Paper BI-10-2014

Secret Sauce for Reporting Health Care Data Using SAS and SQL Server Business Intelligence

Bradley Blackmore, Cincinnati Children's Hospital Medical Center, Cincinnati, OH
Gowri Madhavan, Cincinnati Children's Hospital Medical Center, Cincinnati, OH

ABSTRACT

When thinking about quality improvement in health care, it is technology that facilitates the work. Without a reliable mechanism of capturing, analyzing and reporting measures, quality improvement cannot exist. What if you had a tool that could streamline this process? A tool that could fetch the data you need from multiple sources automatically based on rules and criteria that can change at the pace of healthcare? A tool that can produce SAS-ready SQL, raw datasets, or complex charts quickly? A tool like this does not exist today – at least not from any vendors we've explored. Instead, we have architected a solution internally that is capable of doing all of this. With SQL Server Business Intelligence tools we have developed a flexible and dynamic data warehouse that can extract data out of the electronic medical record (EMR) and dynamically apply context and business rules at touch points along the way. The data is stored in a way that makes aggregate charts and graphs easy to create and lightning fast. Moreover, the detail records are just a click away. Using SQL Reporting Services as a front-end, analysts are able to produce line graphs and Pareto charts on the fly, while extracting SAS datasets ready for more advanced work. This paper describes the architecture of such a solution and how it can be used to automate measures for community health. One such measure includes the monthly number of patient visits to non-urgent care units stratified by communities and clinics – a growing area for improvement work in healthcare.

INTRODUCTION

At Cincinnati Children's Hospital Medical Center (CCHMC), the James M. Anderson Center for Health Systems Excellence is tasked with transformational change and healthcare improvement. Structured around quality improvement (QI), our work involves collaborating with teams across the medical center, serving as a catalyst for achieving sustainable and positive change in patient care and outcomes. As part of the Anderson Center's data systems team, we work at the system level supporting several strategic initiatives, one of which is the Start Strong project. Start Strong is a geographically focused initiative aimed at redesigning the delivery of healthcare in an underserved, high risk community in Cincinnati, OH. The project team hopes to implement programs that can reduce preterm birth rates and urgent care hospitalizations in the community.

The data systems team was asked to produce a variety of measures for the Start Strong initiative. Some of these measures include: monitoring the age in days at the baby's first completed newborn visit; plotting the percent of first completed newborn visits within 7 days of life; and for babies less than 30 days of age who have completed a newborn visit, tracking the number of non-urgent visits to the Emergency Department (ED). The final output were run charts updated monthly, with several stratifications or drilldown options such as locale, clinic, and birth hospital. With historical data available, the team hoped to form a baseline for each of these measures. Once established, the team will monitor the run charts closely while implementing programs and interventions, hoping to shift away from the baseline moving in the right direction.

In 2013, the Data Systems group formed an internal team of Business Intelligence (BI) professionals. The BI team's mission was to automate measures and provide a usable data warehouse for analysts to do their work more efficiently. As part of the BI team, we were given extensive QI training and worked alongside QI analysts. For the analysts involved in the Start Strong project, the complexity and scale of the measures was daunting. Using the typical tools such as SAS, it seemed a monumental task of implementing and supporting these measures. The analysts approached the BI group. A few months later, after borrowing techniques from a previous project, we were able to apply a very innovative type of BI architecture. Not only were we able to meet the needs of the Start Strong project but other projects as well. We created a flexible, dynamic architecture using technology that was attainable and accessible, providing the Start Strong analysts and at least three other teams with run charts and clean, organized and usable SAS data sets quickly.

Making it easier to do their work

At CCHMC the task of capturing, analyzing and reporting QI data is labor intensive. Our data analysts use tools such as SQL Server, SAS, and Business Objects to extract the data from a segregated list of tables hosted in a SQL database known as a Data Store and refreshed nightly (see figure 1). The tables contain filtered data out of the EMR's reporting system; changing the structure or content of these tables is very difficult, usually impossible. As part of the BI team we learned that data analysts need easy access to information inside and outside of the Data Store. They need an efficient way to bind business rules, criteria and calculations to the data being collected; as well as, a way to automatically aggregate the data into charts and reports. If our analysts could deliver results faster, they could iterate through changing requirements more efficiently. It all stems from dealing with changing requirements; you cannot avoid it.

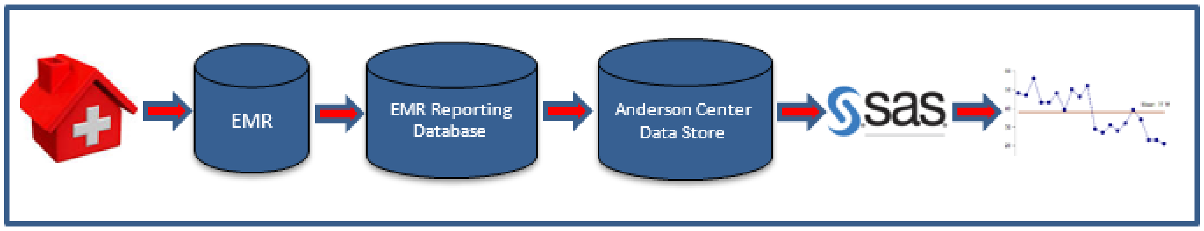


Figure1: Hospital Medical Records Data Flow at CCHMC's Anderson Center Data Systems Group

Start Strong Measures

The Start Strong team asked for a total of nine measures, all complex with multiple criteria and sub groups. In the past, the data analysts would have written SAS programs to churn through data sets and produce Excel charts and graphs. There would be multiple charts on multiple workbooks with a large collection of SAS programs and macros, a large set of files and programs to manage and support. As the Start Strong improvement team begins to explore different ways to better the outcome, they would typically request changes to the reports. Consequently, the data analysts would have to rewrite part of the SAS code and manually correct any Excel sheets.

