МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФГБОУ ВО «УДМУРТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ИНСТИТУТ МАТЕМАТИКИ, ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ
И ФИЗИКИ
КАФЕДРА ТЕОРЕТИЧЕСКИХ ОСНОВ ИНФОРМАТИКИ

РАЗРАБОТКА WEB-ИНТЕРФЕЙСА К MYSQL В DRUPAL 7

ЧАСТЬ I

Учебно-методическое пособие

Ижевск, 2021

УДК 002.6:004.3(075.8)
ББК 32.972.53я73
Л696

Рекомендовано к изданию Учебно-методическим советом УдГУ

Рецензент: кандидат технических наук, доцент, заведующий кафедрой “Автоматизированные системы обработки информации и управления” Ижевского государственного технического университета имени М.Т. Калашникова Мокроусов М.Н.,

Составитель: старший преподаватель кафедры теоретических основ информатики ИМИТиФ Логов А.Г.,

Л696 Разработка web-интерфейса к MYSQL в DRUPAL 7. Часть I: учебно-методическое пособие. / сост. А.Г. Логов. – Ижевск: Издательский центр «Удмуртский университет», 2021. – 62 с.

В учебно-методическом пособии представлены материалы практических занятий проектирования баз данных. В первой части рассматривается знакомство с инструментальной средой phpMyAdmin, разработка форм сбора информации и создание модулей на CMS DRUPAL с доступом к MYSQL.

Рекомендуется студентам второго курса направления подготовки бакалавриата “Прикладная информатика”, а также студентам других направлений, изучающих основы баз данных.

УДК 002.6:004.3(075.8)
ББК 32.972.53я73

Содержание

Предисловие

В учебно-методическом пособие представлены части курса “Базы данных” для студентов второго курса направления подготовки “Прикладная информатика”.

Все программные продукты, используемые в материалах курса, доступны и бесплатны в сети Интернет. Устанавливать программное обеспечение на компьютер пользователя не нужно. Минимальные требования к компьютеру пользователя – наличие браузера Microsoft Edge. Не рекомендуется использовать Chrome от Google в разделе CMS DRUPAL 7, из-за кеширования страниц ядра сайта.

Для работы применяются онлайн инструменты: phpMyAdmin, MySqlServer, любой бесплатный виртуальный хостинг (можно использовать локальный сервер DENVER) и CMS DRUPAL 7.

Исходный программный код примеров постоянно дополняется новыми функциями. Номера строк начального программного кода, с привязкой к конкретным функциям, могут отличаться от промежуточного и завершенного.

Прежде чем приступить к изучению курса “Базы данных”, студент должен успешно пройти курсы “Языки и методы программирования” и “Объектно-ориентированное программирование”. Решение задач, представленных в пособии, улучшит понимание проектирования баз данных в системе управления контентом сайта (DRUPAL).

В результате изучения курса у студентов формируются некоторые общепрофессиональные и

универсальные компетенции, такие как способность проектировать физическую модель базы данных; умение работать в инструментальных средах; способность находить необходимую информацию и саморазвиваться на основе общеобразовательных принципов.

Тема 1. Введение в phpMyAdmin

Управление базами данных – одна из первоочередных задач при разработке информационных систем. Большинство приложений требует создание базы данных перед тем, как они будут установлены. Если используется сетевая база данных, размещенная на хостинге, то в нем уже будет реализован интерфейс для работы с ней.

PhpMyAdmin очень распространенный инструмент для управления базой данных MYSQL. В нем можно реализовать все, от этапа создания базы данных пользователя, резервного копирования, установки прав доступа и переноса базы данных на другой сервер.

После авторизации в phpMyAdmin мы попадаем в главное окно программы (рисунок 1). Оно состоит из трех рабочий частей: боковая панель, главное меню и рабочая область.

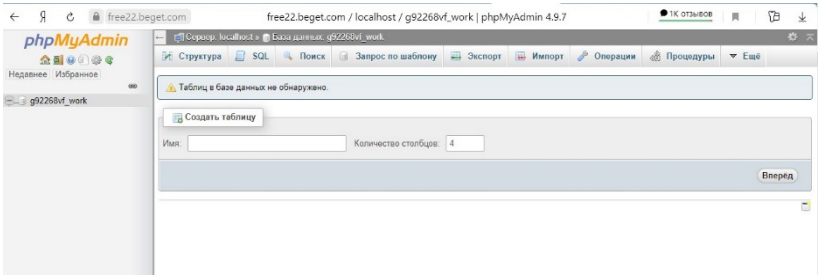


Рисунок 1. Главное окно phpMyAdmin

Боковая панель – располагается слева в виде вертикального столбца. Панель применяется для навигации в базе данных. В этой части, в виде списка, отображаются базы данных и таблицы. При выборе

таблицы, информация из нее отображается в рабочей области.

В самом вверху боковой панели вынесены кнопки возврата на главную страницу, выход из phpMyAdmin, справка, настройки и перезагрузка текущей страницы.

Главное меню – панель инструментов, выполненных в виде вкладок (структура, SQL, поиск, запрос по шаблону, экспорт, импорт, операции, процедуры, триггеры и дизайнер). В нашей работе мы постоянно будем использовать вкладки структура и SQL)

Рабочая область – выводит информацию по текущей действиям в базе данных. Например, структуру таблиц, форму для загрузки SQL-запросов, содержимое таблиц и т.д.

Создание таблиц

Как правило таблицы в базе данных создаются с помощью инструментальных средств (Erwin, Microsoft Visual Studio, Delphi и т.д). Тем не менее, часто для отладки, требуется создать таблицу и в ручном режиме.

Выбираем вкладку **“Структура**” (рисунок 2) в рабочей области.

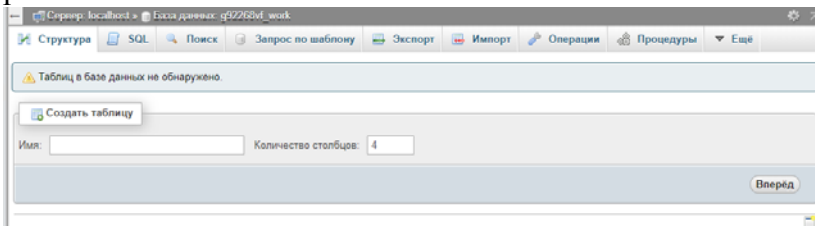


Рисунок 2. Создание таблицы

Для примера, рассмотрим реляционную модель базы данных, в которой есть таблица **customers** со следующими поля и записями в ней (рисунок 3).

id_cust	name	city	date	email
1	Bob	Denver	12.02.2020	bo@gm.com
2	Joe	Atlanta	18.10.2012	joe@gm.com
3	Martin	Boston	16.10.2015	ma@gm.com

Рисунок 3. Таблица customers

В таблице **customers** пять полей (**id_cust**, **name**, **city**, **date** и **email**). Особый разговор про поле **id_cust**. Любая таблица в реляционной базе данных должна иметь минимум одно ключевое поле (поле с уникальными значениями). С помощью такого поля происходит внесение новых записей в таблицу их разделение между собой по ключевым значениям. Также оно может быть внешним ключом в другой, связанной с ним таблице.

Возвращаемся к рисунку 2. В поле **имя** пишем название таблицы **customers.** Количество столбцов это число полей таблицы. Их у нас 5. Завершаем ввод данных кнопкой **Вперед** в правой нижней части рабочей области.

Следующий этап создания таблицы требует ввести названия полей и их тип данных (рисунок 4). Для ключевого поля **id_cust** пока будем использовать целочисленный тип данных (**int**). Позже будет рассказано как правильно задавать ключевые поля. Поля **name**, **city** и **email** создаем строкового типа (VARCHAR) длиной 20 символов. Для поля **date** соответствует тип дата (DATE).

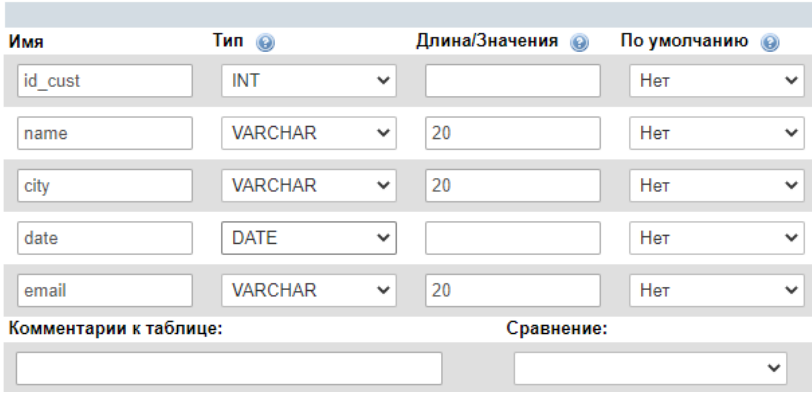


Рисунок 4. Завершение создание таблицы

Кнопкой **Сохранить**, в правой нижней части завершаем создание таблицы **customers**.

Итогом правильного завершения работы будет рисунок 5. Посмотрите в **боковую панель.** Там вы увидите название нашей таблицы.

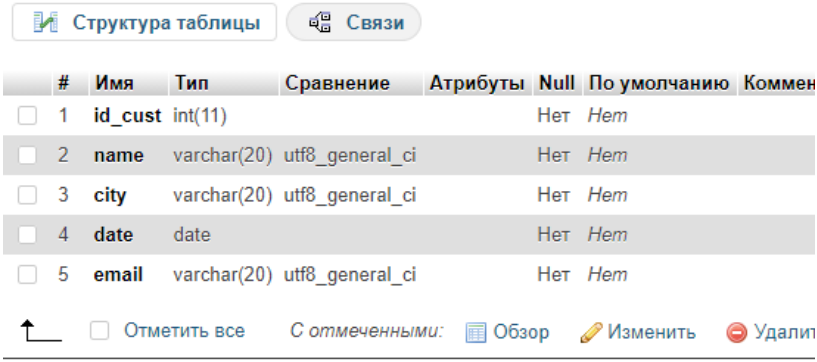


Рисунок 5. Таблица в вкладке “Структура”

Если у нас название полей написано не верно, указан не тот тип данных, порядок следования полей в таблице

другой, создали лишнее поле или пропустили, то все эти недочеты можно исправить в вкладке **Структура**.

Просмотр содержимого таблицы

Все довольно просто. В **боковой панели** нажимаем на таблицу (рисунок 6), содержимое которой хотим просмотреть. У нас пока только одна таблица **customers**.

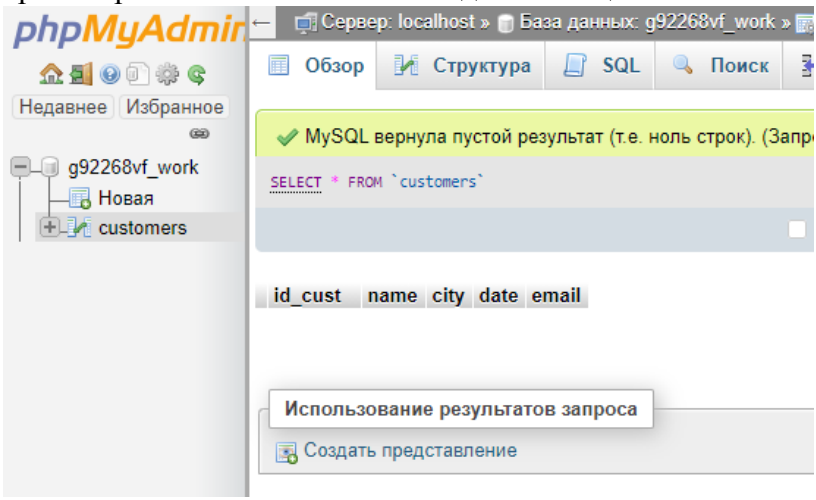


Рисунок 6. Содержимое таблицы customers

На экран будут выведены только названия полей таблицы, поскольку мы еще записи в таблицу не вносили.

