08

Transaksi

by: Ahmad Syauqi Ahsan

Kenapa Transaksi

- Sistem database secara normal adalah diakses oleh banyak user atau proses pada waktu yang bersamaan.
  - Baik query maupun modifikasi (insert, update, delete)
- Tidak seperti sistem operasi (seperti: Windows atau Linux) yang memungkinkan adanya interaksi diantara proses, sebuah DBMS perlu untuk menjaga proses-proses didalamnya untuk terhindar dari masalah yang mungkin timbul dari adanya interaksi antar proses.

Contoh: Interaksi yang buruk

- Anda dan partner anda masing-masing mengambil $100 dari account yang sama pada mesin ATM yang berbeda secara hampir bersamaan.
  - DBMS harus memastikan bahwa pengambilan uang pada salah satu account tidak hilang
- Perbandingan: sebuah OS memungkinkan dua orang untuk meng-edit sebuah dokumen pada waktu yang bersamaan. Jika keduanya menulis, perubahan dari salah satunya akan hilang.

Transaksi

- **Transaksi** = adalah proses-proses yang melibatkan query dan/atau modifikasi pada database
- Umumnya dengan beberapa property yang berkaitan dengan masalah concurrency
- Terbentuk dalam SQL dari statemen tunggal atau kontrol programmer secara eksplisit.

ACID Transaction

- **ACID transaction** adalah:
  - **Atomic**: dikerjakan seluruhnya atau tidak sama sekali.
  - **Consistent**: tetap mempertahankan batasan-batasan (constraints) pada database
  - **Isolated**: yang terlihat oleh user hanya ada satu proses yang dieksekusi pada satu waktu
  - **Durable**: akibat dari suatu proses tetap ada walaupun ada crash.
- ***Opsional***: bentuk yang lebih lemah dari suatu transaksi seringkali juga didukung oleh ACID transaction.

COMMIT

- Statement COMMIT pada SQL menyebabkan sebuah transaksi selesai sempurna.
  - Setiap modifikasi pada database akan tersimpan secara permanen

ROLLBACK

- Statemen ROLLBACK pada SQL juga menyebabkan sebuah transaksi selesai, tetapi dengan cara menggagalkan (abort).
  - Tidak menimbulkan efek apa-apa pada database
- Kesalahan seperti pembagian dengan 0 atau constraint violation (pelanggaran terhadap batasan) dapat juga menyebabkan proses ROLLBACK, bahkan jika sang programmer tidak memintanya

Contoh:Proses-proses yang berinteraksi

- Asusmsi ada relasi (tabel) **Sells**(bar, beer, price), dan misalkan Bar milik Joe menjual hanya bir Bud untuk R2.50 dan bir Miller untuk $3.00
- Sally meng-query **Sells** untuk harga tertinggi dan harga terendah yang diterapkan oleh Joe.
- Joe memutuskan untuk berhenti menjual bir Bud dan bir Miller, tetapi akan menjual bir Heineken saja pada $3.50

Program milik Sally

- Sally mengeksekusi dua statemen SQL berikut yang disebut **(min)** dan **(max)** untuk membantu kita mengingat apa yang mereka kerjakan.

```
(max)     SELECT MAX(price) FROM Sells
          WHERE bar = 'Joe''s Bar';
(min)     SELECT MIN(price) FROM Sells
          WHERE bar = 'Joe''s Bar';
```

Program milik Joe

- Pada waktu yang hampir bersamaan, Joe mengeksekusi langkah-langkah berikut. (del) dan (ins)

```
(del)     DELETE FROM Sells
          WHERE bar = 'Joe''s Bar';
(ins)     INSERT INTO Sells
          VALUES ('Joe''s Bar', 'Heineken',
               3.50);
```

Interleaving of Statements

- Meskipun (max) harus dilakukan sebelum (min), dan (del) dilakukan sebelum (ins), tidak ada batasan lain pada urutan dari statemen-statemen ini, kecuali kita mengelompokkan statemen milik Sally dan/atau milik Joe kedalam transaksi.

