Dea Wahsa Saputri

PERTEMUAN 2

TRANSAKSI DALAM SISTEM

BASIS DATA

Transaksi

- Transaksi basis data mencerminkan dunia transaksi nyata yang dipicu oleh peristiwa seperti membeli produk, mendaftar untuk kursus, atau membuat rekening Koran sebuah bank.
- Semua bagian dari transaksi harus berhasil diselesaikan untuk mencegah masalah integritas data.
- Transaksi adalah setiap tindakan yang membaca dari dan atau menulis ke database.
- Sebuah transaksi dapat terdiri dari pernyataan SELECT sederhana untuk menghasilkan daftar isi ssebuah table.
- Mungkin terdiri dari serangkaian pernyataan UPDATE terkait untuk mengubah nilai-nilai dari atribut di berbagai table.
- Mungkin terdiri dari serangkaian pernyataan INSERT untuk menambahkan baris untuk satu atau lebih table, atau mungkin terdiri dari kombinasi SELECT, UPDATE, dan INSERT.

Contoh transaksi sederhana

START TRANSACTION;

SELECT CUSTOMER.CUST_NUMBER, CUSTOMER.CUST_BALANCE

FROM CUSTOMER

WHERE CUSTOMER.CUST_NUMBER=10016

COMMIT;

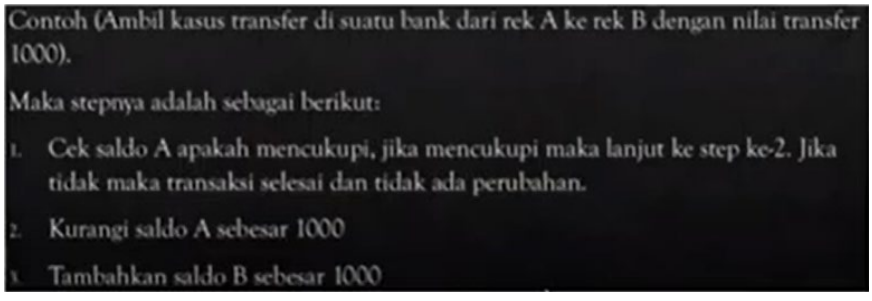
Syarat transaksi

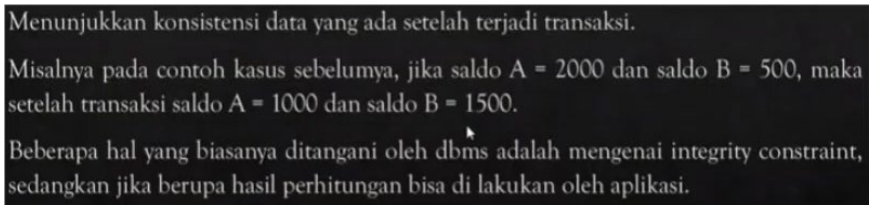
Atomicity

- Atomicity mensyaratkan bahwa semua operasi SQL dari transaksi akan selesai, jika tidak, transaksi tersebut dibatalkan.
- Jika transaksi T1 memiliki empat permintaan SQL, keempat permintaan harus berhasil diselesaikan, jika tidak, seluruh transaksi dibatalkan.
- Transaksi diperlakukan sebagai satu unit, tidak terbagi, kerja logis.

Consistency

- Consistency menunjukkan kondisi permanen dari database yang konsisten.
- Transaksi mengambil database dari satu kondisi konsistenke keadaan konsisten yang lain.
- Ketika transaksi selesai, database harus dalam keadaan konsisten, jika salah satu bagian transaksi melanggar kendala integritas, seluruh transaksi dibatalkan.





Isolation

- Isolation berarti bahwa data yang digunakan (dieksekusi) oleh transaksi T1 tidak dapat digunakan (diakses) oleh transaksi T2 sampai transaksi T1 dengan selesai.
- Property ini sangat berguna dalam lingkungan database multiuser karena beberapa pengguna dapat mengakses dan memperbarui dabase pada saat yang sama.

