

Internet ТЕХНОЛОГИИ

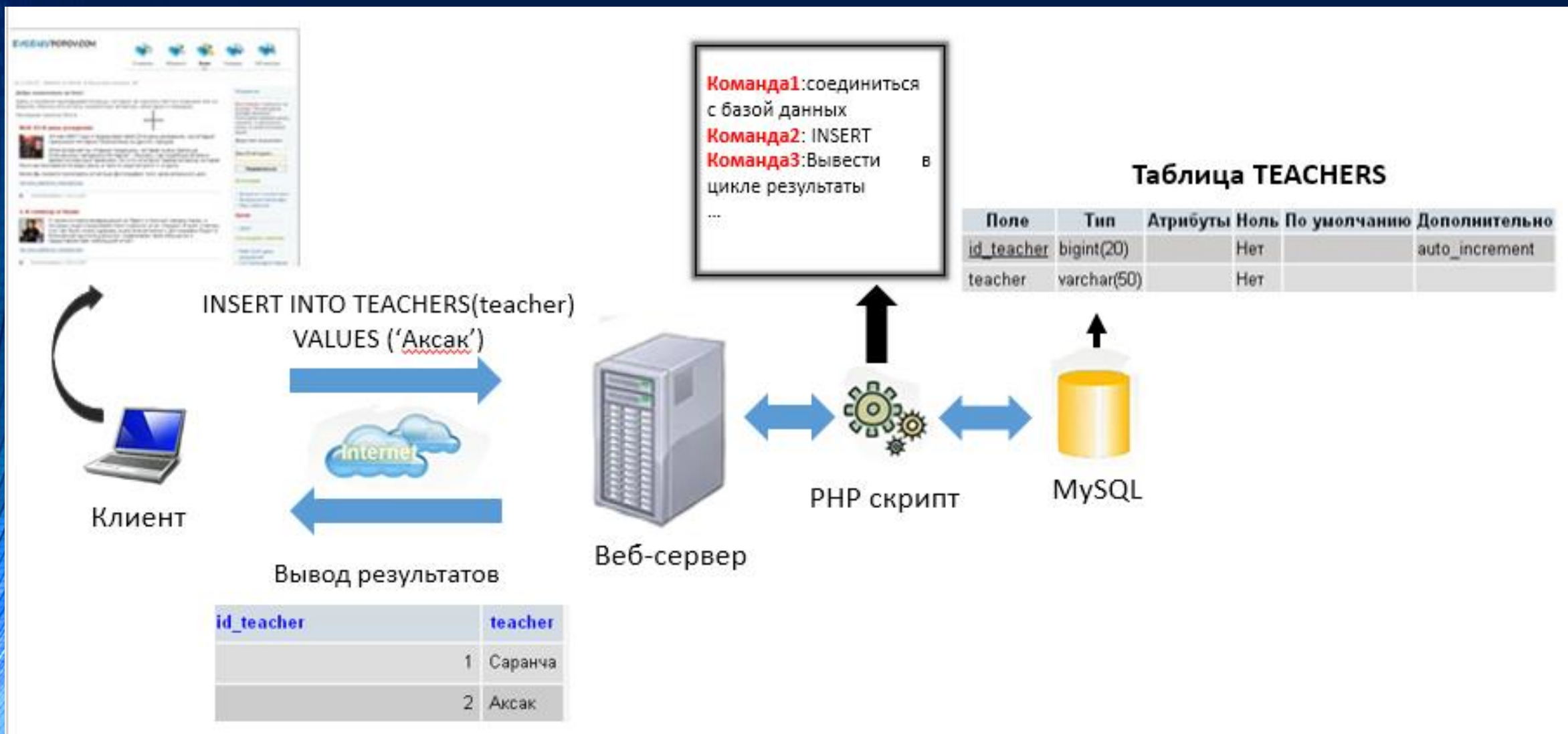
ЛЕКЦИЯ №3
ПРОГРАММНЫЙ ИНТЕРФЕЙС ДОСТУПА
К БАЗАМ ДАННЫХ

Содержание

- Схема взаимодействия пользователя с базой данных.
- Расширения PHP для работы с базой данных.
- Характеристика интерфейса PDO для обеспечения абстракции доступа к базам данных.
- Соединение с базой данных и организация проверок на наличие ошибок.
- Выполнение запросов к базе данных, быстрое получение данных из запроса.
- Выполнение подготовленных запросов и выборка строк из запроса
- Обработка транзакций.
- Заккрытие соединения с базой данных.
- Сравнительная характеристика PDO vs MySQLi, Benchmark.



Схема взаимодействия пользователя с БД



MySQL

MySQL – свободная реляционная система управления базами данных. Входит в состав комплексов серверного ПО WAMP, LAMP и в портативные сборки серверов Денвер, XAMPP, AppServ.



phpMyAdmin – веб-приложение с открытым кодом, написанное на языке PHP и представляющее собой веб-интерфейс для администрирования СУБД MySQL.

PHPMyAdmin позволяет через браузер осуществлять администрирование сервера MySQL, запускать команды SQL и просматривать содержимое таблиц и баз данных.



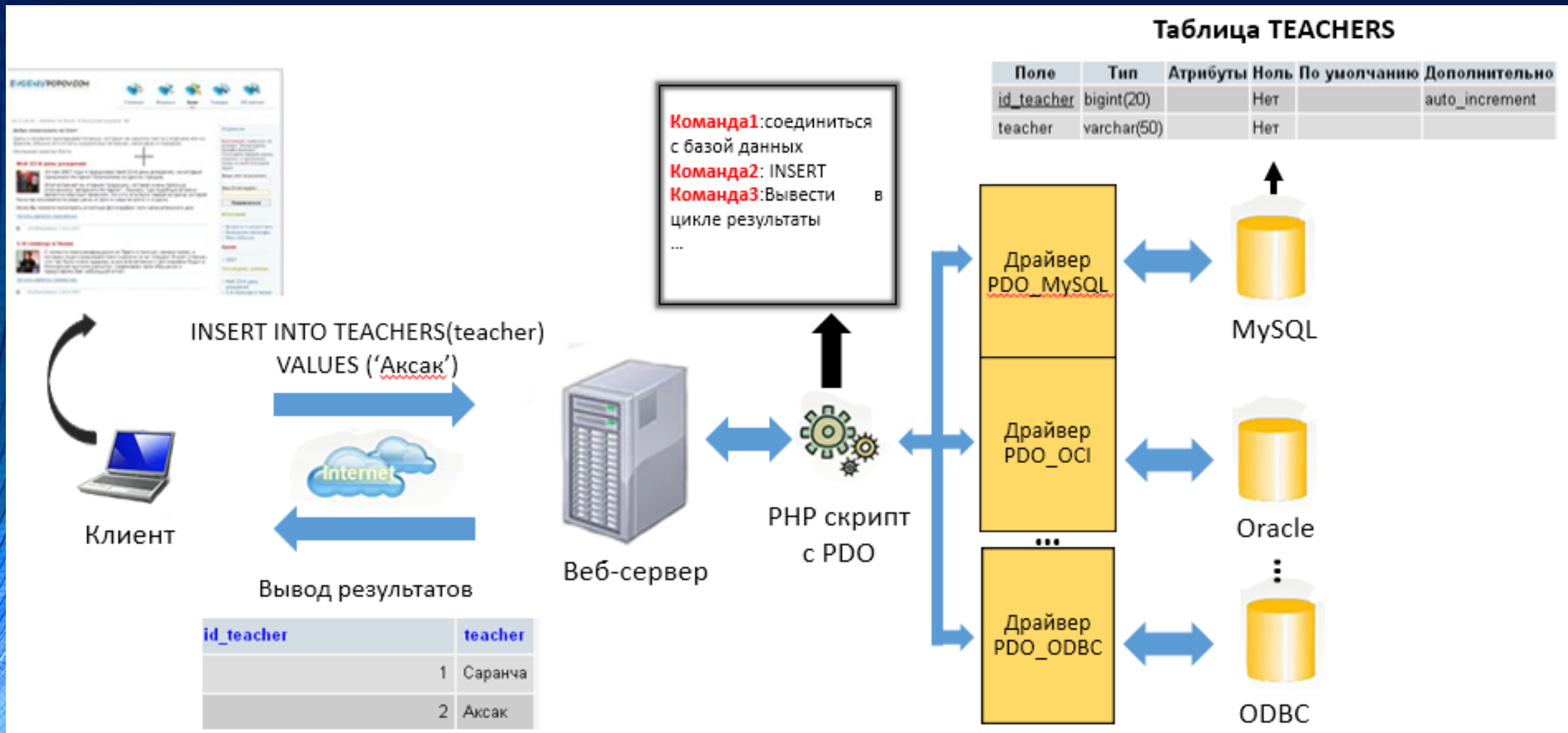
Расширения PHP для работы с MySQL

- MySQL – внедрено в PHP 2.0, оригинальное API MySQL.
- MySQLi – внедрено в PHP 5.0, позволяет получить доступ к функциональности, которую предоставляет MySQL версии 4.1 (текущая версия 5.6.29) и выше.
- PDO – внедрено в PHP 5.1, использует объектно-ориентированный функционал из ядра PHP 5. Для определения доступности библиотеки используется вывод функции `phpinfo()`.

