Techniques for “Anchoring” Subreport Data

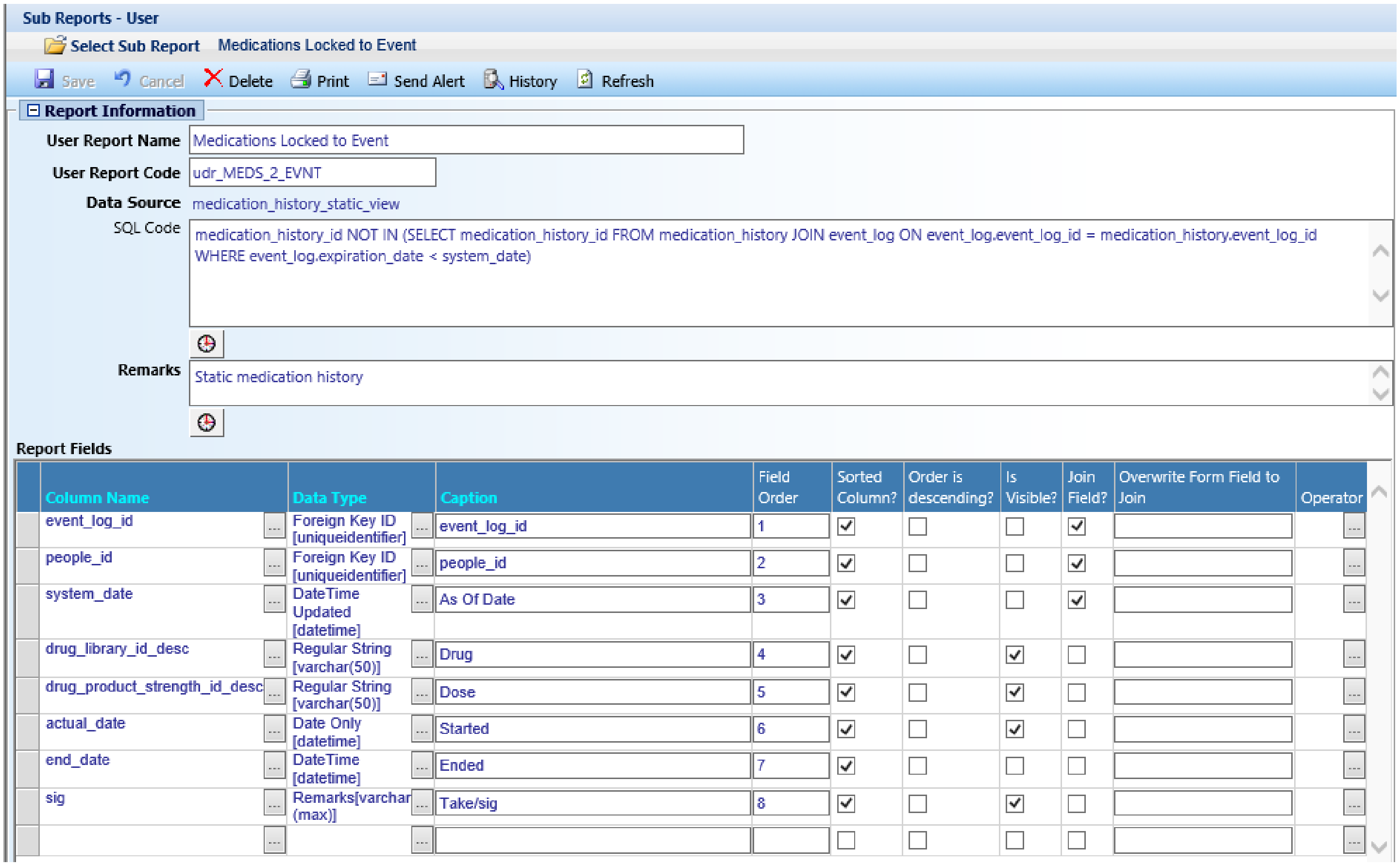
Subreports are a great tool for displaying myEvolv data on forms, but there are times when you want the data that displays on the subreport to stay static. Subreports re-generate on each form open based on the rules setup when you configure the subreport, so if you do not design them carefully, you may end up seeing information other than what was originally on the form when an event was added.

