УДК 004.6

А.Ю. Яцишин
Національний технічний університет України "Київський політехнічний інститут"

ЗАСТОСУВАННЯ ГЕНЕТИЧНОГО АЛГОРИТМУ ДЛЯ ПРОЕКТУВАННЯ ГІБРИДНИХ СХОВИЩ ДАНИХ

Описано застосування генетичного алгоритму до проектування гібридних сховищ даних.

Ключові слова: гібридне сховище даних, проектування, генетичний алгоритм, застосування.

This paper describes application of genetic algorithm to hybrid warehouse design.

Keywords:hybrid data warehouse, design, genetic algorithm, application.

Постановка проблеми

При проектуванні сховищ даних постає питання коректної побудови структур даних для ефективного його функціонування.

У різних наукових публікаціях використовувалися генетичні алгоритми для вирішення тих чи інших питань проектування та оптимізації сховищ даних. Однак рішення, яке б враховувало існування і реляційної, і багатовимірної баз даних в одному сховищі запропоновано не було.

При цьому, актуальним та не вирішеним сьогодні є питання автоматизованого чи напівавтоматизованого проектування гібридних сховищ даних у цілому. Ця проблема випливає з практичної необхідності автоматизованої побудови сховищ даних з оптимальною швидкодією за вибраним критерієм оптимальності сховища.

Аналіз останніх досліджень та публікацій

Питаннями застосування генетичних алгоритмів до побудови сховищ даних займалися та оптимізації сховищ даних Wen-Yang Lin,I-Chung Kuo [1], Chuan Zhang, Xin Yao, Jian Yang [2,5], Ladjel Bellatreche,Kamel Boukhalfa [3], J.-T. Horng, Y.-J. Chang, B.-J. Liu [4], Goran Velinov, Danilo Gligoroski, Margita Kon Popovska [6], Michael Lawrence [7]

У праці [8] розглянуто основні принципи побудови та функціонування сховищ даних. У роботах [9] і [10] розглядаються питання проектування сховищ даних для бюджетування.

Стаття [1] описує генетично-жадний алгоритм для вибору куба даних OLAP. Задачею є вибрати куби даних з N вимірами з 2N можливих. Хромосома складена з 1 для вибраних кубів, 0 – для не вибраних. За функцію придатності взято функцію вартості, яка дорівнює

$$\sum_{i=1}^{n} f_{c_i} E(c_i.M) + g_u \sum_{c \in M} U(c,M), \quad (1)$$

де f_{c_i} – частота звернення до куба і, $E(c_i.M)$ – вартість оцінки куба і при поточній матеріалізації М, g_u – частота вставки в базове відношення, $U(c,M)$ – вартість обслуговування куба і при поточній матеріалізації М. Селекція вибирається шляхом генерації випадкового числа від 0 до 1 і вибору придатнішої хромосоми, якщо це число перевищує 0,75. Схрещування відбувається з імовірністю (точкою схрещування) 0,7. Мутація відбувається з імовірністю pm. Крім генетичного алгоритму, використовується жадний алгоритм для тих рішень, які не є допустимими. Обробка недопустимих рішень відбувається так : якщо пара батько-нащадок мають тих самих предків і вони матеріалізовані, тоді треба розматеріалізувати вузол-нащадок, перерахувати загальну вартість і замінити старе рішення новим. Це рішення може бути використано як елемент MAP для вибору кубів у багатовимірній базі даних, однак воно не вирішує питання побудови цього сховища.

У статті [2] вирішується питання вибору представлень для матеріалізації шляхом «об'єнання» планів виконання запиту. Хромосома складена так: 1 – якщо представлення матеріалізується, 0 – якщо ні. Функція придатності дорівнює Cmax-c(x), де c(x) – функція вартості, а Cmax – максимальне значення c(x) в популяції або в останніх k поколіннях. Використовується одно точкове схрещування з випадково вибраною точкою схрещування від 1 до n, де n – довжина хромосоми. Мутація відбувається за допомогою інвертування біта від 1 до n, що випадково вибирається в хромосомі.

Це рішення може бути використано для вибору матеріалізованих представлень.

