# PostgreSQL – Joining Multiple Tables

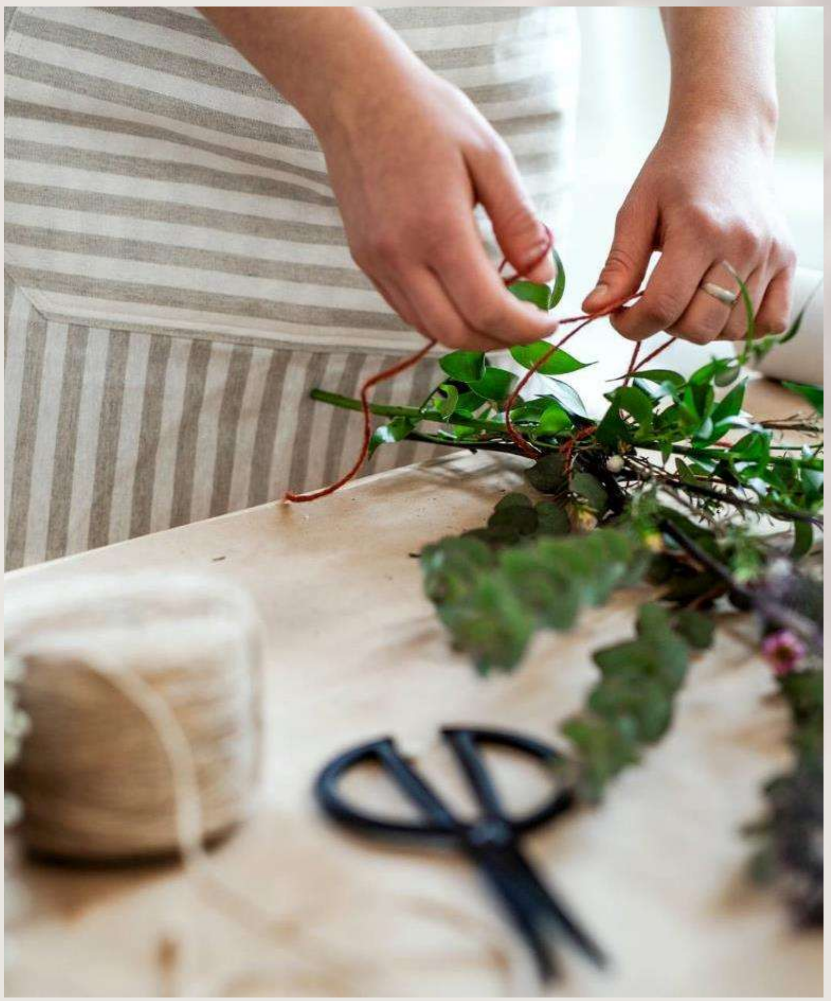
Tessy Badriyah, PhD.

# Topik

- Joins
- Table alias
- Inner Join
- Left Join
- Self-join
- Full Outer Join
- Cross Join
- Natural Join

# JOIN

- Join digunakan untuk menampilkan data dari dua tabel atau lebih.
- Macam2 JOIN:
  - Inner Join
  - Left Join
  - Self-join
  - Full Outer Join
  - Cross Join
  - Natural Join



# Tabel alias

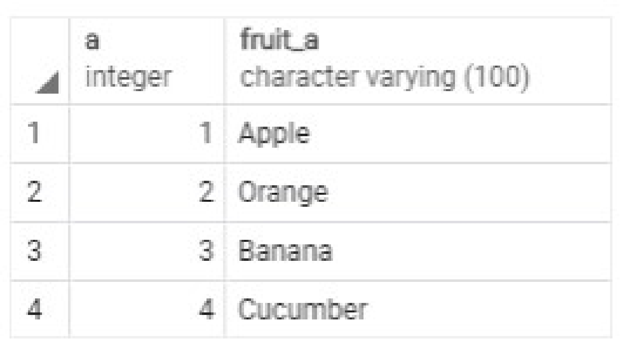
Memberi nama alias pada tabel untuk mempersingkat atau memudahkan penyebutan

**CONTOH:**

```
SELECT F.film_id, F.title
FROM FILM F;
```

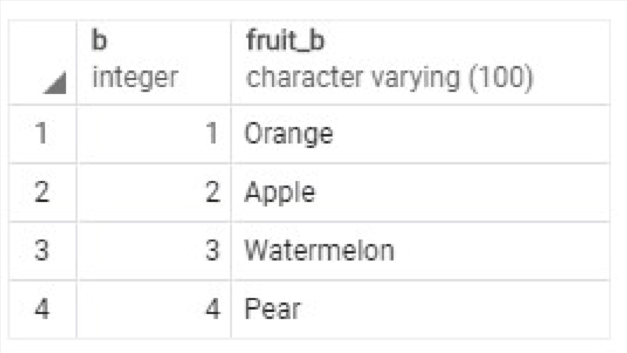
# Tabel yang digunakan untuk contoh

## Tabel basket_a



| | a<br>integer | fruit_a<br>character varying (100) |
|---|---|---|
| 1 | 1 | Apple |
| 2 | 2 | Orange |
| 3 | 3 | Banana |
| 4 | 4 | Cucumber |

