QUIZ

annaufal afif seno - 2023071054

1. Jelaskan beberapa teknik yang dapat digunakan untuk meningkatkan kinerja sebuah sistem basis data!

Meningkatkan kinerja sistem basis data adalah langkah penting untuk memastikan akses data yang cepat dan efisien. Berikut beberapa teknik yang dapat digunakan untuk meningkatkan kinerja sistem basis data:

Indexing
Indeks adalah struktur data yang meningkatkan kecepatan pengambilan data pada tabel. Dengan menambahkan indeks pada kolom yang sering digunakan dalam klausa WHERE, JOIN, atau ORDER BY, sistem basis data dapat menemukan data dengan lebih cepat. Namun, terlalu banyak indeks juga dapat memperlambat operasi INSERT, UPDATE, dan DELETE, karena indeks harus diperbarui setiap kali data berubah.

Denormalisasi
Denormalisasi adalah proses pengurangan tingkat normalisasi dalam desain basis data untuk mengurangi jumlah join yang dibutuhkan untuk mengambil data. Meskipun ini dapat mempercepat query, denormalisasi juga dapat menyebabkan redundansi data, yang memerlukan manajemen ekstra.

Partitioning
Membagi tabel besar menjadi beberapa bagian yang lebih kecil (partisi) berdasarkan kriteria tertentu, seperti rentang tanggal atau nilai numerik. Ini memungkinkan sistem untuk memproses subset data yang lebih kecil dan mempercepat query, terutama dalam basis data dengan skala besar.

Query Optimization
Mengoptimalkan perintah SQL adalah cara penting untuk meningkatkan kinerja. Beberapa langkah yang bisa diambil termasuk:

Menggunakan join yang efisien (misalnya, menggunakan INNER JOIN alih-alih OUTER JOIN jika memungkinkan).
Menghindari penggunaan subquery yang tidak perlu.
Menggunakan klausa LIMIT untuk membatasi jumlah hasil yang diambil dari basis data.
Caching
Penyimpanan hasil query ke dalam memori (cache) dapat mengurangi waktu akses data yang sering diminta. Dengan caching, query yang sama tidak perlu dijalankan berulang kali, sehingga mengurangi beban pada server basis data.

Connection Pooling
Penggunaan connection pooling dapat meningkatkan kinerja dengan mengelola koneksi basis data secara lebih efisien. Daripada membuka dan menutup koneksi setiap kali ada permintaan, connection pooling memungkinkan aplikasi untuk menggunakan kembali koneksi yang ada, yang mengurangi overhead.

Sharding
Sharding adalah teknik membagi basis data menjadi beberapa bagian atau shard, yang masing-masing beroperasi secara mandiri pada server yang berbeda. Ini memungkinkan distribusi beban kerja dan peningkatan kapasitas basis data secara horisontal.

Materialized Views
Materialized view adalah versi precomputed dari query yang sering digunakan. Dengan menyimpan hasil query sebelumnya, sistem dapat menghindari eksekusi ulang query yang kompleks, sehingga mengurangi waktu respons.

Load Balancing
Distribusi beban kerja basis data ke beberapa server melalui load balancing memungkinkan kinerja yang lebih baik, terutama untuk basis data yang sering diakses secara bersamaan oleh banyak pengguna.

Vertical and Horizontal Scaling

Vertical Scaling: Menambah kapasitas server dengan meningkatkan hardware, seperti menambah CPU, RAM, atau storage.
Horizontal Scaling: Menambah jumlah server untuk berbagi beban kerja, yang cocok untuk basis data besar dan terdistribusi.
Teknik-teknik ini bisa diterapkan secara kombinasi tergantung pada kebutuhan dan karakteristik sistem basis data yang digunakan.