У статті [3] розглядаються проблема вибору схеми розбиття сховища даних, використовується горизонтальне розбиття. Кожен атрибут фрагментації може бути представлений масивом з n елементами, де n відповідає числу його піддоменів. Значення цих елементів знаходиться між 1 і n. Якщо два елементи мають ті самі значення, то вони будуть об'єднані для формування одного. Це кодування може бути використане для представлення фрагментів таблиць вимірів і таблиці фактів. Для селекції використовується метод колеса рулетки (він виділяє сектор колеса, що відповідає і-й хромосомі і створює нащадка, якщо згенероване число знаходиться в околі 0 зупиняється усередині призначеного сектора рядка). Використовується двоточковий механізм схрещування. Значення придатності визначається за допомогою призначення балів за двома показниками : поріг і виконання запитів. Поріг визначає, чи перевищує кількість отриманих фрагментів ± 5 відсотків порогу, коли будуть призначені 55 балів, інакше – менше балів. Виконання запитів визнчає, чи перевищує вартість запиту встановлене значення, що обчислюється за допомогою функції вартості. Якщо це так, то призначається менше 3 балів за запит. Функцією вартості є

$$Cost(Q) = \sum_{j=1}^{N} valid(Q, S_i) \prod_{i=1}^{M_j} \left(\frac{Sel_F^{p_i} \times \|F\| \times L}{PS} \right), \quad (2)$$

де Mj,F,L і PS позначають відповідно кількість предикатів вибірки, що визначають фрагмент факту підсхеми типу «зірка» SDEj, кількість кортежів, присутніх в таблиці фактів F, розмір у байтах кортежу таблиці F та розмір сторінки файлової системи (у байтах).

Мутація відбувається з нормою 6–30%.

Це рішення може бути використано як елемент MAP для вибору схеми розбивки бази даних (як реляційної, так і багатовимірної), але тим самим вирішується лише питання оптимізації сховища даних.

У статті [4] розглядається застосування генетичного алгоритму до вибору матеріалізованих представлень. Хромосома складається з двох частин – вибраних планів виконання для кожного з запитів, та вибраних представлень для матеріалізації. Селекція відбувається за допомогою вибору числа від 1 до $\sqrt{n}$, потім це число підноситься до квадрата і вибирається індивід з отриманим порядковим номером, з упорядкованих індивідів за вартістю у порядку її збільшення. Використовується одне точкове схрещування шляхом схрещування обох частин хромосоми: рядка планів виконання запитів та рядка представлень для матеріалізації, кожна схрещується окремо. Мутація відбувається шляхом інвертування бітів в рядку представлень, а також змінює числа у рядку планів виконання, підтримуючи допустимість рішення.

Це рішення також може бути використано для вибору матеріалізованих представлень.

У статті [5] також розглядається проблема вибору матеріалізованих представлень. Як хромосоми взято номери планів виконання запитів, об'єднані в бінарний рядок. Функція придатності дорівнює Cmax-c(x), де c(x) – функція вартості, а Cmax – максимальне значення c(x) в популяції або в останніх k поколіннях. Використовується одно точкове схрещування. Мутація відбувається шляхом випадкового вибору плану виконання запиту, з усіх доступних для вибраного для мутації запиту. Використовується турнірна селекція, де порівнюються індивіди, і кращий з них проходить до наступної популяції. Розмір турніру від 4 до 7 індивідів.

Це рішення є більш раннім варіантом рішення [2] і також може бути використано для вибору матеріалізованих представлень.

Стаття [6] описує вибір індексів, матеріалізованих представлень для покращення реляційних баз даних. Ця стаття описує вибір індексів, матеріалізованих представлень для покращення реляційних баз даних. Використовується генетично-жадний алгоритм. Хромосома формується так. Об'єкт (просторові відношення, агрегатні представлення та індекси) представляється масивом бітів, тобто за частинами, які будуть об'єднані в хромосому. Просторові представлення подають (як особливий тип представлень для матеріалізації) з їх різними варіантами. Кількість бітів, необхідних для представлення кожного просторового відношення з всіма його варіантами, дорівнює k +1, де k – кількість елементів додаткового набору атрибутів. Тому перший біт використовується для представлення просторового відношення, а інші n бітів для представлення додаткових атрибутів. Всі просторові відношення мають бути матеріалізовані,тому кожне з них представляється 1. Атрибут в додатковому наборі при додаванні до його просторового відношення представляється 1,інакше 0. Агреговані представлення подають подібно ом, тобто для кожного представлення, один біт використовується для його представлення (1 – якщо вибране, 0 – не вибране для матеріалізації) і k бітів використовуються для представлення його додаткових атрибутів, тобто варіантів представлень. Атрибути додаткового набору агрегованих представляються у схожий спосіб з атрибутами додаткових наборів просторових відношень. Для кожного агрегованого представлення атрибути міри представляєься n бітами. Атрибут міри, якщо він додається до відповідного представлення, представляється 1, інакше 0. Для кожного представлення має бути доданий якнайменше один атрибут міри. Після представлення кожного подання відбувається представлення їх можливих індексів.Кожний індекс представляється одним бітом, тобто 1 – якщо індекс вибраний, 0 – не вибраний. Кількість і порядок індексів визначаються попередньо.

