SISTEM MONITORING PELANGGAN PASCABAYAR DAN PRABAYAR TBT MENERAPKAN MANAJEMEN TRANSAKSI MENGGUNAKAN METODE TWO PHASE LOCKING

Ni Putu Novita Puspa Dewi[1*], JB Budi Darmawan[1]
Program Studi Teknik Informatika, Fakultas Sains dan Teknologi, Universitas Sanata Dharma
Kampus III, Paingan, Maguwoharjo, Depok, Yogyakarta
Telp: 0274-883037
[*]Email: niputunovitapuspadewi@gmail.com

Abstrak

Sebagai salah satu upaya pengendalian susut non teknis di PT. PLN (Persero) Area Kuala Kapuas, bagian Transaksi Energi melakukan monitoring pelanggan Pascabayar dan Prabayar secara rutin. Monitoring yang dilakukan selama ini dilakukan secara manual dan tidak ada filterisasi untuk data yang sudah diperiksa sehingga pemeriksaan kadang dilakukan berulang kali. PT. PLN (Persero) Area Kuala Kapuas terdiri dari 6 rayon, dimana tiap rayon juga bertanggung jawab untuk melakukan monitoring pelanggan di wilayahnya. Mengingat bahwa petugas yang menggunakan sistem monitoring bukan hanya petugas lapangan melainkan juga petugas yang berada di kantor maka sistem ini menerapkan manajemen transaksi yang melayani multiuser. Sistem yang multiuser ini mengharuskan adanya sebuah penanganan transaksi ketika satu atau beberapa user akan write dan atau read data secara bersamaan. Hal ini dilakukan untuk menghindari masalah proses konkuren seperti diantaranya The Lost Update Problem, The Uncommited Dependency (Dirty Read) Problemdan The Inconsistent Analysis Problem. Sehingga diperlukan manajemen transaksi untuk mengatur kelancaran dari tiap action yang dilakukan oleh user sistem terhadap data. Metode two phase locking yang digunakan akan memberikan lock bagi tiap transaksi yang akan mengakses data, baik read atau write data. Sehingga tiap transaksi akan dibuat menunggu sampe lock dilepaskan untuk dapat mengubah data. Pada penelitian kali ini, Penulis akan mencoba protokol yang sama pada basis data Oracle, dimana Oracle juga mendukung kontrol untuk concurrency dengan teknik 'FOR UPDATE'. Berdasarkan ujicoba pada permasalahan the lost of update menggunakan teknik FOR UPDATE pada basis data Oracle terbukti dapat menjamin konsistensi data hasil monitoring pelanggan.

***Kata Kunci**: sistem monitoring PLN, two phase locking, manajemen transaksi*

1. PENDAHULUAN

Pengendalian susut merupakan salah satu upaya penekanan jumlah susut energi listrik yang disalurkan ke pelanggan. Pengendalian susut energi listrik dibedakan menjadi 2 yaitu secara teknis dan nonteknis. Pengendalian susut teknis berkaitan dengan usaha-usaha pemeliharaan dan pengawasan di pembangkit listrik, sedangkan pengendalian susut non teknis berkaitan dengan administrasi, manajemen kWh Meter, dan penanganan pencurian tenaga listrik.Untuk menghindari kerugian akibat pelanggaran diatas maka perlu dilakukan monitoring untuk pelanggan pasca bayar dan prabayar. Monitoring pemakaian tenaga listrik oleh pelanggan merupakan salah satu bagian dari usaha pengendalian susut non teknis.

Pelanggan yang perlu dimonitor adalah pelanggan pasca bayar kWh 0 dan kWh Maks; dan pelanggan prabayar yang tidak beli token lebih dari 3 bulan. Data Pelanggan Pascabayar dan Prabayar yang harus dimonitor terbilang sangat banyak. Berdasarkan data pada gambar 1 terdapat sekitar 3.803 pelanggan pascabayar di area Kuala Kapuas yang harus dimonitor dalam satu bulan (sumber: data DPM PT.PLN (Persero) Area Kuala Kapuas bulan Juni 2014) dan terdapat sekitar 3.056 pelanggan prabayar tidak beli token yang harus dimonitor (sumber: data DPM PT.PLN (Persero) Area Kuala Kapuas bulan Juni 2015).

PT. PLN (Persero) Area Kuala Kapuas terdiri dari 6 rayon, dimana tiap rayon juga bertanggung jawab untuk melakukan monitoring pelanggan di wilayahnya. Mengingat bahwa petugas yang menggunakan sistem monitoring bukan hanya petugas lapangan melainkan juga petugas yang berada di kantor maka sistem ini menerapkan manajemen transaksi yang melayani *multiuser*. Penerapan sistem,monitoring yang mendukung *multiuser* adalah bagaimana membuat

data monitoring yang disimpan selalu dalam keadaan konsisten walaupun diakses dan di-update secara bersamaan oleh banyak *user*. Permasalahan yang mungkin akan ditemukan saat dua atau lebih transaksi berjalan bersamaan adalah *the lost update problem, the uncommitted dependency problem dan the inconsistent analysis problem* (Connoly *and* Beg, 2002).

