структур папок и файлов которые загружаются пользователями системы. Каждый пользователь в системе обладает своей автономной структурой папок и файлов и может управлять ею (создавать папки, загружать файлы, копировать и перемещать файлы и папки и т.д.). Кроме того пользователи могут делится с другими пользователями своими папками и файлами для просмотра и редактирования. Программная система уже использует реляционную базу данных, т.о. предложено организовать хранение иерархических структур папок и файлов пользователей так же в реляционной базе данных, что упростило бы разработку.

Структура папок и файлов пользователя фактически является семантической сетью с иерархическими отношениями «папка (предок) – файл (потомок)». Так как планируемое количество будущих пользователей программной системы может быть большим (от 20000 и более), а скорость отображения иерархии для конкретного пользователя должна быть достаточно высокой, и пользователь не должен ощущать задержек при работе с системой (то есть в пределах 1-2 секунд), возникает задача хранения семантических иерархических сетей для большого количества пользователей в реляционной БД (причем в таких структурах данных, которые были бы пригодны для быстрого получения иерархии при помощи операций реляционной алгебры).

Основными проблемами, стоящими на пути к решению данной задачи являются:

- У каждого пользователя есть возможность открыть доступ к своим папкам и файлам любому количеству других пользователей в системе. Соответственно при «наивной» реализации этой возможности через копирование части иерархии от одного пользователя к другому появится дополнительное количество избыточных данных, что замедлит в целом работу с семантической сетью.
- Отсутствие возможности использования рекурсивных операций в СУБД (MySQL 5.6), которая используется для работы с базой данных в программной системе iSpring Cloud.

1. Выбор метода хранения иерархических данных в реляционной БД

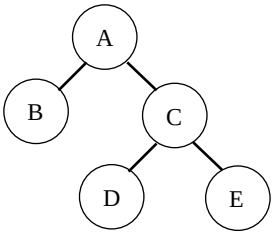
В настоящее время существует множество подходов к управлению иерархическими структурами данных. Наиболее исследованными, обладающими высокой эффективностью хранения и обработки данных в реляционных системах, являются методы управления деревьями [Маликов, 2008]. Основными методами хранения иерархических данных в реляционной системе являются:

- Adjacency List (список смежных вершин);
- Materialized Path (материализованный путь);
- Nested Sets (вложенные множества);
- Closure Table (таблица замыканий).

Рассмотрим подробнее каждый из методов.

1.1. Adjacency List (список смежных вершин)

Отношение «предок-потомок» хранится вместе с данными как простая ссылка на прямого предка.



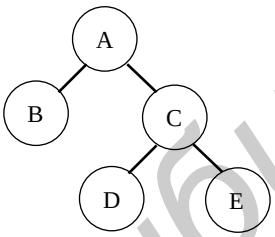
id	parent_id	данные
1	NULL	A
2	1	B
3	1	C
4	3	D
5	3	E

Рисунок 1 – Adjacency List

На рисунке 1 представлен пример иерархической структуры данных и её хранение с использованием этого метода. Основной проблемой при использовании такого способа хранения – невозможность быстро получить всю иерархию полностью, так как получение каждого следующего уровня требует выполнения операции соединения, что накладывает ограничение на максимально возможную глубину структуры (дерева) [Смусенок и др., 2013].

1.2. Materialized Path (материализованный путь)

Для каждого узла в структуре иерархических данных хранится ещё и перечисление всех его предков.



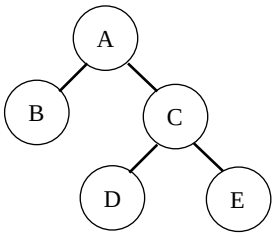
id	path	данные
1	1/	A
2	1/2/	B
3	1/3/	C
4	1/3/4/	D
5	1/3/5/	E

Рисунок 2 – Materialized Path

На рисунке 2 представлен пример иерархической структуры данных и её хранение с использованием этого метода. Так как перечисление предков хранится в одном элементе кортежа, основным недостатком данного метода являются дополнительные накладные расходы на работу с этим элементом: проверки на отсутствие циклов, конкатенация строк и т.д. [Celko, 2012]

1.3. Nested Sets (вложенные множества)

Для каждого узла в структуре иерархических данных добавляется два дополнительных поля: left (левая граница множества, некоторое число, которое меньше чем все числа которые используют потомки текущего узла) и right (правая граница множества, число, которое меньше чем все числа которые используют потомки текущего узла). То есть отношение «предок-потомок» определяется вхождением потомка в промежуток (left; right) предка.



