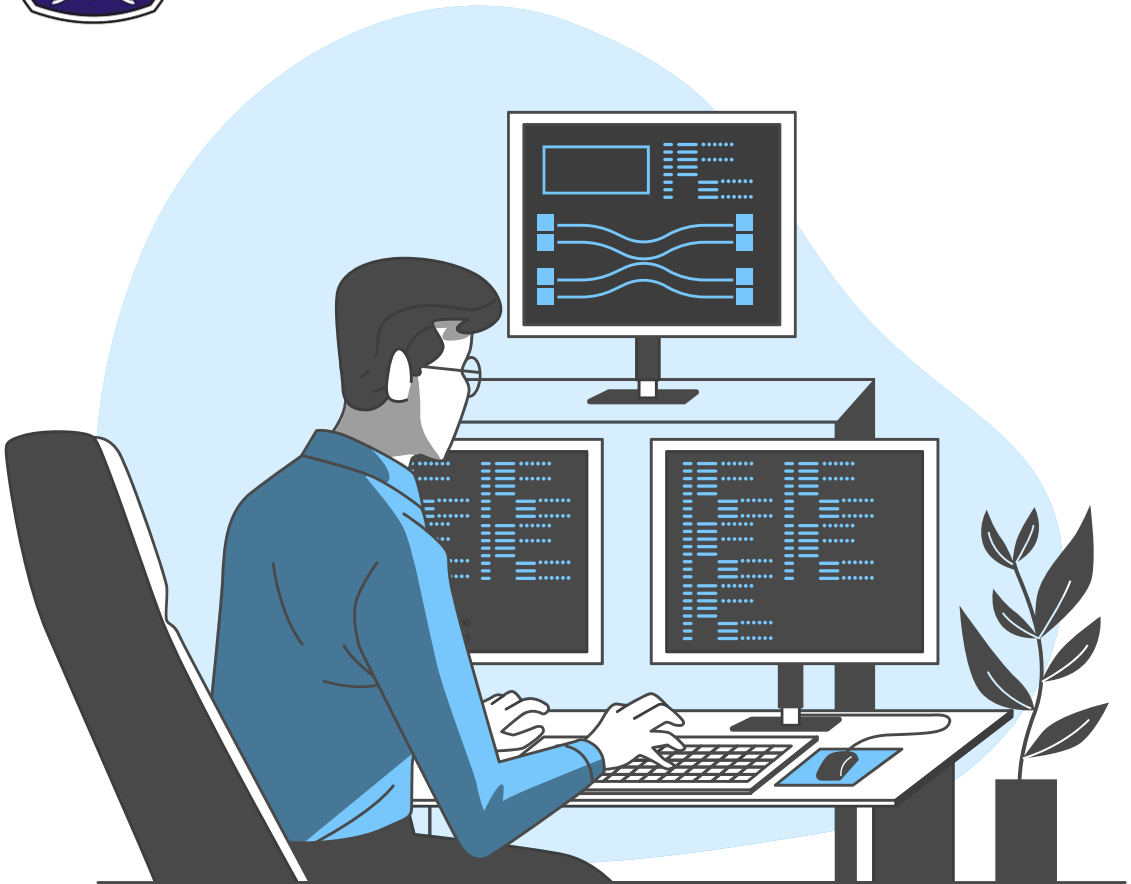


BASIS DATA LANJUT

Pertemuan 10

Operasi Himpunan dan Trigger

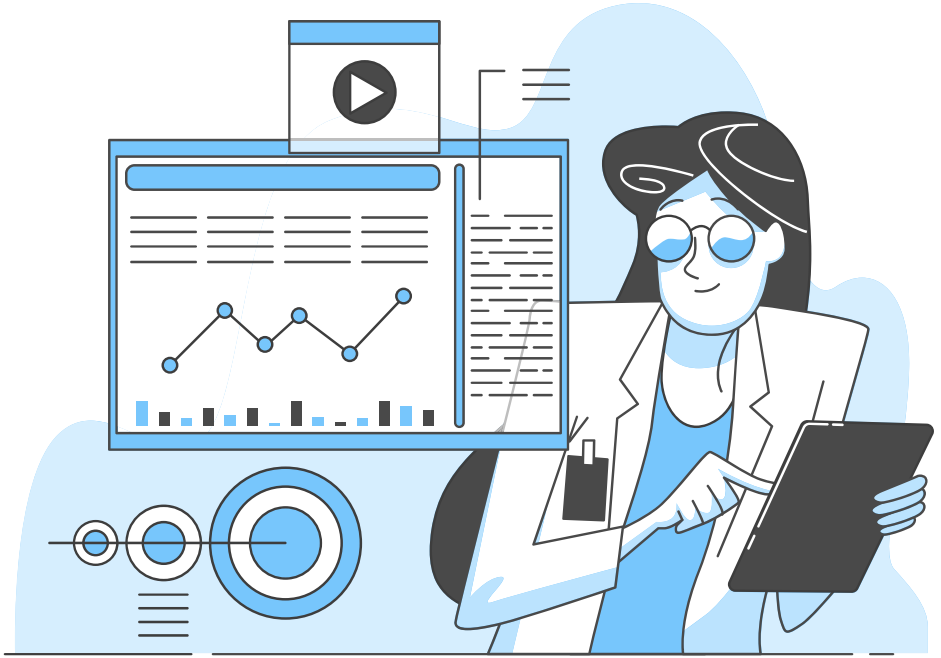
Team Teaching Basis Data Lanjut
JTI - Polinema

Topik

01 OPERASI HIMPUNAN

02 APPLY OPERATOR

03 TRIGGER



Tujuan

- Mahasiswa mampu untuk mengimplementasikan Operasi Himpunan pada SQL Server
- Mahasiswa mampu untuk mengimplementasikan Trigger pada SQL Server

Operasi Himpunan

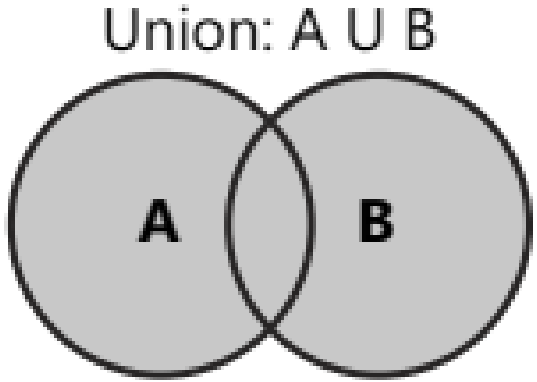
Operasi Himpunan

- Pada ekspresi SQL, terdapat operasi yang berhubungan dengan himpunan pada atribut yang sama.
- Operasi SQL tersebut yaitu: UNION [ALL], INTERSECT dan EXCEPT yang masing-masing memiliki hubungan erat dengan operasi aljabar relasional ∩, ∪, dan -.
