УДК 004.658.2

К.А. НОВОХАТСЬКА
Одеський національний політехнічний університет

УДОСКОНАЛЕННЯ ТЕХНОЛОГІЇ АВТОМАТИЗОВАНОГО ОНОВЛЕННЯ МАТЕРІАЛІЗОВАНИХ ПРЕДСТАВЛЕНЬ ШЛЯХОМ ПОШИРЕННЯ ЇЇ НА ІЄРАХІЧНІ ТА АНАЛІТИЧНІ ЗАПИТИ

Вдосконалюється технологія автоматизованого оновлення матеріалізованих представлень. Запропоновано логічні функції оновлення для матеріалізації ієрархічних запитів та запитів з аналітичними функціями. Розроблено алгоритм перетворення логічних функцій оновлення на тригери SQL. Проведено аналіз журналу виконання трансакцій СКБД для оцінки якості удосконаленої технології генерації функцій оновлення у порівнянні з базовою технологією.

Ключові слова: матеріалізоване представлення, логічна функція оновлення, ієрархічний запит, аналітична функція.

K. NOVOKHATSKA
Odessa National Polytechnic University

TECHNOLOGY IMPROVEMENT OF THE AUTOMATED MAINTENANCE OF MATERIALIZED VIEWS BY EXTENDING IT ON HIERARCHICAL AND ANALYTICAL QUERIES

One of the effective methods for solving the problem of increasing the performance in relational databases is the use of materialized views (MV) – stored results of query execution. Applying MV for resource-intensive and often-repeated queries can significantly shorten the execution time and thereby improve overall system performance. When MV are formed, the task is to maintain the relevance of data stored in the MV. Modern technologies of automated MV maintenance only partially cover the syntax of the SQL language. In particular, such SQL constructs as hierarchical and analytical queries are not supported. This leads to the fact that in many cases the creation of MV update functions is performed manually. The technology of automated materialized views maintenance is being improved. The logical update functions are proposed for the materialization of hierarchical queries and queries with analytic functions. The algorithm for converting the logical update functions to the SQL triggers is developed. The analysis of the DBMS transaction log has been performed to evaluate the quality of the improved technology of update functions generation in comparison with the base technology. Using advanced technology, it was possible to create an average of 7% more update functions than the base technology. The developed technology extends the capabilities of the base method for the materialization of hierarchical queries and queries with analytic functions, but the error in the syntax of the created update functions for these query groups is still quite large (the proportion of logically correct update functions is 62% and 55% respectively).

Keywords: materialized view, logical update function, hierarchical query, analytic function.

Вступ

Одним з ефективних методів вирішення проблеми підвищення продуктивності реляційних баз даних (РБД) є використання матеріалізованих представлень (МП) – збережених результатів виконання запитів. Застосування МП для ресурсномістких і часто повторюваних запитів може значно скоротити час виконання і тим самим підвищити загальну продуктивність системи. Коли МП сформовані, виникає задача підтримки актуальності даних, що зберігаються в МП. Сучасні технології автоматизованого оновлення МП лише частково охоплюють синтаксис мови SQL. Зокрема, не підтримуються такі конструкції мови SQL, як ієрархічні і аналітичні запити. Це призводить до того, що в багатьох випадках створення функцій оновлення МП виконується вручну.

При ручному підході немає можливості проаналізувати всі запити, що надходять в систему, врахувати статистичні і часові показники виконання, і вибрати МП, що дозволяють максимально ефективно оптимізувати роботу конкретної інформаційної системи (ІС). Таким чином, створені вручну МП вирішують лише точкові проблеми продуктивності систем керування базами даних (СКБД). Отже підтримка усіх конструкцій мови SQL у технології автоматизації оновлення МП є актуальною проблемою.

Аналіз літературних даних та постановка проблеми

В [1] розглядається методика генерації функцій поновлення МП для різних видів запитів, які на підставі лексичного аналізу пропонується класифікувати по шести групах. Для даних видів запитів проаналізовано ефективність використання таблиць оновлення і виведені числові коефіцієнти, що дозволяють оцінити ефективність застосування таблиць оновлення на практиці.

