Оптимізація запитів в системах баз даних

Р. Гринвальд, Р. Стаковьяк, Дж. Стерн. Oracle 11g. Основы, 4_е издание. – Пер. с англ. – СПб.: Символ_Плюс, 2009. – 464 с., ил.
Миллсап К., Хольт Д. Oracle. Оптимизация производительности. – Пер. с англ. – СПб: Символ-Плюс, 2006. – 464 с., ил.

Приклади стосуються СУБД Oracle, але така ж чи аналогічна оптимізація може провадитись в інших СУБД.

Оптимізація запитів спрямована на мінімізацію часу відгуку.

Обробка запитів процесором запитів

Підсистема, яка визначає продуктивність СУБД – *процесор запитів* (query processor). На рис.8.1 (Компоненти СУБД) лекції по паралелізму він представлений двома компонентами:

1. *Компілятор запитів* (query compiler) транслює запит у внутрішній формат СУБД – *план запиту* (query plan), тобто послідовність інструкцій, що підлягає виконанню. Компілятор запитів складається з трьох частин:
   - *Синтаксичного аналізатора запитів* (query parser), що на основі тексту запиту створює деревовидну структуру даних;
   - *Препроцесора запитів* (query preprocessor), який виконує семантичний аналіз запиту (перевірку існування відношень та атрибутів у запиті), та перетворення дерева, побудованого аналізатором, у дерево алгебраїчних операторів (у сенсі реляційної алгебри), які відповідають плану запиту;
   - *Оптимізатора запитів* (query optimizer), який виконує перетворення плану запиту в найбільш ефективну послідовність операцій над даними.

   Оптимізатор запитів для прийняття рішень користується метаданими та статистичною інформацією, що накопичена в СУБД. Наприклад, на його рішення суттєво впливає наявність індексу.

2. *Виконуюча машина* (execution engine) відповідає за виконання кожної операції плану запиту. Під час роботи вона спілкується з більшістю інших компонентів СУБД – напряму або через буфери. Для обробки даних виконуюча машина читає дані в буфери. При цьому вона „спілкується” з планувальником завдань, уникаючи звернення до блокованої інформації, а також із менеджером протоколювання, для фіксації змін у протоколі.

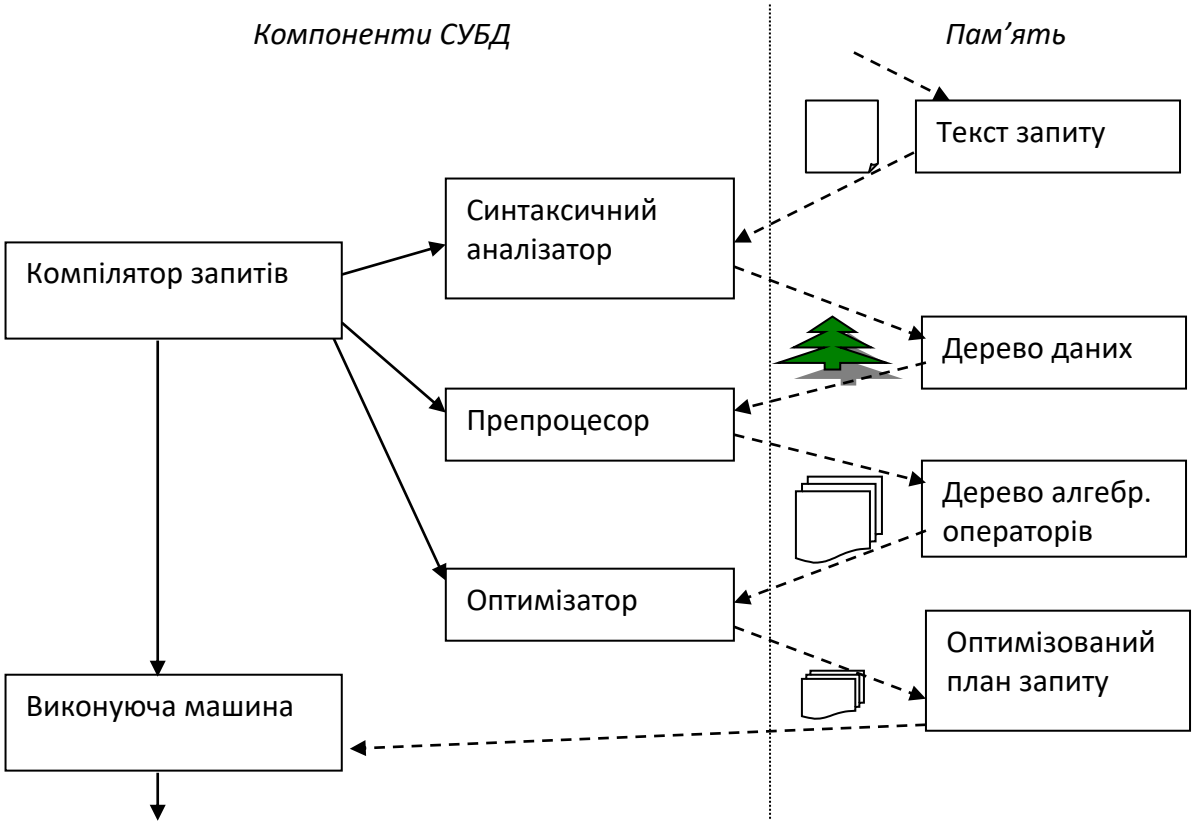

Рисунок 1 - Обробка запитів

Виконуюча машина направляє групу запитів на отримання невеликої порції даних (як правило, рядків (кортежів) таблиці (відношення) *менеджеру ресурсів*, який обізнаний в особливостях розміщення інформації в файлах даних, що містять таблиці, у форматах і розмірах записів у таблицях і у структурах індексів (indexes), які забезпечують суттєве прискорення пошуку даних.