Начиная с PHP 5.3.0 все три расширения используют MySQL Native Driver.



PDO. Унификация доступа к базам данных



Характеристика интерфейса PDO

К основным **задачам** PDO относятся следующие:

- Обеспечение стандартизированного API для реализации основных возможностей различных СУБД. Упрощение интерфейса работы с БД.
- Обеспечение легкой переносимости приложений между различными СУБД или типами сервера БД
- Возможность расширения. Реализация новых возможностей СУБД в PDO.

Преимущества использования PDO:

- **унифицированный** интерфейс;
- компактность, удобство;
- скорость, экономия трафика за счет кэширования подготовленных запросов;
- безопасность.



PDO. Структура

PDO состоит из двух частей:

- ядро – предоставляет интерфейс;
- драйверы – доступ к конкретным СУБД.

Просмотреть список драйверов возможно с помощью метода `PDO::getAvailableDrivers()`:

```
var_dump(PDO::getAvailableDrivers());
```

```
array(2) { [0]=> string(5) "mysql" [1]=> string(6) "sqlite" }
```



PDO. Поддержка СУБД

Поддерживаемая СУБД	Имя драйвера
CUBRID	PDO_CUBRID
FreeTDS / Microsoft SQL Server / Sybase	PDO_DBLIB
Firebird/Interbase 6	PDO_FIREBIRD
IBM DB2	PDO_IBM
IBM Informix Dynamic Server	PDO_INFORMIX
MySQL 3.x/4.x/5.x	PDO_MYSQL
Oracle Call Interface	PDO_OCI
ODBC v3 (IBM DB2, unixODBC and win32 ODBC)	PDO_ODBC
PostgreSQL	PDO_PGSQL
SQLite 3 and SQLite 2	PDO_SQLITE
Microsoft SQL Server	PDO_SQLSRV
4D	PDO_4D



PDO. Классы

В PDO существует несколько классов:

- PDO – базовый класс, который обеспечивает общий доступ к БД;
- PDOStatement – класс, предназначенный для работы с выборкой данных из базы. Объект типа PDOStatement может быть получен в результате выполнения таких методов PDO, как Query() или Prepare();
- PDOException служит для обработки ошибок PDO. Свойства:
 - public array \$errorInfo;
 - protected string \$message - текстовое сообщение об ошибке. Используйте Exception::getMessage(), чтобы получить его содержимое.
 - protected string \$code;



PDO. Источник базы данных, DSN

Синтаксис DSN (DataSourceName):

имя_драйвера:<специфические_опции>

Специфические опции: host, port, dbname, unix_socket, charset.

- mysql:host=localhost; dbname=testdb
- mysql:host=localhost;port=3307; dbname=testdb
- mysql:unix_socket=/tmp/mysql.sock; dbname=testdb
- sqlite::memory:
- sqlite:/opt/databases/mydb.sqlite3

Соединение с базой данных устанавливается при создании объекта PDO:

```
PDO::__construct (string dsn [, string username [, string  
password [, array driver_options]]] )
```



PDO. Опции подключения

- **PDO::ATTR_PREFETCH** (integer) Размер буфера предвыборки. Позволяет регулировать баланс между затрачиваемой памятью и скоростью работы с базой данных.
- **PDO::ATTR_TIMEOUT** (integer) Задаёт время в секундах, в течение которого должен быть завершён обмен с базой данных.
- **PDO::ATTR_CURSOR** (integer) Выбор типа курсора. PDO на данный момент поддерживает **PDO::CURSOR_FWDONLY** и **PDO::CURSOR_SCROLL**.
- **PDO::ATTR_PERSISTENT** (integer) Запрашивать постоянное соединение вместо создания нового подключения.
- **PDO::ERRMODE_SILENT** (integer) Предписание не выбрасывать исключений в случае ошибок. Предполагается, что разработчик скрипта явно проверяет все исключительные ситуации. Это режим по умолчанию.



PDO. Соединение с базой данных

```
$db_driver="mysql"; $host = "localhost"; $database = "iteh2lb2var4";  
$dsn = "$db_driver:host=$host; dbname=$database";
```

```
$username = "root"; $password = "";  
$options = array(PDO::ATTR_PERSISTENT => true, PDO::  
MYSQL_ATTR_INIT_COMMAND => 'SET NAMES utf8');  
try {  
    $dbh = new PDO ($dsn, $username, $password, $options);  
    echo "Connected to database<br>";  
    //$dbh ->query("SET CHARACTER SET utf8");  
}  
catch (PDOException $e) { /* Возвращает сообщение исключения */  
    echo "Error!: " . $e->getMessage() . "<br/>"; die();  
}  
//$dbh->exec("SET CHARACTER SET 'CP1251'");
```



PDO. Опции подключения

1. При соединении с базой данных

```
$dbh = new PDO('mysql:host=localhost;dbname=test', $user, $pass, array(  
PDO::ATTR_PERSISTENT => true, PDO::MYSQL_ATTR_INIT_COMMAND => 'SET  
NAMES \'UTF8\''));
```

2. При использовании некоторых функции выборки данных

```
$sth = $dbh->prepare ($sql, array( PDO::ATTR_CURSOR =>  
PDO::CURSOR_FWDONLY));
```

3. При использовании метода PDO::setAttribute() или PDOStatement::setAttribute

```
$dbh = new PDO($connection_string);  
$dbh->setAttribute(PDO::ATTR_DEFAULT_FETCH_MODE, PDO::FETCH_OBJ);  
$dbh->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
```



PDO. Заккрытие соединения с базой данных

```
<?php
```

```
$dbh = new PDO('mysql:host=localhost;dbname=test', $user, $pass);
```

```
// используем соединение здесь
```

```
//.....
```

```
// теперь закрываем его
```

```
$dbh = null;
```

```
?>
```

Если вы не делаете этого явно, PHP автоматически закрывает соединение после завершения скрипта.



PDO. Выполнение запросов к базе данных

`PDOStatement PDO::query (string $statement)`

Метод `query()` выполняет SQL-запрос *\$statement* без подготовки и возвращает результирующий набор в виде объекта **PDOStatement** или **FALSE** при ошибке.

//Вывод данных.

```
function get Nurse($dbh) {  
    $sql = "SELECT name, date, department FROM nurse";  
    foreach ($dbh->query($sql) as $row) {  
        echo $row['name']."\t";  
        echo $row['date']."\t";  
        echo $row['department']."\n";  
    }  
}
```



PDO. Быстрое получение данных из запроса

int PDOStatement::rowCount (void)

```
$del = $db->query("DELETE FROM users WHERE id =$id");  
$row_count = $del->rowCount();
```

int PDOStatement::columnCount (void)

```
$sth = $dbh->prepare("SELECT name, date FROM nurse");  
$colcount = $sth->columnCount();  
print("Перед вызовом execute() $colcount столбцов. Должно быть 0."); //0  
$sth->execute();  
$colcount = $sth->columnCount();  
print("После вызова execute() $colcount столбцов."); //2
```



PDO. Быстрое получение данных из запроса

```
public string PDO::lastInsertId ([ string $name = NULL ] )
```

```
$db->query("INSERT INTO users SET  
    name='Vasya',address='Here',email='vasya@test.com');  
$insertId=$db->lastInsertId();
```

В зависимости от PDO драйвера этот метод может вообще не выдать осмысленного результата, так как база данных может не поддерживать автоинкремент или последовательности.