Перевагою запропонованої методики є універсальність форми логічного запису, яка зводить будь-який запит до набору простих операцій над атомарними значеннями. Такий підхід дозволяє автоматизувати генерацію тригерів для оновлення МП.

Запропонована класифікація запитів дозволяє поширити розглянутий метод на більшість практичних застосувань. Проте вона не покриває такі типи запитів, як запити з використанням ієрархій або аналітичних функцій.

В [2] розробляється технологія інкрементального обслуговування запитів з агрегаціями і об'єднаннями, також заснована на використанні таблиць змін (*change-table technique*). Обчислення нового значення МП виконується за допомогою застосування оператора оновлення до даних, занесених в таблицю.

Автори розробляють функції оновлення запитів, що містять агрегації і об'єднання, і розраховують

вартість оновлення з урахуванням запропонованого методу. Недоліком цього методу є його реалізація на мові логічноЇ вибірки (*deductive query language*), що не дозволяє застосовувати його на практиці.

В [3] розглядається алгоритм для обчислення SPJ-запитів, що не підтримує реалізацію агрегацій і т.д.

В [4] розглядаються інкрементальні оновлення с використанням моноїдів, обчислюваних на основі множин модифікованих рядків базових таблиць (БТ). Оскільки потужність множини ΔC в порівнянні з потужністю множини C досить мала, то витрати на обчислення функції f для ΔC будуть значно менше, ніж перерахунок всього МП, що доводить ефективність даної технології.

Однак для моноїдів існують деякі обмеження. У більшості випадків вони можуть бути застосовані тільки до лінійних операцій, що ускладнює обслуговування МП, які містять такі функції, як *UNIQUE*, *DISTINCT* тощо. Ще одним недоліком використання моноїдів в сучасних СКБД є їх придатність тільки до атомарних типів даних [5].

В [6, 7] розробляється механізм генерації правил обслуговування МП (*view-maintaining rules*). Правила створюються для операцій вставки, видалення і модифікації рядків з основних таблиць для кожного посилання на таблицю в тексті запиту МП і спрацьовують за принципом тригера для кожної користувальницької транзакції, яка може включати в себе одну або більше операцій модифікації даних.

Для зберігання службових даних про зміни БТ автори використовують набір таблиць (*transition tables*) – по одній таблиці для операцій *INSERT* і *DELETE* і по дві таблиці для *UPDATE* (для старого і нового значення полів). Усередині правила може бути виконана операція *SELECT* для того, щоб при необхідності отримати оновлене значення рядків БТ.

Для генерації правил автори розробили граматику, що описує деякі основні конструкції мови SQL. Граматика дозволяє створювати правила для підзапитів з використанням ключових слів *ANY*, *IN* і *EXISTS*, а також операцій заперечення (*NOT*) і порівняння (>, < і т.д.). Однак нею не підтримуються агрегації і інші поширені функції, що обмежує використання запропонованої технології. Крім того, автори вводять обмеження на обслуговування представлень, що містять в собі дублікати. В такому випадку, результат спрацьовування правил може бути непередбачуваним.

Крім описаних джерел інкрементальні поновлення різних видів запитів також розглядаються в роботах [8–10], однак, запропоновані рішення перетинаються з вже розглянутими технологіями.

Мета та задачі дослідження

Метою даної роботи є підвищення якості технології автоматизованого поновлення МП шляхом поширення даної технології на ієрархічні і аналітичні типи запитів.

Для досягнення поставленої мети необхідно вирішити наступні задачі:

- запропонувати логічні функції оновлення для ієрархічних і аналітичних запитів;
- розробити алгоритм перетворення логічних функцій оновлення на тригери мови SQL.