Фактори загальної вартості запиту

Для порівняння планів виконання запиту і обрання найкращого вводиться умовна вартість запиту, яка має наступні складові:

- *Вартість комунікацій (V)*
  $V = V^k + V^з$

Де V^k – вартість комунікаційного каналу (в мережі), яка зазвичай зв'язана з часом, протягом якого канал відкритий,
$V^з$ - вартість затримок в обробці, що викликаються передачею даних.

- *Вартість доступу до вторинної пам'яті (дисків)*
  - розмір проміжних результатів,
  - кластеризація даних на фізичних сторінках,
  - розмір доступного буферного простору ,
  - швидкість пристроїв, що використовуються.
- *Вартість зберігання*
  - вартість заняття вторинної пам'яті,
  - вартість буферів основної пам'яті.
- *Вартість обчислень* - вартість (час) використання ЦП

Виконання запитів в Oracle

1. **Лексичний і синтаксичний аналіз**. Створюється його внутрішній розріз, що відображає структуру запиту і містить інформацію, яка характеризує об'єкти бази даних, згадані в запиті .
2. **Логічна оптимізація.** Можуть застосовуватися різні перетворення, "поліпшуючі" початкове подання запиту. Серед перетворень можуть бути тільки еквівалентні, після проведення яких виходить внутрішнє подання, семантично еквівалентне початковому (наприклад, приведення запиту до деякої канонічної форми).
3. **Оптимізація.** На основі інформації, яку має в своєму розпорядженні оптимізатор, та набору альтернативних процедурних планів виконання даного запиту відповідно до його внутрішнього подання для кожного плану оцінюється передбачувана вартість виконання запиту. При оцінках використовується статистична інформація про стан бази даних[1], доступна оптимізатору. З отриманих альтернативних планів вибирається найдешевший.
4. **Формування подання для виконання**. За внутрішнім поданням оптимального плану формується виконуване подання плану.
5. **Реальне виконання**.

Оптимізатор. Його призначення. Етапи роботи.

Функція оптимізатора
- вибрати оптимальний (виходячи з набору критеріїв) план виконання запиту.

Завдання оптимізатора:

- Обчислення виразів і операцій
- Перетворення SQL операторів
- Вибір способу оптимізації - за вартістю або по правилах
- Вибір шляхів доступу
- Вибір порядку з'єднань таблиць
- Вибір методу з'єднань таблиць
- Визначення найбільш ефективного плану виконання

Оптимізація по правилах (RULE BASED).

Враховуються тільки способи доступу до даних, із зафіксованими пріоритетами (ваговими коефіцієнтами) по ефективності доступу. Даний підхід використовувався в ранніх версіях ORACLE і містить істотний недолік - він не враховує реального розподілу даних.

Оптимізація за вартістю (COST BASED).

Крім ефективності різних шляхів доступу до даних враховується статистика розподілу даних і ресурсів операційної системи. Наприклад, наскільки збалансоване B-дерево має рівномірний розподіл записів по діапазонах ключів? Скільки можна отримати пам’яті для створення буферів?

[1] Інформація щодо заповненості фізичних записів таблиць і рівномірності розподілу записів по діапазонах ключів у збалансованому дереві.

