УДК 004.652.2

С. А. Смусенок, А. В. Трубилко, А.А. Фурманов

Анализ способов представления иерархических структур в реляционных базах данных с использованием стресс-тестов

Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ»

Проанализированы проблемы хранения иерархических структур в реляционных базах данных. Проведено сравнение производительности существующих подходов к хранению иерархических данных. Для анализа способов представления данных разработан бенчмарк, осуществляющий измерение временных характеристик. Проведен анализ скорости работы различных подходов к хранению данных на основе полученных данных. Сформулированы рекомендации по выбору способов представления иерархических структур в зависимости от поставленной задачи.

Ключевые слова: иерархические структуры, реляционные базы данных, Nested Set, Adjacency List, Closure Tables, Materialized Path.

1. Введение

Большинство хранилищ данных разрабатывается на основе реляционных баз данных. В основном они удовлетворяют требованиям какой-либо конкретной предметной области, но периодически возникают задачи, требующие представления и хранения данных в иерархическом виде.

Существует несколько подходов к построению иерархических структур в реляционных базах данных. [1, 2] В ходе исследования были проанализированы четыре таких подхода:

- вложенное множество (Nested Set);
- список смежных вершин (Adjacency List);
- «замкнутые» таблицы (Closure Tables);
- материализованный путь (Materialized Path).

Целью исследования является анализ результатов серии стресс-тестов различных алгоритмов на различных объемах данных.

Для проведения исследования выбраны следующие операции, так как они являются наиболее часто используемыми:

- извлечение дерева;
- сохранение дерева;
- удаление узла.

Для выполнения исследования был написан бенчмарк, который проводит необходимые измерения. В ходе исследования проводились замеры времени выполнения запросов, а также средней загрузки CPU.

2. Выборка дерева

На рисунке 1 изображены зависимости времени, с, выборки дерева от количества узлов в дереве и объема полезных данных.

Как видно на графике, при небольшом количестве узлов в дереве быстрее всего работает метод «Вложенное множество». При увеличении количества узлов и объема хранимых данных этот метод оказывается медленнее.

Самым медленным при выборке дерева является метод «Замкнутых таблиц». Это происходит по причине того, что для этого метода служебная информация

хранится в отдельной таблице, и при выборке данных необходимо считывать значения из двух таблиц, а не из одной, как это реализовано в остальных методах.

