Paper 4285-2020

Accessing Medicare Data at the Centers for Medicare and Medicaid Services using SAS®

Rick Andrews, Office of the Actuary, Centers for Medicare and Medicaid Services

ABSTRACT

CMS is certainly no stranger to complex data. With billions of Medicare records loaded each year, Fee-for-Service (FFS) databases like the National Claims History (NCH), Standard Analytical Files (SAFs), Integrated Data Repository (IDR), and Chronic Conditions Warehouse (CCW) are getting more involved every day. This paper will discuss some of the nuances among the various systems and will offer tips for accessing each of these environments using SAS Access to Teradata®, Oracle®, and the SAS Grid; utilizing tools such as SAS Enterprise Guide®, SAS Studio, and the IBM® mainframe.

INTRODUCTION

There are two environments for accessing FFS claims: (1) the Baltimore Data Center (BDC), and (2) the Virtual Research Data Center (VRDC). CMS employees can gain access to either of these systems at no charge, provided they have justification. CMS contractors can gain access to both environments, although there may be a fee involved with accessing data within the VRDC. For researchers in the general public, access to the VRDC can be obtained for a fee by contacting the Research Data Assistance Center (ResDAC) at this address, https://www.resdac.org/. The VRDC offers two options to access data, through a virtual connection or by physical receipt. The information required for the request and the fees differ depending on access. There are various data located within both environments; such as, the National Claims History (NCH), Transformed Medicaid Statistical Information System (T-MSIS), and the Enrollment Database (EDB), among others. This paper will concentrate on the NCH.

NATIONAL CLAIMS HISTORY

The NCH files contain billing and utilization information on Medicare beneficiaries enrolled in hospital insurance (Part A) or medical insurance (Part B) of the Medicare program. Information maintained in this system includes, but is not limited to Medicare billing and utilization data, name, health insurance claim number, ethnicity, gender, date of birth, state and county code, zip code, as well as the basis for the beneficiary's Medicare entitlement.

STANDARD ANALYTICAL FILES

The SAFs are final action versions of the NCH and can be obtained from the CMS mainframe, ResDAC, or the CMS Data Extract System (DESY). Note that the CCW Research Identifiable Files (RIFs) also contain final action versions of the NCH in a SAS data warehouse, which makes it easier for researchers to access.

INTEGRATED DATA REPOSITORY

The IDR was implemented under Section 101 of Medicare Prescription Drug, Improvement, and Modernization Act of 2003 (MMA) (Pub. L. 108-173), to house Part D drug information. The system now includes Parts A, B, C and D, and DME claims, beneficiary and provider data sources, along with ancillary data such as contract information, risk scores, and many others. Access to this robust integrated data supports much needed analytics across CMS.

CHRONIC CONDITIONS WAREHOUSE

The CCW was implemented under Section 723 of the MMA to provide less complicated access to CMS, other government agencies, as well as external researchers, such as those based in universities. The environment provides researchers with Medicare and Medicaid beneficiary, claims, and assessment data linked by beneficiary across the continuum of care. In the past, researchers analyzing data files were required to perform extensive analysis related to beneficiary matching, which is no longer needed when using the CCW.

ENVIRONMENTS

BALTIMORE DATA CENTER

There are two avenues for accessing FFS data within the BDC: (1) the IBM mainframe; and, (2) the SAS Enterprise Business Intelligence (EBI) server. The NCH files can be accessed only from the mainframe, whereas the IDR can be accessed from either environment.

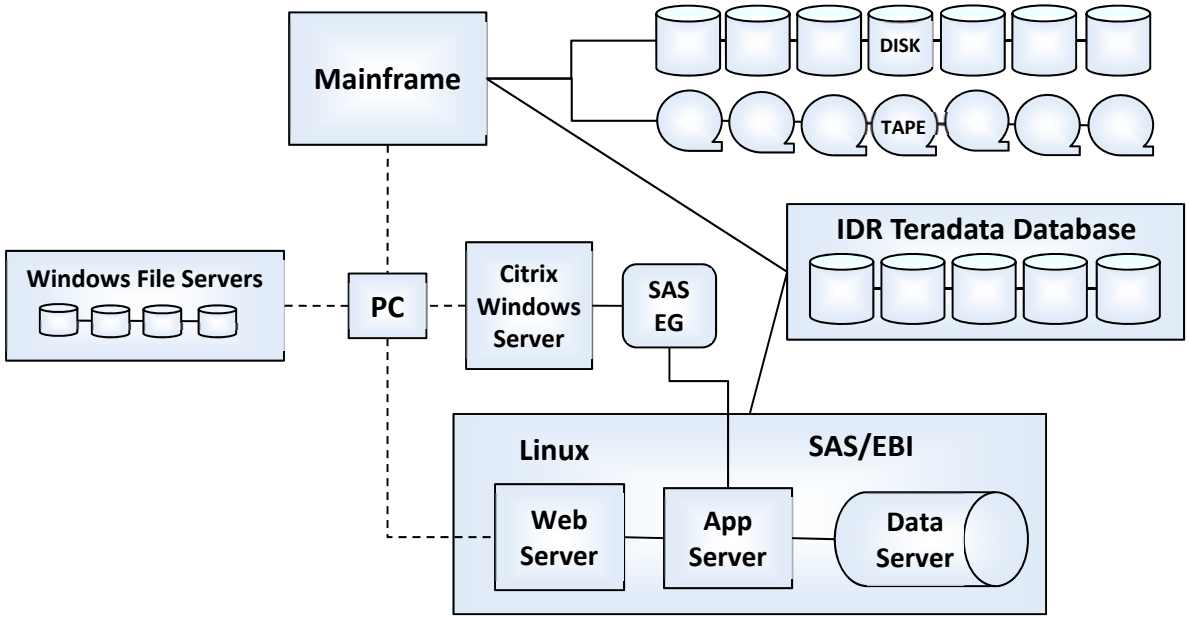


Figure 1: CMS Access Method

VIRTUAL RESEARCH DATA CENTER

The VRDC is a virtual research environment that provides timelier access to Medicare and Medicaid program data in a more efficient and cost effective manner. Researchers working in the CMS VRDC will have direct access to approved data files and be able to conduct their analysis within the CMS secure environment.

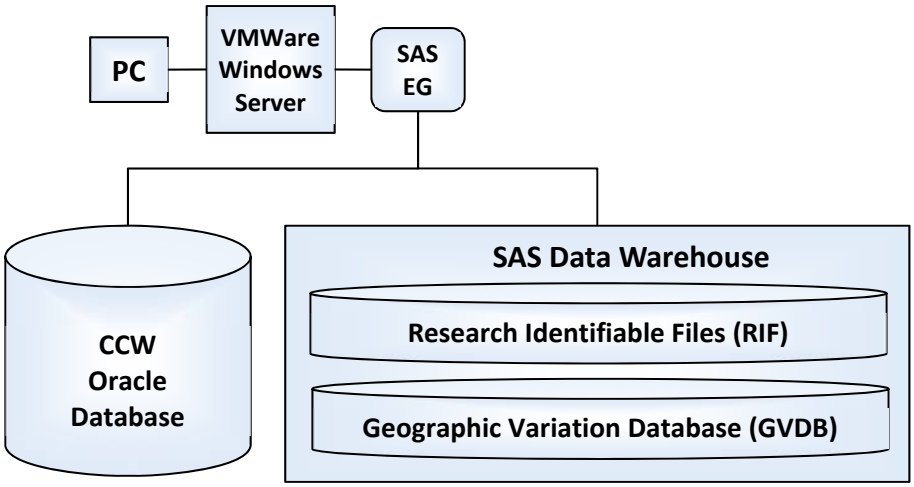


Figure 2: VRDC Access Method

FILE STRUCTURES

NATIONAL CLAIMS HISTORY

The NCH files and the SAFs both have the same file structure. They are variable length flat file on the CMS mainframe on tape data sets. This means a researcher must use a SAS INPUT statement to first read the data and extract the information before doing any analysis. Figure 3 shows the layout for the professional data from CMS Form-1500. Notice there is a fixed header portion and various variable length trailer records. The trailers are basically column-wise tables that contain a varying number of columns based on the number of trailer records.

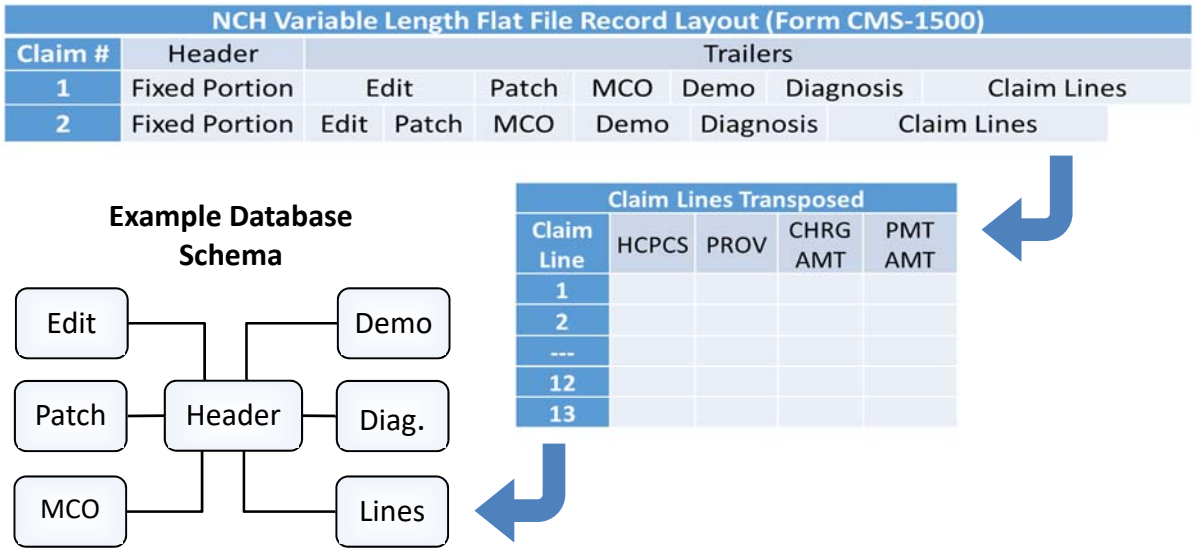


Figure 3: NCH Professional File Layout

Figure 4 is the institutional file layout, which is used for inpatient, outpatient, home heath, hospice, and skilled nursing records from CMS Form-1450. Notice there are more trailers in this layout than in the physician/supplier layout. Also, notice the use of Revenue Centers in lieu of Claim Lines.