Вставка записей в таблицу

На рисунке 3 в таблице **customers** три записи (информация про Bob, Joe и Martin)**.**

Открываем в **Главном меню** (рисунок 7) вкладку **Вставить**

Заполняем столбец **Значения** из таблицы информацией (рисунок 3) первой записи. Кнопкой **Вперед** подтверждаем ввод данных.

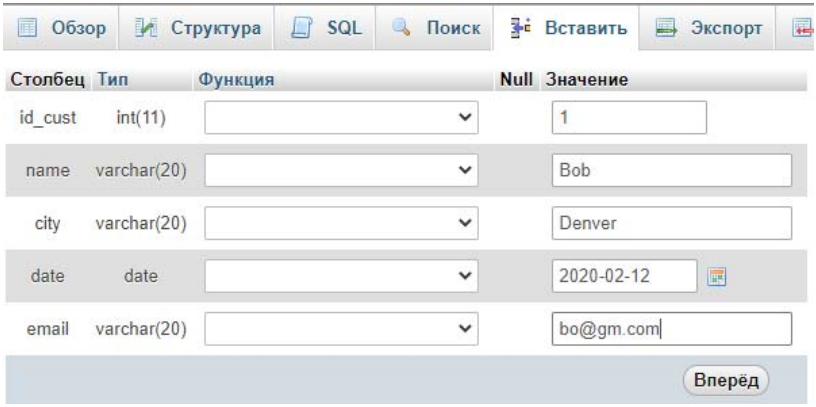


Рисунок 7. Вкладка добавления записей

PhpMyAdmin по нашим данным (рисунок 8) создаст SQL-запрос, выполнить его (смотрите на фразу **Добавлена 1 строка**).

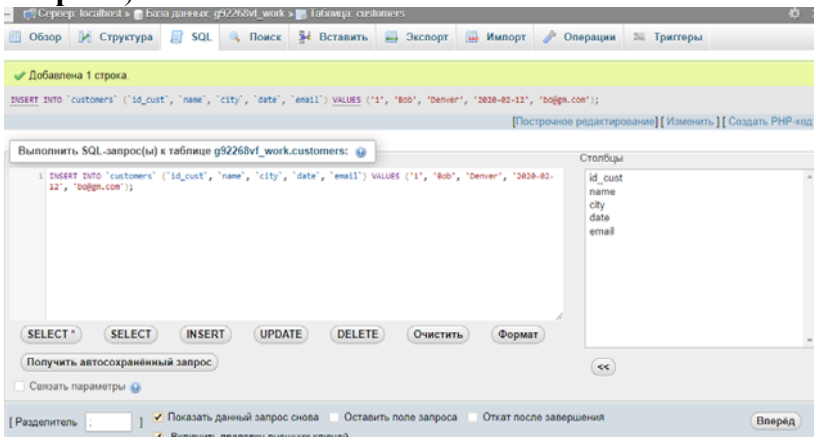


Рисунок 8. Добавление данных в таблицу

Второй раз нажимать на кнопку **Вперед** не нужно. Иначе у нас еще раз добавиться копия этой записи в таблицу. Можно исправить это SQL-запрос на новые

данные, но сейчас используйте вкладку **Вставить**, для внесения оставшихся двух записей таблицы.

Проверить внесения записей в таблицу производим через **Боковую панель** (рисунок 9), нажимая на таблицу **customers.**

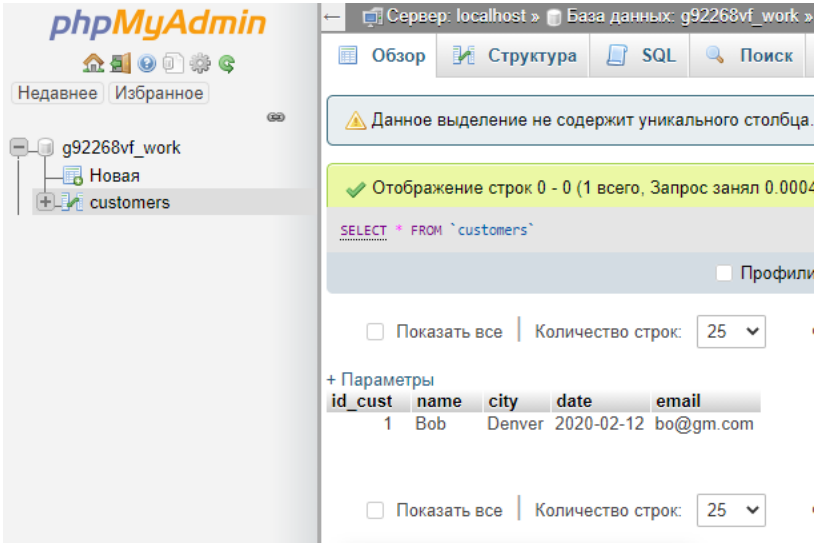


Рисунок 9. Просмотр содержимого таблицы customers

Если вы при вводе данных внесли не верную информацию или есть повторяющиеся строки, то отредактировать ее без SQL-запросов не получиться. У нас ключевое поле **id_cust** 12 имеет тип **INT,** а нужен **SERIAL**. Оставляйте все как загрузили. В следующей теме **SQL-запросы** разберем редактирование записей.

Удаление таблицы

Созданную таблицу можно удалить (рисунок 10) через вкладку **Структура.** Порядок действий следующий:

нажимаем в **Боковой панели** название базы данных (у меня **g92268vf_work**), ставим галочку на против таблицы **customers** и нажимаем на ссылку **Удалить.**

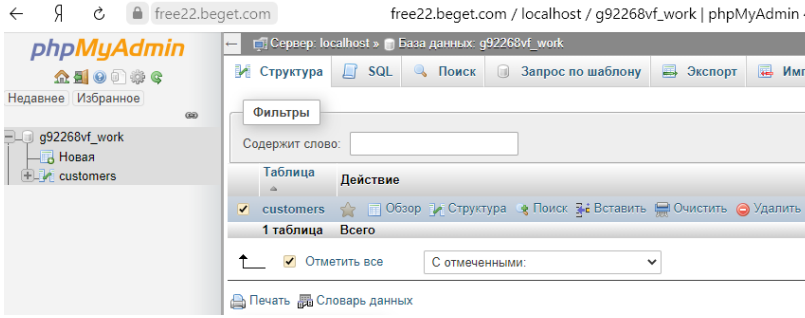


Рисунок 10. Удаление таблицы customers через вкладку “Структура”

Заключение

Тема номер один завершает изучение главных инструментов phpMyAdmin. Вкладки создания, редактирования, просмотра и удаления являются необходимыми для работы в любой базе данных.

Задача 1.1 (индивидуальная)

Создать и заполнить таблицу (6 записей) по следующим вариантам БД (базы данных).

Вариант 1. БД "Библиотека". Учёт востребованности изданий (название издания, количество читателей и дата последней выдачи).

Вариант 2. БД "Больница". Учёт поступления пациентов (фамилия, отделение, диагноз и дата).

Вариант 3. БД "Магазин". Оформление заказов на товары, запасы которых подходят к концу (Название товара, количество проданных товаров, производитель, наличие на складе).

Вариант 4. БД адвоката. Определение эффективности защиты (максимальный срок, минимальный срок, число оправданий, условных сроков и штрафов, дата окончания лицензии адвокатской деятельности).

Вариант 5. БД по продажам недвижимости. Показатели продаж (время нахождения помещения в продаже, тип помещения, стоимость аренды).

Вариант 6. БД "Гостиница". Номера гостиницы (этаж, общее количество мест, число занятых мест, класс номера, стоимость проживания в сутки, тип номера, тип размещения).

Вариант 7. БД "Продажа билетов". Список рейсов (Номер рейса, число билетов, число забронированных билетов, класс, дата вылета, время в пути).

Вариант 8. БД "Спортивный клуб". Списков спортсменов и тренеров (Фамилия, спортсмен или тренер, травмирование, количество соревнований, рейтинг и дата приема на работу).

Вариант 9. БД "Кафедра". Расписания занятий (семестр, название дисциплины, количество часов, номер группы, фамилия преподавателя, вид отчетности).

Вариант 10. БД "Плановый отдел". Составление заказов на поставку сырья и комплектующих (номер заказа, наименования изделия, количество на текущий момент, количество израсходованных, крайняя дата поступления, рекомендуемая дата поступления, фамилия ответственного за доставку).

Тема 2. SQL-запросы

Создадим таблицу **customers** и заполним ее информацией посредством SQL-запросов.

Удаление таблицы

Основное требование при создании новой таблицы, убедиться, что у нас нет старой информации в базе данных.

Если вы выполнили тему **Введение в phpMyAdmin**, то у Вас уже есть таблица **customers**. Повторное создание таблицы приведет к ошибке создания таблицы. Поэтому при проектировании приложений, пишут сценарий, который сначала удаляет старые таблицы (даже если таких данных нет, ничего плохого не случиться, команды просто не будут выполняться). А после команд удаления таблиц пишут команды создания таблиц.

Примечание. Можно сразу создавать новые таблицы. Но это потом слишком усложняет отладку приложений и их переустановку.

Переходим в вкладку **SQL** (рисунок 11).

Синтаксис SQL команды удаления таблицы следующий drop table **customers**;

Операторы **drop table** служебные слова, за ними пишем название таблицы для удаления.

В конце каждого SQL-запроса ставиться точка с запятой. Можно выполнять сразу несколько запросов.

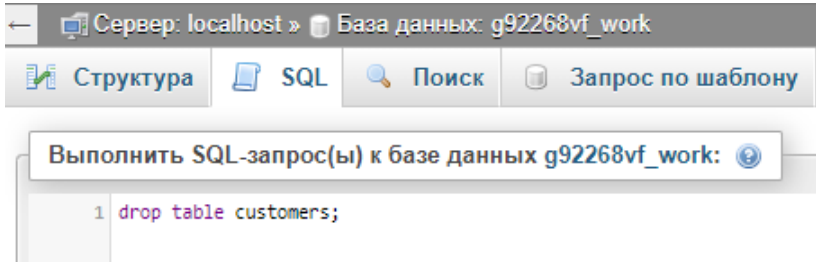


Рисунок 11. Удаление таблицы customers через SQL-запрос

Нажимаем кнопку **Вперед.** Подтверждаем запрос на удаление и видим (рисунок 12), что в **Боковой панели** таблица **customers** исчезла.

Кто выполнял задачи 1.1, аналогично удалите свои таблицы.

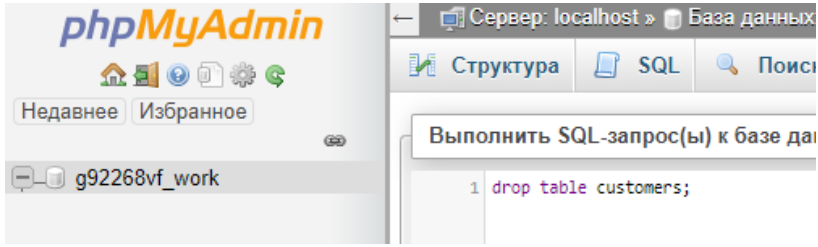


Рисунок 12. Удаление таблицы. Боковая панель

Создание таблицы

Команда создания таблицы (рисунок 13), чуть сложнее.

create table **customers** (**id_cust** integer, **name** varchar(20), **city** varchar(20), **date** date, **email** varchar(20);

Служебными словами (неизменяемая часть команды) будут create table. После указывается название создаваемой таблицы **customers.** В скобках перечисляются названия полей и их тип, разделяемые запятой.

В конце каждой SQL-команды ставиться точка с запятой.

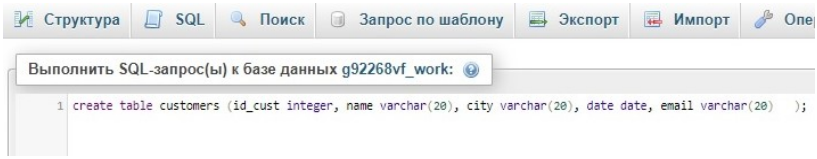


Рисунок 13. Создание таблицы customers

Выполнение команды создания таблицы в середине **Рабочей области** выводит сообщение (рисунок 14).

