Title

PostgreSQL Advanced: Unlock the Power of the World's Most Advanced Open-Source Database

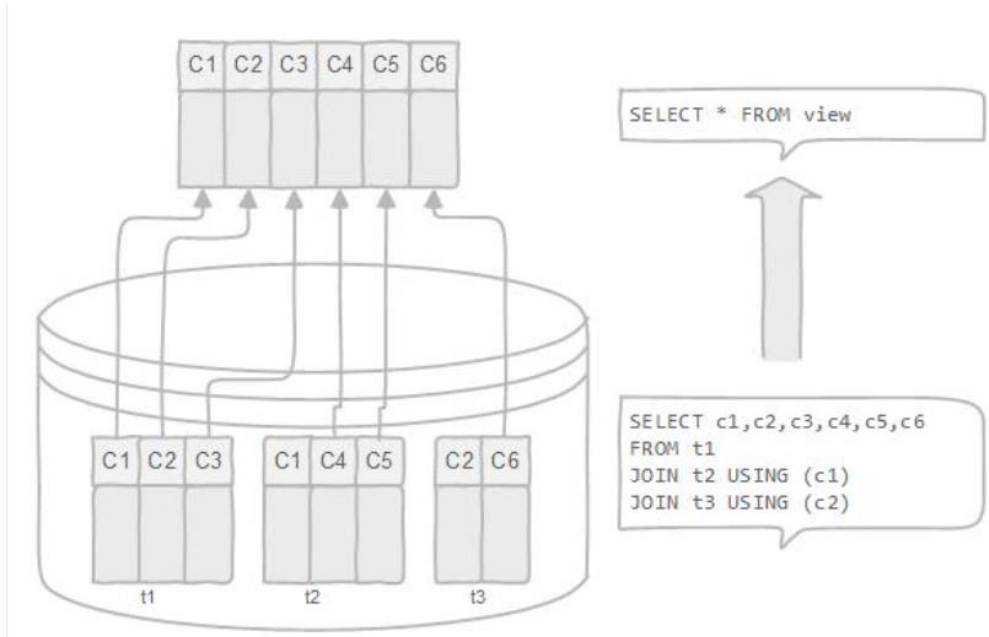
Agenda

- PostgreSQL Views
- PostgreSQL Functions
- PostgreSQL Stored Procedures
- PostgreSQL Triggers
- PostgreSQL Indexes

PostgreSQL Views

A view is defined based on one or more tables which are known as base tables.

A view is useful for wrapping a commonly used complex query.



Create View

```
CREATE OR REPLACE view_name
AS
query
```

Syntax

Example

```
CREATE VIEW customer_master AS
  SELECT cu.customer_id AS id,
    cu.first_name || ' ' || cu.last_name AS name,
    a.address,
    a.postal_code AS "zip code",
    a.phone,
    city.city,
    country.country,
        CASE
            WHEN cu.activebool THEN 'active'
            ELSE ''
        END AS notes,
    cu.store_id AS sid,
    cu.email
   FROM customer cu
     INNER JOIN address a USING (address_id)
     INNER JOIN city USING (city_id)
     INNER JOIN country USING (country_id);
```

Alter and Drop Views

Alter View :

```
ALTER VIEW customer_master RENAME TO customer_info;
```

Drop View:

```
DROP VIEW [ IF EXISTS ] view_name;
```

PostgreSQL Functions

Syntax:

```
create [or replace] function function_name(param_list)
   returns return_type
   language plpgsql
  as
$$
declare
-- variable declaration
begin
 -- logic
end;
$$
```

TatvaSoft

PostgreSQL Functions

Example:

```
create function get_film_count(len_from int, len_to int)
returns int
language plpgsql
as
$$
declare
   film_count integer;
begin
   select count(*)
   into film_count
   from film
   where length between len_from and len_to;

   return film_count;
end;
$$;
```

```
drop function [if exists] function_name(argument_list)
[cascade | restrict]
```

```
select get_film_count(40,90);
```

```
 get_film_count
----------------
            325
(1 row)
```

TatvaSoft

Function Parameters

PL/pgSQL supports three parameter modes: in, out, and inout. A parameter takes the in mode by default if you do not explicitly specify it.

Out parameter:

```
create or replace function get_film_stat(
    out min_len int,
    out max_len int,
    out avg_len numeric)
language plpgsql
as $$
begin

  select min(length),
         max(length),
                 avg(length)::numeric(5,1)
  into min_len, max_len, avg_len
  from film;

end;$$
```

```
select get_film_stat();
```

```
select * from get_film_stat();
```

inout parameter:

```
create or replace function swap(
        inout x int,
        inout y int
)
language plpgsql
as $$
begin
   select x,y into y,x;
end; $$;
```

```
select * from swap(10,20);
```

	x integer	y integer
1	20	10

TatvaSoft

PostgreSQL Stored Procedures

A drawback of user-defined functions is that they cannot execute transactions. In other words, inside a user-defined function, you cannot start a transaction, and commit or rollback it.

```
create [or replace] procedure procedure_name(parameter_list)
language plpgsql
as $$
declare
-- variable declaration
begin
-- stored procedure body
end; $$
```

```
call stored_procedure_name(argument_list);
```

PostgreSQL Triggers

A PostgreSQL trigger is a function invoked automatically whenever an event associated with a table occurs. An event could be any of the following: INSERT, UPDATE, DELETE or TRUNCATE.

A trigger is a special user-defined function associated with a table. To create a new trigger, you define a trigger function first, and then bind this trigger function to a table.

The difference between a trigger and a user-defined function is that a trigger is automatically invoked when a triggering event occurs.

```
CREATE FUNCTION trigger_function()
   RETURNS TRIGGER
   LANGUAGE PLPGSQL
AS $$
BEGIN
   -- trigger logic
END;
$$
```

```
CREATE TRIGGER trigger_name
   {BEFORE | AFTER} { event }
   ON table_name
   [FOR [EACH] { ROW | STATEMENT }]
       EXECUTE PROCEDURE trigger_function
```

Alter and Drop Triggers

Alter Trigger :

```
ALTER TRIGGER trigger_name
ON table_name
RENAME TO new_trigger_name;
```

Drop Trigger:

```
DROP TRIGGER [IF EXISTS] trigger_name
ON table_name [ CASCADE | RESTRICT ];
```

Enable and Disable Triggers

Enable Trigger :

```
ALTER TABLE table_name
ENABLE TRIGGER trigger_name |  ALL;
```

Disable Trigger:

```
ALTER TABLE table_name
DISABLE TRIGGER trigger_name | ALL
```

PostgreSQL Indexes

PostgreSQL indexes are effective tools to enhance database performance. Indexes help the database server find specific rows much faster than it could do without indexes.

Create Index:

```
CREATE INDEX index_name ON table_name [USING method]
(
    column_name [ASC | DESC] [NULLS {FIRST | LAST }],
    ...
);
```

Drop Index:

```
DROP INDEX  [ CONCURRENTLY]
[ IF EXISTS ]  index_name
[ CASCADE | RESTRICT ];
```

TatvaSoft

References

- https://www.postgresqltutorial.com/
- https://www.postgresql.org/docs/

Thank You

THANK YOU

ANY QUESTIONS ?