PDO. Быстрый запуск SQL запроса на выполнение

```
int PDO::exec ( string $statement )
```

```
$dbh = new PDO('odbc:sample', 'db2inst1', 'ibmdb2');  
//$dbh->exec("SET CHARACTER SET 'CP1251'");
```

```
/* Удаляем все записи из таблицы FRUIT */  
$count = $dbh->exec("DELETE FROM fruit WHERE colour = 'red'");
```

```
/* Получим количество удаленных записей */  
print("Deleted $count rows.\n");
```

Deleted 1 rows.



PDO. Экранирование обычной строки

```
string PDO::quote ( string $string [, int $parameter_type =  
PDO::PARAM_STR ] )
```

```
$string = 'Nice';
```

```
print "Неэкранированная строка: $string\n";
```

```
print "Экранированная строка: " . $conn->quote($string) . "\n";
```

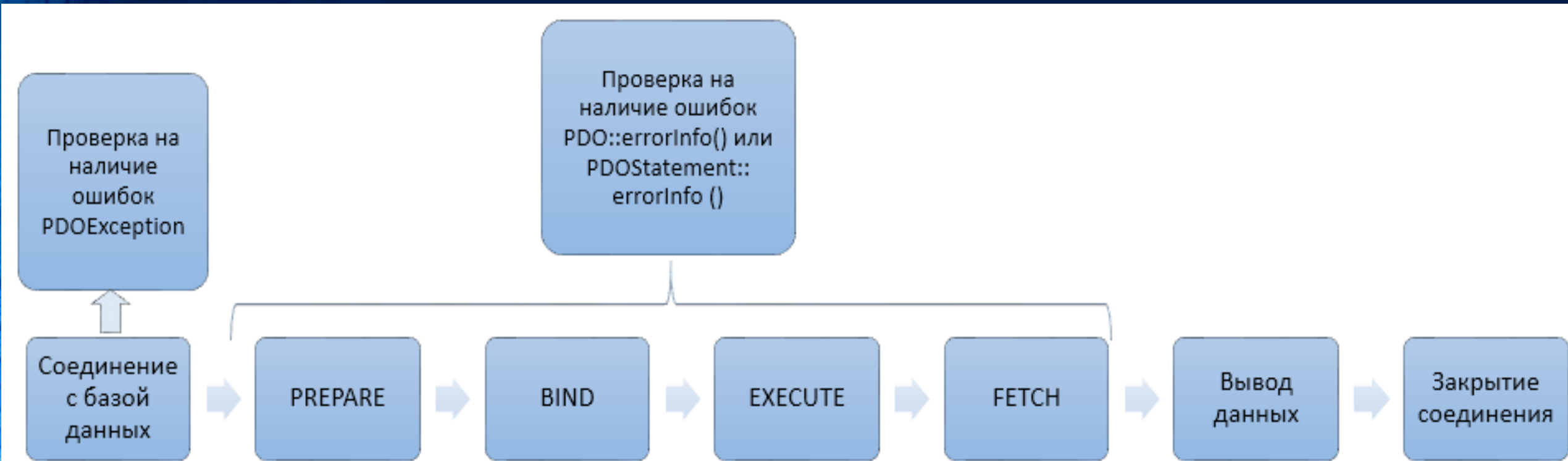
Результат выполнения данного примера:

Неэкранированная строка: Nice

Экранированная строка: 'Nice'



PDO. Схема выборки строк из базы данных с применением подготовленного запроса и вывода результатов



Если драйвер не поддерживает подготавливаемые запросы, PDO умеет их эмулировать.



PDO. Подготовленные запросы. Использование именованных параметров

```
PDOStatement PDO::prepare ( string $statement [, array $driver_options  
= array() ] )
```

```
$sql = 'SELECT name, colour, calories FROM fruit  
WHERE calories < :calories AND colour = :colour';
```

```
/*Создать объект PDOStatement с последовательным курсором*/  
$sth = $dbh->prepare ($sql, array  
(PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY));  
$sth->execute(array(':calories' => 150, ':colour' => 'red'));  
$red = $sth->fetchAll();
```



PDO. Подготовленные запросы. Использование именованных параметров

```
PDOStatement PDO::prepare ( string $statement [, array  
$driver_options = array() ] )
```

```
$sth = $dbh->prepare('SELECT name, colour, calories FROM fruit  
WHERE calories < ? AND colour = ?');  
$sth->execute(array(150, 'red'));  
$red = $sth->fetchAll();
```

PDO может заменять псевдопеременные, если, например, драйвер поддерживает только именованные или, наоборот, только неименованные переменные.



PDO. Запуск подготовленного запроса

```
bool PDOStatement::execute ([ array $input_parameters ] )
```

```
$data = array('Cathy', '9 Dark and  
Twisty Road', 'Cardiff');  
$sth = $dbh->prepare("INSERT  
INTO folks (name, addr, city)  
values (?, ?, ?)");  
$sth->execute($data);
```

Prepared statements не поддерживаются:

- для идентификаторов (поля и т.д.)
- для массивов в операторе IN()
- для массивов в запросах INSERT/UPDATE

```
$calories = 150; $colour = 'red';  
$sth = $dbh->prepare  
('SELECT name, colour, calories FROM  
fruit WHERE calories < :calories AND  
colour = :colour');  
$sth->  
execute(array(':calories' => $calories,  
':colour' => $colour));
```



PDO. Подготовленные запросы. BIND(). Задание значения псевдопеременной.*

```
bool PDOStatement::bindValue ( mixed $parameter , mixed $value [, int  
$data_type = PDO::PARAM_STR ] )
```

```
//Именованные псевдопеременные  
$sth = $dbh->prepare  
( 'SELECT name, colour, calories  
FROM fruit WHERE calories < :calories );  
$sth->bindValue( ':calories', 150,  
PDO::PARAM_INT );  
$sth->execute();
```

```
//Неименованные псевдопеременные  
$sth = $dbh->prepare  
( 'SELECT name, colour, calories  
FROM fruit WHERE calories < ?  
AND colour = ?' );  
$sth->bindValue ( 1, 150, PDO::PARAM_INT );  
$sth->bindValue ( 2, 'red', PDO::PARAM_STR );  
$sth->execute();
```



PDO. Подготовленные запросы. BIND(). Задание значения псевдопеременной.*

```
bool PDOStatement::bindParam ( mixed $parameter , mixed &$variable [,  
int $data_type = PDO::PARAM_STR [, int $length [, mixed $driver_options ]]] )
```

//Именованные псевдопеременные

```
$calories = 150; $colour = 'red';
```

```
$sth = $dbh->prepare
```

```
('SELECT name, colour, calories FROM fruit  
WHERE calories < :calories);
```

```
$sth->bindParam (':calories', $calories,  
PDO::PARAM_INT);
```

```
$sth->execute();
```

//Неименованные псевдопеременные

```
$sth = $dbh->prepare
```

```
('SELECT name, colour, calories  
FROM fruit WHERE calories < ?  
AND colour = ?');
```

```
$sth->bindParam (1, $calories,  
PDO::PARAM_INT);
```

```
$sth->execute();
```



PDO. Подготовленные запросы. BIND(). Задание значения псевдопеременной.*

```
bool PDOStatement::bindColumn ( mixed $column , mixed &$param [, int  
$type [, int $maxlen [, mixed $driverdata ]]] )
```

```
$sql = 'SELECT name, colour, calories FROM fruit';
```

```
$stmt = $dbh->prepare($sql);
```

```
/* Связывание по номеру столбца */
```

```
$stmt->execute();
```

```
$stmt->bindColumn(1, $name);
```

```
$stmt->bindColumn(2, $colour);
```

```
/* Связывание по имени столбца */
```

```
$stmt->execute();
```

```
$stmt->bindColumn('calories', $cals);
```



PDO. Вывод информации об ошибке

array **PDO::errorInfo** (void)

```
$stmt = $dbh->prepare ('bogus sql');  
if (!$stmt) {  
    echo "\nPDO::errorInfo():\n";  
    print_r($dbh->errorInfo());  
}
```

```
$sth = $dbh->prepare  
('SELECT skull FROM bones');  
$sth->execute();  
echo "\nPDOStatement::errorInfo():\n";  
$arr = $sth->errorInfo();  
print_r($arr);
```

Возвращаемый массив содержит следующие поля:

o Код ошибки SQLSTATE (пятисимвольный идентификатор определенный в стандарте ANSI SQL).

1 Код ошибки, заданный драйвером.

2 Выданное драйвером сообщение об ошибке



PDO. Fetch(). Выборка строк из запроса

mixed PDOStatement::fetch ([int \$fetch_style [, int \$cursor_orientation = PDO::FETCH_ORI_NEXT [, int \$cursor_offset = 0]]])

Извлекает следующую строку из результирующего набора объекта PDOStatement. Параметр `fetch_style` определяет, в каком виде PDO вернет эту строку.

