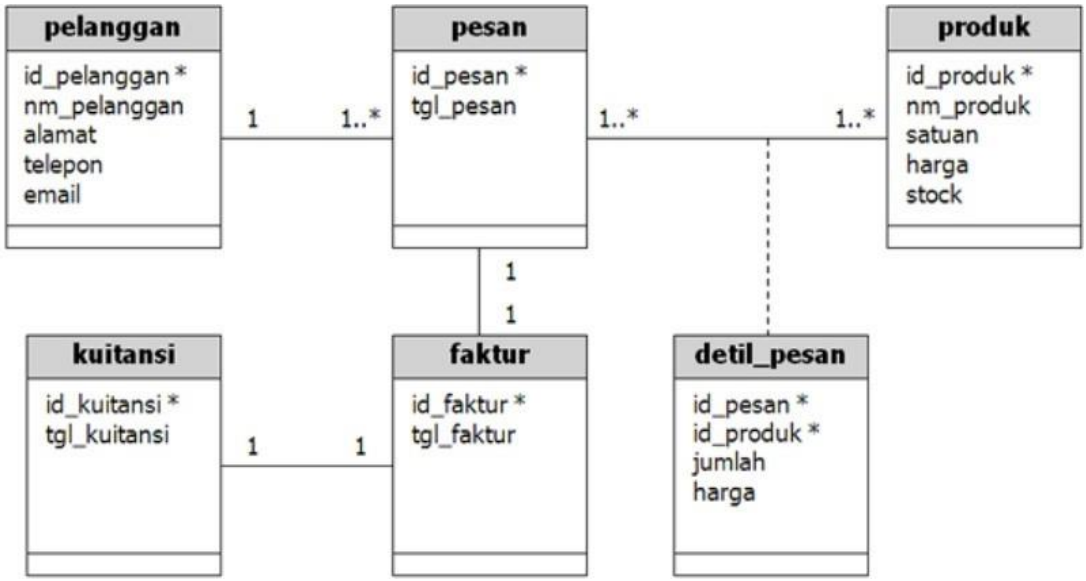
PRAKTIKUM BASIS DATA 2

PERTEMUAN III

TRI WAHYU QUR'ANA, S.Kom, M.Kom

FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS ISLAM KALIMANTAN
MUHAMMAD ARSYAD AL BANJARI
BANJARMASIN
2021

Studi Kasus **pemodelan data konseptual** Sistem Pemesanan (Penjualan) Barang



berikut ini **spesifikasi basis data** dari pemodelan data konseptual di atas:

1. Membuat tabel Pelanggan

```
CREATE TABLE pelanggan (
id_pelanggan varchar(5) NOT NULL,
nm_pelanggan varchar(40) NOT NULL,
alamat text NOT NULL,
telepon varchar(20) NOT NULL,
email varchar(50) NOT NULL,
PRIMARY KEY(id_pelanggan)
);
```

2. Membuat tabel Produk

```
CREATE TABLE produk (
id_produk varchar(5) NOT NULL,
nm_produk varchar(30) NOT NULL,
satuan varchar(10) NOT NULL,
harga decimal(10,0) NOT NULL,
stock int(3) NOT NULL,
PRIMARY KEY (id_produk)
);
```

3. Membuat tabel Pesan

```
CREATE TABLE pesan (
id_pesan int(5) NOT NULLauto_increment,
id_pelanggan varchar(5) NOT NULL,
tgl_pesan date NOT NULL,
PRIMARY KEY(id_pesan),
FOREIGN KEY (id_pelanggan)
REFERENCES pelanggan (id_pelanggan)
);
```

4. Membuat tabel Detail Pesan

```
CREATE TABLE detil_pesan (
id_pesan int (5) NOT NULL,
id_produk varchar (5) NOT NULL,
jumlah int (5) NOT NULL default '0',
harga decimal (10,0) NOT NULL default '0',
PRIMARY KEY (id_pesan,id_produk),
FOREIGN KEY(id_produk)  REFERENCES produk (id_produk),
FOREIGN KEY(id_pesan) REFERENCES pesan (id_pesan)
);
```

