IMPLEMENTASI CONCURENCY CONTROL UNTUK APLIKASI MULTIUSER MENGGUNAKAN DATABASE SQL SERVER

Wiwi Widayani
STMIK AMIKOM Yogyakarta

Abstraksi

Permasalahan yang terjadi pada aplikasi untuk multiuser yaitu akses bersama yang dilakukan oleh beberapa user pada waktu yang bersamaan pada database dapat menyebabkan data yang tidak konsisten.Diperlukan sebuah penanganan berupa Concurrency Control untuk melindungi data dari akses secara simultan oleh multiuser.

Implementasi Concurrency Control akan dilakukan dalam database SQL Server untuk multiuser dalam suatu aplikasi untuk end user berupa aplikasi simulasi untuk Teller Bank dalam mengakses rekening Nasabah Bank.

Kata Kunci :*Concurrency Control, SQL Server,Multiuser,konsisten*

1. Pendahuluan

Permasalahan yang terjadi pada aplikasi untuk multiuser yaitu akses bersama yang dilakukan oleh beberapa user pada waktu yang bersamaan pada database dapat menyebabkan data yang tidak konsisten.Diperlukan sebuah penanganan berupa *Concurrency Control* untuk melindungi data dari akses secara simultan oleh mutliuser.

Implementasi *Concurrency Control* akan dilakukan dalam database SQL Server untuk multiuser dalama suatu aplikasi untuk end user berupa aplikasi untuk Teller Bank dalam mengakses rekening Nasabah Bank. Aplikasi yang dipakai hanya berupa simulasi bukan merupakan program utuh.

1.1 Concurrency control

Concurrency Control adalah proses pengelolaan operasi-operasi yang berjalan bersamaan dalam database dengan tidak saling menggangu satu sama lain.

Kebutuhan *Concurrency Control* dalam management transaksi untuk multiuser menjadi penting, mengingat sistem untuk multiuser memungkinkan terjadinya akses bersama terhadap sebuah database. Akses bersama relative mudah jika seluruh user hanya membaca data, sehingga tidak ada yang melakukan interfensi satu dengan lainnya. Akan tetapi, pada saat dua atau lebih user mengakses database secara simultan dan paling sedikit satu user melakukan update, maka akan terjadi interfensi yang dapat mengakibatkan ketidakkonsistenan data pada database.

1.2 Masalah yang Muncul dalam Akses Bersama dan Hasil yang Diharapkan

Terdapat beberapa masalah yang dapat terjadi yang disebabkan oleh akses bersama oleh multiuser, yaitu :

1) *Lost update problem* (Masalah hilangnya data yang diupdate).
2) *Uncommited dependency problem / dirty real* (Masalah kebergantungan terhadap transaksi yang belum *commit*).
3) *Inconsistent analiysisn problem* (masalah analisa yang tidak konsisten).
4) *Nonrepeteable (atau fuzzy) read*.
5) *Phantom read*.

1.3 Level-Level Isolasi

SQL Server memberikan empat level isolasi. Level isolasi adalah setting yang menentukan level sebuah transaksi menerima data yang tidak konsisten, yaitu tingkatan sebuah transaksi diisolasi dari transaksi lain. Semakin tinggi tingkat isolasi semakin tinggi keakuratan data. Level isolasi yang kita pakai menentukan perilaku lock untuk semua perintah SELECT yang dilakukan. Level-Level

isolasi dari terendah sampai tertinggi adalah : **Read Uncommitted, Read Committed, Repeatable Read** Dan **Serializable**.

- **READ UNCOMMITTED.** Level isolasi terendah, pada level ini transaksi diisolasi hanya untuk menjadi data yang rusak secara fisik tidak dapat dibaca (Read Only).
- **READ COMMITTED.** Level default dari SQL Server. Pada level ini pembacaan data diperbolehkan hanya pada data yang telah di-commit. Data yang telah di-commit adalah data yang merupakan bagian permanent dari database. Pada level ini data yang masih belum di-commit (masih dipakai oleh transaksi) tidak dapat dibaca.
- **REPEATABLE READ.** Pembacaan berulang pada data yang dipakai oleh sebuah transaksi akan menghasilkan data yang sama.
- **SERIALIZABLE**. Level tertinggi isolasi. Transaksi terisolasi dari transaksi lainnya. Hasil yang diperoleh dari transaksi konkuren pada sebuah database sama dengan jika transaksi dijalankan secara serial (satu per satu).