- Bentuk umum dari operasi himpunan, adalah:

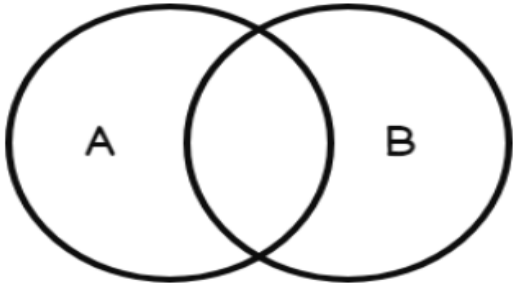
```
Input Query1
<set_operator>
Input Query2
[ORDER BY ...]
```

UNION Operator

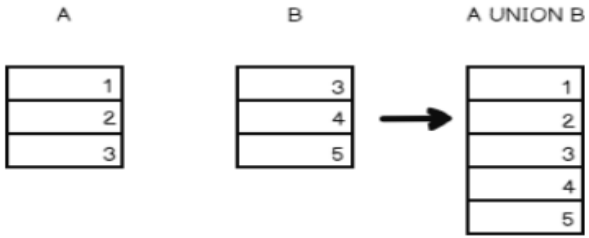
- Dalam teori himpunan, penyatuan dua himpunan adalah himpunan yang berisi semua elemen.
- Dalam T-SQL, operator UNION menyatukan hasil dari dua kueri input. Jika sebuah baris muncul di salah satu set input, maka akan muncul di hasil operator UNION.
- T-SQL mendukung UNION ALL dan UNION (implisit DISTINCT) dari operator UNION



UNION Operator



For example, the table 'A' has 1,2, and 3 and the table 'B' has 3,4,5.