Матеріали та методи дослідження автоматизованого оновлення МП

Розглянемо деяке МП M, що містить поля m_i, $i=\overline{1,n}$. Представимо запит, що формує це МП, як функцію трьох змінних:

$$f(\pi,T,P) \text{ чи } f(f_{вн}(\pi),T,P), \tag{1}$$

де π – набір запрошуваних полів;

T – множина таблиць, що беруть участь у запиті;

P – множина умовних предикатів фрази *WHERE*;

$f_{вн}$ – внутрішня функція (*COUNT*, *DISTINCT* тощо).

Задамо для даного МП таблицю оновлення (ТО) ΔM як різницю множини рядків БТ до і після модифікації. ТО буде містити множину полів МП m_i, $i=\overline{1,n}$, тип функції оновлення (*Insert, Update, Delete*) і множину полів з P. Для кожної операції додавання, видалення і оновлення даних в БТ, замість безпосередньої модифікації МП, необхідно виконати відповідний запис в ΔM с зазначенням старого (*OLD*) і оновленого (*NEW*) значення полів.

Для складання логічної форми запису функції оновлення введемо наступні позначення:

- $\in_p$ – входження елемента предиката в умову P фрази *WHERE*;
- Π – об'єднання стовпців в єдиний рядок, якщо умова істинна;
- $\cup$ – об'єднання рядків;
- $:=$ – присвоювання;
- $/$ – видалення елемента з множини;
- $\&\&$ – кон'юнкція операцій;
- $||$ – диз'юнкція операцій;
- $=$ – умова рівності двох елементів множини;
- $\neq$ – умова нерівності двох елементів множини.

Для оновлення даних МП в [1] було введено оператор $\overset{\Delta}{\leftarrow}{}^{\alpha}_{\beta}$, де α – умовний предикат ($\alpha \in_p P$), а β

– функція оновлення:

$$New(M) = M \overset{\Delta}{\leftarrow}{}^{\alpha}_{\beta} \Delta M \quad (2)$$

Істинність α перевіряється для кожної пари рядків $\langle str_i, \Delta str_j \rangle$, $i = \overline{1,b}$, $j = \overline{1,v}$, що належать таблицям M і ΔM відповідно. Якщо для якого-небудь рядка Δstr_j знайшовся такий рядок str_i, що умова α істинна, то необхідно оновити знайдений рядок відповідно до функції β:

$IF\ \alpha = \text{TRUE THEN}\ \beta\langle \text{str}_\text{i}, \Delta str_j \rangle, i = \overline{1,b}, j = \overline{1,v}\ \text{ELSE RETURN FALSE}$.

Рядок str_i, що узгоджується з рядком Δstr_j відповідно до умови α, оновлюється наступним чином. Задамо функцію β як трійку виду:

$$\langle C, op1(x_a), op2(x_a) \rangle, \quad (3)$$

де C – умовна функція;

op1 та *op2* – алгебраїчні операції або операції алгебри множин;

x_a, $x_a \in str_i$ – деякий атомарний запис, що зберігається в МП.

Даний вираз відповідає наступному предикату:

IF CONDITION = TRUE THEN $op1(x_a)$ ELSE $op2(x_a)$

а. Оновлення ієрархічного запиту

Розглянемо загальну структуру запиту з ієрархією:

SELECT $T.поле_1$, $T.поле_2$, ..., $T.поле_n$ FROM T
CONNECT BY $T.h_1$, $T.h_2$, ... $T.h_m$
START WITH $T.s_1$, $T.s_2$, ... $T.s_z$
WHERE $T.p_1$, $T.p_2$,..., $T.p_k$,

де $T.s_j$, $j = \overline{1,z}$ – умовні предикати, що задають умову вибору кореневих вершин;

$T.h_i$, $i = \overline{1,m}$ – умовні предикати, що задають умову формування ієрархії.

Наведемо алгоритм оновлення даного типу запитів.

1) При виконанні користувальницької операції INSERT:

Крок 1. Отримати дерево по заданій умові формування ієрархії, для якого коренем буде новий рядок.

SELECT *поля_результату* FROM *назва_таблиці*
CONNECT BY *умова_ієрархії*
START WITH NEW.*умова_початку*
WHERE *умови*

Крок 2. Для кожної вершини отриманого дерева виконуємо перевірку на її наявність в основному дереві, яке зберігається в МП.