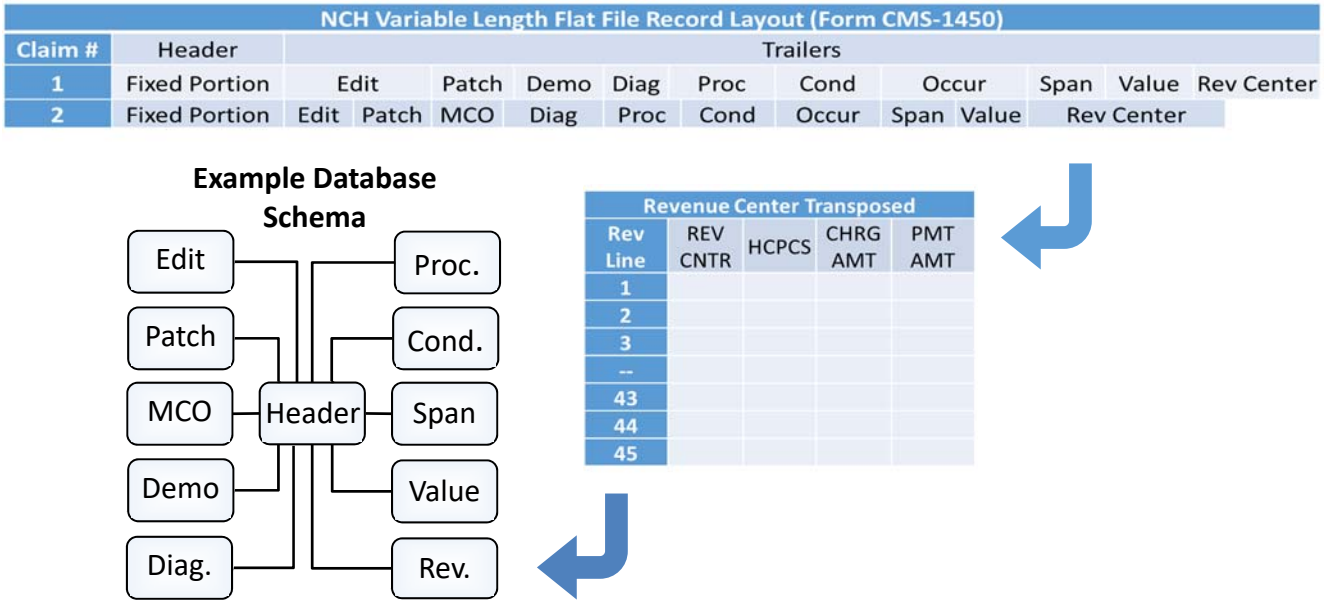


Figure 4: NCH Institutional File Layout

It should be noted that in both Figure 3 and Figure 4 the trailers are transposed from an array of columns into rows of data in the Example Database Schema references. This is the case for data tables in the IDR, CCW, and RIF files. Thus, care should be taken when joining to these tables as this creates a one-to-many scenario that can duplicate data. Also, in all three databases, the header records are output into "claim" tables and the Claim Lines and Revenue Center trailers are output into "claim line" tables. In addition, the other trailers are output into separate tables as well.

INTEGRATED DATA REPOSITORY

The IDR is housed within a Teradata database in the BDC and utilizes a collection of "views" within Virtual Data Marts (VDMs) to reference the underlying physical tables. A view is simply an SQL query that looks like an actual table to the user. This is advantageous because joins and limiting criteria can be performed within the code to make research easier for the user. At present, there are over twenty-five VDMs within the IDR, but this paper will only discuss two of them: Medicare View and Access Layer. The Medicare View, which contains 632 views that represent the underlying tables, for the most part, contain no joins or limiting criteria. This allows more flexibility when writing queries, but forces the user to know the join structure and limiting criteria. The Access Layer, which contains only 21 views, does contain joins and limiting criteria so no need to worry about such things.

Table 1: IDR VDM Examples

VDM	Database	Business Purpose
Medicare View	CMS_VDM_VIEW_MDCR_PRD	Single largest VDM to support ad hoc reporting needs for Medicare Parts A, B, C, and D (632 Views)
Access Layer	CMS_VDM_VIEW_SMNTC_PRD	User-friendly subset of MDCR, organized in a dimensional model requiring fewer joins (21 Views)

Figure 5 is an example of the Medicare View data model that demonstrates why many users feel the IDR is more difficult to use than the Access Layer, or the CCW and RIF tables within the VRDC. Notice that the join structure contains 4, and 5 part keys in order to combine the various tables together. This is an example of a database that is in a file structure called third normal form.

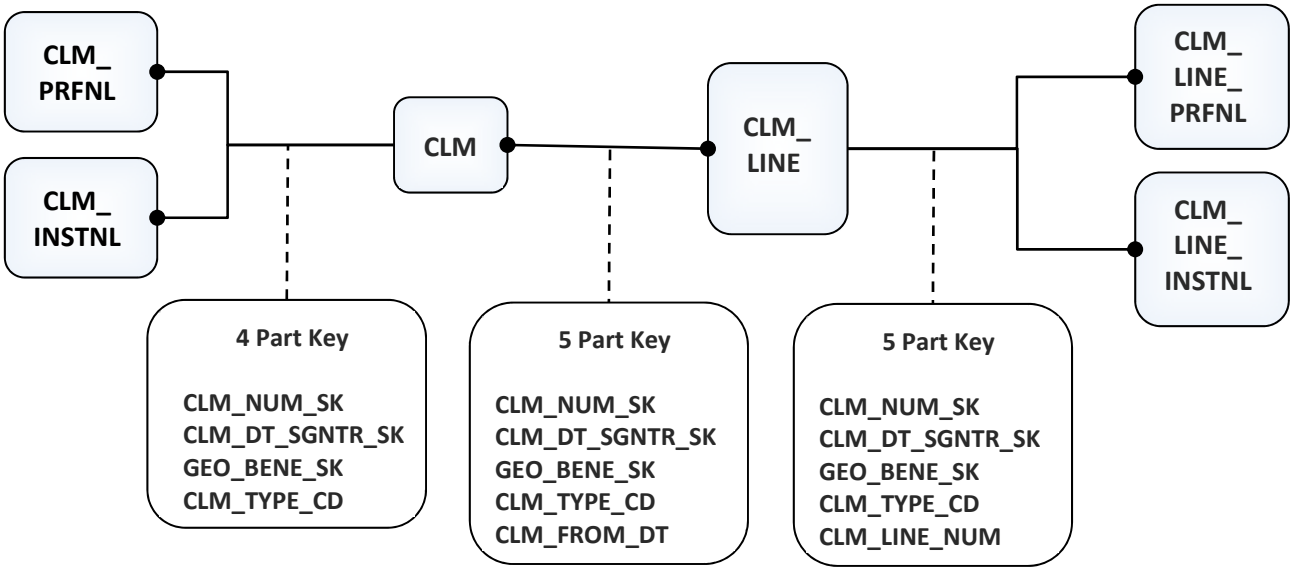


Figure 5: IDR Medicare View Data Model Example

Figure 6 is an example of a simplified view of an IDR join to the geographic tables to obtain the Social Security Administration (SSA) state code. This is an illustration of how the join structure can be simplified using the Medicare View versus the Access Layer.

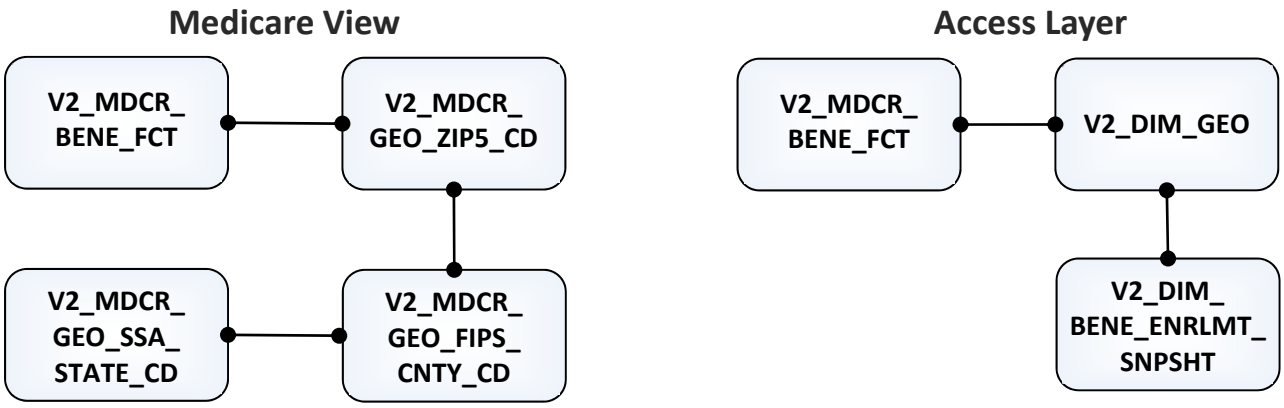


Figure 6: Medicare View versus Access Layer Example

CHRONIC CONDITIONS WAREHOUSE

The CCW is housed within an Oracle database at the VRDC and is designed using a star schema, whereas the IDR utilizes third normal form. The star schema architecture is the simplest data warehouse schema. It is called a star schema because the diagram resembles a star, with points radiating from the center. The center of the star consists of fact table and the points of the star are the dimension tables (STAR,2019). Third normal form is a classical relational-database modeling technique that minimizes data redundancy through normalization. When compared to a star schema, a 3NF schema typically has a larger number of tables due to this normalization process (SCHEMA,2019).

