Dasar Teori

Vertical Splitting

Salah satu cara untuk meningkatkan performansi suatu basis data dapat dilakukan dengan membagi tabel menjadi bagian bagian yang lebih kecil. Vertical Splitting dilakukan dengan memisahkan beberapa kolom dari kolom lainnya pada tabel tersebut. Tujuan vertical splittting adalah untuk mempercepat pengambilan informasi yang seringkali dibutuhkan. Dengan memisahkan kolom yang sering digunakan dari kolom yang jarang digunakan menyebabkan pengambilan suatu data untuk query query yang sering digunakan tidak perlu membaca keseluruhan kolom pada tabel tetapi hanya tinggal mengakses tabel dengan kolom yang sering digunakan yang sudah dipisahkan tadi.

Vertical splitting tetap mempertahankan suatu skema dalam bentuk *normal form*. Dan ada beberapa pertimbangan untuk menentukan kapan vertical splitting akan digunakan, diantaranya jika suatu tabel terdapat beberapa kolom yang lebih sering diakses dibandingkan kolom lainnya atau jika suatu tabel memiliki atribut atau jumlah kolom yang sangat banyak namun tidak selalu semua nilai dari atribut tabel tersebut digunakan bersamaan. Contohnya sebagai berikut :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. Rekening | Cabang | Tanggal akun mulai aktif | ..Informasi lain… | Saldo |
| 1415253637 | Dago1 | 2/10/2014 | ….. | 9,000,000 |
| ….. | ……. | ……. | …… | ……. |

Jika dalam keberlangsungannya, ternyata atribut saldo seringkali diakses, digunakan atau dimodifikasi maka lebih baik untuk memisahkan kolom saldo ke tabel baru sehingga tidak perlu embaca keseluruhan tabel lama hanya untuk mengakses kolom saldo.

|  |  |  |  |
| --- | --- | --- | --- |
| No. Rekening | Cabang | Tanggal akun mulai aktif | …Informasi lain… |
| 1415253637 | Dago1 | 2/10/2104 | ….. |

|  |  |
| --- | --- |
| No. Rekening | Saldo |
| 1415253637 | 9,000,000 |

Colapsing Table

Dalam meningkatkan performansi suatu basis data dapat juga dilakukan dengan cara denormalisasi. Dalam hal ini, meskipun melangar aturan *normal form* denormalisasi masih dapat dilakukan dalam batas tertentu untuk meningkatkan performansi basis data tersebut. Collapsing table dilakukan dengan menyatukan tabel yang terpisah ke dalam sebuah tabel yang lebih besar. Hal ini dapat dilakukan jika kebanyakan user membutuhkan gabungan data dari dua tabel yang berbeda. Tujuannya adalah untuk memangkas waktu dengan menghilangkan proses *join* yang sering dibutuhkan dalam memroses query-query yang sering digunakan.

Ketika melakukan collapsing table perlu diperhatikan kalau setiap data pada kedua tabel yang akan digabungkan harus merupakan *one-to-one relationship*. Dan karena menggabungkan kedua tabel tadi menjadi sebuah tabel yang lebih besar, harus diperhatikan juga agar hasil dari collapsing table ini tidaklah menurunkan performa untuk pemrosesan query lainnya, misalnya jika user masih membutuhkan beberapa data saja dari keselurah tabel tadi maka tabel yang sudah digabungkan sebelumnya dapat “dikembalikan” dengan menggunakan SELECT atau VIEW pada query.

Adding redundant column

Menambahkan kolom yang redundant merupakan salah satu cara denormalisasi lain yang dapat dilakukan untuk meningkatkan performansi dari suatu basis data. Tujuannya adalah untuk mengurangi join yang sering dilakukan padahal hanya atribut tertentu saja yang dibutuhkan dari hasil join tersebut. Misalnya jika terdapat tabel fakultas yang berisi nama fakultas, daftar jurusan pada fakultas tersebut, lokasi tata usaha dan nama dekan kemudian ada tabel mahasiswa dengan atribut data diri mahasiswa dan jurusannya. Namun ternyata dalam penggunaannya seringkali dari data mahasiswa dibutuhkan nama fakultas dari mahasiswa tersebut. Umumnya adalah dengan melakukan join antara tabel fakultas dengan mahasiswa untuk memperoleh nama fakultas berdasarkan jurusan dari mahasiswa. Solusi lainnya adalah dengan menambahkan kolom fakultas pada tabel mahasiswa sehingga dalam memperoleh informasi nama fakultas dari mahasiswa tidak perlu dilakukan proses join terlebih dahulu.

Namun perlu diperhatikan penambahan kolom redundan ini perlu dipertimbangkan matang agar memang penambahan kolom tersebut sangat amat dibutuhkan dan dapat mempercepat performansi dari basis data terebut. Hal ini dikarenakan penambahan kolom redundan juga dapat menyebabkan masalah lain seperti membutuhkan perawatan untuk kolom baru tersebut, maksudnya adalah ketika ada penambahan kolom baru maka setiap baris pada tabel tersebut harus disesuaikan dengan mengisi nilai dari kolom tambahannya untuk baris tersebut. Serta perlu diperhatikan juga jika melakukan penambahan kolom redundan dapat membutuhkan ruang memory yang lebih banyak.

Adding Derived Atribut

Untuk meningkatkan performansi dengan menggunakan denormalisasi dapat dilakukan juga dengan cara menambahkan *derived column* atau *derived attribute.* Maksud dari *derived attribute* adalah atribut yang sebenarnya bisa diperoleh dari atribut lain contohnya adalah usia jika pada tabel sudah terdapat tanggal lahir maka nilai dari usia sebenarnya bisa diperoleh dari mengurangkan tanggal sekarang dengan tanggal lahir tersebut.Tujuan dari penambahan *derived attribute*  untuk menghilangkan beberapa join dan juga mempercepat waktu proses untuk agregasi. Sehingga jika suatu nilai yang dapat diperoleh dari atribut lain seringkali dibutuhkan maka dengan menyediakan kolom yang berisikan nilai tersebut dapat memangkas waktu pemrosesan untuk sekian banyak perhitungan tersebut. Dalam menambahkan *derived attribute* perlu diperhatikan juga pertukaran antara keuntungan performansi dengan penggunaan memory. Karena dengan menambahkan *derived column* berarti memory yang dibutuhkan oleh basis data tersebut juga akan bertambah.