Monthly, to update the graphs with new data, the analyst would usually have to run the SAS programs manually. The Start Strong project was expected to be a tremendous amount of work for one analyst and expected to take several months.

A Framework Built for Change

The concept of knowledge discovery involves the learning that occurs during the development of analytics. In healthcare, it is due to knowledge discovery that many systems go through costly re-writes to produce the output people need. In the ideal world, BI professionals would document all the business requirements needed for a subject area and code them into the ETL (Extract, Transform, and Load) process loading the data in a structured format ready to use. There would be plenty of time and resources to follow a typical development lifecycle consisting of Requirements – Planning – Designing – Developing – and Deployment. In the perfect world, there would be little or no change.

In order to accommodate the kind of change we see in healthcare, we challenged ourselves to think outside of the standard development cycle. We wanted to build something under the expectation that any of the requirements we receive will inevitably change. Coming from a BI background, we also wanted to leverage all the benefits of a BI solution such as consistency and accuracy of data all in one place. As teams come to us with new or changing needs, we want to shorten the development life cycle and flow freely from requirements to testing to deployment.

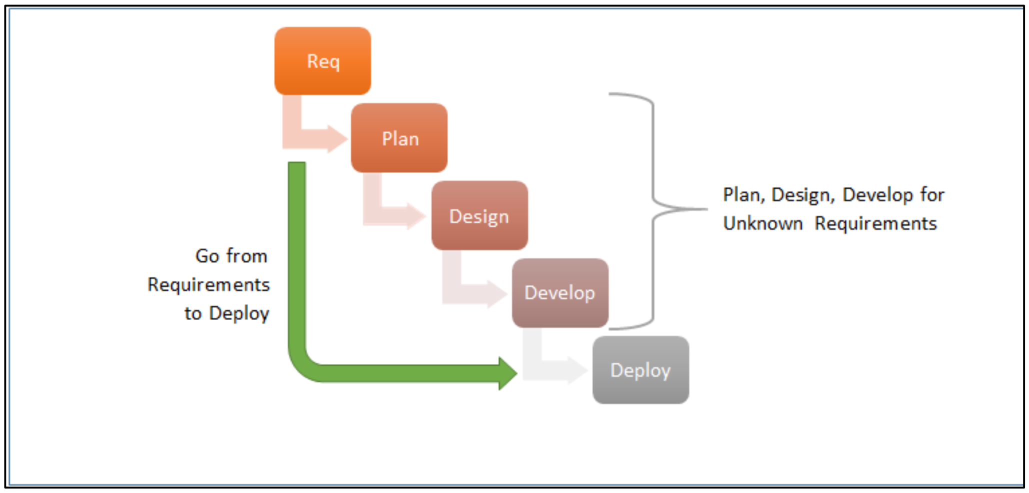

Req
Plan
Design
Develop
Deploy
Go from Requirements to Deploy
Plan, Design, Develop for Unknown Requirements

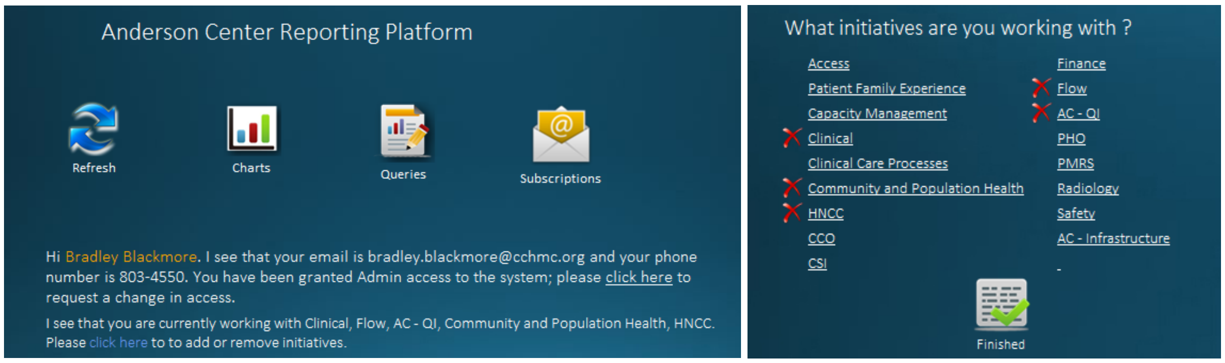
Figure 2: Development Life Cycle Going from Requirements to Deploy

The Presentation Layer

Without a friendly and easy-to-use means of accessing charts and detail records, users would not see the benefits of the system. We could have the most sophisticated mechanism possible for radically storing and churning data but if no one is using it, our mission has failed. We decided to leverage UX (User Experience) design and build a light-weight user friendly tool. We implemented SQL Server Reporting Services (SSRS) reports to display key information without the overhead of building and maintaining a sophisticated web page. The SSRS tool allows users to enter metadata and render charts and reports all in one place.

The Anderson Center Reporting Platform:

Users signing in to the system for the first time are prompted to select an Initiative. There may be several projects within an initiative and each project will contain multiple queries. Once a user selects their initiative, they will only see the queries they need.


