

МОБИЛЬНАЯ
РЕЛЯЦИОННАЯ
СУБД **ЛИНТЕР**[®]

Linter Standard
Linter Bastion
Linter RealTime
Linter Multiversion

PERL-интерфейсы

НАУЧНО-ПРОИЗВОДСТВЕННОЕ ПРЕДПРИЯТИЕ

 **РЕЛАКС**[®]

Товарные знаки

РЕЛЭКС™, ЛИНТЕР® , НЕВОД® , ЛАВ™, 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®, ЛАВ™, 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
Назначение документа	1
Для кого предназначен документ	1
Необходимые предварительные знания	1
Принятые обозначения и соглашения	1
Дополнительные документы	2
Общие сведения о PERL-интерфейсах	3
LinPerl-интерфейс	4
Условия применения	4
Функции	4
Общие сведения	4
Установить соединение с БД	4
Получить параметры соединения с БД	5
Установить пользовательскую кодировку	7
Закрыть соединение с БД	8
Открыть курсор	8
Получить значение параметра курсора	9
Установить значение параметра курсора	11
Закрыть курсор	11
Выполнить курсорный запрос без подготовки	12
Подготовить курсорный запрос для выполнения	12
Привязать значение параметра к курсорному запросу	13
Привязать массив значений параметров к курсорному запросу	13
Выполнить подготовленный курсорный запрос	14
Перейти в заданную строку ответа курсорного запроса	14
Привязать переменную к столбцу	15
Отменить привязку переменной к столбцу	16
Получить значение столбца	16
Получить массив значений строки ответа	17
Получить массивы значений нескольких строк ответа	18
Получить ассоциативный массив значений строки ответа	19
Проверить значение столбца на NULL-значение	19
Добавить нетипизированные данные к первому BLOB-столбцу	20
Добавить типизированные данные к первому BLOB-столбцу	21
Добавить нетипизированные данные к заданному BLOB-столбцу	21
Добавить типизированные данные к заданному BLOB-столбцу	22
Получить тип BLOB-данных	23
Удалить BLOB-данные первого BLOB-столбца	24
Удалить BLOB-данные заданного BLOB-столбца	24
Получить размер BLOB-данных	25
Получить порцию BLOB-данных первого BLOB-столбца	25
Получить порцию BLOB-данных заданного BLOB-столбца	26
Подтверждение выполнения транзакции	27
Отмена выполнения транзакции	27
Получить описание атрибутов столбца	28
Получить последний код завершения	29

Оглавление

Получить описание кода завершения СУБД ЛИНТЕР	29
DBI-интерфейс	31
Общие сведения	31
DBD для СУБД ЛИНТЕР	31
Необходимые условия	31
Установка в ОС типа Unix	31
Динамические атрибуты DBI	32
Методы	32
Создать новый объект	32
Создать новое соединение	32
Выполнение методов DBD-драйвера	33
Получить код завершения последнего метода	33
Прямое обращение к базе данных	34
Подготовка SQL-оператора к выполнению	34
Закрыть соединение с базой данных	35
Подтверждение изменений в базе данных	36
Отмена изменений в базе данных	36
Выполнить SQL-предложение	36
Сохранить данные	37
Останов СУБД ЛИНТЕР	38
Выполнение хранимой процедуры	38
Привязка формальных параметров	39
Выполнение предложения	39
Получить количество записей	40
Выборка данных	40
Получить массив данных	41
Преобразование ESC-последовательностей	42
Завершить обработку предложения	43
Получить порцию BLOB-данных	43
Добавить порцию BLOB-данных	44
Удалить BLOB-данные	44
Приложение 1. Типы данных СУБД ЛИНТЕР	46
Приложение 2. Пример обработки типов данных в DBI-интерфейсе	47
Приложение 3. Коды завершения LinPerl-интерфейса	48
Приложение 4. Коды завершения DBI-интерфейса	49
Указатель функций	50
Указатель методов	51

Предисловие

Назначение документа

Документ содержит описание PERL–интерфейса 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 PERL с использованием СУБД ЛИНТЕР.

Необходимые предварительные знания


Для работы необходимо:

- знать основы реляционных баз данных и языка баз данных SQL;
- знать спецификации DBX;
- знать язык программирования PERL;
- уметь работать в соответствующей операционной системе на уровне простого пользователя.

Принятые обозначения и соглашения

<u>Обозначение</u>	<u>Пример</u>	<u>Значение</u>
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

Предисловие

Обозначение	Пример	Значение
Заглавные буквы	BROWSE	Названия команд, слова, зарезервированные в SQL, ключевые слова
Курсив в <>	<i><return statement></i>	Определяемый элемент синтаксической конструкции
Символ ::=		Равенство по определению. Слева от знака стоит определяемое понятие, справа – собственно определение понятия
Квадратные скобки []	DBSTORE [-d -r -t -u]	Необязательные элементы конструкции. В данном примере ключи не являются обязательными элементами команды
Вертикальная черта	<i><return value></i> ::= <i><value expression></i> NULL	Указывает на то, что все предшествующие ей элементы списка являются необязательными и могут быть заменены любым другим элементом списка после этой черты
Фигурные скобки { }	CODEPAGE {866 1251 KOI8}	Указывают на то, что все находящееся внутри них является единым целым
Многоточие «...»	Характеристики столбца MAKE CHAR(20) MODEL CHAR(20) ... SQL>	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

Дополнительные документы

- СУБД ЛИНТЕР. Архитектура СУБД.
- СУБД ЛИНТЕР. Справочник по SQL.
- СУБД ЛИНТЕР. Справочник кодов завершения.

Общие сведения о PERL-интерфейсах

PERL - это система разработки скриптов, включающая в себя CGI - интерфейс (Common Gateway Interface - стандартный шлюзовой интерфейс, предназначенный для создания серверных приложений HTTP), интерпретатор языка и набор функций для доступа к базам данных (БД) и различным объектам Web-сервера. PERL является удобным средством разработки приложений WWW и интерфейсов к БД в Интернет.

В системе PERL СУБД ЛИНТЕР всегда рассматривается как удаленный SQL-сервер, то есть для доступа к БД создается сетевое соединение. Благодаря этому возможно открывать из одного скрипта либо несколько пользовательских сессий, либо работать с различными SQL-серверами. После установки соединения с сервером можно отправлять и обрабатывать SQL-запросы. При выполнении запроса создается объект, в котором хранится результат выполнения запроса, после чего с помощью специальных функций можно получать отдельные записи запроса.

LinPerl-интерфейс

Условия применения

Для выполнения PERL-программы на компьютере должен быть установлен интерфейс DBI версия 1.02 или выше.

Функции

Общие сведения

- 1) Все символьные параметры функций чувствительны к регистру символов.
- 2) Если длина символьного параметра функции превышает максимально допустимую в СУБД ЛИНТЕР, то значение параметра усекается до допустимой длины и функция выполняется с усеченным значением параметра.
- 3) Все функции возвращают результат типа INTEGER.

Возможные коды завершения:

**Возвращаемое
значение**

Описание

0 (LPE_SUCCESS)

Нормальное завершение, возвращаемое значение отсутствует

Положительное

Возвращаемое функцией значение (нормальное завершение)

Отрицательное

Неудачное завершение функции

- 4) Все возвращаемые значения передаются через параметр(ы) функции.
- 5) Причиной неудачного завершения функции может являться как ошибка PERL-модуля, так и результат обработки функции СУБД ЛИНТЕР. Если причина в СУБД ЛИНТЕР, то возвращается код завершения LPE_LINTER_ERROR. Все остальные отрицательные коды относятся к ошибкам PERL-модуля (см. приложение 3). Для получения дополнительной информации об ошибке используйте функцию GetLastErrorMsg (см. стр. 29).

Установить соединение с БД

Назначение

Функция OpenConnect устанавливает связь между PERL-модулем и СУБД ЛИНТЕР на узле с заданным именем пользователя и паролем.

Синтаксические правила

OpenConnect (<пользователь>, <пароль>, <сервер>, <режим>, <идентификатор соединения>);

<Пользователь>

Имя пользователя БД. Регистрозависимая символьная строка длиной не более 66 символов.

<Пароль>

Пароль пользователя. Регистрозависимая символьная строка длиной не более 18 символов.

<Сервер>

Имя ЛИНТЕР-сервера, с которым необходимо установить соединение. Символьная строка длиной не более 8 символов. Если параметр не задан, предполагается локальный компьютер.

<Режим>

Задаёт режим обработки транзакций и кодовую страницу (значение 0 или логическая комбинация опций <режим транзакции> и <кодовая страница>).

<Режим транзакции>:

- TM_AUTOCOMMIT;
- TM_OPTIMISTIC;
- TM_EXCLUSIVE

<Кодовая страница>:

- CP_1251;
- CP_KOI8;
- CP_866.

Если значение параметра равно 0, значения опций используются по умолчанию.

Возвращаемое значение

<Идентификатор соединения>с БД.

Пример

```
$err = OpenConnect("KENT", "ALEX", "LINTER", CP_866 |
    MODE_EXCLUSIVE, $con);
$err && [code for handling error]
```

См. также функции:

CloseConnect.

Получить параметры соединения с БД**Назначение**

Получить информацию о параметре БД, с которой установлено соединение, или о параметре самого соединения с БД.