Цільова функція задається так. Нехай SCM є станом схеми кубу даних SC з набором AVM AV представлень-кандидатів на матеріалізацію, де кожний з них представляється його варіантом і набором SIM SI індексів-кандидатів. Нехай також всі просторові відношення представляються їх відповідними варіантами. Тоді проблема оптимізації, обмежена витратами на обслуговування є наступною: вибрати стан SCM схеми куба даних SC, яка мінімізує

$$\tau\ (SC, SCM, SQ) = \sum_{Q \in SQ} FQ * P(Q, SCM), \qquad (3)$$

при обмеженні U (SC, SCM) ≤ S, де SQ – набір попередньо визначених запитів, Q F – частота запитів, P(Q,SCM) позначає мінімальну вартість обробки запиту Q в стані SCM схеми SC. Нехай U(SC, SCM) – загальна вартість обслуговування, визначена як:

$$U(SC, SCM) = \sum_{R \in R_{SC}} GR * m(R, SCM) + \sum_{V \in AV_M} GV * m(V, SCM) + \sum_{I \in I_M} GI * m(I, SCM), \qquad (4)$$

де GR, GV and GI – частота оновлення відповідно відношень, представлень та індексів. Нехай m(R, SCM), m(V, SCM) and m(I, SCM) – мінімальна вартість обслуговування відношень, представлень і індексів відповідно в присутності стану SCM. P(Q, SCM) – цільова функція проблеми.

Алгоритми описують так:

Вхідними параметрами алгоритмів є: NC – кількість хромосом (розмір популяції), NG – кількість поколінь, LC – довжина хромосоми (кількість бітів, необхідних для представлення всього простору рішень, NF – кількість фрагментів простору рішень, яка дорівнює кількості кроків жадної процедури, SQ – набір запитів, AV – набір агрегатних представлень. POP(i) – і-те покоління популяції.

```
Algorithm 1: GGLA(NC, NG, LC, NF, AV, SQ)
Begin
AVB:=Ø;
Choose_views(Round(LC/NF), CL, AVB,AVC, AV);
Extend_chromosome(POP(1), AVC, NC);
Evaluate_population(POP(1), SQ);
Evaluate_population_materialization(POP(1));
```

```
j:=2;
For i=2 to NG Do
If Mod(i,Round(NG/NF))=1 Then
Choose_views(Round(LC*j/NF),CL,AVB,
AVC,AV);
Extend_chromosome(POP(i), AVB, NC);
j:=j+1;
End If;
Perform_crossover(POP(i-1), NC);
Perform_mutation(POP(i), NC);
Evaluate_population(POP(i), SQ);
Evaluate_population_materialization(POP(i));
Perform_selection(POP(i), NC);
End For;
End GGLA;
```

Параметрами підалгоритму 1 є : NL – Нова довжина хромосоми (вхідні дані), CL – поточна довжина хромосоми (вхідні/вихідні дані), AVB – набір вибраних представлень (вхідні/вихідні дані), AVC – набір вибраних представлень на поточному кроці (вихідні дані), AV – набір агрегованих представлень (вхідні дані).

```
Subalgorithm 1:Choose_views(NL, CL, AVB, AVC, AV )
Begin
AVC:= Ø ;
While NL>CL Do
B B i AV := AV    V, where Vi has maximal
URi in AV \ AVB;
C C i AV := AV    V ;
CL:= CL+LVi;
End While;
End Choose_views;
```

Перед початком оптимізаційного процесу всі агрегатні представлення упорядковуються за їх коефіцієнтом використовуваності (UR), тобто їх важливості. Загальна ідея – на певних кроках (NF) жадної процедури вибрати підмножини представлень з всіма їхніми бітами (підалгоритм, тобто процедура Choose_views) і додати їх до вже вибраних, тобто об'єднати їх випадково вибрані біти до вже створеної хромосом (процедура Extend_chromosome).Цими процедурами грубо корегується простір рішень. Після кожного кроку жадної процедури виконується тонка оптимізація за допомогою генетичного алгоритму (декілька поколінь в поточному просторі рішень). Процедура Evaluate_population оцінює якість рішень, а процедура Evaluate_population_materialization оцінює вартість обслуговування. Процедури Perform_crossover, Perform_mutation and Perform_selection виконують операції генетичного алгоритму – відповідно схрещування, мутацію і селекцію.