## Tabel basket_b

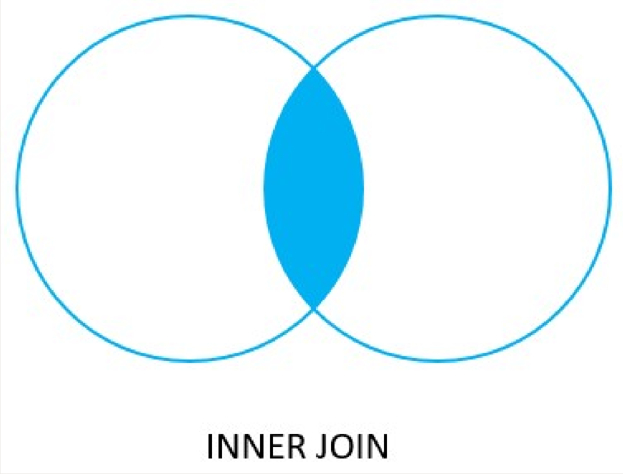


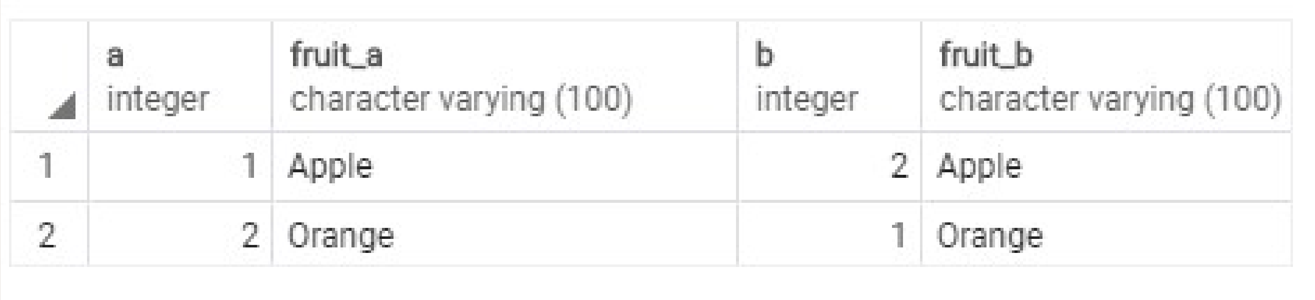
| | b<br>integer | fruit_b<br>character varying (100) |
|---|---|---|
| 1 | 1 | Orange |
| 2 | 2 | Apple |
| 3 | 3 | Watermelon |
| 4 | 4 | Pear |

# Inner join

Inner join dari tabel basket_a dan tabel basket_b berarti mengkombinasikan data dari tabel basket_a yang matching dengan data dari tabel basket_b

```
SELECT a, fruit_a, b, fruit_b

FROM basket_a

INNER JOIN basket_b

ON fruit_a=fruit_b;
```