Синтаксические правила

GetConnectInfo(<идентификатор соединения>, <ассоциативный массив>);

<Идентификатор соединения>

Идентификатор установленного соединения с БД.

<Ассоциативный массив>

Адрес ассоциативного массива.

Возвращаемое значение

<Ассоциативный массив> параметров БД (таблица 1).

Таблица 1. Ассоциативный массив параметров БД

Ключ массива	Возвращаемое значение ключа
VERMAJOR	Старшая версия СУБД
VERMINOR	Младшая версия СУБД
VERBUILD	Номер сборки СУБД
SORTPOOLSIZE	Размер пула сортировки (в страницах по 4 Кбайт)
KERNELPOOLSIZE	Размер пула ядра СУБД (в страницах по 4 Кбайт)
FILEQUEUEUSE	Размер очереди файлов
USERQUEUEUSE	Размер очереди пользователей
TABLEQUEUEUSE	Размер очереди таблиц
COLUMNQUEUEUSE	Размер очереди столбцов
CHANNELQUEUEUSE	Размер очереди каналов
SNAPTIMEOUT	Период времени между операциями полного Snap
KILLTIMEOUT	Тайм-аут опроса существования клиента
BASENAME	Имя БД
SYSLOG	Признак активности журнала транзакций.
SYNC	Признак синхронизации ввода/вывода
LOG	Признак ведения файла-протокола
OS	Идентификатор ОС
NUMOFSORT	Количество файлов сортировки
FLAGS_CSETCHG	Кодовая страница не найдена, используется по умолчанию. Только для версии 6.0 и выше
FLAGS_KERNELENGLOCALE	Используется англоязычная кодовая страница. Только для версии ниже 6.0

Ключ массива	Возвращаемое значение ключа
FLAGS_EXTPASS	Пароль пользователю дан администратором , должен быть изменен
FLAGS_PASSEND	Пароль пользователя должен быть изменен
FLAGS_KERNELINVBYTEORDER	Порядок байт на сервере и клиенте совпадает/не совпадает
FLAGS_KERNELDEMOLIC	БД с демо-лицензией
FLAGS_KERNELDEMOLICEXP	Срок лицензии на демо-БД закончился/не закончился
MAXRECSIZE	Максимальный размер записи в таблице. Только для версии 6. 0 и выше
CHARSET	Текущая кодировка БД. Только для версии 6. 0 и выше
DEFCHARSET	Кодировка по умолчанию БД. Только для версии 6. 0 и выше
USECHARSET	Установлена пользовательская кодировка соединения. Только для версии 6. 0 и выше
USECHARSETNAME	Имя установленной пользовательской кодировки. Только для версии 6. 0 и выше

Пример

```
$err = GetConnectInfo($cur, \%hash);
$build = $hash{"VERBUILD"};
$os = $hash{"OS"};
$err && [code for handling error]
```

См. также функции:

OpenConnect, GetColInfo.

Установить пользовательскую кодировку

Назначение

Установить новую кодировку во всех активных соединениях приложения с БД и во всех курсорах, созданных в этих соединениях.

Синтаксические правила

SetCodepage (<кодовая страница>);

<Кодовая страница>

Имя кодовой страницы.

Пример

```
$err = SetCodepage("KOI8-R");  
$err && [code for handling error]
```

См. также функции:

OpenConnect, GetConnectInfo.

Закрывать соединение с БД

Назначение

Закрывает установленное с БД соединение.

Синтаксические правила

CloseConnect(<идентификатор соединения>);

<Идентификатор соединения>

Идентификатор ранее установленного соединения с БД.



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

Пример

```
$err = CloseConnect($con);  
$err && [code for handling error]
```

См. также функции:

OpenConnect.

Открыть курсор

Назначение

Открывает курсор между PERL-модулем и СУБД ЛИНТЕР.

Синтаксические правила

OpenCursor(<идентификатор соединения>, <курсor>, <идентификатор курсора>);

<Идентификатор соединения>

Идентификатор родительского соединения.

<Имя курсора>

Символьная строка длиной до 18 символов.

<Идентификатор курсора>

Переменная языка PERL



В текущей версии параметр <имя курсора> не поддерживается.

Возвращаемое значение

<Идентификатор курсора>.



Режим транзакции и кодовая страница создаваемого курсора наследуются из родительского соединения.

Пример

```
$err = OpenCursor($con, "MY_CURSOR", $cur);
$err && [code for handling error]
```

См. также функции:

OpenConnect, CloseCursor.

Получить значение параметра курсора

Назначение

Получить значение параметров курсора.

Синтаксические правила

GetCursorOption(<идентификатор курсора>, <параметр курсора>, <значение параметра>);

<Идентификатор курсора>

Идентификатор созданного курсора.

<Параметр курсора>

Идентификатор запрашиваемого значения параметра курсора (таблица 2).

<Значение параметра>

Переменная языка PERL.

Таблица 2. Идентификаторы параметров курсора

Параметр курсора	Описание
CO_CURNAME	Имя курсора
CO_ASYNCMODE	Режим асинхронного выполнения курсора
CO_SYNCMODE	Режим синхронного выполнения курсора
CO_ASYNCDONE	Уведомление о завершении асинхронного вызова
CO_DTFORMAT	Формат данных типа DATE
CO_COLCOUNT	Количество столбцов в ответе

Параметр курсора	Описание
CO_ROWCOUNT	Количество строк в ответе
CO_ERRORROW	Количество строк в сообщении об ошибке
CO_ERRPOS	Указывать местоположение ошибки в строке
CO_TRANSMODE	Режим транзакций (может быть установлен в версии 5.7 и выше)
CO_CURROW	Текущая строка в запросе выборки
CO_CURROWID	RowId текущей строки
CO_CONNECTID	Идентификатор соединения
CO_NODENAME	Имя ЛИНТЕР-сервера, с которым установлено соединение
CO_PRIORITY	Приоритет курсора (в версии 5.7 и выше)
CO_QUERY_PRIORITY	Приоритет текущего запроса (в версии 5.7 и выше)
CO_CANCEL	Отмена выполнения запроса (в версии 5.7 и выше)
CO_PAUSE	Приостанов выполнения запроса (в версии 5.7 и выше)
CO_CONTINUE	Возобновление выполнения запроса (в версии 5.7 и выше)
CO_LAST_ROWID	Предоставление идентификатора последней записи. Должен использоваться только после INSERT-запроса
CO_FETCH_BLOBS_AS_USUAL_DATA	Загружать в строку ответа BLOB-данные (при отключенной опции загрузка BLOB-данных игнорируется). Должен использоваться только после INSERT-запроса

Возвращаемое значение

Значение запрошенного параметра в переменной <значение параметра>.

Пример

```
$err = GetCursorOption($cursor, CO_ROWCOUNT, $rows);  
$err && ... to do something for error handling ...
```

См. также функции:

SetCursorOption, GetConnectInfo, GetColInfo.

Установить значение параметра курсора

Назначение

Установить значение параметра курсора.

Синтаксические правила

```
SetCursorOption(<идентификатор курсора>, <параметр курсора>, <значение  
параметра>);
```

<Идентификатор курсора>

Идентификатор созданного курсора.

<Параметр курсора>

Идентификатор параметра курсора, которому устанавливается новое значение (таблица 2).

<Значение параметра>

Переменная языка PERL, содержащая новое значение параметра.

Пример

```
$err = SetCursorOption($cursor, CO_PRIORITY, 10);  
$err && ... to do something for error handling ...
```

См. также функции:

GetCursorOption.

Закреть курсор

Назначение

Закрывает ранее открытый курсор.

Синтаксические правила

```
CloseCursor(<идентификатор курсора>);
```

<Идентификатор курсора>

Идентификатор созданного курсора.

Пример

```
$err = CloseCursor($cur);  
$err && [code for handling error]
```

См. также функции:

OpenCursor.

Выполнить курсорный запрос без подготовки

Назначение

Выполнить курсорный запрос без предварительной подготовки.

Синтаксические правила

ExecDirect(<идентификатор курсора>, <SQL-запрос>);

<Идентификатор курсора>

Идентификатор созданного курсора.

<SQL-запрос>

Символьная строка, содержащая текст SQL-запроса.



Текст SQL-запроса должен завершаться символом «;».

Пример

```
$err = ExecDirect($cur, "select * from auto;");  
$err && [code for handling error]
```

См. также функции:

Fetch, Prepare, BindParameter, BindParamArray, Execute.

Подготовить курсорный запрос для выполнения

Назначение

Подготовка курсорного запроса для выполнения.

Синтаксические правила

Prepare(<идентификатор курсора>, <SQL-запрос>);

<Идентификатор курсора>

Идентификатор созданного курсора.

<SQL-запрос>

Символьная строка, содержащая текст SQL-запроса с параметрами.



Текст SQL-запроса должен завершаться символом «;».

Пример

```
$err = Prepare($cur, "select * from auto where personid = ?;");  
$err && [code for handling error]
```

См. также функции:

BindParameter, BindParamArray, Execute, ExecDirect.

Привязать значение параметра к курсорному запросу

Назначение

Привязать значения параметра к курсорному запросу.

