

МОБИЛЬНАЯ
РЕЛЯЦИОННАЯ
СУБД

ЛИНТЕР[®]

Linter Standard
Linter Bastion
Linter RealTime
Linter Multiversion

PYTHON-интерфейс

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
Назначение документа	1
Для кого предназначен документ	1
Необходимые предварительные знания	1
Принятые обозначения и соглашения	1
Дополнительные документы	2
Общие сведения	3
Функции интерфейса	4
Соединение с БД	4
Типизация данных	4
Создание типа данных «дата»	5
Создание типа данных «время»	5
Создание типа данных «временная метка»	6
Создание типа данных «дата» из тиков	6
Создание типа данных «время» из тиков	6
Создание типа данных «временная метка» из тиков	7
Создание типа данных «байтовая строка»	7
Глобальные переменные	8
Исключения	9
Функции и методы класса Connection	11
Создание курсора	11
Подтверждение транзакции	11
Отмена транзакции	11
Закрытие соединения	12
Функции и методы класса Cursor	13
Выполнение запроса	13
Пакетная обработка данных	14
Выборка всех строк	15
Выборка следующей строки	15
Выборка следующей порции строк	16
Переход к следующему подмножеству выборки	16
Трансляция запроса	17
Исполнение претранслированного запроса	17
Вызов хранимой процедуры	18

Оглавление

Выделение памяти для параметров запроса	18
Выделение памяти для BLOB-значений	19
Подтверждение курсорной транзакции	19
Отмена курсорной транзакции	19
Закрытие курсора	20
Атрибуты курсора	21
Работа с BLOB-значениями	22
Добавление BLOB-значения	22
Чтение BLOB-значения	22
Очистка BLOB-значения	23
Атрибуты BLOB-значения	23

Предисловие

Назначение документа

Документ содержит описание прикладного (API) интерфейса между СУБД ЛИНТЕР и PYTHON-программой.

Для кого предназначен документ

Документ предназначен для программистов, разрабатывающих приложения для работы с СУБД ЛИНТЕР на языке программирования PYTHON.

Необходимые предварительные знания



Для работы необходимо знать;

- знать основы реляционных баз данных и языка баз данных SQL;
- язык программирования PYTHON;
- 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
Заглавные буквы	BROWSE	Названия команд, слова, зарезервированные в SQL, ключевые слова

Функции

Обозначение	Пример	Значение
Курсив в < >	<return statement>	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	<return value> ::= <value expression> 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. Справочник по SQL.
- СУБД ЛИНТЕР. Справочник кодов завершения.

Общие сведения

Прикладной интерфейс (API) между СУБД ЛИНТЕР и PYTHON-программой обеспечивается PYTHON-драйвером, который представляет собой динамически подгружаемую библиотеку, написанную полностью на языке программирования Си.

Драйвер основывается на спецификации Python Database API Specification v2.0 (<http://www.python.org/peps/pep-0249.html>).

Драйвер экспортирует описанные ниже функции и свойства, доступные из PYTHON-программы.

Для загрузки драйвера в PYTHON-программу необходимо выполнить команду:

```
import LinPy
```

(при этом файл `LinPy.so` должен находиться в `/usr/local/lib/python2.2` или в пути переменной среды `PYTHONPATH`).

Функции интерфейса

Соединение с БД

Назначение

Создание объекта типа Connection и соединение с БД.

Синтаксические правила

connect

```
(user = <пользователь>, password = <пароль> [, database = <сервер>][, mode = <режим>])
```

<пользователь> – имя пользователя БД;

<пароль> – пароль пользователя БД;


<сервер> – имя ЛИНТЕР-сервера (узла локальной сети, на котором находится БД). Если аргумент не задан или имеет пустое значение, то соединение осуществляется с локальной БД;

<режим> – уровень изоляции транзакций по данному соединению:

- M_EXCLUSIVE
- M_OPTIMISTIC
- M_AUTOCOMMIT

Возвращаемое значение

Идентификатор соединения с БД (в случае успешного соединения).

