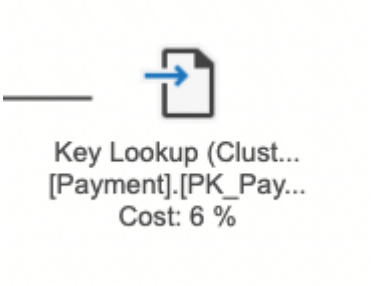
1. Оценка расчёта балансов

`F_CalculatePaymentParticipantBalance`

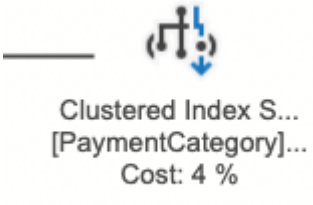
План запроса построен с использованием следующего кода (литерал строки - oid существующего payer'a):

```
DECLARE @id UNIQUEIDENTIFIER;
DECLARE @result int;
SET @id = CAST('6d82168c-a9c2-432b-b3cb-e8dc20777792' AS UNIQUEIDENTIFIER);
SET @result = dbo.F_CalculatePaymentParticipantBalance(@id);
```

2 раза происходит поиск ключа в кластерном индексе по PK_Payment, каждый по ~6% общей стоимости запроса



Остальную стоимость запроса составляют поиски по кластерным/некластерным индексам, все по ~4%



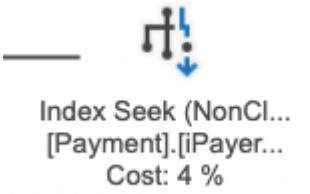


Таблица Top Operations с суммированными и округленными значениями стоимости (из-за округлений общая стоимость может быть меньше 100%):

Operation	Object	Est Cost
Clustered Index Seek	[PaymentData].[dbo].[Payment].[PK_Payment](Clustered)	11
Clustered Index Scan	[PaymentData].[dbo].[AccountType].[PK_AccountType](Clustered)	8
Clustered Index Seek	[PaymentData].[dbo].[Bank].[PK_Bank](Clustered)	8
Clustered Index Seek	[PaymentData].[dbo].[Cashbox].[PK_Cashbox](Clustered)	8
Clustered Index Seek	[PaymentData].[dbo].[Bank].[PK_Bank](Clustered)	7
Clustered Index Scan	[PaymentData].[dbo].[AccountType].[PK_AccountType](Clustered)	7
Clustered Index Scan	[PaymentData].[dbo].[Cashbox].[PK_Cashbox](Clustered)	7
Clustered Index Scan	[PaymentData].[dbo].[AccountType].[PK_AccountType](Clustered)	7
Clustered Index Scan	[PaymentData].[dbo].[AccountType].[PK_AccountType](Clustered)	7
Clustered Index Seek	[PaymentData].[dbo].[PaymentParticipant].[PK_PaymentParticipant](Clustered)	7
Clustered Index Seek	[PaymentData].[dbo].[PaymentCategory].[PK_PaymentCategory](Clustered)	7
Clustered Index Seek	[PaymentData].[dbo].[PaymentParticipant].[PK_PaymentParticipant](Clustered)	7
Index Seek	[PaymentData].[dbo].[Payment].[iPayee_Payment](NonClustered)	7

F_CalculateProjectBalance

План запроса построен с использованием следующего кода (литерал строки - oid существующего project'a):

```
DECLARE @ProjectId UNIQUEIDENTIFIER;
DECLARE @Result int;
SET @ProjectId = CAST('40004eda-e3d6-49ab-85f4-d44ac783c5e6' AS UNIQUEIDENTIFIER);
SET @Result = dbo.F_CalculateProjectBalance(@ProjectId);
```

В плане запроса для данной функции есть секция для вычисления стоимости проекта, а также отдельные секции планов для функций `F_CalculateBalanceByMaterial` и `F_CalculateBalanceByMaterial`, т.к. они вызываются при подсчёте конечного результата целевой функции.

