



# BASIS DATA LANJUT

## **PERTEMUAN 2**

### **PENGANTAR TRANSACT SQL DAN REVIEW QUERY**

TIM BASIS DATA LANJUT JTI 2023



# Outline

- Konsep T-SQL
- Komentar
- Perintah SELECT, DISTINCT
- Penggunaan ALIAS
- Konsep Ekspresi CASE
- Sorting
- Filtering
- Grouping
- Aggregating
- Operasi himpunan

# APA ITU T-SQL?

## SQL

- Structure Query Language; bahasa yang digunakan untuk mengakses dan melakukan manipulasi suatu basis data.

## Transact-SQL (disingkat T-SQL)

- Bahasa basis data SQL yang dikeluarkan oleh perusahaan Microsoft dan Sybase. Microsoft SQL Server hanya mengenali bahasa SQL tipe T-SQL ini.





# JENIS T-SQL

- DDL (Data Definition Language)

Digunakan untuk mendefinisikan struktur basis data, basis data dan tabel.

perintah dasar: **CREATE, ALTER, DROP**

- DML (Data Manipulation Language)

Digunakan untuk melakukan manipulasi atau pengolahan data di dalam tabel.

Perintah dasar: **SELECT, INSERT, UPDATE, DELETE**



# JENIS T-SQL

- DCL (Data Control Language)

Digunakan untuk pengaturan hak akses user baik terhadap server, database maupun tabel.

Perintah dasar: **GRANT, REVOKE**

- TCL (Transaction Control Language)

Digunakan untuk mengatur dan mengontrol transaksi T-SQL. Menjamin transaksi berhasil dilakukan dan tidak melanggar integritas basis data.

Perintah dasar: **BEGIN TRAN, COMMIT TRAN, ROLLBACK**

# KOMENTAR PADA T-SQL

Komentar pada T-SQL digunakan:

- tanda "--" untuk komentar satu baris
- tanda /\* ... \*/ untuk komentar lebih dari satu baris

Contohnya:

```
-- Komentar satu baris
```

```
/* tanda awal komentar multi baris  
   komentar - komentar  
   tanda akhir komentar multi baris */
```



# T-SQL SELECT

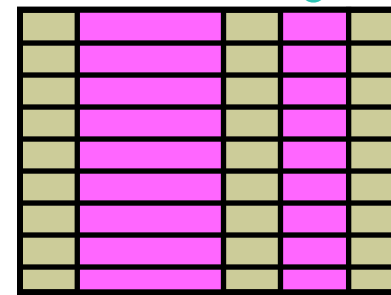
# SELECT

Digunakan untuk menampilkan data dari tabel suatu basis data

Untuk menampilkan data dibutuhkan:

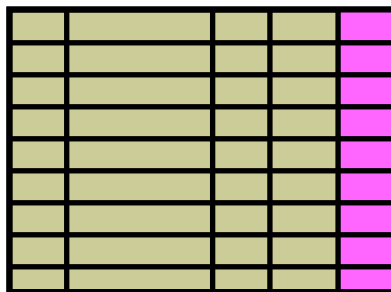
- ✓ Apa saja kolom yang ingin ditampilkan
- ✓ Nama tabel yang akan ditampilkan
- ✓ Kondisi untuk menampilkan data

**Projection**



A 10x5 grid representing a table. The second and fourth columns are highlighted in pink, illustrating the selection of specific columns (projection) from the original table.

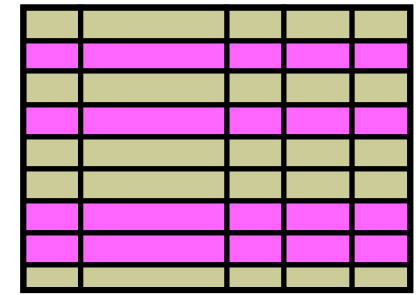
**Table 1**



A 10x5 grid representing a table. The second and fourth columns are highlighted in pink, illustrating the selection of specific columns (projection) from the original table.

**Table 1**

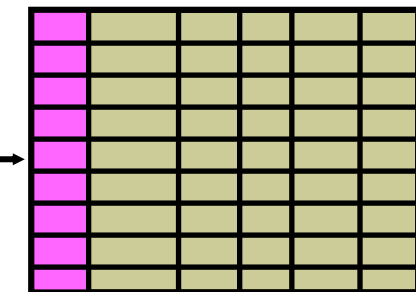
**Selection**



A 10x5 grid representing a table. The first, third, fifth, seventh, and ninth rows are highlighted in pink, illustrating the selection of specific rows (selection) from the original table.

