Первый этап

|  |  |
| --- | --- |
| **Индекс** | **Время выполнения** |
| - | 1330.00 мс |
| USE PaymentData  GO  -- Создание кластерного индекса на таблице Payment  CREATE CLUSTERED INDEX iPayment\_PayerPayeeDate  ON dbo.Payment (Payer, Payee, Date)  GO  -- Создание некластерного индекса на таблице PaymentParticipant  CREATE NONCLUSTERED INDEX iPaymentParticipant\_Balance  ON dbo.PaymentParticipant (Balance)  INCLUDE (Oid, Name, ObjectType, ActiveFrom, InactiveFrom, BankDetails, Balance2, Balance3)  GO  -- Создание некластерного индекса на таблице Project  CREATE NONCLUSTERED INDEX iProject\_Client  ON dbo.Project (Client)  GO  -- Создание некластерного индекса на таблице Project  CREATE NONCLUSTERED INDEX iProject\_Manager\_Foreman  ON dbo.Project (Manager, Foreman)  GO  -- Создание некластерного индекса на таблице PaymentCategory  CREATE NONCLUSTERED INDEX iPaymentCategory\_Flags  ON dbo.PaymentCategory (ProfitByMaterial, CostByMaterial, NotInPaymentParticipantProfit)  GO | 1353.00 мс |

**(100 – операций)**  
Изначально мы решили сделать «комплексное обновление»: накинуть сразу много индексов, которые потенциально могли бы помочь производительности. Но, как показано в таблице, результаты не показали прироста (скорее наоборот). Мы подумали, что необходимо увеличить число итераций в тестировании (как минимум), и делать менее «комплексные» обновления, а добавлять индексы поочередно.

Второй этап

|  |  |
| --- | --- |
| **Индекс** | **Время выполнения** |
| * (т.е. все дефолтно) | 34103.00 мс |
| CREATE NONCLUSTERED INDEX iPayment\_Date  ON dbo.Payment (Date) | 28363.00 мс |
| -- Создание некластерного индекса на столбце CostByMaterial  CREATE NONCLUSTERED INDEX iPaymentCategory\_CostByMaterial  ON dbo.PaymentCategory (CostByMaterial)  INCLUDE (Oid, Name, OptimisticLockField, GCRecord, ProfitByMaterial, NotInPaymentParticipantProfit)  GO  -- Создание некластерного индекса на столбце ProfitByMaterial  CREATE NONCLUSTERED INDEX iPaymentCategory\_ProfitByMaterial  ON dbo.PaymentCategory (ProfitByMaterial)  INCLUDE (Oid, Name, OptimisticLockField, GCRecord, CostByMaterial, NotInPaymentParticipantProfit)  GO | 26583.00 мс |
| CREATE NONCLUSTERED INDEX iPayment\_Date  ON dbo.Payment (Date)  -- Создание некластерного индекса на столбце CostByMaterial  CREATE NONCLUSTERED INDEX iPaymentCategory\_CostByMaterial  ON dbo.PaymentCategory (CostByMaterial)  INCLUDE (Oid, Name, OptimisticLockField, GCRecord, ProfitByMaterial, NotInPaymentParticipantProfit)  GO  -- Создание некластерного индекса на столбце ProfitByMaterial  CREATE NONCLUSTERED INDEX iPaymentCategory\_ProfitByMaterial  ON dbo.PaymentCategory (ProfitByMaterial)  INCLUDE (Oid, Name, OptimisticLockField, GCRecord, CostByMaterial, NotInPaymentParticipantProfit)  GO | 25886.00 мс |

(1000 операций)  
  
Мы увеличили число итераций, добавили индекс на дату.  
Идея: данные о платежах чаще всего добавляются в хронологическом порядке (хотя в теории может быть и не так), поэтому хранить их в отсортированном виде, при этом не перестраивая индекс с каждой операцией вставки кажется неплохой идеей  
  
Затем мы проанализировали триггер T\_payment\_AI, решили посмотреть на индексы PaymentCategory, обнаружили там явную недостаточность 😊 Таблица используется на постоянной основе, т.к. платеж зависит от типа, следовательно добавление индексов в теории должно было дать прирост, в чем мы убедились на практике, добавив пару индексов  
  
Затем мы объединили все три индекса вместе и получили итоговый результат