 В случае ошибки возникнет исключение, функция не завершится и код возврата не сформирует. Это поведение относится ко всем функциям PYTHON-интерфейса.

Примеры


```
# подсоединение пользователя SYSTEM с паролем MANAGER к локальной БД  
connection = LinPy.connect('SYSTEM', 'MANAGER')
```

```
# подсоединение пользователя BORIS с паролем 123 к БД UNCLE  
# в режиме автофиксации изменений  
connection = LinPy.connect(  
user = 'BORIS',  
password = '123',  
database = 'UNCLE',  
mode = LinPy.M_AUTOCOMMIT)
```

Типизация данных

Функции типизации данных предназначены для создания в PYTHON-программе типов данных, используемых в СУБД ЛИНТЕР, но не поддерживаемых языком программирования PYTHON. Например, если данные предназначены для столбца типа DATE, они должны быть подготовлены для БД в формате «дата-время» СУБД ЛИНТЕР, поскольку параметры метода executeXXX() не типизированы. Когда ядро СУБД получает строковый объект PYTHON, оно не знает, нужно ли его привязывать как простой символьный столбец (CHAR) или как дату (DATE).

Для преодоления этой проблемы PYTHON-интерфейс обеспечивает набор функций для создания объектов, специфичных для БД ЛИНТЕР.

 Атрибут объекта курсора `description` возвращает информацию о каждом из столбцов результата запроса. Код типа `type_code` должен быть проверен на **равенство** одному из объектов типа, описанных ниже. Объекты типа могут быть равны более чем одному коду типа (например, `DATETIME` может быть равен кодам типа для столбцов даты, времени и временной метки).

Создание типа данных «дата»

Назначение

Создание объекта, содержащего значение «дата».

Синтаксические правила

`Date(<год>, <месяц>, <день>)`

<год> – целочисленное положительное значение в диапазоне 1-9999;

<месяц> – целочисленное положительное значение в диапазоне 1-12;

<день> – целочисленное положительное значение в диапазоне 1-31.

Возвращаемое значение

Объект типа «дата» (в случае успешного создания).

Пример

```
# первое января 2003 года
d = Date(2003, 1, 1)
```

Создание типа данных «время»

Назначение

Создание объекта, содержащего значение «время».

Синтаксические правила

`Time(<час>, <минута>, <секунда>[, <тик>])`

<час> – целочисленное положительное значение в диапазоне 0-23;

<минута> – целочисленное положительное значение в диапазоне 1-60;

<секунда> – целочисленное положительное значение в диапазоне 1-60;

<тик> – целочисленное положительное значение в диапазоне 1-100.

Возвращаемое значение

Объект типа «время» (в случае успешного создания).

Пример

```
# полдень
t = Time(12, 0, 0)
```

Создание типа данных «временная метка»

Назначение

Создание объекта, содержащего значение «временная метка» (timestamp).

Синтаксические правила

Timestamp(<год>, <месяц>, <день>, <час>, <минута>, <секунда>[, доли секунды])

<год> – целочисленное положительное значение в диапазоне 1-9999;

<месяц> – целочисленное положительное значение в диапазоне 1-12;

<день> – целочисленное положительное значение в диапазоне 1-31.

<час> – целочисленное положительное значение в диапазоне 0-23;

<минута> – целочисленное положительное значение в диапазоне 1-60;

<секунда> – целочисленное положительное значение в диапазоне 1-60;

<доли секунды> – целочисленное положительное значение в диапазоне 1-100.

Возвращаемое значение

Объект типа «временная метка» (в случае успешного создания).

Пример

```
# 5-е апреля 2003 года 6 часов 7 минут 8 секунд и 9 сотых секунды
ts = Timestamp(2003, 4, 5, 6, 7, 8, 9)
```

Создание типа данных «дата» из тиков

Назначение

Создание объекта, содержащего значение «дата», из заданного количества тиков.

Синтаксические правила

DateFromTicks(<тик>)

<тик> – целочисленное положительное значение, возвращаемое функцией `time()`.