Синтаксические правила

```
BindParameter(<идентификатор курсора>, <параметр>, <значение параметра>);
```

<Идентификатор курсора>

Идентификатор созданного курсора.

<Параметр>

Номер параметра в претранслированном (подготовленном с помощью функции Prepare) SQL-запросе в данном курсоре. Нумерация параметров начинается с 1.

<Значение параметра>

Значение, которое должно быть присвоено указанному параметру.



Если ранее уже была сделана привязка параметра, то она будет отменена.

Пример

```
$err = BindParameter($cur, 1, $my_param);  
$err && [code for handling error]
```

См. также функции:

Prepare, Execute, BindParamArray.

Привязать массив значений параметров к курсорному запросу

Назначение

Привязать массив значений параметров к курсорному запросу.

Синтаксические правила

```
BindParameterArray(<идентификатор курсора>, <массив значений> [, <начальный номер>, <конечный номер>]);
```

<Идентификатор курсора>

Идентификатор созданного курсора.

<Массив значений>

Число элементов в массиве параметров должно быть больше или равно количеству параметров в подготовленном запросе.

<Начальный номер>

Номер параметра в претранслированном (подготовленном с помощью функции Prepare) SQL-запросе в данном курсоре, начиная с которого должна выполняться привязка параметров.

<Конечный номер>

Номер параметра в претранслированном (подготовленном с помощью функции Prepare) SQL-запросе в данном курсоре, в котором должна закончиться привязка параметров.

Нумерация параметров начинается с 1.

Если аргументы <начальный номер> или <конечный номер> не заданы, по умолчанию <начальный номер> берется равным 1, а <конечный номер> – последнему номеру параметра в подготовленном SQL-запросе.



Если ранее уже была сделана привязка параметра, то она будет отменена.

Пример

```
$err = BindParamArray($cur, 1, 3, \@my_param);  
$err && [code for handling error]
```

См. также функции:

Prepare, Execute, BindParameter.

Выполнить подготовленный курсорный запрос

Назначение

Выполнение подготовленного курсорного запроса.

Синтаксические правила

Execute(<идентификатор курсора>);

<Идентификатор курсора>

Идентификатор курсора, в котором следует выполнить последний подготовленный SQL-запрос.

Пример

```
$err = Execute($cur);  
$err && [code for handling error]
```

См. также функции:

ExecDirect, Prepare, BindParameter, BindParamArray, Fetch.

Перейти в заданную строку ответа курсорного запроса

Назначение

Перемещение в заданную строку ответа курсорного запроса.

Синтаксические правила

Fetch(<идентификатор курсора>, <направление перемещения>, <номер строки>);

<Идентификатор курсора>

Идентификатор курсора, в котором выполнен запрос выборки данных.

<Направление перемещения>

Задаёт новое местоположение указателя текущей строки ответа.

Возможные значения параметра:

- FETCH_FIRST – перемещение на первую строку;
- FETCH_LAST – перемещение на последнюю строку;
- FETCH_NEXT – перемещение на следующую строку;
- FETCH_PREV – перемещение на предыдущую строку;
- FETCH_ABSNUM – перемещение на заданную строку.

<Номер строки>

Целочисленное положительное значение, задающее порядковый номер строки, в которую надо переместиться в текущей выборке. Параметр используется только с параметром FETCH_ABSNUM.

Нумерация строк начинается с 1.

Пример

```
$err = Fetch($cur, FETCH_NEXT, 0);
$err = Fetch($cur, FETCH_ABSNUM, 100);
$err && [code for handling error]
```

См. также функции:

ExecDirect, Execute

Привязать переменную к столбцу**Назначение**

Привязать переменную к столбцу ответа курсорного запроса.

Синтаксические правила

BindColumn(<идентификатор курсора>, <номер столбца>, <переменная>);

<Идентификатор курсора>

Идентификатор курсора, в котором выполнен запрос выборки данных.

<Номер столбца>

Целочисленное положительное значение, задающее номер столбца в строке выборки, к которому надо привязать переменную. Нумерация столбцов начинается с 1.

<Переменная>

Задаёт переменную, в которую будет помещаться текущее значение выбранного столбца при перемещении по строкам ответа курсорного запроса.

Возвращаемое значение

После выполнения курсорного запроса значение заданного столбца текущей строки ответа будет находиться в привязанной переменной.

Если значение столбца NULL, возвращается NULL-значение.

Если тип данных столбца BLOB и включена настройка CO_FETCH_BLOBS_AS_USUAL_DATA, то возвращается содержимое BLOB-столбца, иначе – пустая строка.

Если тип столбца BOOLEAN, то возвращается строка TRUE или FALSE.

Если тип столбца EXTFILE, то возвращается ошибка.

Пример

```
$err = BindColumn($cur, 1, $make);  
$err && [code for handling error]
```

См. также функции:

GetDataColumn, Fetch, UnbindColumn.

Отменить привязку переменной к столбцу

Назначение

Отменить ранее произведенную привязку переменной к столбцу ответа курсорного запроса.

Синтаксические правила

```
UnbindColumn(<идентификатор курсора>, <номер столбца>);
```

<Идентификатор курсора>

Идентификатор курсора, в котором выполнен запрос выборки данных.

<Номер столбца>

Целочисленное положительное значение, задающее номер столбца в строке выборки, для которого надо отменить привязку переменной. Нумерация столбцов начинается с 1.

Пример

```
$err = UnbindColumn($cur, 1);  
$err && [code for handling error]
```

См. также функции:

BindColumn.

Получить значение столбца

Назначение

Получить в переменной значение заданного столбца текущей строки ответа курсорного запроса.

Синтаксические правила

GetDataColumn(<идентификатор курсора>, <номер столбца>, <переменная>);

<Идентификатор курсора>

Идентификатор курсора, в котором выполнен запрос выборки данных.

<Номер столбца>

Целочисленное положительное значение, задающее номер столбца в строке выборки, значение которого необходимо получить. Нумерация столбцов начинается с 1.

<Переменная>

Задаёт переменную, в которую будет помещено текущее значение выбранного столбца строки ответа курсорного запроса.

Возвращаемое значение

Значение столбца в заданной переменной.

Если значение столбца NULL, значение переменной также NULL.

Если тип данных столбца BLOB и включена настройка CO_FETCH_BLOBS_AS_USUAL_DATA, то возвращается содержимое BLOB-столбца, иначе – пустая строка.

Если тип столбца BOOLEAN, то возвращается строка TRUE или FALSE.

Если тип столбца EXTFILE, то возвращается ошибка.

 Для получения значения столбца типа BLOB рекомендуется использовать функцию BLOBGetData.

Пример

```
$err = GetDataColumn($cur, 1, $make);
$err && [code for handling error]
```

См. также функции:

ExecDirect, Execute, Fetch, BLOBGetData, GetDataRow.

Получить массив значений строки ответа**Назначение**

Получить массив значений текущей строки ответа курсорного запроса.

Синтаксические правила

GetDataRow(<идентификатор курсора>, <массив>);

<Идентификатор курсора>

Идентификатор курсора, в котором выполнен запрос выборки данных.

<Массив>

Адрес массива для записи значений строки ответа.

Возвращаемое значение

Массив значений столбцов текущей строки ответа курсорного запроса. Фактическое количество помещенных в массив значений зависит от размера выделенного массива

Пример

```
$err = ExecDirect($cur, "select * from auto;");  
$err && ... to do something for error handling ...  
$err = GetDataRow($cur, \@res);  
$err && ... to do something for error handling ...  
$make=$res[0];  
$model=$res[1];
```

См. также функции:

ExecDirect, Execute, Fetch, GetDataColumn, GetDataArray, GetDataRows.

Получить массивы значений нескольких строк ответа

Назначение

Получить массивы значений нескольких строк ответа курсорного запроса.

Синтаксические правила

GetDataRowS | GetM (<идентификатор курсора>, <массив>, <количество строк>);

<Идентификатор курсора>

Идентификатор курсора, в котором выполнен запрос выборки данных.

<Массив>

Адрес массива для записи значений строк ответа.

<Количество строк>

Целое неотрицательное значение. Отсчет начинается с 1

Возвращаемое значение

Двумерный массив значений заданного количества строк ответа курсорного запроса.

Отсчет строк ответа начинается с текущей строки курсорного запроса.

Если в курсорном запросе нет нужного количества строк ответа, возвращается сколько есть.



С функциональной точки зрения функции GetDataRowS и GetM идентичны. Отличие в том, что при выполнении функции GetM СУБД ЛИНТЕР использует пакетную загрузку данных в массив, что при большой количестве запрашиваемых строк может дать выигрыш по времени.

Пример

```

$serr = ExecDirect($cur, "select * from auto;");
$serr && ... to do something for error handling ...
$res = GetDataRows($cur, \@res, 5);
$serr && ... to do something for error handling ...
$make1=$res[0][0]; # марка машины в первой строке (1-й столбец)
$model1=$res[0][1] # модель машины в первой строке (2-й столбец)
...
$make3=$res[2][0]; # марка машины в третьей строке (1-й столбец)
$model3=$res[2][1] # модель машины в третьей строке (2-й столбец)

```

См. также функции:

ExecDirect, Execute, Fetch, GetDataColumn, GetDataRow, GetDataArray

Получить ассоциативный массив значений строки ответа**Назначение**

Получить ассоциативный массив значений текущей строки ответа курсорного запроса.