Anderson Center Reporting Platform
Refresh
Charts
Queries
Subscriptions
What initiatives are you working with ?
Finished

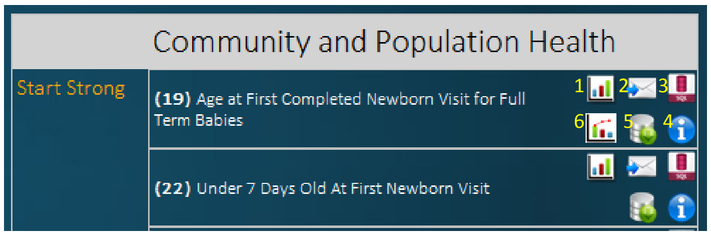
Hi Bradley Blackmore. I see that your email is bradley.blackmore@cchmc.org and your phone number is 803-4550. You have been granted Admin access to the system; please click here to request a change in access.

I see that you are currently working with Clinical, Flow, AC - QI, Community and Population Health, HNCC. Please click here to to add or remove initiatives.

Access	Finance
Patient Family Experience	✗ Flow
Capacity Management	✗ AC - QI
✗ Clinical	PHO
Clinical Care Processes	PMRS
✗ Community and Population Health	Radiology
✗ HNCC	Safety
CCO	AC - Infrastructure
CSI	-

Figure 3: Reporting Platform Home Page

From the home page, users can go to charts, queries and subscriptions. Lets start with Queries: a query as you will learn, is a data request that follows a few basic rules. Queries are given meaningful names that tie back to the operational definition (a wiki document that contains detailed measure requirements and reference information). In the display, each query is organized into projects. In Figure 4 you will see two of the nine Start Strong queries. Next to each query are several icons that provide various options:


Community and Population Health

Start Strong

(19) Age at First Completed Newborn Visit for Full Term Babies

(22) Under 7 Days Old At First Newborn Visit

(1) Run Chart

(2) Contact Query Owner

(3) View Query Syntax

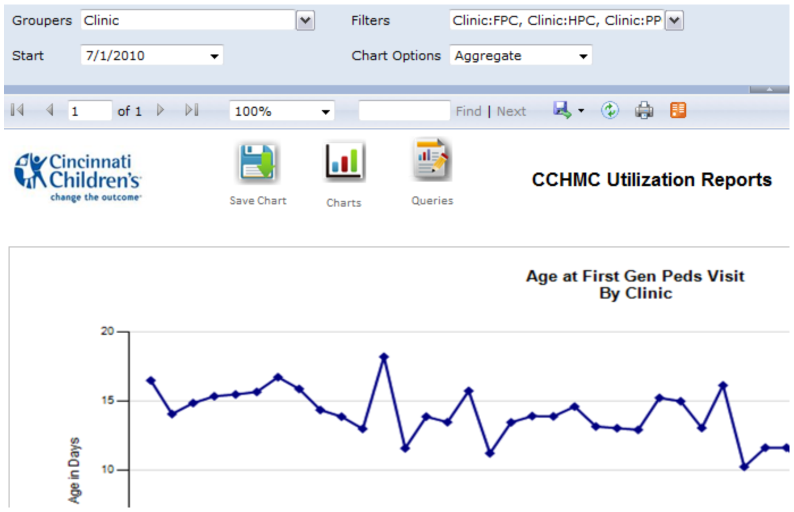
(4) Query Info

(5) Execute Query

(6) Pareto Chart

Figure 4: The Anderson Center's Reporting Platform's Query Dashboard

By clicking on the charting icon, users can render a line graph using the data generated by the query. The line graph will prompt users for a filter, allowing them to visually slice and dice the data. Rendering a run chart is extremely fast given that we have pre-aggregated and stored the values. Charting options include displaying multiple lines, one for each filter selected, or breaking out the chart into one unique chart on its own page. Users can export the chart to PDF, Excel, Word, or as an image. With the appropriate rights, an admin user can save the chart to a library of charts.


Groupers: Clinic

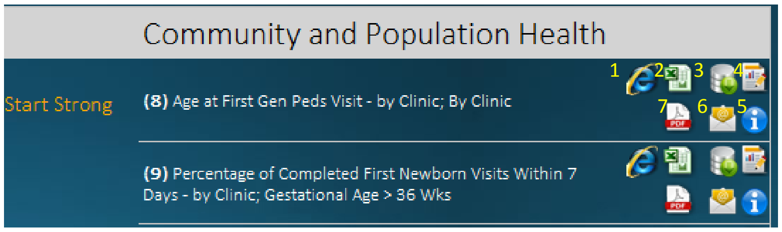
Filters: Clinic:FPC, Clinic:HPC, Clinic:PP

Start: 7/1/2010

Chart Options: Aggregate

Figure 5: Anderson Center Run Chart

Once saved, chart properties such as Titles, Legends, and Axis formatting, can all be customized. The chart can also be scheduled for automatic delivery via e-mail to anyone in the CCHMC directory. In the chart library (Figure 6), there are also one-click options to render the chart as a PDF, Excel, or web page. One of the most widely used features is the option to download all detail data behind the chart. If this is too much data, users can drill through on the chart itself to get the detail records behind each data point. If there are questions about the chart, the business rules and the syntax are one-click away.