	Dirty Read	Nonrepeateable read	Phantom read
Read Uncommitted	Ya	Ya	Ya
Read committed	Tidak	Ya	Ya
Repeatable Read	Tidak	Ya	Tidak
Serializable	Tidak	Tidak	tidak

Level Isolasi Transaksi

1.4 Batasan masalah

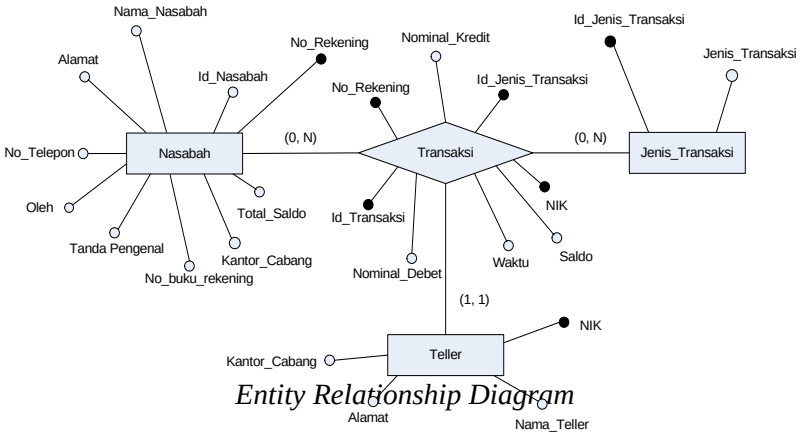
Dalam mengimplementasikan concurrency control pada aplikasi simulasi teller bank, dilakukan batasan sebagai berikut:

- Database diasumsikan database terpusat (*centralized database*)
- Analisa pengujian masalah hanya mencakup *Lost update problem* (Masalah hilangnya data yang diupdate) dan

Inconsistent analiysis problem (masalah analisa yang tidak konsisten)

2. Pembahasan

2.1 Data Modelling



Entity Relationship Diagram

2.2 Pengujian Aplikasi

Pengujian dilakukan menggunakan 2 aplikasi tanpa *concurrency control* yaitu penarikan tunai dan penyetoran tunai yang dijalankan bersama-sama dan 2 aplikasi dengan *concurrency* yaitu penarikan tunai dan penyetoran tunai yang dijalankan bersama-sama , masing-masing memiliki *delay* waktu selama 10 detik. Dengan adanya *delay* maka aplikasi yang berjalan dapat dibuat bertabrakan. Delay ditentukan dalam setiap procedure yang dipakai.

```
WAITFOR DELAY ’00:00:10’
```

Pengujian terhadap 2 aplikasi yang dijalankan bersamaan (bertubrukkan) akan merepresentasikan 2 masalah utama yaitu :

- Masalah hilangnya data yang diubah (*Lost of Update*). Sebagai contoh digunakan, transaksi penarikan tunai dan penyetoran tunai.

- Masalah analisa yang tidak konsisten (*Inconsistent analiysis problem*)

Masalah hilangnya data yang diubah (Lost of Update)

Transaksi Penarikan Tunai dilakukan terhadap nasabah dengan no rekening 00011 sebesar Rp.200000. Transaksi Penyetoran tunai juga dilakukan terhadap nasabah dengan no rekening 00011 sebesar Rp.100000. Jumlah saldo nasabah dengan no rekening 00011 adalah Rp.1500000. Transaksi sama-sama dieksekusi, transaksi penyetoran dieksekusi terlebih dahulu, kemudian transaksi penarikan tunai berikutnya.

Tanpa *Concurrency Control*

Transaksi penyetoran tunai lebih dulu selesai dengan keluaran saldo Rp.1700000 untuk no rekening 00011 dari saldo awal Rp.1500000 + penyetoran Rp.200000, kemudian transaksi penarikan tunai selesai dengan keluaran saldo terakhir Rp.1400000 untuk rekening 001 dari saldo awal Rp 1500000- penarikan Rp.100000.

Dengan demikian, tampak hilangnya data yang diubah oleh transaksi penyetoran tunai sebesar Rp.200000. Karena seharusnya data Saldo nasabah terakhir adalah Rp.1500000 + Rp.200000 – Rp.100000 = Rp. 1600000

Tabel berikut menunjukkan proses yang terjadi pada pengujian aplikasi tanpa *concurrency control* terhadap masalah hilangnya data yang diubah.