PDO::FETCH_ASSOC

PDO::FETCH_NUM

PDO::FETCH_BOTH

PDO::FETCH_BOUND

PDO::FETCH_NAMED

PDO::FETCH_LAZY

PDO::FETCH_OBJ

PDO::FETCH_CLASS

PDO::FETCH_INT



PDO. Режимы выборки. PDO::FETCH_ASSOC

Вызов функции:

```
$sth = $dbh->prepare("select ID_Authors, name from Authors order by  
ID_Authors");
```

```
$sth->execute();
```

```
print_r($sth->fetch(PDO::FETCH_ASSOC));
```

ID_Authors	name

1	'Косминский Е.А.'
2	'Бочаров В.В.'

Вывод результата:

```
Array ( [ID_Authors] => 1 [name] => Косминский Е.А. )
```

Доступ к данным:

```
while($row = $sth->fetch(PDO::FETCH_ASSOC)) {  
    echo "<p>" . $row['ID_Authors'] . "&nbsp;" . $row['name'] . "</p>";  
}
```



PDO. Режимы выборки. PDO::FETCH_NUM

Вызов функции:

```
$sth=$dbh->prepare(" select ID_Authors, name from Authors order by ID_Authors ");  
$sth->execute();  
print_r($result->fetch(PDO::FETCH_NUM));
```

ID_Authors	name

1	'Косминский Е.А.'
2	'Бочаров В.В.'

Вывод результата:

```
Array ( [0] => 1 [1] => Косминский Е.А. )
```

Доступ к данным:

```
while($row = $sth->fetch(PDO::FETCH_NUM)) {  
    echo "<p>" . $row[0] . "&nbsp;" . $row[1] . "</p>";  
}
```



PDO. Режимы выборки. PDO::FETCH_BOTH

Вызов функции:

```
$sth = $dbh->prepare(" select ID_Authors, name from Authors order by ID_Authors ");  
$sth->execute();  
print_r($sth->fetch(PDO::FETCH_BOTH));
```

ID_Authors	name

1	'Косминский Е.А.'
2	'Бочаров В.В.'

Вывод результата:

```
Array ( [ID_Authors] => 1 [0] => 1 [name] => Косминский Е.А. [1] => Косминский Е.А. )
```

Доступ к данным:

```
while($row = $sth-> fetch(PDO::FETCH_BOTH)) {  
    echo "<p>" . $row[0] . "&nbsp;" . $row['name'] . "</p>";  
}
```



PDO. Режимы выборки. PDO_FETCH_OBJ

Вызов функции:

```
$sth = $dbh->prepare(" select ID_Authors, name from Authors order by ID_Authors ");  
$sth->execute();  
print_r($sth->fetch(PDO::FETCH_OBJ));
```

ID_Authors	name
1	'Косминский Е.А.'
2	'Бочаров В.В.'

Вывод результата:

```
stdClass Object ( [ID_Authors] => 1 [name] => Косминский Е.А. )
```

Доступ к данным:

```
while($row = $sth-> fetch(PDO::FETCH_OBJ)) {  
    echo $row->ID_Authors; echo "<br>";  
    echo $row->name; echo "<br>";  
}
```



PDO. Режимы выборки. PDO_FETCH_LAZY

Вызов функции:

```
$sth = $dbh->prepare(" select ID_Authors, name from Authors order by ID_Authors ");  
$sth->execute();  
print_r($sth->fetch(PDO::FETCH_LAZY));
```

ID_Authors	name
1	'Косминский Е.А.'
2	'Бочаров В.В.'

Вывод результата:

```
PDORow Object ( [queryString] => select ID_Authors, name from Authors  
order by ID_Authors [ID_Authors] => 1 [name] => Косминский Е.А. )
```

Доступ к данным:

```
$number=0; //индекс столбца  
while($row = $sth-> fetch(PDO::FETCH_LAZY)) {  
    echo $row->$number;      echo $row->$name;  
}
```



PDO. Режимы выборки. PDO_FETCH_CLASS

Вызов функции:

```
$sth = $dbh->prepare(" select ID_Authors, name from Authors order by ID_Authors ");
$sth->execute();
class author {
    function __construct() { }
}
$result->setFetchMode(PDO::FETCH_CLASS, 'author');
print_r($sth->fetch(PDO::FETCH_CLASS));
```

ID_Authors	name

1	'Косминский Е.А.'
2	'Бочаров В.В.'

Вывод результата:

```
author Object ( [ID_Authors] => 1 [name] => Косминский Е.А. )
```

Доступ к данным:

```
while($row = $sth-> fetch(PDO::FETCH_CLASS)) {
    echo $row ->ID_Authors;    echo $row->name;
}
```



PDO. Режимы выборки. PDO_FETCH_INT

Вызов функции:

```
$sth = $dbh->prepare(" select ID_Authors, name from Authors order by ID_Authors ");
$sth->execute();
class authorzip {
    function __construct() { }
}
$fio = new authorzip();
$result->setFetchMode(PDO::FETCH_INT
```

ID_Authors	name

1	'Косминский Е.А.'
2	'Бочаров В.В.'

Вывод результата:

authorzip Object ([ID_Authors] => 1 [name] => Косминский Е.А.)

Доступ к данным:

```
while($row = $sth-> fetch(PDO::FETCH_INT)) {
    echo $row ->ID_Authors;    echo $row->name;
}
```



PDO. setFetchMode()

`bool PDOStatement::setFetchMode (int $mode)`

`bool PDOStatement::setFetchMode (int $PDO::FETCH_CLASS, string $classname, array $ctorargs)`

`bool PDOStatement::setFetchMode (int $PDO::FETCH_INT, object $object)`

```
$result = $stmt->setFetchMode(PDO::FETCH_NUM);  
while ($row = $stmt->fetch()) {  
    print $row[0]."\t".$row[1] . "\t".$row[2]."\n";  
}
```

ID_Authors	name
1	'Косминский Е.А.'
2	'Бочаров В.В.'



PDO. FetchAll(). Выборка строк из запроса

Используется для возврата массива, содержащего все строки результирующего набора:

```
array PDOStatement::fetchAll ([ int $fetch_style [, mixed $fetch_argument [,  
array $ctor_args = array() ]]] )
```

Вызов функции:

```
$sql = 'SELECT `ID_Authors`, `name` FROM `tren_lk`';
```

```
$sth = $dbh->prepare($sql);
```

```
$sth->execute();
```

```
$result = $sth->fetchAll(PDO::FETCH_NUM);
```

```
print_r($result);
```

Доступ к данным:

```
foreach ($result as $row) {  
    echo $id=$row[0]; echo "<br>";  
}
```

Вывод результата:

```
Array ( [0] => Array ( [0] => 1 [1] => Косминский )  
        [1] => Array ( [0] => 2 [1] => Бочаров В. ) )
```



PDO. FetchAll(). Выборка строки или столбца

```
print("Извлечение первой строки результирующего набора:<br>");
```

Вызов функции:

```
$result = $sth->fetchAll(PDO::FETCH_NUM);
```

```
print_r ($result[1]);
```

Вывод результата:

```
Array ( [0] => 2 [1] => Бочаров В.В. )
```

ID_Authors	name
1	'Косминский Е.А.'
2	'Бочаров В.В.'

```
print("Извлечение первого столбца результирующего набора:<br>");
```

Вызов функции:

```
print_r ($sth->fetchAll(PDO::FETCH_COLUMN,0));
```

Вывод результата:

```
Array ( [0] => 1 [1] => 2 )
```



PDO. FetchAll(). Выборка строки или столбца

```
print("Извлечение уникальных данных первого столбца :<br>");
```

Вызов функции:

```
print_r ($sth->fetchAll(PDO::FETCH_COLUMN|PDO::FETCH_UNIQUE,o));
```

Вывод результата:

```
Array ( [1] => 1 [2] => 2 )
```

```
print("Группировка строк по значениям одного столбца:<br>");
```

Вызов функции:

```
print_r ($sth->fetchAll(PDO::FETCH_COLUMN|PDO::FETCH_GROUP,o); );
```

Вывод результата:

```
Array ( [1] => Array ( [0] => 1 ) [2] => Array ( [0] => 2 ) )
```



PDO. FetchAll(). Выборка строки или столбца

```
print("Создание объекта для каждой строки :<br>");
```

