УДК 621.452.001.57:681.54

Ю. В. Шабатура, д. т. н., доц.;

І. М. Штельмах, асп.;

М. Ю. Шабатура, студ.

ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ОБЧИСЛЮВАЛЬНИХ СИСТЕМ НА ОСНОВІ ГЕНЕТИЧНОЇ ОПТИМІЗАЦІЇ SQL-ЗАПИТІВ ДО БАЗ ДАННИХ

Розглянуто актуальну проблему підвищення ефективності обчислювальних систем інтегрованих в комп'ютерні мережі. Її вирішення в частині роботи з базами даних пропонується у формі автоматичної оптимізації SQL-запитів, яка здійснюється у формі модифікації структури запитів з використанням генетичних алгоритмів.

Формалізовано задачу оптимізації SQL-запитів, запропоновано структурну схему системи оптимізації, розроблено об'єктну модель запиту, яка формується програмою-парсером на основі його тексту та дозволяє модифікувати запит програмно. Наведено алгоритм генетичної оптимізації і його програмну реалізацію, та результати експериментального дослідження, які підтверджують практичну цінність запропонованого підходу.

Вступ

Яскравим прикладом обчислювальних систем інтегрованих в комп'ютерні мережі (ОСІКМ), є пошукові системи в мережі Internet. Практично весь робочий час таких систем витрачається на обробку запитів до баз даних. Швидкість виконання цих запитів та споживання ними обмежених обчислювальних ресурсів обчислювальних систем (ОС) часто стають основною причиною погіршення динаміки ОСІКМ, або навіть відмови в обслуговуванні запиту внаслідок перевищення допустимого часу очікування.

Проведені дослідження показали, що синтаксис SQL запитів в сучасних СУБД є надзвичайно гнучким, і одержати однаковий результат можна, використовуючи різні комбінацій операторів запиту. Разом з тим спосіб побудови SQL запитів суттєво впливає на швидкість їх виконання [1]. Наприклад, при виконанні запиту SELECT на вибірку даних з використанням оператора обмеження WHERE, що визначає задані номера стовпців, швидкість виконання буде залежати від порядку запису цих номерів [2].

Отже сьогодні постає актуальна задача автоматичної оптимізації структури SQL запиту шляхом формування такої комбінації його операторів, яка дозволяє досягти мінімального часу для його виконання.

Постановка задачі

В роботі пропонується новий підхід, який дозволяє суттєво підвищити ефективність обчислювальних систем на основі проведення оптимізацію продуктивності SQL-запитів, яка здійснюється з використанням генетичних алгоритмів.

Об'єктом дослідження обрано СУБД MySQL, яка є найбільш поширеною в сучасних ОСІКМ.

Суть задачі полягає у тому, що на початковому етапі оптимізації в систему надходить запит Q_0, виконання якого приводить до отримання множину даних D_0 через інтервал часу $t = t_0$. Система виконує формування оптимальної комбінації операторів в даному запиті таким чином, щоб одержаний запит Q_{opt} повертав множину даних $D_{opt} = D_0$ за проміжок часу $t = t_{\min} < t_0$.

Функціональна організація системи оптимізації запитів SQL

Функціональна структура системи оптимізації запитів SQL наведена на рис. 1. Вона складаєть-

Рис. 1. Функціональна структура системи оптимізації запитів SQL

ся з двох основних складових: програми-парсера мови SQL та програми, що проводить процедури генетичної оптимізації.

Система функціонує таким чином. Початковий запит Q_0 обробляється програмою-парсером і формує його об'єктну модель $\hat{Q}_0$, а також виконує зворотне перетворення модифікованої об'єктної моделі $\tilde{\hat{Q}}_0$ в SQL запит $\tilde{Q}_0$ для його виконання та визначення змін продуктивності отриманого запиту на основі обчислення значення функції відповідності.

Процедура генетичної оптимізації кодує об'єктну модель запиту у вигляді хромосоми, та здійснює пошук оптимальної хромосоми, яка забезпечує мінімальне значення функції відповідності.

