Часть 2. Практическая работа №7

Работа с базой данных

В этой Практической работе вы:

* узнаете, для чего нужна база данных,
* чем отличаются термины «БД» и «СУБД»,
* что такое реляционная база данных и из чего она состоит,
* познакомитесь с языком баз данных SQL,
* научитесь добавлять данные,
* удалять данные,
* изменять данные
* и выбирать данные из базы,
* узнаете, какими средствами располагает PHP для работы с MySQL и научитесь ими пользоваться.

Замечание. Перед тем, как приступить к Практической работе, необходимо пройти видео Практическая работа «Начинаем работу с MySQL». Вы должно знать, что такое phpMyAdmin и уметь создавать с его помощью базу данных и таблицы в ней.

1. Для чего нужна база данных

База данных - неотъемлемая часть современного веб-приложения. В ней хранится вся информация, необходимая для работы сайта. Исключение составляют файлы: изображения, документы. Они обычно и хранятся в виде файлов в предназначенном для них каталоге на сервере. Однако существуют возможность и такую информацию хранить в базе данных, но ее используют редко.

Из базы данных можно выбирать информацию с помощью специального языка запросов. Операции выборки происходят очень быстро благодаря специфике внутреннего устройства хранилища данных. Использование базы данных позволяет не беспокоиться о совместном доступе к данным (что обязательно произойдет, если ваш сайт посещают, хотя бы 100 человек в день), все функции синхронизации хранилище берет на себя.

1. 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.

Популярные СУБД: Oracle, Microsoft SQL Server, MySQL, Sybase, PostgreSQL.

Сайты PHP чаще всего работают в связке с MySQL, именно эту СУБД мы будем рассматривать в текущем Практическая работае.

База данных проектируется и создается для каждого конкретного проекта, СУБД же выбирается из небольшого списка стандартных средств.

Фраза «я использую на сайте базу данных MySQL» неверная, правильно сказать «я использую на сайте СУБД MySQL».

1. Реляционная база данных

Реляционная база данных состоит из таблиц. Таблица имеет ограниченное количество столбцов (обычно небольшое) и сколь угодно много строк. Существуют и другие типы баз данных (не реляционные), но используются крайне редко. Так что про них можно и вовсе не знать.

Такую популярность реляционная модель получила ввиду своей простоты. Ее можно использовать практически для любых задач! Слово «реляционная» происходит от английского «relation» («отношение»). Таблицы, из которых состоит база данных, как правило, связаны друг с другом, то есть находятся «в отношении».

Предположим, нам нужно хранить информацию о сотрудниках и отделах (это самый распространенный пример). Приведем пример реляционной базы данных, решающих эту задачу.

Нам потребуются две таблицы: сотрудники и отделы, назовем их emps и depts соответственно.

Столбцы таблицы отделов (depts):

* id\_dept (первичный ключ)
* name (название отдела)

Столбцы таблицы сотрудников (emps):

* id\_emp (первичный ключ)
* id\_dept (внешний ключ, ссылающийся на таблицу depts)
* first\_name (имя)
* middle\_name (отчество)
* last\_name (фамилия)

Таблицы реляционной базы данных характерны тем, что каждая должна содержать первичный ключ. Первичный ключ - это столбец с уникальными значениями, то есть по значению этого столбца можно однозначно идентифицировать строку таблицы.

Каждый сотрудник работает в каком-то отделе. Таблица emps ссылается на таблицу depts с помощью столбца id\_dept. Столбец id\_dept в данном случае называется внешним ключом.

Пусть в нашей фирме два отдела: «Бухгалтерия» и «Маркетинг». В бухгалтерии работают Иванов Иван Иванович, Петров Петр Петрович и Сидорова Елена Николаевна. Маркетингом занимаются Ушаков Павел Павлович и Ефремов Илья Викторович.

Таблица depts:

|  |  |
| --- | --- |
| id\_dept | name |
| 1 | Бухгалтерия |
| 2 | Маркетинг |

Таблица emps:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| id\_emp | id\_dept | first\_name | middle\_name | last\_name |
| 1 | 1 | Иван | Иванович | Иванов |
| 2 | 1 | Петров | Петр | Петрович |
| 3 | 2 | Ушаков | Павел | Павлович |
| 4 | 2 | Ефремов | Илья | Викторович |
| 5 | 1 | Сидорова | Елена | Николаевна |

1. Язык SQL

SQL (Structured Query Language - «язык структурированных запросов») - универсальный компьютерный язык, применяемый для создания, модификации и управления данными в реляционных базах данных. Структурными запросы называются потому, что в общем случае могут содержать вложенные подзапросы.

