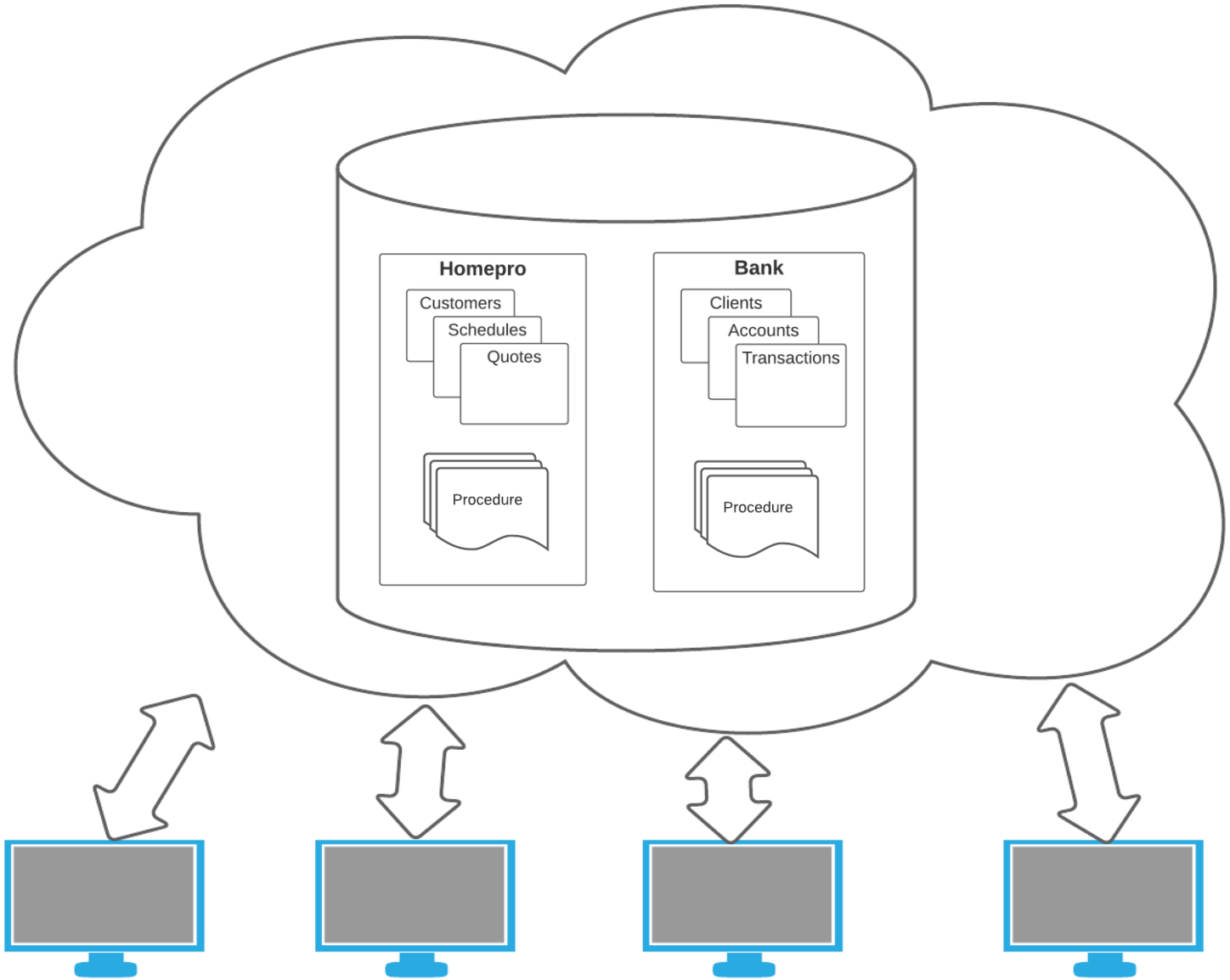
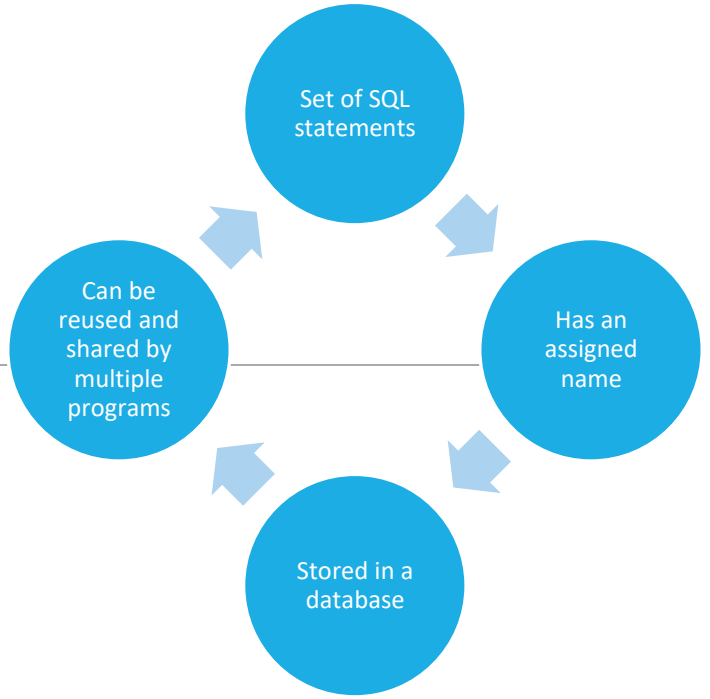