Ketika terdapat 2 *user* yang ingin mengupload data hasil monitoring pada pelanggan yang sama secara bersamaan atau ketika terdapat *user* di kantor yang melakukan pengecekkan (proses *approval* monitoring pada data pelanggan yang sama), maka transaksi antara 2 atau lebih *user* secara bersamaan terhadap data yang sama membutuhkan manajemen transaksi dengan *concurrency control* untuk menjamin konsitensi data yang sudah disebutkan sebelumnya (Connoly *and* Beg, 2002). Penerapan manajemen transaksi terhadap sistem monitoring ini salah satunya dengan menggunakan metode 2PL (2 *Phase Locking*).

Sebelumnya terdapat penelitian yang telah dipublikasikan mengenai *concurrency control* dengan menggunakan 2PL pada basis data MySQL untuk menangani masalah serupa. Penelitian tersebut menyebutkan bahwa MySQL mempunyai kemampuan mendukung transaksi dengan metode 2PL yang dapat menjamin konsistensi data (Darmawan, 2007). Pada penelitian kali ini, Penulis akan mencoba protokol yang sama pada basis data Oracle, dimana Oracle juga mendukung kontrol untuk *concurrency* dengan teknik 'FOR UPDATE'**.** Perintah FOR UPDATE yang digunakan untuk melakukan penguncian agar menjamin konsistensi (Oracle, 2004). Penelitian ini akan berfokus pada penerapan protokol 2PL di tiap proses bisnis monitoring untuk melihat apakah tiga permasalahan diatas ditemukan pada proses monitoring dan menanganinya dengan protokol ini.

2. METODOLOGI

Penulis menggunakan metode pengembangan sistem informasi menggunakan metode FAST (*Framework for the Application of Systems Thinking*) menurut Whitten (2001) yang fasenya meliputi:

1. Definisi lingkup masalah.
   Pada fase ini dilakukan definisi ruang lingkup masalah dengan melakukan pengamatan dan wawancara kepada pihak Transaksi Energi mengenai pengelolaan data-data pelanggan monitoring dan permasalahan yang dihadapi dengan memahami proses bisnis dari monitoring yang dilakukan.
2. Analisa masalah
   Pada fase ini dilakukan analisa masalah yang ada pada sistem monitoring untuk kemudian dapat mendefinisikan sebuah tujuan perbaikan.
3. Analisa Kebutuhan
   Pada fase ini dilakukan analisa kebutuhan-kebutuhan pengguna, untuk mencari tahu apa yang mereka perlukan atau inginkan dari sistem baru. Dalam hal ini berkaitan dengan fokus penelitian, Penulis akan mengamati proses bisnis mana yang memerlukan teknik 2PL untuk mencegah ketidakkonsistenan data.
4. Desain logikal
   Pada fase ini akan dilakukan desain secara logical. Pada fase ini Penulis akan merancang basis data beserta dengan teknik kontrol untuk masalah *concurrency,* dalam hal ini merancang *store procedure* yang menggunakan teknik *locking* dengan 2PL *(*Connolly *and* Beg*, 2002)*.
5. Desain fisikal
   Pada fase ini hal yang dilakukan adalah membangun sistem secara fisik berdasarkan teknologi yang digunakan, desain arsitektur dan desain antarmuka pengguna (*user interface*) untuk menguji implementasi dari 2PL pada proses monitoring.
6. Konstruksi dan Pengujian
   Pada fase ini dilakukan untuk menguji apakah implementasi teknik 2PL berhasil mengatasi masalah *concurrency* pada proses bisnis yang ditangani oleh sistem monitoring. Mekanisme pengujian yang dilakukan akan menggunakan protokol 2PL dan tanpa 2PL, dengan tujuan untuk melihat perbedaan hasil pengujian dan menemukan masalah yang diakibatkan oleh transaksi yang dilakukan bersamaan (Connolly *and* Beg,2002), seperti:

a. *The Lost Update Problem*, merupakan kejadian dimana data transaksi yang telah di*update* dibaca oleh transaksi yang lain kemudian di *update* lagi.
b. *The Uncommited Dependency (Dirty Read) Problem*, merupakan kejadian dimana data transaksi yang dilakukan dibaca oleh transaksi yang lain, lalu dibatalkan tanpa adanya penyimpanan terlebih dahulu oleh transaksi yang pertama sehingga menyebabkan data yang tidak benar.
c. *The Incosistent Analysis Problem*, masalah ini timbul karena data diakses oleh 2 transaksi yang bersamaan, transaksi yang pertama melakukan perubahan dan transaksi yang kedua melakukan analisis sehingga data yang diperoleh menjadi tidak konsisten.

