

**МОБИЛЬНАЯ
РЕЛЯЦИОННАЯ
СУБД**

ЛИНТЕР®

Linter Standard
Linter Bastion
Linter RealTime
Linter Multiversion

PHР-интерфейсы

НАУЧНО-ПРОИЗВОДСТВЕННОЕ ПРЕДПРИЯТИЕ

 **РЕЛАКС®**

Товарные знаки

РЕЛЭКС™, ЛИНТЕР® , НЕВОД® , LAV™, ЛАКУНА являются товарными знаками, принадлежащими ЗАО НПП «Реляционные экспертные системы» (далее по тексту – компания РЕЛЭКС). Прочие названия и обозначения продуктов являются товарными знаками их производителей, продавцов или разработчиков.

Интеллектуальная собственность

Правообладателем продуктов ЛИНТЕР®, НЕВОД®, LAV™, ЛАКУНА является компания РЕЛЭКС (1990–2011). Все права защищены. Данный документ является собственностью компании РЕЛЭКС. Ни одна часть данного документа не может быть воспроизведена, передана, преобразована, сохранена в системе поиска информации, переведена на другой язык или компьютерный язык в какой-либо форме, какими-либо средствами, электронными, механическими, магнитными, оптическими, химическими, ручными или иными без предварительного разрешения компании РЕЛЭКС.

О документе

Материал, содержащийся в данном документе, прошел тщательную проверку, но компания РЕЛЭКС не гарантирует, что документ не содержит ошибок и пропусков. Компания РЕЛЭКС оставляет за собой право в любое время вносить в документ исправления и изменения, пересматривать и обновлять содержащуюся в нем информацию.

Адрес

394006, г. Воронеж, ул. 20-летия Октября, 119.

Тел./факс: (473) 2-711-711, 2-778-333.

e-mail: market@relex.ru.

Адрес для корреспонденции

394000, г. Воронеж, а/я 137.

Техническая поддержка

Отдел поддержки и сопровождения программных продуктов:

телефон: (473) 2-711-711 с 9:00 до 18:00 мск.

e-mail: support@relex.ru, market@relex.ru.

С целью повышения качества разрабатываемых программных средств и предоставляемых услуг в компании РЕЛЭКС действует автоматизированная система учёта и обработки рекламаций. Обо всех обнаруженных недостатках и ошибках в программном продукте и/или документации на него просим сообщать нам на Internet-странице [рекламация](#).

Оглавление

Предисловие	1
Linter PHP-интерфейс	3
Необходимые условия	3
Использование в среде UNIX.....	3
Настройка PHP-интерфейса СУБД ЛИНТЕР	3
Построение Linter PHP-интерфейса как разделяемой библиотеки.....	4
Особенности использования и сборки	5
Использование в среде Win32	6
Настройка PHP-интерфейса	6
Функции PHP-интерфейса СУБД ЛИНТЕР.....	7
Общие сведения	7
Установить соединение с базой данных	8
Закрыть соединение с базой данных	9
Получить параметры соединения с БД	9
Установить кодовую страницу	11
Открыть курсор	11
Закрыть курсор	12
Получить характеристики курсора	12
Установить характеристики курсора.....	13
Выполнить SQL запрос.....	14
Подготовить запрос к выполнению	15
Получить описание свойств параметра	15
Выполнить подготовленный запрос.....	16
Перейти в заданную строку ответа	16
Получить текущую строку ответа	17
Получить строку ответа и переместиться на следующую	17
Получить заданную строку ответа с кэшированием	18
Получить значения нескольких строк ответа, начиная с текущей	18
Получить значение текущей строки в виде ассоциированного массива	19
Получить строку ответа в виде ассоциированного массива и переместиться на следующую..	20
Получить описание свойства столбца	20
Добавить порцию данных к указанному BLOB-столбцу	21
Добавить порцию данных указанного типа к заданному BLOB-столбцу.....	22
Удалить BLOB-данные заданного столбца.....	23
Получить порцию BLOB-данных заданного столбца.....	23
Получить размер BLOB-данных	24
Добавить порцию данных к BLOB-столбцу	24
Добавить порцию данных указанного типа к BLOB-столбцу	24
Удалить BLOB-данные	24
Получить порцию BLOB-данных	25
Получить последний код завершения	25
Получить описание последнего кода завершения.....	25
Linter DBX-интерфейс	27
Необходимые условия	27
Использование в среде Unix	27
Функции DBX-интерфейса СУБД ЛИНТЕР.....	28

Установить соединение с БД.....	28
Закреть соединение с БД.....	29
Сравнить значения столбцов.....	29
Получить сообщение об ошибке.....	30
Выполнить запрос.....	31
Отсортировать выборку данных.....	33
Lintер PEAR::db-интерфейс.....	34
Использование в среде Win32.....	34
Использование в среде UNIX.....	34
DB-основной класс.....	34
Установить соединение с базой данных.....	34
Проверить на ошибку.....	35
DB_Common-интерфейс для доступа к БД.....	36
Получить количество изменённых строк.....	36
Создать последовательность.....	36
Закреть соединение с базой данных.....	37
Удалить последовательность.....	37
Выполнить подготовленный запрос.....	37
Множественное выполнение запроса.....	38
Получить всю результирующую выборку.....	38
Получить заданный столбец результирующей выборки.....	39
Получить информацию о БД.....	40
Выбрать значение поля.....	40
Получить первую строку результирующей выборки.....	40
Ограничение размера результирующей выборки.....	41
Получить следующий номер последовательности.....	42
Подготовить запрос к выполнению.....	42
Передать запрос на выполнение.....	43
Установить граничные кавычки.....	43
Установить формат по умолчанию для выборки данных.....	44
DB_Result-результирующая выборка.....	44
Поместить заданную строку результирующей выборки в переменную.....	44
Получить заданную строку результирующей выборки.....	45
Освободить ресурсы.....	47
Получить следующую результирующую выборку.....	47
Получить количество столбцов результирующей выборки.....	47
Получить количество строк результирующей выборки.....	48
Получить описание результирующей выборки.....	48
Lintер PDO-интерфейс.....	50
Использование в среде UNIX.....	50
Использование в среде Win32.....	50
PDO-класс соединений.....	51
Соединение с БД.....	51
Закреть соединение с БД.....	52
Начать транзакцию.....	52
Подтвердить транзакцию.....	52
Отменить транзакцию.....	53
Установить атрибут соединения с БД.....	54

Получить значение атрибута соединения с БД.....	56
Получить код завершения.....	57
Получить информацию о коде завершения.....	57
Обрамление строки	58
Подготовить запрос к выполнению	58
Выполнить запрос с предоставлением статистики.....	59
Выполнить запрос	60
Получить последнее значение идентификатора записи или последовательности	61
PDOStatement – класс операторов.....	62
Установить атрибут оператора.....	62
Получить значение атрибута оператора.....	63
Получить код завершения.....	63
Получить информацию о коде завершения.....	64
Выполнить подготовленный оператор	64
Получить количество реально обработанных записей.....	65
Получить число столбцов результирующей выборки	66
Получить описание столбца результирующей выборки	67
Привязать значение к параметру	68
Привязать переменную к параметру	69
Привязать переменную к столбцу.....	70
Установить режим выборки данных	72
Получить заданную строку результирующей выборки.....	74
Получить все строки результирующей выборки	75
Получить значение столбца результирующей выборки	77
Освободить ресурсы.....	77
Получить следующий набор данных	78
PDOException-класс исключений.....	79
Пример PHP-скрипта с использованием PDO-интерфейса.....	79
Приложение 1. Типы данных СУБД ЛИНТЕР	80
Приложение 2. Коды завершения Linter PHP-интерфейса	81
Указатель функций Linter PHP–интерфейса	82
Указатель функций Linter DBX–интерфейса	83
Указатель функций Linter Pear::db –интерфейса	84
Указатель функций Linter PDO–интерфейса	85

Предисловие

Назначение документа

Документ содержит описание PHP-интерфейсов с СУБД ЛИНТЕР.

Документ может использоваться для работы с любой версией СУБД ЛИНТЕР. Особенности конкретных версий оговариваются по тексту.

Для кого предназначен документ

Документ предназначен для программистов, разрабатывающих приложения на языке программирования PHP с использованием СУБД ЛИНТЕР.

Необходимые предварительные знания



Для работы необходимо знать:

- основы реляционных баз данных и языка баз данных SQL;
- язык программирования PHP;
- уметь работать в соответствующей операционной системе.

Принятые обозначения и соглашения

Обозначение	Пример	Значение
Курсив	<i>Растровым</i> называется изображение...	Новый термин в тексте
Полужирный шрифт	В этом случае необходимо переносить все физические файлы.	Выделение в тексте
Подчеркнутый шрифт	Подробную информацию о работе программы можно получить на сайте www.dmk.ru .	Адреса страниц Internet
Текст, разделенный знаком ⇒	Выполните команду View □ Properties (Вид □ Свойства).	Последовательность выполнения команд
Текст, заключенный в <>, со знаком + между ними	<Ctrl>+<C>	В <> заключаются клавиши клавиатуры, знак + означает сочетание клавиш
Крупный моноширинный текст	SQL> _q	Текст командной строки
Мелкий моноширинный текст	Page Time Count	Текст программы
Заглавные буквы	BROWSE	Названия команд, слова, зарезервированные в SQL, ключевые слова
Курсив в <>	<return statement>	Определяемый элемент синтаксической конструкции

Оглавление

Обозначение	Пример	Значение
Символ ::=		Равенство по определению. Слева от знака стоит определяемое понятие, справа – собственно определение понятия
Квадратные скобки []	<code>DBSTORE [-d -n -o -p -r -t -u]</code>	Необязательные элементы конструкции. В данном примере ключи не являются обязательными элементами команды
Вертикальная черта	<code><return value> ::= <value expression> NULL</code>	Указывает на то, что все предшествующие ей элементы списка являются необязательными и могут быть заменены любым другим элементом списка после этой черты
Фигурные скобки { }	<code>CODEPAGE {866 1251 KOI8}</code>	Указывают на то, что все находящееся внутри них является единым целым
Многоточие «...»	Характеристики столбца <code>MAKE CHAR(20) MODEL CHAR(20) ... SQL></code>	Означает, что предшествующая часть может быть повторена любое количество раз
Многоточие, внутри которого находится запятая «,...»		Указывает, что предшествующая часть оператора, состоящая из нескольких элементов, разделенных запятыми, может иметь произвольное число повторений
Текст со знаком  на сером фоне	 Если конфигурация страницы-шаблона не учитывала свойств, команда будет выполнена некорректно.	Примечание

Линтер РНР-интерфейс

Необходимые условия

Для использования РНР-интерфейса СУБД ЛИНТЕР на хост-компьютере должен быть установлен интерпретатор языка программирования РНР либо как исполняемый модуль, либо как разделяемая библиотека (модуль) Web-сервера Apache.

Если интерпретатор РНР не установлен, то его исходные тексты и инструкцию по установке можно найти на сайте <http://www.php.net>.

РНР-интерфейс СУБД ЛИНТЕР поддерживает третью, четвертую, пятую и шестую версию РНР.

Использование в ОС QNX совпадает по всем параметрам с использованием в среде UNIX и, для краткости, указание на QNX будет опущено.

Использование в среде UNIX

В составе дистрибутива СУБД ЛИНТЕР поставляются готовые к использованию загружаемые библиотеки РНР-интерфейса, а также исходные тексты библиотек, необходимых для сборки программных интерфейсов ЛИНТЕР. Существует множество версий РНР и модули для каждой версии необходимо собирать отдельно. РНР-интерфейсы ЛИНТЕР работают со всем множеством подверсий РНР 3,4,5 и 6.

Настройка РНР-интерфейса СУБД ЛИНТЕР

Для настройки Линтер РНР-интерфейса:

1. установить (если это не было сделано ранее) интерпретатор языка программирования РНР в операционную систему в виде исполняемого файла или в виде модуля к Web-серверу Apache (см. документацию на РНР и Apache);
2. определить путь к файлу конфигурации интерпретатора РНР и к его разделяемым библиотекам. Для этого необходимо выполнить следующую программу на РНР:

Начало программы `info.php`

```
<?РНР  
phpinfo();  
?>
```

Конец программы `info.php`


Выполнить ее можно двумя способами, в зависимости от установки РНР:

- если интерпретатор РНР используется Web-сервером Apache, то нужно поместить программу в каталог html-документов Web-сервера и выбрать ее как гипертекстовую ссылку из Web-браузера (Netscape, Microsoft Internet Explorer или любой другой);
- если интерпретатор РНР установлен в виде исполняемого модуля, то необходимо выполнить команду запуска программы и результат перенаправить в файл:

```
>php info.php > phpinfo.html
```

3. открыть файл `phpinfo.html` из Вашего Web-браузера и найти в нем опции:
 - `php.ini file path is set to:` (путь к файлу `php.ini`);
 - `extension_dir` (каталог расширений).

4. скопировать файл `libphp3_linter.so` (для версии PHP 3) или `libphp_linter.so` (для PHP 4,5 или 6) из подкаталога `bin` установочного каталога СУБД ЛИНТЕР в каталог расширений PHP или отредактировать опцию `extension_dir` файла `php.ini`, чтобы она указывала на подкаталог `bin` установочного каталога СУБД ЛИНТЕР.
5. выполнить одно из действий:
 - PHP-команду `d1("libphp_linter.so")` для PHP 4,5,6 или `d1("libphp3_linter.so")` для PHP 3 для динамической загрузки PHP-интерфейса в ходе выполнения PHP-программы;
 - внести строку `"extension=libphp_linter.so"` в файл `php.ini` для загрузки Linter PHP-интерфейса перед запуском любых PHP-программ.

 Для загрузки Linter PHP-интерфейса можно использовать также универсальный код, описанный в файле `samples/php/library.php` дистрибутива СУБД ЛИНТЕР.

Построение Linter PHP-интерфейса как разделяемой библиотеки

Построение (сборка) PHP-интерфейса может потребоваться в следующих случаях:

- 1) в состав дистрибутива СУБД ЛИНТЕР не входили готовые к использованию, откомпилированные разделяемые библиотеки Linter PHP-интерфейса;
- 2) установленная на Вашем компьютере версия интерпретатора PHP не совпала с версией, для которой собран PHP-интерфейс СУБД ЛИНТЕР.

Для сборки необходимо иметь:

- установленные заголовочные файлы (или исходные тексты интерпретатора PHP);
- C-компилятор и набор утилит для сборки (`make`, `ld` и т. п.).

Для построения PHP-интерфейса СУБД ЛИНТЕР:

1. запустить программу (скрипт) конфигурации `configure` из корневого каталога дистрибутива СУБД ЛИНТЕР;
2. ответить утвердительно на вопрос о конфигурации дистрибутива для сборки PHP-интерфейсов;
3. определить местоположение заголовочных файлов PHP:
 - методом поиска файлов;
 - явным указанием полного пути (если он известен до выполнения данной операции). Если при вводе была допущена ошибка (по указанному пути файлы не найдены), будет предложено повторить ввод. Отказ от повторного ввода равносителен отказу от сборки PHP-интерфейса для данной версии PHP;
 - редактированием файла `Definition` дистрибутива СУБД ЛИНТЕР вручную, установив переменные `SUPPORT_PHP` или `SUPPORT_PHP4` в "1" для третьей и более поздних версий соответственно. Переменные `PHPINCLUDE` или `PHP4INCLUDE` должны определять полный путь к заголовочным файлам PHP.
4. перейти в каталог `php` дистрибутива СУБД ЛИНТЕР и выполнить команду `make` для сборки интерфейсов. После окончания компиляции и компоновки готовые к использованию разделяемые библиотеки PHP-интерфейса будут находиться в подкаталоге `bin` установочного каталога СУБД ЛИНТЕР.

Особенности использования и сборки

В некоторых случаях можно включить путь к каталогу разделяемых библиотек СУБД ЛИНТЕР (bin) в переменную окружения `LD_LIBRARY_PATH` и использовать команду `d1` без правки `php.ini` файла.

Если PHP-интерпретатор третьей версии установлен в виде исполняемого модуля, то динамически загружаемые расширения не будут работать на некоторых платформах (Linux, FreeBSD). Для активизации загружаемых расширений необходимо перестроить PHP-интерпретатор с добавлением программе сборки опции, предписывающей экспортировать все имена собираемого модуля. Для GNU ld, используемом в FreeBSD и Linux, это опция `-export-dynamic` или `-E`. Т.е. в `makefile` исходных текстов PHP в строку, определяющую переменную `LD_FLAGS`, необходимо добавить `-Wl,-E`. После сборки и установки PHP, согласно документации, и настройки, как описано выше, можно использовать функцию `d1` для загрузки PHP-интерфейса.

Для сборки PHP третьей версии со статическим интерфейсом СУБД ЛИНТЕР:

1. в модуль `internal_functions.c` исходных текстов PHP включить файл `php3_linter.h`. В массив структур `php3_builtin_modules` добавить элемент `{"Linter", linter_module_ptr}`;
2. в корневой `makefile` в макро `INCLUDE` добавить `I<LINTER_HOME>/intl1ib`
3. в макро `LIBS` добавить `L<LINTER_HOME>/intl1ib -lphp3_linter`
где `<LINTER_HOME>` - путь к дереву каталогов дистрибутива СУБД ЛИНТЕР;
4. выполнить `make`.