Значення функції відповідності визначає час виконання запиту SQL на сервері СУБД. Проведення вказаних процедур дозволяє отримати оптимальний запит Q_{opt}, який виконується у СУБД за мінімальний час.

Об'єктна модель SQL-запиту

Розробити програмне забезпечення для автоматичної обробки початкової форми SQL-запиту є задачею, яка не може мати однозначного практичного вирішення. Це пов'язано з тим, що синтаксис мови SQL-запиту представляє собою строкову змінну, з певним набором операторів та назв таблиць, полів і т.п. Тому, для автоматизації процедури оптимізації SQL-запиту була розроблена програма-парсер мови SQL, яка дозволяє отримувати об'єктну модель даного запиту у зручному для подальшої обробки вигляді. Діаграму Об'єктна модель SQL-запиту у формі діаграми класів і зв'язків її компонент показана на рис. 2.

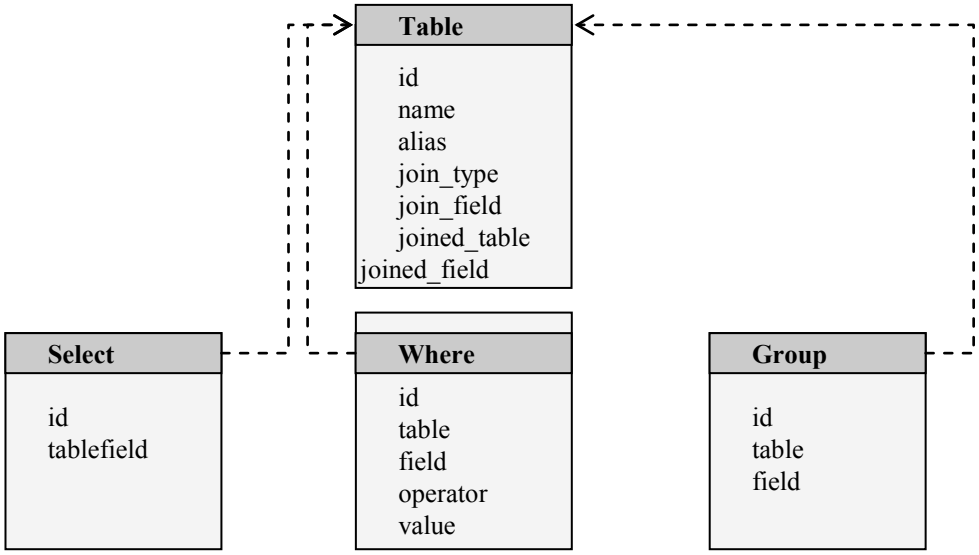

Рис. 2. Об'єктна модель SQL- запиту

Згідно цієї моделі SQL-запит містить такі основні класи об'єктів: Select, Table, Where, Group. Клас Select описує поля, з яких буде складатися вибірка данних, містить атрибути номеру поля (id), коду таблиці до якої належить дане поле (table) та назви поля (field). Клас Table описує таблиці з яких здійснюється вибірка, містить атрибути номеру таблиці (id), її назви (name), коду (alias), типу зв'язку з іншою таблицею (join_type), назви поля, за яким встановлюється зв'язок з іншою таблицею (join_field), коду таблиці, з якою встановлюється зв'язок (joined_table) та назву її поля (joined_field). Клас Where описує обмеження, які накладаються на вибірку. Містить атрибути номера обмеження (id), коду таблиці (table) та поля (field), на які накладається обмеження, оператора обмеження (operator) та його значення (value). Клас Group описує умови групування результатів вибірки, містить атрибути номера умови (id), коду таблиці (table) та назви поля (field), за якою здійснюється групування.

Розроблена об'єктна модель дозволяє повністю описати SQL-запит, може бути легко модифіко-

ваною необхідним чином та знову бути перетвореною у строкову змінну SQL-запиту для подальшого виконання. Однак, наведена модель не є повною, оскільки синтаксис мови SQL є досить складним, тому в процесі практичного впровадження дану модель буде розширено і іншими об'єктами для того, щоб вона могла описати SQL-запит будь-якої складності.