Другий алгоритм називається GGBA – Greedy-Genetic Binary Algorithm. Всі вхідні параметри є тими самими, що і в алгоритмі 1. POP(i,AVk) представляє і-те покоління популяції і складається з фрагмента хромосом, які представляються підмножиною рядків AVk. POP2(i) представляє і-те покоління представлє і-те покоління популяції і складається з цілих хромосом (створених за допомогою об'єднання всіх фрагментів в популяції POP), представлених набором всіх агрегованих представлень AV. Популяція POP2 необхідна для оцінки. Змінна NS – кількість кроків жадної процедури.

В процедурі Concat_chr ми визначаємо новий набір агрегованих представленьAVk := AV2*k 1 AV2*k і нову популяцію POP(i,AVk), яка складається з фрагментів з фрагментів хромосом, створених за допомогою об'єднання фрагментів хромосом популяцій POP(i,AV2*k-1) та POP(i,AV2*k). Фрагменти хромосом обох популяцій упорядковуються від оцінених краще до оцінених гірше, об'єднаних фрагментів у тій же позиції. Автори назвали цю стратегію об'єднання best-to-best. У подібний спосіб, процедура Concat_whole_chr створює POP2(i) як об'єднання з популяцій всіх підмножин наборів POP(i,AVk). Тут також використовується стратегія об'єднання best-to-best. Всі процедури, за виключенням Evaluate_population_materialization, можуть бути паралелізовані для різних фрагментів хромосом, тобто підмножини AVk з AV, що істотно покращує швидкодію алгоритму.

Це рішення являє собою генетично-жадний алгоритм і поєднує переваги як генетичного, так і жадного алгоритмів. Воно може бути використано для оптимізації реляційної бази даних.

Стаття [7] описує мультицільові генетичні алгоритми, які мають на меті як зменшення вартості виконання запитів, так і збільшення швидкодії запитів. Використовується метод колеса рулетки для селекції або турнірна селекція. Функцією придатності є

$$f(a) = \frac{f'(a)}{\sum_{b \in P} sh(d(a,b))}, \tag{5}$$

де d(a,b) – евклідова відстань між цілями a і b, sh – функція розділення, задана в алгоритмі, яка визначається так :

$$sh = \begin{cases} 1-(d/\sigma_{niche})^2, якщо\ d \le \sigma_{niche} \\ 0, в\ протилежному\ випадку \end{cases}, \tag{6}$$

де σ_{niche} – радіус ніші, параметр, який задається на основі оцінки мінімальної бажаної генотипної відстані між індивідами. Також у алгоритмах передбачено використання схрещування і кросинговеру індивідів. Описується також робота з обмеженнями і накладання штрафних функцій до величини придатності.

Алгоритми, представлені у даній статті, є складнішими, ніж у інших статтях, їх особливістю є досягнення оптимального рішення за декількома критеріямм, тобто маємо задачу багатокритеріальної оптимізації. Але розглядається лише проблема оптимізації багатовимірної бази даних класу OLAP за допомогою вибору матеріалізованих представлень. Проблема проектування бази даних не вирішується.

Розглянуті теоретичне рішення дають змогу оптимізувати сховище даних за допомогою індексів, матеріалізованих представлень та фрагментації. Оптимізацією є знаходження оптимальних рішень за заданими цільовими функціями, керуючись заданими функціями придатності. Однак ці методи не використовуються в комплексі, а методи не враховують поєднання реляційної та багатовимірної бази даних.

Тому необхідно запропонованувати такий генетичний алгоритм, який би поєднував індекси і матеріалізовані представлення для оптимізації сховища, а також розміщення таблиць сховища даних у різних базах даних.