Если сборка прошла без ошибок, PHP3-интерфейс готов к работе с СУБД ЛИНТЕР. В этом случае команда `d1` для загрузки PHP-интерфейса не используется – интерфейс всегда готов к работе.

Если используется СУБД ЛИНТЕР с IPC-версией межзадачного обмена, то PHP-интерфейс не будет работать в среде UNIX с PHP, используемом в виде модуля Apache (В настоящее время подобные проблемы могут наблюдаться только для версии ЛИНТЕР 5.7).

Для получения информации о версии СУБД ЛИНТЕР выполните команду:

```
linter -version
```

Наличие строки `IPC` в тексте версии говорит о необходимости дополнительной настройки и перестройки Apache. Для этого перед последней строкой файла `src/include/httpd.h` дистрибутива Apache, содержащей `#endif`, включить строки:

```
#undef SIGUSR1
#define SIGUSR1 SIGHUP
```

и закомментировать выбор (case), относящийся к `SIGUSR1` в файле `src/main/http_main.c` в строке, где компилятор обнаружит ошибку. В остальном Apache собирается и устанавливается согласно документации.

Использование в среде Win32

Настройка PHP-интерфейса

Для настройки PHP-интерфейса СУБД ЛИНТЕР в Win32:

1. поместить интерфейсную библиотеку для соответствующей версии PHP из каталога `intl\php\xxx` в каталог, в котором PHP ищет загружаемые ресурсы. Необходимо подбирать версию библиотеки максимально совпадающую с версией установленного у Вас PHP. Постфикс `_MT` указывает на то, что в данном каталоге лежит модуль, собранный для работы с многопоточной версией PHP. Если в Вашем дистрибутиве ЛИНТЕР отсутствует `dll`, которая соответствует Вашей версии PHP, попытайтесь использовать максимально близкую по версии. Если с ней возникнут проблемы – обратитесь к разработчику за модулем, который будет точно соответствовать Вашей версии PHP.

Для того чтобы найти место, куда поместить PHP-интерфейс, необходимо:

- открыть конфигурационный файл `<WINSYS32DIR>/php.ini` (этот файл может располагаться и в другом каталоге. Определить точный путь к файлу можно так, как указано в разделе конфигурирования PHP-интерфейса для UNIX);
- найти в нем строку `extension_dir=...`

Если в ней уже задан какой-либо каталог – скопировать `php(x)_linter.dll` в этот каталог. Если нет – убрать комментарий в строке и прописать в ней путь к каталогу, где находятся библиотеки `php(x)_linter.dll`.

2. если необходимо, чтобы библиотека `php(x)_linter.dll` всегда автоматически подгружалась при исполнении любой PHP-программы, в разделе `Extensions` добавить строку:
`extension=php(x)_linter.dll`
3. перезапустить Web-сервер (если PHP используется для WEB-приложения).

Функции РНР-интерфейса СУБД ЛИНТЕР

Общие сведения

- 1) Все символьные параметры функций чувствительны к регистру символов.
- 2) Если длина символьного параметра функции превышает максимально допустимую длину в СУБД ЛИНТЕР, то значение параметра усекается до допустимой длины и функция выполняется с усечённым значением параметра.
- 3) Функции возвращают результат типа `int`, `string`, `object` или `array`. В случае ошибки всегда возвращается `int`.

Возможные коды завершения:

<u>Возвращаемое значение</u>	<u>Описание</u>
0 (LPE_SUCCESS)	Нормальное завершение, возвращаемое значение отсутствует
Положительное	Возвращаемое функцией значение (нормальное завершение)
Отрицательное	Неудачное завершение функции.

- 4) Причиной неудачного завершения функции может являться как ошибка РНР-модуля, так и результат обработки функции СУБД ЛИНТЕР. Если причиной является код завершения СУБД ЛИНТЕР, то возвращается `LPE_LINTER_ERROR`. Все остальные отрицательные коды относятся к ошибкам РНР-модуля (см стр. **Ошибка! Закладка не определена.**). Для получения дополнительной информации об ошибке (в том числе и детализации `LPE_LINTER_ERROR`.) используйте функцию `Linter_Last_Error` (см. стр. 24).

Пример обработки кода завершения

```
<?
#
# Linter SQL Server and Linter PHP module errors handler
# Parameters: $con - connection or cursor id;
#             $err - value was returned by any Linter_*
#             routine.
#
function Errors_Handler($con, $err)
{
    if ($err >= 0 && $con >= 0)
        return;
    if ($err == LPE_LINTER_ERROR || $con < 0)
    {
        $lin_err = Linter_Last_Error($con, LINTER_ERROR);
        $sys_err = Linter_Last_Error($con, SYSTEM_ERROR);
        $err_msg = Linter_Error_Msg($con);
        if ($lin_err <= 2) // no more rows
            return;
        printf("Linter error %d (%s) ", $lin_err, $err_msg);
        if ($lin_err >= 2000 && $lin_err < 3000) // SQL syntax
            errors
            {
                $row = $sys_err & 0xFFFF;
                $pos = $sys_err >> 16;
            }
    }
}
```

```
        printf("at row %d, position %d", $row, $pos);
    }
    else
        printf(" system error %d", $sys_err);
    printf("\n");
}
else
    printf("Module internal error %d\n", $err);
Linter_Close_Connect($con);
die;
}
?>
```

- 5) Примеры demo(x).php из подкаталога samples/php установочного каталога СУБД ЛИНТЕР предназначены для запуска из командной строки. Для просмотра их в браузере используйте примеры с постфиксом html, например, demo3html.php

Установить соединение с базой данных

Назначение

Функция `Linter_Open_Connect` устанавливает связь между PHP-модулем и СУБД ЛИНТЕР с заданным именем пользователя и паролем. В функции может быть указано имя ЛИНТЕР-сервера. Если имя ЛИНТЕР-сервера не указано, то модуль подключится к локальной БД или к серверу по умолчанию.

Синтаксические правила

`int Linter_Open_Connect (пользователь, пароль, сервер, режим);`

Пользователь

Имя пользователя БД. Символьная строка длиной не более 66 символов.

Пароль

Пароль пользователя. Символьная строка длиной не более 18 символов.

Сервер

Имя ЛИНТЕР-сервера, с которым необходимо установить соединение. Символьная строка длиной не более 8 символов.

Режим

Задаёт режим обработки транзакций и/или кодовую страницу. Режим обработки транзакций по умолчанию – AUTOCOMMIT. Кодовая страница по умолчанию зависит от ОС. В версиях ЛИНТЕР 5.x в Win32 кодировка по умолчанию 866, а в UNIX – KOI8. В версиях ЛИНТЕР 6.x в Win32 кодировка определяется по кодировке текущей задачи (ОЕМ, если не указано обратное), а в UNIX – по значению переменной LANG. Кодировку по умолчанию можно задать также переменной окружения LINTER_CP (см. документ «СУБД ЛИНТЕР. Интерфейс нижнего уровня»).

Формат:

[режим транзакции] [| кодовая страница]

Режимы транзакции:

- TM_AUTOCOMMIT;
- TM_OPTIMISTIC;
- TM_EXCLUSIVE;
- TIL_READ_COMMITTED (только для версии 6.1);
- TIL_READ_UNCOMMITTED (только для версии 6.1);
- TIL_SERIALIZABLE (только для версии 6.1).

Кодовые страницы (для версии ЛИНТЕР 5):

- CP_1251;
- CP_KOI8;
- CP_866.

Параметр может быть равен 0 (используются значения по умолчанию).

Возвращаемое значение

Идентификатор соединения с БД

Закреть соединение с базой данных

Назначение

Функция `Linter_Close_Connect` закрывает соединение с БД.

Синтаксические правила

`int Linter_Close_Connect` (идентификатор_соединения);



Если соединение является родителем одного или нескольких курсоров, то курсоры тоже будут закрыты.

Получить параметры соединения с БД

Назначение

Функция `Linter_Connect_Info` позволяет получить параметры для заданного соединения по его номеру.

Синтаксические правила

`Int|array Linter_Connect_Info` (идентификатор_соединения);

Идентификатор_соединения

Номер соединения или курсора

Возвращаемое значение

Ассоциированный массив, содержащий параметры указанного соединения. В зависимости от версии ядра СУБД ЛИНТЕР или других условий этот массив может быть пустым (не содержать значений) или содержать часть ниже перечисленных возможных параметров соединения. Программа, которая использует данную функцию, должна предполагать, что не все указанные параметры могут быть возвращены или могут быть

возвращены дополнительные параметры. В последнем случае, необходимо обратиться за обновлением документации к разработчику.

Параметр	Описание
VERMAJOR	Старший номер версии ядра ЛИНТЕР
VERMINOR	Младший номер версии ядра ЛИНТЕР
VERBUILD	Номер сборки версии ядра ЛИНТЕР
MAXRECSIZE	Предельная длина записи в таблице БД (версия 6 и выше)
SORTPOOLSIZE	Размер пула (в страницах) подсистемы сортировки
KERNELPOOLSIZE	Размер пула (в страницах) ядра системы
FILEQUEUE SIZE	Размер очереди файлов
USERQUEUE SIZE	Размер очереди пользователей
TABLEQUEUE SIZE	Размер очереди таблиц
COLUMNQUEUE SIZE	Размер очереди столбцов
CHANNELQUEUE SIZE	Размер очереди каналов
SNAPTIMEOUT	Период времени между операциями полного Snap
KILLTIMEOUT	Таймаут опроса существования клиента
BASENAME	Имя базы данных
SYSLOG	Признак активности журнала транзакций
SYNC	Признак полной синхронности ввода/вывода
LOG	Признак ведения файла протокола
OS	Идентификатор операционной системы (полный список идентификаторов операционных систем см. в документе «СУБД ЛИНТЕР. Интерфейс нижнего уровня»)
NUMOFSORT	Количество процессов сортировки
FLAGS_KERNELINVBYTEORDER	Порядок байтов в данной БД относительно данного клиента (0 – порядок байт у сервера и клиента совпадают; 1– порядок байт у сервера и клиента различаются)
FLAGS_KERNELDEMOLIC	Признак использования демонстрационной версии СУБД ЛИНТЕР
FLAGS_KERNELDEMOLICEXP	Признак истечения срока лицензии СУБД ЛИНТЕР
FLAGS_KERNELENGLOCALE	Ядро использует европейские кодировки вместо русских (для версий 5.x)
FLAGS_CSETCHG	Указанная клиентом в процессе подключения кодовая страница не может быть использована (для версий 6.x)
CHARSET	Идентификатор кодовой страницы данной БД (список доступных кодовых страниц находится в системной таблице \$\$\$CHARSET) (для версий 6.x)
DEFCHARSET	Кодовая страница по умолчанию для БД (для версий

<u>Параметр</u>	<u>Описание</u>
USECHARSET	6.x) Кодовая страница соединения (для версий 6.x)
USECHARSETNAME	Имя кодовой страницы, в которой открыто соединение (для версий 6.x)

Установить кодовую страницу

Назначение

Функция `Linter_Set_Codpage` устанавливает заданную кодовую страницу (только для версий 6.x).

Синтаксические правила

`int Linter_Set_Codpage (кодовая_страница);`

Действие установленной кодовой страницы распространяется на новые создаваемые соединения и курсоры. Для версий ЛИНТЕР 5.x значение игнорируется.

Возвращаемое значение

Возвращает `LPE_LINTER_ERROR` в случае неудачи или `LPE_SUCCESS` в случае успешного завершения.

Пример

См. пример `demo7.php` в подкаталоге `samples/php` установочного каталога СУБД ЛИНТЕР.

Открыть курсор

Назначение

Функция `Linter_Open_Cursor` открывает дочернее соединение (курсor) между PHP-модулем и СУБД ЛИНТЕР.

Синтаксические правила

`int Linter_Open_Cursor (идентификатор_соединения);`

Идентификатор_соединения

Идентификатор родительского соединения.

Возвращаемое значение

Идентификатор нового курсора.

Пример

См. пример `demo4.php` в подкаталоге `samples/php` установочного каталога СУБД ЛИНТЕР.

Закрывать курсор

Назначение

Функция `Linter_Close_Cursor` закрывает указанный курсор.

Синтаксические правила

```
int Linter_Close_Cursor (идентификатор_курсора);
```

Получить характеристики курсора

Назначение

Функция `Linter_Get_Cursor_Opt` позволяет получить характеристики курсора.

Синтаксические правила

```
string |int Linter_Get_Cursor_Opt(идентификатор_курсора,  
характеристика_курсора);
```

Возможные значения параметра характеристика_курсора:

<u>Значение</u>	<u>Описание</u>
<code>CO_NAME</code>	Имя курсора
<code>CO_DT_FORMAT</code>	Формат дата/время
<code>CO_COL_COUNT</code>	Число столбцов в ответе
<code>CO_ROW_COUNT</code>	Число строк в ответе
<code>CO_ERR_ROW</code>	Номер строки, содержащей ошибку
<code>CO_ERR_POS</code>	Положение ошибки в строке
<code>CO_CUR_ROW</code>	Текущая строка курсора
<code>CO_CUR_ROWID</code>	Идентификатор текущей строки курсора
<code>CO_CONNECT_ID</code>	Реальный номер канала связи с СУБД ЛИНТЕР
<code>CO_NODE_NAME</code>	Имя узла ЛИНТЕР-сервера
<code>CO_LAST_ROWID</code>	Последний изменённый, добавленный или удалённый ROWID
<code>CO_UPDATABLE</code>	Тип курсора: 1 – обновляемый; 0 – не обновляемый
<code>CO_PARAM_COUNT</code>	Количество параметров транслируемого запроса
<code>CO_FETCH_BLOBS_AS_USUAL_DATA</code>	Текущий режим извлечения BLOB-данных
<code>CO_QUERY_TYPE</code>	Тип курсорного запроса: 1 – SELECT-запрос; 2 – EXECUTE-запрос; 3 – GRANT;

Значение	Описание
	4 – ALTER;
	0 – остальные запросы.

Возвращаемое значение

Характеристика курсора	Тип возвращаемого значения
CO_NAME	string
CO_DT_FORMAT	string
CO_COL_COUNT	integer
CO_ROW_COUNT	integer
CO_ERR_ROW	integer
CO_ERR_POS	integer
CO_CUR_ROW	integer
CO_CUR_ROWID	integer
CO_CONNECT_ID	integer
CO_NODE_NAME	string
CO_LAST_ROWID	integer
CO_UPDATABLE	integer
CO_PARAM_COUNT	integer
CO_FETCH_BLOBS_AS_USUAL_DATA	integer
CO_QUERY_TYPE	integer

Пример

```
$upd = Linter_Get_Cursor_Opt($con, CO_UPDATABLE);
$typ = Linter_Get_Cursor_Opt($con, CO_QUERY_TYPE);
printf("upd %d type %d \n", $upd, $typ);
```

См. также пример `demo4.php` в подкаталоге `samples\php` установочного каталога СУБД ЛИНТЕР.

Установить характеристики курсора

Назначение

Функция `Linter_Set_Cursor_Opt` позволяет установить характеристики курсора.

Синтаксические правила

```
int Linter_Set_Cursor_Opt (идентификатор_курсора, характеристика_курсора, значение);
```

Возможные значения параметра характеристика_курсора:

<u>Значение</u>	<u>Описание</u>
CO_NAME	Имя курсора
CO_DT_FORMAT	Формат дата/время
CO_FETCH_BLOBS_AS_USUAL_DATA	Режим извлечения BLOB-данных: 1 – позволяет извлекать BLOB-данные как обычные данные (в результате вызова Fetch); 0 – позволяет извлекать BLOB-данные с помощью специальных функций.

Пример

См. пример demo4.php в подкаталоге samples\php установочного каталога СУБД ЛИНТЕР.

Выполнить SQL запрос

Назначение

Функция Linter_Exec_Direct осуществляет выполнение SQL-запроса.

Синтаксические правила

`int Linter_Exec_Direct` (идентификатор_курсора, SQL-запрос);

Текст SQL-запроса должен завершаться символом ';' ;

Запрос выполняется без предварительной трансляции и привязки параметров.

Запрос может быть любым, который поддерживается СУБД ЛИНТЕР

Если процедура возвращает курсор, то результатом будет массив с единственным элементом – идентификатором курсора.

Если запрос возвращает курсор, то после выполнения функции по идентификатору_курсора можно выполнять функции получения данных.

Если выполняется запрос на исполнение хранимой процедуры, то результат можно получить через функцию Linter_Get_Data_Row. Если хранимая процедура вернула курсор, то к нему, в свою очередь, можно применять функции извлечения данных.

Пример

```
linter_exec_direct ( $connection_id, "execute test_proc()");  
$procedure_result = linter_get_data_row ( $connection_id);  
$data = linter_get_data_row ( $procedure_result [0]);
```

После выполнения последнего оператора \$data содержит первую строку данных из курсора.

См. также примеры demo1.php, demo6.php в подкаталоге samples\php установочного каталога СУБД ЛИНТЕР.

Подготовить запрос к выполнению

Назначение

Функция `Linter_Prepere` позволяет подготовить запрос для выполнения, предварительно оттранслировав его.

