UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia
FAKULTAS ILMU KOMPUTER

Slide 11

Trigger and Stored Procedure

CSF2600700 - BASIS DATA

SEMESTER GENAP 2017/2018

These slides are a modification to the supplementary slide of "Database System", 7th edition, Elmasri/Navathe, 2015:
Chapter 7 More SQL: Complex Queries, Triggers, Views, and Schema Modification

PostgreSQL Documentation:

https://www.postgresql.org/docs/

Outline

- Stored Procedure
- Trigger

Stored Procedure

- Sebuah stored procedure adalah kumpulan dari prosedur dan statement SQL yang terdapat pada DBMS.
- Dikenali dengan nama.
- Dieksekusi sebagai sebuah kesatuan (unit).
- Di *postgresql* dikenal sebagai function.

Keuntungan Stored Procedure

- Reuse
- Mengurangi *network traffic* dan meningkatkan *performance*. (tidak ada transmisi individual SQL *statements* pada *network*.)

Simple Example

```
CREATE or REPLACE FUNCTION one()
RETURNS integer AS $$
   SELECT 1 AS result;
$$ LANGUAGE SQL;

CREATE or REPLACE FUNCTION
add_em(x integer, y integer) RETURNS integer
AS $$
   SELECT x + y;
$$ LANGUAGE SQL;
```

HELLO WORLD in SQL

```
create or replace function hello() RETURNS
text AS $$
DECLARE
  hello text;
begin
  hello := 'Hello World!';
  return hello;
END;
$$ LANGUAGE plpgsql;
```

EXECUTE PROCEDURE

Select hello();

EXAMPLE

```
CREATE OR REPLACE FUNCTION sum_salary()
RETURNS integer
AS $$
    DECLARE result integer;
    BEGIN
        Select sum(salary) into result from
employee;
        RETURN result;
    END;
$$ LANGUAGE plpgsql;
```

EXECUTE PROCEDURE
Select sum_salary();

Trigger

- Trigger merupakan kode PL/SQL yang secara otomatis dijalankan oleh DBMS jika suatu event database terjadi.
- Event tersebut bisa berupa operasi
  - INSERT, UPDATE, DELETE
- Sebuah trigger selalu dijalankan sebelum atau sesudah sebuah data row di-INSERT, di-UPDATE atau di-DELETE.
- Sebuah trigger selalu berasosiasi dengan tabel pada basis data.
- Setiap tabel bisa mempunyai satu atau lebih trigger
- Sebuah trigger dieksekusi sebagai bagian dari transaksi yang men-trigger trigger tersebut.

Keuntungan Trigger

- Trigger dapat digunakan untuk memaksakan constraint yang tidak dapat dilakukan pada perancangan dan implementasi DBMS.
- Trigger dapat secara otomatis memberikan pesan warning jika terjadi gangguan pada IC. Penggunaan trigger yg umum adalah untuk meningkatkan referential IC.
- Trigger dapat digunakan untuk update nilai pada tabel, insert tuple pada tabel, dan memanggil stored procedure yg lain.

Trigger and Stored Procedure

CREATE TRIGGER name { BEFORE | AFTER } { event [OR ...] }
ON table [FOR [EACH] { ROW | STATEMENT }]
EXECUTE PROCEDURE funcname (arguments)

