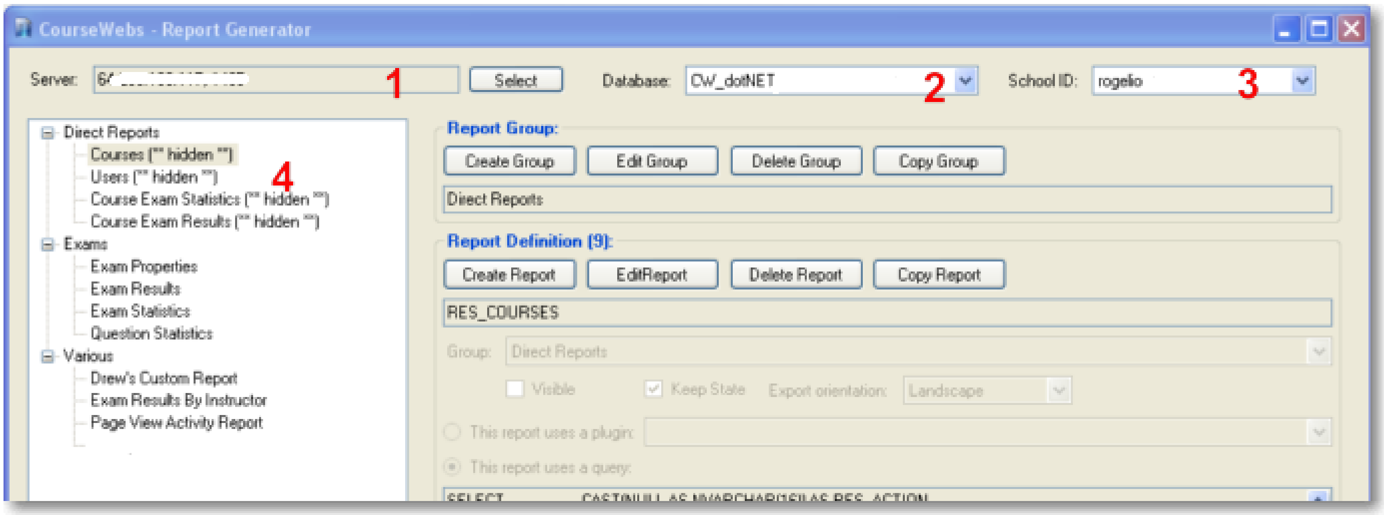
CourseWebs Reporting Tool

Desktop Application Instructions

The CourseWebs Reporting tool is a desktop application that lets a system administrator modify existing reports and create new ones. Changes to existing reports and additional reports show up immediately and do not require the application to be re-compiled. The reporting tool also controls where the reports show up in the application’s menu bar. When creating new reports you will have to supply the SQL statements that the report draws data with.

1. To start

Installation: To install the reporting tool, copy the entire contents of the zip file to a directory of your choice. Run the exe.



Server: 6[redacted]

Database: CW_dotNET

School ID: rogelio

Report Group: Direct Reports

Report Definition: RES_COURSES

Group: Direct Reports

□ Visible ☑ Keep State Export orientation: Landscape

○ This report uses a plugin:

● This report uses a query:

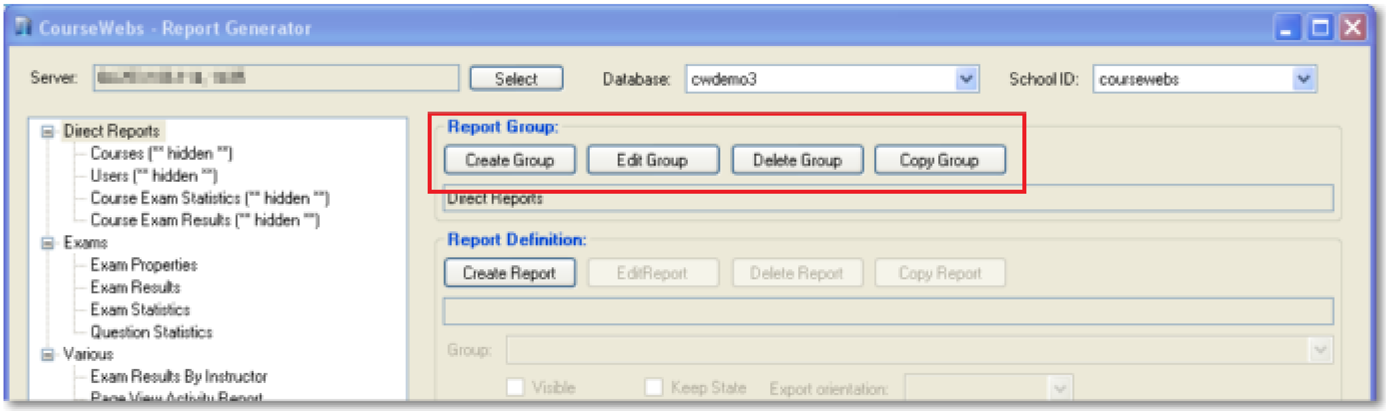
SELECT CAST(NULL AS NVARCHAR(1)) AS RES_ACTION

1. Select the SQL Server that the database is running on.
2. Select the database you want to create/change reports in (this list will only show CourseWebs databases that have the reporting tool capabilities.)
3. Select the School ID you want to create/change reports for.
4. Shows the groups and reports for the selected school in the selected database. Reports and groups are shown here in the exact same order as they will appear in the website. To change that order you can drag and drop the reports and groups around in the treeview. Group names must be unique within a school and report names must be unique within a group (you'll be prompted to change the report name if you drag a report to a group that already contains a report with the same name). You can also move a report from one group to another by dragging it to that other group.

   Reports designated as **hidden** do not show up in the Reports tab. They are accessed by a direct link from CourseWebs, usually from the menu bar. More information below.

