УДК 519.852.67

К.А.Суйковська, студ., V курс, гр. ПІ-41м
Науковий керівник – Ковальчук А.М., к.т.н., доцент
Житомирський державний технологічний університет

ВИКОРИСТАННЯ МОДЕЛІ NESTED SET ДЛЯ ЗБЕРІГАННЯ ДЕРЕВОВИДНИХ СТРУКТУР В БАЗІ ДАНИХ SQL

Перед автором була поставлена задача – зберігання та використання деревовидних структур (категорій оголошень) з застосуванням технології MySQL при розробці веб-сайту оголошень besplatka.ua.

Стандартна реляційна алгебра та побудовані на ній SQL операції не можуть бути застосовані для всіх потрібних маніпуляцій з деревами (ієрархіями). Якщо дерево має довільну глибину, це не дозволяє використовувати SQL вирази для таких операцій, як порівняння місця в ієрархії для двох елементів або визначення належності елемента до певного під-дерева.

Існує декілька підходів для вирішення проблеми і деякі є доступними в системах керування базами даних:

- підтримка ієрархічних типів даних;
- розширення SQL для маніпуляцій з деревами;
- SQL запити можуть бути виражені мовою програмування, яка підтримує ітерації та дозволяє виконувати реляційні операції, як от PL/SQL, T-SQL, або майже будь-якою сучасною мовою програмування;

Деревовидні структури в реляційних базах даних можна зберігати трьома основними способами: Adjacency List (суміжні списки), Matherialized Path (матеріалізовані шляхи) та Nested Set (вкладені множини). Перший спосіб передбачає зберігання батьківського елементу в полі parent_id. Другий – повний шлях до елементу зберігається в полі типу path. Модель Nested Set передбачає зберігання дерева с трьох колонках – left, right та depth, де перші дві зберігають в собі діапазон усіх вкладених елементів.

Перший спосіб – зберігання для кожного елементу ідентифікатору його батька – спосіб, мабуть, найпопулярніший, проте, має значний недолік – SQL запитом можна «пройти» по дереву тільки рівнем вище (маючи на руках parent_id), та рівнем нижче (побудувавши зв'язок на основі того ж parent_id). Кожен наступний рівень передбачає додавання ще одного зв’язку в SQL запиті, що значно нагромаджує сам запит, не кажучи вже про те, що час виконання такого запиту зростає в геометричній прогресії. В такому разі дерево можливо побудувати за допомогою рекурсії. У випадку MySQL ми маємо рекурсію, але тільки на рівні збережених процедур і до 255 рівнів. Також можна застосувати рекурсію, зв’язавши мову програмування та БД, але кількість запитів може бути величезною, тому краще це робити на рівні бази даних.

Другий спосіб – зберігання для кожного елементу його повного шляху в дереві – незручний тим, що переміщення елементів по дереву веде за собою непередбачувану кількість SQL запитів на зміну власне поля path, так як заздалегідь невідомо кількість можливих нащадків елементу, який рухають усередині дерева. Те саме стосується і операції видалення елементу з дерева.

Та, власне, вкладені множини (Nested Set), техніка моделі яких полягає в нумерації вузлів відповідно до обходу дерева. Кожен вузол оброблюється двічі, кожному вузлу надається номер, відповідний до порядкового номеру згідно з обходом. Кожен вузол набуває двох номерів, які зберігаються як два атрибути. Вибірка стає швидкою: належність до дерева (або певної гілки дерева) може бути визначена через порівняння цих номерів. Оновлення дерева вимагає повторної нумерації атрибутів, тому може бути повільною. Проте, в рамках поставленої задачі дерево часто використовується (багато запитів) та рідко змінюється, така модель є найбільш оптимальною.

Дерево складає собою масив вузлів, де кожен вузол має 4 складові – унікальний ідентифікатор, рівень вузла, лівий та правий ключ. Саме у цих двох цифрах закладена вся інформація про дерево. Та, якщо цю інформацію занести до бази даних, то робота з деревом значно спроститься. Щоб проставити ліві та праві ключі, краще всього накласти на дерево асоціацію з лабіринтом без циклів, який потрібно пройти від входу до виходу. Ідея полягає в тому, що потрібно іти лабіринтом, постійно та невідривно торкаючись стіни правою рукою. Таким чином здійснюється обхід від початку до кінця по всім вузлам лабіринту/дерева, причому захід в кожний елемент (комірка для правого та лівого ключів) здійснюється лише один раз. Головною особливістю такого підходу є те, що значення лівого (та правого, але для вибірки достатньо лівого) ключа будь-якого вузла (окрім кореня) знаходиться в діапазоні лівого та правого ключів усіх своїх предків, звідки власне пішла назва цієї моделі – «Вкладені множини».

