УДК 681.3

ПІДХІД ДО ВДОСКОНАЛЕННЯ ОРГАНІЗАЦІЇ ДАНИХ В АВТОМАТИЗОВАНИХ СИСТЕМАХ

І.В.Федоров
Інститут програмних систем НАНУ
e-mail: iffivt@rambler.ru

Однією з найбільш актуальних проблем, які виникають при створенні та супроводженні автоматизованих систем (АС) є проблема забезпечення стійкості цих систем до змін навколишнього середовища та змін вимог їх користувача, тобто їх життєздатності [1].

Ефективність методів забезпечення життєздатності автоматизованих систем багато в чому залежить від організації даних в АС. Вдосконалення методів організації даних дозволить підвищити адаптивність АС при постійній зміні вимог до неї, а також досягти щонайменших втрат при створенні та супроводженні АС.

Автоматизована система (АС) – система, що реалізує інформаційну технологію у сфері управління за спільної роботи управлінського персоналу і комплексу технічних засобів. Вона призначена для автоматизованого збирання, реєстрації, збереження, пошуку, оброблення та видачі інформації за запитами користувачів (управлінського персоналу). Це відбувається на основі використання економіко – математичних методів, моделей, ЕОМ і засобів комунікації. В даний час багато підприємств відчувають необхідність поліпшення своєї автоматизованої системи. У першу чергу це зв'язано з незадоволеністю керівників якістю одержуваної ними інформації і швидкістю її одержання. Тому перед керівництвом постають наступні питання: "Яким же чином організувати дані у залежності від цілей бізнесу, різних етапів розвитку компанії і поточного стану її автоматизації? Як вдосконалити організацію даних таким чином, щоб врахувати найбільш актуальні напрямки розвитку автоматизованої системи?

Особливо потребують вдосконалення даних системи з часовими обмеженнями - темпоральні системи, побудовані на основі темпоральних систем керування базами даних (СКБД) [2]. Однією з головних проблем в таких системах є форма збереження даних з прив'язкою до часу. Основна теза темпоральних систем полягає в тому, що для будь-якого елемента даних, створеного в момент часу t1 і знищеного в момент часу t2, у її БД зберігаються, і є доступними для користувачів, усі його стани в часовому інтервалі [t1, t2].

Дослідження і побудова прототипів темпоральних СКБД звичайно виконуються на основі деякої реляційної СКБД. Темпоральна СКБД — це надбудова над реляційною системою.

Доклад присвячено опису підходу до проектування нових структур даних, що задовольняють високим вимогам, які пред'являються до сучасних застосувань баз даних при створенні автоматизованих систем.

Задачі проектування нових структур даних. Однією із задач створення нових структур даних є організація більш ефективного індексування елементів даних.

З підвищенням швидкості комп'ютерної обробки і ростом обсягів доступної пам'яті змінилися вимоги до організації індексів. Якщо в СКБД перших поколінь головною характеристикою була швидкість доступу до даних, то сьогодні на перший план вийшла швидкість реорганізації індексу через суттєву мінливість БД і неминуче часту перебудову індексу на велику глибину. Тому однією з вимог до індексів є наявність потенційно великого числа вільних місць, що дозволить досягнути рідкої потреби у перебудові індексу.

Типові деревовидні структури даних для організації індексів мають досить суттєві недоліки. Основний недолік - незбалансованість роботи структур після видалення і

вставки деяких елементів [3]. Оскільки під час пошуку при кожному проходженні елементів структури виконується звертання до диску, загальна тривалість пошуку в незбалансованій деревовидній структурі може виявитися зовсім непередбаченою. Тому однією з вимог до створення індексних структур даних є їх оптимізація для застосувань, в яких швидкість оновлення даних, а також швидкість роботи функції пошуку елемента даних є характеристиками, які визначають якість роботи застосування. В просторово-часових базах даних індексування має складну структуру. Пов'язано це зі специфічністю зберігаємих в цих базах даних характеристик елементів даних.

Застосування такого роду потребують засобу керування великою кількістю інформації, в тому числі про реальне місцерозташування елементів даних, а також відображаючому переміщення цих елементів на протязі часу. Зміна властивостей елементів даних є неперервним. Звичайно, це повинно бути відображено в засобі управління даними застосування. Постійне переміщення елементів даних робить недопустимим використання традиційних СКБД в задачах такого характеру, призначених в основному для робіт зі статичними даними. Зміна характеристик елемента даних може бути представлена в базі даних за допомогою параметрів сутності, які змінюються і функції місцерозташування, яка визначає положення елемента даних в просторі в будь-який заданий час. В таких базах даних доцільно використання індексів, заснованих на деревоподібних структурах даних, таких як B-дерево.