Вызов функции:

```
class auth { public $ID_Authors; public $name;}  
print_r ($sth->fetchAll(PDO::FETCH_CLASS,'auth'));
```

Вывод результата:

```
Array ( [0] => auth Object ( [ID_Authors] => 1 [name] => Косминский )  
        [1] => auth Object ( [ID_Authors] => 2 [name] => Бочаров В. ) )
```

Доступ к данным:

```
foreach ($result as $row) {  
    echo $row->ID_Authors; echo $row->name;  
}
```



PDO. FetchObject()

Этот метод является альтернативой вызову PDOStatement::fetch() с параметром PDO::FETCH_CLASS или PDO::FETCH_OBJ.

```
mixed PDOStatement::fetchObject ([ string $class_name = "stdClass" [, array $ctor_args ] ] )
```

Вызов функции:

```
class authors { }  
print_r($result = $sth->fetchObject("authors"));
```

Вывод результата:

```
authors Object ( [ID_Authors] => 1 [name] => Косминский Е.А. )
```

Доступ к данным:

```
echo $result->ID_Authors;  
echo $result->name;
```

Вывод результата:

```
1  
Косминский Е.А.
```



PDO. FetchColumn()

string **PDOStatement::fetchColumn** ([int \$column_number = 0])

Вызов функции:

Получение значения первого столбца следующей строки:

```
print_r($result = $sth->fetchColumn());
```

Получение значения второго столбца следующей строки:

```
print_r($result = $sth->fetchColumn(1));
```

Вывод результата:

Получение значения первого столбца следующей строки:

1

Получение значения второго столбца следующей строки:

Бочаров В.В.



PDO. FetchAll(). Выборка строки или столбца

```
print("Вызов функции для каждой строки :<br>");
```

Вызов функции:

```
function auth ($ID_Authors, $name) {  
    return "{$ID_Authors}: {$name}";  
}
```

```
print_r($sth->fetchAll(PDO::FETCH_FUNC,'auth'));
```

ID_Authors	name
1	'Косминский Е.А.'
2	'Бочаров В.В.'

Вывод результата:

```
Array ( [0] => 1: Косминский [1] => 2: Бочаров В. )
```

Доступ к данным:

```
foreach ($result as $row)  
{ echo $row; }
```



PDO. Транзакции

bool PDO::beginTransaction (void)

bool PDO::commit (void)

/ Начало транзакции, отключение автоматической фиксации */*

\$dbh->beginTransaction();

/ Вставка множества записей по принципу "все или ничего" */*

\$sql = 'INSERT INTO fruit (name, colour, calories) VALUES (?, ?, ?)';

\$sth = \$dbh->prepare(\$sql);

foreach (\$fruits as \$fruit) {

\$sth->execute(array(\$fruit->name, \$fruit->colour, \$fruit->calories,));

}

/ Фиксация изменений */*

\$dbh->commit();

/ Соединение с базой данных снова в режиме автоматической фиксации */*



PDO. Транзакции

bool PDO::rollBack (void)

```
/* Начинаем транзакцию, выключаем автофиксацию */  
$dbh->beginTransaction();
```

```
/* Изменяем схему базы данных и данные в таблицах */  
$sth = $dbh->exec("DROP TABLE fruit");  
$sth = $dbh->exec("UPDATE dessert SET name = 'hamburger'");
```

```
/* Осознаем свою ошибку и откатываем транзакцию */  
$dbh->rollBack();
```

```
/* База данных возвращается в режим автофиксации */
```



Сравнительная характеристика PDO vs MySQLi

Характеристика	PDO	MySQLi
Поддержка баз данных	12 различных драйверов	Только MySQL
API	ОПП	ОПП-процедурная часть
Соединение	Просто	Просто
Именованные параметры	Есть	Нет
Объектное отображение	Есть	Есть
Подготовленные выражения	Есть	Есть
Производительность	Хорошая	Хорошая
Хранимые процедуры	Есть	Есть



MySQLi. Классы

MySQLi имеет 3 основные класса:

- `mysqli` — предоставляет связь между PHP и базой данных MySQL. Необходим для установки соединения с БД;
- `mysqli_stmt` — использование подготовленных выражений;
- `mysqli_result` — представляет результирующий набор, полученный из запроса в базу данных. Объединяет функции для получения результатов запросов, сделанных с помощью `mysqli` или `mysqli_stmt`.



MySQLi. Соединение с базой данных. ОО интерфейс

```
mysqli::__construct() ([ string $host = ini_get("mysqli.default_host")  
[, string $username = ini_get("mysqli.default_user")  
[, string $passwd = ini_get("mysqli.default_pw")  
[, string $dbname = "" [, int $port = ini_get("mysqli.default_port")  
[, string $socket = ini_get("mysqli.default_socket") ]]]]] )
```

Для создания постоянного подключения необходимо перед именем хоста при создании подключения добавить «p:».

По умолчанию, при извлечении из пула соединение сбрасывается. mysqli делает это неявным вызовом функции **mysqli_change_user()** каждый раз, когда подключение используется повторно. С точки зрения пользователя подключение выглядит, как только что созданное. На описанное выше поведение mysqli влияет флаг **MYSQLI_NO_CHANGE_USER_ON_PCONNECT**.



MySQLi. Соединение с базой данных. ОО интерфейс

```
string $mysqli->connect_errno; //При отсутствие ошибок выводит 0.  
string $mysqli->connect_error;
```

```
bool mysqli::set_charset ( string $charset )
```

```
string $mysqli->host_info; //имя хоста сервера и тип подключения.  
string $mysqli->protocol_version; //версия используемого MySQL-  
протокола
```

```
bool mysqli::close ( void ); // Закрывает ранее открытое соединение
```



MySQLi. Соединение с базой данных

```
$mysqli = new mysqli('p:localhost', 'my_user', 'my_password', 'my_db');  
if ($mysqli->connect_error) {  
    die('Ошибка подключения (' . $mysqli->connect_errno . ') '  
        . $mysqli->connect_error);  
}  
echo 'Соединение установлено... ' . $mysqli->host_info . "\n";  
$charset = $mysqli->character_set_name();  
printf ("Текущая кодировка - %s\n", $charset);  
$mysqli->close();
```

//Вывод данных

Соединение установлено... MySQL host info: localhost via TCP/IP

Текущая кодировка - utf8_general_ci



MySQLi. Задание настроек соединения

```
$mysqli = mysqli_init();  
if (!$mysqli) {  
    die('mysqli_init failed');  
}  
  
if (!$mysqli->options(MYSQLI_INIT_COMMAND, 'SET AUTOCOMMIT = 0')) {  
    die('Setting MYSQLI_INIT_COMMAND failed');  
}  
  
if (!$mysqli->real_connect('127.0.0.1', 'root', '', 'Lab1_var1')) {  
    die('Connect Error (' . $mysqli->connect_errno . ') ' . $mysqli->connect_error);  
}  
echo 'Выполнено... ' . $mysqli->host_info . "\n";
```



MySQLi. Соединение с БД. Процедурный интерфейс

```
mysqli mysqli_connect ([ string $host = ini_get("mysqli.default_host")  
[, string $username = ini_get("mysqli.default_user")  
[, string $passwd = ini_get("mysqli.default_pw") [, string $dbname = ""  
[, int $port = ini_get("mysqli.default_port")  
[, string $socket = ini_get("mysqli.default_socket") ]]]]] )
```

```
int mysqli_connect_errno ( void )  
string mysqli_connect_error ( void )
```

```
bool mysqli_set_charset ( mysqli $link , string $charset )  
string mysqli_character_set_name ( mysqli $link )
```

```
string mysqli_get_host_info ( mysqli $link )  
bool mysqli_close ( mysqli $link )
```



Таблица БД для примеров выборки данных

Для хранения данных используется таблица LITERATURE

Структура таблицы

Обзор таблицы

Lab1_var1.LITERATURE	
ID_Book	: int(255)
name	: varchar(50)
date	: date
year	: year(4)
publisher	: varchar(50)
quantity	: varchar(10000)
ISBN	: varchar(25)
number	: varchar(50)
literate	: enum('Book','Journal','Newspaper')
FID_Resource	: int(255)

