Задание 2

Предложить правки запросов без модификации структуры БД такие, что время выполнения запроса получения 3 000 заказов из 50 000 со средним количеством элементов в заказе равным 3 не будет превышать 1-2 сек.

Для выполнения задания используется **Gpt 4.0.**

|  |  |
| --- | --- |
| Promt | Code |
| Optimize the SQL script | CREATE FUNCTION [dbo].[F\_WORKS\_LIST] ()  RETURNS TABLE  AS  RETURN  (  SELECT  w.Id\_Work,  w.CREATE\_Date,  w.MaterialNumber,  w.IS\_Complit,  w.FIO,  CONVERT(VARCHAR(10), w.CREATE\_Date, 104) AS D\_DATE,  dbo.F\_WORKITEMS\_COUNT\_BY\_ID\_WORK(w.Id\_Work, 0) AS WorkItemsNotComplit,  dbo.F\_WORKITEMS\_COUNT\_BY\_ID\_WORK(w.Id\_Work, 1) AS WorkItemsComplit,  dbo.F\_EMPLOYEE\_FULLNAME(w.Id\_Employee) AS EmployeeFullName,  w.StatusId,  ws.StatusName,  CASE  WHEN w.Print\_Date IS NOT NULL OR w.SendToClientDate IS NOT NULL OR w.SendToDoctorDate IS NOT NULL OR w.SendToOrgDate IS NOT NULL OR w.SendToFax IS NOT NULL  THEN 1  ELSE 0  END AS Is\_Print  FROM Works w  LEFT JOIN WorkStatus ws ON w.StatusId = ws.StatusID  WHERE w.IS\_DEL <> 1  ORDER BY w.Id\_Work DESC -- Or w.MaterialNumber DESC if needed  );  GO |
| Optimize the sql script if result = 3,000 & out = 50,000 | CREATE FUNCTION [dbo].[F\_WORKS\_LIST] ()  RETURNS TABLE  AS  RETURN  (  SELECT TOP 3000  w.Id\_Work,  w.CREATE\_Date,  w.MaterialNumber,  w.IS\_Complit,  w.FIO,  CONVERT(VARCHAR(10), w.CREATE\_Date, 104) AS D\_DATE,  (SELECT COUNT(\*) FROM WorkItem wi WHERE wi.Id\_Work = w.Id\_Work AND wi.IS\_Complit = 0) AS WorkItemsNotComplit,  (SELECT COUNT(\*) FROM WorkItem wi WHERE wi.Id\_Work = w.Id\_Work AND wi.IS\_Complit = 1) AS WorkItemsComplit,  dbo.F\_EMPLOYEE\_FULLNAME(w.Id\_Employee) AS EmployeeFullName,  w.StatusId,  ws.StatusName,  CASE  WHEN w.Print\_Date IS NOT NULL OR w.SendToClientDate IS NOT NULL OR w.SendToDoctorDate IS NOT NULL OR w.SendToOrgDate IS NOT NULL OR w.SendToFax IS NOT NULL  THEN 1  ELSE 0  END AS Is\_Print  FROM Works w  LEFT JOIN WorkStatus ws ON w.StatusId = ws.StatusID  WHERE w.IS\_DEL <> 1  ORDER BY w.id\_Work DESC  )  GO |

При уточнении запроса (добавления условия: if result = 3,000 & out = 50,000) качество сгенерированного скрипта повысилось. Как минимум было добавлено ограничение: TOP 3000, что должно значительно повысить скорость выполнения запроса.

Еще одно отличие: dbo.F\_WORKITEMS\_COUNT\_BY\_ID\_WORK(w.Id\_Work, 0)

заменено на подзапрос:

(SELECT COUNT(\*) FROM WorkItems wi WHERE wi.Id\_Work = w.Id\_Work AND wi.IsComplit = 0) AS WorkItemsNotComplit.

Примечание: наверное, стоит убрать сортировку, так как это точно влияет на производительность. Если пользователю понадобится сортировать эти данные, он сможет сделать это отдельным запросом. Тем самым мы избавимся от лишнего перебора значений в тех случаях, когда в этом нет необходимости.

