Konsep Dasar *In Memory Database*

Ari Fadli
fadli.te.unsoed@gmail.com

Peranan DBMS sebagai perangkat lunak pengelola basis data semakin meningkat di era informasi yang berkembang saat ini. Selain itu unjuk kerja DBMS menjadi salah satu alasan dalam memilih DBMS. Hal ini disebabkan karena unjuk kerja mutlak diperlukan untuk menjamin ketepatan dan kecepatan penyampaian informasi. Saat ini muncul sebuah paradigma baru yang disebut dengan *In-Memory Database* yang seolah-olah menjadi solusi dari permasalahan yang berkaitan dengan unjuk kerja DBMS.

Pendahuluan

Database Management System (DBMS) merupakan suatu perangkat lunak yang digunakan untuk mengelola basis data. Pengelolaan basis data ini dilakukan melalui perintah-perintah khusus yang disebut sebagai *statement* SQL. Waktu yang dibutuhkan oleh sebuah DBMS dalam memproses setiap *statement* SQL tersebut akan mempengaruhi unjuk kerjanya.

Dalam DBMS konvensional yang menggunakan media disk sebagai media penyimpanan datanya maka akan menggunakan *memory* sebagai *buffer* untuk data-data yang diolah. Sehingga masih diperlukan adanya operasi I/O ke *disk* bila data yang akan diolah belum ada di *memory*. Sehingga tentunya hal ini akan memberikan dampak pada menurunnya unjuk kerja DBMS.

Sedangkan Konsep yang ditawarkan pada *In Memory Database* (IMDB) adalah bahwa seluruh data akan disimpan di dalam *memory* komputer kemudian menjadikan *memory* komputer sebagai tempat penyimpanan data utama, sehingga secara teoritis *In Memory Database* memiliki unjuk kerja yang lebih baik dari DBMS konvensional.

Pada paradigma IMDB, membuat tidak lagi perlu secara terus menerus melakukan proses I/O *disk* untuk mencari data yang akan diolah, karena semua data sudah ada di *memory*. Sehingga diharapkan dengan meletakkan semua data di *memory* maka proses I/O *disk* dapat diminimalisir bahkan dihilangkan.

Keutamaan IMDB dibandingkan dengan DBMS Konvensional

Mengakses data pada memori utama lebih cepat dan lebih dapat diprediksi daripada mengakses data pada *disk* [1]. Layanan penyimpanan membutuhkan ketersediaan tinggi, kinerja yang baik, dan konsistensi yang tinggi [2]. *In Memory Databases* (IMDBs) [3] memenuhi persyaratan ini dan telah muncul sebagai paradigma baru untuk meningkatkan unjuk kerja transaksi [4]. Karena IMDB dapat diakses langsung dari memori, waktu respons lebih cepat, dan *throughput* transaksi yang lebih baik bila dibandingkan dengan *Database Disk-Resident* (DRDB). Hal ini tentunya sangat sejalan dan dibutuhkan dalam pengembangan aplikasi *real-time* di mana transaksi harus diselesaikan dalam jangka waktu yang ditentukan [5].

Apa bedanya IMDB dengan Caching Basis Data ?