Pada sistem moitoring ini, pengujian untuk membuktikan ketiga masalah diatas hanya dilakukan terhadap *the lost of update problem.* Pembuktian terjadinya masalah *the uncommited dependency problem* tidak dilakukan karena *default* dari Oracle yang tidak *autocommit* pada tiap baris. Pada Oracle terdapat mekanisme proses *write data* secara permanen pada tabel yang tidak bisa dilakukan sebelum proses *commit,* sehingga keadaan ini tidak dapat membuktikan terjadinya permasalahan *the uncommited dependency problem.* Pembuktian terjadinya *The Incosistent Analysis Problem* tidak dilakukan karena tidak ditemui permasalah yang berkaitan dengan *The Incosistent Analysis Problem* pada proses bisnis di sistem monitoring ini.

Simulasi ini dilakukan untuk melihat apa yang terjadi pada saat *upload* data, ubah data, *approve* maupun proses pembatalan yang berlangsung secara bersamaan yang akan ditangani dengan dan tanpa 2PL. Kedua transaksi yang mengakses *row* yang sama pada tabel akan diberikan *delay* untuk membuat agar keduanya saling bertabrakan. Transaksi yang berjalan lebih dulu otomatis akan menjalankan teknik *lock* untuk mengunci *row* yang diaksesnya agar transaksi lain tidak dapat melakukan *write* data pada *row* tersebut. Uji coba ini dilakukan dengan menggunakan 2 *browser browser* yang berbeda.

3. HASIL DAN PEMBAHASAN

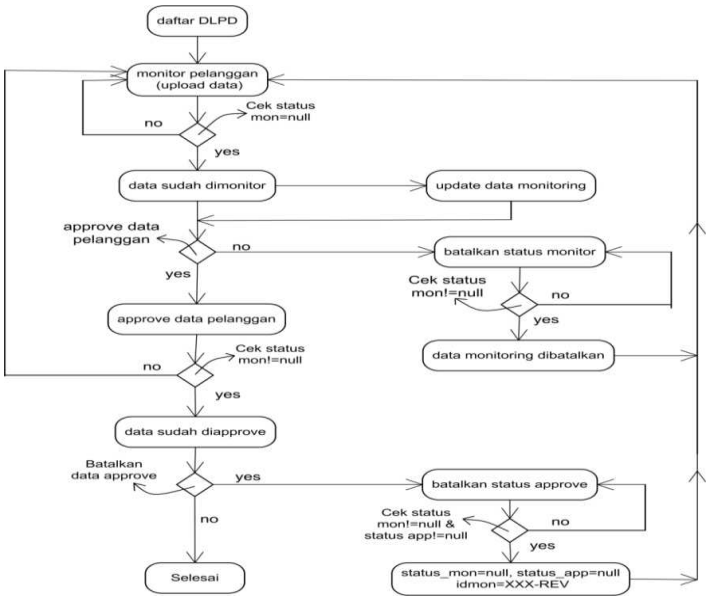
Proses monitoring pelanggan dengan menggunakan sistem monitoring diawali dengan proses upload data pelanggan oleh petugas yang melakukan pengecekkan di lapangan. Setelah itu data pelanggan akan mendapatkan status sudah monitoring. Data tersebut masih bisa diganti (di*update*) untuk kemudian akan ditindak-lanjuti staf di kantor yang akan mengecek keabsahan data tersebut. Jika data lolos pengecekkan, maka data tersebut akan di*approve* oleh staf di kantor dan mendapatkan status sudah *approve,* proses monitoring pada data pelanggan tersebut selesai. Namun, jika tidak lolos maka akan diberikan status ulang monitor (status belum monitor). Data yang sudah di*approve* masih bisa dibatalkan status *approve*-nya, jika dibatalkan maka data akan kembali ke status belum monitor. Alur proses monitoring pelanggan dapat dilihat pada gambar 1. Pada tulisan ini akan dibahas 1 contoh saja untuk mewakili semua proses bisnis yang ada pada sistem.Bagian selanjutnya adalah melakukan uji coba dengan dan tanpa 2PL untuk 2 transaksi yang berlangsung bersamaan untuk permasalah.