Оптимізація SQL-запиту з використанням генетичного алгоритму

Алгоритм генетичної оптимізація SQL-запиту передбачає виконання таких етапів:

1. Отримання об'єктної моделі вхідного запиту $Q_0 \to \overline{Q}_{0obj}$.

2. Кодування об'єктної моделі в двійкову строку (хромосому) шляхом її трансформації $\overline{Q}_{0obj} \to \overline{v}_0$.

3. Генерація початкової популяції хромосом $\overline{V}_{start}$.

4. Розрахунок функції відповідності кожної хромосоми $eval(v_k)$.

5. Відбір оптимальних хромосом.

6. Схрещування та мутація

Даний алгоритм є циклічним, етапи 4—6 виконуються задану кількість ітерацій, з метою пошуку хромосоми SQL-запиту який буде виконуватись найшвидше.

Перетворення вхідного запиту Q_0 в об'єктну модель $\overline{Q}_{0obj}$ здійснюється за допомогою розробленого парсера мови SQL.

Суть оптимізації SQL-запитів у цій роботі полягає в пошуку такої трансформованої об'єктної моделі, яка забезпечить найменшу тривалість виконання запиту. Для здійснення трансформації використовуються схеми оптимізації запитів, наведені в офіційній документації СУБД MySQL [1] та відкриті на основі власних спостережень. Наведемо дві основні схеми трансформації запитів, використані в цій роботі:

1. Обмеження WHERE дає кращі результати, якщо першою в ньому стоїть умова яка відсікає якомога більший набір записів.

2. Підключення допоміжних таблиць може значно збільшувати тривалість виконання SQL-запитів, оскільки під час обробки значних обсягів даних СУБД вимушена створювати тимчасові таблиці для їх зберігання. Заміна таких допоміжних таблиць окремими масивами-словниками безпосередньо в програмі дозволяє прискорити виконання запиту та розвантажити сервер баз даних.

Для кодування першої схеми вводиться множина чисел $W = \{w_1, w_2, \ldots, w_{nw}\}$, які позначають порядковий номер кожного виразу умови, де nw — кількість виразів умови. Числа, які позначають порядковий номер, кодуються у двійковому вигляді, розмірність яких ws залежить від кількості виразів умови nw.

Для кодування другої схеми використано множину двійкових чисел $D = \{d_1, d_2, \ldots, d_{nt}\}$, де d_n — змінна що визначає заміну таблиці з порядковим номером n програмним словником даних, nt — кількість таблиць, які використовуються в запиті.

Таким чином хромосома буде сукупністю множин двійкових чисел, які кодують схеми трансформації. Для наведених двох схем трансформації хромосома набуде вигляду

$$v = W \cup D = \{w_1, w_2, \ldots, w_{nw}, d_1, d_2, \ldots, d_{nt}\}. \quad (1)$$

Для генерації початкової популяції хромосом використовується генератор випадкових чисел, який генерує популяцію ($popsize$) хромосом (двійкових чисел) розмірністю

$$vs = nw * ws + nt. \quad (2)$$

Для ров'язуваної задачі оптимізації функція відповідності еквівалентна цільовій функції (тривалості виконання SQL запиту)

$$eval(v_k) = f\left(x^k\right),\ k = 1, 2, \ldots, popsize, \quad (3)$$

де $f\left(x^k\right)$ отримується на основі виконання SQL-запиту закодованого хромосомою x^k за допомогою процедури обчислення функції відповідності.

За допомогою функції відповідності виконується оцінка хромосоми за ступенем їх пристосова-

ності до виконання критерію оптимізації [3].

Для відбору найпридатніших хромосом використано підхід «колесо рулетки», наведений в [3]. Відбір здійснюється на основі функції розподілу, яка будується пропорційно вичисленим функціям відповідності згенерованих варіантів хромосом. Короткий опис алгоритму наведено нижче:

1. Обчислити значення функції відповідності $eval(v_k)$ для кожної хромосоми згідно (1).