Durability

- Durability memastikan bahwa perubahan transaksi sekali selesai (berkomitmen). Mereka tidak dapat dibatalkan atau hilang, bahkan dalam hal terjadi kegagalan sistem

Contoh:

Kondisi awal : A = 2000 | B = 500

Setelah dilakukan transaksi T1 = A → B (transfer)

Kondisi awal A : 1000, misalnya ditengah perjalanan transaksi ada kegagalan system maka dia harus menjamin bahwa dia bisa kembali menyimpan data itu secara utuh atau data awal. Misalnya kita bisa membayangkan bahwa program didalam system itu ada proses penjumlahan dan pengurangan. Jadi mengurangi saldo di A dan menambah saldo di B, misalnya yang pertama mengurangi saldo di A artinya A itu akan dikurangi pas menuju berikutnya B ditambah 1000 itu ada kegagalan system maka secara ini A sudah tidak 2000 karena telah dikurangi dengan 1000 maka apabila ada kegagalan system maka dia harus dapat menyimpan data itu kembali utuh 2000. Secara garis besar kemampuan untuk menyimpan data secara utuh dan misalnya transaksi selesai bisa menyimpan nilai komit setelah transaksi itu berhasil.

Serializability

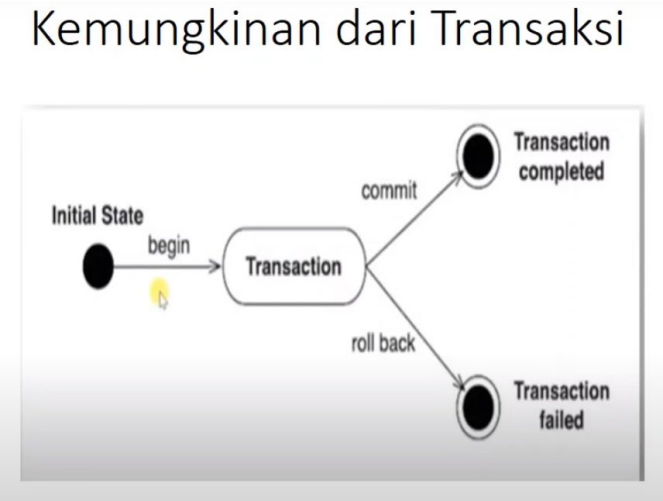
- Serializability memastikan bahwa jadwal untuk pelaksaan bersamaan dari transaksi menghasilkan property results.
- Konsistensi adalah penting dalam database multiuser dan terdistribusi, di mana beberapa transaksi kemungkinan akan dieksekusi secara bersamaan.
- Jika hanya satu transaksi dijalankan, serializability tidak menjadi masalah.

Operasi transaksi

- Commit
  Operasi yang berfunsi menandakan bahwa transaksi telah selesai dilakukan.

- Rollback

Operasi yang berfungsi untuk transaksi itu harus diulang karena adanya fail (kegagalan), diulang di status sebelum transaksi dilakukan.



PERTEMUAN 3

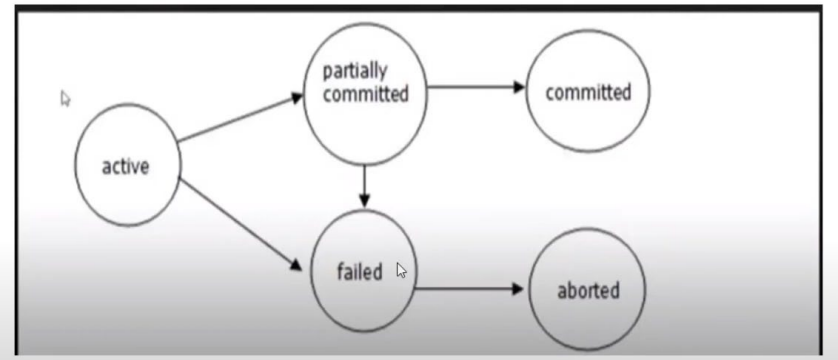
Status transaksi

- **Aktif** (active) merupakan status awal (initial state) sebuah transaksi yang menunjukkan transaksi tersebut masih dieksekusi.
- **Berhasil sebahagian** (partially commited), keadaan yang dicapai transaksi tepat pada saat operasi berakhir dalam transaksi selesai dikerjakan.
- **Gagal** (failed) yaitu merupakan keadaan dimana sebuah transaksi terhenti pengekskusiannya sebelum tuntas sama sekali.
- **Batal** (aborted) yaitu keadaan dimana sebuahh transksi dianggap tidak/belum dikerjakan. Yang tentu dengan terlebih dahulu diawali dengan mengembalikkan semua data yang telah diubah ke nilai-nilai semula(yang menjadi tanggung jawab DBSM)
- **Berhasil sempurna** (commited), keadaan dimana transaksi telah dinyatakan berhasil dikerjakan seluruhnya dan basis data telah mereflesikan perubahan-perubahan yang memang diinginkan transaksi.

