SQL : DML (Continue)

Query Multi-Table

- Dapat menggunakan subqueri asalkan kolom yang dihasilkan berasal dari tabel yang sama.
- Jika kolom yang dihasilkan berasal dari lebih dari satu tabel, maka harus menggunakan join.
- Untuk menampilkan join, harus menyertakan lebih dari satu tabel dalam clause FROM.
- Menggunakan koma sebagai pemisah dan biasanya menyertakan clause WHERE untuk menspesifikasikan kolom-kolom yang di-join.
- Memungkinkan penggunaan alias untuk nama tabel dalam clause FROM.
- Alias dipisahkan dari nama tabel dengan menggunakan spasi.
- Alias dapat digunakan untuk memenuhi syarat nama kolom jika terjadi ambiguitas.

Contoh: Join sederhana
Tampilkan nama seluruh klien yang telah mengunjungi properti beserta komentarnya.

```
SELECT c.clientNo, fName, lName, propertyNo, comment
  FROM Client c, Viewing v WHERE c.clientNo = v.clientNo;
```

- Hanya baris dari kedua tabel yang mempunyai nilai yang sama pada kolom no. klien ((c.clientNo = v.clientNo) yang akan ditampilkan.
- Equivalen dengan equi-join dalam aljabar relasional.

Table 5.24 Result table for Example 5.24.

clientNo	fName	lName	propertyNo	comment
CR56	Aline	Stewart	PG36	
CR56	Aline	Stewart	PA14	too small
CR56	Aline	Stewart	PG4	
CR62	Mary	Tregear	PA14	no dining room
CR76	John	Kay	PG4	too remote

Alternatif pembentukan JOIN

- SQL menyediakan beberapa cara alternatif untuk menspesifikasikan join :

```
FROM Client c JOIN Viewing v ON c.clientNo = v.clientNo
FROM Client JOIN Viewing USING clientNo
FROM Client NATURAL JOIN Viewing
```

- Pada setiap contoh, FROM menggantikan FROM dan WHERE. bagaimanapun, akan menghasilkan tabel dengan dua kolom no klien yang sama.

Contoh: Mengurutkan join
Untuk setiap cabang, tampilkan nomor dan nama staff yang mengatur property dan nomor property yang mereka atur.

```
SELECT s.branchNo, s.staffNo, fName, lName, propertyNo
  FROM Staff s, PropertyForRent p WHERE s.staffNo = p.staffNo
        ORDER BY s.branchNo, s.staffNo, propertyNo;
```

Table 5.25 Result table for Example 5.25.

branchNo	staffNo	fName	lName	propertyNo
B003	SG14	David	Ford	PG16
B003	SG37	Ann	Beech	PG21
B003	SG37	Ann	Beech	PG36
B005	SL41	Julie	Lee	PL94
B007	SA9	Mary	Howe	PA14

Contoh: Join 3 tabel
Untuk setiap cabang, tampilkan staff yang mengatur property, termasuk kota dimana kantor cabang dan property yang diatur tersebut berada.

```
SELECT b.branchNo, b.city, s.staffNo, fName, lName, propertyNo
    FROM Branch b, Staff s, PropertyForRent p
      WHERE b.branchNo = s.branchNo AND s.staffNo = p.staffNo
            ORDER BY b.branchNo, s.staffNo, propertyNo;
```

Table 5.26 Result table for Example 5.26.

branchNo	city	staffNo	fName	lName	propertyNo
B003	Glasgow	SG14	David	Ford	PG16
B003	Glasgow	SG37	Ann	Beech	PG21
B003	Glasgow	SG37	Ann	Beech	PG36
B005	London	SL41	Julie	Lee	PL94
B007	Aberdeen	SA9	Mary	Howe	PA14

- Formulasi alternatif untuk FROM dan WHERE:

```
FROM (Branch b JOIN Staff s USING branchNo) AS bs
        JOIN PropertyForRent p USING staffNo
```