Waktu	Transaksi penyetoran	Transaksi penarikan	Total Saldo
t_1	SELECT @total_saldo = total_saldo FROM nasabah WHERE no_rekening=’00011’		1500000
t_2	Delay	SELECT @total_saldo = total_saldo FROM nasabah WHERE	1500000

		no_rekening=’00011’	
t_3	SET @total_saldo = @total_saldo + @nominal_kredit	Delay	1500000
t_4	UPDATE Nasabah SET total_saldo=@total_saldo WHERE no_rek=’00011’	SET @total_saldo = @total_saldo - @nominal_debet	1700000
t_5		UPDATE Nasabah SET total_saldo=@total_saldo WHERE no_rek=’00011’	1400000

Proses yang terjadi pada pengujian aplikasi tanpa concurrency control
- terhadap masalah hilangnya data yang diubah

Procedure Penarikan Tunai Tanpa *Concurrency* :

```
CREATE PROCEDURE SP_PENARIKAN_TUNAI_TC (
@NO_REKENING CHAR(5), @NIK CHAR(5),
@NOMINAL_DEBET INT)
AS
DECLARE @PESAN_SALAH VARCHAR(50), @TOTAL_SALDO
INT

BEGIN TRAN
SELECT @TOTAL_SALDO=TOTAL_SALDO  FROM NASABAH
WHERE NO_REKENING=@NO_REKENING

IF @NOMINAL_DEBET>=@TOTAL_SALDO
 BEGIN
    PRINT 'TRANSAKSI GAGAL'
    ROLLBACK  TRAN
 END
```

```
ELSE
 BEGIN
      WAITFOR DELAY '00:00:10'
      SET @TOTAL_SALDO=@TOTAL_SALDO –
      @NOMINAL_DEBET
      UPDATE NASABAH SET TOTAL_SALDO =
      @TOTAL_SALDO WHERE
      NO_REKENING = @NO_REKENING

      INSERT INTO TRANSAKSI(ID_JENIS_TRANSAKSI,
      NO_REKENING, NIK, WAKTU, NOMINAL_DEBET,
      NOMINAL_KREDIT)VALUES('02',@NO_REKENING,@NIK,
      GETDATE(),@NOMINAL_DEBET,0)
      COMMIT TRAN
END
```

Procedure Penyetoran Tunai Tanpa *Concurrency* :

```
CREATE PROCEDURE SP_PENYETORAN_TUNAI_TC(
@NO_REKENING CHAR(5),@NIK CHAR(5),
@NOMINAL_KREDIT INT)
AS
DECLARE  @TOTAL_SALDO  INT
BEGIN TRAN
SELECT @TOTAL_SALDO=TOTAL_SALDO FROM
NASABAH
WHERE NO_REKENING=@NO_REKENING
WAITFOR DELAY '00:00:10'
SET @TOTAL_SALDO = @TOTAL_SALDO +
@NOMINAL_KREDIT

UPDATE NASABAH SET TOTAL_SALDO=@TOTAL_SALDO
WHERE NO_REKENING=@NO_REKENING
INSERT INTO
TRANSAKSI(ID_JENIS_TRANSAKSI,NO_REKENING,NIK,WA
KTU,
```

```
NOMINAL_DEBET, NOMINAL_KREDIT) VALUES ('01',
@NO_REKENING, @NIK, GETDATE(), 0,
@NOMINAL_KREDIT)
COMMIT TRAN
```

Dengan *Concurrency Control*

Transaksi penyetoran tunai lebih dulu selesai dengan keluaran saldo Rp.1700000 untuk no rekening 00011 dari saldo awal Rp.1500000 + Rp.200000, kemudian transaksi penarikan tunai selesai dengan keluaran saldo terakhir Rp.1600000 untuk rekening 00011 dari Saldo Rp.1700000 – Rp.100000.

Dengan demikian,tampak bahwa transaksi tidak mengalami masalah hilangnya data yang diubah.

Tabel berikut menunjukkan proses yang terjadi pada pengujian aplikasi tanpa *concurrency control* terhadap masalah hilangnya data yang diubah.