**Table 1**

**Join**



A 10x5 grid representing a table. The first, third, fifth, seventh, and ninth rows are highlighted in pink, illustrating the selection of specific rows (selection) from the original table.

**Table 2**



# SELECT SEMUA DATA PADA TABEL

- Syntax:  
SELECT \* FROM [nama\_tabel];

```
SELECT *  
FROM departments;
```

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting		1700

8 rows selected.

# SELECT DATA TERTENTU PADA TABEL

Syntax:

```
SELECT [nama_kolom], [nama_kolom]  
FROM [nama_tabel]
```

```
SELECT department_id, location_id  
FROM departments;
```

DEPARTMENT_ID	LOCATION_ID
10	1700
20	1800
50	1500
60	1400
80	2500
90	1700
110	1700
190	1700

8 rows selected.



## SELECT BERDASARKAN KONDISI TERTENTU

Perintah WHERE dapat digunakan untuk membatasi baris yang akan ditampilkan

Syntax:

```
SELECT [nama_kolom], [nama_kolom]  
FROM [nama_tabel]  
WHERE [kondisi];
```

```
SELECT last_name, salary  
FROM employees  
WHERE salary <= 3000 ;
```

LAST_NAME	SALARY
Matos	2600
Vargas	2500

# MENGGUNAKAN OPERATOR ARITMATIK

```
SELECT last_name, salary, salary + 300  
FROM employees;
```

LAST_NAME	SALARY	SALARY+300
King	24000	24300
Kochhar	17000	17300
De Haan	17000	17300
Hunold	9000	9300
Ernst	6000	6300

...  
20 rows selected.

# DISTINCT

Query SELECT menampilkan keseluruhan baris, termasuk baris yang terduplikasi  
Perintah DISTINCT digunakan untuk mengeliminasi baris yang terduplikasi

Syntax:

```
SELECT DISTINCT [nama_kolom]  
FROM [nama_tabel];
```

```
SELECT department_id  
FROM employees;
```

1

DEPARTMENT_ID
90
90
90
...

20 rows selected.

```
SELECT DISTINCT  
department_id  
FROM employees;
```

DEPARTMENT_ID
10
20
50

8 rows selected.

2

# ALIAS

Digunakan untuk mengganti nama tabel atau kolom  
Agar lebih singkat dan menyederhanakan penulisan nama tabel atau kolom

Syntax:

```
SELECT [atribut_kolom] AS [alias]  
FROM [nama_tabel];
```

```
SELECT last_name AS name,  
commission_pct comm  
FROM employees;
```

NAME	COMM
King	
Kochhar	
De Haan	

...

20 rows selected.



# T-SQL CASE

# CASE EXPRESSION

Kegunaannya mirip seperti IF-THEN-ELSE

Syntax:

```
SELECT [nama_kolom], [nama_kolom],  
CASE [expr] WHEN [comparison_exp1] THEN [return_expr1]  
    WHEN [comparison_exp2] THEN [return_expr2]  
    WHEN [comparison_exp3] THEN [return_expr3]  
ELSE [else_expr]  
END AS [nama_alias]  
FROM [nama_tabel];
```





# CASE EXPRESSION (1)

```
SELECT last_name, job_id, salary,  
       CASE job_id WHEN 'IT_PROG' THEN 1.10*salary  
                   WHEN 'ST_CLERK' THEN 1.15*salary  
                   WHEN 'SA_REP' THEN 1.20*salary  
       ELSE salary  
       END AS REVISED_SALARY  
FROM   employees;
```

LAST_NAME	JOB_ID	SALARY	REVISED_SALARY
...			
Lorentz	IT_PROG	4200	4620
Mourgos	ST_MAN	5800	5800
Rajs	ST_CLERK	3500	4025
...			
Gietz	AC_ACCOUNT	8300	8300

20 rows selected.



# T-SQL SORTING

# sorting

Suatu proses menyusun kembali data dengan suatu pola tertentu, sehingga tersusun secara teratur menurut aturan tertentu.

Pada SQL, digunakan fungsi ORDER BY untuk menampilkan data secara teratur.

