Basis Data Lanjut

Tool

- MySql server
- Command Prompt

BACKUP RESTORE

- Backup
  - Rumus umum

```
mysqldump [database] > [lokasi dan nama file] -u root -p
```

  - Contoh

```
mysqldump basis_data_lanjut > E:\file.sql -u root -p
```

- Restore
  - Rumus umum

```
mysql [database] < [lokasi dan nama file] -u root -p
```

  - Contoh

```
mysql basis_data_lanjut < E:\file.sql -u root -p
```

Query Select

- Rumus umum sederhana
  - `SELECT [field]|[eskpresi] FROM [Tabel] WHERE [Kondisi] GROUP BY [Field] ORDER BY [Field] [ASC | DESC]`
- Contoh
  - `SELECT * FROM nasabah WHERE nama like '%jono%' ORDER BY alamat`
  - `SELECT no_nasabah, nama, alamat FROM nasabah WHERE nama ='jono' ORDER BY alamat DESC`
  - `SELECT alamat, count(no_nasabah) FROM nasabah GROUP BY alamat`
  - `SELECT nasabah.*, IF(jk=1,'Laki-Laki','Perempuan') AS gender FROM nasabah`

Nested Query (query bersarang)

- Ada kalanya klausa SELECT tidak diikuti dengan nama kolom, melainkan dengan *query* lain.

```
SELECT (SELECT count(*) FROM perpus_anggota WHERE jur='TI') as
jmti, (SELECT count(*) FROM perpus_anggota WHERE jur='Man') as
jmman, (SELECT count(*) FROM perpus_anggota WHERE jur='Akun') as
jmakun
```

- Ada kalanya klausa FROM tidak diikuti oleh nama tabel melainkan diikuti dengan sebuah *query* lain.

```
SELECT alamat, SUM(ti) AS t_ti, SUM(man) AS t_man, SUM(akun) AS
t_akun FROM (SELECT alamat, IF(jur='TI',1,0) AS ti,
IF(jur='Man',1,0) AS man, IF(jur='Akun',1,0) AS akun FROM
perpus_anggota) AS tb_ang GROUP BY alamat
```

- Ada kalanya juga salah satu ruas persamaan dalam klausa WHERE berisi *query* lain.

```
SELECT * FROM simpin_nasabah WHERE id_nas NOT IN (SELECT DISTINCT
no_nas FROM simpin_pinjam)
```

Join
(peggabungan/ menghubungkan/ relasi)

- Digunakan untuk menampilkan kolom tertentu dari satu tabel berdasarkan kolom tabel lainnya
- Memerlukan field (kolom) kunci
- Rumus umum

```
SELECT [daftar_kolom] FROM [tabel_1] [LEFT|RIGHT|INNER] JOIN
[tabel_2] ON [tabel_1].[field_kunci]=[tabel_2].[field_kunci]
```

- Contoh

```
SELECT perpus_pinjam.*,nama FROM perpus_pinjam LEFT JOIN
perpus_anggota ON
perpus_pinjam.id_anggota=perpus_anggota.id_anggota
```

Update Join

- Mengubah data dari suatu tabel berdasarkan tabel lain
- Rumus umum

```
UPDATE [tabel_update] [LEFT|RIGHT|INNER] JOIN
[tabel_sumberdata] ON
[tabel_update].[field_kunci]=[tabel_sumberdata].[field_kunci]
SET

[tabel_update].[field_update]=[tabel_sumberdata].[field_data]
```

- Contoh

```
UPDATE simpin_angsur RIGHT JOIN simpin_pinjam ON
simpin_angsur.no_pinjam=simpin_pinjam.no_pinjam SET
simpin_angsur.jasa=simpin_pinjam.jasa/100*simpin_pinjam.jumlah
```

USER

- Pada umumnya digunakan untuk membatasi hak akses
- Manampilkan data user
  - `SELECT * FROM mysql.user;`
- Mebuat USER

```
CREATE USER [nama_user] IDENTIFIED BY "[password]"
```

