JOIN, VIEW, DAN STORED PROCEDURE

OBJEKTIF :

1. Mahasiswa mampu menggabungkan beberapa tabel menggunakan klausa join.
2. Mahasiswa mampu membuat *virtual table* dan menampilkan *virtual table*.
3. Mahasiswa mampu menyimpan prosedur dan memanggil prosedur yang tersimpan.

5.1 JOIN

JOIN digunakan untuk menggabungkan data dari dua atau lebih tabel, berdasarkan kolom terkait di antara tabel yang akan dilakukan JOIN. Jenis-Jenis JOIN :

- INNER JOIN
- OUTER JOIN
- CROSS JOIN

Sebagai contoh dalam menerapkan klausa JOIN.

Pertama, tampilkan dulu tabel Pengguna dengan kolom-kolomnya yaitu IDPengguna, NamaPengguna, NoHP, Email, Alamat, dan Kota.

```
SELECT*FROM Pengguna;
```

IDPengguna	NamaPengguna	NoHP	Email	Alamat	Kota
1	Sarah	081289987117	sarahayuh@gmail.com	Jl.Pondok 38	Jakarta
2	Dyan	081267761355	dyantriandini@gmail.com	Jl.Sari D35	Bandung
3	Riza	081245677654	riza123@gmail.com	Jl.Angkasa	Bogor
4	Maudy	081278905432	maudy12@gmail.com	Jl.Kemang	Depok
NULL	NULL	NULL	NULL	NULL	NULL

Selanjutnya, tampilkan lagi tabel Pembelian dengan kolom-kolomnya yaitu IDPemesanan, KodePemesanan, IDCustomer, IDBarang, JumlahPemesanan, TanggalPembelian, dan TotalHarga.

```
SELECT*FROM Pembelian;
```

IDPembelian	IDPengguna	IDBarang	TglPembelian	TglPembayaran	TglPengembalian
PMB1	1	BRG1	2020-08-13	2020-08-13	2020-08-20
PMB2	3	BRG3	2020-08-19	2020-08-19	2020-08-25
PMB3	2	BRG2	2020-08-22	2020-08-22	2020-08-28
NULL	NULL	NULL	NULL	NULL	NULL

5.1.1 INNER JOIN

INNER JOIN berfungsi untuk menggabungkan dua tabel atau lebih yang memiliki hubungan sehingga kolom yang dipilih akan ditampilkan. Syarat INNER JOIN adalah kedua tabel harus ada kolom yang sama, yaitu IDPengguna.

Bentuk umum :

```
SELECT select_list FROM t1 INNER JOIN t2 ON join_condition;
```

Penjelasan :

- `select_list` : Merupakan nama kolom yang akan ditampilkan.
- `t1` : Merupakan tabel 1 yang digunakan.
- `t2` : Merupakan tabel 2 yang digunakan.
- `join_condition` : Merupakan sebuah kondisi yang menghubungkan 2 tabel tersebut.

Sebagai contoh, gunakan INNER JOIN untuk menampilkan NamaPengguna, IDBarang, dan TglPembayaran.

```
SELECT NamaPengguna, IDBarang, TglPembayaran FROM Pengguna INNER JOIN Pembelian
ON Pengguna.IDPengguna = Pembelian.IDPengguna;
```

Menghasilkan *output* :

	NamaPengguna	IDBarang	TglPembayaran
▶	Sarah	BRG1	2020-08-13
	Dyan	BRG2	2020-08-22
	Riza	BRG3	2020-08-19

5.1.2 OUTER JOIN

OUTER JOIN terbagi menjadi 2, yaitu LEFT JOIN dan RIGHT JOIN.

LEFT JOIN menampilkan semua data yang berada pada tabel sisi kiri, sedangkan data pada tabel sisi kanan akan bernilai *null* jika datanya tidak ada pada tabel kiri.

Bentuk umum :

```
SELECT select_list FROM t1 LEFT JOIN t2 ON join_condition;
```

Penjelasan :

- `select_list` : Merupakan nama kolom yang akan ditampilkan.
- `t1` : Merupakan tabel 1 yang digunakan.
- `t2` : Merupakan tabel 2 yang digunakan.
- `join_condition` : Merupakan sebuah kondisi yang menghubungkan 2 tabel tersebut.