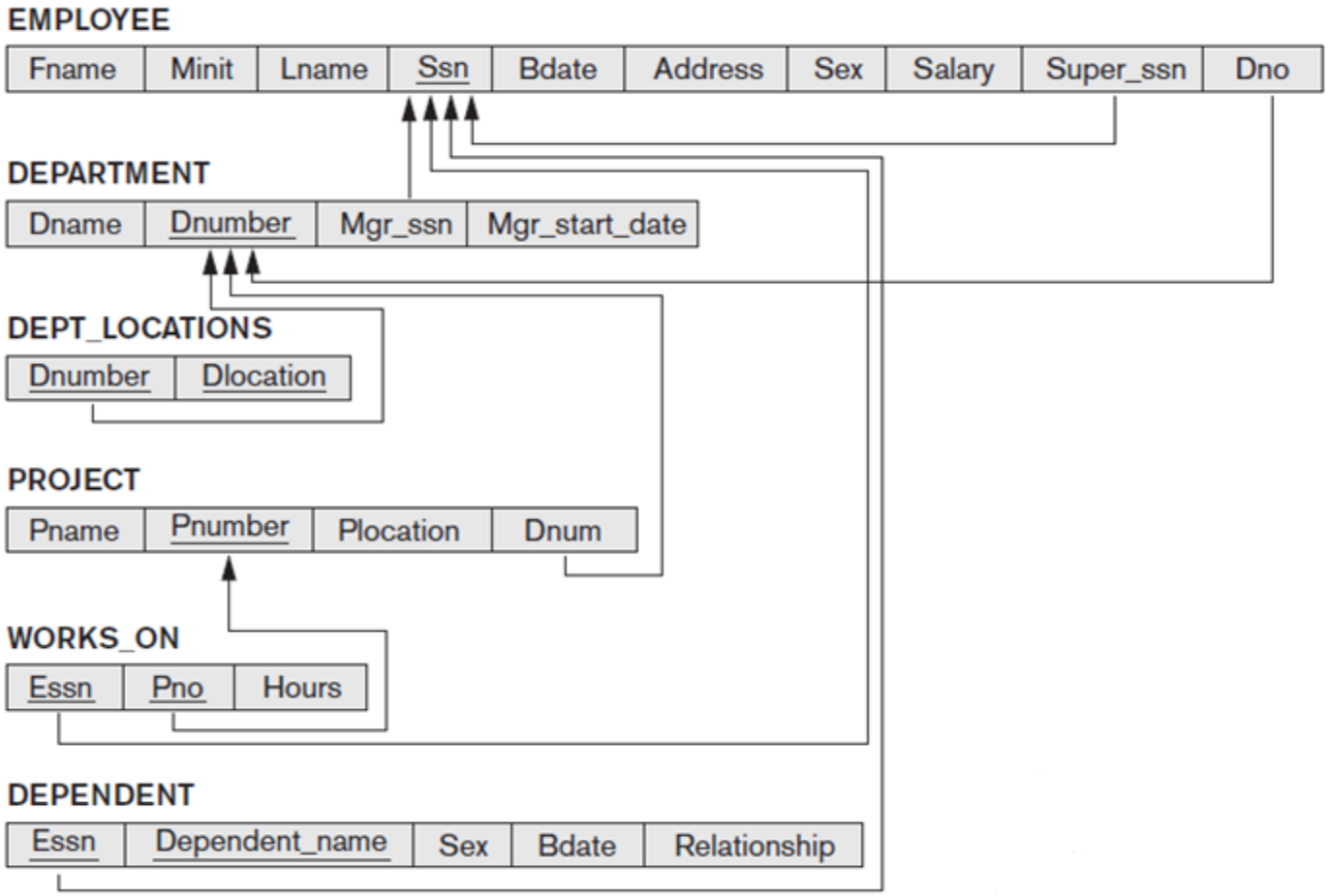
```
CREATE TRIGGER emp_stamp
   BEFORE INSERT
   ON employee
   FOR EACH ROW
   EXECUTE PROCEDURE emp_stamp();
```

CREATE TRIGGER

- - The **CREATE TRIGGER** statement is used to implement, unsurprisingly, triggers
- - Example:
- Suppose we want to check whenever an employee's salary is greater than the salary of her direct supervisor in the COMPANY DB

  Can you guess what events can trigger this situation?