3.1. Pengujian Terhadap Masalah *Lost of Update*

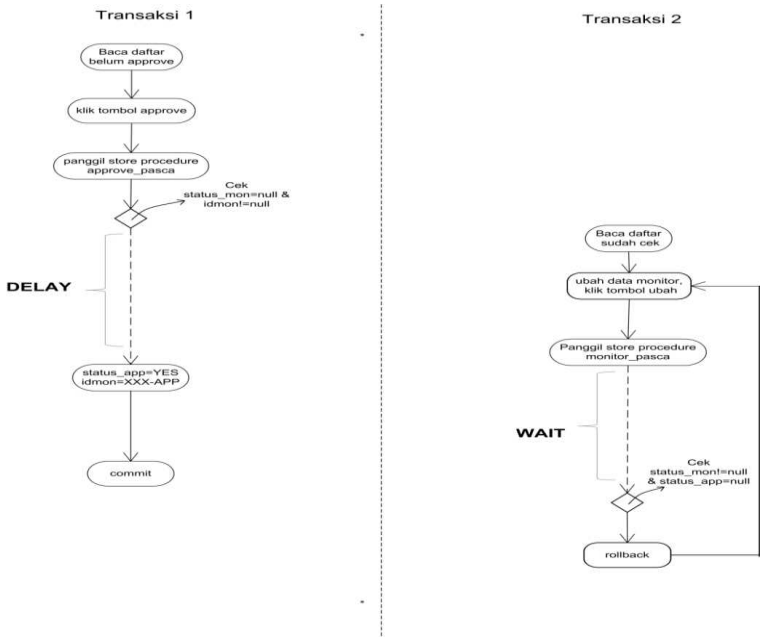
Untuk menguji manajemen transaksi yang diimplementasikan pada mesin di sistem monitoring, maka Penulis melakukan uji coba dengan menjalankan simulasi dimana dilakukan 2 transaksi yang menerapkan metode 2PL (*Two Phase Locking*). Simulasi ini dilakukan dengan memanggil stored procedure tanpa 2PL untuk membuktikan permasalahan *lost of update* dan melakukan simulasi yang memanggil *stored procedure* dengan 2PL untuk menjawab permasalahan *lost of update* yang terjadi pada saat upload data, ubah data, *approve* maupun proses pembatalan yang berlangsung bersamaan. Kedua transaksi yang mengakses *row* yang sama pada tabel akan diberikan *delay* untuk membuat agar keduanya saling bertabrakan. Transaksi yang berjalan lebih dulu otomatis akan menjalakan teknik *lock* untuk menguci *row* yang diaksesnya agar transaksi lain tidak dapat melakukan *write* data pada *row* tersebut. Untuk diagram simulasi percobaan dapat dilihat pada gambar 2.

3.1.1 *Approve* dan Ubah Data Monitoring Pelanggan kWh 0 dengan 2PL

Berikut ini akan dijelaskan simulasi pengujian proses *approve* data pelanggan kwh 0 (transaksi 1) yang pada saat bersamaan ada transaksi 2 yang akan melakukan ubah data monitoring pelanggan tersebut. Transaksi 1 dijalankan oleh admin area dan transaksi 2 dijalankan oleh admin rayon. Aturan untuk mendapatkan IDMON saat mengubah data monitoring adalah belum memiliki status *approve* artinya kolom status_app di tabel DLPD_PASCBAYAR bernilai *null*. Ketika transaksi lain (transaksi 1) yang mendahului transaksi ubah data ini melakukan *locking* terhadap *row* tersebut, maka transaksi ubah data (transaksi 2) tidak dapat melakukan ubah data karena status *approve* yang sekarang sudah bernilai 'YES'. Alur dari simulasi ini secara singkat ditunjukan pada gambar 2.