Схема виконання запиту

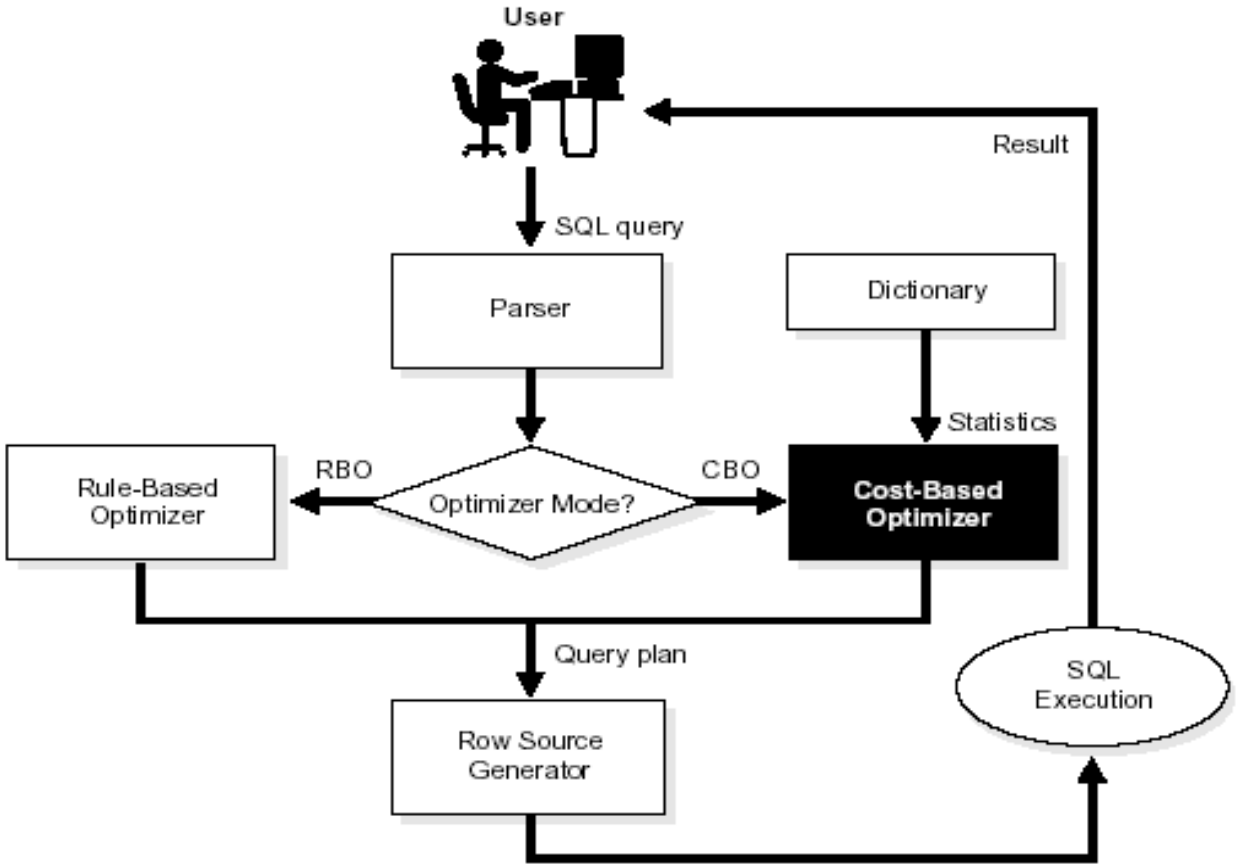

Рисунок 2 – Алгоритм виконання запиту в Oracle

Закон Амдала

У 1967 році Джин Амдал (Gene Amdahl) оприлюднив спостереження, яке пізніше було назване законом Амдала:

> Збільшення продуктивності, що досягається деякими удосконаленням, обмежене часткою загального часу виконання, яка споживається вдосконаленим компонентом.

Іншими словами, в першу чергу слід прагнути до зменшення найбільш значимої складової часу відгуку в найбільш критичних для бізнесу операціях.

Приклад 8.11. Складові часу відгуку в порядку убування долі в загальному часі відгуку для деякого застосунку:

Response Time Component	Duration		#Calls	Dur/Call
SQL*Net message from client	166.6s	91.7%	6,094	0.027338s
CPU service	9.7s	5.3%	18,750	0.000515s
unaccounted-for	2.2s	1.2%		
db file sequential read	1.6s	0.9%	1,740	0.000914s
log file sync	1.1s	0.6%	681	0.001645s
SQL*Net more data from client	0.3s	0.1%	71	0.003521s
SQL*Net more data to client	0.1s	0.1%	108	0.001019s
free buffer waits	0.1s	0.0%	4	0.022500s
SQL*Net message to client	0.0s	0.0%	6,094	0.000007s
db file scattered read	0.0s	0.0%	34	0.001176s
log file switch completion	0.0s	0.0%	1	0.030000s
log buffer space	0.0s	0.0%	2	0.005000s
latch free	0.0s	0.0%	1	0.010000s
direct path read	0.0s	0.0%	5	0.000000s
direct path write	0.0s	0.0%	2	0.000000s
Total	181.8s	100.0%		

Рисунок 3 – Складові часу відгуку в порядку убування долі в загальному часі відгуку для деякого застосунку

Закон Амдала пояснює, чому складові часу відгуку слід розглядати в порядку убування. Саме тому в прикладі 8.11 не слід займатися «проблемою» `CPU service`, не розібравшись попередньо з проблемою `SQL*Net message from client` . Якщо споживання процесорного часу зменшиться на 50%, то час відгуку покращиться всього на 2%. Але якщо вдасться зменшити на ті ж 50% складову `SQL * Net message from client`, то загальний час відгуку зменшиться на 46%.

Для ідентифікації джерел затримки особливо корисні наступні подання, з яких слід почати аналіз проблеми:

V$SYSTEM_EVENT	Надає агреговану загальносистемну інформацію щодо ресурсів, яких чекає екземпляр БД
V$SESSION_EVENT	Надає накопичувальний список подій, які доводилося чекати в кожному сеансі.
V$SESSION_WAIT	Надає детальну посеансову інформацію щодо ресурсів, які сеанс очікує в даний момент або чекав в останній раз.
V$SESSION	Надає інформацію про кожний сеанс, в тому числі про подію, яку він чекає в даний момент або чекав в останній раз.
V$SQL	Надає інформацію стосовно виконаних у сеансі запитів SQL, в т.ч. час процесора і сумарний час
V$WAITSTAT	Надає час чекання під час операцій вводу-виводу

Наприклад, для виконання четвертого подання слід у вікні SQL Commands Oracle Database Express Edition виконати
SELECT * FROM V$SESSION

У Oracle9i є близько 300 подій, що вимагають для аналізу докладних даних з декількох десятків подань V$. Навіть якби вдалося отримати з них дані для строго визначених моментів часу, там будуть агреговані значення, яким бракує деталізації.

Ключ до визначення складових часу відгуку дає аналіз розширеного трасування SQL.