 См. также документацию по стандартному модулю `time` языка PYTHON.

Возвращаемое значение

Объект типа «дата» (в случае успешного создания).

Пример

```
# текущая дата
d = DateFromTicks(time())
```

Создание типа данных «время» из тиков

Назначение

Создание объекта, содержащего значение «время», из заданного количества тиков.

Синтаксические правила

`TimeFromTicks(<тик>)`

`<тик>` – целочисленное положительное значение, возвращаемое функцией `time()`.



См. также документацию по стандартному модулю `time` языка PYTHON.

Возвращаемое значение

Объект типа «время» (в случае успешного создания).

Пример

```
# текущее время
t = TimeFromTicks(time())
```

Создание типа данных «временная метка» из тиков

Назначение

Создание объекта, содержащего значение «временная метка», из заданного количества тиков.

Синтаксические правила

`TimeStampFromTicks(<тик>)`

`<тик>` – целочисленное положительное значение, возвращаемое функцией `time()`.



См. также документацию по стандартному модулю `time` языка PYTHON.

Возвращаемое значение

Объект типа «временная метка» (в случае успешного создания).

Пример

```
# текущие время и дата
ts = TimeStampFromTicks(time())
```

Создание типа данных «байтовая строка»

Назначение

Создание объекта, содержащего значение «байтовая строка». Используется для работы с данными типа `BYTE`, `VARBYTE` СУБД ЛИНТЕР.

Синтаксические правила

`Binary(<строка>)`

`<строка>` – символьная строка шестнадцатеричных цифр.

Возвращаемое значение

Объект типа `BINARY` (в случае успешного создания).

Пример

```
# байтовая строка
bt = Binary('0967fca8dd')
```

Глобальные переменные

PYTHON-программе доступны следующие predefined глобальные переменные:


apiLevel

Строковая константа, содержащая поддерживаемый уровень DB API. В настоящее время содержит строку '2.0'.

threadsafety

Целочисленная константа, обозначающая уровень нитебезопасности, поддерживаемый данным интерфейсом. Возможные значения:


- 0 – модуль не может разделяться нитями;
- 1 – нити могут разделять модуль, но не соединения;
- 2 – нити могут разделять и модуль, и соединения;
- 3 – нити могут разделять модуль, соединения и курсы.

 Разделение в вышеуказанном контексте означает, что две нити могут использовать ресурс, не используя внешний объект синхронизации (mutex) для блокировки. Обеспечить нитебезопасность для внешних ресурсов, управляя доступом только с помощью семафора, не всегда возможно: ресурс может использовать глобальные переменные или другие внешние ресурсы, находящиеся вне вашего контроля.

paramstyle

Строковая константа, обозначающая тип форматирования маркера параметра, ожидаемый интерфейсом. Возможные значения по спецификации DB API 2.0:

Значение	Описание
'qmark'	Стиль вопроса, например: '...WHERE name=?'
'numeric'	Числовой (позиционный) маркер, например: '...WHERE name=:1'
'named'	Именованный маркер, например: '...WHERE name=:name'
'format'	Коды формата ANSI C для функции printf языка Си, например: '...WHERE name=%s'
'pyformat'	Расширенные коды формата языка PYTHON, например: '...WHERE name=%(name)s'

 В СУБД ЛИНТЕР определять параметры можно как с помощью имени (именованные параметры), так и с помощью символа вопроса (?) (позиционные параметры). Предпочтительным (по соображениям быстродействия) является второй способ.

version

Строковая константа, обозначающая версию PYTHON-драйвера.

Исключения

В процессе выполнения PYTHON-программа выдает информационные и диагностические сообщения с помощью следующих исключений:

Warning

Исключение, вызываемое для важных предупреждений, таких, как усечение данных в процессе вставки и т.д. Этот класс является производным от класса PYTHON StandardError (определенного в модуле exceptions).

