МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. В. ДАЛЯ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЕЛЕКТРОНІКИ
КАФЕДРА КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ

До захисту допускається
Завідувач кафедри
_____ Скарга-Бандурова І.С.
«____»_____ 20__ р.

МАГІСТЕРСЬКА РОБОТА

НА ТЕМУ:

Методи підвищення ефективності роботи баз даних в реальному часі

Освітньо-кваліфікаційний рівень "Магістр"
Спеціальність 123 - "Комп'ютерна інженерія" (Спеціалізація - "Комп'ютерні системи і мережі")

Науковий керівник роботи: _____ (підпис) Скарга-Бандурова І.С. (ініціали, прізвище)

Консультант з охорони праці: _____ (підпис) Критська Я.О. (ініціали, прізвище)

Студент: _____ (підпис) Неудакіна Л.В. (ініціали, прізвище)

Група: КСМ-16дм

Сєвєродонецьк 2018

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет Інформаційних технологій та електроніки
Кафедра Компьютерної інженерії
Освітньо-кваліфікаційний рівень магістр
Спеціальність 123 – "Комп'ютерна інженерія"
(шифр і назва)
Спеціалізація "Комп'ютерні системи і мережі"
(шифр і назва)

ЗАТВЕРДЖУЮ:

Завідувач кафедри _____
_____ І.С. Скарга-Бандурова
«_____» _____20____р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Неудакіній Людмилі Валеріївні
(прізвище, ім'я, по батькові)

1. Тема роботи Методи підвищення ефективності роботи баз даних в реальному часі

керівник проекту (роботи) д.т.н., проф. Скарга-Бандурова Інна Сергіївна
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від " " 2017 р. № ____
2. Строк подання студентом роботи
3. Вихідні дані до роботи метод динамічного самонастроювання бази даних для кафедри комп'ютерної інженерії ВНУ ім.В.Даля: опис, документація

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Дослідження існуючих методів і підходів до самонастроювання в системах управління базами даних 2. Дослідження впливу різних стратегій настройки продуктивності бази даних і енергоефективності 3. Розробка методу динамічного самонастроювання бази даних на основі рівняння пропусків буфера, отриманого з аналітичної моделі, і адаптивної нейро-нечіткої методики обліку змін в розмірі бази даних при тривалому використанні програми бази даних
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) електронні плакати

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	ст.викл. Критська Я.О.		

7. Дата видачі завдання _____

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Огляд недавніх досліджень проблем поганої продуктивності бази даних. Визначення вимог до роботи.	05.09.17-19.09.17	
2	Огляд основних технологій точної настройки бази даних.	20.09.17-04.10.17	
3	Опис стратегій до настройки продуктивності бази даних.	05.10.17-19.10.17	
4	Дослідження впливу різних стратегій настройки продуктивності бази даних і енергоефективності.	20.10.17-03.11.17	
5	Розробка методу динамічного самонастроювання бази даних.	04.11.17-18.11.17	
6	Розробка заходів з охороні праці.	19.11.17-03.12.17	
7	Оформлення пояснювальної записки і графічного матеріалу.	04.12.17-18.12.17	
8	Підготовка та подання магістерської роботи до захисту.	19.12.17-18.01.18	

Студент _____ Неудакіна Л.В.
(підпис) (прізвище та ініціали)

Науковий керівник _____ Скарга-Бандурова І.С.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Неудакіна Л.В. Методи підвищення ефективності роботи баз даних в реальному часі.

Досліджені існуючі методи та підходи до самонастроювання в системах управління базами даних. Проведено дослідження впливу різних стратегій настройки продуктивності бази даних і енергоефективності. Розроблено метод динамічного самонастроювання бази даних на основі рівняння пропусків буфера, отриманого з аналітичної моделі, і адаптивної нейро-нечіткої методики обліку змін в розмірі бази даних при тривалому використанні програми бази даних.

Ключові слова: база даних, системи управління базами даних, самоналаштування, продуктивність, рівняння пропусків буфера, нейро-нечітка методика, адміністратор бази даних.

АННОТАЦИЯ

Неудакина Л.В. Методы повышения эффективности работы баз данных в реальном времени.

Исследованы существующие методы и подходы к самонастройке в системах управления базами данных. Проведено исследование влияния различных стратегий настройки производительности базы данных и энергоэффективности. Разработан метод динамической самонастройки базы данных на основе уравнения пропусков буфера, полученного из аналитической модели, и адаптивной нейро-нечеткой методики учета изменений в размере базы данных при длительном использовании программы базы данных.

Ключевые слова: база данных, системы управления базами данных, самонастройка, производительность, уравнение пропусков буфера, нейро-нечеткая методика, администратор базы данных.

ABSTRACT

Neudakina L.V. Methods to improve the performance of databases in real time.

Existing methods and approaches to self-tuning in database management systems are explored. A study was made of the impact of various strategies for tuning database performance and energy efficiency. A method of dynamic database self-tuning based on the buffer missing equation obtained from the analytical model and an adaptive neural-fuzzy methodology for accounting for changes in the size of the database with long-term use of the database program are developed.

Key words: database, database management systems, self-tuning, performance, buffer missing, neural-fuzzy technique, database administrator.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. Продуктивність бази даних та розподіл ресурсів, що використовуються системою управління базами даних (СУБД), безпосередньо пов'язані один з одним. Проблеми діагностики та настройки продуктивності складних і тимчасових завдань утворюються через складні відносини між численними ресурсами СУБД. В наш час підприємствам необхідні дорогі адміністратори баз даних (DBA) для первинної настройки СУБД з метою підвищення продуктивності, а потім - для переінсталяції СУБД з мірою зростання бази даних та зміни робочих навантажень. Для того, щоб знизити вартість власності СУБД, необхідно використовувати автоматичну діагностику та управління ресурсами, що усуває необхідність у DBA. Завдяки автоматизованій системі також можна досягти швидшого реагування СУБД на зміни робочого навантаження, тому що продуктивність може контролюватися 24 години на добу. Автоматизована система діагностики та управління ресурсами працює над підвищенням продуктивності і для статичних, і для динамічних робочих навантажень.

Ключовою проблемою автоматичного управління ресурсами є здатність системи діагностувати проблеми з ресурсами. Перший крок у процесі налаштування ресурсів – це саме діагностика проблеми виділення ресурсів. У цій роботі пропонується розробити метод динамічного самонастроювання бази даних, на основі адаптивної нейро-нечіткої методики та рівнянні пропусків буфера. Формально буде вирішена задача покращення характеристик самонастроювання для СУБД.

Система діагностики включає в себе: порівняння продуктивності робочого навантаження, налаштованої експертом діагностичної системи, і програмного забезпечення майстра настройки продуктивності в тестовій базі даних. Системою успішної діагностики вважається досягнення продуктивності робочого навантаження, що близьке або краще, ніж вказані методи настройки.

Вклад цієї роботи включає в себе: дослідження існуючих методів і підходів до самонастроювання в СУБД, дослідження впливу різних стратегій настройки продуктивності бази даних і енергоефективності та розробку методу динамічного самонастроювання бази даних.

Тому обґрунтованою є тема магістерської роботи, у якій вирішується **науково-прикладне завдання** розроблення моделей і методу інформаційної технології підвищення ефективності роботи баз даних в реальному часі.

Об'єкт дослідження – процеси забезпечення самонастроювання в системах управління базами даних.

Предмет дослідження – моделі та метод інформаційної технології оцінки і забезпечення підвищення ефективності роботи баз даних в реальному часі.

Мета і завдання дослідження.

Метою дослідження є вирішення задачі покращення характеристик самонастроювання для систем управління базами даних.

Для досягнення мети дослідження необхідно вирішити такі завдання:

- дослідження існуючих методів і підходів до самонастроювання в системах управління базами даних;
- дослідження впливу різних стратегій настройки продуктивності бази даних і енергоефективності;
- розробка методу динамічного самонастроювання бази даних на основі рівняння пропусків буфера, отриманого з аналітичної моделі, і адаптивної нейро-нечіткої методики обліку змін в розмірі бази даних при тривалому використанні програми бази даних.

Методи дослідження. Проведені в роботі дослідження основані на адаптивній нейро-нечіткій методиці та рівнянні пропусків буфера, які використовувались при розробленні методу динамічного самонастроювання бази даних.

Наукова новизна отриманих результатів:

1) Удосконалено аналітичні моделі, з отриманням рівняння пропусків буфера і з використанням адаптивної нейро-нечіткої методики шляхом врахування різних варіантів обліку змін в розмірі бази даних при тривалому використанні програми бази даних, що дозволяє підвищити точність динамічного самонастроювання бази даних.

2) Дістали подальшого розвитку оціночні моделі з використанням модулів діагностики і аналізатора шляхом урахування вузьких місць в ресурсах, що дозволяє оцінити продуктивність поточної системи та автоматично налаштувати конкретні параметри ресурсів продуктивності відповідно до реальних значень.

Особистий внесок здобувача полягає у розробленні нових моделей, методів та інструментальних засобів, що дозволяють вирішити поставлені задачі. Усі основні результати отримані автором особисто. Робота над першим розділом магістерської дисертації опублікована у співавторстві.

Апробація матеріалів дисертації. Основні положення, ідеї, висновки магістерської роботи доповідалися та обговорювалися на науково-технічних конференціях «Проблеми інформатики і моделювання» (м. Харків, 2017); «II International Conference TACSIT-2017» (м. Сєвєродонецьк, 2017), «III регіональний Форум IT-Ідея 2017» (м. Сєвєродонецьк, 2017).

Зв'язок з науковими програмами, планами, темами. Дисертаційна робота виконана у Східноукраїнському національному університеті ім. В. Даля у відповідності з державними програмами і планами НДР:

- Проектом 573818-EPP-1-2016-1-UK-EPPKA2-CBHE-JP «Internet of Things: Emerging Curriculum for Industry and Human Applications (ALIOT)» (2016-2019 pp.).

Практичне значення отриманих результатів полягає в тому, що основні наукові положення дисертації реалізовані у виді розрахункових моделей та адаптивної нейро-нечіткої методики, які утворюють метод динамічного самонастроювання бази даних.

Розроблено метод динамічного самонастроювання бази даних на основі рівняння пропусків буфера, отриманого з аналітичної моделі, і адаптивної нейро-нечіткої методики обліку змін в розмірі бази даних при тривалому використанні програми бази даних.

Завдяки використанню цього методу можна вирішити задачу підвищення ефективності роботи баз даних в реальному часі.

Публікації. За темою магістерської роботи з викладенням її основних результатів опубліковано тези 3 доповідей міжнародних конференцій.

Структура та обсяг дисертації. Дисертація складається із вступу, чотирьох розділів, висновків, списку використаних джерел і додатків. Загальний обсяг дисертації складає 108 сторінок, з яких основний текст на 86 сторінках, список використаних джерел із 94 найменувань на 12 сторінках, додатки на 7 сторінках. Робота містить 21 таблицю, та 37 рисунків.

1 ДОСЛІДЖЕННЯ ІСНУЮЧИХ МЕТОДІВ І ПІДХОДІВ ДО САМОНАСТРОЮВАННЯ В СИСТЕМАХ УПРАВЛІННЯ БАЗАМИ ДАНИХ

1.1 Огляд недавніх досліджень проблем поганої продуктивності бази даних і підтримки відповідного робочого рівня під час обчислювальних операцій

Результати розділу опубліковано в тезах шістнадцятої міжнародної науково-технічної конференції «Проблеми інформатики і моделювання» [1].

Ефективне використання ресурсів, покращена продуктивність та низька вартість володіння – це важливі моменти, які вирішує самостійне налаштування. Для втілення в життя систем самоналаштування більшість сучасних систем управління базами даних (СУБД) представили кілька параметрів, що динамічно налаштовуються. Поєднання різноманітних параметрів налаштування призводить до значного підвищення продуктивності як з точки зору часу відповіді запитів, так і загальної пропускної здатності. В залежності від впливу параметрів на продуктивність, кількість та тип робочого навантаження, повинні робитися вибір і ступінь налаштування доступних параметрів. Самоналагоджувальна база даних звільняє від довгого та втомливого ручного налаштування, а також пошуку експертних DBA. Таким чином, знижується загальна вартість володіння всією програмною системою. Система самоналаштування добре підлаштовується до динамічних змін робочого навантаження, а також навантажень користувача в пікові години, забезпечуючи прийнятний час відгуку програми. У цій роботі представлений новий метод, який поєднує в собі здатність до навчання штучної нейронної мережі і здатність нечіткої системи справлятися з неточними входами, для оцінки необхідного ступеня налаштування. Крім того, завдяки проведеним експериментам, модеруються оціночні значення на основі бази знань. Експериментальні дані демонструють суттєве покращення продуктивності в порівнянні з вбудованою функцією самоналаштування СУБД.

Численні підприємства та організації активно використовують СУБД у повсякденному функціонуванні, а також прийняття рішень для ефективного ведення бізнесу і підвищення прибутковості. Неймовірно велика кількість зусиль та коштів буде витрачена на встановлення, навчання та впровадження програми, керованої базою даних. Однак тонке налаштування бази даних на основі кількості і типу робочого навантаження, якій піддається база даних, обділяється увагою. Не дивлячись на те, що більшість СУБД мають значення параметрів налаштування за умовчанням або значення параметрів налаштування, обчислені з використанням правил великого пальця, система може не підтримувати рівень продуктивності в подальшій перспективі. Одна з основних причин цього - несподіване збільшення числа користувачів програми. Це явище притаманне сценаріям веб-додатків, де популярність веб-сайту призводить до величезного

збільшення числа користувачів. З часом шаблон робочого навантаження також може змінюватися. Різко зростаючий розмір БД – є останньою і найважливішою причиною низької продуктивності. Протягом довгого часу (від 3 років) користування СУБД забезпечити необхідний рівень продуктивності неможливо з використанням налаштування за умовчанням або рекомендованого налаштування, заснованого на правилах великого пальця. В результаті, зниження продуктивності спостерігається в пікові робочі години і, через поганий час відгуку, бізнес-додаток починає втрачати своїх користувачів. Також виникає проблема апаратного забезпечення, що підтримує додаток - воно стає застарілим. Як наслідок, страждає продуктивність, і кінцеві користувачі відчувають неприпустимі затримки, що веде до значних втрат у бізнесі. У більшості підприємств є системний адміністратор, який мало або зовсім не знає параметрів налаштування і їх впливу на час відгуку додатків кінцевого користувача. Через некомпетентність DBA, а також бюджетних обмежень підприємств при використанні висококваліфікованого й експертного DBA, може привести до некоректної системи, яка не відповідає цілям продуктивності, встановленим розробниками додатків. Мета полягає в використанні адаптивної нейро-нечіткої методики, яка являє собою комбінацію нейронної мережі [2-4] і нечітких систем [5-7].

1.2 Основні технології точної настройки бази даних

У літературі запропоновано кілька методів для самостійного налаштування СУБД. У статті, опублікованій [8], представлена нова технологія самоналаштування, заснована на аналізі витрат-вигод і методах теорії управління. У цій технології метод з розумною точністю визначає кількість часу обробки, який було б збережено, якби була розподілена додаткова пам'ять. Даний метод має два недоліки. По-перше, метод оцінює час, збережений тільки з розумною точністю. Це означає, що якщо дані не є достатньо точними, розрахункова економія часу буде сильно відрізнятися від того, що було б насправді. По-друге, розрахунковий час накопичується, щоб знайти кумулятивний час збереження, а потім усереднити. У цьому процесі середня розрахункова вартість може бути грубо невірної. Як наслідок, цей метод або переналаштовує, або недооцінює систему, що призводить до поганої продуктивності системи, а також до неефективного використання ресурсів. Крім того, обчислювальні накладні витрати при оцінці вартості вигоди можуть затьмарити переваги протягом часу відгуку всієї системи. Надалі, пропонований метод працює в три етапи до того, як параметри налаштування будуть точно відрегульовані, що призведе до помітної затримки при установці параметрів налаштування на необхідні значення. Метод, запропонований в [9], для динамічного самонастроювання буферів бази даних заснований на рівнянні помилки буфера, отриманому з аналітичної моделі. Незважаючи на те, що рівняння перевірено з використанням експериментальних даних, це рівняння може виявитися непридатним для різних типів робочих

навантажень і умов завантаження користувача. Крім того, рівняння не враховує збільшені розміри бази даних при тривалому використанні програми бази даних, що призводить до розрахункового значення коефіцієнта пропускання буфера і, отже, до формування розмірів буфера.

У документі [10] представлена модель, заснована на цільовому управлінні автономними системами баз даних. Логіка самоконтролю витягує входи датчиків, такі як робоче навантаження, стан і т. д., і визначає значення ефекту, необхідні для фізичної бази даних, або зміни, необхідні для настройки регуляторів, з урахуванням цілей настройки, встановлених адміністратором. Алгоритм вилучення правил визначає, які правила слід застосовувати для точної настройки бази даних, щоб завдання продуктивності, такі як час відгуку або пропускна здатність, задовольнялися з певною точністю. Оскільки параметри налаштування системи бази даних нелінійно пов'язані з перфорацією, будь-то спрощене припущення при моделюванні системи може не відповідати цілям налаштування, встановленим адміністратором баз даних. Системи баз даних у виробничих середовищах повинні володіти довірою, налаштування на основі цієї моделі може не працювати на практиці.

В якості пропонованого рішення можна використовувати набір інструментів для обрізки і колонок [11,12], використовувати технології самовідновлення [13,14], використовувати фізичні налаштування дизайну. Змінено класичний контроль, і для забезпечення стабільності системи використовується триступеневий контроль, що включає моніторинг, аналіз і налаштування [15,16]. Бажано мати механізм управління, який є досить швидкий, щоб налаштувати систему в режимі реального часу. Однак, з трифазним підходом цей метод не працює досить швидко, щоб відповідати цілям продуктивності кінцевих користувачів або розробників додатків. Архітектура, представлена в [17] для бази даних самовідновлення представляє нову архітектуру СУБД, засновану на модульному підході, причому кожен функціональний модуль може контролюватися набором контрольних хуків. Ці контрольні хуки відповідають за збереження поточної інформації про стан або здійснюють моментальний знімок сервера в журналі. Дана архітектура має високі контрольні витрати через те, що при великій кількості параметрів, які підлягають моніторингу, майже кожна інформація про статус модуля повинна зберігатися в журналі, і, якщо це зроблено, часто може знадобитися значна кількість процесорного часу. Ранжування різних параметрів налаштування на основі статистичного аналізу представлено в [18]. Ранжування параметрів засноване на величині впливу, який вони чинять на продуктивність системи для даного робочого навантаження. Формальна база знань для системи самонастроювання бази даних представлена в [19], яка визначає кілька компонентів знань. Компоненти знань включають в себе знання про технології, знання робочого навантаження, знання про діагностику проблеми, знання про дозвіл проблем, знання виконавців і знання залежностей. Налаштування бази даних з використанням концепції віртуалізації представлена в [20], де системні ресурси, такі як процесорний час,

пропускна здатність введення/виведення і пам'ять розподіляються для декількох віртуальних обчислювальних середовищ з кожної СУБД, що працює на кожній з цих віртуальних машин, що відносяться до певного типу робочого навантаження. Налаштування на основі аналізу витрат-вигод [8,21-25] представлене в деяких з таких систем баз даних, як: Oracle 10g і DB2. IBM має самоналагоджувальний диспетчер пам'яті (СНДП), який знову заснований на аналізі витрат. Цей метод оцінки економії на диску і/або часу процесора заданого обсягу пам'яті врятує кожного із споживачів. Дана метрика, тобто час/одиниця пам'яті, використовується для оцінки відносної потреби в пам'яті для всіх споживачів. Обчислення економії часу на одного споживача, а потім оцінка потреби в системній пам'яті - це не тільки дорого, але може також привести до несправності системи, якщо оціночні значення більшою мірою неточні. Також передбачувана економія в часі буде функцією робочого навантаження, кількості активних користувачів, а також характеристик диска. Для визначення коефіцієнта витрат потрібен додатковий простір, що становить 10% або більше від розміру пулу буферів [26]. З документації продукту й опублікованих офіційних документів Oracle 10g має функцію автоматичного управління спільною пам'яттю (АУСП), яка управляє субкомпонентами SGA динамічно на основі характеристик робочого навантаження. АУСП вимагає, щоб користувач поставив обмеження на загальний обсяг пам'яті, яку може споживати база даних, завдання, яке так само складне, як і ручне налаштування бази даних. У цьому огляді літератури показано, що необхідно провести детальне експериментальне дослідження впливу всіх параметрів динамічного налаштування на час відповіді на запит в різних умовах навантаження, а також вплив одного параметра налаштування на інший при зміні регуляторів налаштування під час виконання. Також необхідно сформулювати політики налаштування, що визначають, які параметри налаштування повинні бути налаштовані, в якій мірі і коли. Специфікація політик налаштування повинна бути гнучкою і адаптуватися до умов робочого навантаження, що змінюються. Крім того, існує можливість застосовувати методи машинного навчання [27] для оцінки значень параметрів налаштування з відповідним набором навчальних матеріалів, використовуваних для навчання системи управління, перш ніж застосовуватися для цієї мети.

Далі буде представлено новий метод налаштування продуктивності, заснований на оцінці значень параметрів налаштування з використанням здатності навчання нейронних мереж і здатності справлятися з неточними введеннями даних нечіткої логіки [28]. Входи є неточними, тому що до моменту вилучення значень і їх використання при оцінці параметрів налаштування значення будуть змінені. Крім того, оціночні значення модеруються модулем налаштування, який або масштабується вгору, або вниз, розрахункові значення параметрів налаштування засновані на коефіцієнті впливу параметрів налаштування на час відгуку. Настроювач консультується з базою знань, яка побудована з використанням експериментальних даних про те, наскільки ефективний

кожен параметр налаштування в поліпшенні часу відгуку, в порівнянні з іншими параметрами налаштування. Ця інформація допомагає правильно встановити значення параметрів налаштування, які не недооцінені і не переоцінені. Оскільки модерація заснована на експериментальних фактах, а не на аналітичних або емпіричних моделях, то дія налаштування буде не тільки точною, але й надійною.

1.3 Опис стратегій до настройки продуктивності БД

Більшість СУБД поставляються з декількома параметрами, що налаштовуються, і після установки встановлюються значення за умовчанням. Більшість параметрів налаштування вимагають перезапуск системи після модифікації і для заданого типу робочого навантаження, правильний вибір і налаштування цих параметрів можуть значно підвищити продуктивність. Більшість сучасних систем баз даних, що включають Oracle 10g, DB2, MSSql Server 2010 мають динамічно настроюванні параметри, які дозволяють онлайн-налаштування без будь-якого простою. Наприклад, Oracle 10g має Db_Cache_Size, Large_Pool_Size, Shared_Pool_Size і Java_Pool_Size в якості параметрів, що динамічно настроюються. Однак, розміри цих параметрів налаштування повинні бути встановлені таким чином, щоб сума розмірів всіх параметрів налаштування не перевищувала верхню межу, задану СУБД. Завдання налаштування можна формально викласти як:

Враховуючи набір параметрів налаштування $(Tp_1, Tp_2 \dots Tp_n)$

За умовою, що $\sum Tp_i \leq U_{max}$

Мінімізувати час відповіді на запит $Rt(Tp_1, Tp_2 \dots Tp_n)$

Де U_{max} є верхньою межею перебудованого ресурсу.

Функція управління системою самоналаштування повинна відповідати зазначеному вище обмеженню і досягати необхідної продуктивності, зводячи до мінімуму час відгуку запитів, поданих в СУБД для виконання. Перебудований ресурс в цьому випадку - це пам'ять СУБД. Запитами, відправленими в систему, можуть бути окремі запити або частина певного типу робочого навантаження, що генерується певним кінцевим користувачем. Механізм управління самонастроювання повинен мати низьке обчислювальне навантаження і не повинен призводити до нестабільності системи.

Запропонована архітектура системи зображена на рис. 1.1. Вона включає в себе модуль, який обчислює вхідні параметри для нейро-нечітких модулів, а саме відношення «біт-удар» (BHR), кількість активних користувачів і розмір бази даних. Модуль з нечіткою перевіркою ґрунтується на базі правил і діє на блоковані входи для генерації оціночних значень параметрів налаштування. Добре навчена нейронна мережа оцінює значення параметрів налаштування з

урахуванням поточного навантаження користувача, BHR і бази даних середнього розміру. Функції кожного з модулів системи описуються наступним чином.

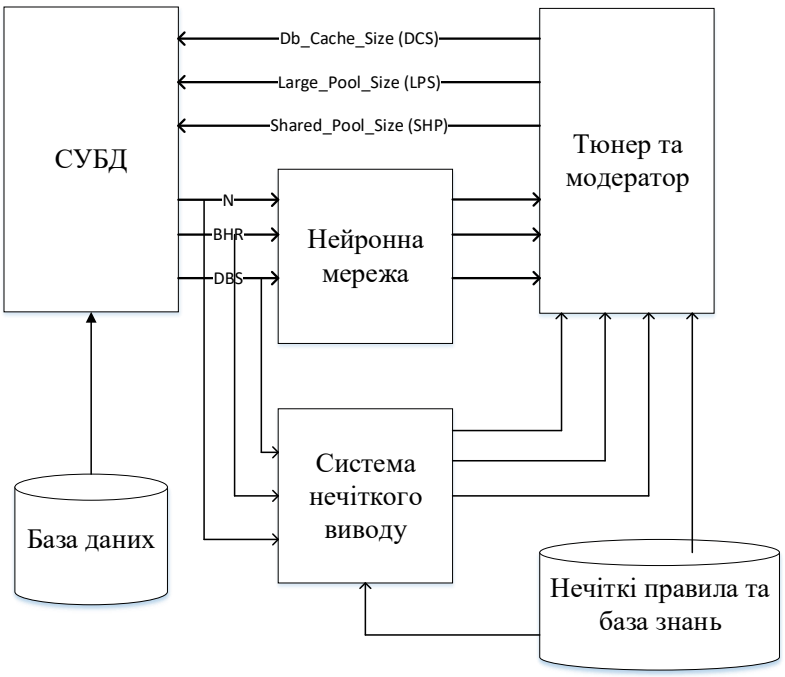


Рисунок 1.1 - Архітектура самоналаштування продуктивності

Калібрування системи для необхідного часу відгуку називається налаштуванням продуктивності. Блок-схема показує функціональні блоки, які складають всю систему управління, зовнішню по відношенню до СУБД. Як видно з архітектури, три важливих вхідних параметра, а саме: кількість користувачів (N), BHR і середній розмір бази даних (DBS) подаються як на нейронні, так і на нечіткі системи для оцінки відповідного розміру параметрів налаштування. Нейронна мережа навчається відповідному набору навчальних даних, отриманого з експериментальних спостережень. Хоча є кілька параметрів, що настроюються, тільки три параметри вибираються для налаштування, оскільки вони мають значний вплив на час відгуку і, що найважливіше, динамічно змінюються без необхідності повертатися назад в СУБД.

Нейронні мережі найкраще підходять для обробки складних систем, які в дійсності нелінійні за своєю природою. Нейронні мережі відносяться до більш широкого предмету, що називається комп'ютерним навчанням, яке займається комп'ютерними рішеннями, які вивчають складний взаємозв'язок між входом і виходом з набору даних емпіричного навчання. Застосування техніки машинного навчання досить широко поширене, в тому числі: аналіз медичних зображень для точного прогнозування помилок, пошук близьких оптимізаційних рішень до завдань NP-Hard, особливо в мережевій області і т. д. Машинне навчання [20] забезпечує ефективний метод прогнозування оціночних значень параметрів настроювання з певного набору навчальних

матеріалів. Як показано на рис. 1.2, нейронна мережа буде мати входи P, певну кількість вузлів в прихованому шарі і один або кілька вихідних вузлів. Нейронна мережа, яка використовується в цій керуючій архітектурі, є прямою мережею зворотного поширення. Використовувана функція активації є сигмоідальною функцією для всіх внутрішніх вузлів. Саме ця функція дає нейронній мережі можливість вчитися і виробляти висновок, для якого вона не навчена. Проте, нейронні мережі потребують чітко визначених наборів навчальних даних для їх належного функціонування. Вихідні вузли мають чисту лінійну активаційну функцію генерації остаточних оціночних значень для певних тестових вхідних даних.

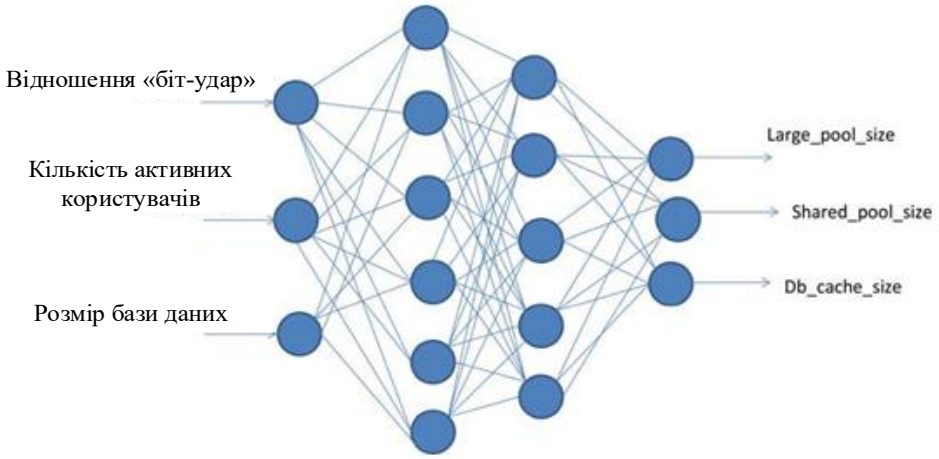


Рисунок 1.2 - Основна структура нейронної мережі з двома прихованими шарами

Нейронні мережі працюють поетапно. На першому етапі мережа навчається з використанням чітко визначеного навчального набору для бажаного висновку. На другому етапі в мережу відображається новий вхід, що називається тестовим входом, який може або не може бути частиною набору навчальних даних, і мережа виробляє бажаний висновок, який найбільш підходить для певних вхідних даних. Для правильної роботи нейронної мережі важливо вибрати правильну функцію активації, швидкість навчання, кількість циклів навчання і значну кількість вузлів у прихованому шарі. У експериментальній установці перший прихований шар має 12 вузлів, а другий прихований шар має 8 вузлів. Для більш точних результатів швидкість навчання встановлена на 0,0001.

Набір даних для навчання заснований на серії експериментів, проведених на сервері Dell 74100 Server, з 2,9 ГГц, на двох 6-ядерних процесорах, 16 ГБ оперативної пам'яті і 500 ГБ жорсткого диска з підтримкою RAID 5. Значення таблиці вказують вхідні та цільові значення для навчання нейронної мережі.

Навчальний набір даних в таблиці 1.1, який використовується для навчання нейронної мережі, містить дані для двох різних розмірів бази даних. Як видно з другого набору даних, великі

значення пулу збільшені з більш високим інкрементним розміром для розміру бази даних 20, ніж для розміру бази даних 10. Це дозволяє нейронній мережі оцінювати більш високі значення великих розмірів пулу, які повинні генеруватися для запитів, які генерують виходи великих обсягів даних так, що будь-яке додаткове посилання або оновлення цього більшого набору даних не призведе до операції доступу до диска. Це призводить до більш швидкої обробки запитів.

Таблиця 1.1 - Набір навчальних даних для нейронної мережі (NN)

Вхід NN	Цільовий покажчик O/P					Вхід NN	Цільовий покажчик O/P				
BHR	N	DBS	DCS	SHP	LPS	BHR	N	DBS	DCS	SHP	LPS
98	5	10	10	80	8	98	5	20	10	80	16
96	10	10	15	88	8	96	10	20	15	88	16
94	15	10	20	88	16	94	15	20	20	88	16
92	20	10	25	88	16	92	20	20	25	88	16
90	25	10	30	96	32	90	25	20	30	96	24
88	30	10	35	96	32	88	30	20	35	96	24
86	35	10	40	96	48	86	35	20	40	96	24
84	40	10	50	120	64	84	40	20	50	120	64
82	45	10	60	120	96	82	45	20	60	120	64
80	50	10	70	124	96	80	50	20	70	124	96
78	60	10	80	132	108	78	60	20	80	132	108
76	70	10	90	132	120	76	70	20	90	132	136
74	80	10	100	132	136	74	80	20	100	132	164

Нечітка логіка є найбільш підходящим вибором для багатьох додатків управління [29] за те, що системи з нечітким керуванням є надійними і можуть бути легко змінені, щоб значно поліпшити продуктивність системи і, найголовніше, вони набагато простіші в реалізації. Крім того, немає необхідності вимірювати швидкість зміни вхідних параметрів, а кількість входів і виходів не обмежена невеликим числом.

Метою цієї системи є аналіз СУБД шляхом активного моніторингу показників ефективності, таких як: відношення пропусків буфера, кількість активних процесів і розмір таблиці, які демонструють ознаки швидкого зростання і ініціюють контрольну міру з використанням нечіткого контролю. Ця архітектура складається з модуля, який безперервно відстежує продуктивність системи, відзначаючи важливі показники продуктивності. Модуль fuzzifier зіставляє ці показники продуктивності з нечіткими змінними. Модуль нечіткого

управління використовує нечіткі правила, що складаються з інструкцій IF THEN про нечіткі вхідні змінні, щоб визначити характер змінних нечіткого виведення. Нечіткими вихідними змінними в цьому випадку будуть параметри налаштування СУБД. Отримавши ступінь налаштування, необхідну в нечітких термінах, модуль defuzzifier генерує чіткі вихідні параметри, які використовуються модулем налагоджувальника для точного налаштування СУБД.

Три вхідних параметра, а саме BHR, кількість активних користувачів (N) і розмір бази даних (DBS) збираються з СУБД і використовуються в якості вхідних даних для системи нечіткого управління. Визначення ключових показників ефективності має ґрунтуватися на ретельному аналізі характеристик, в якому досліджується вплив різних параметрів налаштування на ці показники ефективності. У даній експериментальній установці BHR, завантаження користувача і розмір бази даних є трьома параметрами, які вказують або впливають на продуктивність СУБД, вибираються як вхідні дані в модуль fuzzifier. Функції приналежності, вибрані для вхідних змінних, перераховані в таблиці 1.2.

Таблиця 1.2 - Функції членства, використовувані для різних вхідних параметрів

Функції членства для назви змінної введення / виведення	Функція членства
Коефіцієнт ударів буфера (BHR)	Gaussian
Кількість користувачів (N)	Triangular
Розмір бази даних (DBS)	Triangular
Db_cache_size (DCS)	Gaussian
Shared_pool_ size (SHP)	Trapezoidal
Large_pool_size (LPS)	Trapezoidal

Правила нечіткого управління засновані на загальних логічних міркуваннях і засновані на конструкціях мови програмування, а саме на конструкції управління IF THEN. Лінгвістичні терміни використовуються для опису необхідного ступеня налаштування. Ці правила повинні бути ретельно оформлені, і більшу частину часу потрібна модифікація до отримання бажаних результатів. Наприклад, щоб знайти розмір кешей нових буферів, деякі з правил нечіткості можуть бути сформовані таким чином:

Rule 1 IF BHR is high AND the user load is low THEN set BCS to low

Rule 2 IF BHR is moderate AND the user load is high THEN set BCS to high

Rule 3 IF BHR is low AND the user load is low THEN set BCS to moderate

Rule 4 IF BHR is low and the user load is high THEN set BCS to very high

Rule 5 IF BHR is high and DBS is high THEN set BCS to high

Rule 6 IF BHR is low AND the user load is low THEN set SHP to low

Rule 7 IF BHR is low and the user load is high THEN set SHP to moderate

Rule 8 IF BHR is high and DBS is high THEN set SHP to high

Rule 9 IF BHR is low and the user load is high THEN set BCS to very high

Rule 10 IF BHR is medium and user load is high THEN set LPS to high

Rule 11 IF BHR is low AND the user load is very high THEN set LPS to very large

Rule 12 IF BHR is very low and the user load is high THEN set SHP to large

Rule 13 IF BHR is moderate and DBS is high THEN set DCS to very high

На рисунку 1.3 показано вплив застосування нечітких правил на буферний кеш в залежності від BHR і кількості користувачів. У міру збільшення кількості користувачів розмір кешей буферів також збільшується з урахуванням значення BHR. Наприклад, якщо BHR дуже низький, а число користувачів дуже велике, розмір кеша буфера повинен бути надзвичайно великим. Найважливішою особливістю використання нечіткої логіки є те, що можна додати стільки нечітких правил, скільки необхідно, а також нечіткі змінні можуть бути відповідним чином підібрані для отримання найкращих результатів (рис. 1.3).

