УДК 004.4

ДОСЛІДЖЕНЯ МОДЕЛЕЙ ЗБЕРІГАННЯ ІЄРАРХІЧНОЇ ІНФОРМАЦІЇ У РЕЛЯЦІЙНИХ БАЗАХ ДАНИХ

Соломенцев С.В.
Харківський національний університет радіоелектроніки

Об'єктом дослідження при написанні роботи послужив процес розробки архітектури програм, що передбачають зберігання ієрархічних структур в базі даних, а саме коментарів, організаційних відділів, дерев прийняття рішень та інш. Предметом дослідження роботи стали системи управління базами даних і швидкість їх роботи з тими чи іншими ієрархічними структурами. У роботі розкривається актуальність дослідження за обраним напрямом, ставиться проблема, мета дослідження, а також порівнюється ефективність використання таких ієрархічних структур, як: «Nested set», «Materialized path», «Adjacency list». У висновку описується значимість отриманих результатів.
Ключові слова: таблиця, дерево, ієрархія, структура, ключ.

Постановка проблеми. Сьогодні спостерігається великий інтерес до технологій класу BIG DATA. Постійне зростання об'ємів даних, якими доводиться оперувати великим компаніям вимагає все нових рішень в сфері програмного забезпечення. Накопичена інформація для багатьох організацій є важливим активом, однак обробляти її і витягати з неї користь з кожним днем стає все складніше і дорожче.

Одним з важливих аспектів накопичення та роботи з інформацією є початковий вибір архітектури. Правильно вибрана і складена архітектура програми дозволить досягти кращих результатів у її підтримці, що в свою чергу дозволить значно скоротити витрати компанії.

При правильному підході до складання архітектури програми необхідно врахувати безліч деталей: вибрати відповідну операційну систему, технології, системи управління базами даних, підібрати алгоритми і шаблони проектування. І кожна з цих деталей має ряд особливостей, характерних властивостей та обмежень.

Традиційно виділяються три головні напрями систем управління базами даних за різними моделями даних, на яких ці напрямки засновані – мережеві, ієрархічні, реляційні. Реляційна модель даних, нині найпопулярніша і просунута з усіх існуючих моделей, була спочатку розроблена в кінці 60-х років минулого століття британським вченим, співробітником компанії IBM, Едгаром Коддом. Дана модель заснована на складанні таблиць даних і відносин між ними, що дозволяє досить вільно представляти бізнес модель у вигляді інформаційних таблиць. Кожна таблиця являє собою сутність, її поля або стовпці – атрибути сутності, а записи в таблиці – інформацію. Зв'язки між сутностями представляються у вигляді додаткових полів у таблиці, які дозволяють позначити належність однієї сутності до іншої. Завдяки можливості побудови зв'язків між таблицями та подання відносини один до багатьох компанії мають можливість представляти і зберігати найпростіші бізнес-моделі у вигляді інформації.

З ростом бізнес-моделі ростуть і потреби в побудові більш складних структур даних. Однією з досить цікавих і нетривіальних структур даних для побудови, є ієрархічна структура даних. У першу чергу, це пов'язано з тим, що реляційні бази не пристосовані до зберігання ієрархічних структур (як, наприклад, XML-файли), структура реляційних таблиць представляє собою прості списки. Ієрархічні ж дані мають зв'язок «батько-спадкоємці», яка не реалізована в реляційній структурі.

За допомогою ієрархій можна спростити пошук інформації, управляти сортованими списками даних, робити синтаксичний аналіз виразів, оптимізувати програми. Ієрархії є дуже продуктивними в компонуванні цифрових зображень для отримання різних візуальних ефектів, а також в представленні форм прийняття багатоетапного рішення.

З розвитком моделей зберігання інформації, історично було виділено три моделі зберігання ієрархічної інформації: список суміжних вершин (Adjacency List), вкладені множини (Nested Set), матеріалізований шлях (Materialized Path). Кожна з моделей зберігання ієрархічної інформації має свої переваги, недоліки і спосіб побудови в реляційній базі даних.