Error

Исключение, являющееся базовым классом всех других исключений, связанных с ошибками. Его можно использовать для перехвата всех исключений с помощью единственного утверждения 'except'. Предупреждения не рассматриваются как ошибки и поэтому не должны использовать этот класс в качестве базового. Этот класс является производным от класса PYTHON StandardError (определенного в модуле exceptions).

InterfaceError

Исключение порождается ошибками в PYTHON-интерфейсе. Является производным от класса Error.

DatabaseError

Исключение порождается ошибками обработки данных в БД. Оно является подклассом класса Error.

DataError

Исключение порождается при обработке некорректных данных в БД (деление на ноль, выход величины за пределы допустимых значений и т.д.) Является производным от класса DatabaseError.

OperationalError

Порождение исключения вызывается внешними причинами (неожиданный обрыв соединения, не найдено имя источника данных, невозможно обработать транзакцию, ошибка распределения памяти в процессе обработки и т.д.) Является производным от класса DatabaseError.

IntegrityError

Исключение порождается при нарушении ссылочной целостности БД (например, попытка ссылки на несуществующую внешнюю запись). Является производным от класса DatabaseError.

InternalError

Исключение порождается внутренними ошибками СУБД (например, курсор больше не является допустимым, транзакция не синхронизирована и т.д.) Является производным от класса DatabaseError.

ProgrammingError

Исключение порождается программными ошибками (например, таблица не найдена или уже существует, синтаксическая ошибка в SQL-предложении, задано неверное число параметров и т.д.) Является производным от класса DatabaseError.


NotSupportedError

Исключение порождается в случае использования вызова API. Является производным от класса DatabaseError. Схема наследования исключений показана на Рис. 1.

StandardError

```
StandardError
├── Warning
├── Error
│   ├── InterfaceError
│   ├── DatabaseError
│   │   ├── DataError
│   │   ├── OperationalError
│   │   ├── IntegrityError
│   │   ├── InternalError
│   │   ├── ProgrammingError
│   │   └── NotSupportedError
```

Рис. 1. Схема наследования исключений

 Все исключения, возникающие в PYTHON-драйвере, содержат коды завершения и их текстовую расшифровку. Для получения дополнительных сведений об исключении можно воспользоваться следующими расширенными свойствами, не входящими в спецификацию DBAPI 2.0:

- code код завершения СУБД ЛИНТЕР
- message текстовая расшифровка кода завершения
- linCode синоним свойства code
- apiCode код ошибки прикладного интерфейса
- sysCode код системной ошибки
- object объект, вызвавший исключение

Функции и методы класса Connection

Создание курсора

Назначение

Создание нового объекта курсора для соединения.

Синтаксические правила

`cursor([<режим>][mode = <режим>])`

<режим> – уровень изоляции транзакций для курсора:

- M_EXCLUSIVE;
- M_OPTIMISTIC;
- M_AUTOCOMMIT.

Если не задан, по умолчанию используется режим соединения.

Возвращаемое значение

Объект типа `cursor` (в случае успешного создания).

Пример

```
cursor = connection.cursor()
```

Подтверждение транзакции

Назначение

Подтверждение изменений, внесенных в БД по соединению и подчиненным курсорам.

Синтаксические правила

`commit()`

Описание

По умолчанию установлен режим M_EXCLUSIVE. Для автофиксации изменений в БД необходимо указать режим M_AUTOCOMMIT в методе `connect()`.

Возвращаемое значение

Нет.

Отмена транзакции

Назначение

Откат изменений, внесенных в БД по соединению и подчиненным курсорам.

Синтаксические правила

`rollback()`

Описание

Метод заставляет СУБД выполнить откат к началу все незавершенные транзакции по этому соединению. Закрытие соединения без предварительной явной фиксации изменений приведет к неявному откату.

Возвращаемое значение

Нет.

Закрытие соединения

Назначение

Закрытие соединения.

Синтаксические правила

`close()`

Описание

Функция выполняет немедленное закрытие соединения (а не отложенное, как при вызове при вызове `__del__`). Соединение становится недоступным для последующего использования; При попытке выполнить по нему некоторую операцию (в том числе курсорную) порождается соответствующее исключение.