Terdapat 2 jenis pengurutan:

**ASCENDING** (urut naik), pengurutan data dari nilai yang lebih kecil menuju ke nilai yang lebih besar. (A ke Z atau 1 ke 99)

**DESCENDING** (urut turun), pengurutan data dari nilai yang lebih besar menuju ke nilai yang lebih kecil. (Z ke A atau 99 ke 1)

Syntax

```
SELECT column1, column2, ...  
FROM table_name  
ORDER BY column1, column2,  
... ASC|DESC;
```

Jika hanya menggunakan ORDER BY, tanpa ASC / DESC, maka secara default, data akan terurut naik

# sorting

Contoh:

```
SELECT contactname, address, city, phone  
FROM Sales.Customers  
ORDER BY contactname desc;
```

contactname	address	city	phone
Young, Robin	0123 Grizzly Peak Rd.	Butte	(406) 555-0121
Wojciechowska, Agnieszka	P.O. Box 1234	Lander	(307) 555-0114
Wickham, Jim	Luisenstr. 0123	Münster	0251-456789
Welcker, Brian	4567 Wadhurst Rd.	London	(171) 901-2345
Watters, Jason M.	Gran Vía, 4567	Madrid	(91) 567 8901
Voss, Florian	Strada Provinciale 7890	Reggio Emilia	0522-012345
Veronesi, Giorgio	Taucherstraße 1234	Cunewalde	0372-12345
Veninga, Tjeerd	1234 DaVinci Blvd.	Kirkland	(206) 555-0124
Uppal, Sunil	Estrada da saúde n. 6789	Lisboa	(1) 789-0123
Tuntisangaroorn, Sittichai	6789, rue du Commerce	Lyon	78.90.12.34
Tollefsen, Bjørn	5678, boulevard Charonne	Paris	(1) 89.01.23.45



# T-SQL FILTERING

# TOP N

Digunakan untuk menampilkan record yang berisi sejumlah tertentu yang merupakan urutan teratas atau urutan terbawah dari sekumpulan record.

Syntax

```
SELECT TOP number | column_name (number)  
    FROM table_name  
    WHERE condition;
```

# TOP N

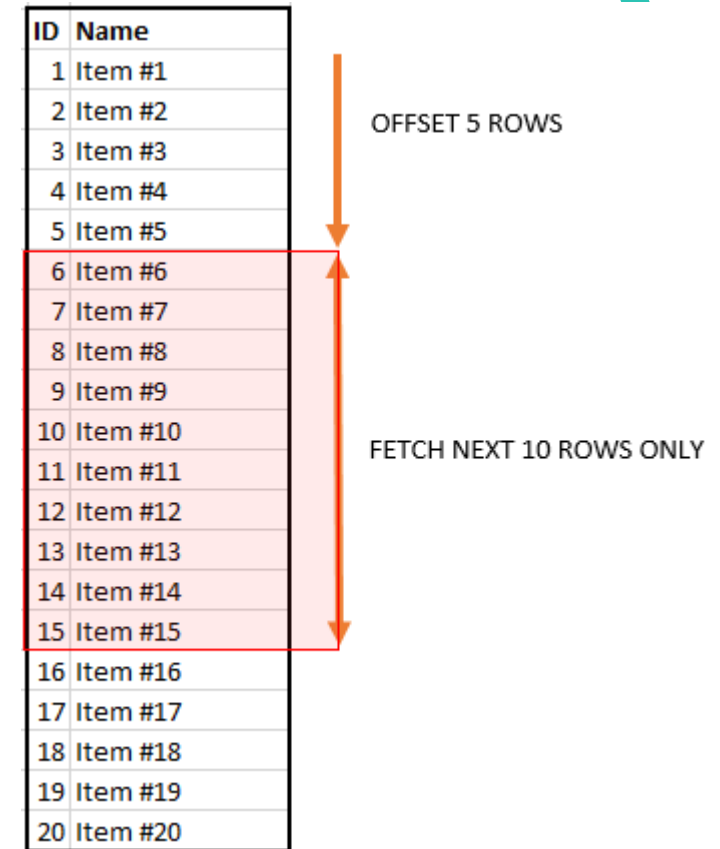
Contoh

```
SELECT TOP 5  
  contactname, address, city, phone  
FROM Sales.Customers;
```

	contactname	address	city	phone
1	Allen, Michael	Obere Str. 0123	Berlin	030-3456789
2	Hassall, Mark	Avda. de la Constitución 5678	México D.F.	(5) 789-0123
3	Peoples, John	Mataderos 7890	México D.F.	(5) 123-4567
4	Amdt, Torsten	7890 Hanover Sq.	London	(171) 456-7890
5	Higginbotham, Tom	Berguvsvägen 5678	Luleå	0921-67 89 01

# OFFSET-FETCH

*OFFSET-FETCH* digunakan bersama dengan klausa `SELECT` dan `ORDER BY` untuk mengambil serangkaian record (range).