Community and Population Health

Start Strong

(8) Age at First Gen Peds Visit - by Clinic; By Clinic

(9) Percentage of Completed First Newborn Visits Within 7 Days - by Clinic; Gestational Age > 36 Wks

(1) Render Chart in a New Window

(2) Render Chart as Excel

(3) Get Detail Data (all)

(4) Query Info

(5) Chart Info

(6) Subscribe to This Chart

(7) Render Chart as PDF

Figure 6: The Anderson Center's Reporting Platform's Charts Dashboard

There are many more features of the Anderson Center Reporting Platform that we could discuss, but we'll stick to just one more: the Query Builder tool. The Query Builder tool displays valuable information about the query, all in one place. With administrator privileges, users can modify existing queries using an interface instead of writing SQL code. Administrators are able to add or remove fields and criteria; they can even make a copy of an existing query and make changes without damaging the existing one. We will cover this tool in more detail later on.

Back to the Architecture

Words of a Sentence

As a result of planning and design, we modeled a BI system that would essentially peel apart all the components of a data request much like taking apart the words of a sentence. By storing the words independently, we are able to pick and choose the words we need; if the word does not exist, we can add to the vocabulary. The sentence is then pieced together and stored in its final form ready to be used later.

We specified a structure to our sentences, similar to the rules of grammar, and as long as the data request can abide by those rules, then the sentence will execute and give you the data you need. Of course, under the premonition that all requirements can change, even the structure of the sentence can change; and it has. With a library of sentence structures we can use, the solution can meet the needs of almost any team.

Sentence Example:

Please show me the list of patients with their names, birth date, and age in days at their first newborn visit to any of the general pediatric clinics for babies who went full term and who live in the following ZIP codes (45xxx,45xxx,45xxx).

Translating the Sentence into a Data Request (Query):

Select PatientName, Birthdate, AgeInDays, VisitDate from ClinicVisits where GestationalAge >= 36 weeks and PatientZip in (45xxx,45xxx,45xxx)

Sentence/Query Components:

Select **[fields]** from **[table]** where **[criteria]**

A typical data request or query consists of the following components: a table or set of tables, a list of fields, a measure expression or formula, a set of criteria to filter the data, and a way to group or aggregate the results. The idea behind our architecture is to produce a structured output based on an unlimited combination of input. The goal is to store the various components of a typical query in the form of metadata (data about data). The mechanism behind the scenes can then fuse together all the components needed and deliver an executable statement. The SQL statement must be independent and be able to run on its own. Users can then take the query and embed it into SAS or any other reporting tool that can read SQL.

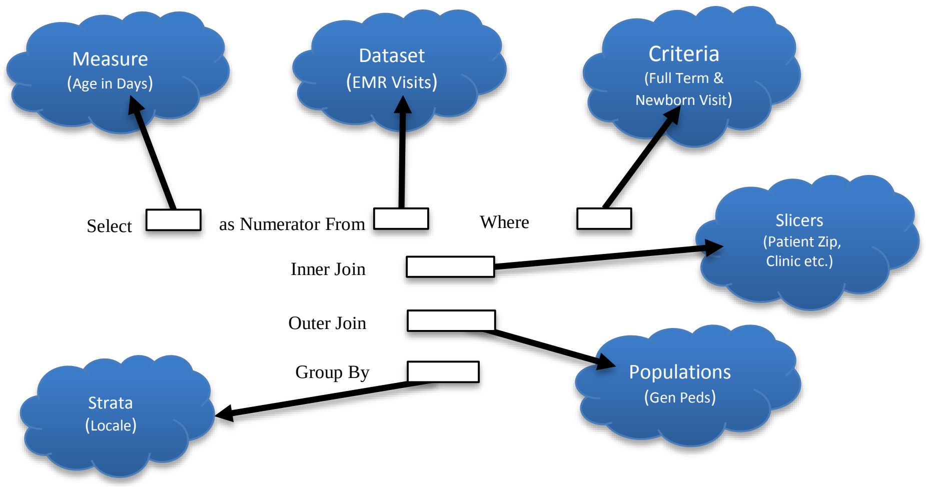


Figure 7: The Components of a Data Request

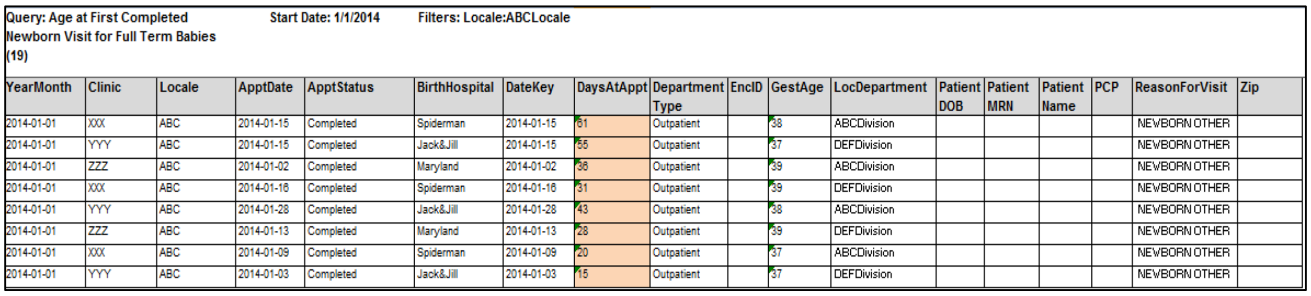
Semantic Layer

The process of metadata driven reporting is not new; often called a semantic layer, Business Objects and other tools already do this. Nonetheless, our team saw the limitations, poor performance, and political hurdles of these tools and decided to build our own. In addition, after evaluating several vendors, we could not find one that was able to work the way we work. We chose to use SQL Server BI tools that we already owned and had experience with; it was accessible and attainable and our team could use it quickly. We also wanted something lightweight that could accommodate a small group of users quickly and grow as demand grows. Finally, we wanted to physically store the aggregated results from our queries in a fact table (table that stores all numeric values and keys to the information about those numbers) to provide instant access to calculated results and faster performance – something that breaks the mold of a typical semantic layer.