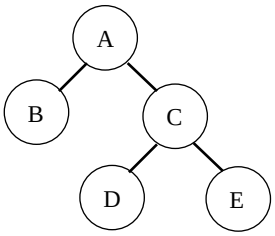
left	right	данные
1	10	A
2	3	B
4	9	C
5	6	D
7	8	E

Рисунок 3 – Nested Sets

На Рисунке 3 представлен пример иерархической структуры данных и её хранение с использованием этого метода. Использование nested sets позволяет избавится от ограничения на максимальную возможную глубину дерева, а совместное использования этого метода с методом Adjacency List позволит извлекать всю структуру иерархических данных при помощи одной операции реляционной алгебры.Однако затраты на вставку новых узлов и перенос требуют большого количества дополнительных операций для пересчета значений left и right, что особенно сильно проявляется на иерархических структурах с большой глубиной вложенности [Karwin, 2010].

1.4. Closure Table (таблица замыканий)

Используется два реляционных отношения – одно перечисляет узлы семантической сети и содержит данные, а другое перечисляет все связи (замыкания) между узлами (в том числе косвенные).



id	данные
1	A
2	B
3	C
4	D
5	E

parent_id	child_id	depth
1	1	0
1	2	1
1	3	1
1	4	2
1	5	2
2	2	0
3	3	0
3	4	1
3	5	1
4	4	0
5	5	0

Рисунок 4 – Closure Table

На Рисунке 4 представлен пример иерархической структуры данных и её хранение с использованием этого метода. Кроме самих данных хранятся метаданные семантической сети – информация для каждого узла, «родителем» каких

«потомков» он является и на каком уровне вложенности. Извлечение иерархической структуры данных при таком методе включает в себя операцию соединения этих отношений по предикатам child_id и parent_id. Извлечение структуры из реляционной системы при использовании данного метода не требует вызова рекурсивных операций, что устраняет проблему отсутствия возможности выполнения рекурсивных операций при помощи используемой в системе СУБД [Karwin, 2010]. Предикат depth может использоваться для выборки одного уровня иерархии, либо в целях оптимизации запросов в СУБД.

Данный метод хранения использует разные реляционные отношения для хранения данных

- об узлах семантической сети;
- о метаданных – связях между узлами в сети,

что позволяет использовать одни и те же узлы в семантических сетях разных пользователей, упрощая реализацию возможности совместного использования одной сети разными пользователями. По этой причине для решения поставленной задачи хранения иерархической семантической сети в реляционной базе данных был выбран метод Closure Table (таблица замыканий).

2. Описание метода хранения иерархической семантической сети

Для решения задачи хранения иерархической семантической сети предлагается добавить еще один предикат в таблицу замыканий для связи иерархической семантической сети с конкретным пользователем программной системы, что позволит хранить иерархии всех пользователей системы и совершать операции над множеством иерархий сразу. Таким образом реляционное отношение будет выглядеть следующим образом: (user_id, parent_id, child_id, depth). Данный метод хранения позволяет производить все основные операции над семантической сетью без задержек, заметных для пользователя системы.

Рассмотрим основные операции.

Выборка всей иерархии папок конкретного пользователя: SELECT * FROM node INNER JOIN closure ON closure.child_id = node.id WHERE closure.user_id = $currUserId

Добавление нового узла в иерархию папок конкретного пользователя: INSERT INTO closure (user_id, parent_id, child_id, depth) VALUES ($userId, $nodeId, $nodeId, 0); INSERT INTO closure (user_id, parent_id, child_id, depth) SELECT $userId, p.parent_id, c.child_id, (p.depth+c.depth+1) FROM closure p, closure c WHERE p.user_id=$userId AND c.user_id=$userId AND p.child_id=$parentId AND c.parent_id = $nodeId;

Первый запрос вставит замыкание узла на самого себя, а второй – добавит необходимые замыкания для всех «предков» нового узла.

Удаление поддерева: DELETE FROM closure WHERE user_id = $userId AND child_id = $nodeId;

Перемещение поддерева представляет собой сначала «отсоединение» поддерева от основной иерархии и последующее «подсоединение» путем добавления новых связей «предок-потомок».

Одним из недостатков описанного метода хранения иерархической семантической сети в реляционной системе является увеличение объема избыточных данных при использовании возможности открытия доступа к редактированию иерархии другим пользователям программной системы. При «наивной» реализации данной возможности через копирование части иерархии от одного пользователя к другому объем избыточных данных может возрасти многократно.

Чтобы исключить данный недостаток предлагается провести денормализацию таблицы замыканий и добавить дополнительный предикат – идентификатор пользователя, который владеет узлом иерархии, с целью введения связей между иерархиями семантических сетей различных пользователей. Таким образом реляционное отношение будет выглядеть следующим образом: (user_id, parent_id, child_id, depth, owner_id). Соответственно при использовании возможности открытия доступа к части иерархии другим пользователям системы, пользователи будут редактировать часть иерархии, которой владеет первый пользователь.

