Schema & Index Tuning

Wednesday, 17 September 2025 20:25

Diberikan skema basis data berikut ini. Atribut dengan garis bawah adalah *primary key*, sedangkan atribut tercetak *miring* adalah *foreign key* (ke atribut *primary key* dengan nama sama):

SUPPLIER = (SID, SNAME, SADDRESS, SCITY, RATING, BALANCE)
PRODUCT = (PID, PNAME, *CID*, WEIGHT)
PRODUCT_CATEGORY = (CID, CNAME, DESCRIPTION)
SUPPLY = (ORDERDATE, *PID*, *SID*, QUANTITY, PRICE, RECEIVEDATE)

Berdasarkan hasil analisis terhadap *workload* sistem yang telah berjalan dan data yang ada di basis data, diperoleh fakta berikut ini.

- Terdapat 10.000 *record* pada SUPPLIER, 12.000 pada PRODUCT, 1.500.000 pada SUPPLY, dan 100 pada PRODUCT_CATEGORY.
- 90% akses terhadap produk selalu membutuhkan nama kategori dari produk tersebut.
- 75% akses terhadap *supply* hanya mengakses data pesanan dari *supplier* yang belum diterima saja. Setiap saat, rata-rata hanya ada 2000 hingga 3000 pesanan dari *supplier* yang belum diterima. Pesanan yang belum diterima ditandai dengan atribut RECEIVEDATE pada suatu baris data pesanan di tabel SUPPLY bernilai NULL.
- Atribut Alamat, Kota, dan Rating dari *supplier* hanya diakses setiap awal bulan, pada saat akan membuat laporan kegiatan pada bulan sebelumnya. Atribut Balance sering sekali berubah, yaitu setiap kali ada pesanan ke *supplier* tersebut.
- Pencarian data supplier dan product sangat sering dilakukan berdasarkan nama.
- Untuk memenuhi kebutuhan dari pihak eksekutif, query untuk mendapatkan total harga terhutang dari seluruh pesanan yang belum diterima, baik per *supplier* maupun per tanggal pesanan, sering dijalankan.

denormalisasi
horizontal splitting
vertical splitting
index tuning
materialized view

Tugas Anda:

Berikan usulan schema tuning untuk meningkatkan kinerja sistem basis data di atas, termasuk kemungkinan untuk memanfaatkan materialized view dan kebutuhan akan indeks. Indeks yang tidak perlu dituliskan adalah indeks yang berkaitan dengan primary key. Untuk setiap usulan:

- Tuliskan jenis tuning-nya dan jelaskan.
- Tuliskan skema baru yang dihasilkan (jika ada).
- Tuliskan kebutuhan proses tambahan di basis data untuk tetap menjaga konsistensi data (jika ada).
- Jika perlu index, tuliskan dengan jelas relasi dan atribut indeks dan jenis indeks yang dibutuhkan.
- Untuk setiap materialized view yang Anda usulkan, tuliskan perintah create view untuk mendapatkannya.

Di akhir jawaban, tuliskan kembali seluruh skema basis data final yang Anda usulkan, termasuk skema materialized view (jika ada).

① kasus "90% akses terhadap produk selalu membutuhkan nama kategori dari produk tersebut"
(a) jenis tuning denormalisasi
karena ukuran dari PRODUCT_CATEGORY sangat kecil (100 records), performance cost dari operasi JOIN menjadi unnecessary, lebih baik menambah redundansi di tabel PRODUCT dengan menambah CNAME

1. Kasus: "90% akses terhadap produk selalu membutuhkan nama kategori dari produk tersebut"
   (a) Jenis tuning: **denormalisasi** dengan redundant column
   Karena ukuran PRODUCT_CATEGORY sangat kecil (100 records), operasi JOIN untuk mendapatkan CNAME dari tabel PRODUCT_CATEGORY menjadi unnecesary, lebih efisien untuk menambahkan atribut CNAME di tabel PRODUCT.
   (b) Schema baru:
   PRODUCT(PID, PNAME, *CID*, CNAME, WEIGHT)
   (c) Kebutuhan proses tambahan di basis data untuk menjaga konsistensi data:
   Diperlukan trigger untuk memastikan konsistensi CNAME pada PRODUCT_CATEGORY dan PRODUCT. Trigger aktif ketika terjadi ~~penambahan atau~~ perubahan CNAME pada tabel PRODUCT_CATEGORY untuk menyelaraskannya dengan tabel PRODUCT. → untuk setiap CID yang berubah CNAMEnya

