УДК 004.023:004.421.2 **doi: 10.26906/SUNZ.2018.1.105**

І.В. Ільїна, Д.С. Кадубенко, Д.О. Ільїн

Харківський національний університет Повітряних Сил імені Івана Кожедуба, Харків

ПІДВИЩЕННЯ ПРОДУКТИВНОСТІ СИСТЕМ УПРАВЛІННЯ БАЗАМИ ДАНИХ

Аналізуються підходи мінімізації часу для отримання необхідної інформації з бази даних, а також критерії та параметри запитів, яких необхідно дотримуватися у процесі розробки. Розглядаються питання створення та оптимізації запитів для підвищення продуктивності, що дозволяють зробити функціонування бази даних більш ефективною. Обґрунтовано необхідності використання даних підходів в різних структурах.

Ключові слова: *база даних, мінімізації часу, СУБД, Oracle, алгоритм пошуку , оптимізація запиту.*

Вступ

Існує безліч підходів до мінімізації часу отримання необхідної інформації з бази даних, які базуються на оптимізації запитів: логічної і семантичної. Ці підходи засновані на внутрішнє перетворення запитів, зміні послідовності виконання операцій реляційної алгебри. Мало досліджена можливість оптимізації запитів в базах даних, що містять дубльованих інформацію. Зміст дублікатів інформації дозволяє будувати різні запити для отримання однієї і тієї ж інформації. Одним з способів контрольованого зберігання і застосування інформації, що дублюється, є використання матеріалізованих уявлень. Матеріалізовані уявлення, що вперше з'явилися в СУБД Oracle, є таблицями бази даних, що зберігають результати виконання запитів. Цілісність даних в цих таблицях підтримується періодичною синхронізацією або використанням інструментів тригерів. Таким чином, для оптимізації запитів з альтернативними маршрутами їх виконання попереду стоїть завдання дослідження методів їх вирішення.

На даний час в різних сферах людської діяльності важливу роль відіграє оперативність прийняття рішень і швидкість отримання інформації, що впливає на ці рішення. Отримати оперативно інформацію з бази даних часто не представляється можливим через наявність великого обсягу даних і складності запитів в інформаційної системи, яка використовується. Тому створення і застосування методів, що дозволяють підвищувати ефективність виконання запитів, є областю досліджень, яка активно розвивається. Під підвищенням ефективності виконання запитів - оптимізацією запитів – розуміється скорочення часу їх виконання.

Головною метою роботи є пошук оптимального плану виконання запитів. Для цього необхідно:

провести аналіз існуючих підходів до оптимізації запитів в базах даних;

розробити і дослідити спеціальні методи побудови альтернативних маршрутів виконання запитів шляхом зміни структури бази даних за рахунок внесення контрольованої надмірності.

Таким чином, для оптимізації запитів з альтернативними маршрутами їх виконання попереду стоїть завдання дослідження методів їх вирішення.

Основна частина

Оптимізація процедур обчислення запитів є, взагалі, обчислювальне важкою, заважає відсутність точної статистичної інформації про базу даних.

Запит - це мовне вираження, яке описує дані, що підлягають вибірці з бази даних. Запити використовуються в декількох середовищах. Найбільш очевидний додаток, в який надходять безпосередні запити кінцевого користувача, що потребує інформації про структуру або вміст бази даних. Якщо потреби користувачів обмежені набором стандартних запитів, вони можуть оптимізуватися вручну шляхом програмування відповідних процедур пошуку і обмеження для користувача. Однак, якщо потрібно задавати непередбачені запити з використанням мови запитів загального призначення, стає необхідною система автоматичної оптимізації запитів. Друге застосування запитів відбувається в транзакціях, які змінюють збережені дані на основі їх поточних значень. Нарешті, вирази, подібні запитам, можуть використовуватися всередині СУБД, наприклад, для перевірки прав доступу [1], підтримки обмежень цілісності [2] і коректної синхронізації паралельного доступу [3].

Економічний принцип вимагає, щоб процедури оптимізації намагалися або максимізувати пропускну здатність при заданому числі ресурсів, або мінімізувати споживання ресурсів при даній пропускної здатності. Оптимізація запитів направлена на мінімізацію часу відгуку для заданого запиту і змішування типів запитів. Ця загальна мета допускає ряд різних операційних цільових функцій. Час відгуку є розумною метою тільки при припущенні, що час користувача є найбільш важливим критичним ресурсом. В іншому випадку можна прагнути безпосередньої мінімізації вартості споживання технічних ресурсів. Обидві мети є у великій мірі взаємно додатковими; при виникання конфліктів цілей зазвичай вирішуються шляхом призначення обмежень на доступні технічні ресурси (наприклад, на розмір буферного простору).