Contoh: Multiple Grouping Column
Tampilkan jumlah nomor property yang ditangani oleh setiap anggota staff dari setiap cabang

```
SELECT s.branchNo, s.staffNo, COUNT(*) AS count
    FROM Staff s, PropertyForRent p
        WHERE s.staffNo = p.staffNo
            GROUP BY s.branchNo, s.staffNo
                ORDER BY s.branchNo, s.staffNo;
```

Table 5.27(a) Result table for Example 5.27.

branchNo	staffNo	count
B003	SG14	1
B003	SG37	2
B005	SL41	1
B007	SA9	1

Computing a Join

- Prosedur untuk membuat suatu hasil dari join :
  1. Bentuk Cartesian product dari tabel yang disebutkan dalam clause FROM.
  2. Jika terdapat clause WHERE, gunakan kondisi pencarian pada setiap baris dari tabel hasil, dan menyimpan/menghasilkan baris yang memenuhi kondisi.
  3. Untuk setiap baris yang disimpan, tentukan nilai untuk setiap item dalam daftar SELECT untuk menghasilkan satu baris tunggal dalam tabel hasil.
  4. Jika DISTINCT digunakan, akan mengeliminasi duplikasi baris dari tabel hasil.
  5. Jika terdapat clause ORDER BY, tabel hasil akan diurutkan.
- SQL menyediakan format SELECT khusus untuk Cartesian product:

```
SELECT [DISTINCT | ALL] {* | columnList}
        FROM Table1 CROSS JOIN Table2
```

Outer Join

- Jika satu baris dari tabel yang di-join tidak cocok (unmatched), baris tersebut akan dihilangkan dari tabel hasil.
- Operasi outer join menyimpan baris yang tidak memenuhi kondisi join.
- Perhatikan tabel berikut :

Branch1

branchNo	bCity
B003	Glasgow
B004	Bristol
B002	London

PropertyForRent1

propertyNo	pCity
PA14	Aberdeen
PL94	London
PG4	Glasgow

- inner join dari kedua tabel diatas :

```
SELECT b.*, p.*  FROM Branch1 b, PropertyForRent1 p
        WHERE b.bCity = p.pCity;
```

Table 5.27(b) Result table for inner join of Branch1 and PropertyForRent1 tables.

branchNo	bCity	propertyNo	pCity
B003	Glasgow	PG4	Glasgow
B002	London	PL94	London

- Tabel hasil mempunyai dua baris dimana kota bernilai sama.
- Tidak ada baris yang sesuai untuk cabang di Bristol dan Aberdeen.
- Untuk menyertakan baris yang tidak cocok (unmatched rows) dalam tabel hasil, digunakan Outer join.

Contoh: Left Outer Join

Tampilkan cabang dan property yang berada pada kota yang sama beserta cabang yang tidak cocok (unmatched branches)

```
SELECT b.*, p.* FROM Branch1 b
        LEFT JOIN  PropertyForRent1 p ON b.bCity = p.pCity;
```

- Menyertakan baris dari tabel pertama (left) yang tidak cocok (unmatched) dengan tabel yang kedua (right).
- Kolom dari tabel kedua diisi dengan NULL.

Table 5.28 Result table for Example 5.28.

branchNo	bCity	propertyNo	pCity
B003	Glasgow	PG4	Glasgow
B004	Bristol	NULL	NULL
B002	London	PL94	London

Contoh: Right Outer Join

Tampilkan cabang dan property pada kota yang sama dan property yang tidak cocok(unmatched properties).

```
SELECT b.*, p.*  FROM Branch1 b RIGHT JOIN
      PropertyForRent1 p ON b.bCity = p.pCity;
```

- Right Outer join menyertakan baris dari tabel kedua (right) yang tidak sesuai dengan baris pada tabel yang pertama (left).
- Kolom dari tabel pertama diisi dengan NULL.