2. Create-, edit- and copy report groups.

Selecting a group in the treeview will enable the edit, delete and copy buttons for the group controls:



Server: [redacted]

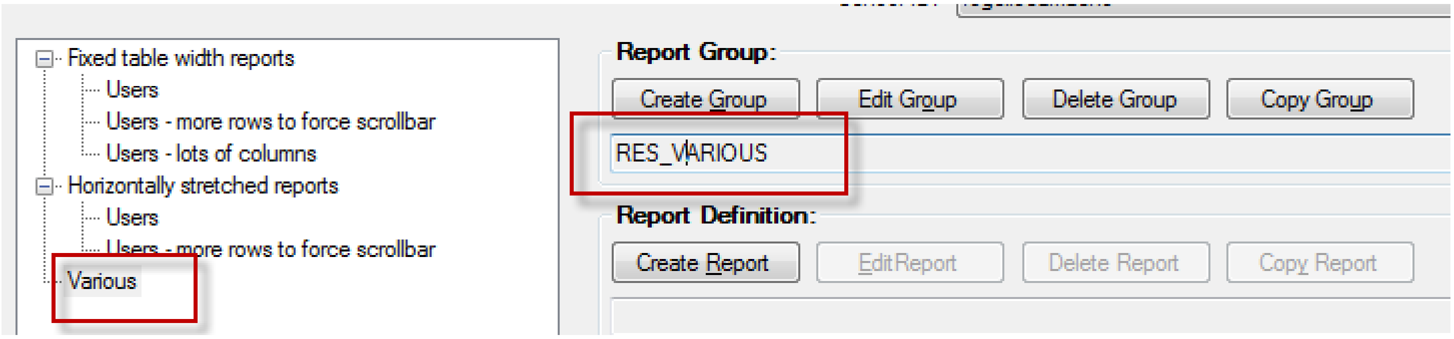
Database: cwdemo3

School ID: coursewebs

Direct Reports

Group:

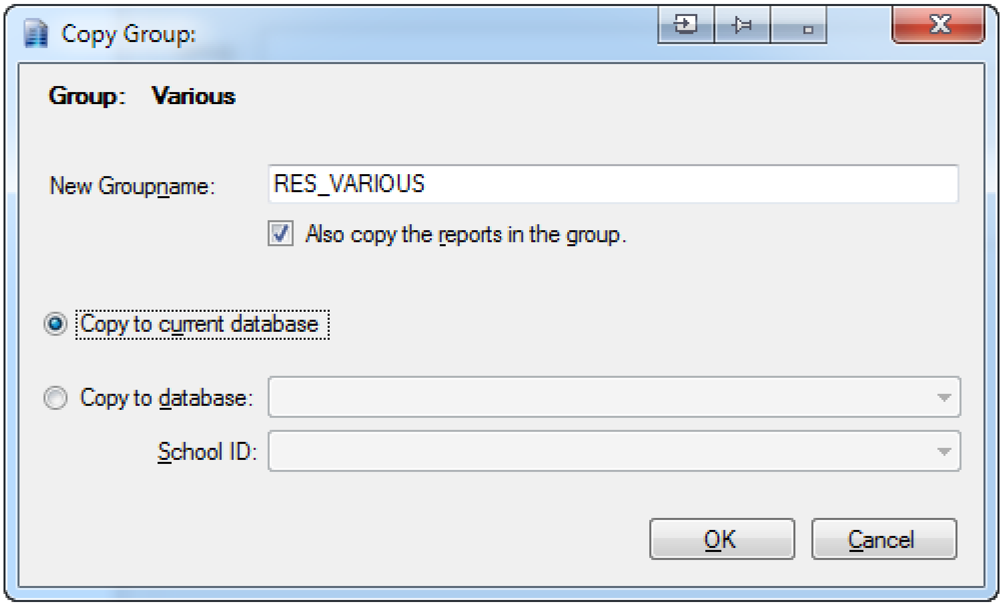
Clicking create or edit will enable the text input box to create a report group or change its name. You can enter resource Id's here instead of plain strings as well:



RES_VARIOUS

You can copy a group within the current database and school id and you can also copy a group to another database that runs on the same SQL server. While copying a group you can opt to create a copy of only the group name, or copy the group name and all of its reports , which is the default.

A group name must be unique within a school, so you'll be prompted if you try to copy a group to a database that already contains a group with the specified name.



Group: Various

New Groupname: RES_VARIOUS

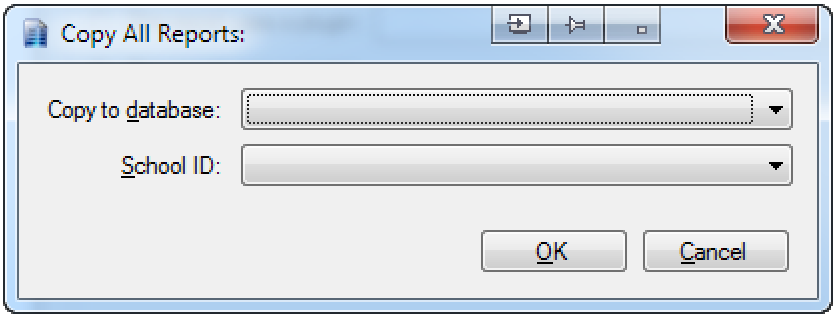
☑ Also copy the reports in the group.

● Copy to current database

○ Copy to database:

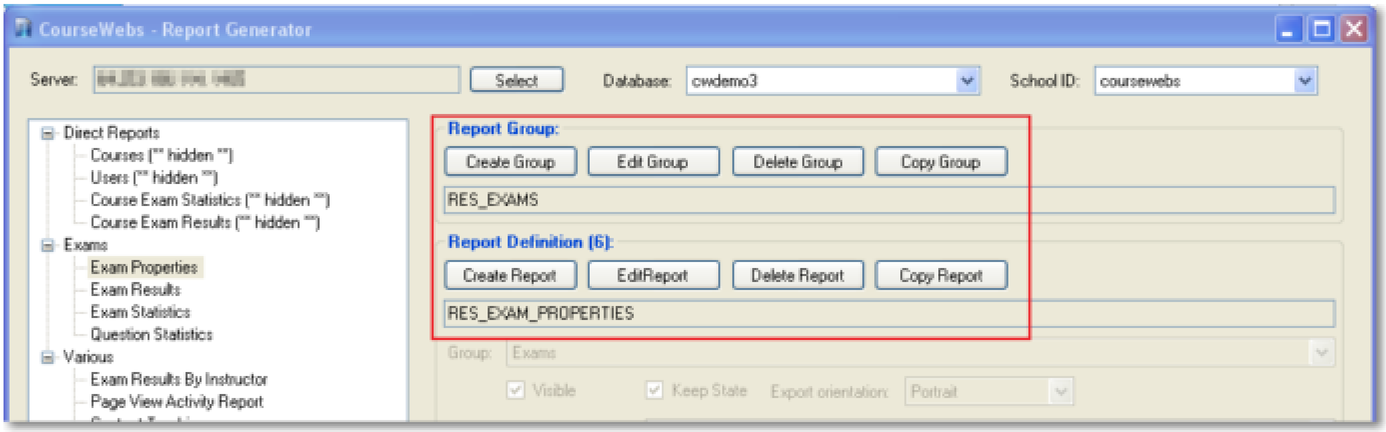
School ID:

With the "Copy All" button below the list of reports you can also copy all groups and reports to another database:



3. Create-, edit- and copy report definitions

Selecting a report in the treeview will enable the edit, delete and copy buttons for both groups as well as reports:



Server: [redacted]

Database: cwdemo3

School ID: coursewebs

RES_EXAMS

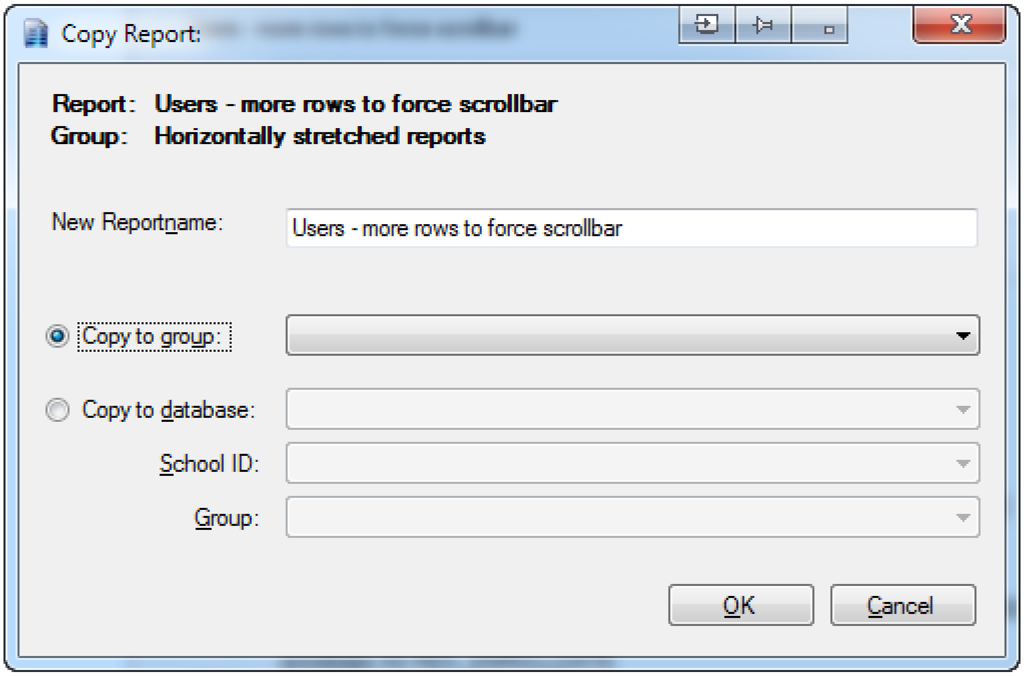
RES_EXAM_PROPERTIES

Group: Exams

☑ Visible ☑ Keep State Export orientation: Portrait

Clicking create or edit will enable the report definition controls:

You can copy a report definition within the current database and school id to another group (or within the same group to another report name) and you can also copy a report definition to a group in another database and school id that runs on the same SQL server:



Report: **Users - more rows to force scrollbar**
Group: **Horizontally stretched reports**

New Reportname: Users - more rows to force scrollbar

● Copy to group:

○ Copy to database:

School ID:

Group:

Creating a report:

1. Fill in the name of the report (again, the name of the report can either be a resource ID or a plain string).
2. Select the group in which to create the report.
3. Specify whether it should be visible and whether the result grid should keep its state:

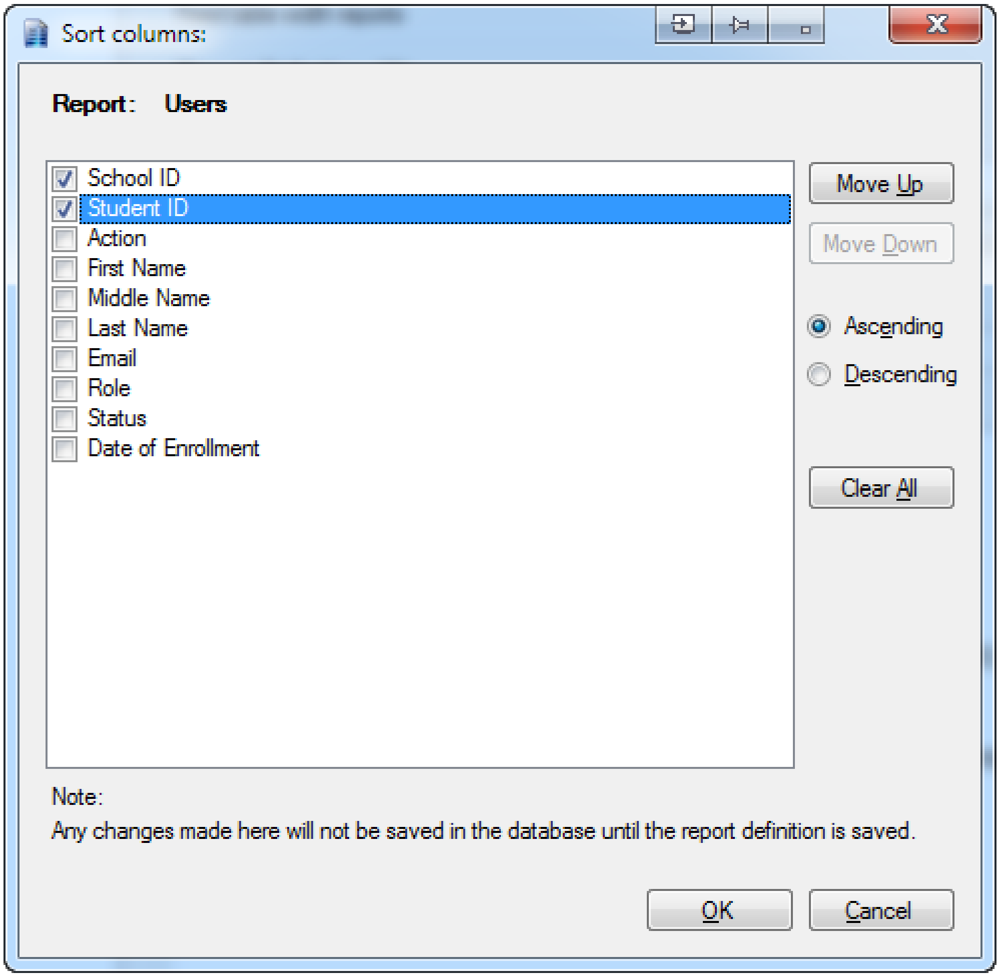
a. Visible reports show on the list of grouped reports in the website.However, reports are also directly accessible, skipping the list with available reports. Should you want to create a report that has its own menu item (such as, the current users list or registrations) you can still create it with the report generator, specify that it should not show up in the list of reports and point the menu item to the report page, specifying the report's ID (which is shown in parentheses after “Report Definition”) in the querystring.
b. The grid can keep its state between page loads. That means that when it is currently showing page 3 and sorted by column X filtered on something Y and you go to another page, it will show the same state as before when returning, instead of page 1 with default sort column and no filter. Situations where this could be useful is when you show a list with edit links to a detail page and when returning from that detail page it would be nice if the list shows what it did before going to that detail page.