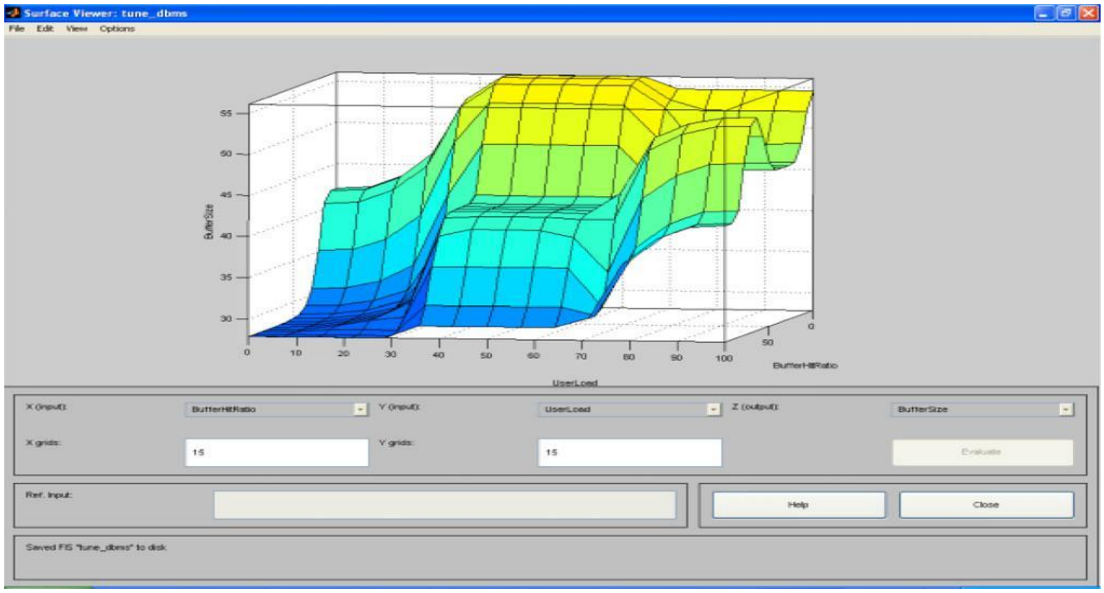


Рисунок 1.3 - Розмір кеша буфера як функція BHR і кількість користувачів

Отримавши значення параметрів налаштування в нечітких термінах, дезактивація повинна бути перенесена знову з використанням деяких функцій-членів для отримання чітких значень для встановлення антен. Потім настроювач ініціює управляючу дію, динамічно змінюючи значення параметра налаштування. У цьому випадку розмір кешу буфера, розмір великого пулу і розмір загального пулу змінюються, і вплив цих параметрів оцінюється з точки зору часу відповіді на запит.

Оціночні значення параметрів налаштування агрегуються шляхом призначення відповідних ваг виходам, які генеруються як нейронними, так і нечіткими підкомпонентами для кожного з параметрів налаштування. Ці оціночні значення можуть в деяких випадках бути неефективними або завищеними, можуть привести до нестійкості системи і, отже, необхідно розробити певну методику модерації, щоб забезпечити ефективне використання параметрів налаштування в поліпшенні часу відповіді на запит. У цьому розділі представлено новий метод модерації, заснований на відносному впливі на параметри налаштування, описані нижче: коефіцієнт впливу параметра налаштування Tp [i] для заданого типу робочого навантаження (j) визначається як:

$$IFactor(Tp[i], Workload\ Type[j]) = \left(\frac{\frac{\delta Rtime}{\delta Tp[i,j]}}{\sum_{i=0}^{TPN}\frac{\delta Rtime}{\delta Tp[i,j]}}\right) \quad (1.1)$$

Де TPN - кількість параметрів, що динамічно настроюються. δRtime - це зміна часу відгуку запиту по відношенню до δTp [i] - зміна параметра i-го налаштування для заданого типу робочого навантаження j.

В принципі, ця формула заснована на нахилах при кожному значенні параметрів налаштування на графіках діаграми часу відгуку в порівнянні з параметрами налаштування, як показано на рис. 1.4, 1.5 і 1.6.

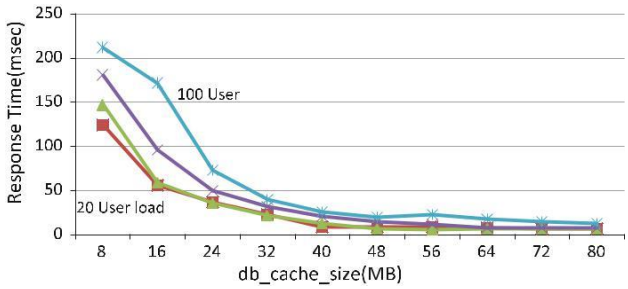


Рисунок 1.4 - Вплив розміру DB_CACHE на час відгуку на запит

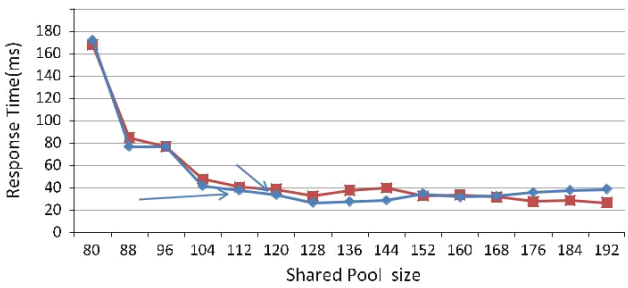


Рисунок 1.5 - Вплив розміру SHARED POOL на час відгуку на запит

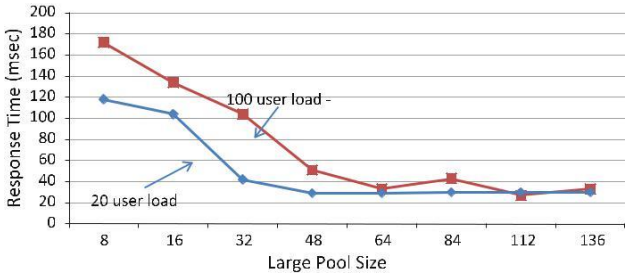


Рисунок 1.6 - Вплив розміру LARGE_POOL на час відгуку на запит

Кожен з параметрів налаштування має свої мінімальні значення. Мінімальні значення DB_CACHE_SIZE, SHARED_POOL_SIZE і LARGE_POOL_SIZE становлять відповідно 8, 80 і 8 Мбайт, і, отже, початкові значення осі X на трьох графіках різні. Крім того, діапазон параметрів налаштування також різний, оскільки їх вплив на час відгуку змінюється по-різному.

Три графіка показують, з якою швидкістю час відгуку запиту зменшується зі збільшенням значень параметрів налаштування. Ці графіки були отримані шляхом проведення експерименту тестування навантаження з використанням типу навантаження TPC-C. Зведення трьох графіків представлено в табличній формі в таблиці 1.3. У таблиці показано вплив кожного параметра налаштування, а також діапазон, в якому вплив є значимим. Експеримент також проводився для визначення впливу навантаження користувача на BHR, як показано в таблиці 1.4, для того, щоб знати, що діапазон BHR перевищує значення, і, відповідно, може бути побудований набір даних навчання.

Таблиця 1.3 - Вплив параметрів налаштування на час відповіді на запит

Параметр налаштування	% Вплив на час відгуку (МБ)	Діапазон налаштування
Db_cache_size	40 ms/8	Wide
Shared_pool_Size	35/8	Narrow
Large_pool_Size	17/8	Wide

Таблиця 1.4 - Вплив навантаження користувача на коефіцієнт попадання в буфер

Вплив навантаження користувача на коефіцієнт попадання в буфер	
Max. 99.12 (немає завантаження)	Min. 82.84 (100 користувачів)

Експериментальні результати показують, що Db_Cache_Size (DCS) має найбільший вплив і, отже, має бути дозволено збільшуватися з великою вагою, ніж два інших параметра. Наступний найвищий параметр - це великий розмір пулу і, отже, він займає друге місце. Нарешті, загальний розмір пулу має найнижчий і, отже, повинен рости повільно з ростом навантаження користувача.

Наступний алгоритм описує кроки, які застосовуються при оцінці нових значень параметрів налаштування, їх агрегації, поміркованості та дії налаштування.

АЛГОРИТМ PerformanceTuneDB (BHR, DBS, N, wLoadType [])

// Обчислюються значення параметрів налаштування $Tp[i \ldots n]$ на основі нейро-нечіткого підходу та модеруються обчислені значення на основі Impact Factor відповідного параметра налаштування.

// Введення: коефіцієнт використання буфера (BHR), розмір бази даних (DBS), кількість активних користувачів (N), тип робочого навантаження (wLoadType) і TPN → кількість параметрів налаштування, NWtype → кількість типів робочого навантаження.

// Виведення: нові значення параметра налаштування $Tp[1 \ldots n]$

Begin for i←1 to TPN do

$TpNN[i] = NN(BHR, DBS, N)$ // Оцінка з використанням підходу NN

$TpFL[i] = FIS(BHR, DBS, N)$ // Оцінка з використанням методу нечіткої логіки

next i

// Обчислити середньозважене значення $TpNN$ та $TpFL$

for i←1 to TPN do

$Tp[i] = 0{,}4 * TpNN[i] + 0{,}6 * TpFL[i]$ // Оцінка Tp за допомогою FL була краща в порівнянні з NN

next i

// Модеруються оціночні значення, засновані на коефіцієнті впливу кожного параметра налаштування

For $j = 1$ to NWtype // Для кожного типу робочого навантаження

For $i \leftarrow 1$ to TPN do // Для кожного параметра налаштування

$IFactor(Tp[i], wLoadType[j]) = \frac{\left(\frac{DeltaRtime}{DeltaTp[i,j]}\right)}{\sum_{i=1}^{TPN} \frac{DeltaRtime}{DeltaTp[i,j]}}$ // Обчислюється вплив налаштування

$Tp[i] = IFactor(Tp[i], wLoadType[j]) * Tp[i]$

performTuning ($Tp[i]$) // Виконання налаштування

next i

next j

end

Алгоритм передбачає, що нейронна мережа вже налаштована і навчена дійсним набором навчальних матеріалів, а також система нечіткого виведення готова до використання з відповідними нечіткими правилами. Дія налаштування виконується шляхом виконання команди в SQL, яка має наступний синтаксис:

ALTER SYSTEM SET DB_CACHE_SIZE = new_valueM scope = both;

Значення має бути зазначено в МБ, додавши M після new_value, а scope = both гарантує, що

зміни повинні бути зроблені негайно під час виконання, і ті ж значення мають підтримуватися навіть після перезапуску.

Налаштування продуктивності було виконано на сервері Dell 74100 Server, що має 2,9 ГГц, на двох 6-ядерних процесорах, 16 ГБ оперативної пам'яті і 500 ГБ жорсткого диска. Використовуваною СУБД була Oracle 10g з активним з'єднанням автоналаштування. Використовуваний тип робочого навантаження був TPC-C, який являє собою сценарій додатка обробки транзакцій. BenchMarkFactory використовувався для створення типу робочого навантаження TPC-C. Отримані результати показують значне поліпшення часу відгуку, як видно з графіка на рис. 1.7. Для користувачів <10 час відповіді двох методів майже однаковий.

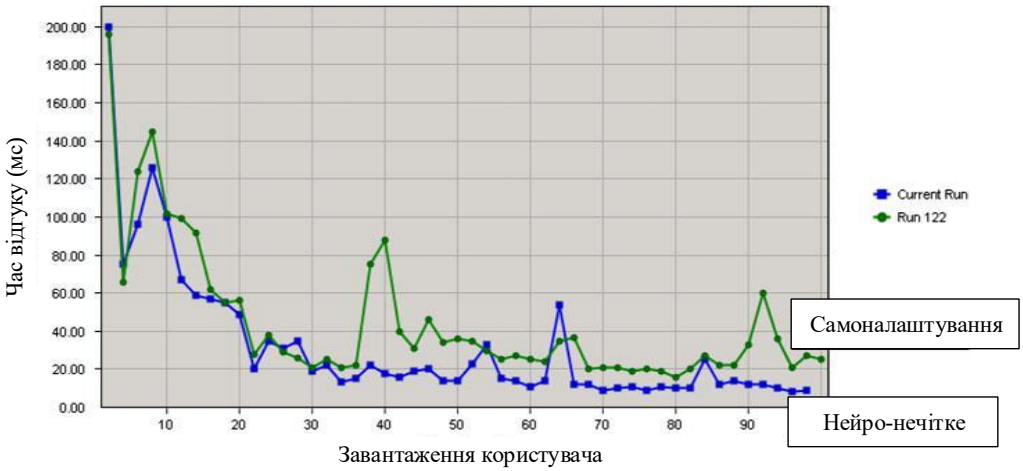


Рисунок 1.7 - Час відгуку від навантаження користувача

Це пов'язано з тим, що після запуску моделювання навантаження в кеші є доступ до не багатьох сторінок. Але поступово, коли все більше і більше користувачів починають використовувати системи - сторінки, запитані одним користувачем, доступні для інших в кеші. Як видно з графіка на рис. 1.7, час відповіді стійкий для великих користувальницьких навантажень, в разі нейро-нечіткого підходу, тоді як при автонастройці системи [21,30] час відгуку зростає для більшого числа користувачів. У першому випадку більш передбачуваний час відгуку, ніж в останньому. Крім того, в середньому є 33% -ве поліпшення за часом відгуку для навантажень користувача більше 32, що свідчить про ефективність розробленого методу в порівнянні з функцією автоналаштування, доступною в Oracle 10g.

У таблиці 1.5 представлені існуючі методи налаштування БД, їхні переваги та недоліки.

Таблиця 1.5 – Методи налаштування бази даних

№	Метод	Короткий опис методу	Переваги	Недоліки
1	Статистика бази даних	Найважливіший ресурс для оптимізатора SQL - це статистика, зібрана для різних таблиць у каталозі. Статистика - це інформація про індекси і їх розподіл по відношенню один до одного.	Визначення найменш дорогого шляху, що задовольняє запит.	Застаріла або відсутня статистична інформація призведе до того, що оптимізатор займе менш оптимізований шлях, що збільшить загальний час відгуку.
2	Створення оптимізованих індексів	Оптимізатор SQL залежить від індексів, визначених для певної таблиці. Жоден індекс не призведе до погіршення ефективності операторів SELECT, і занадто багато індексів сповільнить запити DML (INSERT, UPDATE і DELETE). Тому важливо мати правильний баланс індексу у таблицях.	Індекс прискорює запити, надаючи швидкий доступ до рядків даних в таблиці, аналогічно тому, як предметний покажчик в книзі допомагає швидко знайти бажану інформацію.	Якщо таблиці «бомбардуються» методами INSERT, UPDATE і DELETE, до індексації потрібно поставитися обережно - вона може привести до погіршення продуктивності.
3	Визначення очікуваного зростання	Індекси негативно впливають на запити DML. Один із способів звести до мінімуму цей негативний ефект - вказівка відповідного значення для коефіцієнта заповнення при створенні індексів.	Зниження негативного ефекту на запити DML. Зникає необхідність реорганізовувати сховище даних.	Не знайдено
4	Вказівка підказок оптимізатора в SELECT	Хоча в більшості випадків оптимізатор запитів вибирає відповідний індекс для конкретної таблиці на основі статистики, іноді краще вказувати ім'я індексу в запиті SELECT.	Підвищення ефективності виконання запиту.	Підказки (hints) можна використовувати лише у випадку крайньої необхідності. Намагатися підвищити ефективність за допомогою хінта можна тільки тоді, коли не було знайдено

				іншого рішення.
5	Використання EXPLAIN	Більшість баз даних повертають план виконання будь-якого оператора SELECT, створеного оптимізатором. Цей план дуже корисний для точної настройки SQL-запитів.	Корисний метод при тонкій настройці SQL-запитів. Дозволяє виявити повільні запити і скоротити час на обробку запиту, що згодом може значно прискорити роботу програми.	Недоліки EXPLAIN випливають з недоліків оптимізатора. Бувають випадки, коли він, взагалі, пише не те, що робить, видає дуже мало інформації, коли не виконує запит, а просто складає план.
6	Дві голови краще однієї	Жорсткий диск введення/виведення є одним з найбільш повільних ресурсів на комп'ютері, що стає очевидним під час збільшення розміру бази даних. Багато баз даних дозволяють користувачам розділити свою базу даних на кілька фізичних жорстких дисків. Фактично, деякі навіть йдуть далі і дозволяють розбивати вміст таблиці на кілька дисків.	Низькі витрати, масштабованість, гарне балансування навантаження, висока доступність, простота міграції з однопроцесорних систем.	Складність системи, потенційні проблеми продуктивності.
7	Вибір обмежених даних	Чим менше даних буде отримано, тим швидше буде виконуватися запит. Замість фільтрації на клієнті, як можна більше виконується фільтрації на сервері.	Пришвидшується виконання запиту.	Малоефективно при невеликому наборі результатів.
8	Відкидання індексів перед завантаженням даних	Дуже доречним є метод відкидання індексів таблиці перед завантаженням великої партії даних. Це змушує оператор insert працювати швидше. Після того, як вставки будуть завершені, можна знову створити індекс.	Оператор insert працюватиме швидше.	Потребує створення та використання, окрім основної таблиці, ще й тимчасової, котра найчастіше ускладнює структуру запиту.

1.4 Постановка наукової задачі та обґрунтування методики досліджень

Настройка продуктивності бази даних - це процес її коригування, так що база даних буде функціонувати у повній мірі з урахуванням її поточного або сукупного навантаження. Найбільш сучасні системи баз даних мають динамічно настроювані параметри, що дозволяють встановлювати онлайн-налаштування без втрати часу. Проте вхідні дані дуже неточні, залежні від часу та навантаження, й існує постійна вимога формулювати політику настройки, яка визначає, які параметри слід оптимізувати, до якої міри і коли. Як правило, налаштування за замовчуванням викликають надмірні налаштування чи слабкі настройки системи, що призводить до поганої продуктивності системи та неефективного використання ресурсів. У результаті спостерігається деградація продуктивності та поганий час відгуку. Ці фактори значно зменшують продуктивність бази даних та збільшують витрати на енергію.

Магістерська дисертація присвячена вирішенню задачі покращення характеристик самонастроювання для систем управління базами даних.

Основні задачі роботи:

1. Дослідження існуючих методів і підходів до самонастроювання в системах управління базами даних.

В цьому розділі обговорюється еволюція методів настройки бази даних, що використовуються в самоналагоджувальних базах даних. Більш конкретно, в першій частині представлені:

- огляд недавніх досліджень проблем поганої продуктивності бази даних і підтримки відповідного робочого рівня під час обчислювальних операцій, що можуть викликатись несподіваним зростанням користувачів або збільшенням розміру бази даних протягом тривалого періоду часу.
- основні технології точної настройки бази даних.
- опис стратегій до настройки продуктивності БД.

2. Дослідження впливу різних стратегій настройки продуктивності бази даних і енергоефективності.

У другій частині має бути представлена методологія, яка об'єднує функції автонастройки сучасної системи управління базами даних і динамічного самонастроювання.

Тут необхідно розглянули концептуальний підхід до управління базою даних, що забезпечує необхідний рівень продуктивності.

Детально проаналізувати метод заснований на адаптивній нейро-нечіткій технології для настройки продуктивності систем управління базами даних. Ця методологія може застосовуватися для онлайн-прогнозування критично зростаючого розміру бази даних, яка описує перевантаження

системи, погіршення продуктивності і неприпустимі затримки, що призводять до втрати енергії і бізнесу.

3. Розробка методу динамічного самонастроювання бази даних на основі рівняння пропусків буфера, отриманого з аналітичної моделі, і адаптивної нейро-нечіткої методики обліку змін в розмірі бази даних при тривалому використанні програми бази даних.

У третій частині має бути представлена самоналагоджувальна архітектура для моніторингу, настройки і тренда, а також результати реалізації пропонованої технології. Обговорюється оцінка витрат енергії і підвищення ефективності.

1.5 Висновки до першого розділу

В ході науково-дослідної практики були досліджені існуючі методи та підходи до самонастроювання в системах управління базами даних. Був проведений огляд недавніх досліджень проблем поганої продуктивності бази даних і підтримки відповідного робочого рівня під час обчислювальних операцій, що можуть викликатись несподіваним зростанням користувачів або збільшенням розміру бази даних протягом тривалого періоду часу. Розглянуті основні технології точної настройки бази даних та описані стратегії до настройки продуктивності БД. Також сформульована постановка задачі до магістерської дисертації.

Метод самонастроювання, представлений в цьому розділі, з використанням здатності навчання нейронної мережі і здатності діяти на неточні дані нечіткої логіки, показує значне поліпшення часу відгуку в порівнянні з функцією автоматичного налаштування сучасної СУБД. Більш того, здатність запропонованого методу генерувати майже прямий час відгуку, в порівнянні з методом автоналаштування зі збільшенням навантаження на користувача, дозволяє адміністратору баз даних впроваджувати систему, що має жорсткі вимоги до часу відгуку. Нова методика зміни значення оціночного параметра, заснованого на коефіцієнті Impact, дозволяє уникнути перенастроювання і, таким чином, зберігає пам'ять, яка може використовуватися для більш продуктивних цілей. Проте, необхідні подальші дослідження для встановлення аналогічних фактів в інших СУБД, а також для різних типів робочого навантаження. Крім того, вплив одного параметра налаштування на інший параметр налаштування повинно бути виміряно і включено в коефіцієнт впливу, так що крок модерації може ще більше поліпшити його дію, щоб обмежити значення параметрів налаштування.

1.6 Література до першого розділу

1. Нестеров М.В., Неудакіна Л.В., Скарга-Бандурова І.С. Прогностична модель для налаштування продуктивності систем управління базами даних // Проблеми інформатики і моделювання. Тезиси шістнадцятої міжнародної науково-технічної конференції – Харків: НТУ "ХПІ", 2017. – С. 64.
2. Acedo M.A., Molina M.A., Silva R., Marciano M., Portilla E.A. «Authentication review for nodes in wireless sensor networks» - Revista Electro´nica de Estudios Telema´ticos, 2009 – 9(1):1–23
3. Rubio JJ, Angelov P, Pacheco J (2011) Uniformly stable backpropagation algorithm to train a feedforward neural network. IEEE Trans Neural Netw 22(3):356–366
4. Pe´rez-Cruz JH, Alanis AY, Rubio JJ, Pacheco J (2012) System Identification based on multilayer differential neural networks: a new result. J Appl Math 2012:1–20
5. Rubio JJ (2009) SOFMLS: online self organizing fuzzy modified least square network. IEEE Trans Fuzzy Syst 17(6):1296–1309
6. Iglesias JA, Angelov P, Ledezma A, Sanchis A (2010) An evolving classification of agents behaviors: a general approach. Evol Syst 1(3):161–172
7. Leite D, Ballini R, Costa P, Gomide F (2012) Evolving fuzzy granular modeling from nonstationary fuzzy data streams. Evol Syst 3(2):65–79
8. Satish SK, Saraswatipura MK, Shastry SC (2007) DB2 performance enhancements using Materialized Query Table for LUW Systems, 2007. In: ICONS ’07, second international conference Storm AJ et al (2006) Adaptive self-tuning of memory in DB2. In:VLDB
9. Tran DN, Huynh PC et al (2008) A new approach to dynamic self-tuning of database buffers. In: ACM transactions on storage, vol 4
10. Holze M, Ritter N (2011) System models for goal-driven self-management in autonomic databases. Data Knowl Eng 70:685–701
11. Agarwal S et al (2007) Automated selection of materialized views and indexes. In: VLDB
12. Choudhuri S, Narasayya V (2007) Self tuning database systems: a decade progress. Microsoft Research
13. Koopman P (2004) Elements of the self-healing system problem space. In: IEEE data engineering bulletin
14. Liu P (2005) Design and implementation of self healing database system. In: IEEE conference
15. Nehme RV (2008) Database, heal thyself. In: Data engineering workshop
16. Wiese D, Rabinovitch G (2009) Knowledge management in autonomic database performance tuning. In: Proceedings of 3rd International conference on autonomic and autonomous systems, 2007, p 48

17. Debnath BK, Lilja DJ, Mokbel MF (2008) SARD: a statistical approach for ranking database tuning parameters. In: Data engineering workshop, 2008. ICDEW 2008. IEEE 24th international conference
18. Dageville B, Dias K (2006a) Oracle's self tuning architecture and solutions. In: IEEE data engineering bulletin, vol 29
19. Choudhuri S, Weikum G (2000) Rethinking database system architecture: towards a self tuning RISC style database system. In: VLDB, pp 1–10
20. Cheng SW, Garlan D et al (2006) Architecture based self adaptation in the presence of multiple objectives. In: Proceedings of 2006 international journal of computer systems and engineering
21. Agarwal S, Bruno N, Chaudhari S (2006) AutoAdmin: self tuning database system technology. In: IEEE data engineering bulletin
22. Weikum G, Moenkerngerg A et al (1993) Self-tuning database technology and information services: from wishful thing to viable engineering. In: Parallel and distributed information system
23. Chaudhuri S, Weikum G (2006) Foundations of automated database tuning. In: Data engineering
24. Rabinovitch G, Wiese D (2007) Non-linear optimization of performance functions autonomic database performance tuning. In: IEEE conference
25. Weikum G, Monkenberg A (2002) Self-tuning database technology: from wishful thinking to viable engineering. In: VLDB conference, pp 20–31
26. Chen ANK (2006) Robust optimization for performance tuning of modern database systems. Eur J Oper Res 171:412–429
27. Wang S, Summers RM (2012) Machine learning and radiology. Med Image Anal 16:933–951
28. Hullermeier E (2011) Fuzzy sets in machine learning and data mining. Appl Comput 156:387–406
29. Peng X, Chen B et al (2012) Self-tuning software systems through dynamic quality tradeoff and value-based feedback control loop. J Syst Softw 85:2707–2719
30. Dageville B, Dias K (2006b) Oracle's self tuning architecture and solutions. In: Bulletin of IEEE

2 ДОСЛІДЖЕННЯ ВПЛИВУ РІЗНИХ СТРАТЕГІЙ НАСТРОЙКИ ПРОДУКТИВНОСТІ БАЗИ ДАНИХ І ЕНЕРГОЕФЕКТИВНОСТІ

2.1 Концептуальний підхід до управління базою даних

2.1.1 Статистичний підхід для ранжирування параметрів настройки бази даних

Звукова статистична методологія для кількісного визначення впливу кожного параметра конфігурації і взаємодії між цими параметрами на продуктивність СУБД являє собою повну факторіальну конструкцію, в якій розглядається кожна комбінація вхідних значень параметрів конфігурації. Проте основною проблемою при застосуванні повного факторіала в СУБД є велика кількість параметрів конфігурації. Наприклад, PostgreSQL [1] має принаймні 100 параметрів конфігурації, і всі параметри мають кілька можливих значень. З огляду на робоче навантаження на запит, навіть якщо кожен параметр конфігурації приймає тільки 2 значення, необхідно виконати 2100 експериментів, щоб застосувати повний факторний дизайн для кожного запиту робочого навантаження, що не вважається можливим з точки зору часу і зусиль. Для усунення цієї проблеми, у багатьох випадках адміністратори баз даних покладаються на свій досвід і емпіричні правила для вибору відповідних параметрів конфігурації для настройки. Проте, оскільки евристика з досвіду і інтуїції часто використовується, час і зусилля можуть бути витрачені марно, щоб підвищити продуктивність, налаштувавши ті параметри, які можуть не мати істотного впливу на загальну продуктивність. Неправильні спроби настройки збільшують загальну вартість володіння [2-5]. Ранжування параметрів, що засновані на їх впливі на продуктивність системи, значно допоможе адміністраторам баз даних встановити пріоритети своїх завдань налаштування. Наскільки відомо, немає дослідження, яке статистично забезпечує ранжування параметрів конфігурації на основі їх впливу на продуктивність СУБД.

В цьому пункті буде представлений статистичний підхід, для визначення параметрів конфігурації бази даних (SARD), з використанням схеми методології PLACKETT & BURMAN (P & B) [6] на основі експериментів. Зокрема, SARD вирішує наступну проблему: з огляду на СУБД, набір параметрів конфігурації, діапазон значень для всіх параметрів і робоче навантаження на запит; знайти відносний рейтинг параметрів, заснований на їх впливі на продуктивність. Робоче навантаження може бути набором еталонних запитів, наприклад, запитів TPC, або може бути набором операторів мови обробки даних (DML), зібраних за певний проміжок часу з використанням інструментів профілювання. SARD можна застосовувати для відкидання малозначних параметрів настройки, які мають несуттєвий вплив або ніяк не впливають на продуктивність СУБД.

Головна ідея SARD - провести ряд експериментів, які забезпечують приблизну вибірку

всього простору пошуку. У кожному експерименті значення параметрів систематично змінюються в певному діапазоні допустимих значень. Кожен дослід демонструє, яким буде час відповіді на запит, якщо для настройки сервера бази даних використовується певна комбінація значень параметрів конфігурації. Подальший аналіз зібраних експериментальних даних використовується для оцінки впливу параметрів конфігурації на продуктивність системи. Для зменшення експоненціального числа експериментів, необхідних для застосування повного факторіала, SARD використовує дворівневу факторіальну методологію проектування P & B, засновану на наступних припущеннях: 1) стимулювання системи монотонно неубутними параметрами щодо їх крайніх значень викличе найбільшу відповідь для кожного параметра; і 2) необхідно враховувати лише одно- і двохфакторні параметри. Ухвалення методу проектування P & B допомагає скоротити необхідну кількість експериментів від експоненціального до лінійного.

SARD - це проект експериментального підходу. Основна мета проектування експериментів - зібрати максимальну інформацію про систему з мінімальними зусиллями [7]. Експерименти проводяться на основі даної специфікації для збору інформації про продуктивність системи. Подальший аналіз отриманих експериментальних даних використовується для визначення важливих факторів (параметрів) і наявності взаємодій між факторами. Найпростішою стратегією проектування для кількісної оцінки впливу всіх факторів і взаємодій є застосування повного факторіала, наприклад, ANOVA, в якому вимірюється реакція системи на всі можливі комбінації введення [7]. Однак для цього потрібно експоненціальне число експериментів по числу параметрів.

Для зменшення кількості експериментів, SARD робить кілька припущень. По-перше, для кожного параметра SARD враховує тільки 2 значення: мінімальне і максимальне. Інтуїція полягає в тому, що стимулювання системи входами при їх екстремальних значеннях викличе максимальний діапазон вихідних відповідей для кожного входу. Друге припущення полягає в тому, що викликана відповідь, така як загальний час виконання, є монотонною функцією значень вхідних параметрів. Третє припущення грунтується на принципі розрідженості ефектів: в системній відповіді переважають кілька основних факторів і взаємодії низького порядку; ефект взаємодії більш високого порядку на відповідь не є статистично значущим. Як наслідок, можна сміливо ігнорувати ефекти взаємодії більш високого порядку.

Виходячи з цих припущень, SARD використовує дворівневий факторний дизайн з ім'ям Plackett & Burman (P & B) design [6], який вимагає тільки лінійного числа експериментів в порівнянні з експоненціальним числом експериментів, необхідних повнофакторному дизайну. Для кожного експерименту конструкції P & B значення кожного параметра задається заданою матрицею проектування. У таблиці 2.1 наведено приклад матриці проектування, зображеної стовпцями 2-8. Кожному рядку матриці відповідає один експеримент. Кожна клітинка в матриці

вказує значення, яке буде використовуватися для параметра у відповідному експерименті. Запис в матриці або «+1», або «-1». «+1» відповідає значенню, яке трохи перевищує нормальний діапазон значень для цього параметра, а «-1» відповідає значенню, трохи меншому, ніж нормальний діапазон цього параметра. Значення «+1» і «-1» не обмежуються тільки числовими значеннями. Наприклад, для алгоритму заміни сторінки буфера значення «-1» може бути «RANDOM», а «+1» може бути «CLOCK». Експерименти проводяться шляхом завдання значень відповідно до матриці проектування, і час виконання запиту записується, як в останньому стовпчику таблиці 2.1. Чистий ефект кожного параметра оцінюється шляхом множення значення відповіді на відповідний «+1», або «-1» для кожного рядка і підсумовування значень по всіх рядках. Абсолютне значення чистого ефекту використовується для визначення відносної важливості цього параметра.

SARD використовує методологію проектування P & B для оцінки впливу параметрів конфігурації на продуктивність СУБД. SARD має три основні фази. На першому етапі SARD оцінює вплив P & B кожного параметра конфігурації на продуктивність СУБД для кожного запиту робочого навантаження. На другому етапі для кожного запиту параметри конфігурації ранжуються на основі відносної величини ефекту P & B. На третьому етапі ранжування параметрів конфігурації для всіх індивідуальних запитів об'єднується для визначення остаточного ранжування параметрів для всього робочого навантаження запиту. Для ілюстрації використовується таблиця 2.1 як приклад для запуску, де повинні бути ранжовані сім параметрів конфігурації A, B, C, D, E, F і G. Робоче навантаження складається з трьох запитів: Q1, Q2 і Q3. Останні три стовпці відповідають часу виконання запиту Q1, Q2 і Q3. Тепер детально обговоримо 3 етапи SARD в наступних підрозділах.

A. Фаза I: оцінка параметрів

Спочатку будується матриця проектування P & B, яка дає специфікацію значень, використовуваних для кожного параметра в кожному експерименті. Розмір проектної матриці залежить від кількості параметрів конфігурації, N. Базова матриця дизайну має X рядків і X - 1 стовпців, де X - наступний кратний 4 більше N, тобто $X = (\text{floor}(N/4)+1)*4$. Наприклад, якщо N = 5, то X = 8, а якщо N = 8, то X = 12. Значення X вказує кількість експериментів, які необхідно провести в SARD для збору даних для оцінки ефекту P & B. У розробці оптимальних багатофакторних експериментів [6], P & B рекомендували значення параметрів, «+1» або «-1», для експериментів X = 8, 12, 16, ..., 96, 100. SARD встановлює перший рядок матриці дизайна P & B на основі цих рекомендацій у відповідності зі значенням X. Решта рядків (X - 1) матриці проектування P & B створюються шляхом циклічного зсуву безпосередньо попередньому рядку. Всі записи X-го рядка матриці дизайну P & B встановлені на «-1». Стовпці 2-8 в перших восьми рядках (R1-R8) таблиці 2.1 дають базову матрицю дизайну P & B для N = 7.