ID_Book	name	date	year	publisher	quantity	ISBN	number	literate	FID_Resource
1	SQL: Полное руководство	NULL	2001	Издательская группа BHV	10000	10-15-XX-44	NULL	Book	1
2	Из истории культуры средних веков и Возрождения	NULL	1976	Издательство Наука	1000	44-78-106-X	NULL	Book	2
3	Ежедневный журнал	NULL	2000	NULL	NULL	NULL	№104	Journal	0
4	GLAMOUR.COM	NULL	2010	Microsoft	200000	NULL	№55	Journal	0
5	Независимая газета	2010-09-11	NULL	NULL	NULL	NULL	NULL	Newspaper	0
6	ВЗГЛЯД.РУ	2010-09-10	NULL	NULL	NULL	NULL	NULL	Newspaper	0
7	Из истории культуры средних веков и Возрождения	2013-09-02	2013	BHV	2500	NULL	NULL	Book	3
9	Из истории культуры средних веков и Возрождения	NULL	NULL	Microsoft	500	NULL	NULL	Book	3



MySQLi. Выполнение запросов к базе данных

Объектно-ориентированный стиль

```
mixed mysqli::query ( string $query [, int $resultmode= MYSQLI_STORE_RESULT ] )
```

Процедурный стиль

```
mixed mysqli_query ( mysqli $link , string $query [, int $resultmode =  
MYSQLI_USE_RESULT ] )
```

```
bool mysqli_multi_query ( mysqli $link , string $query )
```

```
void mysqli_free_result ( mysqli_result $result )
```

```
$result = $mysqli->query("select ID_Book, name from literature")
```

```
$result = mysqli_query($link, "select ID_Book, name from literature")
```

```
//Вывод данных
```

```
mysqli_result Object ( [current_field] => 0 [field_count] => 2 [lengths] => [num_rows]  
=> 8 [type] => 0 )
```



MySQLi. Обработка результата

mysqli_result – представляет результирующий набор, полученный из запроса в БД.

Некоторые свойства и методы класса **mysqli_result** :

int \$field_count — количество полей в результирующем наборе

int \$num_rows — число рядов в результирующей выборке

mixed fetch_all — выбирает все строки из результирующего набора и помещает их в ассоциативный массив, обычный массив или в оба

mixed fetch_array — Выбирает одну строку из результирующего набора и помещает ее в ассоциативный массив, обычный массив или в оба

array fetch_assoc — Извлекает результирующий ряд в виде ассоциативного массива

object fetch_fields — Возвращает массив объектов, представляющих поля результирующего набора

object fetch_object — Возвращает текущую строку результата в виде объекта

void free — Освобождает память занятую результатами запроса



MySQLi. Выборка строк из запроса

Объектно-ориентированный стиль

`mixed mysqli_result::fetch_row (void)`

Процедурный стиль

`mixed mysqli_fetch_row (mysqli_result $result)`

```
if ($result = $mysqli->query($query)) {  
    /* выборка данных и помещение их в массив */  
    while ($row = $result->fetch_row()) {  
        printf ("%s (%s)\n", $row[0], $row[1]);  
    }  
}
```



MySQLi. Выборка строк из запроса

Объектно-ориентированный стиль

```
mixed mysqli_result::fetch_array ([ int $resulttype = MYSQLI_BOTH ] )
```

Процедурный стиль

```
mixed mysqli_fetch_array ( mysqli_result $result [, int $resulttype =  
MYSQLI_BOTH ] )
```

Возможные значения параметра:

MYSQLI_ASSOC, MYSQLI_NUM или MYSQLI_BOTH.

```
$buffer = mysqli_query($link, $query);
```

```
mysqli_fetch_array($buffer, MYSQLI_ASSOC);
```

```
//Вывод данных
```

```
Array ( [publisher] => Издательская группа BHV [quantity] => 10000 )
```



MySQLi. Выборка строк из запроса

Объектно-ориентированный стиль

```
mixed mysqli_result::fetch_all ([ int $resulttype = MYSQLI_NUM ] )
```

Процедурный стиль

```
mixed mysqli_fetch_all ( mysqli_result $result [, int $resulttype = MYSQLI_NUM ] )
```

```
mysqli_fetch_all ($list, MYSQLI_NUM)
```

//Вывод данных

```
Array ( [0] => Array ( [0] => 1 [1] => SQL: Полное руководство ) [1] => Array ( [0] => 2  
[1] => Из истории культуры средних веков и Возрождения ) )
```

//Доступ к данным

```
foreach($res as $name) { echo name[1]; }
```



MySQLi. Экранирование данных

Объектно-ориентированный стиль

```
string mysqli::escape_string ( string $escapestr )  
string mysqli::real_escape_string ( string $escapestr )
```

```
$city = $mysqli->real_escape_string($city);  
$mysqli->query("INSERT into myCity (Name) VALUES ('$city')")
```

Процедурный стиль

```
string mysqli_real_escape_string ( mysqli $link , string $escapestr )
```

```
$city = mysqli_real_escape_string($link, $city);  
mysqli_query($link, "INSERT into myCity (Name) VALUES ('$city')")
```



MySQLi. Подготовленные запросы

Подготовка SQL-выражения к выполнению:

```
mysqli_stmt mysqli::prepare ( string $query )
```

```
mysqli_stmt mysqli_prepare ( mysqli $link , string $query )
```

```
$mysqli->prepare(«SELECT * FROM `sk2_articles` WHERE `id` = ?»);
```

```
if ($mysqli->errno) {
```

```
    die('Select Error (' . $mysqli->errno . ') ' . $mysqli->error);
```

```
}
```

```
// второй способ – используя объект mysqli_stmt
```

```
$stmt = $mysqli->stmt_init();
```

```
$stmt->prepare(«SELECT * FROM `sk2_articles` WHERE `id` = ?»);
```

```
if ($stmt->errno) {
```

```
    die('Select Error (' . $stmt->errno . ') ' . $stmt->error);
```

```
}
```



MySQLi. Подготовленные запросы

Привязка переменных к **параметрам** подготавливаемого запроса:

```
bool mysqli_stmt::bind_param ( string $types , mixed &$var1 [, mixed &$... ] )
```

```
bool mysqli_stmt_bind_param ( mysqli_stmt $stmt , string $types , mixed &$var1 [, mixed &$... ] )
```

types: i - integer, d - double, s - string, b - blob.

```
$stmt = $mysqli->prepare("INSERT INTO CountryLanguage VALUES (?, ?, ?, ?)");  
$stmt->bind_param('sssd', $code, $language, $official, $percent);
```

Метки недопустимы в качестве идентификаторов (таких как имена столбцов или таблиц), а также в списке псевдонимов столбцов запроса SELECT



MySQLi. Подготовленные запросы

Привязка переменных к подготовленному запросу для размещения **результата**

```
bool mysqli_stmt::bind_result ( mixed &$var1 [, mixed &$... ] )
```

```
bool mysqli_stmt_bind_result ( mysqli_stmt $stmt , mixed &$var1 [, mixed &$... ] )
```

```
if ( $stmt = $mysqli->prepare("SELECT Code, Name FROM Country ORDER BY Name  
LIMIT 5")) {  
    $stmt->execute();  
    $stmt->bind_result($col1, $col2);  
    while ( $stmt->fetch() ) {  
        printf("%s %s\n", $col1, $col2);  
    }  
    $stmt->close(); /* закрываем запрос */  
}
```



MySQLi. Пример использования

```
<?php
$mysqli = new mysqli('localhost', 'my_user', 'my_password', 'my_db');
if (mysqli_connect_errno()) {
    printf("Ошибка соединения: %s\n", mysqli_connect_error()); exit();
}
$city = "Харьков";
if ($stmt = $mysqli->prepare("SELECT District FROM City WHERE Name=?")) {
    $stmt->bind_param("s", $city); /* связываем параметр */
    $stmt->execute(); /* исполняем запрос */
    $stmt->bind_result($district); /* прикрепляем результаты */
    $stmt->fetch(); printf("%s находится в %s\n", $city, $district);
    $stmt->close();
}
$mysqli->close(); /* закрываем соединение */
?>
```



Подготовленные запросы. Преимущества и недостатки

Преимущества:

- **Скорость** выполнения **серии** запросов выше. Запросы со связываемыми переменными кэшируются сервером, сокращая время синтаксического разбора.
- **Отделение** переменных от запроса. Позволяют установить выражение единожды, а затем многократно исполнить его с разными параметрами.
- **Защита** от SQL-инъекций. Данные не нужно предварительно экранировать.

Недостатки:

- При однократных запросах производительность **ухудшится**.
- Подготавливаться (prepare) могут только **определённые** запросы (не могут использоваться названия полей).
- Сложность **отладки**.