**Вывод компилятора для ф-ции промта (Optimize the sql script if result = 3,000 & out = 50,000):**

**Время синтаксического анализа и компиляции SQL Server:**

**время ЦП = 0 мс, истекшее время = 0 мс.**

**Время работы SQL Server:**

**Время ЦП = 0 мс, затраченное время = 0 мс.**

**(3000 rows affected)**

**Время работы SQL Server:**

**Время ЦП = 109 мс, затраченное время = 224 мс.**

**Completion time: 2024-05-30T01:58:04.9615602+03:00**

Для измерения скорости запроса использовали: SET STATISTICS TIME ON;

**Объяснение изменений:**

* Предложение TOP 3000: ограничивает результирующий набор желаемым количеством строк.
* GROUP BY и SUM вместо отдельных подзапросов: это более эффективный подход, особенно когда среднее количество рабочих элементов в работе относительно невелико.
* Объединение рабочих элементов: объединяем таблицу Works с рабочими элементами, используя id\_work для вычисления количества непосредственно в запросе.

**Как работает эта оптимизация:**

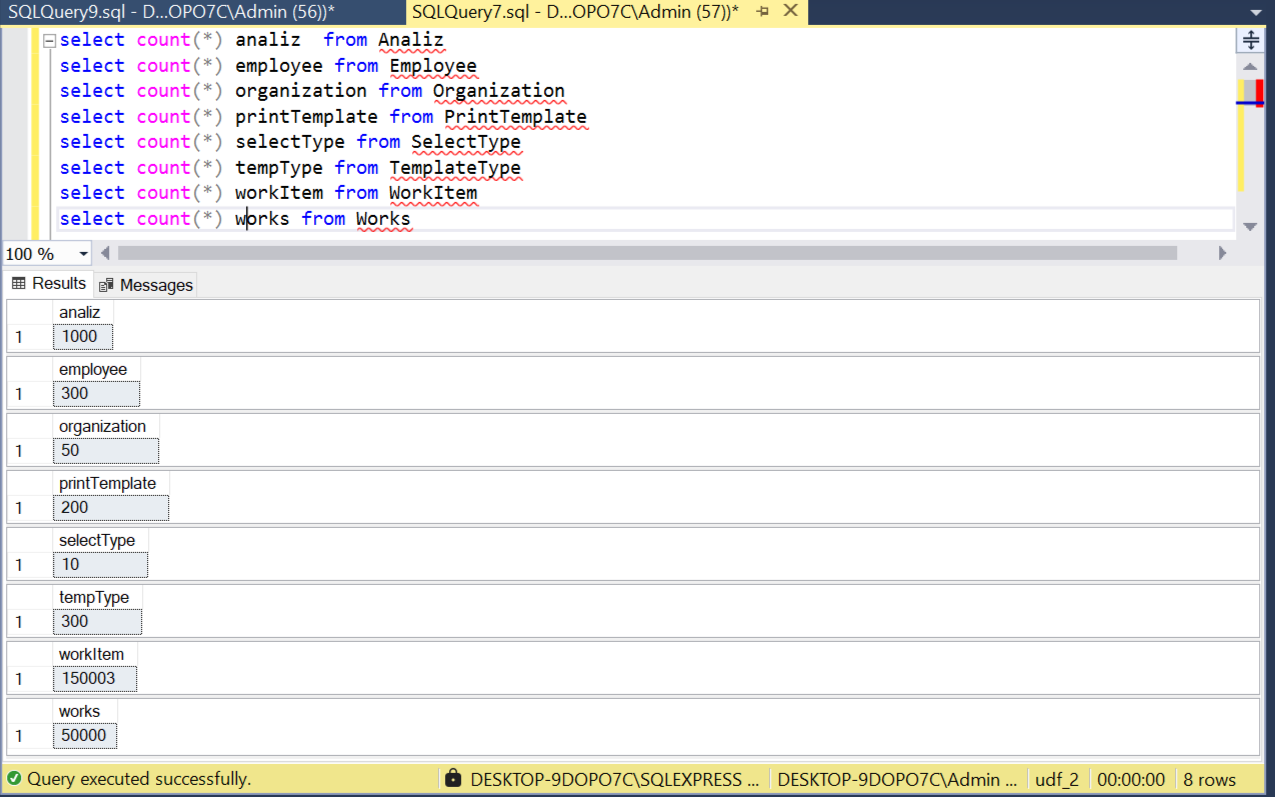
**Эффективная агрегация данных**: Предложение GROUP BY эффективно группирует рабочие элементы, связанные с каждой рабочей записью, а функции суммирования с выражениями CASE предоставляют краткий способ вычисления желаемых значений.

**Сокращение количества запросов**: объединение логики подсчета в одном запросе устраняет необходимость в нескольких запросах или вызовах функций, что приводит к меньшему количеству обращений к базе данных и потенциально более высокой производительности.

**Правильная группировка**: GROUP BY гарантирует, что значения для WorkItemsNotComplit и WorkItemsComplit будут рассчитаны правильно для каждой рабочей записи.