Contoh :

```
(
SELECT 1 ID
   UNION
   SELECT 2
   UNION
   SELECT 3
)
UNION
(
   SELECT 3
   UNION
   SELECT 4
   UNION
   SELECT 5
);
```

UNION ALL Operator

- UNION ALL operator multiset mengembalikan semua baris yang muncul di sembarang multiset input yang dihasilkan dari dua kueri input, tanpa benar-benar membandingkan baris dan tanpa menghilangkan duplikat.
- Dengan asumsi bahwa Query1 mengembalikan baris m dan Query2 mengembalikan n baris, Query1 UNION ALL Query2 mengembalikan baris m + n.

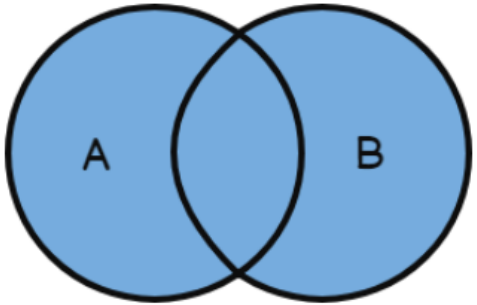
Contoh UNION ALL

```
USE TSQL2012;

SELECT country, region, city FROM HR.Employees
UNION ALL
SELECT country, region, city FROM Sales.Customers;
```

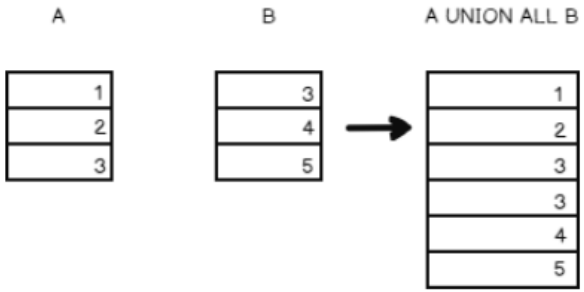
- Perintah tersebut akan menghasilkan 100 baris yang terdiri dari 9 baris berasal dari tabel Employees dan 91 baris dari tabel Customers

```
country         region          city
--------------- --------------- ---------------
USA             WA              Seattle
USA             WA              Tacoma
USA             WA              Kirkland
USA             WA              Redmond
UK              NULL            London
UK              NULL            London
UK              NULL            London
...
Finland         NULL            Oulu
Brazil          SP              Resende
USA             WA              Seattle
Finland         NULL            Helsinki
Poland          NULL            Warszawa

(100 row(s) affected)
```

UNION ALL



For example, the table 'A' has 1,2, and 3 and the table 'B' has 3,4,5.



Contoh :

```
(
SELECT 1 ID
   UNION
   SELECT 2
   UNION
   SELECT 3
)
UNION ALL
(
   SELECT 3
   UNION
   SELECT 4
   UNION
   SELECT 5
);
```

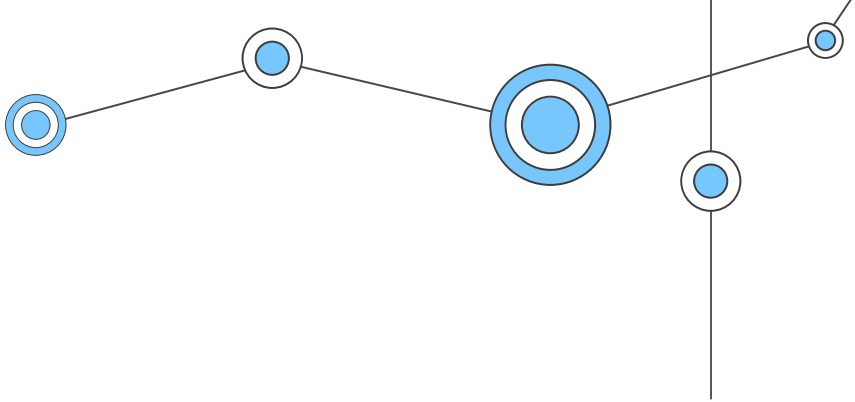
Contoh UNION

```
SELECT country, region, city FROM HR.Employees
UNION
SELECT country, region, city FROM Sales.Customers;
```

