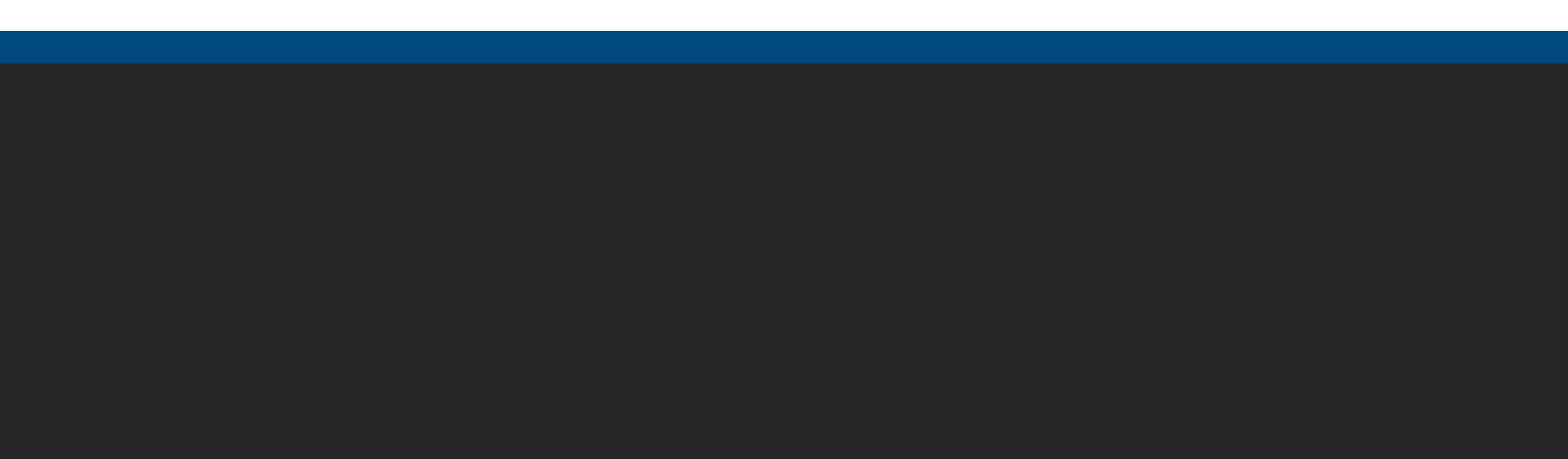
***JTI POLINEMA***

***Team Teaching Basis Data Lanjut 1***



JOBSHEET

**PRAKTIKUM BASIS DATA LANJUT**

**Jurusan Teknologi Informasi POLITEKNIK NEGERI MALANG**

**PERTEMUAN 10**

**SQL SERVER - OPERASI HIMPUNAN & TRIGGER**

|  |  |
| --- | --- |
|  | Jurusan Teknologi Informasi Politeknik Negeri Malang  **Jobsheet 10: Operasi Himpunan & Trigger Mata Kuliah Basis Data Lanjut**  **Pengampu:** Tim Ajar Basis Data Lanjut |

**Tujuan**

Mahasiswa diharapkan dapat:

1. Menerapkan query UNION dan UNION ALL
2. Menerapkan query CROSS APPLY dan OUTER APPLY
3. Menerapkan query EXCEPT dan INTERSECT
4. Menerapkan query TRIGGER (AFTER)
5. Menerapkan query TRIGGER (INSTEAD OF)

**Petunjuk Umum**

1. Ikuti langkah-langkah pada bagian-bagian praktikum sesuai dengan urutan yang diberikan.
2. Jawablah semua pertanyaan bertanda [Soal-X] yang terdapat pada langkah-langkah tertentu di setiap bagian praktikum.
3. Dalam setiap langkah pada praktikum terdapat penjelasan yang akan membantu Anda dalam menjawab pertanyaan-pertanyaan pada petunjuk nomor 3, maka baca dan kerjakanlah semua bagian praktikum dalam jobsheet ini.
4. Tulis jawaban dari soal-soal pada petunjuk nomor 3 pada sebuah laporan yang dikerjakan menggunakan aplikasi word processing (Word, OpenOffice, atau yang lain yang sejenis). Ekspor sebagai file **PDF** dengan format nama sebagai berikut:
   * **10\_Prak.BDL\_Kelas\_NamaLengkapAnda**.pdf
   * Kumpulkan file PDF tersebut sebagai laporan praktikum kepada dosen pengampu.
   * Selain pada nama file, cantumkan juga identitas Anda pada tiap footer halaman laporan tersebut.