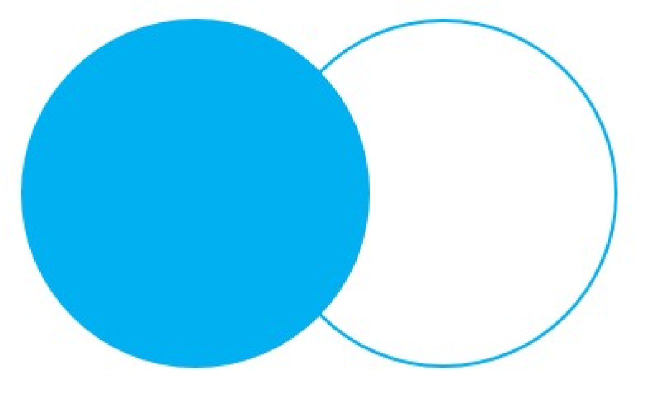


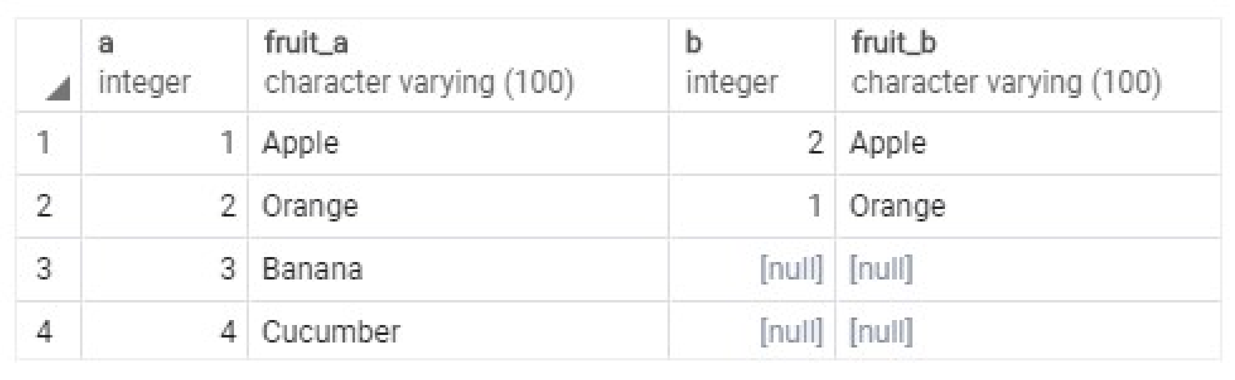
| | a<br>integer | fruit_a<br>character varying (100) | b<br>integer | fruit_b<br>character varying (100) |
|---|---|---|---|---|
| 1 | 1 | Apple | 2 | Apple |
| 2 | 2 | Orange | 1 | Orange |

# Left/Right JOIN

Left atau Right JOIN mengambil bagian semua yang ada di salah satu tabel yang tidak dimiliki oleh tabel lainnya

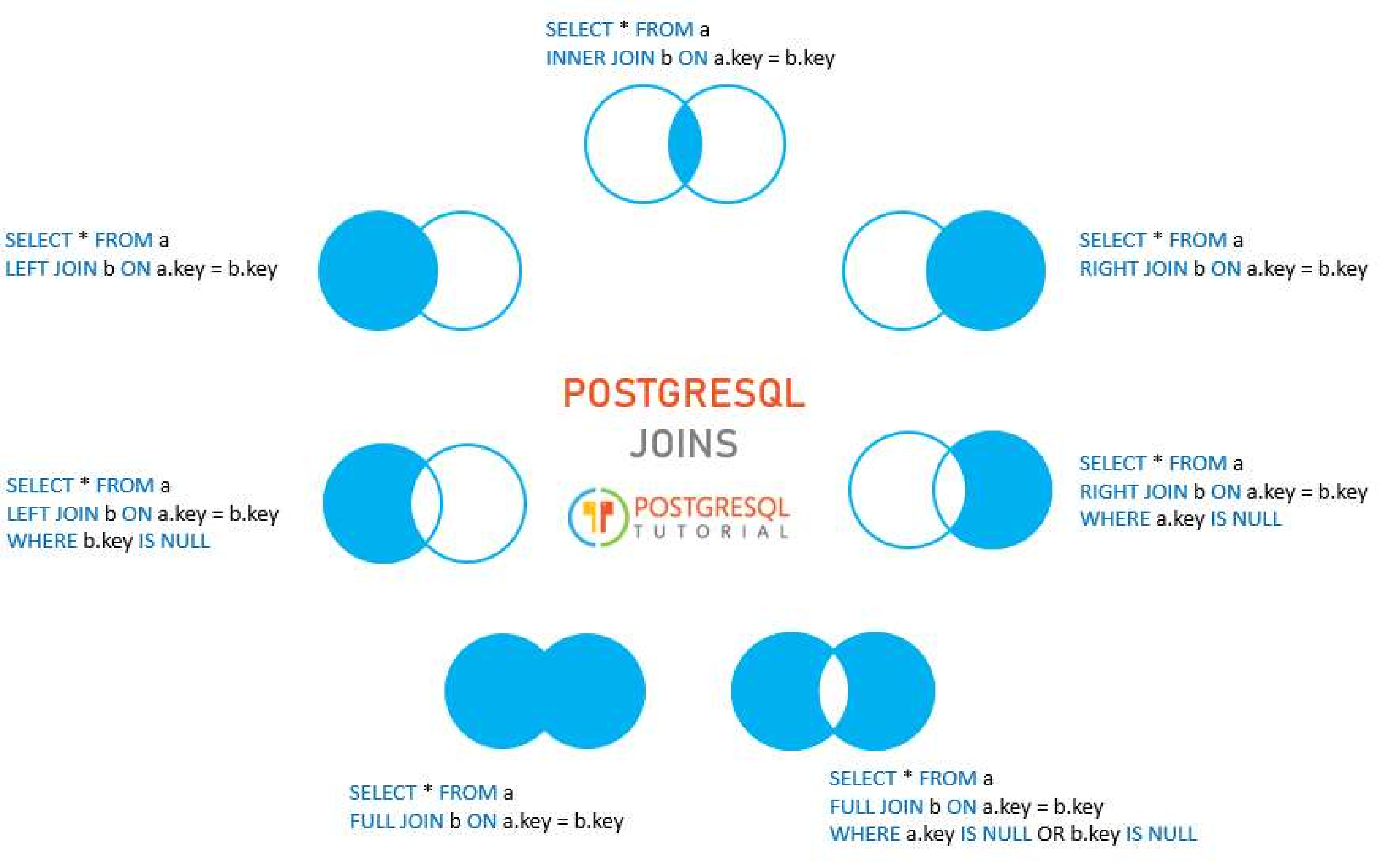
```
SELECT a, fruit_a, b, fruit_b
FROM basket_a
LEFT JOIN basket_b
ON fruit_a=fruit_b;
```