Щоб допустити справедливе порівняння ефективності, функціональні можливості порівнюваних систем виконання запитів повинні бути подібними. Вимога "реляційної повноти", яка запропонована Кодом [4], стала квазістандартом. Загальна вартість, що підлягає мінімізації, складається з таких компонентів:

Вартість комунікацій: Вартість передачі даних з місця, в якому вони зберігаються, в місця, де виконуються обчислення і представляються результати. Ця вартість складається з вартості комунікаційного каналу, яка зазвичай пов'язана з часом, протягом якого канал відкритий, і вартістю затримок в обробці, викликаних передачею. Останній компонент, більш важливий для оптимізації запитів, часто покладається лінійною функцією від обсягу переданих даних.

Вартість доступу до вторинної пам'яті: вартість (або час) завантаження сторінок даних з вторинної пам'яті в основну пам'ять. На цю вартість впливає обирані данні (головним чином, розмір проміжних результатів), кластеризація даних на фізичних сторінках, розмір доступного буферного простору і швидкість пристроїв, які використовуються

Вартість зберігання: вартість заняття вторинної пам'яті і буферів основний пам'яті. Вартість зберігання доречна лише в тому випадку, коли пам'ять стає вузьким місцем системи, і розмір необхідної пам'яті може змінюватися від запиту до запиту.

Вартість обчислень: вартість (або час) використання центрального процесора (ЦП).

На структуру алгоритмів оптимізації запитів впливають співвідношення між цими компонентами вартості. У територіально розподіленої СУБД з відносно повільними комунікаційними каналами переважає вартість комунікаційних затримок, а інші чинники істотні для локальної оптимізація. У централізованих системах домінує час доступу до вторинної пам'яті, хоча для складних запитів досить високою може бути і вартість ЦП [5]. У локально розподілених СУБД всі фактори мають близькі ваги, що призводить до складних функцій і процедур оптимізації.

Дослідження присвячено централізованим базам даних, вартість комунікацій не береться до уваги, тому що в таких системах комунікаційні вимоги не залежать від стратегії виконання запитів. Для оптимізації одиночних запитів вартість зберігання також вважається не першочергово важливою. Ці витрати враховуються тільки при одночасній оптимізації кількох запитів. Залишаються вартість доступу до вторинної пам'яті і вартість використання ЦП (вимірюється числом порівнянь, які потрібно зробити). В основі більшості методів, розроблених для скорочення цієї вартості, лежить ряд загальних ідей: уникати дублювання зусиль; використовувати стандартизовані компоненти; заглядати вперед, щоб уникати зайвих операцій; вибирати найбільш дешеві способи виконання елементарних операцій; вибудовувати їх послідовність оптимальним чином.

Дослідження оптимізації запитів, які представлені в літературі, можна розбити на два класи, які можна охарактеризувати як висхідний і спадний. Дослідники знаходять спільну проблему оптимізації запитів дуже складною. Теоретичні роботи почалися з висхідного підходу, вивчення особливих випадків, таких як оптимальна реалізація важливих операцій і стратегії обчислення для деяких простих підкласів запитів. Надалі дослідники намагалися зібрати з цих початкових результатів більші «будівельні блоки».

Потреба в працюючих системах ініціювала розробку повномасштабних процедур обчислення запитів, що вплинуло на спільність рішень і змусило займатися оптимізацією запитів в евристичної манері [6]. Оскільки часто це не дозволяло досягти конкурентної ефективності систем, сучасною тенденцією є спадний підхід, який забезпечує можливість включення в загальні процедури більшого знання про можливості оптимізації в окремих випадках. У той же час, самі загальні алгоритми посилюються комбінаторними процедурами мінімізації вартості для вибору між стратегіями. Дотримуючись спадного підходу, який застосовує загальну процедуру обчислення, служить каркасом для конкретних методів, розроблених при дослідженні оптимізації запитів.

Крок 1. Знайти внутрішнє уявлення запитів, в яке можуть легко відображатися запити користувачів, що залишає системі всі необхідні ступені свободи для оптимізації виконання запитів.

Крок 2. Застосувати логічні перетворення до подання запиту, які по перше, стандартизують запит, по друге, спрощують запит, щоб уникнути дублювання зусиль, трете покращують запит для спрощення його виконання і створення можливості застосування процедур окремих випадків.