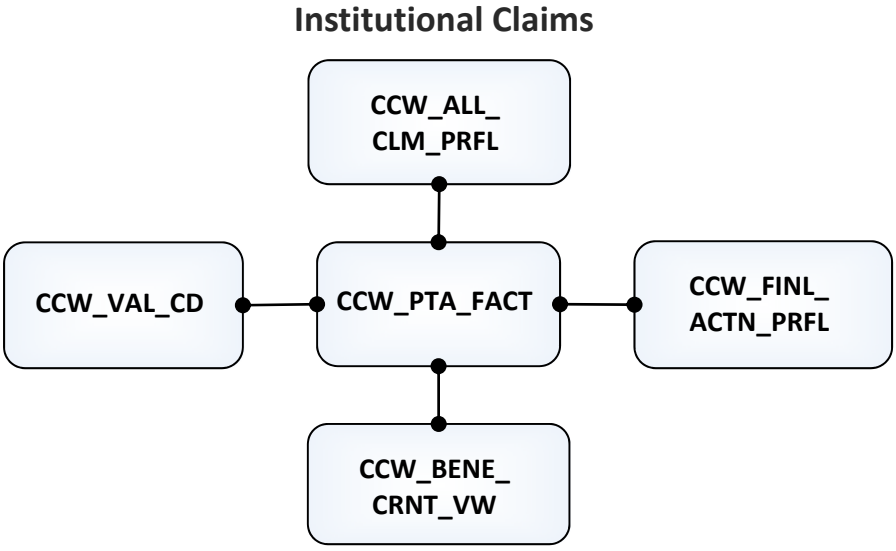


Figure 7: CCW Oracle Data Model Example

```
where CCW_PTA_FACT.CLM_ID        = CCW_VAL_CD.CLM_ID
  and CCW_PTA_FACT.CLM_TYPE_ID   = CCW_ALL_CLM_PRFL.CLM_TYPE_ID
  and CCW_PTA_FACT.FINL_ACTN_ID = CCW_FINL_ACTN_PRFL.FINL_ACTN_ID
  and CCW_PTA_FACT.BENE_ID       = CCW_BENE_CRNT_VW.BENE_ID
```

RESEARCH IDENTIFIABLE FILES

The RIF is a SAS data warehouse that also resides within the VRDC and is designed using a star schema. The data sets are segmented by year, month, and claim type. This lends itself well to SAS Grid processing. A SAS grid computing environment is one in which SAS computing tasks are distributed among multiple computers on a network, all under the control of SAS Grid Manager. In this environment, workloads are distributed across a grid of computers. This workload distribution enables workload balancing, accelerated processing, and job scheduling (GRID,2019). Note that each of the tables can be joined together using the CLM_ID variable.

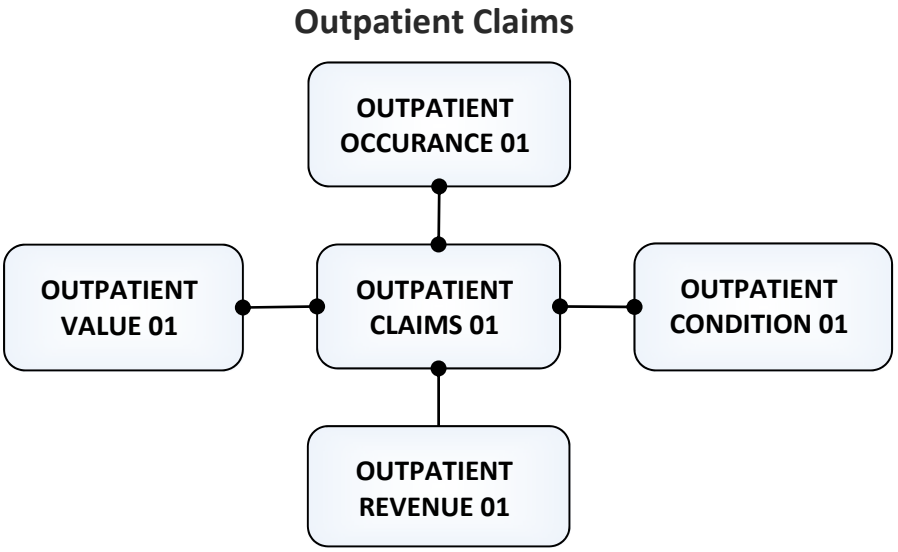


Figure 8: RIF Data Model Example

GEOGRAPHIC VARIATION DATABASE

The GVDB is a suite of easy-to-use SAS analytic files that combine basic Medicare beneficiary and FFS claims data with a range of additional information that make it easier to study geographic variation in spending and service use. The primary source for the Medicare beneficiary and FFS claims data in the GVDB is the CCW Oracle database. Many users find these files to be the easiest to use because the data structure is highly de-normalized and is segmented only by calendar year and whether the claim is institutional versus physician. Also, there are various additional variables that have been added for ease of use. For example, service variables have been added to identify whether a claim is acute-care inpatient, skilled nursing, hospice, and so on. In addition, stay indicators, visit counts, and covered days have been added and all of the diagnosis and procedure codes exist on one record, so there is less need to join tables together. Note that these tables can be joined together using the CLM_ID variable.

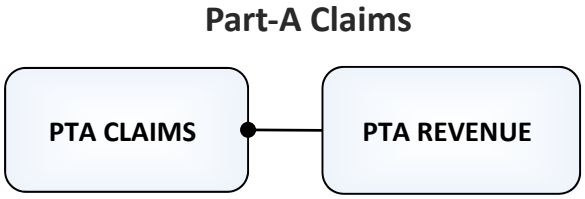


Figure 9: GVDB Data Model Example

FLOW OF DATA

The NCH is fed by the Common Working File (CWF), which gets its data from the Shared Systems. The Fiscal Intermediary Standard System (FISS) handles institutional claims; the Medicare Multi-Carrier Claims System (MCS) handles physician claims; and the Viable Information Processing System (VIPS) Medicare System (VMS) processes durable medical equipment claims. All of this data gets fed into the IDR, which is used by Program Integrity for fraud and abuse, but only the adjudicated FFS Part A and B claims get loaded into the NCH. Note that this data also gets loaded into the most systems of these systems on a daily, weekly, and quarterly basis, which depends on the data feed.

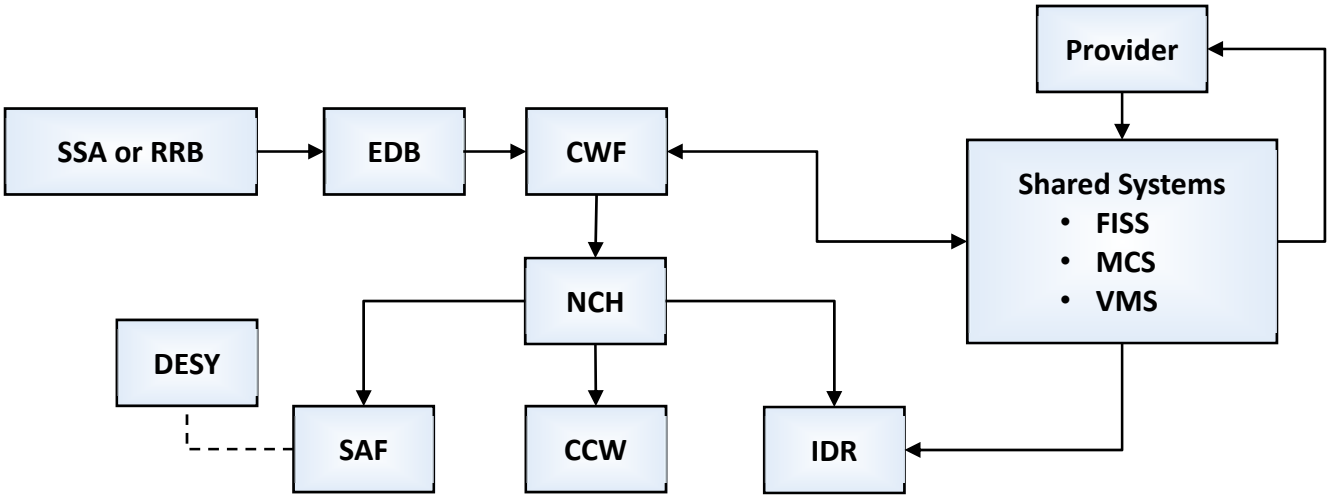


Figure 10: FFS Claim Process Flow

Note that most end-users access the Medicare claims using the IDR, CCW or SAFs and are typically not given access to the NCH files, which are usually accessed only by system programmers. This section is meant to provide additional background on the structure of the NCH data as the SAFs follow a similar pattern. The datasets composing the NCH are received to the CMS mainframe from the CWF on a weekly basis and are accumulated into monthly and quarterly files. The files are based on beneficiary-state and on time frame (inside or outside the 18-month window). The "A*" files are quarterly; the M1 files are monthly; and the M2 and M3 files hold data outside the 18-month window.

CLAIM RUN-OUT

One of the major differences between the various sources of FFS data is claim run-out. According to section 6404 of the Affordable Care Act (ACA), a provider has up to 12 months to submit a claim after the service date. For reference, most claims are submitted within the first three months of service, but sometimes providers submit claims late. Also, if the hospital or the A/B Medicare Administrative Contractor (MAC) discovers an error after a bill is submitted, the hospital submits an adjustment request using the ASC X12 837 institutional claim format or the Form CMS-1450. If a claim has changed as a result of medical review, after an original bill has been forwarded to CMS, adjustment debit/credit bills are required. The corrected bill must be an exact duplicate of the original, except for any changed fields including diagnostic and procedure codes.

Figure 11 illustrates that the NCH "A6" record has a 6-month run-out. The A6 record is an annual file that is no longer updated after the 6-month cut-off. Therefore, new claims can be added to the database or adjustments can be made until that date. Note that the SAFs are created from the A6 record and are often referred to as the 18-month SAF. However, the CCW is updated up to 12 months after the end of the calendar year. This means an additional 6 months of claims and/or adjustments may be received.

Thus, the claim payments may be higher or lower depending on the nature of the adjustments. Note that the IDR is updated weekly and does not use a cut-off date. Therefore, when using the final action indicator within the IDR the claim payments can be quite different than in the other databases because claims can often be adjusted years after they are submitted. Therefore, be cautious when using final action claims within the IDR to perform time series analysis.