2. Apa yang dimaksud dengan indexing? Kapan sebaiknya kita menggunakan indeks dalam sebuah tabel?

a. Indexing dalam basis data adalah teknik yang digunakan untuk mempercepat proses pencarian dan pengambilan data dari tabel. Indeks adalah struktur data khusus yang dibuat berdasarkan satu atau lebih kolom dari tabel, yang memungkinkan sistem basis data menemukan data dengan lebih cepat, tanpa harus memindai seluruh tabel.

Cara Kerja Indexing:
Indeks bekerja seperti indeks di buku. Jika Anda ingin mencari suatu topik di buku, Anda tidak perlu membaca setiap halaman, melainkan bisa langsung menuju indeks di bagian belakang buku yang memberi petunjuk di mana topik tersebut berada. Dalam basis data, indeks memetakan nilai kolom ke lokasi penyimpanan data, sehingga saat query dijalankan, pencarian data bisa dilakukan secara lebih efisien.

Jenis-Jenis Indeks:
Indeks B-Tree (Balanced Tree)
Ini adalah jenis indeks yang paling umum. Struktur B-Tree memungkinkan pencarian, penyisipan, penghapusan, dan pembaruan data dilakukan dalam waktu yang efisien. B-Tree mempertahankan data dalam bentuk yang terurut, sehingga memudahkan pencarian.

Indeks Hash

Indeks hash menggunakan fungsi hash untuk menentukan lokasi penyimpanan data. Indeks ini sangat efisien untuk pencarian nilai eksak, tetapi kurang efektif untuk pencarian rentang (misalnya, BETWEEN atau >, <).

Unique Index
Indeks ini memastikan bahwa nilai di kolom yang diindeks unik di seluruh tabel. Ini digunakan untuk memastikan bahwa tidak ada duplikasi nilai pada kolom tertentu.

Clustered Index
Indeks ini mengatur penyimpanan fisik data di dalam tabel. Hanya bisa ada satu clustered index per tabel, karena indeks ini menentukan urutan penyimpanan data. Data di tabel disimpan dalam urutan sesuai dengan clustered index.

Non-clustered Index
Ini adalah indeks yang terpisah dari tabel data. Indeks ini berisi pointer ke lokasi data dalam tabel, dan memungkinkan beberapa non-clustered index di tabel yang sama.

Manfaat Indexing:
Mempercepat Query: Indeks mempercepat operasi query seperti pencarian data, SELECT, JOIN, dan ORDER BY.
Mengurangi I/O (Input/Output): Dengan indeks, sistem tidak perlu membaca seluruh tabel, sehingga mengurangi jumlah akses disk.
Kekurangan Indexing:
Overhead Penyimpanan: Setiap indeks memerlukan ruang penyimpanan tambahan.
Waktu Update yang Lebih Lambat: Operasi INSERT, UPDATE, dan DELETE bisa lebih lambat karena setiap perubahan data memerlukan pembaruan pada indeks.
Penggunaan Berlebihan: Terlalu banyak indeks dapat memperlambat sistem, terutama untuk basis data yang sering diperbarui.
Jadi, indexing adalah trade-off antara mempercepat pencarian data dan menambah overhead pada penyimpanan serta kecepatan update data.

b. Penggunaan indeks dalam sebuah tabel basis data sangat penting untuk meningkatkan kinerja query, tetapi tidak selalu harus diterapkan secara sembarangan karena indeks juga membawa overhead tambahan. Berikut beberapa situasi di mana sebaiknya Anda menggunakan indeks:

1. Kolom yang Sering Digunakan di Query

Indeks sangat berguna pada kolom yang sering digunakan dalam klausa WHERE, JOIN, atau GROUP BY. Indeks membantu mempercepat pencarian data berdasarkan kondisi ini.

Contoh: Jika Anda sering menanyakan data pelanggan berdasarkan customer_id, menambahkan indeks pada kolom customer_id akan mempercepat query tersebut.

