I have a proposed solution, but it has some caveats that have been highlighted by Lars and Jason before, [namely here](https://lists.launchpad.net/dhis2-devs/msg30976.html) and [in this thread already](https://lists.launchpad.net/dhis2-devs/msg33790.html). I’m using the word Organizational Unit below to follow the convention in DHIS2, but it is interchangeable with ‘Facilities’ or ‘Health Facilities’ or ‘Aid Posts’.

# Assumptions and Caveats

* Organizational units are never ‘reopened’, there is only one each of 'openingdate' and 'closeddate' so that is about all we can do
* Datasets are never removed from an organizational unit even if they don’t submit them anymore, keeps historical reporting rates accurate.
* Organizational units are responsible for submitting a completed datasets for partial time periods (E.G. Org opens on Jan 14th, they are still responsible for January’s completed dataset (if the dataset's frequency is monthly))

# The Math

Since I haven’t done Java in forever, I don’t feel qualified to even play with the source. All my math is done in Postgres, which comes with a lot of limitations, but it allows anyone with any version to use it through the ‘SQL View’ module.

Max Number of Completed Datasets Expected for Opened Organizational Unit (number of timeperiods in the report's duration, e.g. 12 for monthly frequency in yearly report)

Less Excluded Completed Datasets due to Opening Date

Less Excluded Completed Datasets due to Closing Date

Equals Expected Completed Datasets for Organizational Unit

E.G. Report is 2014 Reporting Rates for a Monthly Dataset. Organizational Unit opens February and closes in July. The math is 12(Jan to Dec) – 1(Jan) – 5(Aug to December) = 6 Expected Reports

This is run and summed with all organizational units subscribed to the dataset in question. Currently line one is already implemented inside DHIS2, so I am assuming that information is readily computable either in Java or from the Database. Lines two and three have yet to be implemented. I’m doing it this way because I can actually fit all of this math into a Postgres SELECT query without trouble, so no need for a stored procedure.

# The Math in Query form

The number 12 is just because I know how many time periods are in a year for a dataset with a monthly frequency.

12 - CASE WHEN ouf.openingdate IS NULL THEN integer '12'

WHEN ouf.openingdate > TO\_DATE('2015/01/01', 'YYYY/MM/DD') THEN integer '12'

WHEN ouf.openingdate < TO\_DATE('2014/01/01', 'YYYY/MM/DD') THEN integer '0'

WHEN ouf.openingdate >= TO\_DATE('2014/01/01', 'YYYY/MM/DD')

THEN (CAST(TO\_CHAR(ouf.openingdate, 'MM') AS integer) - 1)

END - CASE WHEN ouf.closeddate IS NULL THEN integer '0'

WHEN ouf.closeddate > TO\_DATE('2015/01/01', 'YYYY/MM/DD') THEN integer '0'

WHEN ouf.closeddate < TO\_DATE('2014/01/01', 'YYYY/MM/DD') THEN integer '12'

WHEN ouf.closeddate >= TO\_DATE('2014/01/01', 'YYYY/MM/DD')

THEN 12 - CAST(TO\_CHAR(ouf.closeddate, 'MM') AS integer)

END AS expectedreps

That looks really busy to me so I will pair it down a bit.

[a] = Time periods in report duration based on dataset frequency.

[b] = Report starting date

[c] = Report ending date

[a] –

CASE

WHEN opendate IS NULL THEN [a] Should never happen, there just in case, remove all

WHEN opendate > [c] THEN [a] Opened after this report, remove all expected datasets

WHEN opendate < [b] THEN 0 Opened before this report starts, remove no datasets

\*\*WHEN opendate >= [b] THEN [a] – opendate Opened sometime this report, remove datasets not covered by opening date

END –

CASE

WHEN closedate IS NULL THEN 0 Not closed, remove no datasets

WHEN closedate > [c] THEN 0 Closes after this report, remove none

WHEN closedate < [b] THEN [a] Closes before this report, remove all

\*\*WHEN closedate >= [b] THEN [a] – closedate Closed sometime in this report, remove datasets not covered by close date

END

The \*\* portion above is the hardest part I think. In our case it is straight forward, I just cast the month number as an integer, but what about a dataset with a weekly frequency inside a quarterly report?

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Daily | Weekly | Monthly | Bimonthly | Quarterly | Six-Monthly | Yearly |
| Daily | 1 | 7 | X | X | X | X | X |
| Weekly |  | 1 | X | X | X | X | X |
| Monthly |  |  | 1 | 2 | 3 | 6 | 12 |
| BiMonthly |  |  |  | 1 | X | 3 | 6 |
| Quartly |  |  |  |  | 1 | 2 | 4 |
| Six-Monthly |  |  |  |  |  | 1 | 2 |
| Yearly |  |  |  |  |  |  | 1 |

Oranges are just a date function, e.g. how many days are in February 2009?