ID	Name
1	Item #1
2	Item #2
3	Item #3
4	Item #4
5	Item #5
6	Item #6
7	Item #7
8	Item #8
9	Item #9
10	Item #10
11	Item #11
12	Item #12
13	Item #13
14	Item #14
15	Item #15
16	Item #16
17	Item #17
18	Item #18
19	Item #19
20	Item #20

OFFSET 5 ROWS

FETCH NEXT 10 ROWS ONLY





# OFFSET

*OFFSET* digunakan untuk menentukan jumlah baris untuk dilewati sebelum mulai mengembalikan baris dari query.

Offset hanya dapat digunakan dengan klausa **ORDER BY**

Nilai **OFFSET** harus lebih besar dari atau sama dengan nol

Syntax

```
SELECT column_name(s)
FROM table_name
WHERE condition
ORDER BY column_name
OFFSET rows_to_skip ROWS;
```

## Contoh:

```
SELECT product_name, list_price
FROM production.products
ORDER BY list_price, product_name
OFFSET 10 ROWS;
```

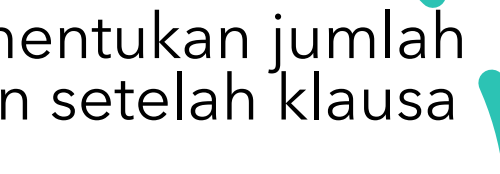
product_name	list_price
Strider Classic 12 Balance Bike - 2018	89.99
Sun Bicycles Lil Kitten - 2017	109.99
Trek Boy's Kickster - 2015/2017	149.99
Trek Girl's Kickster - 2017	149.99
Trek Kickster - 2018	159.99
Trek Precaliber 12 Boys - 2017	189.99
Trek Precaliber 12 Girls - 2017	189.99
Trek Precaliber 12 Boy's - 2018	199.99
Trek Precaliber 12 Girl's - 2018	199.99
Haro Shredder 20 - 2017	209.99
Haro Shredder 20 Girls - 2017	209.99
Trek Precaliber 16 Boy's - 2018	209.99
Trek Precaliber 16 Boys - 2017	209.99
Trek Precaliber 16 Girl's - 2018	209.99
Trek Precaliber 16 Girls - 2017	209.99
Trek Precaliber 20 Boy's - 2018	229.99
Trek Precaliber 20 Girl's - 2018	229.99
Haro Shredder Pro 20 - 2017	249.99



product_name	list_price
Haro Shredder 20 Girls - 2017	209.99
Trek Precaliber 16 Boy's - 2018	209.99
Trek Precaliber 16 Boys - 2017	209.99
Trek Precaliber 16 Girl's - 2018	209.99
Trek Precaliber 16 Girls - 2017	209.99
Trek Precaliber 20 Boy's - 2018	229.99
Trek Precaliber 20 Girl's - 2018	229.99
Haro Shredder Pro 20 - 2017	249.99
Strider Sport 16 - 2018	249.99
Trek MT 201 - 2018	249.99
Sun Bicycles Revolutions 24 - 2017	250.99
Sun Bicycles Revolutions 24 - Girl's - 2017	250.99
Electra Cruiser 1 (24-Inch) - 2016	269.99
Electra Cruiser 1 (24-Inch) - 2016	269.99
Electra Cruiser 1 - 2016/2017/2018	269.99



# FETCH



*FETCH* digunakan untuk menentukan jumlah baris yang akan dikembalikan setelah klausa OFFSET diproses.