**Дополнительно:**

**Наполнение БД.**



Скрипт заполнения:

-- Insert test data into Analiz table

DECLARE @k INT = 1;

WHILE @k <= 1000

BEGIN

INSERT INTO Analiz (IS\_GROUP, MATERIAL\_TYPE, CODE\_NAME, FULL\_NAME, ID\_ILL, Text\_Norm, Price, NormText, UnNormText)

VALUES (CAST(RAND() \* 2 AS BIT), FLOOR(RAND() \* 10), 'Code' + CAST(@k AS VARCHAR), 'Full Name ' + CAST(@k AS VARCHAR),

FLOOR(RAND() \* 100), 'Text Norm', RAND() \* 100, 'Norm Text ' + CAST(@k AS VARCHAR), 'UnNorm Text ' + CAST(@k AS VARCHAR));

SET @k = @k + 1;

END;

GO

-- Insert test data into Employee table

DECLARE @l INT = 1;

WHILE @l <= 100

BEGIN

INSERT INTO Employee (Login\_Name, Name, Patronymic, Surname, Email, Post, CreateDate, UpdateDate, Archived, IS\_Role, Role)

VALUES ('Login' + CAST(@l AS VARCHAR) + CONVERT(VARCHAR(36), NEWID()), 'Name' + CAST(@l AS VARCHAR), 'Patronymic' + CAST(@l AS VARCHAR),

'Surname' + CAST(@l AS VARCHAR), 'Email' + CAST(@l AS VARCHAR) + '@example.com', 'Post' + CAST(@l AS VARCHAR),

GETDATE(), GETDATE(), CAST(RAND() \* 2 AS BIT), CAST(RAND() \* 2 AS BIT), FLOOR(RAND() \* 5));

SET @l = @l + 1;

END;

GO

-- Insert test data into TemplateType table

DECLARE @o INT = 1;

WHILE @o <= 100

BEGIN

INSERT INTO TemplateType (TemlateVal, Comment)

VALUES ('Template Val ' + CAST(@o AS VARCHAR), 'Comment ' + CAST(@o AS VARCHAR));

SET @o = @o + 1;

END;

GO

-- Insert test data into PrintTemplate table

DECLARE @n INT = 1;

WHILE @n <= 100

BEGIN

INSERT INTO PrintTemplate (TemplateName, CreateDate, Ext, Comment, TemplateBody, Id\_TemplateType)

VALUES ('Template Name ' + CAST(@n AS VARCHAR), GETDATE(), '.ext', 'Comment ' + CAST(@n AS VARCHAR),

0x, @n % 100 + 1);

SET @n = @n + 1;

END;

GO

-- Insert test data into Organization table

DECLARE @m INT = 1;

WHILE @m <= 50

BEGIN

INSERT INTO Organization (ORG\_NAME, TEMPLATE\_FN, Id\_PrintTemplate, Email, SecondEmail, Fax, SecondFax)

VALUES ('Org Name ' + CAST(@m AS VARCHAR), 'Template FN ' + CAST(@m AS VARCHAR), @m % 100 + 1,

'email' + CAST(@m AS VARCHAR) + '@example.com', 'secondemail' + CAST(@m AS VARCHAR) + '@example.com',

'123456789' + CAST(@m AS VARCHAR), '987654321' + CAST(@m AS VARCHAR));

SET @m = @m + 1;

END;

GO

-- Insert test data into SelectType table

DECLARE @p INT = 1;

WHILE @p <= 10

BEGIN

INSERT INTO SelectType (SelectType)

VALUES ('Select Type ' + CAST(@p AS VARCHAR));

SET @p = @p + 1;

END;

GO

-- Insert test data into Works table

DECLARE @w INT = 1;

WHILE @w <= 50000

BEGIN

INSERT INTO Works (CREATE\_Date, MaterialNumber, IS\_Complit, FIO, Id\_Employee, StatusId)

VALUES (GETDATE(), RAND() \* 1000, CAST(RAND() \* 2 AS BIT), 'FIO ' + CAST(@w AS VARCHAR), @w % 100 + 1, @w % 10 + 1);

SET @w = @w + 1;

END;

GO

-- Insert test data into WorkItem table

DECLARE @j INT = 1;

DECLARE @i INT = 1;

WHILE @j <= 50000

BEGIN

WHILE @i <= 3

BEGIN

INSERT INTO WorkItem (id\_work, id\_analiz, is\_complit)

VALUES (

CEILING(200001 + RAND() \* (250000 - 200001)),

CEILING(2001 + RAND() \* (3000 - 2001)),

CAST(RAND() \* 2 AS BIT)

);

SET @i = @i + 1;

END

SET @j = @j + 1;

SET @i = 1;

END;

GO