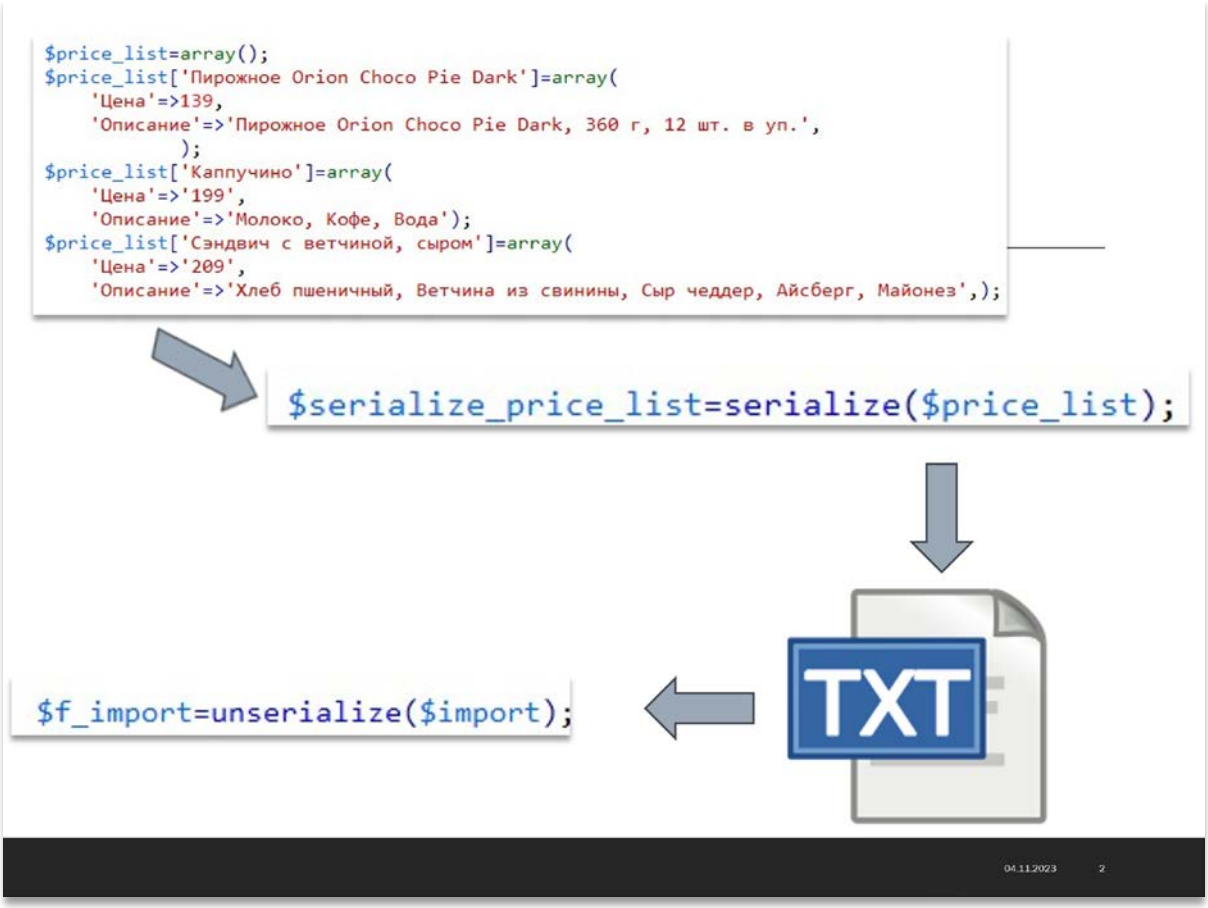
Взаимодействие PHP и MySQL

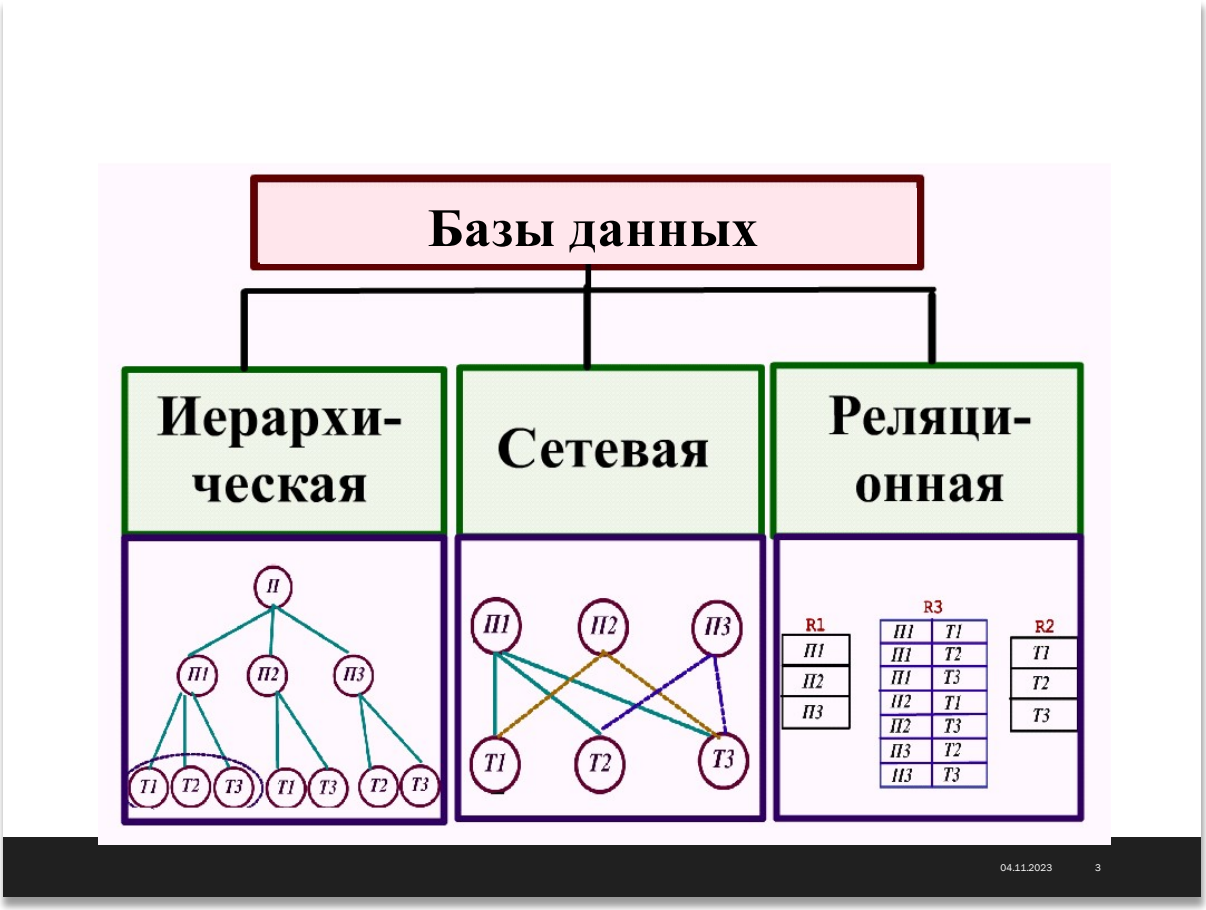


Ранее для долговременного хранения информации мы работали с файлами: помещали в них текст – сериализованный массив, а затем извлекали его, конвертировали снова в массив для последующей работы. Задача длительного хранения информации очень часто встречается в программировании Web-приложений: Хранение контента для формирования динамических WEB-страниц, Учет посетителей, подсчёт посетителей в счётчике, хранение сообщений в форуме, удалённое управление содержанием информации на сайте и т.д.

Использование файловой системы сервера для хранения информации в силу своей функциональности вызывает ряд затруднений для оперативного поиска информации и внесения изменений. Профессиональные приёмы работы с файлами очень трудоёмки: необходимо заботится о помещении в них информации, о её сортировке, извлечении, при этом не нужно забывать, что все эти действия будут происходить на сервере хост-провайдера, где с очень большой вероятностью стоит один из вариантов Unix - следовательно, нужно так же заботится о правах доступа к файлам и их размещении. При этом объём кода значительно возрастает, и совершить ошибку в программе очень просто.

Перечисленные выше задачи с успехом решает применение базы данных, которые сами координируют безопасность информации, ее сортировку, а также дают возможность извлечения и размещения данных с использованием одной строчки. Код с применением базы данных имеет более компактный вид, поэтому и отлаживать его гораздо проще. Помимо этого, не следует забывать и о показателях скорости: выборка информации из базы данных осуществляется более быстро, чем из файлов.

Взаимодействие с базой данных происходит при помощи Системы Управления Базой Данных (СУБД), которая расшифровывает запросы и производит операции с информацией в базе данных. Поэтому более правильно было бы говорить о запросе к СУБД и о взаимодействии с СУБД из Web-приложения. Но так как это несколько усложняет восприятие, далее везде мы будем говорить "база данных", подразумевая при этом СУБД.

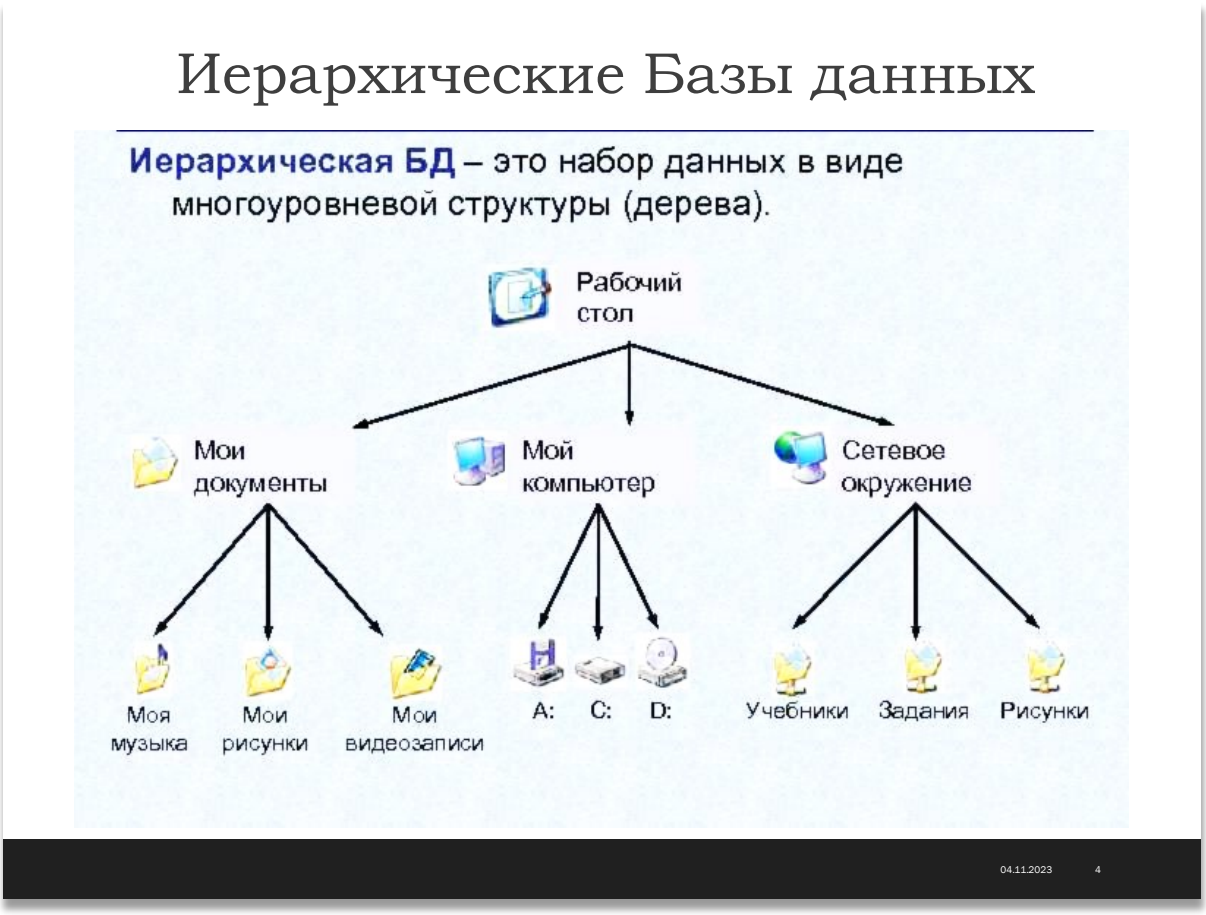


Существуют следующие разновидности баз данных:

иерархические;

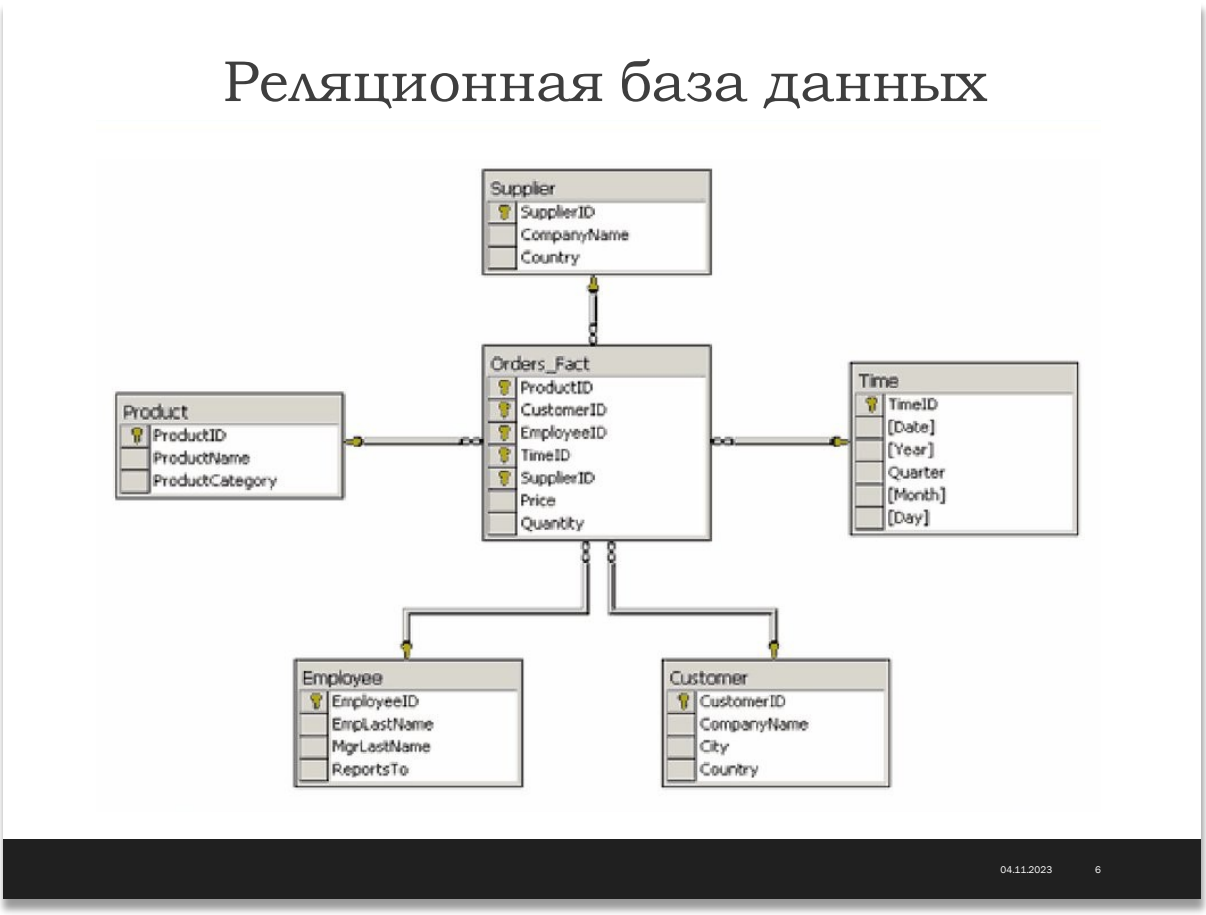
сетевые

реляционные;



Иерархическая база данных основана на древовидной структуре хранения информации. В этом смысле иерархические базы данных очень напоминают файловую систему компьютера.

Сетевая база данных организует данные в иерархической форме, с узлами, связанными отношениями. Данные хранятся в сети узлов и связей, причем каждый узел может иметь множество связей с другими узлами. Иерархическая структура сетевой базы данных определяется типами записей, которые определяют свойства узлов, и наборами, которые группируют узлы вместе на основе их свойств.