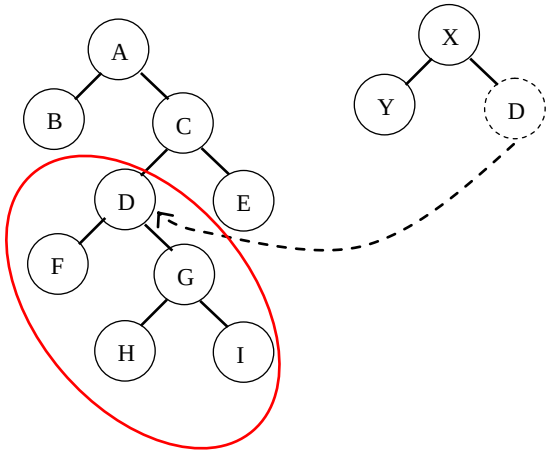


Рисунок 5 – Связь двух семантических сетей

На рисунке 5 в левой части показана иерархия первого пользователя, который открыл доступ к поддереву узла D второму пользователю (его семантическая сеть показана в правой части рисунка). Таким образом в дереве второго пользователя при открытии ему доступа добавляется только один дополнительный узел, который фактически является ссылкой на узел в другой семантической сети, что обеспечивает

построение связей между иерархическими сетями разных пользователей и уменьшение объема дублируемых данных.

Предложенное решение позволяет извлечь иерархическую структуру данных пользователя не более чем за две операции выборки:

- выборка первичных ключей всех пользователей, которые открыли доступ к своей иерархии текущему пользователю. SELECT DISTINCT owner_id FROM closure WHERE user_id = $currUserId;
- выборка всей иерархической структуры, которая доступна текущему пользователю. SELECT * FROM node INNER JOIN closure ON closure.child_id = node.id WHERE closure.user_id = $currUserId OR closure.user_id IN ($foreignTreeOwnerIds)

Заключение

В работе предложено и обосновано решение о хранении иерархической семантической сети в реляционной базе данных. Разработан метод хранения, который позволяет выстраивать связи между однотипными сетями. Представленный метод доведен до практической реализации в программной системе iSpring Cloud.

Библиографический список

[Маликов, 2008] Маликов, А.В. Ориентированные графы в реляционных базах данных / А. В. Маликов // Доклады ТУСУРа, No 2 (18), часть 2, декабрь 2008 С. 100-104

[Смусенок и др., 2013] Анализ способов представления иерархических структур в реляционных базах данных с использованием стресс-тестов / Смусенок С.А. [и др.]; // Открытые информационные и компьютерные интегрированные технологии №62, 2013 С. 107-111

[Karwin, 2010] Karwin, Bill, SQL Antipatterns: Avoiding the Pitfalls of Database Programming / Bill Karwin. – Pragmatic bookshelf. – 2010. - P. 34-53

[Celko, 2012] Celko, Joe, Trees and Hierarchies in SQL for Smarties, Second Edition / Joe Celko. – Morgan Kaufmann. – 2012. – P. 296.

[iSpringCloud, 2014] iSpring Cloud Home Page http://www.ispringcloud.com/ (дата обращения: 27.11.2014).

HIERARCHICAL SEMANTIC NETWORK STORAGE METHOD FOR RELATIONAL DATABASES

Vanyasin N.V., Sidorkina I.G.

Volga State University of Technology, Yoshkar-Ola, Republic of Mari El, Russia

nikita.vanyasin@gmail.com

igs592000@mail.ru

This article proposes a solution to the problem storing a plurality of hierarchical semantic networks in a relational database. Particular attention is paid to optimizing existing storage method, carried out in order to reduce duplication of data. In this paper algorithms combined operations on semantic networks and algorithms for storing hierarchical data in a relational database.

Keywords: semantic networks, metadata, hierarchical data storage.

Introduction

During designing a software system iSpring Cloud (iSpring Solutions Inc.) there was a problem of storage structures of folders and files that are uploaded by users of the system.

Structure of folders and files the user is actually the semantic network with hierarchical relationships "folder (ancestor) - file (descendant)". Thus, here arises a problem of semantic hierarchical networks sorage method for a large number of users in a relational database (and in such data structures that would be suitable for quick hierarchy extraction using relational algebra operations)

Main Part

The most researched, highly effective storage methods for hierarchy in relational systems are tree management method. The basic methods for storing trees in relational systems are adjacency list, materialized path, nested sets and closure table. Extracting hierarchy structures from a relational system with closure table method does not require recursive operations that eliminates the problem of inability to perform recursive operations using the database management system used in the software product.

This storage method uses different relations for storing nodes of a semantic network separate from metadata (the connections between nodes in the network), that allows the use of the same nodes in the semantic networks of different users. This storage method allows you to perform all basic operations on a semantic network without noticeable delays. Also there are proposed improvements to storage method, that are targeted to use one relation for storing sematic networks for all users. The proposed solution allows to extract a hierarchical structure of user data is not more than two sampling operations.

Conclusion

In this whitepaper we propose and justify a decision to keep a hierarchical semantic network in a relational database. Proposed method allows you to build connections between the networks of same type. The presented method is brought to the practical implementation of a software system iSpring Cloud.