Kemungkinan dari transaksi

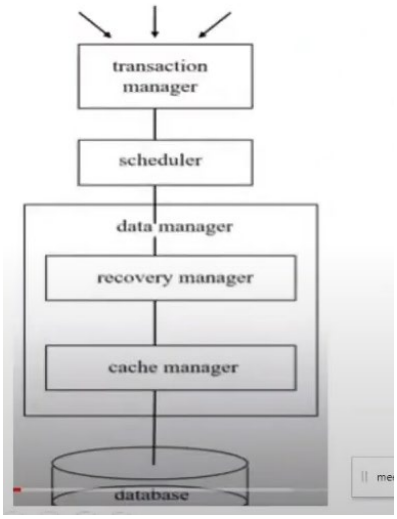
- Jika dilaksanakan lengkap seluruhnya, transaksi tersebut telah di commit dan basis data telah mencapai keadaan konsisten baru.
- Jika transaksi tidak sukses, maka transaksi dibatalkan dan basis data dikembalikan ke keadaan konsisten sebelumnya(roolback).

Siklus transaksi



PERTEMUAN 4

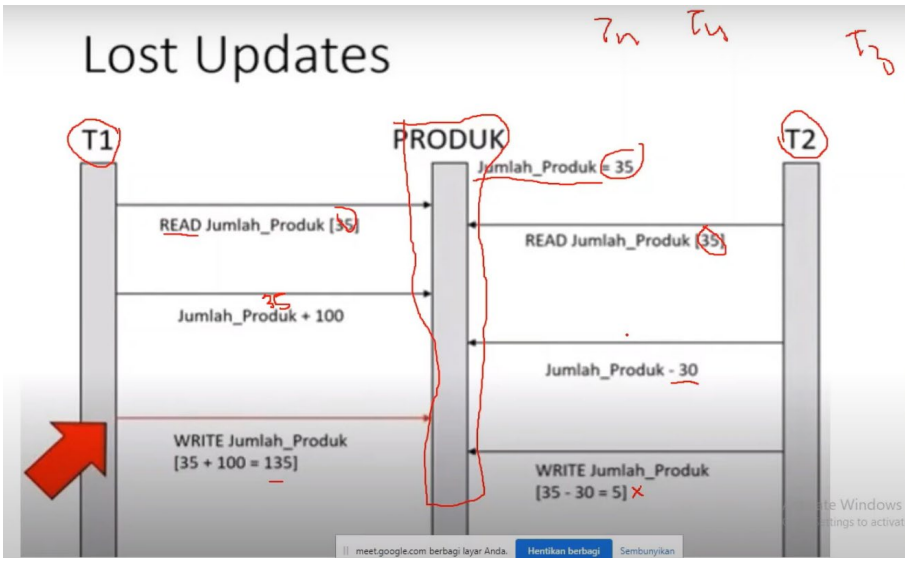
Skema Transaksi

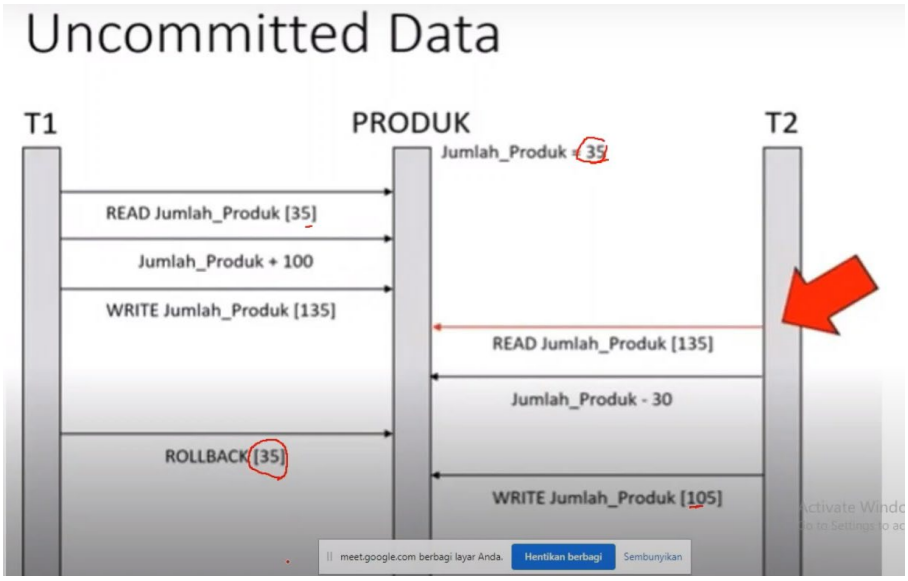


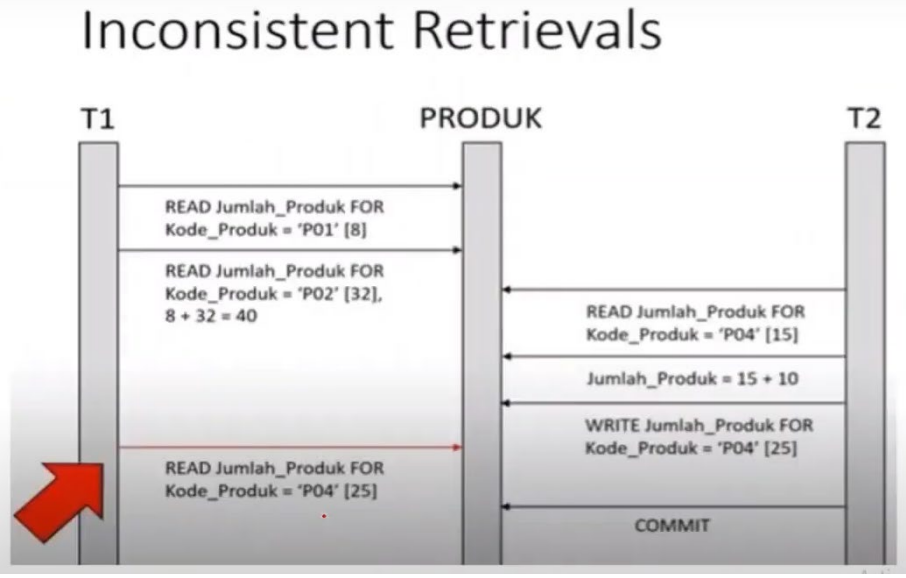
- **Transaction Manager :** pra-pengolahan transaksi basis data mengelola distribusi data untuk trnsaksi multi-user.
- **Scheduler :** menentukan urutan relative transaksi.
- **Data manager :** mengatur penyimpanan data di data store.
- **Recovery Manager :** bertanggung jawab untuk melakukan dan membatalkan, jika terjadi kegagalan system saat memori volatile dan media hilang(error).
- **Cache Manager :** mengelola penyimpanan volatile (cache, dalam data memori), yang beroperasi pada database.

Concurrency Control

- Koordinasi pelaksaan **transaksi simultan** dalam system database multiuser dikenal sebagai control konkurensi.