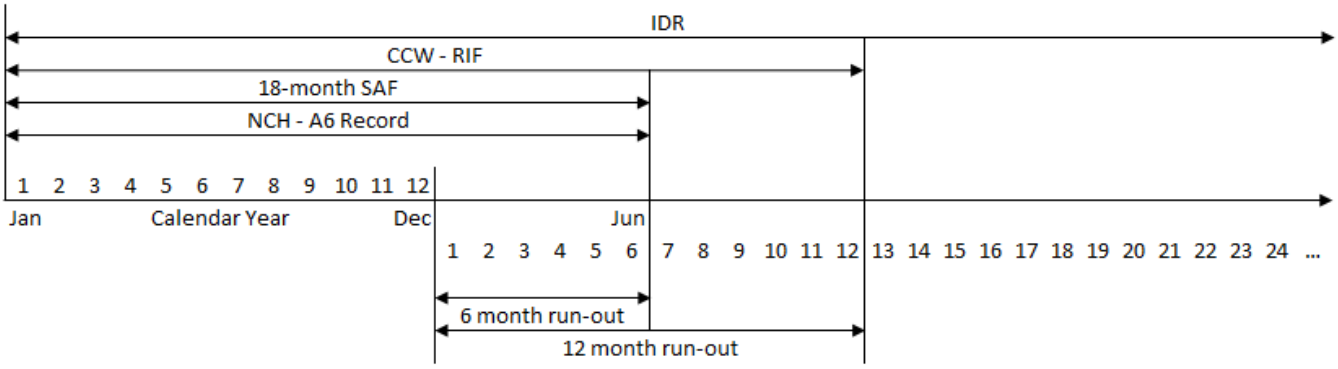


Figure 11: Claim Run-out Example

Table 2 is a comparison of claim run-out for various service types for calendars years 2006 through 2015. Notice that the inpatient dollars consistently increased and stayed constant in 2006-2007; but, in 2008-2012 the dollars started going down after 6 to 12 months. This was due to The Recovery Audit Contractor (RAC) program, which was created through the Medicare Modernization Act (MMA) of 2003 to identify and recover improper Medicare payments to healthcare providers under fee-for-service (FFS) Medicare plans (RAC,2019). Also note that the dollars leveled off more quickly starting around 2013, except for home health dollars, largely a result of the Audit Recovery effort and other claim assessments. The reason that run-out is important is that claim payment levels can change depending on the length of run-out. This is especially significant when using the IDR as described in Figure 11: Claim Run-out Example. If a researcher wishes to replicate an exact number when using the IDR, either debit/credit or as-was processing should be used.

Table 2: Claim Run-out Comparison for CY 2006-2015

Claim Type	Inpatient	Skilled Nursing	Home Health	Hospice	Outpatient	Physician	Durable Equipment	Total	Run-out Months
2006									48
2007									48
2008									48
2009									48
2010									48
2011									48
2012									48
2013									38
2014									26
2015									14

CLAIM COUNTS

There are about 1 billion FFS claims every year, which produce about 3 billion claims lines (or record counts), since there are often multiple services provided (resulting in multiple records) for each claim. Most tables in the VRDC environment (CCW) are segmented by (1) year, (2) institutional or professional, and (3) claim type. Segmenting the data (which is more extensively done in the CCW than in the IDR) makes access easier and reduces the risk of extracting too much data resulting in your query running out of resources and timing out. For example, the 2018 CCW carrier/physician table (claim type = 71 and 72) for January (BCARRIER_LINE_01) contains 157 ***million*** records. In contrast, the claim line table in the IDR (CLM_LINE) contains 188 ***billion*** records. This is because the IDR tables, unlike the CCW, contain all claim types for Parts A, B, C and D, as well as the claims data for the shared systems data from FISS, MCS, and VMS.

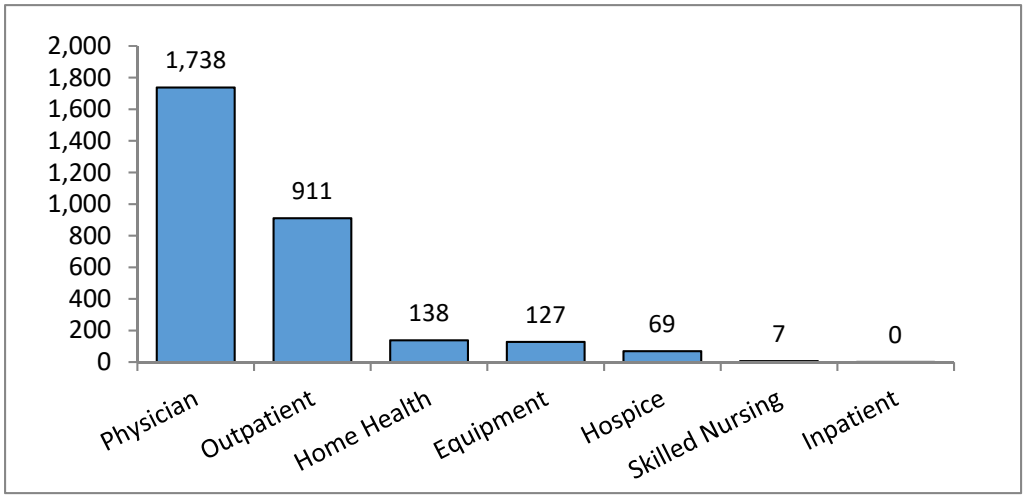


Figure 12: NCH Claim Count Example for One Year (millions)

CLAIM PAYMENTS

The CMS Program Statistics (CPC, 2017) reports $377 billion dollars in annual FFS claim payments. As shown in Figure 13, inpatient payments account for $125 million (33 percent) of those total FFS payments, in contrast to the record counts for inpatient services, which account for only about 0.01 percent of the total claim lines.

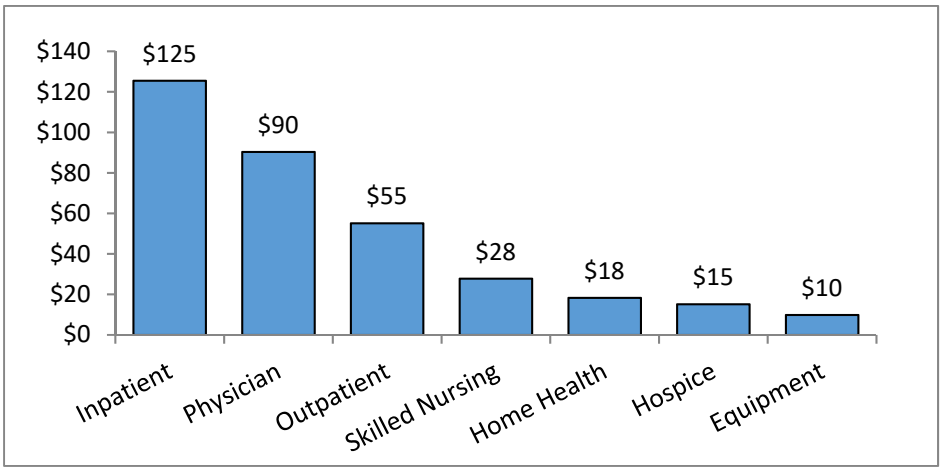


Figure 13: NCH Claim Payment Example for One Year (millions)

IDR AND CCW DIFFERENCES

There are numerous differences between the IDR and the VRDC/CCW. For example, the FFS claims within the IDR only go back to 2006, whereas the claims in the VRDC, in both the CCW Oracle tables and the RIF SAS Data Warehouse, go back to 1999. The reason the IDR only goes back to 2006 is that is when the Part D drug events went into effect they decided to only load the Part A and B claims from 2006 forward to match Part D.

Table 3: Example of Differences between the IDR and VRDC/CCW

Description	VRDC/CCW	IDR
Part A, B and D claims Data	1999 - present	2006 - Present
Part A & B Risk Scores	2006 - Present	2006 – Present
Part D Risk Scores	2006 - Present	2006 – Present
Medicaid data	Medicaid Extract (MAX)	Transformed Medicaid Statistical Information System (TMSIS)
Medicare Provider Analysis & Review (MedPAR)	Start in 1991	Start in 2007
Denominator	1999 - Present	1999 - Present
Common Medicare Enrollment (CME) tables	- Enrollment Database	- Enrollment Database - Medicare Bene. Database - Transplant, Dialysis, and Hospice Tables
Shared Systems Claims (FISS, MCS, VMS)	N/A	2006 - present
Affordable Care Organization (ACO)	N/A	Data loaded June 2012
Hospital-based, Allowed Charges, HPSA	N/A	2011 - Present
Minimum Data Set (MDS)	1999 - present	N/A
Outcome & Assessment Information Set (OASIS)	2000 - present	N/A
Medicare Current Beneficiary Survey (MCBS)	1991 - present)	N/A
National Death Index (NDI)	Started in 1999	N/A

ACCESSING THE DATA

NATIONAL CLAIMS HISTORY

The NCH files exist on the CMS mainframe in variable length flat files on tape data sets and are accessed using batch programs with Job Control Language (Item 1 below). Item 2 shows how to use a LIBNAME statement (although this step can also be done in JCL) to create an output library. Note that the NCH file must be identified within the JCL because it is a tape data set. It is always a good idea to look at a few records first to make sure the INPUT statement is working correctly (Item 3). Item 4 is an efficiency technique where two fields are input and then used to inform the process whether or not to read the remainder of the record. Notice the use of the @ symbol to keep the pointer on the current record.

The code below is an example batch program used to read the NCH data:

```
//MYIDNCH1 JOB (IDRP000),'SAS\NCH',CLASS=H,
//     NOTIFY=&SYSUID,MSGCLASS=Q,MSGLEVEL=(1,1)        1
/*JOBPARM LINES=9999
//*************************************************************
//* PROGRAM: MYID.SGF2020.SASPROG(NCH)
//* PURPOSE: EXAMPLE OF USING SAS TO READ THE NCH
//* CREATOR: RICK ANDREWS, OACT
//* CREATED: 03/29/2020
//*************************************************************
//STEP01 EXEC PROC=SAS
//NCH    DD DSN=P#UTL.#NCHP.CRT.NLK.A1.Y16.S01B,DISP=SHR   2
//SYSIN  DD *

libname MYDATA "&SYSUID..SGF2020.SASDATA " disp=new /*old*/
  space=(cyl,(500,250),rlse);

data mydata.NCH_DATA;
 infile NCH missover; *obs=100;                          3

 input
  @7   CLM_TYPE_CD $2.
  @106 PROVIDER    $6. @;                                 4

 if CLM_TYPE_CD = '60'
 and PROVIDER = '013300' then do;

   input
     @105 CLM_QUERY_CD   $1.
     @229 NPMT_RSN_CD    $2.
     @232 CLM_PMT_AMT ?? PD6.2;                          5

   if  NPMT_RSN_CD  = ' '
   and CLM_QUERY_CD = '0'
   then CLM_PMT_AMT = CLM_PMT_AMT * (-1);
   else CLM_PMT_AMT = CLM_PMT_AMT * (1);

   output;                                               6
 end;
run;
```

Item 5 is an example of using a format modifier to prevent error messages in the log when invalid data are encountered. This usually happens with numeric variables.