5. Membuat tabel Faktur

```
CREATE TABLE faktur (
id_faktur int (5) NOT NULL auto_increment,
id_pesan int (5) NOT NULL,
tgl_faktur date NOT NULL,
PRIMARY KEY (id_faktur),
FOREIGN KEY (id_pesan) REFERENCES pesan (id_pesan)
);
```

6. Membuat tabel Kuitansi

```
CREATE TABLE kuitansi (
id_kuitansi int (5) NOT NULL auto_increment,
id_faktur int (5) NOT NULL,
tgl_kuitansi date NOT NULL,
PRIMARY KEY (id_kuitansi),
FOREIGN KEY (id_faktur) REFERENCES faktur (id_faktur)
);
```

Selanjutnya, untuk memudahkan dalam pemberian contoh, isilah tabel-tabel diatas dengan record (data) secukupnya.

Perintah SELECT dari Banyak Tabel dengan JOIN

Pada praktisnya, terkadang kita juga memerlukan tampilan data yang tidak hanya berasal dari 1 (satu) tabel, namun bisa dari beberapa tabel sekaligus. Contohnya, dari class diagram diatas, kita ingin menampilkan nama pelanggan berikut transaksi yang pernah dilakukannya. Dari contoh tersebut, kita harus bisa menggabungkan minimal dua tabel, yaitu **pelanggan** dan **pesan**. Untuk menggabungkan 2 (dua) atau lebih tabel, kita dapat menggunakan bentuk perintah JOIN.

Inner Join

Dengan inner join, tabel akan digabungkan dua arah, sehingga tidak ada data yang NULL di satu sisi. Sebagai contoh, kita akan menggabungkan tabel **pelanggan** dan **pesan** dimana kita akan menampilkan daftar pelanggan yang pernah melakukan pemesanan (transaksi). Isi tabel pelanggan dan pesan adalah sebagai berikut :

Tabel **pelanggan** (hanya ditampilkan id, nama dan email).

```
+-------------+-----------------+--------------+-------------+--------------------------+
| id_pelanggan | nm_pelanggan    | alamat       | telepon     | email                    |
+-------------+-----------------+--------------+-------------+--------------------------+
| P0001       | Yusuf Hadiwinata | Bekasi       | 085692019009 | yusuf.hadiwinata@gmail.com |
| P0002       | Yoga Rimaldo    | Bandung      | 08571239501 | yoga@computradetech.com    |
| P0003       | Winny           | Jakarta Barat | 0815829221  | winny@computradetech.com   |
| P0004       | Ninda Budianto  | Jakarta Timur | 08171231234 | ninda@computradetech.com   |
+-------------+-----------------+--------------+-------------+--------------------------+
4 rows in set (0.00 sec)
```

Tabel pesan:

```
+----------+-------------+------------+
| id_pesan | id_pelanggan | tgl_pesan  |
+----------+-------------+------------+
|        1 | P0001        | 2008-02-02 |
|        2 | P0002        | 2008-02-05 |
|        3 | P0002        | 2008-02-10 |
|        4 | P0004        | 2008-01-20 |
|        5 | P0004        | 2007-12-14 |
+----------+-------------+------------+
5 rows in set (0.00 sec)
```

Cara 1 : Penggabungan dengan WHERE

Bentuk umum

```
SELECT tabel1.*, tabel2.* FROM tabel1, tabel2
WHERE tabel1.PK=tabel2.FK;
```

Berikut ini perintah SQL untuk menggabungkan tabel pelanggan dan pesan:

```
SELECT pelanggan.id_pelanggan, pelanggan.nm_pelanggan,
pesan.id_pesan, pesan.tgl_pesan
FROM pelanggan, pesan
WHERE pelanggan.id_pelanggan=pesan.id_pelanggan;
```

Hasilnya :

```
+-------------+-----------------+----------+------------+
| id_pelanggan | nm_pelanggan    | id_pesan | tgl_pesan  |
+-------------+-----------------+----------+------------+
| P0001        | Yusuf Hadiwinata |        1 | 2008-02-02 |
| P0002        | Yoga Rimaldo     |        2 | 2008-02-05 |
| P0002        | Yoga Rimaldo     |        3 | 2008-02-10 |
| P0004        | Ninda Budianto   |        4 | 2008-01-20 |
| P0004        | Ninda Budianto   |        5 | 2007-12-14 |
+-------------+-----------------+----------+------------+
5 rows in set (0.01 sec)
```

Pada hasil perintah query di atas terlihat bahwa terdapat 5 (lima) transaksi yang dilakukan oleh 3 (tiga) orang pelanggan. Jika kita lihat kembali isi tabel pelanggan di atas, maka terdapat satu pelanggan yang tidak ditampilkan yaitu yang memiliki id pelanggan P0003, karena belum pernah melakukan transaksi.