A good example of this is diagnoses on an intake assessment. Intake staff may enter a set of diagnoses into myEvolv for the new client and then within 24 hours, a clinician may perform an intake assessment that displays those diagnoses in a subreport. The assessment that the clinician makes is based on the diagnoses listed there. Over the course of a year of treatment at the agency, the client ends up getting additional diagnoses added to their record. When an auditor reviews the intake assessment a year later, the auditor now sees all diagnoses on record, including those added after the intake assessments was completed and may wonder why the new diagnoses were not accounted for in the assessment. By using the techniques below we can keep the information on the subreport looking the same as it did on the day it was created.

Technique 1 – Special View Fields

Occasionally, a sub report data source contains a column that was likely meant for the exact purpose of anchoring the subreport data to the parent form.

One example of this is the **medication_history_static_view**. This subreport view contains a column system_date which when used as a join field will effectively filter the subreport listing to only show the medications that were on file for the client at the time that the parent event was created.



Select Sub Report Medications Locked to Event

User Report Name: Medications Locked to Event

User Report Code: udr_MEDS_2_EVNT

Data Source: medication_history_static_view

SQL Code:

```
medication_history_id NOT IN (SELECT medication_history_id FROM medication_history JOIN event_log ON event_log.event_log_id = medication_history.event_log_id
WHERE event_log.expiration_date < system_date)
```

Remarks: Static medication history

Report Fields

Column Name	Data Type	Caption	Field Order	Sorted Column?	Order is descending?	Is Visible?	Join Field?	Overwrite Form Field to Join	Operator
event_log_id	Foreign Key ID [uniqueidentifier]	event_log_id	1	☑	☐	☐	☑		
people_id	Foreign Key ID [uniqueidentifier]	people_id	2	☑	☐	☐	☑		
system_date	DateTime Updated [datetime]	As Of Date	3	☑	☐	☐	☑		
drug_library_id_desc	Regular String [varchar(50)]	Drug	4	☑	☐	☑	☐		
drug_product_strength_id_desc	Regular String [varchar(50)]	Dose	5	☑	☐	☑	☐		
actual_date	Date Only [datetime]	Started	6	☑	☐	☑	☐		
end_date	DateTime [datetime]	Ended	7	☑	☐	☐	☐		
sig	Remarks[varchar (max)]	Take/sig	8	☑	☐	☑	☐		
				☐	☐	☐	☐		

Unfortunately, these types of special views and columns are rare.

Another type of special field you might find is one that has a calculated end date. Some events in myEvolv are considered active because they have a null end date. This can make it difficult to anchor a subreport because filtering a subreport only on whether the end_date is NULL or NOT NULL will lead to situations where down the road, what lists in the subreport will change as those events become inactive.

A few sub report views, such as the **benefits_assignment view**, have calculated fields that help you work around that issue. The benefits_assignment_view has a column called end_date_adjusted. This column has the value of the event's end_date if it has an end date. If the event has a null end_date, then end_date adjusted has the date 12/31/2100 filled in.

You can use the end_date_adjusted field as a join column on your subreport with a Greater Than (or Equal To) operator and avoid the problem of a null value for end_date because even if the end_date ends up being added later, it will still be greater than the date on the parent form.

Other views with an end_date_adjusted field include:

problems_view, worker_assignment_expanded_view, primary_worker_assignments_view, incident_restraint_view, ppg_history_view, emplpyment_history_view, program_modifier_enrollment_view

If you need data contained in a view that does not have one of these fields and you run into this issue, see Technique 3 for another approach.

A third special field type that might pop up in a view is a different kind of calculated date. These usually end up being things like a calculated column that holds the date of the first of the month or week that the event's actual date took place in.

If you can use one of these views, you can join that column to a column with that same date on the parent form. These are good for monthly summary types of events or weekly summary types of events.