Таблиця 2.1. Стовпці 2-8 і рядки R1-R16 утворюють матрицю проектування P & B для семи параметрів A, B, C, D, E, F і G. Останні три стовпці містять час виконання запитів Q1, Q2, і Q3. Рядки R1-R8 містять базову конструкторську матрицю P & B, а рядки R9-R16 необхідні, якщо використовується foldover [10].

	A	B	C	D	E	F	G	Час виконання		
								Q1	Q2	Q3
R1	+1	+1	+1	-1	+1	-1	-1	34	110	10.2
R2	-1	+1	+1	+1	-1	+1	-1	19	72	10.1
R3	-1	-1	+1	+1	+1	-1	+1	111	89	10.3
R4	+1	-1	-1	+1	+1	+1	-1	37	41	10.3
R5	-1	+1	-1	-1	+1	+1	+1	61	96	10.2
R6	+1	-1	+1	-1	-1	+1	+1	29	57	10.2
R7	+1	+1	-1	+1	-1	-1	+1	79	131	10.3
R8	-1	-1	-1	-1	-1	-1	-1	19	47	10.1
R9	-1	-1	-1	+1	-1	+1	+1	135	107	10.3
R10	+1	-1	-1	-1	+1	-1	+1	56	74	10.3
R11	+1	+1	-1	-1	-1	+1	-1	112	48	10.1
R12	-1	+1	+1	-1	-1	-1	+1	74	91	10.1
R13	+1	-1	+1	+1	-1	-1	-1	55	99	10.3
R14	-1	+1	-1	+1	+1	-1	-1	117	123	10.1
R15	-1	-1	+1	-1	+1	+1	-1	51	77	10.3
R16	+1	+1	+1	+1	+1	+1	+1	76	81	10.2

Поліпшенням, яке підвищує точність базової конструкції P & B, є дизайн P & B з foldover [8], що вимагає додаткових експериментів X. Ці додаткові рядки створюються шляхом зміни знака верхніх записів рядків X. (X + i) -й рядок формується шляхом зміни знака i-го рядка. Стовпці 2-8 в останніх восьми рядках (R9-R16) таблиці 2.1 призводять додаткові конструктивні матричні записи для перевертання для N = 7. Якщо N<(X - 1), означає, що кількість стовпців в матриці проектування P & B це більше, ніж кількість параметрів конфігурації. В цьому випадку додаткові (X - N - 1) останні стовпці матриці P & B розглядаються як манекени, просто ігноруються. Для кожного запиту робочого навантаження i-й експеримент проводиться шляхом завдання значень параметрів конфігурації відповідно з i-м рядком матриці проектування P & B. Ефект кожного параметра конфігурації обчислюється шляхом множення відповідних «+1» або «-1» цього параметра в i- му рядку матриці P & B на час виконання запиту і підсумовування продуктів по всіх рядках проектної матриці.

Для ілюстративного прикладу в таблиці 2.1 N=7, а значення X=8. Для базового випадку потрібно провести вісім експериментів. Якщо використовується foldover, необхідно провести 16 експериментів. Припустимо, що використовується функція foldover. Специфікація значень параметрів повинна використовуватися у всіх 16 експериментах, наведених в стовпцях 2-8 і рядках R1-R16 таблиці 2.1. Перший рядок (R1) матриці дизайну P & B в таблиці 2.1 копіюється з [6] у відповідності зі значенням X = 8. Для запиту Q1 виконується 16 експериментів і записується

час виконання, як показано в дев'ятому стовпчику таблиці 2.1. Таким же чином час виконання запитів Q2 і Q3 записується в 10-й і 11-й стовпці. Тепер чистий ефект першого параметра A для запиту Q1 обчислюється шляхом множення записів у другому стовпці на записи в дев'ятому стовпці і підсумовування по всіх 16 рядках (R1-R16). Для запиту Q1 чистий ефект параметра A оцінюється як:

$$\text{EffectA} = \text{abs}\left((+1 * 34) + (-1 * 19) + \ldots + (-1 * 51) + (+1 * 76)\right) = \text{abs}\,(-109) = 109 \quad (2.1)$$

Аналогічним чином, чистий ефект другого параметра B для запиту Q1 розраховується шляхом множення записів в третьому стовпці на записи в дев'ятому стовпці і підсумовування по всіх 16 рядках (R1-R16). Чистий ефект A для запиту Q2 обчислюється шляхом множення записів у другому стовпці на записи в десятому стовпці і підсумовування по всіх 16 рядках (R1-R16) і т. д. Чисті ефекти P & B всіх 7 параметрів для запитів Q1, Q2 і Q3 показані в таблиці 2.2.

Таблиця 2.2. Ефекти P & B для запитів Q1, Q2 і Q3 [10].

	A	B	C	D	E	F	G	stdev
Q1	109	79	167	193	21	25	177	136.4
Q2	61	161	9	143	39	185	109	123.3
Q3	0.40	0.80	0.00	0.40	0.40	0.00	0.40	0.44

Таблиця 2.3. Нормалізовані ефекти P & B по відношенню до максимального ефекту для запитів Q1 і Q2 [10].

	A	B	C	D	E	F	G
Q1	0.6	0.4	0.9	1	0.1	0.1	0.9
Q2	0.3	0.9	0.0	0.8	0.2	1.0	0.6

SARD також може визначати чутливість запиту до налаштування параметрів. Для кожного запиту SARD обчислює стандартне відхилення мережевих ефектів для всіх параметрів конфігурації. Незалежно від того, наскільки великі ефекти P & B, якщо стандартне відхилення ефектів дуже низьке, це означає, що всі ефекти практично однакові, на продуктивність запиту не вплине зміна налаштувань параметрів конфігурації. В цьому випадку SARD ігнорує рейтинг. Стандартне відхилення чистих P & B-ефектів параметрів для запитів Q1, Q2 і Q3 складає відповідно 136,4, 123,3 і 0,44, як зазначено в останньому стовпчику таблиці 2.2. Однак, оскільки стандартне відхилення P & B-ефектів параметрів конфігурації для запиту Q3 дуже невелике, SARD ігнорує ранжування параметрів для запиту Q3.

1) Деякі обговорення: SARD передбачає, що час виконання запиту є монотонною функцією значень для параметра, тому високі і низькі значення параметра необхідно ретельно вибирати.

Щоб дізнатися (а), чи є час виконання монотонною функцією значень для параметра і (б) високі і низькі значення, так що час виконання є монотонною функцією значень в цьому діапазоні, потрібно або досвід або більше експериментів. З огляду на велику кількість параметрів і велику кількість можливих значень для кожного параметра, це нетривіальна проблема. Тому, як і інші інструменти, система SARD, швидше за все, повинна надаватися постачальником СУБД і не може використовуватися в якості автономного стороннього інструменту, який може бути легко адаптований для різних СУБД.

SARD передбачає, що між параметрами мало взаємодії. Прямо зараз SARD може правильно подбати про взаємодії першого і другого порядку, використовуючи дизайн P & B з foldover. Однак для СУБД, де є велика кількість параметрів, це припущення не завжди виконується. Тому для обґрунтування результатів SARD необхідні знання експертів в галузі управління базами даних. Кількісне визначення впливу цього припущення є предметом майбутньої роботи над SARD.

B. Фаза II: ранжування параметрів для запиту

Після визначення впливу всіх параметрів конфігурації і чутливості запиту наступним кроком є ранжування параметрів. Якщо воно нечутливе до змін параметрів, рейтинг можна сміливо ігнорувати; на продуктивність нечутливого запиту не впливають зміни параметрів.

Таблиця 2.4. Ранжування параметрів конфігурації для запитів Q1 і Q2. Запит Q3 не включений, оскільки він не чутливий до налаштування [10].

Нормалізація до максимального ефекта							
	A	B	C	D	E	F	G
Q1	4	5	3	1	7	7	3
Q2	5	2	7	3	6	1	4

Щоб ранжувати параметри конфігурації, можна просто впорядкувати їх на основі спадного порядку величини ефектів P & B. Однак, проблеми можуть виникнути, якщо деякі ефекти дуже близькі один до одного. Інтуїтивно ця ситуація означає, що ефекти відповідних параметрів практично однакові, але метод сортування привласнює їх іншому порядку ранжирування. Наприклад, існує чотири параметри P, Q, R і S, а ефекти P & B - 1500, 50,6, 51,4 і 3000 відповідно. Ранжування за методом сортування дорівнюватиме 2, 4, 3 і 1. Однак ефекти параметрів Q і R дуже близькі. Оскільки ефекти схожі, вони повинні бути присвоєні одному і тому ж рангу. Щоб уникнути цієї проблеми простого методу сортування, ефекти нормалізуються щодо максимального ефекту, округлюються до першого десяткового дробу і сортуються в порядку убування. Всі параметри з однаковим нормалізованим ефектом присвоюються однаковому рангу. Наприклад, в попередньому прикладі нормовані ефекти параметрів P, Q, R і S дорівнюють відповідно 0,5, 0,0, 0,0 і 1,0. Згідно з методом округлення, ряди складають 2, 4, 4 і 1 відповідно. У

слідуючому прикладі нормалізований ефект P & B для запитів Q1 і Q2 наведено в таблиці 2.3, а рейтинг через округлення наведено в таблиці 2.4.

C. Фаза III: ранжування параметрів для робочого навантаження

Після визначення ранжирування параметрів для кожного окремого запиту наступним кроком є оцінка ранжирування параметрів для всього робочого навантаження. Запити, які нечутливі до налаштування параметрів, не включені в розрахунок ранжирування робочого навантаження.

Щоб оцінити ранг параметрів конфігурації для всього робочого навантаження, ранги підсумовуються по всіх запитах, усереднені і відсортовані в порядку зростання. Найважливіші параметри будуть мати найнижчий кумулятивний ранг. Наприклад, в таблиці 2.4 для всього робочого навантаження середні ранжирування параметрів A, B, C, D, E, F і G складають відповідно 4,5, 3,5, 5,0, 2,0, 6,5, 4,0 і 3,5. Остаточний рейтинг для параметрів робочого навантаження в таблиці 2.1 буде дорівнювати 5, 3, 6, 1, 7, 4 і 3 відповідно. Ранжування показує, що P4 є найбільш важливим параметром, P2 і P7 є другими найбільш важливими параметрами, за якими слідують P6, P1, P3 і P5 по порядку.

Всі експерименти було проведено на машині з двома процесорами Intel XEON 2.0 ГГц w/HT, 2 ГБ оперативної пам'яті і 74 ГБ диском на 10 000 об / хв.

A. Робоче навантаження

Для демонстрації використовувалося робоче навантаження, що складається з п'яти запитів TPC-H, {Q1, Q8, Q9, Q13, Q16}. База даних TPC-H заповнюється програмою генерації даних dbgen з коефіцієнтом масштабування (SF) 1 (розмір даних становить 1 ГБ). Запити створюються за допомогою програми qgen, що поставляється з еталоном TPC-H. Для збору даних запити змінюються шляхом додавання EXPLAIN ANALYZE. Запити виконуються по одному. Більш докладні експериментальні результати з використанням іншого робочого навантаження і повних запитів TPC-H можна знайти в [9].

B. Розглянуті параметри

Для демонстрації використовувався PostgreSql8.2, який має приблизно 100 конфігураційних параметрів [1]. Оскільки запити доступні тільки для читання, багато параметрів не актуальні. Були розглянуті лише ті параметри, які, мабуть, мають відношення до запитів тільки для читання. Також навмисно був обраний тайм-аут контрольної точки параметрів і fsync, які не впливають на запити тільки для читання. Інтуїтивно відносний вплив цього типу параметрів має бути низьким або нульовим для запитів тільки для читання. Високі і низькі значення P & B для параметрів, використовуваних в цій демонстрації, наведені в таблиці 2.5. Всі значення вибираються відповідно до рекомендацій, зроблених в документації PostgreSQL [1]. Для демонстрації вибираються високі і низькі значення для кожного параметра в діапазоні, щоб він виконував монотонність.

У таблиці 2.6 наведені оцінки P & B SARD і ранжування параметрів конфігурації для запитів робочого навантаження, що складаються з TPC-H, {Q1, Q8, Q9, Q13, Q16}. Результати показують, що для Q1 work_mem є найбільш важливим параметром; для Q8 найбільш важливими параметрами є work_mem і maintenance_work_mem; для Q9 random_page_cost, shared_buffers, maintenance_work_mem є найбільш важливими параметрами; для Q13 effective_cache_size, shared_buffers є найбільш важливими параметрами; і для Q16 важливими параметрами є work_mem і shared_buffers. Work_mem займає перше місце в Q1, Q8 і Q9. Maintenance_work_mem займає друге і третє місця відповідно для Q8 і Q9. Shared_buffer займає третє, друге і друге значення відповідно в Q9, Q13 і Q16. Effective_cache_size оцінюється першим в Q3.

Таблиця 2.5. Параметри конфігурації PostgreSQL і їх значення P & B, використовувані в експериментальній установці. Символ * вказує, що значення відносяться до однієї послідовної вартості вибірки сторінок [10].

Параметр	Високе значення	Низьке значення
Effective_cache_size (pages)	81920	8192
Maintenance_work_mem (pages)	8192	1024
Shared_buffers (pages)	16384	1024
Temp_buffers (pages)	8192	1024
Work_mem (KB)	8192	1024
Random_page_cost∗	2.0	5.0
Cpu_tuple_cost∗	0.01	0.03
Cpu_index_tuple_cost∗	0.001	0.003
Cpu_operator_cost∗	0.0025	0.0075
Checkpoint_timeout (seconds)	1800	60
Deadlock_timeout (milliseconds)	60000	100
Max_connections	5	100
fsync	true	false
geqo	true	false
Stats_start_collector	false	true

Таблиця 2.6. Ефекти P & B параметрів конфігурації для робочого навантаження складаються із запитів TPC-H {Q1, Q8, Q9, Q13, Q16} [10].

Параметр	P&B ефекти					Ранжирування				
	Q1	Q8	Q9	Q13	Q16	Q1	Q8	Q9	Q13	Q16
Checkpoint_timeout	63139	8910	602068	4130	1590	15	15	5	15	15
Cpu_index_tuple_cost	10651	42106	149179	2460	1697	15	12	13	15	15
Cpu_operator_cost	81071	46111	349389	480	1714	15	12	11	15	15
Cpu_tuple_cost	58019	74791	48262	5808	1836	15	6	15	15	15
Deadlock_timeout	11923	64500	54715	5214	1546	15	6	15	15	15
Effective_cache_size	75846	11954	304263	238544	979	15	15	11	1	15
fsync	59809	4534	274194	3424	1329	15	15	11	15	15

geqo	3858	82480	529011	853	1679	15	6	7	15	15
Maintenance_work_mem	75774	127031	780976	1351	1048	15	2	3	15	15
Max_connections	97626	77876	628112	3678	1792	15	6	5	15	15
Random_page_cost	8693	43699	1694532	3447	2145	15	12	1	15	15
Shared_buffers	162386	34787	770430	132957	5997	15	12	3	2	2
Stats_start_collector	71420	25181	328104	4278	2122	15	12	11	15	15
Temp_buffers	101473	31217	571133	2713	1238	15	12	7	15	15
Work_mem	5523703	359544	142035	24839	63760	1	1	13	3	1

Таблиця 2.7. Остаточне ранжування параметрів конфігурації для робочого навантаження складається з запитів TPC-H {Q1, Q8, Q9, Q13, Q16} [10].

Ранг	Параметр
1	Work_mem
2	Shared_buffers
3	Maintenance_work_mem
4	Max_connections
5	Effective_cache_size
6	geqo
7	Random_page_cost
8	Checkpoint_timeout
9	Temp_buffers
10	Cpu_tuple_cost
11	Deadlock_timeout
12	Cpu_operator_cost
13	Stats_start_collector
14	Cpu_index_tuple_cost
15	fsync

Припускається, що всі запити мають однакову важливість в робочому навантаженні. Таким чином, інтуїтивно здається, що work_mem є однією з найважливіших конфігурацій для настройки підвищення продуктивності всього робочого навантаження, оскільки вона виглядає як ранжирувана для трьох запитів з п'яти. Shared_buffer також дуже важливий для настройки, оскільки він займає друге місце для двох запитів і третє для іншого запиту. Effective_cache_size і random_page_cost також виглядають багатообіцяючими, оскільки вони ранжуються першими для одного запиту. У таблиці 2.7 наведено загальний рейтинг всіх параметрів конфігурації для робочого навантаження, згенерованого SARD. З результату цікаво відзначити, що загальний рейтинг параметрів конфігурації для робочого навантаження відрізняється від їх ранжування для окремих запитів робочого навантаження. Наприклад, хоча випадкова вартість сторінки спочатку займає перше місце для запиту Q9, але вона не відображається в п'яти найбільш важливих параметрах для даного робочого навантаження. Ще одне цікаве спостереження полягає в тому, що тайм-аут fsync і checkpoint ніколи не представляється найбільш важливим для окремих запитів, а також для всього робочого навантаження запиту. Оскільки запити, які розглядаються тут, тільки

читаються, інтуїтивно ці два параметри повинні мати незначний вплив для підвищення продуктивності спільного робочого навантаження запиту. Експериментальні результати відповідають інтуїції. Низький рейтинг цих двох параметрів допомагає перевірити правильність роботи SARD.

2.1.2 Схеми індексування для самоналаштувальної СУБД

Сьогоднішні програми для корпоративних баз даних часто характеризуються більшим обсягом даних і високим попитом на час відповіді на запит і пропускну здатність транзакцій. Крім інвестування в нове потужне обладнання, настройка бази даних відіграє важливу роль у виконанні вимог. Проте, настройка бази даних вимагає глибоких знань про внутрішні компоненти системи, характеристики даних, додатки і робоче навантаження запитів. Зокрема, вибір індексу є основним завданням настройки. Тут проблема полягає в тому, щоб вирішити, як запити можуть підтримуватися шляхом створення індексів в певних стовпцях. Це вимагає балансу між вигодою індексу і втратами, викликаними споживанням простору і експлуатаційними витратами.

Хоча такі міркування довгий час були невід'ємною частиною проектування фізичної бази даних і добре вивчені, найчастіше це призводить до дуже високої складності і величезним зусиллям, необхідним для забезпечення ефективного вирішення для цього додатка. Протягом останнього року стала очевидною необхідність підтримки системи по цьому завданню.

Хоча дизайн індексу не дуже складний для невеликих або середніх схем і, скоріше, для статичних навантажень запитів, в сценаріях з аналітичним аналізом і безліччю спеціальних запитів, де необхідні індекси не можуть бути передбачені, це може бути досить складно. Конкретним прикладом сценарію є OLAP, де інструменти бізнес-аналітики та / або ROLAP створюють запити в результаті запиту інформації, ініційованого користувачем. Ці інструменти створюють послідовності операторів, включаючи створення і побудову таблиць, вставок і запитів [11].

Найостанніші випуски основних комерційних СУБД, таких як Oracle10g, IBM DB2 версії 8 і SQL Server 2000, забезпечують (обмежену) підтримку для цього сценарію. Вони включають так звані майстри індексів, які здатні аналізувати робоче навантаження (з точки зору витрат на раніше виконані запити) і - на основі деяких евристик - отримують рекомендації по створенню індексу. Це реалізовано з використанням віртуальних індексів, які фізично не створені, але враховуються тільки при оптимізації запитів в манері «що якщо». Хоча ці інструменти використовують інформацію про робоче навантаження, зібрану під час роботи, вони все ще працюють в режимі розробки. Це означає, що адміністратор бази даних повинен прийняти рішення про створення індексу, а створення індексу повністю відокремлено від обробки запитів. Недоліком цього підходу

щодо вищезазначених сценаріїв є те, що запити динамічно генеруються - в кінцевому підсумку на тимчасових таблицях - і тому аналіз автономного робочого навантаження досить скрутний.

Інший недолік радників по індексам пов'язаний з підходом до аналізу робочого навантаження один раз за певний період часу і створенням конфігурації статичного індексу, заснованої на цьому спостереженні. У багатьох додатках використання бази даних змінюється з часом, наприклад, протягом більш тривалих періодів через зміни в робочих процесах, з новими додатками, що працюють в одному наборі даних, або частими змінами, такими як сезонне використання або до кінця бізнес-кварталів. Крім того, зібране робоче навантаження, ймовірно, є неповним і упередженим через вплив збору робочого навантаження на повному обсязі оптимізовану базу даних. Крім того, необхідність зміни конфігурації індексу може бути викликана змінами використовуваного обладнання, такими як процесори, основна пам'ять і диски, що містять базу даних. Все це ставить задачу настройки індексу як безперервний процес з постійною необхідністю взаємодії з DBA. Індекс-консультанти не змінюють цього, вони просто зводять до мінімуму необхідний людський внесок експертних знань.

Спочатку представимо загальний процес виконання запитів побудови індексу. Основною метою підходу є поліпшення часу виконання (можливих майбутніх) запитів шляхом автоматичного створення корисних індексів. Оскільки створення індексів без обмежень може вичерпати обсяг пам'яті, доступний базі даних, припускається, що пул індексів - це індексний простір обмеженого розміру, що діє як постійний індексний кеш. Розмір цього пулу налаштовується адміністратором бази даних як системний параметр. Виходячи з цього припущення, запит обробляється наступним чином:

1) Для даного запиту Q визначаються потенційно корисні індекси;

2) Для запиту Q виводиться оптимізований з точки зору витрат план запиту;

3) Запит Q повторно оптимізований з використанням віртуальних індексів на основі кроку 1);

4) Прибуток наборів віртуальних індексів обчислюється як різниця витрат планів з кроку 2) і з кроку 3). Набір індексів з найбільшим прибутком називається рекомендацією індексу і використовується для поновлення глобальної конфігурації індексу, де зберігаються сукупний прибуток всіх індексів (як матеріалізованих, так і віртуальних). З цієї конфігурації індексу вирішується: створення індексів з набору віртуальних індексів і заміна інших індексів з пулу індексів, якщо для новостворених індексів недостатньо місця.

Вищенаведена дискусія по обробці запитів не враховує ряд важливих питань. По-перше, слід зазначити, що етапи 2) і 3) можна об'єднати. Оптимізатор, заснований на звичайному підході до динамічного програмування, може розглядати доступ до відносин через віртуальні індекси на першій ітерації. Єдиними необхідними модифікаціями алгоритму оптимізації є створення планів

доступу з віртуальними індексами, якщо був обраний оператор сканування таблиці, і не скорочувати план, якщо тільки плани з віртуальними індексами краще. Таким чином, результат етапу оптимізації складається принаймні з двох планів: план без віртуальних індексів і один або кілька планів з використанням віртуальних індексів.

Друге питання - це «особистий інтерес» до запиту. Якщо розглянути тільки кращий план, згенерований на кроці 3) – можна знайти тільки набір індексів, що сприяють поточному запиту Q, тому що необхідно максимізувати вигоду цього запиту (локальна оптимізація). Якщо розглянути всі індекси з кроку 2), які дають позитивний прибуток або, принаймні, без високих втрат – можна було б створити індекси, які, можливо, будуть корисні в майбутньому (для інших запитів). Однак, ця глобальна оптимізація вимагає розгляду більшого числа наборів індексів.

Нарешті, в припущенні обмеженого простору пулу індексів може знадобитися замінити існуючі індекси в пулі іншими індексами, якщо нові віртуальні індекси обіцяють більш високу вигоду, ніж старі. Для цієї мети можливі різні стратегії. Крім класичних стратегій заміни, які були розроблені за останні роки (наприклад, LRU, LFU), прибуток індексу може бути врахований. Однак, для цього потрібно зберегти статистику про глобальні прибутки, наприклад, шляхом моніторингу і акумуляції місцевого прибутку індексу для різних запитів.

Процедура розгляду заяв, описана вище, в деякій мірі підтримується поточними системами управління базами даних. Віртуальні індекси, існуючі, наприклад, в Oracle, дозволяють «as if» - запуск оптимізатора, як на етапі 3), перевіряти корисність індексів без їх матеріалізації. Спираючись на таку віртуальну оптимізацію, все одно потрібно знайти можливі застосовні набори індексів. Оптимізатор DB2 робить ще один крок вперед, надаючи рекомендації по індексу, що охоплюють більшість етапів 1) - 3).

Для роботи з витратами і перевагами індексів як частини автоматичного створення індексу треба розрізняти матеріалізовані і віртуальні (на даний час не матеріалізовані) індекси. Припускається, що статистика для обох типів індексів (віртуальних / матеріалізованих) обчислюється на вимогу: коли певний індекс розглядається вперше, виходить статистична інформація про це.

Набір індексів i1, . . . ,in, в якому використовуються для обробки запиту Q, називається індексний набір і позначається I. Безліч всіх віртуальних індексів I є virt (I), безліч всіх матеріалізованих індексів - mat (I). Нехай вартість (Q) - вартість виконання запиту Q, використовуючи тільки існуючі індекси і вартість (Q, I) - вартість обробки Q, використовуючи додаткові індекси з I. Потім прибуток I для обробки запиту Q просто

$$\text{profit}(Q, I) = \text{cost}(Q) - \text{cost}(Q, I) \qquad (2.2)$$

Очевидно, що якщо virt (I) = ∅, то прибуток (Q, I) = 0.

Щоб оцінити переваги створення певних індексів для інших запитів або вибрати серед кількох можливих індексів для матеріалізації, треба підтримувати інформацію про них. Таким чином, збирається набір всіх матеріалізованих і віртуальних індексів, розглянутих досі в каталозі індексів D = {i1 ,. ,,, Ік}. Тут для кожного індексу і зберігається наступна інформація:

• i.benefit - це перевага, тобто сукупний прибуток індексу;

• i.type ∈ {0, 1} позначає тип індексу з i.type = 1, якщо і матеріалізований, і 0 в іншому випадку;

• i.size - це розмір індексу, який оцінюється на основі типових параметрів, доступних як статистика баз даних, наприклад, розмір атрибута і кількість кортежів у відношенні.

Прибуток індексу, встановленого за запитом, може бути розрахований по-різному, як зазначено нижче. Однак, оскільки використовувалася система DB2 для оцінки, можна було б використовувати оптимізатор і рекомендовані віртуальні індекси. Для цілей оцінки використовувався наступний метод для вилучення оцінки вартості запитів для різних конфігурацій індексів:

1) обчислити витрати на запит без будь-яких індексів, крім індексів первинних ключів, через режим EXPLAIN;

2) вивести рекомендований індекс, встановлений в режимі RECOMMEND INDEXES;

3) обчислити вартість рекомендованого набору індексів.

Таким чином, можна не тільки отримати потенційний прибуток від набору індексів, а й режим радника оптимізатора DB2 також надає статистичну інформацію, таку як потужність і кількість листових вузлів, які дають точну оцінку розміру індексу, необхідного для представлених стратегій.

Підмножина D, що включає всі матеріалізовані індекси, називається індексною конфігурацією C = mat (D). Для такої конфігурації

$$\text{size}(C) = \sum_{i \in C} i.\text{size} \leq \text{MAX SIZE} \qquad (2.3)$$

тобто розмір зміни менше або дорівнює максимальному розміру пулу індексів.

Підтримуючи кумулятивну інформацію про прибутки та витрати за всіма можливими індексами, можна визначити оптимальну конфігурацію індексу для заданого (історичного) робочого навантаження запиту. Припускаючи, що це робоче навантаження також характерне для найближчого майбутнього, проблема створення індексу, в основному, полягає в пошуку конфігурації індексу Cnew, яка максимізує загальну вигоду:

$$\max \sum_{i \in Cnew} i.benefit \quad (2.4)$$

Це може бути досягнуто шляхом матеріалізації віртуальних індексів (в поточну конфігурацію) і / або заміни існуючих індексів. Щоб уникнути перекосу, заміна виконується тільки в тому випадку, якщо різниця між перевагою нової конфігурації Cnew і перевагою поточної конфігурації Ccurr вище заданого порогу. Тут перевага конфігурації обчислюється за допомогою переваги (C) = $\sum_{i \in C} i.benefit$. Крім того, треба враховувати побудову вартості нових індексів $cost_{build}(i)$, які виглядають як негативний прибуток:

$$benefit(Cnew) - benefit(Ccurr) - \sum_{i \in virt(Cnew)} cost_{build}(i) > MIN\_DIFF \quad (2.5)$$

Беручи до уваги сукупний прибуток індексу як критерію прийняття рішень про оптимальну конфігурацію індексу в глобальному масштабі, виникає проблема, пов'язана з історичними аспектами зібраної статистики. Припускаючи, що майбутні запити найбільш схожі на саме останнє робоче навантаження, оскільки зміни у використанні бази даних в середньостроковій або довгостроковій перспективі, статистика повинна представляти поточне робоче навантаження якомога точніше. Менш недавно зібрана статистика повинна мати менший вплив на створення індексів для майбутнього використання. Тому була застосована стратегія старіння для статистики сукупного прибутку на основі ідеї, представленої O'Neil et al. [12].

Як описано вище, легко вирішити, чи може запит локально отримувати вигоду з певної конфігурації індексу шляхом кількісної оцінки прибутку можливих комбінацій індексів з використанням віртуальної оптимізації. Щоб глобально вирішити оптимальну конфігурацію індексу для майбутніх запитів, інформація про можливі прибутки повинна збиратися, стискатися і підтримуватися, щоб найкращим чином представляти поточне робоче навантаження системи і, нарешті, на основі цієї інформації необхідно прийняти рішення, якщо конфігурація індексу може бути змінена в певний момент часу.

При обробці запиту Q статистика повинна оновлюватися шляхом додавання прибутку до кожного залученого індексу. На цьому етапі були розглянуті різні стратегії присвоєння прибутку кожному індексу, що бере участь. Одна з альтернатив пов'язана з тим, що можуть бути різні комбінації індексованих атрибутів, що приносять прибуток при віртуальній оптимізації. У цьому випадку можна або додати прибуток для всіх мінімальних наборів індексів Ii, що приносять прибуток, або отримати тільки прибуток для мінімального індексу, який є локально оптимальним, тобто дає найбільший прибуток, де індекс Ii мінімальний, якщо немає індексу Ij ⊂ Ii, що приносить такий же прибуток. У той час, як перший дає більш повну картину можливих досягнень певних конфігурацій індексів, останній вводить менше накладних витрат, тоді як при великих

навантаженнях все ще забезпечується розумна апроксимація переваг конфігурації.

До сих пір було зосередження на аналізі даного запиту і як зберегти статистику даних, зібраних в результаті віртуальної оптимізації. Тепер виникає питання: чи потрібно оновлювати матеріалізовану конфігурацію індексу? Якщо набір індексів можна замінити підмножиною Irepl ⊆ C = mat (D) поточної матеріалізованої конфігурації індексу, так що

$$\text{benefit}(C \cup I \backslash Irepl) - \text{benefit}(C) - \sum_{i \in virt\,(I)} \text{cost}_{build}\,(i) > \text{MIN\_DIFF} \wedge \text{size}(C \cup I \backslash Irepl) < \text{MAX\_SIZE} \quad (2.6)$$

тобто індекси в Irepl можна відкинути, а ті, що в I, можуть бути створені. Ці умови дозволяють тільки поліпшити конфігурацію індексу відповідно до поточного робочого навантаження і відповідати поставленим вимогам щодо простору покажчиків, а також критеріям, щоб уникнути перебоїв. Для вибору I з локально корисних наборів індексів було розглянуто дві стратегії: з позитивних наборів індексів вибирається тільки локально оптимальний набір індексів або перевіряються всі корисні набори індексів для можливої оптимальної глобальної конфігурації.

Набір індексів заміни Irepl обчислюється з використанням в даний час матеріалізованого набору індексів C = mat (D), застосовуючи жадібний підхід. Для цього сортується C по зростанню до критерію заміни і вибираються найменш вигідні індекси, поки не будуть виконані поставлені вимоги до простору. В якості критеріїв заміни було розглянуто кількість посилань на індекс, сукупний прибуток індексу і відношення прибутку до запиту або посилання на індекс. Тепер, якщо знайдений кандидат на заміщення значно менш вигідний, ніж набір індексів, який досліджується для можливої матеріалізації, конфігурація індексу може бути змінена під час виконання запиту, як описано чи заплановано пізніше.

Підхід для самоналагоджувальних індексних конфігурацій, описаний в попередньому розділі, забезпечує рішення для безперервної настройки на рівні конфігурацій індексів, де конфігурації являють собою набір загальних структур індексів.

Хоча індексування в базах даних з використанням похідних від BTrees, R-Trees, Grid Files, Tries і багатьох інших було успішно застосовано протягом багатьох років, очевидно, що ці структури індексів були задумані з урахуванням самонастроювання. Це відбувається з наступних причин.

Баланс даних, а не структура зі збалансованим доступом: з існуючими структурами даних індексів СУБД оптимізована для всіх існуючих значень певного, можливо, багатовимірного діапазону / домену. Це робиться для досягнення складності O (1) або O (log n) для доступу до єдиного блоку даних, тобто на сторінку, кортежу, об'єкту і т. д. Проте, це робиться без урахування того, що до вузла даних звертаються дуже часто, рідко, або зовсім немає. Отже, перевага структури індексу більше для доступу до даних частіше, ніж для доступу до даних рідше, хоча

обидва вони мають однакову ціну, тобто використовувані системні ресурси.

Груба гранулярність настройки: з існуючими структурами індексу це або все, або нічого: індекс може бути створений шляхом доступу чи ні. Хоча система може виграти, наприклад, від часткових індексів, запропонованих Stonebraker в [13] або неповних і розріджених індексів, це не реалізовано в більшості існуючих СУБД. Більш того, в порівнянні з розміром загальна користь від неповного індексу зростає з O (розмір журналу), тобто чим більше індекс зростає, тим більше зменшується прибуток Δbenefit / Δsize. З огляду на часто величезні вимоги до дискового простору, структури індексів, які підходять для самонастроювання, повинні мати можливість рости або навіть скорочуватися на вимогу.

Нерозуміння використання системних ресурсів: пов'язані з раніше згаданими аспектами, які використовуються в даний час, структури індексів не підходять для розгляду обмежень простору під час виконання. Хоча розміри індексу можна точно оцінити до створення, сам індекс не може відповідати обмеженням простору під час виконання, що треба було б для реалізації адаптивних змін індексів.

Щоб проілюструвати можливий підхід і отримати огляд всіх пов'язаних з цим проблем, було впроваджено просту структуру індексів на основі двійкового дерева, яке адаптується до його використання під час виконання.

Дерево прикладів показано на рисунку 2.1.

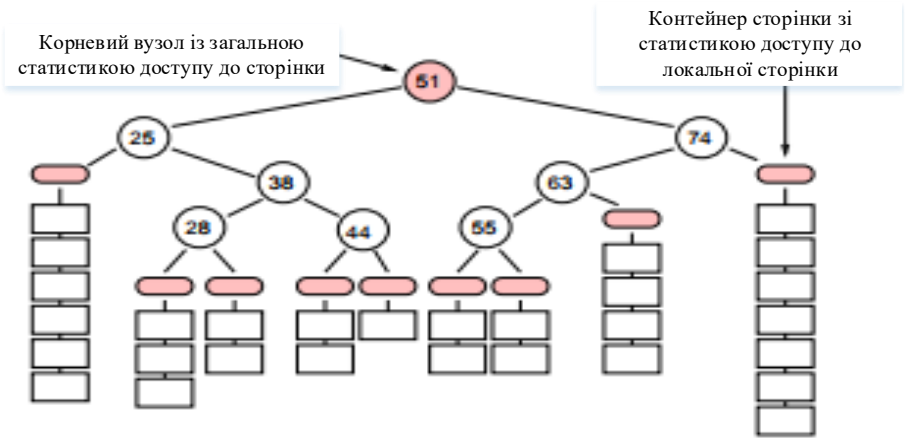


Рисунок 2.1 - Двійкове дерево з балансуванням доступу

У розрідженого індексу є контейнери сторінок як листові вузли, що містять сторінки кортежів з ключами діапазону значень, заданого деревом. Таким чином, в даний час дані організовані індексом. Наслідки для вторинних індексів і багатовимірних схем індексування обговорюються пізніше. Проте, поточний підхід дозволяє призначати системні ресурси дереву, збільшуючи або зменшуючи допустиму кількість вузлів всередині дерева. Пошук в дереві працює

як зазвичай, тоді як пошук в контейнерах сторінок вимагає послідовного сканування через сторінки.