- Perintah tersebut akan menghasilkan 71 baris dikarenakan pada perintah UNION akan menghapus data yang duplikat.

```
country          region           city
---------------  ---------------  ---------------
Argentina        NULL             Buenos Aires
Austria          NULL             Graz
Austria          NULL             Salzburg
Belgium          NULL             Bruxelles
Belgium          NULL             Charleroi
...
USA              WY               Lander
Venezuela        DF               Caracas
Venezuela        Lara             Barquisimeto
Venezuela        Nueva Esparta    I. de Margarita
Venezuela        Táchira          San Cristóbal

(71 row(s) affected)
```

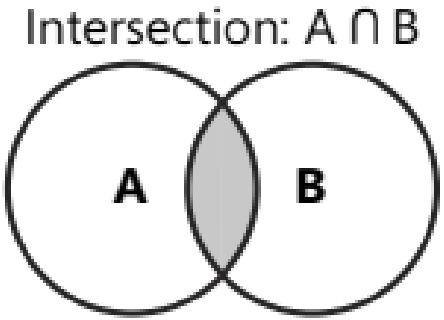
SYNTAX



```
SELECT Column1, Column2, … ColumnN
FROM <table>
[WHERE conditions]
[GROUP BY Column(s]]
[HAVING condition(s)]
UNION
SELECT Column1, Column2, … ColumnN
FROM table
[WHERE condition(s)];
ORDER BY Column1,Column2…
```

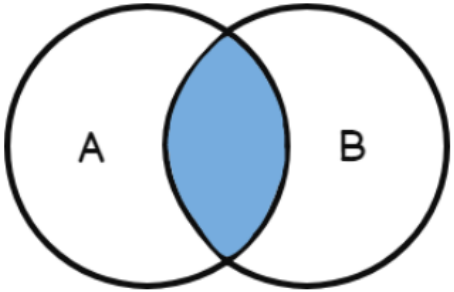
INTERSECT OPERATOR

- Dalam teori himpunan, perpotongan dua himpunan (sebut mereka A dan B) adalah himpunan semua elemen yang termasuk dalam A dan juga dimiliki oleh B.
- Dalam T-SQL, operator set INTERSECT mengembalikan persimpangan set hasil dari dua permintaan input, hanya mengembalikan baris yang muncul di kedua input.

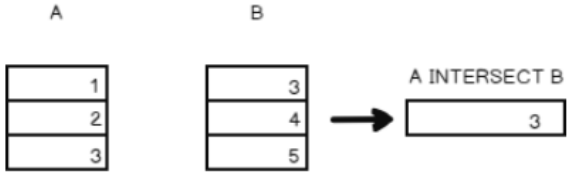


INTERSECT OPERATOR

The interest operator keeps the rows that are common to all the queries



For the same dataset from the aforementioned example, the intersect operator output is given below



Contoh:

```
(
    SELECT 1 ID
    UNION
    SELECT 2
    UNION
    SELECT 3
)

INTERSECT
(
    SELECT 3
    UNION
    SELECT 4
    UNION
    SELECT 5
);
```