- Tujuan dari concurrency control adalah untuk menjamin berhasilnya serialisasi transaksi dalam control lingkungan basis data(DBSM).
- Masalah utama control konkurensi adalah hilangnya data pembaruan, data belum ‘COMMIT’ dan hasil pengolahan tidak konsisten.


Lost Updates
T1
PRODUK
T2
Jumlah_Produk = 35
READ Jumlah_Produk [35]
READ Jumlah_Produk [35]
Jumlah_Produk + 100
Jumlah_Produk - 30
WRITE Jumlah_Produk
[35 + 100 = 135]
WRITE Jumlah_Produk
[35 - 30 = 5]
meet.google.com berbagi layar Anda.
Hentikan berbagi
Sembunyikan


Uncommitted Data
T1
PRODUK
T2
Jumlah_Produk = 35
READ Jumlah_Produk [35]
Jumlah_Produk + 100
WRITE Jumlah_Produk [135]
READ Jumlah_Produk [135]
Jumlah_Produk - 30
ROLLBACK [35]
WRITE Jumlah_Produk [105]
meet.google.com berbagi layar Anda.
Hentikan berbagi
Sembunyikan



Concurrency Control Method : Optimistic

- **Penundaan pengecekan** apakah transaksi memenuhi isolasi dan aturan integritas lainnya (misalnya serializability dan recovery) sampai akhir, tanpa memblokir transaksi yang membaca atau menulis.
- Membatalkan trasaksi untuk mencegah pelanggaran, jika aturan yang diinginkan harus dilanggar atas yang COMMIT.