Серед популярних моделей зберігання ієрархічної інформації можна виділити кілька критеріїв, що виділяють ці моделі і дозволяють вирішувати завдання різного характеру. Серед таких критеріїв: операції вибірки гілок дерева, піддерев цілком, операції перенесення, видалення і вставки гілок, а також окремих вершин. Іншими словами можна сказати, що дані критерії впливають на швидкодію і складність роботи зі структурами.

Правильно обрана модель зберігання ієрархічної інформації в реляційній базі даних, дозволить якісно скласти один з аспектів архітектури додатку, а значить і поліпшити надійність і продуктивність.

Аналіз останніх досліджень і публікацій. Серед багатьох досліджень в цій області, одним з найбільш відомих та повних примірників, можна виділити книгу автора Joe Celko, «Trees and hierarchies in SQL for smarties» [1] яка дуже детально розглядає три основні структури зберігання даних: «Nested Sets», «Materialized path», «Adjacency list». Вона наглядно демонструє як поводить себе дерево при виконанні скрипта призначеного для його модифікації. Ця книга обговорює різні способи, якими б дизайнери баз даних могли вирішити поставлені проблеми зі зберігання ієрархій. Є багато способів для представлення ієрархічних даних в SQL, а кожна глава цієї книги присвячена деталям одного конкретного рішення. Необхідно виділити також цікаві

питання, розглянуті в цій книзі, які виникають при виконанні скрипта з модифікації дерева, а саме: як повинні вести себе піддерева вершини, що видаляється, чи мають вони стати піддеревами родича цієї вершини, чи може ці піддерева повинні прийняти вид одного дерева та стати на місце вершини, що видаляється.

В статті про управління ієрархічними даними в системі управління базами даних MySQL [2], можна знайти ряд скриптів адаптованих під конкретно цю систему. В статті наглядно продемонстровано використання скриптів на тестовій базі даних, а також розглянуті питання: чи потрібно використовувати JOIN запити до бази, якщо так, то наскільки і коли це актуально, як ефективно використовувати GROUP BY та ORDER BY запити маючи ієрархічну структуру даних.

Вадим Тропашко – один з працівників компанії Oracle в своїому блозі дає теоретичне обгрунтування найбільш популярних моделей зберігання ієрархічних даних [3], та розглядає наступні питання: наскільки комплексною є реалізація кожної з моделей, наскільки простою є операція вибору всіх нащадків (піддерев), наскільки простою та ефективною є вибірка усіх предків для заданої вершини, а також наскільки важкою є реорганізація дерев.

Виділення не вирішених раніше частин загальної проблеми. Кожна стаття по-своєму розглядає основні моделі зберігання ієрархічної інформації. В одних примірниках це теоретичне обгрунтування скриптів та можливих рішень з їх вдосконалення, в інших це розгляд практичного їх використання у конкретній системі управління базами даних.

На даний момент існує багато думок з приводу того, які моделі треба використовувати для швидкого читання структури дерева з бази даних, а які для швидкої модифікації того ж дерева. Та на жаль ще не знайдено універсальної моделі яка могла б робити усі ці дії з максимально можливою швидкістю, таким чином щоб перевершити інші існуючі моделі, тож розробникам та архітекторам програмних рішень доводиться робити вибір з того, яка модель максимально відповідає конкретно поставленій задачі.

Якщо взяти всі відомі моделі зберігання даних, то можна помітити, що всі вони дуже непогано почувають себе на невеликих об'ємах даних. Проблема вибору конкретної структури стає в момент коли даних стає дуже багато. Під словом «багато», можна розуміти різні об'єми даних, від десятків записів до мільярдів.

Та чи має стояти вибір конкретної моделі зберігання ієрархічних структур у реляційній базі даних перед власником десяти записів у деревовидному форматі? Звісно ж він буде більш зацікавлений в тому щоб максимально спростити роботу над цими даними і вже в останню чергу буде думати про швидкість обробки дерева. То де знаходиться та сама межа, яка змусить задуматись власника записів про раціональну організацію дерева в базі даних? Саме це питання має виникнути при побудові архітектури програми, для того щоб в майбутньому вона відповідала потребам користувача.