Крок 3. Показати перетворений запитів в можливу послідовність елементарних операцій, для яких відома хороша реалізація і відповідні оцінки вартості. В результаті цього кроку з'являється набір можливих "планів доступу".

Крок 4. Обчислити загальну вартість кожного плану доступу, вибрати найбільш дешевий план і виконати його.

Перші два кроки цієї процедури є в великій мірі незалежними і тому часто можуть бути виконані під час компіляції. Якість кроків 3 і 4 сильно залежить від знання значень в базі даних. Наслідки від залежності від даних є двоякими. По-перше, якщо база даних мінлива, то кроки 3 і 4 можуть бути виконані тільки під час виконання. Це означає, що можливий виграш в ефективності повинен співвідноситися з вартістю самої оптимізації. По-друге, в метабазі даних (наприклад, розширюваному довіднику даних) повинна підтримуватися загальна інформація про структуру бази даних, так само як і статистична інформація про вміст бази даних. Як і в багатьох схожих дослідженнях (наприклад, при управлінні запасами) вартість отримання і підтримки цієї додаткової інформації повинна зіставлятися з її якістю.

Сучасне поняття інформаційної системи має кілька тлумачень. Але всі вони схожі в одному: інфор-

маційна система повинна включати в себе базу даних (БД), систему управління базами даних (СУБД) і клієнтську програму [7]. Інформаційні системи є одним з головних об'єктів, які використовуються при прийнятті, супроводі та контролі управлінських рішень, тому вони використовуються практично на кожному підприємстві. Розрізняють такі інформаційні системи: локальні та інтегровані. Локальні виконують лише якісь окремі конкретні завдання, в той час як інтегровані задовольняють потреби всіх служб, підрозділів та співробітників [8]. Для інтегрованих інформаційних систем характерна розподілена архітектура «клієнт-сервер [5]. Така архітектура має на увазі, що її компоненти розподілені по різних комп'ютерів. Так на одному комп'ютері розташовані БД і СУБД, а на інших клієнтські додатки. Іноді в інформаційних системах крім сервера баз даних є сервери клієнтських додатків. Тоді виникають проміжні ланки, і клієнтські програми взаємодіють не з базою даних, а з сервером додатків, а той вже в свою чергу з СУБД [6].

Інтегровані корпоративні системи містять єдину базу даних, тому дозволяють користувачам в будь-який момент часу отримувати актуальну інформацію. Використання підприємствами інтегрованих інформаційних систем у своїй діяльності є найважливішою конкурентною перевагою, так як оперативне отримання необхідної інформації сильно впливає на якість прийнятих управлінських рішень [5]. Швидкий доступ до необхідної користувачам інформації забезпечується всіма компонентами інформаційних систем, але найбільш важливим з них є бази даних.

База даних – організована відповідно до певних правил і підтримувана в пам'яті комп'ютера сукупність даних, що характеризує актуальний стан деякої предметної області, та використовується для задоволення інформаційних потреб користувачів [7]. До відмітних особливостей баз даних відносяться [9]: обробка та зберігання баз даних повинно здійснюватися в обчислювальної системи; структурованість даних, що зберігаються; опис її логічної структури.

Сьогодні найбільш поширеною моделлю організації даних є реляційна [8]. Реляційна модель побудована на понятті відносини і відповідає таким аспектам: дані є набором відносин; відносини відповідають рівням цілісності (рівні домену, відносини і бази даних); підтримуються оператори обробки даних. Реляційна модель орієнтована на організацію даних у вигляді двовимірних таблиць [7]. Кожна реляційна таблиця – двовимірний масив і має такі властивості: кожен її елемент - один елемент даних; всі осередки в стовпці таблиці однорідні, тобто всі елементи в стовпці мають однаковий тип; кожен стовпець має унікальне ім'я; однакові рядки відсутні; порядок проходження рядків і стовпців довільний.

Все це робить незалежним клієнтську програму від структури бази даних, тобто зміна структури бази даних не буде приводити до необхідності модифікації клієнтського додатку. Це робить реляційну модель даних досить гнучкою. Крім того наявність суворого математичного апарату для роботи з даними – реляційної алгебри, дозволяє точно і однозначно будувати запити для отримання необхідної інформації [1].

Об'єктно-реляційні бази даних реалізують такі додаткові можливості [6]: об'єктну інфраструктуру, що дозволяє користувачам визначати нові типи даних, функції і правила безпосередньо в самих базах даних; реляційні розширювачі над об'єктної інфраструктурою підтримують спеціалізовані додатки.

