Как правило, мы в работе с БД используем одни и те же запросы, либо набор последовательных запросов. Хранимые процедуры позволяют объединить последовательность запросов и сохранить их на сервере. Это очень удобный инструмент, и сейчас вы в этом убедитесь. Начнем с синтаксиса:

CREATE PROCEDURE имя\_процедуры (параметры)

begin

операторы

end

Параметры это те данные, которые мы будем передавать процедуре при ее вызове, а операторы - это собственно запросы. Давайте напишем свою первую процедуру и убедимся в ее удобстве. В [уроке 10](http://www.site-do.ru/db/sql10.php), когда мы добавляли новые записи в БД shop, мы использовали стандартный запрос на добавление вида:

INSERT INTO customers (name, email) VALUE ('Иванов Сергей', 'sergo@mail.ru');

Т.к. подобный запрос мы будем использовать каждый раз, когда нам необходимо будет добавить нового покупателя, то вполне уместно оформить его в виде процедуры:

CREATE PROCEDURE ins\_cust(n CHAR(50), e CHAR(50))

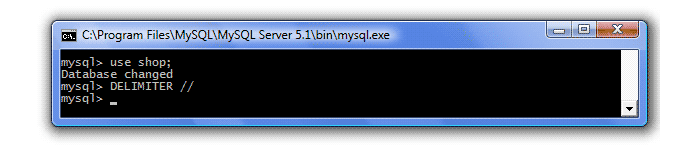
begin

insert into customers (name, email) value (n, e);

end

Обратите внимание, как задаются параметры: необходимо дать имя параметру и указать его тип, а в теле процедуры мы уже используем имена параметров. Один нюанс. Как вы помните, точка с запятой означает конец запроса и отправляет его на выполнение, что в данном случае неприемлемо. Поэтому, прежде, чем написать процедуру необходимо переопределить разделитель с ; на "//", чтобы запрос не отправлялся раньше времени. Делается это с помощью оператора DELIMITER //:

DELIMITER //

  
Таким образом, мы указали СУБД, что выполнять команды теперь следует после //. Следует помнить, что переопределение разделителя осуществляется только на один сеанс работы, т.е. при следующем сеансе работы с MySql разделитель снова станет точкой с запятой и при необходимости его придется снова переопределять. Теперь можно разместить процедуру:

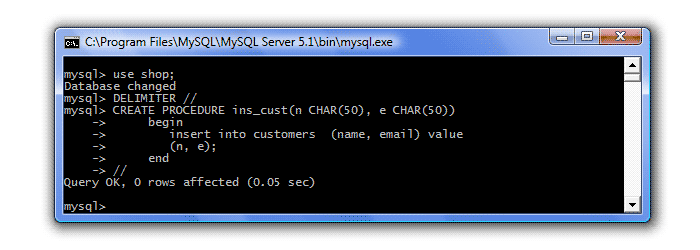
CREATE PROCEDURE ins\_cust(n CHAR(50), e CHAR(50))

begin

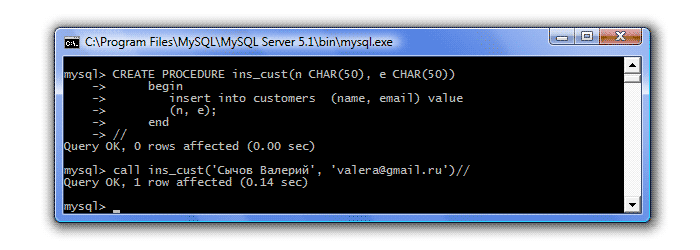
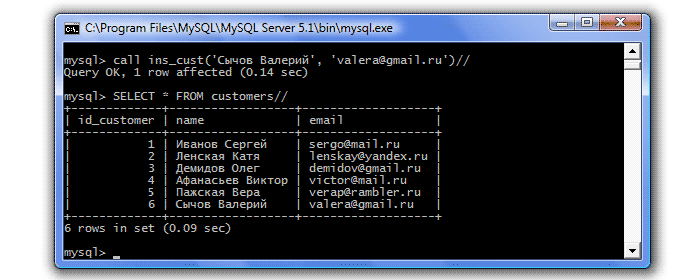
insert into customers (name, email) value (n, e);

end

//

****  
Итак, процедура создана. Теперь, когда нам понадобится ввести нового покупателя нам достаточно ее вызвать, указав необходимые параметры. Для вызова хранимой процедуры используется оператор CALL, после которого указывается имя процедуры и ее параметры. Давайте добавим нового покупателя в нашу таблицу Покупатели (customers):

call ins\_cust('Сычов Валерий', 'valera@gmail.ru')//

  
  
Согласитесь, что так гораздо проще, чем писать каждый раз полный запрос. Проверим, работает ли процедура, посмотрев, появился ли новый покупатель в таблице Покупатели (customers):  
  


Появился, процедура работает, и будет работать всегда, пока мы ее не удалим с помощью оператора

DROP PROCEDURE название\_процедуры.