Homepro
Customers
Schedules
Quotes
Procedure
Bank
Clients
Accounts
Transactions
Procedure

Stored Procedures

A **stored procedure** is a set of structured query language (SQL) statements with an assigned name, which are **stored** in a database management, so it can be reused and shared by multiple programs.



Create procedure basic syntax

Declaration

```
CREATE PROCEDURE <SchemaName>.<ProcedureName>
AS
BEGIN

  --Your code ..
  SELECT

END
```

Usage

```
Execute <schemaName>.<ProcedureName>
```

Example

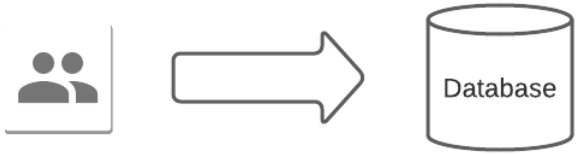
```
CREATE PROCEDURE HomePro.GetAllCustomers
AS
BEGIN
        Select
                CustomerId, FirstName, LastName, ..
        From HomePro.Customers
END
-----
ALTER PROCEDURE HomePro.GetAllCustomers
AS
BEGIN
                .....
END
```

```
Execute HomePro.GetAllCustomers
```

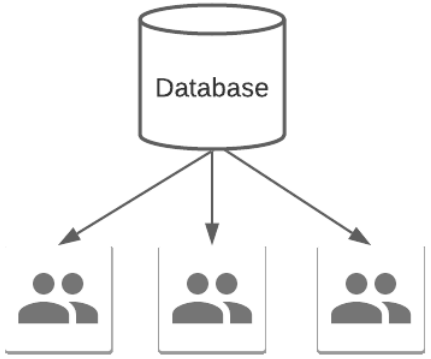
Or

```
Exec HomePro.GetAllCustomers
```

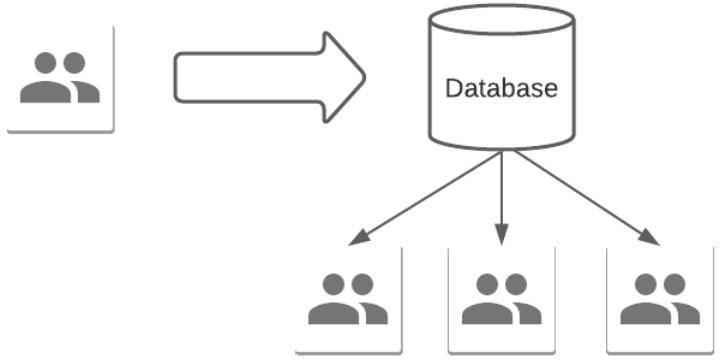
Benefits of using procedures



Store



Share



Distribute

Naming conventions

Schema name	• HomePro.* • Bank.*
Procedures action name: GET, SET, UPDATE and so on	• HomePro.Get* • Bank.Get*
Detail of actions: AllCustomers, ClientsByAge	• HomePro.GetAllCustomers • Bank.GetClientByAge
Alias or owner's name for distinguish	• HomePro.GetAllCustomers_Andrew • Bank.GetClientByAge_Andrew

Parameters

DECLARATION

```
CREATE PROCEDURE Bank.GetClientsByAge_Andrey
        @Age int
AS
BEGIN
    select ClientId, FirstName,
    LastName
    from Bank.Clients
    where age > @Age
END
```

USAGE

```
EXEC Bank.GetClientsByAge_Andrey
        @Age = 10
```

Verify the passed value of parameter

```
CREATE PROCEDURE Bank.GetClientsByAge_Andrey
        @Age int
AS
BEGIN
    if (@Age < 10 or @Age > 100)
    begin
        Raiserror ('The parameter Age is not valid ', 16,10);
        Return
    end

    select ClientId, FirstName, LastName, Age
    from Bank.Clients
    where age > @Age
END
```