

# Tutorials 9 og 10

Prosedyrespråket som følger med PostgreSQL kalles PL / pgSQL, og det lar en utvikle både enkle og mer komplekse brukerdefinerte funksjoner og prosedyrer.

## Tutorial 9:

### 1.Grunnleggende:

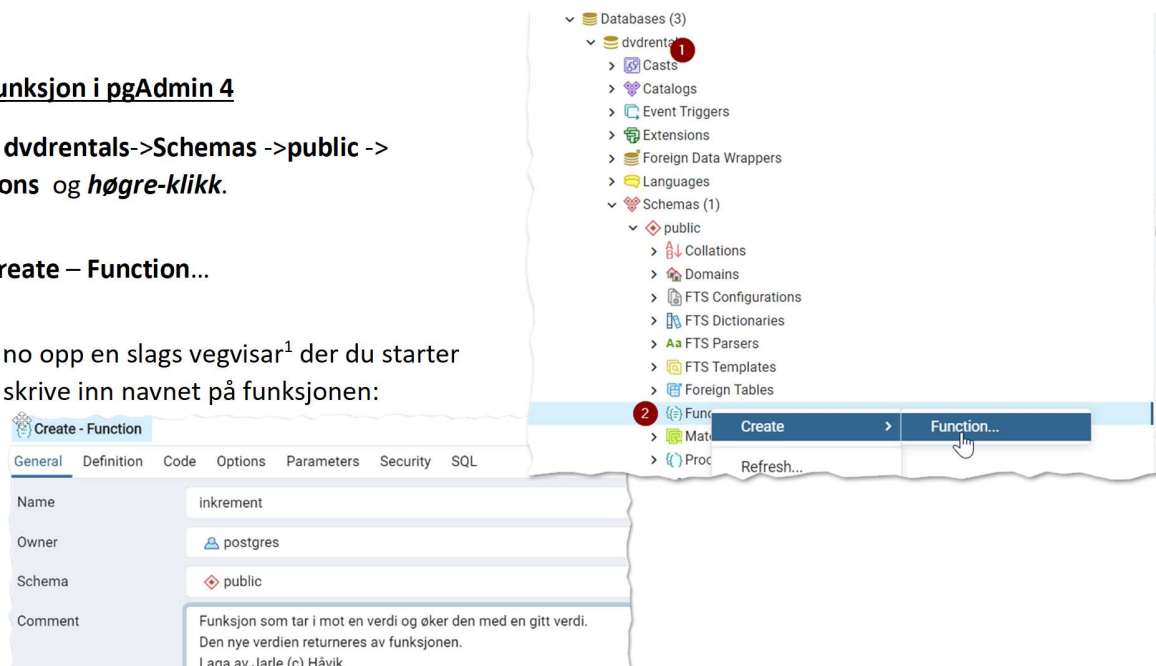
En oversikt over PL/pgSQL og hva den inneholder:

<http://www.postgresqltutorial.com/introduction-to-postgresql-stored-procedures/>

Blokkstrukturen i pl/pgSQL: <http://www.postgresqltutorial.com/plpgsql-block-structure/>

### Opprette en funksjon i pgAdmin 4

- 1) Under **dvdrentals->Schemas ->public -> Functions** og **høgre-klikk**.
- 2) Velg **Create – Function...**
- 3) Du får no opp en slags vegvisar<sup>1</sup> der du starter med å skrive inn navnet på funksjonen:



<sup>1</sup> I pgAdmin får du hjelp til hva de ulike «fanene»/stega i funksjonsdialogen - [http://127.0.0.1:52388/help/help/function\\_dialog.html](http://127.0.0.1:52388/help/help/function_dialog.html) (eller trykk på ?).

- 4) I det neste «steget» definerer du hvilken datatype som skal returneres og hvilket språk funksjonen skal nytte. Pass på at det her er valgt **plpgsql**!