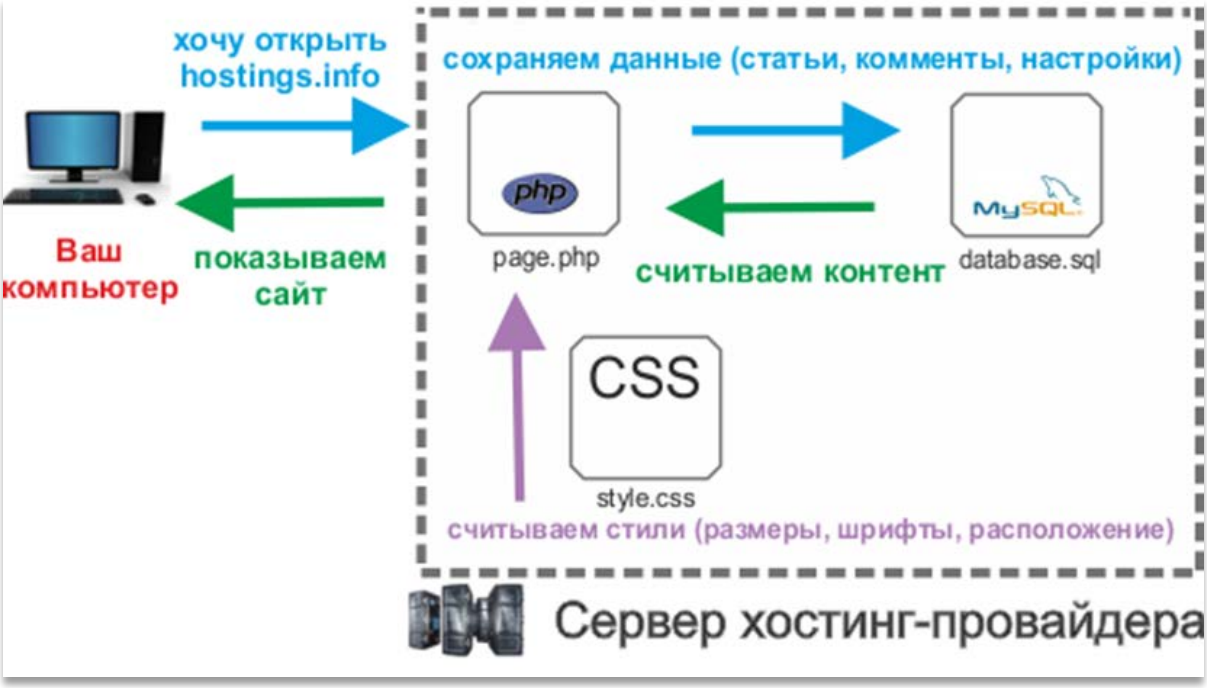
В реляционных базах данных данные собраны в таблицы, которые в свою очередь состоят из столбцов и строк, на пересечении которых расположены ячейки. Запросы к таким базам данных возвращает таблицу, которая повторно может участвовать в следующем запросе. Данные в одних таблицах, как правило, связаны с данными других таблиц, откуда и произошло название "реляционные".

В своих проектах мы будем использовать реляционную базу данных MySQL.

Что такое MySQL?

MySQL – MySQL представляет собой одну из самых распространенных сегодня систем управления базами данных в сети Интернет.

Данная система используется для работы с достаточно большими объемами информации. Однако MySQL идеально подходит как для небольших, так и для крупных интернет-проектов. Немаловажной характеристикой системы является ее бесплатность.



MySQL отличатся хорошей скоростью работы, надежностью, гибкостью. Работа с ней, как правило, не вызывает больших трудностей. Поддержка сервера MySQL автоматически включается в поставку PHP.

Приложение на PHP, использующее для хранения информации базу данных (в частности MySql), всегда работает быстрее приложения, построенного на файлах. Дело в том, что базы данных написаны на языке C++, и написать на PHP программу, которая работала бы с жёстким диском эффективнее базы данных - задача неразрешимая по определению, поскольку программы на PHP в принципе работают медленнее, чем программы на C++, так как PHP - интерпретатор, а C++ - компилятор.

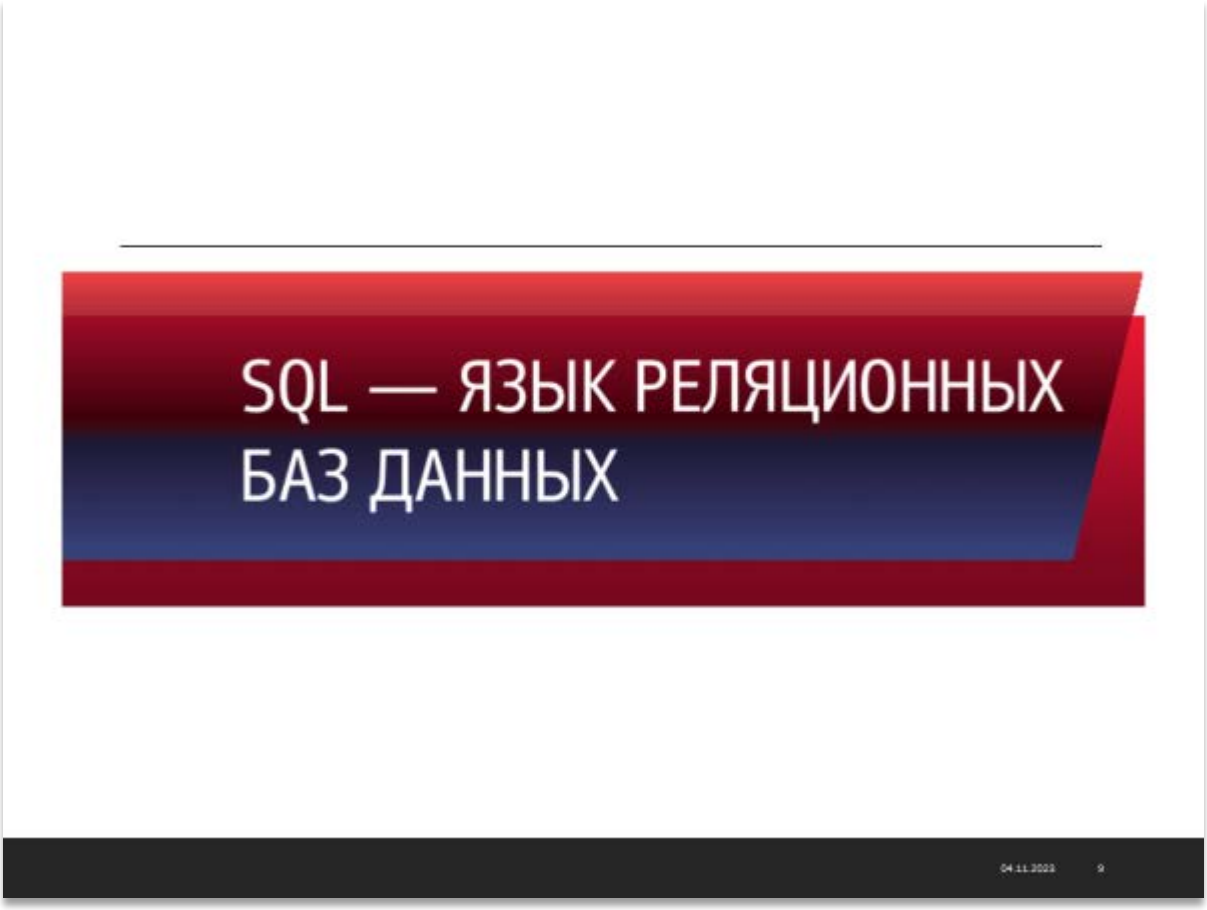
Таким образом, основное достоинство базы данных заключается в том, что она берёт на себя всю работу с жёстким диском и делает это очень эффективно.

Реляционные базы данных, язык запросов SQL

Задача длительного хранения и обработки информации появилась практически сразу с появлением первых компьютеров. Для решения этой задачи в конце 60-х годов были разработаны специализированные программы, получившие название систем управления базами данных (СУБД). СУБД проделали длительный путь эволюции от системы управления файлами, через иерархические и сетевые базы данных.

В конце 80-х годов доминирующей стала система управления реляционными базами данных (СУРБД). С этого времени такие СУБД стали стандартом де-факто, и для того, чтобы унифицировать работу с ними, был

разработан **структурированный язык запросов** (SQL), который представляет собой язык управления именно реляционными базами данных.

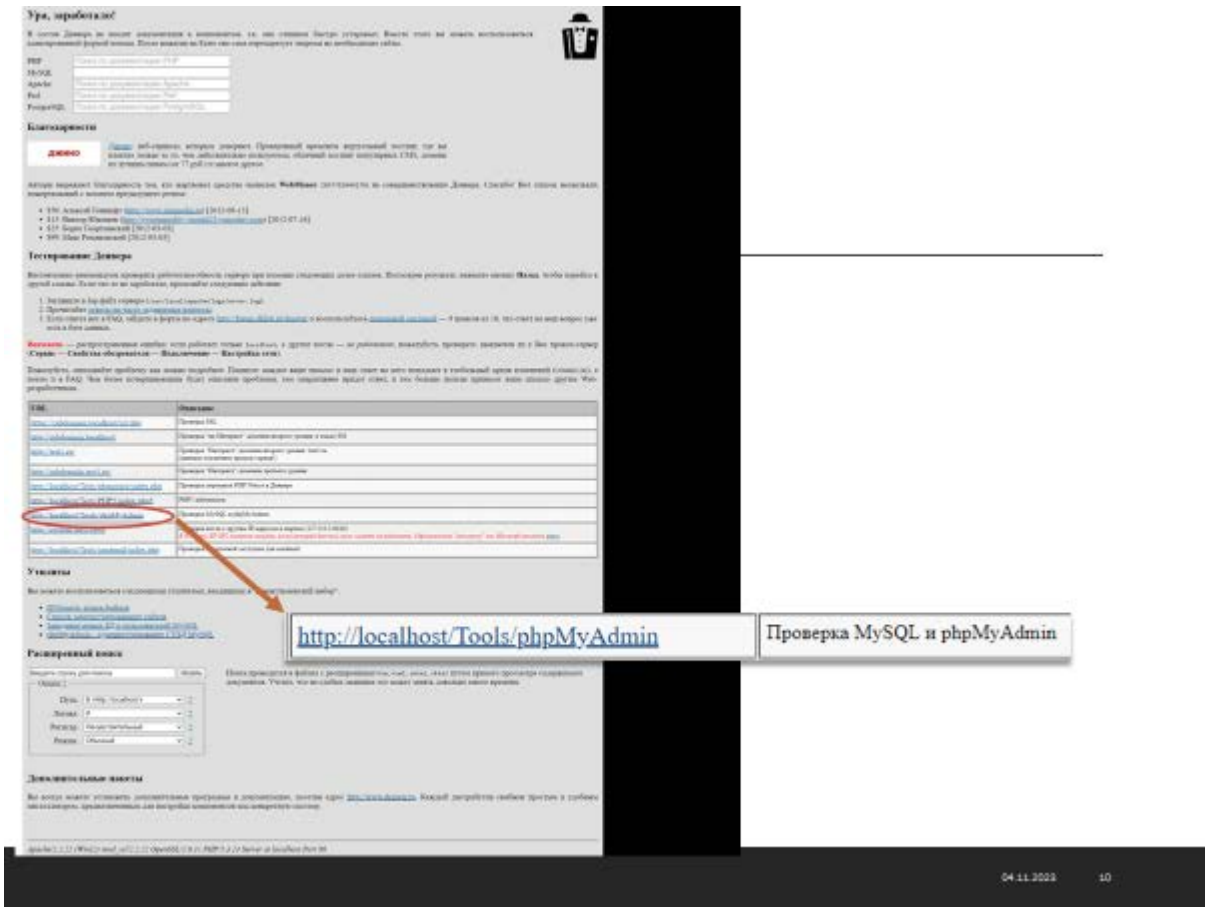


В Web-приложениях, как правило, используются реляционные базы данных. Мы будем рассматривать пример базы данных, на которой строить наш интернет-магазин.

Управлять базой данных MySQL мы будем с помощью WEB-приложения PHPMyAdmin, которое присутствует как и в Денвере, так и на всех хостинговых площадках провайдеров.

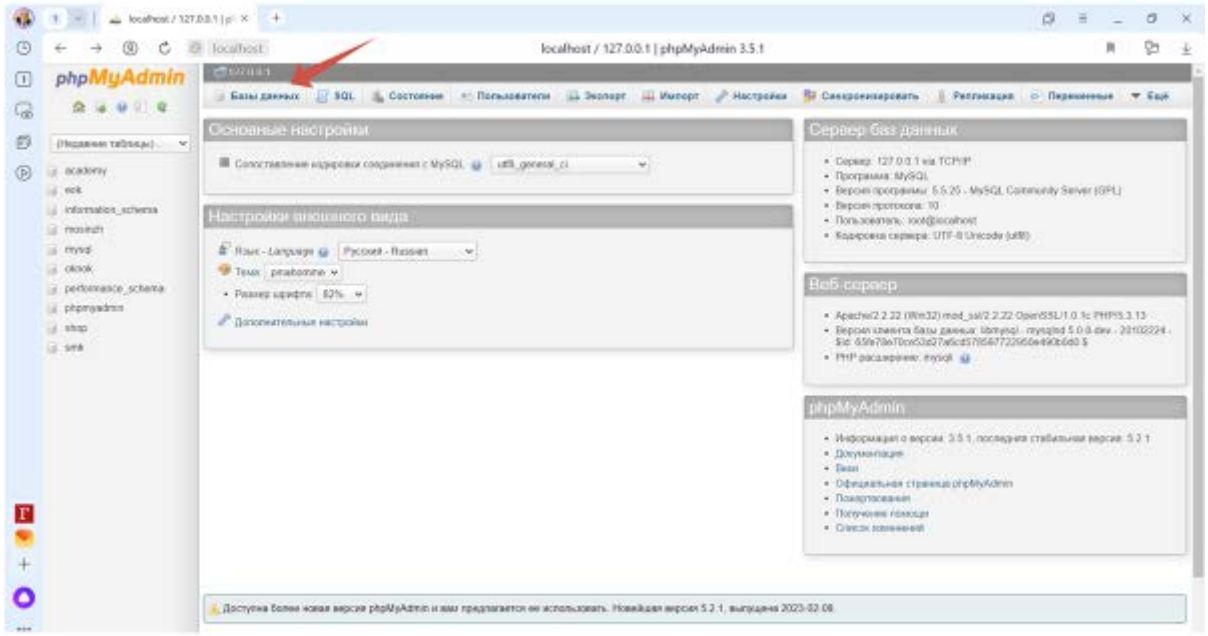
В Денвере PHPMyAdmin находится по адресу:
http://localhost/Tools/phpMyAdmin

Найти её можно легко на главной странице Денвера, которая открывается по адресу http://localhost

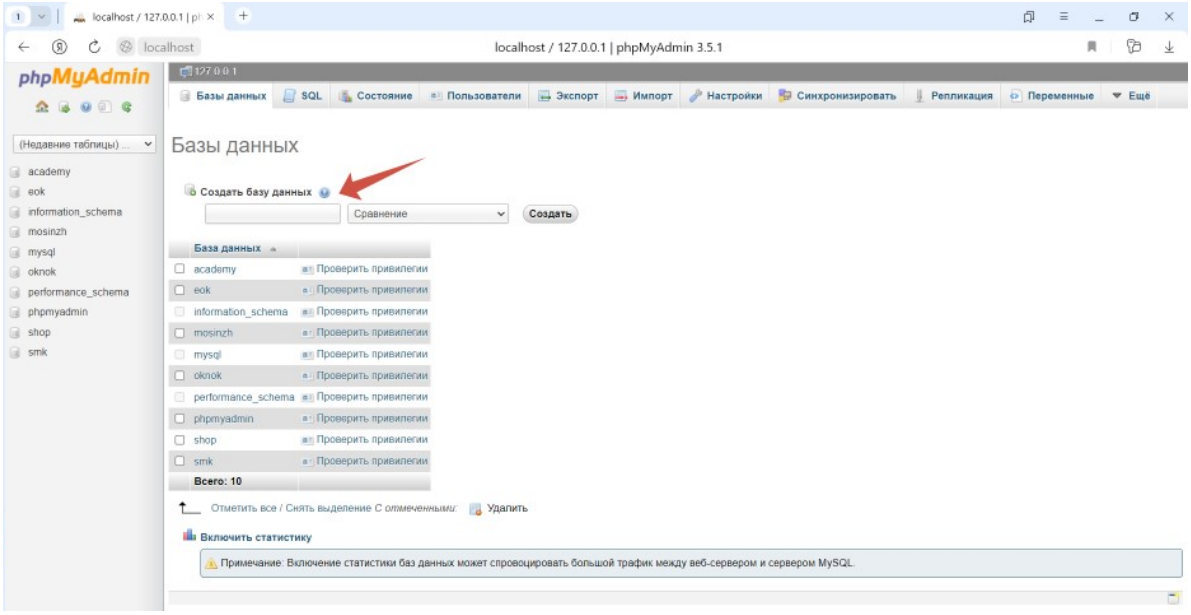


Создавать нашу базу данных и таблицы в ней будем так же с помощью phpMyAdmin.

Чтобы создать базу данных, нужно перейти по ссылке и на главной странице phpMyAdmin кликнуть по вкладке «Базы данных».