Мета статті. Головною метою цієї роботи є виявлення швидкості обробки записів, зберігаючих деревоподібні структури, за допомогою відомих нині моделей: «Adjacency List», «Nested Sets», «Materialized Path». Та виявити при якому об'ємі даних потрібно надати належну увагу до вибору моделі зберігання ієрархічних даних.

Виклад основного матеріалу. Отже згідно з основною метою треба було провести дослідження на предмет роботи трьох моделей зберігання даних. Для перевірки швидкості роботи було взято п'ять конфігурацій дерев: дерево, що складається зі ста вершин та має глибину у п'ять порядків; дерево, що складається з тисячі вершин та має глибину у десять порядків; дерево, що складається з десяти тисяч вершин та має глибину у двадцять порядків; дерево, що складається зі ста тисяч вершин та має глибину у двадцять п'ять порядків; та дерево, що складається з п'ятдесяти тисяч вершин та має глибину у тридцять порядків.

Для тестування роботи моделей, було виділено три віртуальні машини на облаці що надаються платформою Google Cloud. Кожна машина має наступні конфігурації: три гігабайти оперативної пам'яті, п'ятдесят гігабайт жорсткого диску, процесор Intel Sandy Bridge, операційна система Windows Server 2012. Для виконання скриптів було обрано три системи управління базами даних: MS SQL Server, Oracle DB, MySQL.

Для інтеграції з базами даних та автоматизації роботи було обрано програмну мову Python. Вона надає дуже широкий вибір бібліотек для роботи і є багатоплатформовою, тож у майбутньому можна буде перевірити роботу ще й на інших операційних системах.

В кожній системі управління базами даних була створена власна база даних для роботи з деревами, кожна з яких мала стандартні конфігурації, що надаються системою управління при її створенні.

В кожну базу даних було додано по три моделі баз даних: «Adjacency List», «Nested Sets», «Materialized Path», структури яких ми розглянемо трохи пізніше. Кожна з моделей представляє собою таблицю з відповідними полями.

Для дослідження було вибрано наступні операції над деревом:

```
-- Adjacency List Tree Structure
CREATE TABLE `al_tree` (
 `id` bigint(20) NOT NULL auto_increment,
 `parent_id` bigint(20) default NULL,
 `name` varchar(50) NOT NULL,
 PRIMARY KEY (`id`),
 KEY `fk_tree_tree` (`parent_id`),
 CONSTRAINT `fk tree tree` FOREIGN KEY (`parent id`)
 REFERENCES `al tree` (`id`) ON DELETE CASCADE ON UPDATE
CASCADE
)
```

```
-- Nested Set Tree Structure
CREATE TABLE `ns_tree` (
 `id` bigint(20) NOT NULL auto_increment,
 `name` varchar(50) NOT NULL,
 `lft` bigint(20) NOT NULL,
 `rgt` bigint(20) NOT NULL,
 `level` bigint(20) NOT NULL,
 PRIMARY KEY (`id`),
 KEY `nslrl_idx` (`lft`,`rgt`,`level`)
)
```

```
-- Materialized Path Tree Structure
CREATE TABLE `mp_tree` (
 `id` bigint(20) NOT NULL auto_increment,
 `name` varchar(50) NOT NULL,
 `path` varchar(100) NOT NULL,
 `level` int(11) NOT NULL,
 PRIMARY KEY (`id`),
 KEY `mpp_idx` (`path`)
)
```

Рис. 1. Скрипти для створення таблиць моделей баз даних

Джерело: розроблено автором за даними [2]

1) Вибір шляху до конкретного вузла;
2) Вибір піддерева;
3) Вставка нового вузла;
4) Переміщення піддерева;
5) Видалення піддерева.

Всі скрипти що виконувались до баз даних було автоматизовано, та виконано за допомогою програмної мови Python. На рисунку 1 зображено скрипти що використовувались для додавання таблиць у базу даних MySQL.