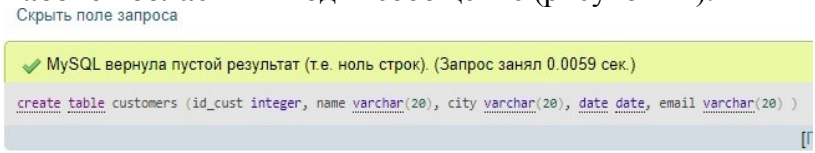


Рисунок 14. Результат команда создание таблицы customers

Фраза **MySQL вернула пустой результат,** не должна вас беспокоить. Это означает, что данный тип команды не возвращает данные, после выполнения. У нас среди SQL-команд этим занимается SQL-оператор **SELECT** (выборка данных из таблицы).

Убедитесь, что в **Боковой панели** появилась таблица **customers** (рисунок 6).

Вставка данных в таблицу

Синтаксис команды **insert** (рисунок 15).

insert into **customers** (**id_cust**, **name**, **city**, **date**, **email**) values (1, 'Bob', 'Denver', '2020-02-12', 'bo@gm.com').

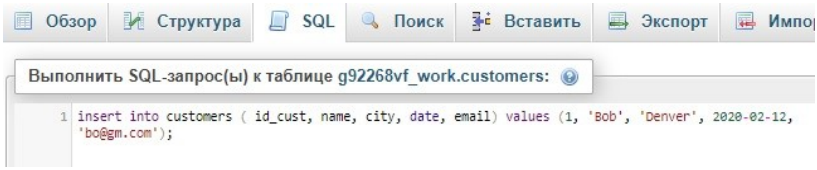


Рисунок 15. Вставка данных в таблицу customers

Если у вас возникает ошибка при выполнении запроса, то вероятнее всего это в поле **date**. Проверьте, что формат даты правильно вносите (год, день и месяц. Все зависит от локализации phpMyadmin). Вторая ошибка, что перед датой и после не поставили апостроф (символ **'** кнопка **Э**, **не путайте с символом на кнопке Ё**)

Удачное завершение у вас будет как на рисунке 9.

Обновление данных в таблице

Для обновления записи в таблице применяется команда update. Область влияния, на изменяемую информацию, определяется только условием. Можно отредактировать одной командой все записи в таблице или только одну конкретную (рисунок 15).

update **customers** set **city**='Texas' where **id_cust** =1;

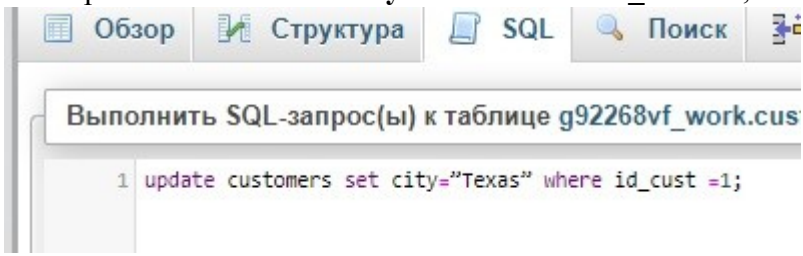


Рисунок 16. Обновление данных в таблице customers

На рисунке 16 команда заменяет текущее значение поля **city** на город **Texas**. Важной опцией является условие, записанное после служебного слова **where**. Изменения затронут только те записи, у которых **id_cust** равен единице. Поле **id_cust** ключевое. Обновление коснется только одной записи.

Можно условие не ставить. Например, записать так.

update **customers** set **city**='Texas'

Все записи в таблице **customers** в поле **city** будут хранить Texas. Пользы от команды **update** без условия мало.

Редактировать одновременно можно значения не только в одном поле. Следующая команда меняет значения полей **name** и **city**, для записей у которых **id_cust** находиться в диапазоне, больше либо равно единице и меньше трех.

update **customers** set **name**='Lucas', **city**='Texas' where **id_cust** >=1 and **id_cust** <3 ;

Можно использовать в условии логическую связку OR(или).

Удаление записей в таблице

SQL-команда удаления похожа по конструкции на **update** (рисунок 17).

delete from **customers** where **id_cust**=1;

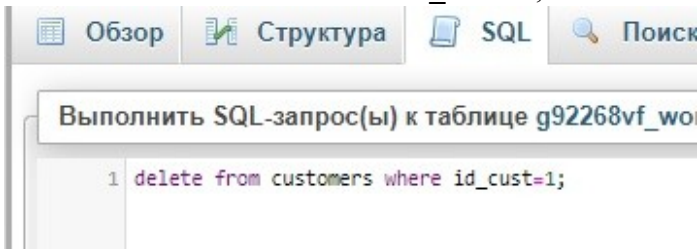


Рисунок 17. Удаление записей в таблице customers

Удаляется записи из таблицы, для которых истинны значения в условии. Можно удалять записи через значение любого поля. Обычно информацию удаляют через ключевые поля.

В этом примере исчезает запись со значением поля **id_cust** равным единице.

Выборка записей

Все предыдущие команды создавали таблицы, вносили данные, изменяли и удаляли записи. Самая важная SQL-команда выборки данных **select**. Она наиболее часто используется при работе с базами данных.

select * from **customers**;

Символ * обозначает, что все поля таблицы **customers** нужно вывести (рисунок 18).

+ Параметры

id_cust	name	city	date	email
1	Bob	Denver	2020-02-12	bo@gm.com
2	Joe	Atlanta	2012-10-18	joe@gm.com
3	Martin	Boston	2015-10-16	ma@gm.com

Рисунок 18. Выборка в таблице customers

Примечание: команда выборки не создает новых таблиц и не удаляет информацию в таблицах. Ее назначение, обратиться к таблице или таблицам базы данных и извлечь информацию из них.

Можно при выводе менять порядок следования полей и их количество (рисунок 19). Поля для вывода указываются через запятую после слова **select**.

select **name**, **email, city** from **customers**;

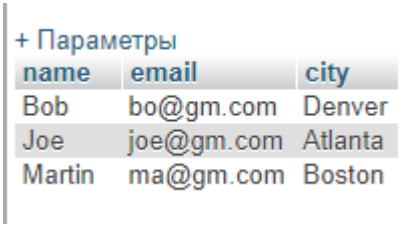


Рисунок 19. Выборка в таблице customers по полям name, email и city

Аналогично командам **update** и **delete** используется условие (рисунок 20). Ищем информацию в таблице, где поле **name** содержит имя **Joe**.

select **name**, **email, city** from **customers** where name='Joe';

Рисунок 20. Выборка в таблице customers по полям name, email и city по условию

Применение команды выборки не ограничивается одной таблицей. База данных информационной системы содержит множество таблиц, связанных между собой внешними ключами. Рассмотрим обращение и вывод информации по нескольким таблицам.

У нас есть в базе данных таблица покупателей **customers** (рисунок 21).

id_cust	name	city	date	email
1	Bob	Denver	12.02.2020	bo@gm.com
2	Joe	Atlanta	18.10.2012	joe@gm.com
3	Martin	Boston	16.10.2015	ma@gm.com

Рисунок 21. Таблица customers

Дополним базу еще тремя таблицами. Таблица с товарами **products** (рисунок 22). Ключевое поле будет **id_prod**. Поле **prod_name** хранит название товара, а поле **price** стоимость товара. Между собой таблица **customers** и **products** не связаны.

id_prod	prod_name	price
1	Monitor	120
2	Phone	50
3	Printer	175

Рисунок 22. Таблица products

Таблица **orders** будет отвечать за номера заказов, оформленные покупателями из таблицы **customers** (рисунок 23)**.** В поле **id_cust** находиться код покупателя. Поле **order_num** отвечает за номер заказа. Оба поля таблицы являются ключевыми. Поле **name**, кроме того, внешний ключ к таблице **customers**.

id_cust	order_num
1	1
1	3
2	5
3	2
3	4

Рисунок 23. Таблица orders

Товары, приобретенные покупателями, хранятся в таблице **purchase**. Ключевое поле **order_num.** Поле **quantity** количество товара. Поля **id_prod** внешний ключ к таблице **products,** а **order_num** к **orders.**

order_num	id_prod	quantity
1	1	2
2	3	1
3	1	3
4	2	1
5	1	4

Рисунок 24. Таблица purchase

Команды создания и заполнения таблиц будут следующие.

create table **products** (id_prod integer, prod_name varchar(20), price varchar(20);

insert into **product**s (id_prod, prod_name, price) values (1, 'Monitor', '120');

create table **orders** (id_cust integer, order_num integer);

insert into **orders** (id_cust, order_num) values (1, 1);

create table **purchase** (order_num integer, id_prod integer, quantity integer);

insert into **purchase** (order_num, id_prod, quantity) values (1, 1, 2);

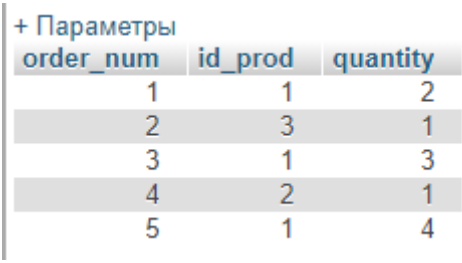
Указаны команда создания таблиц и одна команда вставки. Остальные записи, по аналогии, не сложно написать по содержимому таблиц (рисунок 22, 23 и 24)

После создания таблиц приступаем к написанию сложного запроса выборки по нескольким таблицам.

Выборка записей по нескольким таблицам

Выведем таблицу покупок **purchase** (рисунок 25).

select **order_num, id_prod, quantity** from **purchase**;



+ Параметры

order_num	id_prod	quantity
1	1	2
2	3	1
3	1	3
4	2	1
5	1	4

Рисунок 25. Содержимое таблица purchase

Мы видим, поле **id_prod** хранит код продукта. Как сделать, чтобы вместо кода, отображалось название продукта, соответствующее этому коду. Расшифровка кода находиться в таблице **products**. Обратимся к двум таблицам **purchase** и **products** (рисунок 26).

select purchase**.order_num,** products.prod_name**,** purchase**.quantity** from **purchase, products**;

В запросе к двум и более таблицам нужно указывать, не только названия полей, но и названия таблиц которым они принадлежат. У нас это три поля **purchase.order_num, products.prod_name** и **purchase.quantity.** Сначала идет название таблицы потом поле (между ними ставиться точка). Поле **id_prod** не выводим. Оно нам нужно будет в условии.

Команда запроса написана правильно, но появилось много не верных записей. Изначально было пять покупок, а стало гораздо больше. Это произошло из-за того, что не указаны связи между этими таблицам **purchase** и **products**.

+ Параметры

order_num	prod_name	quantity
1	Monitor	2
1	Phone	2
1	Printer	2
2	Monitor	1
2	Phone	1
2	Printer	1
3	Monitor	3
3	Phone	3
3	Printer	3
4	Monitor	1
4	Phone	1
4	Printer	1
5	Monitor	4
5	Phone	4
5	Printer	4

Рисунок 26. Содержимое таблиц purchase и products

База данных нам вернула всевозможные комбинации трех запрашиваемых полей (**order_num, prod_name** и **quantity**).

Исправляем ошибку. Добавляем в запрос условия связывания таблиц между собой. Таблицы **purchase** и **products** связаны ключевыми полями **id_prod.**

select purchase**.**order_num**,** products.prod_name**,** purchase**.**quantity from purchase, products **where purchase. id_prod= products.id_prod;**

Дополненный условием запрос вернул правильное число записей покупок (их пять). Цель замены кода товара на его название достигнута (рисунок 27).

+ Параметры

order_num	prod_name	quantity
1	Monitor	2
2	Printer	1
3	Monitor	3
4	Phone	1
5	Monitor	4

Рисунок 27. Содержимое таблиц purchase и products с условием связывания ключевых полей

Выведем всю информацию из базы данных. У нас есть таблицы **orders** и **customers.** Ничего нового из **orders** мы не узнаем (кроме кода клиента **id_cust**). Зато, мы через поле **id_cust** таблицы **orders** можем связаться с таблицей **customers** и вывести информацию о покупателях.