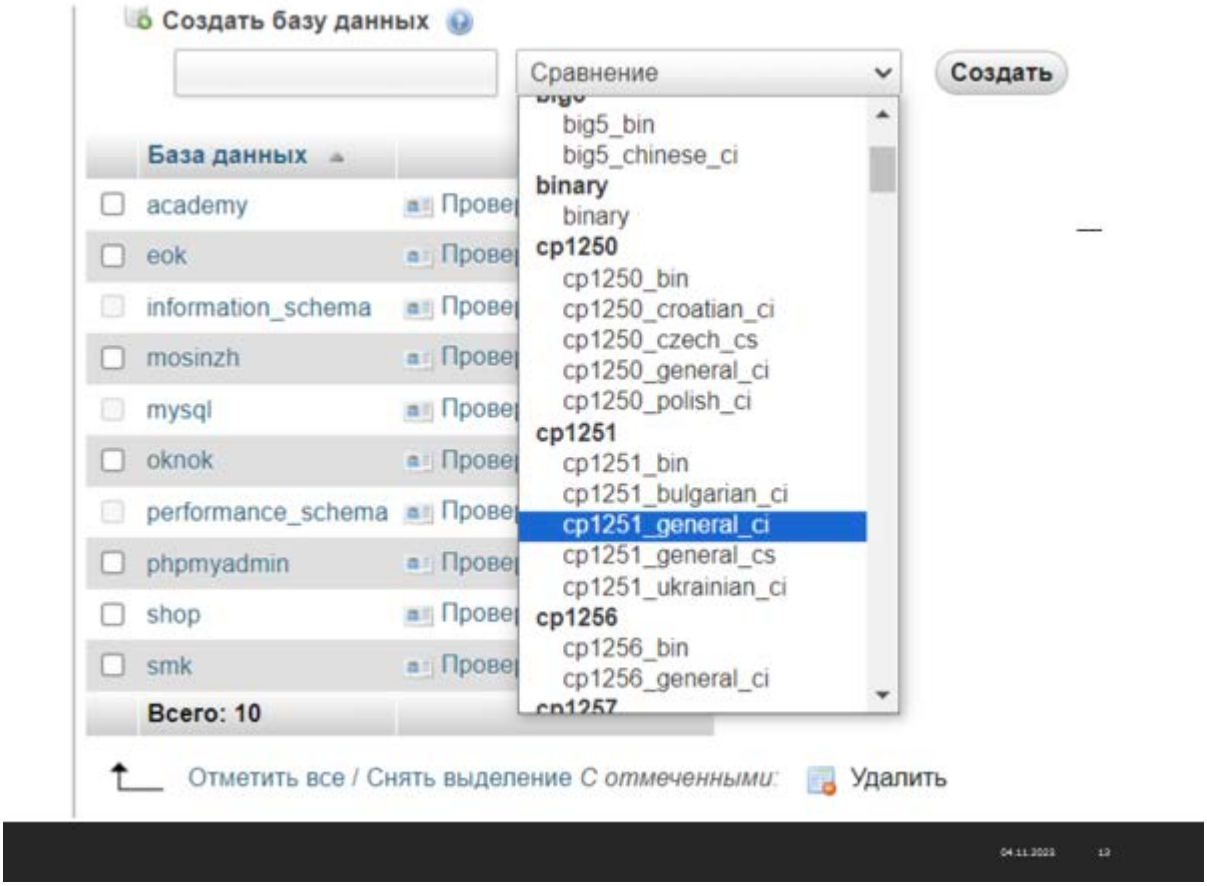


В раскрывшемся окне появится список всех баз данных, которые созданы на web-сервере Денвер, а так же форма для создания новой базы данных:

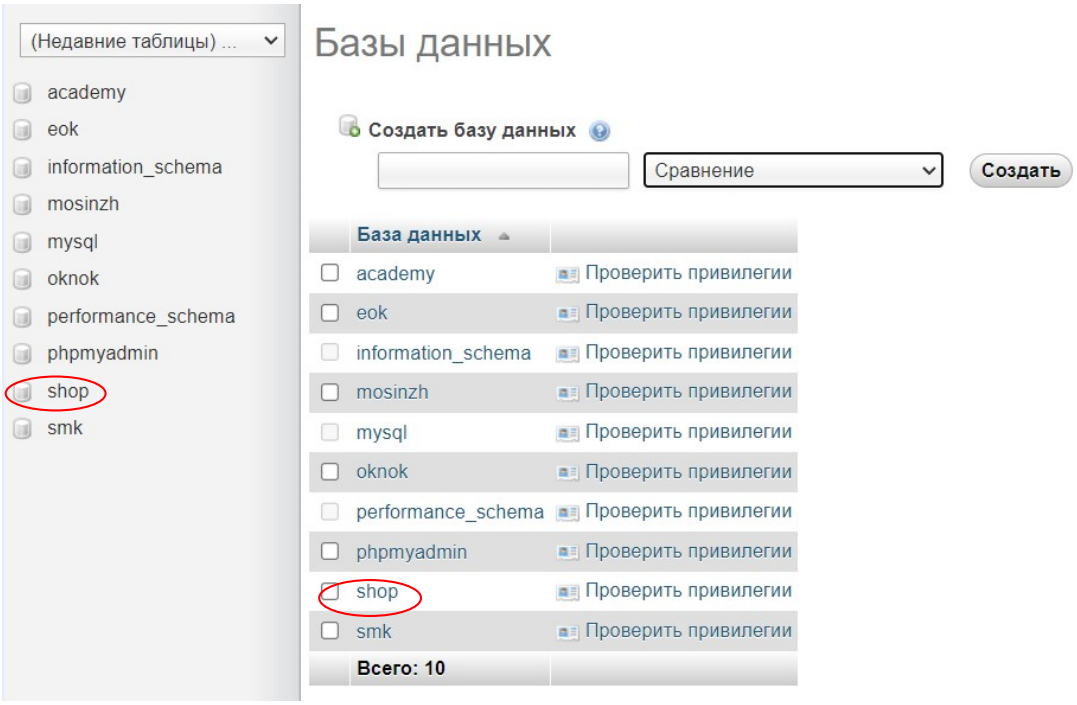


Необходимо ввести имя базы данных и нажать кнопку Создать.

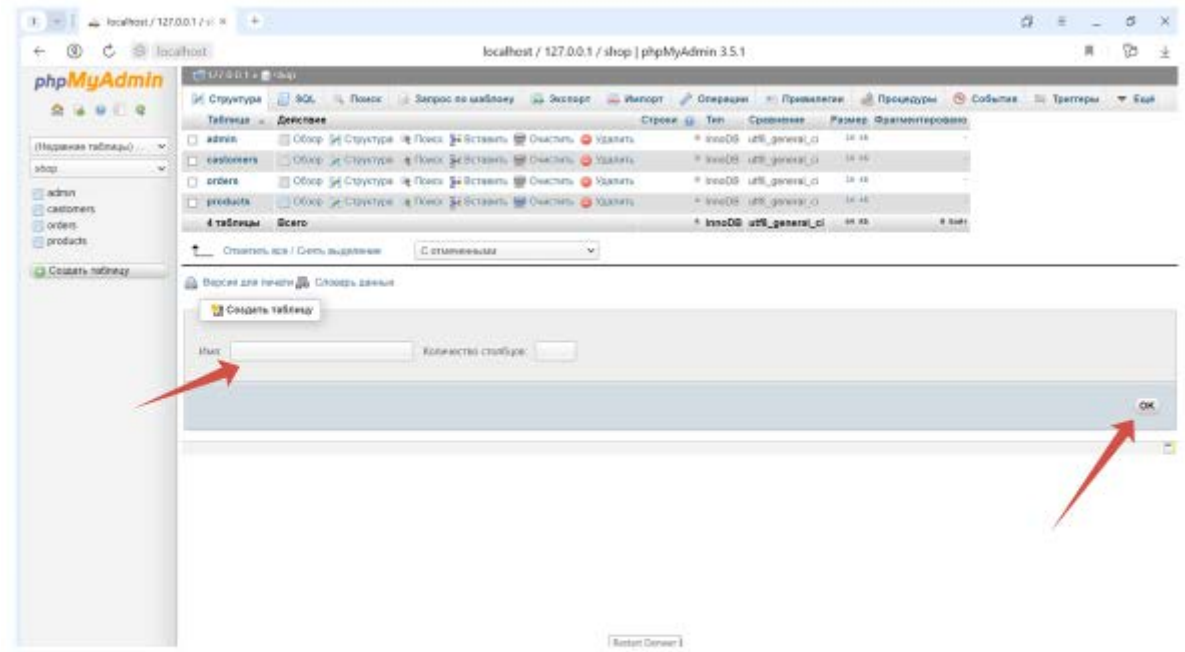
По умолчанию база создаётся с кодировкой utf-8. Нужную кодировку можно установить, выбрав её из списка «Сравнение»



Вновь созданная база данных появится в окне страницы и в левом меню:

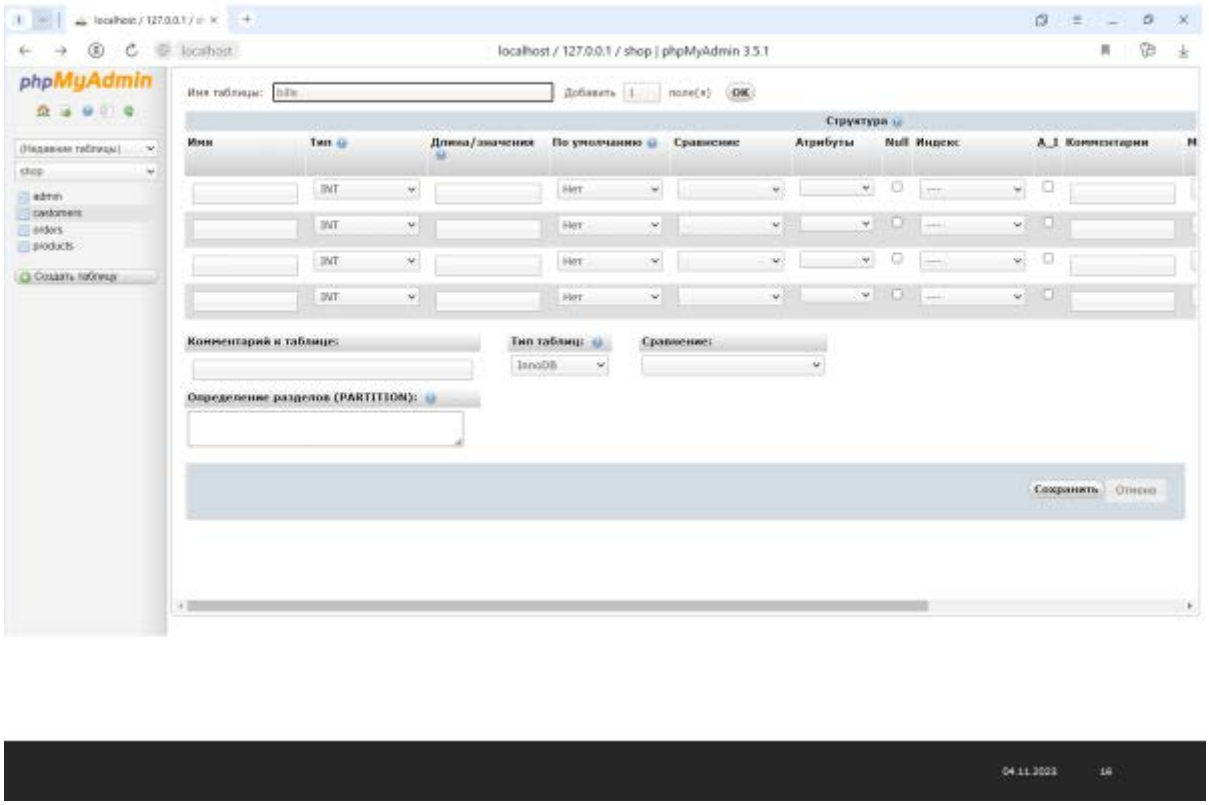


Для перехода в базу нужно кликнуть по имени базы.



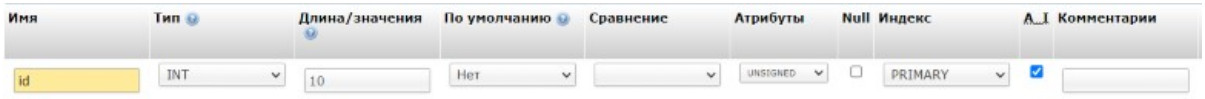
В раскрывшемся окне появится список таблиц, если база только что создано этот список будет пустым.

Чтобы создать новую таблицу, необходимо ввести в поле имя таблицы и нажать кнопку ОК. При необходимости, если вы уже знаете структуру вашей таблицы, можно указать количество столбцов вашей таблицы.



В новом окне появится форма для создания структуры таблицы, в неё вбиваются имена полей, указываются их типы и задаются необходимые свойства.

Задание параметров для первичного ключа



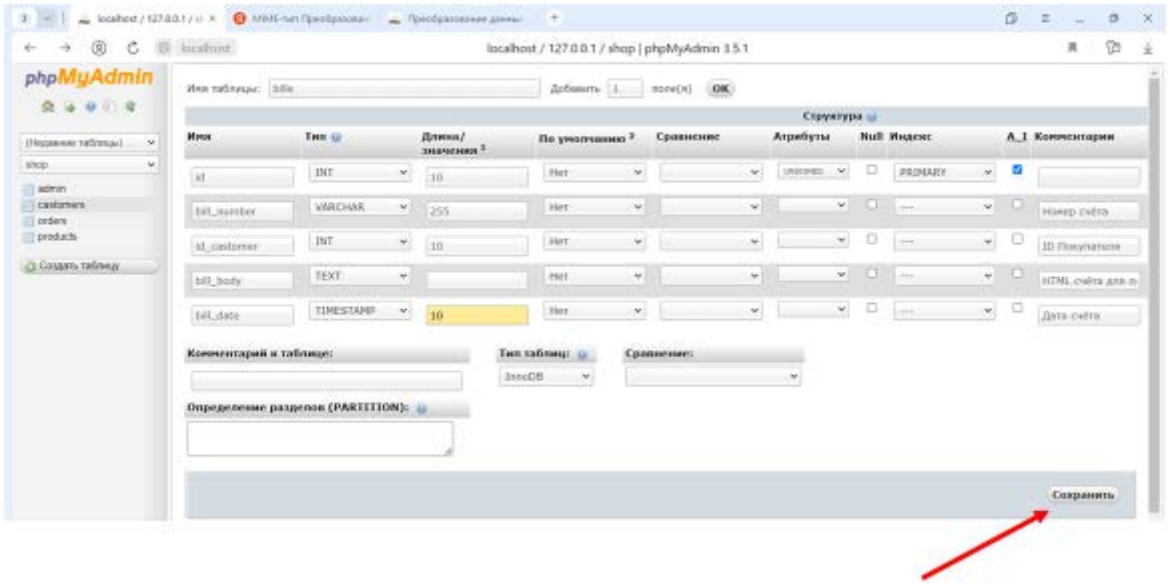
Для первичного ключа обычно задаются следующие свойства:

Имя – id, тип – большое целое число, максимум 10 знаков, атрибут unsigned не допускает отрицательных значений, Индекс – первичный, AI обозначает автоинкрементен, то есть автоматическое заполнение, начиная с единицы.

Остальные поля создаются в основном с указанием типа и размера.

В поле типа TEXT размер не указывается

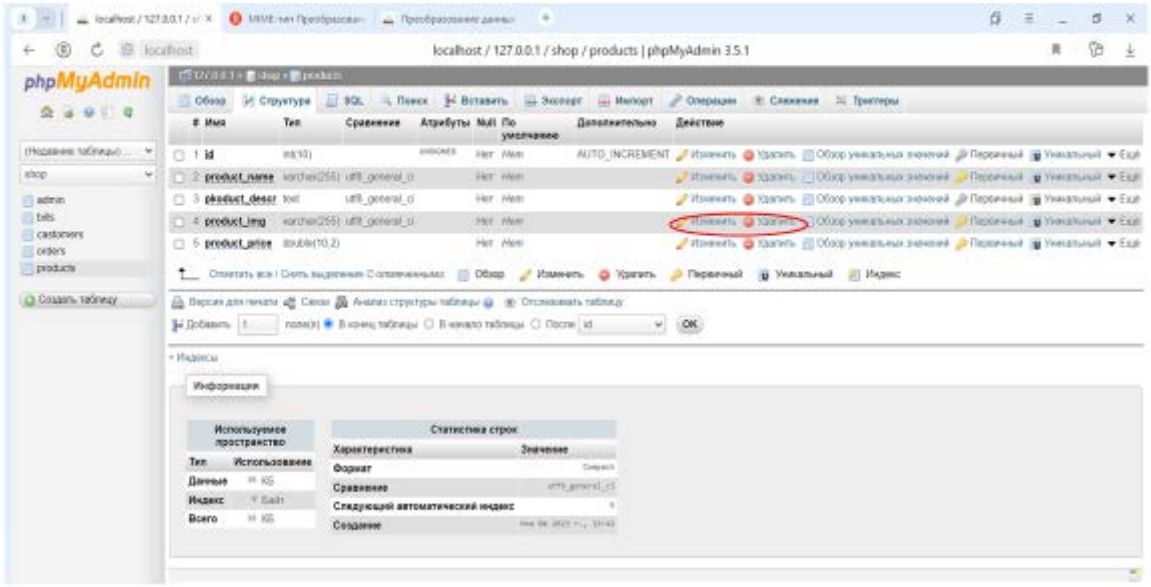
Создание полей таблицы



После заполнения формы необходимо кликнуть по кнопке Сохранить.

Структуру таблицы можно изменить, отредактировав поле или удалив его:

Возможность редактирования структуры таблицы



Таким образом, наша база данных будет включать следующие таблицы:

Таблицы базы данных shop

127.0.0.1 » shop

Таблица	Строки	Тип	Размер	Комментарии
admin	0	InnoDB	16 КБ	Создание: Ноя 04 2023 г., 19:58
bills	0	InnoDB	16 КБ	Таблица учёта счетов Создание: Ноя 05 2023 г., 00:22
castomers	0	InnoDB	16 КБ	Создание: Ноя 04 2023 г., 22:34
orders	0	InnoDB	16 КБ	Создание: Ноя 04 2023 г., 21:07
products	4	InnoDB	16 КБ	Создание: Ноя 04 2023 г., 19:41
5 таблиц	4	--	80 КБ	

Модель реляционной базы данных представляет данные в виде таблиц, разбитых на строки и столбцы, на пересечении которых находятся данные. Пример структуры таблицы products:

Products

Поле	Тип	Комментарии
id	int(10)	Уникальный индекс записи
product_name	varchar(255)	Наименование товара
pkoduct_descr	text	Описание товара
product_img	varchar(255)	Ссылка на изображение товара
product_price	double(10,2)	Цена товара

Таблица содержит 5 полей

Каждое поле представляет собой по сути дела заголовок столбца таблицы.