Ознаки «поганого» коду запиту

1. Код може бути «погано оформлений», в цьому випадку його складно розуміти, відповідно – програмісту складно його змінювати.

```
SELECT Suppliers.CompanyName, Products.ProductName,
Products.QuantityPerUnit, Products.UnitPrice, OrderDetails.UnitPrice,
OrderDetails.Quantity, OrderDetails.Discount, Orders.OrderDate,
Orders.ShipName FROM Suppliers INNER JOIN Products ON
Suppliers.SupplierID = Products.SupplierID INNER JOIN OrderDetails ON
Products.ProductID = OrderDetails.ProductID INNER JOIN Orders ON
OrderDetails.OrderID = Orders.OrderID WHERE (OrderDetails.Discount =
0) AND (Orders.OrderDate > '06/11/2009')
```

Той же запит після переписування (рефакторингу) може виглядати так:

```
SELECT Suppliers.CompanyName, Products.ProductName,
Products.QuantityPerUnit, Products.UnitPrice,
OrderDetails.UnitPrice, OrderDetails.Quantity, OrderDetails.Discount,
Orders.OrderDate, Orders.ShipName
FROM ((  Suppliers INNER JOIN Products ON Suppliers.SupplierID =
Products.SupplierID)
INNER JOIN OrderDetails ON Products.ProductID =
OrderDetails.ProductID)
INNER JOIN Orders ON OrderDetails.OrderID = Orders.OrderID
WHERE       (OrderDetails.Discount = 0) AND (Orders.OrderDate >
'06/11/2009')
```

2. Наявність коду, що дублюється
3. Неможливість візуального подання запиту.

Варто перевірити, чи візуалізується запит в SQL-оболонці, як-от Query Designer.

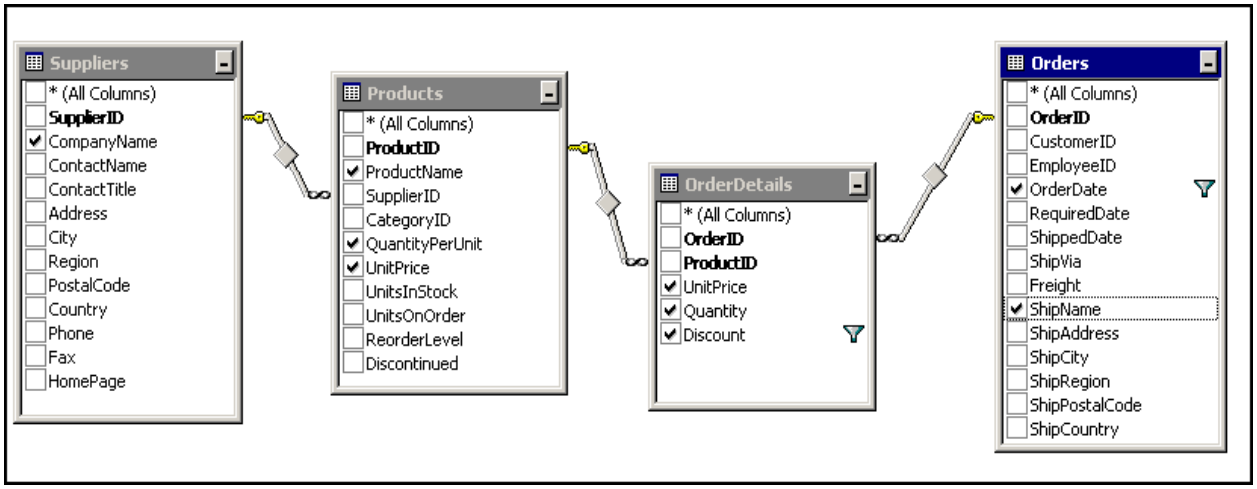

Рисунок 4 - Візуалізація запиту

4. Наявність в запиті «гальмівних» конструкцій:
    - Використання зв'язаних підзапитів;
    - Використання операторів UNION, що примушує потім відкидати дублікати;
    - Використання ключового слова Distinct також передбачає відкидання дублікатів.

Методи модифікації запитів

1. **Виділення користувацької функції**

Це доцільно в разі частого використання деяких стандартних обчислень, що зустрічаються в рамках одного запиту або групи запитів.

2. Утворення розрізів

Замість

```
SELECT Suppliers.CompanyName, Products.ProductName,
Products.QuantityPerUnit,     Products.UnitPrice,
OrderDetails.UnitPrice, OrderDetails.Quantity,
OrderDetails.Discount, Orders.OrderDate, Orders.ShipNameFROM
Suppliers      INNER JOIN Products ON Suppliers.SupplierID =
Products.SupplierID      INNER JOIN OrderDetails ON
Products.ProductID = OrderDetails.ProductID INNER JOIN Orders ON
OrderDetails.OrderID = Orders.OrderID WHERE
(OrderDetails.Discount = 0) AND (Orders.OrderDate >
'06/11/2009')
```

Застосувати

CREATE VIEW ……

Значення дати для фільтру можна передати як параметр розрізу.

Збережені подання компілюються при збереженні і тому виконуються швидше.

3. Позбавлення від підзапитів

```
SELECT Products.ProductName, Products.QuantityPerUnit,
Products.UnitPrice,
(SELECT CompanyName
FROM Suppliers
WHERE Suppliers.SupplierID = Products.SupplierID) AS CompanyName
FROM Products
```

Цей запит можна переписати без використання підзапиту:

```
SELECT Products.ProductName, Products.QuantityPerUnit,
Products.UnitPrice, Suppliers.CompanyName
FROM Products
     LEFT OUTER JOIN Suppliers ON Suppliers.SupplierID =
Products.SupplierID
```

В даному разі JOIN виконує ту ж функцію, що і підзапит.

Зовнішні з’єднання зазвичай приводять до більших витрат, ніж внутрішні, з-з додаткового пошуку неспівпадаючих рядків.

Якщо в запиті лише внутрішні з’єднання, вплив на час виконання умов у реченнях ON та WHERE однаковий. В разі використання зовнішнього з'єднання запити будуть суттєво відрізнятись по часу виконання.

4. Позбавлення від UNION

```
SELECT ProductID, ProductName, UnitPrice   FROM Products   WHERE
ProductName LIKE 'A%‘
UNION
SELECT ProductID, ProductName, UnitPrice FROM Products WHERE
UnitPrice <= 40
```

Модифікований запит (заміна UNION на OR):

```
SELECT ProductID, ProductName, UnitPrice      FROM Products
WHERE ProductName LIKE 'A%' OR UnitPrice <= 40
```

У виразі до модифікації очевидна помилка програміста, на практиці UNION потрібен для злиття записів з різних таблиць.