Item 6 is an example of the debit/credit processing. Since the NCH contains all debit/credit pairs, the claim payment needs to be converted to a negative when a credit is encountered. As an alternative, final action processing can also be employed.

DESY / SAF

The Data Extract System is another mechanism to obtain FFS claims. Although the NCH and SAF files can be processed using DESY, the research community generally use the SAFs because they employ final action processing, making it easier to summarize data.

The SAFs utilize the A6 records from the NCH, which have a 6-month run-out and are often referred to as the 18-month files. This means the results will always be the same no matter when the process is submitted.

One other major difference between the NCH and DESY SAFs is a 50-byte header containing variables created by DESY to keep track of each request.

When the process has completed, the user receives an email indicating the number of records that were output and the name of the library to which the data were output. An example of the filename is listed below.

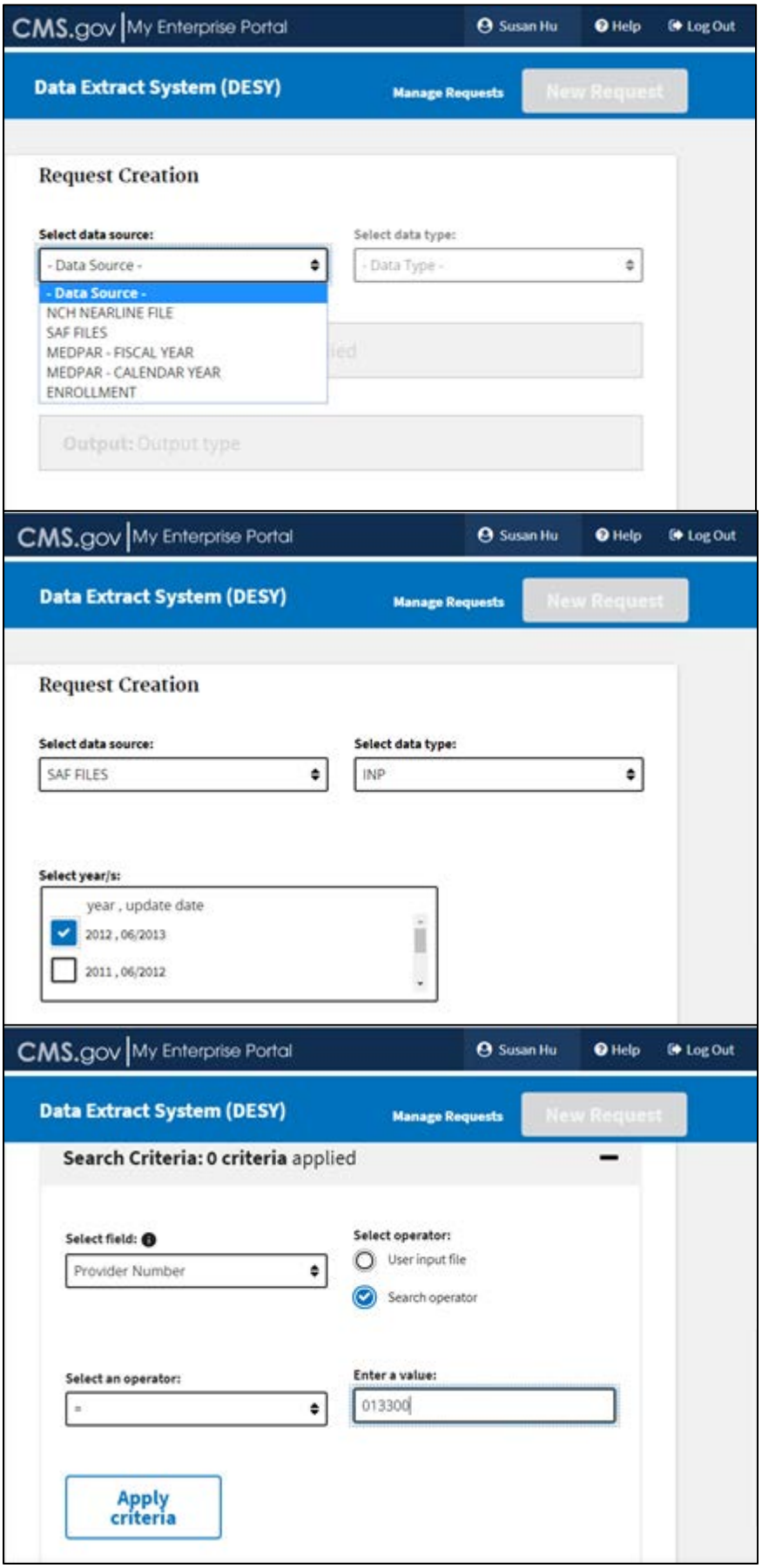
P#DSY.@AAA2049...R0160049.OUT

As a result, the pointers within the INPUT statement should be increased by 50 bytes as shown in this example:

```
input
  @57   CLM_TYPE_CD     $2.
  @155  CLM_QUERY_CD    $1.
  @279  NPMT_RSN_CD     $2.
  @282  CLM_PMT_AMT ??  PD6.2;
```

If the entire record is for either the NCH or SAFs, the INPUT statements have been supplied by CMS and are located on the Internet at the reference listed in (DESY, 2019). Note that CMS employees and contractors can gain access to the SAFs on the mainframe by requesting the following profile:

P#UTL.#SAFP.CRT.AN*.FA.*



Display 1: Example DESY Request

IDR

As mentioned, the IDR exists within the BDC and can be accessed through a mainframe channel, personal computer using a special VPN, or a SAS Enterprise Business Intelligence (EBI) server. SAS/EBI offers a number of ways to process data within the database, including Enterprise Guide®, Enterprise Miner®, implicit and explicit queries, and SAS Studio.

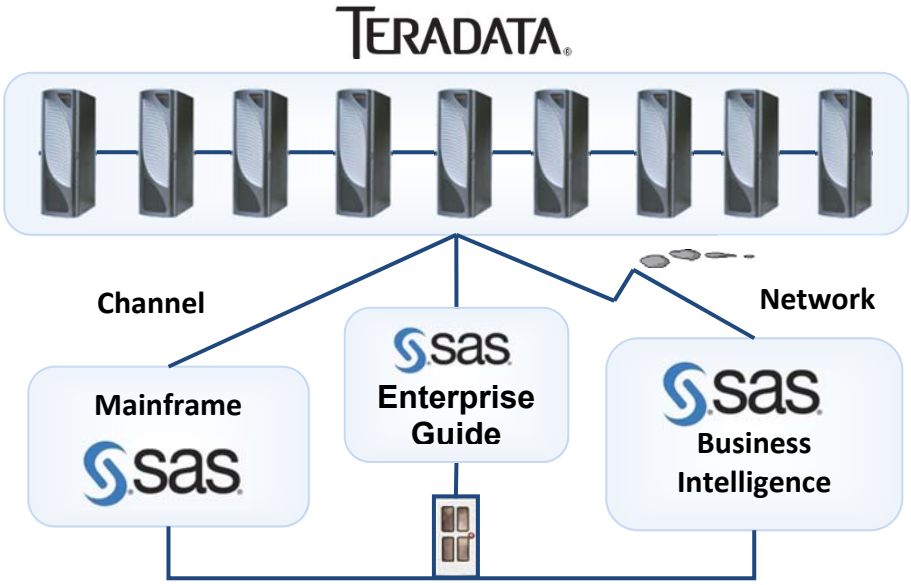


Figure 14: Enterprise Architecture Example

Mainframe

The SAS System offers a product called SAS/Access to Teradata, which presents options to data users that allow them to access data by sending a request to the database implicitly or explicitly. An implicit query is one where SAS attempts to convert traditional SAS programming into SQL to be used by the database. This is very useful, although it can be problematic if used incorrectly as will be demonstrated later. An explicit query is one where the user sends a pass-through query to Teradata for processing. This method removes the possibility of error with the interpretation of programming code, though knowledge of SQL specifically related to Teradata is required.

SAS/Access to Teradata

1 The first thing that happens in a pass-through query is the CONNECT statement. Notice the query is connecting to TERADATA and that various parameters are included, which are used only as examples. These parameters may or may not be required in every instance of a Teradata database, which will be demonstrated later.

2 The second step identifies the output data set to create in SAS. Notice in the FROM clause, CONNECTION TO TERADATA, which sends the pass-thru query via a derived table.

3 The last step is the actual Teradata query that is “passed” through to the database. The use of ANSI standard SQL is recommended so that queries are compatible across platforms and applications.

Example explicit pass-through query:

```
proc sql;
 connect to teradata (                    1
   user= "td_&sysuserid"
   mode= teradata
   tdpid= TPD1 );

 create table work.sas_data_set as
 select *
 from connection to teradata              2
 (
  select THIS, AND, THAT                  3
  from TERADATA_TABLE
 );

 disconnect from teradata;
quit;
```

When accessing the IDR through the mainframe, the user should also create a JCL program. There are a number of changes that need to be made to the JCL code in order to send the query to the production environment to process the data as shown below.