Contoh INTERSECT

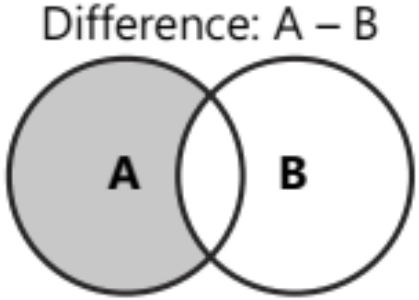
- Operator set INTERSECT secara logis pertama-tama menghilangkan baris duplikat dari dua input multiset yang mengubahnya menjadi set dan kemudian mengembalikan hanya baris yang muncul di kedua set.
- Dengan kata lain, satu baris dikembalikan asalkan muncul setidaknya satu kali di kedua multiset input.

```
SELECT country, region, city FROM HR.Employees
INTERSECT
SELECT country, region, city FROM Sales.Customers;
```

country	region	city
UK	NULL	London
USA	WA	Kirkland
USA	WA	Seattle

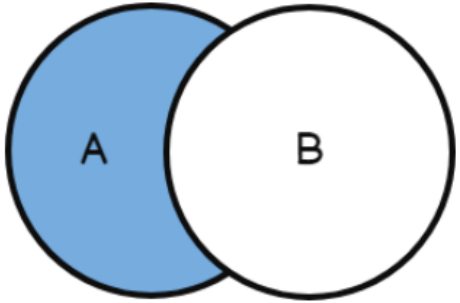
EXCEPT Operator

- Dalam teori himpunan, perbedaan himpunan A dan B (A - B) adalah himpunan elemen yang menjadi milik A dan bukan milik B.
- Himpunan himpunan A - B sebagai A dikurangi dengan anggota B juga pada A.



EXCEPT Operator

The EXCEPT operator lists the rows in the first that are not in the second.



For the same dataset from the aforementioned example, the Except operator output is given below


```
(
    SELECT 1
    UNION
    SELECT 2
    UNION
    SELECT 3
)
EXCEPT
(
    SELECT 3 B
    UNION
    SELECT 4
    UNION
    SELECT 5
);
```

Contoh EXCEPT

- Dalam T-SQL, set perbedaan diterapkan dengan operator set KECUALI. KECUALI beroperasi pada set hasil dari dua kueri input dan mengembalikan baris yang muncul di input pertama tetapi bukan yang kedua.

```
SELECT country, region, city FROM HR.Employees
EXCEPT
SELECT country, region, city FROM Sales.Customers;
```