5. UNION -> UNION ALL

Якщо дублювання неможливі виходячи зі змісту даних, треба використовувати UNION ALL замість UNION. При виконанні UNION ALL не відбувається об’єднання записів, що повторюються.

6. Позбавлення від курсорів

З точки зору продуктивності існують дві проблеми використання курсорів:

- Кожне виконання інструкції FETCH в циклі курсору подібно до виконання інструкції SELECT, що повертає один рядок.
- Інструкція DML на обробку множини записів (як-от UPDATE A SET B WHERE C) оптимізується в цілому, а цикл курсору не може бути так оптимізований. Кожний елемент циклу буде оптимізований і виконаний окремо для кожної ітерації циклу.

Інструкція DML:

```
UPDATE A SET d=e WHERE C
```

Оптимізація

Цикл курсору:

```
WHILE C
BEGIN
     FETCH D INTO E
     UPDATE таблиця_або_подання
     SET список_для_модифікації
     WHERE CURRENT OF курсор_або_змінна
END
```

Оптимізація

Оптимізація

Статистика по таблицях

Дані за статистикою таблиць можна подивитися в поданні User_tables.
(Я побачив у поданні 139 таблиць, жодної створеної користувачем - ВП)

Статистика по індексах

Статистику по індексах можна подивитися в словнику User_indexes.

Шляхи доступу до даних у таблицях

1. **Повне сканування таблиці**

- Для вибірки даних проводиться перебір всіх рядків таблиці, повертаються рядки, що відповідають умовам запиту.
- Може бути ефективніше за використання індексу у випадках, коли необхідно вибрати >25-30% записів в таблиці.

2. **Вибірка за допомогою індексів**

Основними типами індексів є:

- B-tree індекси;
- Двійкові індекси;
- Кластерні індекси.

Найчастіше використовується *B-tree індекс (Balanced Tree).* Рекомендується створювати, якщо:

- Стовпець має широкий діапазон значень
- Стовпець містить велику кількість невизначених значень
- Таблиця великого розміру, і передбачається, що велика частина запитів витягуватиме менше 15% рядків
- Стовпець або стовпці часто використовуються разом в реченні WHERE або умові з'єднання

Оптимізуючі стратегії індексування

SQL Server підтримує два основних типи індексів: кластеризовані і некластеризовані. Обидва реалізуються в вигляді збалансованого дерева.

Кластеризований індекс містить усю таблицю, відсортовану по ключу, тобто його нижній рівень містить реальні рядки таблиці з усіма полями. Якщо таблиця не містить кластеризованого індексу, вона є купою – невідсортованою таблицею.

СУБД автоматично створює індекс для первинного ключа таблиці і для колонок, що входять в обмеження унікальності.

Для яких колонок доцільно створювати індекси?

- Як правило, треба вивести малу кількість рядків таблиці (не більше 10 або 15%), яка містить багато рядків.
- Доцільно індексувати колонку, яка містить широкий діапазон значень, зокрема унікальний номер. Недоцільно – якщо колонка містить обмежену кількість різних значень (як-от «M» і «F» для позначення статі).

У MsSQL Server Інформацію щодо використання індексів можна побачити в системному поданні `sys.dm_db_index_usage_stats` . Його найважливіші колонки:

Database_id Object_id Index_id	Ідентифікатори БД, таблиці або подання, індексу
User_seeks	Кількість операцій пошуку по запитах користувачів
User_scans	Кількість операцій проглядання по запитах користувачів
User_lookups	Кількість уточнюючих запитів по запитах користувачів
User_updates	Кількість операцій оновлення по запитах користувачів

Трасування

Іншим допоміжним засобом оптимізації є **трасування.**

Відтрасувати можна довільну сесію (конекцію) в базі.
Під час трасування у Log-файл операційної системи записуються Sql-вирази, що виконуються, а також деяка супутня інформація - плани запитів, періоди очікування подій, які виконуються в сесії. Траса містить також дуже важливі події, як-от попадання у кеш, періоди очікування з-за блокування, очікування мережі та ін.

Рисунок 5 – Аналіз Log-файлу результатів трасування

Повинен бути систематичний підхід як до пошуку джерела проблеми, так і до реалізації її рішення. Цей підхід передбачає збирання опорних даних щодо використання ресурсів і часу відгуку на внесення змін. Самі зміни слід вносити невеликими порціями, кожен раз спостерігаючи за продуктивністю. Дуже спокусливо виглядає вирішити проблему одним махом, без всяких вимірів, але така тактика зазвичай лише призводить до нових проблем.