Klausa OFFSET bersifat mandatory sedangkan klausa FETCH bersifat opsional

Syntax

```
SELECT column_name(s)
FROM table_name
WHERE condition
ORDER BY column_name
OFFSET rows_to_skip ROWS
FETCH NEXT number_of_rows ROWS
ONLY;
```



# FETCH



Contoh, melewati 10 produk pertama dan menampilkan 10 produk berikutnya:

```
SELECT product_name, list_price  
FROM production.products  
ORDER BY list_price, product_name  
OFFSET 10 ROWS  
FETCH NEXT 10 ROWS ONLY;
```



# T-SQL GROUP BY DAN HAVING

# 'Group by' dan prioritas Logis operasi

- HAVING, SELECT, dan ORDER BY harus mengembalikan sebuah nilai tunggal per group
- Semua kolom pada SELECT, HAVING, dan ORDER BY harus berada dalam klausa GROUP BY atau menjadi input dari fungsi/ekspresi agregasi

Logical Order	Phase	Comments
5	SELECT	
1	FROM	
2	WHERE	
3	GROUP BY	Creates groups
4	HAVING	Operates on groups
6	ORDER BY	

# Group By

SQL berikut akan mengembalikan error:

```
SELECT empid, custid, COUNT(*) AS cnt  
FROM Sales.Order  
GROUP BY empid;
```

Errornya:

Msg 8120, Level 16, State 1, Line 3

Column 'Sales.Orders.**custid**' is invalid in the select list because it is *not contained in either an aggregate function or the GROUP BY* clause.

Mengapa?

Karena kolom **custid** tidak ada di klausa GROUP BY

# Menggunakan 'group by' bersama fungsi agregasi

- Fungsi agregasi umum digunakan di statement SELECT, untuk menghitung berdasarkan kelompok:

```
SELECT CustomerID, COUNT(*) AS cnt  
FROM Sales.SalesOrderHeader  
GROUP BY CustomerID;
```

```
SELECT productid, MAX(OrderQty) AS largest_order  
FROM Sales.SalesOrderDetail  
GROUP BY productid;
```

- Fungsi agregasi boleh meng-agregasi kolom apa saja. Tidak hanya yang di dalam GROUP BY saja.



# Menyaring kelompok data dengan klausa 'Having'

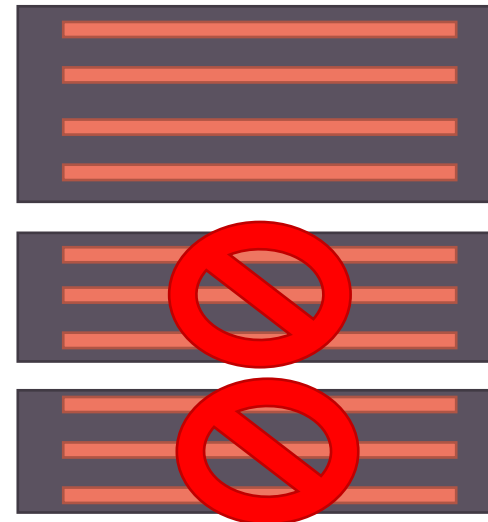
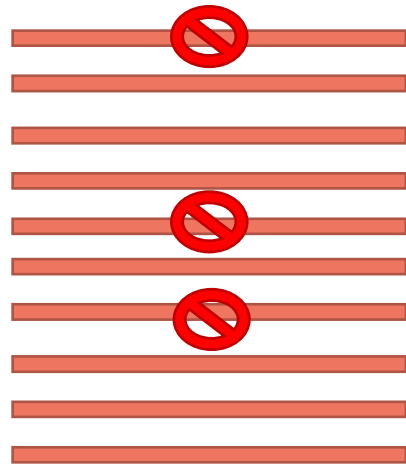
- Klausa HAVING membantu kita untuk memfilter grup-grup yang didapat dari GROUP BY berdasarkan kondisi tertentu.
- Klausa HAVING diproses setelah GROUP BY
- Misal: Mencari pelanggan yang sudah pernah belanja lebih dari 10x...

NoStruk	IdPelanggan	TotalBelanja
1192	Ani	90.000
1193	Budi	100.000
1194	Ani	25.000
Dst...	Dst...	Dst...

```
SELECT IdPelanggan, COUNT(*) AS  
Jumlah_Transaksi  
FROM Penjualan  
GROUP BY IdPelanggan  
HAVING COUNT(*) > 10;
```

# Klausa 'Having' vs. 'WHERE'

- WHERE memfilter baris-baris sebelum grup dibuat
  - Memilih baris mana yang ditampilkan/dimasukkan grup/dikembalikan
- HAVING filters groups
  - Memilih grup mana saja yang ditampilkan/dikembalikan