MySQLi. Транзакции

Включение или отключение автоматической фиксации изменений базы данных:

```
bool mysqli::autocommit ( bool $mode )
```

```
bool mysqli_autocommit ( mysqli $link , bool $mode )
```

Фиксация текущей транзакции:

```
bool mysqli::commit ( void )
```

```
bool mysqli_commit ( mysqli $link )
```

Откат последних изменений, произошедших после последнего вызова функции `mysqli_commit`:

```
bool mysqli::rollback ( void )
```

```
bool mysqli_rollback ( mysqli $link )
```



MySQLi. Транзакции. Пример

```
$load= mysqli_connect("localhost","username","password","dbname", 3306);  
mysqli_autocommit($load, false);  
mysqli_query($load, "SELECT * FROM table1 WHERE id='1'");  
mysqli_query($load, "INSERT INTO table2 VALUES ('1','2')");  
mysqli_commit($load);  
mysqli_query($load, "DELETE FROM table2 WHERE id='1'");  
/*теперь запись удалена, но мы отменяем удаление*/  
mysqli_rollback($load);
```



MySQLi. Пример л/р.

Для хранения данных используется таблица LITERATURE

Структура таблицы

Обзор таблицы

Lab1_var1.LITERATURE	
ID_Book	: int(255)
name	: varchar(50)
date	: date
year	: year(4)
publisher	: varchar(50)
quantity	: varchar(10000)
ISBN	: varchar(25)
number	: varchar(50)
literate	: enum('Book','Journal','Newspaper')
FID_Resource	: int(255)

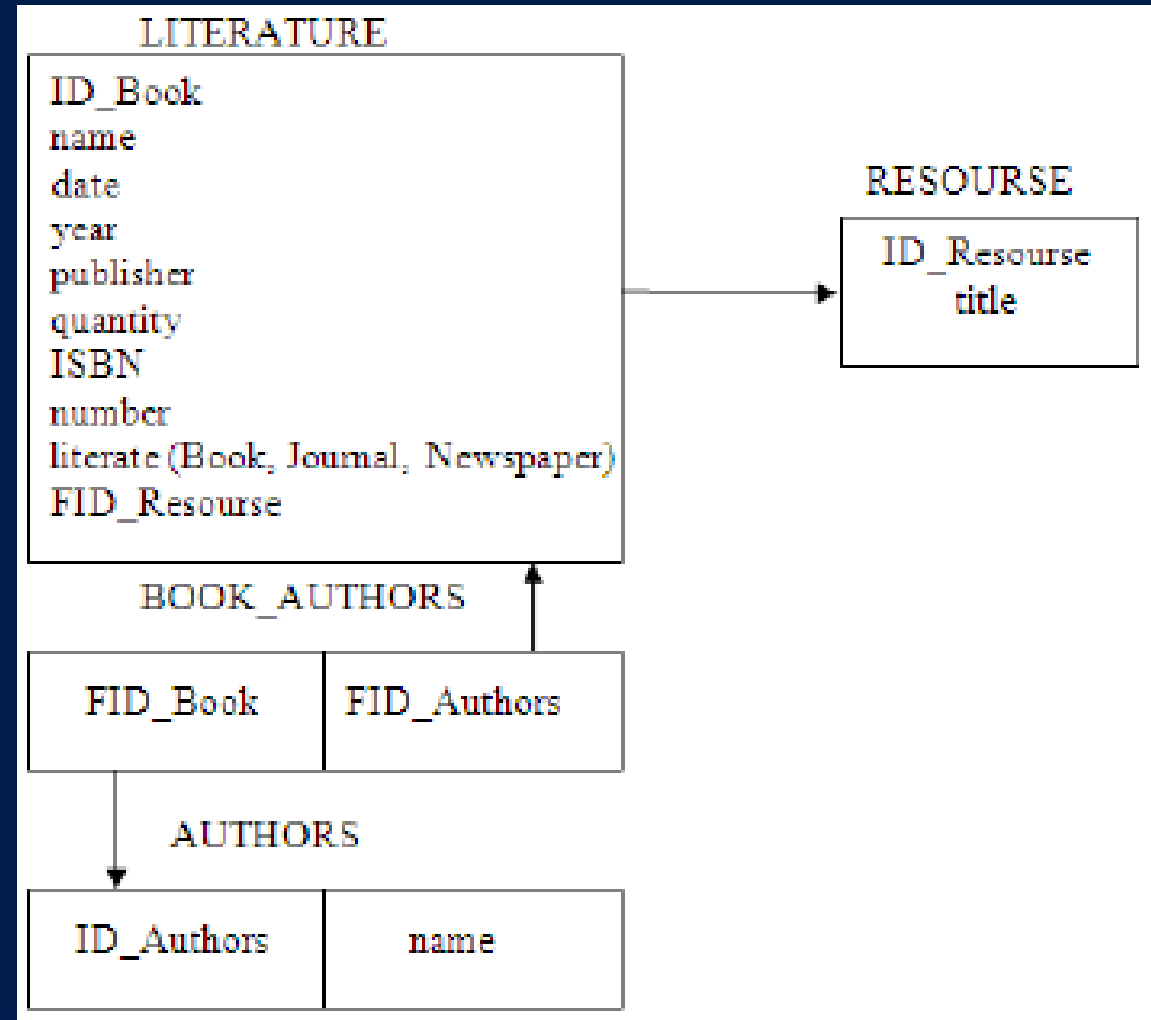
ID_Book	name	date	year	publisher	quantity	ISBN	number	literate	FID_Resource
1	SQL: Полное руководство	NULL	2001	Издательская группа BHV	10000	10-15-XX-44	NULL	Book	1
2	Из истории культуры средних веков и Возрождения	NULL	1976	Издательство Наука	1000	44-78-106-X	NULL	Book	2
3	Ежедневный журнал	NULL	2000	NULL	NULL	NULL	№104	Journal	0
4	GLAMOUR.COM	NULL	2010	Microsoft	200000	NULL	№55	Journal	0
5	Независимая газета	2010-09-11	NULL	NULL	NULL	NULL	NULL	Newspaper	0
6	ВЗГЛЯД.РУ	2010-09-10	NULL	NULL	NULL	NULL	NULL	Newspaper	0
7	Из истории культуры средних веков и Возрождения	2013-09-02	2013	BHV	2500	NULL	NULL	Book	3
9	Из истории культуры средних веков и Возрождения	NULL	NULL	Microsoft	500	NULL	NULL	Book	3



MySQLi. Пример л/р. Выборка строк из запроса

Сформировать запросы, которые будут выводить на экран информацию о:

- книгах, журналах и газетах с указанным именем;
- книгах, журналах и газетах за указанный временной период;
- книгах указанного автора.



MySQLi. Пример л/р. Выборка строк из запроса

```
<form name="form" action="lab.php"
                        method="post">
```

```
<select name="literat">
```

```
<?php
```

```
$list=mysqli_query($link, "select name from literature");
```

```
$res=mysqli_fetch_all($list, MYSQLI_NUM);
```

```
foreach($res as $name)
```

```
    echo "<option value='".$name[o]."'>".$name[o]."</option>";
```

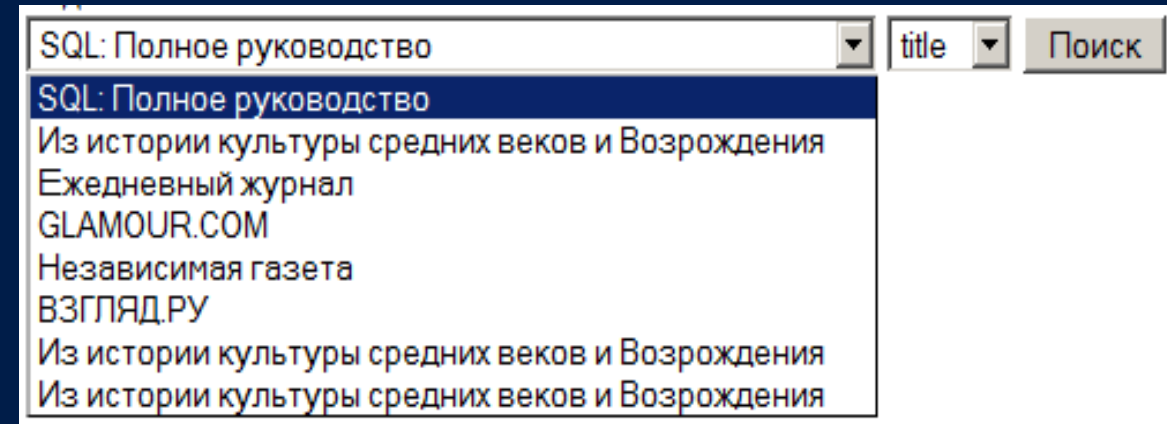
```
?>
```

```
</select>
```

```
<select name="type"> <option>title</option><option>date</option> </select>
```