Вот пример заполненной таблицы products:

id	product_name	pkoduct_descr	product_img	product_price
1	Масло универсальное, БИБИП, 100 мл	Масло универсальное, БИБИП, подойдёт для смазывани...	https://img.fix-price.com/800x800/images/origin/or...	62.00
2	Будильник, FLARX, в ассортименте	Будильник, FLARX, сочетает лаконичность форм и ярк...	https://img.fix-price.com/800x800/images/origin/or...	149.00
3	Светодиодный фонарик, FLARX, 9,5 см, в ассортимент...	Светодиодный фонарик, FLARX, пригодится дома или н...	https://img.fix-price.com/800x800/images/origin/or...	79.00
4	Кондитерское изделие "Choco Pie", Orion, 360 г	Кондитерское изделие “Choco Pie“, Orion – классиче...	https://img.fix-price.com/800x800/pim/pimcore-prod...	139.00

Каждая запись в таблице базы данных представляет собой строку с уникальным индексом id.

Кратко особенности реляционной базы данных можно описать следующим образом:

- Данные хранятся в таблицах, состоящих из столбцов и строк;
- На пересечении каждого столбца и строчки стоит в точности одно значение;
- У каждого столбца есть своё имя, которое служит его названием, и все значения в одном столбце имеют один тип. Например, в столбце id_ все значения имеют целочисленный тип, а в строке product_name - текстовый;
- Столбцы располагаются в определённом порядке, который определяется при создании таблицы, в отличие от строк, которые располагаются в произвольном порядке. В таблице может не быть не одной строчки, но обязательно должен быть хотя бы один столбец;

Запросы к базе данных возвращают результат в виде таблиц.

Индексы

Определение

Индекс - это отсортированный список значений полей, предназначенный для ускорения поиска в базе данных.

Интересны, как правило, не сами индексы, а уникальные индексы.

Уникальный индекс представляет собой список значений, в котором каждое значение уникально. К примеру, в таблице базы данных, содержащей паспортные данные уникальный индекс можно создать для поля "номер паспорта", поскольку каждый такой номер является единственным в своём роде. А вот дата рождения уже не уникальна, поэтому индекс по полю "Дата рождения" не может быть уникальным. Возвращаясь к нашей базе данных Shop, нужно заметить, что дата добавления заказа также не является

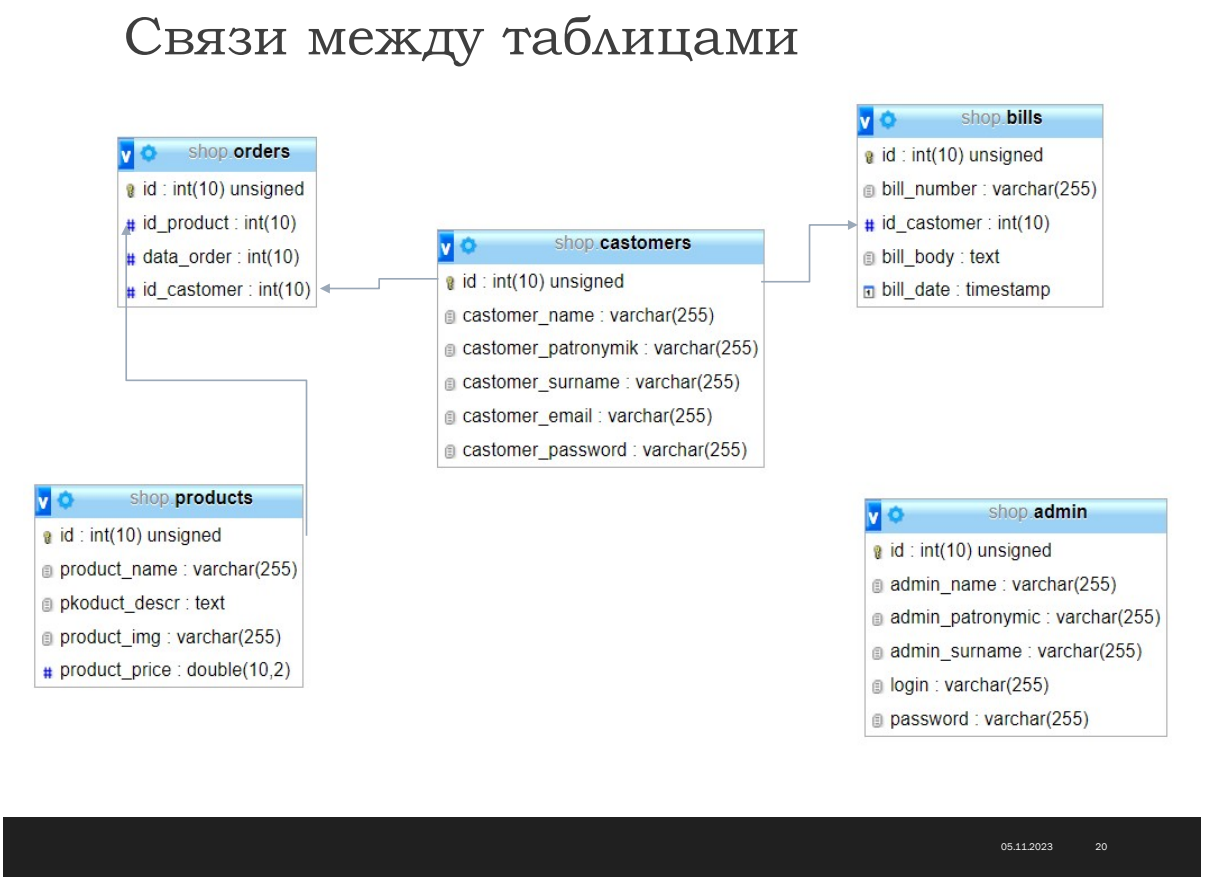
уникальной, так как несколько покупателей могут создать свои заказы одновременно.

Первичные ключи

Первичный ключ (primary key) представляет собой один из примеров уникальных индексов и применяется для уникальной идентификации записей таблицы. Никакие из двух записей таблицы не могут иметь одинаковых значений первичного ключа. Первичный ключ обычно сокращенно обозначают как PK (primary key).

Как мы уже говорили, в реляционных базах данных практически всегда разные таблицы логически связаны друг с другом. Первичные ключи как раз используются для однозначной организации такой связи.

К примеру, в базе данных Shop таблицы products и orders связаны между собой следующим образом:



Первичным ключом таблицы productd является id, а ссылочным ключом в таблице orders является id_products. Таким образом в таблице заказов orders

Способы задания первичного ключа

По способу задания первичных ключей различают **логические** (естественные) ключи и **суррогатные** (искусственные).

Мы создаём суррогатные ключи с именем id

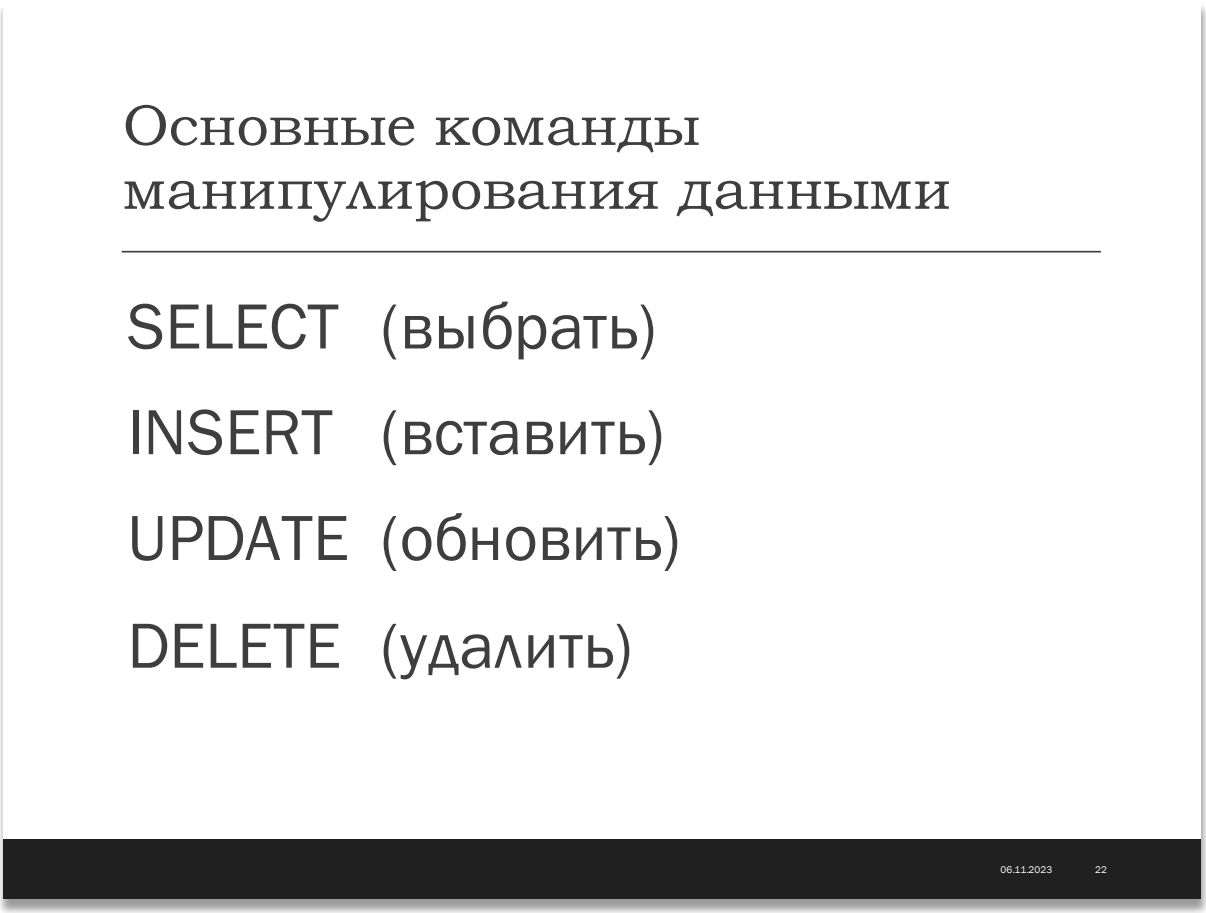
Первичному ключу можно присвоить атрибут auto_increment, позволяющий автоматически генерировать уникальный ключ, если его тип является целочисленным. При вставке записи в базу данных значение ключа выставляется равным нулю, MySQL автоматически вычисляет максимальный номер первичного ключа, увеличивает его на единицу и присваивает это значение первичному ключу новой записи.

Язык SQL

Структурированный язык запросов SQL позволяет производить различные операции с базами данных: создавать таблицы, помещать, обновлять и удалять из них данные, производить запросы из таблиц и т.д. Далее мы последовательно рассмотрим все эти операторы.

Команды SQL

Язык манипулирования данными используется, как это следует из его названия, для манипулирования данными в таблицах баз данных.



Он состоит из 4 основных команд:

SELECT (выбрать)

INSERT (вставить)

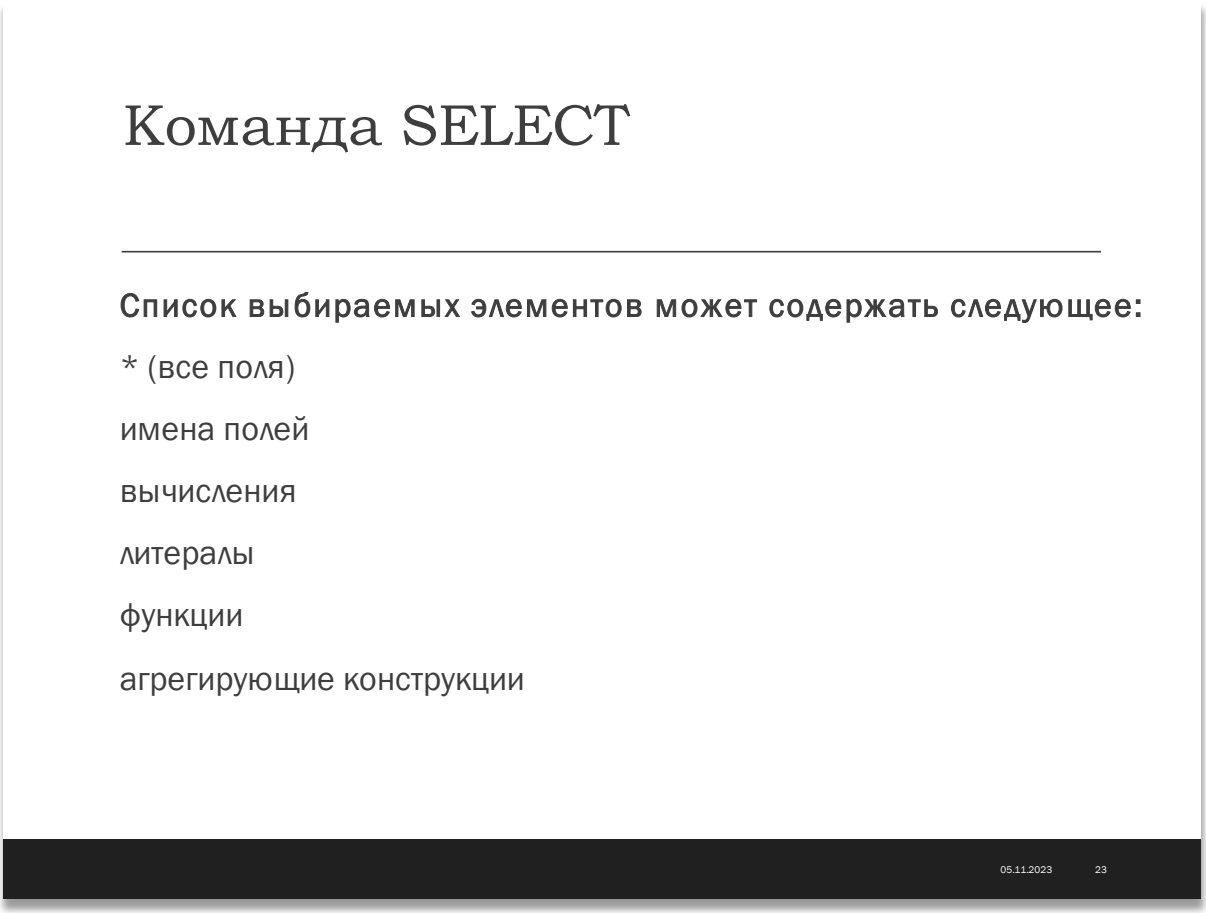
UPDATE (обновить)

DELETE (удалить)

Есть ещё команды языка определения данных, но мы его сейчас рассматривать не будем, так как они автоматически формируются и выполняются WEB-приложением phpMyAdmin/

Команды SQL не чувствительны к регистру, но традиционно они набираются прописными буквами.

Команда SELECT



Список выбираемых элементов может содержать следующее:

* (все поля)

имена полей

вычисления

литералы

функции

агрегирующие конструкции

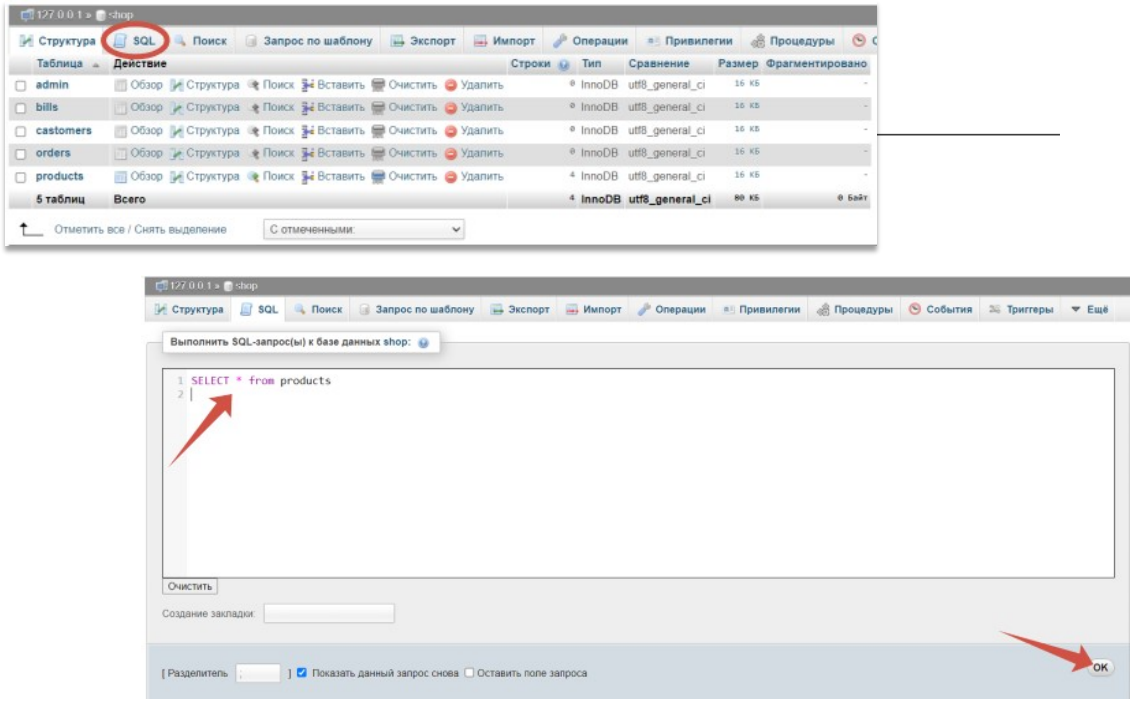
Мы пока будем использовать первые две позиции.