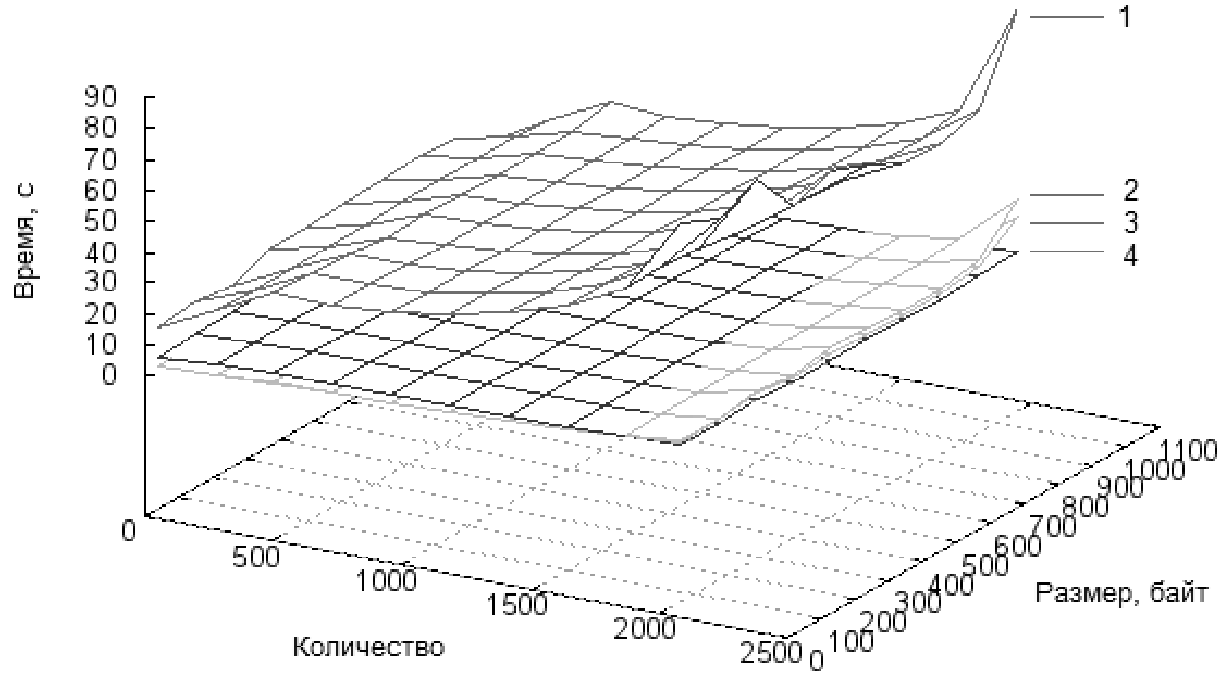

Рис. 1. Выборка дерева:
1 – Closure Tables; 2 – Nested Set; 3 – Materialized Path; 4 – Adjacency List

3. Сохранение дерева

На рис. 2 отражены зависимости времени, с, выполнения запроса при сохранении дерева в базе данных.

При добавлении нового узла в методе «Вложенные множества» пересчитываются ключи, что приводит к перестроению части дерева. Это, в свою очередь, приводит к значительным затратам вычислительных ресурсов.

Сохранение данных в методе «Замкнутые таблицы» выполняется быстрее, чем при использовании метода «Вложенные множества». Это связано с тем, что при добавлении нового узла уже существующие в базе данных узлы дерева не изменяются. Но этот метод работает медленнее остальных вследствие необходимости хранения дополнительных служебных данных.

В методе «Материализованный путь» ключом является строка, а при добавлении нового узла в дерево происходит операция конкатенации строк, что, в свою очередь, негативно сказывается на производительности при записи данных.

Самым быстрым методом при сохранении является «Список смежных вершин». Для добавления нового узла в базу данных необходимо указывать только родительский узел.

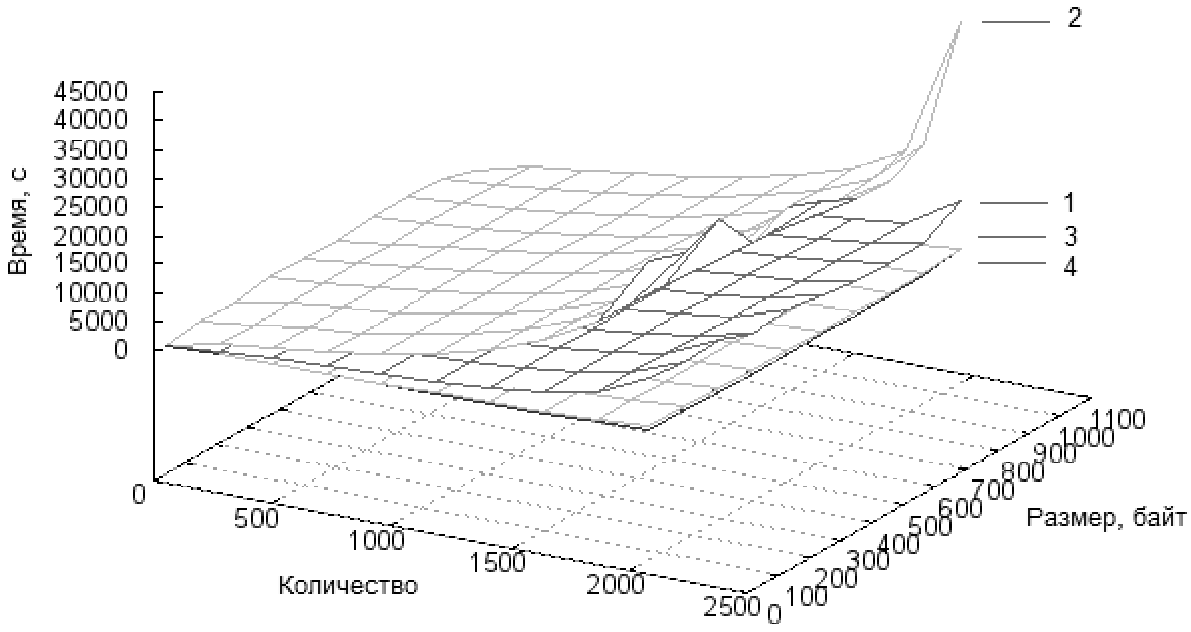

Рис. 2. Сохранение дерева:
1 – Closure Tables; 2 – Nested Set; 3 – Materialized Path; 4 – Adjacency List