Waktu	Transaksi penyetoran	Transaksi penarikan	Total Saldo
t_1	UPDATE nasabah SET total_saldo= total_saldo WHERE no_rekening= ‘00011’		1500000
t_2	SELECT @total_saldo = total_saldo FROM nasabah WHERE no_rekening=’00011’		1500000
t_3		SELECT @total_saldo = total_saldo FROM nasabah WHERE no_rekening=’00011’ (WAIT)	1500000
t_4	SET @total_saldo = @total_saldo +	(WAIT)	1500000

	@nominal_kredit		
t_5	UPDATE Nasabah SETtotal_saldo=@total_saldo WHERE no_rek=’00011’	(WAIT)	1700000
t_6		fetch cr_saldo_penarikan into @total_saldo	1700000
t_7		Delay	1700000
t_8		SET @total_saldo = @total_saldo - @nominal_debet	1700000
t_9		UPDATE Nasabah SETtotal_saldo=@total_saldo WHERE no_rek=’00011’	1600000

Proses yang terjadi pada pengujian aplikasi dengan concurrency control

- terhadap masalah hilangnya data yang diubah

Procedure Penarikan Tunai Dengan *Concurrency* :

```
CREATE PROCEDURE SP_PENARIKAN_TUNAI_DC(
@NO_REKENING CHAR(5),@NIK CHAR(5),
@NOMINAL_DEBET INT)
AS

DECLARE @TOTAL_SALDO  INT
DECLARE CR_SALDO_PENARIKAN CURSOR
FOR SELECT TOTAL_SALDO FROM NASABAH
WHERE NO_REKENING= @NO_REKENING
FOR UPDATE OF TOTAL_SALDO
BEGIN TRAN
OPEN CR_SALDO_PENARIKAN
FETCH CR_SALDO_PENARIKAN INTO @TOTAL_SALDO
```

```
IF @NOMINAL_DEBET >= @TOTAL_SALDO
  BEGIN
        PRINT 'TRANSAKSI GAGAL'
        ROLLBACK TRAN
  END
ELSE
  BEGIN
        WAITFOR DELAY '00:00:10'
        SET @TOTAL_SALDO = @TOTAL_SALDO –
@NOMINAL_DEBET

        UPDATE NASABAH SET TOTAL_SALDO =
        @TOTAL_SALDO  WHERE
        NO_REKENING=@NO_REKENING
        INSERT INTO TRANSAKSI(ID_JENIS_TRANSAKSI,
        NO_REKENING, NIK, WAKTU, NOMINAL_DEBET,
        NOMINAL_KREDIT)VALUES('02',@NO_REKENING,
        @NIK, GETDATE(), @NOMINAL_DEBET,0)
        COMMIT TRAN
 END
CLOSE CR_SALDO_PENARIKAN
DEALLOCATE CR_SALDO_PENARIKAN
```

Procedure Penyetoran Tunai Dengan *Concurrency* :

```
CREATE PROCEDURE SP_PENYETORAN_TUNAI_DC(
@NO_REKENING CHAR(5),@NIK CHAR(5),
@NOMINAL_KREDIT INT)
AS
DECLARE @TOTAL_SALDO  INT

BEGIN TRAN
UPDATE NASABAH SET TOTAL_SALDO=TOTAL_SALDO
WHERE NO_REKENING=@NO_REKENING

SELECT @TOTAL_SALDO=TOTAL_SALDO FROM NASABAH
```

```
WHERE NO_REKENING = @NO_REKENING

WAITFOR DELAY '00:00:10'
SET @TOTAL_SALDO = @TOTAL_SALDO +
@NOMINAL_KREDIT

UPDATE NASABAH SET TOTAL_SALDO=@TOTAL_SALDO
WHERE NO_REKENING=@NO_REKENING

INSERT INTO
TRANSAKSI(ID_JENIS_TRANSAKSI,NO_REKENING,NIK,WA
KTU,
NOMINAL_DEBET,NOMINAL_KREDIT)VALUES('01',
@NO_REKENING, @NIK, GETDATE(), 0,
@NOMINAL_KREDIT)
COMMIT TRAN
```

Masalah analisa yang tidak konsisten (*Inconsistent analiysis problem*)

Penghitungan jumlah total saldo seluruh nasabah memiliki keluaran berupa parameter OUT yaitu nilai_sum yang memuat jumlah total_saldo seluruh nasabah dengan cara menjumlahkannya satu persatu. No rekening 00011 memiliki saldo sebesar Rp.1000000 akan mentransfer saldonya sebesar Rp.200000 ke nasabah dengan no rekening 00012 yang memiliki saldo Rp.1000000, Kemudian transaksi sama-sama dieksekusi. Posisi saldo saat ini adalah sebagai berikut :