Исходное наше условие связывающие таблицы **purchase** и **products** было

where purchase.id_prod= products.id_prod;

Дополняем его связью между **purchase** и **orders** по полю **order_num.**

where purchase.id_prod= products.id_prod **and purchase.order_num= orders.order_num**;

И добавляем завершающую связь **orders** и **customers** по полю **id_cust**

where purchase.id_prod= products.id_prod and purchase.order_num=orders.order_num **and orders.id_cust= customers.id_cust**;

Собираем весь запрос **select** полностью.

Выберем поля для вывода из таблицы **customers (customers.name, customers.city, customers.date, customers.email)**.

Из таблицы **orders** поля для вывода не используем. Но обратите внимание, она нам нужна в запросе **select** для установки связи.

Из таблиц **purchase** и **products** оставляем те поля, что брали изначально (**purchase.order_num, products.prod_name, purchase.quantity**).

Не забудьте указать после служебного слова **from** в **select** названия используемых таблиц (**purchase**, **products**, **orders** и **customers**).

select **customers.name, customers.city, customers.date, customers.email,** purchase**.**order_num**,** products.prod_name, purchase**.**quantity from purchase, products, **orders**, **customers** where purchase.id_prod= products.id_prod
and purchase.order_num=orders.order_num
and orders.id_cust= customers.id_cust;

+ Параметры

name ▲ 1	city	date	email	order_num	prod_name	quantity
Bob	Denver	2020-02-12	bo@gm.com	1	Monitor	2
Bob	Denver	2020-02-12	bo@gm.com	3	Monitor	3
Joe	Atlanta	2012-10-18	joe@gm.com	5	Monitor	4
Martin	Boston	2015-10-16	ma@gm.com	4	Phone	1
Martin	Boston	2015-10-16	ma@gm.com	2	Printer	1

Рисунок 28. Вывод информации по таблицам purchase, products, orders и customers

На рисунке 28 показана все информация (без ключевых полей) которая содержится в базе данных. Обратите внимание, как было пять записей покупок, так их столько же осталось. Если число записей увеличивается, это означает, что у вас не правильно были связаны таблицы в условии.

Можно сортировать выводимые данные в алфавитном порядке, возрастанию или убыванию. Добавьте в конец запроса опцию **order by customers.date** и записи будут отсортированы по дате.

select customers.name, customers.city, customers.date, customers.email, purchase.order_num, products.prod_name, purchase.quantity from purchase, products, orders, customers where purchase.id_prod= products.id_prod
and purchase.order_num=orders.order_num
and orders.id_cust= customers.id_cust **order by customers.date;**

Для сортировки можно использовать любое поле или одновременно несколько полей.

Заключение

В теме 2 рассмотрены основные SQL-команды. Существуют еще другие. Мы их рассмотрим позже, по мере углубленного изучения материала.

Задача 2.1 (индивидуальная)

1. По выбранному варианту темы 1, спроектировать минимум еще три таблицы, дополняющих вашу базу данных (не забывайте создавать внешние ключи, для связи между таблицами).

2. Заполнить таблицы своими значениями (минимум пять записей).

3. К базе данных написать два запроса выборки(**select**). Первый запрос **select** обращается к двум любым таблицам, имеющим связи между собой. Во втором запросе вывести всю информацию из таблиц базы данных.

Тема 3. Создание форм в CMS DRUPAL 7

DRUPAL – это одна из известных систем управления контентом с открытым исходным кодом. С помощью его функционала создаются сложные информационные системы в виде сайтов. Хранение информации реализовано в базе данных MYSQL. Написать интерфейсную часть к базе данных в DRUPAL не сложно.

В это части рассмотрим создание форм сбора информации к базе данных MYSQL. Создание таблиц базы данных и SQL-запросы разберем в четвертой теме.

При проектировании форм нам нужно авторизоваться в DRUPAL (рисунок 29) под учетной записью администратора).

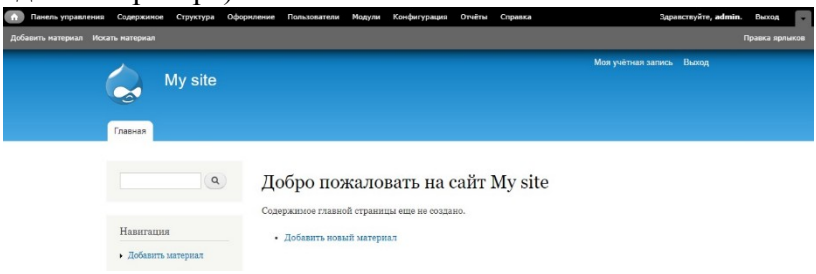


Рисунок 29. Главная страница администратора сайта

В панели инструментов заходим в раздел **Модули.**

Рекомендуется работать пустым ядром CMS DRUPAL. Т.е у вас должен быть только один модуль – **Ядро** (рисунок 30). В DRUPAL есть некоторые недоработки программного обеспечения, посторонние модули могут вызывать ошибки в вашей разработке.

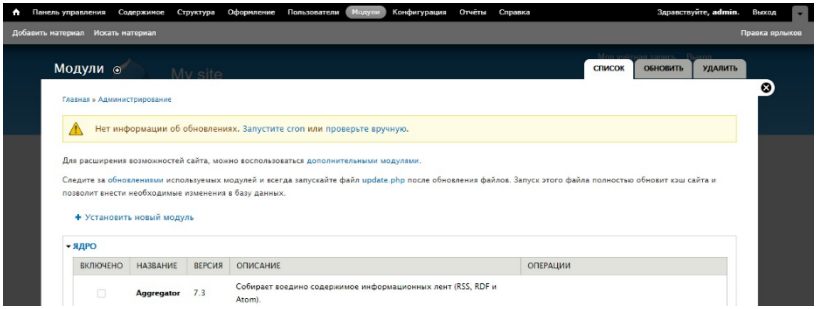


Рисунок 30. Содержимое раздела модули

.

Раздел **Модули** используется для активации, редактирования или удаления модуля. Сюда мы скоро вернемся.

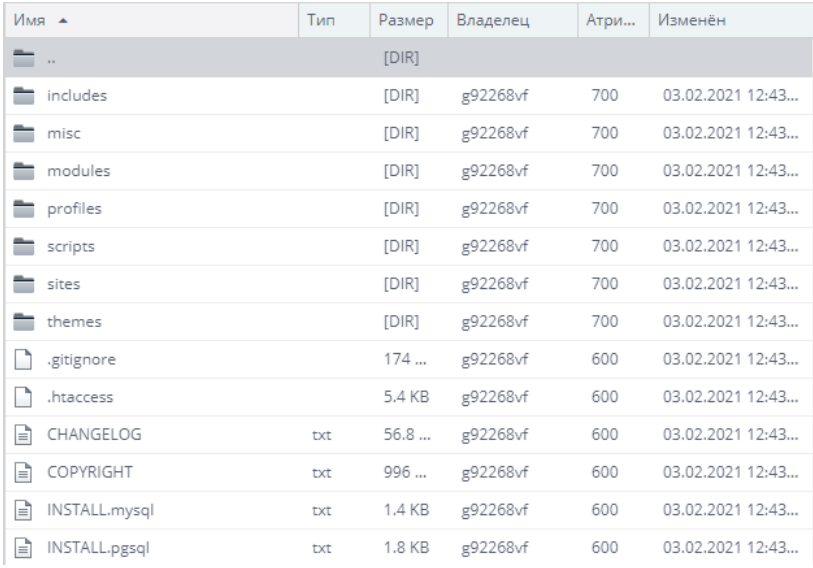
Рассмотрим (рисунок 31) структуру файлов и каталогов DRUPAL. Обычно в хостинге по умолчанию есть папка **public_html**. В ней размешается информация доступная для пользователей сети интернет. После завершения установки DRUPAL будут созданы следующие разделы (**includes, misc, modules, pfofiles, script, sites и themes**).

Скаченные и разрабатываемы модули хранятся по следующему пути: **/sites/all/modules**

Создадим модуль, собирающий информацию для таблицы **customers**, (рисунок 21) из прошлых тем.

На первый этапе разработки в папке **/sites/all/modules** создаем каталог с названием нашего модуля. Название можно дать любое, но не пересекающиеся с уже с существующими модулями. К примеру **customers** (рисунок 32).

Название модуля выбирайте удобным для прочтения и написания. Оно будут постоянно использоваться в работе.



Имя	Тип	Размер	Владелец	Атри...	Изменён
..		[DIR]			
includes		[DIR]	g92268vf	700	03.02.2021 12:43...
misc		[DIR]	g92268vf	700	03.02.2021 12:43...
modules		[DIR]	g92268vf	700	03.02.2021 12:43...
profiles		[DIR]	g92268vf	700	03.02.2021 12:43...
scripts		[DIR]	g92268vf	700	03.02.2021 12:43...
sites		[DIR]	g92268vf	700	03.02.2021 12:43...
themes		[DIR]	g92268vf	700	03.02.2021 12:43...
.gitignore		174 ...	g92268vf	600	03.02.2021 12:43...
.htaccess		5.4 KB	g92268vf	600	03.02.2021 12:43...
CHANGELOG	txt	56.8 ...	g92268vf	600	03.02.2021 12:43...
COPYRIGHT	txt	996 ...	g92268vf	600	03.02.2021 12:43...
INSTALL.mysql	txt	1.4 KB	g92268vf	600	03.02.2021 12:43...
INSTALL.pgsql	txt	1.8 KB	g92268vf	600	03.02.2021 12:43...

Рисунок 31. Содержимое корневого каталога DRUPAL

Рисунок 32. Содержимое каталога modules

В созданном каталоге **customers** создадим два файла **customers.info** и **customers.module.**

Файл **customers.info** устроен просто.

```
1  name = customers
2  description = This module creates a form for customers.
3  core = 7.x
```

Первая строка объявляет названия модуля. Вторая краткое описание. Последняя содержит версию ядра DRUPAL. Больше в работе нам этот файл не понадобиться.

Второй файл **customers.module** описывает содержимого нашего модуля.

```
1  <?php
2
3  function customers_menu() {
4    $items['customers/form1'] = array(
5          'type' => MENU_CALLBACK,
6          'access arguments' => array('доступ к контенту'),
7          'page callback' => 'drupal_get_form',
8          'page arguments'=>array('customers_form1'));
9
10   return $items;
11 }
```

Ядро CMS DRUPAL создано на языке программирования PHP. Первая файла с расширением **module** должна начинаться тэга **<?php,** обозначающей начало программного кода языка PHP. В конце файла тег можно не закрывать.

Почти весь модуль **customers** будет состоять из **hook** (аналог методов в объектно-ориентированном программировании). Наиболее часто используемые методы уже созданы в ядру DRUPAL. Можно создавать свои.

Строки с 3-11 описывают **hook_menu.** Вызов любого hook (строка 3) происходит через название нашего модуля и название hook, разделенных нижним подчеркиванием (в нашем примере **customers_menu**). В начале (строка 4) hook, задаются параметры (каждого хука они свои). В конце hook возвращает значение (строка 10).

В строке 4 определяется путь к нашей, создаваемой форме сбора информации для таблицы **customers.** Чтобы увидеть форму в браузере, нужно после имени домена сайта дописать путь **/customers/form1**. Вместо **form1** можно написать другое имя. Строка 5 определяет тип меню. Строка 6 тип доступа к информации. Строка 7 обработка введенной информации в нашей форме (подробнее эти параметры разберем позже). Строка 8 для нас сейчас наиболее важная. Именно она вызывает еще не созданную форму сбора данных **customers_form1** (реализуется hook_form1). Информации двух файлов **customers.info** и **customers.module** уже достаточно, для активации модуля.

Возвращаемся на страницу **Модули** (рисунок 30) и активируем появившийся модуль **customers** (рисунок 33).

