

虚谷数据库

PHP 标准接口 V11.2.1

开发指南

文档版本 01

发布日期 2024-03-15



版权所有 © 2024 成都虚谷伟业科技有限公司。

声明

未经本公司正式书面许可，任何企业和个人不得擅自摘抄、复制、使用本文档中的部分或全部内容，且不得以任何形式进行传播。否则，本公司将保留追究其法律责任的权利。

用户承诺在使用本文档时遵守所有适用的法律法规，并保证不以任何方式从事非法活动。不得利用本文档内容进行任何侵犯他人权益的行为。

商标声明



为成都虚谷伟业科技有限公司的注册商标。

本文档提及的其他商标或注册商标均非本公司所有。

注意事项

您购买的产品或服务应受本公司商业合同和条款的约束，本文档中描述的部分产品或服务可能不在您的购买或使用范围之内。由于产品版本升级或其他原因，本文档内容将不定期进行更新。

除非合同另有约定，本文档仅作为使用指导，所有内容均不构成任何声明或保证。

成都虚谷伟业科技有限公司

地址：四川省成都市锦江区锦盛路 138 号佳霖科创大厦 5 楼 3-14 号

邮编：610023

网址：www.xugudb.com

前言

概述

本文档介绍虚谷数据库使用 PHP 标准接口进行应用开发的相关指导。

读者对象

本文档主要适用于以下用户：

- 数据库开发人员
- 数据库维护人员
- 数据库管理员

符号约定

在本文中可能出现下列标志，它们所代表的含义如下。

符号	说明
 注意	用于传递设备或环境安全警示信息，若不避免，可能会导致设备损坏、数据丢失、设备性能降低或其它不可预知的结果。
 说明	对正文中重点信息的补充说明。“说明”不是安全警示信息，不涉及人身、设备及环境伤害信息。

修改记录

文档版本	发布日期	修改说明
01	2024-03-15	第一次正式发布

目录

1	PHP 概述	1
2	快速入手	2
2.1	开发环境搭建	2
2.1.1	Windows x86_64	2
2.1.2	Linux x86_64	2
2.2	PHP 版本要求	3
3	函数接口	5
3.1	PDO Class	5
3.1.1	_construct	5
3.1.2	exec	5
3.1.3	lastInsertId	6
3.1.4	prepare	7
3.1.5	query	7
3.1.6	beginTransaction	8
3.1.7	rollback	8
3.1.8	commit	9
3.1.9	quote	9
3.1.10	errorCode	10
3.1.11	errorInfo	10
3.1.12	getAvailableDrivers	11
3.2	PDOStatement Class	11
3.2.1	bindColumn	11
3.2.2	bindParam	12
3.2.3	bindValue	13
3.2.4	execute	14
3.2.5	fetch	15
3.2.6	fetchAll	16

3.2.7	fetchColumn	17
3.2.8	fetchObject	18
3.2.9	rowCount	19
3.2.10	closeCursor	19
3.2.11	columnCount	20
3.2.12	debugDumpParams	20
3.2.13	errorCode	21
3.2.14	errorInfo	21
4	存储过程与函数	23
4.1	输入型参数的存储过程	23
4.2	输出型参数的存储过程	23
4.3	存储函数	24
5	LOB 大对象	25
5.1	插入 BLOB 大对象数据	25
5.2	获取 BLOB 大对象数据	25
6	错误码	26

1 PHP 概述

PHP（PHP: Hypertext Preprocessor）是一种流行的通用开源脚本语言，特别适合于 Web 开发。PHP 作为一种语言程序，在数据处理上具有较高的处理和输出水平，可以广泛应用于 Windows 系统及各类 Web 服务器中。

PDO xugusql 是基于 PDO（PHP Data Objects）扩展进行设计开发的一套适用于 PHP 程序语言访问和操作虚谷数据库的接口扩展。该扩展模块完全依赖于 PDO 扩展进行开发设计，支持 PDO 的接口标准，对外保持了 PDO 原生特性，具备编码一致性、使用灵活性和面向对象思想等优点。