**Praktikum – Bagian 1: UNION & UNION ALL**

|  |  |
| --- | --- |
| **Langkah** | **Keterangan** |
| **1** | Berikut ini adalah sebuah SQL kueri ke tabel 'Production.Products' yang akan menampilkan 'productid' dan 'productname', khusus bagi product yang memiliki 'categoryid' bernilai 4!    Ketik dan eksekusi SQL tersebut dan pastikan hasilnya sesuai dengan gambar berikut: |
| **2** | SQL berikut ini adalah SQL yang menampilkan 'productid' dan 'productname' dari tabel 'Production.Products'. Hasil dari query ini di-*filter* sedemikian rupa sehingga yang tampil hanyalah product yang telah memiliki nilai jual total lebih dari $50.000. |

|  |  |
| --- | --- |
|  | **Keterangan**: Untuk mendapatkan nilai jual total, SQL diatas bekerja dengan cara sebagai berikut:   1. Meng-Inner-joinkan tabel 'Production.Products' dengan tabel 'Sales.OrderDetails' karena data penjualan ada di tabel yang terakhir. 2. Melakukan GROUP BY, berdasarkan 'productid' dan 'productname'-nya 3. Dan yang terakhir, mem-*filter* grup menggunakan HAVING dengan kondisi data yang   **totalNilaiPenjualannya** > 50000   1. Dimana **totalNilaiPenjualan** = ('unitprice' × 'qty')   Eksekusilah SQL diatas tadi dan pastikan hasilnya sesuai dengan gambar berikut: |
| **3** | [Soal-1] Tulis sebuah SQL yang menampilkan hasil pada praktikum-1 langkah-1 & 2 secara sekaligus (gabungan) dengan menggunakan **UNION**!  **Petunjuk**: Letakkan UNION diantara kedua SQL tersebut. Pastikan hasilnya sesuai dengan gambar berikut: |

|  |  |
| --- | --- |
|  |  |
| **4** | [Soal-2] Serupa dengan langkah sebelumnya, kali ini tulislah sebuah SQL yang menampilkan hasil pada praktikum-1 langkah-1 & 2 secara sekaligus (gabungan) dengan menggunakan **UNION ALL**!  Pastikan hasilnya sesuai dengan gambar berikut: |
| **5** | [Soal-3] Apa bedanya UNION & UNION ALL?  **Jawaban: UNION menggabungkan hasil dua query dan menghapus duplikat, sedangkan UNION ALL menggabungkan semua hasil tanpa menghapus duplikat.** |
| **6** | [Soal-4] Tuliskan SQL untuk menampilkan 10 pelanggan dengan nilai pembelian tertinggi pada bulan Januari 2008 serta 10 tertinggi pada bulan Februari 2008. |

|  |  |
| --- | --- |
|  | **Petunjuk**:   1. Buat dahulu query untuk menampilkan data yang bulan-nya Januari lalu UNION-kan dengan bulan Februari. 2. Pada tiap-tiap bulan lakukan INNER JOIN antara tabel 'Sales.Customers' & 'Sales.OrderValue'   Pastikan hasilnya sesuai dengan gambar berikut: |