Крок 2.1. Якщо вершина знайдена, оновлюємо її поля та переходимо до наступної вершини.

Крок 2.2. Якщо вершина не знайдена, додаємо її в дерево з МП.

β = <new $\in$ M, x:=new, M$\cup$new>

2) При виконанні користувальницької операції DELETE:

Крок 1. Отримати дерево по інвертованій умові формування ієрархії, для якого коренем буде новий рядок.

Пояснимо поняття інвертованої умови.

Нехай $h_i = f_{cond}$(PRIOR $T.поле_e$, $T.поле_x$), $e = \overline{1,q}$, $x = \overline{1,q}$, де:

- f_{cond} – деякий умовний оператор,
- PRIOR $T.поле_e$ – поле, відповідне попередньому вузлу ієрархії,
- $T.поле_x$ – поле, відповідне поточному вузлу ієрархії.

Тоді інвертованою будемо називати таку конструкцію:

$\square h_i = f_{cond}$(PRIOR $T.поле_x$, $T.поле_e$), $e = \overline{1,q}$, $x = \overline{1,q}$

Крок 2. Видалити з дерева МП вершини, що зустрічаються в отриманому дереві:

β = <old $\in$ M, M/old, x>

3) При виконанні користувальницької операції UPDATE:

Крок 1. Визначити яке з полів рядка було модифіковано.

Крок 2. Якщо модифіковано поля умови формування ієрархії, необхідно виконати послідовно операції видалення вузла дерева, що відповідає старому значенню, і вставки нової вершини. Інакше переходимо до кроку 3.

Крок 3. Якщо модифіковано поля умови пошуку кореневої вершини, перевірити чи є змінена вершина коренем дерева з МП.

Крок 3.1. Якщо вершина коренева, перевіряємо умову пошуку кореня з новим значенням поля. Якщо умова виконана, переходимо до кроку 4. Інакше видаляємо дерево з МП.

Крок 4. Оновлюємо модифіковану вершину:

β = <old = x, x := new, x>

b. Оновлення запиту з використанням аналітичних функцій

Розглянемо загальну структуру запиту, що містить аналітичні функції:

SELECT f_{analyt}(*T.поле_ф*) OVER (PARTITION BY *поле_гр*
ORDER BY *поле_с* [ASC|DESC] (ROWS|RANGE) BETWEEN
((*n*|UNBOUNDED) PRECEDING|CURRENT ROW) AND
((*m*|UNBOUNDED) FOLLOWING|CURRENT ROW))
FROM *T* WHERE $T.p_1, T.p_2, \ldots, T.p_k$,

де f_{analyt} – аналітична функція;

поле_ф – поле, на основі якого обчислюється аналітична функція;

поле_гр – поле, по якому виконується групування;

поле_с – поле, по якому виконується сортування;

n – нижня межа вікна;

m – верхня межа вікна;

$T.p_i,\ e=\overline{1,q}$ – умовні предикати.

Введемо наступні позначення:

- asc|desc(*поле_с*) – виконання операції ORDER BY *поле_с* ASC|DESC;
- *x* – поточний рядок в курсорі;
- *rownum* – індекс поточного рядка в курсорі;
- *current_row* – рядок МП, що був модифікований в БТ;
- *val* – поле МП, в якому зберігається результат виконання функції.