PDO xugusql 驱动扩展模块依赖于 PDO 扩展模块进行开发，对外具备一致性的访问接口。具体功能特性如下：

- 支持与虚谷数据库单机及集群的连接访问。
- 支持 IPS 特性连接虚谷数据库集群。
- 完全支持 SQL 标准语法。
- 支持大对象数据类型。
- 支持 SQL 语句的预准备。
- 支持多 SQL 语句执行及获取结果集。

2 快速入手

2.1 开发环境搭建

2.1.1 Windows x86_64

操作步骤

1. 访问 PHP 官方网站，根据需求下载对应的版本，以 7.2.28 版本的安装为例。

图 2-1 下载 php 版本

2/18/2020	4:25 PM	24395483	php-7.2.28-nts-Win32-VC15-x86.zip
2/18/2020	4:25 PM	28549499	php-7.2.28-src.zip
2/18/2020	4:25 PM	26252805	php-7.2.28-Win32-VC15-x64.zip
2/18/2020	4:25 PM	24472895	php-7.2.28-Win32-VC15-x86.zip
3/17/2020	4:51 PM	26111480	php-7.2.29-nts-Win32-VC15-x64.zip
3/17/2020	4:51 PM	24393298	php-7.2.29-nts-Win32-VC15-x86.zip
3/17/2020	4:51 PM	28550033	php-7.2.29-src.zip
3/17/2020	4:51 PM	26251246	php-7.2.29-Win32-VC15-x64.zip
3/17/2020	4:51 PM	24473397	php-7.2.29-Win32-VC15-x86.zip
3/1/2018	7:44 AM	25608632	php-7.2.3-nts-Win32-VC15-x64.zip
3/1/2018	7:44 AM	23869868	php-7.2.3-nts-Win32-VC15-x86.zip
3/1/2018	7:45 AM	28091138	php-7.2.3-src.zip

2. 创建 “C:\php” 目录，并将下载好的文件 php-7.2.28-Win32-VC15-x64.zip 解压到 C:\php 目录，并将路径 “C:\php” 添加至环境变量。
3. 联系虚谷数据库产品团队，下载 PDO xugusql 扩展文件。
4. 解压 PDO xugusql 扩展包后进入扩展目录，进行如下操作：

```
/* 拷贝文件XGCSQL.dll */
>cp ./win32/SDK/lib/XGCSQL.dll C:\php\

/* 拷贝文件php_pdo_xugusql.dll */
>cp ./win32/php_pdo_xugusql.dll C:\php\ext\
```

5. 进入 “C:\php\” 目录，手动复制 php.ini 文件。

```
/* 生成php.ini配置文件 */
>cp ./php.ini-production php.ini
```

6. 修改 php.ini 文件，加入对 pdo_xugusql 扩展支持。

```
/* 修改php.ini配置文件，添加如下内容 */
extension=pdo_xugusql
```

2.1.2 Linux x86_64

操作步骤

1. 访问 PHP 官方网站，根据需求下载对应的版本，以 7.2.30 版本的安装为例。

图 2-2 下载 php 版本



2. 解压 PHP 源码包 php-7.2.30.tar.gz，并进行编译安装至 /usr/local/php/ 目录中。

```
/* php 源代码目录 */
./configure --prefix=/usr/local/php/ //根据开发者需求自行加入其他支持参数
make && make install
```

3. 将 /usr/local/php/bin/ 路径配置入环境变量。

4. 查看 PHP 默认指定的 php.ini 路径。

```
/* 查看 php 默认指定的 php.ini 路径 */
php -i | grep php.ini //此处指定的值为 /usr/local/php/lib/php.ini
```

5. 进入 PHP 源码目录，生成 php.ini 文件到上述指定位置。

```
/* 生成 php.ini 到指定位置 */
cp php.ini.production /usr/local/php/lib/php.ini
```

6. 编辑修改 php.ini 配置文件，修改 PHP 的默认扩展路径变量 extension_dir 的值为 /usr/local/php/modules/，并手动创建该目录。

```
/* 编辑修改 php.ini 配置文件，修改如下内容 */
;extension_dir = "./" /* 修改前的内容 */
extension_dir = "/usr/local/php/modules/" /* 修改后 */
```