Concurrency Control Method : Pessimistic

- **Melakukan blok operasi transaksi** jika dapat menimbulkan pelanggaran aturan, sampai kemungkinan pelanggaran menghilang.
- Memblokir operasi biasanya akan berakibat pada pengurangan kinerja.

Concurrency Control Method : Semi Optimistic

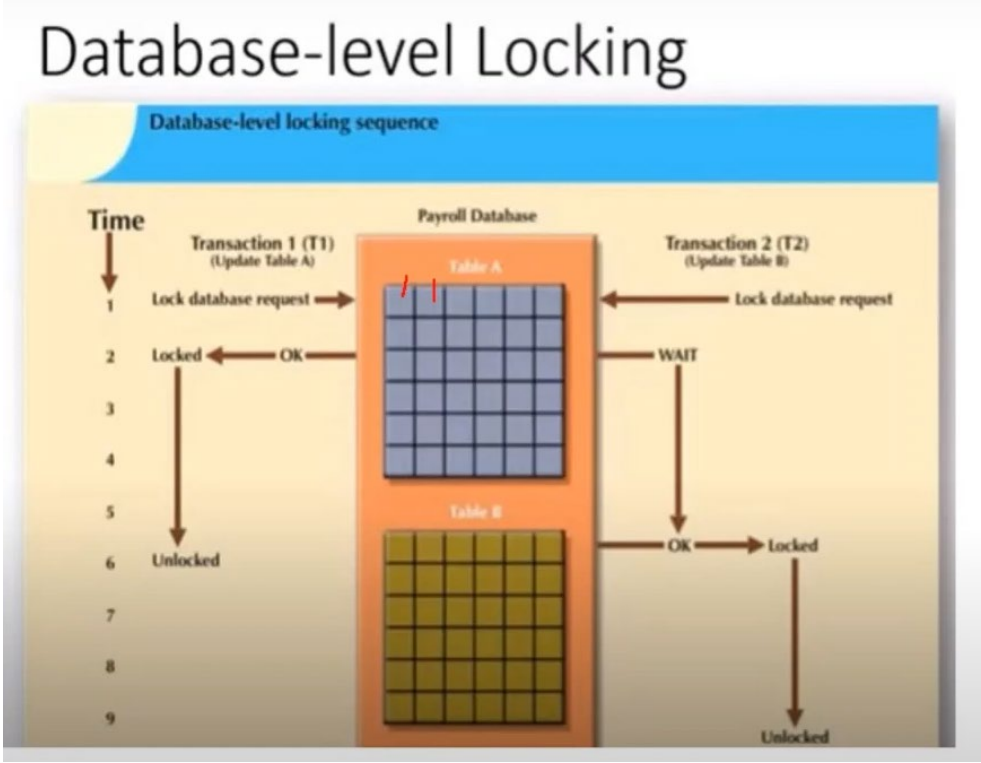
- Melakukan blok operasi dalam beberapa transaksi, jika transaksi tersebut dapat **menyebabkan pelanggaran** beberapa aturan.
- Tidak memblokir dalam situasi lain menggunakan aturan yang akan menunda pemeriksaan (jika diperlukan) sampai akhir transaksi ini, seperti yang dilakukan dengan mekanisme optimis.