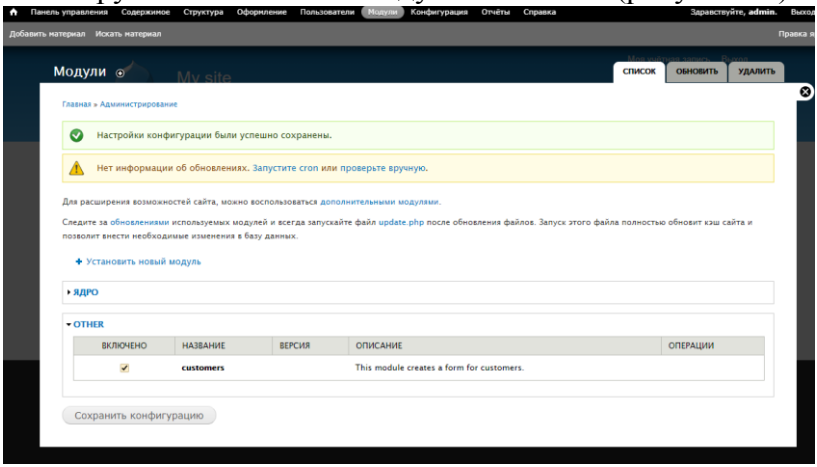


Рисунок 33. Активация модуля customers

Убедитесь, что не возникло ошибок при активации. Должна появиться фраза **Настройки конфигурации были успешно сохранены** (рисунок 33 верхняя часть).

Создадим hook с формой (рисунок 34). Допишем ниже **customers_menu,** новый hook **customers_form1** (строка 13). В 14 строке инициируется массив, содержащий поля формы. Он будет возвращен из hook в строке 22. Строка 16 задает имя поля формы (**name**). Рекомендуется давать такие название полей, как в таблице базы данных, чтоб не запутаться. Аргумент **textfield** (рисунок 35). В строке 17, создает поле ввода текста (существуют **select, radios**, **checkboxes**, **data, submit** и другие).

```
1  <?php
2
3  function customers_menu() {
4    $items['customers/form1'] = array(
5          'type' => MENU_CALLBACK,
6          'access arguments' => array('доступ к контенту'),
7          'page callback' => 'drupal_get_form',
8          'page arguments'=>array('customers_form1'));
9
10   return $items;
11 }
12
13 function customers_form1() {
14     $form = array();
15
16     $form['name']=array(
17         '#type'=>'textfield',
18         '#title'=>t('Имя'),
19         '#description'=>t('Ввведите имя клиента')
20       );
21
22     return $form;
23 }
24
```

Рисунок 34. Поле для сбора имени клиента

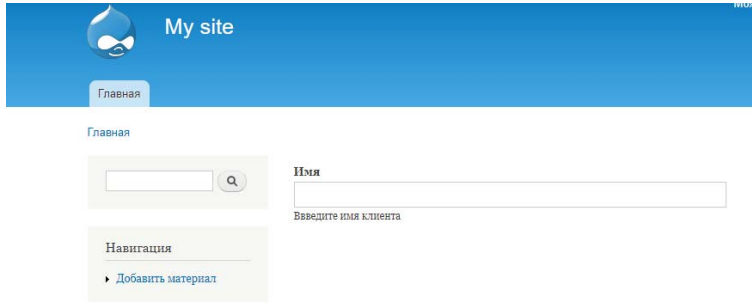


Рисунок 35. Поле textfield в форме

Открывается форма нашего модуля (рисунок 35), как описывалось ранее, добавлением к домену сайта пути **/customers/form1.** Например, ваш сайт имеет домен **mysite.com,** то адрес формы **mysite.com/customers/form1**

Дополним с 21 первой строки в **customers_form1**, оставшиеся поля таблицы **customers (city, date и email).** Поле **id_cust** в форме вводить не будем. Оно ключевое, создаваться будет через базу данных.

Обновим страницу формы (рисунок 36). Все поля у формы выведены. Строки 36-39 создают кнопку отправки формы.

В форму можно вводить данные и отравлять. Но для их получения и обработки нужны hook **validate** и **submit (customers_form1_validate** и **customers_ form1_submit).**

```php
function customers_form1() {
    $form = array();

    $form['name']=array(
        '#type'=>'textfield',
        '#title'=>t('Имя'),
        '#description'=>t('Ввведите имя клиента')
    );
    $form['city']=array(
        '#type'=>'textfield',
        '#title'=>t('Город'),
        '#description'=>t('Ввведите город клиента')
    );
    $form['date']=array(
        '#type'=>'date',
        '#title'=>t('Дата'),
        '#description'=>t('Ввведите дату')
    );
    $form['email']=array(
        '#type'=>'textfield',
        '#title'=>t('Электронный адрес'),
        '#description'=>t('Ввведите email')
    );
    $form['submit']=array(
        '#type'=>'submit',
        '#value'=>t('Отправить')
    );
    return $form;
}
```

Рисунок 36(а). Полный список полей в форме

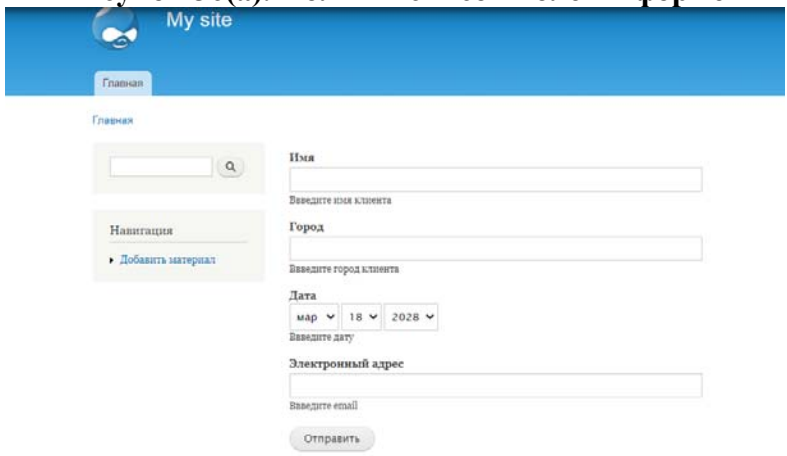


Рисунок 36(б). Форма с полями для таблицы customers

Проверку форму на корректность заполнения полей выполним через hook **validate** (строки 43-48)**.** Название **validate** дописываем к названию обрабатываемой формы.

```
43  function customers_form1_validate($form, $form_state) {
44
45    if(empty($form_state['values']['name'])){
46       form_set_error('name','Не заполненно имя клиента');
47    }
48  }
```

В модуле можно описать несколько форм и проверять каждую.

В строке 45 написано условие, что значение поля **name** (\$form_state[‘values’][‘name’]) вернуло пустой результат (**empty**).

Строка 46 вызывает функцию вывода сообщения об ошибке (form_set_error) с двумя параметрами (рисунок 37). Первый параметр(**name**) выделяет красной рамкой поле в форме, второй выводит сообщение об ошибке над формой.

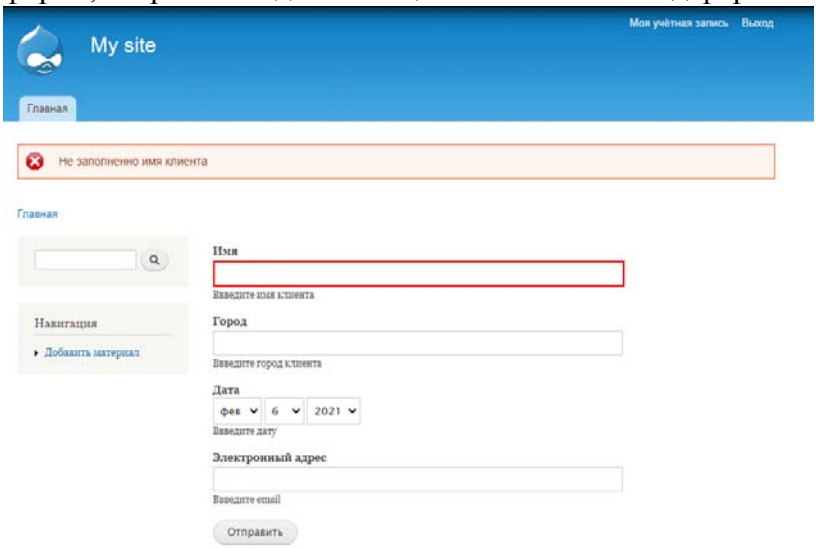


Рисунок 37. Обработка формы с помощью validate

Для проверки корректности введения электронного адреса можно включить фильтр (**filter_var**) в условие (строка 49) с параметром **FILTER_VALIDATE_VAR_EMAIL.**

```
43  function customers_form1_validate($form, $form_state) {
44
45    if(empty($form_state['values']['name'])){
46       form_set_error('name','Не заполненно имя клиента');
47    }
48
49    if(filter_var($form_state['values']['email'], FILTER_VALIDATE_EMAIL) == false) {
50      form_set_error('email','Не корректный email');
51    }
52
53  }
```

Результат применения фильтра показан на рисунке 38.

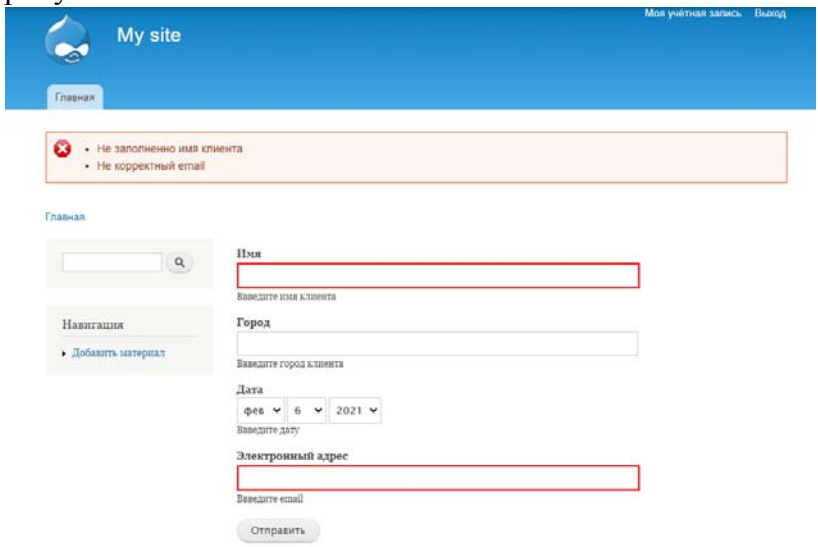


Рисунок 38. Обработка формы фильтром

Оставшиеся поля формы обрабатываются аналогично.

Завершаем работу над модулем (строки 53-55).

```
53  function customers_form1_form1_submit($form, $form_state) {
54      drupal_set_message("Форма была отправлена");
55  }
56
```

Вызывается оправка формы, при правильном заполнении полей. На данном этапе заносить данные из формы в базу данных не будем. Разберем это в следующей главе. В текущем модуле никаких изменений не произойдет. Откроется снова форма (рисунок 36).

Заключение

В теме 3 изучили основы создания модулей в DRUPAL. Создали форму сбора информации и ее обработки.

Задача 3.1 (индивидуальная)

По выбранному варианту темы 1, создать модуль с формой для сбора информации. Обработать все форма формы на корректность заполнения.

Тема 4. Разработка базы данных в CMS DRUPAL 7 и web-интерфейса

В предыдущих частях, мы рассматривали по отдельности базу данных и модуль сбора информации. Создадим небольшую информационную систему для сбора, хранения и обработки информации в базе данных MYSQL через web-интерфейс DRUPAL.

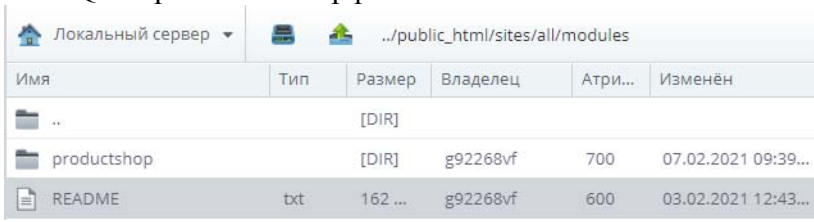


Рисунок 39. Коренной каталог модуля producshop

Разработка модуля с использованием базы данных начинается как прошлой главе. В папке ядра сайта **/sites/all/modules** (рисунок 39) создаем каталог с именем модуля **productshop.** Внутри каталога **producthop** создаем файл описания модуля **producshop.info**.

```
1  name = Processing of goods and orders
2  description = Processing of goods and orders
3  package = «Products shop»
4  core = 7.x
5  version = «7.x-1.0»
6  configure = admin/config/content/productshop
```

Конфигурацию модуля через **producshop.info** распишем подробнее. С командами первой, второй и четвертой строки уже сталкивались (название, описание и версия ядра модуля).