Table 5.29 Result table for Example 5.29.

branchNo	bCity	propertyNo	pCity
NULL	NULL	PA14	Aberdeen
B003	Glasgow	PG4	Glasgow
B002	London	PL94	London

Contoh: Full Outer Join

Tampilkan cabang dan property yang berada pada kota yang sama dan setiap cabang atau property yang kotanya tidak sesuai.

```
SELECT b.*, p.*  FROM Branch1 b FULL
   JOIN PropertyForRent1 p ON b.bCity = p.pCity;
```

- Menyertakan baris yang tidak sesuai dikedua table.
- Kolom yang tidak sesuai diisi dengan NULL.

Table 5.30 Result table for Example 5.30.

branchNo	bCity	propertyNo	pCity
NULL	NULL	PA14	Aberdeen
B003	Glasgow	PG4	Glasgow
B004	Bristol	NULL	NULL
B002	London	PL94	London

EXISTS dan NOT EXISTS

- EXISTS dan NOT EXISTS digunakan hanya untuk subqueri.
- Menghasilkan hasil sederhana benar/salah.
- Benar jika dan hanya jika terdapat sedikitnya satu baris dalam tabel hasil yang dikembalikan oleh subquery.
- Salah jika subquery mengembalikan nilai kosong.
- NOT EXISTS merupakan kebalikan dari EXISTS.
- Sebagai pemeriksaan (NOT) EXISTS untuk keberadaan atau tidaknya baris dalam tabel hasil subquery, subquery dapat terdiri dari beberapa kolom.
- Perintah untuk subqueri menggunakan (NOT) EXISTS seperti berikut :
- (SELECT * ...)

Contoh: Query menggunakan EXISTS

Tampilkan seluruh staff yang bekerja di cabang London.

```
SELECT staffNo, fName, lName, position
  FROM Staff s  WHERE EXISTS (SELECT *  FROM Branch b
            WHERE s.branchNo = b.branchNo AND city = 'London');
```

Table 5.31 Result table for Example 5.31.

staffNo	fName	lName	position
SL21	John	White	Manager
SL41	Julie	Lee	Assistant

- Kondisi pencarian s.branchNo = b.branchNo perlu untuk mempertimbangan record cabang yang benar untuk setiap anggota staff.
- Jika dihilangkan, akan menghasilkan seluruh record staff karena subquery menjadi :
  SELECT * FROM Branch WHERE city='London'

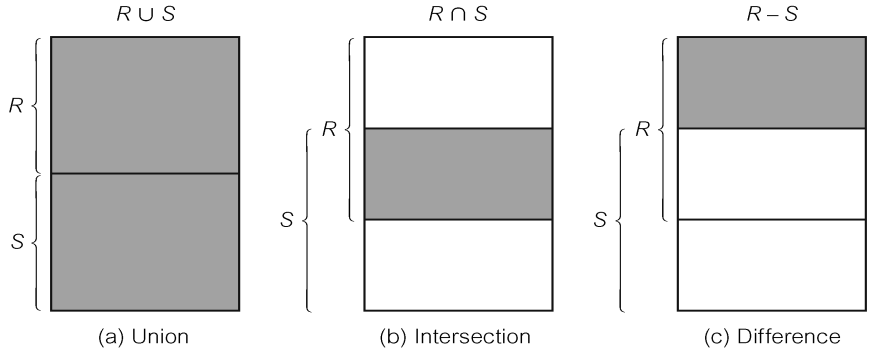
- akan selalu bernilai benar dan query akan menjadi :
  SELECT staffNo, fName, lName, position FROM Staff WHERE true;

- Dapat juga dituliskan dengan menggunakan penggabungan:
  SELECT staffNo, fName, lName, position FROM Staff s, Branch b
  WHERE s.branchNo = b.branchNo AND city = 'London';

Union, Intersect, dan Difference (Except)

- Dapat menggunakan operasi himpunan seperti Union, Intersection, dan Difference untuk kombinasi hasil dari dua atau lebih query menjadi satu tabel.
- Union dari dua tabel, A dan B, maka tabel hasil akan terdiri dari seluruh baris yang ada pada A atau B.
- Intersection, tabel hasil berisi seluruh baris yang ada pada tabel A dan B.
- Difference, tabel hasil berisi seluruh baris yang ada di tabel A tetapi tidak terdapat di B.
- Kedua tabel tersebut harus *union compatible*.
- Format dari setiap clause operator :
  op [ALL] [CORRESPONDING [BY {column1 [, ...]}]]

- Jika disebutkan If CORRESPONDING BY, maka operasi himpunan ditampilkan berdasarkan nama kolom yang disebutkan.
- Jika disebutkan If CORRESPONDING tanpa clause BY, operasi ditampilkan dengan nama kolom yang ada.
- Jika ALL dituliskan, hasil akan mengandung baris duplikat.