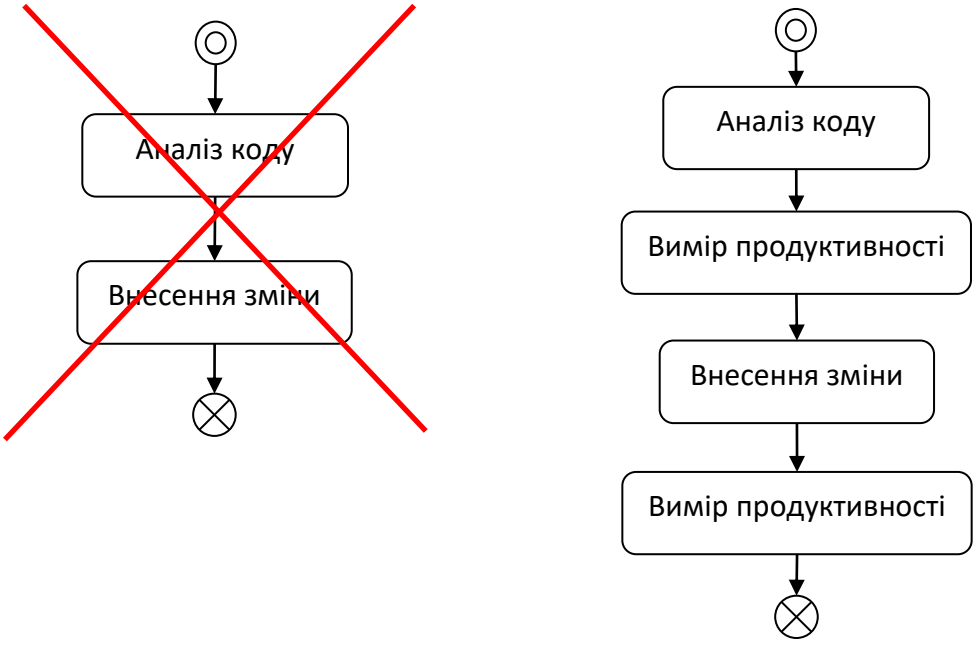

Рисунок 6 - систематичний підхід до пошуку проблеми і до її рішення

Приклад 1. Аналіз плану виконання і визначення вартості виконання запиту у Microsoft SQL Server Management Studio

Для фіксації плану і вартості запиту треба увімкнути режим фіксації. Відчиняємо вікно New Query і в меню Query вмикаємо Include Actual Execution Plan.

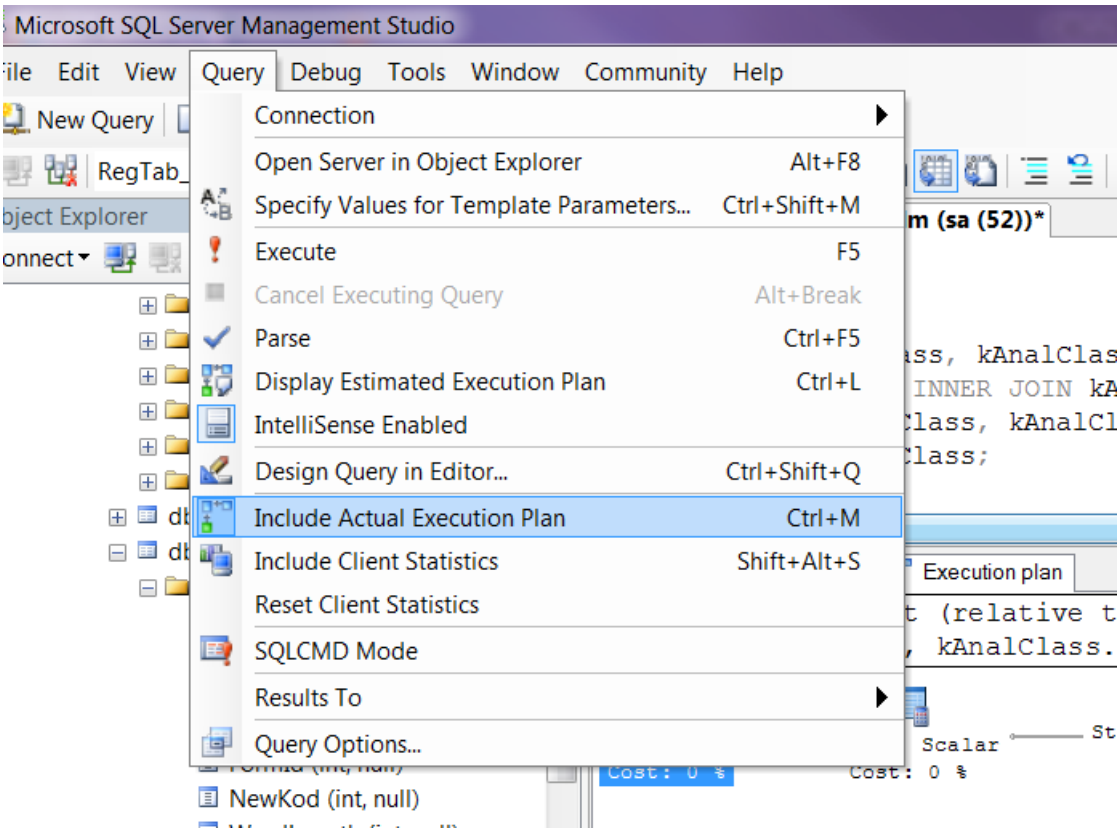

Рисунок 7 - Увімкнення фіксації результатів аналізу запиту в SQL Server Management Studio

Після виконання запиту аналізуємо його план у вкладці Execution Plan.

Виконувався запит на з’єднання двох таблиць, групування ~4000 записів у 19 груп і підрахунок кількості записів у групах.