Реляційна модель має під собою потужний фундамент у вигляді суворого математичного апарату - реляційної алгебри. Сучасне розвиток інформаційних систем зажадало введення нових можливостей, зокрема, кошти визначення нових типів збережених даних і операцій над ними, а також зберігання правил, що дозволяють додаткам спільно використовувати не тільки дані, але і їх поведінку.

У зв'язку з описаними вище аспектами з'явилися розширення реляційних систем баз даних у вигляді об'єктно-реляційної моделі [2]. Оптимізація запитів в базах даних дозволяє значно скоротити час їх виконання, а значить, істотно впливає на швидкість прийняття управлінських рішень, заснованих на аналізі наявної інформації. Існує безліч підходів до мінімізації часу отримання інформації з бази даних, заснованих на оптимізації запитів. Під оптимізацією запиту в реляційних базах даних розуміють вибір способу виконання запитів, коли на основі синтаксичного і семантичного перетворення будується план його виконання, який при існуючої керуючої структурі дає мінімальний час його виконання. Процедуру оптимізації запитів можна розбити на кілька етапів [6].

На першому етапі відбувається синтаксичний і лексичний розбір запиту. Результатом етапу є його внутрішнє подання, яке містить інформацію про об'єкти бази даних, описаних в самому запиті.

Другий етап полягає в логічній оптимізації внутрішнього подання запиту, отриманого на першому етапі. На цьому етапі застосовуються перетворення, що зберігають семантичну цілісність, і призводять його до деякої стандартної формі.

На третьому етапі відбувається вибір набору альтернативних планів виконання даного запиту на основі його внутрішнього уявлення, отриманого на другому етапі.

На останньому етапі відбувається реальне виконання запиту в відповідно до вибраного на попередньому етапі планом.

Логічна оптимізація запитів має на увазі еквівалентні перетворення подання запиту, що використовують правила, які закладені в оптимізаторі. Ці правила, що призводять до оптимізації запиту, є досить умовними, оскільки залежать від загальної організації оптимізатора, яка в свою чергу багато в чому залежить від того, як буде виконуватися етап вибору альтернативних планів реалізації даного запиту [2].

Логічні перетворення є одними з основних перетворень, що призводять внутрішнє уявлення запиту до якогось стандартного, канонічного вигляду. На-

приклад, уявлення предикатів, які описують умови вибірки в даному запиті. Під предикатами, які задають умови вибірки, розуміють звичайні операції порівняння, тобто предикат є порівняння двох арифметичних виразів, наприклад, імена полів відносин і константи [3]. Тобто, в цьому випадку логічна оптимізація зводиться до приведення предикатів загального вигляду до наступного поданням "арифметичний вираз-оператор-константний арифметичний вираз". При цьому ліву і праву частину теж можна привести до певного стандартного вигляду. Це може дозволити знайти загальні арифметичні вирази в різних предиката запиту, що може привести до зменшення всього часу виконання запиту. Під час операцій по приведенню предикатів до канонічного вигляду логічним видається обчислювати вирази констант всюди, де це можливо, а також йти від операції заперечення.

Природним наступним етапом логічних перетворень є приведення умов вибірки до однієї з канонічних форм. В якості таких форм найбільш часто використовуються кон'юнктива і діз'юнктівная нормальна форми. Кон'юнктива нормальна форма являє собою кон'юнкцію предикатів, які в свою чергу є диз'юнкція простих предикатів. Аналогічно, діз'юнктівная нормальна форма являє собою диз'юнкцію предикатів, які в свою чергу є кон'юнкція простих предикатів. Приведення предикатів до конкретної канонічної формі багато в чому залежить від типів використовуваних оптимізаторів [5].

Перед приведенням умов вибірки до канонічного вислову логічним етапом є спрощення самих умов. Одним з яскравих прикладів таких спрощень є заміна кон'юнкції, предикати, які взаємно суперечать, на значення FALSE. Дійсно, взаємне виконання умов, що суперечать, статися не може, чого якраз і вимагає операція кон'юнкції. Ще одним логічним перетворенням є позбавлення від ідентичних предикатів, які можуть виникнути при первісному невдалому конструюванні запитів, або з'явитися у час приведення предикатів до канонічного вигляду.