Синтаксические правила

GetDataArray(<идентификатор курсора>, <массив>);

<Идентификатор курсора>

Идентификатор курсора, в котором выполнен запрос выборки данных.

<Массив>

Адрес ассоциативного массива для записи значений строк ответа.

Возвращаемое значение

Ассоциированный массив значений текущей строки ответа курсорного запроса.

Пример

```

$serr = ExecDirect($cur, "select * from auto;");
$serr && ... to do something for error handling ...
$res = GetDataArray($cur, \@res);
$serr && ... to do something for error handling ...
$make=$res{"MAKE"};
$model=$res{"MODEL"};

```

См. также функции:

ExecDirect, Execute, Fetch, GetDataColumn, GetDataRow

Проверить значение столбца на NULL-значение**Назначение**

Проверить значение заданного столбца текущей строки ответа курсорного запроса на NULL-значение.

Синтаксические правила

TestNull(<идентификатор курсора>, <номер столбца>, <результат проверки>);

<Идентификатор курсора>

Идентификатор курсора, в котором выполнен запрос выборки данных.

<Номер столбца>

Целочисленное положительное значение, задающее номер столбца в текущей строке выборки, значение которого необходимо проверить. Нумерация столбцов начинается с 1.

<Результат проверки>

Переменная, в которую будет помещен результат проверки на NULL-значение.

Возвращаемое значение

Результат проверки в переменной:

- 1 – значение столбца NULL;
- 0 – значение столбца не NULL.

Пример

```
$err = TestNull($cur, 1, $is_null);  
$err && [code for handling error]  
if ($is_null != 0) ...
```

См. также функции:

GetDataColumn, BLOBGetData, GetDataRow, GetDataArray.

Добавить нетипизированные данные к первому BLOB-столбцу

Назначение

Добавить порцию данных к первому найденному BLOB-столбцу (если такой имеется) в текущей строке курсорного запроса.

Синтаксические правила

BLOBAppend(<идентификатор курсора>, <порция данных>);

<Идентификатор курсора>

Идентификатор курсора, в котором выполнен запрос выборки данных.

<Порция данных>

Адрес буфера, содержащего добавляемую порцию данных.

Возвращаемое значение

Порция данных добавляется в конец существующих BLOB-данных.

По умолчанию типу добавляемых BLOB-данных присваивается значение 0 (устанавливается при добавлении первой порции BLOB-данных)

Пример

```
$err = BLOBAppend($cur, $blob);
$err && [code for handling error]
```

См. также функции:

BLOBClear, BLOBGetSize, BLOBGetData, BLOBAdd, BLOBAppendEx

Добавить типизированные данные к первому BLOB-столбцу**Назначение**

Добавить типизированную порцию данных к первому найденному BLOB-столбцу (если такой имеется) в текущей строке курсорного запроса.

Синтаксические правила

BLOBAppendEx(<идентификатор курсора>, <порция данных>, <тип данных>);

<Идентификатор курсора>

Идентификатор курсора, в котором выполнен запрос выборки данных.

<Порция данных>

Адрес буфера, содержащего добавляемую порцию данных.

<Тип данных>

Целочисленное значение, характеризующее тип BLOB-данных.

 Значение типа данных от -11 по -20 зарезервированы СУБД ЛИНТЕР для внутреннего использования.

Возвращаемое значение

Порция данных добавляется в конец существующих BLOB-данных.

Тип BLOB-данных устанавливается при добавлении только первой порции данных, при добавлении остальных порций игнорируется.

Пример

```
$err = BLOBAppendEx($cur, $blob, $blob_type);
$err && [code for handling error]
```

См. также функции:

BLOBAppend, BLOBClear, BLOBGetSize, BLOBGetData, BLOBAddEx.

Добавить нетипизированные данные к заданному BLOB-столбцу**Назначение**

Добавить порцию данных к заданному BLOB-столбцу (если такой имеется) в текущей строке курсорного запроса.

Синтаксические правила

BLOBAdd(<идентификатор курсора>, <порция данных>, <номер столбца>);

<Идентификатор курсора>

Идентификатор курсора, в котором выполнен запрос выборки данных.

<Порция данных>

Адрес буфера, содержащего добавляемую порцию данных.

<Номер столбца>

Целочисленное положительное значение, задающее номер столбца в текущей строке выборки, к которому необходимо добавить порцию данных. Нумерация столбцов начинается с 1



В СУБД ЛИНТЕР версии ниже 5.9 аргумент <номер столбца> игнорируется.

Возвращаемое значение

Порция данных добавляется в конец существующих BLOB-данных.

По умолчанию типу добавляемых BLOB-данных присваивается значение 0 (устанавливается при добавлении первой порции BLOB-данных).

Пример

```
$err = BLOBAdd($cur, $blob);  
$err && [code for handling error]
```

См. также функции:

BLOBAppend, BLOBAddEx, BLOBPurge, BLOBFetch.

Добавить типизированные данные к заданному BLOB-столбцу

Назначение

Добавить типизированную порцию данных к заданному BLOB-столбцу (если такой имеется) в текущей строке курсорного запроса.

Тип BLOB-данных устанавливается при добавлении только первой порции данных, при добавлении остальных порций игнорируется.



В СУБД ЛИНТЕР версии ниже 5.9 аргумент *номер столбца* игнорируется.

Синтаксические правила

BLOBAddEx(<идентификатор курсора>, <порция данных>, <номер столбца>, <тип данных>);

<Идентификатор курсора>

Идентификатор курсора, в котором выполнен запрос выборки данных.

<Порция данных>

Адрес буфера, содержащего добавляемую порцию данных.

<Номер столбца>

Целочисленное положительное значение, задающее номер столбца в текущей строке выборки, к которому необходимо добавить порцию данных. Нумерация столбцов начинается с 1

<Тип данных>

Целочисленное значение, характеризующее тип BLOB-данных.

 Значение типа данных от -11 по -20 зарезервированы СУБД ЛИНТЕР для внутреннего использования.

Возвращаемое значение

Порция данных добавляется в конец существующих BLOB-данных.

Тип BLOB-данных устанавливается при добавлении только первой порции данных, при добавлении остальных порций игнорируется.

Пример

```
$err = BLOBAddEx($cur, $blob, 3, $blob_type);
$err && [code for handling error]
```

См. также функции:

BLOBAppend, BLOBAppendEx, BLOBAdd, BLOBPurge, BLOBFetch.

Получить тип BLOB-данных

Назначение

Получить тип BLOB-данных заданного BLOB-столбца (если такой имеется) в текущей строке курсорного запроса .

Синтаксические правила

BLOBGetType(<идентификатор курсора>, <тип данных>, <номер столбца>);

<Идентификатор курсора>

Идентификатор курсора, в котором выполнен запрос выборки данных.

<Тип данных>

Целочисленная переменная языка PERL.

<Номер столбца>

Целочисленное положительное значение, задающее номер столбца в текущей строке выборки, для которого необходимо получить тип его BLOB-данных. Нумерация столбцов начинается с 1

Возвращаемое значение

Тип BLOB-данных в переменной <Тип данных>.

Пример

```
$err = BLOBGetType($cur, $blob_type, $column);
$err && [code for handling error]
```

См. также функции:

BLOBAppendEx, BLOBAddEx, BLOBGetSize.

Удалить BLOB-данные первого BLOB-столбца

Назначение

Удалить полностью BLOB-данные первого найденного BLOB-столбца (если такой имеется) текущей строки курсорного запроса.

Синтаксические правила

BLOBClear(<идентификатор курсора>);

<Идентификатор курсора>

Идентификатор курсора, в котором выполнен запрос выборки данных.

Пример

```
$err = BLOBClear($cur);  
$err && [code for handling error]
```

См. также функции:

BLOBAppend, BLOBGetSize, BLOBGetData, BLOBPurge

Удалить BLOB-данные заданного BLOB-столбца

Назначение

Удалить полностью BLOB-данные заданного BLOB-столбца (если такой имеется) текущей строки курсорного запроса.

Синтаксические правила

BLOBPurge(<идентификатор курсора>, <номер столбца>);

<Идентификатор курсора>

Идентификатор курсора, в котором выполнен запрос выборки данных.

<Номер столбца>

Целочисленное положительное значение, задающее номер столбца в текущей строке выборки, у которого необходимо удалить его BLOB-данные. Нумерация столбцов начинается с 1.



В СУБД ЛИНТЕР версии ниже 5.9 аргумент <номер столбца> игнорируется.

Пример

```
$err = BLOBPurge($cur, $column);  
$err && [code for handling error]
```

См. также функции:

BLOBAppend, BLOBAdd, BLOBGetSize, BLOBGetData, BLOBFetch, BLOBClear.

Получить размер BLOB-данных

Назначение

Получить размер BLOB-данных заданного BLOB-столбца текущей строки курсорного запроса.

Синтаксические правила

`BLOBGetSize(<идентификатор курсора>, <размер> [, <номер столбца>]);`

<Идентификатор курсора>

Идентификатор курсора, в котором выполнен запрос выборки данных.

<Размер>

Переменная языка PERL.