2. Обчислити загальну функцію відповідності популяції

$$F = \sum_{k=1}^{popsize} \left(eval\ (v_k) - \min_{j=1,popsize} \left\{ eval\ (v_j) \right\} \right). \tag{4}$$

3. Обчислити ймовірність відбору p_k для кожної хромосоми v_k

$$p_k = \frac{eval(v_k) - \min\limits_{j=1,popsize} \left\{ eval\ (v_j) \right\}}{F}, \quad k = 1, 2, \ldots, popsize\ . \tag{5}$$

4. Обчислити сукупну ймовірність q_k для кожної хромосоми v_k

$$q_k = \sum_{j=1}^{k} p_j\ , \qquad k = 1, 2, \ldots, popsize\ . \tag{6}$$

Процес відбору починається з обертання колеса $popsize$ разів; при цьому за кожен цикл обирається одна хромосома:

1. Генеруємо випадкове число r з інтервалу $[0,\ 1]$.

2. Якщо $r \le q_1$, то вибираємо першу хромосому v_1; інакше вибираємо k-ту хромосому v_k $(2 \le k \le popsize)$ таку що $q_{k-1} < r \le q_k$.

Для схрещування хромосом використано метод з одною точкою обміну, наведений в [3]. У відповідності з цим методом, випадково обирається одна точка обміну, відносно якої міняються місцями частини хромосом-батьків. Алгоритм процедури схрещування наведено на рис. 3, де sp — ймовірність схрещування.

Мутація полягає в зміні одного або більше генів з ймовірністю рівною коефіцієнту мутації mp. При роботі з бінарними хромосомами мутація полягає в інверсії відповідного біта. Випадковим чином обираються $mn = Round(mp * [vs * popsize])$ цілі числа з інтервалу $[1, vs * popsize]$ та утворюють множину $M = \{m_1, m_2, \ldots, m_{mn}\}$. Ця множина чисел визначає порядкові номери бітів із загальної сукупності бітів хромосом, які підлягають мутації. Процедура мутації здійснюється за алгоритмом, наведеним на рис. 4.

```
begin
  k :=0 ;
 while (k ≤ popsize) do
     r_k := random [0, 1] ;
     if (r_k < sp) then
    вибрати хромосому v_k для схрещування;
     end;
     k: = k + 1 ;
   end;
end.
```

Рис. 3. Алгоритм процедури схрещування

```
begin
  k := 0 ;
 while (k ≤ popsize) do
   i := 0;
   while (i ≤ vs) do
if((k * vs + i) in M) then v_k[i] = not v_k[i];
        end;
         i: = i + 1;
      end;
      k: = k + 1 ;
end;
 end.
```

Рис. 4. Алгоритм процедури мутації

Результати експериментального дослідження і його аналіз

Для проведення експериментального дослідження вибрано складний SQL-запит який використовується в реальній системі аналізу даних продаж та здійснює вибірку з чотирьох таблиць, обробляє понад 300 000 запитів. Модель такого запиту на мові SQL наведено на рисунку 5. Тривалість виконання склала $t = t_0 = 28{,}832$ с.

```
select ap.customer_id, af.fabric, ap.value, ap.type, cc.kurs, ap.month from ArtikelsPrognozes ap join tbl_customers c ON c.Auftr_geb_ = ap.customer_id join currency cc ON cc.cur_name = c.Wahrg left join tbl_article_fabrics af ON af.customer = ap.customer_id AND af.article = ap.article where ap.year ='2008' and ap.value > 0 group by ap.customer_id
```

Рис. 5. Початковий SQL-запит

За допомогою парсера SQL здійснено перетворення $Q_0 \to \overline{Q}_{0obj}$ та одержано об'єктну модель, показану на рис. 6.

На основі об'єктної моделі одержується кількість виразів умови $nw = 2$ та кількість таблиць, які використовуються в запиті $nt = 4$. Хромосома має вигляд $v = [$хх хх хххх$]$, де перші дві пари двійкових чисел позначають порядкові номери виразів умови, чотири останні числа позначають необхідність заміни таблиць програмними словниками даних.