Оптимізатор запитів повинен вибирати послідовність операцій виконання, що приводить до найбільш мінімальних витрат фізичних ресурсів, тобто повинен забезпечувати ефективний фізичний план. Для того, щоб фізичний план виконання запитів був ефективним, крім використання алгебраїчних законів необхідно групувати однакові асоціативно-комутативні оператори, в тому числі оператори з'єднання. Використання логічних перетворень запитів не залежить від конкретного виду бази даних. Але в реальних реляційних базах даних є деяка сукупність правил і знань, що використовуються для різних цілей, в тому числі і для збереження цілісності бази даних.

Системи управління базами даних використовують ці обмеження для збереження цілісності бази даних. Запити до бази даних повинні формулюватися, використовуючи і спираючись на знання підтримки цілісності бази даних, що зберігаються [4]. Сучасний розвиток обчислювальної техніки дозволяє не сильно турбуватися про обсяги інформації, що зберігається. Набагато більш критичною є завдання швидкого пошуку і вибору з усього обсягу необхідної інформації. Крім того, сучасні СУБД мають достатньо великий і нескладний інструментарій для підтримки цілісності інформації в базах даних. Тому зберігання дубльованої інформації вже можна не відносити до недоліків.

Другим із зазначених недоліків є негнучкість матеріалізованих уявлень. Можливість їх використання обмежується вбудованими в клієнтську програму запитів. Адже навіть при невеликому корегуванні запиту матеріалізоване уявлення не даватиме необхідної в запиті інформації. Можливий шлях подолання цього недоліку полягає в наступному: при зміні запиту матеріалізоване уявлення може залишитися його частиною. Так як матеріалізоване уявлення вже є результатом виконання ряду операцій, то заміна цих операцій в запиті може значно прискорити час виконання. Актуальною стає розробка методик прискорення часу виконання запитів, які враховують денормалізація. Семантичні методи оптимізації, засновані на внесення та зберіганні дубльованої інформації в базі даних. Зазвичай одним із завдань, що вирішуються при проектуванні баз даних, є нормалізація відносин. Нормалізація полягає в видаленні надлишкової інформації і створенні однозначності в збережених даних. Нормалізація - це приведення відносин до нормальної формі [6].

У зв'язку з відсутністю дубльованої інформації та однозначності її визначення в нормалізованих базах даних не можна побудувати різні варіанти запитів, що дають семантично рівнозначні результати. При сучасному технологічному розвитку нормалізованих баз (відсутність дубльованої інформації зменшує обсяг пам'яті, необхідної для її зберігання, і не може викликати невідповідність в дублікатах, що зберігаються в різних місцях; однозначність визначення виключає можливість помилок у виданих даних [4]) стають і їх недоліками. У нормалізованих базах для отримання необхідної інформації необхідно створювати запити, що містять велику кількість з'єднаних таблиць. Розміри багатьох сучасних інформаційних систем складають терабайти, а кількість записів наближається до сотнею мільйонів. Тому операції з'єднання таблиць можуть виконуватися досить довго і гальмувати роботу всієї системи. У цьому випадку кращим варіантом було б збереження додаткової дубльованої інформації (необхідне збільшення обсягів пам'яті не є критичним), яка б дозволяла значно скоротити кількість операцій з'єднання таблиць в запитах і зменшити час їх виконання.

Денормалізацію не можна вважати просто операцією зворотною до нормалізації [8]. По-перше, на відміну від нормальних не існує денормальних форм. По-друге, немає строгих математично обґрунтованих правил виконання денормалізації. Проведення денормалізації – процес неформалізований та творчий. Якість денормалізації характеризується наступними параметрами:

1) сума часів на виконання запитів до бази і на підтримку її цілісності за певний проміжок часу (годину, добу, тиждень і т.д.). Денормализация вважається обґрунтованою, якщо ця сума менше, ніж час виконання запитів до бази за такий же період часу до денормалізації;

2) інформація, що видається, відповідає актуальному стану бази даних. Вона повинна бути однозначною в незалежності від того, з яких джерел зберігання дубльована інформація отримана. Багато в чому якість денормалізації буде залежати від досвіду і знань адміністратора баз даних, котрий її проводить.

Використання дубльованої інформації в інформаційних системах іноді дозволяє не тільки значно скоротити час виконання запитів, але і спростити процедуру написання запитів. Така можливість реалізується в сховищах даних сучасних ERP-систем. В них реалізована аналітична система обробки даних OLAP [5]. Вона заснована на понятті OLAP кубів. Найбільш поширеними схемами сховищ даних є схеми типу «Зірка» або «сніжинка». В основі цих схем лежать поняття таблиць фактів і таблиць вимірів. Таблиці фактів зазвичай містять всю повну інформацію про досліджувані об'єкти. Таблиці вимірювань пов'язані з таблицею фактів і задають з неї зрізи.