Skema COMPANY



Employee's salary > Supervisor's salary

- Suppose we want to check whenever an employee's salary is greater than the salary of her direct supervisor in the COMPANY DB

  Can you guess what events can trigger this situation?
- - Inserting a new employee record
- - Updating an employee's salary
- - Updating an employee's supervisor

Employee's salary > Supervisor's salary

- Suppose we want to check whenever an employee's salary is greater than the salary of her direct supervisor in the COMPANY DB

```
CREATE TRIGGER SALARY_VIOLATION
BEFORE INSERT OR UPDATE OF SALARY, SUPERVISOR_SSN
   ON EMPLOYEE
FOR EACH ROW
WHEN (NEW.SALARY > (SELECT SALARY FROM EMPLOYEE
            WHERE SSN = NEW.SUPERVISOR_SSN))
     EXECUTE_PROCEDURE
       INFORM_SUPERVISOR(NEW.SUPERVISOR_SSN, NEW.SSN);
```

Typical TRIGGER components

- ECA = Event, Condition, Action

```
CREATE TRIGGER SALARY_VIOLATION
BEFORE INSERT OR UPDATE OF SALARY, SUPERVISOR_SSN
   ON EMPLOYEE
FOR EACH ROW
WHEN (NEW.SALARY > (SELECT SALARY FROM EMPLOYEE
            WHERE SSN = NEW.SUPERVISOR_SSN))
     EXECUTE_PROCEDURE
         INFORM_SUPERVISOR(NEW.SUPERVISOR_SSN, NEW.SSN);
```

The **event(s)**: These are usually DB updates.

Specified after the keyword BEFORE (or AFTER).

Typical TRIGGER components

- ECA = Event, Condition, Action

```
CREATE TRIGGER SALARY_VIOLATION
BEFORE INSERT OR UPDATE OF SALARY, SUPERVISOR_SSN
   ON EMPLOYEE
FOR EACH ROW
WHEN (NEW.SALARY > (SELECT SALARY FROM EMPLOYEE
            WHERE SSN = NEW.SUPERVISOR_SSN))
     EXECUTE_PROCEDURE
         INFORM_SUPERVISOR(NEW.SUPERVISOR_SSN, NEW.SSN);
```

The **condition**: the check whether the rule action should be executed. If no condition, the execution will be executed whenever the event occurs.

If there is a condition, the condition is first evaluated, and only if it is true will the rule action be executed.

The condition is specified in the WHEN clause.

Typical TRIGGER components

- ECA = Event, Condition, Action

```
CREATE TRIGGER SALARY_VIOLATION
BEFORE INSERT OR UPDATE OF SALARY, SUPERVISOR_SSN
   ON EMPLOYEE
FOR EACH ROW
WHEN (NEW.SALARY > (SELECT SALARY FROM EMPLOYEE
            WHERE SSN = NEW.SUPERVISOR_SSN))
     EXECUTE PROCEDURE INFORM_SUPERVISOR(NEW.SUPERVISOR_SSN, NEW.SSN);
```

The **action** to be taken: The action can be SQL statements or a stored procedure or function.

In this example, the action is to execute the stored procedure `INFORM_SUPERVISOR`.

Contoh

Contoh mengacu pada tabel COMPANY pada *slide* sebelumnya.

- Contoh 1 mengecek bahwa SALARY dari EMPLOYEE tidak boleh bernilai negatif.
- Contoh 2 mengakumulasi jumlah jam kerja EMPLOYEE pada PROJECT.
- Contoh 3 menghitung gaji total EMPLOYEE yang merupakan akumulasi SALARY dengan tunjangan yang bergantung pada jam kerja pada PROJECT.