7. 联系虚谷数据库产品团队，下载 PDO xugusql 扩展文件。

8. 修改 php.ini 文件，加入对 pdo_xugusql 扩展支持。

2.2 PHP 版本要求

虚谷数据库 PHP 语言访问接口的整个开发过程基于 PHP 7.1.33 完成，对应的 Zend 引擎版本为 Zend Engine v3.1.0，因此在使用 PDO xugusql 访问接口进行开发时建议安装 PHP 7.1.33 及以上版本。

说明

PHP 5.1 及以上版本附带 PDO，在 PHP 5.0 中是作为一个 PECL 扩展使用。PDO 需要 PHP 5 核心的新 OO 特性，因此不能在较早版本的 PHP 上运行。

3 函数接口

3.1 PDO Class

3.1.1 __construct

函数原型

```
public PDO::__construct(string $dsn [, string ...])
```

功能

创建一个表示连接到请求数据库的数据库连接 PDO 实例。

参数

表 3-1 __construct 参数信息

参数名	描述
dsn	数据源名称，包含了请求连接到数据库的信息，格式为‘PDO 驱动名称：连接语法’，其中连接语法以键值对的形式出现并且以分号分隔开。PDO xugusql 支持的连接键有 IP、IPS、DB、PORT、USER、PWD、CHAR_SET、AUTO_COMMIT，其中 IP、IPS 键值信息不允许同时出现在一个连接中

3.1.2 exec

函数原型

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

功能

执行一条 SQL 语句，并返回受此语句影响的行数。

参数

表 3-2 exec 参数信息

参数名	描述
statement	即将被预处理和执行的 SQL 语句

使用范例

```
<?php
$db = new PDO('xugusql:ip=192.168.78.130;port=5138;db=SYSTEM;user=
  SYSDBA;pwd=SYSDBA');
$db->exec("INSERT INTO PHP_TEST VALUES(102, 'WHO
  ', 0.90, '2020-02-21 16:09:00')");
?>
```

3.1.3 lastInsertId

函数原型

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

功能

返回最后插入行的 ID。

参数

表 3-3 lastInsertId 参数信息

参数名	描述
name	接收返回序列的对象名称

使用范例

```
<?php
$db = new PDO('xugusql:ip=192.168.78.130;port=5138;db=SYSTEM;
  user=SYSDBA;pwd=SYSDBA');

$db->exec("INSERT INTO PHP_TEST VALUES(102, 'WHO
  ', 0.90, '2020-02-21 16:09:00')");

$SEQ = $db->lastInsertId();
echo "The last ID is:" . $SEQ. "\n"
?>
```

3.1.4 prepare

函数原型

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

功能

为 PDO Statement::execute() 方法准备待执行的 SQL 语句。SQL 语句可以包含零个或多个参数占位标记，格式是参数名 (:name) 或问号 (?) 的形式，当它执行时将用真实数据取代。在同一个 SQL 语句中，命名形式和问号形式不能同时使用；只能选择其中一种参数形式。

参数

表 3-4 prepare 参数信息

参数名	描述
statement	必须是对目标数据库服务器有效的 SQL 语句模板
drv_opts	数组包含一个或多个 KEY=>VALUE 键值对，为返回的 PDOStatement 对象设置属性

使用范例

```
<?php

$db = new PDO('xugusql:ip=192.168.78.130;port=5138;db=SYSTEM;
  user=SYSDBA;pwd=SYSDBA');

$stmt = $db->prepare("SELECT C2 FROM PHP_TEST WHERE C1 = ?");
$stmt->execute(array(101));

?>
```

3.1.5 query

函数原型

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

功能

PDO::query() 在单次函数调用内执行 SQL 语句，以 PDOStatement 对象形式返回结果集（若存在数据的情况下）。

参数

表 3-5 query 参数信息

参数名	描述
statement	需要准备、执行的 SQL 语句

使用范例

```
<?php
$db = new PDO('xugusql:ip=192.168.78.130;port=5138;db=SYSTEM;
    user=SYSDBA;pwd=SYSDBA');

foreach($db->query("SELECT C1, C2 FROM PHP_TEST;") as $row){
    echo $row['C1']."\t";
    echo $row['C2']."\n";
}
?>
```