PERTEMUAN 5

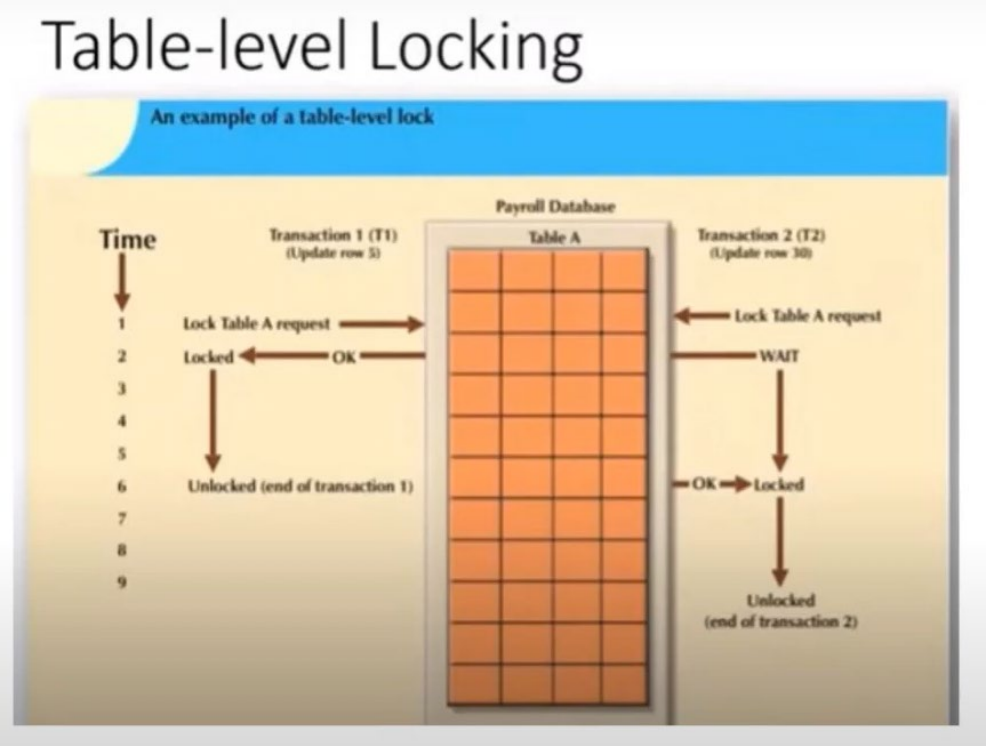
LOCKING

Locking Method

- Metode penguncian menjamin penggunaan eksklusif dari item data ke transaksi yang sedang berjalan (isolasi data).
- Sebuah transaksi akan dikunci sebelum akses data, kunci dilepaskan ketika transaksi selesai, sehingga transaksi lain dapat mengunci item data untuk penggunaan eksklusif selanjutnya.
- **Lock Granularity** menunjukkan tingkat penggunaan penguncian.

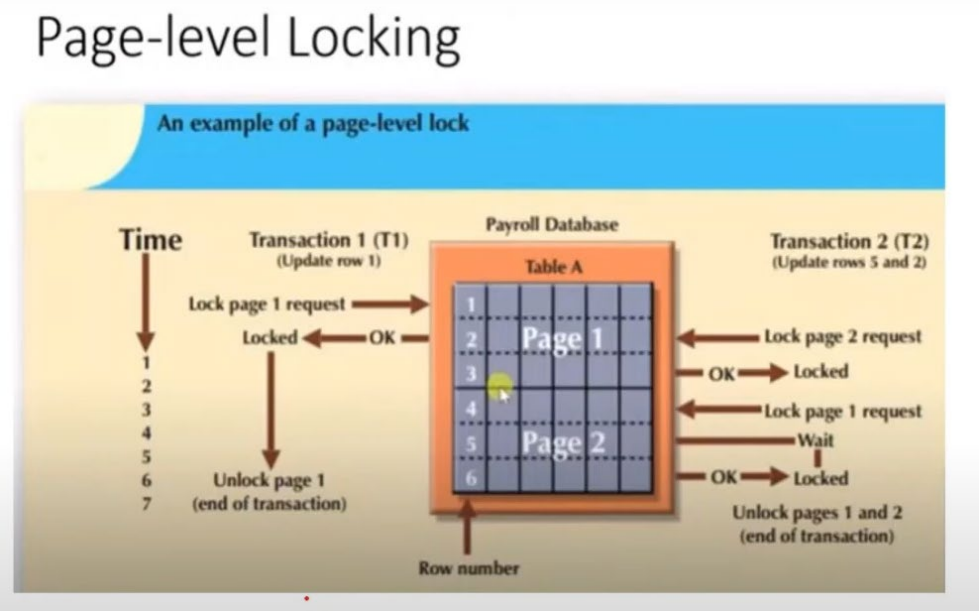


- T1 meminta request database untuk mengakses table A → kunci diberi sampai selesai → di unlock sampai update (tidak dapat digunakan oleh transaksi lain).
- Pada saat dikunci → T2 ingin update table B pada database yang sama → harus wait sampai T1 unlock dan memberikan pemberitahuan bahwa T1 sudah selesai → transaksi 2 bisa mengakses.
- 2 transaksi mengakses table berbeda maka tidak masalah.