Розглянемо оператор оновлення аналітичної функції на прикладі наступної синтаксичної конструкції:

COUNT(*поле_ф*) OVER (PARTITION BY *поле_гр* ORDER BY ASC *поле_с*
ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW)

При вставці елемента необхідно виконати наступну послідовність кроків:

Крок 1. Знайти множину рядків МП, яка задовольняє умові групування. Із знайденої множини необхідно вибрати рядок, який передує новому рядку, і записати значення поля *val* зі знайденого рядка в аналогічне поле нового рядка:

<new.поле_гр=x.поле_гр, desc(поле_с), x>
<new.поле_гр=x.поле_гр &&
x.поле_с<new.поле_с &&
rownum=1, new.val:=x.val, new>

Якщо рядок не знайдено, аналогічним чином необхідно виконати пошук наступного рядка після нового:

<new.поле_гр=x.поле_гр, asc(поле_с), x>
<new.поле_гр=x.поле_гр &&
x.поле_с>new.поле_с &&
rownum=1, new.val:=x.val, new>

Крок 2. Виконати вставку нового рядка:

<true, M∪new, x>

Крок 3. Якщо множина рядків МП була попередньо упорядкована за зменшенням, необхідно відновити початковий порядок прямування елементів:

*<new.*поле*_гр=x.поле_гр, asc(поле_с), x>*

Крок 4. Інкрементуємо значення поля *val* нового рядка і усіх наступних рядків МП, які відповідають умові групування:

<new.поле_гр=x.поле_гр &&
поле_с≥new.поле_с &&
new.поле_ф≠null, x.val+1, x>

Дана послідовність кроків вказана для фрази *ORDER BY ASC* в синтаксисі запиту. Якщо в запиті зазначено зворотний порядок сортування, необхідно в усіх кроках змінити операції сортування та порівняння (> й <) на протилежні.

Нижче приведено спрощений запис запитів в термінах SQL, що відповідає описаної послідовності кроків:

*SELECT val INTO val_var FROM (SELECT **
FROM MP WHERE поле_гр=new.поле_гр ORDER BY поле_с DESC)

WHERE поле_с<new.поле_с AND rownum<2;

INSERT INTO MP (val, поле_гр, поле_с, поле_ф)
VALUES (val_var, new.поле_гр, new.поле_с, new.поле_ф);

UPDATE MP SET val=val+1
WHERE поле_гр=new.поле_гр AND поле_с> new.поле_с;

При видаленні елемента виконуємо наступне:

Крок 1. Видаляємо старий рядок з МП:

<true, M/old, x>

Крок 2. Декрементуємо значення поля *val* усіх рядків, що передують видаленому та задовольняють умові групування:

<old.поле_гр = x.поле_гр &&
поле_сold.поле_с &&
old.поле_ф ≠ null, x.val-1, x>

При зворотному порядку сортування елементів МП, модифікуємо усі наступні рядки групи:

<old.поле_гр = x.поле_гр &&
поле_с ≥ old.поле_с &&
old.поле_ф ≠ null, x.val-1, x>

Операція модифікації рядка МП, у випадку, коли змінюється значення поля групування, зводиться до послідовності виконання операцій видалення рядка з МП зі старим значенням і додаванням нового рядка. Якщо змінюється поле функції і нове значення дорівнює *null*, виконуємо декремент поля *val* для усіх рядків, наступних за поточним у цій групі:

<new.поле_гр ≠ old.поле_гр, delete(old) && insert(new),
<new.поле_ф ≠ old.поле_ф && new.поле_ф = null,

<old.поле_гр = x.поле_гр &&
поле_с ≥ old.поле_с &&
old.поле_ф ≠ null, x.val-1, x>, x>, x>

c. Перетворення логічного запису функції оновлення на SQL

На рис. 1 наведено розроблений загальний алгоритм генерації функцій оновлення в термінах SQL.

Наведений алгоритм дозволяє перетворити розроблені вище функції оновлення у логічній формі запису до вигляду, придатному для подальшої генерації SQL тригерів для оновлення МП. В ході знаходження у запиті агрегацій, групувань, об'єднань тощо формуються відповідні проміжні функції оновлення, які потім склеюються у результуючу функцію оновлення.

Результати дослідження автоматизованої технології оновлення МП

Порівняємо якість удосконаленої в даній роботі технології генерації функцій оновлення з методом, запропонованим раніше в [1].