Sebagai contoh, gunakan LEFT JOIN untuk menampilkan NamaPengguna, IDBarang, dan TglPembayaran.

```
SELECT NamaPengguna, IDBarang, TglPembayaran FROM Pengguna LEFT JOIN Pembelian
ON Pengguna.IDPengguna = Pembelian.IDPengguna;
```

Menghasilkan *output* :

	NamaPengguna	IDBarang	TglPembayaran
▶	Sarah	BRG1	2020-08-13
	Dyan	BRG2	2020-08-22
	Riza	BRG3	2020-08-19
	Maudy	NULL	NULL

Pada *output* menghasilkan nilai *null* pada kolom IDBarang dan TglPembayaran, karena tabel sisi kanan datanya tidak ada pada tabel kiri.

RIGHT JOIN menampilkan semua data yang berada pada tabel sisi kanan, sedangkan data pada tabel sisi kiri akan bernilai *null* jika datanya tidak ada pada tabel kanan.

Bentuk umum :

```
SELECT select_list FROM t1 RIGHT JOIN t2 ON join_condition;
```

Penjelasan :

- `select_list` : Merupakan nama kolom yang akan ditampilkan.
- `t1` : Merupakan tabel 1 yang digunakan.
- `t2` : Merupakan tabel 2 yang digunakan.
- `join_condition` : Merupakan sebuah kondisi yang menghubungkan 2 tabel tersebut.

Sebagai contoh, gunakan RIGHT JOIN untuk menampilkan NamaPengguna, IDBarang, dan TglPembayaran.

```
SELECT NamaPengguna, IDBarang, TglPembayaran FROM Pengguna RIGHT JOIN Pembelian
ON Pengguna.IDPengguna = Pembelian.IDPengguna;
```

Menghasilkan *output* :

	NamaPengguna	IDBarang	TglPembayaran
▶	Sarah	BRG1	2020-08-13
	Riza	BRG3	2020-08-19
	Dyan	BRG2	2020-08-22

5.1.3 CROSS JOIN

CROSS JOIN akan menghasilkan kombinasi dari baris pada tabel pertama dengan baris di tabel kedua. Secara umum jika masing-masing tabel memiliki n dan m baris, hasil CROSS JOIN akan memiliki baris n x m.

Bentuk umum :

```
SELECT  select_list FROM t1 CROSS JOIN t2;
```

Penjelasan :

- `select_list` : Merupakan nama kolom yang akan ditampilkan.
- `t1` : Merupakan tabel 1 yang digunakan.
- `t2` : Merupakan tabel 2 yang digunakan.

Sebagai contoh, gunakan CROSS JOIN untuk mengalikan baris antara kolom NamaPengguna, IDBarang, dan TglPembayaran.

```
SELECT NamaPengguna, IDBarang, TglPembayaran FROM Pengguna CROSS JOIN Pembelian;
```

Menghasilkan *output* :

	NamaPengguna	IDBarang	TglPembayaran
▶	Sarah	BRG1	2020-08-13
	Dyan	BRG1	2020-08-13
	Riza	BRG1	2020-08-13
	Maudy	BRG1	2020-08-13
	Sarah	BRG3	2020-08-19
	Dyan	BRG3	2020-08-19

	NamaPengguna	IDBarang	TglPembayaran
	Riza	BRG3	2020-08-19
	Maudy	BRG3	2020-08-19
	Sarah	BRG2	2020-08-22
	Dyan	BRG2	2020-08-22
	Riza	BRG2	2020-08-22
	Maudy	BRG2	2020-08-22

5.2 VIEW

VIEW atau *virtual table* adalah salah satu objek *database* yang fungsinya menampilkan data dari tabel yang sudah ada.

Sebagai contoh dalam menerapkan VIEW.