3.1.6 beginTransaction

函数原型

```
public PDO::beginTransaction(void) : bool
```

功能

关闭自动提交模式。自动提交模式被关闭的同时，通过 PDO 对象实例对数据库做出的更改直到调用 PDO::commit() 结束事务才被提交。调用 PDO::rollBack() 将回滚对数据库做出的更改并将数据库连接返回到自动提交模式。

使用范例

```
<?php
$db = new PDO('xugusql:ip=192.168.78.130;port=5138;db=SYSTEM;
    user=SYSDBA;pwd=SYSDBA');

$db->beginTransaction();
$db->exec("INSERT INTO PHP_TEST VALUES(102, 'WHO
    ', 0.90, '2020-02-21 16:09:00')");
$db->commit();
?>
```

3.1.7 rollBack

函数原型

```
public PDO::rollBack(void) : bool
```

功能

回滚由 PDO::beginTransaction() 发起的当前事务。

使用范例

```
<?php
$db = new PDO('xugusql:ip=192.168.78.130;port=5138;db=SYSTEM;
    user=SYSDBA;pwd=SYSDBA');

$db->beginTransaction();
$db->exec("INSERT INTO PHP_TEST VALUES(102, 'WHO
    ', 0.90, '2020-02-21 16:09:00')");
$db->rollBack();
?>
```

3.1.8 commit

函数原型

```
public PDO::commit(void) : bool
```

功能

提交一个事务，数据库连接返回到自动提交模式直到下次调用 PDO::beginTransaction() 开始一个新的事务为止。

使用范例

```
<?php
$db = new PDO('xugusql:ip=192.168.78.130;port=5138;db=SYSTEM;
    user=SYSDBA;pwd=SYSDBA');

$db->beginTransaction();
$db->exec("INSERT INTO PHP_TEST VALUES(102, 'WHO
    ', 0.90, '2020-02-21 16:09:00')");
$db->commit();
?>
```

3.1.9 quote

函数原型

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

功能

PDO::quote() 为输入的字符串添加引号（如果有需要），并对特殊字符进行转义，且引号的风格和底层驱动适配。

参数

表 3-6 quote 参数信息

参数名	描述
str	要添加引号的字符串
param_type	为驱动提示数据类型，以便选择引号风格

使用范例

```
<?php
$db = new PDO('xugusql:ip=192.168.78.130;port=5138;db=SYSTEM;
    user=SYSDBA;pwd=SYSDBA');

$C4='2020-02-24 09:56:00';
$db->exec("INSERT INTO PHP_TEST VALUES(104, 'QUOTE', 0.2, ".$db
    ->quote($C4).");");

?>
```

3.1.10 errorCode

函数原型

```
public PDO::errorCode(void) : mixed
```

功能

获取跟数据库句柄上一次操作相关的 SQLSTATE。

使用范例

```
<?php
$db = new PDO('xugusql:ip=192.168.78.130;port=5138;db=SYSTEM;
    user=SYSDBA;pwd=SYSDBA');

$db->exec("INSERT INTO PHP_TEST VALUES(101, 'PHP
    ', 3.90, '2020-02-21 16:09:00')");
echo "error Code:"
$db->errorCode()

?>
```

3.1.11 errorInfo

函数原型

```
public PDO::errorInfo(void) : mixed
```

功能

获取与数据库句柄上的最后一次操作关联的扩展错误信息。

使用范例

```
<?php

$db = new PDO('xugusql:ip=192.168.78.130;port=5138;db=SYSTEM;
    user=SYSDBA;pwd=SYSDBA');

$db->exec("INSERT INTO PHP_TEST VALUES(102, 'PHP
    ', 3.90, '2020-02-21 16:09:00')");
echo "error Code:"
print_r($db->errorInfo());

?>
```

