Задание 1

Рассматривается проблема такого побочного эффекта параллелизации транзакций как возможные "*грязные прочтения*" (Dirty reads). То есть прочтение незафиксированной (незакоммиченной) информации Если, например, произойдет откат выполняемой транзакции уже после того как другой процесс считает изменные данные, то такие данные уже будут невалидны (dirty data).

Поэтому важно выбрать подоходящий уровень изоляции транзакций.

- Read Commited уровень изоляции позволяет транзакции читать данные, которые могут быть залочены другими транзакциями.
- С другой стороны, Read Committed не может прочитать ресурс, который был зблокирован, залочен.

Поэтому *Процесс 2* точно прочитает `1`, так как дождется разблокирования ресурсов, занятых *Процессом 1*.
Процесс 3 так же считает `1`, если представленная таблица точно демонстрирует поочередность исполняемых процессов (то есть если нет никаких задержек и тд.)

Задание 2

```
select PriceAssetId, month_2021, min(ClosePrice), max(ClosePrice)
from
    (select PriceAssetId, extract(month from PriceDate) as month_2021
    from
        (select *
        from dbo.tblClosePrice
        where year(PriceDate) = '2021') as t1)
group by month_2021
order by PriceAssetId
```

Задание 3 (2?)

запрос 1

```
select ClientId, Name, sum(PaymentSum) as TotalSum
from tblClientPayments
join tblClient on tblClient.ClientId = tblClientPayments.ClientId
group by ClientId
```

запрос 2

```
select ClientId, Name, TotalSum
from tblClientPayments
join
    (select ClientId, sum(PaymentSum) as TotalSum
    from tblClientPayments
    join tblClient on tblClient.ClientId = tblClientPayments.ClientId
    group by ClientId) as t1
on t1.ClientId = tblClientPayments.ClientId
where TotalSum > 7000 or PaymentDate > '2022-03-05'
```

запрос 3

```
select ClientId, Name
from
    (select ClientId, year, count(month) as paymentsInYear
    from
        (select ClientId, extract(year from PaymentDate) as year,
        extract(month from PaymentDate) as month
        from tblClientPayments
        group by ClientId)
    group by year) as t1
where paymentsInYear >= 2
```

запрос 4

```
update tblClientPayments
set PaymentSum = (PaymentSum - (0.1 * PaymentSum))
where ClientId =
    (select ClientId
    from tblClientPayments
    join tblClient on tblClientPayments.ClientId = tblClient.ClientId
    where Name = 'Кузнецова Анна Андреевна') as t1
and PaymentDate = '2022-03-02'
```

запрос 5

```
set @id = (select ClientId from ClientId where Name = 'Петров Петр Петрович')
set @currDate = getDate()
insert into tblClientPayments(ClientId, PaymentDate, PaymentSum)
values(@id, @currDate, 18000)
```

Задание 4 (3?)

Пункт 1

```
select t2.curr_price_date as DateT,
t2.PriceAssetId as AssetId,
t2.curr_close_price as PriceT,
t2.prev_close_price as PriceT1,
(100 - (cast(curr_close_price as float) * 100) / curr_close_price) asDivergence
from
    (select ClosePrice as curr_close_price,
    t1.close_price as prev_close_price,
    PriceAssetId,
    PriceDate as curr_price_date,
    max(t1.price_date) as previous_date
    from dbo.tbClosePrice
        (select PriceDate as price_date,
        PriceAssetId as asset_id,
        ClosePrice as close_price
        from dbo.tbClosePrice) as t1
    join t1 on t1.PriceAssetId = dbo.tbClosePrice.PriceAssetId
    where PriceAssetId = t1.price_asset_id
    group by PriceAssetId
    having max(PriceDate) < t1.price_date) as t2
group by AssetId
having abs(Divergence) > 30
```

Пункт 2

```
CREATE INDEX asset_id_index ON dbo.tbClosePrice (PriceAssetId);
CREATE INDEX date_index ON dbo.tbClosePrice (PriceDate);
```

Задание 5 (4?)

Операции `in/not in` часто замедляют выполнение запроса, поэтому попробуем избавиться от этих выражений, переписав запрос. Также вместо прибавления 7 дней, поставим неделю, неуверен, что это как-то повлияет на производительность, но можно проверить. Также перепишем последний `not in` на `c.ctpTypeId != 2 and c.ctpTypeId != 2` , хотя скорее всего оптимизатор и так упрощает это выражение, приводя его к предложенному виду

```
select o.RcvId
from dbo.Operations o
left join dbo.Counterparty c
on o.RcvId = c.RcvId
where c.RcvId = null
and dateadd(week, 1, o.OperDate) >= cast (getdate() as date)
and c.ctpTypeId != 2 and c.ctpTypeId != 2
```

И так как выполняется `left join` , можно добавить индексов по этим параметрам

```
CREATE INDEX operation_id_index ON dbo.Operation (RcvId);
CREATE INDEX counterparty_id_index ON dbo.Counterparty (RcvId);
```

Вообще, например, в postgresql индексы на ключи добавляются автоматически, не уверен, что это справедливо и для mssql, поэтому добавил их на всякий случай.

Задание 6 (5?)

```
namespace Train;

static class Program
{
    public static void Main(string[] args)
    {
        for (int i = 0; i < 100; i++)
        {
            var train = new RandomTrain();
            Console.Out.WriteLine($"Result of algorithm: real length = " +
                                  $"{train.RealLength}, found = {train.FindLength()}"
        }
    }
}

public class Wagon
{
    private bool _light;

    public bool Light
    {
        get => _light;
    }

    public Wagon(bool turnOnLight)
    {
        _light = turnOnLight;
    }

    public void TurnOnLight()
    {
        _light = true;
    }

    public void TurnOffLight()
    {
        _light = false;
    }
}

public class RandomTrain
{
    private readonly LinkedList<Wagon> _train;
    private readonly int _realLength;
    private LinkedListNode<Wagon> _currentWagon;

    public LinkedList<Wagon> TrainWagons
    {
        get => _train;
    }
```

```
public LinkedListNode<Wagon> CurrentWagon
{
    get => _currentWagon;
}

public int RealLength
{
    get => _realLength;
}

public RandomTrain()
{
    Random random = new Random();
    int length = random.Next(2, 10000);
    _realLength = length;
    _train = new LinkedList<Wagon>();
    for (var i = 0; i < length; ++i)
    {
        _train.AddLast(new Wagon(random.Next(100) > 20));
    }

    _currentWagon = _train.First ?? throw new InvalidOperationException();
}

public void TurnOnLightInCurrentWagon()
{
    _currentWagon.Value.TurnOnLight();
}

public void TurnOffLightInCurrentWagon()
{
    _currentWagon.Value.TurnOffLight();
}

public bool IsLightInWagon()
{
    return _currentWagon.Value.Light;
}

public void GoNextWagon()
{
    _currentWagon = _currentWagon.Next ?? _currentWagon.List.First;
}

public void GoToPreviousWagon()
{
    _currentWagon = _currentWagon.Previous ?? _currentWagon.List.Last;
}

public int FindLength()
```

```
    {
        int count;
        do
        {
            count = 0;
            TurnOnLightInCurrentWagon();
            GoNextWagon();
            while (!IsLightInWagon())
            {
                ++count;
                GoNextWagon();
            }

            TurnOffLightInCurrentWagon();
            for (var i = 0; i <= count; ++i)
                GoToPreviousWagon();
        } while (IsLightInWagon());

        return count + 1;
    }
}
```