Bahan Ajar Pemicu 1

Topik : Optimasi dan Manipulasi Query SQL

Nama : Dapunta Adyapaksi Ratyanasja
NRP : 5025231187
Kelas : Manajemen Basis Data E

Index

Penjelasan
Index adalah struktur data untuk membantu pencarian data dalam tabel secara cepat. Ini memungkinkan DBMS menemukan baris data tanpa harus melakukan full scan.

Manfaat
- Meningkatkan performa query menggunakan klausa seperti WHERE, ORDER BY, dan GROUP BY.
- Mengurangi beban I/O karena tidak perlu membaca seluruh data dalam tabel.

Proses Pencarian dengan Index
Ketika sebuah query dijalankan, DBMS akan memeriksa apakah ada index yang sesuai untuk kolom yang digunakan dalam kondisi pencarian. Jika ada, DBMS akan melompat langsung ke lokasi data yang relevan, sehingga waktu akses menjadi lebih cepat.

Jenis-Jenis Index
- Primary Index
  Dibuat secara otomatis ketika mendefinisikan primary key pada tabel.
- Unique Index
  Memastikan bahwa nilai dalam kolom tersebut tidak ada yang duplikat.
- Composite Index
  Index yang melibatkan lebih dari satu kolom, berguna ketika query sering menggunakan kombinasi kolom untuk filtering.
- Clustered dan Non-Clustered Index
  - Clustered Index : Menentukan susunan fisik data dalam tabel (hanya satu per tabel).
  - Non-Clustered Index : Index yang disimpan secara terpisah dari data fisik tabel, memungkinkan beberapa index per tabel.

Contoh Penerapan Index

Contoh 1 : Pencarian berdasarkan department_id

Table karyawan

```
CREATE TABLE karyawan (
   id INT PRIMARY KEY,
   name VARCHAR(100),
   department_id INT
);
```

Buat index

```
CREATE INDEX idx_department ON karyawan(department_id);
```

Penggunaan index

```
SELECT * FROM karyawan WHERE department_id = 2;
```

Kesimpulan

Dengan adanya index idx_department, DBMS dapat langsung mencari baris yang memiliki department_id = 2 tanpa melakukan full scan pada table karyawan.

Contoh 2 : Filter berdasarkan tahun pada kolom tanggal

Table orders

```
CREATE TABLE orders (
   id_orders VARCHAR(15) PRIMARY KEY,
   name VARCHAR(100),
   order_date DATE
);
```

Query tidak optimal

```
SELECT * FROM orders WHERE YEAR(order_date) = 2020;
```

Fungsi YEAR(order_date) diterapkan pada setiap baris data sehingga menyebabkan full table scan yang bisa memperlambat performa.

Query optimal

```
CREATE INDEX idx_order_date ON orders(order_date);

SELECT * FROM orders
   WHERE order_date >= '2020-01-01'
   AND order_date < '2021-01-01';
```

Dengan menggunakan kondisi range pada kolom order_date secara langsung, query memungkinkan DBMS untuk memanfaatkan idx_order_date.

Eksekusi query menjadi lebih cepat karena pencarian data hanya melibatkan index lookup, bukan pemrosesan setiap baris dengan fungsi.

JOIN

Penjelasan
JOIN adalah operasi dalam SQL yang digunakan untuk menggabungkan data dari dua atau lebih tabel berdasarkan kolom yang memiliki relasi (misalnya, foreign key). Dengan JOIN, data yang tersebar di tabel-tabel berbeda dapat ditampilkan dalam satu query sehingga memudahkan analisis secara komprehensif.

Manfaat
- Menggabungkan data yang terkait dari beberapa tabel.
- Meminimalkan redundansi data dengan menyimpan informasi secara terpisah.
- Memudahkan pembuatan laporan dan analisis data yang memerlukan informasi dari berbagai sumber tabel.

Penggunaan JOIN
JOIN digunakan untuk menghubungkan tabel-tabel dengan kondisi tertentu. Kondisi ini biasanya berupa kecocokan nilai antara kolom yang berelasi pada masing-masing tabel.

Jenis-Jenis JOIN

1. INNER JOIN

Penjelasan
- Mengembalikan baris yang memiliki kecocokan di kedua tabel yang di-join.
- Jika tidak ada kecocokan, baris tersebut tidak akan muncul pada hasil query.

Contoh Penerapan
```
SELECT a.*, b.*
FROM customers a
INNER JOIN orders b ON a.customer_id = b.customer_id;
```

Catatan
- Hanya menampilkan data yang terdapat di kedua tabel (keduanya harus memiliki nilai yang cocok).
- Penggunaan "JOIN" tanpa prefiks tambahan secara otomatis dianggap sebagai INNER JOIN.

2. LEFT JOIN (Left Outer Join)

Penjelasan
- Mengembalikan semua baris dari tabel sebelah kiri (tabel pertama), dan hanya baris yang cocok dari tabel sebelah kanan.
- Jika tidak ada kecocokan di tabel kanan, kolom dari tabel tersebut akan berisi nilai NULL.

Contoh Penerapan
```
SELECT a.*, b.*
FROM customers a
LEFT JOIN orders b ON a.customer_id = b.customer_id;
```

Catatan
- Sangat berguna ketika ingin menampilkan semua data dari tabel utama, meskipun beberapa baris tidak memiliki data terkait di tabel kedua.