4. Удаление узла

На рис. 3 показаны зависимости времени, с, выполнения запроса при удалении узлов дерева.

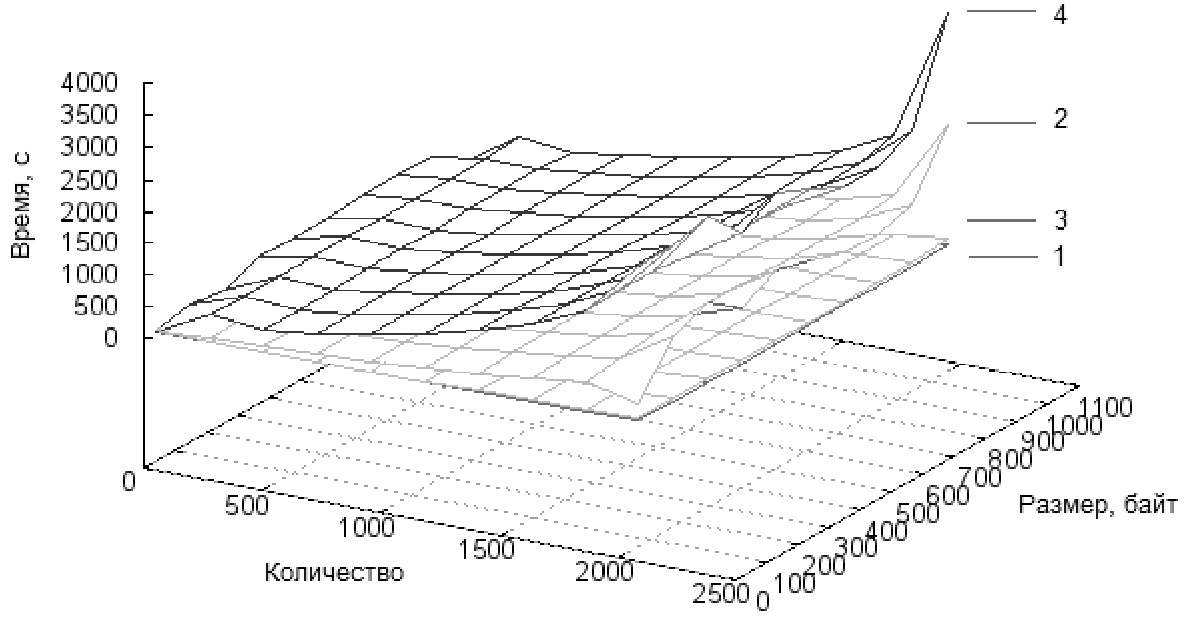

Рис. 3. Удаление дерева:
1 – Closure Tables; 2 – Nested Set; 3 – Materialized Path; 4 – Adjacency List

Быстрее всего удаление узлов в дереве происходит при использовании метода «Замкнутые таблицы». Это достигается благодаря тому, что в базе данных хранятся все пути в дереве.

Вторым по скорости удаления данных является «Материализованный путь». Это связано с затратами на удаление строковых значений, хранящих информацию о полном пути от корня дерева до узла.

Следующий в очереди — метод «Вложенные множества». При удалении узла в дереве пересчитываются ключи правой части дерева, что приводит к изменению данных так же, как и при добавлении нового узла.

Самым медленным является «Список смежных вершин». Результат объясняется тем, что каждый узел хранит информацию о своем родителе, и для нахождения полного пути в дереве необходимо использовать рекурсивные запросы.

5. Результаты тестирования

В таблице 1 представлены результаты замеров времени выполнения запросов на извлечении, сохранении и удалении данных при размере полезных данных 1024 байт.