Contoh: Strange Interleaving

- Misalkan urutan eksekusi adalah seperti berikut:
  (max) (del) (ins) (min)

Harga dari Joe	{2,50; 3.00}	{2,50; 3.00}		{3.50}
Statement	(max)	(del)	(ins)	(min)
Hasil	3.00			3.50

- Sally akan melihat bahwa nilai MAX < MIN

Menyelesaikan permasalahan dengan Transaksi

- Jika kita mengelompokkan statemen-statemen dari Sally (max) dan (min) kedalam satu transaksi, maka dia tidak akan dapat melihat ketidakkonsistensian ini.
- Sally akan melihat harga-harga milik Joe pada satu waktu yang sama.
  - Bisa jadi sebelum atau sesudah Joe merubah harga ataupun ditengah-tengah, nilai **(max)** dan **(min)** akan di hitung dari harga yang sama pada waktu yang sama pula.

Masalah yg Lain: ROLLBACK

- Misalkan Joe mengeksekusi (del)(ins) tidak sebagai transaksi. Tetapi setelah proses eksekusi statemen-statemen ini, dia mengeluarkan statemen ROLLBACK
- Jika Sally mengeksekusi statemennya setelah (ins) tetapi sebelum rollback, dia melihat nilai 3.50, yang tidak pernah ada dalam database.

Solusi

- Jika Joe mengeksekusi (del)(ins) sebagai transaksi, akibat yang ditimbulkan tidak dapat dilihat oleh orang lain sampai transaksi tersebut di-COMMIT.
  - Jika transaksi tersebut di-ROLLBACK, maka akibat dari transaksi tersebut tidak akan pernah terlihat

Isolation Level (Level Isolasi)

- SQL mendefinisikan 4 ***level isolasi*** = pilihan interaksi yang diijinkan oleh transaksi yang dijalankan pada waktu yang hampir bersamaan.
- Hanya ada satu level ("serializable") = ACID transaction
- Setiap DBMS mengimplementasikan transaksi dengan caranya masing-masing

Memilih Level Isolasi

- Didalam sebuah transaksi, kita bisa mengatakan:

  SET TRANSACTION ISOLATION LEVEL X

  where X =

  1. SERIALIZABLE
  2. REPEATABLE READ
  3. READ COMMITTED
  4. READ UNCOMMITTED

Serializable Transactions

- Jika Sally = (max)(min) dan Joe = (del)(ins), dimana masing-masing adalah transaksi.
- Dimana Sally menjalankannya dengan level isolasi SERIALIZABLE.
- Sehingga Sally akan melihat database baik sebelum maupun setelah Joe mengeksekusi transaksi. Tetapi tidak pernah ditengah-tengahnya.

Level Isolasi adalah Pilihan

- Pilihan anda, misalkan menjalankan secara serializable, hanya berakibat bagaimana anda melihat suatu database, tetapi tidak bagi orang lain.
- Contoh: Jika Joe menjalankan transaksi secara serializable, tetapi Sally tidak. Maka Sally mungkin melihat tidak ada harga pada Bar milik Joe.
  - Terlihat oleh Sally seperti kalau dia menjalankan transaksinya ditengah-tengah transaksi milik Joe.

Transaksi Read-Committed

- Jika Sally menjalankan dengan level isolasi READ COMMITTED, kemudian dia dapat melihat hanya data-data yang sudah di commit saja, tetapi bisa jadi datanya akan berubah pada waktu yang berbeda.
- Contoh: dibawah READ COMMITTED, interleaving (max)(del)(ins)(min) adalah dimungkinkan, selama Joe melakukan commit.
  - Sally akan melihat MAX < MIN

Transaksi Repeatable-Read

- Kebutuhannya adalah seperti read-committed, ditambah: jika data dibaca lagi, maka segala sesuatu yang terlihat pertama kali akan juga terlihat pada yang kedua kali.
  - Tetapi yang kedua dan pembacaan berikutnya mungkin akan melihat lebih banyak tuple.

Contoh: Repeatable Read

- Misalkan Sally menjalankan transaksi dengan REPEATABLE-READ, dan urutan eksekusinya adalah (max)(del)(ins)(min).
  - (max) melihat nilai 2.50 dan 3.00
  - (min) dapat melihat 3.50, tetapi harus juga melihat 2.50 dan 3.00, karena mereka terlihat oleh pembacaan sebelumnya oleh (max).

Read Uncommitted

- Sebuah transaksi yang berjalan dibawah READ UNCOMMITTED dapat melihat data didalam database, bahkan jika data tersebut ditulis oleh transaksi yang belum di-commit (dan mungkin juga tidak akan pernah di-commit).
- Contoh: Jika Sally menjalankan transaksi dibawah READ UNCOMMITTED, dia dapat melihat harga 3.50 bahkan jika Joe pada akhirnya menggagalkan (abort) transaksinya.

Tanya Jawab

Terima Kasih