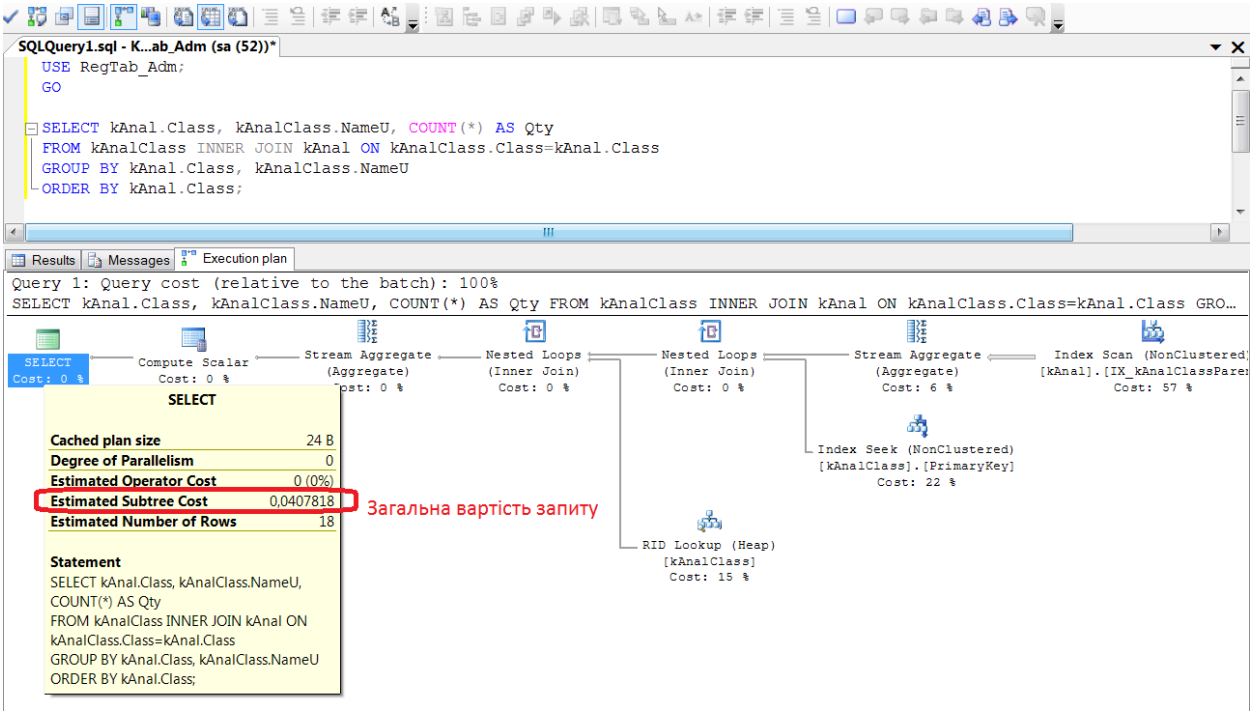

Рисунок 8 - Подання плану і вартості запиту в SQL Server Management Studio

Бачимо, що агрегація тягне 57+6=63% вартості запиту, з'єднання – 15+22=37%.

Приклад 2. Технологія підготовки звіту в системі класу ERP у середовищі Microsoft Access XP, Microsoft SQL Server 2000

Для видачі звіту потрібні :

- Візуальна форма звіту, яку забезпечує генератор звітів Ms Access.
- Recordset (набір даних) – результат виконання запиту до бази.

Перший і найпростіший шлях видачі звіту – на основі приєднаних до середовища MsAccess серверних таблиць БД:

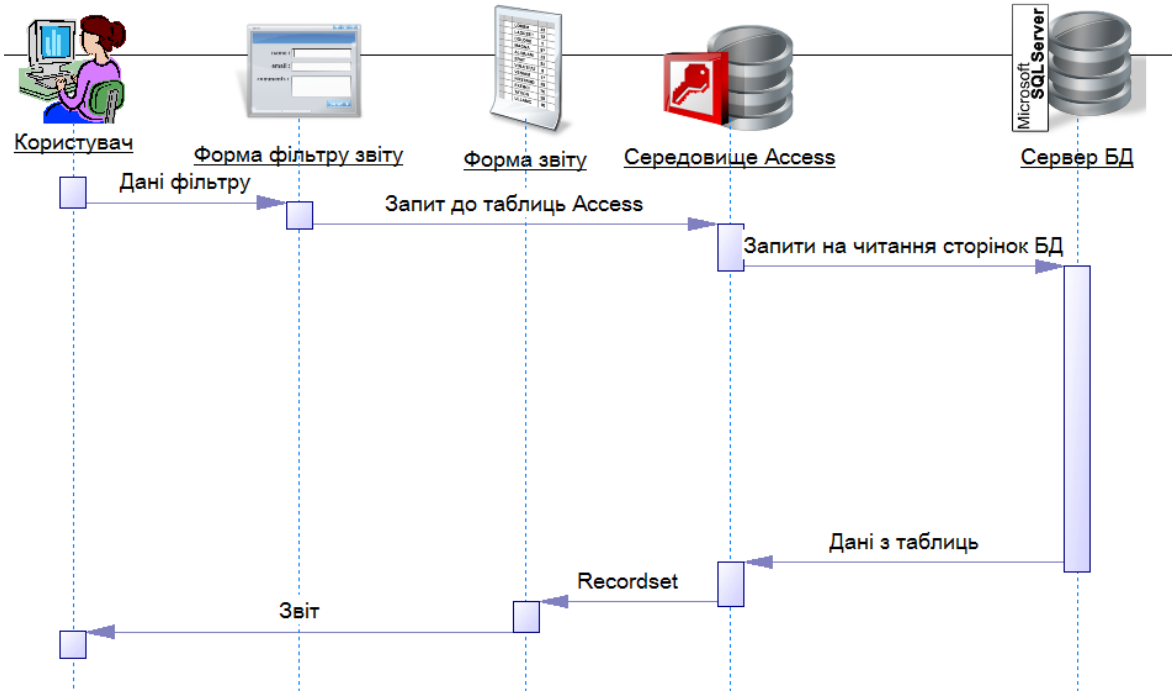

Рисунок 9 - Видача звіту за допомогою середовища MsAccess

Другий шлях видачі звіту - на основі запиту до сервера, який формується в формі фільтру звіту MsAccess і виконується на сервері.