2. Kasus: "75% akses terhadap supply hanya mengakses data pesanan dari supplier yang belum diterima saja. Setiap saat, rata-rata hanya ada 2000 hingga 3000 pesanan dari supplier yang belum diterima. Pesanan yang belum diterima ditandai dengan atribut RECEIVEDATE pada suatu baris data pesanan di tabel SUPPLY bernilai NULL" → jumlah yang sering diakses << keseluruhan tabel
   (a) Jenis tuning: **horizontal splitting** HORIZONTAL SPLITTING ✓
   Karena sebagian besar akses terhadap tabel SUPPLY hanya mengakses data pesanan dari supplier yang belum diterima, maka tabel dapat dibagi menjadi dua berdasarkan nilai atribut RECEIVEDATE. RECEIVED_SUPPLY berisi data SUPPLY dengan RECEIVEDATE yang NOT NULL, sedangkan UNRECEIVED_SUPPLY berisi data SUPPLY dengan RECEIVEDATE yang NULL.
   (b) Schema baru:
   RECEIVED_SUPPLY(ORDERDATE, *PID*, *SID*, QUANTITY, PRICE, RECEIVEDATE)
   UNRECEIVED_SUPPLY(ORDERDATE, *PID*, *SID*, QUANTITY, PRICE, RECEIVEDATE)
   (c) Kebutuhan proses tambahan di basis data untuk menjaga konsistensi data: disjoin
   Data pada RECEIVED_SUPPLY harus disjoint dengan data UNRECEIVED_SUPPLY. Nilai RECEIVEDATE pada tabel RECEIVED_SUPPLY harus NOT NULL, sedangkan data RECEIVEDATE pada tabel UNRECEIVED_SUPPLY harus NULL. Jika pesanan dari supplier yang belum diterima pada tabel UNRECEIVED_SUPPLY berubah status menjadi diterima, baris data tersebut harus dipindahkan dari tabel UNRECEIVED_SUPPLY ke RECEIVED_SUPPLY.

   indexing lebih cocok untuk searching

3. Kasus: "Atribut Alamat, Kota, dan Rating dari supplier hanya diakses setiap awal bulan, pada saat akan membuat laporan kegiatan pada bulan sebelumnya. Atribut Balance sering sekali berubah, yaitu setiap kali ada pesanan ke supplier tersebut."
   (a) Jenis tuning: **vertical splitting**
   Atribut Alamat, Kota, dan Rating dari supplier hanya diakses setiap awal bulan. Pada saat akan membuat laporan kegiatan pada bulan sebelumnya, atribut Balance sering berubah saat setiap kali ada pesanan ke supplier tersebut sehingga perlu dilakukan vertical splitting pada tabel SUPPLIER.

(b) Schema baru:
SUPPLIER = (SID, SNAME, SADDRESS, SCITY, RATING)
SUPPLIER_ BALANCE= (SID, BALANCE)
FK: SUPPLIER_BALANCE (SID) → SUPPLIER (SID)

(c) Kebutuhan proses tambahan di basis data untuk menjaga konsistensi data:
Karena ada foreign key reference dari tabel SUPPLIER_BALANCE ke SUPPLIER_MAIN, SID pada tabel SUPPLIER_BALANCE merupakan salah satu SID pada SUPPLIER. Lalu, semua SID yang muncul pada tabel SUPPLIER juga muncul di tabel SUPPLIER_BALANCE.

4. Kasus: "Pencarian data supplier dan product sangat sering dilakukan berdasarkan nama."
Jenis tuning: **B+-tree indexing**
   (a) Menambahkan secondary index pada relasi SUPPLIER pada atribut SNAME, karena pencarian data supplier sering didasarkan pada nama.
   (b) Menambahkan secondary index pada relasi PRODUCT pada atribut SNAME, karena pencarian data product sering didasarkan pada nama.

hash : equality (sama dengan)
B+-tree : general

5. Kasus: "Untuk memenuhi kebutuhan dari pihak eksekutif, query untuk mendapatkan total harga terhutang dari seluruh pesanan yang belum diterima, baik per supplier maupun per tanggal pesanan, sering dijalankan."
Jenis tuning: **materialized view**

```
CREATE_VIEW TotalHutang AS
SELECT s.SID, su.ORDERDATE, SUM(su.QUANTITY*su.PRICE)
FROM UNRECEIVED_SUPPLY su JOIN SUPPLIER s ON su.SID = s.SID
GROUP BY s.SID, su.ORDERDATE
```

sebaiknya dipecah, tapi bisa juga disatukan

Skema Basis Data Final:
PRODUCT(PID, PNAME, *CID*, CNAME, WEIGHT)
RECEIVED_SUPPLY(ORDERDATE, *PID*, *SID*, QUANTITY, PRICE, RECEIVEDATE)
UNRECEIVED_SUPPLY(ORDERDATE, *PID*, *SID*, QUANTITY, PRICE, RECEIVEDATE)
SUPPLIER = (SID, SNAME, SADDRESS, SCITY, RATING)
SUPPLIER_ BALANCE= (SID, BALANCE)