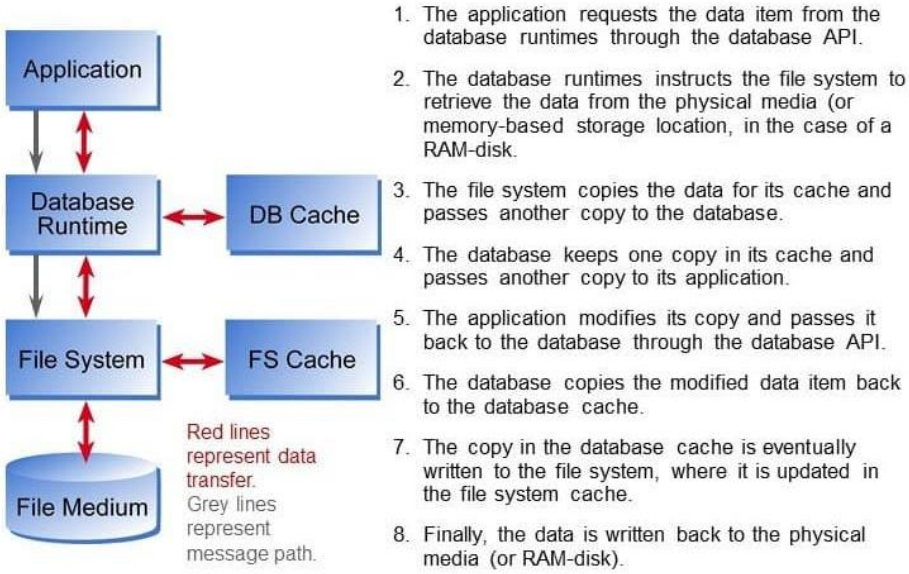
Sebagaimana yang tertulis dalam bagian pendahuluan bahwa *In Memory Database* (IMDB) adalah sebuah konsep yang akan membawa seluruh data tersimpan di dalam *memory* komputer dan dan kemudian menjadikan *memory* komputer sebagai tempat penyimpanan data utama.

Sedangkan dalam konsep *caching* merupakan proses dimana basis data pada disk menyimpan catatan tentang apa saja yang sering diakses dalam memori sehingga membuat akses lebih cepat. Namun konsep *caching* ini hanya mempercepat pengambilan basis data terbaca, tapi sebaliknya jika ada penulisan basis data berupa pembaruan catatan atau pembuatan catatan baru masih harus ditulis melalui cache ke dalam *disk*. Sehingga dengan demikian konsep *chacing* basis data ini hanya berlaku untuk sebagian proses saja. Selain itu, pengelolaan *cache* itu sendiri merupakan proses yang membutuhkan memori dan sumber daya CPU yang substansial sehingga secara unjuk kerja masih berada dibawah *In Memory Database* [6].

Jika IMDB membawa data ke *memory*, lalu mengapa tidak membuat DBMS Konvensional ke *disk*RAM

Memilih solusi dengan menempatkan seluruh basis data pada *disk*RAM akan mempercepat proses baca tulis basis data, namun sayangnya DBMS masih terprogram untuk penyimpanan ke dalam *disk*, sehingga justru akan membutuhkan atau memfasilitasi untuk melakukan proses penulisan kedalam disk, seperti misalnya *caching* dan proses I/O.

Selain hal tersebut DBMS konvensional harus mampu mendeliver nilai ke berbagai lokasi saat digunakan. Sebagaimana yang ditunjukan pada Gambar 1 menunjukkan proses yang diperlukan aplikasi untuk membaca sepotong data dari *database* pada *disk*, memodifikasinya dan menulis catatan itu kembali ke *database*. Tampak dari Gambar 1 tersebut bahwa ini ini memerlukan siklus waktu dan CPU, dan tidak dapat dihindari dalam *database* tradisional, bahkan ketika itu berjalan pada *disk* RAM [6].



Gambar-1. Mekanisme Penyimpanan Data dalam Konsep Database Konvensional [8]

Unjuk Kerja Database Konvensional, Database Konvensional pada ondiskRAM, dan In Memory Database

Dalam sebuah penelitian dihasilkan bahwa dengan memindahkan database konvensional ke *disk*RAM menghasilkan akses baca yang hampir 4x lebih cepat dan update basis data yang lebih dari 3x lebih cepat dibandingkan dengan database konvensional.

Sedangkan ketika digunakan In Memory Database dihasilkan lebih cepat 4x untuk proses baca dan 420x lebih cepat untuk proses tulis jika dibandingkan dengan *database* konvensional diskRAM [7].

Perbedaan Lainnya

Sebagaimana yang telah disebutkan bahwa DBMS konvensional akan menggunakan *disk* sebagai *storage* utamanya sehingga sangat dipengaruhi oleh mekanisme proses I/O. Sedikit dan banyaknya penggunaan mekanisme proses I/O akan berdampak pada konsumsi memori dan CPU dan hal ini akan mempengaruhi kinerjanya Selain hal tersebut DBMS konvensional juga menyimpan banyak data berlebih, adanya data duplikat dalam struktur indeks, yang memungkinkan DBMS konvensional ini untuk mengambil data dari indeks selain dengan menggunakan mekanisme file I/O [6].