Возвращаемое значение

Нет.

Функции и методы класса `Cursor`

Объекты класса `Cursor` представляют курсор СУБД, используемый для управления контекстом операции выборки.

Выполнение запроса

Назначение

Подготовка и выполнение SQL-запроса.

Синтаксические правила

`execute(<операция>[, <спецификация значений>])`

`<спецификация значений> ::=`
 `<спецификация именованных параметров>`
 `| <спецификация словарных параметров>`
 `| <спецификация неименованных параметров>`

`<спецификация именованных параметров> ::=`
 `(<имя>=<значение>[, <имя>=<значение>...])`

`<спецификация словарных параметров> ::=`
 `<словарная переменная>`
 `| {<имя>=<значение>[, <имя>=<значение>...]}`

`<спецификация неименованных параметров> ::=`
 `(<значение>[, <значение>...])`

`<операция>` – текст SQL-запроса к СУБД (синтаксис SQL-запроса с параметрами описан в документе «СУБД ЛИНТЕР. Справочник по SQL». См. также атрибут модуля `paramstyle`).

Описание

Ссылка на SQL-запрос сохраняется курсором. Если объект операции SQL-запроса передается снова на обработку, курсор может оптимизировать свое поведение (не выполнять повторную трансляцию SQL-запроса, а использовать его претранслированный вариант). Это наиболее эффективно для алгоритмов, в которых много раз используются одна и та же операция с привязкой различных параметров.

Привязка значений к именованным параметрам выполняется по именам параметров, например:

```
cursor.execute("insert into PEOPLE (ID, NAME, BDAY) values (:I, :N, :D)",  
I = 1, N = "BORIS", D = Date(1980, 5, 27))
```

В результате будет добавлена строка со значением полей:

```
ID=1, NAME="BORIS", BDAY= Date(1980, 5, 27)
```

Привязка значений к неименованным параметрам выполняется в порядке следования значений, например:

```
cursor.execute("insert into PEOPLE (ID, NAME) values (?, ?)",  
(1, "BORIS"))
```

В результате будет добавлена строка со значением полей:

Функции и методы класса `Cursor`

`ID=1, NAME="BORIS", BDAY=` значение по умолчанию (если определено или `NULL`).

Привязка значений к словарным параметрам выполняется в соответствии со словарными именами параметров, например;

```
cursor.execute("insert into PEOPLE (ID, NAME, BDAY) values (:I, :N, :D)",
               {"I" : 1, "N" : "BORIS", "D" : Date(1980, 5, 27)})
```

В результате будет добавлена строка со значением полей:

```
ID=1, NAME="BORIS", BDAY= Date(1980, 5, 27)
```

Возвращаемое значение

Нет.

Пример

```
# запрос без параметров
cursor.execute("insert into PEOPLE (ID, NAME, BDAY) values (1, 'BORIS', "\
'1980.05.27')")
```

Пакетная обработка данных

Назначение

Подготовка и выполнение SQL-запроса в режиме пакетной обработки.

Синтаксические правила

`executemany` (<операция>[,<спецификация значений> [,<спецификация значений>]]...

<операция> – текст SQL-запроса к СУБД (синтаксис SQL-запроса с параметрами описан в документе «СУБД ЛИНТЕР. Справочник по SQL». См. также атрибут модуля `paramstyle`).

<спецификация значений> – см. описание функции `execute()`.

Описание

Функция позволяет за один вызов подготовить SQL-запрос и затем выполнить его для всего пакета данных.

Пакет данных представляет собой массив, каждый элемент которого содержит набор значений параметров исполняемого SQL-запроса.

Возвращаемое значение

Нет.

Примеры

```
cursor.executemany("insert into PEOPLE (ID, NAME) values (:I, :N)",
                  [{"I" : 1, "N" : "BORIS"},
                   {"I" : 2, "N" : "SASHA"}])
```

```
cursor.executemany("insert into PEOPLE (ID, NAME) values (?, ?)",
                  [(1, "BORIS"),
                   (2, "SASHA")])
```

Выборка всех строк

Назначение

Получение всех строк ответа запроса выборки.