Синтаксические правила

`int Linter_Prepere` (идентификатор_курсора, SQL_запрос);

После трансляции появляется возможность определить количество и тип параметров запроса, указать значения параметров и исполнить запрос.

Если в запросе нет параметров, то он может быть исполнен или без предварительной трансляции (с помощью `Linter_Execute_Direct`) или с помощью `Linter_Prepere/Linter_Execute`. В случае, если в запросе указаны параметры, то обязательно должны использоваться функции `Linter_Prepere/Linter_Execute`.

Пример

См. пример `demo2.php` в подкаталоге `samples\php` установочного каталога СУБД ЛИНТЕР.

Получить описание свойств параметра

Назначение

С помощью функции `Linter_Get_Param_Prop` можно получить описание параметра подготовленного запроса.

Синтаксические правила

`object | int Linter_Get_Param_Prop` (идентификатор_курсора, номер_параметра);

Количество параметров можно узнать с помощью функции `Linter_Get_Cursor_Opt` с характеристикой `CO_PARAM_COUNT`.

В случае, если задан неверный номер параметра, возвращается ошибка `LPE_INVALID_COL_NUM`.

Нумерация параметров начинается с 1.

Возвращаемое значение

Описание свойств заданного столбца в виде следующей структуры:

<u>Свойства объекта</u>	<u>Тип</u>	<u>Описание</u>
<code>name</code>	<code>string</code>	Имя параметра (если задано. Если нет – пустая строка)
<code>type</code>	<code>integer</code>	Тип параметра (см. описание в документе «СУБД ЛИНТЕР. Интерфейс нижнего уровня»)
<code>type_name</code>	<code>string</code>	Имя типа параметра
<code>length</code>	<code>integer</code>	Длина параметра
<code>precision</code>	<code>integer</code>	Точность параметра (имеет смысл только для типа данных <code>DECIMALS</code>)
<code>scale</code>	<code>integer</code>	Масштаб параметра (имеет смысл только для типа данных <code>DECIMALS</code>)

Выполнить подготовленный запрос

Назначение

Функция `Linter_Execute` осуществляет выполнение подготовленного запроса.

Синтаксические правила

`int Linter_Execute` (идентификатор_курсора, [значение_параметра [, ...]]);

Значение_параметра

Значение, которое должно быть присвоено параметру в подготовленном запросе.

Число элементов списка должно быть равно числу параметров в подготовленном запросе.

Если число элементов списка больше, то лишние параметры игнорируются.

Обработка результатов запроса может проводиться так же, как указано при описании функции `Linter_Execute_Direct`.

Перейти в заданную строку ответа

Назначение

Функция `Linter_Fetch` осуществляет переход в заданную строку ответа.

Синтаксические правила

`int Linter_Fetch` (идентификатор_курсора, указатель_строки, [номер_строки]);

Возможные значения параметра указатель_строки:

<u>Значение</u>	<u>Описание</u>
<code>FETCH_FIRST</code>	Выбрать первую строку
<code>FETCH_LAST</code>	Выбрать последнюю строку
<code>FETCH_NEXT</code>	Выбрать следующую строку
<code>FETCH_PREV</code>	Выбрать предыдущую строку
<code>FETCH_ABSNUM</code>	Выбрать строку по абсолютному номеру

Если указателем_строки задан `FETCH_ABSNUM`, то должен быть указан номер искомой строки.

Функция применима только в случае, если по идентификатору_курсора был исполнен запрос `SELECT` (или идентификатор возвращён после исполнения хранимой процедуры, возвращающей курсор).

При переходе к указанной строке ответа никакие действия над данными не выполняются, проверяется только доступность строки.

Нумерация строк начинается с 1.

Пример

См. пример `demo8.php` в подкаталоге `samples\php` установочного каталога СУБД ЛИНТЕР.

Получить текущую строку ответа

Назначение

Функция `Linter_Get_Data_Row` позволяет получить значение текущей строки ответа или выходные значения хранимой процедуры.

Синтаксические правила

`array | int Linter_Get_Data_Row (идентификатор_курсора [,NULL_индикатор]);`

Функция применима только в случае, если по идентификатору_курсора был исполнен запрос `SELECT` (или идентификатор возвращён после исполнения хранимой процедуры, возвращающей курсор). Перед её использованием может быть вызвана функция `Linter_Fetch`.

Если вызвать данную функцию сразу после исполнения запроса, то будет возвращена первая строка ответа.

Возвращаемое значение

Строка ответа в виде одномерного массива данных. Доступ к элементу массива (строки) выполняется по относительному номеру элемента в массиве.

Если `NULL_индикатор` задан и не равен 0, то данные возвращаются в формате:

`{значение 1 столбца}{признак NULL 1 столбца}[, ...].`

то есть нечетные элементы массива содержат собственно значения, четные – признак `NULL`-значения.

Если установлен режим `CO_FETCH_BLOBS_AS_USUAL_DATA`, то `BLOB`-данные (если есть) помещаются в результирующий массив.

Если по идентификатору_курсора был исполнен запрос на выполнение хранимой процедуры, то первым элементом возвращаемого массива будет значение, возвращаемое процедурой. Если это курсор, то его потом можно использовать во всех функциях `PHP-интерфейса`.

Если процедура завершилась исключением, то массив содержит единственный элемент - код исключения.

Пример

См. пример `demo1.php` в подкаталоге `samples\php` установочного каталога СУБД `ЛИНТЕР`.

Получить строку ответа и переместиться на следующую

Назначение

Функция `Linter_Fetch_Row` возвращает данные текущей строки ответа и осуществляет переход в следующую строку.

Синтаксические правила

`array | int Linter_Fetch_Row (идентификатор_курсора [,NULL_индикатор]);`

Функции Linter PHP-интерфейса

Функция `Linter_Fetch_Row` работает аналогично функции `Linter_Get_Data_Row`. Ее отличие в том, что после получения данных она автоматически позиционируется на следующую строку, что позволяет в цикле выбирать все данные ответа.

Возвращаемое значение

Строка ответа в виде одномерного массива данных или `LPE_LINTER_ERROR`, если все строки были уже выданы. Если установлен режим `CO_FETCH_BLOBS_AS_USUAL_DATA`, то BLOB-данные (если есть) помещаются в результирующий массив.

Пример

См. пример `demo2.php` в подкаталоге `samples\php` установочного каталога СУБД ЛИНТЕР.

Получить заданную строку ответа с кэшированием

Назначение

Функция `Linter_GETM` позволяет получить заданную строку ответа. В процессе работы функция получает данные большими порциями и кэширует их на клиенте.

Синтаксические правила

`array | int Linter_GETM (идентификатор_курсора, номер_строки);`

Нумерация строк начинается с 1.

С помощью этой функции нельзя получить признаки NULL-значений и BLOB-данные

Возвращаемое значение

Строка ответа в виде одномерного массива данных. Доступ к элементу массива (строки) выполняется по относительному номеру элемента в массиве.

Получить значения нескольких строк ответа, начиная с текущей

Назначение

Функция `Linter_Get_Data_Rows` позволяет получить значения нескольких строк ответа. Функция всегда возвращает признак NULL-значения.

Синтаксические правила

`array | int Linter_Get_Data_Rows (идентификатор_курсора, количество_строк);`


Возвращаемое значение

Двумерный массив данных из столбцов ответа из заданного количества_строк. Доступ к элементу массива данных выполняется по номеру строки и номеру столбца элемента в массиве.

Если установлен режим `CO_FETCH_BLOBS_AS_USUAL_DATA`, то BLOB-данные (если есть) помещаются в соответствующий элемент.

Пример

```
...
$cols = Linter_Get_Cursor_Opt($cursor_id,
                               CO_COL_COUNT);
$rows = Linter_Get_Cursor_Opt($cursor_id,
                               CO_ROW_COUNT);
$data = Linter_Get_Data_Rows($cursor_id, $rows);
if (is_array($data))
{
    for($i = 0; $i < $rows; $i++)
    {
        for($j = 0; $j < $cols; $j++)
            printf("%s ", $data[$i][$j]);
        print("\n");
    }
}
...
```

 Количество строк в массиве ответа может быть меньше требуемого, если достигнут конец выборки.

Получить значение текущей строки в виде ассоциированного массива

Назначение

Функция `Linter_Get_Data_Array` позволяет получить значение текущей строки в виде ассоциированного массива.

Синтаксические правила

`array | int Linter_Get_Data_Array (идентификатор_курсора);`

Возвращаемое значение


Строка ответа в виде массива данных. Доступ к элементу массива (строке) данных выполняется по имени столбца в курсорном запросе.

Если установлен режим `CO_FETCH_BLOBS_AS_USUAL_DATA`, то BLOB-данные (если есть) помещаются в соответствующий элемент.

Пример

После выполнения запроса
`select id,"name" from table;`
 доступ к столбцам ответа возможен по именам столбцов: `ans["ID"], ans["name"]`.

См. также пример `demo1.php` в подкаталоге `samples\php` установочного каталога СУБД ЛИНТЕР.

 Имена столбцов чувствительны к регистру, то есть `ans["id"]` и `ans["ID"]` – разные столбцы.

Получить строку ответа в виде ассоциированного массива и переместиться на следующую

Назначение

Функция `Linter_Fetch_Array` позволяет получить текущую строку ответа в виде ассоциированного массива и перейти на следующую строку. Функция позволяет последовательными вызовами перебрать все строки выборки.

Синтаксические правила

`array | int Linter_Fetch_Array (идентификатор_курсора);`


Функция `Linter_Fetch_Array` работает аналогично функции `Linter_Get_Data_Array`. Ее отличие в том, что после получения данных она автоматически позиционируется на следующую строку, что позволяет в цикле выбирать все данные ответа.

Возвращаемое значение

- 1) строка ответа в виде массива данных. Доступ к элементу массива (строки) выполняется по имени столбца в запросе;
- 2) `LPE_LINTER_ERROR`, если все строки были уже выданы.

Пример

См. пример `demo2.php` в подкаталоге `samples\php` установочного каталога СУБД ЛИНТЕР.

 Имена столбцов чувствительны к регистру, то есть `ans["id"]` и `ans["ID"]` – разные столбцы.

Получить описание свойства столбца

Назначение

Функция `Linter_Get_Col_Prop` предоставляет описание свойств столбца ответа.

Синтаксические правила

`object | int Linter_Get_Col_Prop (идентификатор_курсора, номер_столбца);`

Нумерация столбцов начинается с 1.

Если задан неверный номер столбца, то возвращается код завершения `LPE_INVALID_COL_NUM`.

Возвращаемое значение

Описание свойств заданного столбца в виде следующей структуры:

<u>Свойства объекта</u>	<u>Тип</u>	<u>Описание</u>
Name	string	Имя столбца
Table	string	Имя таблицы, из которой выбран столбец (если есть)
User	string	Владелец таблицы, из которой выбран столбец (если есть)

Свойства объекта	Тип	Описание
Type	integer	Тип столбца (см. документ «СУБД ЛИНТЕР. Интерфейс нижнего уровня»)
type_name	string	Имя типа столбца
Length	integer	Длина столбца
precision	integer	Точность столбца (имеет смысл только для типа данных DECIMAL)
Scale	integer	Масштаб столбца (имеет смысл только для типа данных DECIMAL)
is_null	integer	Индикатор NULL-значения для текущей строки. Альтернативный способ получения значения NULL-индикатора.

Пример

См. пример `demo3.php` в подкаталоге `samples\php` установочного каталога СУБД ЛИНТЕР.

Добавить порцию данных к указанному BLOB-столбцу

Назначение

Функция `Linter_Blob_Add` добавляет порцию данных к указанному BLOB-столбцу в текущей строке курсорного запроса.

Синтаксические правила

`int Linter_Blob_Add` (идентификатор_курсора, порция_данных, размер_порции, номер_столбца);

Порция_данных

Данные для добавления


Размер_порции

Количество байт из параметра «порция_данных», которые будут добавлены к BLOB.

Номер_столбца

Параметр задает номер столбца, к которому должна быть добавлена порция данных.

Нумерация столбцов в строке начинается с 1.

 Примечание для всех функций, работающих с BLOB-данными.

Аргумент «номер столбца» задает порядковый номер столбца в текущей записи предшествующего запроса `SELECT/INSERT/UPDATE`, а не в исходной таблице.

Например, после выполнения последовательности операторов

```
create or replace table tst(i int autoinc, d int default 0, b1 blob);
insert into tst(b1) values ('');
update tst set d=100, b1='BLOB-данные';
```

для добавления порции BLOB-данных в аргументе `номер_столбца` надо задавать значение 2.

Пример

См. пример `demo5.php` в подкаталоге `samples\php` установочного каталога СУБД ЛИНТЕР.

Добавить порцию данных указанного типа к заданному BLOB-столбцу

Назначение

Функция `Linter_Blob_Add_Ex` добавляет порцию данных указанного типа к заданному BLOB-столбцу в текущей строке курсорного запроса.

Синтаксические правила

`int Linter_Blob_Add_Ex` (идентификатор_курсора, порция_данных, размер_порции, номер_столбца, тип_данных);

Порция_данных

Данные для добавления

Размер_порции

Количество байт из параметра «порция_данных», которые будут добавлены к BLOB-данным. Для оптимальной работы рекомендуется, чтобы размер порции был кратен 4048 байт.

Номер_столбца

Параметр задает номер столбца, к которому должна быть добавлена порция данных.

Нумерация столбцов в строке начинается с 1.

Тип_данных

Параметр задает тип добавляемых данных (текст, графика, анимация, музыка и т.п.). Типы BLOB-данных в СУБД ЛИНТЕР не стандартизованы. Каждый пользователь может вводить свою собственную систему типизации.

Функция присваивает BLOB-столбцу указанный тип данных только в случае, если BLOB предварительно был очищен. При последующих вызовах `тип_данных` игнорируется.

Параметр `тип_данных` – целое число.

Пример

См. пример `demo5.php` в подкаталоге `samples\php` установочного каталога СУБД ЛИНТЕР.

Удалить BLOB-данные заданного столбца

Назначение

Функция `Linter_Blob_Purge` удаляет BLOB-данные заданного столбца текущей строки.

Синтаксические правила

`int Linter_Blob_Purge` (идентификатор_курсора, номер_столбца);

Номер_столбца

Параметр задает номер столбца, из которого должны быть удалены BLOB-данные.

Нумерация столбцов в строке начинается с 1.

Пример

См. пример `demo5.php` в подкаталоге `samples\php` установочного каталога СУБД ЛИНТЕР.

Получить порцию BLOB-данных заданного столбца

Назначение

Функция `Linter_Blob_Fetch` извлекает порцию BLOB-данных заданного столбца текущей строки.

Синтаксические правила

`string | int Linter_Blob_Fetch` (идентификатор_курсора, начало_порции, размер_порции, номер_столбца);

Начало_порции

Параметр задает относительный номер байта, с которого начинается порция данных.

Размер_порции

Параметр указывается в байтах.

Номер_столбца

Параметр задает номер столбца, из которого необходимо извлечь BLOB-данные.

Нумерация столбцов в строке начинается с 1.

Возвращаемое значение

Запрошенная порция данных.

Пример

См. пример `demo5.php` в подкаталоге `samples\php` установочного каталога СУБД ЛИНТЕР.

Получить размер BLOB-данных

Назначение

Функция `Linter_Blob_Get_Size` предоставляет размер BLOB-данных текущей строки ответа.

Синтаксические правила

```
int Linter_Blob_Get_Size (идентификатор_курсора [номер_столбца]);
```

Возвращаемое значение

Размер BLOB-данных столбца, указанного в параметре `номер_столбца`. Если `номер_столбца` не задан, по умолчанию принимается 1.

Пример

См. пример `demo5.php` в подкаталоге `samples\php` установочного каталога СУБД ЛИНТЕР.

Добавить порцию данных к BLOB-столбцу

Назначение

Функция `Linter_Blob_Append` добавляет порцию данных к первому BLOB-столбцу в текущей строке ответа.

 Функция устарела. Вместо неё рекомендуется использовать `Linter_Blob_Add`.


Синтаксические правила

```
int Linter_Blob_Append (идентификатор_курсора, порция_данных, размер_порции);
```

Добавить порцию данных указанного типа к BLOB-столбцу

Назначение

Функция `Linter_Blob_Append_Ex` добавляет порцию данных указанного типа к первому BLOB-столбцу в текущей строке ответа.

 Функция устарела. Вместо неё рекомендуется использовать `Linter_Blob_Add_Ex`,

Синтаксические правила

```
int Linter_Blob_Append_Ex (идентификатор_курсора, порция_данных, размер_порции, тип_данных);
```

Удалить BLOB-данные

Назначение

Функция `Linter_Blob_Clear` удаляет BLOB-данные первого BLOB-столбца текущей строки ответа.

 Функция устарела. Вместо неё рекомендуется использовать `Linter_Blob_Purge`.

Синтаксические правила

```
int Linter_Blob_Clear (идентификатор_курсора);
```

Получить порцию BLOB-данных

Назначение

Функция `Linter_Blob_Get_Data` позволяет получить порцию Blob-данных первого BLOB текущей строки ответа.