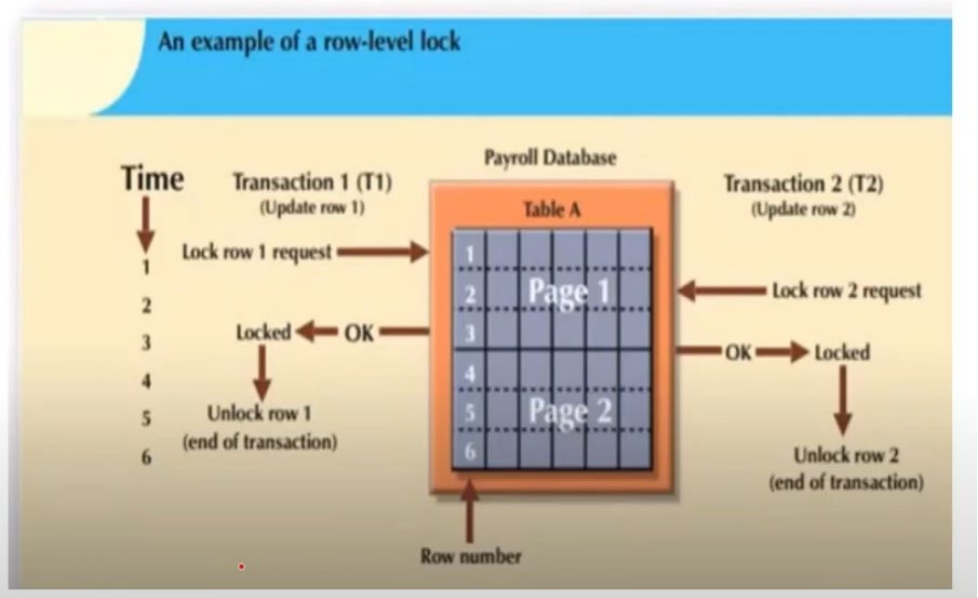
- Database terdiri dari banyak table
- Table yang dikunci
- T1 update row5, T2 update row 30
- T1 meminta request table A (update row 5)
- Kunci diberikan
- T1 berhasil mengunci table A
- T2 ingin mengupdate table A, walaupun baris berbeda harus menunggu sampai update T1 selesai
- Ketika T1selesai, maka T2 bisa update dan table dikunci juga sampai T2 selesai

Page-level Locking



- Table dalam database bisa dibagi menjadi beberapa halaman
- Table A dibagi menjadi 2 page → 1 page 2 baris
- Transaksi berbeda dipage yang berbeda diperbolehkan karena yang di lock adalah pagenya
- T1 update row 1 pada page 1
  T2 update row 2 pada page 1 dan row 5 page 2
- T1 request page 1 → terkunci
- T2 harus menunggu T1 unlocked page 1
- T2 update page 1

Row-level Locking



- Meski page sama, namun row berbeda tidak jadi masalah
- T1 update baris 1 dan T2 update baris 2, bisa dilaksanakan pada waktu yang sama
- T1 hanya mengunci row 1
- Masalah akan timbul kalau database sama, table sama, page sama, baris sama
- DBSM akan melihat scheduler untuk menentukan prioritas
- T1 me lock, T2 menunggu sampai T1 selesai, T2 update

Field-level Locking

- Kunci tingkat field memungkinkan transaksi konkuren untuk mengakses baris yang sama selama mereka memerlukan penggunaan berbagai bidang(atribut) dalam baris itu.
- Kunci biner (binary locks) hanya memiliki dua kondisi : terkunci (1) atau tidak terkunci (0).

  - Jika granuality semakin kecil, cost dan resources akan semakin besar → proses melambat.
  - Locking table paling popular digunakan granularity sedang, cost dan resources akan efisien.