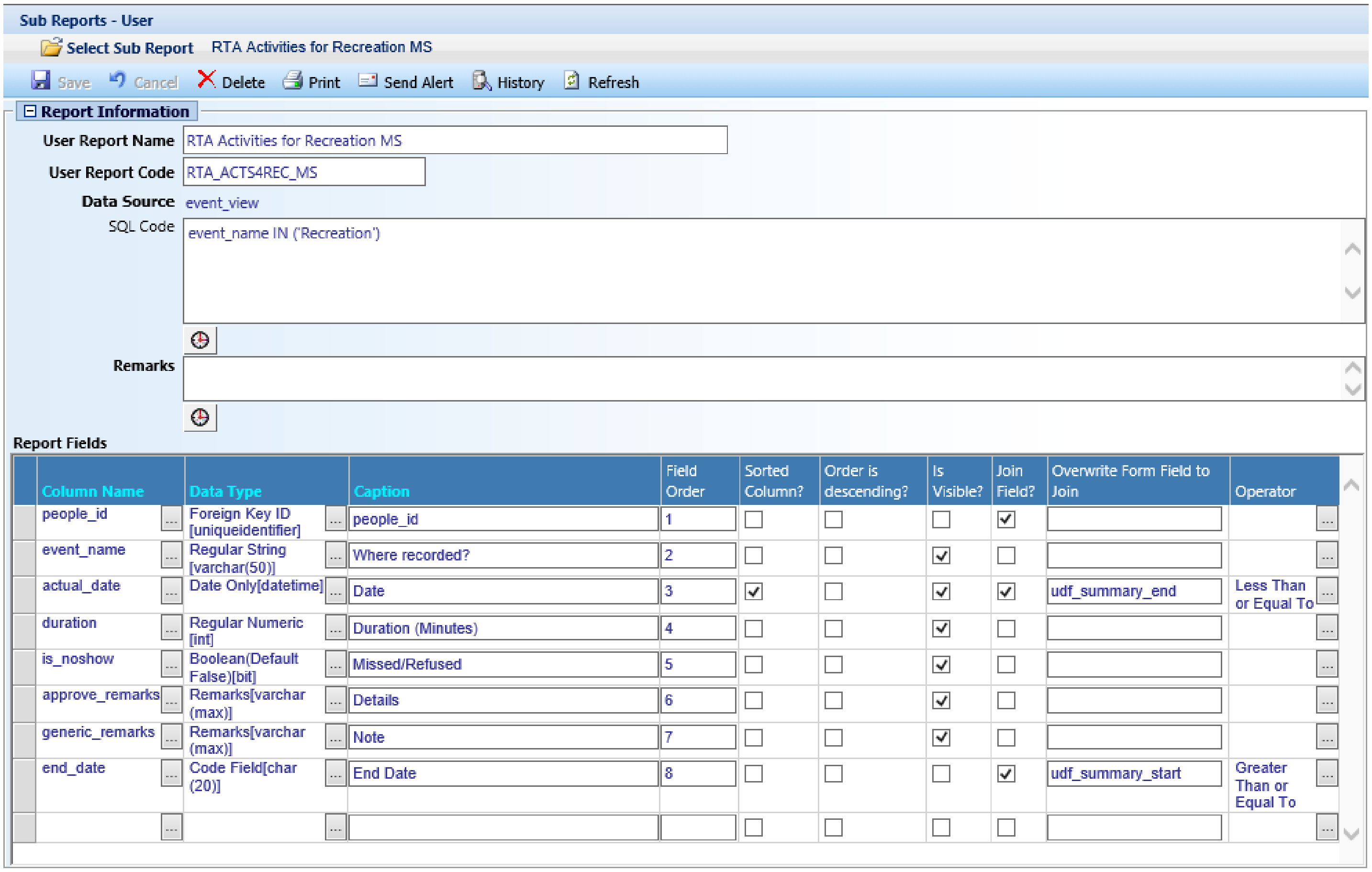
These are some of the views that include a month_start column: **tests_events_expended_view, payroll_details_view, missing_authorizations_view, event_expanded_view, event_view, service_with_notes_and_other_act_view**.

These are some of the views that include a calendar_week_start or week_start: **staff_payroll_weekly_view, attendance_details_view, service_details_view, tests_events_expended_view, services_not_billed_view, events_linked_to_goals_view, event_expanded_view, events_linked_to_goals_weekly_view, event_view**.