 Функция устарела. Вместо неё рекомендуется использовать `Linter_Blob_Fetch`

Синтаксические правила

```
string | int Linter_Blob_Get_Data (идентификатор_курсора, начало_порции,
размер_порции);
```

Получить последний код завершения

Назначение

Функция `Linter_Last_Error` возвращает последний код завершения по соединению. Функция может возвращать дополнительный (системный) код завершения.

Синтаксические правила

```
int Linter_Last_Error ([идентификатор_курсора,] тип);
```

Идентификатор_курсора

Идентификатор открытого соединения.

Если соединение не открыто, код завершения нужно получать с помощью запроса `Linter_Last_Error(LINTER_ERROR)`, т.е. без указания соединения.

Тип

Параметр задает тип запрашиваемого кода завершения: `LINTER_ERROR` или `SYSTEM_ERROR`.

Возвращаемое значение

Последний код завершения указанного типа.

Получить описание последнего кода завершения

Назначение

Функция `Linter_Error_Msg` возвращает текстовое описание последнего кода завершения запроса, выданного СУБД ЛИНТЕР.

Функции Linter PHP-интерфейса

Синтаксические правила

string | int `Linter_Error_Msg` (идентификатор_курсора);

Возвращаемое значение

Текстовое описание последнего кода завершения СУБД ЛИНТЕР.



Для получения корректного описания в БД должна существовать таблица `ERRORS`.

Линтер DBX-интерфейс

DBX-интерфейс – независимый от БД интерфейс доступа из PHP-модуля к БД. Доступ к БД выполняется с помощью сравнительно небольшого набора функций (вызовов). Эти функции взаимодействуют с СУБД не напрямую, а через промежуточный модуль, который используется для доступа к конкретной БД (рис . 1) (в частности, для доступа к БД ЛИНТЕР используется модуль DBX-интерфейса СУБД ЛИНТЕР). Таким образом, PHP-модуль, выполняющий доступ к некоторой БД только с помощью набора функций DBX-интерфейса, может быть без изменения использован для доступа к любой другой БД, в систему управления которой входит модуль DBX-интерфейса. Можно сказать, что DBX-интерфейс является аналогом ODBC-интерфейса для PHP-приложений.

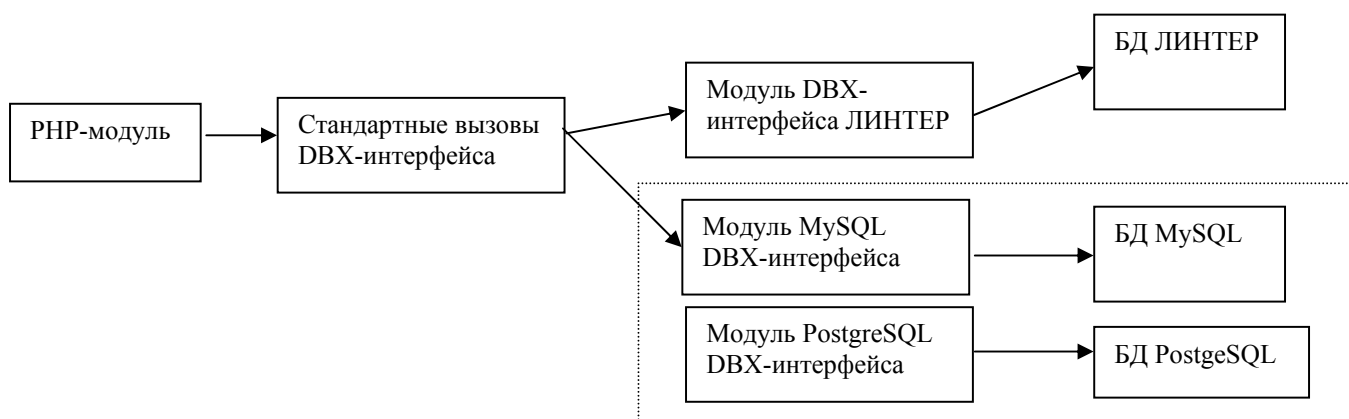


Рис. 1. Схема взаимодействия PHP-модуля с СУБД

Необходимые условия

При работе с СУБД ЛИНТЕР с помощью данного расширения должна быть загружена библиотека `php(x)_linter.dll`.

Использование в среде Unix

Для включения поддержки СУБД ЛИНТЕР в DBX-интерфейс:

1. Получить исходные тексты DBX-интерфейса (например, с сайта).
2. Скопировать файлы `dbx_linter.c` и `dbx_linter.h` (из каталога `<Linter_root>\intl\PHP\dbx` для Windows, `<Linter_root>/PHP/dbx` для UNIX) в каталог с исходными текстами расширения.
3. Изменить файл `dbx.c`, как написано в инструкции `howto_extend_dbx.htm`, которая находится в каталоге с исходными текстами расширения,
4. Собрать расширение `dbx` в соответствии с рекомендациями по сборке расширения.

Функции DBX-интерфейса СУБД ЛИНТЕР

Установить соединение с БД

Назначение

Функция `dbx_connect` устанавливает связь между PHP-модулем и СУБД ЛИНТЕР на узле с заданным именем пользователя и паролем.

Синтаксические правила

object **dbx_connect** (модуль, сервер, "", пользователь, пароль);

Модуль

Имя DBX-модуля для СУБД ЛИНТЕР: `DBX_LINTER`.

Сервер

Имя ЛИНТЕР-сервера, с которым необходимо установить соединение. Символьная строка длиной не более 8 символов.

Пользователь

Имя пользователя БД. Символьная строка длиной не более 66 символов.

Пароль

Пароль пользователя. Символьная строка длиной не более 18 символов.

Возвращаемое значение

Идентификатор объекта соединения при нормальном завершении, `FALSE` – при ошибке.

Примеры

1) соединение с ЛИНТЕР-сервером по умолчанию

```
if (!($link = dbx_connect(DBX_LINTER, "", "", "SYSTEM",  
"MANAGER")))  
    die("Error in dbx_connect");
```

2) соединение с конкретным ЛИНТЕР-сервером

или так (при подключении к удаленному серверу БД с именем "HOST1"):

```
if (!($link = dbx_connect(DBX_LINTER, "DEMO", "", "SYSTEM",  
"MANAGER")))  
    die("Error in dbx_connect");
```

Закреть соединение с БД

Назначение

Функция `dbx_close` закрывает ранее установленное соединение между PHP-модулем и СУБД ЛИНТЕР.

Синтаксические правила

```
bool dbx_close (идентификатор);
```

Идентификатор

Идентификатор ранее установленного соединения с БД.

Возвращаемое значение

TRUE – при нормальном завершении, FALSE – при ошибке.

Пример

```
$link = dbx_connect(DBX_LINTER, "DEMO", "", "SYSTEM", "MANAGER")
        or die ("Ошибка соединения");

print("Успешное соединение с БД");
dbx_close($link);
```

Сравнить значения столбцов

Назначение

Функция `dbx_compare` сравнивает значения указанных столбцов из двух строк таблицы.

Синтаксические правила

```
int dbx_compare (строка1, строка2, столбец [, флаг]);
```

Строка1

Массив значений первой сравниваемой строки таблицы.

Строка2

Массив значений второй сравниваемой строки таблицы.

Столбец

Имя сравниваемого столбца.

Флаг

Задаёт направление и тип сравнения.

Возможные направления:

- `DBX_CMP_ASC` – в порядке возрастания;
- `DBX_CMP_DESC` – в порядке убывания.

Возможные типы сравнения:

- `DBX_CMP_NATIVE` – без конвертации типов;
- `DBX_CMP_TEXT` – сравнивать как строки;

Функции Linter DBX-интерфейса

- `DBX_CMP_NUMBER` - сравнивать численно.

Одно из направлений и одна из констант типа могут комбинироваться битовой операцией OR (`|`).

Если параметр `flag` не задан, по умолчанию используется значение

`DBX_CMP_ASC | DBX_CMP_NATIVE`.

Возвращаемое значение

Результат сравнения:

- 1) 0, если значение строка1 [столбец] равно значению строка2[столбец];
- 2) 1, если значение строка1[столбец] больше значения строка2[столбец] (в случае, если значение параметра флаг равно `DBX_CMP_ASC`);
- 3) -1, если значение строка1[столбец] меньше значения строка2[столбец] в случае, если значение параметра флаг равно `DBX_CMP_ASC`).

Результат сравнения меняется на противоположный при замене значения параметра флаг на `DBX_CMP_DESC`.

Пример

```
function user_auto ($a, $b) {
    $rv = dbx_compare ($a, $b, "model", DBX_CMP_DESC);
    if ( !$rv ) {
        $rv = dbx_compare ($a, $b, "year", DBX_CMP_NUMBER);
    }
    return $rv;
}

$link = dbx_connect (DBX_LINTER, "", "", "SYSTEM", "MANAGER")
or die ("Ошибка соединения");

$result = dbx_query ($link, "SELECT year, make, model FROM auto ORDER BY make");
// данные в $result теперь упорядочены по производителям автомобилей (make)

dbx_sort ($result, "user_auto");
// данные в $result теперь упорядочены по марке автомобиля (model) (по убыванию), затем по году выпуска (year)

dbx_close ($link);
```

Получить сообщение об ошибке

Назначение

Функция `dbx_error` предоставляет сообщение об ошибке последнего вызова функции в DBX-модуле СУБД ЛИНТЕР (не просто в соединении).

Синтаксические правила

string **dbx_error** (идентификатор);

Идентификатор

Идентификатор ранее установленного соединения с БД.

Возвращаемое значение

Строка с сообщением об ошибке от последнего вызова функции LINTER DBX-модулем. Если есть несколько соединений с одним и тем же модулем, выдаётся просто последняя ошибка. Если имеются соединения в различных модулях, возвращается самая последняя ошибка для модуля, соответствующего указанному параметру соединения.

Пример

```
$link = dbx_connect(DBX_LINTER, "DEMO", "", "SYSTEM", "MANAGER")
        or die ("Ошибка соединения");

$result = dbx_query($link, "selet count(*)from auto");
if ( $result == 0 ) {
    echo dbx_error ($link);
}
dbx_close ($link);
```

Выполнить запрос

Назначение

Функция `dbx_query` передаёт SQL-запрос на выполнение и получает все результаты его обработки (если имеются).

Синтаксические правила

object **dbx_query** (идентификатор, запрос [,флаги]);

Идентификатор

Идентификатор ранее установленного соединения с БД.

Запрос

Текст SQL-запроса.

Флаги

Флаги управления возвращаемой информацией. Может быть любая комбинация следующих констант с битовой операцией OR (!):

DBX_RESULT_INDEX

Всегда установлен, то есть возвращённый объект имеет свойство `data`, которое является двухмерным массивом, индексированным цифрами. Например, в выражении `data[2][3]` 2 – это количество строк (или записей) ответа, а 3 – это количество столбцов (или полей) в каждой строке ответа. Индексация начинается с 0;

DBX_RESULT_ASSOC

Возвращённый объект содержит информацию, относящуюся также и к `DBX_RESULT_INFO`, даже если она и не специфицирована;

DBX_RESULT_INFO

Предоставлять информацию о столбцах (имя и тип данных);

DBX_RESULT_ASSOC

Устанавливает соответствие между значением поля и его именем, используемом как ключ, и делает это соответствие доступным свойству `data` возвращаемого объекта.

Ассоциированные результаты являются на самом деле ссылками на численно индексированные данные, так что модификация `data[0][0]` имеет такой же эффект, что и модификация `data[0]['имя первого столбца']`.

DBX_RESULT_INDEX используется всегда, независимо от фактического значения параметра флаги. Это означает, что эффективными являются только следующие комбинации:

- DBX_RESULT_INDEX;
- DBX_RESULT_INDEX | DBX_RESULT_INFO;
- DBX_RESULT_INDEX | DBX_RESULT_INFO | DBX_RESULT_ASSOC - по умолчанию, если параметр флаги не задан.

Возвращаемое значение

Результат выполнения SQL-запроса:

- 1) объект типа `object` (только в том случае, если SQL-запрос порождает строки ответа);
- 2) 1 - успешная обработка SQL-запроса, не порождающего строк ответа (например, `CREATE TABLE ...`);
- 3) 0 - неудачная обработка любого SQL-запроса.

Возвращаемый объект `object` имеет 4 или 5 свойств, в зависимости от значения параметра флаги:

`handle`

Правильный дескриптор присоединённой БД. Может использоваться в функциях, где фигурирует в качестве параметра функции, например:

```
$result = dbx_query ($link, "SELECT make FROM auto");  
//вызываем функцию php интерфейса СУБД ЛИНТЕР для получения свойств первого  
столбца запроса  
$columns = Linter_Get_Col_Prop ($result->handler, 1);
```

`cols` и `rows`

Содержат количество столбцов (полей) и строк (записей), соответственно, например:

```
$result = dbx_query ($link, 'SELECT make FROM auto');  
echo $result->rows; // количество строк  
echo $result->cols; // количество столбцов
```

`info` (по желанию)

Двумерный массив, состоящий из именованных строк (`name` и `type`), для получения информации о столбце. Возвращается только тогда, когда установлен флаг DBX_RESULT_INFO или DBX_RESULT_ASSOC, например:

```
$result = dbx_query ($link, 'SELECT make, model, year FROM auto',  
DBX_RESULT_INDEX | DBX_RESULT_INFO);  
  
for ($i = 0; $i < $result->cols; $i++) {  
echo $result->info['name'][$i] . "\n";  
echo $result->info['type'][$i] . "\n";  
}
```

data

Содержит результаты выполнения запроса, возможно, ассоциированные также с именами столбцов, в зависимости от установленного значения параметра флага. Если установлен флаг `DBX_RESULT_ASSOC`, то к столбцам можно обращаться по их именам `$result->data[2]["имя столбца"]`, например (вывод результатов в HTML-формате):

```
$result = dbx_query ($link, 'SELECT make, model, year FROM auto');
echo "<table>\n";
foreach ( $result->data as $row ) {
echo "<tr>\n";
foreach ( $row as $field ) {
echo "<td>$field</td>";
}
echo "</tr>\n";
}
echo "</table>\n";
```

Отсортировать выборку данных

Назначение

Функция `dbx_sort` выполняет сортировку результирующих данных поискового SQL-запроса с помощью пользовательской функции сравнения.

Синтаксические правила

`bool dbx_sort(выборка, функция);`

Выборка

Результирующая выборка поискового SQL-запроса.

Функция

Имя пользовательской функции сравнения данных.

Возвращаемое значение

Результат сортировки:

- 1) `TRUE` – нормальное завершение;
- 2) `FALSE` – ошибка.

Если возможно, сортировку лучше выполнять не данной функцией, а с помощью конструкции `ORDER BY` SQL-оператора.

Линтер PEAR::db-интерфейс

PEAR ([PHP Extension and Application Repository](#)) – то библиотека классов языка программирования [PHP](#) с открытым исходным кодом. Она входит в стандартную поставку PHP. Чтобы класс вошёл в PEAR, он должен соответствовать очень жёстким правилам. Например, без особой необходимости нельзя создавать класс с задачей, аналогичной уже существующему классу.

В рамках PEAR был создан специальный [стиль оформления PHP-кода](#), которого должны придерживаться все классы в библиотеке. Этот стиль стал наиболее распространённым стандартом стиля PHP-кода в Интернете.

Библиотека PEAR содержит набор модулей, похожих на Perl CPAN (или TeX CTAN), для PHP. Код в PEAR разделен на так называемые «пакеты» (packages). Каждый из пакетов – набор классов и утилит, написанных на PHP и представляющих собой решение какой-нибудь распространённой проблемы. При этом все пакеты тесно связаны между собой, например, для доступа к БД данных используется пакет PEAR::Db.

Использование в среде Win32

Для включения поддержки СУБД ЛИНТЕР в расширение PEAR::Db в среде Win32 необходимо скопировать файл `linter.php` из подкаталога `\intl\PHP\pear` установочного каталога СУБД ЛИНТЕР в подкаталог `<php_root>/PEAR/DB`. При этом должен быть доступен модуль PHP-интерфейса СУБД ЛИНТЕР.

Использование в среде UNIX

Для включения поддержки СУБД ЛИНТЕР в расширение PEAR::Db в среде UNIX необходимо скопировать файл `linter.php` из подкаталога `\php\pear` установочного каталога СУБД ЛИНТЕР в подкаталог `<php_root>/PEAR/DB`. При этом должен быть доступен модуль PHP-интерфейса СУБД ЛИНТЕР.

DB-основной класс

Установить соединение с базой данных

Назначение

Функция `DB::connect` устанавливает соединение между PHP-модулем и СУБД ЛИНТЕР на узле с заданным именем пользователя и паролем.

Синтаксические правила

```
mixed connect (string $dsn [, array $options]);
```

\$dsn

Строка, содержащая параметры соединения.

Формат строки:

```
linter://<пользователь>:<пароль>@[сервер].
```

Пользователь

Имя пользователя БД. Символьная строка длиной не более 66 символов.

Пароль

Пароль пользователя. Символьная строка длиной не более 18 символов.

Сервер

Имя ЛИНТЕР-сервера, с которым необходимо установить соединение. Символьная строка длиной не более 8 символов.

\$options

Опции соединения (в СУБД ЛИНТЕР не используются).