Використаємо для цієї задачі журнал виконання трансакцій СКБД за одноденний, тижневий і місячний період роботи досліджуваної ІС. Класифікуємо запити, виконані за вказаний період:

- простий запит (I);
- простий запит з групуванням (II);
- простий багатотабличний запит (III);
- підлеглі запити на вибірку, корельовані запити (IV);
- просте об'єднання за рівністю (V);
- багатотабличний запит (VI);
- ієрархічний запит (VII);
- запит з аналітичними функціями (VIII).

В результаті простого аналізу перерахованих вище журналів трансакцій отримані наступні статистичні дані (табл. 1).

Таблиця 1

Розподіл запитів за різні періоди часу

Часовий період	Кількість запитів виду SELECT	Частка запитів від загального числа, %							
		I	II	III	IV	V	VI	VII	VIII
1 день	11352	9	5	2	15	11	48	7	3
1 тиждень	59626	12	4	4	18	13	37	8	4
1 місяць	242094	10	6	2	13	17	41	8	3

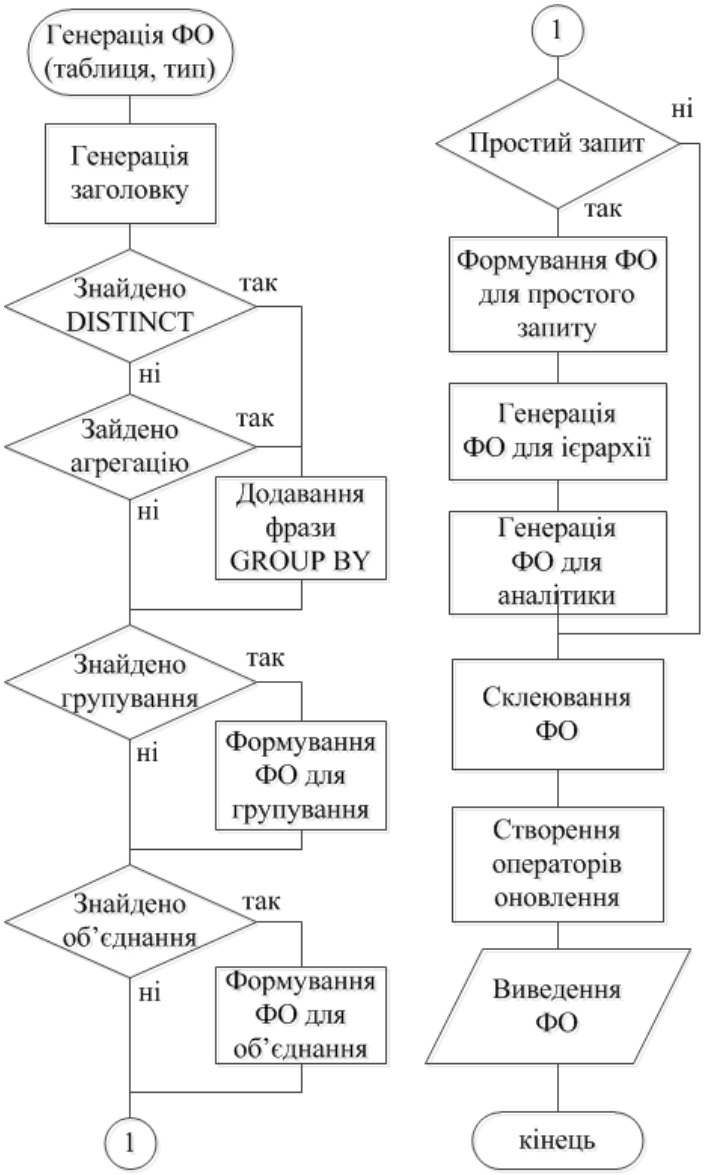

Рис. 1. Загальний алгоритм генерації функцій оновлення

Для оцінки якості розробленої технології генерації функцій оновлення були матеріалізовані всі унікальні запити, що зустрічалися в журналі трансакцій за певний період часу. Отримані результати були усереднені і зазначені в процентних співвідношеннях в табл. 2.