<Номер столбца>

Целочисленное положительное значение, задающее номер столбца в текущей строке выборки, для которого необходимо получить тип его BLOB-данных. Нумерация столбцов начинается с 1. Если аргумент не задан, по умолчанию используется 1.

Возвращаемое значение

Размер BLOB-данных (в байтах) в переменной <размер>.

Пример

```
$err = BLOBGetSize($cur, $blob_size);
$err && [code for handling error]
```

См. также функции:

`BLOBAppend`, `BLOBAdd`, `BLOBClear`, `BLOBPurge`, `BLOBGetData`, `BLOBFetch`,
`BLOBGetType`

Получить порцию BLOB-данных первого BLOB-столбца

Назначение

Получить порцию BLOB-данных первого найденного BLOB-столбца (если такой имеется) текущей строки курсорного запроса.

Синтаксические правила

`BLOBGetData(<идентификатор курсора>, <начало порции>, <размер>, <буфер>);`

<Идентификатор курсора>

Идентификатор курсора, в котором выполнен запрос выборки данных.

<Начало порции>

Относительный номер байта, с которого начинается требуемая порция данных. Нумерация байтов начинается с 1.

<Размер>

Переменная языка PERL, задающая размер требуемой порции данных.

<Буфер>

Буфер для размещения порции данных.

Возвращаемое значение

Порция BLOB-данных в в заданном <буфере>.

Реальная длина переданных данных в переменной <размер>.

Пример

```
$err = BLOBGetData($cur, 1, 1000, $blob);  
$err && [code for handling error]
```

См. также функции:

BLOBAppend, BLOBClear, BLOBGetSize, GetDataColumn, BLOBFetch.

Получить порцию BLOB-данных заданного BLOB-столбца

Назначение

Получить порцию BLOB-данных заданного BLOB-столбца (если такой имеется) текущей строки курсорного запроса.

Синтаксические правила

BLOBFetch(<идентификатор курсора>, <ачало порции>, <размер>, <буфер>, <номер столбца>);

<Идентификатор курсора>

Идентификатор курсора, в котором выполнен запрос выборки данных.

<Начало порции>

Относительный номер байта, с которого начинается требуемая порция данных. Нумерация байтов начинается с 1.

<Размер>

Переменная языка PERL, задающая размер требуемой порции данных.

<Буфер>

Буфер для размещения порции данных.

<Номер столбца>

Целочисленное положительное значение, задающее номер столбца в текущей строке выборки, из которого необходимо извлечь порцию BLOB-данных. Нумерация столбцов начинается с 1. Если аргумент не задан, по умолчанию используется 1.



В СУБД ЛИНТЕР версии ниже 5.9 аргумент <номер столбца> игнорируется.

Возвращаемое значение

Порция BLOB-данных в в заданном <буфере>.

Реальная длина переданных данных в переменной <размер>.

Пример

```
$err = BLOBFetch($cur, 1, 1000, $blob, $column);  
$err && [code for handling error]
```

См. также функции:

BLOBAppend, BLOBAdd, BLOBClear, BLOBPurge, BLOBGetData, GetDataColumn.

Подтверждение выполнения транзакции

Назначение

Подтвердить изменения, внесенные в БД, при выполнении текущей транзакции курсорного запроса.

Синтаксические правила

Commit(<идентификатор курсора>);

<Идентификатор курсора>

Идентификатор курсора, в котором внесены изменения в БД.

Возвращаемое значение

Все изменения данных, выполненные при обработке транзакции, фиксируются в БД.

Пример

```
$err = Commit($cur);  
$err && [code for handling error]
```

См. также функции:

ExecDirect, Execute, Rollback

Отмена выполнения транзакции

Назначение

Отменить изменения, внесенные в БД, при выполнении текущей транзакции курсорного запроса.

Синтаксические правила

Rollback(<идентификатор курсора>);

<Идентификатор курсора>

Идентификатор курсора, в котором внесены изменения в БД.

Возвращаемое значение

Все изменения данных, выполненные при обработке транзакции, отменяются и в БД не сохраняются.

Пример

```
$err = Rollback($cur);  
$err && [code for handling error]
```

См. также функции:

ExecDirect, Execute, Commit

Получить описание атрибутов столбца

Назначение

Получить описание атрибутов заданного столбца текущей строки курсорного запроса.

Синтаксические правила

GetColInfo(<идентификатор курсора>, <номер столбца>, <массив>);

<Идентификатор курсора>

Идентификатор курсора, в котором выполнен запрос выборки данных.

<Номер столбца>

Целочисленное положительное значение, задающее номер столбца в текущей строке выборки. Нумерация столбцов начинается с 1.

<Массив>

Адрес массива для значений атрибутов столбца.

Возвращаемое значение

Ассоциативный массив значений атрибутов столбца (таблица 3).

Таблица 3. Ассоциативный массив значений атрибутов столбца

Ключ массива	Возвращаемое значение ключа
is_null	Столбец допускает (значение 1) / не допускает (значение 0) NULL-значение.
type	Числовое значение типа данных столбца (см. приложение1).
type_name	Имя типа данных столбца (см. приложение1). Имя указывается без префикса LDT_.
length	Длина данных.
precision	Точность представления (для вещественных значений).
scale	Масштаб (для вещественных значений).
name	Имя столбца.
table	Имя таблицы, которой принадлежит столбец.
user	Имя владельца таблицы.

Пример

```
$err = GetColInfo($cur, 1, $type, $len, $name);
$err && [code for handling error]
```

См. также функции:

ExecDirect, Execute, GetConnectInfo, GetCursorOption.

Получить последний код завершения

Назначение

Получить код завершения СУБД ЛИНТЕР или операционной системы последнего курсорного запроса.

Синтаксические правила

`GetError(<идентификатор курсора>, <код СУБД>, <код ОС>);`

<Идентификатор курсора>

Идентификатор курсора, в котором выполнен SQL-запрос.

<Код СУБД>

Переменная языка PERL.

<Код ОС>

Переменная языка PERL.

Возвращаемое значение

<u>Переменная</u>	<u>Значение</u>
<код СУБД>	Код завершения последнего выполненного SQL-запроса в данном курсоре
<код ОС>	Код завершения операционной системы, соответствующий последнему выполненному SQL-запросу в данном курсоре

Пример

```
$err = GetError($cur, $linerr, $syserr);
$err && [code for handling error]
```

См. также функции:

`GetErrorMsg`.

Получить описание кода завершения СУБД ЛИНТЕР

Назначение

Получить описание кода завершения СУБД ЛИНТЕР.

Синтаксические правила

`GetErrorMsg(<идентификатор курсора>, <код СУБД>, <текст сообщения>);`

<Идентификатор курсора>

Идентификатор курсора, в котором выполнен SQL-запрос.

<Код СУБД>


Числовой код завершения обработки SQL-запроса ядром СУБД ЛИНТЕР.

<Текст сообщения>

Переменная языка PERL.

Возвращаемое значение

Текстовое описание заданного кода завершения СУБД ЛИНТЕР в переменной <текст сообщения>.

 Для выполнения функции в БД должна существовать таблица SYSTEM.ERRORS. В противном случае возвращается пустая строка.

DBI-интерфейс

Общие сведения

DBI-интерфейс (Data Base Interface) является интерфейсом доступа к БД, написанным в форме PERL-модуля. Этот модуль определяет набор «методов» и «атрибутов», через которые осуществляется доступ к БД из PERL-программы. Конкретная реализация методов специфична для каждой БД и реализуется с помощью DBD-драйверов (Data Base Driver), разрабатываемых для каждой СУБД. Этот драйвер также имеет форму PERL-модуля.

DBI работает как переключатель между PERL-программой и DBD-драйвером.

Более подробно о DBI можно узнать на сайте

<http://www.symbolstone.org/technology/perl/DBI>

DBD для СУБД ЛИНТЕР

Необходимые условия

Драйвер требует Perl 5.005 и DBI версии не ниже 1.02

Установка в ОС типа Unix

Для установки интерфейса СУБД ЛИНТЕР с PERL-программой:

1. Перейти в подкаталог perl-dbi установочного каталога СУБД ЛИНТЕР. Подкаталог содержит исходные тексты драйвера и файл сборки DBD-драйвера ЛИНТЕР для DBI интерфейса языка PERL.
2. Выполнить команду `make`.

После успешного построения драйвера в каталоге `bin` дистрибутива СУБД ЛИНТЕР будет создан подкаталог `DBD` с файлами `Linter.pm` и `DBD/Linter.so`.

Получить доступ к DBD- драйверу можно следующими способами:

1. Добавить к переменной окружения `PERL5LIB` путь до каталога `bin` дистрибутива.
2. Скопировать файл `Linter.so` и каталог `DBD` с файлом `Linter.pm` в один из каталогов библиотек PERL. Полный список каталогов библиотек PERL можно получить, с помощью команды `'perl -v'`
3. Выполнить команду `make install`.
4. Выполнить команду `make pinstall` (генерация и установка с помощью стандартного `Makefile.PL`).