Возвращаемое значение

Результат соединения с БД:

- mixed – созданный DB-объект соединения (нормальное завершение).
- DB_Error - объект – при ошибке соединения.

Примеры

```
$dbh =&DB::connect("linter://SYSTEM:MANAGER@", $options);
$dbh =&DB::connect("linter://SYSTEM:MANAGER@HOST1", $options);
<?
require_once("DB.php");
$dsn = 'linter://SYSTEM:MANAGER@';
$db = DB::Connect($dsn);
if (DB::isError($db))
{
    echo $dbh->getUserInfo(); // содержит код ошибки ЛИНТЕРа и ее
    описание
    echo $dbh->getMessage(); // стандартное описание PEAR ошибки
    die;
}
...
?>
```

Проверить на ошибку

Назначение

Функция DB::isError проверяет код завершения на принадлежность его к группе ошибок.

Синтаксические правила

```
boolean isError (mixed $value);
```

\$value

Проверяемый код завершения.

Возвращаемое значение

Результат проверки:

- true – код завершения соответствует ошибке (DB_Error объект);

- `false` – в противном случае.

Пример

Получение сообщений об ошибках из `Pear Error` объектов.

```
<?php
...
$res = $db->query('selct * from auto');
if (DB::isError($res)) {
    // Получение краткого сообщения об ошибке
    echo $res->getMessage();
}
?>
```

DB_Conmon-интерфейс для доступа к БД

Получить количество изменённых строк

Назначение

Функция `DB_Conmon::affectedRows` предоставляет информацию о количестве реально изменённых строк в процессе выполнения запроса.

Синтаксические правила

`mixed affectedRows ()`

Возвращаемое значение

Результаты обработки запроса:

- `mixed` – количество изменённых строк (добавленных или модифицированных) в случае успешного выполнения запроса;
- `DB_Error` – в случае ошибки при выполнении запроса.

Пример

```
// Сколько строк реально удалено
$sql = "delete from clients";
$db->query("delete from auto where model='BLACK' and year=70");
echo 'Я удалил ' . $db->affectedRows() . 'строк';
```

Создать последовательность

Назначение

Функция `DB::createSequence` создает новую последовательность.

Синтаксические правила

`mixed createSequence(string $seq_name)`

`$seq_name`

Имя последовательности.

Возвращаемое значение

Результаты обработки запроса:

- `integer` – `DB_OK` в случае успешного выполнения запроса;
- `DB_Error` – в случае ошибки при выполнении запроса.

Закреть соединение с базой данных

Назначение

Функция `DB_Common::disconnect` закрывает ранее установленное соединение с БД.

Синтаксические правила

`boolean disconnect ()`

Возвращаемое значение

- `true` – в случае успешного выполнения;
- `false` – в случае ошибки.

Удалить последовательность

Назначение

Функция `DB_Common::dropSequence` удаляет последовательность.

Синтаксические правила

`integer dropSequence (string $seqName)`

`$seq_name`

Имя последовательности.

Возвращаемое значение

Результаты обработки запроса:

- `integer` – `DB_OK` в случае успешного выполнения запроса;
- `DB_Error` – в случае ошибки при выполнении запроса.

Выполнить подготовленный запрос

Назначение

Функция `DB_Common::execute()` выполняет подготовленный SQL-запрос.

Синтаксические правила

`mixed execute(resource $stmt, array $data)`

`$stmt`

Текст подготовленного SQL-запроса.

`$data`

Числовой массив, содержащий данные для привязки к запросу.

Описание

Функция выполняет привязку параметров к подготовленному запросу и выполняет его.

Возвращаемое значение

Результаты обработки запроса:

- `DB_Result` - результирующая выборка в случае нормального выполнения запроса;
- `DB_Error` - в случае ошибки.

Многократное выполнение запроса

Назначение

Функция `DB_Common::executeMultiple` выполняет повторение подготовленного запроса.

Синтаксические правила

mixed `executeMultiple` (string `$stmt`, array `$data`)

`$stmt`

Идентификатор запроса функции `prepare()`.

`$data`

Нумерованный массив, содержащий данные для внесения в запрос. Нумерация начинается с 0. Для каждой строки этого массива вызывается функция `execute()`.

Описание

Количество повторов определяется размерностью массива, содержащего входные данные для каждого повторяемого запроса. Для каждой строки этого массива вызывается функция `execute()`. Если при выполнении какого-то запроса произошла ошибка, оставшиеся запросы будут проигнорированы.

Возвращаемое значение

Результаты обработки запроса:

- `resource` - объект класса `DB_Result` в случае успешного выполнения запроса со всеми строками входного массива;
- `DB_Error` - в случае ошибки при выполнении запроса.

Получить всю результирующую выборку

Назначение

Функция `DB_Common::getAll` предоставляет все строки результирующей выборки.

Синтаксические правила

mixed `&getAll` (string `$query` [, array `$params` [, integer `$fetchmode`]])

`$query`

Текст SQL-запроса или выражение для его подготовки.

\$params

Массив значений параметров для привязки их к подготавливаемому запросу (если запрос – подготавливаемый).

\$fetchmode

Режим выборки. Если не задан, по умолчанию используется DB_FETCHMODE_DEFAULT.

Возвращаемое значение

Результаты обработки запроса:

- mixed – двумерный массив ответов в случае успешного выполнения запроса;
- DB_Error – в случае ошибки при выполнении запроса.

Получить заданный столбец результирующей выборки

Назначение

Функция DB_Common::getCol предоставляет значения заданного столбца результирующей выборки.

Синтаксические правила

```
mixed getCol(string $query [, mixed $col [, array $params]])
```

\$query

Текст SQL-запроса или выражение для его подготовки.

\$col

Если тип аргумента integer – номер столбца, начинающийся с 0; если string – имя столбца. Значение по умолчанию 0.

\$params

Массив значений параметров для привязки их к подготавливаемому запросу (если запрос – подготавливаемый).

Возвращаемое значение

Результаты обработки запроса:

- mixed – массив значений заданного столбца, индексируемый с 0, в случае успешного выполнения запроса;
- DB_Error – в случае ошибки при выполнении запроса.

Пример

```
// $all_client_names = array('make', 'model');  
$all_client_names = $db->getCol('select make, model from auto');  
?>
```

Получить информацию о БД

Назначение

Функция `DB_Common::getListOf` предоставляет внутреннюю информацию о БД.

Синтаксические правила

```
mixed getListOf(string $type)
```

\$type

Тип запрашиваемой о БД информации: “tables”, “views”, “users”;

Возвращаемое значение

Результаты обработки запроса:

- `mixed` – запрашиваемые данные в случае успешного выполнения запроса;
- `DB_Error` – в случае ошибки при выполнении запроса.

Выбрать значение поля

Назначение

Функция `DB_Common::getOne` предоставляет значение поля из первого столбца первой строки ответа результирующей выборки.

Синтаксические правила

```
mixed getOne(string $query [, array $params])
```

\$query

Текст SQL-запроса или выражение для его подготовки.

\$ params

Массив значений параметров для привязки их к подготавливаемому запросу (если запрос – подготавливаемый).

Возвращаемое значение

Результаты обработки запроса:

- `mixed` – значение поля в случае успешного выполнения запроса;
- `DB_Error` – в случае ошибки при выполнении запроса.

Пример

```
$numrows = $db->getOne('select count(*) from auto');
```

Получить первую строку результирующей выборки

Назначение

Функция `DB_common::getRow` предоставляет первую строку ответа результирующей выборки.

Синтаксические правила

mixed **getRow** (string \$query [, array \$params [, integer \$fetchmode]])

\$query

Текст SQL-запроса или выражение для его подготовки.

\$params

Массив значений параметров для привязки их к подготавливаемому запросу (если запрос – подготавливаемый).

\$fetchmode

Режим выборки. Если не задан, по умолчанию используется DB_FETCHMODE_DEFAULT.

Возвращаемое значение

Результаты обработки запроса:

- mixed – первая строка ответа в виде индексруемого с 0 массива в случае успешного выполнения запроса;
- DB_Error – в случае ошибки при выполнении запроса.

Ограничение размера результирующей выборки**Назначение**

Функция DB_Common::limitQuery ограничивает количество строк ответа результирующей выборки.

Синтаксические правила

mixed **limitQuery**(string \$query, integer \$from, integer \$count)

\$query

Текст SQL-запроса.

\$from

Номер строки, с которой надо начинать выборку.

\$count

Количество выбираемых строк.

Возвращаемое значение

Результаты обработки запроса:

- resource – объект класса DB_Result в случае успешного выполнения запроса;
- DB_Error – в случае ошибки при выполнении запроса.

Получить следующий номер последовательности

Назначение

Функция `DB_Common::nextId` предоставляет следующий (свободный) номер в заданной последовательности.

Синтаксические правила

```
mixed nextId(string $seq_name, boolean $on_demand)
```

`$seq_name`

Имя последовательности.

`$on_demand`

Порядок предоставления номера:

- `true` – создать новую последовательность (с заданным в аргументе `$seq_name` именем), затем выбрать из нее номер;
- `false` – выбрать номер из существующей последовательности.

Значение по умолчанию – `true`.

Возвращаемое значение

Результаты обработки запроса:

- `mixed` – номер последовательности в случае успешного выполнения запроса;
- `DB_Error` – в случае ошибки при выполнении запроса.

Пример

```
$id = $db->nextID('mySequence');
```

Подготовить запрос к выполнению

Назначение

Функция `DB_Common::prepare()` подготавливает SQL-запрос к выполнению.

Синтаксические правила

```
resource prepare(string $query)
```

`$query`

Текст SQL-запроса, который должен быть подготовлен.

Описание

Функция подготавливает SQL-запрос для последующего его выполнения с помощью функции `execute()`.

Возвращаемое значение

Результаты обработки запроса:

- `resource` – идентификатор подготовленного запроса в случае нормального выполнения запроса;

- `DB_Error` – в случае ошибки.

Пример

```
$alldata = array(
    array(1, 'one'),
    array(2, 'two'),
    array(3, 'three')
);

$sth = $dbh->query("CREATE OR REPLACE TABLE TST (I INT, C CHAR(5))");
$sth = $dbh->prepare("INSERT INTO TST VALUES (?,?)");
foreach ($alldata as $row) {
    $dbh->execute($sth, $row);
}
```

Передать запрос на выполнение

Назначение

Функция `DB_Common::query()` посылает SQL-запрос ядру СУБД ЛИНТЕР.

Синтаксические правила

`mixed query(string $query [, array $params])`

`$query`

Текст SQL-запроса.

`$params`

Значения, которые должны быть привязаны к подготавливаемому запросу (если необходимо).

Описание

Функция подготавливает (при необходимости) SQL-запрос и передает его СУБД для выполнения.

Возвращаемое значение

Результаты обработки запроса:

- `DB_Result` – результат выполнения поискового запроса в случае нормального выполнения запроса;
- `DB_OK` – результат выполнения непоискового запроса в случае нормального выполнения запроса;
- `DB_Error` – в случае ошибки.

Установить граничные кавычки

Назначение

Функция `DB::quote` устанавливает кавычки в начале и в конце символьной строки.

Синтаксические правила

`string quote(string $string)`

`$string`

Строка, которая должна быть ограничена кавычками.

Возвращаемое значение

`string` – исходная строка с граничными кавычками.

Установить формат по умолчанию для выборки данных

Назначение

Функция `DB_Common::setFetchMode` устанавливает формат, который будет использоваться по умолчанию для представления данных результирующей выборки.

Синтаксические правила

`mixed setFetchMode(integer $fetchmode [, string $object_class])`

`$fetchmode`

Формат по умолчанию:

- `DB_FETCHMODE_ORDERED;`
- `DB_FETCHMODE_OBJECT;`
- `DB_FETCHMODE_ASSOC,`

Перечисленные форматы можно задавать совместно с `DB_FETCHMODE_FLIPPED` при помощи логической операции `OR` (см. «FETCH-секцию» для я дополнительной информации).

`$object_class`

Класс, объект которого будет возвращен методом `fetch`, если форматом по умолчанию выбран `DB_FETCHMODE_OBJECT`. Если никакого класса не задано, по умолчанию будет выполняться приведение строки ответа к объекту. Это даёт возможность использовать и расширять класс `'DB_Row'`.

Возвращаемое значение

Результат установки формата:

- заданный формат установлен – не возвращается ничего;
- `PEAR_ERROR` – если `$fetchmode` содержит неизвестное значение.

DB_Result-результатирующая выборка

Поместить заданную строку результирующей выборки в переменную

Назначение

Функция `DB_Result::fetchInto()` размещает заданную строку результирующей выборки в указанной переменной.

Синтаксические правила

mixed **fetchInto**(mixed \$arr [, integer \$fetchmode [, integer \$rownum]])

\$arr

Ссылка на переменную, в которую должна быть размещена строка выборки.

\$ fetchmode

Формат выбираемых строк. По умолчанию DB_FETCHMODE_DEFAULT.

\$rownum

Номер строки для выборки. По умолчанию NULL.

Возвращаемое значение

Результаты обработки запроса:

- DB_OK - в случае успешного выполнения запроса;
- NULL - если заданная строка не найдена;
- DB_Error - в случае ошибки при выполнении запроса.

Пример

```
<?php
...
while ($result->fetchInto($row, $optional_fetchmode)
{
    $id = $row[0];
}
?>
```

Получить заданную строку результирующей выборки

Назначение

Функция DB_Result::fetchRow() предоставляет заданную строку результирующей выборки.

Синтаксические правила

mixed **fetchRow** ([integer \$fetchmode [, integer \$rownum]])

\$ fetchmode

Формат выбираемых строк. По умолчанию DB_FETCHMODE_DEFAULT.

\$rownum

Номер строки для выборки. По умолчанию NULL.

Описание

DB_Result содержит результат выполнения запроса к БД. Ссылка на экземпляр объекта DB_Result возвращается функциями [query\(\)](#) или [execute\(\)](#).

Возвращаемое значение

Результаты обработки запроса:

- array – массив значений строки в случае успешного выполнения запроса;
- NULL – если заданная строка не найдена;
- DB_Error – в случае ошибки при выполнении запроса.

Примеры

1)

```
while ($row = $result->fetchRow()) {
    // По умолчанию поведение метода fetchRow() – это возвращение
    // ассоциативного массива, такого как:
    // $row = array (
    //0 => <first column data>,
    //1 => <second column data>
    //);
    echo $row[0]."<br>\n";
}
?>
```

2)

```
$row = $db->getRow("select year, model from auto where personid=100",
DB_FETCHMODE_OBJECT);
print_r($row);
// Пусть типом класса по умолчанию объекта «строка» будет
db_row Object
(
    [year] => 70
    [model] => BMV
    [color] => BLACK
)
```

Доступ к данным объекта:

```
$year = $row->year;
$model = $row->model;
$color = $row->color;
```

Возможны два метода доступа.

```
// первый метод:
while ($row = $result->fetchRow(DB_FETCHMODE_ASSOC))
{
    // $row = array(
    // 'year' => <данные столбца year>,
    // 'model' => <данные столбца model>

    // 'color' => <данные столбца color>

    //);
    $id = row['id'];
}

// Второй метод (используется по умолчанию)
$db->setfetchmode(DB_FETCHMODE_ASSOC);
// устанавливаем метод по умолчанию
$result->query($sql);
while ($row = $result->fetchRow())
```

```
{  
    $year = row['year'];  
    $model = row['model'];  
    $color = row['color'];  
}
```

Освободить ресурсы

Назначение

Функция `DB_Result::free()` освобождает ресурсы, выделенные для хранения строк результирующей выборки.

Синтаксические правила

```
mixed free()
```

Возвращаемое значение

Результаты обработки запроса:

- `true` – ресурсы освобождены;
- `DB_Error` – ресурсы не освобождены.

Получить следующую результирующую выборку

Назначение

Функция `DB_Result::nextResult()` предоставляет следующую результирующую выборку в случае выполнения пакета запросов.

Синтаксические правила

```
boolean nextResult ()
```

Возвращаемое значение

Результаты обработки запроса:

- `true` – результирующая выборка содержится в пакете запросов;
- `false` – результирующие выборки исчерпаны.

Получить количество столбцов результирующей выборки

Назначение

Функция `DB_Result::numCols()` предоставляет информацию о количестве столбцов результирующей выборки.

Синтаксические правила

```
integer numCols ()
```

Возвращаемое значение

Результаты обработки запроса:

- `integer` – количество столбцов в случае успешного выполнения функции;
- `DB_Error` – в случае ошибки при выполнении функции.

Получить количество строк результирующей выборки

Назначение

Функция `DB_Result::numRows()` предоставляет информацию о количестве строк результирующей выборки.

Синтаксические правила

```
integer numRows ()
```

Возвращаемое значение

Результаты обработки запроса:

- `integer` – количество строк результирующей выборки в случае успешного выполнения функции;
- `DB_Error` – в случае ошибки при выполнении функции.

Пример

```
$db = DB::connect($dsn);
$sql = 'select * from auto';
$res = $db->query($sql);

// Проверить код завершения
// ...

// Кол-во строк:
echo $res->numRows();
// Кол-во столбцов:
echo $res->numCols();
// Информация о запросе:
print_r ($res->tableInfo());
```

Получить описание результирующей выборки

Назначение

Функция `DB::tableInfo` предоставляет описание результирующей выборки.