Apakah database akan hilang, jika ada sistem crash?

IMDB memiliki sebuah tools yang akan melakukan logging pada setiap transaksi (*transaction logging*) yang terjadi yang disebut sebagai *savepoints* dan ditulis kedalam non-volatile media. Jika sistem mengalami kegagalan maka IMDB ini memiliki mekanisme *rolls back* yaitu kembali ke transaksi terkahir dan mekanisme *rolls forward* (yaitu menyelesaikan setiap transaksi yang belum selesai),sesaat sebelum sistem mengalami crash

Ketersediaan *Non-volatile* RAM atau NVRAM sangat mendukung paradigma IMDB ini. Satu jenis NVRAM adalah *batery*-RAM. RAM ini memiliki baterai yang dapat digunakan saat perangkat dimatikan atau kehilangan sumber daya, sehingga konten isi memori tetap ada. Generasi terbaru mengganti baterai dengan super-capasitor. Selain teknologi *super-capasitor* terdapat jenis RAM *feroelektrik* (FeRAM), RAM *magnetoresistif* (MRAM) dan RAM perubahan fase (PRAM) dirancang untuk memelihara informasi ketika kehilangan daya [6].

Aplikasi apa yang cocok dengan konsep IMDB ?

In Memory Database ini paling umum digunakan dalam aplikasi yang menuntut akses data yang sangat cepat, penyimpanan dan manipulasi, dan dalam sistem yang biasanya tidak memiliki *disk* tetapi tetap harus mengelola jumlah data yang cukup besar. Selain alasan tersebut IMDB juga baik digunakan pada sistem tertanam yang memiliki sumber daya terbatas [6].

Apakah sama IMDB dan "Basis Data Tertanam"?

'Basis Data Tertanam' tidak ada kaitannya dengan sistem tertanam, istilah Basis Data Tertanam memiliki arti bahwa sistem manajemen data base dinyatakan sebagai satu set pustaka kode objek yang nantinya akan dikaitkan dengan kode objek aplikasi. Dengan kata lain, fungsionalitas basis data menjadi bagian dari aplikasi itu sendiri, di ruang alamat yang sama.

a) Sistem basis data tertanam dapat berupa *In Memory Database* atau basis data persisten (DBMS Konvensional berbasis *disk*).
b) *In Memory Database* dapat berupa sistem basis data tertanam, atau dapat berupa sistem basis data klien / server.
c) Sistem basis data klien / server dapat berupa *In Memory Database*, atau dapat berupa sistem basis data persisten.

Dalam IMDB, bagaimana Data Disimpan ?

Proses menyimpan data berdasarkan pendekatan IMDB memiliki dua mekanisme yaitu penyimpanan data berorientasi kolom dan berorientasi baris. Penyimpanan yang berorientasi baris dan penyimpanan berbasis kolom. Perhatikan Contoh berikut

Penyimpanan data yang berorientasi baris, maka data akan secara runtun di letakan pada satu baris yang sama
001:10,Smith,Joe,80000;
002:12,Jones,Mary,50000;
003:11,Johnson,Cathy,44000;
004:22,Jones,Bob,55000;

Penyimpanan data yang berorientasi kolom membuat semua nilai berada pada kolom bersama, lalu nilai kolom berikutnya, dan seterusnya.
10:001,12:002,11:003,22:004;
Smith:001,Jones:002,Johnson:003,Jones:004;
Joe:001,Mary:002,Cathy:003,Bob:004;
80000:001,50000:002,44000:003,55000:004;

Database dalam memori harus tetap mempertahankan properti "ACID". Properti ACID yang dimaksud adalah *Atomicity, Consistency, Isolasi, dan Durability*. Properti ACID yang dimaksud merupakan seperangkat sifat yang menjamin bahwa database transaksi diproses andal.