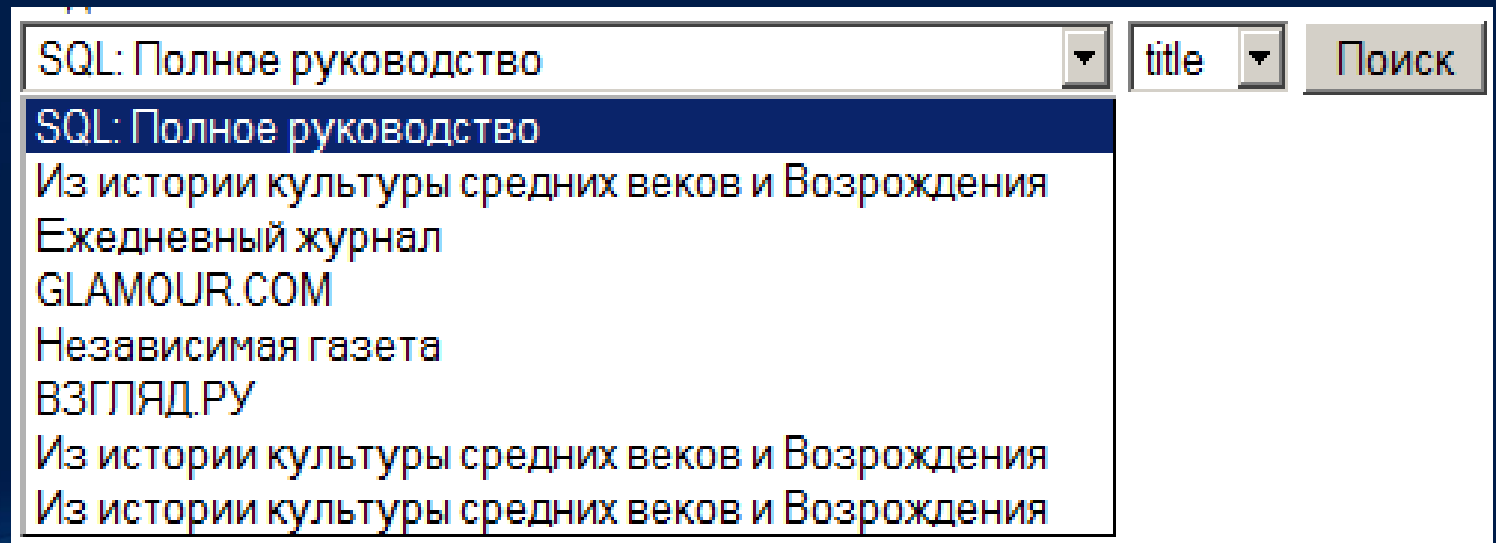
```
<input type="submit" name="form_submit" value="Поиск"><br>
```

```
</form>
```



MySQLi. Пример л/р. Выборка строк из запроса

```
if ($_POST['type']=='title') {  
    $title=$_POST["literat"];  
    $book = mysqli_real_escape_string($link, $title);  
    $query = "select publisher, quantity from literature where name='".$book.'";  
    $buffer = mysqli_query($link, $query);  
}
```



The screenshot shows a web application interface with a search bar. The search bar contains the text "SQL: Полное руководство" and a dropdown arrow. To the right of the search bar is a button labeled "Поиск". Below the search bar, a dropdown menu is open, displaying a list of search results. The first result is "SQL: Полное руководство", which is highlighted. The other results are "Из истории культуры средних веков и Возрождения", "Ежедневный журнал", "GLAMOUR.COM", "Независимая газета", "ВЗГЛЯД.РУ", "Из истории культуры средних веков и Возрождения", and "Из истории культуры средних веков и Возрождения".

Search Results
SQL: Полное руководство
Из истории культуры средних веков и Возрождения
Ежедневный журнал
GLAMOUR.COM
Независимая газета
ВЗГЛЯД.РУ
Из истории культуры средних веков и Возрождения
Из истории культуры средних веков и Возрождения

MySQLi. Пример л/р. Выборка строк из запроса

```
if(isset($buffer)) {  
    echo '<table width="60%" border="1px" >';  
    echo '<tr bgcolor="#CoCoCo"><th>Publisher</th><th>Quantity</th></tr>';  
    while ($result=mysqli_fetch_array($buffer, MYSQLI_ASSOC)) {  
        $publisher=$result['publisher'];  
        $quantity=$result['quantity'];  
        echo "<tr align='center'><td>$publisher</td><td>$quantity</td></tr>";  
    }  
    echo '</table>';  
}  
mysqli_close($link);
```

SQL: Полное руководство	title	Поиск
Publisher		Quantity
Издательство Наука		1000
BHV		2500
Microsoft		500

BENCHMARK

PDO results for 100k queries

<u>query</u>	<u>time in seconds</u>
insert	13.079864025116
select	19.150141954422
update	16.263291120529
delete	14.891561985016

MySQLi results for 100k queries

<u>query</u>	<u>time in seconds</u>
insert	19.463269948959
select	27.461564064026
update	22.705169916153
delete	21.583913087845

PDO results for 1M queries

<u>query</u>	<u>time in seconds</u>
insert	133.66887807846
select	197.49514484406
update	164.97567796707
delete	150.77637290955

MySQLi results for 1M queries

<u>query</u>	<u>time in seconds</u>
insert	202.65716481209
select	283.04056501389
update	231.89973783493
delete	232.20053887367

NoSQL

NoSQL (англ. not only SQL) — ряд подходов, направленных на реализацию хранилищ баз данных, имеющих существенные отличия от моделей, используемых в традиционных реляционных СУБД с доступом к данным средствами языка SQL.

- Key-value



- Graph database



- Document-oriented



- Column family



NoSQL

В NoSQL вместо ACID может рассматриваться набор свойств **BASE**:

базовая доступность (англ. basic availability) — каждый запрос гарантированно завершается (успешно или безуспешно).

гибкое состояние (англ. soft state) — состояние системы может изменяться со временем, даже без ввода новых данных, для достижения согласования данных.

согласованность в конечном счёте (англ. eventual consistency) — данные могут быть некоторое время рассогласованы, но приходят к согласованию через некоторое время.

РСУБД	Пример NoSQL: MongoDB
Database	Database
Table	Collection
Row	Document (JSON, BSON)
Column	Field

MongoDB

Основные особенности:

- Документоориентированное хранение (JSON-подобная схема данных). Коллекции в MongoDB не привязаны к заранее определенной схеме (аналог записей с разным набором полей в одной таблице).
- Open source.
- Эффективное хранение двоичных данных больших объёмов.
- Журналирование операций, модифицирующих данные в базе данных.
- не имеет аналога **join**.
- отсутствие понятия «**транзакции**». Атомарность гарантируется только на уровне целого документа, то есть частичного обновления документа произойти не может.
- отсутствие понятия «**изоляции**». Любые данные, которые считываются одним клиентом, могут параллельно изменяться другим клиентом.

MongoDB

```
{  
  "_id" : ObjectId("5114eobd42..."),  
  "first" : "John",  
  "last" : "Doe",  
  "age" : 39,  
  "interests" : [  
    "Reading",  
    "Mountain Biking ]  
  "favorites": {  
    "color": "Blue",  
    "sport": "Soccer"}  
}
```

```
<?php  
$m = new MongoClient();  
$db = $m->selectDB("demo");  
$collections = $db->listCollections();  
  
foreach ($collections as $collection) {  
  echo "amount of documents in $collection: ";  
  echo $collection->count(), "\n";  
}  
?>
```

Вопросы

- В чем состоят преимущества применения интерфейса доступа к базам данных PDO?
- Каким образом осуществляется управление подключениями к базам данных?
- Как проверить наличие ошибок при соединении с базой данной или выполнении SQL-запроса?
- Для каких целей предназначены подготовленные запросы?
- Каким образом задается значение именованной или неименованной псевдопеременной в подготовленном SQL запросе?
- Какие вы знаете режимы выборки данных? В чем отличие доступа к данным у каждого из режимов?
- Определение транзакции, пример использования?