Step by Step, How Do We Use It?

Let's step through a typical Data Request from our analysts:

1. Read and understand the question: What are the ages, in days, of babies at their first completed newborn visit for certain ZIP codes?
2. What are the components needed to answer that question?
    a. **A measure:** Age in Days
    b. **Dataset:** EMR Encounters
    c. **Criteria:** Full Term babies, newborn visits only
    d. **Dimensions:** Patient Dimension looking at Patient ZIP code, Patient Name, Patient DOB
    e. **Populations:** General Pediatrics patients only
    f. **Strata:** group by Patient

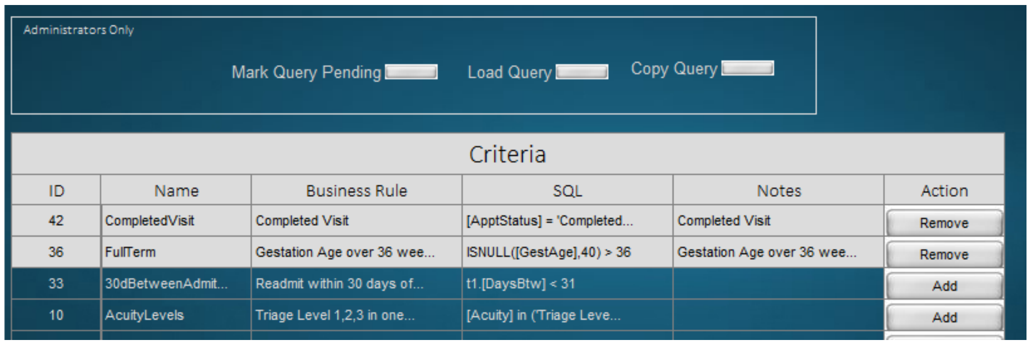


Query: Age at First Completed Newborn Visit for Full Term Babies (19) **Start Date: 1/1/2014** **Filters: Locale:ABCLocale**

YearMonth	Clinic	Locale	ApptDate	ApptStatus	BirthHospital	DateKey	DaysAtAppt	Department Type	EncID	GestAge	LocDepartment	Patient DOB	Patient MRN	Patient Name	PCP	ReasonForVisit	Zip
2014-01-01	XXX	ABC	2014-01-15	Completed	Spiderman	2014-01-15	61	Outpatient		38	ABCDivision					NEWBORN OTHER	
2014-01-01	YYY	ABC	2014-01-15	Completed	Jack&Jill	2014-01-15	55	Outpatient		37	DEFDivision					NEWBORN OTHER	
2014-01-01	ZZZ	ABC	2014-01-02	Completed	Maryland	2014-01-02	36	Outpatient		39	ABCDivision					NEWBORN OTHER	
2014-01-01	XXX	ABC	2014-01-16	Completed	Spiderman	2014-01-16	31	Outpatient		39	DEFDivision					NEWBORN OTHER	
2014-01-01	YYY	ABC	2014-01-28	Completed	Jack&Jill	2014-01-28	43	Outpatient		38	ABCDivision					NEWBORN OTHER	
2014-01-01	ZZZ	ABC	2014-01-13	Completed	Maryland	2014-01-13	28	Outpatient		39	DEFDivision					NEWBORN OTHER	
2014-01-01	XXX	ABC	2014-01-09	Completed	Spiderman	2014-01-09	20	Outpatient		37	ABCDivision					NEWBORN OTHER	
2014-01-01	YYY	ABC	2014-01-03	Completed	Jack&Jill	2014-01-03	15	Outpatient		37	DEFDivision					NEWBORN OTHER	

Figure 8: Detail records from a Start Strong query

3. Browse the metadata library to see if we can reuse existing definitions
    a. Is there a measure we can reuse? Yes, Age in Days was used in a previous project.
    b. Is there a dataset we can use? Yes, the EMR encounter data set already exists.
    c. Are there criteria we can use? No, the full term criterion is not available yet.
4. If not, we need to add the criteria to the metadata library
    a. It looks like we need to add a new criterion for Full Term Babies
5. Tell the Query Builder tool which definitions to pull out of the library and hit the BUILD QUERY button



Criteria

ID	Name	Business Rule	SQL	Notes	Action
42	CompletedVisit	Completed Visit	[ApptStatus] = 'Completed...	Completed Visit	Remove
36	FullTerm	Gestation Age over 36 wee...	ISNULL([GestAge],40) > 36	Gestation Age over 36 wee...	Remove
33	30dBetweenAdmit...	Readmit within 30 days of...	t1.[DaysBtw] < 31		Add
10	AcuityLevels	Triage Level 1,2,3 in one...	[Acuity] in ('Triage Leve...		Add