4. Specify whether it should use a custom plugin for the report or just a query. Plugins are not implemented yet but will allow more advanced reporting capabilities..
5. Create the query to load the report with. You can either do that in the small text box or press the "Open in external window" button to create it in a full size textbox. That full size textbox accepts tabs and has a different font for better readability of the query, so that's probably where you'd want to work in. The column names (usually aliases) of the query will be used for the text to be displayed in the column header of the result grid. To be able to also show these reports multilingual, those column names can be resource IDs or regular strings (using regular strings keeping in mind the restrictions SQL server puts on column names, of course....). You can also add resource IDs as result values in the query. Taking the query being used in the example it shows a CASE statement containing [RES]RES_STUDENT[/RES]; those "tags" will be replaced with a translation found in the resource file for resource id RES_STUDENT. **Note that you're working live on the destination database here; the textbox will happily accept updates, inserts and deletes, so care should be taken!** Apart from the resource string place holders, the query can contain two other place holders; {SchoolID} and {CourseID}. Those two are case sensitive.
6. You can test the query by pressing "Execute query". That will popup a grid with the results in a similar way that the grid with results will show in the website, except that it can't parse the HTML that might be in the result list and it can't be paged.
7. Save the query. An invalid query can't be saved it; will always be executed before saving to test if the query doesn't generate errors.

The buttons "Sort Columns", "Column Formats", "Column Alignments", "Autostretch Column", "Column Templates", "Invisible Columns" and "Unsortable Columns" will not be enabled while creating a new report. That's because they need the query to be valid in order to show the list of result columns. So, after having saved the query, press the "Edit Report" button to enable those buttons. Note that none of the changes made in any of the screens behind these buttons will be saved to the database until you actually press the "Save" button on the main screen.

1. Sort Columns.

Use this to specify the default sort order of the result set. It is important not to specify an order by in the query but rather let the grid itself handle the sorting. That's because it will show those little sort-glyph images for the columns it is being sorted on.



Sort columns:

Report: Users

- ☑ School ID
- ☑ Student ID
- □ Action
- □ First Name
- □ Middle Name
- □ Last Name
- □ Email
- □ Role
- □ Status
- □ Date of Enrollment

● Ascending

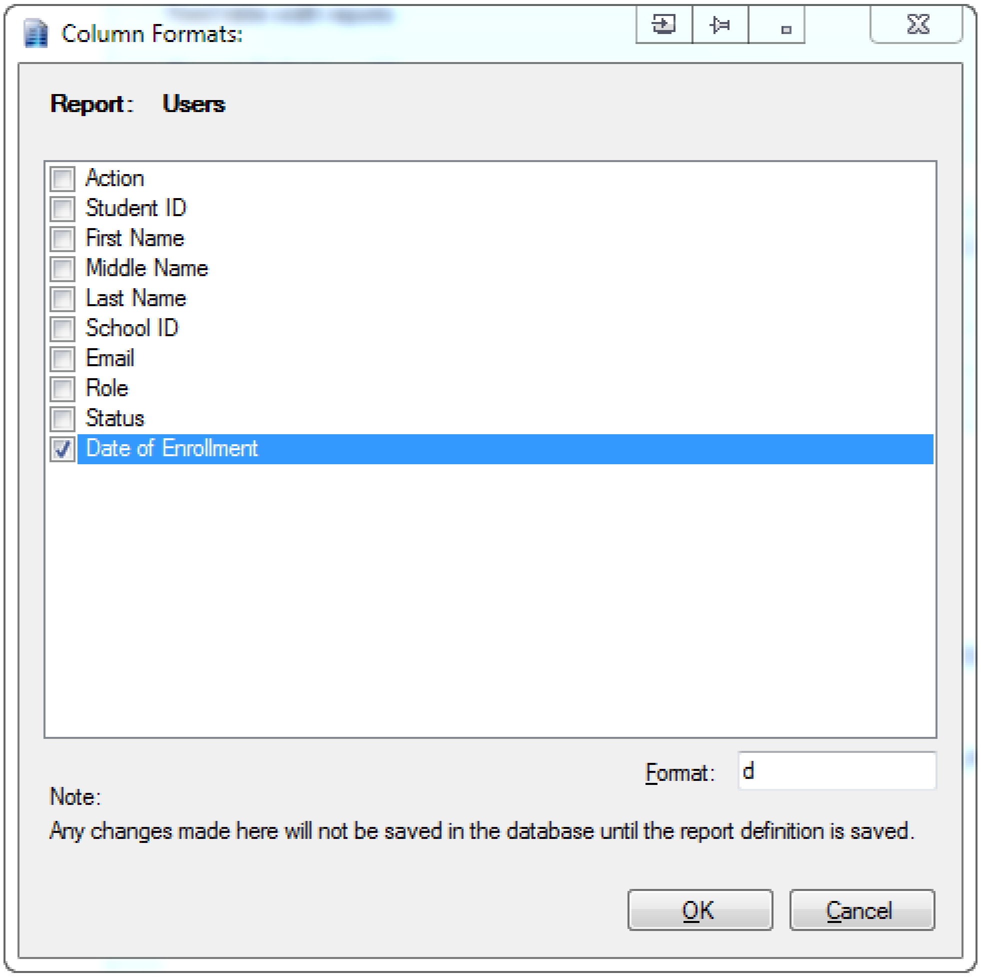
○ Descending

Note:
Any changes made here will not be saved in the database until the report definition is saved.

You can sort on multiple columns, sort ascending or descending and you can set the sort preference (ie; which column to sort on first, which second, etc, etc) by using the "Move up" and "Move down" buttons (the upmost column is the first to be sorted on). Columns that are specified to be sorted on are always at the top of the list and in their order of preference; this list doesn’t show the actual order that the columns will appear in the result set.