3.1.12 getAvailableDrivers

函数原型

```
public PDO::getAvailableDrivers(void) : mixed
```

功能

返回一个有效可用 PDO 驱动力的数组。

使用范例

```
<?php

print_r(PDO::getAvailableDrivers());

?>
```

3.2 PDOStatement Class

3.2.1 bindColumn

函数原型

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

功能

安排一个特定的变量绑定到一个查询结果集中的列。每次调用 PDOStatement::fetch() 或 PDOStatement::fetchAll() 都将更新所有绑定到列的变量。

参数

表 3-7 bindColumn 参数信息

参数名	描述
column	结果集中的列号（从 1 开始索引）或列名串
param	即将绑定到列的 PHP 变量名称串
type	通过 PDO::PARAM_* 常量指定的参数的数据类型串

使用范例

```
<?php

$db = new PDO('xugusql:ip=192.168.78.130;port=5138;db=SYSTEM;
    user=SYSDBA;pwd=SYSDBA');

$stmt = $db->prepare("SELECT C1, C2 FROM PHP_TEST;");
$stmt->execute();

$stmt->bindColumn(1, $C1);
$stmt->bindColumn(2, $C2);

while($row = $stmt->fetch(PDO::FETCH_BOUND)) {
    echo $C1."\t".$C2."\n";
}

?>
```

3.2.2 bindParam

函数原型

```
PDOStatement::bindParam(mixed $parameter, mixed $var [ , int
    $type... ]) : bool
```

功能

绑定一个 PHP 变量到用作预处理的 SQL 语句中的对应命名占位符或问号占位符。不同于 PDOStatement::bindValue(), 此变量作为引用被绑定, 并只在 PDOStatement::execute() 被调用的时候才取其值。

参数

表 3-8 bindParam 参数信息

参数名	描述
parameter	参数标识符，对于使用命名占位符的预处理语句，应该类似:name 形式的参数名
var	绑定到 SQL 语句参数的 PHP 变量名
type	使用 PDO::PARAM_* 常量明确地指定参数的类型

使用范例

```
<?php

$db = new PDO('xugusql:ip=192.168.78.130;port=5138;db=SYSTEM;
    user=SYSDBA;pwd=SYSDBA');

$c1=101;

$stmt = $db->prepare("SELECT * FROM PHP_TEST WHERE C1 = ?");
$stmt->bindParam(1, $c1, PDO::PARAM_INT);
$stmt->execute();

$row = $stmt->fetch();
print_r($row);

?>
```

3.2.3 bindValue

函数原型

```
PDOStatement::bindValue(mixed $parameter, mixed $value [ , int
    $type... ]) : bool
```

功能

绑定一个值到用作预处理的 SQL 语句中的对应命名占位符或问号占位符。

参数

表 3-9 bindValue 参数信息

参数名	描述
parameter	参数标识符，对于使用命名占位符的预处理语句，应该类似:name 形式的参数名
value	绑定到参数的值
type	使用 PDO::PARAM_* 常量明确地指定参数的类型

使用范例

```
<?php

$db = new PDO('xugusql:ip=192.168.78.130;port=5138;db=SYSTEM;
    user=SYSDBA;pwd=SYSDBA');

$C2='PHP';

$stmt = $db->prepare("INSERT INTO PHP_TEST VALUES
    (100,?,9.0,'2020-02-24 13:49:00');");
$stmt->bindValue(1, $C1, PDO::PARAM_STR);
$stmt->execute();
?>
```

3.2.4 execute

函数原型

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

功能

执行预处理过的语句。如果预处理过的语句含有参数标记，必须选择下面其中一种做法：

- 调用 PDOStatement::bindParam() 绑定 PHP 变量到参数标记：如果有的话，通过关联参数标记绑定的变量来传递输入值和取得输出值
- 传递一个只作为输入参数值的数组

参数

表 3-10 execute 参数信息

参数名	描述
input_parameters	一个元素个数和将被执行的 SQL 语句中绑定的参数一样多的数组

使用范例

```
<?php

$db = new PDO('xugusql:ip=192.168.78.130;port=5138;db=SYSTEM;
  user=SYSDBA;pwd=SYSDBA');

$stmt = $db->prepare("SELECT * FROM PHP_TEST;");
$stmt->execute();

?>
```