Reds are complicated… I don’t have a solution in every case, but I had an idea to use the timeperiod table with something like the following:

SELECT COUNT(\*) AS “Number of Reports”

FROM timeperiod

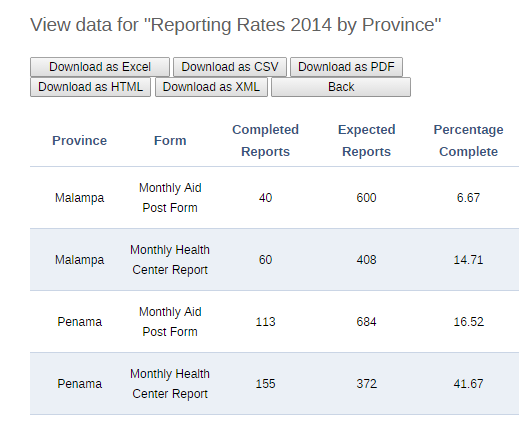
WHERE enddate > [org’s start date]

AND enddate < [org’s end date]

AND periodtypeid = [dataset’s period type]

Thankfully, all that is an aside in Vanuatu, because there will always be 12 expected dataset completions in a yearly report, and 1 in a monthly report. We’ve already implemented this via an SQL view here to good effect. I hope this helps somebody in the group.

Final Result



Note that if there are future time periods, you will have to divide the result by (how far into report duration you are / total expected reports)

So for Penama Province, Monthly Health Center Report, you get 41.67 / (10 / 12) or about 50% reporting rate so far in 2014.

Going forward, it would be nice to address the issue of facilities that close and reopen. I have some ideas I’m kicking around to do this, but all will require changes to the java, gui, and database.

The full query for our implementation is included below. Cheers!

SELECT oup.name AS "Province",

ds.name AS "Form",

SUM(cr.completedreports) AS "Completed Reports",

SUM(12 - CASE WHEN ouf.openingdate IS NULL THEN integer '12'

WHEN ouf.openingdate > TO\_DATE('2015/01/01', 'YYYY/MM/DD') THEN integer '12'

WHEN ouf.openingdate < TO\_DATE('2014/01/01', 'YYYY/MM/DD') THEN integer '0'

WHEN ouf.openingdate >= TO\_DATE('2014/01/01', 'YYYY/MM/DD') THEN (CAST(TO\_CHAR(ouf.openingdate, 'MM') AS integer) - 1)

END - CASE WHEN ouf.closeddate IS NULL THEN integer '0'

WHEN ouf.closeddate > TO\_DATE('2015/01/01', 'YYYY/MM/DD') THEN integer '0'

WHEN ouf.closeddate < TO\_DATE('2014/01/01', 'YYYY/MM/DD') THEN integer '12'

WHEN ouf.closeddate >= TO\_DATE('2014/01/01', 'YYYY/MM/DD') THEN 12 - CAST(TO\_CHAR(ouf.closeddate, 'MM') AS integer)

END) AS "Expected Reports",

ROUND(SUM(cr.completedreports) /

SUM(12 - CASE WHEN ouf.openingdate IS NULL THEN integer '12'

WHEN ouf.openingdate > TO\_DATE('2015/01/01', 'YYYY/MM/DD') THEN integer '12'

WHEN ouf.openingdate < TO\_DATE('2014/01/01', 'YYYY/MM/DD') THEN integer '0'

WHEN ouf.openingdate >= TO\_DATE('2014/01/01', 'YYYY/MM/DD') THEN (CAST(TO\_CHAR(ouf.openingdate, 'MM') AS integer) - 1)

END - CASE WHEN ouf.closeddate IS NULL THEN integer '0'

WHEN ouf.closeddate > TO\_DATE('2015/01/01', 'YYYY/MM/DD') THEN integer '0'

WHEN ouf.closeddate < TO\_DATE('2014/01/01', 'YYYY/MM/DD') THEN integer '12'

WHEN ouf.closeddate >= TO\_DATE('2014/01/01', 'YYYY/MM/DD') THEN 12 - CAST(TO\_CHAR(ouf.closeddate, 'MM') AS integer)

END) \* 100, 2) AS "Percentage Complete"

FROM organisationunit AS ouf

INNER JOIN organisationunit AS oui ON (ouf.parentid = oui.organisationunitid)

INNER JOIN organisationunit AS oup ON (oui.parentid = oup.organisationunitid)

INNER JOIN datasetsource AS dss ON (ouf.organisationunitid = dss.sourceid)

INNER JOIN dataset AS ds ON (dss.datasetid = ds.datasetid)

LEFT JOIN

(SELECT cdv.datasetid, cdv.sourceid, COUNT(cdv.periodid) AS completedreports

FROM completedatasetregistration AS cdv

INNER JOIN period AS p ON (cdv.periodid = p.periodid)

WHERE p.startdate >= TO\_DATE('2014/01/01', 'YYYY/MM/DD')

AND p.enddate <= TO\_DATE('2015/01/01', 'YYYY/MM/DD')

GROUP BY datasetid, sourceid) AS cr ON (cr.datasetid = ds.datasetid AND cr.sourceid = ouf.organisationunitid)

WHERE ds.name IN ('Monthly Aid Post Form','Monthly Health Center Report')

GROUP BY oup.name, ds.name

ORDER BY oup.name, ds.name