Убедиться в работоспособности интерфейса можно, выполнив пример из каталога `samples/dbi` дистрибутива ЛИНТЕР. Для подключения модуля примера ОС необходимо изменить значение переменной окружения:

```
PERL5LIB=%PERL5LIB%;<LINTER_TOP_DIR>\INTLIB\PERL
```

Динамические атрибуты DBI

Динамические атрибуты содержат информацию о результате выполнения последнего вызванного метода. Их значения должны анализироваться или использоваться до вызова другого метода.

<u>Атрибут</u>	<u>Описание</u>
DBI::errmsg	Числовой код завершения последнего выполненного метода
DBI::errstr	Текст кода завершения последнего выполненного SQL-запроса к СУБД ЛИНТЕР
DBD::Linter::VERSION	Версия DBD- драйвера СУБД ЛИНТЕР

Методы

Создать новый объект

Назначение

Метод `Driver` позволяет создать новый объект класса `Linter`.

Пакет

Package `Linter`.

Прототип

```
$drh = DBI->install_driver('Linter');
```

Возвращаемые значения

<u>Переменная</u>	<u>Описание</u>
<code>\$drh</code>	Описатель драйвера (в случае нормального завершения) и <code>undef</code> – в случае внутренней ошибки DBI



Метод автоматически вызывается методом `install_driver` в DBI.

Создать новое соединение

Назначение

Метод `Connect` осуществляет создание нового соединения с БД и объекта класса «соединение».

Пакет

Package `Linter::dr`

Прототип

```
$dbh = $drh->connect  
($dbname, $username, $password, [%attributes], 'Linter');
```

<u>Параметр</u>	<u>Описание</u>
<code>\$dbname</code>	Имя БД длиной не более 8 символов или " для соединения по умолчанию. Если длина имени больше 8 символов, то она усекается

<u>Параметр</u>	<u>Описание</u>
<code>\$username</code>	Имя пользователя БД (регистр символов учитывается) длиной не более 66 символов. Если длина имени больше 66 символов, то она усекается
<code>\$password</code>	Пароль пользователя БД (регистр символов учитывается) длиной не более 18 символов. Если длина пароля больше 18 символов, то она усекается
<code>\%attributes</code>	Ссылка на список атрибутов создаваемого соединения: <code>%attributes=</code> <code>('mode' =></code> <code>{autocommit optimis`tic exclusive},</code> <code>'codepage' =></code> <code>{CP_866 CP_1251 KOI8})</code> Значения по умолчанию: <code>'mode' =>autocommit</code> <code>'codepage'=>CP_866</code>

Возвращаемые значения

<u>Переменная</u>	<u>Описание</u>
<code>\$dbh:</code>	
<ul style="list-style-type: none"> описатель БД undef 	<p>Нормальное завершение</p> <p>Ошибка соединения с БД</p>
<code>\$DBI::errstr</code>	Диагностическое сообщение (в случае ошибки соединения с БД)

Выполнение методов DBD-драйвера

Назначение

Метод `Func` осуществляет выполнение методов, присущих только DBD-драйверу СУБД ЛИНТЕР.

Пакет

`Package Linter::dr`

Прототип

```
$drh->func(@function_paramaters, 'function_name');
```

Получить код завершения последнего метода

Назначение

Методом `Err` можно получить код завершения последнего выполненного метода.

Пакет

`Package Linter::dr`

Прототип

Этот метод доступен через вызов метода `func`.
`$drh->func('err')`

Прямое обращение к базе данных

Назначение

Метод `Lint` осуществляет прямое обращение к БД с помощью внутреннего (Call) интерфейса СУБД ЛИНТЕР.

Пакет

`Package Linter::dr`

Прототип

```
$rv = $drh->  
func(\%CBL, \ $arg2, \ $arg3, \ $arg4, \ $arg5, 'inter');
```

Значения полей контрольного блока обмена (CBL) зависят от выполняемой команды СУБД ЛИНТЕР (см. документ «СУБД ЛИНТЕР. Интерфейс нижнего уровня»).

Поля CBL можно заполнять двумя способами:

Первый способ:

```
%CBL = (  
    'CodErr' => \ $coderr,  
    'Prior' => \ $prior,  
    'NumChan' => \ $numchan,  
    'UserName' => \ $username,  
    'Command' => \ $command,  
    'Node' => \ $node,  
    'RowId' => \ $rowid,  
    'RowCount' => \ $rowcount,  
    'PrzExe' => \ $przexe,  
    'SysErr' => \ $syserr,  
    'LnBufRow' => \ $lnbufrow,  
    'Reserve' => \ $reserve,  
);
```

Второй способ:

```
$CBL{ 'NumChan' } = \ $numchan;  
$CBL{ 'Command' } = \ $command;  
$CBL{ 'PrzExe' } = \ $przexe;  
$CBL{ 'LnBufRow' } = \ $lnbufrow;  
...
```

Подготовка SQL-оператора к выполнению

Назначение

Метод `Prepare` выполняет подготовку SQL-оператора к выполнению и создает объект класса «предложение».

Пакет

`Package Linter::db`

Прототип

```
$sth = $dbh->prepare($statement)
```

<u>Параметр</u>	<u>Описание</u>
\$statement	Текст SQL-оператора, заканчивающийся знаком «;». SQL-оператор может включать неименованные (?) и/или именованные динамические параметры (см. документ «СУБД ЛИНТЕР. Справочник по SQL»(раздел «SQL-операторы с параметрами»)

Возвращаемые значения

<u>Переменная</u>	<u>Описание</u>
\$dbh:	
<ul style="list-style-type: none"> описатель предложения undef 	<p>Нормальное завершение</p> <p>Ошибка подготовки предложения</p>
\$DBI::errstr	Диагностическое сообщение (в случае ошибки подготовки предложения)

Закреть соединение с базой данных**Назначение**

Метод Disconnect закрывает соединение с БД, открытое ранее с помощью метода Connect.

Пакет

Package Linter::db

Прототип

```
$dbh->disconnect([$action])
```

<u>Параметр</u>	<u>Описание</u>
\$action	<p>Действие при закрытии соединения:</p> <ul style="list-style-type: none"> commit – сохранить в БД все изменения последней транзакции по соединению \$dbh; rollback - отменить изменения в БД <p>По умолчанию выполняется rollback</p>

Возвращаемые значения

<u>Переменная</u>	<u>Описание</u>
\$dbh:	
<ul style="list-style-type: none"> 1 0 	<p>Нормальное завершение</p> <p>Ошибка закрытия</p>
\$DBI::errstr	Диагностическое сообщение (в случае ошибки)

Подтверждение изменений в базе данных

Назначение

Метод `Commit` осуществляет подтверждение изменений в БД, произведенных последней выполнявшейся транзакцией.

Пакет

Package Linter::db

Прототип

```
$dbh->commit()
```

Возвращаемые значения

<u>Переменная</u>	<u>Описание</u>
<code>\$dbh:</code>	
<ul style="list-style-type: none">• 1• 0	Нормальное завершение Ошибка обработки транзакции
<code>\$DBI::errstr</code>	Диагностическое сообщение (в случае ошибки)

Отмена изменений в базе данных

Назначение

Метод `Rollback` выполняет отмену изменений в БД, произведенных последней выполнявшейся транзакцией.

Пакет

Package Linter::db

Прототип

```
$dbh->$dbh->rollback([$savepoint])
```



В текущей версии откат транзакции к точкам сохранения не поддерживается.

Возвращаемые значения

<u>Переменная</u>	<u>Описание</u>
<code>\$dbh:</code>	
<ul style="list-style-type: none">• 1• 0	Нормальное завершение Ошибка обработки транзакции
<code>\$DBI::errstr</code>	Диагностическое сообщение (в случае ошибки)

Выполнить SQL-предложение

Назначение

Метод `Do` осуществляет выполнение SQL-предложения (кроме `execute procedure`).

Пакет

Package Linter::db

Прототип

```
$dbh->do($statement, \%attributes, @values)
```


<u>Параметр</u>	<u>Описание</u>
<code>\$statement</code>	SQL-предложение
<code>%attributes</code>	Зарезервировано для будущего применения.
<code>@values</code>	Массив значений динамических параметров (если SQL- предложение содержит параметры)

Возвращаемые значения

<u>Переменная</u>	<u>Описание</u>
<code>\$dbh:</code>	
<ul style="list-style-type: none"> • количество обработанных записей • значение '0E0' • undef 	<p>Нормальное завершение</p> <p>Нет обработанных записей</p> <p>Ошибка обработки SQL-предложения</p>
<code>\$DBI::errstr</code>	Диагностическое сообщение (в случае ошибки)

Сохранить данные**Назначение**

Метод `Snapshot` выполняет принудительное сохранение данных на диск для повышения надежности.

 СУБД ЛИНТЕР обрабатываемые данные хранит в своем внутреннем буфере и выгружает их на диск по мере его заполнения. Метод `Snapshot` заставляет СУБД выгрузить данные на диск не дожидаясь заполнения буфера.

Пакет

```
Package Linter::db.
```

Прототип

```
$dbh->func($database, 'snap')
```

<u>Параметр</u>	<u>Описание</u>
<code>\$database</code>	Имя базы данных

Возвращаемые значения

<u>Переменная</u>	<u>Описание</u>
<code>\$dbh:</code>	
<ul style="list-style-type: none"> • 1 • 0 	<p>Нормальное завершение</p> <p>Ошибка выполнения метода</p>

Останов СУБД ЛИНТЕР

Назначение

Метод Shut выполняет останов СУБД ЛИНТЕР.