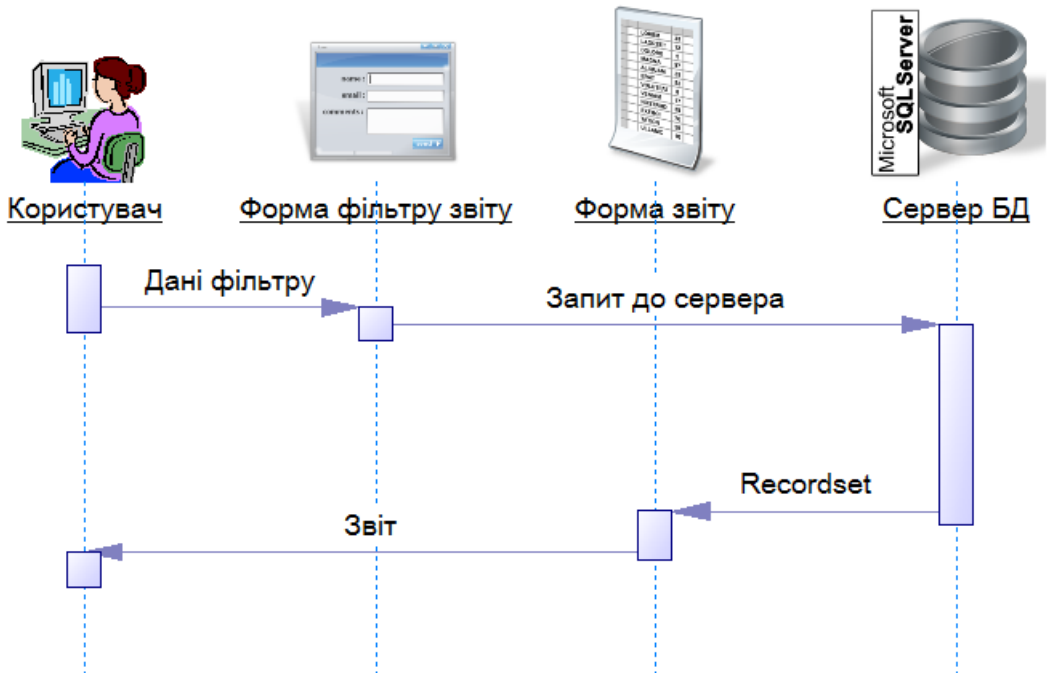

Рисунок 10 - Видача звіту за допомогою запиту до сервера MsSQL

Третій шлях видачі звіту - на основі запиту до сервера, який звертається до проміжних даних, збережених в TempDb.

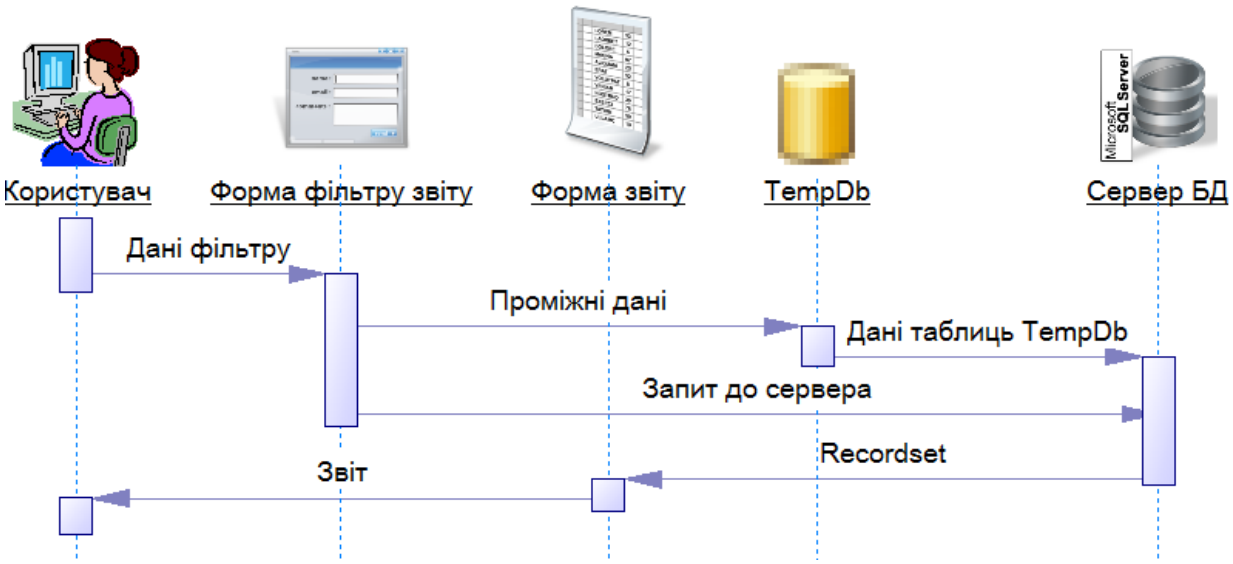

Рисунок 11 - Видача звіту за допомогою запиту до сервера MsSQL і за участі тимчасової бази в ОП TempDb

Звіт – «Головна книга» - аналіз оборотів у кредит бухгалтерського рахунку і в дебет кореспондуючих рахунків з початку року. Він потребував агрегації десятків тисяч бухгалтерських проводок з мільйона проводок у таблиці. Першим шляхом ніхто не пробував його видавати. Другим шляхом він видавався 3 хвилини, що неприйнятно для користувача. Третім шляхом він видавався 3-4 секунди.