Таблица стоимостей операций для секции вычисления Cost:

Operation	Object	Est Cost
Clustered Index Seek	[PaymentData].[dbo].[PaymentParticipant].[PK_PaymentParticipant](Clustered)	15
Clustered Index Seek	[PaymentData].[dbo].[PaymentParticipant].[PK_PaymentParticipant](Clustered)	14
Clustered Index Seek	[PaymentData].[dbo].[PaymentCategory].[PK_PaymentCategory](Clustered)	14
Clustered Index Seek	[PaymentData].[dbo].[Project].[PK_Project](Clustered)	14
Clustered Index Seek	[PaymentData].[dbo].[Client].[PK_Client](Clustered)	14
Index Seek	[PaymentData].[dbo].[Payment].[iProject_Payment](NonClustered)	14
Clustered Index Seek	[PaymentData].[dbo].[Payment].[PK_Payment](Clustered)	14

Стоимости операций для остальных функций будут рассмотрены ниже.

F_CalculateBalanceByMaterial

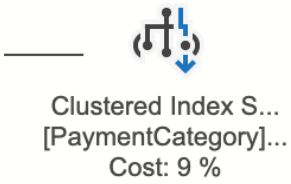
План запроса построен с использованием следующего кода (литерал строки - oid существующего project'а):

```
DECLARE @ProjectId UNIQUEIDENTIFIER;
DECLARE @Result int;
SET @ProjectId = CAST('40004eda-e3d6-49ab-85f4-d44ac783c5e6' AS UNIQUEIDENTIFIER);
SET @Result = dbo.F_CalculateBalanceByMaterial(@ProjectId);
```

План запроса у этой функции делится на 2 ветки: вычисление Profit и вычисление Cost.

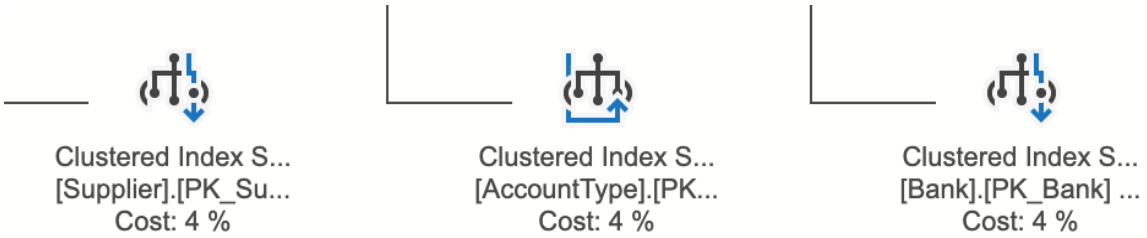
В ветке Profit:

Поиск по индексу платёжных категорий - 9%



Поиск по индексу получателей денег - 6%

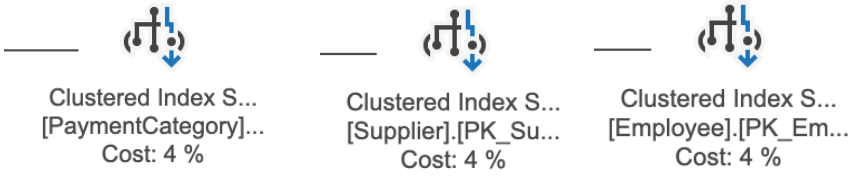
Поиск по индексам плательщиков, типов аккаунта и банков - по 4%



И далее 10 поисков по различным индексам - по 3%

В ветке Cost:

Поиск по индексам категорий, плательщиков и сотрудников - по 4%



И далее так же 10 поисков по различным индексам - по 3%

Таблица Top Operations с суммированными и округленными значениями стоимости:

Operation	Object	Est Cost
Clustered Index Seek	[PaymentData].[dbo].[PaymentCategory].[PK_PaymentCategory](Clustered)	15
Clustered Index Seek	[PaymentData].[dbo].[Supplier].[PK_Supplier](Clustered)	10
Clustered Index Seek	[PaymentData].[dbo].[Supplier].[PK_Supplier](Clustered)	7
Clustered Index Scan	[PaymentData].[dbo].[AccountType].[PK_AccountType](Clustered)	6
Clustered Index Seek	[PaymentData].[dbo].[Bank].[PK_Bank](Clustered)	6
Clustered Index Seek	[PaymentData].[dbo].[Project].[PK_Project](Clustered)	5
Clustered Index Scan	[PaymentData].[dbo].[AccountType].[PK_AccountType](Clustered)	5
Clustered Index Scan	[PaymentData].[dbo].[Cashbox].[PK_Cashbox](Clustered)	5
Clustered Index Scan	[PaymentData].[dbo].[AccountType].[PK_AccountType](Clustered)	5
Clustered Index Seek	[PaymentData].[dbo].[Bank].[PK_Bank](Clustered)	5
Clustered Index Scan	[PaymentData].[dbo].[AccountType].[PK_AccountType](Clustered)	5
Clustered Index Scan	[PaymentData].[dbo].[Cashbox].[PK_Cashbox](Clustered)	5
Clustered Index Seek	[PaymentData].[dbo].[PaymentParticipant].[PK_PaymentParticipant](Clustered)	5
Clustered Index Seek	[PaymentData].[dbo].[PaymentParticipant].[PK_PaymentParticipant](Clustered)	5
Index Seek	[PaymentData].[dbo].[Payment].[iProject_Payment](NonClustered)	5
Clustered Index Seek	[PaymentData].[dbo].[Payment].[PK_Payment](Clustered)	5

F_CalculateBalanceByWork

План запроса построен с использованием следующего кода (литерал строки - oid существующего project'a):

```
DECLARE @ProjectId UNIQUEIDENTIFIER;
DECLARE @Result int;
SET @ProjectId = CAST('40004eda-e3d6-49ab-85f4-d44ac783c5e6' AS UNIQUEIDENTIFIER);
SET @Result = dbo.F_CalculateBalanceByWork(@ProjectId);
```

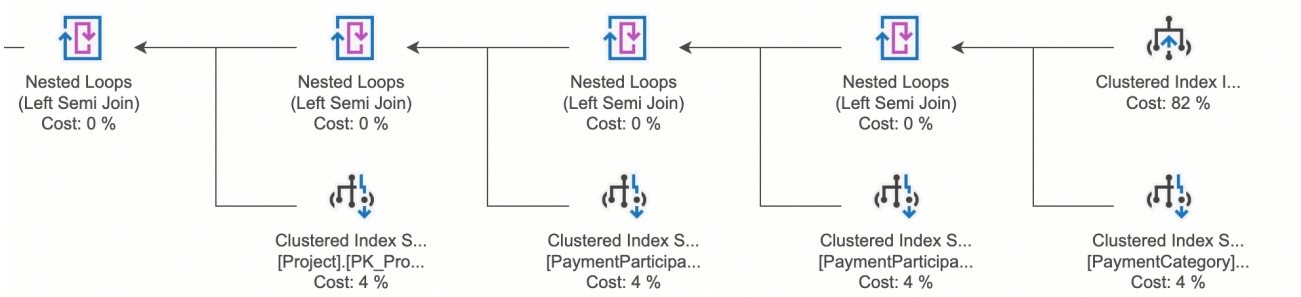
Тут план запроса расходится на те же 2 ветки, однако в данном случае ветка Cost выполняется гораздо быстрее ветки Project.

По стоимости обращения к индексам в этом запросе примерно равны, поэтому выделить особо нечего.

Таблица:

Operation	Object	Est Cost
Clustered Index Seek	[PaymentData].[dbo].[PaymentParticipant].[PK_PaymentParticipant](Clustered)	7
Clustered Index Scan	[PaymentData].[dbo].[Cashbox].[PK_Cashbox](Clustered)	7
Clustered Index Seek	[PaymentData].[dbo].[Bank].[PK_Bank](Clustered)	7
Clustered Index Scan	[PaymentData].[dbo].[AccountType].[PK_AccountType](Clustered)	7
Clustered Index Scan	[PaymentData].[dbo].[AccountType].[PK_AccountType](Clustered)	7
Clustered Index Scan	[PaymentData].[dbo].[AccountType].[PK_AccountType](Clustered)	7
Clustered Index Seek	[PaymentData].[dbo].[Project].[PK_Project](Clustered)	6
Index Seek	[PaymentData].[dbo].[Payment].[iProject_Payment](NonClustered)	6
Clustered Index Seek	[PaymentData].[dbo].[Payment].[PK_Payment](Clustered)	6
Clustered Index Seek	[PaymentData].[dbo].[PaymentParticipant].[PK_PaymentParticipant](Clustered)	6
Clustered Index Seek	[PaymentData].[dbo].[PaymentCategory].[PK_PaymentCategory](Clustered)	6
Clustered Index Seek	[PaymentData].[dbo].[Client].[PK_Client](Clustered)	6
Clustered Index Seek	[PaymentData].[dbo].[Cashbox].[PK_Cashbox](Clustered)	6
Clustered Index Seek	[PaymentData].[dbo].[Bank].[PK_Bank](Clustered)	6
Clustered Index Seek	[PaymentData].[dbo].[AccountType].[PK_AccountType](Clustered)	6