|  | a<br>integer | fruit_a<br>character varying (100) | b<br>integer | fruit_b<br>character varying (100) |
|---|---|---|---|---|
| 1 | 1 | Apple | 2 | Apple |
| 2 | 2 | Orange | 1 | Orange |
| 3 | 3 | Banana | [null] | [null] |
| 4 | 4 | Cucumber | [null] | [null] |

# LEFT, RIGHT dan FULL



```
SELECT * FROM a
INNER JOIN b ON a.key = b.key
```

```
SELECT * FROM a
LEFT JOIN b ON a.key = b.key
```

```
SELECT * FROM a
RIGHT JOIN b ON a.key = b.key
```

```
SELECT * FROM a
LEFT JOIN b ON a.key = b.key
WHERE b.key IS NULL
```

```
SELECT * FROM a
RIGHT JOIN b ON a.key = b.key
WHERE a.key IS NULL
```

```
SELECT * FROM a
FULL JOIN b ON a.key = b.key
```

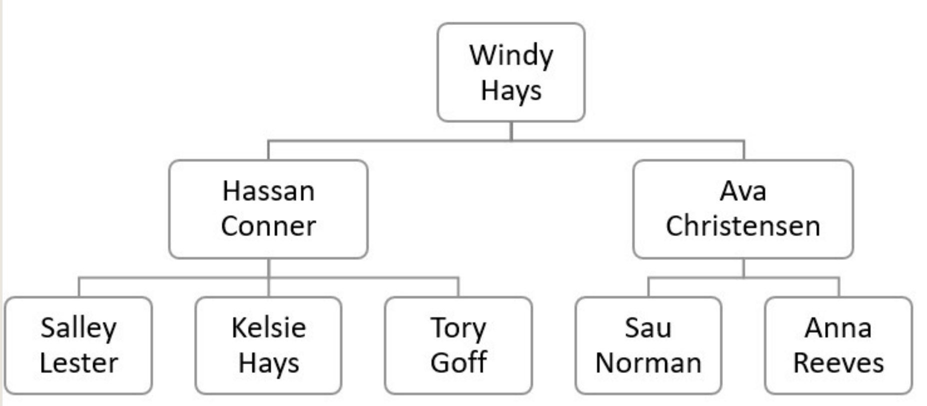
```
SELECT * FROM a
FULL JOIN b ON a.key = b.key
WHERE a.key IS NULL OR b.key IS NULL
```

# Self  Join

## Diberikan hirarki dalam suatu Perusahaan sbb:

### Perintah pembuatan tabel:

```
CREATE TABLE employee ( employee_id INT PRIMARY KEY,
first_name VARCHAR (255) NOT NULL, last_name VARCHAR
(255) NOT NULL, manager_id INT, FOREIGN KEY
(manager_id) REFERENCES employee (employee_id) ON
DELETE CASCADE ); INSERT INTO employee ( employee_id,
first_name, last_name, manager_id )

VALUES (1, 'Windy', 'Hays', NULL),

(2, 'Ava', 'Christensen', 1),

(3, 'Hassan', 'Conner', 1),

(4, 'Anna', 'Reeves', 2),

(5, 'Sau', 'Norman', 2),

(6, 'Kelsie', 'Hays', 3),

(7, 'Tory', 'Goff', 3),

(8, 'Salley', 'Lester', 3);
```