# Klausula 'Having' vs. 'WHERE'

- Menggunakan fungsi/ekspresi COUNT(\*) pada klausa HAVING sangat berguna untuk menyelesaikan permasalahan bisnis umum:
- **Misal:** Tampilkan customer yang sudah pesan lebih dari sekali:

```
SELECT Cust.Customerid, COUNT(*) AS cnt
FROM Sales.Customer AS Cust
JOIN Sales.SalesOrderHeader AS Ord ON Cust.CustomerID =
ORD.CustomerID
GROUP BY Cust.CustomerID
HAVING COUNT(*) > 1;
```

- **Misal:** Tampilkan produk yang sudah lebih dari 10x dipesan

```
SELECT Prod.ProductID, COUNT(*) AS cnt
FROM Production.Product AS Prod
JOIN Sales.SalesOrderDetail AS Ord ON Prod.ProductID = Ord.ProductID
GROUP BY Prod.ProductID
HAVING COUNT(*) >= 10;
```

# Contoh HAVING

```
SELECT
    column_name1,
    column_name2,
    aggregate_function (column_name3) column_alias
FROM
    table_name
GROUP BY
    column_name1,
    column_name2
HAVING
    column_alias > value;
```

Instead, you must use the aggregate function expression in the `HAVING` clause explicitly as follows:

```
SELECT
    column_name1,
    column_name2,
    aggregate_function (column_name3) alias
FROM
    table_name
GROUP BY
    column_name1,
    column_name2
HAVING
    aggregate_function (column_name3) > value;
```




SQL Server HAVING  
Clause  
([sqlservertutorial.net](http://sqlservertutorial.net))






# 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  $\cap$ ,  $\cup$ , dan  $-$ .
- Bentuk umum dari operasi himpunan, adalah:



Input Query1  
<set\_operator>  
Input Query2  
[ORDER BY ...]

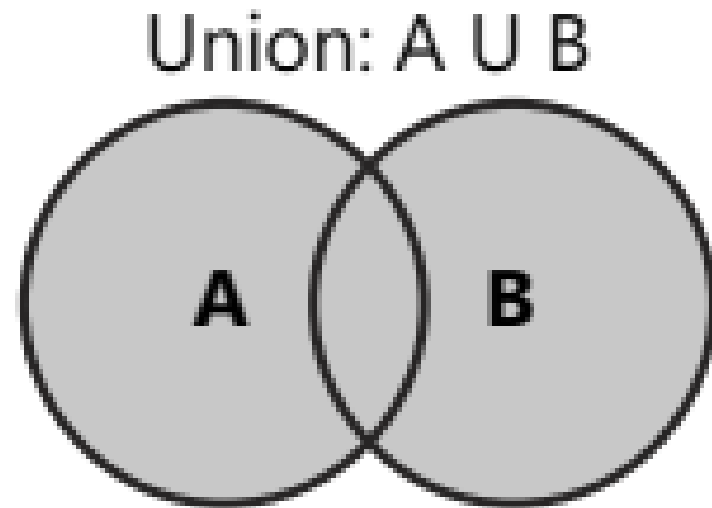


# UNION Operator

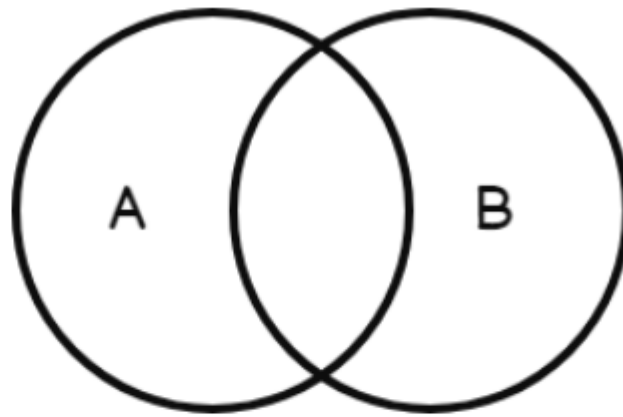
Dalam teori himpunan, penyatuan dua himpunan adalah himpunan yang berisi semua elemen.

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.