Команда SELECT

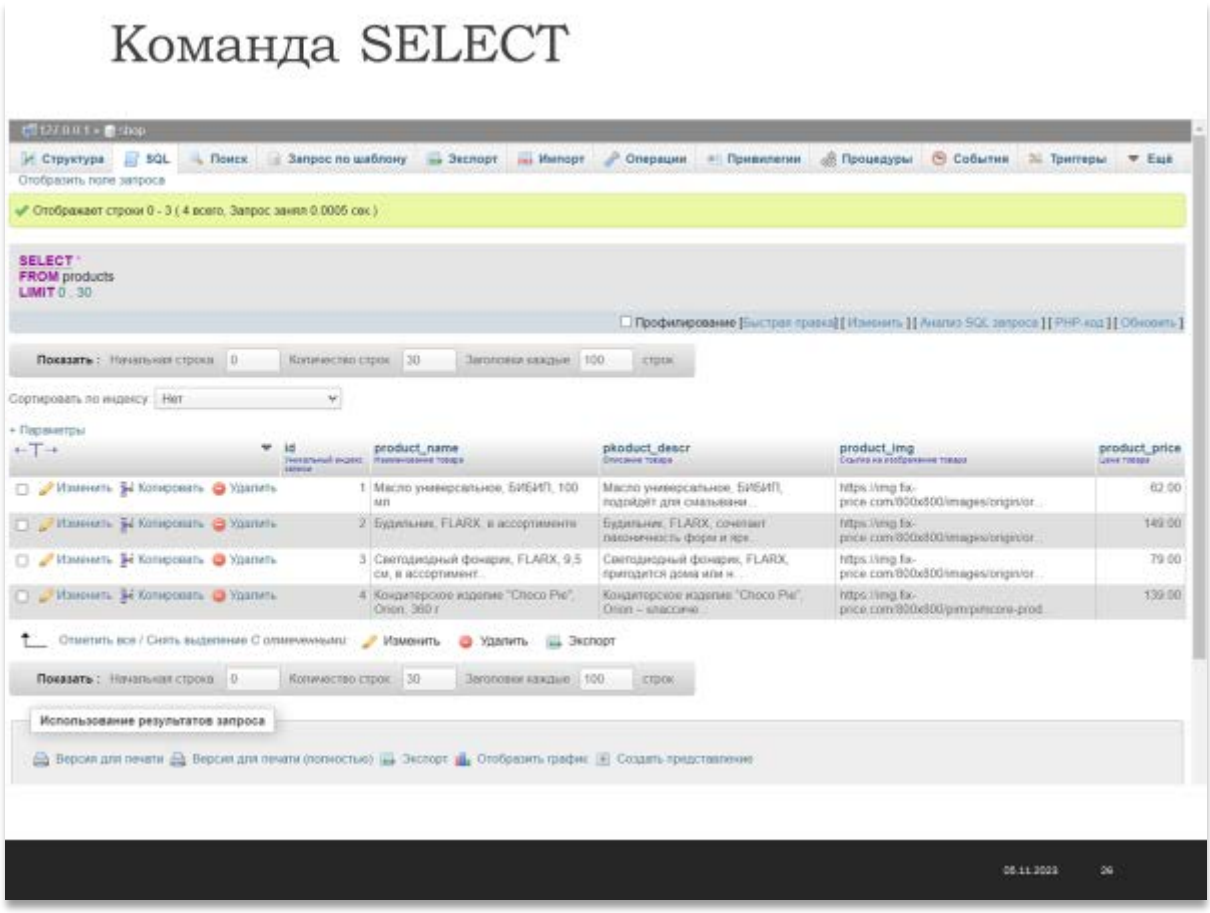
```
SELECT * from products
```

Вобьём эту команду в поле SQL в phpMyAdmin, предварительно кликнув по соответствующей вкладке, и нажмём ОК:

Команда SELECT

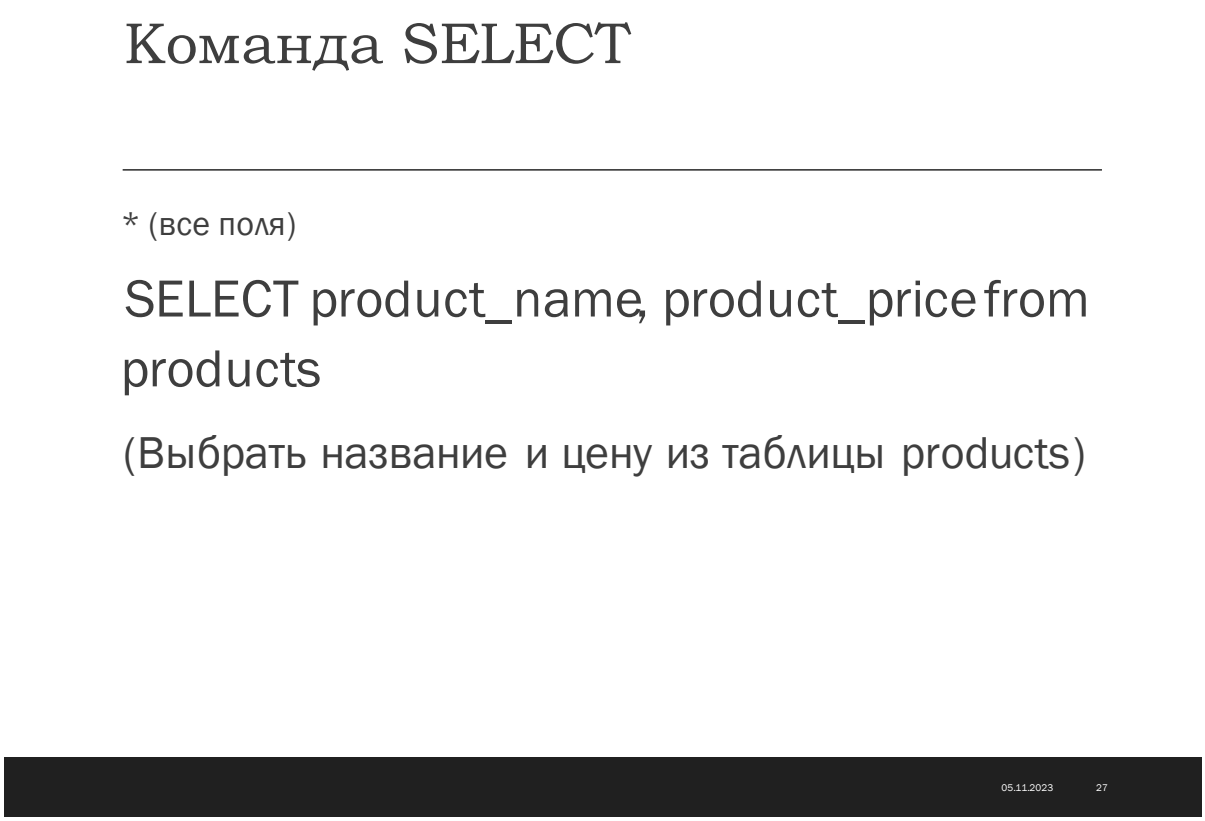


Получим следующий результат:

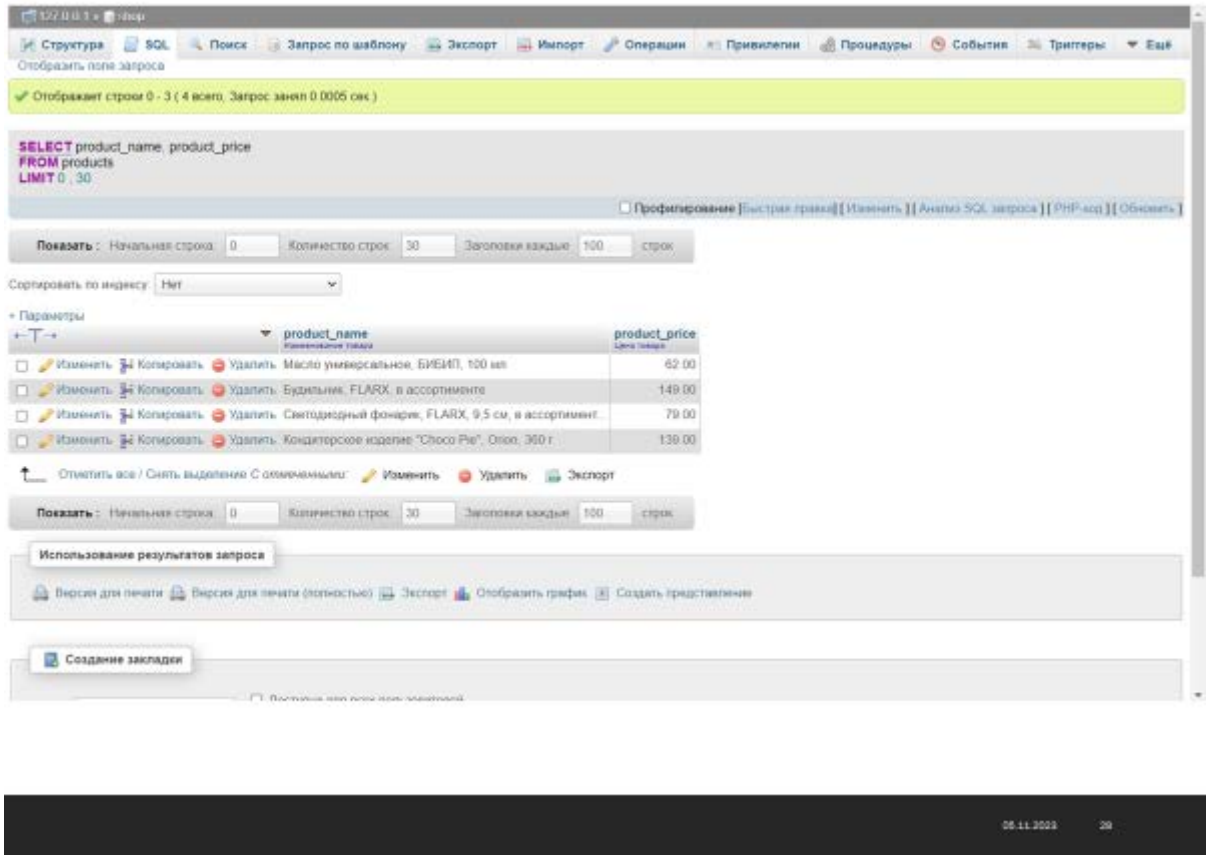


Теперь применим имена полей:

```
SELECT product_name, product_price from products
```



Получим:

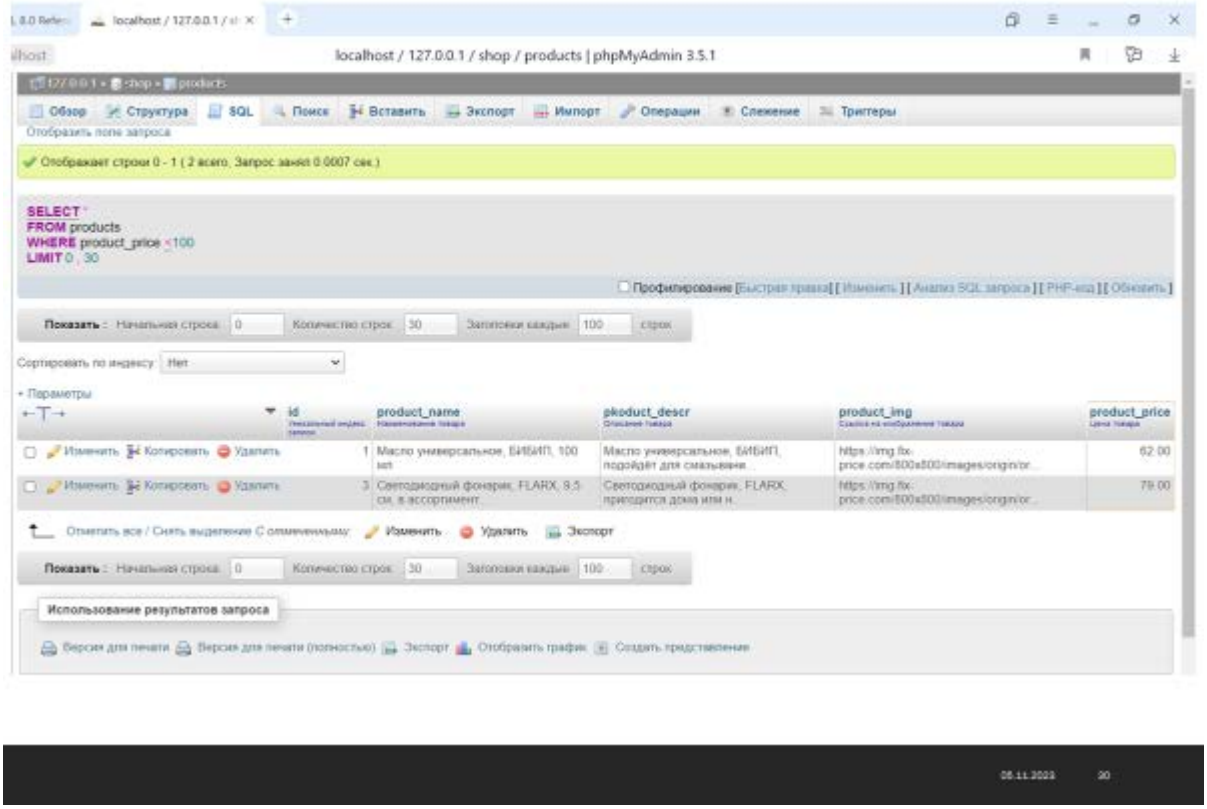


Выбор с условием

Предположим, что мы хотим получить список товаров цена которых меньше 100 рублей.

```
SELECT * from products where product_price<100
```

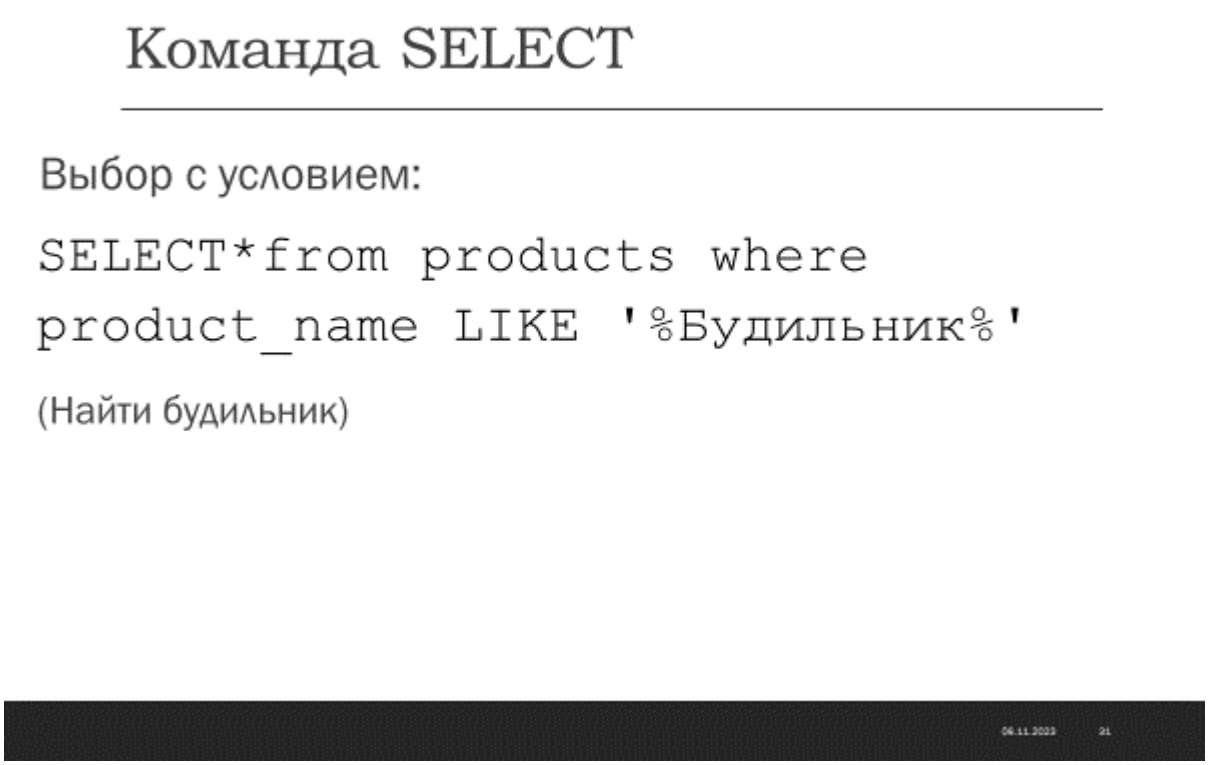
Вставим этот запрос в поле формы SQL phpMyAdmin,
Получим результат:



Теперь предположим, что мы хотим найти товар Будильник.

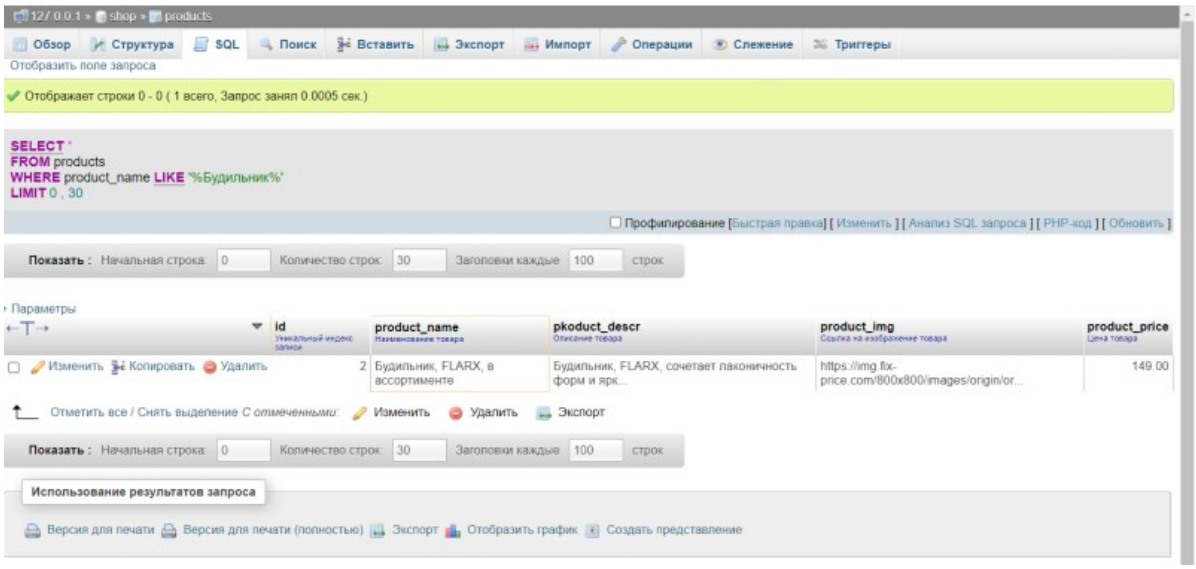
Запрос на выборку будет выглядеть так:

```
SELECT * from products where product_name LIKE '%Будильник%'
```



В результате отыщутся записи, в которых в поле product_name содержится строка Будильник.