Pertama, tampilkan dulu tabel Pengguna dengan kolom-kolomnya yaitu IDPengguna, NamaPengguna, NoHP, Email, Alamat, dan Kota.

```
SELECT*FROM Pengguna;
```

	IDPengguna	NamaPengguna	NoHP	Email	Alamat	Kota
	1	Sarah	081289987117	sarahayuh@gmail.com	Jl.Pondok 38	Jakarta
	2	Dyan	081267761355	dyantriandini@gmail.com	Jl.Sari D35	Bandung
	3	Riza	081245677654	riza123@gmail.com	Jl.Angkasa	Bogor
*	NULL	NULL	NULL	NULL	NULL	NULL

Selanjutnya, tampilkan lagi tabel Pembelian dengan kolom-kolomnya yaitu IDPemesanan, KodePemesanan, IDCustomer, IDBarang, JumlahPemesanan, TanggalPembelian, dan TotalHarga.

```
SELECT*FROM Pembelian;
```

	IDPembelian	IDPengguna	IDBarang	TglPembelian	TglPembayaran	TglPengembalian
▶	PMB1	1	BRG3	2020-08-13	2020-08-13	2020-08-20
	PMB2	3	BRG3	2020-08-19	2020-08-19	2020-08-25
	PMB3	2	BRG2	2020-08-22	2020-08-22	2020-08-28
*	NULL	NULL	NULL	NULL	NULL	NULL

5.2.1 CREATE VIEW

CREATE VIEW atau membuat VIEW atau digunakan untuk membuat VIEW.

Bentuk umum :

```
CREATE VIEW view_name AS SELECT select_list FROM table_name;
```

Penjelasan :

- `view_name` : Merupakan nama view yang akan dibuat.
- `select_list` : Merupakan nama kolom-kolom yang akan di ambil.
- `table_name` : Merupakan nama tabel yang akan digunakan.

Sebagai contoh, gunakan VIEW untuk membuat *virtual table* yang bernama PenjualanAgustus dan menggunakan INNER JOIN.

```
CREATE VIEW PenjualanAgustus AS SELECT NamaPengguna, IDBarang, TglPembayaran FROM Pengguna INNER JOIN Pembelian ON Pengguna.IDPengguna = Pembelian.IDPengguna;
```

Menghasilkan *output* :

	NamaPengguna	IDBarang	TglPembayaran
▶	Sarah	BRG1	2020-08-13
	Dyan	BRG2	2020-08-22
	Riza	BRG3	2020-08-19

5.2.2 UPDATABLE VIEW

UPDATABLE VIEW merupakan perintah untuk mengubah data yang berada pada tabel VIEW.

Bentuk umum :

```
UPDATE VIEW view_name SET column_1 = value_1 WHERE condition;
```

Penjelasan :

- `view_name` : Merupakan nama view yang akan digunakan.
- `column_1` : Merupakan nama kolom yang akan diubah.
- `value_1` : Merupakan data yang baru.
- `condition` : Merupakan kondisi dari data yang akan diubah.

Sebagai contoh, gunakan UPDATE untuk mengubah IDBarang pada NamaPengguna Sarah menjadi BRG3.

```
UPDATE PenjualanAgustus SET IDBarang = 'BRG3' WHERE NamaPengguna = 'Sarah';
```

Menghasilkan *output* :

	NamaPengguna	IDBarang	TglPembayaran
▶	Sarah	BRG3	2020-08-13
	Dyan	BRG2	2020-08-22
	Riza	BRG3	2020-08-19

Pada *output* IDBarang pada NamaPengguna Sarah sudah berubah menjadi BRG3.

5.3 STORED PROCEDURE

Di dalam *database*, terdapat beberapa objek *database*. Salah satunya yaitu Stored procedure. Stored procedure merupakan kumpulan perintah/prosedur tersimpan yang digunakan untuk mengakses tabel secara tidak langsung, yang berfungsi untuk membatasi hak akses.

Parameter membuat stored procedure lebih fleksibel. Stored procedure mempunyai 3 parameter, yaitu :

- Parameter IN
- Parameter OUT
- Parameter INOUT

Bentuk umum :

```
[IN | OUT | INOUT] parameter_name data_type[(length)]
```

Penjelasan :

- `parameter_name` : Merupakan nama parameter yang dibuat.
- `data_type` : Merupakan tipe data yang digunakan.
- `length` : Merupakan panjang karakter dari tipe data yang digunakan.

Sebagai contoh, gunakan parameter IN untuk membuat prosedur tersimpan menggunakan parameter IN :

```
DELIMITER //
CREATE PROCEDURE SPPengguna(IN KotaPengguna VARCHAR(20))
BEGIN
SELECT * FROM Pengguna WHERE Kota = KotaPengguna;
END //
DELIMITER ;
```

Perintah delimiter ini bukan bentuk umum dari stored procedure, tetapi jika ingin menggunakan stored procedure harus menggunakan delimiter di awal dan di akhir query.

