Release notes for version 2.5

Behavior changes

Important! This release changes behavior of some transformations and expressions. You might need to modify your projects to make them work correctly in the new version (e.g. adjust column names after **Aggregate** transformation). Create a backup copy of your projects before updating to the new version.

1. Default column names for **Aggregate** transformation:

Old behavior	New behavior
Column names for aggregates always contain aggregation type. For a sum of **Amount**, the aggregate column name would be **Sum of Amount**.	Column names for aggregates are the same as original column names. For a sum of **Amount**, the aggregated column name would still be **Amount**. Aggregation type is added only when there is more than one aggregation for the same column. For instance, if both sum and max are calculated for column **Amount**, the aggregated columns would get names **Sum of Amount** and **Max of Amount** respectively.

2. Arithmetical operations with empty cells:

Old behavior	New behavior
empty() * empty() produced error *#Can't multiply not numbers*.	empty() * empty() equals to 0.

3. **Iterate** transformation in *Iterate and Append* mode now automatically adds columns with parameter values used for iterations. For instance, if a project iterates through a list of file names, than the final table gets additional column with file names used for each iteration step.

Old behavior	New behavior
Result tables are appended into one final table that becomes the transformation output.	One or more columns are added to every result table – one column per each parameter used for the iteration. Then the result tables are appended into one final table that becomes the transformation output.

4. Syntax errors in expressions now halt project execution. Previously they returned *#Syntax error* as an expression result and the project kept running.

5. **Import from QVD** changed behavior if a dual double value fails to convert into a decimal number, then the text part of the dual is used. Previously this returned *#Can't convert value into a decimal number* (EasyMorph doesn't support floats/doubles).

Old behavior	New behavior
When importing dual double value 123E100 it converted to error value (*#Can't convert value into a decimal number*).	When importing dual double value 123E100 it converts into text '123E100'.

Also, this release modifies the project file structure and therefore projects created or saved in this version can't be opened in older versions of EasyMorph.

What's new

Database connectors and Connector Manager

Version 2.5 introduces native database connectors for the following database types:

- Oracle database
- MS SQL Server
- MySQL
- SQLite

A native DB connector means that EasyMorph already includes client software to connect to a database without ODBC, OLE DB or any other type of database client previously installed.

Connectors in EasyMorph can be embedded into a project -- in this case the project keeps using the same connection settings on different computers. Alternatively, connectors can be stored in a shared repository created automatically by EasyMorph -- in this case many users can share the same connection settings. Having shared connector repositories is also convenient for migrating projects from dev to test/prod environments (shared connectors are simply referenced by name). To reduce a chance of error, a shared repository can be protected with a password for adding new or modifying existing connectors. Using a connector for importing data doesn't require a password.

Connector Manager is a new dialog that is used to manage connectors. It allows creating, editing, deleting connectors, as well as copying connectors between projects and repositories in any direction. You can also use Connector Manager to create new repositories and select active repository.

Connector Manager can be found in menu Project, or invoked by pressing F7.

Sandboxes

EasyMorph is shifting towards data analysis and the sandboxes are the first step in that direction. A sandbox is a regular table which initial dataset can be populated either from the clipboard, or copied from another transformation. Sandboxes remain static and don't change during full project re-

calculations. Although, sandbox data is not meant to be persistent and is not saved with project. Therefore, when a project is re-opened its sandboxes are empty.

Sandboxes can be used in various scenarios that require answering spontaneous questions, or for one-time data manipulations. For instance:

- Copy/paste data into EasyMorph from Excel or another EasyMorph project. Merge copied data (which is now in a sandbox) with calculated data using **Merge** transformation for one-time reconciliation.
- Copy output of any transformation into a sandbox for quick analysis and ad hoc data manipulation.
- Make a snapshot of a table, reload the project with different parameters and compare updated data in the table with the snapshot in order to see the difference.

Sandbox tables start from the **Sandbox** transformation. To send a transformation's output to a sandbox right-click the transformation icon in table and choose *Send to sandbox.*

New transformations

Import from database transformation loads data from a database specified by a connector. The editor of this transformation has a visual query builder. You can also switch to custom SQL if needed. WHERE clause and Custom SQL allow free-form SQL syntax with EasyMorph parameters. Parameters names should be wrapped in curly braces.

Group transformation allows assigning a value (e.g. group name) to rows with values selected in a list. For instance, if column **Color** contains colors, you can assign 'Warm colors' to red, orange and brown colors, making them a group. This transformation is somewhat equivalent to IN operator in SQL syntax. E.g. now you can use condition `[Color group] = 'Warm colors'` which would effectively mean `[Color] IN ('red', 'orange', 'brown')` in SQL syntax.

Split column into rows takes delimited values of a text field and splits them into rows. For example, this table:

Column 1	Column 2
10	AA, BB
20	CC

would be transformed as below (using comma as separator)

Column 1	Column 2
10	AA
10	BB
20	CC

Spreadsheet metadata transformation generates a list of sheet names of a spreadsheet. When it's used with **Import from Excel** transformation and iterations it makes possible importing data from spreadsheets with varying sheet names and number of sheets.

Changes in existing transformations

The changes below don't affect calculations in projects created in older versions, although they are not backward compatible.

Calculate new column transformation now has a new setting – position in table where the new column(s) will be inserted. By default it's *Leftmost*.

Look up and replace transformation has been renamed to **Look up**. Now it allows creating a new column as well as using expressions for calculating a default result.

Import from Excel now supports defining sheet name through a parameter. Together with the new **Spreadsheet metadata** transformation and iterations it makes possible importing data from spreadsheets with varying sheet names and number of sheets.

Import from ODBC transformation now allows using EasyMorph parameters in SQL queries. Parameters names should be wrapped in curly braces. For instance:

```
SQL SELECT * FROM MYTABLE WHERE YEAR = {Year}
```

Double quotes for text literals

Now it's possible to use double quotes to specify text literals in expressions the same way as it works with single quotes. With this change you don't have to remember whether it should be single or double quotes in expressions.

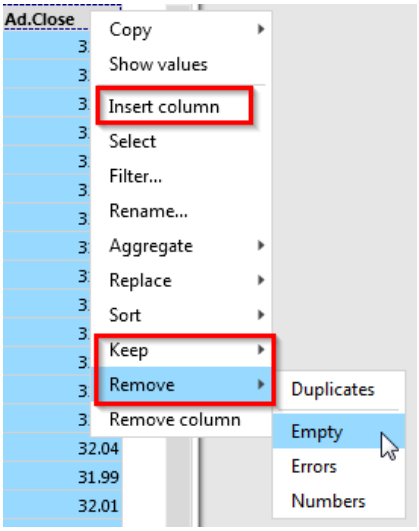
`"Some text"` (notice double quotes) is exactly the same as `'Some text'`.

`upper("Please don't " & 'stop the music')` is a correct expression.

UI improvements

Right-click column menu has got some updates:

- "Insert column" – insert a new empty column right after selected column (it automatically creates **Calculate new column** transformation).
- "Keep" and "Remove" submenus – automatically create **Deduplicate** or **Keep duplicates**, or **Filter by condition** depending on chosen submenu item.
- "Keep duplicates" has been removed from the menu as it's now equivalent to Remove > Duplicates command.



Bugfixes

- Clicking on table cells caused significant delays if **Filter** properties editor was open and it contained a big number of unique values in the list
- **Filter** properties editor didn't save correctly error values
- Undo didn't work properly for some transformations
- Some transformation properties editors automatically applied changes on closing without asking for confirmation