Technique 2 – Between Two Dates

This technique works well for situations where you have a start and end date on the parent form that you want to use to constrain the date range of items listing in the sub report. For example, if you have a monthly summary type of event and want to use a subreport to list the events between those two dates. The trick here is to find a sub report view that has enough dates to work with. For these you will need two date columns, typically actual_date and end_date or date_entered to use for join columns because you can only use a column once in the subreport.

This example uses the **event_view**.



Select Sub Report: RTA Activities for Recreation MS

User Report Name: RTA Activities for Recreation MS

User Report Code: RTA_ACTS4REC_MS

Data Source: event_view

SQL Code: event_name IN ('Recreation')

Remarks:

Report Fields

Column Name	Data Type	Caption	Field Order	Sorted Column?	Order is descending?	Is Visible?	Join Field?	Overwrite Form Field to Join	Operator
people_id	Foreign Key ID [uniqueidentifier]	people_id	1	☐	☐	☐	☑		
event_name	Regular String [varchar(50)]	Where recorded?	2	☐	☐	☑	☐		
actual_date	Date Only[datetime]	Date	3	☑	☐	☑	☑	udf_summary_end	Less Than or Equal To
duration	Regular Numeric [int]	Duration (Minutes)	4	☐	☐	☑	☐		
is_noshow	Boolean(Default False)[bit]	Missed/Refused	5	☐	☐	☑	☐		
approve_remarks	Remarks[varchar (max)]	Details	6	☐	☐	☑	☐		
generic_remarks	Remarks[varchar (max)]	Note	7	☐	☐	☑	☐		
end_date	Code Field[char (20)]	End Date	8	☐	☐	☐	☑	udf_summary_start	Greater Than or Equal To
				☐	☐	☐	☐		

The parent event in this example has two user_defined fields on it to capture the summary start and summary end dates. The subreport uss the Overwrite Form Field to Join property to use those fields instead of the actual_date and end_date fields on the parent form.

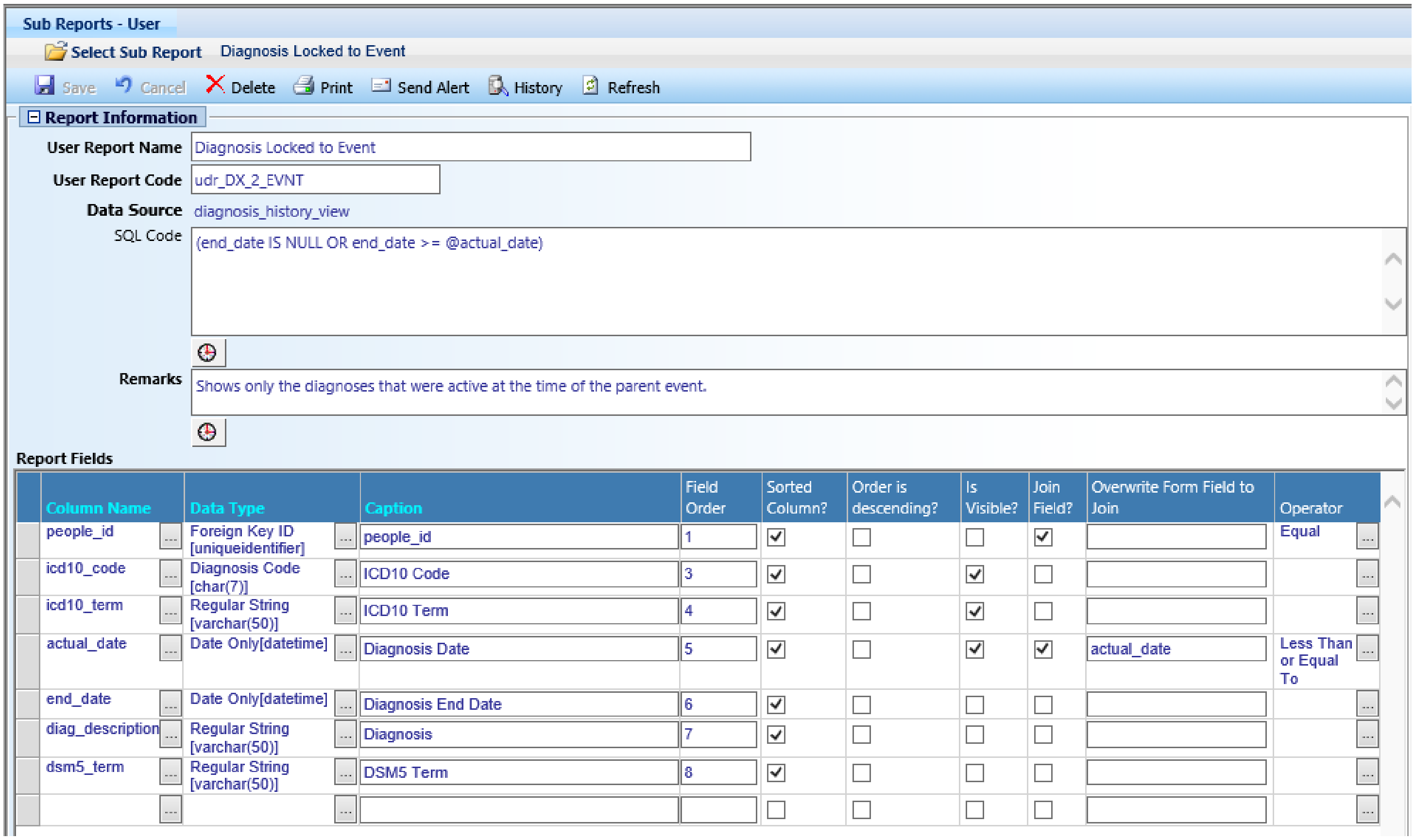
By using the operators of Less Than or Equal To and Greater Than or Equal To, the subreport only pulls in the Recreation events between those two dates on the parent form.