Gunakan perintah CALL untuk memanggil stored procedure yang sudah dibuat. Disini terdapat ketentuan bahwa hanya akan menampilkan pengguna dari kota Bogor saja :

```
CALL SPPengguna('Bogor');
```

Menghasilkan *output* :

	IDPengguna	NamaPengguna	NoHP	Email	Alamat	Kota
▶	3	Riza	081245677654	riza123@gmail.com	Jl.Angkasa	Bogor

5.2.1 CREATE STORED PROCEDURE

CREATE STORED PROCEDURE digunakan untuk membuat Stored Procedure.

Bentuk umum :

```
CREATE PROCEDURE procedure_name(parameter_list)
BEGIN statements;
END //
```

Penjelasan :

- `procedure_name` : Merupakan nama stored procedure yang dibuat.
- `parameter_list` : Merupakan daftar parameter yang ingin dibuat.
- `statements` : Perintah yang digunakan, contohnya select*from atau insert.

Sebagai contoh, gunakan CREATE STORED PROCEDURE untuk membuat prosedur tersimpan yang bernama SPPenjualanAgustus :

```
CREATE PROCEDURE SPPenjualanAgustus()
BEGIN
SELECT NamaPengguna, IDBarang, TglPembayaran FROM Pengguna
INNER JOIN Pembelian ON Pengguna.IDPengguna = Pembelian.IDPengguna;
END //
DELIMITER ;
```

Perintah delimiter ini bukan bentuk umum dari stored procedure, tetapi jika ingin menggunakan stored procedure harus menggunakan delimiter di awal dan di akhir query.

5.3.2 CALL STORED PROCEDURE

CALL STORED PROCEDURE digunakan untuk memanggil dan menampilkan hasil dari Stored Procedure yang telah dibuat.

Bentuk umum :

```
CALL stored_procedure_name;
```

Penjelasan :

- `stored_procedure_name` : Merupakan nama stored procedure yang akan dipanggil.

Gunakan perintah CALL untuk memanggil stored procedure yang sudah dibuat.

```
CALL SPPenjualanAgustus;
```

Menghasilkan *output* :

	NamaPengguna	IDBarang	TglPembayaran
▶	Sarah	BRG3	2020-08-13
	Dyan	BRG2	2020-08-22
	Riza	BRG3	2020-08-19

5.3.3 Keuntungan dan Kerugian Stored Procedure

Berikut adalah beberapa keuntungan menggunakan *Stored Procedure* yaitu :

- Stored Procedure mempermudah pengetikan *query*, karena *query* yang sama dan berulang maka cukup panggil store procedurenya saja tanpa menggetik *query* dari awal lagi.
- Stored Procedure mengurangi lalu lintas antara aplikasi dan server *database*.
- Stored Procedure membuat *database* lebih aman.

Berikut adalah beberapa kerugian menggunakan Stored Procedure yaitu :

- Penggunaan memori meningkat jika menggunakan banyak Stored Procedure.

- Penyelesaian masalah yang sulit.
- Tidak mudah mengembangkan dan memelihara Stored Procedure.

Referensi :

[1] Fathansyah. 2018. *Basis Data Revisi Ketiga*. Bandung: Informatika.

[2] www.mysqltutorial.org. 2020. "MySQL CREATE VIEW", https://www.mysqltutorial.org/create-sql-views-mysql.aspx, diakses pada 20 Juli 2020.

[3] www.mysqltutorial.org. 2020. "MySQL UPDATABLE VIEW", https://www.mysqltutorial.org/create-sql-updatable-views.aspx, diakses pada 22 Juli 2020.

[4] www.mysqltutorial.org. 2020. "MySQL CREATE and CALL STORED PROCEDURE", https://www.mysqltutorial.org/getting-started-with-mysql-stored-procedures.aspx, diakses pada 22 Juli 2020.

[5] www.mysqltutorial.org. 2020. "MySQL JOIN", https://www.mysqltutorial.org/mysql-join/, diakses pada 19 Agustus 2020.