Cara 2 : Penggabungan dengan INNER JOIN

Bentuk umum

```
SELECT tabel1.*, tabel2.*
FROM tabel1 INNER JOIN tabel2
ON tabel1.PK=tabel2.FK;
```

Berikut ini perintah SQL untuk menggabungkan tabel pelanggan dan pesan:

```
SELECT pelanggan.id_pelanggan, pelanggan.nm_pelanggan,
pesan.id_pesan, pesan.tgl_pesan
FROM pelanggan INNER JOIN pesan
ON pelanggan.id_pelanggan=pesan.id_pelanggan;
```

Hasilnya:

```
+--------------+------------------+----------+------------+
| id_pelanggan | nm_pelanggan     | id_pesan | tgl_pesan  |
+--------------+------------------+----------+------------+
| P0001        | Yusuf Hadiwinata |        1 | 2008-02-02 |
| P0002        | Yoga Rimaldo     |        2 | 2008-02-05 |
| P0002        | Yoga Rimaldo     |        3 | 2008-02-10 |
| P0004        | Ninda Budianto   |        4 | 2008-01-20 |
| P0004        | Ninda Budianto   |        5 | 2007-12-14 |
+--------------+------------------+----------+------------+
5 rows in set (0.00 sec)
```

OUTER JOIN

Dengan outer join, tabel akan digabungkan satu arah, sehingga memungkinkan ada data yang NULL (kosong) di satu sisi. Outer Join terbagi menjadi 2 (dua) yaitu LEFT JOIN dan RIGHT. Berikut ini bentuk umum dan contohnya:

LEFT JOIN

Bentuk umum

```
SELECT tabel1.*, tabel2.*
FROM tabel1 LEFT JOIN tabel2
ON tabel1.PK=tabel2.FK;
```

Berikut ini perintah SQL untuk menggabungkan tabel pelanggan dan pesan:

```
SELECT pelanggan.id_pelanggan, pelanggan.nm_pelanggan,
pesan.id_pesan, pesan.tgl_pesan
FROM pelanggan LEFT JOIN pesan
ON pelanggan.id_pelanggan=pesan.id_pelanggan;
```

Hasilnya :

```
+--------------+------------------+----------+------------+
| id_pelanggan | nm_pelanggan     | id_pesan | tgl_pesan  |
+--------------+------------------+----------+------------+
| P0001        | Yusuf Hadiwinata |        1 | 2008-02-02 |
| P0002        | Yoga Rimaldo     |        2 | 2008-02-05 |
| P0002        | Yoga Rimaldo     |        3 | 2008-02-10 |
| P0003        | Winny            |     NULL | NULL       |
| P0004        | Ninda Budianto   |        4 | 2008-01-20 |
| P0004        | Ninda Budianto   |        5 | 2007-12-14 |
+--------------+------------------+----------+------------+
6 rows in set (0.00 sec)
```

Berbeda dengan hasil sebelumnya (inner join), penggunaan left join akan menampilkan juga data pelanggan dengan id P0003, walaupun pelanggan tersebut belum pernah bertransaksi. Dan pada kolom id_pesan dan tgl_pesan untuk pelanggan P0003 isinya NULL, artinya di tabel kanan (pesan) pelanggan tersebut tidak ada.