Синтаксические правила

`fetchall()`

Описание

Атрибут курсора `arraysize` влияет на производительность этой операции.

Исключение `Error` (или его подкласс) порождается, если результатом выборки является пустое множество строк или запрос выборки вообще не подавался.

Возвращаемое значение

Нет.

Пример

```
cursor.execute('select * from auto fetch first 5 order by ')
cursor.fetchall()
```

Выборка следующей строки

Назначение

Получение следующей строки ответа запроса выборки.

Синтаксические правила

`fetchone()`

Описание

Исключение `Error` (или его подкласс) порождается, если результатом выборки является пустое множество строк или запрос выборки вообще не подавался.

Возвращаемое значение

- 1) Следующая строка ответа запроса выборки. Если после выполнения курсорного запроса строки не выбирались, выдается первая строка ответа;
- 2) `None` (строк ответа больше нет)

Пример

```
cursor.execute('select distinct make from auto ')
cursor.fetchone()
| ALPINE      |
cursor.fetchone()
| AMERICAN MOTORS |
cursor.fetchone()
| BMW         |
```

Выборка следующей порции строк

Назначение

Получение следующей порции строк ответа запроса выборки.

Синтаксические правила

`fetchmany(<количество>)`

`<количество>` – количество строк для выборки за один вызов. Если аргумент не задан, число строк для выборки определяется атрибутом курсора `arraysize`.

Описание

Функция предоставляет следующую порцию строк ответа запроса выборки.

Если количество запрошенных строк превышает количество выбранных (при первом вызове функции) или оставшихся (после предыдущих вызовов функции), может быть возвращено меньшее число строк.

Исключение `Empty` (или его подкласс) порождается, если результатом выборки является пустое множество строк или запрос выборки вообще не подавался.

Для достижения оптимальной производительности в большинстве случаев лучше не указывать размер порции, а пользоваться атрибутом `arraysize`. Если размер порции указывается, то желательно, чтобы он сохранял одну и ту же величину при всех вызовах функции `fetchmany()`.

Возвращаемое значение

Пустая последовательность, если больше нет никаких строк.

Пример

```
cursor.execute('select distinct make from auto');
cursor.fetchmany(3)
|ALPINE          |
|AMERICAN MOTORS|
|BMW            |

cursor.fetchmany(2)
|CHRYSLER       |
|CITROEN        |
```

Переход к следующему подмножеству выборки

Назначение

Перемещение курсора к следующему подмножеству пакетного запроса выборки.

Синтаксические правила

`nextset()`

Описание

Метод заставляет курсор перейти на следующее имеющееся подмножество пакетного запроса выборки, игнорируя все оставшиеся строки текущего подмножества.

Исключение порождается, если предыдущий вызов `executeXXX()` вернул пустую выборку, либо выборка данных вообще не выполнялась.

 В данной версии метод не реализован, т.к. СУБД не поддерживает возможности генерации нескольких выборок по одному курсору.

Возвращаемое значение

- 1) TRUE – курсор перемещен к следующему подмножеству;
- 2) None – список подмножеств выборки исчерпан.

Если функция возвращает TRUE, то последующие обращения к методам выборки будут возвращать строки из следующего подмножества выборки.

Пример

```
cursor.execute("select distinct model from auto where color=? and make=?
fetch first 3",
[("BLACK", 'ALPINE' ),,
 ("BLUE", 'BMW'),
 ("WHITE", '')])
```

```
|A-310          |
Выдано строк   : 1
```

```
|3.0 CSI        |
Выдано строк   : 1
```

```
|124 SPORT COUPE |
|1275 GT         |
|1302 S         |
Выдано строк   : 3
```

```
cursor.execute("select distinct model from auto where color=? and make=?
fetch first 3",
[("BLACK", 'ALPINE' ),,
 ("BLUE", 'BMW'),
 ("WHITE", '')])
cursor.nextset()
```

```
|A-310          |
```

```
cursor.nextset()
```

```
|3.0 CSI        |
```

```
cursor.nextset()
```

```
|124 SPORT COUPE |
```

```
|1275 GT         |
```

```
|1302 S         |
```

Трансляция запроса

Назначение

Трансляция SQL-запроса.