- Contoh

```
CRETAE USER jono IDENTIFIED BY "jono1234"
```

- Menghapus USER

```
DROP USER [nama_user]
```

HAK AKSES USER (Privileges)

- Digunakan untuk memberikan batasan akses pada pengguna tertentu
- Rumus umum

```
GRANT [hak_akses] ON [nama_database].[nama_tabel] TO
[nama_user]
```

- Contoh
  - GRANT SELECT, UPDTAE ON basis_data_lanjut.simpin_anggota to jono
  - GRANT SELECT (nama, alamat), UPDATE (alamat) ON basis_data_lanjut.perpus_anggota TO jono
  - GRANT ALL ON basis_data_lanjut.* to jono
  - GRANT ALL ON *.* to jono

DAFTAR HAK AKSES

No	Hak Akses	Keterangan
1	ALL [PRIVILEGES]	Memberikan seluruh hak akses, kecuali GRANT OPTION
2	ALTER	Hak akses untuk merubah tabel (ALTER TABLE)
3	ALTER ROUTINE	Hak akses untuk merubah stored routines
4	CREATE	Hak akses untuk membuat tabel dan database
5	CREATE ROUTINE	Hak akses untuk membuat dan menghapus stored routine
6	CREATE TABLESPACE	Hak akses untuk membuat, mengubah dan menghapus tablespaces dan log file
7	CREATE TEMPORARYTABLES	Hak akses untuk membuat tabel sementara CREATE TEMPORARY TABLE
8	CREATE USER	Hak akses untuk membuat, menghapus, dan mengubah user (CREATE USER, DROP USER, RENAME USER, dan REVOKE ALL PRIVILEGES)
9	CREATE VIEW	Hak akses untuk membuat dan mengubah views
10	DELETE	Hak akses untuk menghapus data (DELETE)

DAFTAR HAK AKSES

No	Hak Akses	Keterangan
11	DROP	Hak akses untuk menghapus database, tabel dan view
12	EVENT	Hak akses untuk membuat event
13	EXECUTE	Hak akses untuk menjalankan stored routines
14	FILE	Hak akses untuk membuat server membaca maupun membuat file
15	GRANT OPTION	Hak akses untuk memberikan hak akses kepada user lainnya.
16	INDEX	Hak akses untuk membuat dan menghapus index
17	INSERT	Hak akses untuk menambahkan data (query INSERT)
18	LOCK TABLES	Hak akses untuk mengunci tabel (LOCK TABLES)
19	PROCESS	Hak akses untuk melihat seluruh proses (SHOW PROCESSLIST)
20	PROXY	Hak akses untuk proses proxy

DAFTAR HAK AKSES

No	Hak Akses	Keterangan
21	REFERENCES	Belum diimplementasikan
22	RELOAD	Hak akses untuk operasi FLUSH
23	REPLICATION CLIENT	Hak akses untuk mengubah urutan master dan slave server
24	REPLICATION SLAVE	Hak akses untuk server replikasi untuk membaca log event biner dari server master
25	SELECT	Hak akses untuk melihat data (query SELECT)
26	SHOW DATABASES	Hak akses untuk melihat seluruh database (SHOW DATABASES)
27	SHOW VIEW	Hak akses untuk melihat pembuatan view (SHOW CREATE VIEW)
28	SHUTDOWN	Hak akses untuk mysqladmin shutdown
29	SUPER	Hak akses untuk fungsi administrasi server, seperti CHANGE MASTER TO, KILL, PURGE BINARY LOGS, SET GLOBAL, dan perintah debug mysqladmin
30	TRIGGER	Hak akses untuk operasi trigger
31	UPDATE	Hak akses untuk memperbaharui data (UPDATE)

Variabel

- Variabel di dalam mysql diawali dengan @ untuk variabel user (per sesi) dan @@ untuk variabel global. Urutan eksekusinya juga sesuai dengan letaknya, mulai dari kiri ke kanan.
- program:

```
i := 1

j := i+1
```