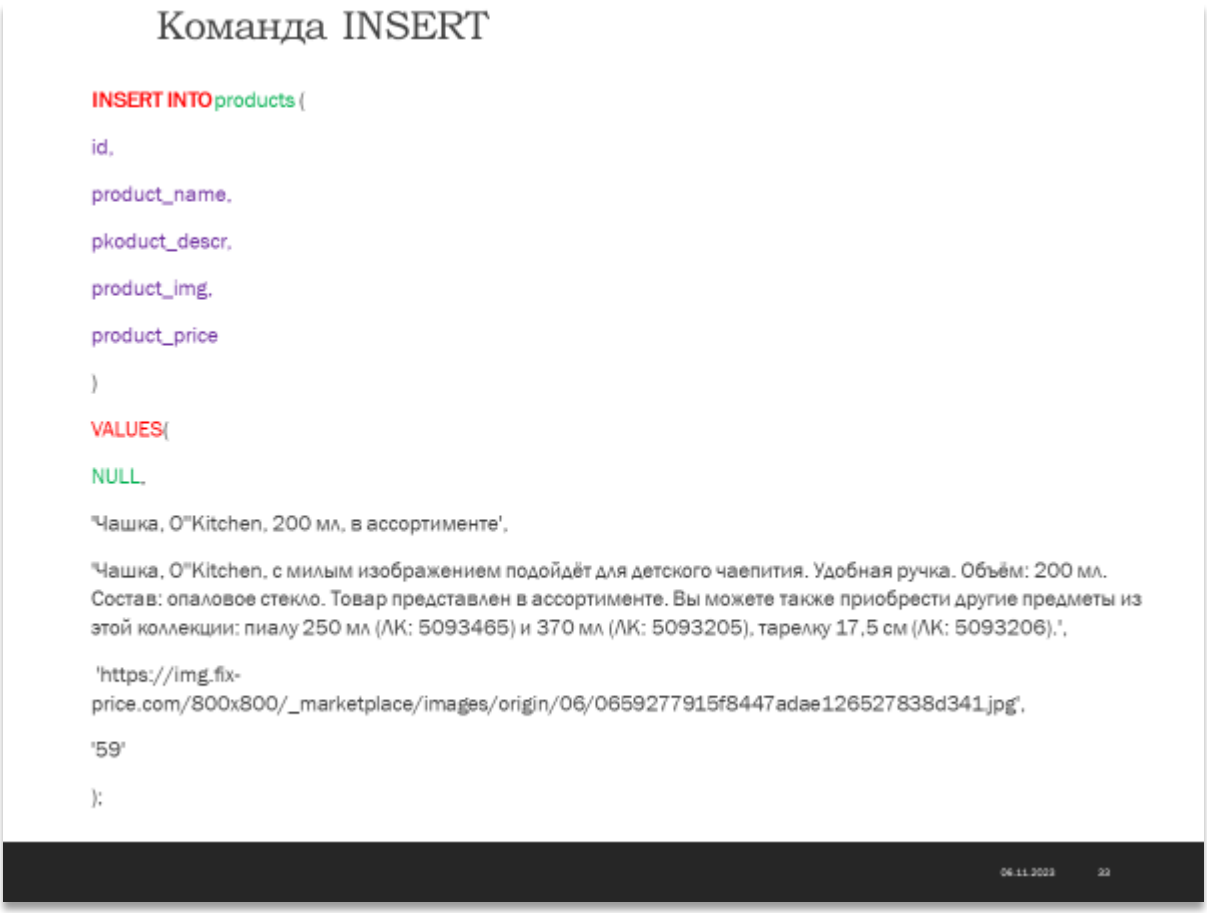
Знаки % означают любые символы перед и до искомой строки.



Команда INSERT

Данная команда используется для добавления записей в таблицу

```
INSERT INTO products (
id,
product_name,
pkoduct_descr,
product_img,
product_price
)
VALUES(
NULL,
'Чашка, O''Kitchen, 200 мл, в ассортименте',
'Чашка, O''Kitchen, с милым изображением подойдёт для детского чаепития.
Удобная ручка. Объём: 200 мл. Состав: опаловое стекло. Товар представлен в
ассортименте. Вы можете также приобрести другие предметы из этой коллекции:
пиалу 250 мл (ЛК: 5093465) и 370 мл (ЛК: 5093205), тарелку 17,5 см (ЛК:
5093206).',
 'https://img.fix-
price.com/800x800/_marketplace/images/origin/06/0659277915f8447adae126527838d
341.jpg',
'59'
);
```



Здесь в первых круглых скобках перечисляются имена полей, а во вторых круглых скобках перечисляются их значения в той же последовательности. Заметьте, что полю id присваивается значение NULL, потому что id имеет свойство авто инкрементирования. Все строковые значения заключаются в кавычки.

Команда UPDATE

Данная команда используется для изменения записи

```
UPDATE products
SET product_img = 'https://img.fix-
price.com/800x800/_marketplace/images/origin/71/7110fdb94724a3d10e94fea7d7084
2f0.jpg'
WHERE id =6;
```

Здесь в предложении SET конкретному полю присваивается новое значение.

Условие WERE в данном примере определяет уникальный индекс записи, в которой нужно сделать изменение.

Если условие WERE не указывать, то изменениям подвергнутся все записи таблицы.

В предложении SET можно указать необходимые поля и присваиваемые им значения, разделенные запятыми.

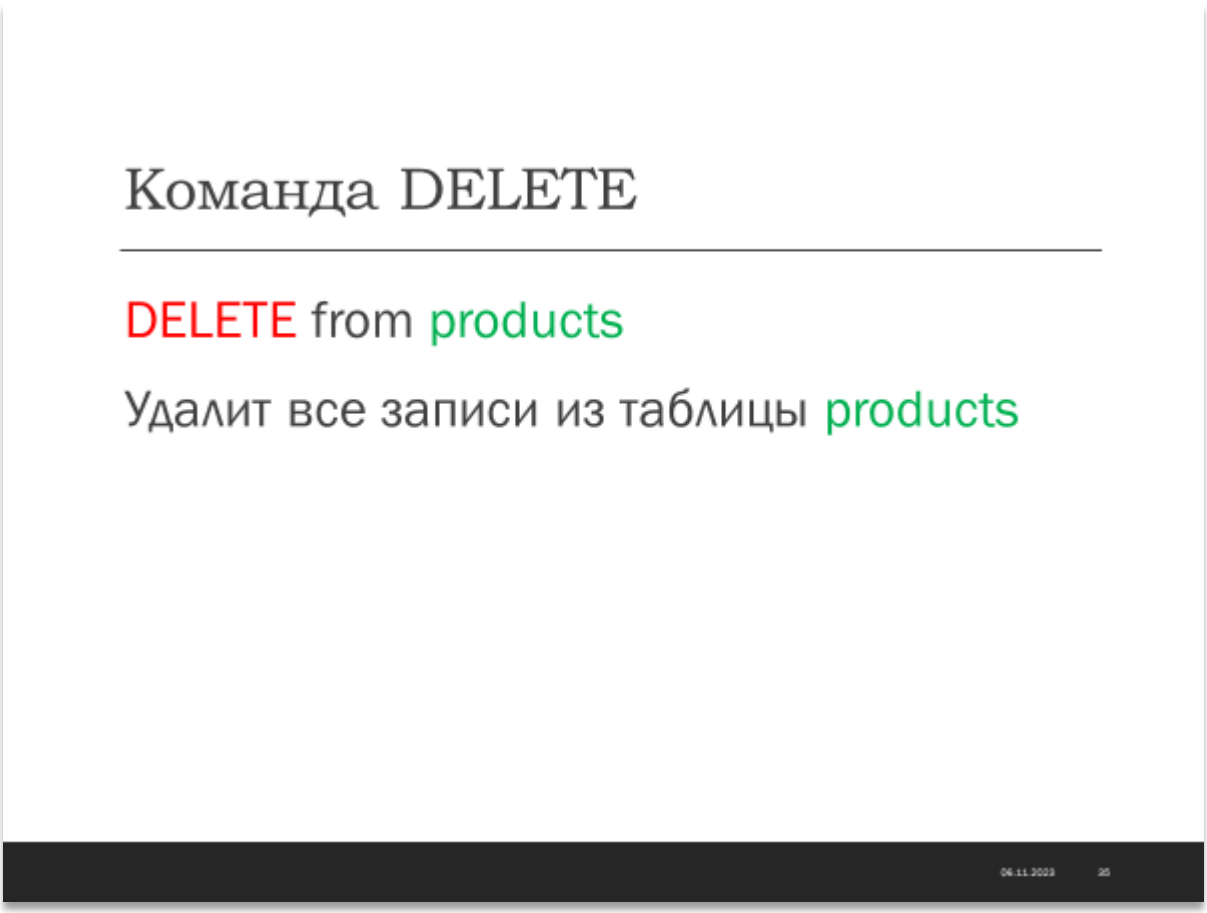
Команда DELETE

Команда DELETE удаляет записи из таблицы

Например команда

```
DELETE from products
```

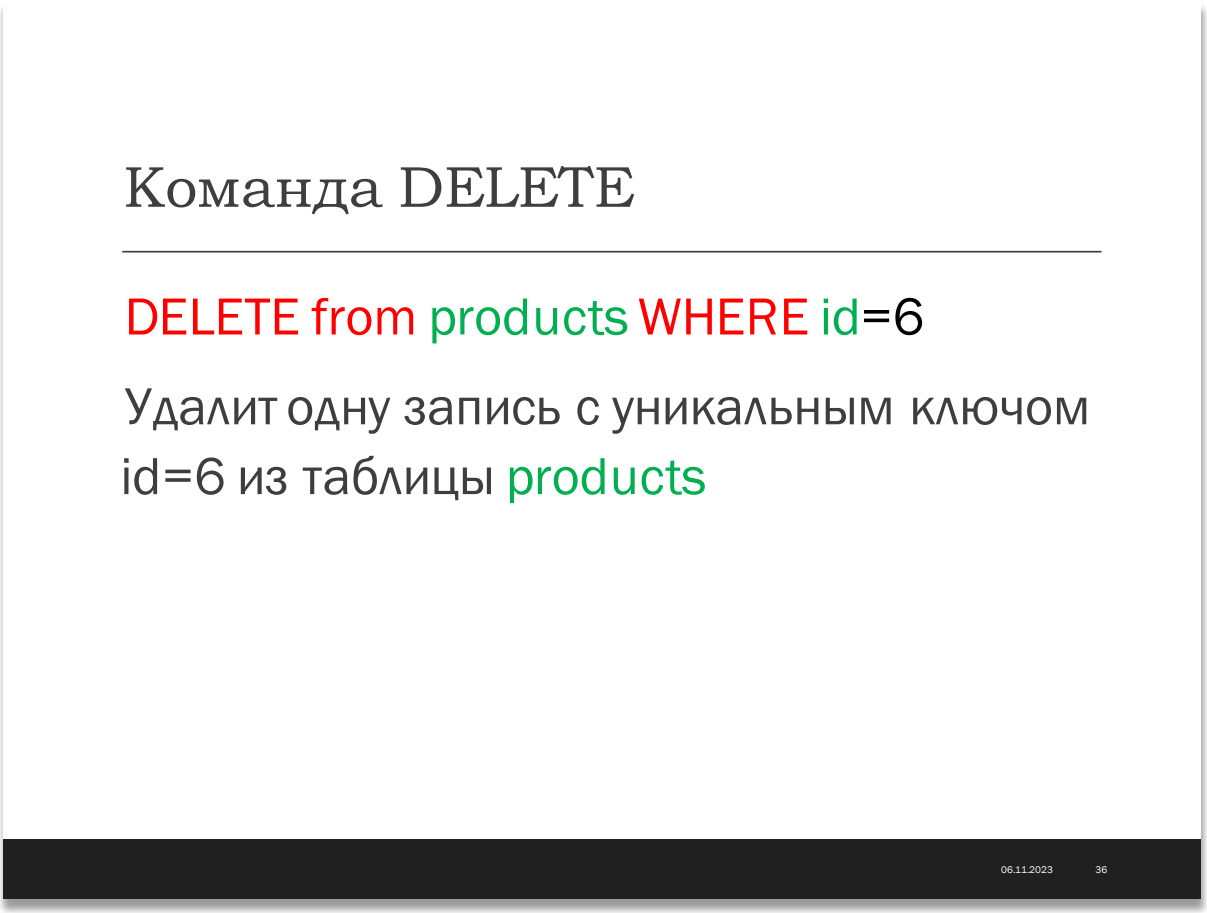
Удалит все записи из таблицы products.



На практике чаще всего удаляются не все, а конкретные определённые записи. Поэтому используется условие WHERE.

```
DELETE from products WHERE id=6
```

Удалит одну запись с уникальным ключом id=6 из таблицы products

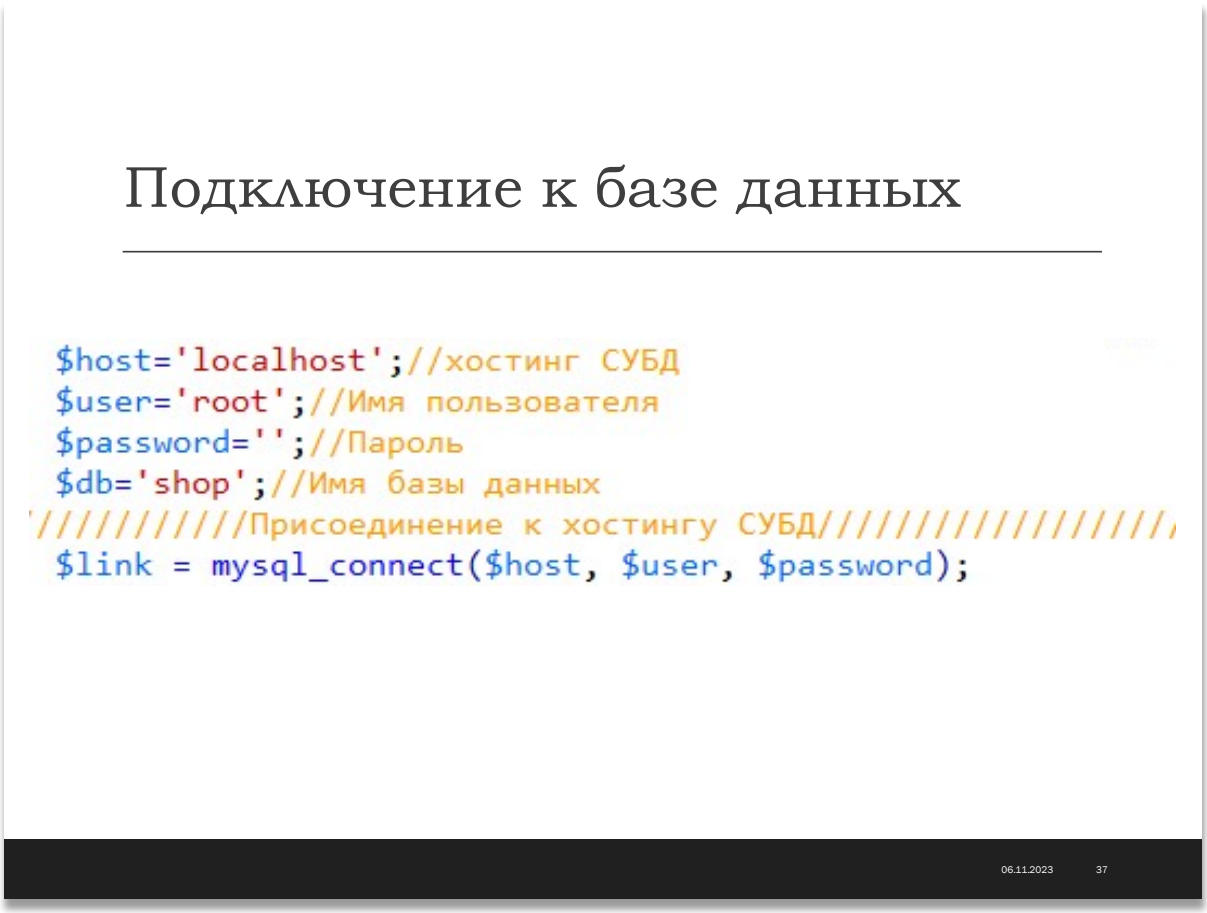


Взаимодействие с базой данных из сценариев PHP

Подключение к базе данных

```
$host='localhost';//хостинг СУБД
	$user='root';//Имя пользователя
	$password='';//Пароль
	$db='shop';//Имя базы данных
//Присоединение к хостингу СУБД//
	$link = mysql_connect($host, $user, $password);
```

Для подключения к серверу СУБД используется функция mysql_connect().



Параметрами этой функции являются: URL хоста, имя пользователя и пароль. Все данные для подключения предоставляются хостинг-провайдерами.