Синтаксические правила

prepare(<запрос>)

<запрос> – текст SQL-запроса (возможно, с параметрами).

Исполнение претранслированного запроса

Назначение

Выполнение претранслированного ранее SQL-запроса.

Синтаксические правила

`executemanyprepared`([<спецификация значений> [, <спецификация значений>]]...)

<спецификация значений> – см. описание функции `executemany()`.

Описание

Выполняется привязка переданных значений к последнему претранслированному по соединению SQL-запросу и последующее его выполнение.

Вызов хранимой процедуры

Назначение

Вызов хранимой процедуры БД

Синтаксические правила

`callproc`(<имя процедуры>[, <список параметров>])

Описание

<список параметров> должен содержать значение для каждого параметра.

Возвращаемое значение

Возвращается модифицированная копия переданных параметров. Input-параметры остаются нетронутыми, Output и input/output параметры могут быть заменены новыми значениями. Процедура может вернуть выборку, доступ которой возможен через стандартные методы `fetchXXX()`.

Пример

```
result = cursor.callproc("myproc", ("hi", 5, var))
```

Выделение памяти для параметров запроса

Назначение

Предварительное выделение памяти для хранения значений параметров SQL-запросов.

Синтаксические правила

`setinputsizes`(<размеры>)

<размеры> ::= <тип параметра> | <значение> | **none**

<тип параметра> – спецификация типа параметра, для которого запрашивается выделение памяти (например, `IntType`);

<значение> – максимальная длина параметра строкового типа (в символах)

Описание

Если в спецификации размера параметра указано none, предварительное выделение памяти под значение параметра выполняться не будет (это бывает полезным при работе с большими объемами данных).

Функция может использоваться перед вызовом метода `execute()`.

Возвращаемое значение

Нет.

Выделение памяти для BLOB-значений

Назначение

Определение размера буфера для загрузки BLOB-столбцов.

Синтаксические правила

`setoutputsize(<размер>[, <столбец>)`

<размер> – размер буфера для хранения значений BLOB_столбца (в байтах)

<столбец> – номер BLOB-столбца, для которого выделяется память

Описание

Номер столбца задается как индекс в запросе выборки. Если столбец не задан, устанавливается размер по умолчанию для всех BLOB-столбцов в курсоре.

Метод может использоваться перед вызовом метода `executeXXX()`.

Возвращаемое значение

Нет.

Подтверждение курсорной транзакции

Назначение

Подтверждение транзакции по курсору.

Синтаксические правила

`commit()`

Описание

Метод аналогичен методу `commit()` для соединения, но производит фиксацию изменений в БД только по курсору.

Отмена курсорной транзакции

Назначение

Отмена транзакции по курсору.

Синтаксические правила

`rollback()`

Описание

Метод аналогичен методу `rollback()` для соединения, но производит фиксацию изменений в БД только по курсору.

Закрытие курсора

Назначение

Закрытие курсора.

Синтаксические правила

`close()`

Описание

Функция выполняет немедленное закрытие курсора (а не отложенное, как при вызове `__del__`). С момента вызова и далее курсор становится неиспользуемым; в случае попытки обратиться к некоторой операции курсора порождается соответствующее исключение.

Атрибуты курсора

В таблице приведены доступные атрибуты курсора.

