netlink
digital solutions

DATABASE

www.netlink.com

Stored Procedure

- A stored procedure is a prepared SQL code that you can save, so the code can be reused over and over again.
- So if you have an SQL query that you write over and over again, save it as a stored procedure, and then just call it to execute it.
- You can also pass parameters to a stored procedure, so that the stored procedure can act based on the parameter value(s) that is passed.

- In Postgres, the main functional difference between a function and a stored procedure is that a function returns a result, whereas a stored procedure does not.
- This is because the intention behind a stored procedure is to perform some sort of activity and then finish, which would then return control to the caller.

Stored Procedure

- A stored procedure is a prepared SQL code that you can save, so the code can be reused over and over again.
- So if you have an SQL query that you write over and over again, save it as a stored procedure, and then just call it to execute it.
- You can also pass parameters to a stored procedure, so that the stored procedure can act based on the parameter value(s) that is passed.

- In Postgres, the main functional difference between a function and a stored procedure is that a function returns a result, whereas a stored procedure does not.
- This is because the intention behind a stored procedure is to perform some sort of activity and then finish, which would then return control to the caller.

Stored Procedure

- So far, you have learned how to define user-defined functions using the create function statement.
- 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.

- A stored procedure is a prepared SQL code that you can save, so the code can be reused over and over again.
- So if you have an SQL query that you write over and over again, save it as a stored procedure, and then just call it to execute it.
- You can also pass parameters to a stored procedure, so that the stored procedure can act based on the parameter value(s) that is passed.

- In Postgres, the main functional difference between a function and a stored procedure is that a function returns a result, whereas a stored procedure does not.
- This is because the intention behind a stored procedure is to perform some sort of activity and then finish, which would then return control to the caller.

Advantages

Advantages of using PostgreSQL stored procedures:

- Reduce the number of round trips between applications and database servers. All SQL statements are wrapped inside a function stored in the PostgreSQL database server so the application only has to issue a function call to get the result back instead of sending multiple SQL statements and wait for the result between each call.

- Increase application performance because the user-defined functions and stored procedures are pre-compiled and stored in the PostgreSQL database server.

- Reusable in many applications. Once you develop a function, you can reuse it in any applications.

Example

Syntax:

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

```
-------table creation-------

create table accounts (
    id int generated by default as identity,
    name varchar(100) not null,
    balance dec(15,2) not null,
    primary key(id)
);
```

```
insert into accounts(name,balance)
values('Bob',10000);

insert into accounts(name,balance)
values('Alice',10000);
```

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

Example

```
create or replace procedure transfer(
   sender int,
   receiver int,
   amount dec
)
language plpgsql
as $$
begin
    -- subtracting the amount from the sender's account
    update accounts
    set balance = balance - amount
    where id = sender;

    -- adding the amount to the receiver's account
    update accounts
    set balance = balance + amount
    where id = receiver;

    commit;
end;$$
```

```
call stored_procedure_name(argument_list);

call transfer(1,2,1000);
call transfer (2,1,2000);

SELECT * FROM accounts;
```


netlink
digital solutions
Thank you
www.netlink.com