Here is an example of an explicit IDR pass-through query using the CMS mainframe:

```
//MYIDIDR1 JOB (IDRP000),'SAS/TERADATA',GROUP=IDRSASP,
//     NOTIFY=&SYSUID,MSGCLASS=Q,MSGLEVEL=(1,1),CLASS=O
/*JOBPARM LINES=9999
/*JBS BIND DB2P
//*************************************************************
//* PROGRAM: MYID.SGF2012.SASPROG(IDR1)
//* PURPOSE: EXAMPLE OF USING THE MAINFRAME AGAINST THE IDR
//* CREATOR: RICK ANDREWS, OACT
//* CREATED: 03/29/2020
//*************************************************************
//JS010  EXEC PROC=SAS,
//    LOAD='HCFA1.@TERADTA.P1.APPLOAD',
//    COND=(0,NE)
//SYSIN  DD *

libname MYDATA "&SYSUID..SGF2020.SASDATA" disp=old /*new*/
  space=(cyl,(500,250),rlse);

proc sql;
 connect to teradata (
  USER=&SYSUID PASS='' TDPID=TDP1
  MODE=TERADATA DATABASE=CMS_VDM_VIEW_MDCR_PRD );

 create table MYDATA.SGF2020_IDR as
 select *
 from connection to TERADATA
  (
   explain /*<-- Use EXPLAIN to evaluate query */
   select
   /*top 10 /*<-- Prevent runaway query */
      CLM.CLM_BLG_PRVDR_OSCAR_NUM
     ,sum(CLM.CLM_PMT_AMT) as CLM_PMT_AMT

   from V2_MDCR_CLM as CLM

   where CLM.CLM_TYPE_CD = 60
     and CLM.CLM_FINL_ACTN_IND = 'Y'
     and CLM.CLM_BLG_PRVDR_OSCAR_NUM = '013300'
     and CLM.CLM_THRU_DT between '2016-01-01' and '2016-12-31'

   group by
      CLM.CLM_BLG_PRVDR_OSCAR_NUM
  );
 disconnect from TERADATA;
quit;
```

Mainframe IDR Program Description

1 When accessing the IDR, special parameters must be used within the JOBCARD. Set the GROUP parameter to “IDRSASP” and the CLASS parameter to “O”.

2 Since end-users of the CMS mainframe are required to write programs within the Development environment, the process must be bound to the Production LPAR by using the Job Binding Service (JBS). Thus, a BIND command is used with the DB2P parameter.

3 In order to create a mainframe channel to the IDR, use the following LOAD module: HCFA1.@TERADTA.P1.APPLOAD and submit it within the EXEC statement.

4 This example utilizes a SAS/ACCESS explicit pass-through query and is done using a CONNECT statement within the SQL procedure. Note that the parameters used are specific to the mainframe and the DATABASE option that is set to CMS_VDM_VIEW_MDCR_PRD is optional.

5 The CREATE TABLE statement is still a part of the SAS code, but notice the CONNECTION to TERADATA parameter in the FROM clause. This tells SAS that everything within the open and close parenthesis is Teradata-specific SQL.

6 Teradata has an option to create an EXPLAIN plan, which provides the user a guess as to how many observations will be returned and how long the query will take to run. It is a very good idea to always submit an "explain plan" when building a new query. Figure 15 below is an example of an explain plan. The last few rows of the report indicate that 84,304 records will be returned and the query will only take 0.42 seconds. If the estimated rows returned or time to run is very high, the query should not be submitted.

7 Teradata also has an option called TOP that will output only the "top" number of rows after the query has finished. Just as in the mainframe NCH program, it is a good idea to look at the first few rows before moving forward. Note that the TOP function is commented out in the example above. Once the explain plan has been evaluated, comment it out and uncomment the TOP function. After the first few rows have been appraised, it is also a good idea to set the TOP function to a value higher than the expected rows to be returned. This will prevent the unexpected download of hundreds of millions of records.

8 Notice that the example here is using the final action indicator of "Y". This will keep the most current "final" claim, if an adjustment has been made. Keep in mind that the IDR performs final action processing every week. Thus, unlike the SAFs or the RIF within the VRDC, the total claim payment amounts can change.

67	7) We do an all-AMPs RETRIEVE step in TD_Map1 from Spool 5 (Last Use)
68	by way of an all-rows scan into Spool 1 (group_amps), which is
69	built locally on the AMPs. The size of Spool 1 is estimated with
70	no confidence to be 84,304 rows (3,119,248 bytes). The estimated
71	time for this step is 0.00 seconds.
72	-> The contents of Spool 1 are sent back to the user as the result of
73	statement 1. The total estimated time is 0.42 seconds.

Figure 15: Teradata Explain Plan Example

Debit/Credit Processing

Debit/Credit processing is used by the adjudication systems (FISS, MCS, and VMS) to apply adjustments to FFS claims. When an adjustment is made, a debit/credit pair is created. The credit negates the original debit and, if necessary (but not always), a new debit record is created. Final action keeps the latest debit, which makes processing much easier for researchers. If debit/credit is desired, the example below describes the IDR code needed.

Description

1. Note that the EXPLAIN clause has been commented out and the TOP clause has been uncommented. This is not a part of debit/credit processing, but instead is an example showing what to do after the EXPLAIN plan has been evaluated.
2. When performing debit/credit in the IDR, there is no need to look at the non-payment reason or claim query codes as the database contains a variable for the claim adjustment type code. If the code contains the number 0 or 2, then the claim payment amount is multiplied by 1, which essentially does nothing. If the code is the number 1, then the claim payment needs to be multiplied by negative one.
3. Notice that the final action indicator has been removed.
4. Also note that the claim effective date has been added with a date six months after the end of the calendar year of the incurred date of the claim. This will provide a summary with a six month run-out, similar to the "A6" record of the NCH.

Note: One issue with using debit/credit processing is that it needs to be done on all aggregate functions.

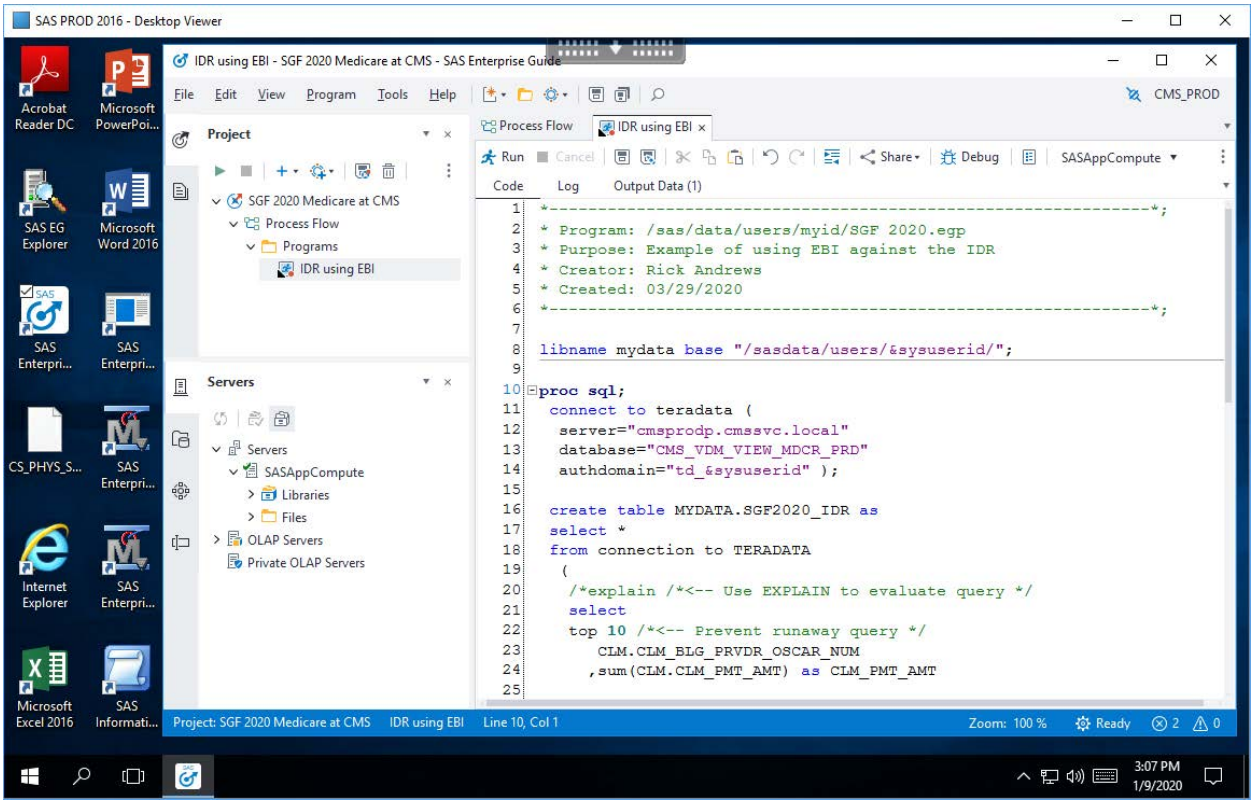
Below is an example of using debit/credit processing using the IDR:

```
/*explain /*<-- Use EXPLAIN to evaluate a query */
select
top 10 /*<-- Prevent a runaway query */                    1
   CLM.CLM_BLG_PRVDR_OSCAR_NUM

  ,sum(CLM.CLM_PMT_AMT *
       case CLM.CLM_ADJSTMT_TYPE_CD
         when '0' then 1
         when '2' then 1
         when '1' then -1                                  2
         else 0
       end)
      as CLM_PMT_AMT

from V2_MDCR_CLM as CLM                                    3

where CLM.CLM_TYPE_CD = 60
  and CLM.CLM_BLG_PRVDR_OSCAR_NUM = '013300'
  and CLM.CLM_THRU_DT between '2016-01-01' and '2016-12-31'
  and CLM.CLM_EFCTV_DT <= '2017-06-30'

group by
   CLM.CLM_BLG_PRVDR_OSCAR_NUM                             4
```

SAS Enterprise Business Intelligence

SAS Enterprise Business Intelligence (EBI) is employed using a client/server environment utilizing Citrix®, which is a Windows terminal emulator that allows a user to logon to an external box, similar to the mainframe IBM Host On-Demand application. It allows a user to submit interactive or batch processes using either a SAS/ACCESS implicit or explicit pass-through query.