Fields:

Id	1	2	3	4	5	6
Table	ap	af	ap	ap	cc	ap
Field	customer_id	fabric	value	type	kurs	month

Tables:

Id	Name	Alias	JoinType	JoinField	JoinedTable	JoinedField
1	ArtikelsPrognozes	ap				
2	tbl_customers	c	JOIN	Auftr_geb_	ap	customer_id
3	currency	cc	JOIN	cur_name	c	Wahrg
4	tbl_article_fabrics	af	LEFT JOIN	article	ap	article

Where rules:

Id	Table	Field	Operator	Value
1	ap	year	=	'2008'
2	ap	value	>	0

Рис. 6. Об'єктна модель вхідного запиту

Після генерації початкової популяції хромосом та проведення генетичної оптимізації, на 258-й ітерації, було одержано оптимізований запит зтрансформований хромосомою $v = [10010001]$, модель даного запиту на мові SQL наведено на рис. 7. Графік процесу оптимізації наведено на рис. 8.

```
select ap.customer_id, ap.value, ap.type, cc.kurs, ap.month from ArtikelsPrognozes ap join tbl_customers c ON c.Auftr_geb_ = ap.customer_id join currency cc ON cc.cur_name = c.Wahrg where ap.value>0 and ap.year='2008' group by ap.customer_id
```

Рис. 7. Оптимізований SQL-запит

Як видно з рисунку 7, одна з таблиць запиту була замінена програмним словником даних, а також було замінено порядок виразів умови, що дозволило одержати необхідну вибірку даних $D_{opt} = D_0$ за проміжок часу $t = t_{\min} = 1{,}034$ с, що склало 3 % від тривалості виконання початкового запиту.

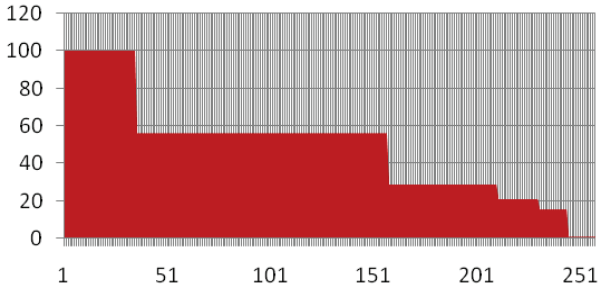

Рис. 8. Часова діаграма процесу оптимізації

На рисунку 8 показано графік залежності часу виконання оптимізованого запиту від номера ітерації генетичного алгоритму.

Висновки

1. Розроблено новий підхід, який дозволяє підвищити ефективність функціонування ОСІКМ шляхом оптимізації SQL-запитів.

2. Розроблено об'єктну модель SQL-запиту, яка дозволяє автоматично аналізувати та оптимально модифікувати його структуру.

3. Створено програму-парсер мови SQL та програму для генетичної оптимізації, що дозволило використати розроблений підхід на практиці.

4. Результати експериментальних досліджень засвідчили практичну цінність розробленого підходу, зокрема його високу ефективність для оптимізації складних запитів.

СПИСОК ЛІТЕРАТУРИ

1. Optimizing SELECT and Other Statements // Reference Manual for the MySQL Database System, v. 5.1. — Режим доступу: http://dev.mysql.com/doc/refman/5.0/en/query-speed.html.

2. Оптимизация производительности баз данных для Web. / Ахмед Абуалсемид // Библиотека Интернет Индустрии I2R.ru. — Режим доступу: www.i2r.ru/static/480/out_11207.shtml.

3. Ротштейн А. П. Интеллектуальные технологии в идентификации / А. П. Ротштейн. — Винница: Універсум– Вінниця, 1999, — 300 с.

Рекомендована кафедрою метрології та промислової автоматики

Надійшла до редакції 8.09.08
Рекомендована до друку 20.10.08

Шабатура Юрій Васильович — професор, ***Штельмах Ігор Миколайович*** — аспірант.
Кафедра метрології та п ромислової автоматики;
Шабатура Максим Юрійович — студент Інституту автоматики, електроніки та комп'ютерних систем управління.
Вінницький національний технічний університет