

Урок №7

Работа с базой данных.

В этом уроке Вы:

- узнаете, для чего нужна база данных;
- чем отличаются термины «БД» и «СУБД»;
- что такое реляционная база данных и из чего она состоит;
- какие существуют типы связей между данными;
- познакомитесь с языком баз данных SQL;
- научитесь добавлять данные;
- удалять данные;
- изменять данные;
- выбирать данные из базы;
- научитесь работать с MySQL средствами PHP.

1. Для чего нужна база данных

На прошлом занятии мы разобрались с тем, как можно хранить информацию в файлах. Однако в большинстве случаев такой подход будет доставлять массу неудобств при работе с большими объёмами данных. Например, представьте, что Вы храните в файле информацию о сотрудниках фирмы. Каждая строка – запись об одном человеке. В определённый момент встаёт задача найти сотрудника по имени и фамилии. При работе с файлом нет функции, которая сделает это автоматически. Соответственно, придётся самостоятельно писать не самый простой алгоритм анализа всех строк.

А при работе с базой данных уже есть специальная команда, с помощью которой мы сможем решить эту задачу за считанные секунды. По сути, база данных и является набором файлов с информацией, но в ней уже реализованы стандартные механизмы по совершению наиболее востребованных операций с записями.

То есть, база данных позволяет разработчику сэкономить массу времени при работе с информацией. Файлы, при этом, с сервера никуда не исчезают. Например, изображения и документы, также как и раньше, хранятся в каталогах. А вот вся текстовая информация переносится именно в базу данных.

2. Отличие «БД» от «СУБД»

БД – база данных. Под этим термином понимается информация, которую вы храните.

СУБД – система управления базой данных. Это программа, которая предоставляет доступ внешним приложениям к базе данных, обеспечивает ее работу.

Существуют различные популярные СУБД: Oracle, Microsoft SQL Server, MySQL, Sybase, PostgreSQL итд.

Сайты PHP чаще всего работают в связке с MySQL, поэтому именно данную СУБД мы будем рассматривать в текущем уроке.

3. Реляционная база данных

Существует множество различных способов организации информации в БД. Основные типы баз данных:

- Иерархические
- Объектные
- Сетевые
- Объектно-реляционные
- Функциональные
- Реляционные

Нас интересует именно последний тип, так как именно с ним придётся работать при создании сайтов.

Реляционная база данных состоит из **таблиц** и **связей** между ними. Как известно из жизни, любая таблица – это набор столбцов и строк. Столбцов создаётся определённое количество, а строк может быть сколько угодно. Столбец является более важным элементом, чем строка, так как последняя отвечает лишь за конкретную запись в таблице, а столбец – за наличие определённой характеристики у всех записей.

Представим, что нам нужно хранить информацию о сотрудниках фирмы. При проектировании таблицы сначала нужно просто представить, а какие вообще данные о каждом человеке нам нужны. Например, мы решили, что будем хранить фамилию, имя и отчество сотрудника. Тогда таблица может выглядеть следующим образом:

id_emp	first_name	middle_name	last_name
1	Иван	Иванович	Иванов
2	Петр	Петрович	Петров
3	Павел	Павлович	Павлов
4	Елена	Ивановна	Иванова
5	Елена	Петровна	Петрова

Обратите внимание, что в таблице создаётся дополнительное поле **id_emp**. Это так называемый **первичный ключ** – уникальной номер конкретной строки, по которому мы всегда сможем её получить. У каждой таблицы в реляционной базе данных должен быть **первичный ключ**.

Теперь немного усложним задачу. Представим, что фирма состоит из нескольких отделов, и нам необходимо хранить информацию о том, в каком именно отделе работает сотрудник. Сначала посмотрим на неправильную реализацию:

id_emp	dept	first_name	middle_name	last_name
1	ИТ	Иван	Иванович	Иванов
2	Бухгалтерия	Петр	Петрович	Петров
3	ИТ	Павел	Павлович	Павлов
4	ИТ	Елена	Ивановна	Иванова
5	Бухгалтерия	Елена	Петровна	Петрова

Почему же так делать неправильно? Если внимательно присмотреться к столбу **dept**, станет видно, что в нём присутствует повторяющаяся информация. В итоге, если у нас вдруг измениться название отдела, придётся менять столько строчек в таблице, сколько сотрудников в нём работало. Поэтому необходимо сделать две таблицы, разбив информацию следующим образом:

Таблица **depts** (отделы):

id_dept	name
1	Бухгалтерия
2	ИТ

Таблица **emps** (сотрудники):

id_emp	id_dept	first_name	middle_name	last_name
1	2	Иван	Иванович	Иванов
2	1	Петр	Петрович	Петров
3	2	Павел	Павлович	Павлов
4	2	Елена	Ивановна	Иванова
5	1	Елена	Петровна	Петрова

Суть изменения заключается в том, что в таблице **emps** в столбце **id_dept** у нас хранится только номер отдела, а не его название. Данное поле называется внешним ключом, ссылающимся на таблицу **depts**.

