Database best practices

Best practices (all versions)

> **Avoid methods that do automatic html conversion**, like `query_result` and `fetchByAssoc(...)`

> **Never concatenate variables (especially coming from request) into the sql string**

> **Always use the no_html** versions (`query_result_no_html` or `fetchByAssoc(..., -1, false)`) or the new shortcut methods (see below)

> **Always use prepared statement** (`pquery`), to avoid SQL injections

> In `SELECT` statements, **select only the necessary columns**, avoid `*` if you don't need all the columns

> In `INSERT` statements, **always specify the column names** (to avoid breaking the query in case of new columns added later)

Shortcut methods (version ≥ 26.01)

New utility methods have been added to easy query execution and retrieval of rows. None of these functions will do any html conversion of the result.

- `rows()`: Iterate over rows:

```
$res = $adb->pquery("SELECT * FROM table WHERE column = ?", [$param]);
foreach ($res->rows() as $row) {
  // ... use $row
}
```

This is equivalent of:

```
$res = $adb->pquery("SELECT * FROM table WHERE column = ?", [$param]);
while ($row = $adb->fetchByAssoc($res, -1, false)) {
  // ... use $row
}
```

- `row($index = 0)`: Get a single row:

```
$res = $adb->pquery("SELECT * FROM table WHERE column = ?", [$param]);
$row1 = $res->row(1); // get the second row (index starts at 0)
```

- `allRows()`: Read all rows (note: may use a lot of memory if there are many rows):

```
$res = $adb->pquery("SELECT * FROM table WHERE column = ?", [$param]);
$allrows = $res->allRows();
```

- `col($column)`: Return a specific column (by index or by name) from the result set (note: may use a lot of memory if there are many rows):

```
$res = $adb->pquery("SELECT * FROM table WHERE column = ?", [$param]);

$col = $res->col();     // first column
$col = $res->col(2);    // 3rd column (0-based index)
$col = $res->col('id'); // "id" column
```

- `allCols()`: Return all the columns from the result set (note: may use a lot of memory if there are many rows). The return value is an array, where each key is the name of the column and the value, an array of the values for that column:

```
$res = $adb->pquery("SELECT id, name FROM table WHERE column = ?", [$param]);
$all = $res->allCols();

/*
$all looks like:
Array (
    [id] => Array
        (
```

```
            [0] => 1
            [1] => 6
        )
    [name] => Array
        (
            [0] => first value
            [1] => something else
        )
)
*/
```

- `queryGetAll($sql, $params = [], $offset = 0, $limit = -1)`: Combines `pquery` and `allRows`:

```
// get all rows from query (no parameters)
$rows = $adb->queryGetAll("SELECT column1, column2 FROM table");

// get all rows from query (single parameter)
$rows = $adb->queryGetAll("SELECT * FROM table WHERE column = ?", $param);

// with limit, select first 5 rows
$rows = $adb->queryGetAll("SELECT * FROM table WHERE column = ? AND id < ?", [$param,
6], 0, 5);
```

- `queryGetFirst($sql, $params = [])`: Get the first row returned by the query:

```
// no need to specify LIMIT
$row = $adb->queryGetFirst("SELECT column1, column2 FROM table");
```

- `queryGetFirstValue($sql, $params = [], $column = 0)`: Get a single value from the first row returned:

```
// first column, if nothing specified
$column1 = $adb->queryGetFirstValue("SELECT column1, column2 FROM table");

// by name
$column2 = $adb->queryGetFirstValue("SELECT column1, column2 FROM table", [],
```

```
'column2');
```

- `queryHasRows($sql, $params = [])`: Return true if the query returns at least one row:

```
$hasRows = $adb->queryHasRows("SELECT column1 FROM table WHERE id > ?", [10]);
```

- `queryGetAllColumns($sql, $params = [], $offset = 0, $limit = -1)`: Combines `pquery` and `allCols`:

```
// get all columns from query (no parameters)
$cols = $adb->queryGetAllColumns("SELECT column1, column2 FROM table");

// get all columns from query (single parameter)
$cols = $adb->queryGetAllColumns("SELECT * FROM table WHERE column = ?", $param);

// with limit, select first 5 rows and return them as columns
$cols = $adb->queryGetAllColumns("SELECT * FROM table WHERE column = ? AND id < ?",
[$param, 6], 0, 5);
```

- `queryGetFirstColumn($sql, $params = [])`: Get the first column returned by the query:

```
// no need to specify LIMIT
$col1 = $adb->queryGetFirstColumn("SELECT column1, column2 FROM table");
```

> In version 26.01, there is still a main source of queries transforming data to html: **reading records**. So all queries reading records to be displayed in ListView, DetailView, EditView, Reports... are still converting to html.
> This behaviour was kept to avoid too many breaks with existing custom code (uitypes, views, presave, ...)

Aliased methods (vtenext ≥ 26.01)

Some methods in PearDatabase have now new aliases, to better remember that a html conversion can take place:

- `query_result_html` -> alias of `query_result`
- `fetch_array_html` -> alias of `fetch_array`

- `fetchByAssocHtml` -> alias of `fetchByAssoc`, with 3rd parameter set to true
- `fetchByAssocNoHtml` -> alias of `fetchByAssoc`, with 3rd parameter set to false

It is recommended not to use the html methods, but if really necessary, try to use the "html" aliases.

New methods (version ≥ 26.04)

- `pqueryTimeout(string $sql, ?array $params, int $timeout, bool $dieOnError=false)`: Executes a SELECT statement with a timeout in ms (only for MySql connections). Returns the string `TIMEOUT` if a timeout occured.

```
$res = $adb->pqueryTimeout("SELECT * FROM table ORDER BY date_field", null, 3000);
if ($res === 'TIMEOUT') {
  // took longer than 3s
  // alternative code to execute the query in background
  return;
else {
  // use $res
}
```

Revision #17
Created 2026-01-26 13:10:20 UTC by manuel.tagliapietra
Updated 2026-04-20 14:42:02 UTC by manuel.tagliapietra