country	region	city
USA	WA	Redmond
USA	WA	Tacoma

Contoh EXCEPT

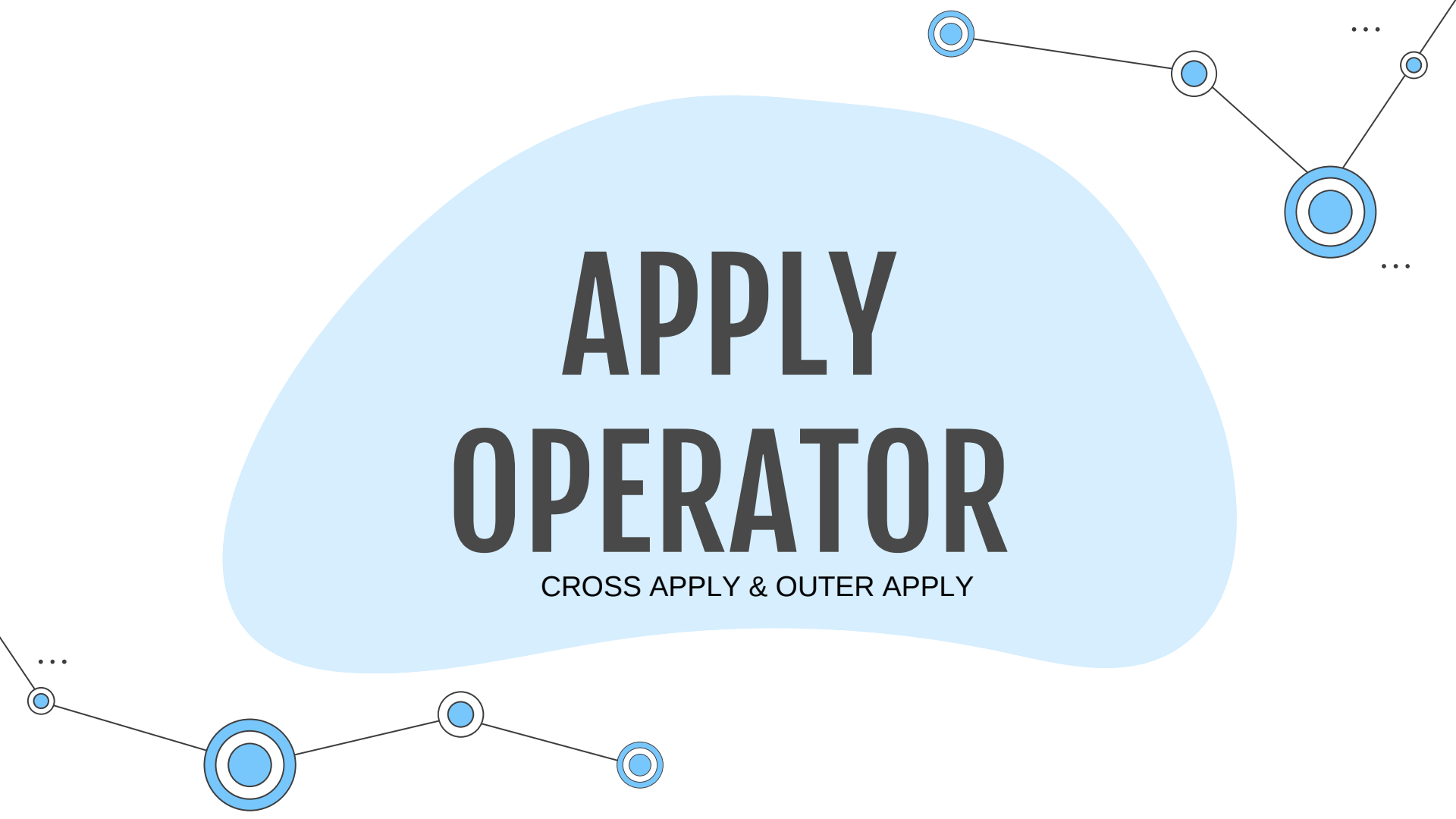
- Kueri berikut mengembalikan lokasi berbeda yang merupakan lokasi pelanggan tetapi bukan lokasi karyawan

```
SELECT country, region, city FROM Sales.Customers
EXCEPT
SELECT country, region, city FROM HR.Employees;
```

```
country         region          city
--------------- --------------- ---------------
Argentina       NULL            Buenos Aires
Austria         NULL            Graz
Austria         NULL            Salzburg
Belgium         NULL            Bruxelles
Belgium         NULL            Charleroi
...
USA             WY              Lander
Venezuela       DF              Caracas
Venezuela       Lara            Barquisimeto
Venezuela       Nueva Esparta   I. de Margarita
Venezuela       Táchira         San Cristóbal

(66 row(s) affected)
```

APPLY OPERATOR

CROSS APPLY & OUTER APPLY



APPLY Operator

- SQL Server mendukung **table valued functions**, fungsi yang mengembalikan data dalam bentuk tabel.
- Operasi **JOIN** di SQL Server digunakan untuk menggabungkan dua atau lebih tabel.
- Namun, operasi JOIN tidak dapat digunakan untuk menggabungkan tabel dengan output dari **table valued functions.**
- Ada dua jenis utama **APPLY** Operator. 1) **CROSS APPLY** dan 2) **OUTER APPLY**.
- Operator **CROSS APPLY** secara semantik mirip dengan operator INNER JOIN.
- Dimana CROSS APPLY akan mengambil catatan-catatan itu dari **table valued function** dan tabel yang berkaitan dengan hanya menampilkan data yang ada pada kedua tabel
- Di sisi lain, OUTER BERLAKU mengambil semua catatan dari **table valued functions** dan tabel.