4. Язык SQL

SQL (Structured Query Language - «язык структурированных запросов») - универсальный компьютерный язык, применяемый для создания, модификации и управления данными в реляционных базах данных.

Т.е., SQL – это язык для общения с базой данных. Любую операцию, от создания таблицы до выборки данных, можно осуществить только посредством запроса на языке SQL.

Все запросы делятся на два вида:

- DDL
- DML

К DDL (Data Definition Language – «язык описания данных») относятся запросы, меняющие структуру базы данных. Например, создание таблицы, удаление таблицы, добавление столбца к существующей таблице.

К DML (Data Modification Language – «язык модификации данных») относятся запросы, меняющие содержимое базы данных, то есть, операции над строками таблиц. Сюда относится вставка, удаление, изменение и выборка строк.

Мы сосредоточимся на изучении именно DML-запросов, так как они осуществляются гораздо чаще. По сути, DDL-операции не должны осуществляться из PHP скрипта, так как структура базы данных должна быть заранее продумана и сделана один раз при её создании.

Пример DDL-операции – создание таблицы **depts**:

```
CREATE TABLE depts
(
    id_dept INT NOT NULL,
    name VARCHAR(32) NOT NULL,
    PRIMARY KEY (id_dept)
)
```

Обычно данные команды не прописываются вручную, а производятся с помощью специальной утилиты **phpMyAdmin**.

Если Вы используете **denwer**, то можете открыть её по следующему адресу:
<http://localhost/tools/phpMyAdmin>

Обязательно посмотрите видео по работе с **phpMyAdmin**:

[ТУТ ССЫЛКА НА НОВОЕ ВИДЕО ИЗ ВЕБ-СТАРТА](#)

Теперь перейдём к рассмотрению DML-операций.

4.1 Вставка строк

Для вставки строк в языке SQL служит оператор **INSERT**. Вот так можно наполнить базу данных сотрудников и отделов:

```
INSERT INTO depts (id_dept, name) VALUES ('1', 'Бухгалтерия');
INSERT INTO depts (id_dept, name) VALUES ('2', 'ИТ');

INSERT INTO emps (id_emp, id_dept, last_name, first_name, middle_name)
VALUES ('1', '2', 'Иван', 'Иванович', 'Иванов');

INSERT INTO emps (id_emp, id_dept, last_name, first_name, middle_name)
VALUES ('2', '1', 'Петр', 'Петрович', 'Петров');

INSERT INTO emps (id_emp, id_dept, last_name, first_name, middle_name)
VALUES ('3', '2', 'Павел', 'Павлович', 'Павлов');

INSERT INTO emps (id_emp, id_dept, last_name, first_name, middle_name)
VALUES ('4', '2', 'Елена', 'Ивановна', 'Иванова');

INSERT INTO emps (id_emp, id_dept, last_name, first_name, middle_name)
VALUES ('5', '1', 'Елена', 'Петровна', 'Петрова');
```

4.2 Удаление строк

Предположим, руководство решило уволить всех сотрудников отдела маркетинга. В этом случае поможет оператор **DELETE**, удаляющий строки из таблицы:

```
DELETE FROM emps
WHERE id_dept = '2'
```

Увольнение конкретного сотрудника выглядит похожим образом:

```
DELETE FROM emps
WHERE id_emp = '2'
```

Обратите внимание, запись о сотруднике мы удаляем, используя не его фамилию, а первичный ключ. Связано это с тем, что однофамильцы могут присутствовать в фирме, а первичный ключ всегда уникален.

4.3 Изменение строк

Елена Петровна вышла замуж за Павла Павловича и поменяла фамилию. Для изменения строк таблицы служит оператор **UPDATE**:

```
UPDATE emps
SET last_name = 'Иванова'
WHERE id_emp = '5'
```

Главное, не забывать указывать при изменении строк параметр **WHERE**. В противном случае, при отсутствии **WHERE** в данном примере, фамилии поменялись бы у всех сотрудников.

4.4 Выборка строк

За выборку строк отвечает оператор SELECT. С его помощью можно составлять сложнейшие запросы, выбирающие данные сразу из множества таблиц. В этом уроке нас интересуют лишь самые простые примеры.

Все сотрудники:

```
SELECT *
  FROM emps
```

Сотрудники ИТ-отдела:

```
SELECT *
  FROM emps
 WHERE id_dept = '1'
```

Сколько всего в фирме работает человек?

```
SELECT count(*)
  FROM emps;
```

Сколько сотрудников в ИТ-отделе?

```
SELECT count(*)
  FROM emps
 WHERE id_dept = '2';
```

Сотрудники отдела бухгалтерии, отсортированные по фамилии, имени, отчеству:

```
SELECT *
  FROM emps
 WHERE id_dept = '1'
 ORDER BY last_name, first_name, middle_name ASC
```

Сотрудники отдела бухгалтерии, отсортированные в обратном алфавитном порядке:

```
SELECT *
  FROM emps
 WHERE id_dept = '1'
 ORDER BY last_name, first_name, middle_name DESC
```

На данном этапе мы рассмотрели основные DML-операции. Очень важно, что мы видели синтаксис именно языка SQL, т.е., в PHP-коде нельзя просто брать и писать подобные команды. Для этого нужно применять специальные функции, которые мы рассмотрим в следующем разделе.