Реорганізація і балансування на основі доступу можуть бути накидані тільки через обмеження простору. В принципі це працює наступним чином: кореневий вузол відстежує всі звернення до сторінки і поточний розмір дерева. Відповідно, він обчислює баланс

$$balance = \frac{all\ page\ accesses}{number\ of\ page\ containers} \tag{2.7}$$

для всіх контейнерів сторінок. Для контейнерів сторінок pc зберігається кількість всіх сторінок (pc). Якщо листовий вузол з двома контейнерами сторінок pc1 і pc2 виконує

$$\text{pageAccesses (pc1)} + \text{pageAccesses (pc2)} < \text{balance} \tag{2.8}$$

він віддаляється і додається в пул вільних вузлів, який підтримується як кількість вільних вузлів у корені дерева. Два контейнера сторінок об'єднані і пов'язані з попередником в дереві. Якщо число звернень до сторінок одного контейнера сторінки виконується

$$\text{pageAccesses (pc1)} > 2 * \text{balance} \tag{2.9}$$

і, крім того, в пулі вузлів є вільні вузли, новий вузол, вставлений до медіанного ключем з контейнера, і контейнер розділяється на 2 відповідно до нових контейнерів. Зростаюче дерево працює просто за рахунок збільшення вільного пулу вузлів і спираючись на раніше накидані алгоритми реорганізації. Скорочення може бути зроблено аналогічним чином, але також може примусово негайно відновити ресурси, видаливши вузли і об'єднавши їх контейнери сторінок, що мають найменший доступ до сторінок.

В даний час ці реорганізації, а також поновлення статистики виконуються під час операцій пошуку. З причин, обговорюваних пізніше, вони також можуть бути відкладені, щоб уникнути проблем з паралелізмом.

На основі цієї схеми реорганізації дерево може регулювати обмеження за розміром, а також використовувати дані, наприклад, дерево на рисунку 2.1 може індексувати дані з ключами, рівномірно розподіленими в діапазоні від 1 до 100, тоді як звернення зазвичай розподіляються із середнім значенням ключа 50, тобто мається набагато більше доступу в середині діапазону. Тому дерево глибше до цього діапазону і має більш дрібні контейнери сторінок для ефективного пошуку часто використовуваних даних.

Для експериментальної оцінки було протестовано аналогічний сценарій рівномірно

розподілених даних (100000 кортежів, близько 100 кортежів на сторінку, дерево від 1000 вузлів) і нормально розподілені точні зіставлення.

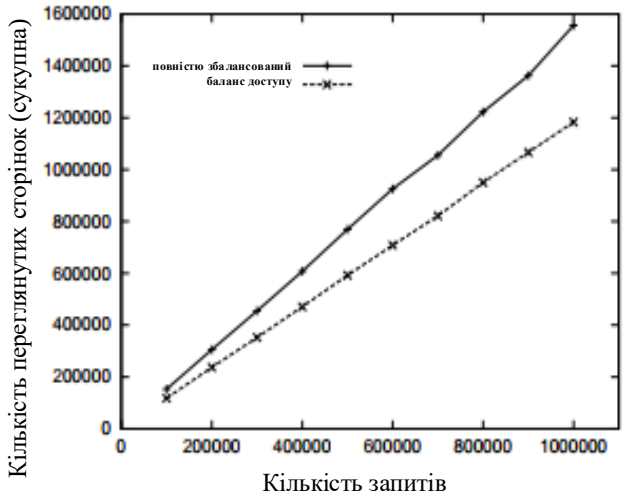


Рисунок 2.2 - Порівняння доступу до сторінок

На рисунку 2.2 показані результати доступу до сторінок. Оскільки недоцільно порівнювати представлений підхід з щільним індексом, замість цього порівнюється з альтернативною реалізацією дерева з фіксованим розміром без автоматичного балансування на основі доступу, який повністю збалансований з самого початку. Хоча в обох випадках складність одного пошуку залишається в порядку O (log n), дерево з розподіленим балансом перевершує повністю збалансоване дерево з лінійним коефіцієнтом, оскільки доступ до більш часто використовуваних даних більш ефективний.

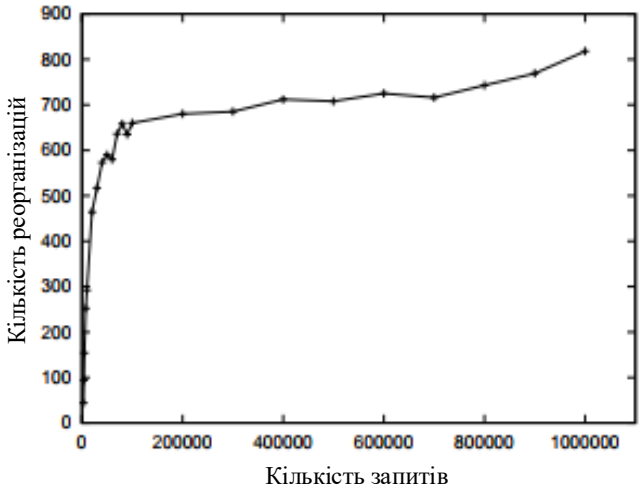


Рисунок 2.3 - Кількість реорганізацій дерев

На рисунку 2.3 ілюстрована кількість реорганізацій дерев, викликана кількістю точних

збігів. Після приблизно 100 000 пошуків дерево досягає стабільної структури з невеликою кількістю реорганізацій, необхідних згодом.

Замість того, щоб створювати індекси в конфігурації чи ні, тепер можна привласнювати простір індексам там, де це необхідно, в більш складній схемі індексування. І, як обговорювалося раніше, навіть невеликі індекси можуть принести величезну користь. Таким чином, краще мати невелику кількість дуже великих індексів, ніж наявність великої кількості цих невеликих індексів.

2.1.3 Самоналаштування динамічних ресурсів для робочих навантажень

Налаштування динамічних ресурсів - один з аспектів самоналаштувальної технології. Метою є автоматичне регулювання різних параметрів ресурсів для досягнення мети продуктивності при роботі системи.

Ресурси системи баз даних можна розділити на фізичні і логічні ресурси. Фізичні ресурси - це апаратні аспекти, а їх значення обмежені станом обладнання, таким як процесор, пам'ять, диск і т. д.; логічні ресурси надаються СУБД, а їх значення обмежені СУБД, такими як кількість робочих потоків, кількість блокувань і т. д. [15]. Фізичні і логічні ресурси є інтерактивними і взаємозалежними. Складна обробка запитів призведе до значної зміни вимог до пам'яті, процесору і диску. Самонастройка продуктивності СУБД стає складним завданням через складні взаємодії і різні ефекти на продуктивність системи серед цих ресурсів [16]. Взагалі кажучи, внутрішні алгоритми управляються деякими параметрами, що настроюються, наданими СУБД. Значення цих параметрів впливають на продуктивність цільової системи. При запуску системи самонастройка параметрів зі зміною різних робочих навантажень може в деякій мірі впливати на продуктивність системи.

В області самонастроювання продуктивності бази даних існує більше досліджень з автоматичного розподілу пам'яті. Спосіб самонастроювання пам'яті містить автоматичний розподіл пулу буферів, використання пам'яті в розподіленій системі, попередню вибірку даних і т. д.. В основному сфокусований алгоритм являє собою планування пулу буферів. Метою цих алгоритмів є поліпшення швидкості захоплення буфера.

Алгоритм, запропонований в [16], який автоматично регулює рівні мультипрограмування і розподіл пам'яті – для досягнення набору цілей часу відгуку для кожного класу для складних робочих навантажень в СУБД. Він використовує набір евристик і методів оцінки для управління механізмом зворотного зв'язку. Майстер налаштування буфера пулу описаний в [17] – для визначення ефективних розмірів пулу буферів. Майстер, заснований на самоналагоджувальному алгоритмі, званому алгоритмом динамічної реконфігурації (DRF), який використовує жадібні евристики – для пошуку перерозподілу, який приносить користь цільовому класу транзакцій.

Структура адаптації робочого навантаження пропонується в [18]. Menascé [19] описує підхід, в якому аналітичні моделі продуктивності поєднуються з комбінаторними методами пошуку для проектування контролерів, які періодично запускаються для визначення найкращої конфігурації системи з урахуванням її робочого навантаження. У систему вбудований новий підхід до самоконтролюючої комп'ютерної системи, механізми, необхідні для самонастроювання параметрів конфігурації, щоб постійно задовольнялися вимоги до якості обслуговування системи.

Механізм зворотного зв'язку може бути як фундаментальною архітектурою системи самонастроювання. У цьому евристичному механізмі основними компонентами є: монітор продуктивності, аналізатор продуктивності і оптимізатор продуктивності. Система починає працювати з визначеними параметрами конфігурації. Моніторний модуль постійно контролює різні дані про продуктивність цільової системи протягом періоду роботи і передає результати в модуль аналізатора. Якщо модуль аналізатора виявляє продуктивність цільової системи аж до визначених порогових значень, вона відправить деякі інструкції для деяких параметрів, які необхідно відрегулювати. Після настройки цих параметрів, якщо мета виконання не може бути виконана, система увійде в процес рециркуляції спостереження, аналізу та настройки [20].

Архітектура прототипу оптимізації найкраще описана на рисунку 2.4. Модулями структури є моніторинг продуктивності, оцінка продуктивності, діагностика ресурсів, набір правил знань, аналізатор продуктивності і регулятор ресурсів.

Монітор продуктивності динамічно контролює продуктивність цільової системи з мінливими робочими навантаженнями. Він виконує регулярні інтервали з виконанням алгоритму контролера і збирає використання ресурсів і стан продуктивності динамічних уявлень і словники даних в системі. Оцінка продуктивності оцінює продуктивність поточної системи. І вона дає інформацію про те, чи слід оптимізувати систему на основі принципу оціночної моделі. Діагностика ресурсів виявляє вузьке місце в ресурсах і вирішує, чи потрібно налаштовувати СУБД. Набір правил використовується для зберігання набору правил оптимізації продуктивності. Набір знань надає інформацію для модулів діагностики і аналізатора, в яких викладається план настройки параметрів ресурсу продуктивності. Аналізатор продуктивності вирішує, як налаштувати параметри і видає план оптимізації в допустимому діапазоні параметрів. Регулятор ресурсів автоматично налаштовує конкретні параметри ресурсів продуктивності відповідно до реальних значень, переданих модулем аналізатора.

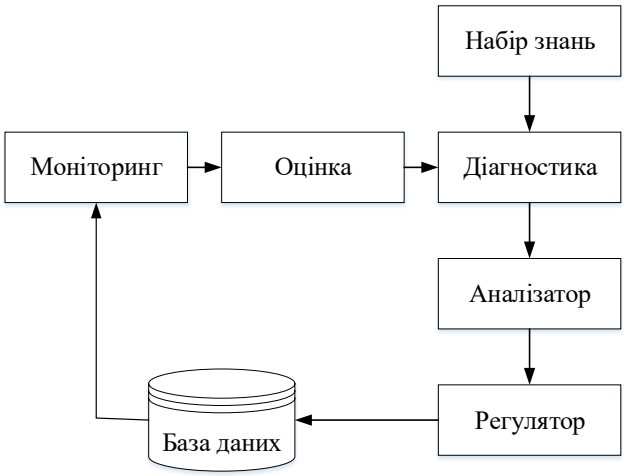


Рисунок 2.4 - Структура оптимізації

Основна робота по розробці оцінки - як оцінити ці індекси ресурсів. Щоб спростити роботу, вибираються основні ресурси, які суттєво впливають на продуктивність системи, включаючи пропускну здатність транзакцій, ставлення кеш-пам'яті, час відгуку, ставлення відмови запиту і коефіцієнт використання CPU. Був побудований індекс продуктивності бази даних для кількісної оцінки цих ресурсів. Індекс продуктивності бази даних показаний в таблиці 2.8.

Кожному індексу продуктивності дається набір балів, який представляє стан поточної продуктивності системи. Беручи за приклад коефіцієнт попадання в буферний кеш, конкретний принцип оцінки проілюстрований в таблиці 2.9.

Таблиця 2.8. Індекс продуктивності бази даних [21]

Показник	Індекс	Тип
Час відгуку	>85%	100
Запит відмови	>85%	100
Пропускна здатність транзакції	>85%	100
Коефіцієнт використання CPU	>85%	100
Коефіцієнт попадання кеша буфера	>95%	100
Коефіцієнт попадання кеша бібліотеки	>98%	100
Коефіцієнт попадання в кеш-пам’ять	>98%	100
Сортування по пам’яті	>98%	100
Коефіцієнт попадання буфера в журнал	>98%	100

Таблиця 2.9. Принцип оцінки [21]

Індекс	Тип
>98%	100
95-98%	80
90-94%	60
<90%	0

Взагалі кажучи, продуктивність краще, коли оцінка вище. Оцінка дорівнює нулю, і деякі

частини системи мають проблеми. Згідно з індексом продуктивності, можна було б отримати поточний стан системи з статистичної інформації і використовувати цю оціночну модель для визначення стану продуктивності системи. Цільова формула визначається як,

$$QOS = r1 * (1 - R0/Rmax) + r2 * (1 - P0/Pmax) + r3 * (1 - Xmin/X0) + r4 * (1 - CPUmin/CPU0) + r5 * (a1 * Rate1 + a2 * Rate2 + \ldots + an * Raten) \quad (2.10)$$

де ri (ri≥0 і r1 + r2 + r3 + r4 + r5 = 1) представляє вагу, яка вказує відносну важливість кожного показника продуктивності. Визначається R0 як середній час відгуку, Rmax - як найдовший, який допустимий, X0 - середня пропускна здатність системи, Xmin - як мінімальна допустима, P0 - середнє відхилення запиту, Pmax - максимально допустиме, CPU0 - як середнє відношення використання ЦП, CPUmin як мінімальне допустиме, Ratei як різні коефіцієнти попадання в кеш, ai як вага (ai≥0 і a1 + a2 + ... + an = 1). Показник QOS означає якість обслуговування цільової системи. R0, P0, X0, CPU0, Ratei можуть використовуватися в якості основних індексів для вимірювання коефіцієнта використання системних ресурсів.

Оскільки робочі навантаження постійно змінюються, продуктивність системи змінюється відповідно. Проблема, що виникає, повинна бути вирішена вчасно, тому алгоритм не може бути складним. Використовується алгоритм Hillclimbing для оптимізації параметрів продуктивності системи. H (h1, h2, h3) - векторна безліч, де h1, h2, h3 - регульовані параметри, які можуть впливати на продуктивність системи. Кожен вектор має максимальне і мінімальне значення. Значення «сусіда» H визначається як один з векторів hi, змінений на +1 або -1. Визначимо V = (h, h, h) як змінну векторну сукупність. У векторному безлічі V є тільки один h дорівнює 1, другий дорівнює 0. Беручи V = (1, 0, 0), наприклад, H + V означає, що це значення (h1 + 1, h2, h3). Пошук повторюється при кожному новому відвіданому векторі до тих пір, поки не будуть виконані вимоги QOS і кожен індекс продуктивності.

Як показано на рисунку нижче, час відгуку являє собою час виконання певних серій транзакцій, на сеансах присутні різні номери клієнтів, підключених до сервера, TXN є різна кількість виконаних транзакцій. Коли кількість сеансів або транзакцій невелика, час відповіді на виконання не відрізняється від і без санкціонування системи. Однак час відгуку швидко збільшується без самонастроювання системи. Але час контролюється під власним налаштуванням системи.

Як показують цифри 4 і 5, TPS являє транзакції в секунду. Він дає посилання на пропускну здатність системи. Різні сеанси, що виконуються з певною кількістю транзакцій, мають різні TPS. Значення TPS велике, так як кількість сеансів збільшується до максимальної можливості системи. При самоналаштуванні системи баз даних TPS вище, ніж без самоналаштування системи в міру

збільшення кількості сеансів. Коли кількість транзакцій невелика, TPS майже такий же, як і без самоналаштування системи. Однак, коли число транзакцій збільшується, TPS з самоналаштування системи збільшується швидше, ніж без самоналаштування системи. І TPS без самоналаштування системи зберігається на певному рівні невеликої вартості.

Після самоналаштування системи час відгуку зменшується, а TPS збільшується в кожній групі. Це означає, що продуктивність системи має деякі поліпшення завдяки самоналаштуванню ресурсів продуктивності.

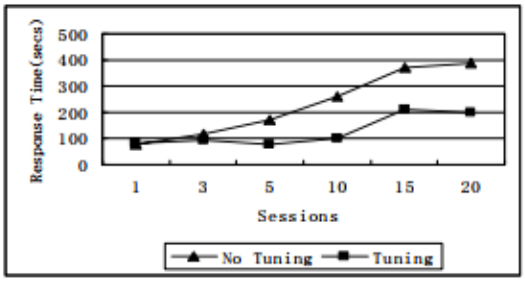


Рисунок 2.5 - Час відгуку для різних сеансів [21]

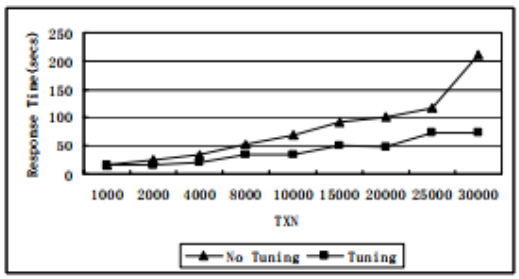


Рисунок 2.6 - Час відгуку для різних транзакцій [21]

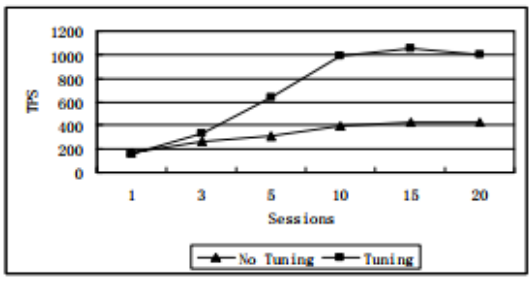


Рисунок 2.7 - TPS для різних сеансів [21]

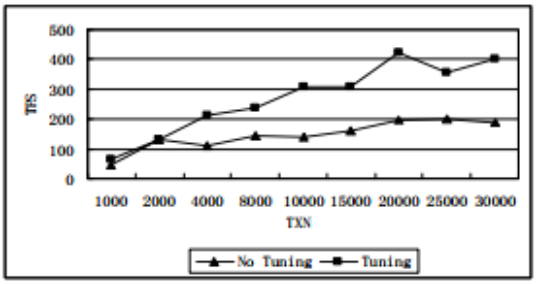


Рисунок 2.8 - TPS для різних транзакцій [21]

2.2 Аналіз методу, заснованого на адаптивній нейро-нечіткій технології для настройки продуктивності систем управління базами даних

Зростаючі витрати на енергію в великих центрах обробки даних є рушійною силою для енергозберігаючих обчислень. Точно так же, як продуктивність грає центральну роль в оцінці систем, швидко зростає споживання енергії, що важливо для мінімізації витрат на IT. Тому метою являється краще розуміння енергетичних характеристик систем баз даних на сучасному обладнанні.

Основні внески полягають в наступному:

1) Докладне дослідження профілів потужності для основних операторів бази даних на сучасному масштабованому обладнанні. На відміну від інших контекстів [22, 23], було виявлено, що потужність ЦП не змінюється лінійно з використанням ЦП, а використання - поганий проксі для потужності ЦП;

2) Ретельне дослідження впливу апаратних і програмних регуляторів на енергоефективність складних запитів в двох широко використовуваних двигунах: PostgreSQL і System-X1;

3) Майже у всіх випадках було виявлено, що на відміну від попередніх досліджень, найбільш ефективна конфігурація є найбільш енергоефективною.

Щоб оцінити можливості для оптимізації, необхідно почати з вимірювання потужності компонентів системи від холостого до повного використання. На рисунку 2.9 показано розмикання живлення однієї конфігурації 8-ядерного (двопроцесорного) тестового апарату. Важливим фактором для будь-якого дослідження ефективності використання енергії є використовувана апаратна настройка.

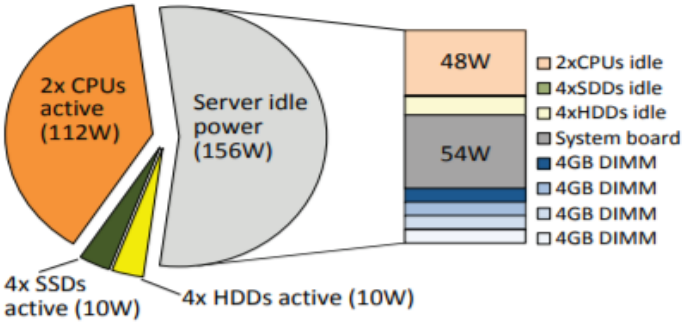


Рисунок 2.9 - Порушення потужності випробувальної машини [37]

Права половина кругової діаграми - це споживання енергії на холостому ходу і становить близько половини пікової потужності. Як показує розбивка, трьома основними компонентами, які сприяють убутку, є бездіяльність двох процесорів, фіксована потужність ОЗУ і інші компоненти системної плати. Проста потужність жорстких дисків відносно невелика, а споживання SSD близько до нуля. Ліва половина кругової діаграми показує додаткову потужність, споживану при повному використанні всіх ЦП і дисків. Однак твердотільні накопичувачі і жорсткі диски використовують аналогічну потужність, однак домінуючими компонентами є два процесори (+ 112 Вт).

Найбільшим споживачем енергії в системі є процесори. На рисунку 2.10 показано загальне споживання потужності ЦП для декількох операторів, так як змінюється кількість використовуваних сердечників, від 1 до 8 (кожне ядро використовується на 100%, точка 0 - це бездіяльна потужність). Рисунок 2.10 (а) відповідає політиці планування процесу, орієнтованої на продуктивність, тоді як на рисунку 2 (б) використовується політика економії енергії.

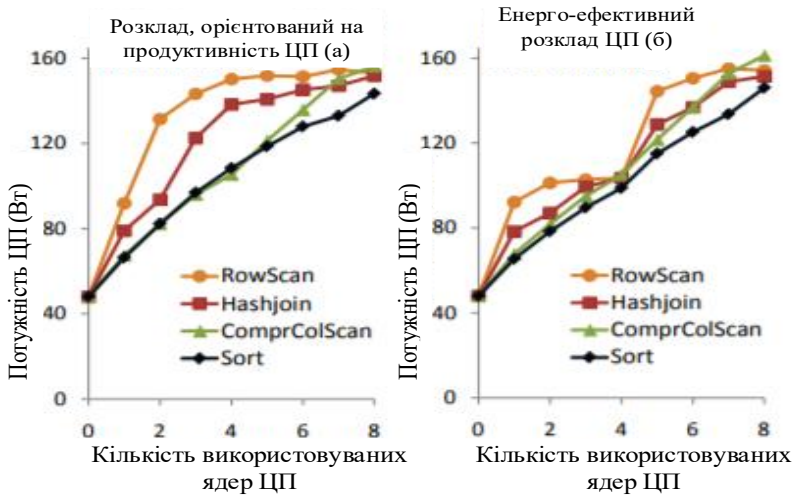


Рисунок 2.10 - Потужність центрального процесора проти використання для операторів бази даних

Дивно, але різні оператори можуть змінюватися більш ніж на 60% при споживанні енергії (наприклад, сортування і сканування рядків з використанням двох ядер).

Визначається енергоефективність як відношення корисної роботи до використовуваної енергії, яка збігається з відношенням продуктивності до потужності:

$$EE = \frac{\text{Work done}}{\text{Energy}} = \frac{\text{Work done}}{\text{Power} \times \text{Time}} = \frac{\text{Perf}}{\text{Power}} \quad (2.11)$$

Оскільки програмне забезпечення бази даних багате налаштованими параметрами, від констант системного рівня до планування і виконання запитів, а експерименти досі вказують на кілька параметрів, які можуть вплинути на використання енергії, ці параметри можуть потенційно вплинути на енергоефективність. Найбільш перспективними кнопками є ті, які можуть безпосередньо торгувати циклами ЦП для часу доступу до диска, оскільки це два ресурси зі значно різними профілями використання живлення. Такі компроміси існують в методах доступу (послідовне сканування в порівнянні з кластерним і некластерізованим скануванням індексів), орієнтованих на стовпці і доступ до запису по рядках, методи стиснення (легкий і великоваговий) і алгоритми об'єднання. Недавня робота припустила, що деякі з цих регуляторів можуть бути багатообіцяючими [24, 25], а інші показали конкретні випадки, коли ефективність використання енергії поліпшується [26-28] за рахунок продуктивності.

На рисунку 2.11 показана ефективність енергоспоживання запиту 5 TPC-H в базі даних System-X для 128 різних конфігурацій. Цей графік є параметричним графом, в якому обидві осі - енергетична ефективність (вісь y) і продуктивність (по осі x) - є залежними змінними; вони обидві залежать від конфігурації. Енергоефективність йде рука об руку з продуктивністю. Майже завжди є конфігурація, яка може ще більше підвищити енергоефективність, просто покращуючи продуктивність. Навіть для точок, близьких до пікових характеристик, їх енергоефективність складає менше 10%.

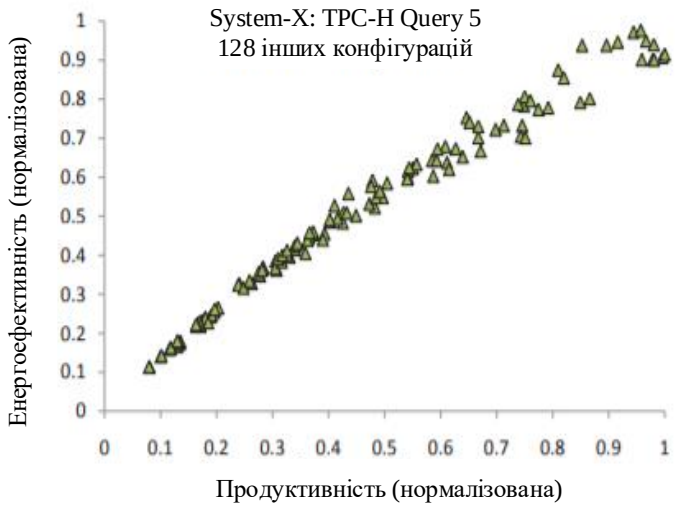


Рисунок 2.11 - Ефективність використання енергії проти продуктивності System-X

Отримані результати безпосередньо контрастують з недавньою роботою [27,28], яка передбачає, що енергоефективність і продуктивність часто є двома різними цілями оптимізації.

Енергоефективність обчислювального обладнання в центрах обробки даних є важливим завданням з кількох причин. По-перше, витрати на живлення та охолодження починають обганяти вартість обладнання [29]. По-друге, збільшення використання енергії має негативні наслідки для щільності, надійності і масштабованості в центрі обробки даних. Нарешті, збільшення використання енергії в центрах обробки даних викликало екологічні проблеми, з якими провідні уряди в усьому світі прагнуть регулювати потужність IT підприємства. З цих причин бачимо зрушення в промисловості і дослідженнях в сторону оптимізації енергоефективності.

Цей зсув широкий: від чіпів до центрів обробки даних. Розробники чіпів розглянули безліч технологій енергозбереження, таких як динамічне масштабування частоти і напруги (DVFS), оптимізація маршрутизації тактових імпульсів, асиметричні багатоядерні процесори і т. д. Системні архітектори запропонували стратегії для динамічного управління станами потужності DRAM, управлінням швидкістю диска або обертовими дисками. Тепер бачимо нові енергоефективні редизайн платформи, які відповідають SLA продуктивності невеликого, але важливого класу робочих навантажень, наприклад веб-серверів і аналізу даних [29-31]. Нарешті, в недавній роботі пропонується оптимізація на рівні ансамблю, наприклад, переміщення робочих навантажень для задоволення обмежень потужності і температури або консолідація для поліпшення використання енергії [32]. Архітектори центрів обробки даних досліджують цілісний редизайн, який обробляє центр обробки даних як окремий комп'ютер [33].

З цією метою деякі з них закликають до пропорційного використання енергії [34]. Абсолютно пропорційний компонент не використовує живлення, коли воно не використовується, і використовує тільки потужність, пропорційну її продуктивності. Оскільки енергоефективність - це

співвідношення продуктивності і потужності, пропорційне обладнання забезпечує постійну енергоефективність при всіх режимах продуктивності. При такому обладнанні не потрібно використовувати методи більш високого рівня для адаптації до найбільш ефективної точки. Конструктори апаратного забезпечення забезпечують максимальну ефективність, і розробники програмного забезпечення продовжують турбуватися про продуктивність.

Сьогодні, однак, компоненти навряд чи пропорційні енергії, але їх динамічний діапазон потужності і пропорційність неухильно поліпшуються. Спочатку постачальники процесорів представили малопотужні версії своїх процесорів для мобільного ринку. Комерційні продукти вже включають в себе гачки для управління статичними станами DRAM, а нові контролери пам'яті знаходяться на шляху, який динамічно регулюють ці стани. Існують також нові типи незалежної пам'яті, такі як PCRAM [35] і memristor [36], які хочуть замінити DRAM в цілому. Для дисків бачимо нові диски з декількома швидкостями віджиму, а також більш потужні приводи з великою кількістю станів сну. Нарешті, SSD знаходяться на шляху до заміни жорстких дисків в контекстах, де ємність не є обмежуючим фактором.

Використовуючи ці нові компоненти, все ще залишається питання про те, як найкращим чином зібрати їх в більш великі енергозберігаючі системи.

Поточна установка з одним вузлом використовує сучасні (з 2009 року) процесори, DRAM і SSD. Ця настройка вбудована в майбутні тенденції в компонентах: процесор має кілька активних станів потужності, а SSD - найсучасніші. Крім того, система збалансована для продуктивності; диски, коли вони повністю використовуються, можуть підтримувати всі ядра в простих операціях, таких як сортування і змішування.

Використовуваний сервер бази даних використовує як корпоративні жорсткі диски (HDD), так і твердотільні накопичувачі (SSD). На відміну від пам'яті, ці компоненти демонструють вищу мінливість потужності, що відповідає за 15% від загальної активної потужності. Однією із загальних характеристик в обох технологіях зберігання є наявність двох робочих станів, вільних і активних, кожен з яких споживає різну потужність.

Для вивчення пропорційності потужності твердотільних накопичувачів і жорстких дисків використовується мінімальне ядро зберігання рядків / сховища рядків (7), яке емулює механізм зберігання бази даних на основі рядків або стовпців, а також може виконувати просту оцінку та його узагальнення предикатів. Ядро використовує пряме введення-виведення для зменшення накладних витрат ЦП при доступі до блоків даних і асинхронному IO для підвищення пропускної здатності пристрою.

Щоб виміряти енергоспоживання SSD в залежності від використання пристрою, було сконфігуровано ці пристрої в RAID-0 (чергування), і використовувалося ядро для послідовного читання 100 ГБ файлу; розмір блоку був встановлений на рівні 128 КБ. Утилізація пристрою

розраховувалася як відношення виміряної до максимальної пропускної здатності. Предикати підвищеної складності поступово застосовувалися для введення службових даних ЦП, які ефективно зменшували виміряну пропускну здатність. На рисунку 2.12 описано використання енергії чотирма SSD-пристроями в залежності від використання пристрою. Як видно на рисунку 2.12, твердотільні накопичувачі демонструють ідеальну пропорційність енергії; вони споживають майже нічого без навантаження і демонструють лінійну потужність при додатковому навантаженні.

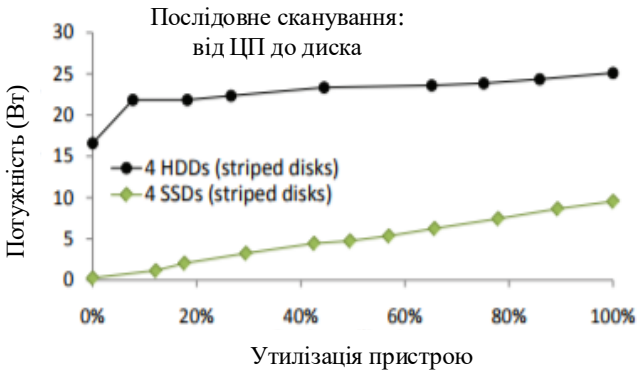


Рисунок 2.12 - Енергоспоживання SSD та HDD як функції використання пристрою

На рисунку 2.12 також показаний профіль потужності жорстких дисків, який вимірюється за використанням тієї ж процедури. Жорсткі диски явно не пропорційні енергії, тому що вони споживають майже половину потужності в режимі очікування і приблизно 80% максимальної потужності, коли пристрій стає активним. Первісна вартість енергії відбувається від обертання диска, і припускається, що другий стрибок відбувається від використання додаткових схем і кешей. Після цього жорсткі диски демонструють лінійне збільшення з використанням. Погана пропорційність приводів може значно знизити енергоефективність, особливо коли ці пристрої недовикористовуються.

Цікаво, що на процесори припадає 85% збільшення потужності при завантаженні в системі. Було запропоновано кілька моделей потужності, які припускають лінійну кореляцію між використанням ЦП і споживанням енергії. Однак лінійні моделі для повної потужності ЦП є хорошими апроксимаціями при дуже специфічних умовах: а) операції пов'язані з ЦП, б) немає загальних ресурсів між ядрами ЦП і в) не застосовуються методи управління живленням.

Однак ці умови не виконуються для операцій з базою даних, оскільки вони не завжди пов'язані з ЦП. Крім того, в сучасних багатоядерних процесорах елементи обробки (ядра) зазвичай мають кілька ресурсів, таких як частина ієрархії кеша і шина пам'яті. Крім того, зазвичай застосовуються апаратні і програмні методи управління живленням. Сучасні процесори динамічно

змінюють частоту процесора в залежності від застосовуваного робочого навантаження. На рівні програмного забезпечення операційні системи використовують енергозберігаючі методи планування, які мінімізують споживання енергії ЦП.

Незважаючи на те, що системи баз даних сумно відомі великою кількістю параметрів, що настроюються, вибираються ті, які потенційно можуть надати найбільший вплив на енергоефективність. Зокрема, треба змінити наступні «ручки» на рівні бази даних: а) вибір алгоритму / плану, б) внутрішньооператорский паралелізм (# ядер з одним оператором), в) паралелізм між запитами (кількість незалежних запитів, що виконуються паралельно), г) фізична компоновка (сканування рядків і стовпців), д) макет сховища (чергування) і е) вибір носія даних (HDD проти SDD). Також треба змінити регулятори рівня платформи: перш за все планується політика і настройка частоти.

Для того, щоб знайти плани, спрямовані на підвищення ефективності використання енергії, необхідно буде оптимізувати оптимізатори для моделювання витрат енергії і перепризначення їх критеріїв вибору плану.

Методи доступу є основними будівельними блоками кожного плану запитів і в багатьох випадках визначають продуктивність запиту. Тому, природно, починається з вивчення того, як різні методи доступу впливають на ефективність використання енергії.