Contoh 1

```
CREATE OR REPLACE FUNCTION emp_stamp()
  RETURNS trigger AS
  $$
   BEGIN
      IF (NEW.salary < 0) THEN
     RAISE EXCEPTION '% cannot have negative
  salary',
      NEW.lname;
      END IF;
      RETURN NEW;
  END;
  $$
  LANGUAGE plpgsql;
```

Contoh 1

```
CREATE TRIGGER emp_stamp
   BEFORE INSERT
   ON employee
   FOR EACH ROW
   EXECUTE PROCEDURE emp_stamp();
```

Contoh 1

- INSERT INTO EMPLOYEE VALUES ('Donald','T','Duck','123456788','1950-11-11’, 'Manhattan','M',-100,NULL,1);

```
postgres=# select * from employee;
  fname   | minit |  lname   |    ssn    |   bdate    | address  | sex |  salary  | super_ssn | dno
----------+-------+----------+-----------+------------+----------+-----+----------+-----------+-----
 John     | B     | Smith    | 123456789 | 1965-01-09 | Fondren  | M   | 30000.00 | 333445555 |   5
 Franklin | T     | Wong     | 333445555 | 1955-12-08 | Houston  | M   | 40000.00 | 888665555 |   5
 Joyce    |       | English  | 453453453 |            |          | F   | 25000.00 | 987654321 |   5
 Ramesh   |       | Narayan  | 666884444 |            |          | F   | 38000.00 | 888665555 |   5
 James    |       | Borg     | 888665555 |            |          | M   | 55000.00 | 333445555 |   1
 Jennifer | S     | Wallace  | 987654321 | 1941-06-19 | Bellaire | F   | 43000.00 | 333445555 |   4
 Ahmad    |       | Jabbar   | 987987987 |            |          | M   | 25000.00 | 987654321 |   4
 Alicia   | J     | Zelaya   | 999887777 | 1968-01-19 | Castle   | F   | 25000.00 |           |   4
(8 rows)


postgres=# INSERT INTO EMPLOYEE VALUES('Donald','T','Duck','123456788','1950-11-11','Manhattan','M','
-100',NULL,1);
ERROR:  Duck cannot have negative salary
CONTEXT:  PL/pgSQL function emp_stamp() line 2 at RAISE
```

Contoh 1

- INSERT INTO EMPLOYEE VALUES ('Donald','T','Duck','123456788','1950-11-11','Manhattan','M',20000.00, NULL,1);

```
postgres=# INSERT INTO EMPLOYEE VALUES('Donald','T','Duck','123456788','1950-11-11','Manhattan','M','
20000',NULL,1);
INSERT 0 1
postgres=# select * from employee;
  fname   | minit |  lname   |    ssn    |   bdate    |  address  | sex |  salary  | super_ssn | dno
----------+-------+----------+-----------+------------+-----------+-----+----------+-----------+-----
 John     | B     | Smith    | 123456789 | 1965-01-09 | Fondren   | M   | 30000.00 | 333445555 |   5
 Franklin | T     | Wong     | 333445555 | 1955-12-08 | Houston   | M   | 40000.00 | 888665555 |   5
 Joyce    |       | English  | 453453453 |            |           | F   | 25000.00 | 987654321 |   5
 Ramesh   |       | Narayan  | 666884444 |            |           | F   | 38000.00 | 888665555 |   5
 James    |       | Borg     | 888665555 |            |           | M   | 55000.00 | 333445555 |   1
 Jennifer | S     | Wallace  | 987654321 | 1941-06-19 | Bellaire  | F   | 43000.00 | 333445555 |   4
 Ahmad    |       | Jabbar   | 987987987 |            |           | M   | 25000.00 | 987654321 |   4
 Alicia   | J     | Zelaya   | 999887777 | 1968-01-19 | Castle    | F   | 25000.00 |           |   4
 Donald   | T     | Duck     | 123456788 | 1950-11-11 | Manhattan | M   | 20000.00 |           |   1
(9 rows)
```