Формулювання цілей статті

Ціллю статті є запропонувати генетичний алгоритм автоматизованого проектування гібридних сховищ даних, який б дозволив проектувати гібридне сховище даних на основі знань про предметну область (опису таблиць даних) та вимог до сховища даних (опису запитів даних)

Виклад основного матеріалу

За [6], сховище даних можна представити у вигляді пятірки

$$DW=<DB,rf,rm,RM,func>, \tag{7}$$

Де DB – бази даних; rf – множина відношень фактів для обох складових; RF – схема для rf; rm – множина відношень метаданих для обох складових; RM – схема для rm; func – множина процедур прийняття рішень.

При проектуванні сховища DB,rf,RF та func залишаються незмінними, оскільки маємо реляційну та багатовимірну бази даних незалежно від того, як вони спроектовані, те ж саме стосується відношень фактів (даних) та процедур прийняття рішень. Отже, при заданні математичної моделі потрібно враховувати такі змінні і цільову функцію, які впливають на RF (схеми баз даних).

Проаналізувавши вищенаведені алгоритми і евристики, можна сказати, що жоден з них не застосовний повною мірою до гібридних сховищ даних, бо не враховує особливостей одночасно реляційних та багатовимірних баз даних. Однак вищенаведені рішення можуть бути застосовані частково.

З огляду на необхідність оптимізації гібридного сховища даних запропоновано використати два механізми покращення роботи сховища – застосування індексів і матеріалізованих представлень.

Індексом будемо вважати об'єкт бази даних, який дає змогу отримувати дані з таблиць швидше за допомогою збереження словників даних, які будуються за одним чи декількома стовпцями таблиці, на якій визначається індекс. У реляційних БД поля індексу доцільно визначати за допомогою генетичного алгориму : у хромосому включаються гени, які відображають включення поля в індекс таблиці. В багатовимірних базах даних побудова індексів найчастіше інкапсульована, а тому створення індексів є або неможливим, або зайвим, оскільки при наявності двох наборів індексів продуктивність бази даних може бути знижена. Однак принцип індексування в алгоритмі може поширюватися і на багатовимірну базу даних. Індексуються ті поля, при індексуванні яких ЦФ на цій таблиці є мінімальною.

Матеріалізованим представленням вважатимемо збережену окрему від баз даних таблицю або таблиці даних, деякі поля яких вибирають генетичним алгоритмом. У хромосому включаються гени, які відображають включення поля в матеріалізоване представлення. Цей підхід повністю застосовний для реляційних баз даних, однак не застосовний для багатовимірних, оскільки розривання полів таблиці, що відповідає зрізу куба, може привести до порушення цілісності даних. Матеріалізуються тільки ті поля, при включенні яких у матеріалізоване представлення ЦФ на цій таблиці є мінімальною.

Визначимо цільову фінкцію алгоритму. Відповідно до постановки задачі критерієм оптимальності сховища є кількість доступів до даних, тобто операцій читання даних, які необхідно провести для виконання запиту до сховища даних.

Оскільки спроектувати базу даних під всі запити однаково ефективно неможливо, доцільно проектувати її під конкретні популяції запитів. Тому цільова функція задачі буде розраховуватися під перший запит популяції даних.

Цільова функція задачі має вигляд:

$$z = \sum_{i=1}^{n} A_i , \tag{8}$$

де n – кількість таблиць запиту, A_i – кількість доступів до i-ї таблиці реляційної бази даних або зрізу багатовимірної

Але в практичній діяльності, враховуючи особливості використання різних баз даних, і те, що той самий запит може виконуватися по-різному залежно від виконаної оптимізації баз даних, а також неможливість отримання інформації про кількість доступів з ядра бази даних можна замінити на

$$z = \sum_{i=1}^{n} t_i , \tag{9}$$

де n – кількість таблиць запиту, t_i – час доступу до i-ї таблиці реляційної бази даних або зрізу багатовимірної, T_j - – час виконання операції з'єднання над таблицями $i_1=j$ та $i_2=j+1$

Величини t_i залежать від трьох факторів: розміщення таблиць, індексування та матеріалізованості полів.