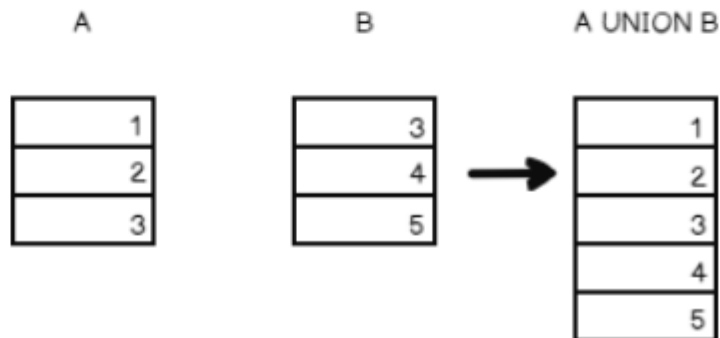
T-SQL mendukung UNION ALL dan UNION (implisit DISTINCT) dari operator UNION



# 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
```

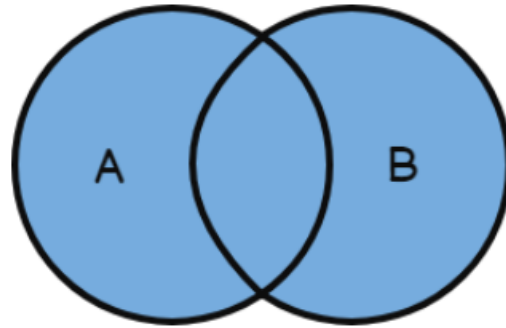
```
SELECT country, region, city FROM Sales.Customers;
```

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

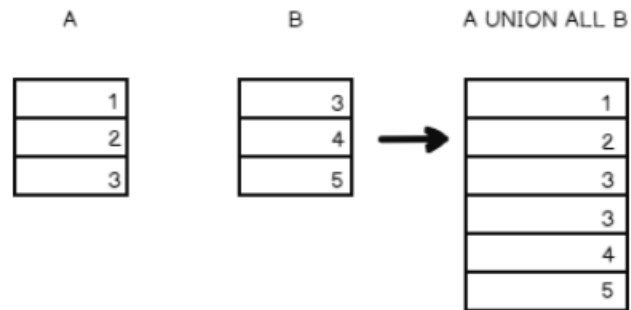
country	region	city
-----	-----	-----
USA	WA	Seattle
USA	WA	Tacoma
USA	WA	Kirkland
USA	WA	Redmond
UK	NULL	London
UK	NULL	London
UK	NULL	London
...		
Finland	NULL	Oulu
Brazil	SP	Resende
USA	WA	Seattle
Finland	NULL	Helsinki
Poland	NULL	Warszawa

(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)

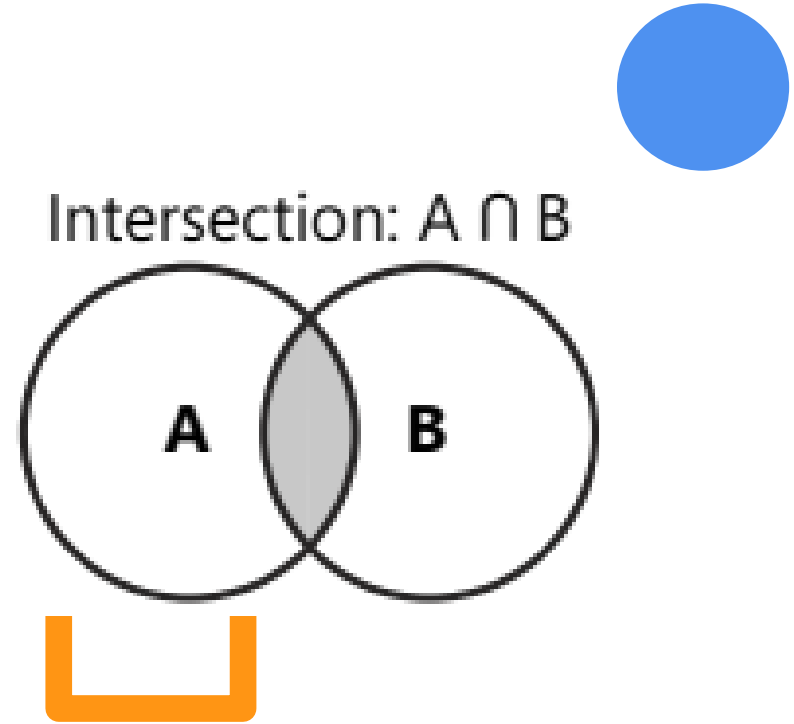
# SYNTAX

```
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.