2. Column Formats.

Use this to specify a format for the text to be shown in a column. This will be necessary if you want to overrule the default format chosen by either dotnet itself or by the reporting tool.



Column Formats:

Report: Users

☐ Action
☐ Student ID
☐ First Name
☐ Middle Name
☐ Last Name
☐ School ID
☐ Email
☐ Role
☐ Status
☑ Date of Enrollment

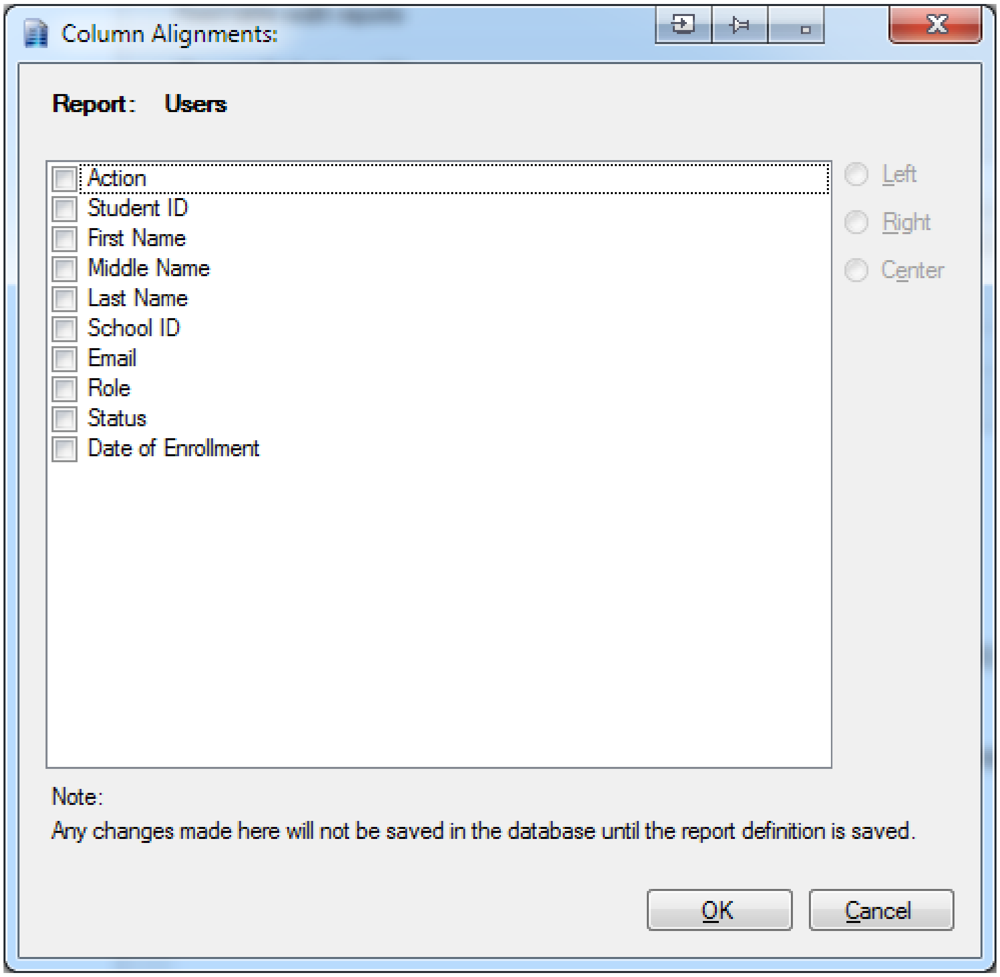
Format: d

Note:
Any changes made here will not be saved in the database until the report definition is saved.

The format string should be a valid dotnet format string and those depend on the datatype of the column (ie, date columns accept different format strings than number columns). Given the wide variety of data types and their format strings, there's no check on whether the format string is valid; it will just show an error when executing the query in the preview screen. Here's a list of format strings.

3. Column Alignments.

Use this to overrule the default alignment of columns:



Column Alignments:

Report: Users

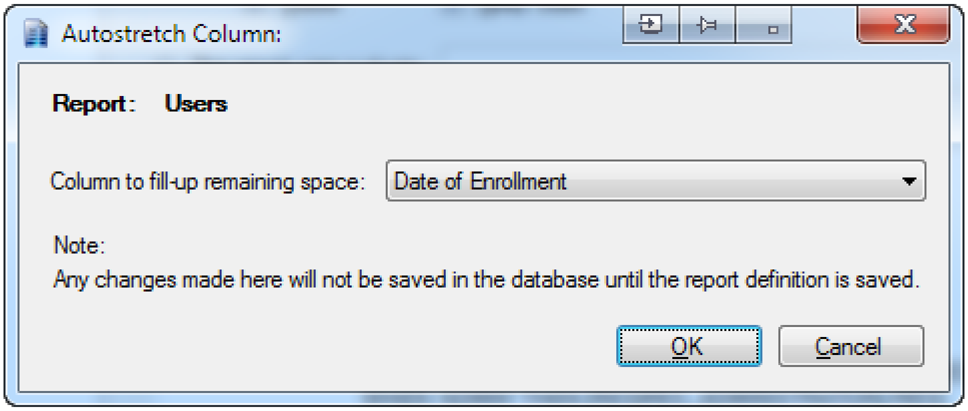
□ Action
□ Student ID
□ First Name
□ Middle Name
□ Last Name
□ School ID
□ Email
□ Role
□ Status
□ Date of Enrollment

○ Left
○ Right
○ Center

Note:
Any changes made here will not be saved in the database until the report definition is saved.

Numbers are aligned right and everything else left. Should you want a left aligned number, or a centered or right aligned string then you need to specify that here.