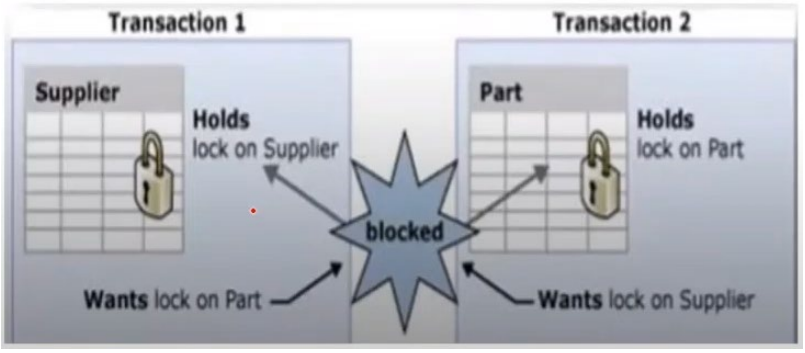
Contoh :

MySQL : Locking Tables

```
LOCK TABLES trans READ, customer WRITE;
SELECT SUM(value) FROM trans
    WHERE customer_id=some_id;
UPDATE customer
    SET total_value=sum_from_previous_statement
    WHERE customer_id=some_id;
UNLOCK TABLES;
```

Deadlocks

- Kebuntuan terjadi bila dua transaksi menunggu tanpa batas waktu untuk membuka data.
- Misalnya, kebuntuan terjadi apabila dua transaksi, T1 dan T2, ada dalam modus berikut : T1 = akses data item X dan Y, T2 = akses data item Y dan X.



T1 menunggu T2 dan T2 menunggu T1 (saling menunggu)

Deadlock Control

1. **Deadlock prevention.** Transaksi meminta kunci baru dibatalkan bila ada kemungkinan bahwa kebuntuan dapat terjadi. Semua perubahan yang dibuat oleh transaksi ini 'roll-back' dan semua kunci yang diperoleh oleh transaksi dilepaskan. Transaksi ini kemudian dijadwalkan ulang untuk eksekusi.

2. **Deadlock detection.** DBSM secara berkala menguji database untuk kebuntuan. Jika kebuntuan ditemukan, salah satu transaksi akan dibatalkan dan transaksi lainnya berjalan terus.

3. **Deadlock avoidance.** Transaksi ini harus mendapatkan semua kunci yang dibutuhkan sebelum dapat dieksekusi.

PERTEMUAN 7

RECOVERY

Database recovery management

- Pemulihan basis data mengembalikan basisdata dari kondisi tertentu(biasanya tidak konsiten) ke kondisi sebelumnya yang konsisten.
- Teknik pemulihan didasarkan pada properti transaksi atomik: semua bagian dari transaksi harus diperlakukan sebagai unit kerja logis tunggal di mana semua operasi diterapkan dan diselesaikan untuk menghasilkan basis data yang konsisten.

Transaction log

- DBSM menggunakan log transaksi untuk melacak semua transaksi yang memperbarui databse**.**
- Informasi yang disimpan dalam log ini digunakan oleh DBSM untuk kebutuhan pemulihan yang dipicu oleh pernyataan ROLLBACK, terminasi abnormal suatu program, kegagalan system atau disk crash.

4 konsep proses pemulihan

- Protokol write-ahead-log memastikan bahwa log transaksi selalu ditulis sebelum data diperbarui.
- Redundant transaction log memastikan bahwa kegagalan fisik tidak akan mengganggu kemapuan DBSM untuk memulihkan data.
- Buffer basis data adalah area penyimpanan sementara di memori utama yang digunakan untuk mempercepat operasi disk.
- Checkpoint basis data adalah operasi di mana DBSM menulis semua buffer yang telah diperbarui dalam memori disk, operasi ini dicatat dalam log transaksi.

Prosedur pemulihan transaksi

- Deferred-write : operasi transaksi tidak segera memperbarui basis data fisik, hanya log transaksi yang diperbarui.Database diperbarui secara fisik hanya dengan data dari transaski dengan menggunakan informasi dari log transaksi.
- Write-through : database segera diperbaharui oleh operasi transaksi selama eksekusi transaksi, bahkan sebelum transaksi mencapai COMMIT. Jika transaksi tersebut dibatalkan, digunakan operasi ROLLBACK.