5. Средства PHP для работы с MySQL

Порядок работы с базой данных в PHP похож на порядок работы с файлами, но имеет определённые особенности:

- 1) установить соединение с сервером;
- 2) выбрать конкретную базу данных;
- 3) выполнить операции;
- 4) закрыть соединение.

Рассмотрим по очереди все эти шаги. Для подключения к базе в PHP существует специальная функция **mysql_connect**. Чаще всего она вызывается с тремя параметрами:

```
mysql_connect($server, $username, $password);
```

Если вы тестируете сайт на локальном компьютере, и у вас установлен пакет Денвер, то параметры должны быть следующими:

```
$server = 'localhost'; // Адрес сервера, на котором находится база
$username = 'root';    // Имя пользователя
$password = '' ;        // Пароль
```

После установки соединения с сервером необходимо выбрать конкретную базу данных. Для этого служит функция **mysql_select_db**:

```
mysql_select_db ($db_name);
```

где \$db_name – имя базы, с которой Вы хотите работать.

Теперь можно совершать действия, которые мы рассматривали в предыдущем пункте. За выполнение запроса к базе отвечает функция **mysql_query**, которая принимает строку с командой на языке SQL и возвращает результат её выполнения, например:

```
$result = mysql_query("DELETE FROM emps WHERE id_emp = '2'");
```

Для запросов, не подразумевающих получение данных из базы (INSERT, UPDATE, DELETE), функция возвращает true в случае успешного выполнения операции и false в противном случае.

Для SELECT-запросов функция возвращает дескриптор с результатом выборки в случае успешного выполнения операции и false в противном случае.

Получается, что после выполнения запроса на выборку данных, нам необходимо его обработать. Как правило, это представляет собой перегонку системного типа данных в массивы. Например:

```
$result = mysql_query('SELECT * FROM emps');
$epms = array();

while($row = mysql_fetch_assoc($result))
    $epms[] = $row;
```

Внимательно разберёмся с данным кодом. В переменной \$result оказывается результат выборки из базы - список всех сотрудников.

Однако, пока что он храниться в непригодном для нас виде. Поэтому создаём пустой массив \$epms, в который будем добавлять информацию о каждой записи.

Теперь запускаем цикл, который ориентируется на функцию **mysql_fetch_assoc**. Она извлекает очередную строку из выборки данных и возвращает её в переменную \$row в виде ассоциативного массива. Ключами данного массива являются названия столбцов таблицы, а значениями – данные из конкретной строки. Когда строки закончатся, **mysql_fetch_assoc** вернёт значение false, и цикл завершится.

На выходе мы получим двумерный массив \$epms с информацией о сотрудниках. В нём будет столько же элементов, сколько строк вернула база в результате выборки. Каждый его элемент – ассоциативный массив с информацией об одном сотруднике.

Данная перегонка является стандартной. Её достаточно освоить один раз, а затем лишь применять на практике.

Также существуют и другие полезные функции при работе с базой, например:

mysql_num_rows - число строк, содержащееся в результате выборки данных;

mysql_affected_rows - число строк, затронутых последним запросом INSERT, UPDATE или DELETE;

mysql_error – сообщение о последней ошибке, возникшей в ходе запроса;

mysql_insert_id – id записи, добавленной последним запросом INSERT;

mysql_close – закрывает соединение с сервером MySql.

Самоконтроль

- ✓ Зачем нужна база данных
- ✓ Чем БД отличается от СУБД
- ✓ Из чего состоит реляционная база данных
- ✓ Из чего состоят таблицы в БД
- ✓ Что в таблице важнее: столбец или строка
- ✓ Что такое первичный ключ
- ✓ В чём отличие DDL и DML операций
- ✓ Какие из вышеперечисленных операций совершаются чаще
- ✓ Как создать базу данных и таблицы через phpMyAdmin
- ✓ Вставка строк
- ✓ Удаление строк
- ✓ Изменение строк
- ✓ Выборка строк
- ✓ Порядок работы с базой данных в PHP

Домашнее задание

После предыдущего урока Вы реализовали галерею изображений с помощью файлов и папок. Теперь необходимо сделать то же самое, но для хранения имен картинок следует использовать базу данных.

Базовый блок

Создать галерею изображений, состоящую из двух страниц:

index.php – просмотр всех фотографий (уменьшенных копий);

photo.php – просмотр конкретной фотографии (изображение оригинального размера).

В базе данных создать таблицу, в которой будет храниться информация о картинках.

Продвинутый блок

На странице просмотра фотографии полного размера под картинкой должно быть указано число ее просмотров.

На странице просмотра галереи список фотографий должен быть отсортирован по популярности. Популярность - число кликов по фотографии.