(a) Union (b) Intersection (c) Difference

Contoh penggunaan UNION

Tampilkan seluruh kota yang terdapat kantor cabang ataupun property.

```
(SELECT city FROM Branch
      WHERE city IS NOT NULL) UNION
            (SELECT city FROM PropertyForRent
                  WHERE city IS NOT NULL);
```

– Or
(SELECT * FROM Branch
WHERE city IS NOT NULL) UNION
CORRESPONDING BY city
(SELECT * FROM PropertyForRent
WHERE city IS NOT NULL);

- Akan menghasilkan tabel dari kedua query, kemudian digabungkan.

Table 5.32 Result table for Example 5.32.

city
London
Glasgow
Aberdeen
Bristol

Contoh penggunaan INTERSECT

Tampilkan seluruh kota yang terdapat kantor cabang dan property.
(SELECT city FROM Branch) INTERSECT (SELECT city FROM PropertyForRent);
Or
(SELECT * FROM Branch) INTERSECT CORRESPONDING BY city
(SELECT * FROM PropertyForRent);

Table 5.33 Result table for Example 5.33.

city
Aberdeen
Glasgow
London

Dapat juga dituliskan tanpa operator INTERSECT :
SELECT b.city FROM Branch b PropertyForRent p WHERE b.city = p.city;
Or
SELECT DISTINCT city FROM Branch b WHERE EXISTS
(SELECT * FROM PropertyForRent p WHERE p.city = b.city);

Contoh penggunaan EXCEPT

Tampilkan seluruh kota yang terdapat kantor cabang tetapi tidak ada property.
(SELECT city FROM Branch) EXCEPT (SELECT city FROM PropertyForRent);
Or
(SELECT * FROM Branch) EXCEPT CORRESPONDING BY city
(SELECT * FROM PropertyForRent);

- Dapat dituliskan tanpa operator EXCEPT:
SELECT DISTINCT city FROM Branch WHERE city NOT IN
(SELECT city FROM PropertyForRent);
Or

SELECT DISTINCT city FROM Branch b
WHERE NOT EXISTS
(SELECT * FROM PropertyForRent p
WHERE p.city = b.city);

Table 5.34 Result table for Example 5.34.

city
Bristol

INSERT…VALUES
Digunakan untuk memasukan nilai record pada suatu tabel.

Bentuk umum :
INSERT INTO TableName [(columnList)] VALUES (dataValueList)

Dimana :
- *columnList* bersifat optional; jika dihilangkan, SQL mengasumsikan seluruh kolom seperti yang terdapat pada CREATE TABLE.
- Kolom yang dihilangkan, harus dideklarasikan sebagai NULL ketika tabel dibuat, kecuali jika ditentukan nilai DEFAULT pada saat pembuatan kolom.
- *dataValueList* harus sesuai dengan *columnList* seperti :
    - Jumlah items pada setiap *list* harus sama.
    - Posis untuk setiap item dalam *list* harus sesuai;
    - tipe data setiap item dalam *dataValueList* harus sesuai dengan tipe data dari kolom yang dimaksud.