Таблиця 2

Результати матеріалізації запитів

Тип запиту	Базовий метод, %	Удосконалений метод, %
I	93	93
II	92	92
III	89	89
IV	80	80
V	93	93
VI	69	69
VII	0	62
VIII	0	55

Гістограма, яка відображає якість розробленої технології генерації функцій оновлення у порівнянні з базовою, наведена на рис. 2.

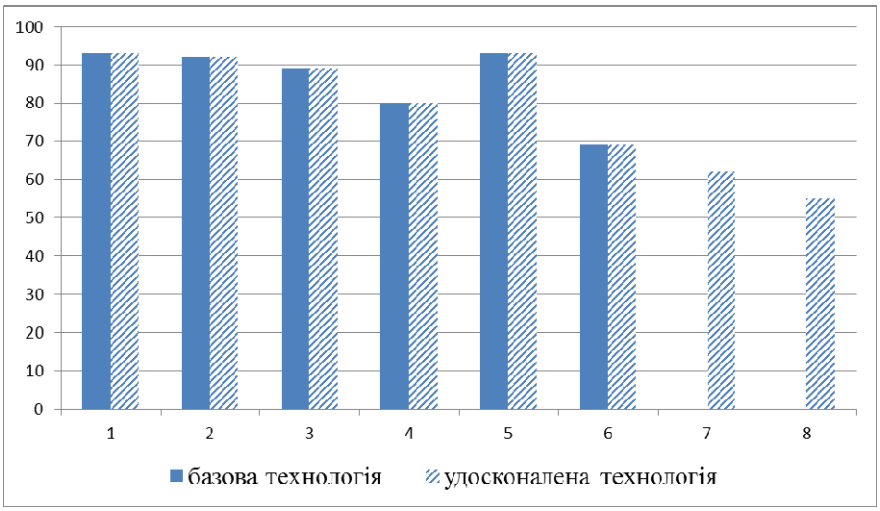

Рис. 2. Порівняння якості технологій генерації функцій оновлення

Гістограма показує, для скількох запитів кожної групи вдалося створити функції оновлення і який відсоток з них виявився логічно вірним. Для груп VII і VIII базовий метод не містить алгоритмів генерації функцій оновлення, тому частка оброблених запитів для цих груп дорівнює 0 %.

Результати дослідження підтвердили можливість запропонованого методу забезпечити створення у середньому на 7 % більше функцій оновлення в порівнянні з базовою технологією. Частка логічно вірних функцій оновлення дорівнює 62 % та 55 % для ієрархічних та аналітичних видів запитів відповідно, що свідчить про можливість подальшої оптимізації створених функцій оновлення.

Обмеженням на використання отриманих результатів є генерація функцій оновлення для складних запитів, що містять одночасно ієрархії, аналітичні функції, кореляції тощо. Дане обмеження викликано все ще досить високим відсотком (до 45 %) помилок синтаксису при генерації запитів.

Дані дослідження є продовженням роботи зі створення автоматизованої технології створення і обслуговування матеріалізованих представлень у методі інкрементального оновлення МП.

Отримані результати доцільно використовувати для оптимізації продуктивності ІС, у яких в якості моделі представлення даних обрана РБД.

Висновки

Була вдосконалена технологія автоматизованого оновлення МП шляхом поширення її на ієрархічні запити та запити з аналітичними функціями:

1. Складено функції оновлення у логічній формі запису для ієрархічних і аналітичних запитів. Для кожної операції додавання, видалення і оновлення даних у базових таблицях надано операції застосовні до таблиць оновлення замість безпосередньої модифікації МП.

2. Розроблено алгоритм перетворення логічних функцій оновлення на тригери мови SQL, що поширює можливості базового методу для матеріалізації ієрархічних запитів і запитів з аналітичними функціями.

З використанням удосконаленої технології вдалося створити у середньому на 7% більше функцій оновлення в порівнянні з базовою технологією, проте помилка в синтаксисі створених функцій оновлення для даних груп запитів ще досить велика.