Висновки і пропозиції. Провівши дослідження роботи на трьох системах управління базами даних, ми отримали результати що показують скільки часу було витрачено на роботу кожного скрипта.

З результатів було сформовано таблицю 1, яка відображає середні дані швидкості з усіх протестованих систем управління базами даних. Дані впорядковані по операціям, що приведені вище, а також по моделям зберігання ієрархічних даних.

З таблиці можна побачити що усі моделі зберігання ієрархічних структур дуже впевнено почуваються з деревами до десяти тисяч вузлів. Коли ж до справи стають великі об'єми даних, ми бачимо дуже різку зміну у швидкості, для вибірки даних в моделі Adjacency List, для переміщення вузла в Materialize Path та Nested Sets, та для видалення та додання вузлів в моделі Nested Sets.

Зі скриптів можна зробити висновок що всі вони мають стандартний вигляд, що можна знайти в джерелах. Але кожна з узятих систем управління базами даних має свої розширення для зберігання ієрархічних систем. Наприклад в Microsoft SQL Server є поля типів: xml, hierarchyid, що представляють собою розширену реалізацію моделі Materialized Path. Тому одним з планів на майбутнє є також перевірення цих розширень на швидкість та розгляд їх у порівнянні зі стандартними реалізаціями.

Таблиця 1

Перелік часу виконання скриптів

Кількість вузлів	Шлях до вузла, сек	Вибір піддерева, сек	Вставка вузла, сек	Переміщення вузла, сек	Видалення вузла, сек
Adjacency List					
100	0,0002	0,004	0,0006	0,0004	0,0001
1000	0,0004	0,03	0,004	0,0004	0,0001
10000	0,0006	0,3	0,04	0,001	0,0001
100000	0,001	2,6	0,06	0,001	0,0003
500000	0,001	45	0,06	0,001	0,0004
Nested Set					
100	0,0003	0,0002	0,004	0,02	0,003
1000	0,0004	0,001	0,03	0,2	0,03
10000	0,004	0,008	0,4	1,4	0,4
100000	0,08	0,2	100	214	59
500000	0,4	0,6	1400	3613	1624
Materialized Path					
100	0,0001	0,0001	0,0008	0,03	0,0006
1000	0,0006	0,001	0,001	0,2	0,001
10000	0,006	0,01	0,007	1	0,01
100000	0,1	0,2	1	42	1
500000	0,6	0,6	2	2800	2

Джерело: розроблено автором за даними [1]

Також планами на майбутнє є перевірення впливу таких факторів як: використання жорсткого диску типу SSD, використання операційної системи Linux, використання більшої кількості ядер процесору, оптимізації на рівні системи управління базами даних.

Список літератури:

1. Celko Joe. Joe Celko's Trees and Hierarchies in SQL for Smarties [Text] / Joe Celko. – Morgan Kaufmann, 2004. – 240 pages.
2. Mike Hillyer. Managing Hierarchical Data in MySQL [Electronic Resource]. – Mode of access: URL: http://mikehillyer.com/articles/managing-hierarchical-data-in-mysql/ Title from the screen.
3. Vadim Tropashko. One more Nested Intervals vs. Adjacency List comparison [Electronic Resource]. – Mode of access: URL: https://vadimtropashko.wordpress.com/2008/08/09/one-more-nested-intervals-vs-adjacency-list-comparison/ Title from the screen.

Соломенцев С.В.
Харьковский национальный университет радиоэлектроники

ИССЛЕДОВАНИЕ МОДЕЛЕЙ ХРАНЕНИЯ ИЕРАРХИЧЕСКОЙ ИНФОРМАЦИИ В РЕЛЯЦИОННЫХ БАЗАХ ДАННЫХ

Аннотация
Объектом исследования при написании работы послужил процесс разработки архитектуры приложений, предусматривающих хранения иерархических структур в базе данных, а именно, комментариев, организационных отделов, деревьев принятия решений и др. Предметом исследования стали популярные системы управления базами данных и скорость их работы с теми или иными иерархическими структурами. В работе раскрывается актуальность исследования по выбранному направлению, ставится проблема и цель исследования, а также сравнивается эффективность использования таких иерархических структур, как: «Nested set», «Materialized path», «Adjacency list». В выводе описывается значимость полученных результатов.
Ключевые слова: таблица, дерево, иерархия, структура, ключ.