Пакет

Package Linter::db.

Прототип

```
$dbh->func($database, 'shut')
```

<u>Параметр</u>	<u>Описание</u>
\$database	Имя базы данных

Возвращаемые значения

<u>Переменная</u>	<u>Описание</u>
\$dbh:	
• 1	Нормальное завершение
• 0	Ошибка выполнения метода

Выполнение хранимой процедуры

Назначение

Метод Proc осуществляет выполнение хранимой процедуры.

Пакет

Package Linter::db

Прототип

```
$dbh->func($proc_name, @proc_parameters, 'proc')
```

```
@proc_parameters ::=  
  ([ $input_parameter, ]  
  [ \ $input_output_paramater, ]  
  [ \ $output_parameter ])
```

<u>Параметр</u>	<u>Описание</u>
\$proc_name	Имя хранимой процедуры
@proc_parameters	Массив параметров процедуры: <ul style="list-style-type: none">• \$input_parameter (параметры типа IN)• \$input_output_paramater (параметры типа INOUT)• \$output_parameter(параметры типа OUT)

Возвращаемые значения

<u>Переменная</u>	<u>Описание</u>
\$dbh:	
<ul style="list-style-type: none"> • скалярное значение или объект типа «предложение» • undef 	Нормальное завершение
\$DBI::errstr	Ошибка выполнения процедуры Диагностическое сообщение (в случае ошибки)

Привязка формальных параметров

Назначение

Метод `Bind_param` выполняет привязку формальных параметров SQL-оператора к значениям.

Пакет

Package Linter::st

Прототип

```
$sh->bind_param
($parameter_number,
 $parameter_value
 [, $parameter_type])
```

<u>Параметр</u>	<u>Описание</u>
\$parameter_number	Порядковый номер привязываемого параметра. Нумерация начинается с 1
\$parameter_value	Значение привязываемого параметра (в том числе допускается и NULL-значение)
\$parameter_type	Тип привязываемого параметра. Зарезервировано для дальнейшего использования

Возвращаемые значения

<u>Переменная</u>	<u>Описание</u>
\$sh:	
<ul style="list-style-type: none"> • 1 • 0 	Нормальное завершение Ошибка выполнения метода
\$DBI::errstr	Диагностическое сообщение (в случае ошибки)

Выполнение предложения

Назначение

Метод `Execute` осуществляет выполнение предложения, подготовленного методом `Prepare`.

Пакет

Package Linter::st

DBI-интерфейс

Прототип

```
$sh->execute ([@paramaters_array])
```

<u>Параметр</u>	<u>Описание</u>
@paramaters_array	Массив значений параметров, привязываемых к формальным параметрам SQL-предложения

Возвращаемые значения

<u>Переменная</u>	<u>Описание</u>
\$sh:	
<ul style="list-style-type: none">• количество обработанных записей• значение '0E0'• undef	Нормальное завершение Нет обработанных записей Ошибка обработки SQL-предложения
\$DBI::errstr	Диагностическое сообщение (в случае ошибки)

Получить количество записей

Назначение

С помощью метода `Rows` можно получить количество записей, выбранное (модифицированное) с помощью метода `Execute`.

Пакет

Package Linter::st

Прототип

```
$sh->rows ()
```

Возвращаемые значения

<u>Переменная</u>	<u>Описание</u>
\$sh:	
<ul style="list-style-type: none">• количество обработанных записей• <0	Нормальное завершение Ошибка обработки SQL-предложения
\$DBI::errstr	Диагностическое сообщение (в случае ошибки)

Выборка данных

Назначение

Метод `Fetchrow_arrayref` осуществляет выборку данных (в виде ссылки на массив), полученных с помощью метода `Execute`.

Пакет

Package Linter::st

Прототип

```
$ary = $sh-fetchrow_arrayref ([\%attributes])  
%attributes=
```

```
(
'direction'=> {'first' | 'last' | 'next' | 'prev' | 'number'}
'number'    => {1, 2, ...}
)
```

Параметр	Описание
'direction'	<p>Задаёт направление перемещения в массиве ответов выполненного SQL-оператора (как правило, SELECT):</p> <ul style="list-style-type: none"> 'first' - получить первую запись; 'last' - получить последнюю запись; 'next' - получить следующую запись; 'prev' - получить предыдущую запись; 'number'- получить запись с заданным номером

Возвращаемые значения

Переменная	Описание
\$ary:	
<ul style="list-style-type: none"> ссылка на массив данных, содержащий выбранную запись undef 	<p>Нормальное завершение</p> <p>Ошибка выполнения метода</p>
\$DBI::errstr	Диагностическое сообщение (в случае ошибки)



Количество выбранных записей возвращают методы Execute и Rows.

Получить массив данных

Назначение

С помощью метода Fetchrow_array можно получить результаты метода Execute в виде массива или списка данных.

Пакет

Package Linter::st

Прототип

```
$ary = $sh-fetchrow_array ([\%attributes])

%attributes=
(
'direction'=> {'first' | 'last' | 'next' | 'prev' | 'number'}
'number'    => {1, 2, ...}
)
```

Параметр	Описание
'direction'	<p>Задаёт направление перемещения в массиве ответов выполненного SQL-оператора (как правило, SELECT):</p> <ul style="list-style-type: none"> 'first' - получить первую запись; 'last' - получить последнюю запись; 'next' - получить следующую запись; 'prev' - получить предыдущую запись; 'number'- получить запись с заданным номером

Возвращаемые значения

<u>Переменная</u>	<u>Описание</u>
<code>\$ary</code> :	
<ul style="list-style-type: none">• массив или список данных, содержащий выбранную запись• undef	Нормальное завершение Ошибка выполнения метода
<code>\$DBI::errstr</code>	Диагностическое сообщение (в случае ошибки)



Количество выбранных записей возвращают методы `execute` и `rows`.

Пример

```
# выборка из БД каждой строки в виде списка
while (($name,$tip,$tip1)=$sh->fetchrow_array)
{
    print "$name $tip1 $tip2\n";
}

# выборка из БД каждой строки в виде массива
while (@row=$sh->fetchrow_array)
{
    print "$row[0] $row[1] $row[2]\n";
}
```

Цикл `while` выполняется до тех пор, пока метод `fetchrow_array` не вернёт `false`.

Преобразование ESC-последовательностей

Назначение

С помощью метода `Native_sql` можно получить SQL-выражение с преобразованными ESC-последовательностями.

Пакет

Package Linter::db

Прототип

```
$sh = $dbh->native_sql($statement)
```

<u>Параметр</u>	<u>Описание</u>
<code>\$statement</code>	Текст SQL-оператора с преобразованными ESC-последовательностями.

Возвращаемые значения

<u>Переменная</u>	<u>Описание</u>
<code>\$dbh</code> :	
<ul style="list-style-type: none">• описатель предложения• undef	Нормальное завершение Ошибка подготовки предложения
<code>\$DBI::errstr</code>	Диагностическое сообщение (в случае ошибки подготовки предложения)

Завершить обработку предложения

Назначение

Метод `Finish` позволяет закончить обработку предложения и освободить все ресурсы, выделенные методам `prepare`, `execute` и `fetch` для работы с предложением.

Пакет

Package Linter::st

Прототип

```
$sh->finish()
```


Возвращаемые значения

Переменная	Описание
\$ary:	
• 1	Нормальное завершение
• 0	Ошибка выполнения метода
\$DBI::errstr	Диагностическое сообщение (в случае ошибки)

Получить порцию BLOB-данных

Назначение

Метод `Blob_read` позволяет получить порцию BLOB-данных.

 Метод применяется к таблице с одним BLOB-столбцом и только после того, как предложение подготовлено, выполнено и выбрана хотя бы одна запись, содержащая поле типа BLOB.

Пакет

Package Linter::st

Прототип

```
$blob = $sh->
blob_read($field,$offset,$length[, \ $blobref])
```

Параметр	Описание
\$field	Номер выбираемого BLOB-столбца (отсчет начинается с 1)
\$offset	Смещение в байтах порции данных в BLOB-столбце
\$length	Длина порции данных (в байтах)
\$blobref	Ссылка на буфер, в который должна быть помещена порция данных

Возвращаемые значения


Переменная	Описание
\$blob :	

• порция данных	Нормальное завершение
• undef	Ошибка выполнения метода
<code>\$blobref</code>	Ссылка на буфер с порцией данных
<code>\$DBI::errstr</code>	Диагностическое сообщение (в случае ошибки)

Добавить порцию BLOB-данных

Назначение

Метод `Blob_write` позволяет добавить порцию BLOB-данных,

 Метод применяется только после того, как предложение подготовлено, выполнено и выбрана хотя бы одна запись, содержащая поле типа BLOB.