Література

1. Новохатская Е. А. Методика генерации функций обновления в методе инкрементального обновления материализованных представлений / Е. А. Новохатская, Ю. Н. Возовиков // Вісник СумДУ. Серія «Технічні науки». – 2011. – № 3. – С. 82–96.

2. Gupta H. Incremental Maintenance of Aggregate and Outerjoin Expressions / H. Gupta, I. S. Mumick // Information Systems. – 2006. – Vol. 31 (6). – P. 435–464.

3. Gluche D. Incremental Updates for Materialized OQL Views. / D. Gluche, T. Grust, C. Mainberger, M. H. Scholl // Proc. of the 5th International Conference on Deductive and Object-Oriented Databases. – 1997. – P. 52–66.

4. Grust T. Query Evaluation in CROQUE – Calculus and Algebra Coincide / T. Grust, J. Kröger, D. Gluche, A. Heuer, M. H. Scholl // Proc. of the 15th British National Conferenc on Databases. – 1997.

5. Buneman P. Comprehension Syntax / P. Buneman, L. Libkin, D. Suciu, V. Tannen, L. Wong // SIGMOD Record. – 1994. – Vol. 23 (1). – P. 87–96.

6. Fegaras L. A Uniform Calculus for Collection Types / L. Fegaras // Technical Report 94–030. – Oregon Graduate Institute of Science & Technology, 1994.

7. Fegaras L. An Algebraic Framework for Physical OODB / L. Fegaras, D. Maier // Proceedings of the 5th International Workshop on Database Programming Languages. – 1995. – P. 6–8.

8. Nica A. Incremental maintenance of materialized views with outerjoins / A. Nica // Information Systems. – 2011. – vol. 31 (6). – P. 435–464.

9. Larson P.-A. Efficient Maintenance of Materialized Outer-Join Views / P.-A. Larson, J. Zhou // Proceedings of the 23rd International Conference on Data Engineering. – 2007. – P. 56–65.

10. Griffin T. Incremental maintenance of views with duplicates / T. Griffin, L. Libkin // Proceedings of the ACM SIGMOD International Conference on Management of Data. – 1995. – P. 316–327.

References

1. Novokhatska, K., Vozovikov, Y. (2011). The method of update function generation for the incremental maintenance of materialized views. Visnyk SumDU. Seriya "Technichni nauky", 82–96.

2. Gupta, H., Mumick, I.S. (2006). Incremental Maintenance of Aggregate and Outerjoin Expressions. Information Systems, 31 (6), 435–464.

3. Gluche, D., Grust, T., Mainberger, C., Scholl, M.H. (1997). Incremental Updates for Materialized OQL Views, Proc. of the 5th International Conference on Deductive and Object-Oriented Databases, 52–66.

4. Grust, T., Kröger, J., Gluche, D., Heuer, A. (1997). Query Evaluation in CROQUE – Calculus and Algebra Coincide, Proc. of the 15th British National Conferenc on Databases.

5. Buneman, P., Libkin, L., Suciu, D., Tannen, V., Wong, L. (1994). Comprehension Syntax, SIGMOD Record, 23 (1), 87–96.

6. Fegaras, L. (1994). A Uniform Calculus for Collection Types, Technical Report 94–030, Oregon Graduate Institute of Science & Technology.

7. Fegaras, L., Maier, D. (1995) An Algebraic Framework for Physical OODB, Proc. of the 5th International Workshop on Database Programming Languages, 6–8.

8. Nica, A. (2012). Incremental maintenance of materialized views with outerjoins, Information Systems, 37 (5). – p. 430–442.

9. Larson, P.A., Zhou, J. (2007). Efficient Maintenance of Materialized Outer-Join Views, Proc. of the 23rd International Conference on Data Engineering, 56–65.

10. Griffin, T., Libkin, L. (1995). Incremental maintenance of views with duplicates, Proc. of the ACM SIGMOD International Conference on Management of Data, 316–327.

Рецензія/Peer review : 15.11.2017 р. Надрукована/Printed :06.12.2017 р.
Рецензент: к.т.н., проф. Кунгурцев О.Б.