- mysql:

```
select @i:=1, @j:=@i+1
```

Storeprocedure

- Salah satu objek *routine* yang tersimpan pada database MySQL dan dapat digunakan untuk menggantikan berbagai kumpulan perintah yang sering kita gunakan
- Sangat berguna ketika kita tidak ingin user mengakses table secara langsung, atau dengan kata lain membatasi hak akses user dan mencatat operasi yang dilakukan. Dengan demikian resiko kebocoran dan kerusakan data dapat lebih diminalisir.
- Variabel parameter dalam stored procedure dibedakan menjadi 3 jenis:
  1. IN : Variabel parameter hanya dapat digunakan untuk menerima input saja. IN juga sebagai nilai default dari variabel parameter.
  2. OUT : Variabel parameter hanya dapat digunakan untuk menyimpan hasil output saja.
  3. INOUT : Variabel parameter digunakan untuk menerima input dan menyimpan hasil output.

Storeprocedure

- Rumus umum

```
CREATE PROCEDURE nama_stored_procedure ([nama_variabel, [...]])
BEGIN
        Isi_stored_procedure
END
```

- Contoh

```
DELIMITER $$

CREATE PROCEDURE jumlah_siswa(vKota VARCHAR(50), OUT iJumlah INT)
    BEGIN
        SELECT COUNT(*) INTO iJumlah FROM siswa WHERE kota = vKota;
    END$$

DELIMITER ;
```

- Menggunakan

```
CALL jumlah_siswa('Depok', @iJumlah);

SELECT @iJumlah;
```

TRIGGER

- Trigger merupakan store procedure yang dijalankan secara automatis saat user melakukan modifikasi data pada tabel Rumus umum
- Event yang bisa digunakan untuk mengekseskusi triger
  - BEFORE INSERT – dijalankan ketika data di masukan ke dalam table.
  - AFTER INSERT – dijalankan setelah data masuk ke dalam table.
  - BEFORE UPDATE – dijalankan sebelum proses update data.
  - AFTER UPDATE – dijalankan setelah proses proses update data.
  - BEFORE DELETE – dijalankan sebelum proses delete data.
  - AFTER DELETE – dijalankan setelah proses delete data.
- Referensi "OLD" dan "NEW“
  - Karena trigger digunakan pada saat terjadi perubahan row data, maka kita perlu referensi ke row sebelum dan sesudah perubahan. Untuk ini ada dua alias yang berfungsi untuk hal tersebut yaitu **OLD** dan **NEW**.
  - Sesuai namanya, OLD digunakan untuk referensi sebelum perubahan dan NEW untuk referensi sesudah perubahan.

TRIGGER

- Menampilkan Trigger yang sudah ada

```
SHOW TRIGGERS
```

- Membuat Trigger

```
CREATE TRIGGER [nama_trigger] [event] ON [nama_table]
FOR EACH ROW
BEGIN
      //isi trigger berupa query
END;
```

- Menghapus Trigger

```
DROP TRIGGER [nama_trigger]
```

TRIGGER

- Contoh

Command Prompt - mysql -u root -p

```
MariaDB [basis_data_lanjut]> DELIMITER |
MariaDB [basis_data_lanjut]> CREATE TRIGGER del_perpus_pinjam AFTER DELETE ON perpus_pinjam
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO log set tgl=now(), pengguna=user(),
    -> ket=CONCAT('Menghapus pinjam :', OLD.id_pinjam,':',OLD.id_anggota,':',OLD.id_item);
    -> END;
    -> |
Query OK, 0 rows affected (0.09 sec)

MariaDB [basis_data_lanjut]> DELIMITER ;
MariaDB [basis_data_lanjut]>
```

Command Prompt - mysql -h PSI-UNSIQ -u jono -p

```
Database changed
MariaDB [basis_data_lanjut]> DELETE FROM perpus_pinjam WHERE id_pinjam=6;
Query OK, 1 row affected (0.54 sec)

MariaDB [basis_data_lanjut]>
```

TRIGGER

- Hasil