MATERI IV
VIEW DAN INDEX

A. Tujuan

B. Dasar Teori

1. View

View adalah object *database* yang tidak berisi data apapun. View berisi query dari satu atau beberapa tabel yang disimpan di server sebagai objek dan direferensikan seperti tabel. Akan tetapi hasil view tidak menyimpan hasil query, hanya definisi atau sintaks query yang disimpan di dalam *database*. Ketika data pada tabel yang direferensi oleh view berubah, maka view juga akan menghasilkan perubahan data yang ditampilkan. Penggunaan view sama halnya penggunaan table, dimana hasil query pada view dapat menggunakan klausa JOIN dari table lainnya pada bagian FROM.

Sebagai contoh jika ingin menampilkan kembali Nip, nama dosen, mata kuliah yang diampu serta menampilkan dosen yang tidak mengampu mata kuliah, maka akan lebih baik jika dibuat view dan mengeksekusi view daripada menuliskan query secara berulang-ulang. Sintaks dasar pembuatan view adalah sebagai berikut :

```
CREATE
    [OR REPLACE]
    VIEW view_name [(column_list)]
    AS select_statement
```

Sebagai contoh akan dibuat view dengan nama "vw_ips" untuk menampilkan Indeks Prestasi mahasiswa prodi Teknik Informatika (kode prodi = "0304") dengan persamaan untuk menghitung Indeks Prestasi adalah sebagai berikut :

$$IPs = \frac{\sum(SKS * Mutu\ Nilai}{jumlah\ SKS\ ditempuh}$$

Sintak view "vw_ips" yang dibuat adalah sebagai berikut :

```
CREATE OR REPLACE VIEW vw_ips
AS
SELECT m.Nim as NIM,m.Nama_Mhs as NAMA,
SUM(CASE
        WHEN k.Nilai= 'A' THEN 4*mk.sks
        WHEN k.Nilai= 'B+' THEN 3.25*mk.sks
        WHEN k.Nilai= 'B' THEN 3*mk.sks
        WHEN k.Nilai= 'C+' THEN 2.25*mk.sks
        WHEN k.Nilai= 'C' THEN 2*mk.sks
        WHEN k.Nilai= 'D' THEN 1*mk.sks
ELSE
        0*mk.sks
END)/SUM(mk.sks)
AS IP_SEMESTER
FROM krs as k
JOIN mahasiswa AS m ON k.Nim = m.Nim
JOIN matakuliah AS mk ON k.Kode_MK = mk.Kode_MK
WHERE k.Nim LIKE "%0304%"
GROUP BY m.Nim
```

Untuk mengesekusi view “vw_ips” dilakukan menggunakan query layaknya menampilkan data dari tabel, yaitu : *SELECT * FROM vw_ips*

Sebagaimana disebutkan, jika data pada tabel yang direferensi view berubah, maka view juga akan menampilkan perubahan data. Untuk membuktikannya, ubahlah nilai beberapa record dari tabel “krs”, kemudian eksekusi kembali view “vw_ips”.

2. Index di Mysql

Indeks di dalam tabel digunakan untuk menemukan baris dengan nilai kolom spesifik dengan cepat. Tanpa indeks, MySQL harus memulai dengan baris pertama dan kemudian membaca seluruh tabel untuk menemukan baris yang relevan. Semakin besar tabel, semakin banyak waktu yang dibutuhkan untuk membaca isi tabel. Jika tabel memiliki indeks untuk kolom yang dimaksud, MySQL dapat dengan cepat menentukan posisi yang dicari di tengah file data tanpa harus melihat semua data. Ini jauh lebih cepat daripada membaca setiap baris secara berurutan.

Index adalah struktur data yang dapat meningkatkan kecepatan operasi pada tabel. Index dapat dibuat menggunakan satu atau beberapa kolom. Pertimbangan kolom yang akan digunakan pada index adalah kolom atau *field* tersebut digunakan sebagai kriteria dalam proses searching data, dalam hal ini *field* yang digunakan dalam klausa WHERE pada query. Jenis index pada MySQL adalah INDEX, UNIQUE, dan FULLTEXT. INDEX adalah index sederhana yang digunakan untuk mempercepat proses pembacaan data dari suatu tabel, UNIQUE index digunakan untuk memastikan bahwa tidak ada nilai yang sama