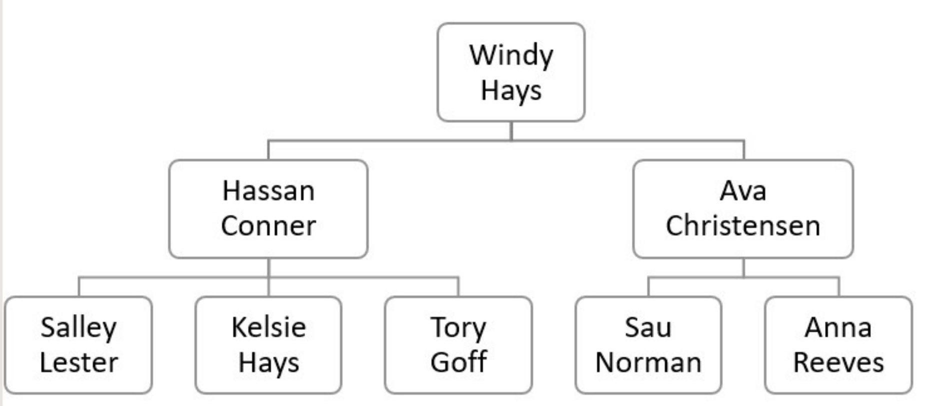
# Self Join

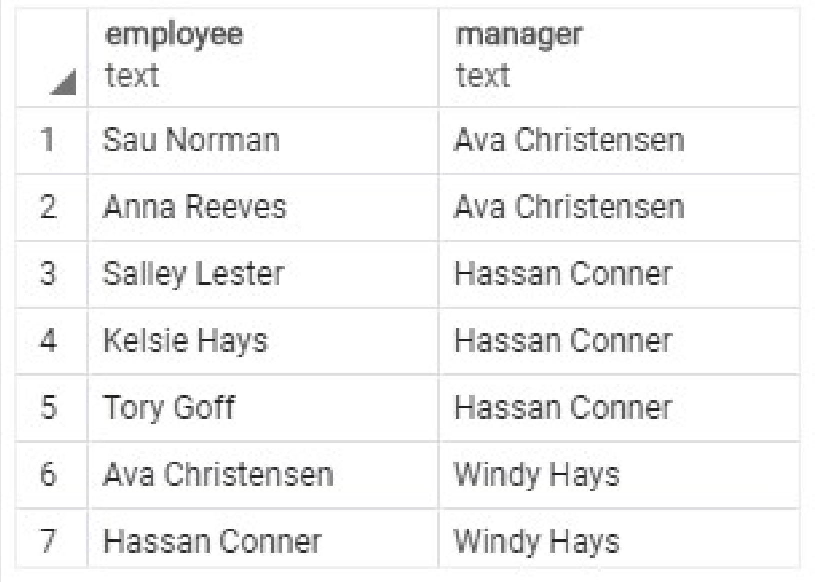
## Contoh penggunaan self join:

**Tampilkan data karyawan (employee) beserta manager dari karyawan tersebut!**

**Jawaban:**

```
SELECT P.FIRST_NAME, P.LAST_NAME,
              M.FIRST_NAME, M.LAST_NAME
FROM EMPLOYEE P
INNER JOIN EMPLOYEE M
ON P.MANAGER_ID=M.EMPLOYEE_ID;
```





| | employee<br>text | manager<br>text |
|---|---|---|
| 1 | Sau Norman | Ava Christensen |
| 2 | Anna Reeves | Ava Christensen |
| 3 | Salley Lester | Hassan Conner |
| 4 | Kelsie Hays | Hassan Conner |
| 5 | Tory Goff | Hassan Conner |
| 6 | Ava Christensen | Windy Hays |
| 7 | Hassan Conner | Windy Hays |

# Left outer, Right outer, Full outer Join

## Buat dua tabel: EMPLOYEES dan DEPARTMENTS

### Perintah pembuatan tabel:

```
DROP TABLE IF EXISTS departments;
DROP TABLE IF EXISTS employees;
CREATE TABLE departments (
        department_id serial PRIMARY KEY,
        department_name VARCHAR (255) NOT NULL
);
CREATE TABLE employees (
        employee_id serial PRIMARY KEY,
        employee_name VARCHAR (255),
        department_id INTEGER
);
```

### Perintah pengisian tabel:

```
INSERT INTO departments (department_name)
VALUES    ('Sales'),
          ('Marketing'),
          ('HR'),
          ('IT'),
          ('Production');
INSERT INTO employees (
          employee_name,
          department_id)
VALUES    ('Bette Nicholson', 1),
          ('Christian Gable', 1),
          ('Joe Swank', 2),
          ('Fred Costner', 3),
          ('Sandra Kilmer', 4),
          ('Julia Mcqueen', NULL);
```

# Left outer, Right outer, Full outer Join

## Contoh penggunaan full outer join:

**Tampilkan nama karyawan dan nama departemen (termasuk karyawan yg tdk punya departemen dan departemen yang tidak punya karyawan)**