3.2.5 fetch

函数原型

```
PDOStatement::fetch( [int $fetch_style [, int $orien [, int
  $offset]]] ) : mixed
```

功能

从一个 PDOStatement 对象相关的结果集中获取下一行。fetch_style 参数决定 PDO 如何返回行（其缺省情况下的默认值为 PDO::FETCH_BOTH），fetch_style 参数的值及其含义列举如图 3-1 所示。

图 3-1 fetch_style 枚举

fetch_style	描述
PDO :: FETCH_ASSOC	返回一个索引为结果集列名的数组
PDO :: FETCH_BOTH	返回一个索引为结果集列名和以0开始的列号的数组
PDO :: FETCH_BOUND	分配结果集中的列给bindColumn()方法绑定的PHP变量
PDO :: FETCH_INTO	更新一个被请求类已存在的实例，映射结果集中的列到类中命名的属性
PDO :: FETCH_LAZY	结合使用FETCH_BOTH和FETCH_OBJ,创建供用来访问的对象变量名
PDO :: FETCH_NUM	返回一个索引为0开始的结果集列号的数组
PDO :: FETCH_OBJ	返回一个属性名对应结果集列名的匿名对象

参数

表 3-11 fetch 参数信息

参数名	描述
fetch_style	控制下一行如何返回给调用者，此值必须是 PDO::FETCH_* 系列常量中的一个
orien	此值必须是 PDO::FETCH_ORI_* 系列常量中的一个，默认为 PDO::FETCH_ORI_NEXT
offset	此值指定结果集中想要获取行的绝对行号

使用范例

```
<?php

$db = new PDO('xugusql:ip=192.168.78.130;port=5138;db=SYSTEM;
    user=SYSDBA;pwd=SYSDBA');

$stmt = $db->prepare("SELECT * FROM PHP_TEST;");
$stmt->execute();

$row = $stmt->fetch();
print_r($row);
?>
```

3.2.6 fetchAll

函数原型

```
PDOStatement::fetchAll([int $style [, mixed $arg [, array $ctor
    = array()]]) : array
```

功能

返回一个包含结果集中所有行的数组。

参数

表 3-12 fetchAll 参数信息

参数名	描述
style	控制返回数组的内容如同 PDOStatement::fetch() 文档记载的一样
arg	根据 fetch_style 参数的值，此参数有不同的意义
ctor	当 fetch_style 参数为 PDO::FETCH_CLASS 时，自定义类的构造函数的参数

使用范例

```
<?php

$db = new PDO('xugusql:ip=192.168.78.130;port=5138;db=SYSTEM;
    user=SYSDBA;pwd=SYSDBA');

$stmt = $db->prepare("SELECT * FROM PHP_TEST WHERE C1= 101;");
$stmt->execute();

$row = $stmt->fetchAll();
print_r($row);
?>
```

3.2.7 fetchColumn

函数原型

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

功能

从结果集中的下一行返回单独的一列，如果没有数据，则返回 FALSE。

参数

表 3-13 fetchColumn 参数信息

参数名	描述
column_number	希望从行里取回的列的索引数字（以 0 开始的索引），缺省情况下获取第一列

使用范例

```
<?php

$db = new PDO('xugusql:ip=192.168.78.130;port=5138;db=SYSTEM;
    user=SYSDBA;pwd=SYSDBA');

$stmt = $db->prepare("SELECT * FROM PHP_TEST;");
$stmt->execute();

$res = $stmt->fetchColumn();
print "C1:". $res. "\n";

$res = $stmt->fetchColumn(1);
print "C2:". $res. "\n"

?>
```

3.2.8 fetchObject

函数原型

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

功能

获取下一行并作为一个对象返回。此函数（方法）是使用 PDO::FETCH_CLASS 或 PDO::FETCH_OBJ 风格的 PDOStatement::fetch() 的一种替代。