4. Autostretch Column.
Here you can specify the column that you want to have stretched to fill any remaining space on the page:



Autostretch Column:

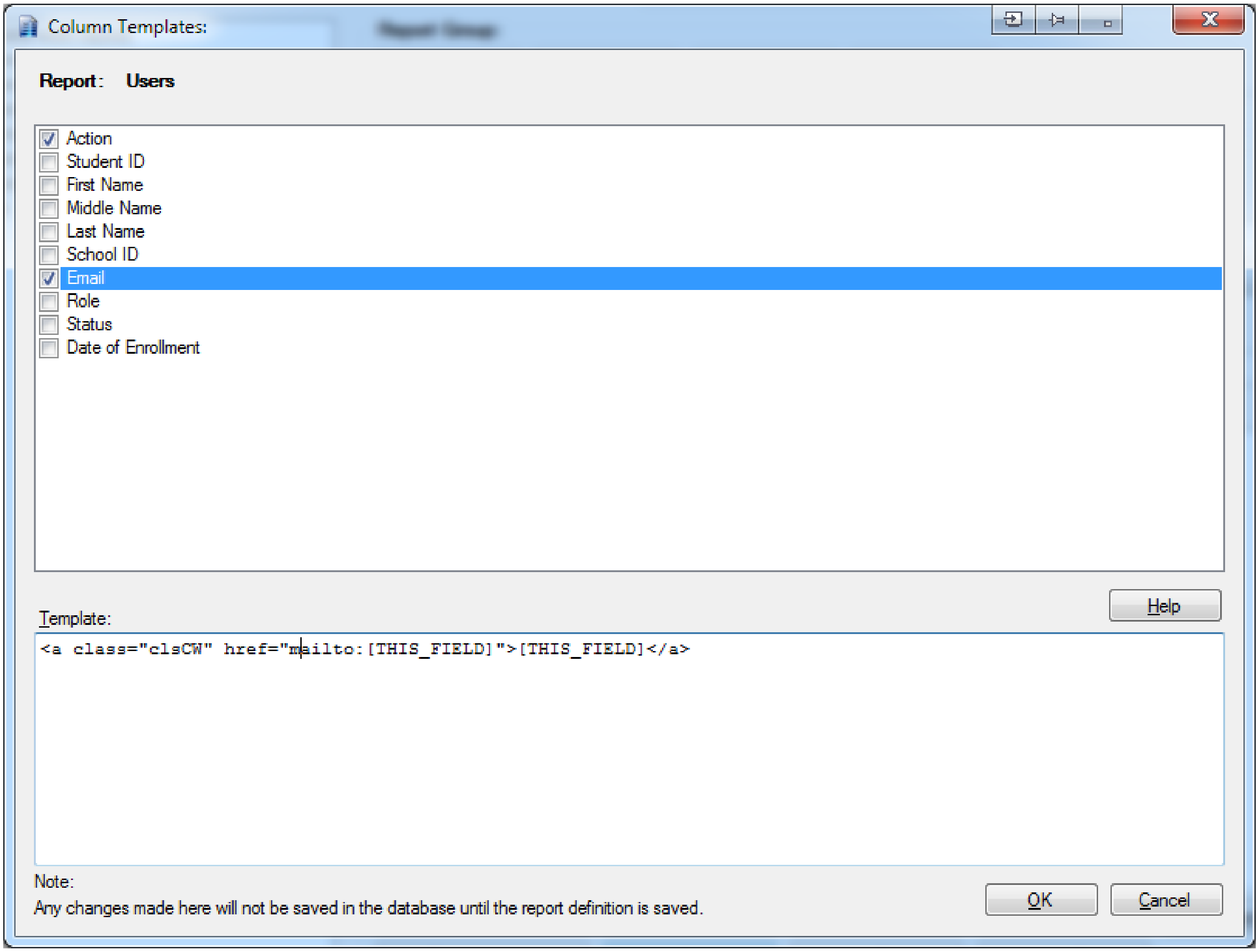
Report: Users

Column to fill-up remaining space: Date of Enrollment

Note:
Any changes made here will not be saved in the database until the report definition is saved.

The result grid always shows the entire contents of a column horizontally. If you have a lot of columns or wide columns (or just a small screen), it will start scrolling horizontally. However, if you have a grid with just a few columns, it won't fill up the screen horizontally because the total length of the columns is less than the screen width. If you want the grid to fill up the remaining space then you can specify a column here to do that; that column will always be at least wide enough to show its contents and it will stretch out to push the right border of the grid to the right border of the screen if possible.

5. Column Templates.
This is probably the most difficult one, but also the most important one. Here you can specify a template string that will be used to show in the grid. This template string can contain any text, HTML and some place holders:



Column Templates: [redacted]

Report: Users

- ☑ Action
- □ Student ID
- □ First Name
- □ Middle Name
- □ Last Name
- □ School ID
- ☑ Email
- □ Role
- □ Status
- □ Date of Enrollment

Template:

```
<a class="clsCW" href="mailto:[THIS_FIELD]">[THIS_FIELD]</a>
```

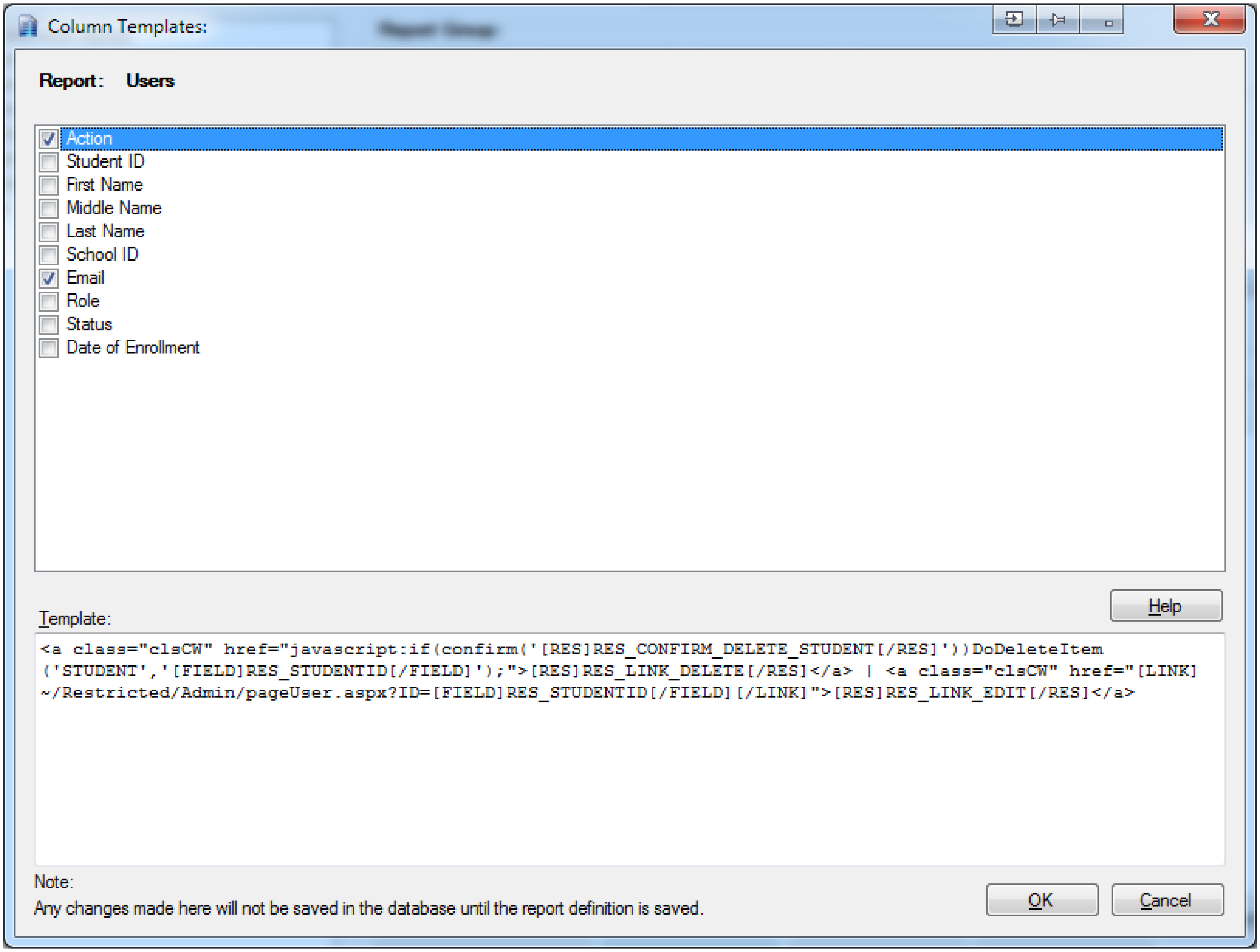
Note:
Any changes made here will not be saved in the database until the report definition is saved.