Синтаксические правила

```
mixed tableInfo (DB_Result $result [, mode $mode])
```

\$result

Результирующая выборка.

\$mode

Формат предоставляемой информации.

- 1) По умолчанию:
 - [0]["table"] – имя таблицы;
 - [0]["name"] – имя поля;
 - [0]["type"] – тип данных поля;
 - [0]["len"] – длина поля;
 - [0]["flags"] – флаги поля;
- 2) DB_TABLEINFO_ORDER
 - ["num_fields"] – количество записей в данной структуре;
 - [0]["table"] – имя таблицы;
 - [0]["name"] – имя поля;
 - [0]["type"] – тип данных поля;
 - [0]["len"] – длина поля;
 - [0]["flags"] – флаги поля;
 - ["order"]["имя поля"] номер элемента "имя поля" в данном массиве. Используется в случае, если предполагается доступ к полям по их именам, а не порядковым номерам. Проверка: `if (isset($result['meta']['myfield'])) { ...`
- 3) DB_TABLEINFO_ORDERTABLE
 - Возвращает массив той же структуры, что и при DB_TABLEINFO_ORDER, но с дополнением: ["ordertable"][table name][field name] номер поля с именем "field name". Это позволяет обрабатывать поля из разных таблиц с одинаковыми именами.

Возвращаемое значение

Результаты обработки запроса:

- mixed – массив описаний в соответствии с заданным форматом в случае успешного выполнения запроса;
- DB_Error – в случае ошибки при выполнении запроса.

Lintер PDO-интерфейс

PDO-интерфейс (PHP Data Objects) представляет собой расширение для языка [PHP](#), позволяющее разработчику иметь простой и удобный интерфейс для доступа к БД данных из PHP-скриптов. PDO-интерфейс реализован в виде 3-х классов (PDO, PDOStatement и PDOException), которые обеспечивают абстрактный (т. е. независимый от конкретной СУБД) доступ к БД. Работа этих классов с разными СУБД обеспечивается за счет подключения pdo-драйвера конкретных СУБД, которые интерпретируют команды PDO в команды той или иной СУБД. В целом, принцип работы PDO-интерфейса схож с принципом работы DBX- и ODBC-интерфейсов.

Использование в среде UNIX

В составе дистрибутива СУБД ЛИНТЕР поставляются готовые к использованию загружаемые библиотеки PDO-интерфейса, а также исходные тексты библиотек, необходимых для сборки программных интерфейсов СУБД ЛИНТЕР.

Для настройки Linter PDO-интерфейса необходимо настроить PHP-интерфейс (см. стр. 3 данной документации) и подключить модули `php_pdo.so` и `php_pdo_linter.so` одним из следующих способов:

- PHP-командами `d1("php_pdo.so")` и `d1("php_pdo_linter.so")` для динамической загрузки Linter PDO-интерфейса в ходе выполнения PHP-программы;
- внесением строк `"extension=php_pdo.so"` и `"extension=php_pdo_linter.so"` в файл `php.ini` для загрузки Linter PDO-интерфейса перед запуском любых PHP-программ.

Использование в среде Win32

Для настройки PDO-интерфейса СУБД ЛИНТЕР в среде Win32:

1. поместить интерфейсные библиотеки (`php_pdo.dll` и `php_pdo_linter.dll`) в каталог, в котором PHP ищет загружаемые ресурсы.

Для того чтобы найти место, куда поместить PDO-интерфейс, необходимо:

- открыть конфигурационный файл `<WINSYS32DIR>/php.ini` (этот файл может располагаться и в другом каталоге. Определить точный путь к файлу можно так, как указано в разделе конфигурирования PHP-интерфейса для UNIX);
- найти в нем строку `extension_dir=...`

Если в ней уже задан какой-либо каталог – скопировать `php_pdo.dll` и `php_pdo_linter.dll` в этот каталог. Если нет – убрать комментарий в строке и прописать в ней путь к каталогу, где находятся данные библиотеки.

2. если необходимо, чтобы интерфейс всегда автоматически подгружалась при исполнении любой PHP-программы, в разделе `Extensions` добавить строки:
`extension=php_pdo.dll`
`extension=php_pdo_linter.dll`
3. перезапустить Web-сервер (если PHP используется для WEB-приложения).

PDO-класс соединений

Объекты данного класса представляют собой соединения с БД. Инициализация нового объекта эквивалентна созданию соединения.

Соединение с БД

Назначение

Создание нового объекта класса PDO.

Синтаксические правила

```
object new PDO(string $dsn, $user, $password [,array $options]);
```

\$dsn

Строка, содержащая параметры соединения.

Формат строки:

```
linter:node=<сервер>;dbname=<название БД>
```

<сервер>::=имя ЛИНТЕР-сервера, с которым необходимо установить соединение. Символьная строка длиной не более 8 символов.

<название БД>::=название БД, к которой необходимо получить доступ. Символьная строка не более 18 символов.

\$user

Строка, содержащая имя пользователя БД.

\$password

Строка, содержащая пароль пользователя БД.

\$options

Опции соединения – массив пар <атрибут значение>. Формат атрибутов и их значений аналогичен формату в функции PDO::setAttribute().

Описание

Функция создает новый объект класса PDO.

Возвращаемое значение

Результат соединения с БД:

- object – созданный PDO-объект соединения (нормальное завершение);
- PDOException-объект – при ошибке соединения.

Пример

```
<?php
try
{
$dbh = new PDO("linter:node=;dbname=DEMO", "SYSTEM", "MANAGER");
/* Тут работа с БД
...
*/
```

```
        $dbh = null;
    }
    catch (PDOException $e)
    {
        print "Error!: " . $e->getMessage() . "<br/>";
        die();
    }
}
```

Закрывать соединение с БД

Назначение

Закрывает соединение с БД.

Синтаксические правила

```
$dbh = null;
```

\$dbh

Ранее инициализированный PDO-объект.

Описание

Функция уничтожает ранее созданный PDO-объект.

Возвращаемые значения

Нет.

Начать транзакцию

Назначение

Начать транзакцию в БД.

Синтаксические правила

```
bool PDO::beginTransaction();
```

Описание

Функция завершает текущую транзакцию БД (если она есть) и начинает новую.

Возвращаемое значение

Результат инициирования транзакции:

- true – начата новая транзакция;
- false – неуспешное инициирование новой транзакции.

Подтвердить транзакцию

Назначение

Подтверждение и завершение текущей транзакции.

Синтаксические правила

```
bool PDO::commit();
```

Описание

Функция сохраняет (фиксирует) в БД внесенные текущей транзакцией изменения и закрывает транзакцию.

Возвращаемое значение

Результат завершения транзакции:

- true – данные сохранены в БД;
- false – неуспешное завершение транзакции; данные в БД не сохранены.

Отменить транзакцию

Назначение

Отмена текущей транзакции.

Синтаксические правила

```
bool PDO::rollBack();
```

Описание

Функция отменяет в БД внесенные текущей транзакцией изменения.

Возвращаемое значение

Результат отмены транзакции:

- true – изменения в БД не сохранены;
- false – неуспешная отмена транзакции, данные, возможно, остались в системном журнале БД.

Пример

```
<?php

function countRows($action) {
    global $dbh;
    $select = $dbh->prepare('SELECT COUNT(*) FROM test');
    $select->execute();
    $res = $select->fetchColumn();
    return "Counted $res rows after $action.\n";
}

$dbh = new PDO("linter:node=;dbname=DEMO", "SYSTEM", "MANAGER");

$dbh->exec('CREATE TABLE test(id INT NOT NULL PRIMARY KEY, val
VARCHAR(10)) ');
$dbh->exec("INSERT INTO test VALUES(1, 'A')");
$dbh->exec("INSERT INTO test VALUES(2, 'B')");
$dbh->exec("INSERT INTO test VALUES(3, 'C')");
```

```
$delete = $dbh->prepare('DELETE FROM test');

echo countRows('insert');

$dbh->beginTransaction();
$delete->execute();
echo countRows('delete');
$dbh->rollBack();

echo countRows('rollback');

$dbh->beginTransaction();
$delete->execute();
echo countRows('delete');
$dbh->commit();

echo countRows('commit');

$dbh = null;
?>
```

Установить атрибут соединения с БД

Назначение

Установка (изменение) атрибутов соединения с БД.

Синтаксические правила

```
bool PDO::setAttribute (int attribute, mixed value);
```

attribute

Идентификатор атрибута.

value

Новое значение устанавливаемого атрибута.

Атрибуты и их допустимые значения:

- 1) PDO::ATTR_CASE – управление преобразованием имен столбцов.
Допустимые значения:
 - PDO::CASE_LOWER – преобразовывать к нижнему регистру;
 - PDO::CASE_NATURAL – оставлять имена столбцов как есть;
 - PDO::CASE_UPPER – преобразовывать к верхнему регистру.
- 2) PDO::ATTR_ERRMODE – управление режимом вывода сообщений.
Допустимые значения:
 - PDO::ERRMODE_SILENT – выводить сообщения только об ошибках;
 - PDO::ERRMODE_WARNING – выводить предупреждения;
 - PDO::ERRMODE_EXCEPTION – выводить сообщения об исключениях.

- 3) PDO::ATTR_ORACLE_NULLS – управление преобразованием NULL-значений и пустых строк.
Допустимые значения:
 - PDO::NULL_NATURAL – оставлять значение как есть;
 - PDO::NULL_EMPTY_STRING – преобразовывать пустую строку в NULL-значение;
 - PDO::NULL_TO_STRING – преобразовывать NULL-значение в пустую строку.
- 4) PDO::ATTR_STRINGIFY_FETCHES – управление преобразованием числовых значений.
Допустимые значения:
 - true – преобразовывать числовые значения в строковый вид;
 - false – оставлять числовые значения как есть.
- 5) PDO::ATTR_STATEMENT_CLASS – устанавливает пользовательский класс для выражений, наследуемый от PDOStatement.
Допустимые значения:
 - array(string classname, array(mixed ctor_args)) – название класса и массив параметров конструктора.
- 6) PDO::ATTR_AUTOCOMMIT – управление autocommit-режимом обработки транзакций.
Допустимые значения:
 - 0 – включить autocommit-режим;
 - 1 – выключить autocommit-режим.
- 7) PDO::ATTR_CURSOR_NAME – устанавливает имя курсора.
Допустимые значения:
 - string cursorName – имя курсора.
- 8) PDO::ATTR_FETCH_TABLE_NAMES – управление добавлением имени таблицы перед именем столбца.
Допустимые значения:
 - true – предварять (через точку) имя столбца именем таблицы;
 - false – не добавлять имя таблицы к имени столбцу.
- 9) PDO::ATTR_FETCH_CATALOG_NAMES – управление добавлением имени БД перед именем столбца (таблицы).
Допустимые значения атрибута:
 - true – предварять (через точку) имя столбца(таблицы) именем БД;
 - false – не добавлять имя БД к имени столбцу (таблицы).
- 10) PDO::ATTR_DT_FORMAT – устанавливает формат значений типа «дата-время».
Допустимые значения:
 - string newDtFormat – шаблон значений типа «дата-время». Формат шаблона аналогичен формату значений типа «дата-время» СУБД ЛИНТЕР (см. документ «СУБД ЛИНТЕР. Справочник по SQL»).

11) PDO::ATTR_FETCH_BLOB_AS_DATA – управление представлением извлекаемых BLOB-данных.

Допустимые значения:

- true – представлять BLOB-данные в бинарном виде;
- false – представлять BLOB-данные в текстовом виде.

Описание

Функция устанавливает заданные атрибуты соединения.

Возвращаемое значение

Результат установки атрибутов:

- true – атрибуты соединения установлены;
- false – неуспешная попытка установки атрибутов.

Получить значение атрибута соединения с БД

Назначение

Получение значения атрибута соединения с БД.

Синтаксические правила

```
mixed PDO::getAttribute (int attribute);
```

attribute

Идентификатор запрашиваемого атрибута.

Кроме атрибутов, перечисленных в пункте «Установить атрибут соединения с БД», дополнительно может быть запрошена информация о следующих атрибутах:

- 1) PDO::ATTR_DRIVER_NAME – название драйвера, используемого для доступа из клиентских PHP-приложений к СУБД ЛИНТЕР (возвращается значение linter).
- 2) PDO::ATTR_CURSOR – тип курсора.
Возможные значения:
 - PDO::CURSOR_SCROLL – скролируемый курсор (перемещение по выборке вперед/назад).
- 3) PDO::ATTR_SERVER_VERSION – версия сервера БД.
Возможные значения:
 - string ServerVersion – строка, содержащая номер версии, релиза, сборки СУБД ЛИНТЕР (например, 5.9.25).
- 4) PDO::ATTR_CONNECTION_STATUS – статус соединения.
Возможные значения:
 - 0 – соединение с БД не установлено;
 - 1 – соединение с БД установлено.

- 5) PDO::ATTR_SERVER_INFO – информация о сервере БД в виде ассоциированного массива, содержащего параметры соединения (см. пункт «Получить параметры соединения с БД» в разделе «Linter PHP-интерфейс»).
- 6) PDO::ATTR_CLIENT_VERSION – версия используемого драйвера `php_pdo_linter`.
- 7) PDO::ATTR_MAX_COLUMN_LEN – максимальная длина ответа.

Описание

Функция предоставляет текущее значение запрошенного атрибута соединения.

Возвращаемое значение

Результат выполнения функции:

- значение запрошенного атрибута (удачное завершение);
- NULL-значение – неудачное завершение.

Получить код завершения

Назначение

Получение последнего кода завершения.

Синтаксические правила

```
string PDO::errorCode();
```

Описание

Функция возвращает SQLSTATE-код завершения последнего обработанного ядром СУБД ЛИНТЕР SQL-запроса.

Возвращаемое значение

Пяти символьное алфавитно-цифровое значение, определяемое стандартом ANSI SQL-92 для идентификации кодов завершения.

Получить информацию о коде завершения

Назначение

Получение расширенной информации о коде завершения.

Синтаксические правила

```
array PDO::errorInfo();
```

Описание

Функция возвращает расширенную информацию о коде завершения последнего обработанного ядром СУБД ЛИНТЕР SQL-запроса.

Возвращаемое значение

Массив, содержащий информацию о коде завершения.

Элементы массива:

- 0 – код завершения в формате SQLSTATE (см. пункт «Получить код завершения»);
- 1 – числовой код завершения ядра СУБД ЛИНТЕР;
- 2 – текстовая расшифровка кода завершения ядра СУБД ЛИНТЕР.

Обрамление строки

Назначение

Обрамление строки заданными символами.

Синтаксические правила

```
string PDO::quote(string string [,int parameter_type]);
```

string

Исходная строка.

parameter_type

Тип данных исходной строки. Допустимое значение PDO::PARAM_STR (оно же используется и по умолчанию).

Описание

Функция выполняет следующие действия:

- обрамляет кавычками исходную строку;
- заменяет внутри исходной строки одиночные кавычки на пару одиночных кавычек.

Возвращаемое значение

- 0 – нормальное завершение;
- 1 – ошибочный аргумент функции.

Примеры

Исходная строка	Результирующая строка
abc	'abc'
ab\'c	'ab\'c'
ab'c	'ab''c'
ab'\'c	'ab''\'c'

Подготовить запрос к выполнению

Назначение

Подготовка SQL-оператора к выполнению.

Синтаксические правила

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

statement

SQL-оператор, подготавливаемый для обработки СУБД ЛИНТЕР.

driver_options

Массив пар <атрибут значение>, которые будут присвоены новому PDOStatement-объекту.

Описание

Функция подготавливает SQL-оператор для последующего выполнения его функцией PDOStatement::execute. Текст SQL-оператора может содержать ноль или более именованных (:<имя>) или нумерованных (?) параметров, значение которым будет присваиваться в момент обработки SQL-оператора ядром СУБД ЛИНТЕР.

Требования к тексту SQL-оператора:

- если SQL-оператор содержит параметры, то они должны быть однотипными: либо все именованные, либо все нумерованные;
- уникальный параметр можно задавать для любого значения, которое должно быть передано на обработку (функции PDOStatement::execute кроме имен объектов БД);
- дублирование именованных параметров запрещается;
- нельзя привязывать несколько значений одному именованному параметру;
- для выполнения SQL-оператора с различными значениями параметров необходимо каждый раз вызывать функцию PDO::prepare, а затем PDOStatement::execute.

Возвращаемое значение

PDOStatement-объект, соответствующий заданному SQL-оператору.

Пример

```
<?php
...
$select = $dbh->prepare('SELECT COUNT(*) FROM auto where
color=:color_auto');
$select->execute(array('color_auto'=>'BLACK'));
$res = $select->fetchColumn();

echo ('BLACK auto: '.$res);
?>
```

Выполнить запрос с предоставлением статистики

Назначение

Выполнение оператора модификации данных с предоставлением статистики реально модифицированных записей.

Синтаксические правила

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

statement

SQL-оператор.

Описание

Функция выполняет SQL-запрос и возвращает количество реально обработанных (добавленных, удаленных, модифицированных) записей.

Функция не возвращает количество строк ответа поискового запроса (SELECT) (для этого необходимо использовать функцию PDO::query).