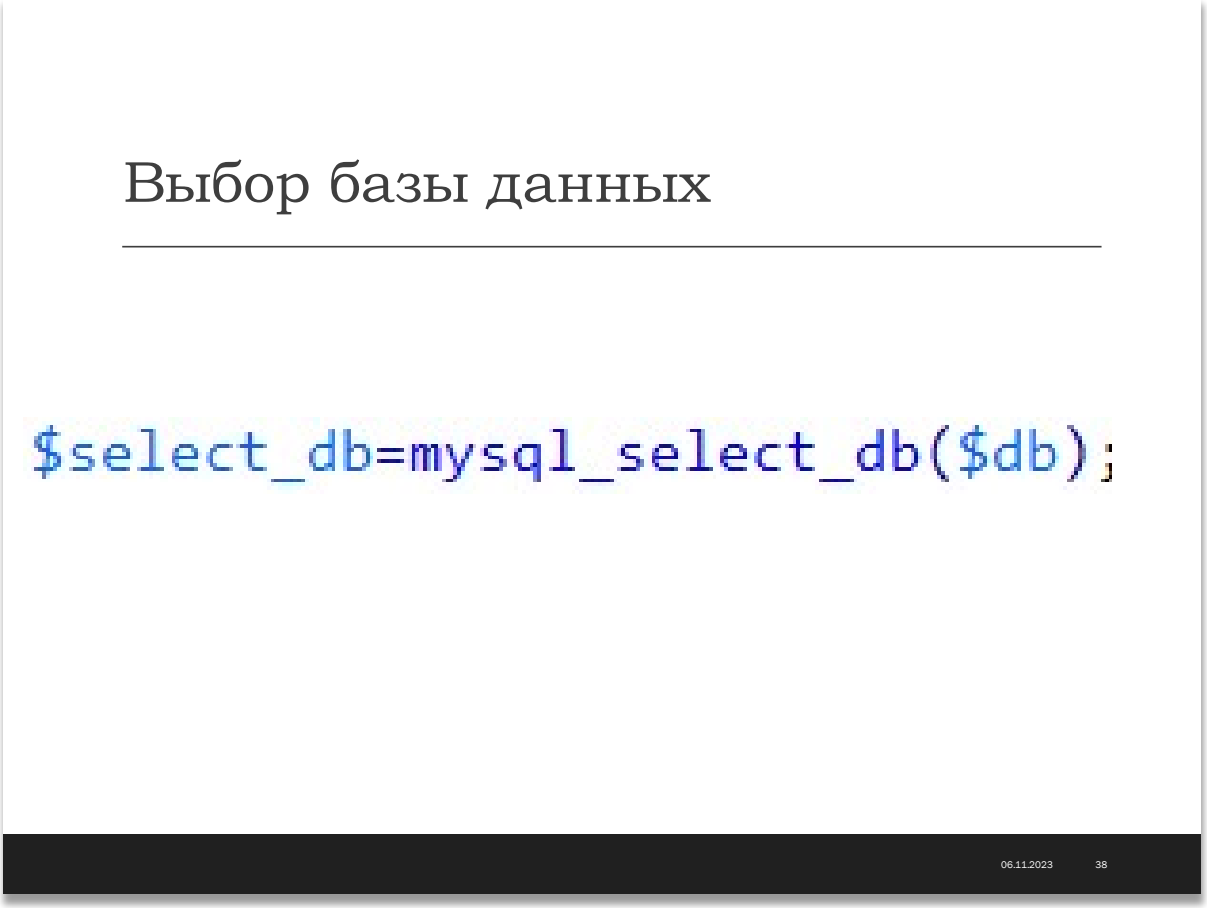
В Денвере по умолчанию используются:

- Имя хоста: localhost;
- Имя пользователя: root;
- Пароль: пустая строка.

После подключения необходимо произвести выбор базы данных:

```
$select_db=mysql_select_db($db);
```

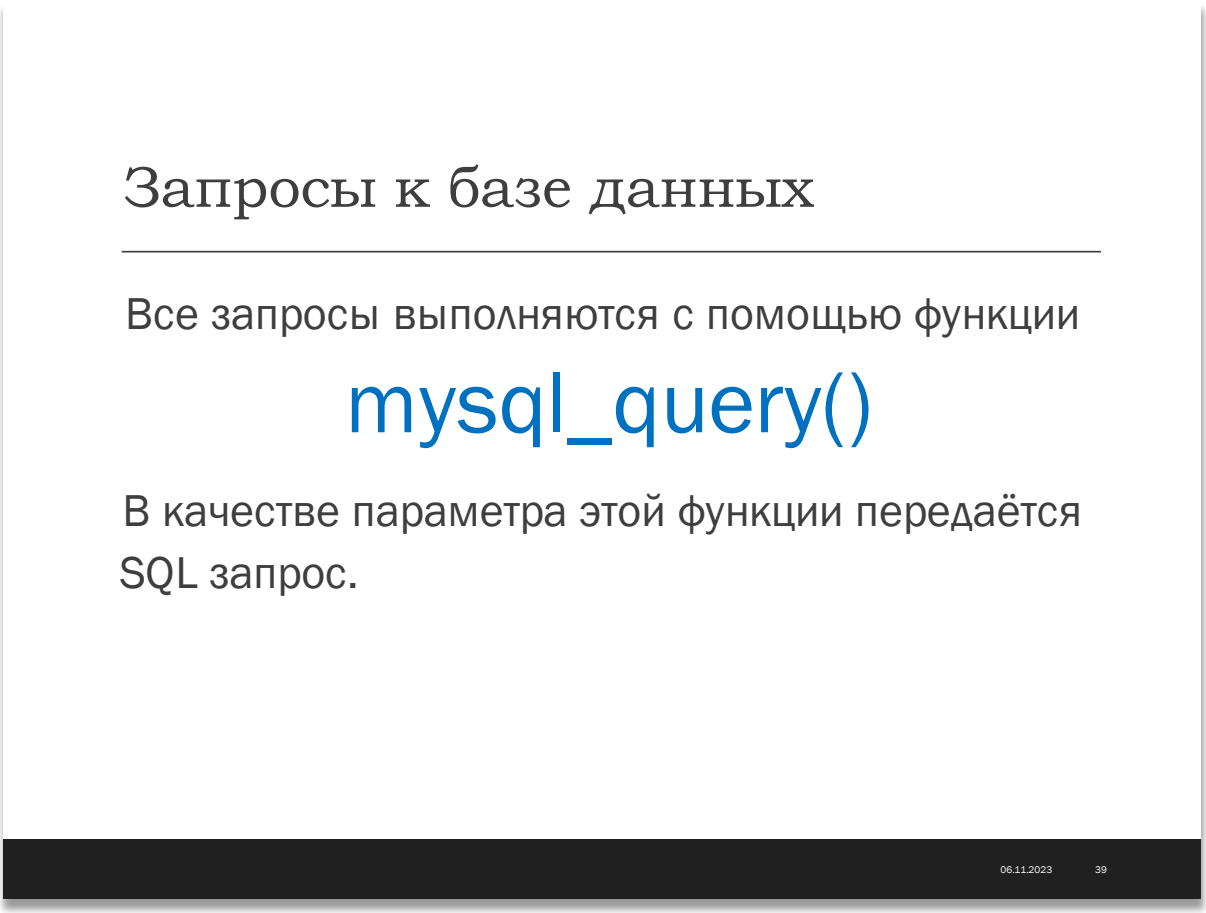
Для этого используем функцию mysql_select_db(), параметром которой является имя базы данных.



Выполнение запросов к БД

Все запросы выполняются с помощью функции mysql_query().

В качестве параметра этой функции передаётся SQL запрос.



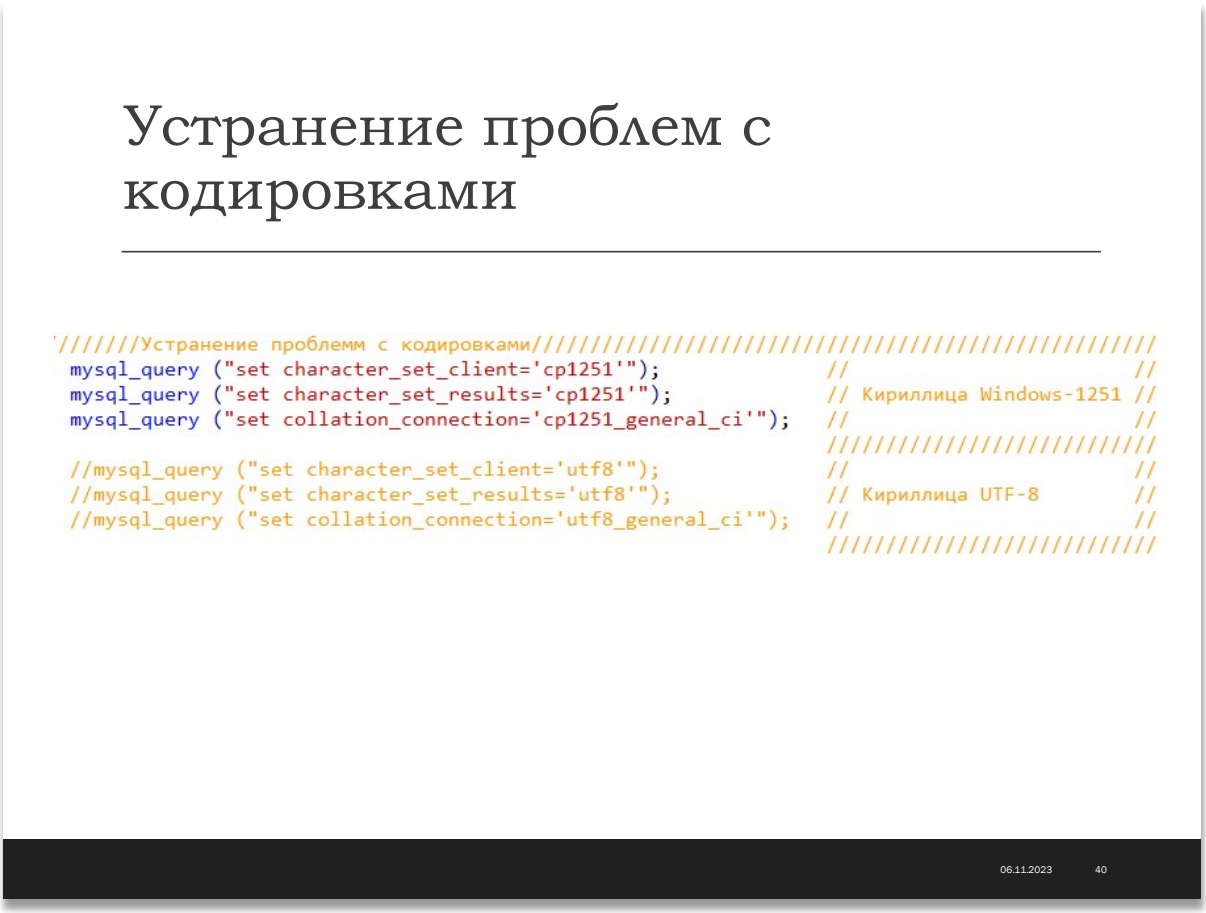
Устранение проблем с кодировками

Если в результате выполнения результатов запросов браузер некорректно отображает кириллицу, исправить положение помогают следкющие установочные запросы:

```
mysql_query ("set character_set_client='cp1251'");
mysql_query ("set character_set_results='cp1251'");
mysql_query ("set collation_connection='cp1251_general_ci'");
//mysql_query ("set character_set_client='utf8'");
//mysql_query ("set character_set_results='utf8'");
//mysql_query ("set collation_connection='utf8_general_ci'");
```

Эти запросы устанавливают необходимую кодировку. Их рекомендуется разместить сразу после выбора базы данных. В данном примере установлена кодировка Windows-1251.

Если необходимо установить кодировку utf-8, необходимо будет закомментировать первые три строки и убрать комментарии последних трёх строк.



Выборка всех записей SELECT

```
//Формирование SQL запроса на выборку всех записей//
$query='SELECT * from products';
//Отправка запроса//
$result=mysql_query($query);
```

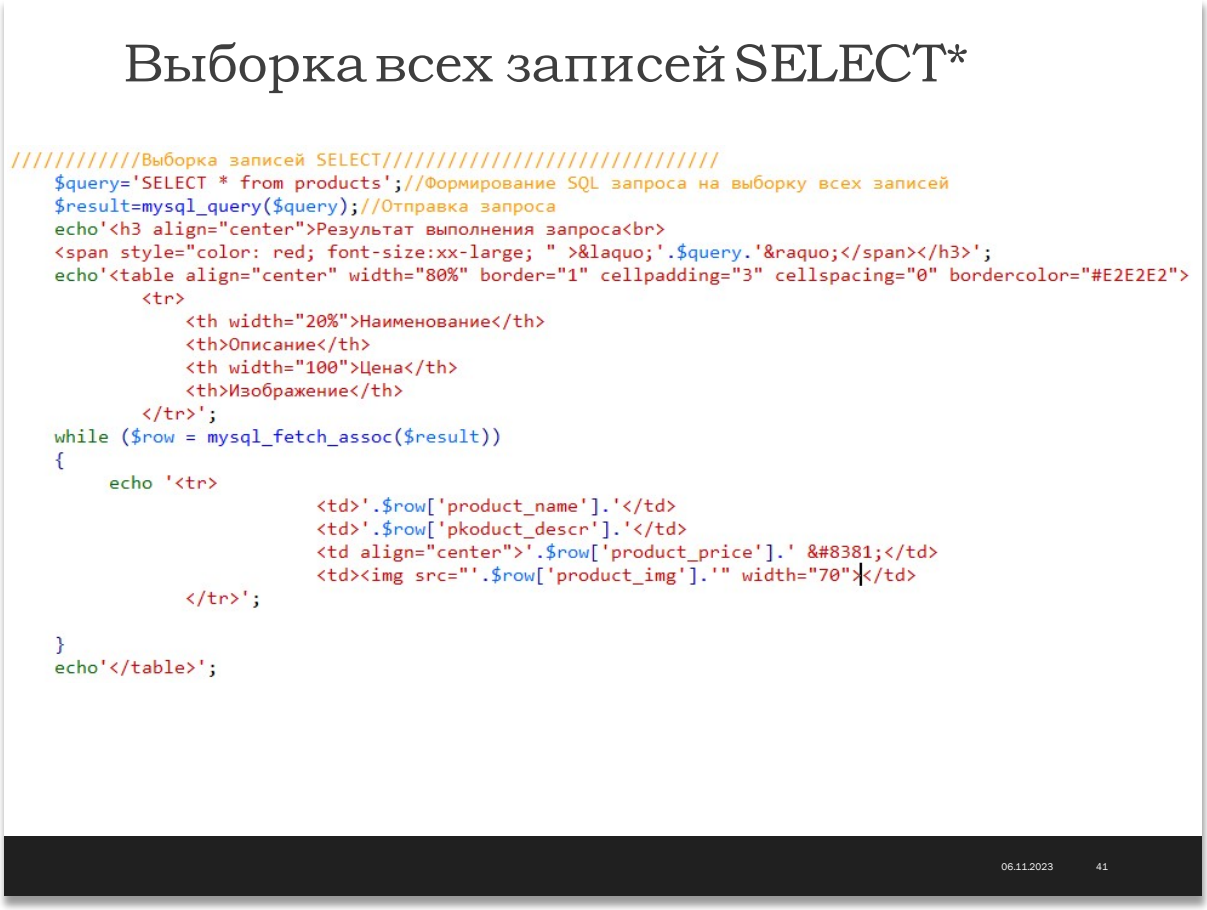
Этот код запрашивает набор всех записей из таблицы product.

Чтобы получить доступ к записям, можно воспользоваться функцией mysql_fetch_assoc(). Функция mysql_fetch_assoc возвращает ассоциативный массив, который содержит следующую строку результирующего набора в виде пары "имя поля" - "значение".

Так как строк в таблице может быть несколько, необходимо использовать цикл, например while:

```
echo'<table align="center" width="80%" border="1" cellpadding="3"
cellspacing="0" bordercolor="#E2E2E2">
                  <tr>
                        <th width="20%">Наименование</th>
                        <th>Описание</th>
                        <th width="100">Цена</th>
                        <th>Изображение</th>
                  </tr>';
      while ($row = mysql_fetch_assoc($result))
      {
             echo '<tr>
                                          <td>'.$row['product_name'].'</td>
      <td>'.$row['pkoduct_descr'].'</td>
```

```
      <td>'.$row['product_price'].' &#8381;</td>
      <td><img src="'.$row['product_img'].'" width="70"></td>
      </tr>';
      }
      echo'</table>';
```



Добавление записи в таблицу

В реальном проекте удобнее всего для добавления записей использовать форму.

Давайте напишем сценарий, позволяющий добавлять товары в таблицу products.

Для начала создадим форму, аналогичную той, которую использовали в нашем интернет-магазине:

```
<?
//////////////отображение формы заполнения прайс-листа///////
echo'<form action="?p=1" method="post">';
echo'<table align="center" width="600" border="1" cellpadding="3"
cellspacing="0" bordercolor="#E2E2E2">
<tr>
      <td>Наименование товара</td>
      <td> <input  type="text" name="product_name"></td>
</tr>
<tr>
      <td>Цена товара </td>
      <td><input  type="text" name="product_price"></td>
</tr>
```

```
<tr>
      <td>Описание товара</td>
<td><textarea name="product_descr" cols="30" rows="5"></textarea></td>
</tr>
<tr>
      <td>Ссылка на изображение товара</td>
      <td><input  type="text" name="product_img"></td>
</tr>
<tr>
      <td colspan="2"><p align="center"><input type="submit"
name="Добавить"></p></td>
</tr>
</table>';
echo'</form>';
?>
```

Разместим этот код в отдельном файле, например в test_db_3.php.