pada suatu tabel. Perbedaaanya dengan PRIMARY KEY adalah, jika di dalam PRIMARY KEY tidak diijinkan ada data yang bernilai NULL, sedangkan di dalam UNIQUE index nilai NULL diperbolehkan. Sedangkan FULLTEXT index digunakan untuk pencarian *full-text searching*, yaitu memungkinkan pencarian tertentu dalam tabel dengan cara melakukan perbandingan string. Struktur sintak pembuatan index di Mysql adalah sebagai berikut :

```
CREATE [UNIQUE|FULLTEXT|SPATIAL] INDEX index_name
    [index_type]
    ON tbl_name (index_col_name,...)
    [index_type]

index_col_name:
    col_name [(length)] [ASC | DESC]

index_type:
    USING {BTREE | HASH}
```

Pengunaan index umumnya digunakan untuk mempercepat proses pencarian data di tabel. Untuk membuktikannya, ikuti langkah-langkah berikut :

1) Tulis sintaks query INNER JOIN untuk menampilkan NIP,Nama Dosen yang mengampu Mata Kuliah dengan jumlah mahasiswa lebih dari 3 mahasiswa sebagai berikut :

```
SELECT k.Kode_MK,mk.Nama_MK,d.Nip,d.Nama_Dosen, COUNT(k.Kode_MK) as jml_mahasiswa
FROM krs as k
JOIN matakuliah as mk ON k.Kode_MK=mk.Kode_MK
JOIN jadwal as j ON mk.Kode_MK=j.Kode_MK
JOIN dosen as d ON j.Nip=d.Nip
GROUP BY k.Kode_MK
HAVING jml_mahasiswa >= 3
```

2) Eksekusi query pada nomor 1, kemudian perhatikan *explain* hasil eksekusi query tersebut seperti ditunjukkan pada gambar berikut:

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	j	ALL	(Null)	(Null)	(Null)	(Null)	8	Using temporary; Using files
1	SIMPLE	d	eq_ref	PRIMARY	PRIMARY	38	func	1	Using where
1	SIMPLE	mk	eq_ref	PRIMARY	PRIMARY	32	func	1	Using where
1	SIMPLE	k	ALL	(Null)	(Null)	(Null)	(Null)	15	Using where; Using join buff

Pada kolom table dan type, untuk table j (jadwal) dan k (krs) terlihat sebelum adanya index, seluruh data pada kedua tabel tersebut akan ditelusuri.

3) Selanjutnya buat index pada tabel jadwal untuk kolom Kode_MK dan NIP, dan tabel KRS untuk kolom Kode_MK dan NIM. Sintak pembuatan index untuk kedua tabel tersebut adalah sebagai berikut :

```
CREATE INDEX jadwal_idx_kode_mk
ON jadwal(Kode_MK,NIP) USING BTREE;

CREATE INDEX krs_idx_kode_mk
ON krs(Kode_MK,NIM) USING BTREE;
```

4) Eksekusi kembali query pada nomor 1, kemudian perhatikan *explain* hasil eksekusi query sebagaimana ditunjukkan pada gambar berikut :

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	j	index	(Null)	nama_idx_kod	28	(Null)	8	Using index; Using tempora
1	SIMPLE	d	eq_ref	PRIMARY	PRIMARY	38	func	1	Using where
1	SIMPLE	mk	eq_ref	PRIMARY	PRIMARY	32	func	1	Using where
1	SIMPLE	k	index	(Null)	krs_idx_kode_n	25	(Null)	15	Using where; Using index; U

Setelah pembuatan index pada kedua tabel yaitu jadwal dan krs, maka hasil pembacaan pada kedua tabel tersebut tidak lagi pada seluruh data, akan tetapi sesuai index berdasarkan *field* yang terlibat di nilai klausa WHERE.

Berdasarkan hasil eksekusi query di atas tidak menunjukkan perbedaan yang signifikan antara pengunaan index dan tanpa penggunaan index. Hal tersebut disebabkan jumlah data pada tabel "matakuliah", "jadwal", dan "dosen" yang tidak besar. Untuk membuktikan hasil yang signifikan dalam penggunaan index, ikuti langkah-langkah berikut :

1) Langkah pertama adalah membuat table dan men-*generate* record pada tabel, sehingga terdapat sekitar 7 juta record. Query yang digunakan adalah sebagai berikut :

```
create table big_table as
select @baris := @baris+1 as baris
from ( select @baris := 0 ) x
join ( select 1 kolom from information_schema.tables ) a
join ( select 1 kolom from information_schema.tables ) b
join ( select 1 kolom from information_schema.tables ) c
```

Tabel "big_table" yang dibuat belum memiliki index. Dibutuhkan sekitar 2 menit untuk men-*generate* data pada tabel "big_table".