Contoh 2

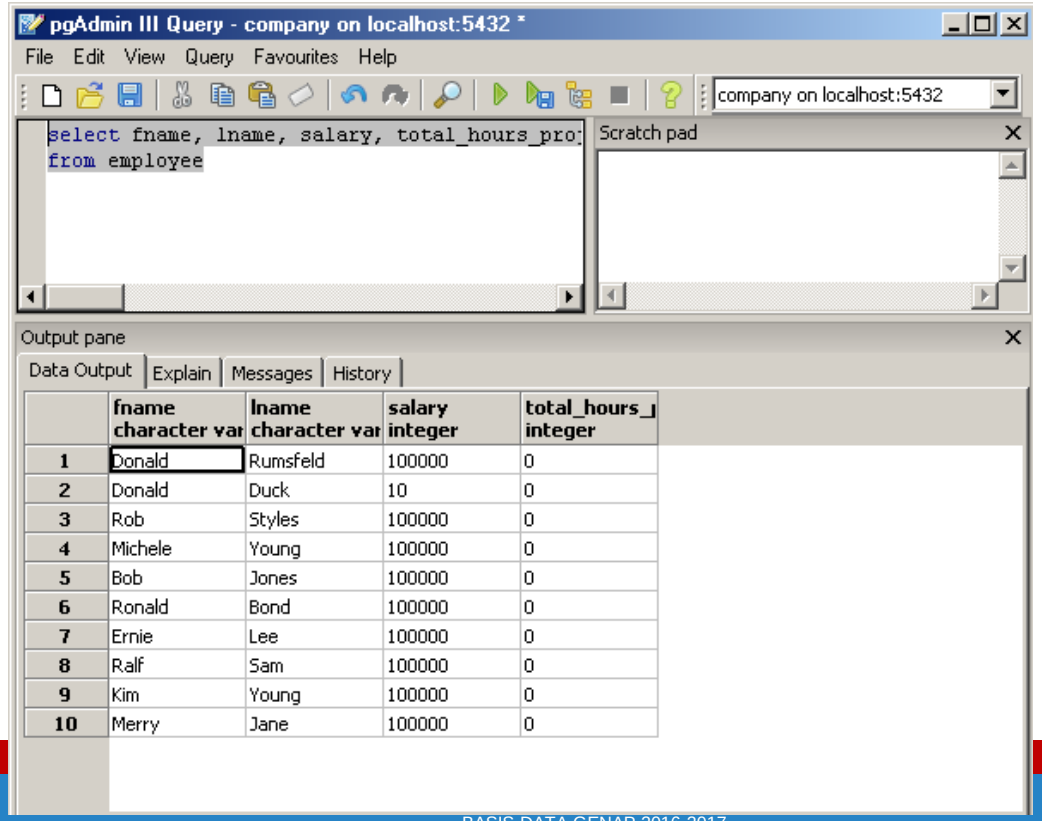
```
CREATE OR REPLACE FUNCTION emp_total_hours_proj()
  RETURNS "trigger" AS
  $$
   BEGIN
      IF (TG_OP = ‘INSERT’) THEN
  UPDATE employee SET total_hours_project =
  total_hours_project + NEW.hours
        WHERE   ssn = NEW.essn;
  RETURN NEW;

      ELSIF (TG_OP = ‘DELETE’) THEN
  UPDATE employee SET total_hours_project =
  total_hours_project - OLD.hours
        WHERE   ssn = OLD.essn;
  RETURN OLD;
      END IF;
  END;
  $$
  LANGUAGE plpgsql;
```

Contoh 2

```
CREATE TRIGGER emp_total_hours_proj
   AFTER INSERT OR DELETE
   ON works_on
   FOR EACH ROW
   EXECUTE PROCEDURE emp_total_hours_proj();
```

Contoh 2

- Sebelum meng-assign WORKS_ON
- SELECT FNAME, LNAME, SALARY, TOTAL_HOURS_PROJECT FROM EMPLOYEE



	fname character var	lname character var	salary integer	total_hours_ integer
1	Donald	Rumsfeld	100000	0
2	Donald	Duck	10	0
3	Rob	Styles	100000	0
4	Michele	Young	100000	0
5	Bob	Jones	100000	0
6	Ronald	Bond	100000	0
7	Ernie	Lee	100000	0
8	Ralf	Sam	100000	0
9	Kim	Young	100000	0
10	Merry	Jane	100000	0