SAS Enterprise Guide



Display 2: EBI example against the IDR

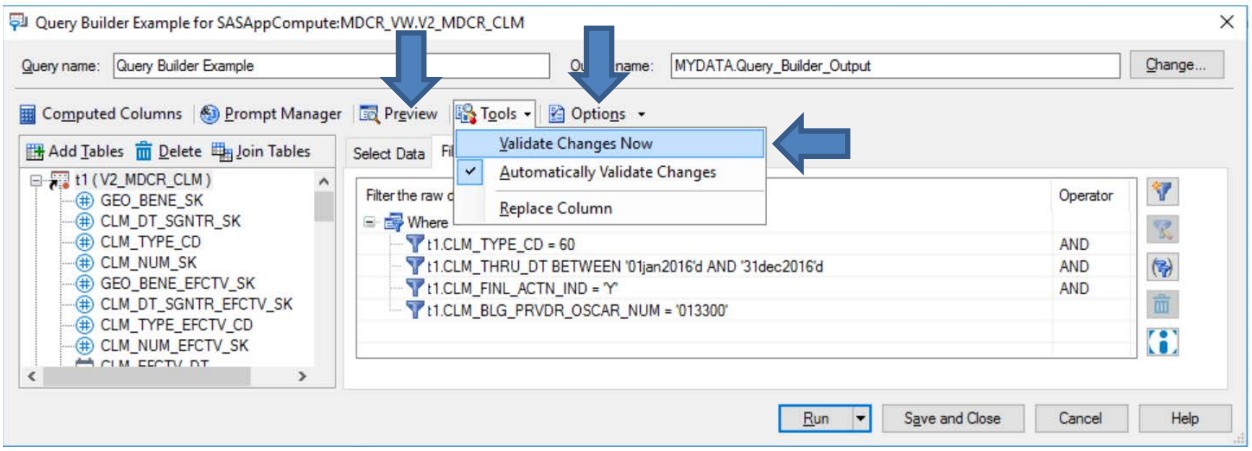
Most researchers utilize the SAS Enterprise Guide product to access the IDR. The differences between the mainframe Enterprise Guide are many, but the differences in the programming code are not. Other than removing the JCL, the only code that needs to change is the following:

```
libname mydata base "/sasdata/users/&sysuserid/";

proc sql;
  connect to teradata (
    server="cmsprodp.cmssvc.local"
    database="CMS_VDM_VIEW_MDCR_PRD"
    authdomain="td_&sysuserid" );
```

The only code that needs to change is the file path within the LIBNAME statement and the connection string within the CONNECT statement. The remainder of the query runs exactly the same, which can be very helpful.

Query Builder

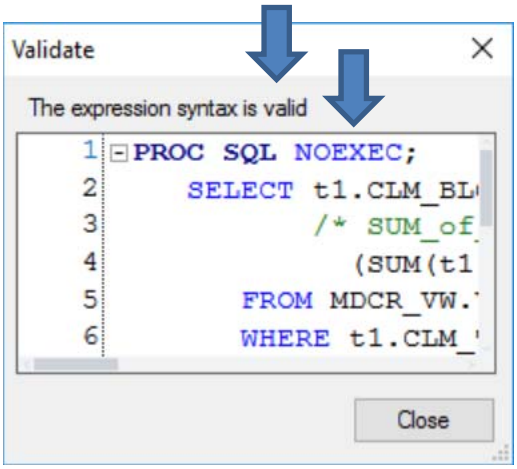
The Query Builder enables a user to extract data from one or more tables according to the criteria needed. It is a point and click tool, which allows a researcher to drag and drop values within a Graphical User Interface (GUI) without having to know any SAS programming code. In the background, the Query Builder generates Structured Query Language (SQL) code that can be viewed and used as the basis for a new program.



Display 3: Query Builder example against the IDR

Implicit Example

In this example, the Query Builder is used to execute an "implicit" query, where SAS is expected to generate the SQL necessary to summarize data from a given database, Teradata in this example.



Display 4: Validate Example

This section is not meant to describe all of the features of the Query Builder, but instead provide a couple that deserve merit. One is the **NOEXEC** option used when choosing the "Validate Changes Now" option within the Tools pull-down menu shown in Display 3: Query Builder example against the IDR. This will present the window described in Display 4, validating the query. Any other result should be scrutinized. Another good thing to do is click on the Options button and limit the number of rows to save in the output, which will add the **OUTOBS** option. Clicking on the Preview button will present the programming shown below:

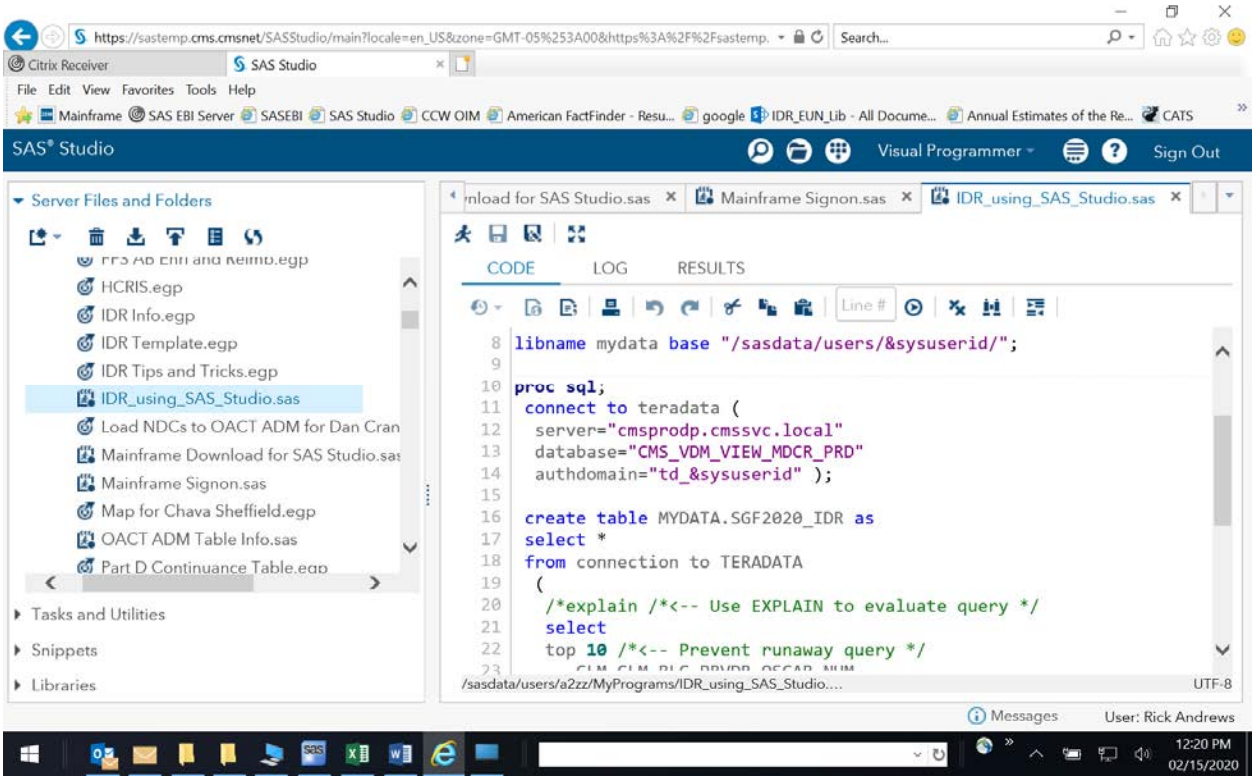
```
PROC SQL OUTOBS=100;
   CREATE TABLE MYDATA.Query_Builder_Output AS
   SELECT t1.CLM_BLG_PRVDR_OSCAR_NUM,
          /* SUM_of_CLM_PMT_AMT */
            (SUM(t1.CLM_PMT_AMT)) FORMAT=17.2 AS SUM_of_CLM_PMT_AMT
      FROM MDCR_VW.V2_MDCR_CLM t1
      WHERE t1.CLM_TYPE_CD = 60
        AND t1.CLM_FINL_ACTN_IND = 'Y'
        AND t1.CLM_BLG_PRVDR_OSCAR_NUM = '013300'
        AND t1.CLM_THRU_DT BETWEEN '01jan2016'd AND '31dec2016'd
      GROUP BY t1.CLM_BLG_PRVDR_OSCAR_NUM
      ORDER BY t1.CLM_BLG_PRVDR_OSCAR_NUM;
QUIT;
```

Notice the SQL code generated from the Query Builder does not contain a CONNECT statement. This is because it uses an implicit pass-through query, which utilizes a LIBNAME statement to create a connection to the database as shown in the code below:

```
libname MDCR_VW teradata database=CMS_VDM_VIEW_MDCR_PRD
  server="cmsprodp.cmssvc.local" authdomain=TD_&sysuserid;
```

SAS Studio

SAS Studio is a developmental web application for SAS that can be used to access data files, libraries, and existing programs and to write new programs through a web browser. Predefined tasks in SAS Studio can also be used to generate SAS code. When using SAS Studio, SAS software is being executed behind the scenes. SAS Studio connects to a SAS server in order to process SAS commands. The SAS server can be hosted in a cloud environment, in a local environment, or in a copy of SAS on a local machine. After the code is processed by the SAS server, the results are returned to SAS Studio in the browser (STUDIO, 2020).



Display 5: SAS Studio Example against the IDR

The same programming code used within Enterprise Guide can be executed from SAS Studio. The only exception is that the program needs to be saved on the server as a true (dot).SAS program such as the one shown in Display 6 as IDR_using_SAS_Studio.sas.

VRDC

CMS offers a secure way of accessing its program data through virtual access to the CMS Virtual Research Data Center (VRDC). The CMS VRDC is a virtual research environment that provides timelier access to Medicare and Medicaid program data in a more efficient and cost effective manner. Researchers working in the CMS VRDC have direct access to approved data files and are able to conduct their analysis within the CMS secure environment. They also have the ability to download aggregated reports and results to their own personal workstation (VRDC, 2020).

The CMS VRDC:

- Satisfies all CMS privacy and security requirements
- Enables researchers to access and perform their own analyses and manipulation of CMS data using the CMS infrastructure
- Enables researchers to upload external data files into their workspace to analyze with the approved CMS data files
- Provides access to the Research Identifiable Files
- Provides access through a Virtual Private Network and virtual desktop.