**Praktikum – Bagian 2: CROSS APPLY & OUTER APPLY**

|  |  |
| --- | --- |
| **Langkah** | **Keterangan** |
| **1** | **APPLY**: Adalah fasilitas yang memungkinkan kita untuk menerapkan inner-query maupun TVF (Table-Valued Function) ke setiap hasil yang didapat oleh outer-querynya.  APPLY ada 2:   1. CROSS APPLY: Hanya mengembalikan baris yang inner-query atau TVF-nya ada nilainya 2. OUTER APPLY: Mengembalikan baris baik yang inner-query atau TVF-nya ada nilainya maupun tidak. Kalau tidak ada hasilnya, maka digandeng-kan dengan NULL.   Supaya Anda paham, ketik dan eksekusilah SQL berikut ini lalu perhatikan hasilnya!\    Pastikan hasilnya sesuai dengan gambar berikut: |

|  |  |
| --- | --- |
|  | **Keterangan:** SQL diatas menampilkan 'orderid' dan 'productname' dari produk-produk yang ada di tabel 'Production.Products' dengan mencantumkan 'orderid' dari 2 pemesanan terakhir yang melibatkan masing-masing produk. |
| **2** | Untuk memahami penggunaan APPLY dengan TVF, tulis dan eksekusilah SQL berikut ini: |

|  |  |
| --- | --- |
|  | Eksekusilah SQL diatas terlebih dahulu supaya fungsinya tersimpan di database.  Kemudian tulis dan jalankan SQL dibawah ini untuk menampilkan data 3 pembelian teratas yang dilakukan oleh customer pada tabel 'Sales.Customers' dimana data tersebut meliputi 'productid', 'productname', & 'totalsalesamount'.    **Keterangan**:   1. Pada SQL tersebut digunakan fungsi '**fnGetTop3ProductsForCustomer**' yang dibuat sebelumnya untuk mendapatkan 3 baris 'productid', 'productname', 'totalsalesamount' milik masing-masing customer. 2. Pada SQL tersebut digunakan **CROSS APPLY**, sehingga setiap 'custid' yang dihasilkan oleh outer-querynya, akan mendapatkan 3 hasil karena fungsi '**fnGetTop3ProductsForCustomer**' mengembalikan 3 hasil.   Pastikan hasilnya seperti pada gambar berikut:  C:\Users\Yoppy Yunhasnawa\OneDrive\Ngajar Polinema\Ngajar SMBD\08. Operasi Himpunan & Triggers\Jobsheet_IMG\06. 3 Pembelian Terakhir Customer.PNG |

|  |  |
| --- | --- |
| **3** | [Soal-5] Ubahlah CROSS APPLY pada Praktikum bagian-2.2 menjadi **OUTER APPLY**, sehingga hasilnya menjadi seperti gambar berikut:  C:\Users\Yoppy Yunhasnawa\OneDrive\Ngajar Polinema\Ngajar SMBD\08. Operasi Himpunan & Triggers\Jobsheet_IMG\07. 3 Pembelian Terakhir Customer + Customer yang belum pernah beli.PNG  Perhatikan hasilnya, sekarang ada 267 baris. 2 baris tambahan yang muncul tersebut adalah pelanggan-pelanggan yang belum pernah melakukan pembelian, sehingga fungsi 'fnGetTop3ProductsForCustomer()' mengembalikan nilai **NULL**. |
| **4** | [Soal-6] Modifikasilah SQL yang telah Anda buat dari bagian sebelumnya sehingga SQL tersebut HANYA menampilkan customer yang tidak pernah membeli produk.  **Petunjuk**: Tambahkan WHERE **<?>** IS NULL. Dimana **<?>** adalah sebuah nama kolom. Pastikan hasilnya seperti gambar berikut:  C:\Users\Yoppy Yunhasnawa\OneDrive\Ngajar Polinema\Ngajar SMBD\08. Operasi Himpunan & Triggers\Jobsheet_IMG\08. Customer yang tidak pernah beli.PNG |
| **5** | Hapus fungsi yang dibuat pada praktikum bagian 2.2 dengan mengeksekusi SQL berikut:  C:\Users\Yoppy Yunhasnawa\OneDrive\Ngajar Polinema\Ngajar SMBD\08. Operasi Himpunan & Triggers\Jobsheet_IMG\09. Hapus fungsi top3.PNG |