Возвращаемое значение

Количество реально обработанных строк

Пример

```
<?php
...
$rows = $dbh->exec("DELETE FROM auto WHERE color = 'abc'");
echo("Deleted $rows rows.\n");
?>
```

Выполнить запрос

Назначение

Подготовка и выполнение SQL-оператора.

Синтаксические правила

PDOStatement PDO::query (string statement);

statement

SQL-оператор для подготовки и последующего немедленного выполнения.

Описание

Функция выполняет SQL-оператор и возвращает результирующую выборку данных (если она не пуста) как PDOStatement объект.

Если данный SQL-оператор должен выполняться многократно (с различными значениями одних и тех же параметров) лучше один раз его подготовить к выполнению (с помощью функции PDO::prepare) и затем периодически использовать функцию PDOStatement::execute.

Перед очередным вызовом функции PDO::query с поисковым SQL_оператором предыдущая результирующая выборка данных должна быть обработана полностью (т. е. все строки ответа должны быть извлечены клиентским приложением из выборки), в противном случае необходимо использовать функцию PDOStatement::closeCursor для освобождения ресурсов, выделенных для хранения строк ответа предыдущей выборки

Возвращаемое значение

Результаты поискового запроса в виде PDOStatement-объект.

Пример

```
<?php
function getFruit($conn) {
    $sql = 'SELECT make, model FROM auto limit 3';
    foreach ($conn->query($sql) as $row) {
        print $row['MAKE'] . "\t";
        print $row['MODEL'] . "\n";
    }
}
?>
```

Результат выполнения данного примера:

```
FORD          MERCURY COMET GT V8
ALPINE        A-310
AMERICAN MOTORS MATADOR STATION
```

Получить последнее значение идентификатора записи или последовательности

Назначение

Получение идентификатора последней вставленной записи или очередное значение последнее значение последовательности.

string PDO::lastInsertId ([string name]);

name

Имя последовательности.

Описание

Функция предоставляет значение псевдостолбца LAST_ROWID или значение CURVAL заданной последовательности (см. документ «СУБД ЛИНТЕР. Справочник по SQL»).

Возвращаемое значение

Если имя последовательности не задано, возвращается системный идентификатор последний добавленной в БД записи по данному соединению.

Если имя последовательности задано, возвращается последнее выбранное из данной последовательности значение.

Пример

```
<?php
...
$dbh->exec("CREATE or replace public sequence test_seq
Increment by 5 minvalue 0;");
$dbh->exec("select TEST_SEQ.NEXTVAL;");
$dbh->exec("select TEST_SEQ.NEXTVAL;");
$res = $dbh->lastInsertId("TEST_SEQ");
echo ("LastSeqValue is $res\n");
?>
```

Результат выполнения данного примера:

```
LastSeqValue is 5
```

PDOStatement – класс операторов

Объект данного класса представляет собой подготовленный SQL-оператор. Если оператор является поисковым (SELECT) оператором, то объект содержит и набор данных, полученный в результате выполнения этого оператора.

Установить атрибут оператора

Назначение

Установка (изменение) атрибутов поискового SQL-оператора.

Синтаксические правила

```
bool PDOStatement::setAttribute (int attribute, mixed value);
```

attribute

Идентификатор атрибута.

value

Новое значение устанавливаемого атрибута.

Атрибуты и их допустимые значения:

- 1) PDO::ATTR_CURSOR_NAME – устанавливает имя курсора.
Допустимые значения:
 - string cursorName – имя курсора.
- 2) PDO::ATTR_DT_FORMAT – устанавливает формат значений типа «дата-время».
Допустимые значения:
 - string newDtFormat – шаблон значений типа «дата-время». Формат шаблона см. в документе «СУБД ЛИНТЕР. Справочник по SQL».
- 3) PDO::ATTR_FETCH_BLOB_AS_DATA – управляет представлением извлекаемых BLOB-данных.
Допустимые значения:
 - true – представлять BLOB-данные в бинарном виде;
 - false – представлять BLOB-данные в текстовом виде.

Описание

Функция устанавливает заданные атрибуты поискового SQL-оператора. Для непоисковых операторов установка атрибутов игнорируется.

Возвращаемое значение

- true – атрибуты оператора установлены;
- false – неуспешная попытка установки атрибутов.

Получить значение атрибута оператора**Назначение**

Предоставление текущего значения атрибута поискового SQL-оператора.

Синтаксические правила

mixed PDOStatement::getAttribute (int attribute);

attribute

Идентификатор запрашиваемого атрибута.

Кроме атрибутов, перечисленных в пункте «Установить атрибут поискового оператора», дополнительно может быть запрошена информация о следующих атрибутах:

- 1) PDO::ATTR_CURSOR – тип курсора.
Возможные значения:
 - PDO::CURSOR_SCROLL – скролируемый курсор (перемещение по выборке вперед/назад).
- 2) PDO::ATTR_MAX_COLUMN_LEN – максимальная длина ответа.

Описание

Функция предоставляет текущее значение запрошенного атрибута поискового оператора связанного с данным PDOStatement-объектом.

Возвращаемое значение

- значение запрошенного атрибута (удачное завершение);
- NULL-значение – неудачное завершение.

Получить код завершения**Назначение**

Получение кода завершения последнего выполненного оператора.

Синтаксические правила

string PDOStatement::errorCode();

Описание

Функция предоставляет SQLSTATE-код завершения последнего обработанного ядром СУБД ЛИНТЕР SQL-оператора, связанного с данным PDOStatement-объектом..

Возвращаемое значение

Пяти символьное алфавитно-цифровое значение, определяемое стандартом ANSI SQL-92 для идентификации кодов завершения.

Получить информацию о коде завершения

Назначение

Получение расширенной информации о коде завершения последнего выполненного SQL-оператора..

Синтаксические правила

```
array PDOStatement::errorInfo();
```

Описание

Функция возвращает расширенную информацию о коде завершения последнего обработанного ядром СУБД ЛИНТЕР SQL-оператора, связанного с данным PDOStatement-объектом.

Возвращаемое значение

Массив, содержащий информацию о коде завершения.

Элементы массива:

- 0 – код завершения в формате SQLSTATE;
- 1 – числовой код завершения ядра СУБД ЛИНТЕР;
- 2 – текстовая расшифровка кода завершения ядра СУБД ЛИНТЕР.

Пример

```
<?php
$sth = $dbh->prepare('CREATE TABLE AUTO (id int)');
$sth->execute();

echo "\nPDOStatement::errorInfo():\n";
$arr = $sth->errorInfo();
print_r($arr);
?>
```

Результат выполнения данного примера:

```
PDOStatement::errorInfo():
Array
(
    [0] => 42S01
    [1] => 1503
    [2] => Таблица уже существует.
)
```

Выполнить подготовленный оператор

Назначение

Выполнение подготовленного SQL-оператора.

Синтаксические правила

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

input_parameters

Ассоциативный массив значений привязываемых параметров.

Описание

Если подготовленный оператор содержит не привязанные формальные параметры, функция привязывает переданные ей в аргументе фактические значения формальных параметров к SQL-оператору, после чего выполняет его.

Если подготовленный оператор содержит формальные параметры, а в функции массив значений параметров не передается (аргумент input_parameters не задан), то привязка значений параметров должна быть выполнена предварительно с помощью функции PDOStatement::bindParam.

Возвращаемое значение

- true - успешное выполнение функции;
- false - ошибка при выполнении функции.

Пример

```
<?php
    $c="BLACK";
    $w=3000;
    $sth = $dbh->prepare('SELECT count(*) from auto WHERE color = :c
AND weight > :w');
    $sth->execute(array(':c'=>$c, ':w'=>$w));
    $res = $sth->fetchColumn();

    print_r($res);
?>
```

Результат выполнения данного примера:

```
178
```

Получить количество реально обработанных записей

Назначение

Предоставление информации о результатах выполнения запроса манипулирования данными.

Синтаксические правила

```
int PDOStatement::rowCount();
```

Описание

Функция предоставляет информацию о количестве реально добавленных, удаленных или модифицированных записей при выполнении последнего DELETE, INSERT или UPDATE-оператора.

Возвращаемое значение

Количество реально обработанных записей

Пример

```
<?php
...
$dbh->exec('create or replace table test (i int)');

$sql = $dbh->prepare('insert into test(i) values(?)');

$sql->execute(array(1));
$sql->execute(array(1));
$sql->execute(array(2));
$sql->execute(array(3));

$sql = $dbh->prepare('delete from test where i>2');
$sql->execute();

$count = $sql->rowCount();
print("Deleted $count rows.\n");
...
?>
```

Результат выполнения данного примера:

```
Deleted 1 rows.
```

Получить число столбцов результирующей выборки

Назначение

Предоставление количества столбцов в результирующей выборке поискового SQL-оператора.

Синтаксические правила

```
int PDOStatement::columnCount();
```

Описание

Функция предоставляет информацию о количестве столбцов в результирующей выборке последнего поискового SQL-оператора

Если `columnCount` привязана к `PDOStatement`-объекту, полученному при выполнении функции `PDO::query`, то информация предоставляется сразу же.

Если же `PDOStatement`-объект был возвращен функцией `PDO::prepare`, то `columnCount` предоставит информацию только после выполнения функции `PDOStatement::execute`.

Если `columnCount` привязана к `PDOStatement`-объекту, не формирующему результирующую выборку, возвращается 0.

Возвращаемое значение

Количество столбцов в результирующей выборке.

Пример

```
<?php
...
    $sth = $dbh->prepare("select * from auto limit 1");
    $sth->execute();
    $colcount = $sth->columnCount();
    print("Result set has $colcount columns.\n");
...
?>
```

Результат выполнения данного примера:

```
Result set has 13 columns.
```

Получить описание столбца результирующей выборки

Назначение

Предоставление информации о характеристиках заданного столбца результирующей выборки.

Синтаксические правила

```
mixed PDOStatement::getColumnMeta (int column);
```

column

Номер столбца в результирующей выборке, информацию о котором необходимо получить. Нумерация столбцов начинается с нуля.

Описание

Функция предоставляет метаданные для заданного столбца результирующей выборки.

Возвращаемые значения

- массив, содержащий метаданные столбца;
- false, если столбец не найден.

Элементы массива:

- 1) string **native_type** – тип данных столбца в терминологии PHP;
- 2) string **decl_type** – тип данных столбца в терминологии СУБД ЛИНТЕР;
- 3) int **flags** – логическая комбинация флагов, представляющих атрибуты столбца. Зарезервирован.
- 4) string **name** – имя столбца.
- 5) int **len** – максимальная длина значений столбца.
- 6) int **precision** – точность данных столбца.
- 7) int **pdo_type** – значение константы, соответствующей типу данных столбца в терминологии PDO.



Метаданные столбца в полном объеме можно получить напрямую с помощью SQL-запроса к системной таблице \$\$\$ATTRI СУБД ЛИНТЕР.

Пример

```
<?php
...
$select = $dbh->query('SELECT * FROM auto');
$meta = $select->getColumnMeta(0);
var_dump($meta);
...
?>
```

Результат выполнения данного примера:

```
array(7) {
  ["native_type"]=>
  string(6) "string"
  ["linter:decl_type"]=>
  string(4) "char"
  ["flags"]=>
  array(0) {
  }
  ["name"]=>
  string(4) "MAKE"
  ["len"]=>
  int(20)
  ["precision"]=>
  int(0)
  ["pdo_type"]=>
  int(2)
}
```

Привязать значение к параметру

Назначение

Присвоение значения формальному параметру SQL-оператора.

Синтаксические правила

```
bool PDOStatement::bindValue (mixed parameter, mixed value [,int
date_type]);
```

parameter

Идентификатор параметра. Для именованных параметров – это имя параметра, для нумерованных параметров – порядковый номер параметра (отсчет начинается с 1).

value

Значение, присваиваемое параметру.

data_type

Тип данных параметра:

- 1) PDO::PARAM_NULL–NULL-значение;
- 2) PDO::PARAM_INT–целочисленный;
- 3) PDO::PARAM_STR–строковый (char, varchar);
- 4) PDO::PARAM_LOB–BLOB;
- 5) PDO::PARAM_STMT – зарезервировано;

6) PDO::PARAM_BOOL—логический.

Если аргумент не задан, по умолчанию подразумевается PDO::PARAM_STR.

Описание

Функция присваивает значение именованному или нумерованному параметру в подготовленном SQL-операторе.

Возвращаемые значения

- true - нормальное завершение;
- false – ошибка привязки параметра.

Пример

```
<?php
...
$make = 'FORD';
$color = 'RED';
$stmt = $dbh->prepare('SELECT count(*) FROM auto
    WHERE color = ? AND make = ?');
$stmt->bindValue(1, $color, PDO::PARAM_STR);
$stmt->bindValue(2, $make, PDO::PARAM_STR);
$stmt->execute();
echo $stmt->fetchColumn();
?>
```

Результат выполнения данного примера:

5

Привязать переменную к параметру

Назначение

Привязка PHP-переменной к заданному параметру SQL-оператора.

Синтаксические правила

```
bool PDOStatement::bindParam (mixed parameter, mixed &variable [,int
data_type [, int length [, mixed driver_options]]]);
```

parameter

Идентификатор параметра. Для именованных параметров – это имя параметра, для нумерованных параметров – порядковый номер параметра (отсчет начинается с 1).

variable

Имя PHP-переменной.

data_type

Тип данных параметра (см. функцию PDOStatement::bindValue).

Для возвращения INOUT-параметра из хранимой процедуры необходимо использовать оператор OR для установки бита PDO::PARAM_INPUT_OUTPUT.

length

Длина данных параметра.

driver_options

Аргумент зарезервирован для последующего применения.

Описание

Функция привязывает переменную к именованному или нумерованному параметру в подготовленном SQL-операторе. В отличие от функции `PDOStatement::bindValue` в текст подготовленного оператора подставляется не значение, а ссылка на переменную, значение которой будет использоваться в момент выполнения оператора.

Возвращаемые значения

- `true` – нормальное завершение;
- `false` – ошибка привязки переменной.

Пример

```
<?php
...
$val_weight = 3500;
$val_color = 'RED';
$stmt = $dbh->prepare('SELECT count(*) FROM auto WHERE
weight>:val_weight AND color=:val_color');
$stmt->bindParam(':val_weight', $val_weight, PDO::PARAM_INT);
$stmt->bindParam(':val_color', $val_color, PDO::PARAM_STR, 10);
$val_color = 'BLACK';
$stmt->execute();
echo $stmt->fetchColumn();
...
?>
```

Результат выполнения данного примера:

131

Привязать переменную к столбцу

Назначение

Привязка PHP-переменной к заданному столбцу результирующей выборки SQL-оператора.

Синтаксические правила

```
bool PDOStatement::bindColumn (mixed column, mixed &param [, int type[,
int $length [, mixed $driver_options ]]] );
```

column

Идентификатор столбца – имя столбца или его порядковый номер в результирующей выборке. Нумерация столбцов начинается с 1.

param

Имя PHP-переменной.

type

Тип данных параметра. (см. функцию PDOStatement::bindValue).

Length

Максимальная длина данных.

driver_options

Зарезервирован для последующего применения.

Описание

Функция привязывает PHP-переменную к заданному столбцу результирующей выборки, в результате чего при каждом вызове функции PDOStatement::fetch() или PDOStatement::fetchAll() привязанная переменная будет получать значение указанного столбца в текущей строке выборки.



Данная функция должна вызываться после выполнения функции PDOStatement::execute().

Возвращаемые значения

- true - нормальное завершение;
- false – ошибка привязки переменной.

Пример

```
<?php
...
$data1 = 0;
$data2 = 0;
$stmt = $dbh->prepare('SELECT 100 as column1, 500 as column2');
$stmt->execute();
$stmt->bindColumn('COLUMN1', $data1);
$stmt->bindColumn(2, $data2);
$stmt->fetch();
echo $data1."\n".$data2;
```

...

?>

Результат выполнения данного примера:

100

500

Установить режим выборки данных

Назначение

Установка режима выборки данных для поискового SQL-оператора.

Синтаксические правила

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

mode

Константа, соответствующая определенному режиму извлечения данных.