Проще говоря, SQL - это язык, на котором можно общаться с базой данных. Любую операцию, от создания таблицы до выборки данных, возможно, осуществить только посредством запроса на языке SQL. Запрос, как правило, отправляется внешней программой к СУБД. Та в свою очередь выполняет его, производя необходимые операции над базой данных, и возвращает результат.

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.

Вернемся к базе данных отделов и сотрудников.

С помощью такого запроса можно создать таблицу depts:

CREATE TABLE depts (

id\_dept INT NOT NULL,

name VARCHAR(32) NOT NULL, PRIMARY KEY (id\_dept)

)

А таким запросом - таблицу emps:

CREATE TABLE emps (

id\_emp INT NOT NULL, id\_dept INT NOT NULL, first\_name VARCHAR(32) NOT NULL, middle name VARCHAR(32), last\_name VARCHAR(32) NOT NULL,

PRIMARY KEY (id\_emp),

FOREIGN KEY (id\_dept) REFERENCES depts (id\_dept)

)

Обычно DML операции приходится производить один раз - при создании базы данных. Их чаще выполняет не вручную, а с помощью специальных утилит. Для MySQL такой утилитой является phpMyAdmin. С помощью ее графического интерфейса можно создать таблицы, не задумываясь о синтаксисе оператора CREATE TABLE.

Скрипт, как правило, работает со строками таблиц, не меняя структуру базы данных, то есть производит только DML операции: вставку, удаление, изменение и выборку строк.

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', 'Маркетинг | ); |  |
| INSERT INTO | emps | (id emp, id dept, last | name, first name, | middle | name) |
| VALUES | d', | '1', 'Иванов', 'Иван', | 'Иванович'); |  |  |
| INSERT INTO | emps | (id emp, id dept, last | name, first name, | middle | name) |
| VALUES | '2', | '1', 'Петров', 'Петр', | 'Петрович'); |  |  |
| INSERT INTO | emps | (id emp, id dept, last | name, first name, | middle | name) |
| VALUES | '3', | '2', 'Ушаков', 'Павел', | 'Павлович'); |  |  |
| INSERT INTO | emps | (id emp, id dept, last | name, first name, | middle | name) |
| VALUES | ('4', | '2', 'Ефремов', 'Илья', | 'Викторович'); |  |  |
| INSERT INTO | emps | (id emp, id dept, last | name, first name, | middle | name) |
| VALUES | ('5', | '1', 'Сидорова', 'Елена | ', 'Николаевна'); |  |  |

1. 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'



1. Изменение строк

Елена Николаевна вышла замуж за Ивана Ивановича и поменяла фамилию. Для изменения строк таблицы служит оператор UPDATE. Воспользуемся им:

UPDATE emps

SET last name = 'Иванова'

WHERE id emp = '5'

1. Выборка строк

Для выборки строк служит оператор SELECT. С его помощью можно составлять сложнейшие запросы, выбирающие данные сразу из множества таблиц. В рамках текущего Практическая работаа рассмотрим лишь самые простые примеры.

Все сотрудники:



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

|  |  |  |
| --- | --- | --- |
| SELECT \* |  |  |
| FROM | emps |  |
| WHERE | id dept | = ' 1' |
| ORDER | BY last | name, first name, middle name |

Как зовут сотрудника № 1?

SELECT last name, first name, middle name FROM emps

WHERE id\_emp = '1'

Сколько всего в фирме работает человек?

SELECT count(\*)

FROM emps;

1. Средства PHP для работы с MySQL

int mysql\_connect (

[ string $server = ini\_get("mysql.default\_host") [, string $username = ini\_get("mysql.default\_user") [, string$password = ini\_get("mysql.default\_password") [, bool $new\_link = false [, int $dient\_flags = 0]]]]])

Перед тем, как работать с базой данных, необходимо установить с ней соединение. Для этого служит функция mysql\_connect(). Результат выполнения функции - дескриптор соединения, который пригодится, только если вы собираетесь работать сразу с несколькими подключениями. В большинстве случаем это не требуется, и результат выполнения функции проверяют лишь на неравенство false (что означает, что подключение прошло успешно).

Функция принимает множество параметров и все они необязательны. Подробнее о их назначении можно прочитать в документации. Чаще всего функция 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 (если вы, конечно, не меняли конфигурацию):

$server = 'localhost';

$username = 'root';

$password = '';

Когда сайт работает на удаленном сервере, параметр $server остается «localhost», $username и $password определяются при создании пользователя базы данных.

int mysql\_select\_db(string Sdbname [, int $link\_identifler])

До того как послать первый запрос серверу MySQL, необходимо указать, с какой базой данных мы собираемся работать. Для этого и предназначена функция mysql\_select\_db().