**Praktikum – Bagian 3: EXCEPT & INTERSECT**

|  |  |
| --- | --- |
| **Langkah** | **Keterangan** |
| **1** | Berikut ini adalah sebuah statement SELECT yang menampilkan kolom 'custid' dari tabel Sales.Orders. Dimana hasilnya difilter lagi sehingga yang tampil hanyalah pelanggan yang sudah membeli lebih dari 20 produk yang berbeda (berdasarkan kolom 'productid' dari tabel 'Sales.OrderDetails').    Ketik dan eksekusilah SQL tersebut sehingga hasilnya serupa dengan gambar berikut:  C:\Users\Yoppy Yunhasnawa\OneDrive\Ngajar Polinema\Ngajar SMBD\08. Operasi Himpunan & Triggers\Jobsheet_IMG\10. Customer dgn 20 produk berbeda.PNG |

|  |  |
| --- | --- |
| **2** | [Soal-7] Buatlah sebuah statement SELECT yang menampilkan kolom 'custid' dari tabel 'Sales.Orders'. Saring hasilnya sehingga yang tampil hanyalah pelanggan yang berasal dari USA kecuali SEMUA pelanggan yang muncul pada hasil query pada praktikum bagian 3.1.  **Petunjuk**: Tambahkan sebuah query untuk mendapatkan customer dari USA dan tambahkan operator EXCEPT didepan query praktikum-3 langkah-1.  Pastikan hasilnya seperti pada gambar berikut:  C:\Users\Yoppy Yunhasnawa\OneDrive\Ngajar Polinema\Ngajar SMBD\08. Operasi Himpunan & Triggers\Jobsheet_IMG\11. Customer dari USA kecuali semua yang dgn 20 produk berbeda.PNG |
| **3** | Berikut ini adalah sebuah statement SELECT yang menampilkan kolom 'custid' dari tabel 'Sales.Orders'. Hasilnya kemudian di-filter sedemikian rupa sehingga hanya customer yang telah berbelanja **lebih dari $10.000** yang tampil. Nilai belanja customer-customer tersebut didapatkan dari perkalian kolom 'qty' dan 'unitprice' yang ada di tabel 'Sales.OrderDetails'.    Ketik dan eksekusi SQL diatas lalu pastikan hasilnya seperti pada gambar berikut: |
|  | C:\Users\Yoppy Yunhasnawa\OneDrive\Ngajar Polinema\Ngajar SMBD\08. Operasi Himpunan & Triggers\Jobsheet_IMG\12. Customer dgn pengeluaran lbh dari 10,000.PNG |
| **4** | [Soal-8] Salin SQL pada bagian 3.1 tambahkan operator INTERSECT dibelakangnya, kemudian salin-tempel SQL pada bagian 3.3 dibelakang operator INTERSECT tadi. Jalankan, dan perhatikan hasilnya.  Pastikan hasilnya seperti pada gambar berikut:  C:\Users\Yoppy Yunhasnawa\OneDrive\Ngajar Polinema\Ngajar SMBD\08. Operasi Himpunan & Triggers\Jobsheet_IMG\13. EXCEPT & INTERSECT.PNG |

