

# Расширение PHP для работы с YottaDB

Данное расширение предназначено для работы с переменными MUMPS в СУБД *YottaDB* и предоставляет следующие функции:

```
ydb_init([env])
ydb_finalize()
ydb_set(var, keys, ..., val)
ydb_get(var, keys, ...)
ydb_getstring(var, keys, ...)
ydb_kill(var, keys, ...)
ydb_data(var, keys, ...)
ydb_order(var, keys, ..., dir)
ydb_error()
ydb_strerror()
ydb_zsinfo(&id, &severity, &text)
```

Большинство функций в случае ошибки возвращают `false`. Код ошибки можно получить с помощью функции `ydb_error`, возвращающей его в виде целого числа. Все коды ошибок можно разделить на коды ошибок, определенных данным расширением и коды ошибок, возникающих в *YottaDB*. Для кодов ошибок, определённых расширением, имеются следующие константы:

```
YDB_ERR_LASTCLD — больше нет индексов для обхода.
YDB_ERR_NUMOFRANGE — слишком большое числовое значение.
YDB_ERR_STR2LONG — превышена максимальная длина строки.
```

Коды ошибок *YottaDB* определены в формате `YDB_ERR_ID`, где `ID` — идентификатор сообщения *YottaDB* (см. [1], раздел «ZMessage Codes»). При получении одного из таких кодов, можно вызвать функцию

```
ydb_zsinfo(&id, &severity, &text)
```

, которая возвращает информацию из внутренней переменной *YottaDB* `$ZSTATUS` (см. [2], разделы «8. Intristic Special Variables» и «13. Error Processing») через передаваемые по ссылке строковые аргументы `id`, `severity` и `text`.

Поле `id` является текстовым и содержит идентификатор сообщения *YottaDB* (см. [1], раздел «ZMessage Codes»). Числовое поле `severity` указывает на серьёзность ошибки и может принимать одно из значений, определённых следующими константами:

```
YDB_SEVERITY_ERROR
YDB_SEVERITY_FATAL
YDB_SEVERITY_INFORMATIONAL
YDB_SEVERITY_SUCCESS
YDB_SEVERITY_WARNING
```

, которые соответствуют аналогичным константам, приведенным в [3], в разделе «Severity». Текстовое поле `text` содержит текстовое описание ошибки.

Текстовое описание существует для всех ошибок. Его можно получить с помощью функции `ydb_strerror`, возвращающей это описание как строку. При внутренних

ошибках базы возвращается тот же текст, что функция `ydb_zstatus` помещает в свой аргумент `text`.

## 1. Начало и завершение работы

Функция

```
ydb_init([env])
```

предназначена для подготовки окружения к работе с базой данных и всегда вызывается прежде, чем будут вызваны любые другие функции данного расширения. Её единственный аргумент, `[env]`, опционален и является ассоциативным массивом, где ключами являются имена переменных окружения, которые необходимо задать при инициализации базы данных, а элементами — значения этих переменных (полный список переменных окружения, используемых для настройки YottaDB, см. в [4], раздел «3. Basic Operations»).

При отсутствии аргумента `[env]` или каких либо его элементов, соответствующих переменным окружения, необходимым для инициализации работы с базой данных, будет выполнено подключение со значениями этих переменных в соответствии с настройками YottaDB в локальной системе. В случае успеха данная функция возвращает `true`, в случае ошибки — `false`.

Функция

```
ydb_finalize()
```

возвращает окружение в то состояние, в котором оно было до последнего вызова `ydb_init`. Эта функция вызывается всегда, после того, как работа с базой данных завершена. Как и `ydb_init`, в случае успеха возвращает `true`, а при ошибке возвращает `false`.

## 2. Функции для работы с переменными MUMPS

Функции `ydb_set`, `ydb_get`, `ydb_getstring`, `ydb_kill`, `ydb_data` и `ydb_order`, предназначены для работы с переменными *MUMPS*.

Первые аргументы, `var` и произвольное число аргументов `keys`, для всех этих функций задают переменную или элемент массива, с которым необходимо выполнить действие. Аргумент `var` является строкой, указывающей имя переменной. Если имя переменной начинается с символа «^», то она является глобальной, иначе переменная является локальной и исчезает после вызова `ydb_finalize`.

В случае, если переменная `var` является массивом, за ней следует произвольное число аргументов `keys`, указывающих индексы её элемента, с которыми необходимо выполнить действие, идущие от старшего индекса к младшему.