Константа	Режим выборки данных
PDO::FETCH_LAZY	Каждая строка выборки должна быть представлена в виде объекта, содержащего набор переменных, имена которых совпадают с именами столбцов результирующей выборки.
PDO::FETCH_ASSOC	Каждая строка выборки должна быть представлена в виде массива, индексируемого по именам столбцов результирующей выборки. Если строка ответа содержит дубликаты столбцов, возвращается только одно значение для этого имени столбца.
PDO::FETCH_NAMED	Каждая строка выборки должна быть представлена в виде массива, индексируемого по именам столбцов результирующей выборки. Если строка ответа содержит дубликаты столбцов, возвращается массив их значений.
PDO::FETCH_NUM	Каждая строка выборки должна быть представлена в виде массива, индексируемого по номерам столбцов результирующей выборки. Нумерация столбцов начинается с 0.
PDO::FETCH_BOTH	Каждая строка выборки должна быть представлена в виде массива, индексируемого по имени и номеру столбца результирующей выборки. Нумерация столбцов начинается с 0.
PDO::FETCH_OBJ	Каждая строка выборки должна быть представлена в виде объекта, содержащего свойства, имена которых соответствуют именам столбцов результирующей выборки.
PDO::FETCH_BOUND	Значения столбцов результирующей выборки должны присваиваться переменным, привязанным к оператору с помощью функции PDOStatement::bindParam() or PDOStatement::bindColumn(). Данный режим выборки должен всегда возвращать true.
PDO::FETCH_COLUMN	Должно возвращаться только одно значение указанного столбца результирующей выборки.
PDO::FETCH_CLASS	Должен возвращаться новый экземпляр заданного класса,

Константа	Режим выборки данных
	отображающий столбцы в именованные свойства этого класса.
PDO::FETCH_INTO	Должен возвращаться существующий обновленный экземпляр заданного класса, отображающий столбцы в именованные свойства этого класса.
PDO::FETCH_FUNC	Каждая строка выборки должна передаваться на обработку пользовательской функции.
PDO::FETCH_GROUP	Должен возвращаться ассоциированный массив, содержащий значения заданной группы столбцов результирующей выборки. Для задания группы столбцов должна использоваться битовая маска флага PDO::FETCH_COLUMN совместно с флагом данного режима.
PDO::FETCH_UNIQUE	Должен возвращаться ассоциированный массив, содержащий уникальные значения заданной группы столбцов результирующей выборки. Для задания группы столбцов должна использоваться битовая маска флага PDO::FETCH_COLUMN совместно с флагом данного режима.
PDO::FETCH_CLASSTYPE	Имя класса определяется по значению первого столбца результирующей выборки. Используется совместно с режимом PDO::FETCH_CLASS, т. е. PDO::FETCH_CLASS PDO::FETCH_CLASSTYPE.

Возвращаемые значения

- 1 – нормальное завершение;
- false – ошибка установки режима выборки.

Пример

```
<?php
...
$sql = 'SELECT make, model, color FROM auto fetch first 3';
try {
    $stmt = $dbh->query($sql);
    $result = $stmt->setFetchMode(PDO::FETCH_NUM);
    while ($row = $stmt->fetch()) {
        print $row[0] . "\t" . $row[1] . "\t" . $row[2] . "\n";
    }
}
catch (PDOException $e) {
    print $e->getMessage();
}
?>
```

Результат выполнения данного примера:

FORD	MERCURY COMET GT V8	BLACK
ALPINE	A-310	WHITE
AMERICAN MOTORS	MATADOR STATION	BROWN

Получить заданную строку результирующей выборки

Назначение

Предоставление заданной строки результирующей выборки.

Синтаксические правила

```
mixed PDOStatement::fetch ([int fetch_style [, int cursor_orientation [, int cursor_offset]]]);
```

`fetch_style`

Режим выборки данных (одна из констант вида `PDO::FETCH_*`). Допустимые значения см. в пункте «Установить режим выборки данных». По умолчанию используется режим `PDO::FETCH_BOTH`.

`cursor_orientation`

Константа, определяющая возвращаемую строку результирующей выборки.

Константа	Возвращаемая строка
<code>PDO::FETCH_ORI_NEXT</code>	Следующая строка по отношению к текущей строке.
<code>PDO::FETCH_ORI_PRIOR</code>	Предыдущая строка по отношению к текущей строке.
<code>PDO::FETCH_ORI_FIRST</code>	Первая строка выборки.
<code>PDO::FETCH_ORI_LAST</code>	Последняя строка выборки.
<code>PDO::FETCH_ORI_ABS</code>	Строка с указанным в аргументе <code>offset</code> порядковым номером. Отсчет строк выборки начинается с 1. Е
<code>PDO::FETCH_ORI_REL</code>	Строка со смещением, задаваемым в аргументе <code>offset</code> , по отношению к текущей строке выборки. Если значение <code>offset</code> положительное, строка выбирается из последующих строк выборки, если отрицательное – из предыдущих.

Во всех случаях, когда результирующая выборка пуста или задаваемая строка выборки не существует, возвращается `NULL`.

Если номер выбираемой строки не задан, по умолчанию используется `PDO::FETCH_ORI_NEXT`.

`offset`

Абсолютный или относительный номер выбираемой строки (см. `PDO::FETCH_ORI_ABS`, `PDO::FETCH_ORI_REL`).

Описание

Функция возвращает запрашиваемую строку результирующей выборки (если она существует). PDOStatement-объект, соответствующий данной выборке, должен быть скролируемым курсором (т. е. должен быть установлен атрибут PDO::CURSOR_SCROLL при подготовке к выполнению поискового SQL-оператора с помощью функции [PDO::prepare\(\)](#)).

Возвращаемые значения

- true – запрошенная строка предоставлена;
- false – запрошенная строка не найдена.

Пример

```
<?php
...
    $sql = "SELECT 'Иванов'
           union
           SELECT 'Петров'
           union
           SELECT 'Сидоров'
           union
           SELECT 'Смирнов'";
    $stmt = $dbh->prepare($sql);
    $stmt->execute();
    $row = $stmt->fetch(PDO::FETCH_NUM, PDO::FETCH_ORI_ABS, 3);
    print $row[0]."\n";
}
?>
```

Результат выполнения данного примера:
Сидоров

Получить все строки результирующей выборки

Назначение

Получение всех строк результирующей выборки.

Синтаксические правила

```
array PDOStatement::fetchAll ([int fetch_style [, int column_index]]);
```

fetch_style

Режим выборки данных (одна из констант вида PDO::FETCH_*). Допустимые значения см. в пункте «Установить режим выборки данных». По умолчанию используется режим PDO::FETCH_BOTH.

column_index

Если установлен режим выборки PDO::FETCH_COLUMN, аргумент задает номер столбца, из которого нужно извлекать данные. По умолчанию 0.

Описание

Функция возвращает все строки результирующей выборки.

Каждая строка выборки будет представлена либо в виде массива собственно значений столбцов, либо в виде объекта со свойствами, имена которых соответствуют именам столбцов выборки (свойства объекта содержат ссылки на значение столбца, например, на BLOB-значение, которое трудно разместить непосредственно в массиве).

Если объем результирующей выборки будет очень большим, то может не хватить ресурсов операционной системы или сервера базы данных. В этом случае необходимо либо изменить алгоритм PHP-приложения, либо уменьшить объем выборки, передаваемый сервером БД PHP-приложению с помощью средств SQL (условия WHERE, предикаты (BETWEEN, IN и др.), группировка GROUP BY, удаление дубликатов DISTINCT и т. п.).

Возвращаемые значения

Массив значений результирующей выборки (нормальное завершение).

Если массив пустой, то запрошенных данных нет, либо задан некорректный оператор.

Пример

```
<?php
...
$sth = $dbh->prepare("select firstnam, name from auto, person
where auto.personid=person.personid and make='FORD'
and model='LINCOLN CONTINENTAL'
and color='YELLOW' and year=71");
$sth->execute();

/* Получить фамилию и имя владельцев автомобилей марки Lincoln
Continetal производства компании Ford желтого цвета 1971 года
выпуска */
print("Список владельцев автомобилей:\n");
$result = $sth->fetchAll();
print_r($result);
?>
```

Результат выполнения данного примера:

Список владельцев автомобилей:

```
Array
(
    [0] => Array
        (
            [FIRSTNAM] => JUSTIN
            [0] => JUSTIN
            [NAME] => SHAW
            [1] => SHAW
        )
)
```

Получить значение столбца результирующей выборки

Назначение

Предоставление значение заданного столбца результирующей выборки.

Синтаксические правила

```
string PDOStatement::fetchColumn ([int column_number]);
```

column_number

Номер столбца результирующей выборки, значение которого требуется получить. Нумерация столбцов начинается с 0. Если аргумент не задан, по умолчанию данные извлекаются из первого столбца.

Описание

Функция возвращает значение заданного столбца из следующей строки результирующей выборки.

Возвращаемые значения

- значение запрошенного столбца выборки;
- false – нет данных (конец результирующей выборки или несуществующий столбец)

Пример

```
<?php
...
    $sth = $dbh->prepare("SELECT 100,200,300");
    $sth->execute();
    $result = $sth->fetchColumn(2);
    print($result);
...
?>
```

Результат выполнения данного примера:
300

Освободить ресурсы

Назначение

Освобождение ресурсов, выделенных для хранения результирующей выборки.

Синтаксические правила

```
string PDOStatement::closeCursor();
```

Описание

Функция закрывает курсор и освобождает ресурсы, выделенные для хранения результирующей выборки.

Возвращаемые значения

- true – в случае успешного завершения;
- false – в случае ошибки.

Пример

```
<?php
/* Создание PDOStatement-объекта*/
$stmt = $dbh->prepare('SELECT * FROM auto');

/* Создание второго PDOStatement-объекта */
$otherStmt = $dbh->prepare('SELECT * from person');

/* Выполнение первого оператора */
$stmt->execute();

/* Получаем только одну строку из результирующей выборки. */
$stmt->fetch();

/* Остальные строки результирующей выборки не нужны. Закрываем этот
курсор и освобожденные ресурсы драйвер будет использовать для
обработки друго оператора */
$stmt->closeCursor();

/* После освобождения ресурсов можно выполнять второй оператор */
$otherStmt->execute();
?>
```

Получить следующий набор данных

Назначение

Получение доступа к следующему набору данных, связанному с данным PDOStatement-объектом.

Синтаксические правила

```
bool PDOStatement::nextRowset();
```

Описание

СУБД ЛИНТЕР не поддерживает работу с несколькими наборами данных, поэтому данная функция всегда возвращает false.

Возвращаемое значение

false – нет доступа к к следующему набору данных..

PDOException-класс исключений

Класс исключений PDO-интерфейса является потомком PHP-класса Exception. См. раздел «Исключения» («Exceptions») документации к PHP для более подробной информации о работе с данным классом.

Пример PHP-скрипта с использованием PDO-интерфейса

```
<?php
$dbh = new PDO("linter:node=;dbname=DEMO","SYSTEM","MANAGER");

$dbh->exec('CREATE TABLE test(id INT NOT NULL PRIMARY KEY, val1
VARCHAR(10), val2 VARCHAR(16))');

$select = $dbh->prepare('SELECT COUNT(id) FROM test');

$data = array(
    array('10', 'Abc', 'zxy'),
    array('20', 'Def', 'wvu'),
    array('30', 'Ghi', 'tsr'),
    array('40', 'Jkl', 'qpo'),
    array('50', 'Mno', 'nml'),
    array('60', 'Pqr', 'kji'),
);

// Insert using question mark placeholders
$stmt = $dbh->prepare("INSERT INTO test VALUES(?, ?, ?)");

foreach ($data as $row) {
    $stmt->execute($row);
}
$select->execute();
$num = $select->fetchColumn();
echo 'There are ' . $num . " rows in the table.\n";

$select->closeCursor();

// Insert using named parameters
$stmt2 = $dbh->prepare("INSERT INTO test VALUES(:first,
:second, :third)");
foreach ($data as $row) {
    $stmt2->execute(array(':first'=>($row[0] + 5),
':second'=>$row[1],
':third'=>$row[2]));
}

$select->execute();
$num = $select->fetchColumn();
echo 'There are ' . $num . " rows in the table.\n";
$dbh = null;
?>
```

Приложение 1

Типы данных СУБД ЛИНТЕР

<u>Тип данных</u>	<u>Описание</u>
LDT_INTEGER	Целые числа
LDT_BIGINT	Длинное целое
LDT_SMALLINT	Короткое целое
LDT_REAL	Вещественные числа
LDT_DOUBLE	Вещественное двойной точности
LDT_NUMERIC	Числа с фиксированной точкой ¹⁾
LDT_DECIMAL	
LDT_DATE	Дата-время
LDT_BLOB	Blob-данные
LDT_CHAR	Символьные строки фиксированной длины
LDT_VARCHAR	Символьные строки переменной длины
LDT_BYTE	Байтовые строки фиксированной длины
LDT_VARBYTE	Байтовые строки переменной длины
LDT_BOOLEAN	Логическое значение
LDT_NCHAR	Unicode-строки фиксированной длины ²⁾
LDT_NVARCHAR	Unicode-строки переменной длины ²⁾
LDT_EXTFILE	Внешние файлы

¹⁾ Данные типа NUMERIC (DECIMAL) выдаются в виде строк. Это влияет на результаты жесткого сравнения (с помощью == и на результаты функции gettype()). Чтобы получать этот тип данных в виде LDT_DOUBLE, необходимо в файл настроек php (php.ini) добавить следующие строки

```
:  
[linter]  
linter.decimal_as_double = On
```

(это работает для php версии >= 4.1.0)

См. пример demo9.php в подкаталоге samples\php установочного каталога СУБД ЛИНТЕР.

²⁾ При работе со строками типа NCHAR и NVARCHAR необходимо помнить, что соответствующая этому типу данных кодировка в PHP называется UCS-2. При этом порядок байтов зависит от архитектуры компьютера и его нужно задавать самостоятельно (UCS-2LE или UCS-2BE). Для работы с различными кодировками в PHP можно использовать, например, модуль mbstrings:

```
if ( !extension_loaded("mbstring") ) dl("php_mbstring.dll")  
Linter_Exec_Direct ($con, "select nchar_string from table1 ;");  
$data = Linter_Get_Data_Array ($con);  
$value = mb_convert_encoding ($data[0], "CP1251", "UCS-2LE");
```

См. пример demo7.php в подкаталоге samples\php установочного каталога СУБД ЛИНТЕР.

Приложение 2

Коды завершения Linter PHP-интерфейса

<u>Имя константы</u>	<u>Значение</u>	<u>Описание</u>
LPE_SUCCESS	0	Успешное завершение
LPE_LINTER_ERROR	-1	Произошла ошибка СУБД ЛИНТЕР
LPE_NO_MEMORY	-2	Недостаточно памяти
LPE_INVALID_PARAM	-3	Функции был передан неверный параметр
LPE_INVALID_CONNECT	-4	Неверный идентификатор связи
LPE_INVALID_CONNECT	-5	Неверный идентификатор курсора
LPE_INVALID_COL_NUM	-6	Неверный номер колонки
LPE_INVALID_PARAM_NUM	-7	Неверный номер параметра
LPE_INVALID_PARAM_NAME	-8	Неверное имя параметра
LPE_INVALID_STATE	-9	Недопустимое состояние курсора
LPE_INVALID_DIRECT	-10	Неверное направление выбора
LPE_UNKNOWN_OPTION	-11	Неизвестная опция курсора
LPE_BAD_DT_FORMAT	-12	Неверный формат дата/время
LPE_NO_BLOB_COLUMN	-13	Отсутствие Blob-столбца в курсоре
LPE_NOT_CURSOR	-14	Объект не является курсором. Возможно, это связь.
LPE_WRONG_PARAM_COUNT	-15	Функции было передано недопустимое число параметров
LPE_NOT_IMPLEMENTED	-999	Свойство не реализовано

Указатель функций Linter PHP-интерфейса

Linte Blob_Add, 21
Linte Blob_Add_Ex, 22
Linte Blob_Append, 24
Linte Blob_Append_Ex, 24
Linte Blob_Clear, 25
Linte Blob_Fetch, 23
Linte Blob_Get_Data, 25
Linte Blob_Get_Size, 24
Linte Blob_Purge, 23
Linte Close_Connect, 9
Linte Close_Cursor, 12
Linte Connect_Info, 9
Linte Error_Msg, 26
Linte Exec_Direct, 14
Linte Execute, 16
Linte Fetch, 16
Linte Fetch_Array, 20
Linte Fetch_Row, 18
Linte Get_Col_Prop, 15, 20
Linte Get_Cursor_Opt, 12
Linte Get_Data_Array, 19
Linte Get_Data_Row, 17
Linte Get_Data_Rows, 18
Linte GETM, 18
Linte Last_Error, 25
Linte Open_Connect, 8
Linte Open_Cursor, 11
Linte Prepare, 15
Linte Set_Codepage, 11
Linte Set_Cursor_Opt, 13

Указатель функций Linter DBX-интерфейса

dbx_close, 29

dbx_compare, 29

dbx_connect, 28

dbx_error, 30

dbx_query, 31

dbx_sort, 33

Указатель функций Linter Pear::db – интерфейса

affectedRows, 36
connect, 34
createSequence, 36
disconnect, 37
dropSequence, 37
execute, 37
executeMultiple, 38
fetchInfo, 44
fetchRow, 45
free, 47
getAll, 38
getCol, 39
getListOff, 40
getOne, 40
getRow, 40
isError, 35
limitQuery, 41
nextId, 41
nextResult, 47
numCols, 47
numRows, 48
prepare, 42
query, 43
quote, 43
setFetchMode, 44
tableInfo, 48

Указатель функций Linter PDO–интерфейса

new PDO, 51

PDO

beginTransaction, 52

commit, 53

rollback, 53

setAttribute, 54

getAttribute, 56

errorCode, 57

errorInfo, 57

quote, 58

prepare, 59

exec, 59

query, 60

lastInsertId, 61

PDOStatement

setAttribute, 62

getAttribute, 63

errorCode, 63

errorInfo, 64

execute, 65

rowCount, 65

columnCount, 66

getColumnMeta, 67

bindValue, 68

bindParam, 69

bindColumn, 71

setFetchMode, 72

fetch, 74

fetchAll, 76

fetchColumn, 77

closeCursor, 78

nextRowset, 78