|  |  |
| --- | --- |
|  | [Soal-9] Dapatkah Anda menyimpulkan, customer yang bagaimana yang tampil pada hasil query bagian 3.4 ini?  **Jawaban: hasil query ini akan menampilkan ID pelanggan (custid) yang memiliki total transaksi lebih dari 10.000 dan yang pernah membeli lebih dari 20 jenis produk unik.** |
| **5** | [Soal-10] Salin keseluruhan query pada bagian-3.4, modifikasi SQL tersebut dengan cara mengapit statement SELECT sebelum operator INTERSECT dengan tanda kurung '(' dan ')'.  Eksekusilah SQL tersebut dan pastikan hasilnya seperti pada gambar berikut:  C:\Users\Yoppy Yunhasnawa\OneDrive\Ngajar Polinema\Ngajar SMBD\08. Operasi Himpunan & Triggers\Jobsheet_IMG\14. Presedence.PNG    [Soal-11] Apakah hasilnya berbeda dengan SQL pada bagian-3.4? Mengapa demikian? Dapatkah Anda menjelaskan tentang urutan prioritas (*precedence*) operator yang digunakan pada SQL bagian ini?  **Jawaban: Kedua query SQL akan menghasilkan output yang identik. Perbedaan utamanya hanya terletak pada strukturnya, di mana query kedua menggunakan subquery yang diberi alias 'result'. Meskipun menggunakan subquery, hal ini tidak mengubah data yang dihasilkan. Subquery tersebut hanya berfungsi sebagai pembungkus untuk operasi INTERSECT. Walaupun subquery menggunakan SELECT \* untuk mengambil semua kolom, pada kenyataannya data yang diproses tetap hanya kolom custid saja. Dengan kata lain, hasil akhir tetap sama - yaitu daftar custid yang memenuhi kedua kondisi yang diminta.** |

**Praktikum – Bagian 4: TRIGGER (AFTER)**

|  |  |
| --- | --- |
| **Langkah** | **Keterangan** |
| **1** | TRIGGER: Trigger adalah semacam stored procedure (fungsi yang tidak mengembalikan nilai) spesial yang akan dieksekusi ketika ada sebuah event yang terjadi pada suatu tabel.  Trigger ada 2:   * TRIGGER **AFTER**: Trigger yang MENAMBAHKAN suatu aksi * TRIGGER **INSTEAD OF**: Trigger yang MENCEGAH suatu aksi   Trigger **AFTER INSERT**: Adalah trigger yang akan dieksekusi ketika ada operasi INSERT berhasil (selesai, after) dilakukan pada tabel yang dipasangi trigger tersebut.  Misalkan kita ingin membuat, ketika tabel pemesanan (Sales.Orders) diisi, maka secara otomatis tabel detailnya diisi dengan data default, maka kita bisa menggunakan TRIGGER AFTER INSERT.  Ketikkan SQL berikut pada SSMS dan eksekusilah!    Jalankan SQL berikut untuk menambahkan data baru ke tabel Sales.Orders sehingga memicu ter-eksekusinya Trigger yang kita buat diatas tadi. |

|  |  |
| --- | --- |
|  | Jika benar, maka akan menampilkan pesan berikut:    Serta hasil seperti dibawah: |
| **2** | Trigger **AFTER UPDATE**: Adalah trigger yang akan dieksekusi ketika ada operasi UPDATE berhasil (selesai, after) dilakukan pada tabel yang dipasangi trigger tersebut.  **Contoh kasus**: Misalkan pada tabel 'Sales.OrderDetails' terdapat kolom 'unitprice' dimana kolom ini mengacu pada kolom yang sama pada 'Production.Product'. Akan tetapi, jika pada tabel 'Production.Products' kita ubah 'unitprice' sebuah produk, 'unitprice' yang ada di 'Sales.OrderDetails' tidak otomatis berubah. Agar harga di tabel 'OrderDetails' otomatis berubah ketika tabel 'Products' diupdate kita dapat menggunakan TRIGGER AFTER UPDATE.  Jalankan SQL berikut untuk membuat TRIGGER yang menyelesaikan contoh kasus diatas: |

|  |  |
| --- | --- |
|  | Eksekusilah SQL berikut, untuk mengetes TRIGGER yang telah Anda buat tadi:    Sehinga menghasilkan pesan seperti dibawah ini:  C:\Users\Yoppy Yunhasnawa\OneDrive\Ngajar Polinema\Ngajar SMBD\08. Operasi Himpunan & Triggers\Jobsheet_IMG\TRIGGERS\03. Trigger AFTER UPDATE Messages.PNG  Dan hasil seperti berikut: |