Таблица 1.

Зависимость времени (с) выполнения запроса от количества узлов в дереве

	Количество узлов	Materialized Path, с	Nested Set, с	Adjacency List, с	Closure Tables, с
Извлечение	7	27.79	78.7	7.24	6.159
	31	30.08	76.6	9.702	6.195
	127	33.56	116.3	12.83	10.27
	511	40.93	241.3	27.18	27.8
	2047	51.7	491	73.95	83
Сохранение	7	136.8	166.4	91.65	138.7
	31	265	358.2	146	276.6
	127	876.3	1914	541.5	1236
	511	3054	17692	2817	9084
	2047	11976	200005	7836	63124
Удаление	7	146.4	83.84	78.29	96.28
	31	122.1	85.32	195.7	106.8
	127	138.8	266.6	5198	133.9
	511	274.8	742.2	1182	242
	2047	720	7070	17352	388

6. Выводы

Результатом исследований является реализация API для работы с различными типами деревьев. Выявлены достоинства и недостатки приведенных подходов с учётом их различных применений, а также произведено сравнение производительности при различных объемах хранимых данных.

На основе полученных данных можно сделать вывод, что нет абсолютно лучшего способа для хранения иерархических структур. В зависимости от поставленной задачи следует выбирать подходящий способ с учетом различных критериев, таких, как время выполнения запроса, объем хранимых данных и загрузка CPU, основным из которых является время выполнения запроса. Так, для задач, в которых наиболее критичным является критерий скорости выборки

данных, наилучшим является «Вложенное множество»; для задач, в которых необходимо вносить большое количество изменений в дерево, - «Список смежных вершин». Метод «замкнутые» таблицы позволяет легко производить выборку отдельных поддеревьев, однако, на вставку нового узла требуется больше времени, чем в остальных методах.

В дальнейшем планируется проведение аналогичных стресс-тестов в других СУБД.

Список литературы

1. Celko, Joe, Trees and Hierarchies in SQL for Smarties, Second Edition / Joe Celko. – Morgan Kaufmann. – 2012. – 296 p.

2. Dong, Guozhu, Maintaining the transitive closure of graphs in SQL / Guozhu Dong, Leonid Libkin, Jianwen Su and Limsoon Wong // Int. Journal of Information Technology. – 1999. – №5. P. 46-78.

3. Tropashko, Vadim, SQL Design Patterns: Expert Guide to SQL Programming / Vadim Tropashko. – Rampant Techpress. – 2007. – 254 p.

4. Karwin, Bill, SQL Antipatterns: Avoiding the Patfails of database programming / Bill Karwin. – Pragmatic bookshelf. – 2010. – 328 p.

Рецензент: д.т.н., проф. Харченко В.С., Национальный аэрокосмический университет им. Н.Е. Жуковсого «ХАИ», г. Харьков.

Аналіз способів зберігання ієрархічних структур у реляційних базах даних з використанням стресс-тестів

Проаналізовано проблеми зберігання ієрархічних структур в реляційних базах даних. Проведено порівнання продуктивності існуючих підходів до зберігання ієрархічних даних. Для аналізу способів представлення даних розроблено бенчмарк, який здійснює вимірювання часових характеристик. Проведено аналіз швидкості роботи різних підходів до зберігання даних на основі отриманих даних. Сформульовано рекомендації щодо вибору способів зберігання ієрархічних структур залежно від поставленого завдання.

Ключові слова: ієрархічні структури, реляційні бази даних, вкладена множена, список суміжних вершин, замкнуті таблиці, матеріалізованний шлях.

Analyzes of the ways of storing hierarchical structures in relational databases using stress-tests

This article analyzes the problem of storing hierarchical data structures in relational databases. Compared the performance of existing approaches for storing hierarchical data. For the analysis of data representation developed benchmark, which performs measurements of time characteristics. Analyzed performance of different approaches to store the data on the basis of the obtained data. Made recommendations on the choice of ways to represent hierarchical structures, depending on the task.

Keywords: hierarchical structures, relational databases, nested set, adjacency list, closure tables, materialized path.