NO_REKENING	NAMA_NA...	ALAMAT	NO...	TA...	ID...	N...	KANTOR...	TOTAL_SALDO
00011	ANGRAINI	GEJAY...	0...	KT...	0...	...	CAB.G...	1000000
00012	JOKO K...	JL.JE...	0...	SIM C	0...	...	CAB.K...	1000000
00115	AYU FI...	KAUMA...	0...	KT...	0...	...	CAB.M...	1000000

Tanpa *Concurrency Control*

Transaksi transfer saldo mengeluarkan hasil untuk total saldo no rekening 00011= Rp.800000 dan no rekening 00012= Rp.1200000, transaksi total saldo seluruh nasabah selesai, tampil total saldo seluruh

nasabah dengan keluaran yang tidak sama antara jumlah total saldo seluruh nasabah yang sebenarnya Rp.3000000 dengan hasil jumlah total saldo dari prosedur Rp.3200000.

Dengan demikian, tampak masalah analisa yang tidak konsisten pada transaksi total saldo seluruh nasabah, antara jumlah total saldo pada tabel nasabah dengan variabel yang menyimpan data jumlah total saldo dalam prosedur.

Tabel berikut menunjukkan proses yang terjadi pada pengujian aplikasi tanpa *concurrency control* terhadap masalah analisa yang tidak konsisten.

Waktu	Transaksi transfer saldo	Transaksi total saldo	Total saldo nasabah	Var @total_saldo
t_1	Select @total_saldo=total_saldo from nasabah Where no_rekening= ‘00011’	Declare cursor_saldo cursor for select total_saldo from nasabah	1000000	
t_2	Delay	Fetch next from cursor_saldo into @total_saldo	1000000	0
t_3	Set @total_saldo=@total_saldo - @nominal_debet	Delay	1000000	0
t_4	Update nasabah set total_saldo=@total_saldo Where no_rekening=’00011’	Set Jum_tot_saldo= jum_tot_saldo + @tot_saldo	**800000**	**1000000**
t_5	Select @total_saldo = total_saldo from nasabah		1000000	1000000

	Where no_rekening =’00012’			
t_6	Set @total_saldo=@total_ saldo + @nominal_kredit		1000000	1000000
t_7	Update nasabah set total_saldo=@total_saldo Where no_rekening=’00012’	Fetch next from cursor_saldo into @total_saldo	**1200000**	1000000
		Set Jum_tot_saldo= jum_tot_saldo + @tot_saldo	1200000	**1200000**

Proses yang terjadi pada pengujian aplikasi tanpa concurrency control - terhadap masalah analisa yang tidak konsisten.

Procedure Transfer saldo Tanpa *Concurrency* :

```
CREATE PROCEDURE SP_TRANSFER_SALDO_TC(
@NO_REKENING_ASAL CHAR(5),@NIK
CHAR(5),@NOMINAL_DEBET INT,
@NO_REKENING_TUJUAN CHAR(5)
)
AS
DECLARE @TOTAL_SALDO INT,@NOMINAL_KREDIT INT
SET @NOMINAL_KREDIT=@NOMINAL_DEBET
BEGIN TRAN
SELECT @TOTAL_SALDO=TOTAL_SALDO FROM NASABAH
WHERE NO_REKENING=@NO_REKENING_ASAL

IF @NOMINAL_DEBET>=@TOTAL_SALDO
  BEGIN
    PRINT 'TRANSAKSI GAGAL'
    ROLLBACK TRAN
```

```
  END
ELSE
BEGIN

WAITFOR DELAY '00:00:10'
SET @TOTAL_SALDO=@TOTAL_SALDO -
@NOMINAL_DEBET
UPDATE NASABAH SET TOTAL_SALDO=@TOTAL_SALDO
WHERE NO_REKENING=@NO_REKENING_ASAL

SELECT @TOTAL_SALDO=TOTAL_SALDO FROM NASABAH
WHERE NO_REKENING=@NO_REKENING_TUJUAN
SET @TOTAL_SALDO=@TOTAL_SALDO +
@NOMINAL_KREDIT
UPDATE NASABAH SET TOTAL_SALDO=@TOTAL_SALDO
WHERE NO_REKENING=@NO_REKENING_TUJUAN

INSERT INTO
TRANSAKSI(ID_JENIS_TRANSAKSI,NO_REKENING,NIK,WA
KTU,
NOMINAL_DEBET,NOMINAL_KREDIT)VALUES('03',@NO_RE
KENING_ASAL,@NIK,GETDATE(), @NOMINAL_DEBET,0)

INSERT INTO
TRANSAKSI(ID_JENIS_TRANSAKSI,NO_REKENING,NIK,WA
KTU,
NOMINAL_DEBET,NOMINAL_KREDIT)VALUES('04',@NO_RE
KENING_TUJUAN, @NIK,GETDATE(),
0,@NOMINAL_KREDIT)
COMMIT TRAN
END
```