På samme plass må vi også legge inn eventuelle argumenter som skal tas med inn i funksjonen.

Siden en funksjon kun skal returnere en verdi definerer du her alle som IN.

Det er ikke påkrevd med argumenter til en funksjon.

- 5) Det neste vi må gjøre er å skrive inn koden for hva som skal skje i funksjonen -> her skal startverdien økes/redueres med den verdien som er angitt som andre argument.

General Definition **Code** Options P.

```
1 BEGIN
2   return startverdi + steg;
3 END
```

- 6) Hopp over dei andre «fanene» og går til SQL for å se hvilke SQL utsagn Create-Function dialogen har bygd opp.

**Create - Function**

General **Definition** Code Options Parameters Security SQL

Return type: bigint (1)

Language: plpgsql (2)

Arguments: |

Data type: internal, c, sql, plpgsql (3)

**Create - Function**

General **Definition** Code Options Parameters Security SQL

Return type: bigint (1)

Language: plpgsql (2)

Arguments:

Data type	Mode	Argument name	Default
bigint (4)	IN (5)	startverdi (6)	0 (7)
integer	IN (5)	steg (6)	1

```
1 CREATE FUNCTION public.akkumulator(IN startverdi bigint DEFAULT 0, IN steg integer DEFAULT 1)
2   RETURNS bigint
3   LANGUAGE 'plpgsql'
4
5
6 AS $BODY$
7 BEGIN
8   return startverdi + steg;
9 END
10 $BODY$;
11
12 ALTER FUNCTION public.akkumulator(bigint, integer)
13 OWNER TO postgres;
```

- 7) Trykker så på Save og Funksjonen er oppretta.

For å eksekvere den skriv vi t.d.

Query Editor Query History

```
1 select akkumulator(234,-5) as "234-5 er";
```

Data Output Explain Messages Notifications

234-5 er
bigint
1 229

En oversikt over forskjellige funksjonsparametere fin du her:

<http://www.postgresqltutorial.com/plpgsql-function-parameters/>

Fremgangsmåte for å deklareere funksjonsvariabler finn du her:

<http://www.postgresqltutorial.com/plpgsql-variables/>

## **2.Kontroll struktur:**

Tre former for IF-utsagn: <http://www.postgresqltutorial.com/plpgsql-if-else-statements/>

To former for CASE-utsagn: <http://www.postgresqltutorial.com/plpgsql-case-statement/>

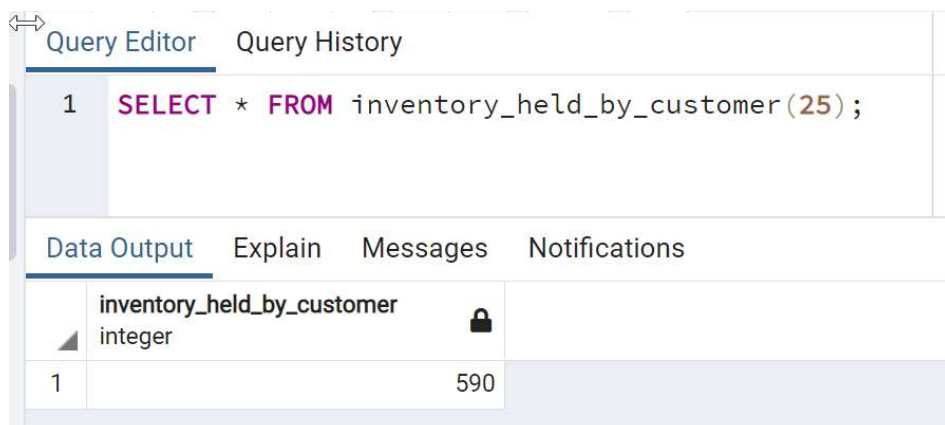
Ulike iterasjonsstrukturer: <http://www.postgresqltutorial.com/plpgsql-loop-statements/>