Проектування нової структури даних. При проектуванні індексу у вигляді B-дерева варто врахувати, що ступінь B-дерева значно впливає на час пошуку, корисне застосування кожної вершини і продуктивність модифікації даних. Загалом конфігурація B-дерева і правила балансу пристосовано до інтенсивного відновлення даних, тобто до мінливості даних у БД великого розміру [4].

Для досягнення якнайменших втрат при реорганізації індексу та для більш швидкого доступу до даних можна запропонувати наступний спосіб розщеплення та об'єднання вузлів дерева.

Алгоритм забезпечує можливість зберігання в кожному вузлі n ключів, Кількість ключів в корні може бути 3n. Попередньо виконавши перевірку на порожність нащадка під час вставки, спускаємось до нього. Якщо нащадок пустий, ключі, які знаходяться в ньому і двох суміжних до нього вузлах, об'єднуються і перерозподіляються. Якщо два суміжних вузла також заповнені, то додається вузол. Таким чином, ми отримуємо вже чотири вузла, кожний з яких на ¾ заповнений. Під час видалення попередньо виконуємо перевірку на наповненість нащадка наполовину, далі опускаємося до нього. Якщо нащадок заповнений наполовину, ключі нащадка і двох суміжних вузлів об'єднуються і перерозподіляються. Якщо два суміжних вузла самі заповнені наполовину, вони зливаються у два вузли, кожний з яких повний на ¾ . Ми, таким образом, опиняємося посередині між наповненістю наполовину і повною наповненістю, що дозволяє нам сподіватися на однакове число вставок і видалень.

Якщо під час вставки виявиться, що корінь повний, ми розподіляємо ключі по чотирьом новим вузлам, кожний з яких заповнений на ¾ . Ці дії призведуть до збільшення висоти дерева. Під час видалення алгоритм досліджує нащадків. Якщо є тільки три нащадки і вони заповнені наполовину, переносимо їх зміст в корінь, це призведе до зменшення висоти дерева.

Отже, коротше кажучи, ми збираємо три вузла, а потім розщеплюємо їх. При вставці, коли нам потрібен додатковий вузол, ми розщеплюємо на чотири вузла. При видаленні, коли вузол треба видалити, ми розщеплюємо на два вузли. Цей спосіб зберігає збалансованість дерева. Максимальна висота росте не швидше, ніж логарифм числа елементів.

Висновки. Структуру типу даних, що базується на представленні B+ -дерева,

можливо вдосконалити, використавши відмінний від класичного спосіб розщеплення та об'єднання вузлів дерева. В реалізації структури даних потрібно забезпечити необхідність множинних посилань на одні й ті ж дані, дозволити здубльовані ключі. В межах одного індексу всі ключі повинні бути однієї довжини. Для утримання вузлів в пам'яті до тих пір, доки пам'яті вистачає, потрібно реалізувати гнучку схему буферизації. Якщо очікується доступ до чого-небудь упорядкованого потрібно зменшити кількість звертань до диску. Але при цьому треба обов'язково дотримуватися стандартних властивостей B+-дерева. Необхідно врахувати, що операція доступу до диску означає посекторний обмін; типовий розмір сектора - 256 байтів. Потрібно прирівняти розмір вузла с розміром сектора і згрупувати разом декілька ключів в кожному вузлі, щоб зменшити кількість операцій обміну. Всі ключі мають зберігатися в листах, там також зберігається й інформаційна частина вузла. У внутрішніх вузлах зберігаються копії ключів - вони допомагають шукати потрібний лист. Лівий показник веде до ключів, які менше заданого значення, правий - ключам, які більші або рівні. Також необхідно врахувати особливості роботи с батьківськими вузлами під час операцій вставки та видалення. Коли модифікується перший ключ в листі, дерево проходить від листа до кореня. Останній з показників (більше або рівно), знайдений при спуску по дереву, і є тим, що потребує модифікації, щоб відобразити нове значення ключа. Оскільки всі ключі повторюються в листах, ми можемо зв'язувати їх для послідовного доступу.

Продовження дослідження в цьому напрямку дозволить покращити властивості дерева, удосконалити алгоритми сортування елементів даних у разі вставки в незаповнений вузол, провести аналіз взаєморозташування елементів даних на одному рівні дерева, вести спеціальну метрику відстані між елементами даних. Це дасть змогу будувати індексну структуру більш оптимальну щодо пошуку даних.

Список літератури

1. Ігнатенко П.П. Проблеми забезпечення життєздатності програмних систем та підходи до їх вирішення// Пробл. програм.- 2002. - № 3-4. – с. 58 – 73.

2. Дейт К. Дж. Введение в системы баз данных. – М.: Вильямс, 2000, - 848с.

3. Томас Кормен, Чарльз Лейзерсон, Рональд Ривест. Алгоритмы: построение и анализ. - М.: МЦНМО, 1990, - 960 с.

4. А. Ахо, Д. Хопкрофт, Д. Ульман Построение и анализ вычислительных алгоритмов. - М.: Наука, 1989, - 360с.