Также посмотрел на план запроса insert одной записи платежа, оказалось, что вставка в индекс в нём занимает 82% от общего времени.



Operation	Object	Est Cost
Clustered Index Insert(Insert)	[PaymentData].[dbo].[Payment].[PK_Payment](Clustered)	82
Clustered Index Seek	[PaymentData].[dbo].[Project].[PK_Project](Clustered)	4
Clustered Index Seek	[PaymentData].[dbo].[PaymentParticipant].[PK_PaymentParticipant](Clustered)	4
Clustered Index Seek	[PaymentData].[dbo].[PaymentParticipant].[PK_PaymentParticipant](Clustered)	4
Clustered Index Seek	[PaymentData].[dbo].[PaymentCategory].[PK_PaymentCategory](Clustered)	4

Стоит отметить, что Azure Data Studio (по крайней мере на macOS) почему-то не показывает планы запросов для сработавших триггеров, хотя в стандартных утилитах для работы с SQL Server (на Windows) они указаны.

Дальнейшие наблюдения были получены с использованием SQL Server Management Studio (на Windows) с построением плана запроса для операции вставки платежа (полученный план запроса находится в файле insert_payment_smss.sqlplan в директории с планами запросов в репозитории).

Округленные значения:

- 3% - вставка платежа в таблицу
- 24% - обновление баланса у новых участников:
  - У плательщика (вычисление 5% + обновление 7%)
  - У получателя (вычисление 5% + обновление 7%)
- 22% - обновление баланса у старых участников:
  - У плательщика (вычисление 5% + обновление 6%)
  - У получателя (вычисление 5% + обновление 6%)
- 26% - обновление баланса у новых проектов (вычисление 8% + обновление 18%)
- 26% - обновление баланса у старых проектов (вычисление 8% + обновление 18%)

2. Сценарий оптимизации

Т.к. **Оператор** не имеет доступа к балансам, не имеет смысла тратить время на их пересчет при внесении каждого нового платежа, уменьшая эффективность работы сотрудника.

На данную ситуацию стоит спроецировать логику работы паттерна **Unit Of Work**: при выполнении одного запроса к API множество репозиториев работают с одним UoW, а в конце выполнения запроса UoW сохраняет суммарные изменения в базу данных.

В нашем случае роль UoW будет выполнять Оператор. Тогда придётся отключить триггер `T_Payment_AI`. Оператор добавит все платежи, после чего можно будет инициировать пересчёт балансов. Перед этим стоит как-то просуммировать платежи с одинаковыми участниками+проектами: составить список уникальных сущностей, для которых потребуется пересчёт балансов. Также следует создать таблицу, в которой будут отмечаться периоды работы оператора, и на её изменения повесить триггер по пересчёту балансов (также можно рассмотреть варианты когда функция пересчёта вызывается оператором вручную, либо по мере накопления определенного числа запросов). В записях этой таблицы стоит хранить отметку, с которой начинаются новые платежи (это может быть oid первого нового платежа или его create date), чтобы триггер итерировался именно по новым платежам, а не по всем существующим.

Таким образом, пересчёт балансов станет отложенной операцией, а данные по балансам и платежам будут согласованными в конечном счёте (после завершения работы оператора и операций пересчёта),

3. Недостатки сценария

- Балансы не актуальны в период работы оператора
- Оператору и Аналитику невозможно работать параллельно по мере добавления платежей, работать им придётся друг за другом.

- Есть предположение, что пересчёт платежей будет выполняться очень долго из-за их количества, но это нужно будет замерять.