Для всіх непаралельних запитів (одиночний процес з використанням тільки одного ядра) варіації потужності процесора від різних одноядерних додатків в поєднанні зі змінами в потужності зберігання не переміщують голку в порівнянні з фіксованими витратами на електроенергію системи. Ці результати показують, що в цьому випадку немає необхідності міняти оптимізатор.

На рисунку 2.13 показана ефективність використання енергії та продуктивність кешуючого хеш-з'єднання, оскільки змінюється кількість використовуваних сердечників. Лівий і правий графіки обчислюють енергоефективність, використовуючи тільки потужність процесора і загальну потужність системи, відповідно. На правому графіку показана сильна лінійна кореляція між енергоефективністю і продуктивністю, а для точок поблизу будь-якого заданого рівня продуктивності ефективність практично не змінюється.

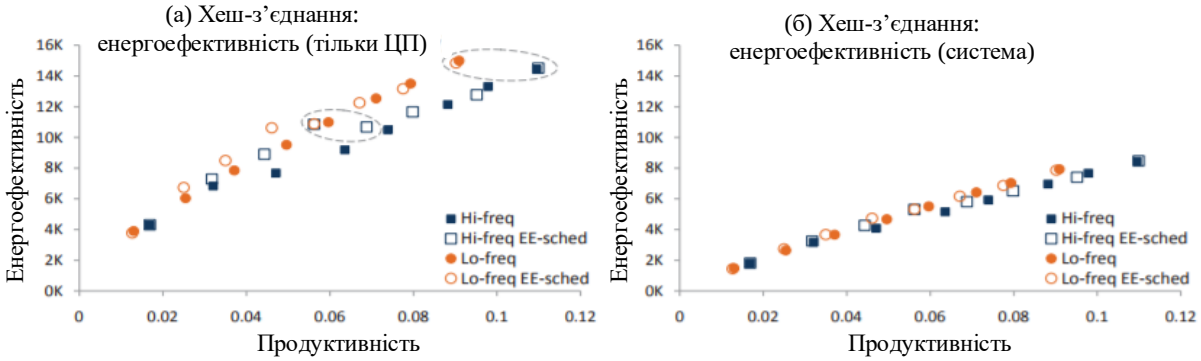


Рисунок 2.13 - Ефективність використання енергії порівняно з продуктивністю для паралельного входу в хеш-пам'ять

На лівому графіку показана крива енергоефективності, яка включає тільки потужність процесора. Знову ж таки, бачимо слабший, але все ж лінійний зв'язок. Хоча продуктивність поліпшується в цьому випадку, енергоефективність залишається тією ж, оскільки додана потужність іншого процесора скасовує переваги продуктивності. Вище була розглянута енергоефективність під час виконання одного запиту. Проте, системи баз даних зазвичай виконують кілька запитів одночасно. Тому далі буде розглянуто вплив енергоефективності на збільшення паралелізму (кількість запитів, які виконуються паралельно) в системі.

Підводячи підсумок, незалежно від складності запиту і того, які кнопки використовуються, тобто методи доступу, алгоритми операторів, тип і рівень паралелізму, енергоефективність і продуктивність йдуть рука об руку. Енергоефективність, яка визначається як відношення корисної роботи до використовуваної енергії, еквівалентна співвідношенню продуктивності і потужності. Коли додаються системні компоненти по одному або використовується більше ресурсів, як потужність, так і продуктивність системи змінюються. Загальна потужність збільшується, і очікується, що продуктивність також покращиться. У будь-який момент енергоефективність поліпшується, коли відносне поліпшення продуктивності перевищує відносне збільшення потужності:

$$\frac{\Delta\,\text{Perf}}{\text{Perf}} > \frac{\Delta\,\text{Power}}{\text{Power}} \tag{2.12}$$

Коли обидві сторони рівні, є баланс відносних поліпшень і, таким чином, підтримується така ж ефективність. Коли відношення змінюється на протилежне – знаходимося в області спадної ефективності. Тобто збільшення продуктивності від додаткового ресурсу не виправдовує його енерговитрати. Ще одне важливе міркування в центрах обробки даних - пікове споживання енергії. Стійки, особливо старіші, мають обмежену ємність для подачі електроенергії, а при

перевантаженні можуть привести до спрацьовування запобіжників. Пікове використання потужності також може збільшити температуру, що перевищує ємність для охолодження або значно збільшити вартість охолодження. В результаті, центрам обробки даних часто необхідно забезпечити виконання бюджетів потужності і покладатися на узгоджені контролери на всіх рівнях від стійки до вузла, щоб забезпечити їх дотримання. На рівні вузлів апаратні механізми можуть швидко регулювати потужність, але мало знають про продуктивність додатків.

2.3 Висновки до другого розділу

В ході роботи були досліджені три основні концептуальні підходи до управління базою даних: статистичний підхід для ранжирування параметрів настройки бази даних (SARD), схеми індексування для самоналаштувальної СУБД, самоналаштування динамічних ресурсів для робочих навантажень. Кожен з підходів має як свої переваги, так і недоліки, про які було зазначено в дослідженні, на основі проведених есперименів. Завдяки проведеному аналізу, для подальшого опрацьовування був обраний підхід самоналаштування динамічних ресурсів для робочих навантажень.

Результати експериментів стосовно енергоефективності показують, що потужність процесора, що використовується різними операторами, може сильно варіюватися: на 60% для одного і того ж використання ЦП і що потужність ЦП не є лінійної з використанням. Було виявлено, що найбільш ефективна конфігурація також являється найбільш енергоефективною. У тих небагатьох випадках, коли цього не було, енергоефективність не покращилася більш ніж на 10%. Цей взаємозв'язок є результатом великих витрат на електроенергію в сучасних серверних компонентах. Однак дослідження вказують на два перспективні напрямки для підвищення енергоефективності: консолідація ресурсів на недовикористовуваних вузлах для економії енергії без шкоди для продуктивності, а також альтернативне енергоефективне обладнання для зниження витрат на фіксовану потужність.

2.4 Література до другого розділу

1. “PostgreSQL DBMS Documentation”, http://www.postgresql.org/.
2. S. Chaudhuri and G. Weikum, “Rethinking Database Architecture: Towards s Self-tuning RISC-style Database System,” in Proceedings of VLDB, 2000, pp. 1–10.
3. K. Dias, M. Ramacher, U. Shaft, V. Venkataramamani, and G. Wood, “Automatic Performance Diagnosis and Tuning in Oracle,” in Proceedings of CIDR, 2005, pp. 1110–1121.
4. G. Group, “The Total Cost of Ownership: The Impact of System Management Tools,” 1996.

5. H. Group, “Achieving Faster Time-to-Benefit and Reduced TCO with Oracle Certified Configurations,” March, 2002.
6. R. Plackett and J. Burman, “The Design of Optimum Multifactorial Experiments,” in Biometrika Vol. 33 No. 4, 1946, pp. 305–325.
7. D. J. Lilja, Measuring Computer Performance A practioner‘s Guide. Cambridge University Press, 2000.
8. D. Montgomery, Design and Analysis of Experiments. Wiley, 2001.
9. B. Debnath, J. Skarie, D. Lilja, and M. Mokbel, “SARD: A Statistical Approach for Ranking Database Tuning Parameters,” Laboratory for Advanced Research in Computing Technology and Compilers Technical Report, no. ARCTiC 07-11, 2007.
10. Biplob K. Debnath, David J. Lilja, Mohamed F. Mokbel, “SARD: A Statistical Approach for Ranking Database Tuning Parameters”, University of Minnesota, Twin Cities, USA
11. T. Kraft, H. Schwarz, R. Rantzau, and B. Mitschang. CoarseGrained Optimization: Techniques for Rewriting SQL Statement Sequences. In VLDB’2003, pages 488–499, 2003.
12. E. J. O’Neil, P. E. O’Neil, and G. Weikum. The LRU-K Page Replacement Algorithm For Database Disk Buffering. In SIGMOD’1993, pages 297–306, 1993.
13. M. Stonebraker. The case for partial indexes. Sigmod Record, 18(4):4–11, Dec. 1989.
14. Kai-Uwe Sattler, Eike Schallehn, Ingolf Geist, “Towards Indexing Schemes for Self-Tuning DBMS”, TUIlmenau, UniversityofMagdeburg
15. Ivan T. Bowman, David Toman Abdelkader Hameurlain, F. Morvan, “CPU and incremental memory allocation in dynamic parallelization of SQL queries”, Parallel Computing, 2002, pp. 525–556.
16. K. P. Brown, M. Mehta, M. J. Carey, and M.Livny, “Towards Automated Performance Tuning For Complex Workloads”, Proceedings of the 20th Very Large Data Base Conference, Santiago, Chile, 1994.
17. P. Martin, W. Powley,Zheng Min, K.Romanufa, “Experimental study of a self-tuning algorithm for DBMS buffer pools”, Journal of Database Management, Apr.2005, pp. 1-20.
18. Baoning Niu, P. Martin, W. Powley, R. Horman and P. Bird, “Workload Adaptation in Autonomic DBMSs”, Proceedings of CASCON, 2006, pp. 161 - 173.
19. D. A. Menascé, and M. N. Bennani, “On the Use of Performance Models to Design Self-Managing Computer Systems”, Proceedings of 2003 Computer Measurement Group Conference, Dallas, TX. USA, Dec.7-12, 2003, pp. 1-9.
20. D. Botzer, and O. Etzion, “Self-Tuning of the Relationships among Rules' Components in Active Databases Systems” IEEE Transactions On Knowledge And Engineering, Mar.2004, pp. 375-379.

21. Fu Duan, Yongjie Han, Qiuyong Zhao, Keming Xie, “Towards Self-tuning of Dynamic Resources for Workloads”, Taiyuan University of Technology, Taiyuan, Shanxi, P.R.China, 2008
22. S. Rivoire, P. Ranganathan, and C. Kozyrakis. A comparison of high-level full-system power models. In HotPower, 2008.
23. X. Fan, W.-D. Weber, and L. A. Barroso. Power provisioning for a warehouse-sized computer. In ISCA, 2007.
24. G. Graefe. Database servers tailored to improve energy efficiency. In Software Engineering for Tailor-made Data Management, pages 24–28, 2008.
25. S. Harizopoulos, M. A. Shah, J. Meza, and P. Ranganathan. Energy efficiency: The new holy grail of data management systems research. In CIDR, 2009.
26. J. Meza, M. A. Shah, P. Ranganathan, M. Fitzner, and J. Veazey. Tracking the power in an enterprise decision support system. In ISLPED ’09, pages 261–266, 2009.
27. W. Lang and J. M. Patel. Towards eco-friendly database management systems. In CIDR, 2009.
28. Z. Xu, Y. Tu, and X. Wang. Exploring power-performance tradeoffs in database systems. In ICDE, 2010.
29. J. Hamilton. Internet-scale data center power efficiency. In CIDR, 2009.
30. D. G. Andersen, J. Franklin, M. Kaminsky, A. Phanishayee, L. Tan, and V. Vasudevan. Fawn: a fast array of wimpy nodes. In SOSP, pages 1–14, 2009.
31. S. Rivoire, M. A. Shah, P. Ranganathan, and C. Kozyrakis. Joulesort: a balanced energy-efficiency benchmark. In SIGMOD ’07, pages 365–376, 2007.
32. N. Tolia, Z. Wang, et al. Delivering energy proportionality with non energy-proportional systems – optimizing the ensemble. In HotPower, 2008.
33. R. Raghavendra, P. Ranganathan, et al. No power struggles: A unified multi-level power management architecture for the data center. In ASPLOS, 2008.
34. L. A. Barroso and U. Holzle. The case for ¨ energy-proportional computing. IEEE Computer, 40(12):33–37, 2007.
35. Numonyx. Phase change memory (pcm): A new memory technology to enable new memory usage models. Online, 2009. http://www.numonyx.com/ Documents/WhitePapers/ Numonyx PhaseChangeMemory WhitePaper.pdf.
36. D. Strukov, G. Snider, D. Stewart, and R. S. Williams. The missing memristor found. Nature, 453:80–83, 2008.
37. Dimitris Tsirogiannis, Stavros Harizopoulos, Mehul A. Shah, “Analyzing the Energy Efficiency of a Database Server”, University of Toronto, HP Labs

3 РОЗРОБКА МЕТОДУ ДИНАМІЧНОГО САМОНАСТРОЮВАННЯ БАЗИ ДАНИХ

3.1 Самоналагоджувальна архітектура для моніторингу, настройки і тренда

Бізнес-дані завжди ростуть із кілобайта, мегабайта, гігабайта, терабайта, петабайта і так далі. Неможливо уникнути цього збільшення швидкості даних до тих пір, поки бізнес ще не запущений. Через цю проблему налаштування баз даних є важливою частиною інформаційної системи. Налаштування баз даних з економічної точки зору є все більш складною задачею. Загальна вартість володіння (TCO) інформаційними технологіями повинна бути значно зменшена за рахунок мінімізації витрат людей. Фактично, помилки в операціях та адміністрування інформаційних систем є єдиними причинами відмови системи та неприйнятної продуктивності [1]. Один із способів вирішення проблеми загальної вартості володіння - це зробити інформаційні системи більш самоврядними. Особливо складною частиною амбіціозного видіння створення автономних систем бази даних є автоматизація налаштування продуктивності бази даних. В цьому пункті пояснюється прогрес, досягнутий до справжнього часу по цій важливій проблемі. Зокрема, пропонується архітектура та алгоритм для цієї проблеми.

Витрати на апаратне забезпечення швидко падають, а витрати на персонал залишаються відносно статичними. Це призводить до того, що витрати на ручну настройку на людину перевищують витрати на прискорене обладнання (рисунок 3.1).

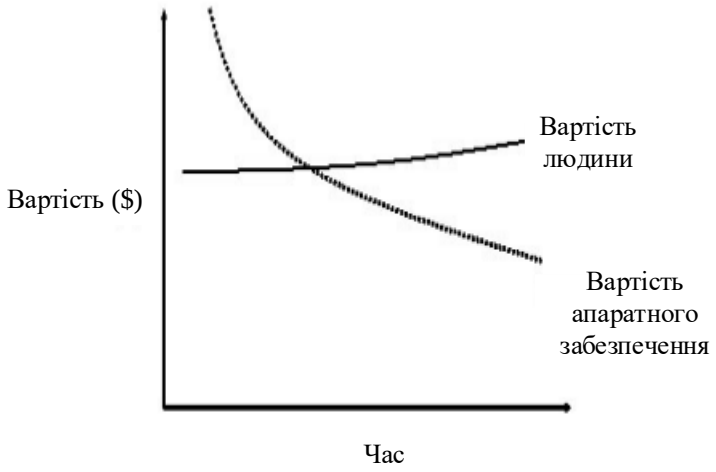


Рисунок 3.1 - Вартість апаратного забезпечення проти людських витрат

Більшість великих баз даних управляються адміністраторами баз даних, які відповідають за хорошу продуктивність бази даних, але ручний фізичний дизайн є трудомістким і дуже виснажливим, оскільки адміністратор бази даних (DBA) повинен знайти переваги різних індивідуальних конструктивних елементів, які можуть взаємодіяти один з одним. Мотивовані не

тільки складністю налаштування, а й необхідністю зниження загальної вартості володіння в своїх продуктах, деякі комерційні постачальники СУБД пропонують автоматизовані засоби фізичного проектування з декількома функціями. При різкому зниженні вартості апаратного та програмного забезпечення витрати на персонал і тюнінг персоналу перевищують вартість володіння системою баз даних [2]. Проблема фізичного дизайну пов'язана з пошуком потенційно дуже великого простору різних конфігурацій кандидатів. Пошук простору альтернативних конфігурацій недоцільний. Тому останні інструменти фізичного проектування засновані на жадібних евристиках, які скорочують простір пошуку. Повторні звернення до оптимізатора кожного разу, коли треба оцінити запит в іншій конфігурації, накладають серйозне вузьке місце на виконання фізичних дизайнерів. Грунтуючись на експериментальних результатах, 90% часу настройки витрачається на результати очікування оптимізатора замість оцінки потенційно перспективних конфігурацій [3,4].

Більшість систем реагуватимуть на збільшення навантаження з деяким ступенем зниження продуктивності. Здатність системи приймати більш високе навантаження називається масштабністю, а зміна системи для обробки більш високого навантаження є синонімом настройки продуктивності.

Систематична настройка виконується наступним чином:

- Оцінка проблеми і встановлення числових значень, які класифікують прийнятну поведінку;
- Вимірювання продуктивності системи перед її модифікацією;
- Визначення частини системи, яка має вирішальне значення для підвищення продуктивності. Це називається вузьким місцем;
- Зміна цієї частини системи, щоб видалити вузьке місце.

Проблема продуктивності може бути ідентифікована повільними або не реагуючими системами. Зазвичай це відбувається через високе завантаження системи, що призводить до того, що якась частина системи досягає межі в своїй здатності реагувати. Ця межа в системі називається вузьким місцем. Для підвищення продуктивності використовується кілька методів.

Дані для сучасних підприємств і управління базами даних часто пов'язані з комплексним плануванням, управлінням часом і реалізацією стандартних завдань системи. Автоматизація баз даних допомагає підприємствам краще управляти своїми операціями з базами даних, скорочуючи час простою, а також загальний час, що витрачається на управління базою даних. Автоматизація в будь-якому місці працює з будь-якою базою даних SQL, такими як Oracle, MS SQL, Sybase, SQL DB2 і т. д. На відміну від інших рішень для автоматизації, вона не вимагає значного навчання. Проста, зручна у використанні, але потужна, вона може автоматизувати будь-яке завдання бази даних. [5]

Адміністратор бази даних відповідає за підвищення продуктивності системи баз даних.

Виявлення погіршення продуктивності досягається за рахунок постійного моніторингу параметрів продуктивності системи. Було запропоновано кілька методів, в тому числі використання матеріалізованих уявлень і індексів, таблиці обрізки і набори колонок, використання методів самовідновлення, використання фізичної настройки дизайну і т. д., які активно відстежують показники продуктивності системи, аналізують симптоми і автоматично налаштовують СУБД для підвищення продуктивності. Зниження продуктивності пов'язано зі збільшенням робочого навантаження в системі. Це підвищене навантаження повинно бути зведене до мінімуму для підвищення швидкості реакції системи. Для досягнення цієї мети або адміністратор зменшує деяке навантаження, закриваючи деякі файли, або збільшує ОЗУ. Адміністратор повинен постійно перевіряти, чи можна з регулярним твердженням розрахувати коефіцієнт буферного кешу. Грунтуючись на цьому коефіцієнті попадання, адміністратор бази даних визначає, чи має бути виділено більшу кількість ОЗУ. Це завдання зниження навантаження за рахунок збільшення ОЗУ вимагає ручного втручання і, отже, можуть знадобитися навіть роки для завершення. [5]

Проте, Oracle управляє потребами оперативної пам'яті відповідно до вимог кожного завдання, використовуючи складні алгоритми для підвищення швидкості роботи оперативної пам'яті. Oracle DBA може динамічно деактивувати оперативну пам'ять, а також перерозподіляти її. Але оскільки адміністратор бази даних є нормальною людиною, він не може розрахувати фактичний обсяг оперативної пам'яті, необхідний додатку.

Через це обмеження DBA розподіл ОЗУ вручну для оптимізації продуктивності системи баз даних стає складним і дорогим завданням.

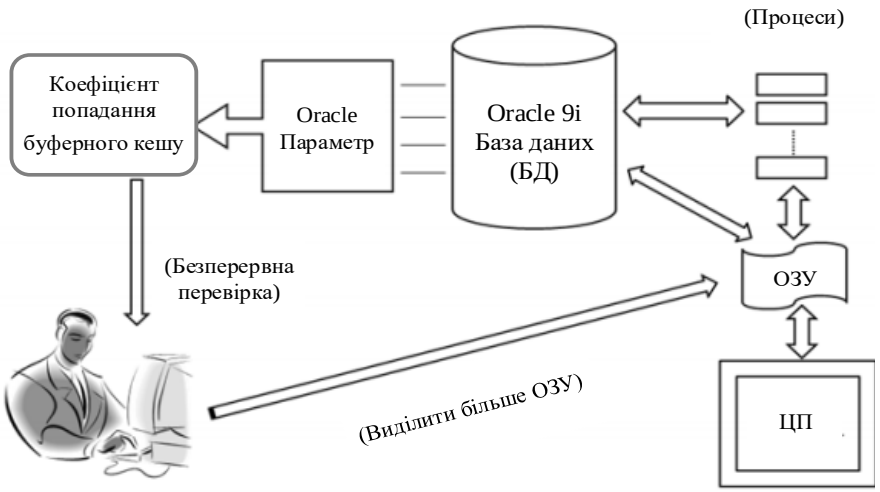


Рисунок 3.2 - Ручна настройка бази даних

Іноді виділяється більше ОЗУ, ніж потрібно, що забирає додаткову частину ОЗУ. [1] Таким чином, існує велика потреба в динамічних можливостях виділення пам'яті для створення бази

даних самонастроювання. В Oracle Database 10g функція автоматичної настройки, така як автоматичне керування пам'яттю, дозволяє системі бази даних виявляти недоліки і налаштовувати області основної пам'яті відповідно до нових вимог до середовища Oracle. Тому дослідники в даний час зосереджені на розробці методів самонастроювання, таких як проект автоматичної настройки COMFORT [3] або підхід MAPE, даний IBM [4] для безперервної адаптації.

Ранжування різних параметрів настройки на основі статистичного аналізу представлено в [6]. Ранжування параметрів засноване на величині впливу, який вони чинять на продуктивність системи для даного робочого навантаження. Представлена формальна база знань для системи самонастроювання бази даних, в якій визначені кілька компонентів знань, які включають знання політики, знання робочого навантаження, знання з діагностики проблем, знання про дозвіл проблем, знання про ефекторів і знання про залежності. Архітектура, представлена в цьому пункті, включає витяг корисної інформації з системного журналу, а також з СУБД з використанням системних запитів. Ця інформація, зібрана протягом певного періоду часу, потім використовується для запуску SQL-скриптів для бажаного часу відгуку вихідного сигналу. Потім структура додатка оцінює ступінь корекції, яка повинна застосовуватися до ключових системних параметрів, які допомагають збільшити продуктивність системи. Класичний контроль змінений, і для забезпечення стабільності системи використовується триступеневе управління, що включає контроль, аналіз та налаштування [6]. Архітектура, представлена для бази даних самовідновлення, служить основою для нової архітектури, представленої в цьому пункті. Тут представлена нова архітектура СУБД, заснована на модульному підході, де в кожному функціональному модулі можна контролювати набір контрольних гачків. Ці контрольні гачки відповідають за збереження поточної інформації про стан або збереження моментального знімку сервера в журналі. Ця архітектура має значні накладні витрати на моніторинг, через те, що при великій кількості параметрів, які необхідно контролювати, інформація про статус кожного модуля повинна зберігатися в журналі, і якщо це зроблено, часто можна з'їсти багато процесорного часу. Більш того, ця архітектура більше орієнтована на зцілення системи і не розглядає настройку СУБД для підвищення продуктивності.

Багато бізнес-додатків вимагають використання складних систем баз даних, які необхідно адмініструвати і оптимізувати для підвищення продуктивності. Як було запропоновано в [7], слід уникати фізичної настройки, так як це дорого. Оскільки фізичний дизайн бази даних страждає від різних обмежень, пропонується нова автоматизована архітектура баз даних на основі сценаріїв для досягнення високого рівня продуктивності. Архітектура, показана на рисунку 3.3, використовується для ідентифікації симптомів і зміни параметрів ключової системи. Файл журналу системи СУБД буде основним джерелом інформації, яка перевіряє поточний статус системи. Інструмент інтелектуального аналізу даних стискає дані на більш дрібну інформаційну

базу, оскільки файл журналу може містити величезний обсяг даних. Архітектура має три основних будівельних блоку, що складаються з Data Miner, Script і Tuner. Після вилучення значущої інформації ступінь необхідної корекції оцінюється за пропонованим сценарієм і алгоритмам.

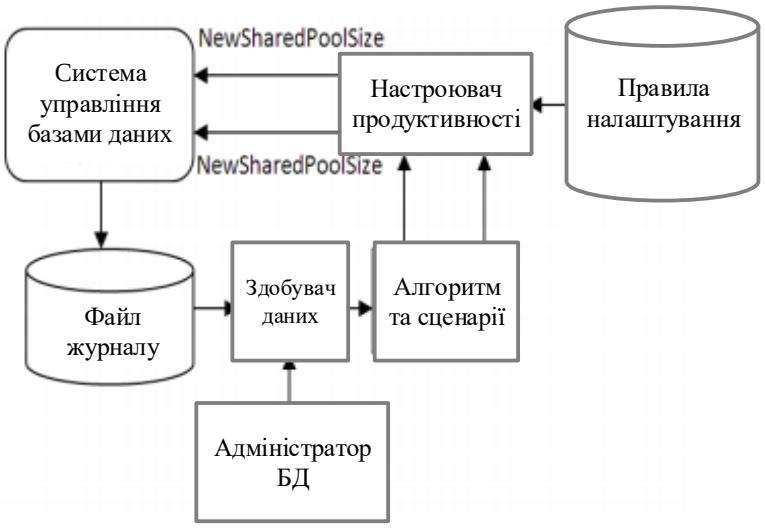


Рисунок 3.3 - Архітектура настройки на основі скриптів

Ці алгоритми і скрипти будуть налаштовувати базу даних з використанням різних правил настройки, а також системних параметрів. Тим не менше, кілька параметрів можуть бути змінені одночасно для підвищення продуктивності. Алгоритм оцінює необхідний розмір буфера на основі поточних вхідних параметрів СУБД, і тюнер застосовує необхідну корекцію до розміру буфера на основі правил настройки. Найголовніше, що внутрішня коригуюча міра, така як зміна розміру буфера СУБД, що використовується при обробці запитів, досліджується в цій архітектурі.

У цьому дослідженні надана самоналагоджувальна архітектура системи баз даних, як показано на рисунку 3.4, щоб підвищити продуктивність системи. Оскільки DBA відповідає за адміністрування та оптимізацію різних завдань, він може або збільшити ОЗУ, або зменшити навантаження на процесор з метою оптимізації продуктивності. Але це буде трудомістким методом, оскільки DBA є нормальною людиною, яка не може виконувати складні обчислення протягом кількох секунд, як комп'ютерна система. [3]

DBA може не знати точно, скільки ОЗУ має бути виділено для підвищення продуктивності системи. Отже, пропонується підхід до автоматизації цієї задачі оптимізації DBA, тобто завдання, яке DBA повинен зробити для підвищення продуктивності, тепер буде виконуватися комп'ютерною системою протягом невеликих тимчасових графіків.

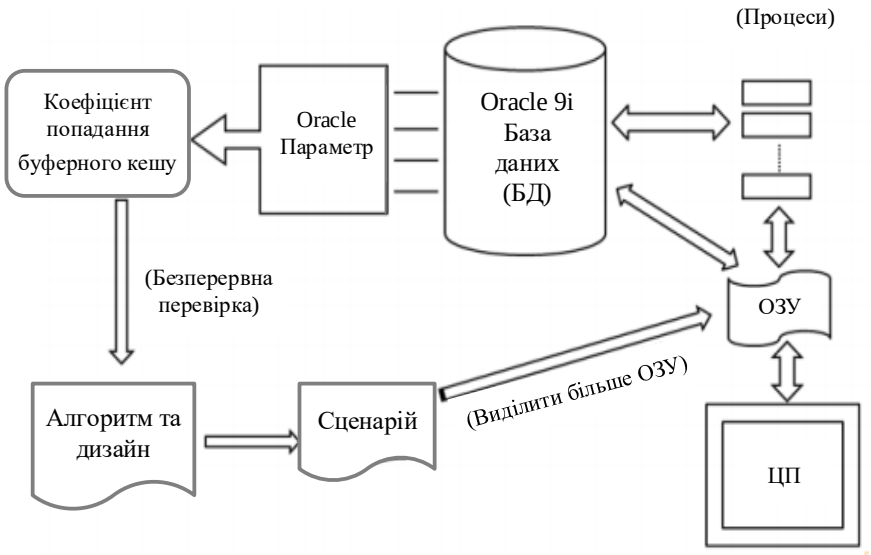


Рисунок 3.4 - Автоматизована розробка баз даних

Алгоритм визначає три змінні: ΔRT абревіатура для зміни часу відгуку, BUFFER_SIZE позначає поточний розмір буфера, CACHE_SIZE відповідає розміру кеш-пам'яті.

```
ALGORITHM

1. dbTuner (ESTIMATED_CACHE_SIZE)
2. Begin
3.      Run application, algorithm and process
4.              Calculate the change in response time (ΔRT)
5.              If (ΔRT>0)
                {
                        Run Script
                        {
                                BUFFER_SIZE = BUFFER_SIZE + 1
        Allocate more RAM and update CACHE_SIZE
        }
    Else IF (ΔRT<0)
    {
     Run Script
     {
             BUFFER_SIZE = BUFFER_SIZE - 1
        Reduce RAM and update CACHE_SIZE
        }
     }
6.   Go To Step 4
6.   Stop application, algorithm and process
7.  End
```

У таблиці 3.1 наведені дані про підготовку зразка. Для тестування пропонованої системи був використаний набір даних для навчання розміром 100. Як видно з таблиці, розмір буфера

налаштовується для збільшення розміру таблиці, кількості призначених для користувача процесів і співвідношення буферної місії, так що час виконання запиту зменшується і пам'ять використовується ефективно.

Таблиця 3.1 - Приклад набору даних навчання

Розмір таблиці	Співвідношення буферної місії	Розмір спільного пулу (у Мб)	Розмір буфера (у Мб)
1000	0.9624	32	4
1000	0.9152	32	4
1000	0.9791	32	8
1000	0.9613	32	8
2000	0.9371	32	8
2000	0.9453	40	8
3000	0.8931	40	16
3000	0.8253	40	16

Налаштування бази даних може стати досить складним, але Oracle9i пропонує адміністратору безпрецедентну можливість управляти PGA і SGA. Поки Oracle9i не перетвориться в повністю самоналаштувальну архітектуру, DBA буде відповідати за настройку динамічної конфігурації системної ОЗУ. Автоматизовані скрипти налаштування SGA можуть використовуватися, щоб дозволити DBA рости і скорочувати області SGA. Ці сценарії розміщуються в dbms_job для запланованої обробки. Oracle надає розширені уявлення в процесі v$process, v$pgastat, щоб можна було відслідковувати поведінку області сортування RAM. Уявлення v$ в Oracle9i також дають розуміння про використання ОЗУ для окремих операторів SQL в кеші бібліотеки.

3.2 Результати реалізації пропонованої технології

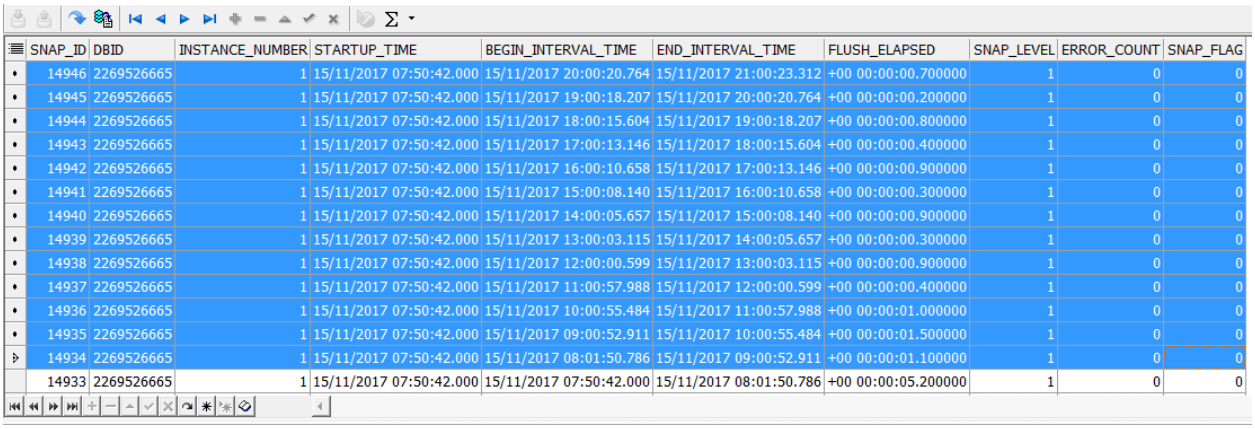
Знаходження проблемної області для настройки - це дуже непросте завдання. Вона не тільки залежить від кількості процесорів, внутрішньої пам'яті, дискового простору. Це сукупність таких властивостей як: наскільки підключення можливо до бази даних, час виконання запитів користувачів, вибір OLTP OLAP, і т.д.

Фактори, що впливають на продуктивність бази даних:

- Визначення пам'яті для структур бази даних;
- Визначення вимог вводу / виводу різних частин бази даних;
- Налаштування операційної системи для оптимальної роботи бази даних.

Для того щоб подивитися статистику вибирається часовий проміжок, за який період треба побачити статистику:

select * from dba_hist_snapshot order by snap_id desc



SNAP_ID	DBID	INSTANCE_NUMBER	STARTUP_TIME	BEGIN_INTERVAL_TIME	END_INTERVAL_TIME	FLUSH_ELAPSED	SNAP_LEVEL	ERROR_COUNT	SNAP_FLAG
14946	2269526665	1	15/11/2017 07:50:42.000	15/11/2017 20:00:20.764	15/11/2017 21:00:23.312	+00 00:00:00.700000	1	0	0
14945	2269526665	1	15/11/2017 07:50:42.000	15/11/2017 19:00:18.207	15/11/2017 20:00:20.764	+00 00:00:00.200000	1	0	0
14944	2269526665	1	15/11/2017 07:50:42.000	15/11/2017 18:00:15.604	15/11/2017 19:00:18.207	+00 00:00:00.800000	1	0	0
14943	2269526665	1	15/11/2017 07:50:42.000	15/11/2017 17:00:13.146	15/11/2017 18:00:15.604	+00 00:00:00.400000	1	0	0
14942	2269526665	1	15/11/2017 07:50:42.000	15/11/2017 16:00:10.658	15/11/2017 17:00:13.146	+00 00:00:00.900000	1	0	0
14941	2269526665	1	15/11/2017 07:50:42.000	15/11/2017 15:00:08.140	15/11/2017 16:00:10.658	+00 00:00:00.300000	1	0	0
14940	2269526665	1	15/11/2017 07:50:42.000	15/11/2017 14:00:05.657	15/11/2017 15:00:08.140	+00 00:00:00.900000	1	0	0
14939	2269526665	1	15/11/2017 07:50:42.000	15/11/2017 13:00:03.115	15/11/2017 14:00:05.657	+00 00:00:00.300000	1	0	0
14938	2269526665	1	15/11/2017 07:50:42.000	15/11/2017 12:00:00.599	15/11/2017 13:00:03.115	+00 00:00:00.900000	1	0	0
14937	2269526665	1	15/11/2017 07:50:42.000	15/11/2017 11:00:57.988	15/11/2017 12:00:00.599	+00 00:00:00.400000	1	0	0
14936	2269526665	1	15/11/2017 07:50:42.000	15/11/2017 10:00:55.484	15/11/2017 11:00:57.988	+00 00:00:01.000000	1	0	0
14935	2269526665	1	15/11/2017 07:50:42.000	15/11/2017 09:00:52.911	15/11/2017 10:00:55.484	+00 00:00:01.500000	1	0	0
14934	2269526665	1	15/11/2017 07:50:42.000	15/11/2017 08:01:50.786	15/11/2017 09:00:52.911	+00 00:00:01.100000	1	0	0
14933	2269526665	1	15/11/2017 07:50:42.000	15/11/2017 07:50:42.000	15/11/2017 08:01:50.786	+00 00:00:05.200000	1	0	0

Рисунок 3.5 - Статистика за заданий часовий проміжок

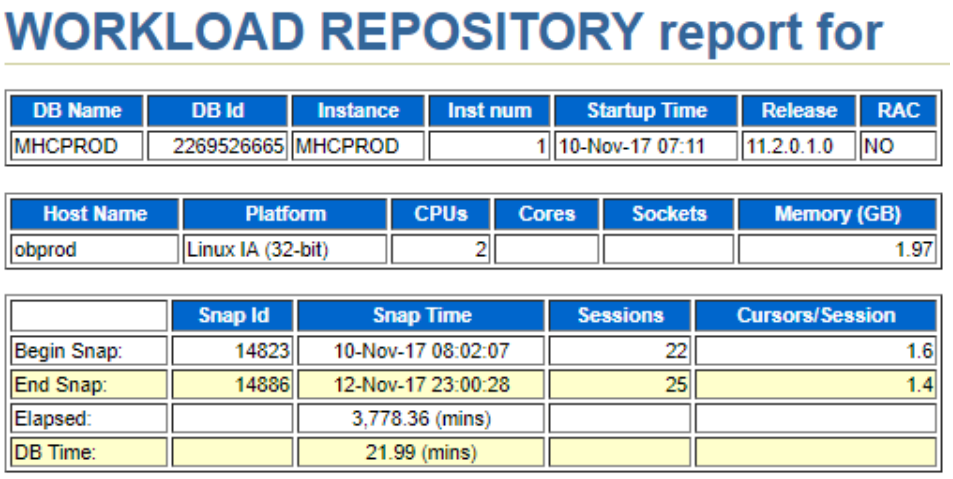
select * from table (DBMS_WORKLOAD_REPOSITORY.awr_report_html(l_dbid=> 2269526665,

l_inst_num=>1,

l_bid=>14933,

l_eid=>14945))

Вибираються проміжки і створюється звіт у вигляді html-файла.