**Jawaban:**

```
SELECT employee_name, department_name
FROM employees e
FULL OUTER JOIN departments d
ON d.department_id = e.department_id;
```

# Cross JOIN

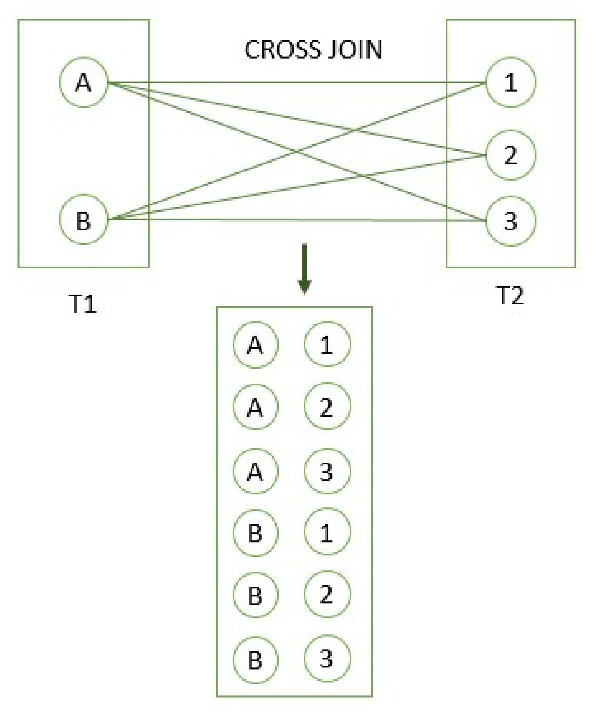
Cross join mengkombinasikan semua data dari dua tabel

**Pembuatan tabel:**

```
CREATE TABLE T1 (label CHAR(1) PRIMARY KEY);

CREATE TABLE T2 (score INT PRIMARY KEY);
```

**Pengisian tabel:**

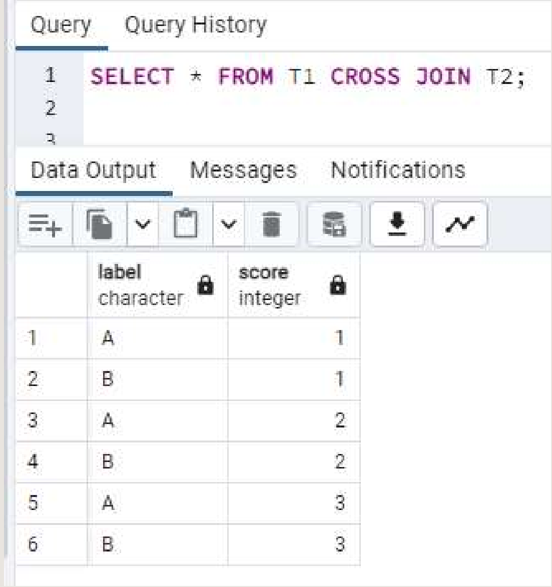
```
INSERT INTO T1 (label)

VALUES  ('A'),('B');

INSERT INTO T2 (score)

VALUES  (1), (2), (3);
```



# Cross JOIN

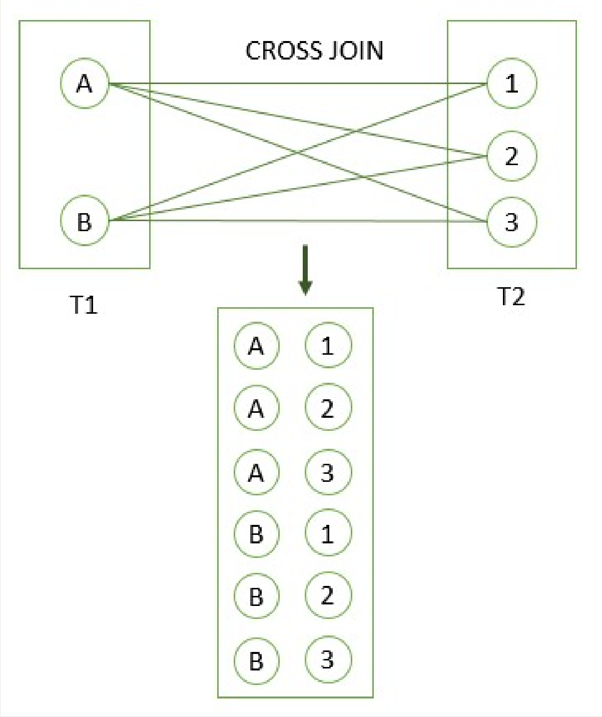
## Penggunaan cross join:

```
SELECT * FROM T1 CROSS JOIN T2;
```



```
SELECT * FROM T1 CROSS JOIN T2;
```