## **3. Håndtering av unntak (exceptions):**

<http://www.postgresqltutorial.com/plpgsql-errors-messages/>

For å kalle en funksjon i PostgreSQL, bruk "Query Editor" og skriv dette utsagnet:

"SELECT \* FROM inventory\_held\_by\_customer(25);"



The screenshot shows a PostgreSQL Query Editor window. The 'Query Editor' tab is active, displaying the query: `1 SELECT * FROM inventory_held_by_customer(25);`. Below the query editor, the 'Data Output' tab is active, showing the results of the query. The results are displayed in a table with two columns: 'inventory\_held\_by\_customer' and 'integer'. The first row shows the value 590.

	inventory_held_by_customer	integer
1		590

Query Editor

Query History

Scratch Pad

1

SELECT \* FROM rental;

Data Output

Explain

Messages

Notifications

	rental_id [PK] integer	rental_date timestamp without time zone	inventory_id integer	customer_id smallint	return_date timestamp without time zone
6	15455	2005-08-23 01:05:00	3991	238	2005-08-26 22:56:00
7	15456	2005-08-23 01:07:01	2451	262	2005-08-24 23:28:01
8	15457	2005-08-23 01:07:37	4539	306	2005-08-26 19:46:37
9	15458	2006-02-14 15:16:03	25	590	[null]
10	15459	2005-08-23 01:09:48	2058	346	2005-08-24 04:52:48
11	15460	2005-08-23 01:10:42	2907	20	2005-08-28 20:49:42
12	15461	2005-08-23 01:13:52	4542	103	2005-08-30 00:44:52

Spørringen over kaller på en funksjon som er skrevet i PL/pgSQL og som er lagret med namnet "inventory\_held\_by\_customer", funksjonen tar som argument (inventory\_id) og returnerer en integer verdi (customer\_id). Funksjonene søker igjennom rental-tabellen etter inventory\_id = "25" – som er den parameterverdi vi sender inn som argument til funksjonen (p\_inventory\_id). I tillegg sjekker en på predikatet return\_date=NULL, som indikerer at dvd-en ennå ikke er returnert av denne kunden.

Functions

- > f\_group\_concat(text,text)
- > f\_add(int4,int4)
- > f\_adding(int4,int4)
- > f\_dummy()
- > f\_film\_in\_stock(int4,int4)
- > f\_film\_not\_in\_stock(int4,int4)
- > f\_get\_customer\_balance(int4,timesta
- > f\_group\_concat(text)
- > f\_helverden()
- > f\_inventory\_held\_by\_customer(int4)
- > f\_inventory\_in\_stock(int4)
- > f\_last\_day(timestamp)
- > f\_last\_updated()
- > f\_rewards\_report(int4,numeric)
- > f\_rania()

Function parameters

Dependencies

Properties / ...

Permissions

CREATE OR REPLACE FUNCTION public.inventory\_held\_by\_customer(p\_inventory\_id integer)

RETURNS integer

LANGUAGE plpgsql

AS \$function\$

DECLARE

v\_customer\_id INTEGER;

BEGIN

SELECT customer\_id INTO v\_customer\_id

FROM rental

WHERE return\_date IS NULL

AND inventory\_id = p\_inventory\_id;

RETURN v\_customer\_id;

END \$function\$

;

Vert deklarert for at vi skal ta vare på den og returnere den i slutten av funksjonen. Viktig her at datatype er lik datatypen som skal returneres

## **Tutorial 10: Andre funksjoner**

Definere flere funksjoner med samme navn, men forskjellig argument liste:

<http://www.postgresqltutorial.com/plpgsql-function-overloading/>

Utvikling av en funksjon som returnerer en tabell:

<http://www.postgresqltutorial.com/plpgsql-function-returns-a-table/>

Bruk av konstanter for å lette lesbarheten og vedlikeholdet:

<http://www.postgresqltutorial.com/plpgsql-constants/>