Gambar 1. **Alur proses monitoring pelanggan**



Gambar 2. **Diagram simulasi Approve dan Ubah Data Monitoring Pelanggan kWh0**

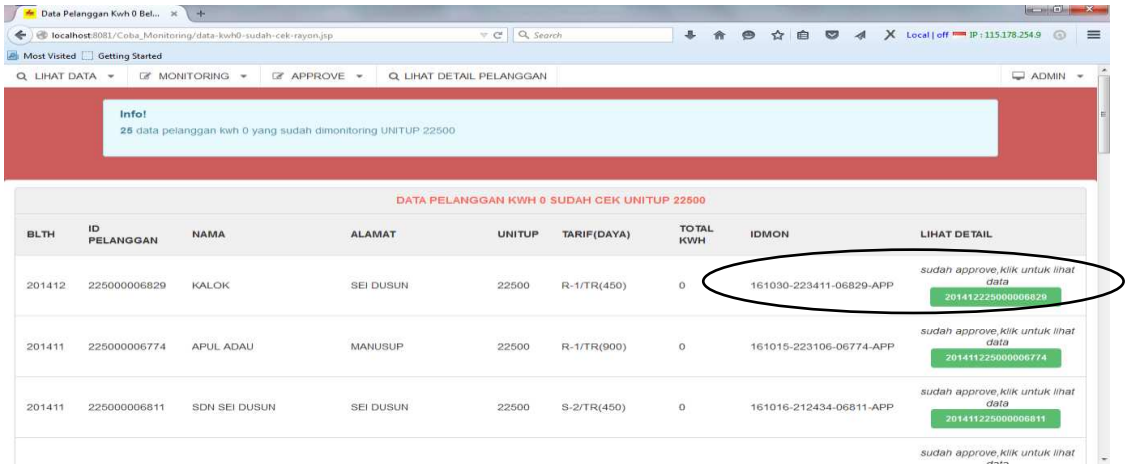
Pada simulasi ini *stored procedure* untuk *approve* data akan diberi *delay* untuk menunggu agar transaksi 2 berjalan bersamaan dengan transaksi 1. Pemanggilan *stored procedure* ini

dimaksudkan untuk mencegah masalah *lost of update* dimana data yang akan di*approve* akan berubah ketika proses ubah data yang berjalan setelahnya (transaksi 2) telah *commit* terlebih dahulu sebelum proses *approve* oleh transaksi 1 selesai. Pemberian *delay* ini dilakukan dengan memberi perintah DBMS_LOCK.SLEEP (*number in seconds*) setelah perintah *locking*. DMBS_LOCK.SLEEP ini akan membuat *delay* selama waktu yang ditentukan. Pada gambar 4 adalah potongan *stored procedure approve_pascabayar* dengan menggunakan perintah DBMS_LOCK.SLEEP dan implementasi dari teknik FOR UPDATE untuk melakukan *locking* dengan memberikan *exlusive lock* pada transaksi 1.

Hasil dari simulasi ini menunjukkan riwayat data monitoring pelanggan dalam keadaan konsisten (gambar 5), karena transaksi 1 berhasil *approve* dan transaksi 2 tidak dapat melakukan ubah data monitoring yang sudah di*approve* oleh transaksi 1. Pada saat transaksi 2 gagal melakukan ubah data dikarenakan data telah dalam keadaan *approve* oleh transaksi 1, maka transaksi 2 akan melakukan mekanisme *rollback*, sebab transaksi 2 tidak berhasil mendapat IDMON (mekanisme pemerolehan IDMON dilakukan oleh *function* tambah_idmon pada gambar 6). *Stored procedure* lengkap ditunjukkan pada gambar 7.

```
select idmon, status_app into v_idmon, v_status_mon from dlpd_pascabayar
where idpel=v_idpel and blth=v_blth
for update of idpel,blth;
v_idmon:=function_tambah_idapp_pasca(v_idpel,v_blth);
p_idmon:=v_idmon;
dbms_lock.sleep(20);
```

Gambar 4. Prosedur *approve*_pascabayar dengan DBMS_LOCK.SLEEP selama 20 detik



Gambar 5. Data pelanggan dengan nama 'Kalok' tetap dalam kondisi sudah *approve* dengan nomor referensi IDMON yang sama ketika *approve*

```
IF (v_status_mon is null) and (v_status_app is null)
then v_idmon:=v_date||'-'||v_idmon||'-'||'MON';
--update prabayar set id_mon=v_idmon, status_mon='YES' where idpel=v_idpel and blth=v_blth;
elsif (v_status_mon is not null) and (v_status_app is null)
then v_idmon:=v_date||'-'||v_idmon||'-'||'REV';
--update prabayar set id_mon=v_idmon where idpel=v_idpel and blth=v_blth;
else
v_idmon:=null;
END IF;
end if;
```

Gambar 6. Kutipan fungsi tambah_idmon_pasca jika v_status_app tidak *null* maka idmon = *null*