Каждый из аргументов `keys` может являться строкой, целым числом, числом плавающей точкой или массивом РНР с целочисленными ключами, содержащим перечисленные значения перечисленных типов. Если аргумент `keys` является строкой или числом, то он указывает один индекс. Если же `keys` является массивом, то он содержит несколько идущих подряд, индексов, от старшего к младшему.

Имя переменной и индексов должны соответствовать правилам именования переменных MUMPS (см. [2], раздел «5. General Language Features of M»). Также на имена переменных и индексов накладываются ограничения самой YottaDB: длина имени переменной не должна превышать 31 символ, количество индексов не должно превышать 31. Помимо этого существуют ограничения на объем памяти, занимаемый всеми ключами вместе с именем переменной (см. [4], раздел «4. Global Directory Editor»).

Возможны следующие ошибки, связанные с передачей индексов:

- YDB\_ERR\_MAXNRSUBSCRIPTS, если количество индексов, переданных функции, превышает 31.
- YDB\_ERR\_KEY2BIG, если объем памяти, занимаемый переданным именем глобальной переменной превышает ограничение, заданное настройками базы данных.
- YDB\_ERR\_GVSUBOFLOW, если объем памяти, занимаемый переданными *индексами вместе с именем глобальной переменной* превышает ограничение, заданное настройками базы данных.
- YDB\_ERR\_NUMOFRANGE, если индекс, переданный в качестве числа с плавающей точкой или целого выходит за пределы ограничений, определённых в YottaDB для такого числа (см. приложение).
- YDB\_ERR\_STR2LONG, если индекс, переданный в качестве строки, превышает максимально возможную длину строки, определённую настройками базы данных.

Функция

```
ydb_set(var, keys, ..., val)
```

устанавливает значение переменной или элемента массива MUMPS. Аргумент `val` может быть строкой, целым числом, числом с плавающей точкой и булевым значением. Булево значение `false` преобразуется в число 0, а значение `true` преобразуется в 1. В случае ошибки возвращает `false`, иначе возвращает `true`. Помимо ошибок связанных с передачей имени переменной и индексов, данная функция может вернуть следующие ошибки:

- YDB\_ERR\_NUMOFRANGE, если аргумент `val`, является числом с плавающей точкой или целым и выходит за пределы ограничений, определённых в YottaDB для такого числа (см. приложение).
- YDB\_ERR\_STR2LONG, если аргумент `val`, является строкой и превышает максимально возможную длину строки, определённую настройками базы данных.

Функция

```
ydb_get(var, keys, ...)
```

возвращает значение переменной или элемента массива MUMPS. Определение типа выполняется автоматически — это может быть число с плавающей точкой, целое число или строка. Если значение является числом без десятичного разделителя и других символов, за исключением знака «+» или «-» в начале, длина которого не превышает 18 цифр, возвращается целое число (тип `integer`). Если же значение является любым другим числом, возможным в MUMPS, возвращается число с плавающей точкой (тип `float`). При любом другом значении возвращается строка (тип `string`). Если значение содержит число только частично, также возвращается строка. В случае ошибки возвращается `false`.

Помимо ошибок, связанных с передачей имени переменной и индексов данная функция может вернуть следующие ошибки:

- `YDB_ERR_OOFRANGE`, если значение переменной MUMPS является целым числом и оно больше максимально возможного или меньше минимально возможного целого числа в PHP (см. приложение).
- `YDB_ERR_OOFRANGE`, если значение переменной MUMPS является числом с десятичным разделителем и оно выходит за пределы ограничений, определенных для чисел с плавающей точкой в PHP (см. приложение).
- `YDB_ERR_STR2LONG`, если значение переменной MUMPS является строкой и его длина превышает максимально возможную длину строки, определенную настройками PHP.
- `YDB_ERR_GBUNDEF`, если запрашивается глобальная переменная, которой не существует.
- `YDB_ERR_LVUNDEF`, если запрашивается локальная переменная, которой не существует.
- `YDB_ERR_INVSVN`, если запрашивается встроенная специальная переменная, которой не существует.

В случае, если переменная MUMPS содержит числовое значение, которое невозможно преобразовать в значение PHP соответствующего типа, можно использовать функцию

```
ydb_getstring(var, keys, ...)
```

, возвращающую значение любой переменной MUMPS в виде строки. В случае ошибки данная функция, как и другие подобные функции, возвращает `false`. Список ошибок, которые может вернуть данная функция, аналогичен списку возвращаемых ошибок для функции `ydb_get`, за исключением отсутствия ошибки `YDB_ERR_OOFRANGE`.