The above screenshot contains the HTML to show a clickable email link in the grid. Everything in that template will be copied directly over to the result grid, except that the place holders will be replaced with what they're holding place for. You can use the following placeholders:

- [RES][/RES]: Specifies a resource ID
- [RES_QS][/RES_QS]: Specifies a resource ID; result will be querystring-encoded.
- [FIELD][/FIELD]: Specifies the name of a column in the result set. This tag will be replaced with the value found in that column for the current row. If that column has a format specified then it will be replaced with the formatted value.
- [LINK][/LINK]: Dotnet-absolute path to a webpage, which will be transformed to a relative path in the result set.
- [LINK_QS][/LINK_QS]: Path to a webpage; will be querystring-encoded.
- [THIS_FIELD]: Will be replaced with the (formatted, if specified) value of this column. Templates that contain this place holder will output an empty string to the result set if the value of the current row is empty in the column that this template has been specified for, because in that case the resulting string would be useless anyway.
- [REPORT_ID]: The row ID of the current report. Could be useful to pass along to other pages in the querystring.
- [THIS_REPORT]: Creates a querystring-encoded link back to the current report.

The above example is rather straight-forward. It specifies a html email link with the [THIS_FIELD] place holder. Given that this is the template for the email column, it means that the email address will be put in that link. And, if the column is empty for the current row, it will show nothing in the resultset.

The following example is a little more complicated:



Column Templates:

Report: Users

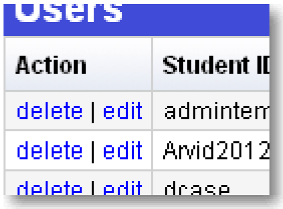
☑ Action
□ Student ID
□ First Name
□ Middle Name
□ Last Name
□ School ID
☑ Email
□ Role
□ Status
□ Date of Enrollment

Template:

```
<a class="clsCW" href="javascript:if(confirm('[RES]RES_CONFIRM_DELETE_STUDENT[/RES]'))DoDeleteItem
('STUDENT','[FIELD]RES_STUDENTID[/FIELD]');">[RES]RES_LINK_DELETE[/RES]</a> | <a class="clsCW" href="[LINK]
~/Restricted/Admin/pageUser.aspx?ID=[FIELD]RES_STUDENTID[/FIELD][/LINK]">[RES]RES_LINK_EDIT[/RES]</a>
```

Note:
Any changes made here will not be saved in the database until the report definition is saved.

This shows two links, separated by a pipe character:

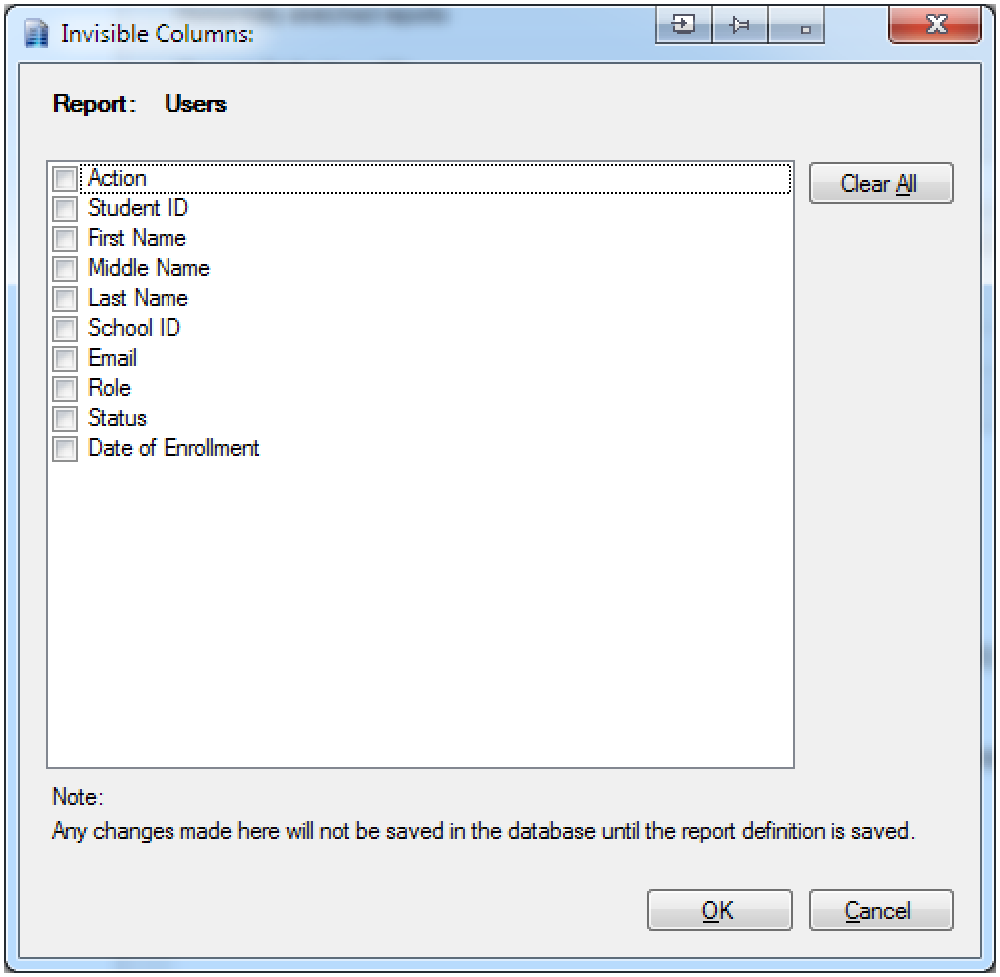


Users

Action	Student I
delete | edit	adminter
delete | edit	Arvid2012
delete | edit	dcase