|  |  |
| --- | --- |
|  | C:\Users\Yoppy Yunhasnawa\OneDrive\Ngajar Polinema\Ngajar SMBD\08. Operasi Himpunan & Triggers\Jobsheet_IMG\TRIGGERS\04. Trigger Table Result.PNG |
| **3** | Trigger **AFTER DELETE**: Adalah TRIGGER yang dieksekusi ketika sebuah operasi DELETE dilakukan pada suatu tabel.  **Contoh kasus**: Perhatikan tabel 'Sales.OrderDetails', pada tabel tersebut terdapat kolom 'productid' yang merupakan Foreign Key yang mengacu pada tabel 'Production.Products'. Misalkan kita ingin supaya: ketika sebuah 'productid' dihapus semuanya dari tabel 'OrderDetails' maka kolom 'discontinued' diubah nilainya menjadi '1', kita dapat menggunakan TRIGGER AFTER DELETE.  [Soal-12] Buatlah TRIGGER yang dapat menyelesaikan permasalahan pada contoh kasus diatas!  Lalu jalankan SQL berikut agar TRIGGER yang Anda buat tereksekusi:      Pastikan ***message***-nya seperti berikut:      Dan ***result***-nya seperti dibawah: |

**Praktikum – Bagian 5: TRIGGER (INSTEAD OF)**

|  |  |  |
| --- | --- | --- |
| **Langkah** | **Keterangan** | |
| **1** | Buat dulu tabel backup dengan cara membuka dan mengeksekusi file 'SQLQuery- EmployeesBackup.sql' yang disertakan bersama jobsheet ini.  ***Lokasi****: <Folder Jobsheet>\Resources\SQLQuery-EmployeesBackup.sql*  Isi tabel HR.EmployeesBackup dengan isi yang sama persis dari tabel HR.Employees dengan cara mengeksekusi SQL berikut | |
| **2** | Trigger **INSTEAD OF INSERT**: Trigger ini akan mencegah user melakukan insert pada tabel 'HR.Employee', alih-alih membiarkan INSERT terjadi pada tabel tersebut, trigger berikut ini akan 'membelokkan' data yang diinsert ke tabel 'HR.EmployeesBackup' yang kita buat sebelumnya.  Buatlah TRIGGER yang menyelesaikan permasalahan diatas dengan mengeksekusi SQL berikut: | |
| ***Team Teaching Basis Data Lanjut 21*** | |  |

|  |  |
| --- | --- |
|  | Lalu tes TRIGGER tadi dengan mengeksekusi SQL INSERT berikut:    Akan menghasilkan baris baru pada tabel 'EmployeesBackup' dan tabel 'Employees' tidak akan ada perubahan. |
| **3** | Trigger **INSTEAD OF UPDATE**: Mencegah user melakukan UPDATE pada suatu tabel.  [Soal-13] Dengan cara yang serupa dengan langkah sebelumnya, buatlah TRIGGER yang mencegah user melakukan UPDATE ke table 'HR.Employee'. Ketika ada UPDATE yang terjadi, terapkan hasilnya ke tabel 'HR.EmployeesBackup'!  Lalu jalankan SQL berikut agar TRIGGER yang Anda buat tereksekusi:    Apabila TRIGGER yang Anda buat benar maka ***message***-nya akan tampil seperti berikut: |
| **4** | Trigger **INSTEAD OF DELETE**: Mencegah user melakukan DELETE pada suatu tabel.  [Soal-14] Buatlah TRIGGER yang mencegah user melakukan DELETE ke table 'HR.Employee'. Ketika ada DELETE yang terjadi, jangan biarkan ada data pada tabel tersebut yang hilang!  Hapus data yang sama 'HR.EmployeesBackup'!  Lalu jalankan SQL berikut agar TRIGGER yang Anda buat tereksekusi: |

|  |  |
| --- | --- |
|  | Apabila TRIGGER yang Anda buat benar maka ***message***-nya akan tampil seperti berikut: Dan akan mehapus 1 baris pada table *backup*, sementara di tabel aslinya datanya tetap ada. |

***--- Selamat Mengerjakan ----***