RIGHT JOIN

Bentuk umum

```
SELECT tabel1.*, tabel2.*
FROM tabel1 RIGHT JOIN tabel2
ON tabel1.PK=tabel2.FK;
```

Berikut ini perintah SQL untuk menggabungkan tabel pelanggan dan pesan:

```
SELECT pelanggan.id_pelanggan, pelanggan.nm_pelanggan,
pesan.id_pesan, pesan.tgl_pesan
FROM pelanggan RIGHT JOIN pesan
ON pelanggan.id_pelanggan=pesan.id_pelanggan;
```

Hasilnya :

```
+--------------+------------------+----------+------------+
| id_pelanggan | nm_pelanggan     | id_pesan | tgl_pesan  |
+--------------+------------------+----------+------------+
| P0001        | Yusuf Hadiwinata |        1 | 2008-02-02 |
| P0002        | Yoga Rimaldo     |        2 | 2008-02-05 |
| P0002        | Yoga Rimaldo     |        3 | 2008-02-10 |
| P0004        | Ninda Budianto   |        4 | 2008-01-20 |
| P0004        | Ninda Budianto   |        5 | 2007-12-14 |
+--------------+------------------+----------+------------+
5 rows in set (0.00 sec)
```

Dengan right join, tabel yang menjadi acuan adalah tabel sebelah kanan (tabel pesan), jadi semua isi tabel pesan akan ditampilkan. Jika data pelanggan tidak ada di tabel pelanggan, maka isi tabel pesan tetap ditampilkan.

Menggabungkan Tiga Tabel

Untuk menggabungkan tiga tabel atau lebih, pada dasarnya sama dengan penggabungan 2 (dua) tabel. Sebagai contoh misalnya kita akan menampilkan barangbarang yang dipesan beserta nama barang dan harganya untuk pemesanan dengan nomor 1. Berikut ini perintah SQL-nya:

```
SELECT pesan.id_pesan, produk.id_produk, produk.nm_produk,
detil_pesan.harga, detil_pesan.jumlah
FROM pesan, detil_pesan, produk
WHERE pesan.id_pesan=detil_pesan.id_pesan AND
detil_pesan.id_produk=produk.id_produk
AND pesan.id_pesan='1' ;
```

Hasilnya:

```
+----------+-----------+------------+-------+--------+
| id_pesan | id_produk | nm_produk  | harga | jumlah |
+----------+-----------+------------+-------+--------+
|        1 | B0001     | Buku Tulis |  2700 |      2 |
|        1 | B0003     | Penggaris  |  3000 |      3 |
|        1 | B0004     | Pensil     |  2000 |      1 |
+----------+-----------+------------+-------+--------+
3 rows in set (0.01 sec)
```

Pengelompokkan Hasil Query dengan GROUP BY

Hasil query terkadang perlu dikelompokkan berdasarkan kriteria atau kondisi tertentu. Misalnya kita akan menampilkan jumlah barang yang dibeli untuk masing masing transaksi (pemesanan). Perhatikan perintah query berikut ini dan lihat hasilnya:

```
SELECT pesan.id_pesan, pesan.tgl_pesan,
detil_pesan.jumlah
FROM pesan, detil_pesan
WHERE pesan.id_pesan=detil_pesan.id_pesan;
```

Hasilnya:

```
+----------+------------+--------+
| id_pesan | tgl_pesan  | jumlah |
+----------+------------+--------+
|        1 | 2008-02-02 |      2 |
|        1 | 2008-02-02 |      3 |
|        1 | 2008-02-02 |      1 |
+----------+------------+--------+
3 rows in set (0.00 sec)
```

Agar jumlah barang ditampilkan per-transaksi (pemesanan), maka kita dapat menggunakan fungsi GROUP BY dan juga SUM untuk menjumlahkan jumlah barang.

Berikut ini perintah query dengan group by dan count.

```
SELECT pesan.id_pesan, pesan.tgl_pesan,
SUM (detil_pesan.jumlah) as jumlah
FROM pesan, detil_pesan
WHERE pesan.id_pesan=detil_pesan.id_pesan
GROUP BY id_pesan;
```

Hasilnya:

```
+----------+------------+--------+
| id_pesan | tgl_pesan  | jumlah |
+----------+------------+--------+
|        1 | 2008-02-02 |      6 |
+----------+------------+--------+
1 row in set (0.05 sec)
```

Selain hasil di atas, kita juga dapat menggunakan tambahan WITH ROLLUP di belakang group by untuk menampilkan jumlah total seluruh barang.