Contoh

- Dibuat suatu Table valued function

```
SELECT * FROM fnGetBooksByAuthorId(3)
```

Results | Messages

	id	book_name	price	author_id
1	4	Book4	400	3
2	6	Book6	400	3

```
CREATE FUNCTION fnGetBooksByAuthorId(@AuthorId int)
RETURNS TABLE
AS
RETURN
(
SELECT * FROM Book
WHERE author_id = @AuthorId
)
```

Contoh

- Dilakukan INNER JOIN pada Table Valued Function

```
SELECT A.author_name, B.id, B.book_name, B.price
FROM Author A
INNER JOIN fnGetBooksByAuthorId(A.Id) B
ON A.id = B.author_id
```

Messages

```
Msg 4104, Level 16, State 1, Line 76
The multi-part identifier "A.Id" could not be bound.
```

Contoh

- Dilakukan CROSS APPLY dan OUTER APPLY pada Tabel Valued Function

```
SELECT A.author_name, B.id, B.book_name, B.price
FROM Author A
CROSS APPLY fnGetBooksByAuthorId(A.Id) B
```

Results | Messages

	author_name	id	book_name	price
1	Author1	1	Book1	500
2	Author2	2	Book2	300
3	Author1	3	Book3	700
4	Author3	4	Book4	400
5	Author5	5	Book5	650
6	Author3	6	Book6	400

```
SELECT A.author_name, B.id, B.book_name, B.price
FROM Author A
OUTER APPLY fnGetBooksByAuthorId(A.Id) B
```

Results | Messages

	author_name	id	book_name	price
1	Author1	1	Book1	500
2	Author1	3	Book3	700
3	Author2	2	Book2	300
4	Author3	4	Book4	400
5	Author3	6	Book6	400
6	Author4	NULL	NULL	NULL
7	Author5	5	Book5	650
8	Author6	NULL	NULL	NULL
9	Author7	NULL	NULL	NULL

TRIGGER

DML TRIGGER

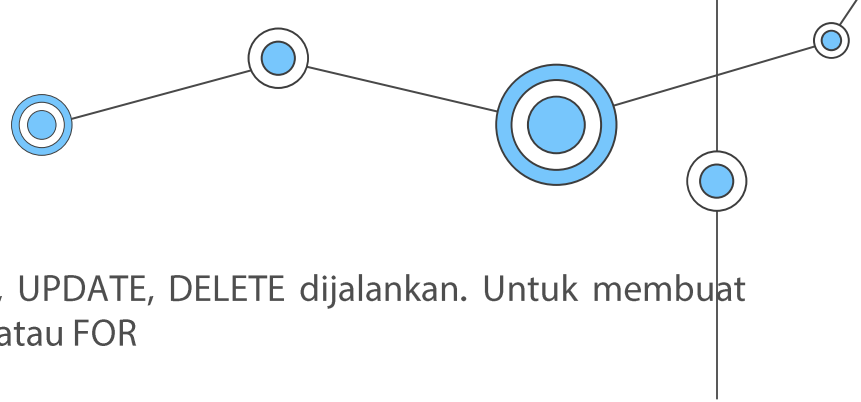
Trigger

- Trigger adalah jenis khusus dari stored procedure yang dijalankan sebagai respons terhadap tindakan tertentu pada tabel seperti penyisipan, penghapusan atau pembaruan data.
- Trigger merupakan objek database yang terikat ke sebuah tabel dan dieksekusi secara otomatis.
- Satu-satunya cara untuk melakukan trigger adalah dengan melakukan tindakan yang diperlukan terhadap tabel di mana trigger ditugaskan.

Trigger

- Jenis Trigger yang ada pada SQL Server dapat dibagi menjadi 2 bagian, yaitu: DML Trigger dan DDL Trigger.
- DML Trigger di SQL Server akan dijalankan ketika peristiwa DML terjadi, yaitu ketika terjadi pembaharuan data, penghapusan data, dan penambahan data.
- DDL Trigger di SQL Server akan dijalankan ketika peristiwa DDL terjadi, yaitu: ketika terjadi proses CREATE, ALTER, DROP dan perintah DDL lainnya.