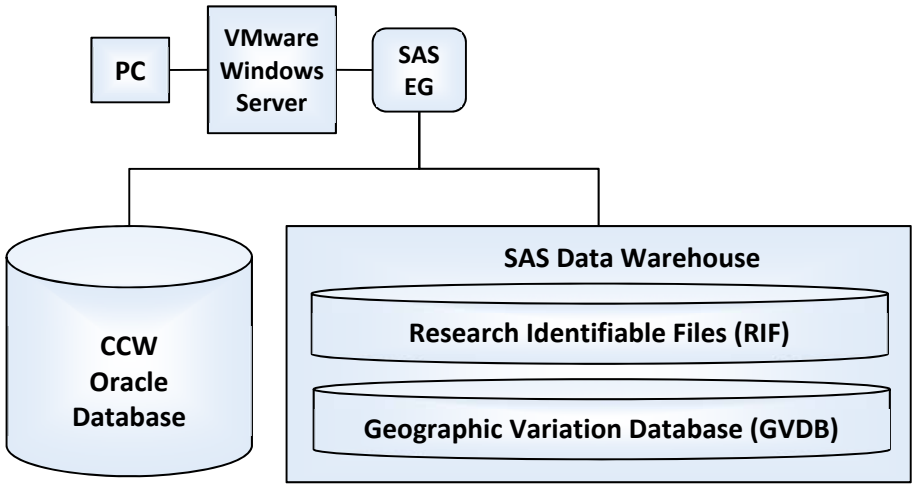


Figure 16: VRDC Access Method

As previously mentioned, the VRDC has three areas where FFS claims can be obtained – CCW, RIF, and GVDB. The CCW is housed within an Oracle database, whereas the RIF and GVDB are housed within a SAS data warehouse. All of these options are accessed using a product called VMware Horizon®.

CCW

Many researchers use SAS Enterprise Guide to access data within the VRDC, but only Power Users are given access to the CCW and ability to utilize SAS Access to Oracle to summarize data from the CCW database. This database was the first data source created to satisfy Section 723 of the MMA and is the basis used for the RIF and GVDB SAS data warehouses.

The following program code is an example of summarizing data from the CCW:

SAS/Access to Oracle

```
%include "&myfiles_root./sasnetrc.sas"/nosource2;

proc sql outobs=10 /*<-- Prevent runaway query      1    */;
 connect to oracle (
   user=&ouser. pw="&opassp." path="&opathp." Preserve_comments=yes );

 create table &sysuserid.sl.SGF2020_CCW as
 select *
 from connection to oracle
  (                                          2
   select /*+ parallel(8)*/
      CCW_PRVDR_AT_TIME_OF_CLM.PRVDR_NUM
     ,sum(CCW_PTA_FACT.clm_pmt_amt) as Claim_Payment

   from CCW_OWNER.CCW_PTA_FACT
       ,CCW_OWNER.CCW_ALL_CLM_PRFL
       ,CCW_OWNER.CCW_FINL_ACTN_PRFL
       ,CCW_OWNER.CCW_PRVDR_AT_TIME_OF_CLM      3

   where CCW_PTA_FACT.CLM_TYPE_ID  = CCW_ALL_CLM_PRFL.CLM_TYPE_ID
     and CCW_PTA_FACT.FINL_ACTN_ID = CCW_FINL_ACTN_PRFL.FINL_ACTN_ID
     and CCW_PTA_FACT.PRVDR_AT_TIME_OF_CLM_ID =
         CCW_PRVDR_AT_TIME_OF_CLM.PRVDR_ID

     and CCW_ALL_CLM_PRFL.NCH_CLM_TYPE_CD = '60'
     and CCW_FINL_ACTN_PRFL.FINL_ACTN_CD  = 'Y'
     and CCW_PRVDR_AT_TIME_OF_CLM.PRVDR_NUM = '013300'
     and CCW_PTA_FACT.CLM_THRU_DT between '01-JAN-2016'
                                      and '31-DEC-2016'
     /*and ROWNUM < 11 /*<-- Used for testing */

   group by                                    4
     CCW_PRVDR_AT_TIME_OF_CLM.PRVDR_NUM
  );
 disconnect from oracle;
quit;
```

1. Notice that the **OUTOBS** option is being used to prevent a runaway query and that the Oracle option is utilized to connect to the CCW database.
2. Also note that the **PARALLEL** option is being used to run the Oracle pass-through query in parallel, if appropriate. This can greatly increase the speed of the query.
3. For the most part, the star schema typically reduces the number of joins needed for a query, although, in this case, it was determined that splitting the data increases the query's efficiency. Notice that the joins needed are much simpler than the IDR.
4. Finally, the variables needed for the WHERE clause are very similar to the IDR and the **ROWNUM** can be used to limit the number of records read from the database while testing the query.

Research Identifiable Files

The RIF are segmented by claim type, year, and month. Thus, summarizing all FFS data for one calendar year, seven claim types, and twelve months would yield 84 SAS data sets that would need to be aggregated.

Below is an example of summarizing one month of inpatient RIF data:

```
proc summary data=RIF2016.INPATIENT_CLAIMS_01 nway missing;
  class PRVDR_NUM;
  var CLM_PMT_AMT;
  where NCH_CLM_TYPE_CD = '60'
    and PRVDR_NUM = '013300';
  output out=MYDATA.SGF2020_RIF_SMRY_01 sum=;
quit;
```

One way to cycle over each month is to create a SAS macro to group the data by month and then to subsequently summarize the results by year as shown below:

```
%macro runit;

 * Ensure final data set does not exist;
 proc delete data=work.SGF2020_RIF_DS; run;

 * Cycle by month;
 %do i=1 %to 12;

 * Append leading zero to months 1-9;
 data _null_;
   call symput('mth',put(&i,z2.));
 run;

 * Summarize by provider and month;
 proc summary data=RIF2016.INPATIENT_CLAIMS_&mth nway missing;
   class PRVDR_NUM;
   var CLM_PMT_AMT;
   where NCH_CLM_TYPE_CD = '60'
     and PRVDR_NUM = '013300';
   output out=work.SGF2020_RIF_TEMP sum=;
 quit;

 * Append temporary summary for further aggregation;
 proc append data=work.SGF2020_RIF_TEMP out=work.SGF2020_RIF_DS; run;

 %end;

%mend runit;
%runit;

* Summarize all 12 months;
proc summary data=work.SGF2020_RIF_DS nway missing;
  class PRVDR_NUM;
  var CLM_PMT_AMT;
  output out=MYDATA.SGF2020_RIF_DS sum=;
run;
```

Geographic Variation Database

The GVDB was created to enable researchers and policymakers to evaluate geographic variation in the utilization and quality of health care services for the Medicare FFS population. The Geographic Variation Public Use File includes demographic, spending, utilization, and quality indicators at the state level, hospital referral region (HRR) level, and county level (GVDB, 2020).

Below is an example of using the GVDB:

```
proc summary data=GVCLM24.PTA_CLAIM_2016 nway missing;
  class PROVIDER_ID;
  var ACTUAL_PMT;
  where NCH_CLM_TYPE_CD = '60'
    and PROVIDER_ID = '013300';
  output out=MYDATA.SGF2020_GVDB_SMRY sum=CLM_PMT_AMT;
quit;
```

CONCLUSION

With the advent of the IDR and CCW environments, obtaining CMS data has become easier and faster than ever. The IDR was initially meant to be the repository of all Part D claims, enrollment, risk, and provider data; whereas the CCW was created to develop research initiatives and propose intervention demonstration programs to provide better health care for chronically ill Medicare beneficiaries. Since their inceptions, both environments have gone through many changes to include all Part A, B, C, and D data so either system can be used for most projects. If a researcher is looking for the easiest method for obtaining data, the CCW is a good choice as the system was designed for ease of use. If there are data elements that do not currently exist in the CCW, the IDR is a good choice as it was meant to house all of the data elements and was designed to run extremely fast on a Massively Parallel Processing system.

REFERENCES

CCW. 2006. "Medicare Part D Data". Accessed January 7, 2020. https://www.federalregister.gov/documents/2006/10/18/06-8750/medicare-program-medicare-part-d-data

NCH. 2006. "System of Records". Accessed January 5, 2020. https://www.federalregister.gov/documents/2006/11/20/E6-19505/privacy-act-of-1974-report-of-a-modified-or-altered-system-of-records

STAR. 2019. "Star Schema". Accessed January 8, 2020. https://www.datawarehouse4u.info/Data-warehouse-schema-architecture-star-schema.html

SCHEMA. 2019. "Schema Modeling Techniques". Accessed January 8, 2020. https://docs.oracle.com/cd/B10500_01/server.920/a96520/schemas.htm

GRID. 2019. "SAS® Grid Manager". Accessed January 14, 2020. https://support.sas.com/software/products/gridmgr/index.html

RAC. 2019. "Recovery Audit Program". Accessed January 12, 2020. https://www.cms.gov/Research-Statistics-Data-and-Systems/Monitoring-Programs/Medicare-FFS-Compliance-Programs/Recovery-Audit-Program/Index.html

DESY. 2019. "Data Extract System". Accessed January 10, 2020. https://www.cms.gov/Research-Statistics-Data-and-Systems/CMS-Information-Technology/desy/

RIF. 2019. “Data Administration”. Accessed January 9, 2020. https://www.cms.gov/Research-Statistics-Data-and-Systems/CMS-Information-Technology/DataAdmin

ANDREWS. 2011. “Teradata® for the SAS® Programmer”. Accessed January 14, 2020. http://support.sas.com/resources/papers/proceedings11/019-2011.pdf

STUDIO. 2020. “SAS Studio”. Accessed January 11, 2020. https://support.sas.com/software/products/sas-studio/faq/SASStudio_whatis.htm

VRDC. 2020. “Virtual Research Data Center”. Accessed January 8, 2020. https://www.resdac.org/cms-virtual-research-data-center-vrdc

GVDB. 2020. “Geographic Variation”. Accessed January 15, 2020. https://www.cms.gov/Research-Statistics-Data-and-Systems/Statistics-Trends-and-Reports/Medicare-Geographic-Variation/GV_PUF

CONTACT INFORMATION

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

Rick Andrews
Centers for Medicare and Medicaid Services
Richard.Andrews@cms.hhs.gov
(410) 786-6395