3.1.2 *Approve* dan Ubah Data Monitoring Pelanggan kWh 0 tanpa 2PL

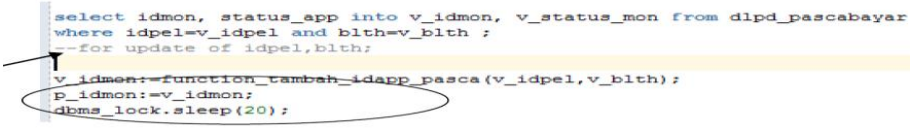
Jika sebelumnya telah dilakukan simulasi untuk menangani masalah *lost of update* dengan menggunakan *method 2 phase locking*, maka pada simulasi ini Penulis ini menunjukan kasus transaksi 1 *approve* data A yang akan disela dengan transaksi 2 ubah data A menjadi B. Tanpa adanya 2PL maka data yang diapprove oleh transaksi 1 bukan lagi data A melainkan data B. Yang dimaksudkan disini adalah id monitoring (idmon) sebagai id untuk referensi data sebelumnya yang berubah ketika approve disela dengan ubah data. Ini akan menyebabkan data menjadi tidak konsisten.

```
create or replace procedure approve_pascabayar
...
is
...

begin

select idmon, status_app into v_idmon, v_status_mon from
dlpd_pascabayar
where idpel=v_idpel and blth=v_blth
for update of idpel,blth;
v_idmon:=function_tambah_idapp_pasca(v_idpel,v_blth);
p_idmon:=v_idmon;
dbms_lock.sleep(20);

select idmon into v_versi_sebelum from dlpd_pascabayar
where idpel=v_idpel and blth=v_blth;

IF v_idmon is not null then
    select
    keadaan_mcb,ket,tgl_mon,verifikasi,koordinat,gambar
    into
    v_mcb,v_ket,v_tgl_mon,v_verifikasi,v_koordinat,v_gam
    bar
    from record_dlpd_pascabayar where
    idmon=v_versi_sebelum;
    update dlpd_pascabayar set status_app='YES',
    idmon=v_idmon where idpel=v_idpel and blth=v_blth;
if v_versi_sebelum is not null
then
insert into record_dlpd_pascabayar
(idpel,blth,idmon,koordinat,tgl_mon,verifikasi,ket,keadaan_
mcb,versi_sebelum,user_id,gambar,tgl_app)
values
(v_idpel,v_blth,v_idmon,v_koordinat,v_tgl_mon,v_verifikasi,
v_ket,v_mcb,v_versi_sebelum,v_user,v_gambar,to_char(S
YSDATE,'DD-MON-YY'));

else
  raise ex_no_data_found;
  end if;
else
  raise ex_no_data_found;
  end IF;

commit;
  p_status:=1;

EXCEPTION
WHEN ex_no_data_found THEN
rollback;
p_status:=0;
WHEN OTHERS THEN
rollback;
p_status:=0;

end approve_pascabayar;
```

Gambar 7. Isi *stored procedure* approve_pascabayar

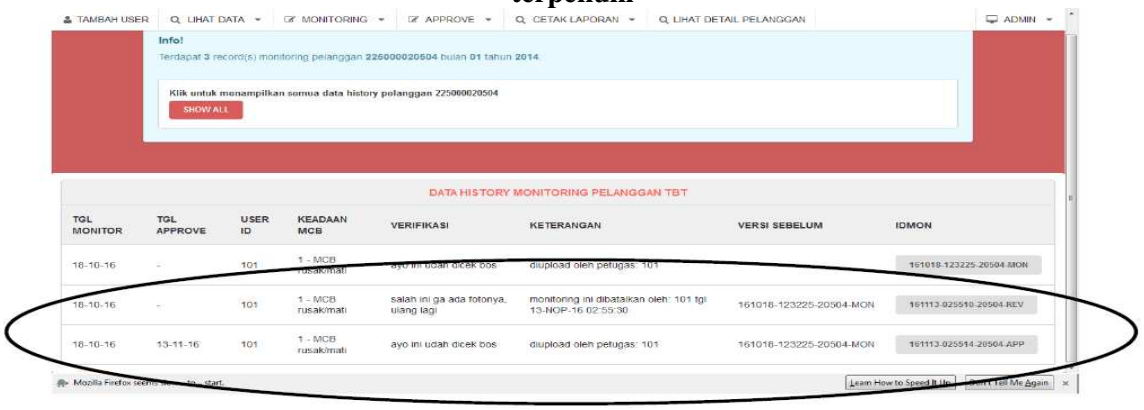
Data yang akan di*approve* adalah data dengan idpel 225000010899 dengan blth 201401. Transaksi 1 akan mengakses halaman kwh-0-detail-approve sedangkan transaksi 2 akan mengakses halaman kwh0-sudah-cek. Untuk membuat agar kedua transaksi ini bertabrakan, maka sama seperti ujicoba sebelumnya, *delay* selama 20 detik akan dibuat untuk transaksi 1 sehingga transaksi 2 dapat menyela. Pada *stored procedure* approve_pascabayar klausa FOR UPDATE OF akan dihilangkan agar tidak ada proses *locking* yang terjadi (gambar 8).

Karena terjadi delay, transaksi 1 tidak dapat langsung mengubah status monitor menjadi *null*. Pada saat yang bersamaan transaksi 2 berhasil melakukan *approve* sehingga membuat status *approve* bernilai YES. Setelah *delay* berakhir, data tetap dapat dibatalkan. Berdasarkan aturan monitoring, data pelanggan tidak dapat di*approve* jika status monitornya bernilai *null* maupun sebaliknya. Disinilah terjadinya data yang tidak konsisten yang ditunjukan pada gambar 9. Perhatikan data riwayat monitoring pelanggan pada gambar 10 dan 11, terjadi ketidakkonsistenan akibat transaksi 2 yang berhasil *commit* sebelum transaksi 1.