TRIGGER



1. **AFTER**, trigger akan dijalankan setelah proses INSERT, UPDATE, DELETE dijalankan. Untuk membuat after trigger kita dapat menggunakan statement AFTER atau FOR

2. **INSTEAD OF**, trigger akan dijalankan sebelum proses INSERT, UPDATE, DELETE dijalankan. Pada SQL Server versi sebelumnya fasilitas Instead Of ini tidak disediakan.

Contoh DML Trigger UPDATE

- Dicontohkan terdapat 2 table dengan perintah berikut:

```
CREATE TABLE Locations (LocationID int, LocName varchar(100))

CREATE TABLE LocationHist (LocationID int, ModifiedDate DATETIME)
```

- Contoh perintah Trigger pada UPDATE:

```
CREATE TRIGGER TR_UPD_Locations ON Locations
FOR UPDATE
NOT FOR REPLICATION
AS

BEGIN
  INSERT INTO LocationHist
  SELECT LocationID
    ,getdate()
  FROM inserted
END
```

Contoh DML Trigger UPDATE

- **TR_UPD_Locations** = nama Trigger
- **Locations** = Nama Tabel
- **FOR UPDATE** = DML Event
- Perintah di antara **BEGIN … END** adalah perintah yang secara spesifik dilakukan ketika trigger dijalankan.

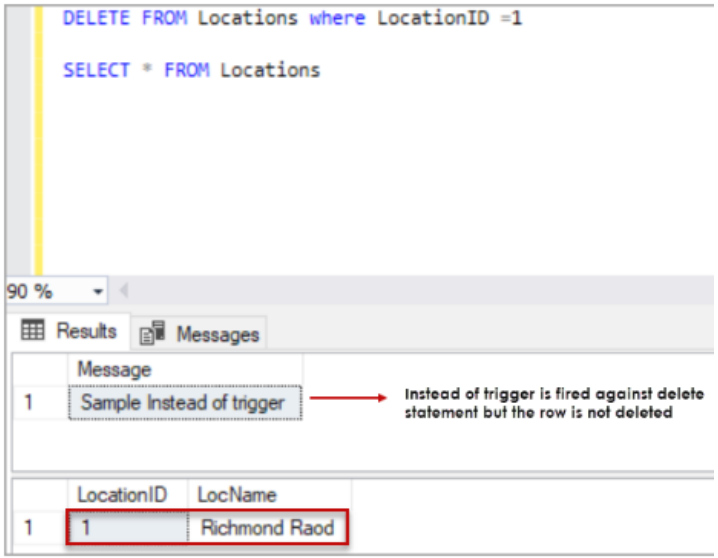
INSTEAD OF Trigger

- Trigger ini diaktifkan sebelum peristiwa DML dan data aktual tidak dimodifikasi dalam tabel.
- Misalnya, jika kita menentukan **INSTEAD OF** Trigger untuk DELETE pada tabel, saat pernyataan DELETE dijalankan terhadap tabel, INSTEAD OF Trigger diaktifkan dan blok T-SQL di dalam Trigger di SQL Server dijalankan tetapi penghapusan yang sebenarnya tidak terjadi.

Contoh

```
CREATE TRIGGER TR_DEL_Locations ON Locations
INSTEAD OF DELETE
AS
BEGIN
  Select 'Sample Instead of trigger' as [Message]
END
```

- Ketika Trigger telah dibuat untuk perintah DELETE, maka ketika perintah DELETE dilakukan, akan muncul pesan “Sample …”dan apabila dilakukan SELECT, maka data masih tetap ada



Aktif / Non Aktif Trigger

- Untuk menonaktifkan Trigger:

```
DISABLE TRIGGER TR_UPD_Locations2 on Locations
```

- Untuk mengaktifkan Trigger:

```
ENABLE TRIGGER TR_UPD_Locations2 on Locations
```

- Menonaktifkan SEMUA Trigger:

```
DISABLE TRIGGER ALL ON Locations
```

Menghapus Trigger

- Untuk menghapus Trigger yang telah dibuat, dapat dilakukan dengan cara:

```
DROP TRIGGER TRL_UPD_Locations2
```

Thanks!

Do you have any questions?

Team Teaching Matakuliah Basis Data Lanjut
JTI POLINEMA