Таблица. Атрибуты курсора

Атрибут	Тип	Назначение
<code>description</code>	read-only	<p>Содержит описание столбцов запроса выборки в виде последовательности: (<описание столбца1>) (<описание столбца2>)...</p> <p>Описание столбцов имеет вид:</p> <ul style="list-style-type: none">• <code>name</code> – имя столбца;• <code>type_code</code> – тип данных столбца (анализируется путем сравнения с объектами типа);• <code>display_size</code> – длина отображаемого значения (для строковых значений);• <code>internal_size</code> – длина хранимого в БД значения (для строковых значений);• <code>precision</code> – точность (для вещественных значений);• <code>scale</code> – масштаб (для вещественных значений);• <code>null_ok</code> – допустимость NULL-значений (1 – да, 0 – нет ?).
<code>rowcount</code>	read-only	<p>Характеризует число обработанных строк:</p> <ul style="list-style-type: none">• выбранных (в SQL-запросах типа SELECT);• модифицированных (в SQL-запросах типа UPDATE, INSERT) последним вызовом <code>executeXXX()</code>. <p>Атрибут принимает значение -1 в случае, если над курсором не выполнен ни один вызов <code>executeXXX()</code> или если PYTHON-интерфейс не может определить количество обработанных строк в последнем выполненном SQL-запросе.</p>
<code>arraysize</code>	read/write	Задаёт число строк, выбираемых с помощью метода <code>fetchmany()</code> за один вызов
<code>mode</code>	read-only	Задаёт уровень изоляции курсорных транзакции
<code>name</code>	read/write	Задаёт имя курсора для операций с конструкцией WHERE CURRENT OF

Работа с BLOB-значениями

Если в запросе выборки присутствуют BLOB-столбцы, то значение курсорного атрибута `arraysize` игнорируется – выборка всегда производится по одной записи.

Добавление BLOB-значения

Назначение

Добавление BLOB-значения.

Синтаксические правила

`write(<объект> [, <тип>])`

<объект> – объект типа `buffer`, `string`, `unicode`, `array`;

<тип> – целочисленное значение.

Описание

Добавляемое BLOB-значение помещается в конец столбца, к которому относится объект этого метода

Аргумент <тип> задает тип добавляемых данных (текст, графика, анимация и т. п.). Тип добавляемых данных не контролируется СУБД.

Возвращаемое значение

Пример

```
cur.execute("select I, B from TEST where I = 10")
(i, blob) = cur.fetchone()
blob.clear()
blob.write('Hello World')
```

Чтение BLOB-значения

Назначение

Чтение BLOB-значения.

Синтаксические правила

`read([<смещение> [, <длина>]])`

<смещение> – целочисленное значение;

<длина> – целочисленное значение.

Описание

Аргумент <смещение> задает смещение считываемой порции BLOB-данных (отсчет начинается с 1); если не задан – выборка начинается с начала BLOB-данных.

Аргумент <длина> задает размер (в байтах) считываемой порции BLOB. Если не задан, чтение производится до конца BLOB-значения.

Если заданная длина порции больше длины всего BLOB-значения, то выдается BLOB-значение с заданного смещения и до конца.

Возвращаемое значение

Порция BLOB-данных.

Пример

```
cur.execute("select * from TEST")
print "description: %s" % str(cur.description)
result = cur.fetchone()
print "result: %s" % str(result)
result = cur.fetchone()
print "result: %s" % str(result)
(i, blob) = result
print "blob length:", blob.length
print "blob type:", blob.type
buf = blob.read(1, blob.length)
```

Очистка BLOB-значения

Назначение

Очистка BLOB-значения.

Синтаксические правила

`clear()`

Возвращаемое значение

Нет.

Пример

```
cur.execute("select I, B from TEST where I = 10")
(i, blob) = cur.fetchone()
blob.clear()
```

Атрибуты BLOB-значения

Для BLOB-столбцов доступны следующие атрибуты для чтения:

- 1) `length` – текущая длина BLOB- значения;
- 2) `type` - тип BLOB-значения.

Пример

```
cur.execute("select * from TEST")
print "description: %s" % str(cur.description)
result = cur.fetchone()
print "result: %s" % str(result)
result = cur.fetchone()
print "result: %s" % str(result)
(i, blob) = result
print "blob length:", blob.length
print "blob type:", blob.type
```