Как было сказано в начале урока, процедуры позволяют объединить последовательность запросов. Давайте посмотрим, как это делается. Помните в [уроке 11](http://www.site-do.ru/db/sql11.php) мы хотели узнать, на какую сумму нам привез товар поставщик "Дом печати"? Для этого нам пришлось использовать вложенные запросы, объединения, вычисляемые столбцы и представления. А если мы захотим узнать, на какую сумму нам привез товар другой поставщик? Придется составлять новые запросы, объединения и т.д.

Проще один раз написать хранимую процедуру для этого действия.

Казалось бы, проще всего взять уже написанные в уроке 11 представление и запрос к нему, объединить в хранимую процедуру и сделать идентификатор поставщика (id\_vendor) входным параметром, вот так:

CREATE PROCEDURE sum\_vendor(i INT)

begin

CREATE VIEW report\_vendor AS SELECT magazine\_incoming.id\_product, magazine\_incoming.quantity,

prices.price, magazine\_incoming.quantity\*prices.price AS summa FROM magazine\_incoming, prices

WHERE magazine\_incoming.id\_product= prices.id\_product AND id\_incoming=

(SELECT id\_incoming FROM incoming WHERE id\_vendor=i);

SELECT SUM(summa) FROM report\_vendor;

end

//

Но так процедура работать не будет. Все дело в том, что в представлениях не могут использоваться параметры. Поэтому нам придется несколько изменить последовательность запросов. Сначала мы создадим представление, которое будет выводить идентификатор поставщика (id\_vendor), идентификатор продукта (id\_product), количество (quantity), цену (price) и сумму (summa) из трех таблиц Поставки (incoming), Журнал поставок (magazine\_incoming), Цены (prices):

CREATE VIEW report\_vendor AS SELECT incoming.id\_vendor,

magazine\_incoming.id\_product, magazine\_incoming.quantity,

prices.price, magazine\_incoming.quantity\*prices.price AS summa

FROM incoming, magazine\_incoming, prices

WHERE magazine\_incoming.id\_product= prices.id\_product AND

magazine\_incoming.id\_incoming= incoming.id\_incoming;

А потом создадим запрос, который просуммирует суммы поставок интересующего нас поставщика, например, с id\_vendor=2:

SELECT SUM(summa) FROM report\_vendor WHERE id\_vendor=2;

Вот теперь мы можем объединить два этих запроса в хранимую процедуру, где входным параметром будет идентификатор поставщика (id\_vendor), который будет подставляться во второй запрос, но не в представление:

CREATE PROCEDURE sum\_vendor(i INT)

begin

CREATE VIEW report\_vendor AS SELECT incoming.id\_vendor,

magazine\_incoming.id\_product, magazine\_incoming.quantity,

prices.price, magazine\_incoming.quantity\*prices.price AS summa

FROM incoming, magazine\_incoming, prices

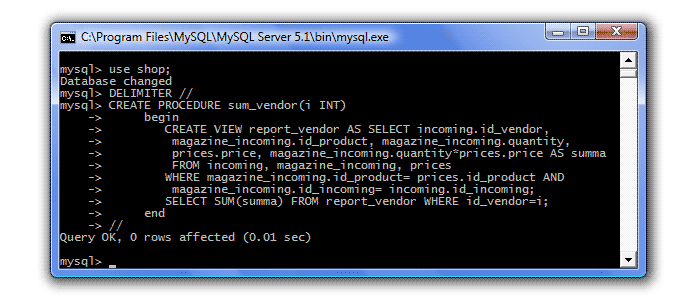
WHERE magazine\_incoming.id\_product= prices.id\_product AND

magazine\_incoming.id\_incoming= incoming.id\_incoming;

SELECT SUM(summa) FROM report\_vendor WHERE id\_vendor=i;

end

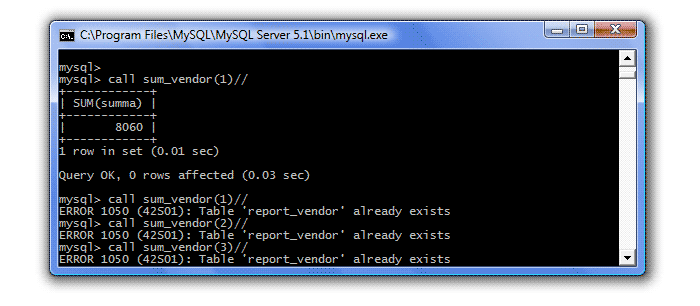
//

  
  
Проверим работу процедуры, с разными входными параметрами:

call sum\_vendor(1)//

call sum\_vendor(2)//

call sum\_vendor(3)//



Как видите, процедура срабатывает один раз, а затем выдает ошибку, говоря нам, что представление report\_vendor уже имеется в БД. Так происходит потому, что при обращении к процедуре в первый раз, она создает представление. При обращении во второй раз, она снова пытается создать представление, но оно уже есть, поэтому и появляется ошибка. Чтобы избежать этого возможно два варианта.