Procedure Mencari Jumlah Total Saldo Tanpa *Concurrency* :

```
CREATE PROCEDURE SP_JUM_TOT_SALDO_TC
```

```
AS
BEGIN
DECLARE CURSOR_SALDO CURSOR FOR SELECT
TOTAL_SALDO FROM NASABAH
DECLARE @JUMLAH_SEBENARNYA INT,@JUMLAH
INT,@TOTAL_SALDO INT

 SET @JUMLAH=0
 OPEN CURSOR_SALDO
 FETCH NEXT FROM CURSOR_SALDO INTO
@TOTAL_SALDO
 WAITFOR DELAY '00:00:10'

  WHILE @@FETCH_STATUS=0
    BEGIN
      FETCH NEXT FROM CURSOR_SALDO INTO
@TOTAL_SALDO
     --PRINT @TOTAL_SALDO
     SET @JUMLAH=@JUMLAH +@TOTAL_SALDO
          --PRINT @JUMLAH
    END

PRINT @JUMLAH
SELECT @JUMLAH_SEBENARNYA=SUM(TOTAL_SALDO)
FROM NASABAH
PRINT @JUMLAH_SEBENARNYA
CLOSE CURSOR_SALDO
DEALLOCATE CURSOR_SALDO
END
```

Dengan *Concurrency Control*

Transaksi transfer saldo mengeluarkan hasil untuk total saldo no rekening 00011= Rp.800000 dan no rekening 00012= Rp.1200000, transaksi total saldo seluruh nasabah selesai, tampil total saldo seluruh nasabah dengan keluaran yang tidak sama antara jumlah total saldo

seluruh nasabah yang sebenarnya Rp.3000000 dengan hasil jumlah total saldo dari prosedur Rp.3000000.

Tabel berikut menunjukkan proses yang terjadi pada pengujian aplikasi dengan *concurrency control* terhadap masalah analisa yang tidak konsisten.

Waktu	Transaksi transfer saldo	Transaksi total saldo	Total saldo nasabah	Var @total_saldo
	Set transaction isolation level serializable	Declare cursor_saldo cursor for select total_saldo from nasabah		
t_1	Select @total_saldo=total_saldo from nasabah Where no_rekening= ‘00011’	Set transaction isolation level serializable	1000000	0
t_2	Delay	Fetch next from cursor_saldo into @total_saldo (wait)		0
t_3	Set @total_saldo=@total_saldo - @nominal_debet	Delay	1000000	0
t_4	Update nasabah set total_saldo=@total_saldo Where no_rekening= ’00011’	(Wait)	**800000**	0
t_5	Select @total_saldo = total_saldo from	Set Jum_tot_saldo= jum_tot_saldo +	1000000	**800000**

	nasabah Where no_rekening =’00012’	@tot_saldo		
t_6	Set @total_saldo=@total_saldo + @nominal_kredit	(wait)	1000000	800000
t_7	Update nasabah set total_saldo=@total_saldo Where no_rekening=’00012’	Fetch next from cursor_saldo into @total_saldo	**1200000**	800000
		Set Jum_tot_saldo= jum_tot_saldo + @tot_saldo		**1200000**

Proses yang terjadi pada pengujian aplikasi dengan concurrency control

- terhadap masalah analisa yang tidak konsisten.