It uses the [RES] place holders to show the javascript confirmation question and to show the "delete" and "edit" text of the links. It uses the [FIELD] place holder to pass the value found in the "RES_STUDENT" column on to the javascript delete function and to put it in the "ID" querystring of the [LINK] place holder that specifies the absolute path to the page where the profile can be edited and which will convert it to a path relative to the report page. Note that such columns show up weird in the query results window of the desktop app because it will just show the html that would be outputted by the report on the website.

6. Invisible Columns.
Those are useful if you have a template specified for column X and in that template you have included a [FIELD] place holder pointing to column Y, but you don't want column Y to be a column in the result set; the value will still be available in the result set while parsing the template of column X, but it won't show up itself as a separate column in the grid.



Invisible Columns:

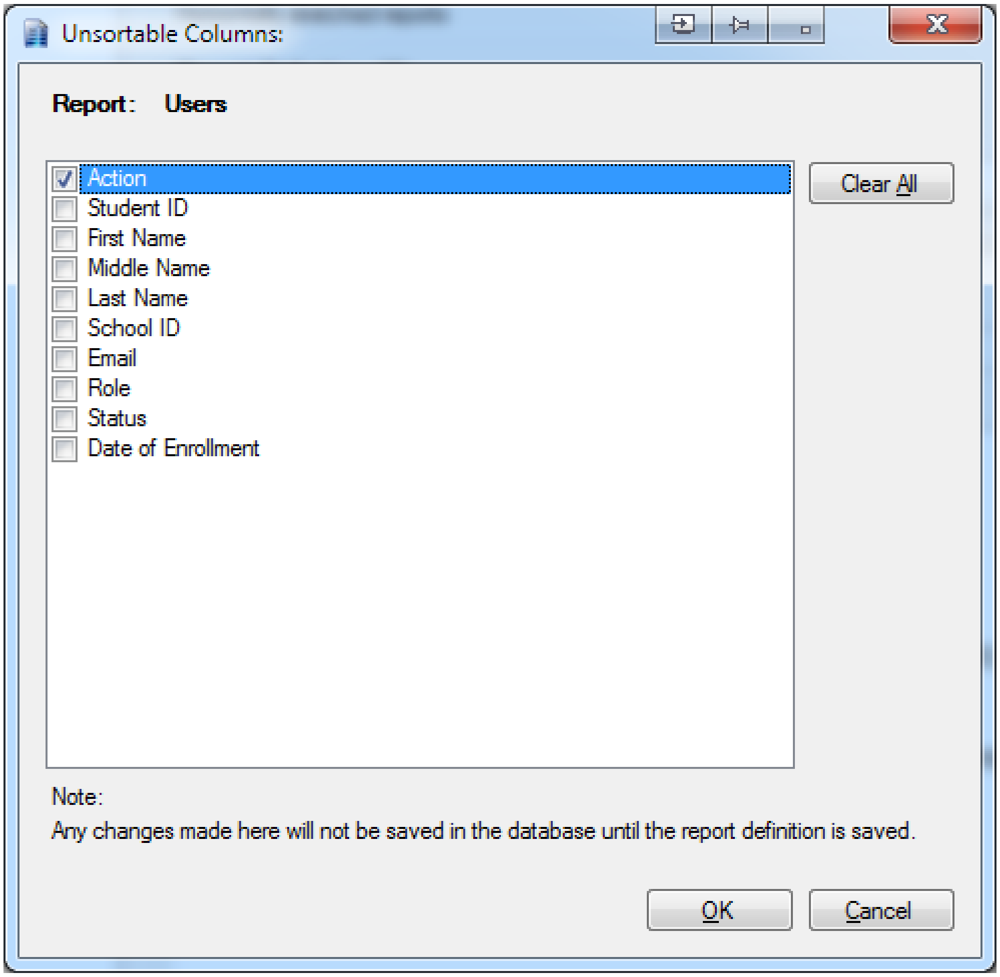
Report: Users

- □ Action
- □ Student ID
- □ First Name
- □ Middle Name
- □ Last Name
- □ School ID
- □ Email
- □ Role
- □ Status
- □ Date of Enrollment

Note:
Any changes made here will not be saved in the database until the report definition is saved.

7. Unsortable columns

If there's a column in the grid that you don't want the user to be able to sort upon, or if there's a column that causes problems when being sorted upon, you can mark it here as "not able to be sorted upon":



Report: Users

☑ Action
□ Student ID
□ First Name
□ Middle Name
□ Last Name
□ School ID
□ Email
□ Role
□ Status
□ Date of Enrollment

Note:
Any changes made here will not be saved in the database until the report definition is saved.

Note that the grid will sort on the actual data found in the result set returned by the query. If there are column format or column templates active, sorting might result in unexpected results, because those are NOT part of the result set returned by the query.
For instance the "Action" column. That column contains a whole bunch of HTML and text in the result set (although you only SEE "delete | edit", of course). However, in the query it is specified as NULL. That's because it has to be in the query in order to show the column in the result set, but it can never contain any value of meaning in the actual dataset being returned by the query. So, should you allow sorting on that column, it would sort a dataset on a column that contains NULL in every row. The result would be unexpected. Especially if, instead of showing "delete | edit" it would contain some value from another column, say "First name"; you would sort on a column that doesn't actually contains that value being displayed in the data set and the result would be a completely random list that you THINK is sorted on first name, because that's what you see in the grid, but in fact shows them as completely randomly ordered.
In contrast, the email column: that column also contains a whole bunch of HTML, yet it only shows the email address in the grid. And that address is what's in the data set returned by the query so sorting upon that column does exactly what you would expect it to do. Same for the date column. It is projected in the grid as a string, but the data set contains an actual date, so sorting on that column results in what one would expect when one sorts a date column, even though it shows as a formatted string.

4. A few notes

- The Reporting Tool can't read customized resources created in the extra xml file that the web app uses to read those. That means that the desktop app will show just the resource ID while the web app will show the correct translation.
- Groups without visible reports will not show in the website.