参数

表 3-14 fetchObject 参数信息

参数名	描述
class	创建类的名称
接下一页	

参数名	描述
ctor	此数组的元素被传递给构造函数

使用范例

```
<?php

$db = new PDO('xugusql:ip=192.168.78.130;port=5138;db=SYSTEM;
    user=SYSDBA;pwd=SYSDBA');

$stmt = $db->prepare("SELECT * FROM PHP_TEST;");
$stmt->execute();

$obj = $stmt->fetchObject();
var_dump($obj);

?>
```

3.2.9 rowCount

函数原型

```
PDOStatement::rowCount(void) : int
```

功能

PDOStatement::rowCount() 返回上一个由对应的 PDOStatement 对象执行 DELETE、INSERT 或 UPDATE 语句影响的行数。

使用范例

```
<?php

$db = new PDO('xugusql:ip=192.168.78.130;port=5138;db=SYSTEM;
    user=SYSDBA;pwd=SYSDBA');

$stmt = $db->prepare("update php_test set C1 = 111 where C1
    = 110;");
$stmt->execute();

$count = $stmt->rowCount();
echo "row count:". $count. "\n\n"

?>
```

3.2.10 closeCursor

函数原型

```
public PDOStatement::closeCursor(void) : bool
```

功能

PDOStatement::closeCursor() 释放到数据库服务的连接，以便发出其他 SQL 语句，但使语句处于一个可以被再次执行的状态。

使用范例

```
<?php

$db = new PDO('xugusql:ip=192.168.78.130;port=5138;db=SYSTEM;
    user=SYSDBA;pwd=SYSDBA');

$stmt1 = $db->prepare("SELECT C1 FROM PHP_TEST;");
$stmt1->execute();

$stmt1->bindColumn(1, $C1);
$stmt1->fetch();
$stmt1->closeCursor();

?>
```

3.2.11 columnCount

函数原型

```
public PDOStatement::columnCount(void) : int
```

功能

PDOStatement::columnCount() 返回由 PDOStatement 对象代表的结果集中的列数，若没有结果集，则返回 0。

使用范例

```
<?php

$db = new PDO('xugusql:ip=192.168.78.130;port=5138;db=SYSTEM;
    user=SYSDBA;pwd=SYSDBA');

$stmt = $db->prepare("SELECT * FROM PHP_TEST;");
$stmt->execute();
$CNT = $stmt->columnCount();

print $CNT."\n"
?>
```

3.2.12 debugDumpParams

函数原型

```
public PDOStatement::debugDumpParams(void) : bool
```

功能

直接打印一条预处理语句包含的信息。提供正在使用的 SQL 查询、所用参数的数目、参数的清单、参数名、用一个整数表示的参数类型、键名或位置、值、以及在查询中的位置。

使用范例

```
<?php
$db = new PDO('xugusql:ip=192.168.78.130;port=5138;db=SYSTEM;
    user=SYSDBA;pwd=SYSDBA');

$c1 = 103;
$stmt = $db->prepare("SELECT * FROM PHP_TEST WHERE C1 = ?");
$stmt->bindParam(1, $c1);
$stmt->execute();

$stmt->debugDumpParams();

?>
```

3.2.13 errorCode

函数原型

```
public PDOStatement::errorCode(void) : string
```

功能

PDOStatement::errorCode 获取跟上一次语句句柄操作相关的 SQLSTATE。

使用范例

```
<?php
$db = new PDO('xugusql:ip=192.168.78.130;port=5138;db=SYSTEM;
    user=SYSDBA;pwd=SYSDBA');

$stmt = $db->prepare("select * from php_test where c1 = ?");
$stmt->execute();

echo 'test errorCode:'. $stmt->errorCode(). "\n\n";

?>
```

3.2.14 errorInfo

函数原型

```
public PDOStatement::errorInfo(void) : array
```

功能

PDOStatement::errorInfo() 返回一个关于上一次语句句柄执行操作的错误信息的数组。

使用范例

```
<?php
$db = new PDO('xugusql:ip=192.168.78.130;port=5138;db=SYSTEM;
    user=SYSDBA;pwd=SYSDBA');

$stmt = $db->prepare("INSERT INTO PHP_TEST VALUES(1, ?, ?, ?);");
$stmt->execute();

$info = $stmt->errorInfo();
print_r($info);

?>
```