Contoh INSERT … VALUES
Masukkan baris baru ke dalam tabel Staff untuk seluruh kolom

```
INSERT INTO Staff
   VALUES ('SG16', 'Alan', 'Brown', 'Assistant', 'M', Date'1957-05-25', 8300, 'B003');
```

Contoh INSERT dengan Default
Masukkan baris baru ke dalam tabel staff untuk kolom-kolom tertentu

```
INSERT INTO Staff (staffNo, fName, lName, position, salary, branchNo)
   VALUES ('SG44', 'Anne', 'Jones',  'Assistant', 8100, 'B003');
-- Or
   INSERT INTO Staff
         VALUES ('SG44', 'Anne', 'Jones', 'Assistant', NULL, NULL, 8100, 'B003');
```

INSERT … SELECT
Bentuk kedua dari INSERT, memungkinkan banyak baris disalin dari satu tabel ke tabel lainnya.

bentuk umum :
```
INSERT INTO TableName [ (columnList) ]
   SELECT ...
```

Contoh INSERT … SELECT
Diasumsikan terdapat tabel StaffPropCount yang berisi nama staff dan nomor property yang mereka atur

StaffPropCount(staffNo, fName, lName, propCnt)

Tambahkan StaffPropCount menggunakan tabel Staff dan PropertyForRent :
```
INSERT INTO StaffPropCount (SELECT s.staffNo, fName, lName, COUNT(*)
      FROM Staff s, PropertyForRent p
            WHERE s.staffNo = p.staffNo
                  GROUP BY s.staffNo, fName, lName)
```

```
UNION
        (SELECT staffNo, fName, lName, 0  FROM Staff
                WHERE staffNo NOT IN
                        (SELECT DISTINCT staffNo      FROM PropertyForRent));
```

Table 5.35 Result table for Example 5.37.

staffNo	fName	lName	propCount
SG14	David	Ford	1
SL21	John	White	0
SG37	Ann	Beech	2
SA9	Mary	Howe	1
SG5	Susan	Brand	0
SL41	Julie	Lee	1

- Jika bagian kedua dari UNION dihilangkan, maka staff yang tidak mengatur property tidak akan ditampilkan.

UPDATE
Digunakan untuk memodifikasi nilai dari suatu record dalam tabel dengan kondisi tertentu

Bentuk umum :

```
UPDATE TableName
  SET columnName1 = dataValue1
        [, columnName2 = dataValue2...]
  [WHERE searchCondition]
```

Dimana :
- *TableName* dapat berupa nama tabel asal atau *updatable view*.
- Clause SET menspesifikasikan nama dari satu atau lebih kolo yang akan di-update.
- Clause WHERE Bersifat optional:
  - Jika dihilangkan, kolom yang disebutkan akan di-update untuk seluruh baris dalam tabel.
  - Jika disebutkan, hanya baris yang memenuhi syarat saja yang akan di-update.
- *dataValue* yang baru harus sesuai dengan tipe data pada kolom yang bersangkutan.

Contoh UPDATE seluruh baris
Berikan pada seluruh staff kenaikan sebesar 3%

```
UPDATE Staff SET salary = salary*1.03;
```

Berikan seluruh manager kenaikan sebesar 5%

```
UPDATE Staff   SET salary = salary*1.05  WHERE position = 'Manager';
```

Contoh UPDATE banyak kolom
Naikan jabatan David Ford (staffNo='SG14') menjadi Manager dan ubah gajinya menjadi £18,000.

```
UPDATE Staff   SET position = 'Manager', salary = 18000
        WHERE staffNo = 'SG14';
```

DELETE
Digunakan untuk menghapus record dario tabel jika memenuhi kondisi tertentu

Bentuk umum :

```
DELETE FROM TableName
  [WHERE searchCondition]
```

Dimana :

- *TableName* dapat berupa nama tari tabel asal atau *updatable view.*
- *searchCondition* bersifat optional; Jika dihilangkan, maka seluruh baris akan dihapus, tetapi tidak menghapus tabelnya. Jika disebutkan, hanya baris yang memenuhi syarat saja yang akan dihapus.

Contoh DELETE baris tertentu :

Hapus seluruh viewings yang terkait dengan properti PG4.

```
DELETE FROM Viewing
        WHERE propertyNo = 'PG4';
```

Hapus seluruh record dari tabel Viewing

```
DELETE FROM Viewing;
```