Berikut ini perintah query dan hasilnya:

```
SELECT pesan.id_pesan, pesan.tgl_pesan,
SUM (detil_pesan.jumlah) as jumlah
FROM pesan, detil_pesan
WHERE pesan.id_pesan=detil_pesan.id_pesan
GROUP BY id_pesan WITH ROLLUP;
```

Hasilnya:

```
+----------+------------+--------+
| id_pesan | tgl_pesan  | jumlah |
+----------+------------+--------+
|        1 | 2008-02-02 |      6 |
|     NULL | 2008-02-02 |      6 |
+----------+------------+--------+
2 rows in set (0.03 sec)
```

HAVING

Perintah query berikut ini akan menampilkan jumlah item (jenis) barang untuk tiap transaksi.

```
SELECT pesan.id_pesan, COUNT (detil_pesan.id_produk) as jumlah
FROM pesan, detil_pesan
WHERE pesan.id_pesan=detil_pesan.id_pesan
GROUP BY pesan.id_pesan
```

Hasilnya:

```
+----------+--------+
| id_pesan | jumlah |
+----------+--------+
|        1 |      3 |
+----------+--------+
1 row in set (0.00 sec)
```

WHERE tidak dapat diterapkan pada fungsi agregrasi seperti COUNT, SUM, AVG dll. Untuk menyeleksi suatu fungsi agregasi, kita tidak dapat menggunakan WHERE, namun kita dapat menggunakan HAVING.

Berikut ini perintah query yang menggunakan HAVING:

```
SELECT pesan.id_pesan, COUNT (detil_pesan.id_produk) as jumlah
FROM pesan, detil_pesan
WHERE pesan.id_pesan=detil_pesan.id_pesan
GROUPBY pesan.id_pesan
HAVING jumlah > 2 ;
```

Hasilnya:

```
+----------+--------+
| id_pesan | jumlah |
+----------+--------+
|        1 |      3 |
+----------+--------+
1 row in set (0.00 sec)
```

SubSELECT

Perintah query SubSELECT memungkinkan untuk melakukan query di dalam query. Misalnya kita akan menampilkan data yang kondisinya merupakan hasil dari query lain. Perintah SubSELECT memiliki banyak variasi.

Berikut ini beberapa variasi bentuk perintah SubSELECT.

- Menampilkan daftar pelanggan yang pernah melakukan transaksi (pemesanan).

```
SELECT id_pelanggan, nm_pelanggan FROM pelanggan
WHERE id_pelanggan IN (SELECT id_pelanggan FROM pesan);
```

Hasilnya sebagai berikut:

```
+--------------+------------------+
| id_pelanggan | nm_pelanggan     |
+--------------+------------------+
| P0001        | Yusuf Hadiwinata |
| P0002        | Yoga Rimaldo     |
| P0004        | Ninda Budianto   |
+--------------+------------------+
3 rows in set (0.01 sec)
```

- Menampilkan data pemesanan dengan jumlah barang terbanyak.

```
SELECT id_pesan, jumlah FROM detil_pesan
WHERE jumlah = ( SELECT MAX (jumlah) FROM detil_pesan);
```

Hasilnya sebagai berikut:

```
+----------+--------+
| id_pesan | jumlah |
+----------+--------+
|        1 |      3 |
+----------+--------+
1 row in set (0.00 sec)
```

Menampilkan Record secara Random

MySQL memiliki fungsi khusus yang dapat digunakan untuk menampilkan record secara acak (random).

Berikut ini contoh perintah query untuk menampilkan data pelanggan secara acak (random):

```
SELECT id_pelanggan, nm_pelanggan, email
FROM pelanggan ORDER BY RAND() ;
```

Salah satu hasilnya sebagai berikut:

```
+--------------+------------------+----------------------------+
| id_pelanggan | nm_pelanggan     | email                      |
+--------------+------------------+----------------------------+
| P0004        | Ninda Budianto   | ninda@computradetech.com   |
| P0001        | Yusuf Hadiwinata | yusuf.hadiwinata@gmail.com |
| P0002        | Yoga Rimaldo     | yoga@computradetech.com    |
| P0003        | Winny            | winny@computradetech.com   |
+--------------+------------------+----------------------------+
4 rows in set (0.02 sec)
```