Figure 9: The Query Builder tool – add or remove Criteria

6. Check the SQL syntax to see if the tool generated the query as you expected (snapshot of syntax)

```
SELECT *
FROM    (SELECT DimDate.[YearMonth]                AS [YearMonth],
          DimDepartment.[DepartmentType]           AS [DepartmentType],
          DimPatient.[PatientMRN]                  AS [PatientMRN],
          t1.[EncID]                               AS [EncID],
          DimPatient.[PatientName]                 AS [PatientName],
          DimPatient.[PatientDOB]                  AS [PatientDOB],
          t1.[DaysAtAppt]                          AS [DaysAtAppt],
          DimPatient.[Zip]                         AS [Zip],
          t1.[DateKey]                             AS [DateKey],
          t1.[ReasonForVisit]                      AS [ReasonForVisit],
          t1.[LocDepartment]                       AS [LocDepartment],
          t1.[GestAge]                             AS [GestAge],
          t1.[ApptStatus]                          AS [ApptStatus],

…………………………………….

FROM    abcdatabase.dbo.chmc_abc_demo AS d
          INNER JOIN defdatabase.dbo.tblpatientvisit AS pv ON
          pv.pat_mrn_id = d.pat_mrn_id
          LEFT JOIN abcdatabase.bmi.ped_gest_age AS PGA ON
          PGA.pat_mrn_id = d.pat_mrn_id
          LEFT JOIN defdatabase.dbo.tblFinalDx DX ON DX.PAT_ENC_CSN_ID =
          pv.PAT_ENC_CSN_ID AND
          DX.LINE = 1
          LEFT JOIN defdatabase.dbo.tlkICD9 icd9 ON icd9.Icd9 = DX.
      CURRENT_ICD9_LIST
          WHERE  pv.prc_name LIKE '%NEWBORN%' AND
             pv.department_name IN ('CCC PEDS', 'ABC HEALTH', 'DEF
             PRIMARY CARE') AND
                  appt_time > '1/1/2012') AS t1

………………………………………..

GROUP  BY
DimDate.[YearMonth],
DimDepartment.[DepartmentType],
DimPatient.[PatientMRN],
t1.[EncID],
DimPatient.[PatientName],
DimPatient.[PatientDOB],
t1.[DaysAtAppt],
DimPatient.[Zip],
t1.[DateKey],
t1.[ReasonForVisit],
t1.[LocDepartment],
t1.[GestAge],
t1.[ApptStatus],
CASE WHEN [POPID] = 5 THEN [Attribute2]
END,CASE WHEN LEFT([Zip], 5) IN ('12345') THEN 'ABC LOCALE'
WHEN LEFT([Zip], 5) IN ('12346') THEN 'DEF LOCALE'
ELSE 'Other'
END,CONVERT(DATE, t1.[DateKey]),ISNULL(t1.[Clinic], 'Not in Clinic'))
AS F
```

Figure 10: SQL syntax from the Start Strong Query

7. Hit the LOAD QUERY button to load the data into the data warehouse and automatically generate charts and graphs
8. Optional: Take the SQL syntax out to SAS and build SAS datasets for more advanced analytics

Advanced Analytics with SAS

We take the SQL syntax produced by the reporting system to the next level by generating SAS datasets for additional manipulations, such as stratification by physicians or nurses, redefining the 'term' babies to include 'late preterm babies', or identifying the babies who come for a new born visit by 9 days of life. Complex adjustments to the data can be accomplished in one or two data steps in SAS. Furthermore, this can be completed without affecting the original SQL and without adding additional steps to the SQL reporting platform. What's more, if the SAS data changes become standard, they can be implemented into the query builder tool and automated.

Step by Step, How Do We Use It in SAS?

We will step through a SAS program and how it can leverage the SQL produced by our framework.

In addition to using SQL language on SAS datasets, PROC SQL allows you to execute native SQL directly against the database. We are able to do this by using the SAS/ACCESS to OLEDB (Object Linking and Embedding Database) engine. The OLEDB connection allows us to connect to a Microsoft SQL Server database and pass the SQL code to the database for execution. The connection parameters identify which database to connect to and how to connect. In our environment, database security is integrated with our network logon hence the "Integrated Security" syntax (line 2). The connection parameters including security will be specific to each database and environment. For more information refer to the SAS/ACCESS documentation.

```
proc sql ;                                                        (1)
        connect to oledb as EMR_PT (provider=SQLOLEDB dsn = 'AA01BB\AA01BB'
        schema = dbo properties = ("Initial Catalog" = Example_Main,
        "Integrated Security"=SSSS));                             (2)
```

In the above code we provide an alias for connection named "EMR_PT". This mechanism allows us to reference the connection by name later. Once we have the connection to the database open we can execute SQL statements.

```
create table nb_termbabies  as                                    (3)
            select *
                  from connection to  EMR_PT                      (4)
                        (
                  SELECT *                                        (5)
                        FROM
                  (SELECT dimdate.[yearmonth]          AS [YearMonth],
                        dimdepartment.[departmenttype] AS [DepartmentType],
                        dimpatient.[patientmrn]        AS [PatientMRN],
                        t1.[encid]                     AS [EncID],
                        dimpatient.[patientname]       AS [PatientName],
                        dimpatient.[patientdob]        AS [PatientDOB],
                        t1.[daysatappt]                AS [DaysAtAppt],
                        dimpatient.[zip]               AS [Zip],
                        t1.[datekey]                   AS [DateKey],
                        t1.[reasonforvisit]            AS [ReasonForVisit],
                        t1.[locdepartment]             AS [LocDepartment],
                        t1.[gestage]                   AS [GestAge],
                        dimpatient.[lastpcp]           AS [LastPCP],
                        t1.[apptstatus]                AS [ApptStatus],
                        CASE
                              WHEN [popid] = 2 THEN [attribute1]
                        END
                        ……………………..

                        CASE
                              WHEN LEFT ([zip], 5) IN ( '12345' ) THEN 'abc'

                        …………………. AS F
                        )
      ;
      quit;                                                               (6)
```

In line (3) above, we have an outer CREATE TABLE statement that specifies the name of the SAS dataset that we will create – NB_TERMBABIES. The FROM statement in line (4) references our database connection EMR_PT, indicating the creation of our SAS Dataset from the of the database query. In line (5), we have the SQL generated by the Anderson Center reporting framework that we inserted manually. By using SQL passthrough, we can take the native SQL generated by our metadata and copy it directly into SAS. SAS passes it to the database for execution. The SQL code can be tweaked as needed but must be consistent with standard SQL syntax. In line (6), we close the connection to the database.