WORKLOAD REPOSITORY report for

DB Name	DB Id	Instance	Inst num	Startup Time	Release	RAC
MHCPROD	2269526665	MHCPROD	1	10-Nov-17 07:11	11.2.0.1.0	NO

Host Name	Platform	CPUs	Cores	Sockets	Memory (GB)
obprod	Linux IA (32-bit)	2			1.97

	Snap Id	Snap Time	Sessions	Cursors/Session
Begin Snap:	14823	10-Nov-17 08:02:07	22	1.6
End Snap:	14886	12-Nov-17 23:00:28	25	1.4
Elapsed:		3,778.36 (mins)		
DB Time:		21.99 (mins)		

Рисунок 3.6 – Звіт у вигляді html-файла

Shared Pool Statistics

	Begin	End
Memory Usage %:	32.45	87.33
% SQL with executions>1:	50.27	80.21
% Memory for SQL w/exec>1:	47.73	64.52

Top 5 Timed Foreground Events

Event	Waits	Time(s)	Avg wait (ms)	% DB time	Wait Class
DB CPU		1,500		113.68	
db file sequential read	24,446	40	2	3.02	User I/O
db file scattered read	9,499	29	3	2.19	User I/O
log file sync	4,265	11	3	0.85	Commit
resmgr:cpu quantum	116	6	51	0.44	Scheduler

Host CPU (CPUs: 2 Cores: Sockets:)

Load Average Begin	Load Average End	%User	%System	%WIO	%Idle
0.31	0.15	0.2	0.3	0.1	99.5

Instance CPU

%Total CPU	%Busy CPU	%DB time waiting for CPU (Resource Manager)
0.5	99.2	0.4

Memory Statistics

	Begin	End
Host Mem (MB):	2,022.0	2,022.0
SGA use (MB):	528.0	528.0
PGA use (MB):	46.8	51.4
% Host Mem used for SGA+PGA:	28.43	28.66

Рисунок 3.7 - Продовження звіту у вигляді html-файла

Використовуючи зібрану статистику за різні дні можна виділити пікові навантаження і підлаштовувати параметри під потрібний результат. У більшості випадків встановлених стандартних налаштувань при установці бази даних з використанням помічника досить, але якщо профіль обраний невірно, або ж кількість користувачів збільшується підключаються до бази даних, в цьому випадку потрібно робити настройку в ручному режимі змінюючи параметри, але найкраще конфігурувати нову установку бази даних вже з відомими показниками і зробити відновлення даних. У цьому випадку можна розвантажити основну базу даних і поступово перенаправляти на нову базу даних, збираючи статистику.

Завантаженість ЦП - дуже важливий показник, якщо завантаженість ЦП вище 80-95%, то цілком імовірно, що база даних перезапуститься з втратою даних або вийде з ладу. ЦП пов'язаний з читанням та записом в пам'ять, дисковий простір.

Завантаженість ЦП в більшості залежить від написаного SQL-запиту і в подальшому з обробкою даних, вибраних з таблиць.

SELECT TOP_LEVEL_SQL_ID,

```
        sql_id,
        sql_plan_hash_value,
        sql_plan_line_id,
        current_obj#,
        event,
        SUM (TIME_WAITED),
        COUNT (*)
   FROM active_session_history
  WHERE user_id = 101
GROUP BY TOP_LEVEL_SQL_ID,
        sql_id,
        sql_plan_line_id,
        event
```

TOP_LEVEL_SQL_ID	SQL_ID	SQL_PLAN_HASH_VALUE	SQL_PLAN_LINE_ID	CURRENT_OBJ#	EVENT	SUM(TIME_WAITED)	COUNT(*)
4x2w3685u5cyn	4x2w3685u5cyn	3283153649		-1	log file sync	4962	1
gryf3m7wamxpz	gryf3m7wamxpz	851969126	1	74812	direct path read	2143	4
gryf3m7wamxpz	gryf3m7wamxpz	851969126	35	74808	db file sequential read	565	1
0dnf14xkzp6b2	0dnf14xkzp6b2	0		-1		0	1
1m71sw8mg77wk	1m71sw8mg77wk	2075335828	50	74812		0	1
4x2w3685u5cyn	4x2w3685u5cyn	3283153649	10	-1		0	1
gryf3m7wamxpz	gryf3m7wamxpz	851969126	15	74812		0	1
gryf3m7wamxpz	gryf3m7wamxpz	851969126	1	74808		0	1
0m9qn06zz4bvq	0m9qn06zz4bvq	3455776041	21	-1		0	1
gryf3m7wamxpz	gryf3m7wamxpz	851969126	1	74812		0	15
c39mugjqqgsjx	c39mugjqqgsjx	3405442970		74812		0	1
2sjkzdkv73kr6	2sjkzdkv73kr6	2075335828		74966		0	1
c3skvng973g55	c3skvng973g55	0		-1		0	1
0m9qn06zz4bvq	0m9qn06zz4bvq	3455776041	4	-1		0	1

Рисунок 3.8 - Активна історія сеансів

Необхідно звернути увагу на другий рядок.

```
select * from v$sqlarea where sql_id = 'gryf3m7wamxpz' ;
```

Далі можна побачити, що містить цей запит.

SQL_FULLTEXT	SQL_ID	SHARABLE_MEM	PERSISTENT_MEM	RUNTIME_MEM	SORTS	VERSION_COUNT	LOADED_VERSIONS	OPEN_VERSIONS	USERS_OPENING
(HUGECLOB)	gryf3m7wamxpz	87557	63428	60264	5	1	1	0	0

Рисунок 3.9 - Вміст запиту

У колонці SQL_FULLTEXT міститься запит:

Select ID_APPNT,ID_APPNT_TYPE,START_DATE, END_DATE, EVENT_LENGTH, EVENT_PID, REC_PATTERN, REC_TYPE,FEC,APPNT_NAME_RU,APPNT_NAME_UA,ID_PATIENT,USERNAME_PATIENT,DATE_OF_BIRTH,ID_GENDER,ID_SECTOR,ID_OFFICE,PARENT_SEC,DESCRIPTION_OTDEL,ID_STAF,MHC_ROLE,USERNAME_STAFF,WADD_IDSECTOR,IDO_WADD,DO_WADD,PS_WADD,IDF_WADD,ROLE_WADD, USERNAME_WADD,PID_APPNT FROM MHC.MHC_V_EVENTS_REG

Цей запит зайняв найбільше часу.

Застосувавши функцію Explain Plan на цей запит, можна його розібрати.

Id	Operation	Name	E-Rows	E-Bytes	E-Temp	Cost (%CPU)	E-Time	Pstart	Pstop
0	SELECT STATEMENT		2478	4169K		1298 (1)	00:00:16		
1	SORT ORDER BY		2478	4169K	4968K	1298 (1)	00:00:16		
* 2	HASH JOIN		2478	4169K		402 (2)	00:00:05		
* 3	TABLE ACCESS FULL	MHC_T_PERSON	2002	76076		19 (0)	00:00:01		
* 4	HASH JOIN		2478	4077K		382 (2)	00:00:05		
* 5	TABLE ACCESS FULL	MHC_T_PERSON	2002	92092		19 (0)	00:00:01		
* 6	HASH JOIN		2478	3966K		363 (2)	00:00:05		
7	INDEX FULL SCAN	MHC_I_PATIENT_FK01	304	2432		1 (0)	00:00:01		
* 8	HASH JOIN		2478	3946K		361 (2)	00:00:05		
9	VIEW	index$_join$_019	11	77		3 (34)	00:00:01		
* 10	HASH JOIN								
11	INDEX FAST FULL SCAN	MHC_I_STAFF_FK01	11	77		1 (0)	00:00:01		
12	INDEX FAST FULL SCAN	MHC_I_STAFF_PK	11	77		1 (0)	00:00:01		
* 13	HASH JOIN		2478	3929K		358 (1)	00:00:05		
14	TABLE ACCESS FULL	MHC_T_OFFICES	7	294		3 (0)	00:00:01		
* 15	HASH JOIN		2478	3828K		355 (1)	00:00:05		
16	TABLE ACCESS FULL	MHC_T_SECTORS	11	187		19 (0)	00:00:01		
* 17	HASH JOIN		2478	3787K		335 (1)	00:00:05		
18	TABLE ACCESS FULL	MHC_T_TYPE_APPOINTMENTS	5	290		19 (0)	00:00:01		
* 19	HASH JOIN		2478	3646K		316 (1)	00:00:04		
20	NESTED LOOPS								
21	NESTED LOOPS		11	1144		23 (5)	00:00:01		
* 22	HASH JOIN		11	726		12 (9)	00:00:01		
23	NESTED LOOPS								
24	NESTED LOOPS		11	649		9 (0)	00:00:01		
25	TABLE ACCESS FULL	MHC_T_OFFICES	7	294		3 (0)	00:00:01		
* 26	INDEX RANGE SCAN	MHC_I_SECTORS_FK01	4			0 (0)	00:00:01		
27	TABLE ACCESS BY INDEX ROWID	MHC_T_SECTORS	2	34		2 (0)	00:00:01		
28	VIEW	index$_join$_013	11	77		3 (34)	00:00:01		
* 29	HASH JOIN								
30	INDEX FAST FULL SCAN	MHC_I_STAFF_FK01	11	77		1 (0)	00:00:01		

Рисунок 3.10 - Застосування функції Explain Plan на запит

```
Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("EW"."ID_PERSON"="BW"."ID_PERSON")
   3 - filter("BW"."ID_GENDER" IS NOT NULL)
   4 - access("AP"."ID_PERSON"="BP"."ID_PERSON")
   5 - filter("BP"."ID_GENDER" IS NOT NULL)
   6 - access("A"."ID_PATIENT"="AP"."ID_PATIENT")
   8 - access("AW"."ID_STAF"="EW"."ID_STAF")
  10 - access(ROWID=ROWID)
  13 - access("AW"."ID_OFF"="DW"."ID_OFF")
  15 - access("A"."WADD_IDSECTOR"="AW"."ID_SECTOR")
  17 - access("A"."ID_APPNT_TYPE"="B"."ID_APPNT_TYPE")
  19 - access("A"."ID_SECTOR"="AST"."ID_SECTOR")
  22 - access("ES"."ID_STAF"="AST"."ID_STAF")
  26 - access("AST"."ID_OFF"="DS"."ID_OFF")
  29 - access(ROWID=ROWID)
  32 - access("ES"."ID_PERSON"="BS"."ID_PERSON")
  33 - filter("BS"."ID_GENDER" IS NOT NULL)
```

Рисунок 3.11 - Предикативна інформація щодо запиту

Також можна отримати статистику по запиту іншим шляхом.

DBID	SQL_ID	PLAN_HASH_VALUE	ID	OPERATION	OPTIONS	OBJECT_NODE	OBJECT#	OBJECT_OWNER	OBJECT_NAME	OBJECT_ALIAS	OBJECT_TYPE	OPTIMIZER	PARENT_ID	DEPTH	POSITION	SEARCH_COLUMNS	COST	CARDINALITY	BYTES
2269526665	gryf3m7wamxpz	851969126	0	SELECT STATEMENT								ALL_ROWS		0	1298	0	1298		
2269526665	gryf3m7wamxpz	851969126	1	SORT	ORDER BY								0	1	1	0	1298	2478	4269594
2269526665	gryf3m7wamxpz	851969126	2	HASH JOIN									1	2	1	0	402	2478	4269594
2269526665	gryf3m7wamxpz	851969126	3	TABLE ACCESS	FULL		74828	MHC	MHC_T_PERSON	BW@SEL$5	TABLE		2	3	1	0	19	2002	76076
2269526665	gryf3m7wamxpz	851969126	4	HASH JOIN									2	3	2	0	382	2478	4175430
2269526665	gryf3m7wamxpz	851969126	5	TABLE ACCESS	FULL		74828	MHC	MHC_T_PERSON	BP@SEL$3	TABLE		4	4	1	0	19	2002	92092
2269526665	gryf3m7wamxpz	851969126	6	HASH JOIN									4	4	2	0	363	2478	4061442
2269526665	gryf3m7wamxpz	851969126	7	INDEX	FULL SCAN		156935	MHC	MHC_I_PATIENT_FK01	AP@SEL$3	INDEX		6	5	1	0	1	304	2432
2269526665	gryf3m7wamxpz	851969126	8	HASH JOIN									6	5	2	0	361	2478	4041618
2269526665	gryf3m7wamxpz	851969126	9	VIEW				MHC	index$_join$_019	EW@SEL$5	VIEW		8	6	1	0	3	11	77
2269526665	gryf3m7wamxpz	851969126	10	HASH JOIN									9	7	1	0			
2269526665	gryf3m7wamxpz	851969126	11	INDEX	FAST FULL SCAN		74979	MHC	MHC_I_STAFF_FK01	indexjoin$_alias$_001@SEL$4985B87C	INDEX		10	8	1	0	1	11	77
2269526665	gryf3m7wamxpz	851969126	12	INDEX	FAST FULL SCAN		74981	MHC	MHC_I_STAFF_PK	indexjoin$_alias$_002@SEL$4985B87C	INDEX (UNIQUE)		10	8	2	0	1	11	77
2269526665	gryf3m7wamxpz	851969126	13	HASH JOIN									8	6	2	0	358	2478	4024272
2269526665	gryf3m7wamxpz	851969126	14	TABLE ACCESS	FULL		74827	MHC	MHC_T_OFFICES	DW@SEL$5	TABLE		13	7	1	0	3	7	294
2269526665	gryf3m7wamxpz	851969126	15	HASH JOIN									13	7	2	0	355	2478	3920196
2269526665	gryf3m7wamxpz	851969126	16	TABLE ACCESS	FULL		74826	MHC	MHC_T_SECTORS	AW@SEL$5	TABLE		15	8	1	0	19	11	187
2269526665	gryf3m7wamxpz	851969126	17	HASH JOIN									15	8	2	0	335	2478	3878070
2269526665	gryf3m7wamxpz	851969126	18	TABLE ACCESS	FULL		74820	MHC	MHC_T_TYPE_APPOINTMENTS	B@SEL$2	TABLE		17	9	1	0	19	5	290
2269526665	gryf3m7wamxpz	851969126	19	HASH JOIN									17	9	2	0	316	2478	3734346
2269526665	gryf3m7wamxpz	851969126	20	NESTED LOOPS									19	10	1	0			
2269526665	gryf3m7wamxpz	851969126	21	NESTED LOOPS									20	11	1	0	23	11	1144
2269526665	gryf3m7wamxpz	851969126	22	HASH JOIN									21	12	1	0	12	11	726

Рисунок 3.12 - Статистика по запиту

На ID# 3,5,14,16,18,25,35.

Таблиця, організована купою, не зберігає рядки в будь-якому конкретному порядку. Для цього запит розбирається більш детальніше.

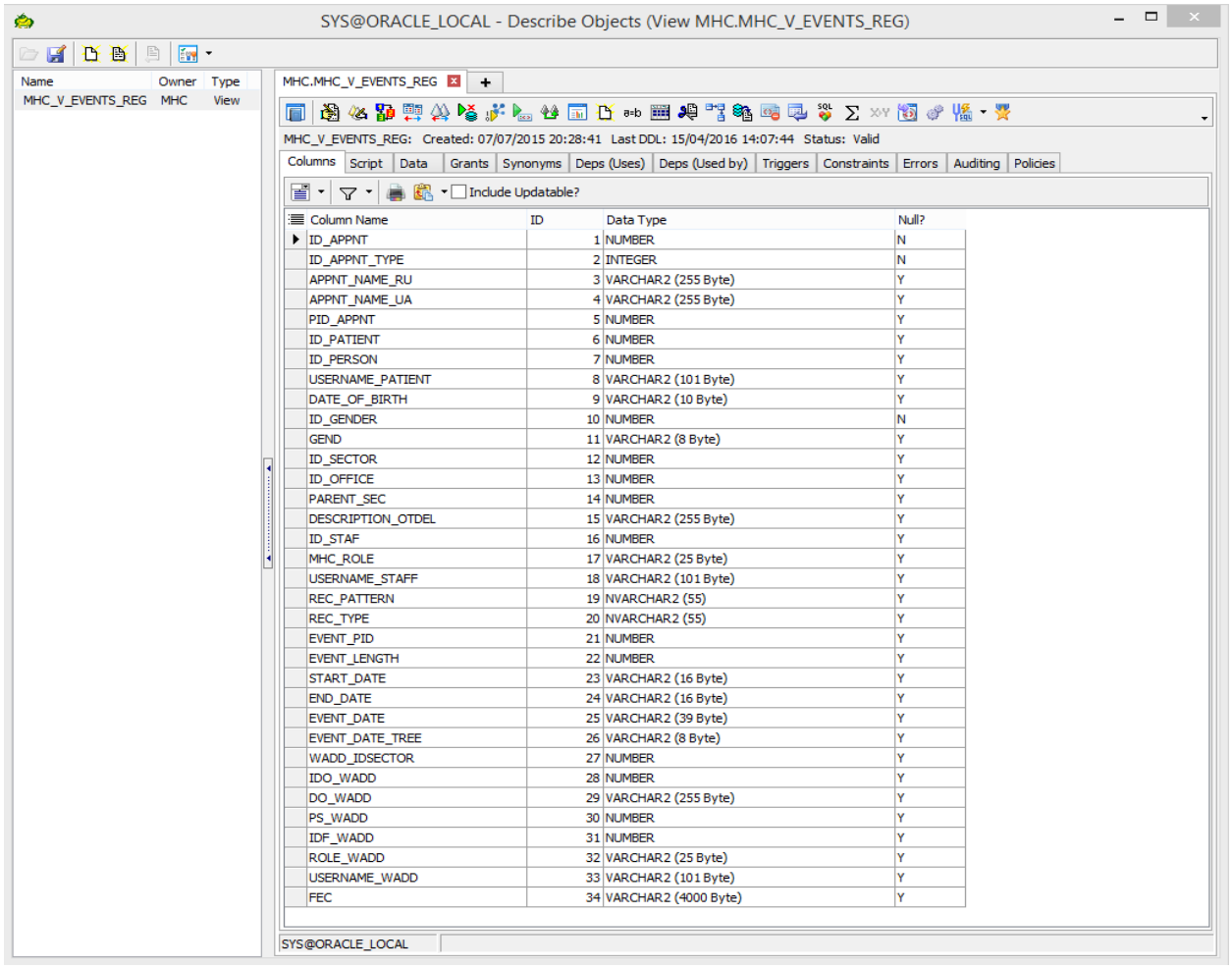


Рисунок 3.13 - Створення таблиці

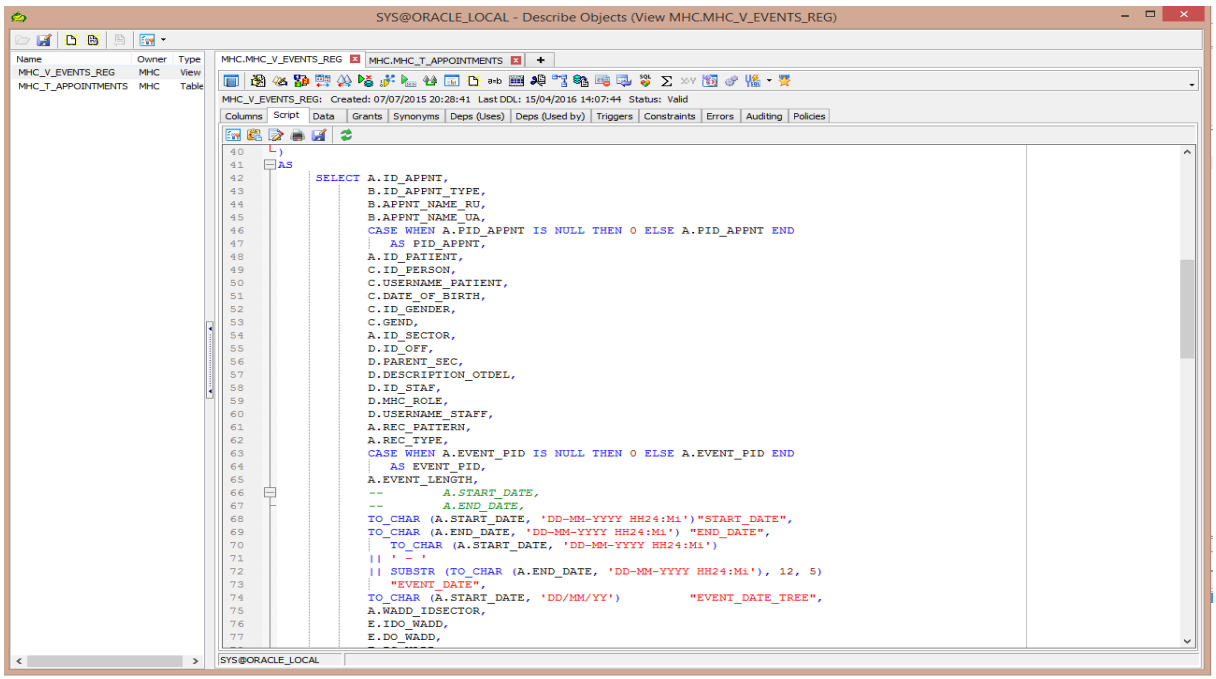


Рисунок 3.14 - SQL-script створеної таблиці

Вибирається SQL-script створеної таблиці. З запиту видно, що таблиця MHC.MHC_T_APPOINTMENTS займає більше часу. Використовуючи Optimizer Hints в SQL-запиті, необхідно досягти настройки швидкодії та економії енергії. Перевірка COST-значення з різними показниками Optimizer Hints для таблиць, чим менше значення - тим швидше працює запит.

```
set autotrace on
SELECT /*+ full (a) */ COUNT(*) FROM MHC.MHC_T_APPOINTMENTS a WHERE ID_SECTOR = 12;
SELECT COUNT(*) FROM MHC.MHC_T_APPOINTMENTS a WHERE ID_SECTOR = 12;
SELECT COUNT(*) FROM MHC.MHC_T_APPOINTMENTS a;

  COUNT(*)
----------
        36
1 row selected.

Execution Plan
----------------------------------------------------------
   0      SELECT STATEMENT Optimizer Mode=ALL_ROWS (Cost=292 Card=1 Bytes=3)
   1    0   SORT AGGREGATE (Card=1 Bytes=3)
```

```
  2    1     PARTITION RANGE ALL (Cost=292 Card=620 Bytes=1 K)
  3    2       TABLE ACCESS FULL MHC.MHC_T_APPOINTMENTS (Cost=292 Card=620 Bytes=1
K)

Statistics
----------------------------------------------------------
         0  recursive calls
         0  db block gets direct
         0  consistent gets - examination
         0  physical reads direct
         0  redo writes
         0  Workload Capture: unsupported user calls
         0  Workload Capture: errors
         0  Workload Replay: dbtime
         0  OS System time used
         0  OS Maximum resident set size
         1  rows processed

  COUNT(*)
----------
        36
1 row selected.

Execution Plan
----------------------------------------------------------
  0      SELECT STATEMENT Optimizer Mode=ALL_ROWS (Cost=2 Card=1 Bytes=3)
  1    0   SORT AGGREGATE (Card=1 Bytes=3)
  2    1     INDEX RANGE SCAN MHC.MHC_I_APPOINTMENTS_FK04 (Cost=2 Card=620 Bytes=1
K)

Statistics
----------------------------------------------------------
         0  recursive calls
         0  db block gets direct
         1  consistent gets - examination
```

```
        0  physical reads direct
        0  redo writes
        0  Workload Capture: unsupported user calls
        0  Workload Capture: errors
        0  Workload Replay: dbtime
        0  OS System time used
        0  OS Maximum resident set size
        1  rows processed

  COUNT(*)
----------
      2478
1 row selected.

Execution Plan
----------------------------------------------------------
   0      SELECT STATEMENT Optimizer Mode=ALL_ROWS (Cost=3 Card=1)
   1    0   SORT AGGREGATE (Card=1)
   2    1     INDEX FAST FULL SCAN MHC.MHC_I_APPOINTMENTS_PK (Cost=3 Card=2 K)

Statistics
----------------------------------------------------------
        0  recursive calls
        0  db block gets direct
        0  consistent gets - examination
        0  physical reads direct
        0  redo writes
        0  Workload Capture: unsupported user calls
        0  Workload Capture: errors
        0  Workload Replay: dbtime
        0  OS System time used
        0  OS Maximum resident set size
        1  rows processed
```

Також, використовуючи Oracle Enterprise Manager, можна дивитися статистику.

Нижче представлена статистика по активним сесіям, завантаженості ЦП, топовій активності по SQL-запитам.

Рисунок 3.15 - Статистика по активним сесіям, завантаженості ЦП, топовій активності по SQL-запитам (а)

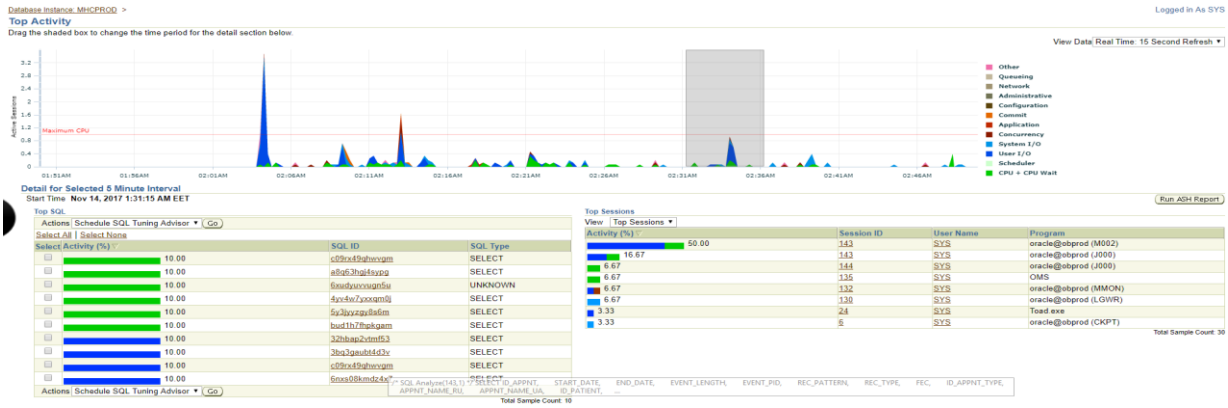


Рисунок 3.16 - Статистика по активним сесіям, завантаженості ЦП, топовій активності по SQL-запитам (б)

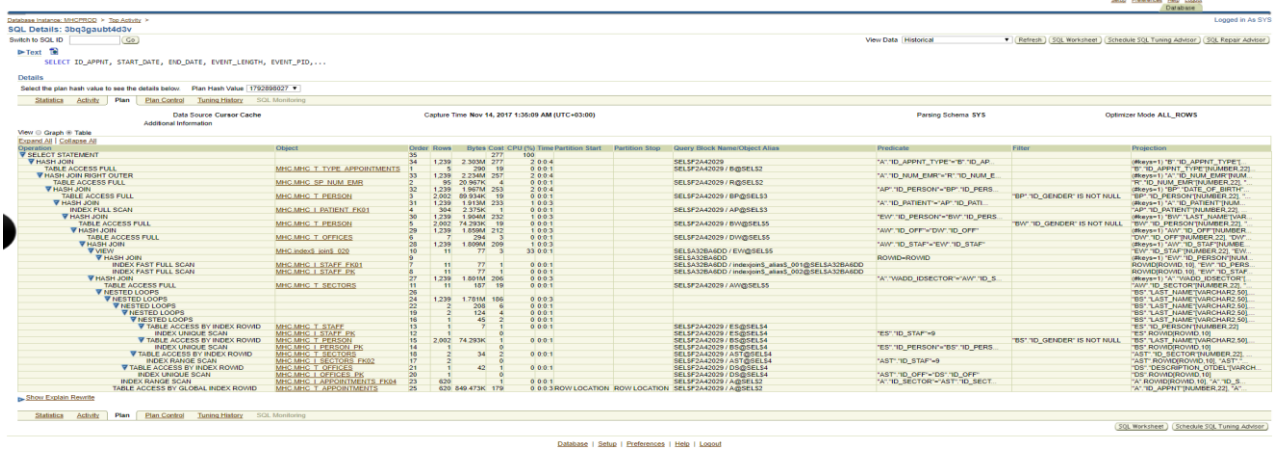


Рисунок 3.17 - Статистика по активним сесіям, завантаженості ЦП, топовій активності по SQL-запитам (в)

3.3 Висновки до третього розділу

У цьому розділі була розглянута самоналагоджувальна архітектура для моніторингу, настройки і тренда. За допомогою порівняння вартості апаратного забезпечення проти людських витрат, було доведено, що витрати на ручне налаштування перевищують витрати на прискорене обладнання. А це, в свою чергу означає, що самоналагоджування баз даних – є невід'ємною частиною економії часу та енергії на володіння СУБД.

Також був створений метод динамічного самонастроювання бази даних, та представлені результати пропонованої технології. Завдяки введеним запитам за заданий часовий проміжок була відображена статистика активних сесій, завантаженості ЦП, топової активності. Створені звіти по запитам у вигляді html-файла. Використовуючи Optimizer Hints в SQL-запиті, було досягнуто налаштування швидкодії та економії енергії. Проведена перевірка COST-значення з різними показниками Optimizer Hints для таблиць (при меншому значенні запит працює швидше).

3.4 Література до третього розділу

1. Foundations of Automated Database Tuning, VLDB '06, September 12–15, 2006, Seoul, Korea.Copyright 2006 VLDB Endowment, ACM
2. Rethinking Database System Architecture: Towards a Selftuning RISC-style Database System in Cairo, Egypt, 2000
3. AutoAdmin:Self-Tuning Database Systems Technology, Copyright 2006 IEEE
4. Self-Tuning Database Systems: A Decade of Progress, Copyright
5. I. Alagiannis, Towards Adaptive, Flexible, and Self-tuned Database Systems, (DIAS, I&C, EPFL) in EDIC-ru/05.05.2009
6. SQL Memory Management in Oracle9i, Hong Kong, China, 2002
7. D. Burleson, Oracle Tuning, The Definitive Reference Second Edition.
8. Hitesh KUMAR SHARMA, Aditya SHASTRI, Ranjit BISWAS, Architecture of Automated Database Tuning Using SGA Parameters, 2012

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ. ЕКОЛОГІЯ

В даному розділі проведено аналіз потенційних небезпечних та шкідливих виробничих факторів, причин пожеж. Розглянуті заходи, які дозволяють забезпечити гігієну праці і виробничу санітарію. На підставі аналізу розроблені заходи з техніки безпеки та рекомендації з пожежної профілактики.

Завданням даної магістерської роботи було розроблення моделей і методу інформаційної технології підвищення ефективності роботи баз даних в реальному часі, і як результат було створено метод динамічного самонастроювання бази даних на основі рівняння пропусків буфера, отриманого з аналітичної моделі, і адаптивної нейро-нечіткої методики обліку змін в розмірі бази даних при тривалому використанні програми бази даних. За цим методом в подальшому розроблятиметься реальна система, яка значно полегшить процес підвищення ефективності роботи баз даних в реальному часі. Так як в процесі проектування використовувалося програмне забезпечення, то аналіз потенційно небезпечних і шкідливих виробничих чинників виконується для персонального комп'ютера, на якому буде розроблений метод динамічного самонастроювання бази даних.

4.1 Аналіз небезпечних та шкідливих факторів при виробництві (експлуатації) виробу

На підставі аналізу небезпечних та шкідливих факторів при виробництві (експлуатації), пожежної безпеки можуть бути надалі вирішені питання необхідності забезпечення працюючих достатньою кількістю освітлення, вентиляції повітря, організації заземлення, тощо.

Роботу, пов'язану з ЕОП з ВДТ, у тому числі на тих, які мають робочі місця, обладнані ЕОМ з ВДТ і ПП, виконують із забезпеченням виконання НПАОП 0.00.-1.28-10 «Правил охорони праці під час експлуатації електронно-обчислювальних машин», які встановлюють вимоги безпеки до обладнання робочих місць, до роботи із застосуванням ЕОМ з ВДТ і ПП. Переважно роботи за проектами виконують у кабінетах чи інших приміщеннях, де використовують різноманітне електрообладнання, зокрема персональні комп’ютери (ПК) та периферійні пристрої.

Основними робочими характеристиками персонального комп'ютера є наступні:

– робоча напруга U = +220В ± 5%;
– робочий струм I = 2А;
– споживана потужність P = 350 Вт.

Робочі місця мають відповідати вимогам Державних санітарних правил і норм роботи з візуальними дисплейними терміналами електронно-обчислювальних машин, затверджених постановою Головного державного санітарного лікаря України від 10.12.98 N 7 [1].

За умов роботи з ПК виникають наступні небезпечні та шкідливі чинники: несприятливі мікрокліматичні умови, освітлення, електромагнітні випромінювання, забруднення повітря шкідливими речовинами (джерелом, яких можуть бути: принтер, сканер та інші джерела виділення багатьох хімічних речовин - напр., озону, оксидів азоту та аерозолів високодисперсних частинок тонера), шум, вібрація, електричний струм, електростатичне поле, напруженість трудового процесу та інше.

Аналіз небезпечних та шкідливих виробничих факторів виконується у табличній формі (табл. 4.1).