Procedure Transfer saldo Dengan *Concurrency* :

```
CREATE PROCEDURE SP_TRANSFER_SALDO_DC(
@NO_REKENING_ASAL CHAR(5),@NIK
CHAR(5),@NOMINAL_DEBET INT,
@NO_REKENING_TUJUAN CHAR(5)
)
AS
DECLARE @TOTAL_SALDO INT,@NOMINAL_KREDIT INT
SET @NOMINAL_KREDIT=@NOMINAL_DEBET
BEGIN TRAN
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE

SELECT @TOTAL_SALDO=TOTAL_SALDO FROM NASABAH
WHERE NO_REKENING=@NO_REKENING_ASAL
```

```
IF @NOMINAL_DEBET>=@TOTAL_SALDO
  BEGIN
    PRINT 'TRANSAKSI GAGAL'
    ROLLBACK TRAN
  END
ELSE
BEGIN

WAITFOR DELAY '00:00:10'
SET @TOTAL_SALDO=@TOTAL_SALDO -
@NOMINAL_DEBET
UPDATE NASABAH SET TOTAL_SALDO=@TOTAL_SALDO
WHERE NO_REKENING=@NO_REKENING_ASAL

SELECT @TOTAL_SALDO=TOTAL_SALDO FROM NASABAH
WHERE NO_REKENING=@NO_REKENING_TUJUAN
SET @TOTAL_SALDO=@TOTAL_SALDO +
@NOMINAL_KREDIT
UPDATE NASABAH SET TOTAL_SALDO=@TOTAL_SALDO
WHERE NO_REKENING=@NO_REKENING_TUJUAN

INSERT INTO
TRANSAKSI(ID_JENIS_TRANSAKSI,NO_REKENING,NIK,WA
KTU,
NOMINAL_DEBET,NOMINAL_KREDIT)VALUES('03',@NO_RE
KENING_ASAL,@NIK,GETDATE(),
@NOMINAL_DEBET,0)
INSERT INTO
TRANSAKSI(ID_JENIS_TRANSAKSI,NO_REKENING,NIK,WA
KTU,
NOMINAL_DEBET,NOMINAL_KREDIT)VALUES('04',@NO_RE
KENING_TUJUAN,@NIK,GETDATE(),
0,@NOMINAL_KREDIT)
COMMIT TRAN
END
```

Procedure Mencari Jumlah Total Saldo Dengan *Concurrency* :

```
CREATE  PROCEDURE SP_JUM_TOT_SALDO_DC
AS
BEGIN TRAN
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE
DECLARE CURSOR_SALDO CURSOR FOR SELECT
TOTAL_SALDO FROM NASABAH
DECLARE @JUMLAH_SEBENARNYA INT,@JUMLAH
INT,@TOTAL_SALDO INT

 SET @JUMLAH=0
 OPEN CURSOR_SALDO
 FETCH NEXT FROM CURSOR_SALDO INTO
@TOTAL_SALDO
 WAITFOR DELAY '00:00:10'

  WHILE @@FETCH_STATUS=0
    BEGIN
      FETCH NEXT FROM CURSOR_SALDO INTO
@TOTAL_SALDO
     --PRINT @TOTAL_SALDO
          if @@error<>0
          rollback tran
     SET @JUMLAH= @JUMLAH + @TOTAL_SALDO
          --PRINT @JUMLAH
    END
PRINT @JUMLAH
SELECT @JUMLAH_SEBENARNYA=SUM(TOTAL_SALDO)
FROM NASABAH
PRINT @JUMLAH_SEBENARNYA
CLOSE CURSOR_SALDO
DEALLOCATE CURSOR_SALDO
```

3. Penutup

Aplikasi dapat mencegah masalah hilangnya data yang diubah (*Lost of Update*) dan juga masalah analisa yang tidak konsisten (*the inconsistent analysis problem*)

Concurrency control dalam SQL Server dapat dicapai dengan beberapa cara, seperti menggunakan :

Statement SELECT .. FOR UPDATE

Isolation level SERIALIZABLE

Statement UPDATE pada diri sendiri sebelum melakukan proses baca dengan maksud mengubah data yang dibaca.

Daftar Pustaka

Connolly,T.,Begg, C.,*DATABASE SYSTEM A Practical Approach To Design*, Implementation And Management, Addison Wesley,2002.

Ir.Inge Martina, 36 Jam Belajar Komputer Microsoft SQL Server 2000, Elexmedia Komputindo,2003.

Dusan Petkovic, *SQL SERVER 7 A BEGINNER'S GUIDE*, Osborne/ Mcgraw-Hill,1999.