In line (7), to meet our needs, we execute several data step manipulations using SAS code to restructure the dataset.

```
data nbvisits_expand
(drop= numerator denominator rate  apptdate yearmonth);
 set nb_termbabies ;                                                       (7)

*gestage is string, convert to number;
gage=input(gestage, 8.);

***nbvisit is in string, convert to date;
nb_visit_date=input(datekey,ANYDTDTE11.);
format nb_visit_date date11.;

***extracting the month from edvisit;
nbvisit_mon=intnx('month',nb_visit_date,0,'e');
format nbvisit_mon date9.;

zipf=substr(zip,1,5);
drop zip;

run;
```

In line (8), we then produce the analytics. For this measure, we are able to calculate the frequencies of babies with a newborn visit within seven or nine days of life by ZIP code and locale.

***An example of annotations of comments from SAS syntax given below;

```
***by locale/ adjust for term definition to include or exclude late preterm
babies, or days of life for seven or nine;

title1 'Frequency of Age in Days by Locale';
proc freq data=nbvisits_expand;                                            (8)
      *where  gage >=37 and daysatappt<=7;
      *where  gage >=35 and daysatappt<=9;
      where  gage >=35;
      tables nbvisit_mon*locale/  nocol nopercent norow;
      tables gage daysatappt;
run;
title;
```

In order to produce Excel charts, as many of our customers typically request, we use the Excel libname engine. In step (9) through step (10) we output datasets for each locale and save them in an Excel libname.

```
data nbvisits_abc_locale (keep=nb_visit_date daysatappt)
      nbvisits_def_locale(keep=nb_visit_date daysatappt);   (9)

      set nbvisits_xmr;

      if '01jan2014'd <=nbvisit_mon <='31jul2014'd;

      if locale = 'ABC LOCALE' then
            output nbvisits_def_locale;

      if locale = 'DEF LOCALE' then
            output  nbvisits_abc_locale;
run;

proc sort data= nbvisits_def_locale out=nbvisits_def_locale_sorted;
      by nb_visit_date daysatappt; run;


proc sort data= nbvisits_abc_locale out=nbvisits_abc_locale_sorted;
      by nb_visit_date daysatappt; run;

libname nbvisit
'T:\username\PopulationHealth\nbvisits\nbvisits_01012014_07312014.xls';
                                                                           (10)

data nbvisit. Nbvisits_def_locale_sorted;
      set nbvisits_def_locale_sorted; run;


data nbvisit. Nbvisits_abc_locale_sorted;
      set nbvisits_abc_locale_sorted; run;


libname nbvisit clear;
```

Once the summary data is in Excel we generally use standardized measurement charts, such as the XMR chart, to view the impacts of quality improvement over time.