Первый - вынести представление из процедуры. То есть мы один раз создадим представление, а процедура будет лишь к нему обращаться, но не создавать его. Предварительно не забудет удалить уже созданную процедуру и представление:

DROP PROCEDURE sum\_vendor//

DROP VIEW report\_vendor//

CREATE VIEW report\_vendor AS SELECT incoming.id\_vendor,

magazine\_incoming.id\_product, magazine\_incoming.quantity,

prices.price, magazine\_incoming.quantity\*prices.price AS summa

FROM incoming, magazine\_incoming, prices

WHERE magazine\_incoming.id\_product= prices.id\_product AND

magazine\_incoming.id\_incoming= incoming.id\_incoming//

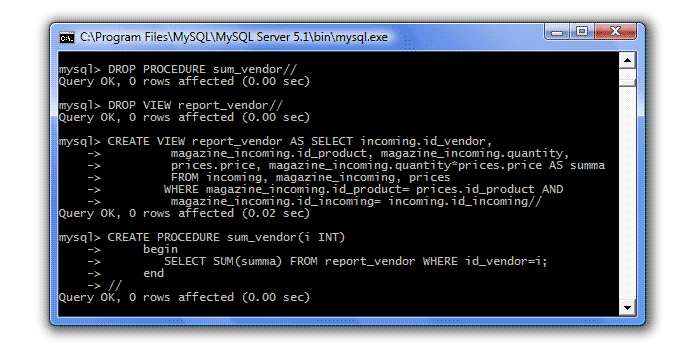
CREATE PROCEDURE sum\_vendor(i INT)

begin

SELECT SUM(summa) FROM report\_vendor WHERE id\_vendor=i;

end

//

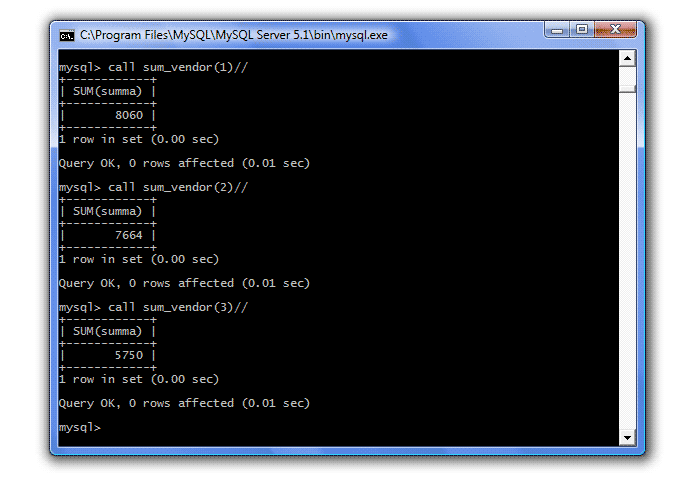


Проверяем работу:

call sum\_vendor(1)//

call sum\_vendor(2)//

call sum\_vendor(3)//

  
  
Второй вариант - прямо в процедуре дописать команду, которая будет удалять представление, если оно существует:

CREATE PROCEDURE sum\_vendor(i INT)

begin

DROP VIEW IF EXISTS report\_vendor;

CREATE VIEW report\_vendor AS SELECT incoming.id\_vendor,

magazine\_incoming.id\_product, magazine\_incoming.quantity,

prices.price, magazine\_incoming.quantity\*prices.price AS summa

FROM incoming, magazine\_incoming, prices

WHERE magazine\_incoming.id\_product= prices.id\_product AND

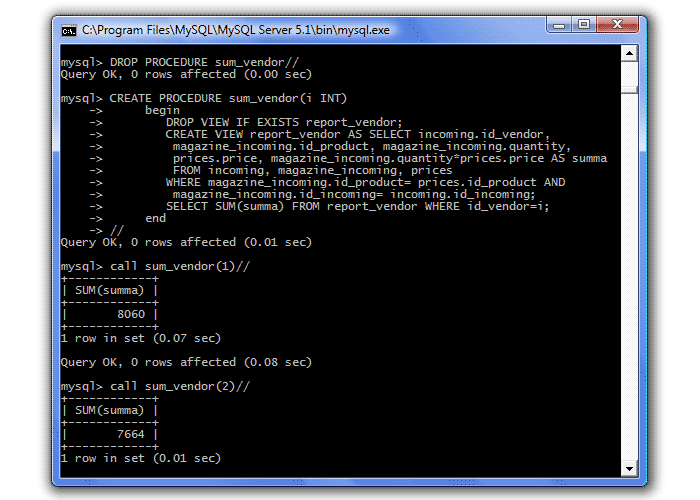
magazine\_incoming.id\_incoming= incoming.id\_incoming;

SELECT SUM(summa) FROM report\_vendor WHERE id\_vendor=i;

end

//

Перед использованием этого варианта не забудьте удалить процедуру sum\_vendor, а затем проверить работу:



Как видите, сложные запросы или их последовательность действительно проще один раз оформить в хранимую процедуру, а дальше просто обращаться к ней, указывая необходимые параметры. Это значительно сокращает код и делает работу с запросами более логичной.