2. Kolom yang Digunakan untuk Sorting

Jika Anda sering melakukan operasi ORDER BY pada suatu kolom, indeks di kolom tersebut bisa mempercepat proses pengurutan.

Contoh: Query yang sering melakukan sorting berdasarkan created_at (tanggal pembuatan), seperti SELECT * FROM orders ORDER BY created_at, akan lebih cepat jika kolom created_at diindeks.

3. Kolom yang Sering Digunakan dalam Join

Pada query yang menggabungkan dua atau lebih tabel (join), indeks pada kolom yang digunakan sebagai kunci join dapat mempercepat proses penggabungan data.

Contoh: Jika Anda sering melakukan join antara tabel orders dan customers menggunakan customer_id, sebaiknya indeks dibuat pada kolom customer_id di kedua tabel.

4. Kolom dengan Nilai Unik

Kolom yang memiliki nilai unik, seperti kunci utama (primary key), hampir selalu diindeks secara otomatis oleh sistem basis data. Anda juga bisa menambahkan indeks unik pada kolom lain untuk memastikan keunikan nilai, serta mempercepat pencarian berdasarkan kolom tersebut.

Contoh: email di tabel users bisa diindeks dengan unique index karena biasanya setiap pengguna memiliki email yang unik.

5. Kolom dengan Selektivitas Tinggi

Indeks paling efektif pada kolom yang memiliki selektivitas tinggi, yaitu kolom dengan banyak nilai unik. Selektivitas tinggi memungkinkan indeks untuk mempersempit hasil pencarian dengan lebih cepat.

Contoh: Kolom seperti social_security_number di tabel pegawai biasanya memiliki selektivitas tinggi karena setiap orang memiliki nomor unik, sehingga akan efektif diindeks.

6. Tabel yang Sering Dibaca daripada Di-update

Jika tabel lebih sering dibaca daripada diperbarui (read-heavy table), indeks sangat membantu. Sebaliknya, jika tabel lebih sering di-update (write-heavy table), menambahkan indeks dapat memperlambat operasi update.

Contoh: Tabel log atau tabel pelaporan yang tidak sering diperbarui bisa mendapatkan manfaat besar dari indeks karena query baca (read) lebih dominan.

7. Kolom yang Digunakan untuk Filter Data

Jika Anda sering melakukan pencarian berdasarkan rentang nilai (misalnya, tanggal, angka, atau rentang harga), indeks pada kolom tersebut akan mempercepat pencarian.

Contoh: SELECT * FROM transactions WHERE amount BETWEEN 1000 AND 5000 akan lebih cepat jika kolom amount diindeks.

Situasi Di Mana Indeks Tidak Sebaiknya Digunakan:

Kolom dengan Banyak Nilai Duplikat (Selektivitas Rendah) Indeks tidak efisien pada kolom yang memiliki banyak nilai duplikat, seperti kolom boolean (misalnya, is_active dengan nilai TRUE atau FALSE). Dalam kasus seperti ini, indeks tidak akan banyak membantu mempercepat query.

Contoh: Kolom gender atau status yang hanya memiliki beberapa nilai (misalnya M/F atau active/inactive) mungkin tidak ideal untuk diindeks.
Tabel Kecil Indeks tidak terlalu berguna pada tabel kecil, karena dalam tabel kecil, operasi pencarian yang menggunakan full table scan (membaca seluruh tabel) tidak akan memakan banyak waktu. Indeks pada tabel kecil justru bisa menambah overhead.

Contoh: Tabel dengan hanya 100 baris data tidak memerlukan indeks.
Kolom yang Sering Di-update Kolom yang sering berubah (misalnya kolom yang di-update setiap kali ada perubahan status transaksi) bisa menambah overhead karena setiap perubahan akan memicu pembaruan indeks. Jadi, hindari indeks pada kolom yang sering di-update.

Contoh: Kolom seperti last_login_time atau stock_quantity mungkin tidak cocok untuk diindeks jika sering diperbarui.