Теперь создадим отдельный файл db_include.php, который будет отвечать за подключение к базе данных. Запишем туда знакомый нам код:

```
<?
      header('Content-Type: text/html; charset=windows-1251');
      $host='localhost';//хостинг СУБД
      $user='root';//Имя пользователя
      $password='';//Пароль
      $db='shop';//Имя базы данных
/////Присоединение к хостингу СУБД///
      $link = mysql_connect($host, $user, $password);
      if(!$link)
      echo '<h4 font-color="red">Ошибка соединения</h4>';
      else/////Выбор базы данных///
      $select_db=mysql_select_db($db);
      if(!$select_db)
      echo'<h4 font-color="red">Ошибка соединения с базой данных</h4>';
///Устранение проблемм с кодировками///
      mysql_query ("set character_set_client='cp1251'");
      mysql_query ("set character_set_results='cp1251'");
      mysql_query ("set collation_connection='cp1251_general_ci'");
      //mysql_query ("set character_set_client='utf8'");
      //mysql_query ("set character_set_results='utf8'");
      //mysql_query ("set collation_connection='utf8_general_ci'");
?>
```

Файл include_db.php

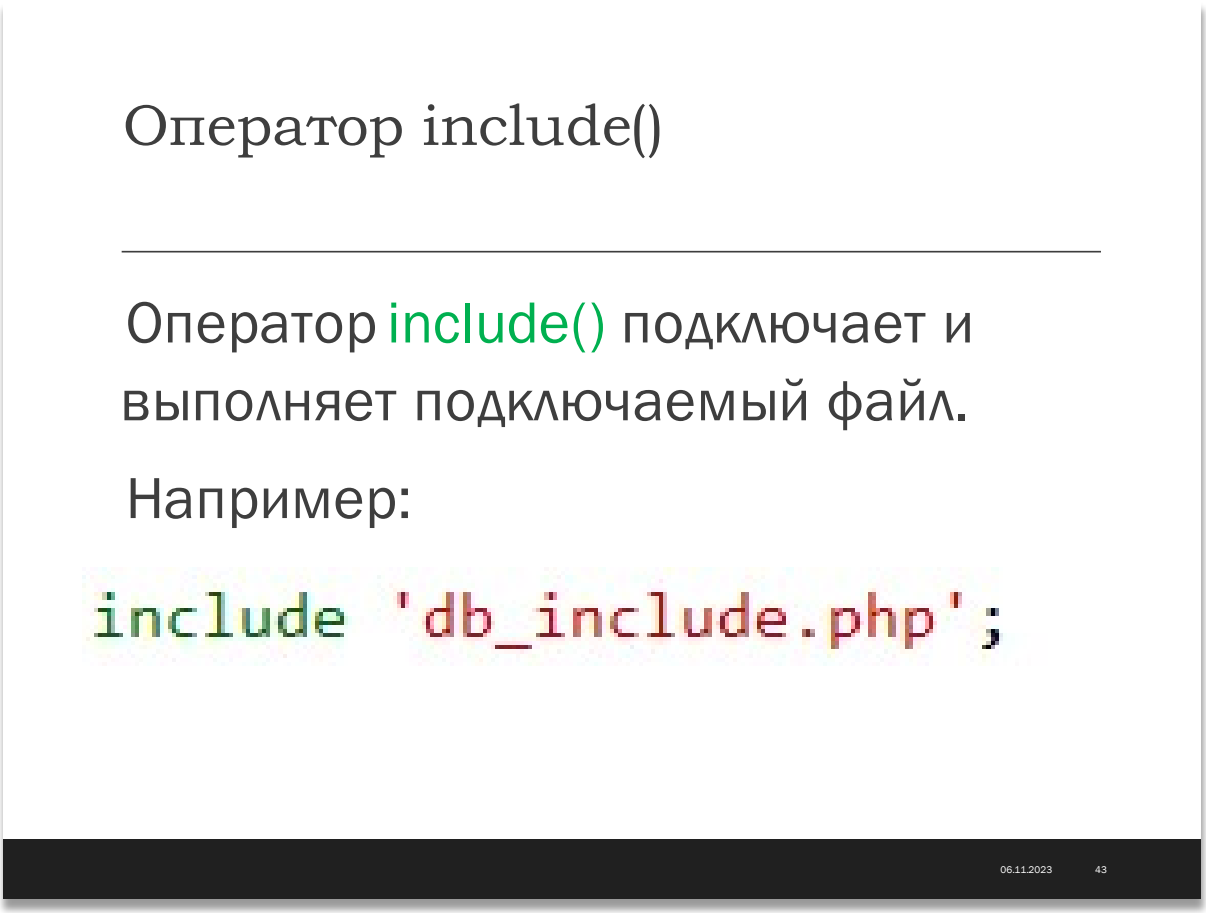
```php
<?
    header('Content-Type: text/html; charset=windows-1251');
    $host='localhost';//хостинг СУБД
    $user='root';//Имя пользователя
    $password='';//Пароль
    $db='shop';//Имя базы данных
/////Присоединение к хостингу СУБД///
    $link = mysql_connect($host, $user, $password);
    if(!$link)
    echo '<h4 font-color="red">Ошибка соединения</h4>';
    else/////Выбор базы данных///
    $select_db=mysql_select_db($db);
    if(!$select_db)
    echo'<h4 font-color="red">Ошибка соединения с базой данных</h4>';
///Устранение проблемм с кодировками///
    mysql_query ("set character_set_client='cp1251'");
    mysql_query ("set character_set_results='cp1251'");
    mysql_query ("set collation_connection='cp1251_general_ci'");

    //mysql_query ("set character_set_client='utf8'");
    //mysql_query ("set character_set_results='utf8'");
    //mysql_query ("set collation_connection='utf8_general_ci'");
?>
```

06.11.2023 42

Оператор include()

Оператор include() подключает и выполняет подключаемый файл, например:

```
include 'db_include.php';
```



Вставим эту строчку в начало файла db_3.php с формой сразу под дескриптором <?

```
<?
include 'db_include.php';
////////////отображение формы заполнения прайс-листа///////
echo'<form action="?p=1" method="post">';
echo'<table align="center" width="600" border="1" cellpadding="3"
cellspacing="0" bordercolor="#E2E2E2">
<tr>
      <td>Наименование товара</td>
      <td> <input  type="text" name="product_name"></td>
</tr>
<tr>
      <td>Цена товара </td>
      <td><input  type="text" name="product_price"></td>
</tr>
<tr>
      <td>Описание товара</td>
<td><textarea name="product_descr" cols="30" rows="5"></textarea></td>
</tr>
<tr>
      <td>Ссылка на изображение товара</td>
      <td><input  type="text" name="product_img"></td>
</tr>
<tr>
      <td colspan="2"><p align="center"><input type="submit"
name="Добавить"></p></td>
</tr>
</table>';
echo'</form>';
```

```
?>
```

Подключение файла db_include.php

```
<?
 include 'db_include.php';
///отображение формы заполнения прайс-листа////
    echo'<form action="?p=1" method="post">';
    echo'<table align="center" width="600" border="1" cellpadding="3" cellspacing="0" bordercolor="#E2E2E2">
            <tr>
                <td>Наименование товара</td>
                <td> <input  type="text" name="product_name"></td>
            </tr>
            <tr>
                <td>Цена товара </td>
                <td><input  type="text" name="product_price"></td>
            </tr>
            <tr>
            <td>Описание товара</td>
            <td><textarea name="product_descr" cols="30" rows="5"></textarea></td>
            </tr>
            <tr>
            <td>Ссылка на изображение товара</td>
            <td><input  type="text" name="product_img"></td>
            </tr>
            <tr>
            <td colspan="2"><p align="center"><input type="submit" name="Добавить"></p></td>
            </tr>
            </table>
            ';
    echo'</form>';
?>
```


Теперь добавим сюда строки, отвечающие за приём параметров и данных из формы, а также обработчик формы, отвечающий за добавление новой записи в базу данных в таблицу products.

```
/////Приём данных из формы///////////
      $product_name=$_POST['product_name'];
      $product_price=$_POST['product_price'];
      $product_descr=$_POST['product_descr'];
      $product_img=$_POST['product_img'];
      ///////Приём параметров сценария///////
      $p=$_GET['p'];
      //Обработчик формы добавления записи//
      if($p==1)
      {
            $query="INSERT INTO products (
                  id,
                  product_name,
                  pkoduct_descr,
                  product_img,
                  product_price
                  ) VALUES(
                        NULL,
                        '$product_name',
                        '$product_descr',
                        '$product_img',
                        '$product_price'
                        )";
            $result=mysql_query($query);//Отправка запроса
            if(!$result)
```

```
            {
                  echo'<p>Ошибка выполнения запроса</p>';
                  echo $query;
            }
      }
```

```
/////Приём данных из формы//////////
   $product_name=$_POST['product_name'];
   $product_price=$_POST['product_price'];
   $product_descr=$_POST['product_descr'];
   $product_img=$_POST['product_img'];
   ///////Приём параметров сценария///////
   $p=$_GET['p'];
   //Обработчик формы добавления записи//
   if($p==1)
   {
       $query="INSERT INTO products (
           id,
           product_name,
           pkoduct_descr,
           product_img,
           product_price
           ) VALUES(
               NULL,
               '$product_name',
               '$product_descr',
               '$product_img',
               '$product_price'
               )";
       $result=mysql_query($query);//Отправка запроса
       if(!$result)
       {
           echo'<p>Ошибка выполнения запроса</p>';
           echo $query;
       }
   }
```

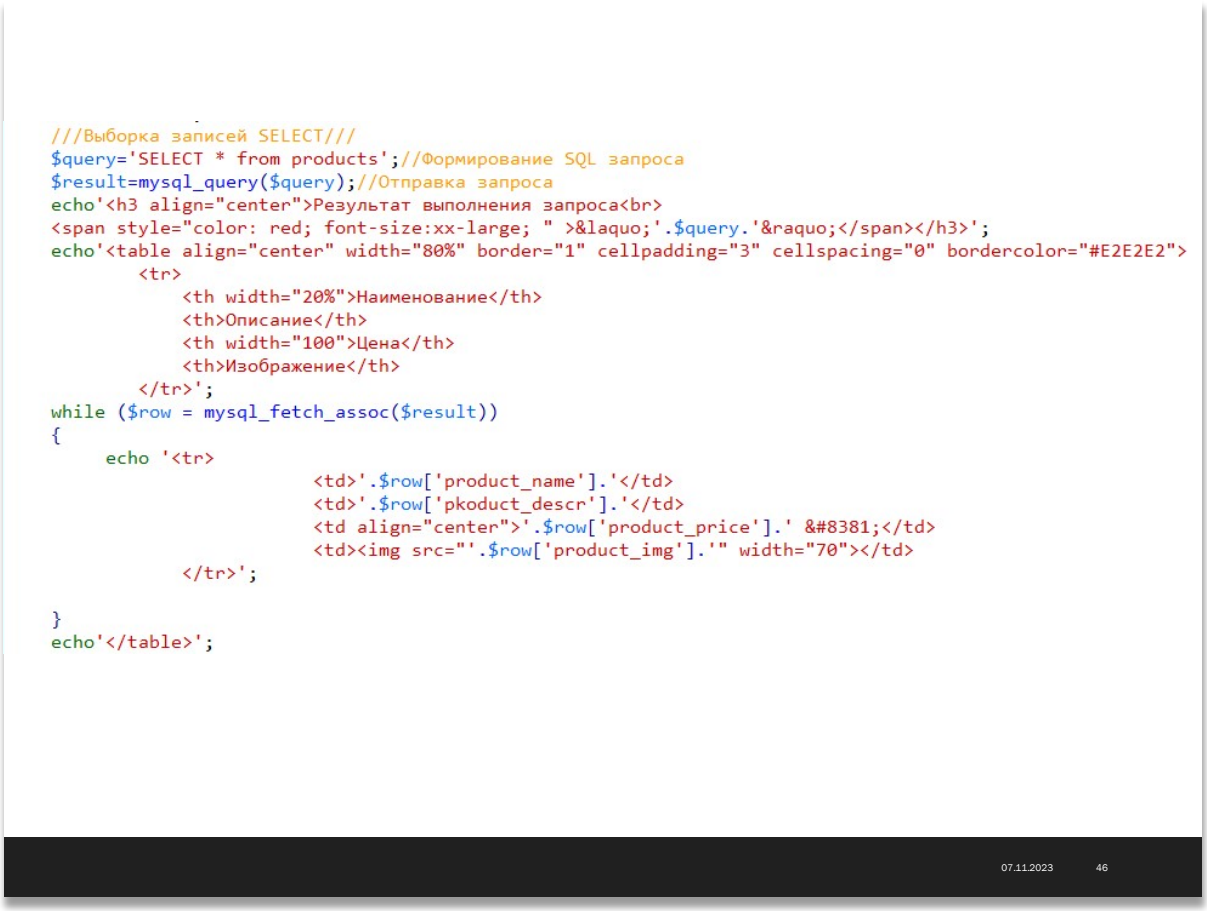
07.11.2023 45

Разместим этот код в файле db_3.php под строчкой include 'db_include.php';.

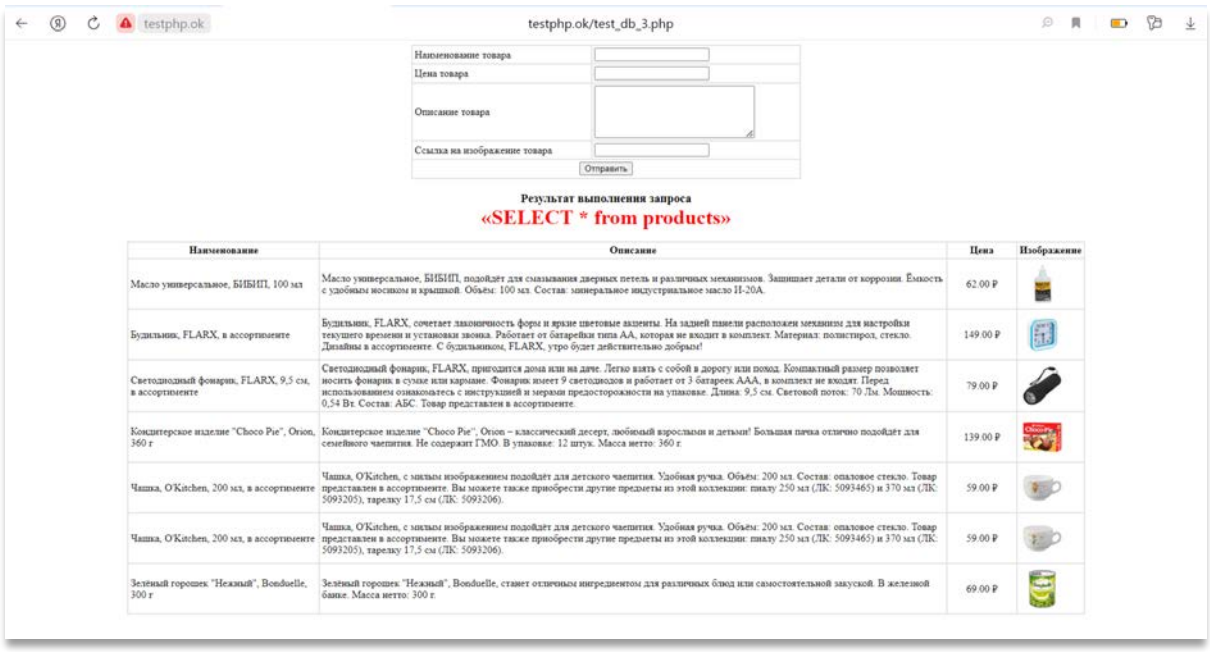
Теперь прямо под формой вставим уже знакомый нам код отображения нашей таблицы products:

```
///Выборка записей SELECT///
      $query='SELECT * from products';//Формирование SQL запроса
      $result=mysql_query($query);//Отправка запроса
      echo'<h3 align="center">Результат выполнения запроса<br>
      <span style="color: red; font-size:xx-large; "
>&laquo;'.$query.'&raquo;</span></h3>';
      echo'<table align="center" width="80%" border="1" cellpadding="3"
cellspacing="0" bordercolor="#E2E2E2">
                  <tr>
                        <th width="20%">Наименование</th>
                        <th>Описание</th>
                        <th width="100">Цена</th>
                        <th>Изображение</th>
                  </tr>';
      while ($row = mysql_fetch_assoc($result))
      {
            echo '<tr>
                                          <td>'.$row['product_name'].'</td>
                                          <td>'.$row['pkoduct_descr'].'</td>
                                          <td
align="center">'.$row['product_price'].' &#8381;</td>
```

```
                                                <td><img
src="'.$row['product_img'].'" width="70"></td>
                        </tr>';
      }
      echo'</table>';
```



Результат в браузере после запуска сценария должен получиться примерно такой:



Теперь мы можем заполнять таблицу products новыми товарами, используя эту форму и контролировать наполнение таблицы.

Остался один нюанс. Если после заполнения и отправки формы обновить страницу браузера, браузер спросит о повторной отправке формы. Если ответить да, то произойдёт повторная отправка формы и в таблице появится дублирующая запись.

Избавиться от этой досадной неприятности можно, используя отправку специального заголовка.

Добавим в обработчик формы заголовок header('Location:?p=0') в условие проверки записи в базу данных

```
if(!$result)
            {
                  echo'<p>Ошибка выполнения запроса</p>';
                  echo $query;
            }
            else
            header('Location:?p=0' );
```

Данный заголовок обнуляет параметр p, делает его равным нулю, что не позволяет запустить обработчик после обновления страницы.

Добавление заголовка, сбрасывающего значение параметрар в обработчик формы

```
if(!$result)
{
    echo'<p>Ошибка выполнения запроса</p>';
    echo $query;
}
else
header('Location:?p=0' );
```