При будь-якому вигляді денормалізації змінюється структура бази даних. Зміна структури бази даних шляхом денормалізації виявляється виправданим в тих випадках, якщо час виконання запитів в нереструктурізованій базі даних буде більше ніж час виконання запитів в реструктуризованій базі і час виконання додаткової обробки, необхідної для підтримки цілісності даних [6].

Висновки

Розглянуто загальне поняття сучасних інформаційних систем. Виділена найпопулярніша модель зберігання даних - реляційна. Також розглянута актуальність оптимізації запитів в реляційних базах даних. Виділені основні види оптимізації: логічна і семантична. Систематизовані підходи до денормалізації структури бази даних. Розглянута денормалізація бази даних з використанням матеріалізованих уявлень, а також критерії ефективності використання матеріалізованих уявлень.

Розглянуті підходи до оптимізації запитів засновані на зміні їх внутрішньої структури, тобто визначенні послідовності виконуваних операцій - вибору оптимального плану виконання запиту.

Денормалізована структура бази даних містить дублікати інформації, що зберігається в різних таблицях, що дозволяє створювати різні запити з використанням різних таблиць для отримання однієї і тієї ж інформації.

Список літератури

1. *Griffiths, P.G., and Wade, B. W. 1976. An authorization mechanism for a relational database system. ACM Trans. Database Syst. 1, 3 (Sept.), 242-255.*
2. *Stonebraker, M. 1975. Implementation of integrity constraints and views by query modification. In Proceedings of the ACM-SIGMOD International Conference on Management of Data (San Jose, Calif., May 14-16). ACM, New York, pp. 65-78.*
3. *Reimer, M. 1983. Solving the phantom problem by predicative optimistic concurrency control. In Proceedings of the 9th International Conference on Very Large Data Bases (Florence, Italy). VLDB Endowment, Saratoga, Calif., pp. 81-88.*
4. *Codd, E.F. 1971. A database sublanguage founded on the relational calculus. In Proceedings of the ACM-SIGFIDET Workshop, Data Description, Access, and Control (San Diego, Calif., Nov. 11-12). ACM, New York, pp. 35-68.*
5. *Gotlieb, L. R. 1975. Computing joins of relations. In Proceedings of the ACM-SIGMOD International Conference on Management of Data (San Jose, Calif., May 14-16). ACM, New York, pp. 55-63.*
6. *Astrahan, M.M., and Chamberlin, D.S. 1975. Implementation of a structured English query language. Commun. ACM 18, 10 (Oct.), 580-588.*
7. *Бойко В.В., Савинков В.М. Проектирование баз данных информационных систем // М.: Финансы и статистика, 1989. – С. 351.*
8. *Бочаров Е.П. Интегрированные корпоративные информационные системы. Учебное пос., 2007. – 288 с.*
9. *Коннолли Т., Бегг К., Страчан А. Базы данных: проектирование, реализация и сопровождение. Теория и практика. 2-е изд. Пер. с англ. М.: Издат. дом «Вильямс», 2000. –1120 с.*

Надійшла до редколегії 22.12.2017

Рецензент: д-р техн. наук, проф. К.С. Козелкова, Державний університет телекомунікацій, Київ.

ПОВЫШЕНИЕ ПРОИЗВОДИТЕЛЬНОСТИ СИСТЕМ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ

И.В. Ильина, Д.С. Кадубенко, Д.А. Ильин

Анализируются подходы минимизации времени для получения необходимой информации из базы данных, а также критерии и параметры запросов, которых необходимо придерживаться в процессе разработки. Рассматриваются вопросы создания и оптимизации запросов для повышения производительности, что позволяют сделать функционирование базы данных наиболее эффективными. Обоснована необходимость использования данных подходов в разных структурах.

***Ключевые слова**: база данных, минимизации времени, СУБД, Oracle, алгоритм поиска, оптимизация запроса.*

IMPROVING THE PERFORMANCE OF DATABASE MANAGEMENT SYSTEMS

I.V. Ilina, D.S Kadubenko, D.O. Ilin

The article discusses time minimization approaches for obtaining the necessary information from the database, as well as criteria and parameters, adherence to the development process, creation and maintenance of optimization of productivity enhancement, make it work most efficiently. Justified the need for availability in various structures now.

***Key words:** database, time minimization, DBMS, Oracle, search algorithm, query optimization*