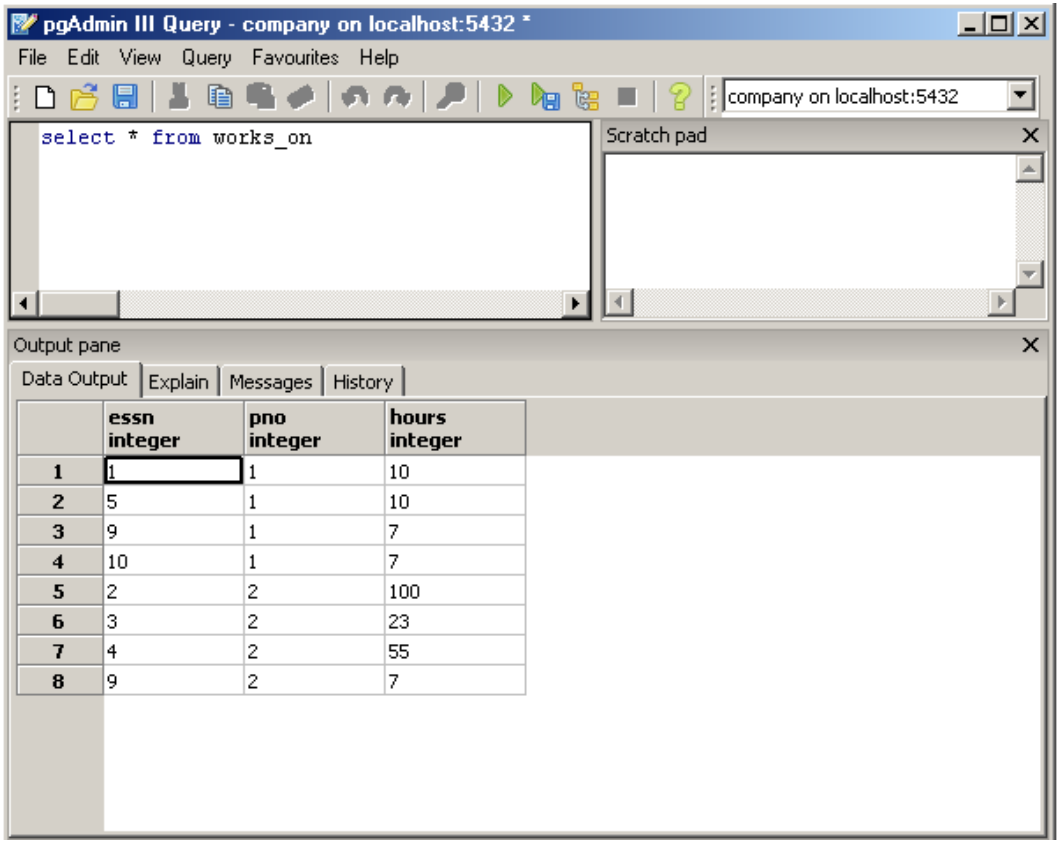
Contoh 2

- INSERT INTO PROJECT VALUES('1', 'New York', 'NYS Project', '1');
- INSERT INTO WORKS_ON VALUES('1', '1', '10');
- INSERT INTO WORKS_ON VALUES('5', '1', '10');
- INSERT INTO WORKS_ON VALUES('9', '1', '7');
- INSERT INTO WORKS_ON VALUES('10', '1', '7');

- INSERT INTO PROJECT VALUES('2', 'New York', 'NYS2 Project', '1');
- INSERT INTO WORKS_ON VALUES('2', '2', '100');
- INSERT INTO WORKS_ON VALUES('3', '2', '23');
- INSERT INTO WORKS_ON VALUES('4', '2', '55');
- INSERT INTO WORKS_ON VALUES('9', '2', '7');