Строка номер 3 создает категорию в разделе инструментов **Модули** DRUPAL. В прошлой работе модуль относился к категории **OTHER** (рисунок 33).

Можно указать версию разрабатываемого модуля (строка 5). Доступ к страницам модуля (строка 6), реализуем через ссылку **Настроить** раздела **Модули,** в столбце **Операции** (рисунок 40)**.**

Работа модуля описывается файлом **productshop.module.**

```
1  <?php
2  function productshop_block_info()
3  {
4      $blocks['productshop'] = array(
5          'info'  => t('Товары и клиенты'),
6          'cache' => DRUPAL_CACHE_PER_ROLE, // по умолчанию
7      );
8      return $blocks;
9  }
10
11 function productshop_menu()
12 {
13     $items = array();
14     $items['admin/config/content/productshop'] = array(
15         'title'            => 'База данных клиентов и товаров',
16         'description'      => 'Создание и редактрование товаров и клиентов ',
17         'page callback'    => 'customers_list',
18         'access arguments' => array('administer site configuration'),
19     );
20
21     return $items;
22 }
```

Строки 2-9 описывает используемые блоки.

Структура модуля будет сложнее прошлого примера. В модуле создается блок с названием (строка 4) **productshop.** В **с**троке 5 задаем краткую информацию о блоге. Строка 6 описывает правило кеширования страниц сайта.

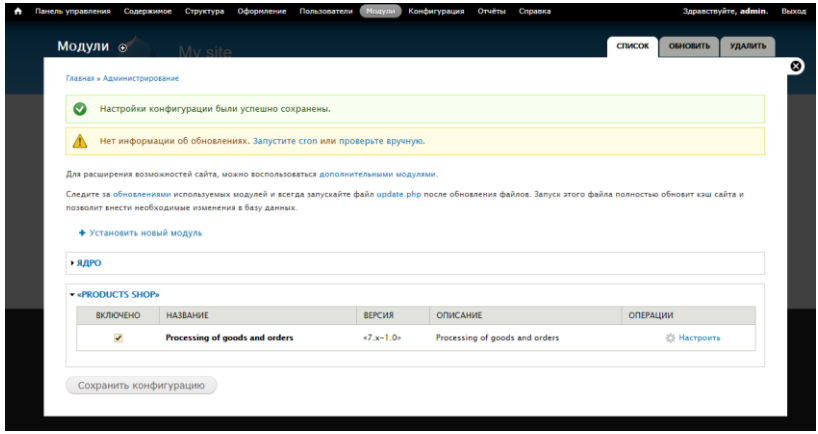


Рисунок 40. Активация модуля producshop

Реализуем более удобный доступ к страницам модуля. Все структура страниц модуля описывается в **productshop_menu** (строка 11). Вынесем ссылку на модуль в **Конфигураци** (рисунок 41), раздел **Работа с содержимым**. Строка 15 задает ссылку на модуль. Краткое описание в строке16. В строке 17 определяем, создаваемую нами функцию (**customers_list**), вывода содержимого страницы.

В 24 строке название функции вывода информации (**customers_list**). Объявляем два кеш-массива **$header** и **$row** вывода заголовка таблицы **customers** (рисунок 18).

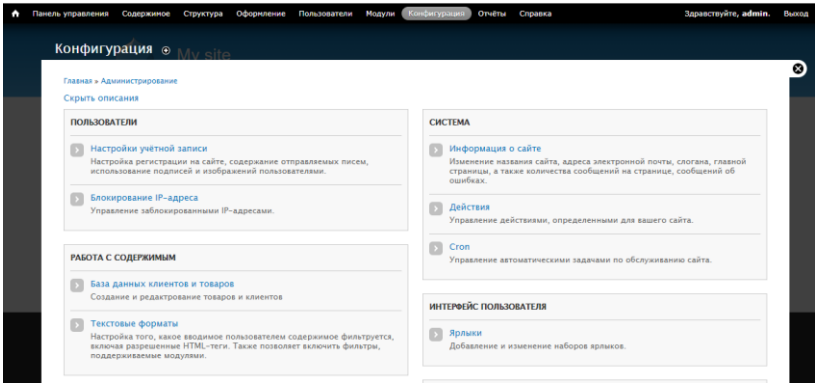


Рисунок 41. Ссылка к модулю в конфигурации

Базу данных с таблицами мы еще не создали, но вывод уже можем посмотреть. Строки 27-28 – это заголовки столбцов. Строка 33 создает по столбцам (рисунок 42) HTML-таблицу. В **Конфигурации** ссылка **База данных клиентов и товаров** уже активна.

```
11  function productshop_menu()
12  {
13      $items = array();
14      $items['admin/config/content/productshop'] = array(
15          'title'            => 'База данных клиентов и товаров',
16          'description'      => 'Создание и редактрование товаров и клиентов ',
17          'page callback'    => 'customers_list',
18          'access arguments' => array('administer site configuration'),
19      );
20
21      return $items;
22  }
23
24  function customers_list()
25  {
26      $header = array(
27          array('data' => t('Имя')),
28          array('data' => t('Город')),
29          array('data' => t('Дата')),
30          array('data' => t('Email'))
31      );
32      $row="";
33      return theme('table', array(
34          'header' => $header,
35          'rows'   => $row,
36      ));
37  }
```

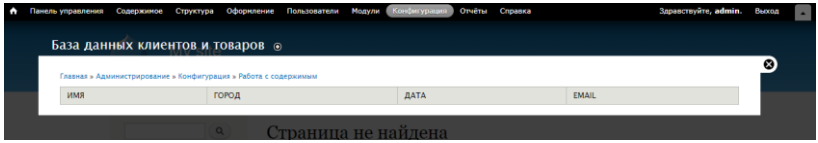


Рисунок 42. Вывод заголовков полей таблицы

Создадим в базе данных таблицу **customers.** Инициализация таблиц происходит в отдельном файле с расширением **install.**

Содержимое файла **productshop.install**

```
1  <?php
2  function productshop_uninstall()
3  {
4      cache_clear_all('productshop', 'cache', TRUE);
5      drupal_uninstall_schema('customers');
6      menu_rebuild();
7  }
8
9  function productshop_schema()
10 {
11     $schema['customers'] = array(
12         'fields'      => array(
13             'id_cust'     => array('type' => 'serial',  'size'=>'normal',  'not null' => TRUE),
14             'name'        => array('type' => 'varchar', 'length' => 255, 'not null' => TRUE),
15             'city'        => array('type' => 'varchar', 'length' => 255, 'not null' => TRUE),
16             'date'        => array('mysql_type' => 'datetime', 'not null' => TRUE),
17             'email'       => array('type' => 'varchar', 'length' => 255, 'not null' => TRUE),
18
19         ),
20         'primary key' => array('id_cust')
21     );
22
23     return $schema;
24 }
```

В файле применяются два hook **uninstall** и **schema.** В начале мы удаляем всю старую информацию в модуле **productshop** (строка 4). В строке 5 удаляем таблицу **customers.** Если, в ядре сайта модуль не был установлен, ничего происходит. В строках с 11 по 23 написан sql-запрос **create table** в синтаксисе DRUPAL. Объявляются имена полей и их тип.

Примечание. Обратите внимание на строки 13 и 20. Создается ключевое поле **id_cust**. В начале ему указывается тип **serial (**автоматически увеличивается номер следующей записи на единицу). Строка 20 фиксирует его ключевым полем.

После создания файла **productshop.install** в разделе **Модули** деактивируем модуль **customers** и снова активируем (рисунок 40).

Заходим в **phpMyAdmin,** ищем в нем таблицу **customers** (перед названием таблицы будет префикс DRUPAL). В этом пример DRUPAL дал имя **dpcustomers.** Префикс может быть и другим. В разделе **Структура** проверяем тип полей нашей таблицы (рисунок 43) **customers**.

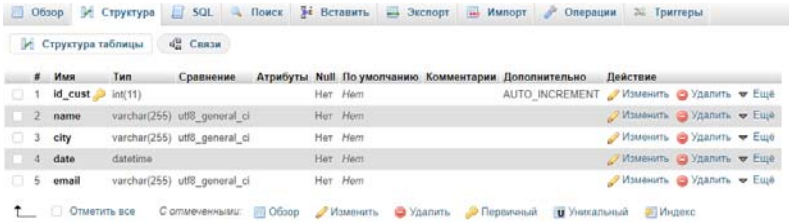


Рисунок 43. Структура таблицы customers

В поле **id_cust** имеется значек ключа и служебное слово AUTO_INCREMENT. Таблица создана корректно, можно с ней дальше работать.

Если таблица не создалась, то необходимо переустановить модуль **customers**.

Переустановка модуля customers

Файл **productshop.install** выполняется в целях безопасности, только один раз. Любые изменения в нем, после активации не вызываются.

В начале заходим в раздел **Модули** (рисунок 40) и убираем галочку перед модулем и сохраняем конфигурацию.

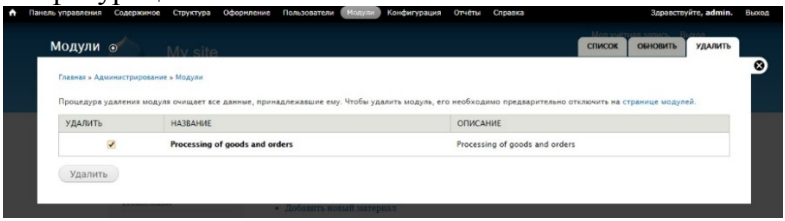


Рисунок 44. Вкладка удаления модуля

В этом же разделе выбираем вкладку (рисунок 44) **Удалить.** Выбираем наш модуль и удаляем. Копируем папку **customers** (рисунок 39) из каталога **/sites/all/modules** в любое другое место.

Удаляем папку **customers** в каталоге **/sites/all/modules.**

Чистим **кеш** (рисунок 45) в ядре сайта (**Конфигурация -> Производительнось)**

Копируем обратно в каталог **/sites/all/**modules наш модуль **customers.**

Повторяем шаги активации модуля (рисунок 39 и шаги ниже рисунка).