Таблиця 4.1 – Аналіз небезпечних і шкідливих виробничих факторів

Небезпечні і шкідливі виробничі фактори	Джерела факторів (види робіт)	Кількісна оцінка	Нормативні документи
1	2	3	4
фізичні			
- підвищена температура поверхонь обладнання	експлуатація ЕОМ, принтерів, сканерів чи/або серверного обладнання для роботи	2	[2]
- підвищений рівень шуму на робочому місці	-//-	2	[3]
- підвищений рівень вібрації	-//-	2	[4] [5]
- підвищена або знижена вологість повітря	-//-	2	[2]
- підвищена або знижена рухливість повітря	-//-	1	[2]
- підвищений рівень іонізуючого випромінення в робочій зоні	-//-	2	[2] [6]
- підвищений рівень електромагнітного випромінення	-//-	2	[6]
- підвищений рівень напруги електричної мережі, замикання якої може відбутися через тіло людини	-//-	4	[7] [8]
- підвищений рівень статичної електрики	-//-	2	[7]
- підвищена напруженість електричного поля	-//-	2	[6]
- підвищена	-//-	2	[6]

напруженість магнітного поля			
- недостатність природного світла	порушення умов праці (вимог до приміщень)	2	[9]
- недостатнє освітлення робочої зони	порушення гігієнічних параметрів виробничого середовища	3	[9]
- підвищена яскравість світла	порушення умов праці (організації місця праці- налагодження моніторів)	1	[10]
- понижена контрастність	-//-	1	[10]
хімічні:			
- загазованість повітря робочої зони, яка впливає на організм людини через органи дихання та надає токсичну і канцерогенну дію	від експлуатації сканерів, принтерів для роботи – O_3, оплавлення електричних і комутаційних кабелів, резисторів, конденсаторів, напівпровідникових діодів, транзисторів й інше в ЕОМ та системах кондиціонування повітря - CO, CO_2, SO_2, P_2O_5, H_2S, HCl, H, NH_3, ClF_3, F_2O_2, F_2O_3, SeO_2. SeF_6, TeF_6, $COCl_2$, SO_2F_2, інш.	3	[11] [12] [13] [14]
психофізіологічні:			
- нервово-психічна перевантаження (розумове, перенапруження аналізаторів-зорових)	- пошук інформації для постановки теми; - пошук та аналіз аналогів і літератури; - пошук наявних технологій, моделювання та аналіз алгоритмів; - виконання роботи за темою диплома, тестування; - оформлення роботи	4	[15] [10]
- фізичні (статичне – сидіння)	порушення умов праці (організації місця праці- сидіння користувача,) та організації робочого часу - безпервна робота)	2	[15] [10]

4.2 Гігієнічні вимоги до параметрів виробничого середовища

4.2.1 Мікроклімат

Оптимальні значення для температури, відносної вологості й рухливості повітря для зазначеного робочого місця відповідають [2] і наведені в табл. 4.2:

Таблиця 4.2 – Норми мікроклімату робочої зони об’єкту

Період року	**Категорія робіт**	**Температура С^0**	**Відносна вологість %**	**Швидкість руху повітря, м/с**
Холодна	легка-1 а	22 - 24	40 – 60	0,1
Тепла	легка-1 а	23 - 25	40 – 60	0,1

Дане приміщення обладнане системами опалення, кондиціонування повітря або припливно-витяжною вентиляцією. У приміщенні на робочому місці забезпечуються оптимальні значення параметрів мікроклімату: температури, відносної вологості й рухливості повітря у відповідності до [2]. Рівні позитивних і негативних іонів у повітрі мають відповідати [2]. Для забезпечення оптимальних параметрів мікроклімату в приміщенні проводяться перерви в роботі співробітників, з метою його провітрювання. Існують спеціальні системи кондиціонування, які забезпечують підтримання в приміщенні балансу оптимальних параметрів мікроклімату. Контроль параметрів мікроклімату в холодний і теплий період року здійснюється не менше 3-х разів на зміну (на початку, середині, в кінці).

4.2.2 Освітлення

У приміщенні, де розташовані ЕОМ передбачається природне бічне освітлення, рівень якого відповідає ДБН В. 2.5-28:2015 [16]. Джерелом природного освітлення є сонячне світло. Регулярно повинен проводитися контроль освітленості, який підтверджує, що рівень освітленості задовольняє ДБН і для даного приміщення в світлий час доби достатньо природного освітлення.

Розрахунок освітлення.

Для будівель виробництв світловий коефіцієнт приймається в межах 1/6 - 1/10:

$$\sqrt{a^2+b^2}\cdot S_b = (1/8 \div 1/10)\cdot S_n \qquad (4.1)$$

де S_b – площа віконних прорізів, м^2;

S_n – площа підлоги, м^2.

$S_n = a \cdot b = 5 \cdot 5 = 25$ м^2

$S_{вік} = 1/8 \cdot 25 = 3{,}125$ м^2

Приймаємо 2 вікна площею $S = 1{,}6$ м^2 кожне.

Світильники загального освітлення розташовуються над робочими поверхнями в рівномірно-прямокутному порядку. Для організації освітлення в темний час доби передбачається

обладнати приміщення, довжина якого складає 5 м, ширина 5 м, світильниками ЛПО2П, оснащеними лампами типа ЛБ (дві по 80 Вт) з світловим потоком 5400 лм кожна.

Розрахунок штучного освітлення виробляється по коефіцієнтах використання світлового потоку, яким визначається потік, необхідний для створення заданої освітленості при загальному рівномірному освітленні. Розрахунок кількості світильників *n* виробляється по формулі (4.2):

$$n = \frac{E \cdot S \cdot Z \cdot K}{F \cdot U \cdot M} \tag{4.2}$$

де *E* – нормована освітленість робочої поверхні, визначається нормами – 300 лк;

S – освітлювана площа, м^2; S = 25 м^2;

Z – поправочний коефіцієнт світильника (Z = 1,15 для ламп розжарювання та ДРЛ; Z = 1,1 для люмінесцентних ламп) приймаємо рівним 1,1;

K – коефіцієнт запасу, що враховує зниження освітленості в процесі експлуатації – 1,5;

U – коефіцієнт використання, залежний від типу світильника, показника індексу приміщення і т.п. – 0,575

M – число люмінесцентних ламп в світильнику – 2;

F – світловий потік лампи – 5400лм (для ЛБ-80).

Підставивши числові значення у формулу (4.2), отримуємо:

$$\mathrm{n} = \frac{300 \cdot 25 \cdot 1{,}1 \cdot 1{,}5}{5400 \cdot 0{,}575 \cdot 2} \approx 2.$$

Приймаємо освітлювальну установку, яка складається з 2-х світильників, які складаються з двох люмінесцентних ламп загальною потужністю 160 Вт, напругою – 220 В.

4.2.3 Шум та вібрація, електромагнітне випромінювання

Рівень шуму, що супроводжує роботу користувачів персональних комп’ютерів (зумовлений як роботою системних блоків, клавіатури, так і друкуванням на принтерах, а також зовнішніми чинниками), коливається у межах 50–65 дБА [3]. Шум такої інтенсивності на тлі високого ступеня напруженості праці негативно впливає на функціональний стан користувачів. Тому на практиці рекомендують знижувати фактичний рівень шуму у приміщеннях, де створюють комп’ютерні програми, виконують теоретичні та творчі роботи, проводять навчання до 40 дБА, а в приміщеннях, де виконують роботу, що потребує зосередженості, — до 55 дБА. У залах опрацювання інформації та комп’ютерного набору рівні шуму не повинні перевищувати 65 дБА.

Шум часто є причиною зниження рівня працездатності, підвищення рівня загальної та професійної захворюваності, частоти виробничих травм. Шум є загальнобіологічним подразником, який негативно впливає на всі органи і системи організму. У разі тривалого систематичного впливу шуму може виникнути патологія з переважним ураженням слуху, центральної нервової і серцево- судинної систем.

Для зниження шуму на шляху його поширення передбачається розміщення в приміщенні штучних поглиначів. Для зниження рівня шуму стелю або стіни вище 1.5 - 1.7 метра від підлоги повинні облицьовуватися звукопоглинальним матеріалом з максимальним коефіцієнтом звукопоглинання в області частот 63-8000 Гц. Додатковим звукопоглинанням в КВТ можуть бути фіранки, підвішені в складку на відстані 15-20 см. Від огорожі, виконані з щільної, важкої тканини. У приміщенні з ЕОМ коректований рівень звукової потужності не перевищує 45 дБА. Оскільки рівень шуму не перевищує гранично допустимих величин, які встановлені санітарними нормами, заходи для зниження шуму не проводяться.

Віброізоляція можливо здійснювати за допомогою спеціальної прокладки під системний блок, який послаблює передачу вібрацій робочого столу. Вібрація на робочому місці в приміщенні, що розглядається, відповідає нормам [3]. Допустимий рівень вібрацій на робочому місці: для 1 ступеня шкідливості до 3 дБ; для 2-3 - 1-6 дБ; для 3 - більше 6 дБ.

Для захисту від електромагнітного випромінювання передбачаються наступні заходи:

1) застосування нових плазмових моніторів, LG W2271TC,

2) віддалення робочого місця не менше, ніж на 0,4-0,5 м, оскільки напруженість електричного поля зменшується при віддаленні від джерела поля,

3) встановлення раціональних режимів роботи персоналу (обмеження часу перебування),

4) раціональне розміщення в робочому приміщенні устаткування, що випромінює електромагнітну енергію.

4.2.4 Вентилювання

У приміщенні, де знаходяться ЕОМ, повітрообмін реалізується за допомогою природної організованої вентиляції (вентиляційні шахти) і установки в віконному отворі автономного кондиціонера БК-2000. Цей метод забезпечує приток потрібної кількості свіжого повітря, що визначається в СНіП (30 $м^3$ на годину на одного працюючого).

Також має здійснюватися провітрювання приміщення, в залежності від погодних умов, тривалість повинна бути не менше 10 хв. Найкращий обмін повітря здійснюється при наскрізному провітрюванні.

4.3 Заходи з організації виробничого середовища та попередження виникнення надзвичайних ситуацій

Розрахунок захисного заземлення (забезпечення електробезпеки будівлі).

Загальний опір захисного заземлення визначається за формулою:

$$R_{ззп} = \frac{R_з \cdot R_п}{R_п \cdot n \cdot \eta_з + R_з \cdot \eta_п}, \quad (4.3)$$

де Rз - опір заземлення, якими когут бать труби, опори, кути і т.п., Ом;

Rш - опір опори, яке з'єднує заземлювачі, Ом;

п - кількість заземлювачів;

ηз - коефіцієнт екранування заземлювача; приймається в межах 0,2 ÷ 0,9; ηз = 0,7

ηш - коефіцієнт екранування сполучної стійки; приймається в межах 0,1 ÷ 0,7; ηш = 0,5;

Опір заземлення визначається за формулою:

$$R_з = \frac{\rho}{2\pi \cdot l} \cdot \left(\ln \frac{2 \cdot l}{d} + \frac{1}{2} \ln \frac{4 \cdot t + l}{4 \cdot t - l} \right), \quad (4.4)$$

де ρ - питомий опір ґрунту, залежить від типу грунту, Ом·м;

для піску - 400 ÷ 700 Ом·м; приймаємо ρ = 400 Ом·м;

l - довжина заземлювача, м; для труб - 2-3 м; l = 3 м;

d - діаметр заземлювача, м; для труб - 0,03-0,05 м; d = 0,05 м;

t - відстань від середини забитого в ґрунт заземлювача до рівня землі, м; t = 2 м.

$$R_з = \frac{400}{2 \cdot 3{,}14 \cdot 3} \left(\ln \frac{2 \cdot 3}{0{,}05} + \frac{1}{2} \ln \frac{4 \cdot 2 + 3}{4 \cdot 2 - 3} \right) = 110, \text{ Ом}$$

Опір смуги, що з'єднує заземлювачі, визначається за формулою:

$$R_ш = \frac{\rho}{2\pi \cdot L} \cdot \ln \frac{2 \cdot L^2}{b \cdot t^1}, \quad (4.5)$$

де L - довжина смуги, що з'єднує заземлювачі (м) і приблизно дорівнює периметру будівлі: $П_{буд.}$ = 42·2 + 38·2 = 160 м; L = 160 м;

b - ширина смуги, м; b = 0,03 м;

t_1 - глибина заземлення від рівня землі, м; t_1 = 0,5 м.

$$R_n = \frac{400}{2 \cdot 3{,}14 \cdot 160} \cdot \ln \frac{2 \cdot 160^2}{0{,}03 \cdot 0{,}5} = 5{,}99, \text{ Ом}$$

Кількість заземлювачів захисного заземлення визначається за формулою:

$$n = \frac{2 \cdot R_з}{4 \cdot \eta_з}, \tag{4.6}$$

де 4 - допустимий загальний опір, Ом;

2 - коефіцієнт сезонності.

Визначаємо загальний опір захисного заземлення:

$$R_{ззп} = \frac{110 \cdot 5{,}99}{5{,}99 \cdot 79 \cdot 0{,}7 + 110 \cdot 0{,}5} = 1{,}7 \text{ Ом}$$

Висновок: дане захисне заземлення буде забезпечувати електробезпеку будівлі, так як виконується умова: $R_{ззп} < 4$ Ом.

3) При виникненню пожеж при роботі на ПЕОМ від таких можливими джерел запалювання як:

– іскри і дуги коротких замикань;

– перегрів провідників, резисторів та інших радіодеталей ПЕОМ, від тривалої перевантаження та наявність перехідного опору;

– іскри при розмиканні і розмиканні ланцюгів;

– розряди статичної електрики;

– необережному поводженню з вогнем, а також вибухи газо-повітряних і паро-повітряних сумішей.

Важливу увагу слід звернути на пожежну безпеку підприємства в цілому і окремих його приміщень. В приміщеннях не повинно накопичуватися сміття, непотрібний папір, мотлох та ін. речі, які не використовуються у виробничому процесі. Наявний вільний аварійний вихід за межі

приміщення в разі пожежі, бути передбачені вогнегасники. Вони повинні бути в робочому стані і перевірятися згідно з нормами. У приміщеннях повинна бути пожежна сигналізація, вогнегасник. У разі виникнення пожежі необхідно повідомити в найближчу пожежну частину, убезпечити інших працівників і по можливості прийняти кроки по запобіганню можливих наслідків та усуненню пожежі.

4.4 Охорона навколишнього природного середовища

4.4.1 Загальні дані з охорони навколишнього природного середовища

Діяльність за темою магістерської роботи, а саме: створення прототипу інформаційної системи в процесі її виконання впливає на навколишнє природнє середовище і регламентується нормами діючого законодавства: Законом України «Про охорону навколишнього природного середовища», Законом України «Про забезпечення санітарного та епідемічного благополуччя населення», Законом України «Про відходи», Законом України «Про охорону атмосферного повітря», Законом України «Про захист населення і територій від надзвичайних ситуацій техногенного та природного характеру», Водний кодекс України.

Основним екологічним аспектом в процесі діяльності за даними спеціальностями є процеси впливу на атмосферне повітря та процеси поводження з відходами, які утворюються, збираються, розміщуються, передаються на відалення (знешкодження), утилізацію, тощо в ІТ галузі.

Вплив на атмосферне повітря при нормальних умовах праці не оказує, бо не має в приміщенні сканерів, принтерів та інших джерел викиду забруднюючих речовин в повітря робочої зони.

В процесі діяльності створення прототипу інформаційної системи виникають процеси поводження з відходами ІТ галузі. Нижче надано перелік відходів, що утворюються в процесі роботи:

1) Змінні носії інформації - IV клас небезпеки
2) Відходи друкуючих пристроїв - IV клас небезпеки
3) Макулатура - IV клас небезпеки
4) Побутові відходи - IV клас небезпеки

4.4.2 Визначення впливу та заходів щодо поводження з відходами ІТ галузі

З метою визначення та прогнозування впливу відходів на навколишне середовище, своєчасного виявлення негативних наслідків, їх запобігання відповідно до Закону України «Про

відходи» повинен здійснюватися моніторинг місць утворення, зберігання, і видалення відходів. Відомості про місце утворення та місце розташування відходів зазначаються на «План схемі місці розміщення відходів організації / виробництва» та наводяться у таблиці 4.3, а Відомості про склад і властивості відходів, що утворюються, а також ступінь їх небезпечності для навколишнього природного середовища та здоров'я людини у табл. 4.4.

Таблиця 4.3 - Відомості про місце утворення та місце розташування відходів

№ з/п	Код та найменування відходів за ДК -005-96	Технологічний процес або виробництво, де утворюються відходи / клас небезпеки	Місце розташування відходу, тара та її кількість, місткість, розміри у разі наявності майданчиків розташування відходів необхідно зазначити тип покриття та наявність даху)	№ на схемі (додається масштабна схема місць розміщення відходів)
1	2	3	4	5
1	7720.3.1.01 Відходи комунальні (міські) змішані, у т.ч. сміття з урн (Побутові відходи)	4	зовнішній майданчик зберігання побутових відходів біля буд .84 S=5м² V= 2,08м³- 2од.	8401-ТХ
2	7710.3.1.01 Макулатура паперова та картонна (Макулатура)	4	буд .84 4 поверх в кім. 412 S =5,0 м.^{2}	8401-ТХ
3	Змінні носії інформації	4	буд. 84, кім. 412 V=0,0005 м^3	8401-ТХ
4	Відходи друкуючих пристроїв.	4	буд. 84, кім. 412 V=1,0 м^3	8401-ТХ

Таблиця 4.4 – Відомості про склад і властивості відходів, що утворюються, а також ступінь їх небезпечності для навколишнього природного середовища та здоров'я людини

№ п/п	Назва відходу	Клас небезпечності	Хімічний (у долях відсотків складників або інших одиницях виміру) та морфологічний склад	Фізико-хімічні властивості	Негативний вплив на навколишнє середовище та здоров'я людини
1	2	3	4	5	6
1	Макулатура	IV	**Цинк** - 0,000053 – 0,000056 **Zn** **Свинець** - 0,000049 – 0,000051 **Pb** **Хром** - 0,000051 – 0,000054 **Cr** **Мідь** - 0,000033 – 0,000035 **Cu**	Уривки та обрізки з паперових мішків **Цинк** $T_{кип.}$= 913°C $T_{плав.}$= 4,19°C **Свинець** $T_{кип.}$= 1751°C $T_{плав.}$= 327,3°C **Хром** $T_{кип.}$= 1890°C $T_{плав.}$= 2480°C **Мідь** $T_{кип.}$= 2580°C	Негативний вплив на ОС і людини визначається його хімічним складом. **цинк** Малотоксичний для теплокровних тварин при надходженні з їжею і питної водою-концентрація в питній воді 11,2 ... 26,6 мг / л переноситься без будь-яких ознак інтоксикації. Дуже корисний для флори, будучи одним з найважливіших мікроелементів харчування, однак лише в концентрації до 0,2 мг / л, крім того, елемент силяється до кумуляції в грантах. Дуже токсичний для водних організмів, порушуючи процеси самоочищення водойм і стаючи токсичним для іхтіофауни в концентрації 0,15 ... 5,0 мг / л. Мутагенна і онкогенна небезпеку [17]. **свинець** У природних водах міститься в концентрації 0,001 - 0,023 мг / л. У концентрації 2,0 мг / л надає воді металевий присмак. Можливо має мутагенну і канцерогенну дію, значно збільшує токсичну дію інших металів. В конценрації 1,90 мг / л згубно діє на дафній, концентрація 0,1 мг / л погіршує процеси самоочищення водойм. Свинець токсичний для рослин в концентрації понад 5,0 мг / кг ґрунту. Помірно токсичний. Викликає хронічне отруєння. Має здатність вражати центральну і периферичну нервову систему, кістковий мозок і кров, судини, синтез білка, генетичний апарат клітини [17]. **хром** Міститься в природних водах в

			Целюлоза - 97,299814 – 96,999804 **$(C_6H_{10}O_5)_n$** **Вода** - 2,7 – 3,0	$T_{плав.}$= 1083°C **Целюлоза** $T_{возг. с обуглив.} \geq$ 100°C	концентрації 0,001 ... 0,112 мг / л. LK50 Cr (VI) для риб-30,0 ... 50,0 мг / л, LK50 Cr (III) для риб- 117,0 мг / л. Низькі концентрації хрому позитивно впливають на ріст рослин, проте полив водою С / Г культур з концентрацією хрому 10,0 ... 50,0 мг / л гальмує їх розвиток. На тварин надає загально токсичне, подразнююче, кумулятивне, алергенну, канцерогенну і мутагенну дію. Володіє канцерогенними властивістю [17] **мідь** У природних водах міститься в концентраціях 0,001 ... 0,98 мг / л. У концентрації 0,5 мг / л забарвлює воду, в концентрації> 1,0 мг / л-помітно збільшує мутність води. Дуже токсична як для водних організмів, так і для рослин. У концентрації 0,001 мг / л гальмує розвиток синьо зелених водоростей, LK50 практично для всіх видів риб становить 0,18 ... 1,35 мг / л (короп, карась, окунь, щука, сом). Куммулюється ґрунтом і рослин-ями. У концентрації 0,1 ... 0,2 мг / л надає токсичну дію на ріст рослин. Високотоксичний метал. Викликає гостре отруєний-ня, має широкий спектр токсичної дії [17] **целюлоза** Нетоксична. Досить легко підвержен біодеструкції лігнін- і целюлозоруйнуючими бактеріями і деякими класами низших грибів. У зв'язку з нетоксичністю LD50 для тваринах не установлена. Токсичність визначається за вмістом важких металів, здатних мігрувати з неї в навколишнє середовище. При попаданні на ґрунт, в воду і атмосферне повітря чинить негативний вплив на ОС і здоров'я людини.
2	Побутові відходи	IV	Побутові відходи - 100 – 100, в т. ч.: Папір -30 - 17; [**$(C_6H_{10}O_5)_n$** - целюлоза]	Целюлоза $T_{возг. с обуглив.} \geq$ 100°C	Негативний вплив на ОС і людини визначається його хімічним складом. **целюлоза** Нетоксична. Досить легко піддавав біодеструкції лігнін- і целюлозоруйнуючими бактеріями і деякими класами низших грибів. У зв'язку з нетоксичністю LD50 для тваринах не установлена. Токсичність визначається за вмістом важких металів, здатних мігрувати з неї в навколишнє середовище

			Поліетилен -20 – 24; **(- CH_2 - CH_2 -)$_n$** Деревина -5 – 3; [**($C_6H_{10}O_5$)$_n$** - целюлоза, лігнін] Матеріали текстильні -4 – 3; [**($C_6H_{10}O_5$)$_n$** - целюлоза Мінеральні домішки (пісок, глина) -4 – 9 Харчові відходи -37 –44;	Поліетилен - $T_{размяг.}$ ≥ 150°C Твердий матеріал рослинного походження, не розчиняється у воді. Целюлоза, лігнін $T_{возг. с обуглив.}$ ≥ 120°C Твердий матеріал рослинного походження, не розчиняється у воді. Целюлоза $T_{возг. с обуглив.}$ ≥ 100°C Харчові відходи $T_{биоразл.}$ ≥ 4° C	**поліетилен** Нетоксичний для всіх видів флори і фауни в зв'язку з дуже високою біологічною інертністю. Нерозчинний у водних середовищах і не впливає на санітарний режим водойм. Використання його не вимагає запобіжних заходів. Отруєння можливі при виробництві та переробці плівки, в результаті виділення окису вуглецю, альдегідів, органічних кислот [18]. **деревина** Нетоксична. Досить легко піддається біодеструкції лігнін- і целюлозоруйнуючими бактеріями і деякими класами нижчих грибів. У зв'язку з нетоксичністю LD50 для тварин не встановлена. Деревина нетоксична при використання. Але дія деревного пилу при рубці і переробці деревини викликає захворювання дихальних шляхів і шкіри. текстильне волокно Нетоксична в зв'язку з біогенним походженням, проте для біодеструкції необхідна наявність вологи. Нетоксична при використанні. Токсична дія виникає (як результат механічні дії - наслідок пилу) при виробництві тканів і при переробці вторинних матеріалів; слабкий алерген [19]. **Глина** нетоксична

4.4.3 Висновки до четвертого розділу

В результаті проведеної роботи було зроблено аналіз умов праці, шкідливих та небезпечних чинників, з якими стикається робітник. Було визначено параметри і певні характеристики приміщення для роботи над запропонованим проектом написаному в дипломній роботі, описано, які заходи потрібно зробити для того, щоб дане приміщення відповідало необхідним нормам і

було комфортним і безпечним для робітника. Приведені рекомендації щодо організації робочого місця, а також важливу інформацію щодо пожежної та електробезпеки. Була наведена схема, розміри приміщення та наведено значення температури, вологості й рухливості повітря, необхідна кількість і потужність ламп та інші параметри, значення яких впливає на умови праці робітника, а також – наведені інструкції з охорони праці, техніки безпеки при роботі на комп'ютері.

А також визначені основні екологічні аспекти впливу на навколишнє природнє середовище та зазначені заходи щодо поводження з ними.

4.4.4 Література до четвертого розділу

1.	ДСанПіН 3.3.2.007-98	Гігієнічні вимоги до організації роботи з візуальними дисплейними терміналами електронно-обчислювальних машин
2.	ДСН 3.3.6.042-99	Санітарні норми мікроклімату виробничих приміщень
3.	ДСН 3.3.6.037-99	Санітарні норми виробничого шуму, ультразвуку та інфразвуку
4.	ДСН 3.3.6.039-99	Санітарні норми виробничої загальної та локальної вібрації
5.	ДСТУ ГОСТ 12.1.012-90	ССБТ. Вибрационная безопасность. Общие требования
6.	ГОСТ 12.1.006-84	ССБТ. Электромагнитные поля радиочастот. Общие требования безопасности. Допустимые уровни на рабочих местах и требования к проведению контроля.
7.	ГОСТ 12.1.030-81	ССБТ. Электробезопасность.Защитное заземление. Зануление
8.	ГОСТ 13109-97	„Электрическая энергия. Совместимость технических средств электромагнитных. Нормы качества электроэнергоснабжения общего назначения”
9.	ДБН В.2.5-28:2015	Природнє і штучне освітлення
10.	ДСанПіН 3.3.2.007-98	Гігієнічні вимоги до організації роботи з візуальними дисплейними терміналами електронно-обчислювальних машин
11.	НПАОП 40.1-1.21-98	Правила безпечної експлуатації електроустановок споживачів
12.	ДБН В.2.5-67:2013	Опалення, вентиляція та кондиціонування
13.	ГОСТ 12.1.005-88.	ССБТ. Загальні санітарно-гігієнічні вимоги до воздуху робочої зони
14.	ГОСТ 12.1.044-89	ССБТ. Пожаровзрывоопасность веществ и материалов. Номенклатура показателей и методы их определения.
15.	НПАОП 0.00-1.28-10	Правила охорони праці під час експлуатації електронно-обчислювальних машин
16.	НПАОП 0.00-4.15-98	Про розробку інструкцій з охорони праці
17.	Закон України	Про охорону навколишнього природного середовища
18.	ДСТУ ISO 1A001:2006 (ISO 1A001:200A)	Системы экологического управления. Требования и руководящие указания по применению
19.	Кодекс	Кодекс законів про працю України

Висновки та рекомендації

За результатами дослідження методів підвищення ефективності роботи баз даних в реальному часі сформовано наступні висновки:

1) Вплив різних методів на налаштування СУБД має як свої переваги, так і недоліки. Тому необхідно досконально проаналізувати декілька методів, щоб вибрати конкретний в залежності від очікувань.

2) Самоналаштування СУБД, на відміну від ручного налаштування, має більші переваги.

Налаштування продуктивності бази даних - це процес її коригування так, що база даних буде функціонувати у повній мірі з урахуванням її поточного або сукупного навантаження. Найбільш сучасні системи баз даних мають динамічно настроювані параметри, що дозволяють встановлювати онлайн-налаштування без втрати часу. Проте вхідні дані дуже неточні, залежні від часу та навантаження, й існує постійна вимога формулювати політику настройки, яка визначає, які параметри слід оптимізувати, до якої міри і коли. Як правило, налаштування за замовчуванням викликають надмірні налаштування чи слабкі настройки системи, що призводить до поганої продуктивності системи та неефективного використання ресурсів. У результаті спостерігається деградація продуктивності та поганий час відгуку. Ці фактори значно зменшують продуктивність бази даних та збільшують витрати на енергію. Налаштування бази даних – задача не з легких. Але для кожного підприємства, працюючого з СУБД, важливо вирішити цю задачу. Найчастіше для виконання налаштування запрошується адміністратор бази даних. При цьому підприємство несе великі витрати на DBA та може залишитись без бажаного результату. Тому з’явився другий вихід – самоналаштування СУБД. Системи самоналаштування повністю автоматизовані, тому зникає необхідність у експертному DBA. Отже, витрати часу та грошей на ручне налаштування теж зникають.

Дані для сучасних підприємств і управління базами даних часто пов'язані з комплексним плануванням, управлінням часом і реалізацією стандартних завдань системи. Автоматизація баз даних допомагає підприємствам краще управляти своїми операціями з базами даних, скорочуючи час простою, а також загальний час, що витрачається на управління базою даних. Автоматизація в будь-якому місці працює з будь-якою базою даних SQL, такими як Oracle, MS SQL, Sybase, SQL DB2 і т. д. На відміну від інших рішень для автоматизації, вона не вимагає значного навчання. Проста, зручна у використанні, але потужна, вона може автоматизувати будь-яке завдання бази даних.

Oracle управляє потребами оперативної пам'яті відповідно до вимог кожного завдання, використовуючи складні алгоритми для підвищення швидкості роботи оперативної пам'яті. Oracle

DBA може динамічно деактивувати оперативну пам'ять, а також перерозподіляти її. Але оскільки адміністратор бази даних є нормальною людиною, він не може розрахувати фактичний обсяг оперативної пам'яті, необхідний додатку. Через це обмеження DBA розподіл ОЗУ вручну для оптимізації продуктивності системи баз даних стає складним і дорогим завданням.

На початку роботи була поставлена мета: вирішити задачу покращення характеристик самонастроювання для СУБД. Для досягнення цієї мети були досліджені існуючі методи і підходи до самонастроювання в СУБД; досліджений вплив різних стратегій настройки продуктивності БД і енергоефективності; розроблений метод динамічного самонастроювання бази даних на основі рівняння пропусків буфера, отриманого з аналітичної моделі, і адаптивної нейро-нечіткої методики обліку змін в розмірі бази даних при тривалому використанні програми бази даних.

Результати показали, що спостереження за статистикою СУБД протягом деяких проміжків часу, дозволяє вчасно підлаштовувати параметри СУБД під необхідний результат. Іноді достатньо стандартного встановленого помічника, але якщо профіль обраний невірно, або ж кількість користувачів БД збільшується - в цьому випадку потрібно робити настройку в ручному режимі, змінюючи параметри. Але кращим виходом із ситуації буде конфігурація нової установки бази даних вже з відомими показниками та відновлення даних. У цьому випадку можна розвантажити основну базу даних і поступово перенаправляти на нову базу даних, збираючи статистику. Завантаженість ЦП, без сумніву, дуже важливий показник. Адже при завантаженості ЦП вище 80-95% швидше за все база даних перезапуститься з втратою даних або вийде з ладу. ЦП пов'язаний з читанням та записом в пам'ять, дисковий простір. Завантаженість ЦП в більшості залежить від написаного SQL-запиту і в подальшому з обробкою даних, вибраних з таблиць. При налаштуванні СУБД необхідно виводити статистику за певний час та в першу чергу дивитися на завантаженість ЦП. Статистика не тільки дозволить вчасно побачити перегрузку та запобігти збою роботи СУБД, але й підвищити швидкодію та знизити витрати на електроенергію.

У подальшому планується спробувати інші існуючі методи самоналаштування СУБД, оцінити їхні переваги та недоліки, щоб вдосконалити власний метод динамічного самоналаштування БД. Також для покращення розробленого методу необхідно буде провести більше експериментів у пропонованій проблемній області.

Перелік використаних джерел

1. Нестеров М.В., Неудакіна Л.В., Скарга-Бандурова І.С. Прогностична модель для налаштування продуктивності систем управління базами даних // Проблеми інформатики і моделювання. Тезиси шістнадцятої міжнародної науково-технічної конференції – Харків: НТУ "ХПІ", 2017. – С. 64.
2. Acedo M.A., Molina M.A., Silva R., Marciano M., Portilla E.A. «Authentication review for nodes in wireless sensor networks» - Revista Electro´nica de Estudios Telema´ticos, 2009 – 9(1):1–23
3. Rubio JJ, Angelov P, Pacheco J (2011) Uniformly stable backpropagation algorithm to train a feedforward neural network. IEEE Trans Neural Netw 22(3):356–366
4. Pe´rez-Cruz JH, Alanis AY, Rubio JJ, Pacheco J (2012) System Identification based on multilayer differential neural networks: a new result. J Appl Math 2012:1–20
5. Rubio JJ (2009) SOFMLS: online self organizing fuzzy modified least square network. IEEE Trans Fuzzy Syst 17(6):1296–1309
6. Iglesias JA, Angelov P, Ledezma A, Sanchis A (2010) An evolving classification of agents behaviors: a general approach. Evol Syst 1(3):161–172
7. Leite D, Ballini R, Costa P, Gomide F (2012) Evolving fuzzy granular modeling from nonstationary fuzzy data streams. Evol Syst 3(2):65–79
8. Satish SK, Saraswatipura MK, Shastry SC (2007) DB2 performance enhancements using Materialized Query Table for LUW Systems, 2007. In: ICONS ’07, second international conference Storm AJ et al (2006) Adaptive self-tuning of memory in DB2. In:VLDB
9. Tran DN, Huynh PC et al (2008) A new approach to dynamic self-tuning of database buffers. In: ACM transactions on storage, vol 4
10. Holze M, Ritter N (2011) System models for goal-driven self-management in autonomic databases. Data Knowl Eng 70:685–701
11. Agarwal S et al (2007) Automated selection of materialized views and indexes. In: VLDB
12. Choudhuri S, Narasayya V (2007) Self tuning database systems: a decade progress. Microsoft Research
13. Koopman P (2004) Elements of the self-healing system problem space. In: IEEE data engineering bulletin
14. Liu P (2005) Design and implementation of self healing database system. In: IEEE conference
15. Nehme RV (2008) Database, heal thyself. In: Data engineering workshop
16. Wiese D, Rabinovitch G (2009) Knowledge management in autonomic database performance tuning. In: Proceedings of 3rd International conference on autonomic and autonomous systems, 2007, p 48

17. Debnath BK, Lilja DJ, Mokbel MF (2008) SARD: a statistical approach for ranking database tuning parameters. In: Data engineering workshop, 2008. ICDEW 2008. IEEE 24th international conference
18. Dageville B, Dias K (2006a) Oracle's self tuning architecture and solutions. In: IEEE data engineering bulletin, vol 29
19. Choudhuri S, Weikum G (2000) Rethinking database system architecture: towards a self tuning RISC style database system. In: VLDB, pp 1–10
20. Cheng SW, Garlan D et al (2006) Architecture based self adaptation in the presence of multiple objectives. In: Proceedings of 2006 international journal of computer systems and engineering
21. Agarwal S, Bruno N, Chaudhari S (2006) AutoAdmin: self tuning database system technology. In: IEEE data engineering bulletin
22. Weikum G, Moenkerngerg A et al (1993) Self-tuning database technology and information services: from wishful thing to viable engineering. In: Parallel and distributed information system
23. Chaudhuri S, Weikum G (2006) Foundations of automated database tuning. In: Data engineering
24. Rabinovitch G, Wiese D (2007) Non-linear optimization of performance functions autonomic database performance tuning. In: IEEE conference
25. Weikum G, Monkenberg A (2002) Self-tuning database technology: from wishful thinking to viable engineering. In: VLDB conference, pp 20–31
26. Chen ANK (2006) Robust optimization for performance tuning of modern database systems. Eur J Oper Res 171:412–429
27. Wang S, Summers RM (2012) Machine learning and radiology. Med Image Anal 16:933–951
28. Hullermeier E (2011) Fuzzy sets in machine learning and data mining. Appl Comput 156:387–406
29. Peng X, Chen B et al (2012) Self-tuning software systems through dynamic quality tradeoff and value-based feedback control loop. J Syst Softw 85:2707–2719
30. Dageville B, Dias K (2006b) Oracle's self tuning architecture and solutions. In: Bulletin of IEEE
31. "PostgreSQL DBMS Documentation", http://www.postgresql.org/.
32. S. Chaudhuri and G. Weikum, "Rethinking Database Architecture: Towards s Self-tuning RISC-style Database System," in Proceedings of VLDB, 2000, pp. 1–10.
33. K. Dias, M. Ramacher, U. Shaft, V. Venkataramamani, and G. Wood, "Automatic Performance Diagnosis and Tuning in Oracle," in Proceedings of CIDR, 2005, pp. 1110–1121.
34. G. Group, "The Total Cost of Ownership: The Impact of System Management Tools," 1996.
35. H. Group, "Achieving Faster Time-to-Benefit and Reduced TCO with Oracle Certified Configurations," March, 2002.
36. R. Plackett and J. Burman, "The Design of Optimum Multifactorial Experiments," in Biometrika Vol. 33 No. 4, 1946, pp. 305–325.