Contoh 2

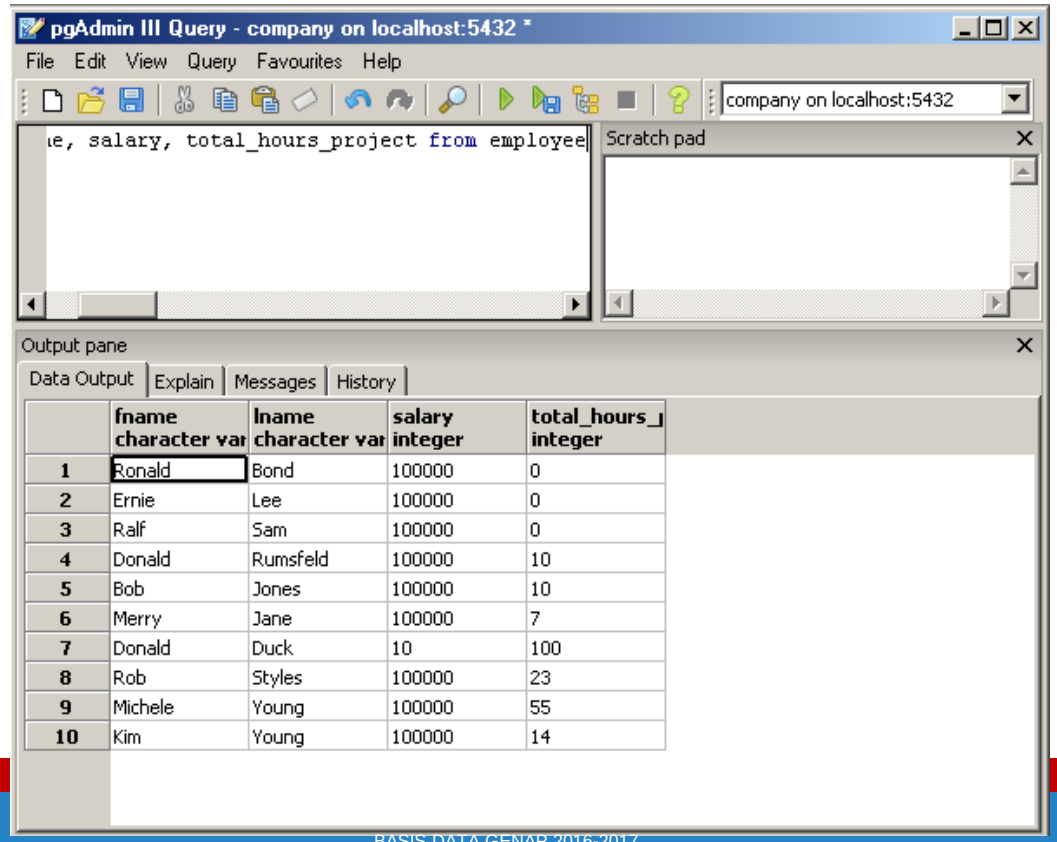
- Tabel WORKS_ON



	essn integer	pno integer	hours integer
1	1	1	10
2	5	1	10
3	9	1	7
4	10	1	7
5	2	2	100
6	3	2	23
7	4	2	55
8	9	2	7

Contoh 2

- Setelah meng-assign WORKS_ON
- **SELECT FNAME, LNAME, SALARY, TOTAL_HOURS_PROJECT FROM EMPLOYEE**