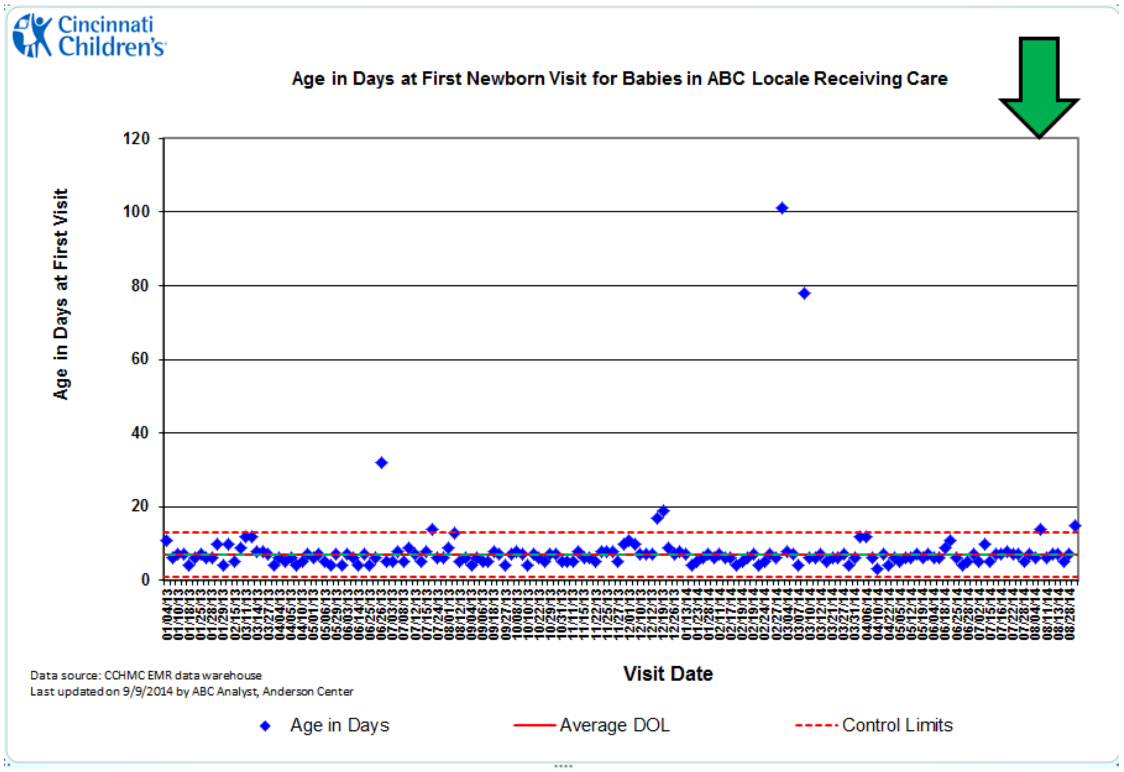


Figure 10: XMR (Individual Measurement) chart for age in days at first new born visit.

UPCOMING ENHANCEMENTS

We are currently developing SAS code to query the metadata stored in the SQL Server database to automatically import the SQL syntax needed for each dataset. One advantage is that our SAS programs will stay in sync with any changes to the SQL in the metadata. In addition, our SAS programs will be more concise and reusable. Below is an example of the SAS code we hope to implement.

```
PROC SQL;
      Select  SQL_CODE   into :SQL
/* CREATES a SASMACRO variable SQL to hold the SQL_CODE */
      From BI_PLATFORM.TABLE
/* This would need to be the LIBRARY and TABLE that has the code */
      Where query = 19
      ;
Run;
```

By using a macro variable, we could dynamically generate the SQL code and have the SAS program execute it for us.

```
proc sql;
      connect to oledb as EMR_PT (provider=SQLOLEDB dsn = &database_server
      schema = dbo properties = ("Initial Catalog" = Example_Main,
      "Integrated Security"=SSSS));
      create table nb_termbabies  as
            select *
                  from connection to  EMR_PT
                        (
                        &SQL
                              ) ;
quit;
```

DIFFERENT TYPES OF USERS

People consume data differently. Our system had to be robust and flexible enough accommodate a variety of data requests. Likewise, our system had to account for multiple types of users. Our users include BI professionals, QI Analysts, Data Analysts, Quality Control professionals in the hospital and even medical staff.

The system had to provide information to analysts in the form of run charts, pareto charts, large data dumps, filtered data, or in the form of a stand-alone SQL query. The solution we provided allows analysts to print and send charts, build their own reports using the SQL query, even embed the SQL code into a SAS program to do more advanced work. For our enterprise-wide users we automatically send the summary data to our company-wide reporting system to be displayed on dashboards and charts throughout the organization. From the enterprise reporting system, users can request access to detail and the system will handle all the security permissions needed to manage private healthcare information (PHI) correctly.

CONCLUSION

In healthcare, specifically in the areas of quality improvement and analytics, change in requirements is unavoidable. We recommend taking a different angle to this problem. We are not asking you to hire BI experts or purchase expensive BI tools. We are also not telling you to go out and build a dynamic data warehouse on day one. We would like to inspire you to think differently. Start to think along the lines of “Change is good.” For the next project build something that not only expects change, but grows deeper and richer for it.

Whether you use SAS or SQL server as your platform, try to build a framework that can split apart all the components of the request into a library of reusable chunks. Even if your library starts with a small set of measure rules, try to leverage the concept of dynamic code. Give it a try. Use a program – either a SQL stored procedure or SAS macro – to fetch the metadata you need from your library and stitch together a select statement and then execute it. Your goal should be to loop through each measure in your library, and as you add new ones, let your engine create usable queries and datasets automatically. Eventually you will progress to using this framework to accommodate multiple similar projects; then as you mature, you may even build a dynamic data warehouse of your own.

REFERENCES

Provost, L., & Murray, S. (2011). *The Health Care Data Guide*. San Francisco, CA: Jossey-Bass
Montgomery, D. (2009). *Introduction to Statistical Quality Control* (6th ed.). John Wiley & Sons.
Turban, E., Sharda, R., Aronson, J., & King, D. (2010). *Business Intelligence A Managerial Approach* (2nd ed.). Pearson Education Limited.
Kimball, R., & Ross, M. (2002). *The Data Warehouse Toolkit* (2nd ed.). John Wiley & Sons.
Larson, B. (2009). *Delivering Business Intelligence with Microsoft SQL Server 2008*. The McGraw Hill companies.

ACKNOWLEDGMENTS

We thank the team members who have contributed for this paper, with their subject knowledge expertise and they include:

- Courtney Brown, MD, instructor, Division of General and Community Pediatrics, CCHCM;
- Alan Leach, MS, MIS, SAS Developer, Intellicog, Inc;
- Michael Platt, MS, Senior Analyst- Quality and Transformation, Data Systems Team, Anderson Center, CCHMC.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Name: Bradley Blackmore, MBA
Enterprise: Cincinnati Children's Hospital and Medical Center
Address: 3333 Burnet Avenue, MLC 5040
City, State ZIP: Cincinnati, OH 45229
Work Phone: 513-803-4550
E-mail: bradley.blackmore@cchmc.org

Name: Gowri Madhavan, MPH
Enterprise: Cincinnati Children's Hospital and Medical Center
Address: 3333 Burnet Avenue, MLC 5040
City, State ZIP: Cincinnati, OH 45229
Work Phone: 513-803-1847
E-mail: gowri.madhavan@cchmc.org