37. D. J. Lilja, Measuring Computer Performance A practioner‘s Guide. Cambridge University Press, 2000.
38. D. Montgomery, Design and Analysis of Experiments. Wiley, 2001.
39. B. Debnath, J. Skarie, D. Lilja, and M. Mokbel, “SARD: A Statistical Approach for Ranking Database Tuning Parameters,” Laboratory for Advanced Research in Computing Technology and Compilers Technical Report, no. ARCTiC 07-11, 2007.
40. Biplob K. Debnath, David J. Lilja, Mohamed F. Mokbel, “SARD: A Statistical Approach for Ranking Database Tuning Parameters”, University of Minnesota, Twin Cities, USA
41. T. Kraft, H. Schwarz, R. Rantzau, and B. Mitschang. CoarseGrained Optimization: Techniques for Rewriting SQL Statement Sequences. In VLDB’2003, pages 488–499, 2003.
42. E. J. O’Neil, P. E. O’Neil, and G. Weikum. The LRU-K Page Replacement Algorithm For Database Disk Buffering. In SIGMOD’1993, pages 297–306, 1993.
43. M. Stonebraker. The case for partial indexes. Sigmod Record, 18(4):4–11, Dec. 1989.
44. Kai-Uwe Sattler, Eike Schallehn, Ingolf Geist, “Towards Indexing Schemes for Self-Tuning DBMS”, TUIlmenau, UniversityofMagdeburg
45. Ivan T. Bowman, David Toman Abdelkader Hameurlain, F. Morvan, “CPU and incremental memory allocation in dynamic parallelization of SQL queries”, Parallel Computing, 2002, pp. 525–556.
46. K. P. Brown, M. Mehta, M. J. Carey, and M.Livny, “Towards Automated Performance Tuning For Complex Workloads”, Proceedings of the 20th Very Large Data Base Conference, Santiago, Chile, 1994.
47. P. Martin, W. Powley,Zheng Min, K.Romanufa, “Experimental study of a self-tuning algorithm for DBMS buffer pools”, Journal of Database Management, Apr.2005, pp. 1-20.
48. Baoning Niu, P. Martin, W. Powley, R. Horman and P. Bird, “Workload Adaptation in Autonomic DBMSs”, Proceedings of CASCON, 2006, pp. 161 - 173.
49. D. A. Menascé, and M. N. Bennani, “On the Use of Performance Models to Design Self-Managing Computer Systems”, Proceedings of 2003 Computer Measurement Group Conference, Dallas, TX. USA, Dec.7-12, 2003, pp. 1-9.
50. D. Botzer, and O. Etzion, “Self-Tuning of the Relationships among Rules' Components in Active Databases Systems” IEEE Transactions On Knowledge And Engineering, Mar.2004, pp. 375-379.
51. Fu Duan, Yongjie Han, Qiuyong Zhao, Keming Xie, “Towards Self-tuning of Dynamic Resources for Workloads”, Taiyuan University of Technology, Taiyuan, Shanxi, P.R.China, 2008
52. S. Rivoire, P. Ranganathan, and C. Kozyrakis. A comparison of high-level full-system power models. In HotPower, 2008.

53. X. Fan, W.-D. Weber, and L. A. Barroso. Power provisioning for a warehouse-sized computer. In ISCA, 2007.
54. G. Graefe. Database servers tailored to improve energy efficiency. In Software Engineering for Tailor-made Data Management, pages 24–28, 2008.
55. S. Harizopoulos, M. A. Shah, J. Meza, and P. Ranganathan. Energy efficiency: The new holy grail of data management systems research. In CIDR, 2009.
56. J. Meza, M. A. Shah, P. Ranganathan, M. Fitzner, and J. Veazey. Tracking the power in an enterprise decision support system. In ISLPED '09, pages 261–266, 2009.
57. W. Lang and J. M. Patel. Towards eco-friendly database management systems. In CIDR, 2009.
58. Z. Xu, Y. Tu, and X. Wang. Exploring power-performance tradeoffs in database systems. In ICDE, 2010.
59. J. Hamilton. Internet-scale data center power efficiency. In CIDR, 2009.
60. D. G. Andersen, J. Franklin, M. Kaminsky, A. Phanishayee, L. Tan, and V. Vasudevan. Fawn: a fast array of wimpy nodes. In SOSP, pages 1–14, 2009.
61. S. Rivoire, M. A. Shah, P. Ranganathan, and C. Kozyrakis. Joulesort: a balanced energy-efficiency benchmark. In SIGMOD '07, pages 365–376, 2007.
62. N. Tolia, Z. Wang, et al. Delivering energy proportionality with non energy-proportional systems – optimizing the ensemble. In HotPower, 2008.
63. R. Raghavendra, P. Ranganathan, et al. No power struggles: A unified multi-level power management architecture for the data center. In ASPLOS, 2008.
64. L. A. Barroso and U. Holzle. The case for ¨ energy-proportional computing. IEEE Computer, 40(12):33–37, 2007.
65. Numonyx. Phase change memory (pcm): A new memory technology to enable new memory usage models. Online, 2009. http://www.numonyx.com/ Documents/WhitePapers/ Numonyx PhaseChangeMemory WhitePaper.pdf.
66. D. Strukov, G. Snider, D. Stewart, and R. S. Williams. The missing memristor found. Nature, 453:80–83, 2008.
67. Dimitris Tsirogiannis, Stavros Harizopoulos, Mehul A. Shah, "Analyzing the Energy Efficiency of a Database Server", University of Toronto, HP Labs
68. Foundations of Automated Database Tuning, VLDB '06, September 12–15, 2006, Seoul, Korea.Copyright 2006 VLDB Endowment, ACM
69. Rethinking Database System Architecture: Towards a Selftuning RISC-style Database System in Cairo, Egypt, 2000
70. AutoAdmin:Self-Tuning Database Systems Technology, Copyright 2006 IEEE
71. Self-Tuning Database Systems: A Decade of Progress, Copyright

72. I. Alagiannis, Towards Adaptive, Flexible, and Self-tuned Database Systems, (DIAS, I&C, EPFL) in EDIC-ru/05.05.2009
73. SQL Memory Management in Oracle9i, Hong Kong, China, 2002
74. D. Burleson, Oracle Tuning, The Definitive Reference Second Edition.
75. Hitesh KUMAR SHARMA, Aditya SHASTRI, Ranjit BISWAS, Architecture of Automated Database Tuning Using SGA Parameters, 2012
76. ДСанПіН 3.3.2.007-98 Гігієнічні вимоги до організації роботи з візуальними дисплейними терміналами електронно-обчислювальних машин
77. ДСН 3.3.6.042-99 Санітарні норми мікроклімату виробничих приміщень
78. ДСН 3.3.6.037-99 Санітарні норми виробничого шуму, ультразвуку та інфразвуку
79. ДСН 3.3.6.039-99 Санітарні норми виробничої загальної та локальної вібрації
80. ДСТУ ГОСТ 12.1.012-90 ССБТ. Вибрационная безопасность. Общие требования
81. ГОСТ 12.1.006-84 ССБТ. Электромагнитные поля радиочастот. Общие требования безопасности. Допустимые уровни на рабочих местах и требования к проведению контроля.
82. ГОСТ 12.1.030-81 ССБТ. Электробезопасность.Защитное заземление. Зануление
83. ГОСТ 13109-97 „Электрическая энергия. Совместимость технических средств электромагнитных. Нормы качества электроэнергоснабжения общего назначения”
84. ДБН В.2.5-28:2015 Природнє і штучне освітлення
85. ДСанПіН 3.3.2.007-98 Гігієнічні вимоги до організації роботи з візуальними дисплейними терміналами електронно-обчислювальних машин
86. НПАОП 40.1-1.21-98 Правила безпечної експлуатації електроустановок споживачів
87. ДБН В.2.5-67:2013 Опалення, вентиляція та кондиціонування
88. ГОСТ 12.1.005-88 ССБТ. Загальні санітарно-гігієнічні вимоги до воздуху робочої зони
89. ГОСТ 12.1.044-89 ССБТ. Пожаровзрывоопасность веществ и материалов. Номенклатура показателей и методы их определения.
90. НПАОП 0.00-1.28-10 Правила охорони праці під час експлуатації електронно-обчислювальних машин
91. НПАОП 0.00-4.15-98 Про розробку інструкцій з охорони праці
92. Закон України Про охорону навколишнього природного середовища
93. ДСТУ ISO 1A001:2006 (ISO 1A001:200A) Системы экологического управления. Требования и руководящие указания по применению
94. Кодекс Кодекс законів про працю України

ДОДАТОК А

Електронні плакати

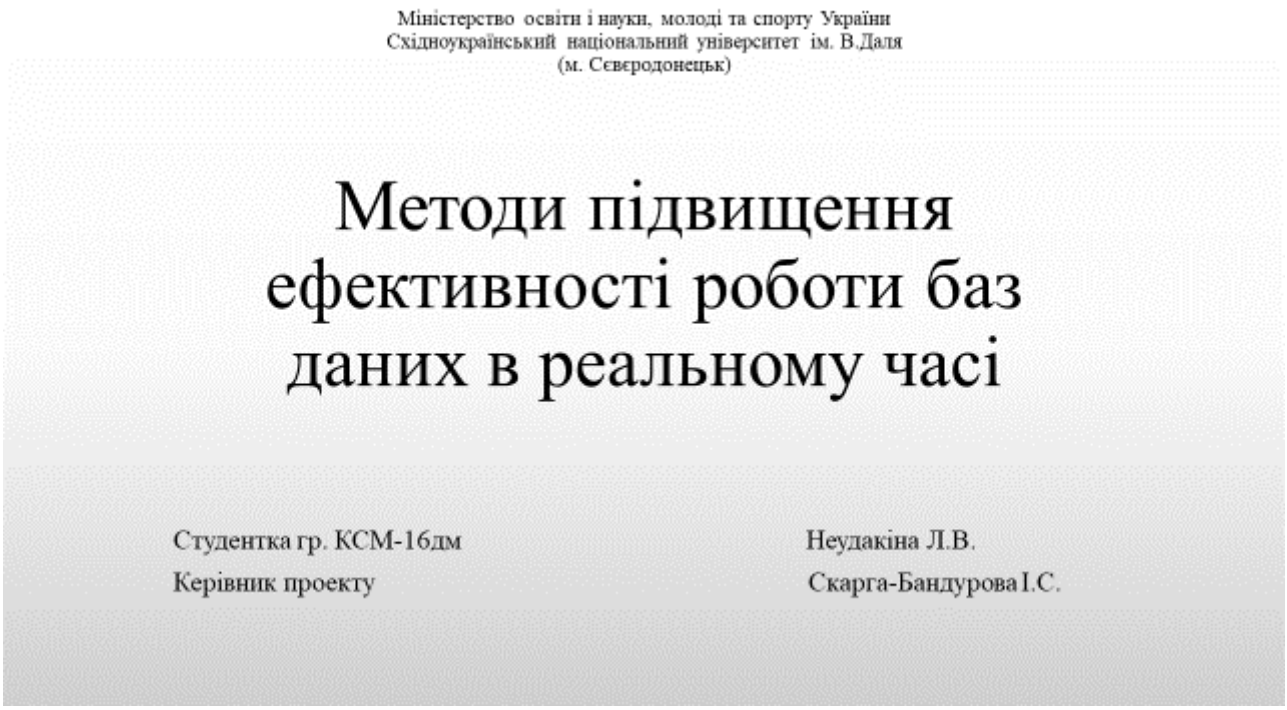


Рисунок А.1 – Титульна сторінка

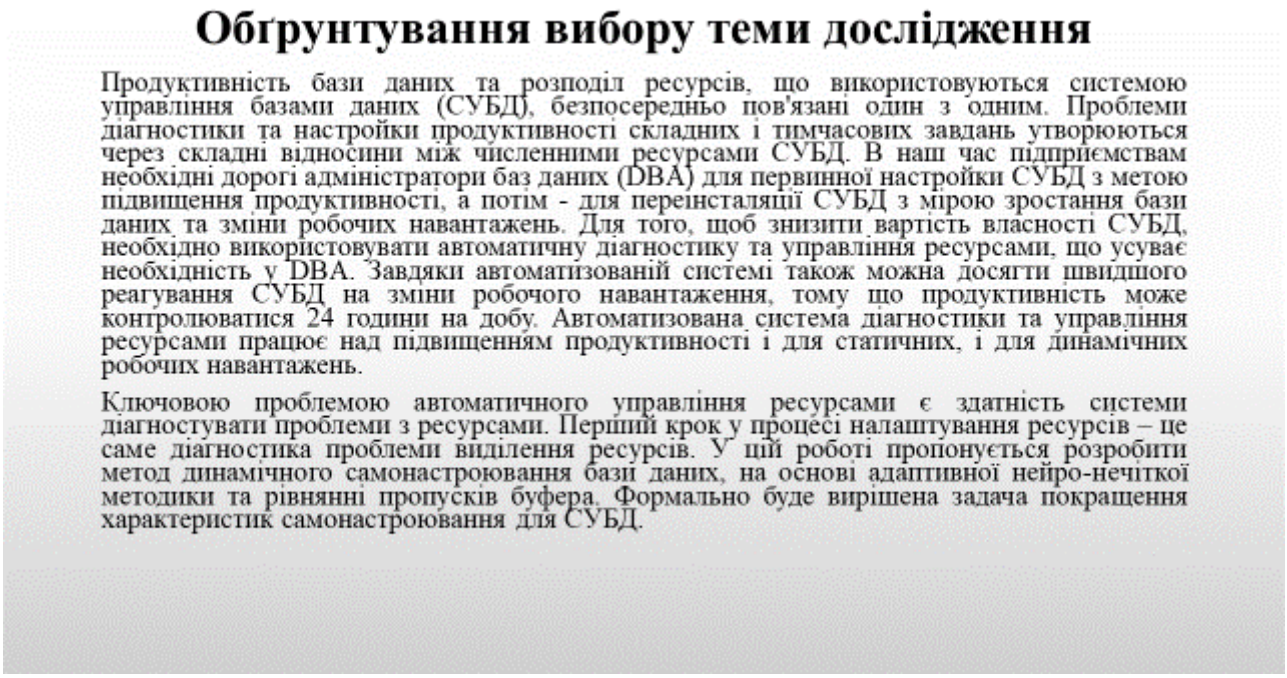


Рисунок А.2 – Обгрунтування вибору теми дослідження

Мета і завдання дослідження

Метою дослідження є вирішення задачі покращення характеристик самонастроювання для систем управління базами даних.

Для досягнення мети дослідження необхідно вирішити такі завдання:

- дослідження існуючих методів і підходів до самонастроювання в системах управління базами даних;

- дослідження впливу різних стратегій настройки продуктивності бази даних і енергоефективності;

- розробка методу динамічного самонастроювання бази даних на основі рівняння пропусків буфера, отриманого з аналітичної моделі, і адаптивної нейро-нечіткої методики обліку змін в розмірі бази даних при тривалому використанні програми бази даних.

Рисунок А.3 – Мета і завдання дослідження

Методи дослідження. Проведені в роботі дослідження основані на адаптивній нейро-нечіткій методиці та рівнянні пропусків буфера, які використовувались при розробленні методу динамічного самонастроювання бази даних.

Практичне значення отриманих результатів полягає в тому, що основні наукові положення дисертації реалізовані у виді розрахункових моделей та адаптивної нейро-нечіткої методики, які утворюють метод динамічного самонастроювання бази даних.

Розроблено метод динамічного самонастроювання бази даних на основі рівняння пропусків буфера, отриманого з аналітичної моделі, і адаптивної нейро-нечіткої методики обліку змін в розмірі бази даних при тривалому використанні програми бази даних.

Завдяки використанню цього методу можна вирішити задачу підвищення ефективності роботи баз даних в реальному часі.

Рисунок А.4 – Методи дослідження та практичне значення отриманих результатів

Рисунок А.5 – Архітектура самоналаштування продуктивності

Налаштування продуктивності

Калібрування системи для необхідного часу відгуку називається налаштуванням продуктивності. Блок-схема показує функціональні блоки, які складають всю систему управління, зовнішню по відношенню до СУБД. Як видно з архітектури, три важливих вхідних параметра, а саме: кількість користувачів (N), BHR і середній розмір бази даних (DBS) подаються як на нейронні, так і на нечіткі системи для оцінки відповідного розміру параметрів налаштування. Нейронна мережа навчається відповідному набору навчальних даних, отриманого з експериментальних спостережень. Хоча є кілька параметрів, що настроюються, тільки три параметри вибираються для налаштування, оскільки вони мають значний вплив на час відгуку і, що найважливіше, динамічно змінюються без необхідності повертатися назад в СУБД.

Нейронні мережі найкраще підходять для обробки складних систем, які в дійсності нелінійні за своєю природою. Нейронні мережі відносяться до більш широкого предмету, що називається комп'ютерним навчанням, яке займається комп'ютерними рішеннями, які вивчають складний взаємозв'язок між входом і виходом з набору даних емпіричного навчання. Застосування техніки машинного навчання досить широко поширене, в тому числі: аналіз медичних зображень для точного прогнозування помилок, пошук близьких оптимізаційних рішень до завдань NP-Hard, особливо в мережевій області і т. д. Машинне навчання забезпечує ефективний метод прогнозування оціночних значень параметрів настроювання з певного набору навчальних матеріалів.

Метою цієї системи є аналіз СУБД шляхом активного моніторингу показників ефективності, таких як: відношення пропусків буфера, кількість активних процесів і розмір таблиці, які демонструють ознаки швидкого зростання і ініціюють контрольну міру з використанням нечіткого контролю.

Рисунок А.6 – Налаштування продуктивності

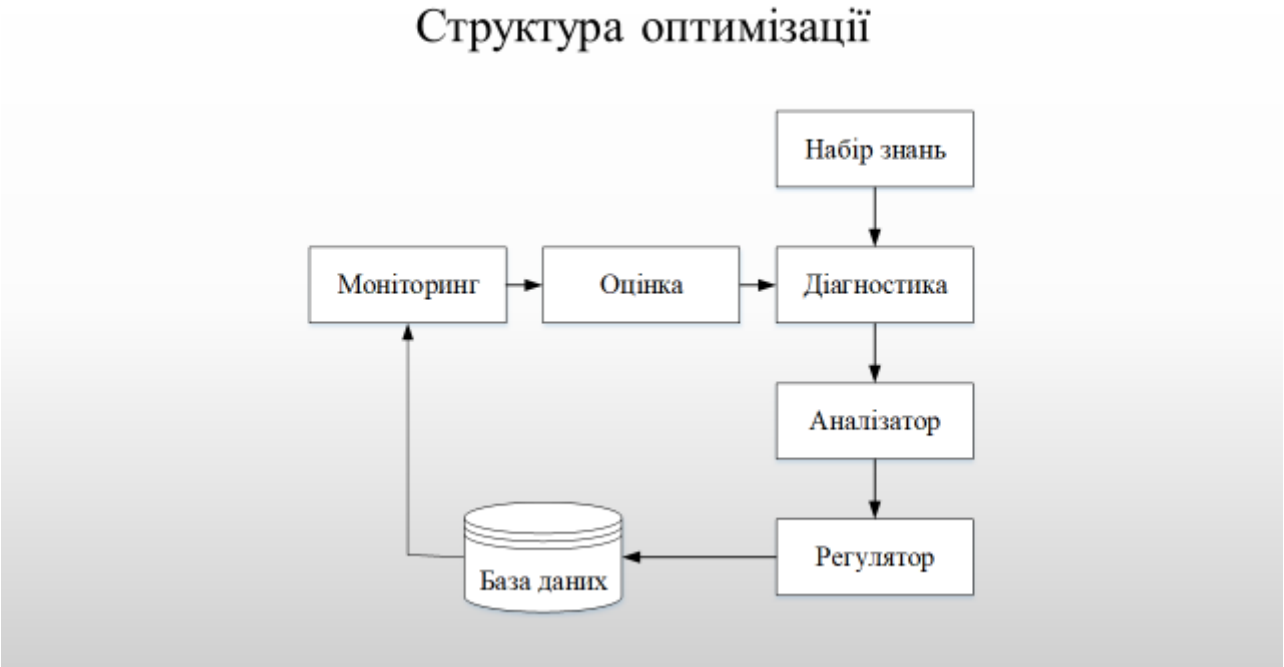


Рисунок А.7 – Структура оптимізації

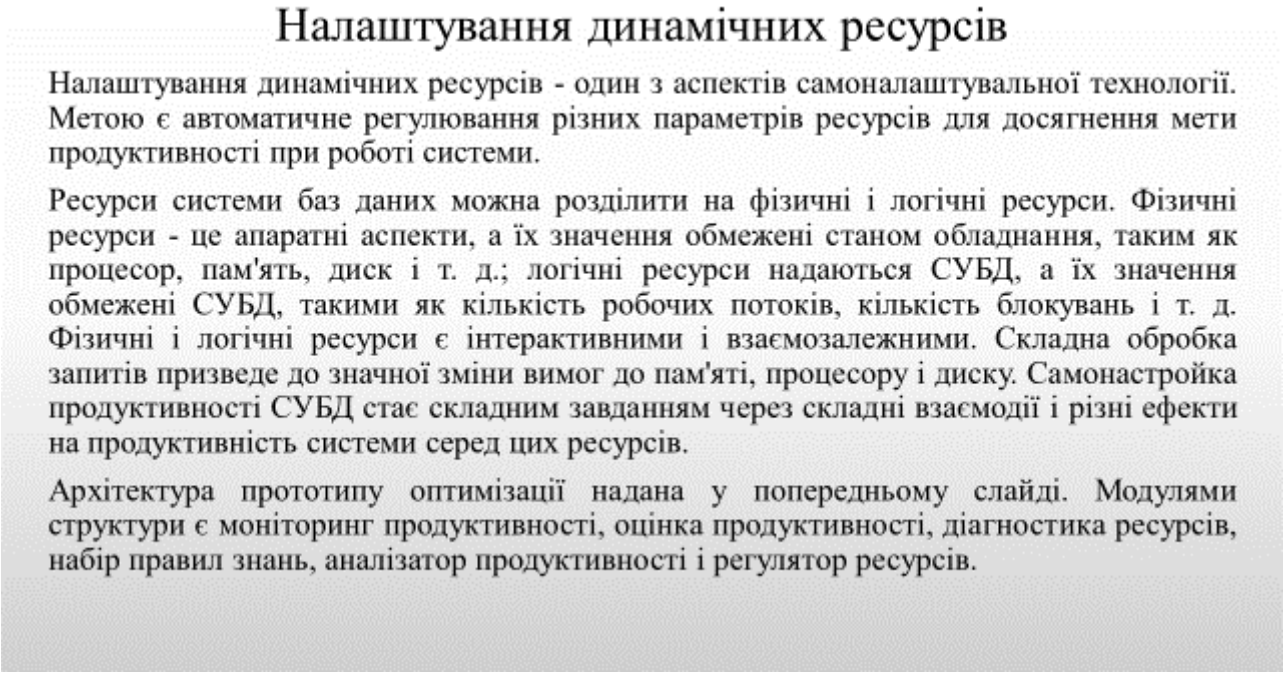


Рисунок А.8 – Налаштування динамічних ресурсів

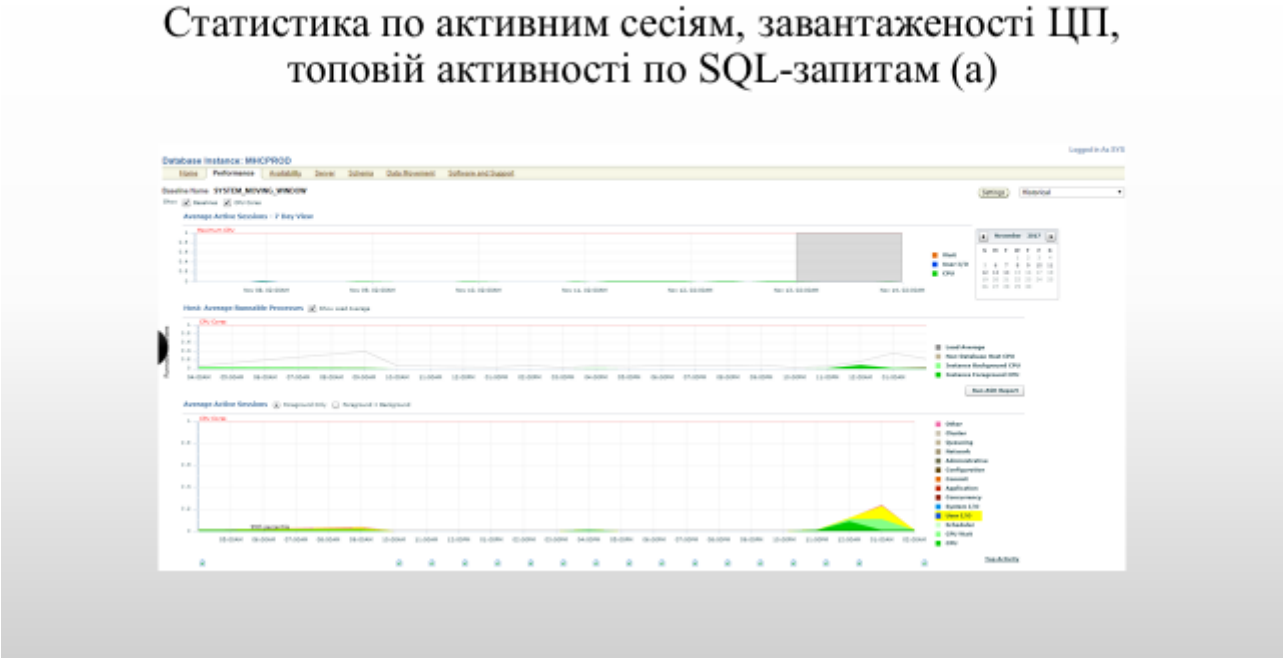


Рисунок А.9 – Статистика по активним сесіям, завантаженості ЦП, топовій активності по SQL-запитам (а)

Рисунок А.10 – Статистика по активним сесіям, завантаженості ЦП, топовій активності по SQL-запитам (б)

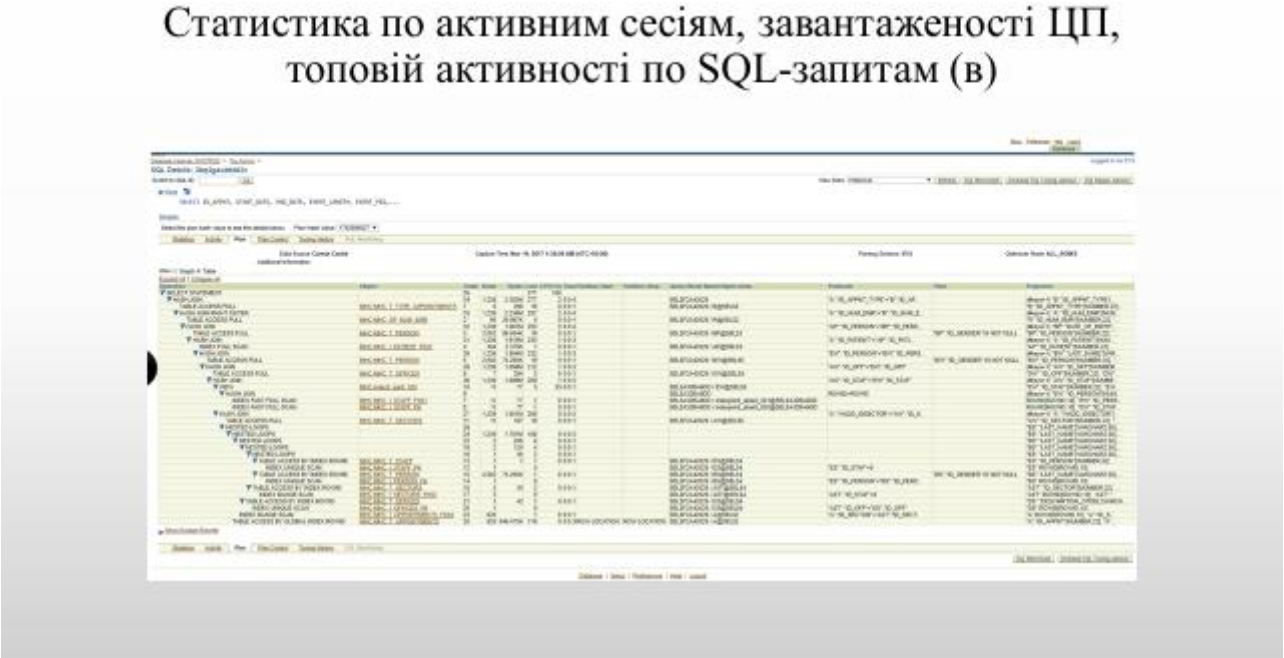


Рисунок А.11 – Статистика по активним сесіям, завантаженості ЦП, топовій активності по SQL-запитам (в)

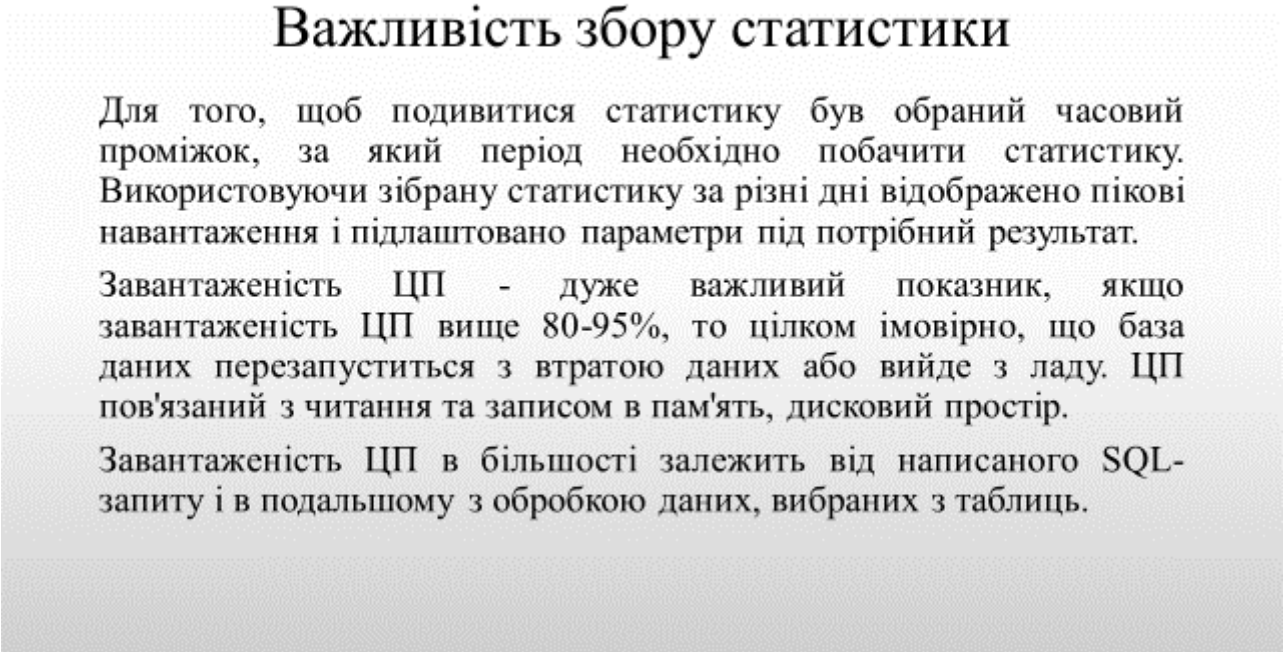


Рисунок А.12 – Важливість збору статистики

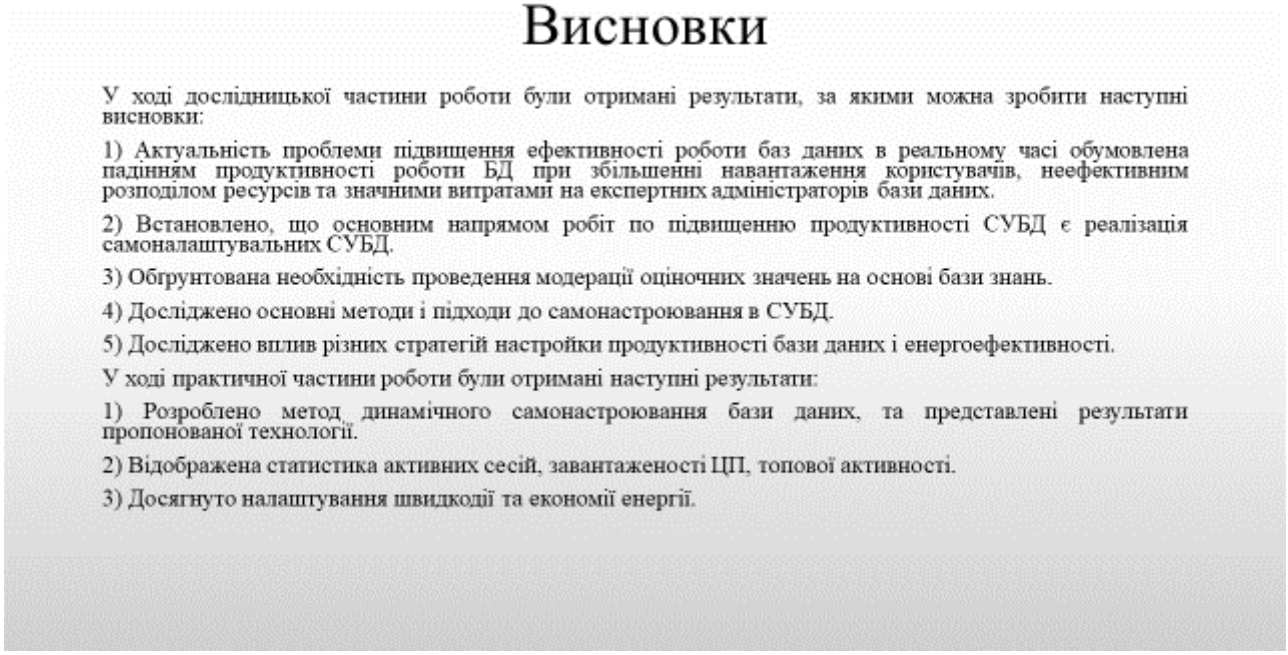


Рисунок А.13 - Висновки