Функция

```
ydb_kill(var, keys, ...)
```

удаляет переменную или элемент массива MUMPS. Возвращает `true` в случае успеха и `false` в случае ошибки. Данная функция может вернуть только ошибки, связанные с передачей имени переменной и индексов. В случае, если удаляется несуществующая переменная или элемент массива, функция возвращает `true` не совершая никаких действий.

Функция

```
ydb_data(var, keys, ...)
```

возвращает целое число, содержащее информацию о состоянии переменной или элемента массива MUMPS:

- Если переменная или элемент массива не определён, возвращается 0.
- Если переменная или элемент массива имеет значение, но не имеет потомков, возвращается 1.
- Если переменная или элемент массива не имеет значения, но имеет потомков, возвращается 10.
- Если переменная или элемент массива имеет значение и имеет потомков, возвращается 11.
- В случае ошибки возвращает `false`. Данная функция может только ошибки, связанные с передачей имени переменной и индексов.

Функция

```
ydb_order(var, keys, ..., dir)
```

используется для итерации по индексам массива MUMPS. Аргумент `dir` указывает порядок обхода индексов: если он равен 1 возвращается следующий индекс в переданном ей массиве, если же он равен -1, возвращается предыдущий индекс. Порядок обхода индексов определяется по правилам сортировки значений MUMPS. В случае ошибки, возвращается `false`. Помимо ошибок, связанных с передачей имени переменной и индексов, данная функция может вернуть следующие ошибки:

- `YDB_ERR_LASTCLD`, если закончились индексы для обхода.
- `YDB_ERR_ORDER2`, если аргумент `dir` не равен 1 или -1.

## 3. Примеры

Определение глобальной переменной «`^country`» со значением «Russia»:

```
ydb_set("^country", "Russia");
```

Определение глобального массива «`^person`»:

```
$p = array("Komi Republic", "Syktyvkar");
ydb_set("^person", $p, 1, "name", "Ivan");
ydb_set("^person", $p, 1, "surname", "Ivanov");
ydb_set("^person", $p, 2, "name", "Vasilii");
ydb_set("^person", $p, 2, "surname", "Vasil'ev");

ydb_set("^person", "Komi Republic", "Ukhta", 1,
        "name", "Dmitiy");
ydb_set("^person", "Komi Republic", "Ukhta", 1,
        "surname", "Dmitriev");

$p = array("Kirovskaya Oblast'", "Kirov");
ydb_set("^person", $p, 1, "name", "Pyotr");
ydb_set("^person", $p, 1, "surname", "Petrov");
ydb_set("^person", $p, 2, "name", "Aleksey");
ydb_set("^person", $p, 2, "surname", "Alekseev");
```

Вывод значения глобальной переменной `^country`:

```
echo ydb_get("^country");
```

результат:

```
Russia
```

Вывод значения значений отдельных элементов глобального массива «^person»:

```
$p = array("Komi Republic", "Syktyvkar");  
echo ydb_get("^person", $p, 2, "name"),  
      " ", ydb_get("^person", $p, 2, "surname"), "\n";
```

```
$p = array("Kirovskaya Oblast", "Kirov");  
echo ydb_get("^person", $p, 1, "surname"),  
      " ", ydb_get("^person", $p, 1, "surname"), "\n";
```

Результат:

```
Vasiliy Vasil'ev  
Pyotr Petrov
```

Удаление элемента глобального массива «^person» вместе со всеми его дочерними элементами:

```
ydb_kill("^person", "Komi Republic", "Ukhta");
```

Получение информации о глобальной переменной «^country» и отдельных элементах глобального массива «^person»:

```
echo ydb_data("^country"), "\n";  
echo ydb_data("^person", "Komi Republic", "Ukhta", 2), "\n";  
echo ydb_data("^person", "Komi Republic", "Syktyvkar",  
              1, "name"), "\n";  
echo ydb_data("^person", "Komi Republic"), "\n";
```

Результат:

1  
0  
1  
10

Получение отдельных индексов глобального массива «`^person`»:

```
echo ydb_order("^person", "Komi Republic",  
               "Syktyvkar", 1), "\n";  
echo ydb_order("^person", "Komi Republic",  
               "Syktyvkar", 1), "\n";
```