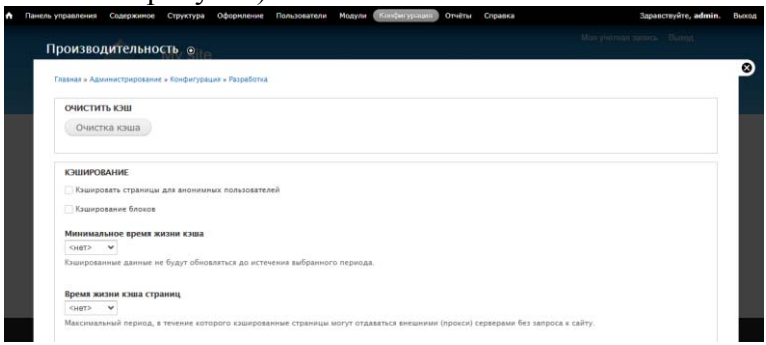


Рисунок 45. Очистка кэша ядра сайта

Создадим две вкладки **Список клиентов** и **Добавить клиента** для работы с таблицей **customers** (рисунок 46)**.**

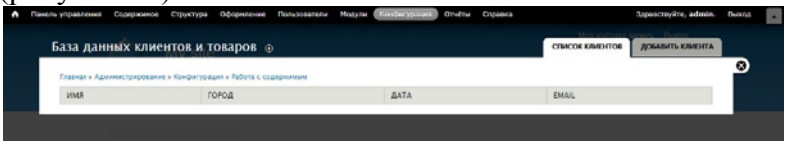


Рисунок 46. Вкладки модуля

Допишем в hook **productshop_menu** операторы создания элементов вкладок. Строки 21-25 первая вкладка. В 26-33 описана вторая вкладка. Строки 21 и 26 назначают ссылки на страницы вкладок. Параметр **title** задает имя вкладки (рисунок 46)**.** Опция **type,** со значением **MENU_DEFAULT_LOCAL_TASK** фиксирует вкладку, открываемую по умолчанию (в ней будет открываться информация из 17 параметра **page_callback** – функция **customers_list**). Номер вкладки задается **weight.** Для второй вкладки строкой 29 вызывается форма сбора полей **productshop_form**.

```
11  function productshop_menu()
12  {
13      $items = array();
14      $items['admin/config/content/productshop'] = array(
15          'title'            => 'База данных клиентов и товаров',
16          'description'      => 'Создание и редактрование товаров и клиентов ',
17          'page callback'    => 'customers_list',
18          'access arguments' => array('administer site configuration'),
19      );
20
21      $items['admin/config/content/productshop/list'] = array(
22          'title'  => 'Список клиентов',
23          'type'   => MENU_DEFAULT_LOCAL_TASK,
24          'weight' => 1,
25      );
26      $items['admin/config/content/productshop/add'] = array(
27          'title'            => 'Добавить клиента',
28          'page callback'    => 'drupal_get_form',
29          'page arguments'   => array('productshop_form'),
30          'access arguments' => array('administer site configuration'),
31          'type'             => MENU_LOCAL_TASK,
32          'weight'           => 2,
33      );
34      return $items;
35
36  }
```

В теме номер три был описан hook **form** (рисунок 34) без передачи ему аргументов. Будем использовать форму для сбора информации полей таблицы и их редактирование.

Аргументы передаются после названия (строка 38) hook **function productshop_form($form, &$form_state, $rss=null).**

Первые два параметра хранят данные о структуре формы и ее характеристик. Нам нужен третий аргумент **$rss.** Это кеш массив. Через него будем передавать значения полей в форму для редактирования.

Основа формы взята из предыдущей темы (рисунок 36), с небольшими изменениями. Для удобства описания опции в полях формы расположены в другом порядке (не влияет на работу hook).

В поле появились две новые опции (**required** и **default_value**). С помощью первой опции (рисунок 47), после каждого имени поля выводиться красная звездочка, требующая обязательного заполнения перед отправкой.

Вторая опция (строки 44, 51, 58, 65) заполняют поля формы передаваемыми значениями.

Запись **$rss? $rss['city']: ''** описывает условный оператор. Если условие истинно, выполняется команда от знака вопроса, до двоеточия, иначе после двоеточия.

```php
function productshop_form($form, &$form_state, $rss = null)
{
    $form['name'] = array(
        '#title'         => t('Имя'),
        '#description'   => t('Ввведите имя клиента'),
        '#type'          => 'textfield',
        '#default_value' => $rss ? $rss['name'] : '',
        '#required'      => true,
    );
    $form['city'] = array(
        '#title'         => t('Город'),
        '#description'   => t('Ввведите город клиента'),
        '#type'          => 'textfield',
        '#default_value' => $rss ? $rss['city'] : '',
        '#required'      => true,
    );
    $form['date'] = array(
        '#title'         => t('Дата'),
        '#description'   => t('Ввведите дату'),
        '#type'          => 'date',
        '#default_value' => $rss ? $rss['date'] : '',
        '#required'      => true,
    );
    $form['email'] = array(
        '#title'         => t('Электронный адрес'),
        '#description'   => t('Ввведите email'),
        '#type'          => 'textfield',
        '#default_value' => $rss ? $rss['city'] : '',
        '#required'      => true,
    );
    $form['submit'] = array(
        '#type'  => 'submit',
        '#value' => $rss ? t('Внести изменения') : t('Добавить новую запись'),
    );

    if ($rss) {
        $form['id_cust'] = array(
            '#type'  => 'value',
            '#value' => $rss['id_cust'],
        );
    }

    return $form;
}
```

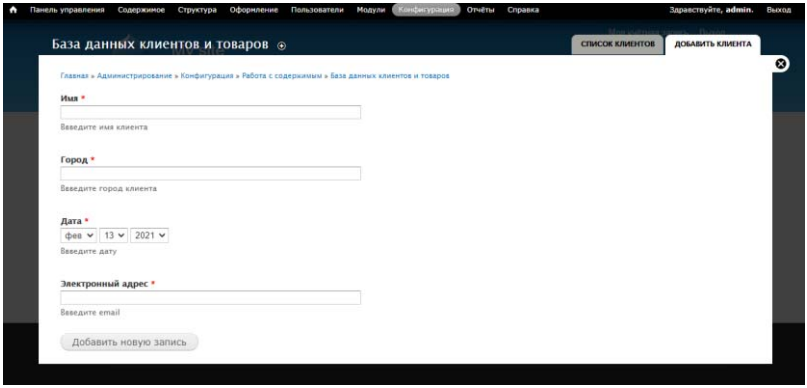


Рисунок 47. Форма productshop_form

При вызове из вкладки добавить клиента значение **$rss** пустое (в строке 29 **'page arguments' => array('productshop_form')** нет передаваемых значений.) Наша форма будет вызвана без передаваемых значений.

Строка 70 выведет название кнопки отправки формы в зависимости от выполняемого действия (**Внести изменения** или **Добавить новую запись**)

Строки 73-78 создают скрытое поле **id_cust** (аналог типа **hidden** в HTML). В нем передаем ключевое поле номера записи в таблице **customers** (рисунок 43).

Следующим шагом будет отправка формы hook **submit**. Описание проверки полей hook **validate** пропускается. Выполняется аналогично, как в теме номер три.

При написании hook **productshop_form** использовалось поле типа **date**. Формат записи у него **день-месяц-год.** В базе данных в таблице **customers** тип поля **datetime** с форматом **год-месяц-день.**

Установим модуль **Date (**www.drupal.org/project/date) для конвертирования даты (рисунок 48).

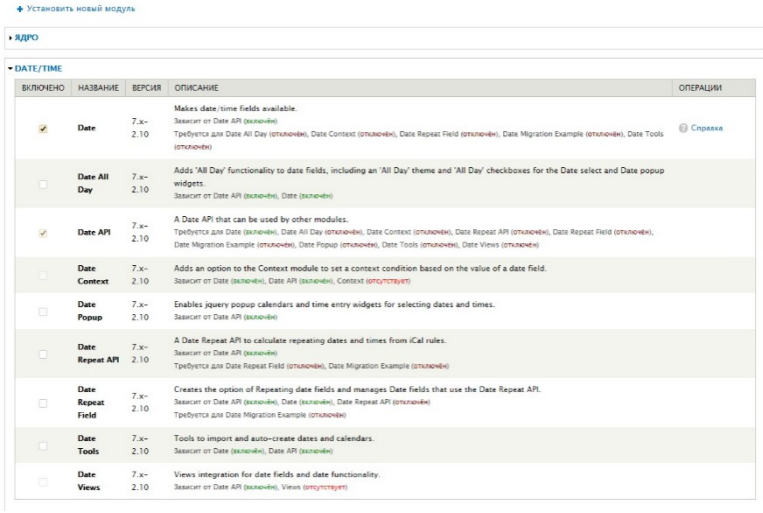


Рисунок 48. Активация модуля Date

Активируем опции **Date** и **Data API** (рисунок 48)**.**

Преобразуем поле формы **date** (строка 54-61) в новый тип поля **data_select** (строка 57), установленного модуля **DATE**.

В строке 58 выставляем порядок вывода года, месяца и дня.

В ядре DRUPAL реализовано мало функций разработки информационной системы. Необходимые модули устанавливаются дополнительно с официального сайта **drupal.org**.

```
38  function productshop_form($form, &$form_state, $rss = null)
39  {
40      $form['name'] = array(
41          '#title'         => t('Имя'),
42          '#description'   => t('Ввведите имя клиента'),
43          '#type'          => 'textfield',
44          '#default_value' => $rss ? $rss['name'] : '',
45          '#required'      => true,
46      );
47      $form['city'] = array(
48          '#title'         => t('Город'),
49          '#description'   => t('Ввведите город клиента'),
50          '#type'          => 'textfield',
51          '#default_value' => $rss ? $rss['city'] : '',
52          '#required'      => true,
53      );
54      $form['date'] = array(
55          '#title'         => t('Дата'),
56          '#description'   => t('Ввведите дату'),
57          '#type'          => 'date_select',
58          '#date_format'   => ('Y-m-d'),
59          '#default_value' => $rss ? $rss['date'] : '',
60          '#required'      => true,
61      );
62      $form['email'] = array(
63          '#title'         => t('Электронный адрес'),
64          '#description'   => t('Ввведите email'),
65          '#type'          => 'textfield',
66          '#default_value' => $rss ? $rss['city'] : '',
67          '#required'      => true,
68      );
69      $form['submit'] = array(
70          '#type'  => 'submit',
71          '#value' => $rss ? t('Внести изменения') : t('Добавить новую запись'),
72      );
73      if ($rss) {
74          $form['id_cust'] = array(
75              '#type'  => 'value',
76              '#value' => $rss['id_cust'],
77          );
78      }
79      return $form;
80  }
```

Переходим к hook **submit** (строки 82-103) **productshop_form_submit**.

В строках 85-90 в кеш массив **$rss** извлекаем значения полей **name**, **city**, **date** и **email** из формы. Если вноситься новая запись в базу данных, выполняется строка 97. При редактировании записи в таблице вызывается строка 92.

Условие (строка 92) определяет существует ли скрытое поле формы **id_cust.**

```
82  function productshop_form_submit($form, &$form_state)
83  {
84
85      $rss = array(
86          'name'        => $form_state['values']['name'],
87          'city'        => $form_state['values']['city'],
88          'date'        => $form_state['values']['date'],
89          'email'       => $form_state['values']['email'],
90      );
91      // save edit data
92      if (isset($form_state['values']['id_cust'])) {
93          $rss['id_cust'] = $form_state['values']['id_cust'];
94          drupal_write_record('customers', $rss, 'id_cust');
95          drupal_set_message(t('Данные клиента изменены '));
96      } // add new data
97      else {
98          drupal_write_record('customers', $rss);
99          drupal_set_message(t('Новый клиент добавлен.'));
100     }
101
102     drupal_goto('admin/config/content/productshop');
103 }
```

За внесение записи в базу данных в таблицу **customers** отвечают строки 94 и 98.

Оператор **drupal_set_message** выводит на экран сообщение.

Заполните поля формы и их отправьте (рисунок 50). Зайдите в phpMyAdmin (рисунок 49) и проверьте, что данные формы были внесены в таблицу.

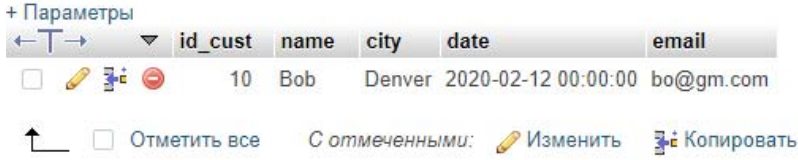


Рисунок 49. Содержимое таблицы customers

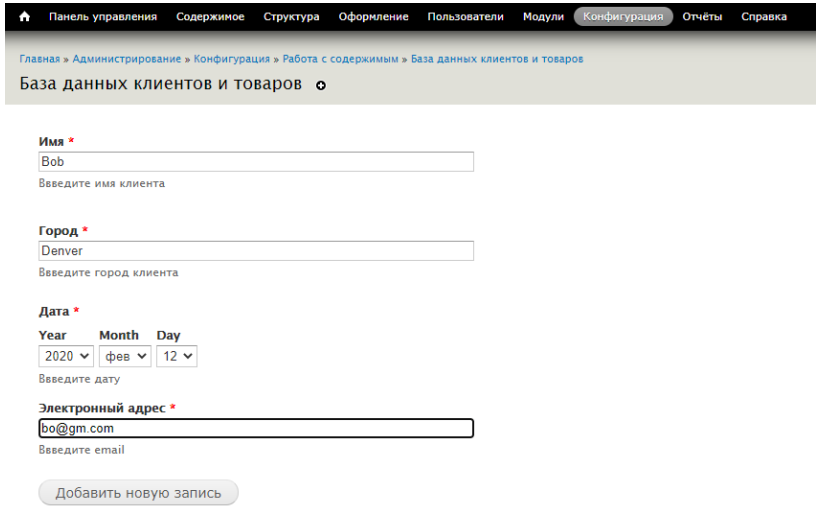


Рисунок 50. Новый тип date_select модуля Date

В примере ключевое поле **id_cust** имеет значение равное десяти. Оно создается автоматически базой данных. Номер ключа может отличаться от исходного (рисунок 3).