Technique 3 – Parameters in SQL Code

NOTE: This technique breaks subreports in NX as of version 10.1.0150

This is a technique that opens up a lot of possibilities because it allows you to use parameters in the SQL Code property of the subreport. One of the things that can be frustrating in trying to anchor a subreport is that some kinds of events do not have end_date values if they are active. This is the case with things like diagnoses. So if you want to only show the diagnoses that were current or active (null end_date) without filtering out diagnoses that were subsequently ended (now NOT null end_date), you might find yourself stuck.

This example uses the diagnosis_history_view to solve the problem outlined above. It will pull in the diagnoses that were active at the time of the event, which means any diagnosis that has a start date prior to the actual_date of the parent event and has either a null end_date or an end_date that is after the actual date of the parent event. This way, if a diagnosis is ended after the parent event, it does not disappear from the subreport for having a NOT NULL end_date.



Select Sub Report: Diagnosis Locked to Event

User Report Name: Diagnosis Locked to Event

User Report Code: udr_DX_2_EVNT

Data Source: diagnosis_history_view

SQL Code:

```
(end_date IS NULL OR end_date >= @actual_date)
```

Remarks: Shows only the diagnoses that were active at the time of the parent event.

Report Fields

Column Name	Data Type	Caption	Field Order	Sorted Column?	Order is descending?	Is Visible?	Join Field?	Overwrite Form Field to Join	Operator
people_id	Foreign Key ID [uniqueidentifier]	people_id	1	☑	□	□	☑		Equal
icd10_code	Diagnosis Code [char(7)]	ICD10 Code	3	☑	□	☑	□		
icd10_term	Regular String [varchar(50)]	ICD10 Term	4	☑	□	☑	□		
actual_date	Date Only[datetime]	Diagnosis Date	5	☑	□	☑	☑	actual_date	Less Than or Equal To
end_date	Date Only[datetime]	Diagnosis End Date	6	☑	□	□	□		
diag_description	Regular String [varchar(50)]	Diagnosis	7	☑	□	□	□		
dsm5_term	Regular String [varchar(50)]	DSM5 Term	8	☑	□	□	□		
				□	□	□	□		

The following has worked for me, but this is not documented anywhere so this is all speculative on my part.

In order to use the parameter (@actual_date), you must use the 'Overwrite Form Field to Join' property for the column, even if you overwrite it with the same column it would have used without it. You must also therefore select an 'Operator'. You then have a parameter with the name of the <u>'Column Name'</u> to Join with the '@' appended that can be used in your SQL Code.