Solomentsev S.V.
Kharkiv National University of Radioelectronics

RESEARCH MODELS OF HIERARCHICAL STORAGE OF INFORMATION IN RELATIONAL DATABASES

Summary
The object of research is the process of developing application architecture and providing storage for such hierarchies in the database as comments, organizational departments, decision trees. The subject of the research work is popular database management systems and how fast they works with variety of hierarchical structures. The work reveals the relevance of studies in the chosen direction, raises the problem and the purpose of the study, and it also outlines effectiveness of using such relational hierarchies as: «Nested set», «Materialized path», «Adjacency list». The conclusion describes the significance of the results.
Keywords: table, tree, hierarchy, structure, key.

УДК 664:144:641.1

CONFECTIONS WITH A LONG-TERM STORAGE

Shapovalova N.P.
National University of Food Technologies

The influence of additives and packaging methods for extending the shelf life guaranteed in stamped confectionery and changes their quality during storage. In the process of studies used generally recognized and modern methods. On the basis of these studies was demonstrated that the addition of churned confectionery algae increases the number and diameter of the pores, causing more adsorption properties. Porous samples should be longer soft, that are desirable consumer properties for finished products. Argued the possibility of extending the long-term of storage of the new products from 30 to 90 days by simultaneous use in Lamidan and Cycorlact.
Keywords: active water, adsorption-desorption of water, hydrocolloids, churned confectionery, Lamidan.

Introduction. The term of storage of confectionery is determined by two complex parameters of quality. The first complex indicator of quality can be differentiated on the organoleptic indicators (appearance, taste, smell, texture) products, physico-chemical (moisture content, forms of moisture bond) and indicators that characterize the biological value (content in the product of proteins, fats, carbohydrates, vitamins, minerals). These indicators should remain fairly constant throughout the guaranteed term storage.

The second group includes indicators of safety, especially – microbiological. In the case where at least one parameter of the second group reaches threshold, confectionery are unfit for consumption. To increase the term of storage is necessary to stabilize the indicators of the first group and to slow the changes of the second one.

In evaluating the quality and the term of fitting the food one of the key physical and chemical parameters is content of moisture, which primarily affects the growth of microorganisms. With decreasing the content of moisture decreases the intensity of proliferation of microorganisms and in reaching a certain moisture content stops. However, for microbial development are important not absolute humidity, and the availability of water for microbial development. This indicator is called water activity [1].

Water activity is determined by the value of the equilibrium relative humidity, and serves to quantify the energy of the moisture with the material:

$$a_w = (P.B.B.) / 100 = Pn / Ps \quad (1)$$

where a_w – water activity in decimal fractions;

P.B.B. – equilibrium relative humidity at which the product absorbs moisture and doesn't give it to the environment.

The indicator aw for pure water without additives equal to 1, the stronger bond water in food, the lower the figure to the value [2].

«Water activity» can vary from 0 to 1. The different types of confectionery with water activity vary from 0.4 to 0.95. The control of water activity indicator allows predicting the processes that occur while storage of confectionery. For values of indicators of water activity all confectionary are divided into three groups:

I group – products with the indicators of water activity of 0.65 or less – low humidity 10-13% less. These include sweets with praline, marzipan shells and chocolate.

II group – intermediate moisture products (a_w from 0.65 to 0.95). Moisture products are between 13 and 35%. These include sweets from the churned, jelly, fruit and jelly shells.

III group – product with high humidity – over 35% and 0.9. They are a group of biscuits and sponge cakes.

In products with low humidity suppressed activity of microorganisms. They held the oxidation