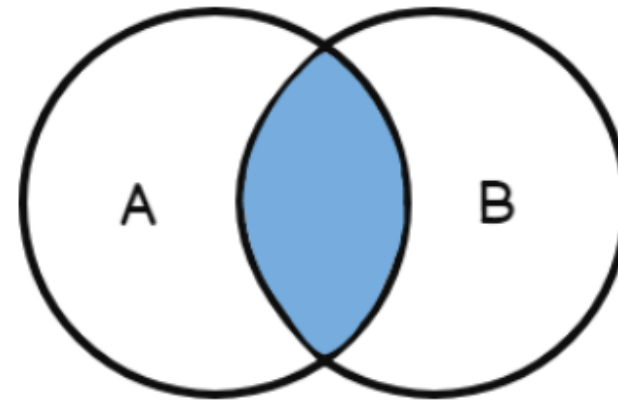


# INTERSECT OPERATOR

Contoh:

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

The intersect 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 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;
```

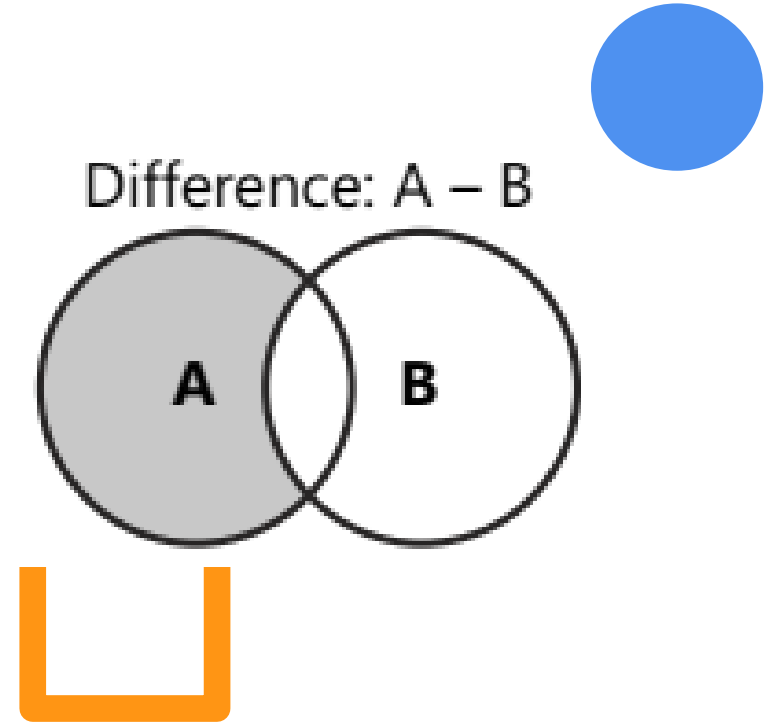
country	region	city
UK	NULL	London
USA	WA	Kirkland
USA	WA	Seattle



# EXCEPT Operator

Dalam teori himpunan, perbedaan himpunan A dan B ( $A - B$ ) adalah himpunan elemen yang menjadi milik A dan bukan milik B.

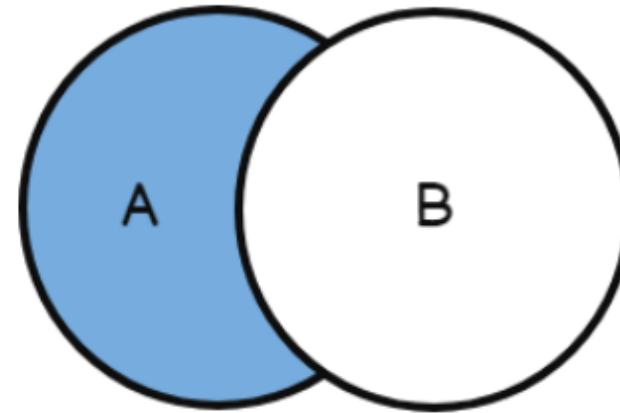
Himpunan himpunan  $A - B$  sebagai A dikurangi dengan anggota B juga pada A.



# EXCEPT Operator

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

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



# 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)		

# Thanks!

Do you have any questions?



Team Teaching Matakuliah Basis Data Lanjut  
JTI POLINEMA

[jti.polinema.ac.id](http://jti.polinema.ac.id)