4 存储过程与函数

4.1 输入型参数的存储过程

概述

执行输入型参数的存储过程的用法与其执行普通的带参数的 SQL 语句用法基本一致，参数以值传递的形式给予存储过程，参数不接受在执行存储过程期间内部的赋值。

使用范例

```
<?php

$db = new PDO('xugusql:ip=192.168.78.130;port=5138;db=SYSTEM;
    user=SYSDBA;pwd=SYSDBA');

$pval = 7;
$stmt = $db->prepare("exec test_proce_1(:argv);");

$stmt->bindParam(':argv', $pval, PDO::PARAM_STR);
$stmt->execute();

?>
```

4.2 输出型参数的存储过程

概述

执行输出型参数的存储过程的用法与执行输出型参数的存储过程稍有区别，即在进行参数绑定阶段时，应当指定参数的类型为 PDO::PARAM_INPUT_OUTPUT。输出型参数会接受在执行存储过程期间内部对参数值的更改，从而达到一个存储过程内部与外界的信息交互目的。

使用范例

```
<?php

$db = new PDO('xugusql:ip=192.168.78.130;port=5138;db=SYSTEM;
    user=SYSDBA;pwd=SYSDBA');

$stmt = $db->prepare("exec test_proce_1(:argv);");

$stmt->bindParam(':argv', $pval, PDO::PARAM_INPUT_OUTPUT);
$stmt->execute();
echo "pval : ".$pval."\n";

?>
```

4.3 存储函数

概述

执行带参数的存储函数与执行带参数的存储过程用法基本一致，其执行存储过程与执行存储函数的主要区别在于存储函数需要额外绑定一个接收返回值的参数。

使用范例

```
<?php

$db = new PDO('xugusql:ip=192.168.78.130;port=5138;db=SYSTEM;
    user=SYSDBA;pwd=SYSDBA');

$pval = 7;
$stmt = $db->prepare("exec test_func_1(:argv);");

$stmt->bindParam(':argv', $pval, PDO::PARAM_INT);
$stmt->bindParam(':relv', $relv, PDO::PARAM_INPUT_OUTPUT);
$stmt->execute();

echo "pval : ".$pval."\n";
echo "relv : ".$relv."\n";

?>
```

5 LOB 大对象

5.1 插入 BLOB 大对象数据

使用范例

```
<?php
$db = new PDO('xugusql:ip=192.168.78.130;port=5138;db=SYSTEM;
    user=SYSDBA;pwd=SYSDBA');
$llob=fopen('/home/T/mac-002.jpg', "r");

$stmt = $db->prepare("INSERT INTO PHP_LOB VALUES (1, ?, NULL,
    NULL);");
$stmt->bindValue(1, $f_blob, PDO::PARAM_LOB);

$stmt->execute();
fclose($llob);
?>
```

5.2 获取 BLOB 大对象数据

使用范例

```
<?php
.....
$llob=fopen('./002.jpg', "wb");

$stmt = $db->prepare("SELECT C2 FROM PHP_LOB;");
$stmt->execute();

$stmt->bindColumn(1, $contex, PDO::PARAM_LOB);
$stmt->fetch(PDO::FETCH_BOUND);

while(!feof($contex)){
    fwrite($f_clob, fgets($contex, 102400));
}

fclose($llob);
?>
```

6 错误码

部分错误码详细信息如表 6-1 所示。

表 6-1 错误码信息

错误码	错误标识	描述
E5021	ERR_TAB_NO_EXIST	表或视图不存在
E5022	ERR_TAB_EXIST	表已存在
E7002	ERR_SEQ_NO_EXIST	序列值发生器不存在
E10001	ERR_FUNC_NO_EXIST	违反唯一值约束
E13001	ERR_DUP_VAL_ON_INDEX	违反唯一值约束
E17005	ERR_DATA_LEN	违反唯一值约束
E19009	ERR_NO_DATA_FOUND	查询无结果
E19132	ERR_PARSE	语法错误



成都虚谷伟业科技有限公司

联系电话：400-8886236

官方网站：www.xugudb.com