Реализовать индексы, повышающие производительность операций вставки и изменения платежей без модификации программных компонент

Для начала необходимо замерить, как ведет себя имеющаяся таблица при вставке в нее нового элемента. Для этого напишем функцию, которая перед началом выполнения создает переменную с текущим timestamp, затем проводит вставку значений в таблицу, и наконец вычитает из текущего timestamp сохраненный перед выполнением. Так как операция эта достаточно быстрая и время работы системы может занимать весомое время от времени выполнения всей функции, проведем эту вставку большое число раз (к примеру, 1000)

```
USE PaymentData;
GO

DECLARE @iter INT = 0;
DECLARE @max_iter INT = 1000;
DECLARE @startTime DATETIME, @endTime DATETIME, @duration DECIMAL(18,2);

PRINT 'Начало вставки платежей...';

SET @startTime = GETUTCDATE();

WHILE @iter < @max_iter
BEGIN
  DECLARE @r_payee UNIQUEIDENTIFIER = (SELECT TOP (1) PaymentParticipant.Oid
FROM dbo.PaymentParticipant ORDER BY NEWID());
  DECLARE @r_payer UNIQUEIDENTIFIER = (SELECT TOP (1) PaymentParticipant.Oid
FROM dbo.PaymentParticipant ORDER BY NEWID());
  DECLARE @r_category UNIQUEIDENTIFIER = (SELECT TOP (1) PaymentCategory.Oid
FROM dbo.PaymentCategory ORDER BY NEWID());
  DECLARE @r_project UNIQUEIDENTIFIER = (SELECT TOP (1) Project.Oid FROM
dbo.Project ORDER BY NEWID());

  INSERT dbo.Payment(
    Oid,
    Amount,
    Category,
    Project,
    Justification,
    Comment,
    Date,
```

```
    Payer,
    Payee,
    OptimisticLockField,
    GCRecord,
    CreateDate,
    CheckNumber,
    IsAuthorized,
    Number
  ) VALUES (
    NEWID(),
    69,
    @r_category,
    @r_project,
    '0P0',
    'test',
    '2023-06-05 00:00:00.000',
    @r_payer,
    @r_payee,
    -6673,
    5412,
    '2023-07-05 00:00:00.000',
    N'1234',
    0,
    N'12312'
  );

  SET @iter = @iter + 1;
END;

SET @endTime = GETUTCDATE();
SET @duration = DATEDIFF(ms, @startTime, @endTime);

PRINT 'Вставка платежей завершена.';
PRINT 'Время выполнения: ' + CAST(@duration AS VARCHAR(20)) + ' мс';
```

Время выполнения запроса составляет 31124мс.

Теперь поочередно будем добавлять индексы на наиболее подходящие поля.
Одним из best-practice является добавление индекса на таблицы, содержащие временные данные. Рассмотрим поведение системы при добавлении индекса на поле Payment.Date

```
CREATE NONCLUSTERED INDEX iPayment_Date
ON dbo.Payment (Date)
```

После добавления индекса время выполнения запросов начало составлять 25424мс

Следующим шагом добавим индекс на столбец ProfitByMaterial

```
CREATE NONCLUSTERED INDEX iProfitByMaterial
ON dbo.PaymentCategory (ProfitByMaterial)
INCLUDE (Oid, Name, OptimisticLockField, GCRecord, CostByMaterial,
NotInPayementPracticipantProfit)
```

После добавление второго индекса время выполнения запросов стало составлять 24012мс

Сейчас попробуем добавить индексы вообще на все существующие таблицы

```
CREATE NONCLUSTERED INDEX iPayment_Date
ON dbo.Payment (Date)

CREATE NONCLUSTERED INDEX iCostByMaterial
ON dbo.PaymentCategory (CostByMaterial)
INCLUDE (Oid, Name, OptimisticLockField, GCRecord, ProfitByMaterial,
NotInPaymentParticipantProfit)
GO

CREATE CLUSTERED INDEX iPayment_PayerPayeeDate
ON dbo.Payment (Payer, Payee, Date)
GO

CREATE NONCLUSTERED INDEX iPaymentParticipant_Balance
ON dbo.PaymentParticipant (Balance)
INCLUDE (Oid, Name, ObjectType, ActiveFrom, InactiveFrom, BankDetails,
Balance2, Balance3)
GO


CREATE NONCLUSTERED INDEX iProject_Client
ON dbo.Project (Client)
GO

CREATE NONCLUSTERED INDEX iProject_Manager_Foreman
ON dbo.Project (Manager, Foreman)
GO


CREATE NONCLUSTERED INDEX iPaymentCategory_Flags
ON dbo.PaymentCategory (ProfitByMaterial, CostByMaterial,
```

```
NotInPaymentParticipantProfit)
GO
```

Время выполнения запросов теперь составляет 28052мс

Из этого мы можем заключить, что добавление лишних индексов может ухудшить производительность системы. Однако при адекватном подходе можно ускорить выполнение запросов на несколько десятков процентов.