Возвращаемся к функции **customers_list** (рисунок 42). Функция генерировала HTML-таблицу с заголовками таблиц **Имя**, **Город**, **Дата** и **Email**. В строке 114 (смотри ниже) добавлен столбец (рисунок 51) **Действия**.

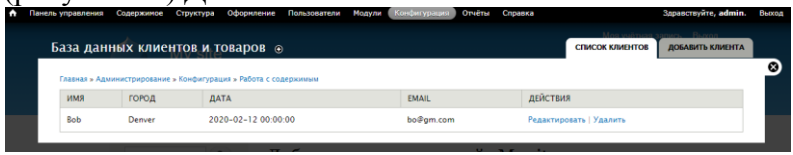


Рисунок 51. Вывод дополненной функции customers_list

```
107 function customers_list()
108 {
109     $header = array(
110         array('data' => t('Имя')),
111         array('data' => t('Город')),
112         array('data' => t('Дата')),
113         array('data' => t('Email')),
114         array('data' => t('Действия'))
115     );
116     $rss = db_select('customers', 'n')
117         ->fields('n', array('id_cust', 'name', 'city','date','email'))
118         ->execute()->fetchAll();
119     $row = array();
120     if ($rss) {
121         foreach ($rss as $rss_cust) {
122             $actions = array(
123                 l(t('Редактировать'),   'admin/config/content/productshop/' . $rss_cust->id_cust . '/edit'),
124                 l(t('Удалить'),         'admin/config/content/productshop/' . $rss_cust->id_cust . '/delete'),
125             );
126             $row [] = array(
127                 array('data' => $rss_cust->name),
128                 array('data' => $rss_cust->city),
129                 array('data' => $rss_cust->date),
130                 array('data' => $rss_cust->email),
131                 array('data' => implode(' | ', $actions)),
132             );
133         }
134     }
135     return theme('table', array(
136         'header' => $header,
137         'rows'   => $row,
138     ));
139 }
```

Строки с 116–118 выполняют SQL-запрос к таблице **customers** (строка 116). В 117 перечисляются названия полей таблицы. Сам запрос выполняется в 118 строке. Строка 118 инициирует массив **$row**.

Если SQL-запрос вернул не пустое значение (строка 120), то в цикле считываются значения из SQL-запроса (строки 121–133). Для удаления и изменения записи создадим две ссылки: **Редактировать** и **Удалить** (строки 123 и 124). Переменная **$rss_cust ->id_cust** добавить в ссылку значение ключевого поля. Необходимо добавить эти ссылки в hook **menu** (будет описано ниже).

Итоговая запись из запроса создается с 126 по 132 строку. В кеш массив **$row** передаются данные из SQL-запроса построчно. Строки 135-138 выводят в HTML-таблицу все собранную информацию (рисунок 51) из таблицы **customers.**

Добавим в hook **menu** ссылку удаления записи из таблицы **customers** (строки 34–41)**.**

```
11  function productshop_menu()
12  {
13      $items = array();
14      $items['admin/config/content/productshop'] = array(
15          'title'            => 'База данных клиентов и товаров',
16          'description'      => 'Создание и редактрование товаров и клиентов
17          'page callback'    => 'customers_list',
18          'access arguments' => array('administer site configuration'),
19      );
20
21      $items['admin/config/content/productshop/list'] = array(
22          'title'  => 'Список клиентов',
23          'type'   => MENU_DEFAULT_LOCAL_TASK,
24          'weight' => 1,
25      );
26      $items['admin/config/content/productshop/add'] = array(
27          'title'            => 'Добавить клиента',
28          'page callback'    => 'drupal_get_form',
29          'page arguments'   => array('productshop_form'),
30          'access arguments' => array('administer site configuration'),
31          'type'             => MENU_LOCAL_TASK,
32          'weight'           => 2,
33      );
34      $items['admin/config/content/productshop/%/delete'] = array(
35          'title'            => 'Удалить запись',
36          'page callback'    => 'customers_delete',
37          'page arguments'   =>  array(4),
38          'access arguments' =>  array('administer site configuration'),
39          'type'             =>  MENU_CALLBACK,
40      );
41      return $items;
42
43  }
```

В 34 строке пишем путь, указанный в функции **customers_list().** Знак **%** означает, что этот параметр изменяемый. Строка 35 – заголовок таблицы. Опция **page callback** вызовет функцию **customers_delete** (опишем ниже). В адресе страницы **admin/config/content/ productshop/%/delete** знак **%** находиться на четвертой позиции, если считать, что admin нулевой элемент. Строкой 37 (**page ragument**) считываем значение, хранимое в позиции знака процента (**id_cust**).

Функция **customers_delete** удаляет запись из таблицы **customers.** На вход подается значение, извлеченное из позиции с символом % (в нем хранилось значения ключа **id_cust**). Строки с 152–154 создают SQL-запрос **delete** к таблице **customers** (строки 152-153). В 154 выполняется

запрос. Оставшиеся строки выводят сообщение и возвращают на первую вкладку **Список клиентов** (можно указать другой путь).

```
150  function customers_delete($rss)
151  {
152      $rss_deleted = db_delete('customers')
153          ->condition('id_cust', $rss)
154          ->execute();
155      drupal_set_message(t('Запись удалена!'));
156      drupal_goto('admin/config/content/productshop');
157  }
158
```

Проверьте в работе модуля, что удаление записи происходит корректно (не забудьте почистить кеш).

Завершается описание модуля, созданием функции редактирования записи.

В hook **menu** дописываем последнюю ссылку редактирования страницы (строки 40–46).

При удалении записи, достаточно было знать только номер ключевого поля. Для этого использовался символ процента. Для редактирования записи, нужно обратиться с SQL-запросом к таблице, считать поля по ключевому полю и полученные данные вывести в форму редактирования. В строке 40 добавляем, после процента, название переменной (**rss**) для хранения ключевого поля.

```
11  function productshop_menu()
12  {
13      $items = array();
14      $items['admin/config/content/productshop'] = array(
15          'title'            => 'База данных клиентов и товаров',
16          'description'      => 'Создание и редактрование товаров и клиентов ',
17          'page callback'    => 'customers_list',
18          'access arguments' => array('administer site configuration'),
19      );
20      $items['admin/config/content/productshop/list'] = array(
21          'title'  => 'Список клиентов',
22          'type'   => MENU_DEFAULT_LOCAL_TASK,
23          'weight' => 1,
24      );
25      $items['admin/config/content/productshop/add'] = array(
26          'title'            => 'Добавить клиента',
27          'page callback'    => 'drupal_get_form',
28          'page arguments'   => array('productshop_form'),
29          'access arguments' => array('administer site configuration'),
30          'type'             => MENU_LOCAL_TASK,
31          'weight'           => 2,
32      );
33      $items['admin/config/content/productshop/%/delete'] = array(
34          'title'            => 'Удалить запись',
35          'page callback'    => 'customers_delete',
36          'page arguments'   =>  array(4),
37          'access arguments' =>  array('administer site configuration'),
38          'type'             =>  MENU_CALLBACK,
39      );
40      $items['admin/config/content/productshop/%rss/edit'] = array(
41          'title'            => 'Редактировать запись',
42          'page callback'    => 'drupal_get_form',
43          'page arguments'   =>  array('productshop_form',4),
44          'access arguments' =>  array('administer site configuration'),
45          'type'             =>  MENU_CALLBACK,
46      );
47      return $items;
48
49  }
50
```

Обратите внимание в строке 28 форма (**productshop_form**) вызывается без параметра. В новой ссылке меню, (строка 43) мы добавили позицию взятия значения ключевого поля из адреса (как это было при удалении записи). При создании hook **productshop_form** был создан механизм вызова формы с пустыми полями или с заполненными значениями из записи таблицы (опция **default_value**).

Для извлечения значений полей из таблицы **customers** добавим hook **load** (строки 166–173).

```
166  function rss_load($id)
167  {
168      $rss = db_select('customers', 'n')
169          ->fields('n', array('id_cust', 'name', 'city', 'date', 'email'))
170          ->condition('n.id_cust', $id)
171          ->execute()->fetchAssoc();
172      return $rss;
173  }
```

В название добавляем имя переменной rss перед **load.** Строки 168–170 создают SQL-запрос **select** с выборкой по ключевому полю **id_cust** (строка 170). Строкой 171 выполняется запрос. В 172 строке возвращается кеш массив со значениями полей таблицы.

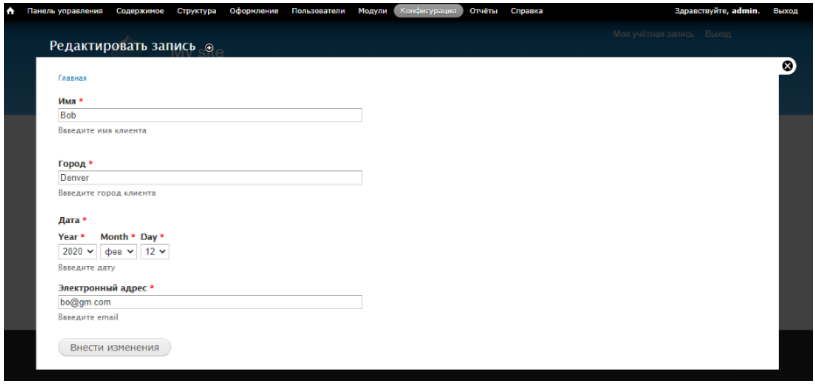


Рисунок 52. Вывод формы с заполненными полями для редактирования

При переходе по ссылке **Редактировать** (рисунок 51) открывается форма с заполненными полями (рисунок 52).

Примечание. Если у вас вместо значения email выводиться название города, подумайте, что нужно изменить в **productshop_form.**

Заключение

Темой номер 4 заканчивается первая часть знакомства с базой данных MYSQL и CMS DRUPAL.

Во второй части будет рассмотрен вопрос проектирования базы данных и создания более сложных информационных систем.

Задача 4.1 (индивидуальная)

По выбранному варианту темы 1, создать еще две вкладки вывода содержимого таблицы (ссылки **редактировать** и **удалить** должны быть реализованы) и внесения записи в таблицу. Выбирайте таблицы, которые не содержат внешних ключей.

Список литературы

1. Колисниченко Д.Н. Drupal 7 Руководство пользователя. – Москва: Диалектика, 2011. – 252 с.

2. Осипов Д.Л. Технология проектирования баз данных. – Москва: ДМК Пресс, 2019 – 497 с.

3. Дейт К. Дж. Введение в системы баз данных. – Москва; Диалектика, 2019. – 1327 с.

4. Алексей Черных – Drupal 7. Бесплатная система управления сайтом. Москва: Эксмо, 2011. – 208 с.

5. Мелансон Б., Нордин Д., Луиси Ж. Профессиональная разработка сайтов на Drupal 7. Издательский дом Питер 2013. – 688 с.

Учебное издание

Составитель
Логов Алексей Генритович

РАЗРАБОТКА WEB-ИНТЕРФЕЙСА К MYSQL В DRUPAL 7

ЧАСТЬ I

Учебно-методическое пособие

Авторская редакция

Отпечатано с оригинал-макета заказчика

Подписано в печать 17.03.2021. Формат 60x84 $^{1}/_{16}$.
Тираж 50 экз. Заказ № 482.

Типография Издательского центра
«Удмуртский университет»
426034, Ижевск, ул. Университетская, 1, корп. 2.
Тел. 68-57-18