| | label<br>character | score<br>integer |
|---|---|---|
| 1 | A | 1 |
| 2 | B | 1 |
| 3 | A | 2 |
| 4 | B | 2 |
| 5 | A | 3 |
| 6 | B | 3 |



| | |
|---|---|
| A | 1 |
| A | 2 |
| A | 3 |
| B | 1 |
| B | 2 |
| B | 3 |

# Natural JOIN

## Buat dua tabel: CATEGORIES dan PRODUCTS

### Perintah pembuatan tabel:

```
CREATE TABLE categories (
        category_id serial PRIMARY KEY,
        category_name VARCHAR (255) NOT NULL
);
CREATE TABLE products (
        product_id serial PRIMARY KEY,
        product_name VARCHAR (255) NOT NULL,
        category_id INT NOT NULL,
        FOREIGN KEY (category_id) REFERENCES
categories (category_id)
);
```

### Perintah pengisian tabel:

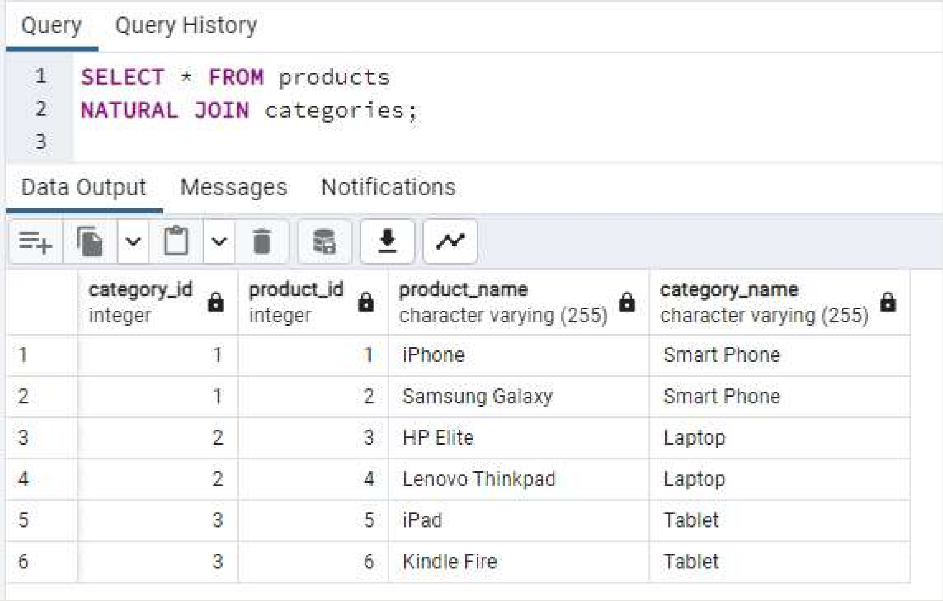
```
INSERT INTO categories (category_name)
VALUES
        ('Smart Phone'),
        ('Laptop'),
        ('Tablet');

INSERT INTO products (product_name, category_id)
VALUES
        ('iPhone', 1),
        ('Samsung Galaxy', 1),
        ('HP Elite', 2),
        ('Lenovo Thinkpad', 2),
        ('iPad', 3),
        ('Kindle Fire', 3);
```

# Natural JOIN

## Contoh penggunaan natural join

**SELECT * FROM products**

**NATURAL JOIN categories;**



```
SELECT * FROM products
NATURAL JOIN categories;
```

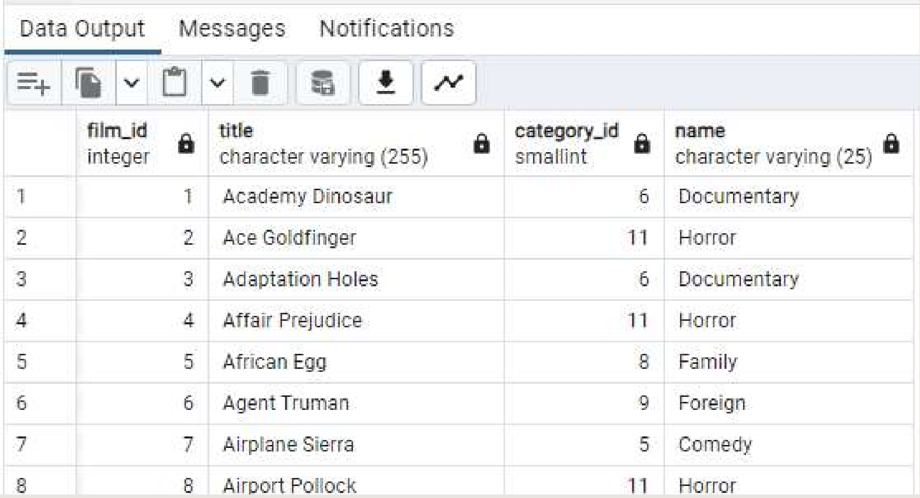
| | category_id<br>integer | product_id<br>integer | product_name<br>character varying (255) | category_name<br>character varying (255) |
|---|---|---|---|---|
| 1 | 1 | 1 | iPhone | Smart Phone |
| 2 | 1 | 2 | Samsung Galaxy | Smart Phone |
| 3 | 2 | 3 | HP Elite | Laptop |
| 4 | 2 | 4 | Lenovo Thinkpad | Laptop |
| 5 | 3 | 5 | iPad | Tablet |
| 6 | 3 | 6 | Kindle Fire | Tablet |