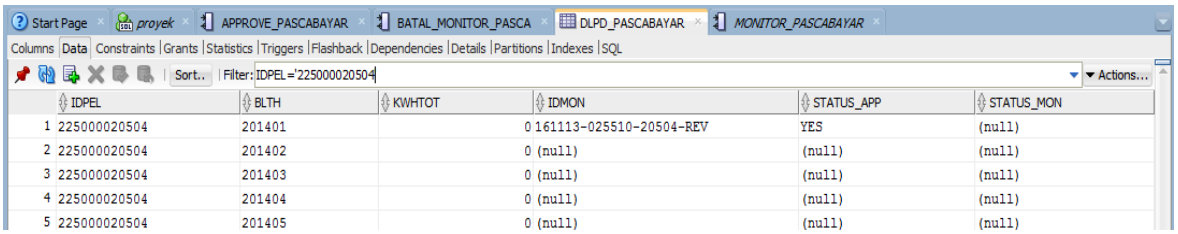


Gambar 8. Klausa FOR UPDATE OF dinonaktifkan,delay diset 20 detik.

Gambar 9. Pada daftar pelanggan sudah *approve*, data pelanggan memiliki status *approve* dengan tanda silang, ini terjadi karena syarat status monitor tidak sama dengan *null* tidak terpenuhi



Gambar 10. Pada data *history* monitoring pelanggan tersebut, referensi idmon versi sebelum menjadi tidak konsisten, ada 2 *record* yang memiliki versi sebelum yang sama.



Gambar 11. Status monitoring pelanggan dengan idpel 225000020504 untuk blth 201401 adalah *null* sedangkan status *approve* adalah YES.

Tabel 2. Proses yang terjadi saat *approve* data pelanggan dan ubah data pelanggan (subbab 3.1.1)

Waktu	Transaksi 1	Transaksi 2	Status *approve*	Idmon
t1	select idmon, status_app into v_idmon, v_status_mon from dlpd_pascabayar where idpel=v_idpel and blth=v_blth for *update* of idpel,blth; v_versi_sebelum:=v_idmon;		*null*	161020-154938-06829-REV
t2	v_idmon:=function_tambah_idapp_pasca(v_idpel,v_blth); p_idmon:=v_idmon;	select idmon, status_mon into v_idmon, v_status_mon from dlpd_pascabayar where idpel=v_idpel and blth=v_blth for *update* of idpel,blth;	*null*	161020-154938-06829-REV

t3	IF v_idmon is not null then
	
	select keadaan_mcb,ket,tgl_mon,verifikasi,koordinat,gambar into v_mcb,v_ket,v_tgl_mon,v_verifikasi,v_koordinat,v_gambar from record_dlpd_pascabayar where idmon=v_versi_sebelum;
	
	update dlpd_pascabayar set **status_app='YES', idmon=v_idmon** where idpel=v_idpel and blth=v_blth;
		WAIT	YES	161030-223411-06829-APP

t4	if v_versi_sebelum is not null then	WAIT	YES	161030-223411-06829-APP
	insert into record_dlpd_pascabayar (idpel,blth,idmon,koordinat,tgl_mon,verifikasi,ket,keadaan_mcb,versi_sebelum,user_id,gambar,tgl_app) values (v_idpel,v_blth,v_idmon,v_koordinat,v_tgl_mon,v_verifikasi,v_ket,v_mcb,v_versi_sebelum,v_user,v_gambar,to_char(SYSDATE,'DD-MON-YY'));			
t5	commit; p_status:=1;	v_idmon:=function_tambah_idmon_pasca(v_idpel,v_blth); p_idmon:=v_idmon;	YES	161030-223411-06829-APP
t6		else raise ex_no_data_found;	YES	161030-223411-06829-APP
t7		rollback; p_status:=0;	YES	161030-223411-06829-APP

Tabel 5. 3 *Approve* data pelanggan dan ubah data pelanggan tanpa protokol 2PL (subbab 3.1.2)