Розміщення таблиць у реляційній чи багатовимірній базі даних можна представити змінними розміщення таблиць L_t, де t – порядковий номер таблиці. Але з огляду на можливу наявність зв'язків між таблицями для зменшення кількості змінних доцільним є введення поняття *області сховища даних*. Областю сховища даних будемо вважати таку множину таблиць сховища даних, у якій на кожній таблиці визначено зовнішній ключ над деякою іншою таблицею цієї ж множини. Тому змінними ЦФ, що представляють цей фактор, є L_a :

$$L_a = \begin{cases} 1, \text{якщо всі таблиці області даних } a \text{ розміщені в багатовимірній БД} \\ 0, \text{якщо всі таблиці області даних } a \text{ розміщені в реляційній БД} \end{cases} \quad (10)$$

Індексування полів таблиць представляється змінними I_c:

$$I_c = \begin{cases} 1, \text{якщо індекс містить поле } c \\ 0, \text{в протилежному випадку} \end{cases} \quad (11)$$

Матеріалізованість полів таблиць представляється змінними M_c:

$$M_c = \begin{cases} 1, \text{якщо індекс містить поле } c \\ 0, \text{в протилежному випадку} \end{cases} \quad (12)$$

Враховуючи введені змінні, ЦФ має такий остаточний вигляд:

$$z = \sum_{i=1}^{n} t_i(L_a, \{I_c\}, \{M_c\}), \quad (13)$$

де n – кількість таблиць запиту, a – область, що містить таблицю i, $\{I_c\}$ та $\{M_c\}$ – ознаки індексування та матеріалізації полів c таблиці i.

Така ЦФ є такою, що використовується в генетичних алгоритмах при f=z(i), $i = < L_a, I_{c0}, I_{c0+1}, ..., I_c, M_{c0}, M_{c0+1}, ..., M_c >$ і дає змогу в однакових інших умовах оточнення (апаратне, програмне забезпечення) отримати часову оцінку отримання даних з таблиць запиту певної популяції. А оскільки сховище даних буде проектоване під всі запити з цієї популяції, то вони будуть виконуватися більш оптимально, ніж при універсальному проектуванні даних (під всі запити).

Оскільки змінними моделі є ознаки розміщення таблиці даних, наявності індекса та матеріалізованості стовпця, область всіх можливих рішень можна записати як $\{0,1\}^{a+2*c}$ тобто цю область складають всі набори $< L_1, L_2, ..., L_n, I_1, I_2, ..., I_n, M_1, M_2, ..., M_n >$, де $L_i, I_i, M_i \in \{0,1\}$.

На ОДР накладаються такі обмеження:

– забороняється розміщення таблиць, що містять зовнішні ключі та нечислові поля, в багатомірних базах даних, оскільки такі таблиці фактів не підтримуються;

– забороняється створення індексів в багатовимірній базі даних, що випливає з природи багатовимірних баз даних, оскільки ця операція найчастіше інкапсульована в двигуні бази даних і виконується автоматично;

– забороняється включення зовнішніх ключів в поля, що входять до матеріалізованих представлень таблиць, які містяться в багатовимірній базі даних, оскільки це робить неможливим коректне виконання запитів до багатовимірної з бази даних.

Для задання генетичного алгоритму треба визначити хромосому, процедуру селекції, мутації та схрещування.

Виходячи з змінних математичної моделі, хромосомою є набір $< L_1, L_2, ..., L_n >$, де L_i – ознака розміщення таблиці даних (0 – в реляційній БД, 1 – в багатовимірній БД).

У цій роботі використовуються такі характеристики алгоритму :

- Імовірність (частота) мутації = 0,05.
- Імовірність (частота) схрещування = 0,8.
- Селекція за методом колеса рулетки.
- Одноточкове схрещування.

Частоти підібрано так, що вони відповідають реальним біологічним генетичним процесам, отже, є оптимальними для вибору алгоритму. Селекція за методом рулетки вибрано тому, що вона є найбільш ефективною. Одноточкове схрещування вибране тому, що в комбінації з мутацією доволі чітко відтворює реальні біологічні процеси.

Наведемо сам алгоритм:

Алгоритм проектування гібридного сховища даних.

Вхідні дані : поточна структура сховища даних, запит популяції.

Вихідні дані : оптимальна структура сховища даних.

1. Перевірити належність останнього виконаного запиту до поточної популяції запитів. Популяцією запитів вважається послідовність запитів однакового типу (точкового чи інтервального) до однієї і тієї самої області сховища даних.

2. Якщо запит належить до нової популяції, перейти на крок 2,інакше перейти на крок 5.

