SQL Performance Tuning Examples - 6

Case 1: SQL 2019 , 2 PR , 2 DR, 200 GB Memory, 16 VCpu , Heavy OLTP AI BOT application. Table takes 50 milliseconds at an average to insert one record. 100 BOTs are inserting records simultaniously.Throughput is not sufficient and queue is buildup day time and clears in the night time when the load is reduced. Next day same tasks are repeated. Table had instead of insert trigger that slow down the inserts.

```
    RefNum Int,
    OrderID Int,
    Amount Decimal
```

Nonclustered Index on RefNum, OrderID was there. They were checking whether RefNum, OrderID and Amount is already present. Then dont insert the record if matching record found. Otherwise insert it.

```
    If Amount is null
        Then insert records
    else
    Being
        if not exists(select 1 from mytable where refnum=@refnum and orderID =@orderid and Amount = @Amount)
            Insert record
    end
```

They are enforcing uniqueness among existing records for all 3 columns. Same time they want to insert NULL values for amount field if it is NULL.

They couldnt achieve it through UNIQUE constraint in SQL Server since it allows only single NULL value. Rest of the RDBMS Oracle, MySQL and PostgreSQL allows multiple NULL values by default. We have option to control the behaviour. Not sure why SQL Server didnt follow the same.

But we can achieve the same bevaviour through Indexes.

We recreated the Index with UNIQUE option for RefNum and OrderID where Amount is not NULL and disabled the Trigger. It brought down the insertion from 50 to 7 milliseconds.

Case 2: Same application. Inserts are blocked with exclusive locks. Parent & Child tables. Child table insert 15 to 100 records for single row in Parent table. Same time Parent table gets updated for status and amount fields.

```
        Customer
            CustID int primary key,
            Status smallint,
            Amount decimal,
            .....
        CustomerTrans
            TranID int primary key,
            CustID int foreign key references Customer(CustID),
            ....
```

CustomerTrans insertions are blocked by foreign key check, because Customer gets updated. Application opens explicit transaction and runs 3 to 5 indiviual statements and finally commits. So locks are held for a while in Customer table. Same time CustomerTrans insertion is waiting for Customer to check its reference ID existance. Ideally it should allow to check when we dont update Primary key column of parent table. But it is designed to block foreign key reference check if parent tables are in use. I tested with enabling ReadCommitted Snapshot , But behaviour remains same. The amount of load is more in child table compare to parent table. So we changed

```
        Customer
                CustID int primary key nonclustered,
                Status smallint,
                Amount decimal,
                .....
        CustomerTrans
                TranID int primary key,
                CustID int foreign key references Customer(CustID),
```

We changed parent table Primary key from clustered index to nonclustered index and added another field with AccountID which is unique clustered.

Now CustomerTrans table is able to check parent table while it is in use.

We opted for this solution since child table utilization is heavy. We dont have better option if Parent table heavily used and it will cause key lookups for any search. Generally we dont update our Primary key. Only insert or delete actions will be performed. So there should be an internal option to force not to update Primary key , so that it can be checked easily other than delete queries in parent table.