# Summary

- Telah dipelajari macam2 JOIN beserta contoh2nya.

# Contoh soal:

Buat perintah JOIN pada skema dvdrental, sehingga muncul data seperti pada gambar berikut:

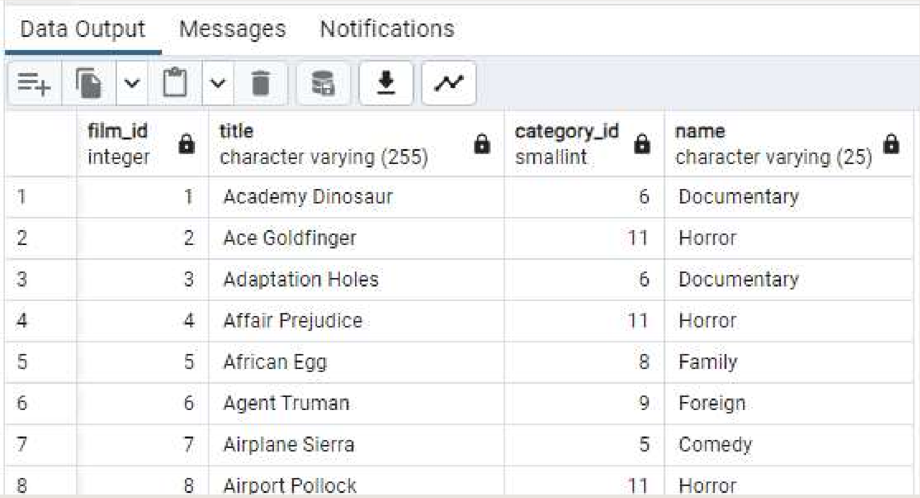


| | film_id<br>integer | title<br>character varying (255) | category_id<br>smallint | name<br>character varying (25) |
|---|---|---|---|---|
| 1 | 1 | Academy Dinosaur | 6 | Documentary |
| 2 | 2 | Ace Goldfinger | 11 | Horror |
| 3 | 3 | Adaptation Holes | 6 | Documentary |
| 4 | 4 | Affair Prejudice | 11 | Horror |
| 5 | 5 | African Egg | 8 | Family |
| 6 | 6 | Agent Truman | 9 | Foreign |
| 7 | 7 | Airplane Sierra | 5 | Comedy |
| 8 | 8 | Airport Pollock | 11 | Horror |

# Contoh soal:

Jawaban:

```
SELECT f.film_id, f.title, fc.category_id, c.name
FROM FILM f
inner join FILM_CATEGORY fc on f.film_id=fc.film_id
inner join CATEGORY C on FC.category_id=C.category_id
ORDER BY FILM_ID;
```



| | film_id<br>integer | title<br>character varying (255) | category_id<br>smallint | name<br>character varying (25) |
|---|---|---|---|---|
| 1 | 1 | Academy Dinosaur | 6 | Documentary |
| 2 | 2 | Ace Goldfinger | 11 | Horror |
| 3 | 3 | Adaptation Holes | 6 | Documentary |
| 4 | 4 | Affair Prejudice | 11 | Horror |
| 5 | 5 | African Egg | 8 | Family |
| 6 | 6 | Agent Truman | 9 | Foreign |
| 7 | 7 | Airplane Sierra | 5 | Comedy |
| 8 | 8 | Airport Pollock | 11 | Horror |

Tessy.Badriyah@gmail.com

# Thank you

- Google scholar https://scholar.google.co.id/citations?user=jBGQ0QUAAAAJ&hl=en
- ORCID https://orcid.org/0000-0003-1905-3797
- SINTA https://sinta.kemdikbud.go.id/authors/profile/6195589/