3. Сформулювати початкову хромосому як поточний стан схеми сховища даних і додати її до поточного покоління. Також додати до поточного покоління всіх особин, у яких $\forall c\, I_c = 0$ та $\forall c\, M_c = 0$, тобто ознаки матеріалізації та індексування покласти дорівнюють нулю.

4. Поки структура отриманого сховища не є оптимальною, робити наступне.

4.1. Вибрати батьків i1, i2 з популяції Ik за допомогою оператора селекції

4.2. Побудувати i' по i1, i2 за допомогою оператора схрещування

4.3. Модифікувати i'за допомогою оператора мутації

4.4. Якщо сформована хромосома не є допустимою, змінити її, задавши гени таким чином, що вона буде допустимою.

4.5. Перебудувати сховище даних відповідно до генів хромосоми i'.

4.6. Виконати запит, отримати нове значення ЦФ.

4.7. Якщо f*<f(i'), то f* := f(i').

4.8. Якщо $f_k^* - f_{k+1}^* < \varepsilon$, перейти на крок 5, інакше перейти на п.4.8

4.9. Покласти k = k+1 і перейти на крок 2.1.

5. Кінець алгоритму.і'представляє собою оптимальну структуру сховища даних для областей, таблиці яких беруть участь у запиті та типу запиту (інтервального чи точкового).

Висновки

У статті проаналізовано існуючі підходи до оптимізації сховищ даних. Розглянуті теоретичне рішення дають змогу оптимізувати сховище даних за допомогою індексів,матеріалізованих представлень та фрагментації. Однак ці методи не використовуються в комплексі, а методи не враховують поєднання реляційної та багатовимірної бази даних.

Тому запропоновано генетичний алгоритм, який би поєднував індекси і матеріалізовані представлення для оптимізації сховища. У подальших дослідженнях слід розглянути питання оптимізації роботи алгоритму і дослідити його роботу у різних програмних та апаратних середовищах.

1. Wen-Yang Lin. A Genetic Selection Algorithm for OLAP Data Cube, / Wen-Yang Lin,I-Chung Kuo – Knowledge and information systems 2004, vol. 6 2.Chuan Zhang.An Evolutionary Approach to Materialized Views Selection in a Data Warehouse Environment. Systems/.Chuan Zhang, Xin Yao, Jian Yang – Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on Volume 31, Issue 3, Aug 2001 3.Ladjel Bellatreche. An Evolutionary Approach to Schema Partitioning Selection in a Data Warehouse / Ladjel Bellatreche,Kamel Boukhalfa – Lecture notes in computer science, Congrès DaWaK 2005 : data warehousing and knowledge discovery: International conference on data warehousing and knowledge discovery No7, Copenhagen, DANEMARK 2005 , vol. 3589 4.J.-T. Horng. Applying evolutionary algorithms to materialized view selection in a data warehouse/ J.-T. Horng, Y.-J. Chang, B.-J. Liu – Soft Computing – A Fusion of Foundations, Methodologies and Applications, Volume 7, Number 8 / August, 2003 5. Chuan Zhang. Evolving Materialized views in a Data Warehouse / Chuan Zhang, Xin

Yao, Jian Yang – Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on Volume 2 6. Goran Velinov. Framework for Generalization and Improvement of Relational Data / Goran Velinov, Danilo Gligoroski, Margita Kon Popovska – IJCSNS International Journal of Computer Science and Network Security, VOL.8 No.3, March 2008 7.Michael Lawrence. Multiobjective genetic algorithms for materialized view selection in OLAP data/ Michael Lawrence – Proceedings of the 8th annual conference on Genetic and evolutionary computation,2006 8. Шаховська Н.Б. Сховища та простори даних: Монографія. / Шаховська Н.Б, Пасічник В.В. – Львів: Видавництво Національного університету «Львівська політехніка», 2009. – 244 с. 9. Яцишин Ю.В. База знань для формування переліку бюджетних програм інформаційно-аналітичної системи управління державними фінансами, III Міжнародна науково-технічна конференція "Комп'ютерні науки та інформаційні технології" (CSIT-2008) / Яцишин Ю.В, Яцишин А.Ю. – Львів, 2008. 12. 10. Яцишин А.Ю. Проектування гібридного сховища даних для роботи з державним бюджетом України. II Всеукраїнська науково-практична конференція «Інформаційні технології та автоматизація – 2009» / Яцишин А.Ю. – Одеса, 2009.