Waktu	Transaksi 1	Transaksi 2	Status monitor	Status *approve*	Idmon	Versi_sebelum
t1	select idmon, status_app into v_idmon, v_status_mon from dlpd_pascabayar where idpel=v_idpel and blth=v_blth;		YES	*null*	161029-124151-10899-MON	

t2	v_idmon:=function_tambah_idapp_pasca(v_idpel,v_blth);
	p_idmon:=v_idmon;
		select idmon, status_mon into v_idmon, v_status_mon from dlpd_pascabayar where idpel=v_idpel and blth=v_blth;
		
		v_idmon:=function_tambah_idmon_pasca(v_idpel,v_blth);
		p_idmon:=v_idmon;
			YES	*null*	161029-124151-10899-MON	

t3	delay	IF v_idmon is not null then
		select idmon into v_versi_sebelum from dlpd_pascabayar where idpel=v_idpel and blth=v_blth;
		p_versisebelum:=v_versi_sebelum;
		update dlpd_pascabayar set status_mon='YES', idmon=v_idmon where idpel=v_idpel and blth=v_blth;
		
		if v_versi_sebelum is not null then
		insert into record_dlpd_pascabayar (idpel,blth,idmon,koordinat,tgl_mon,ve
			YES	*null*	161113-014129-10899-REV	161029-124151-10899-MON

		rifikasi,ket,keadaan_mcb,versi_sebelum,user_id) values (v_idpel,v_blth,v_idmon,v_koordinat,to_char(v_date,'DD-MON-YY'),v_verifikasi,v_ket,v_mcb,v_versi_sebelum,v_user);				
t4	delay	commit	YES	*null*	161113-014129-10899-REV	161029-124151-10899-MON

t5	select idmon into v_versi_sebelum from dlpd_pascabayar where idpel=v_idpel and blth=v_blth;
	IF v_idmon is not null then
	
	select keadaan_mcb,ket,tgl_mon,verifikasi,koordinat,gambar into v_mcb,v_ket,v_tgl_mon,v_verifikasi,v_koordinat,v_gambar from record_dlpd_pascabayar where idmon=v_versi_sebelum;
			YES	*null*	161113-014129-10899-REV	161029-124151-10899-MON

t6	*update* dlpd_pascabayar set **status_app='YES', idmon=v_idmon** where idpel=v_idpel and blth=v_blth;
	if v_versi_sebelum is not null then
	insert into record_dlpd_pascabayar (idpel,blth,idmon,koordinat,tgl_mon,verifikasi,ket,keadaan_mcb,versi_sebel
			YES	YES	161113-014126-10899-APP	161113-014129-10899-REV

	um,user_id) values (v_idpel,v_blth,v_idmon,v_koordinat,to_char(v_date,'DD-MON-YY'),v_verifikasi,v_ket,v_mcb,v_versi_sebelum,v_user);					
t7	commit		YES	YES	161113-014126-10899-APP	161113-014129-10899-REV

Setelah dilakukan pengujian terhadap transaksi 1 dan transaksi 2 dengan mengimplementasikan 2PL dan tanpa 2PL, terlihat perbedaan data *history* monitoring pelanggan dengan dan tanpa 2PL. Hal ini dapat membuktikan bahwa terjadi permasalahan ketidakkonsistenan data akibat terjadi *the lost of update problem* pada saat kedua transaksi bertabrakan. Permasalahan ini kemudian dapat ditangani dengan metode 2PL yang diimplementasikan pada basis data Oracle dengan menggunakan teknik FOR UPDATE yang terlihat pada data *history* monitoring pelanggan dengan 2PL yang konsisten.

4. KESIMPULAN

Sistem monitoring pelanggan berhasil diimplementasikan dengan menggunakan protokol 2PL. Berdasarkan ujicoba pada permasalahan *the lost of update* menggunakan teknik FOR UPDATE pada basis data Oracle terbukti dapat menjamin konsistensi data hasil monitoring pelanggan.

DAFTAR PUSTAKA

Connolly, Thomas & Begg, Carolyn E. ,(2002) , *Database Systems: A practical Approach to Design, Implementation, and Management, 3rd Ed.,* Pearson Education limited, England.

Darmawan, J.B.B., (2007) , *Teknik Kontrol Concurrency Menggunakan 2pl Dalam Mysql Untuk Menangani Masalah Dalam Concurrency*. Seminar Nasional Fakultas Sains dan Teknologi, Universitas Sanata Dharma, Yogyakarta.

Oracle Corporation, (2002) , *Java Stored Procedures Developer's Guide*, Oracle Corporation.

Oracle University, (2004) , *Oracle Database 10g: SQL Fundamentals I* , Oracle University.

Oracle University, (2004) , *Oracle Database 10g: SQL Fundamentals II* , Oracle University.

Raharjo, B., Heryanto, I., dan Haryano, A., (2012) , *Mudah Belajar Java Edisi Revisi Kedua,* Informatika , Bandung.

Whitten, J.L., and Bentley, L.D., (1998) , *Systems Analysis & Design Methods*, 4th Ed., Irwin/McGraw-Hill International Co., New York.