Результат:

1  
2

Обход всех дочерних элементов другого элемента глобального массива «`^person`»:

```
$p = array("Kirovskaya Oblast'", "Kirov");  
while (($k = ydb_order("^person", $p, 1))) {  
    $v = ydb_get("^person", $p, $k, "name");  
    ...  
}  
  
assert(ydb_error() == YDB_ERR_LASTCLD);
```

## Приложение. Преобразование разных типов данных между PHP и YottaDB

При преобразовании типов данных между YottaDB и PHP существуют ограничения. Они появляются из-за того, что одни и те же типы данных в PHP и YottaDB имеют разное внутреннее представление.

YottaDB хранит значения всех переменных и элементов массивов в виде строк, но при этом, при выполнении арифметических вычислений, преобразует данные строки в числа. Числа и строки в MUMPS имеют следующие ограничения:

- Максимальная длина строки определяется настройками базы данных и при этом никогда не может превышать 1мб.
- Целое число не может состоять более чем из 18 цифр.
- Число с плавающей точкой может иметь до 18 значащих цифр и должно лежать в диапазоне  $[10^{-43}; 10^{47}]$ .

RНР также имеет свои ограничения, связанные с типами данных Си и зависящие от архитектуры компьютера:

- Максимальная длина строки определяется настройками RНР, но не может превышать 2гб для 32-битных архитектур и практически не имеет ограничений для 64-битных архитектур.
- Целое число должно лежать в диапазоне  $[-2^{31} + 1; 2^{31} - 1]$  для 32-битных архитектур и в диапазоне  $[-2^{63} + 1; 2^{63} - 1]$  для 64-битных архитектур.
- Число с плавающей точкой на большинстве архитектур соответствует числу двойной точности из стандарта *IEEE 754-2008*. Такое число может иметь от 15 до 17 значащих цифр и лежит в диапазоне  $[10^{-308}; 10^{308}]$ .

Если ограничения для строк зависят как от настроек базы данных, так и от настроек RНР, то ограничения для чисел зависят только от архитектуры и их можно представить в следующем виде:

Тип	из YottaDB в RНР		из RНР в YottaDB	
	Зн. цифры	Диапазон	Зн. цифры	Диапазон
32-битное целое	-	$[-2^{31}+1; 2^{31}-1]$	-	$[-2^{31}+1; 2^{31}-1]$
64-битное целое	18	$[-2^{63}+1; 2^{63}-1]$	18	$[-2^{63}+1; 2^{63}-1]$
Число с плавающей точкой	15-17	$[1E-43; 1e47]$	15	$[1E-43; 1e47]$

Как уже было упомянуто выше, если число при преобразовании выходит за пределы этих ограничений, возвращается ошибка. Причем, в случае с целыми числами ошибка возвращается всегда, а в случае с числами с плавающей точкой происходит преобразование с потерей точности и ошибка возникает только тогда, когда число выходит за пределы диапазона своих значений.

## Ссылки

1. <https://docs.yottadb.com/MessageRecovery> — Message and Recovery Procedures Manual.
2. <https://docs.yottadb.com/ProgrammersGuide> — M programmer's Guide
3. <https://docs.yottadb.com/MultiLangProgGuide> — MultiLanguage Programmers Guide.
4. <https://docs.yottadb.com/AdminOpsGuide> — Administration and Operations Guide.



## Соображения

- У варианта с функциями, имеющими произвольное число аргументов, есть недостатки, возможно от него придётся отказаться, например, когда нужно передать сразу два элемента массива.
- К функции `ydb_data`, возможно, стоит добавить дополнительные мини-функции для работы с её возвращаемыми значениями: `YDB_HAVEVAL(n)` и `YDB_HAVECLD(n)`.
- Наверное, еще стоит добавить аналог команд `LOCK` и `MERGE`, а возможность вызова пользовательских функций.
- Функция `ydb_getstring`, по сути, представляет часть отдельного API, где типы переменной не определяются автоматически. Возможно к ней стоит добавить функции `ydb_getint` и `ydb_getfloat`. Или вообще отказаться от автоматического определения типа. Хотя мне кажется, что определение типов не самая плохая идея для РНР.
- Я определенно не до конца разобрался с числами с плавающей точкой. Что двигаться дальше, придётся.
- Индексы могут принимать значение `null`, надо будет разобраться с этим.