业务逻辑可以通过程序存储在PostgreSQL中，这个程序就是存储过程。存储过程使执行业务逻辑的代码可以从应用程序中移动到数据库。从而，代码存储一次能够被多个程序使用。

在用户定义的函数中，不能启动事务，并提交或回滚它，PostgreSQL 11 引入了支持事务的存储过程。

**创建存储过程**

语法格式：

create [or replace] procedure procedure\_name(parameter\_list)

language plpgsql

as $$

declare

-- variable declaration

begin

-- stored procedure body

end; $$

存储过程中的参数可以有in和inout模式。但没有out模式

存储过程不返回值。不能return在存储过程中使用带有值的语句，但是，可以使用不带值的return来停止存储过程

如果要从存储过程中返回值，可以将参数与inout模式一起使用

示例1：创建一个转账的存储过程。

create table accounts (id int,name varchar(100),amount decimal(15,2));

insert into accounts values(1,'zhangsan',500);

insert into accounts values(2,'lisi',1000);

create or replace procedure transfer(id1 int,id2 int,p\_amount decimal)

language plpgsql

as $$

begin

update accounts set amount = amount - p\_amount where id = id1;

commit;

raise exception 'this is test';

update accounts set amount = amount + p\_amount where id = id2;

end $$;

示例2：创建一个转账的函数。

create or replace function transfer2(id1 int,id2 int,p\_amount decimal)

returns int

language plpgsql

as $$

begin

update accounts set amount = amount - p\_amount where id = id1;

commit;

raise exception 'this is test';

update accounts set amount = amount + p\_amount where id = id2;

return 1;

end $$;

示例3：函数必须有返回值

create or replace function transfer3()

returns void

language plpgsql

as $$

begin

raise notice 'this is test';

end $$;

**调用存储过程**

调用存储过程，使用CALL语句：

call stored\_procedure\_name(argument\_list);

例如，调用存储过程从zhangsan账号转账200到lisi账户。

call transfer(1,2,200);

**管理存储过程**

删除存储过程

drop procedure if exists transfer, transfer2;

重命名存储过程：

ALTER PROCEDURE transfer3 RENAME TO transfer33;