Она уведомляет PHP, что в дальнейших операциях с соединением $link identifier (или с

последним открытым соединением, если указанный параметр не задан) будет использоваться база данных $dbname. Функция возвращает true в случае успеха, иначе false.

int mysql\_query(string $query [, int $link\_identifier])

Запрос к базе данных. Текст запроса формулируется на языке SQL. Для запросов на выборку данных функция возвращает идентификатор результата в случае успеха и false в случае ошибки. Для запросов, не подразумевающих результат (INSERT, UPDATE, DELETE) функция в случае успеха возвращает true.

array mysql\_fetch\_array(int $result)

Функция извлекает очередную строку результата выборки данных. В качестве параметра принимает идентификатор, полученной вызовом функции mysql\_query(). Возвращает массив, состоящий из значений каждого столбца текущей строки. В качестве ключа - порядковый номер столбца (начиная с нуля). Когда извлекать больше нечего, функция возвращает false.

|  |  |  |
| --- | --- | --- |
| $server = 'localhost'; $username = 'root'; $password = '';  $dbname = 'lesson8'; |  |  |
| mysql connect($server, | $username, | $password); |
| mysql select db($dbname); | |  |
| $result = mysql query( | SELECT id | dept, name FROM depts'); |

echo '<li>';

while ($row = mysql fetch array($result)) {

echo '<a href="dept.php?id dept=' . $row[0] echo $row[1]; echo '</a>'; echo '</li>';

’>';

}

array mysql\_fetch\_assoc(int $result)

Функция аналогична mysql\_fetch\_array(), но возвращает ассоциативный массив, в котором в качестве ключа выступает имя столбца.

$result = mysql query('SELECT id dept, name FROM depts');

while ($row = mysql fetch assoc($result))

{

echo '<li> ' ;

echo '<a href="dept.php?id dept=' . $row['id dept'] . '">';

echo $ row['name'] ;

echo '</a>'; echo '</li>';

}

int mysql\_num\_rows(int $result)

Функция возвращает число строк, содержащееся в результате выборки данных.

$result = mysql query('SELECT id dept, name FROM depts');

$count = mysql num rows($result);

for ($i = 0; $i < $count; $i++)

{

$row = mysql fetch array($result); echo '<li>' ;

echo '<a href="dept.php?id dept=' . $row['id dept'] . '">';

echo $ row['name'] ;

echo '</a>'; echo '</li>';

}

int mysql\_affected\_rows([resource $link\_identifier])

Функция возвращает число строк, затронутых последним запросом INSERT, UPDATE или DELETE.

mysql\_query("DELETE FROM emps WHERE id\_dept='2'");

$count = mysql affected rows();

echo 'Уволены все сотрудники из отдела маркетинга. Их было $count чел.';

int mysql\_errno([ int $link\_identifier]) string mysql\_error([ int $link\_identifier])

Если в процессе работы с MySQL возникают ошибки, то сообщение об ошибке и ее номер можно получить с помощью этих двух функций. Первая возвращает номер последней зарегистрированной ошибки. Вторая - строку, содержащую текст сообщения об ошибке. Ее удобно применять в отладочных целях.

$result = mysql query($query text);

if ($result == false)

{

$err code = mysql errno();

$err text = mysql error();

die("Ошибка MySQL #$err code: $err text" .

"<br/>" .

"при выполнении SQL запроса: query text");

}

Резюме

Тема баз данных слишком обширна, чтобы уместить ее в один Практическая работа. Однако собранные в документе знания помогут начать изучение этой области и использовать базы данных в ваших проектах уже сейчас!

Задания:

В прошлом Практическая работае домашним заданием было реализовать галерею фотографий с помощью

файлов и папок. Теперь необходимо доработать сайт с использованием базы данных.

Требования:

1. На странице просмотра фотографии полного размера под картинкой должно быть указано число ее просмотров.
2. На странице просмотра галереи список фотографий должен быть отсортирован по популярности. В начале списка должны находиться наиболее популярные фотографии. Популярность определяется числом кликов по фотографии (просмотров увеличенного изображения).

Комментарии:

* при загрузке фотографии ей следует выделять уникальный идентификатор;
* помимо идентификатора в базе следует хранить тип картинки («gif», «jpg» и т.д.) и число кликов по ней;
* уменьшенные изображения следует хранить в отдельной папке с именами вида «идентификатор.тип»;
* изображения полного размера также следует хранить в отдельной папке с именами вида «идентификатор.тип».