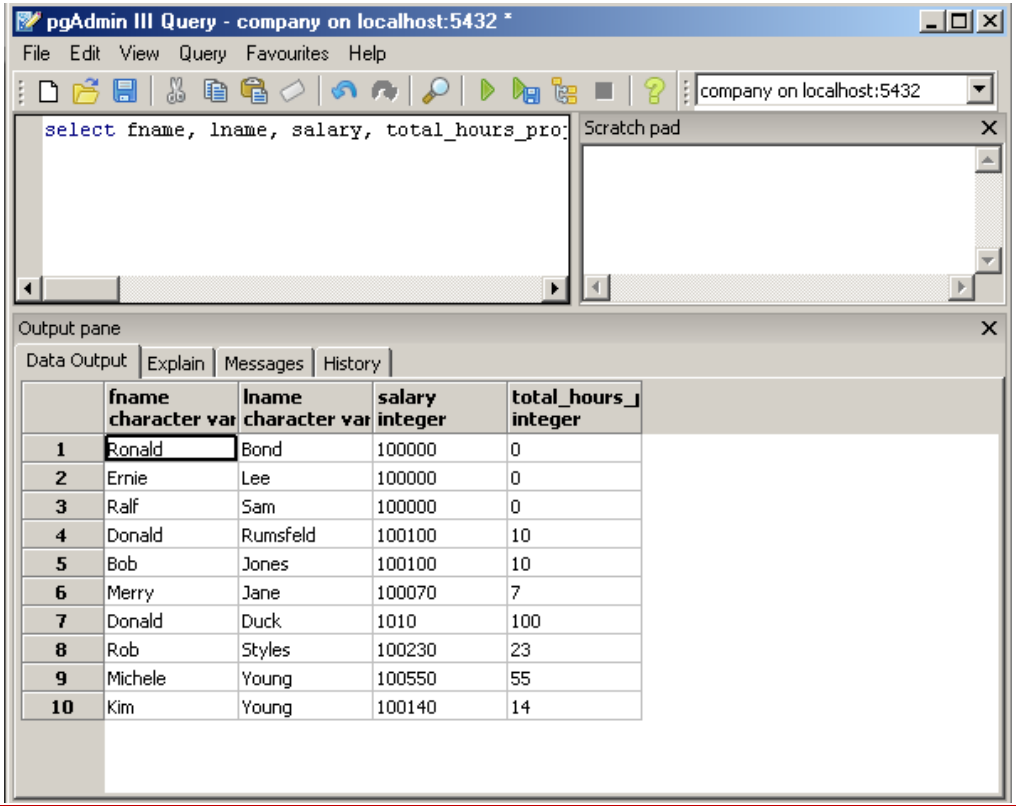
	fname character var	lname character var	salary integer	total_hours_ integer
1	Ronald	Bond	100000	0
2	Ernie	Lee	100000	0
3	Ralf	Sam	100000	0
4	Donald	Rumsfeld	100000	10
5	Bob	Jones	100000	10
6	Merry	Jane	100000	7
7	Donald	Duck	10	100
8	Rob	Styles	100000	23
9	Michele	Young	100000	55
10	Kim	Young	100000	14

Contoh 3

```
CREATE OR REPLACE FUNCTION
  procedure_salary() RETURNS void
AS $$
DECLARE
BEGIN
 update employee set salary = salary +
  total_hours_project * 10
  where total_hours_project > 0;
  END;
$$ LANGUAGE plpgsql;
```

Contoh 3

- Untuk meng-execute dapat digunakan query sebagai berikut:
- **SELECT * FROM PROCEDURE_SALARY();**
- **SELECT fname, lname, salary, total_hours_proj FROM EMPLOYEE;**



	fname character var	lname character var	salary integer	total_hours_ integer
1	Ronald	Bond	100000	0
2	Ernie	Lee	100000	0
3	Ralf	Sam	100000	0
4	Donald	Rumsfeld	100100	10
5	Bob	Jones	100100	10
6	Merry	Jane	100070	7
7	Donald	Duck	1010	100
8	Rob	Styles	100230	23
9	Michele	Young	100550	55
10	Kim	Young	100140	14

Trigger vs Stored Procedure