Визначимо які дані можна вибирати з таблиці шляхом запиту:

1. Власне, саме дерево
   SELECT id, name, level FROM my_tree ORDER BY left_key

   В результаті отримано список, де кожен нащадок знаходиться після свого батька. Слід зауважити, що лівий ключ визначає не лише положення елементу в дереві, а й порядок

відносно сусідів, тому, видозмінивши отриманий список, додаючи до назви відступи у кількості рівня елементу, отримаємо дерево в класичному представленні/вигляді.

2. Вибір підпорядкованої гілки (усіх нащадків)
   SELECT id, name, level FROM my_tree WHERE left_key >= $left_key AND right_key <= $right_key ORDER BY left_key
3. Вибір батьківської гілки (тобто усіх предків)
   SELECT id, name, level FROM my_tree WHERE left_key <= $left_key AND right_key >= $right_key ORDER BY left_key
4. Вибір гілки, у якій приймає участь поточний вузол
   SELECT id, name, level FROM my_tree WHERE right_key > $left_key AND left_key < $right_key ORDER BY left_key

В цілому, застосовуючи в умові запиту ключів вузла, можна вибрати будь-які дані, пов'язані з цим вузлом.

Отже, можна виділити основні правила та прикмети даної моделі:

1. Лівий ключ завжди менше правого;
2. Найменший лівий ключ дорівнює одиниці;
3. Найбільший правий ключ дорівнює подвоєній кількості вузлів;
4. Різниця між правим та лівим ключем завжди число непарне;
5. Якщо рівень вузла – число непарне, то лівий ключ завжди непарне число, аналогічно з парними числами.
6. Ключі завжди унікальні, незалежно від того правий він чи лівий.
7. Кількість підпорядкований вузлів (нащадків) для кожного елементу можна визначити за формулою

   $$count = (right - left - 1)/2$$

   Отже, щоб визначити, є у елементу нащадки чи ні, не потрібно робити додаткові запити в базу даних – якщо різниця між правим та лівим ключами складає одиницю, елемент є кінцевим у гілці.

Розглянемо операції зміни дерева – створення вузла, переміщення по дереву та видалення з дерева.

Для створення вузла потрібно знати рівень та правий ключ батьківського вузла, або максимальний правий ключ, якщо у нового вузла не буде батьківського. Створення нового вузла пройде в 3 кроки – оновлення ключів існуючого дерева, вузлів, що стоять за батьківським вузлом. (UPDATE my_tree SET left_key = left_key + 2, right_ key = right_ key + 2 WHERE left_key > $right_ key); наступний – оновити батьківську гілку (UPDATE my_tree SET right_key = right_key + 2 WHERE right_key >= $right_key AND left_key < $right_key); та власне створення нового вузла (INSERT INTO my_tree SET left_key = $right_key, right_key = $right_key + 1, level = $level + 1).

Операція видалення вузла не набагато складніша, проте потрібно врахувати, що у вузла, що видаляється, можуть бути нащадки. Тому, видалення з дерева також проходить поетапно – спочатку власне видаляються ті вузли, у яких лівий ключ не менший за лівий ключ потрібного елементу і правий ключ не більший за правий ключ цього елементу. Наступним кроком буде оновлення батьківського елементу – зменшення його правого ключа на величину, що дорівнює різниці правого та лівого ключів елементу на видалення, до якої додається одиниця. Останнє – оновлення наступних вузлів (тобто тих, що стоять правіше) – від їх лівих та правих ключів також віднімається значення того ж виразу.

Переміщення вузла – найскладніша дія в управлінні деревом. Вузол може переміщуватися в дві різні області: вищестоящий і нижчестоящих вузлів.

1. Вверх по дереву включає в себе:
   a. Перенесення гілки (вузла) в підпорядкування нижчестоящому по дереву вузлу;
   b. Перенесення гілки вверх без зміни батьківського вузла (зміна порядку вузлів);
2. Вниз по дереву:
   a. Перенесення вузла в корінь дерева;
   b. Перенесення вузла вниз без зміни батьківського вузла (зміна порядку вузлів);
   c. Підняття вузла на рівень вище;
   d. Переміщення вузла вниз по дереву.

Висновок. У даній роботі була розглянута і описана задача зберігання деревовидних структур в реляційних базах даних. Було запропоновано використання моделі «Вкладені множини» та описано переваги методу над іншими.

ВІДОМОСТІ ПРО АВТОРІВ:

СУЙКОВСЬКА Катерина Андріївна, магістрант групи ПІ-41м кафедри програмного забезпечення систем Житомирського державного технологічного університету.