Пакет

Package Linter::st

Прототип

```
$rv = $sh->func($field,$length, \ $blob,'blob_write');
```

<u>Параметр</u>	<u>Описание</u>
<code>\$field</code>	Номер BLOB-столбца, к которому добавляются BLOB-данные (отсчет начинается с 1)
<code>\$length</code>	Длина добавляемой порции данных (в байтах)
<code>\$blob</code>	Ссылка на буфер, в котором содержится добавляемая порция данных

Возвращаемые значения

<u>Переменная</u>	<u>Описание</u>
<code>\$blob</code> :	
• 1	Нормальное завершение
• 0	Ошибка выполнения метода
<code>\$DBI::errstr</code>	Диагностическое сообщение (в случае ошибки)

Удалить BLOB-данные

Назначение

С помощью метода `Blob_delete` осуществляется удаление BLOB-данных.

 Метод применяется только после того, как предложение подготовлено, выполнено и выбрана хотя бы одна запись, содержащая поле типа BLOB.

Пакет

Package Linter::st

Прототип

```
$sh->func([$field],'blob_delete');
```

<u>Параметр</u>	<u>Описание</u>
<code>\$field</code>	Номер BLOB-столбца, из которого удаляются BLOB-данные (отсчет начинается с 1)

Возвращаемые значения

<u>Переменная</u>	<u>Описание</u>
<code>\$blob</code> :	
• 1	Нормальное завершение
• 0	Ошибка выполнения метода
<code>\$DBI::errstr</code>	Диагностическое сообщение (в случае ошибки)

Приложение 1

Типы данных СУБД ЛИНТЕР

Тип данных	Числовое значение	Описание
LDT_CHAR	1	Символьные строки фиксированной длины (максимальная длина 4000 байтов)
LDT_INTEGER	2	Целые числа (4 байта)
LDT_REAL	3	Вещественные числа (4 байта)
LDT_DATE	4	Дата-время (16 байтов)
LDT_NUMERIC	5	Числа с фиксированной точкой (16 байтов)
LDT_BYTE	6	Байтовые строки фиксированной длины (максимальная длина 4000 байтов)
LDT_BLOB	7	BLOB-данные (максимальная длина 2 Гбайта)
LDT_VARCHAR	8	Символьные строки переменной длины (максимальная длина 4000 байтов)
LDT_VARBYTE	9	Байтовые строки переменной длины (максимальная длина 4000 байтов)
LDT_BOOLEAN	10	Логическое значение (5 байтов) Из PERL-программы BOOLEAN-значение видно как целое число со значениями 0/1 (False/True).
LDT_NCHAR	11	UNICODE- строки фиксированной длины (максимальная длина 2000 UNICODE-символа)
LDT_NVARCHAR	12	UNICODE- строки переменной длины (максимальная длина 2000 UNICODE-символа)
LDT_EXTFILE	1	Внешний файл. Из PERL-программы тип данных EXTFILE виден как строка длиной до 512 байтов. Чтобы привязать переменную типа EXTFILE, надо использовать функцию EXTFILE(). например: <pre>\$dbh->do('create table extfile_test(i int, e extfile);'); \$sh=\$dbh->prepare('insert into extfile_test values(?,EXTFILE(?));');</pre>
LDT_BIGINT	2	Длинное целое (8 байтов)
LDT_SMALLINT	2	Короткое целое (2 байта)
LDT_DOUBLE	3	Вещественное двойной точности (8 байтов)

Типы данных BIGINT, INTEGER, SMALLINT имеют одинаковый числовой код. Фактический тип данных должен определяться с помощью длины возвращаемых данных.

Приложение 2

Пример обработки типов данных в DBI-интерфейсе

```
#!/usr/bin/perl
require DBI;
import DBI;

$t->{'t'} = "t";

$drh = DBI->install_driver('Linter');

print "Version = $drh->{Version}";

$database='';
$username='SYSTEM';
$auth='MANAGER';
$dbh = DBI->connect($database, $username, $auth, \%attr, 'Linter');

if(defined $dbh){
    $dbh->do('drop table bigint_test;');
    $dbh->do('create table bigint_test(b boolean, i int, j int, bi bigint, ch
varchar(15), ch1 char(20));');
    $dbh->do('insert into bigint_test values(true, 10, 10, 42949672950,
\'ddd\',\'\\\');');
    $dbh->do('insert into bigint_test values(false, 20, 10, 429496729500,
\'ddd\',\'\\\');');
    $dbh->do('insert into bigint_test values(true, 30, 10, -42949672950,
\'ddd\',\'\\\');');
    $dbh->do('insert into bigint_test values(false, 40, 10, -429496729500,
\'ddd\',\'\\\');');

    $sh=$dbh->prepare('insert into bigint_test values(?,?,?,?,?,?);');
    if(defined $sh){
        # @param=(100, 100, "-42949672950", "fff", "ff");
        @param=(1, 100, 100, -42949672950000, "fff", "ff");
        $sh->execute(@param);
        $sh->execute(@param);
        $sh->execute(@param);
    }

    $sh->finish;
    $sh=$dbh->prepare('select b,i,bi,j,ch,ch1 from bigint_test;');
    $t = $sh->{TYPE};

    print "\n$t->[0], $t->[1], $t->[2], $t->[3], $t->[4], $t->[5]\n";

    #die "\n";

    $rv = $sh->execute();
    while(1){
        @row=$sh->fetchrow_array;
        last unless defined $row[0];
        print "\n-> $row[0] $row[1] $row[2] $row[3] $row[4] $row[5]<-\n";
        print $row[1] /100, "\n";
    };

    }else{
        die("\nconnection error\n");
    }

    die "\n";
}
```

Приложение 3

Коды завершения LinPerl-интерфейса

<u>Имя константы</u>	<u>Значение</u>	<u>Описание</u>
LPERR_SUCCESS	0	Успешное завершение
LPERR_NOMEMORY	-1	Недостаточно памяти
LPERR_INVCURSOR	-2	Неверный идентификатор курсора/соединения
LPERR_INVCOLUMN	-3	Неверный номер столбца
LPERR_INVPARAMNUM	-4	Неверный номер параметра
LPERR_INVSTATE	-5	Неверное значение состояния курсора/соединения
LPERR_INVDIRECT	-6	Неверное направление перемещения по курсору
LPERR_UNKNOWNOPT	-7	Неизвестная опция курсора
LPERR_BADDTFORMAT	-8	Неверный формат данных типа «дата/время»
LPERR_NOBLOBCOL	-9	Отсутствие BLOB-столбца в курсоре
LPERR_NOPARENT	-10	Для курсора нет родительского соединения
LPERR_INVPARAMETER	-11	Функции было передано недопустимое число параметров
LPERR_LINERROR	-12	Внутренняя ошибка СУБД ЛИНТЕР
LPERR_NOTIMPLEMENTED	-999	Свойство не реализовано

Приложение 4

Коды завершения DBI-интерфейса

<u>Имя константы</u>	<u>Значение</u>	<u>Описание</u>
NOERR	0	Успешное завершение
NOTFOUND	100	Нет данных
ERR_NOCTH	-3050	Неверный номер контекста
ERR_NODBH	-3051	Неверный номер соединения
ERR_NOSH	-3052	Неверный номер предложения
ERR_NOTCURSOR	-3053	Канал не курсор
ERR_NOTPREPARED	-3054	Предложение не подготовлено
ERR_INVALIDX	-3055	Неверный номер привязываемого параметра
ERR_NOREF	-3056	Параметр не определен
ERR_INVTYP	-3057	Неверный тип параметра
ERR_INVVAL	-3058	Неверное значение параметра
ERR_EXECPROC	-3059	Хранимая процедура не может исполняться посредством метода <code>execute</code> , <code>do</code>
ERR_NOTMAINCH	-3060	Канал – курсор
ERR_CORRUPT	-3061	Испорчено содержимое внутренних структур
ERR_NOMEM	-3062	Нет памяти
ERR_SPNOTSUPP	-3063	ЛИНТЕР 4.xx не поддерживает вызовы хранимых процедур
ERR_NOCOMPAT	-3064	Несовместимые версии библиотеки и сервера
ERR_NONSELECT	-3065	Попытка выполнить <code>fetch</code> для не-SELECT запроса

Указатель функций

BindColumn, 15
BindParameter, 13
BindParameterArray, 13
BLOBAdd, 21
BLOBAddEx, 22
BLOBAppend, 20
BLOBAppendEx, 21
BLOBClear, 24
BLOBFetch, 26
BLOBGetData, 25
BLOBGetSize, 25
BLOBGetType, 23
BLOBPurge, 24
CloseConnect, 8
CloseCursor, 11
Commit, 27
ExecDirect, 12
Execute, 14
Fetch, 15
GetColumnInfo, 28
GetConnectInfo, 5
GetCursorOption, 9
GetDataArray, 19
GetDataColumn, 17
GetDataRow, 17
GetDataRowS, 18
GetError, 28
GetErrorMsg, 29
GetM, 18
OpenConnect, 4
OpenCursor, 8
Prepare, 12
Rollback, 27
SetCodepage, 8
SetCursorOption, 11
TestNull, 19
UnbindColumn, 16

Указатель методов

Bind_param, 38
Blob_delete, 43
Blob_read, 42
Blob_write, 42
Commit, 34
Connect, 31
Disconnect, 34
Do, 35
Driver, 31
Err, 32
Execute, 38
Fetchrow_array, 40, 41
Fetchrow_arrayref, 39
Finish, 41
Func, 32
Linter, 33
Prepare, 33
Proc, 37
Rollback, 35
Rows, 39
Shut, 36
Snap, 36