2) Jalankan query untuk menampilkan jumlah record pada table "big_table". Query yang digunakan adalah sebagai berikut :

```
SELECT COUNT(*) FROM big_table
```

Hasil eksekusi query di atas ditunjukkan pada gambar berikut :

```
SELECT COUNT(*)  FROM big_table
> OK
> Time: 3.744s
```

Berdasarkan hasil eksekusi query untuk menampilkan jumlah record pada table "big_table" membutuhkan waktu 3 detik lebih. Selanjutnya akan ditampilkan *execution plan* query COUNT(*) pada table "big_table" menggunakan query sebagai berikut :

```
explain
select COUNT(*) from big_table ;
```

Hasil *execution plan* perintah query "COUNT(*)" ditunjukkan pada gambar berikut :

id	select_type	table	type	possible_keys	key	key_len	ref	rows
1	SIMPLE	big_table	ALL	(Null)	(Null)	(Null)	(Null)	7173708

Sesuai hasil *execution plan* terlihat bahwa MySql melakukan *full scan* atau membaca seluruh data pada tabel "big_table". Hal tersebut ditandai pada parameter "*type=ALL*".

3) Selanjutnya, jalankan query yang menggunakan klausa WHERE seperti query berikut :

```
SELECT * FROM big_table
WHERE baris="7188048"
```

Eksekusi query di atas membutuhkan waktu sekitar 3.742 detik. Selanjutnya hasil *execution plan* ditunjukkan pada gambar berikut :

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	big_table	ALL	(Null)	(Null)	(Null)	(Null)	7173708	Using where

Berdasarkan hasil *execution plan*, MySql melakukan *full scan* yang ditandai pada parameter "*type=ALL"* dan *"rows=7173708"*. Artinya MySql melakukan proses pembacaan 7.173.708 baris pada table "big_table" untuk menemukan satu baris data pada *field* "baris" dengan nilai "7188048".

4) Berikutnya untuk meningkatkan performa table "big_table", dibuat index berdasarkan *field* "baris". Query pembuatan index adalah sebagai berikut :

```
CREATE INDEX big_table_pk on big_table(baris) USING BTREE;
```

5) Jalankan lagi query pada langkah 2. Setelah adanya index pada table "big_table", waktu eksekusi query menjadi 3.299 detik. Kemudian hasil *execution plan* ditunjukkan pada gambar berikut :

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	big_table	index	(Null)	big_table_pk	9	(Null)	7173708	Using index

Terlihat bahwa MySqk tidak lagi menggunakan *full scan table*, akan tetapi menggunakan penelusuran berdasarkan *index* yang ditandai pada parameter "*type=index"*.

6) Jalankan kembali query pada langkah ke 3 untuk membuktikan penggunaan index, apakah MySql masih melakukan melakukan proses pembacaan 7.173.708 baris pada table "big_table" untuk menemukan satu baris data pada *field* "baris" dengan nilai "7188048".

```
SELECT * FROM big_table
WHERE baris="7188048"
> OK
> Time: 0.002s
```

Berdasarkan hasil di atas menunjukkan adanya peningkatan performa pada tabel "big_table". Waktu eksekusi sebelum menggunakan index adalah 3.742 detik, sedangkan waktu eksekusi setelah menggunakan index hanya 0.002 detik. Hasil *execution plan* query ditunjukkan pada gambar berikut:

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	big_table	ref	big_table_pk	big_table_pk	9	const	1	Using index

C. Evaluasi dan Pertanyaan

1. Buatlah view di dalamnya terdapat perintah JOIN untuk menampilkan NIM, Nama Mahasiswa dan jumlah SKS yang ditempuh !
2. Buatlah view di dalamnya terdapat perintah JOIN untuk menampilkan Nip, Nama Dosen dan jumlah SKS yang diampu !
3. Jelaskan cara kerja index dengan karakteristik B-TREE dan HASH ! berikan contoh proses pembacaan data pada index berkarakteristik B-TREE dan HASH.