Transaksi adalah sebuah unit eksekusi dari program yang mengakses dan memungkinkan update berbagai macam tipe data. Transaksi hrs memperhatikan Konsistensi Database. Selain itu ACID merupakan salah satu konsep paling penting dari teori database transaksional, ACID memiliki makna sebagai berikut :

- *Atomicity* mengacu pada kemampuan database untuk menjamin bahwa baik semua bagian transaksi dilakukan atau tidak sama sekali. Jika salah satu bagian dari transaksi gagal, seluruh transaksi gagal.
- *Consistency* memastikan data dapat dikembalikan dalam keadaan sebelum transaksi dimulai, jika terjadi kegagalan.
- *Isolation* memastikan transaksi yang masih dalam proses dan belum dilakukan (*committed*) harus tetap terisolasi terhadap transaksi lainnya.
- *Durability* memastikan data yang telah disimpan (*committed data*) disimpan oleh sistim sebagaimana keadaannya , bahkan jika dalam keadaan kegagalan sistim dan restart sistem, data tersebut tersedia dalam tahapan dan keadaan yang benar [9].

- Meskipun *In Memory Database* dapat sedikit mengabaikan dalam hal *Durability* didasarkan pada berdasarkan penyimpanan yang tidak persisten [10].

Penutup

In Memory Database (IMDB) merupakan sebuah paradigma *database* dimana data akan disimpan di dalam *memory* komputer kemudian menjadikan *memory* komputer sebagai tempat penyimpanan data utama. Dengan menjadikan *memory* sebagai lokasi penyimpanan utama membuat secara teoritis In Memory *Database* memiliki unjuk kerja yang lebih baik dari DBMS konvensional (*on disk*) *In Memory Database* memiliki kecepatan lebih cepat 4x untuk proses baca dan 420x lebih cepat untuk proses tulis jika dibandingkan dengan database konvensional *disk*RAM pendekatan IMDB memiliki dua mekanisme yaitu penyimpanan data berorientasi kolom dan berorientasi baris.

Referensi

[1]. Telecommunication Systems Signs up as a Reseller of TimesTen; Mobile Operators and Carriers Gain Real-Time Platform for Location-Pased Services [N]. Business Wire, 2002-06-24.

[2]. Camargos I, Pedone F, Schmidt R. A Primary-Backup Protocol for In-Memory Database Replication [C]//Proceedings of 5th IEEE International Symposium on Network Computing and Applications (NCA'06), Jul 10-12, 2008, Cambridge, MA, USA. Los Alamitos, CA, USA: IEEE Computer Society, 2006: 204-211.

[3]. Garcia-Molina H, Salem K. Main Memory Database Systems: An Overview. IEEE Transactions on Knowledge and Data Engineering, 1992, 4(6): 509-516.

[4]. Blott S, Korth H F. An Almost-Serial Protocol for Trans-Action Execution in Main-Memory Database sSystems [C]//Proceedings of the 28th International Conference on Very Large Data Bases (VLDB'02), Aug 20-23, 2002, Hong Kong, China. San Francisco, CA, USA: Morgan Kaufmann Publishers, 2002: 706-717.

[5]. Deng Kun, LIAO Guoqiong, Huang Yukun, et al. A Novel Storage Architecture of In-Memory Databases Supporting Real-Time E-Commerce Applications [C]//Proceedings of the International Conference on Management of E-Commerce and E-Government (ICMECG'08), Oct 17-19, 2008, Nanchang, China. Washington, DC, USA: IEEE, 2008: 252-256.

[6]. Doyle, Conan, Sir Arthur, In-Memory Database Questions & Answers, https://www.mcobject.com/in_memory_database/, diakses pada 22 Juni 2019

[7]. Doyle, Conan, Sir Arthur, White papers for developers from McObject https://www.mcobject.com/download/in-memory-vs-ram-disk-databases-a-linux-based-benchmark/, diakses pada 22 Juni 2019

[8]. Graves, Steve, IMDSs Bring Off-the-Shelf Database Software to the Electronics Mainstream,https://www.electronicdesign.com/embedded/imdss-bring-shelf-database-software-electronics-mainstream, diakses pada 22 Juni 2019

[9]. Anonymous, ACID, https://idbigdata.com/official/explandict/acid/, diakses pada 22 Juni 2019

[10]. Raima Database Manager, In Memory Database, https://raima.com/in-memory-database/, , diakses pada 22 Juni 2019