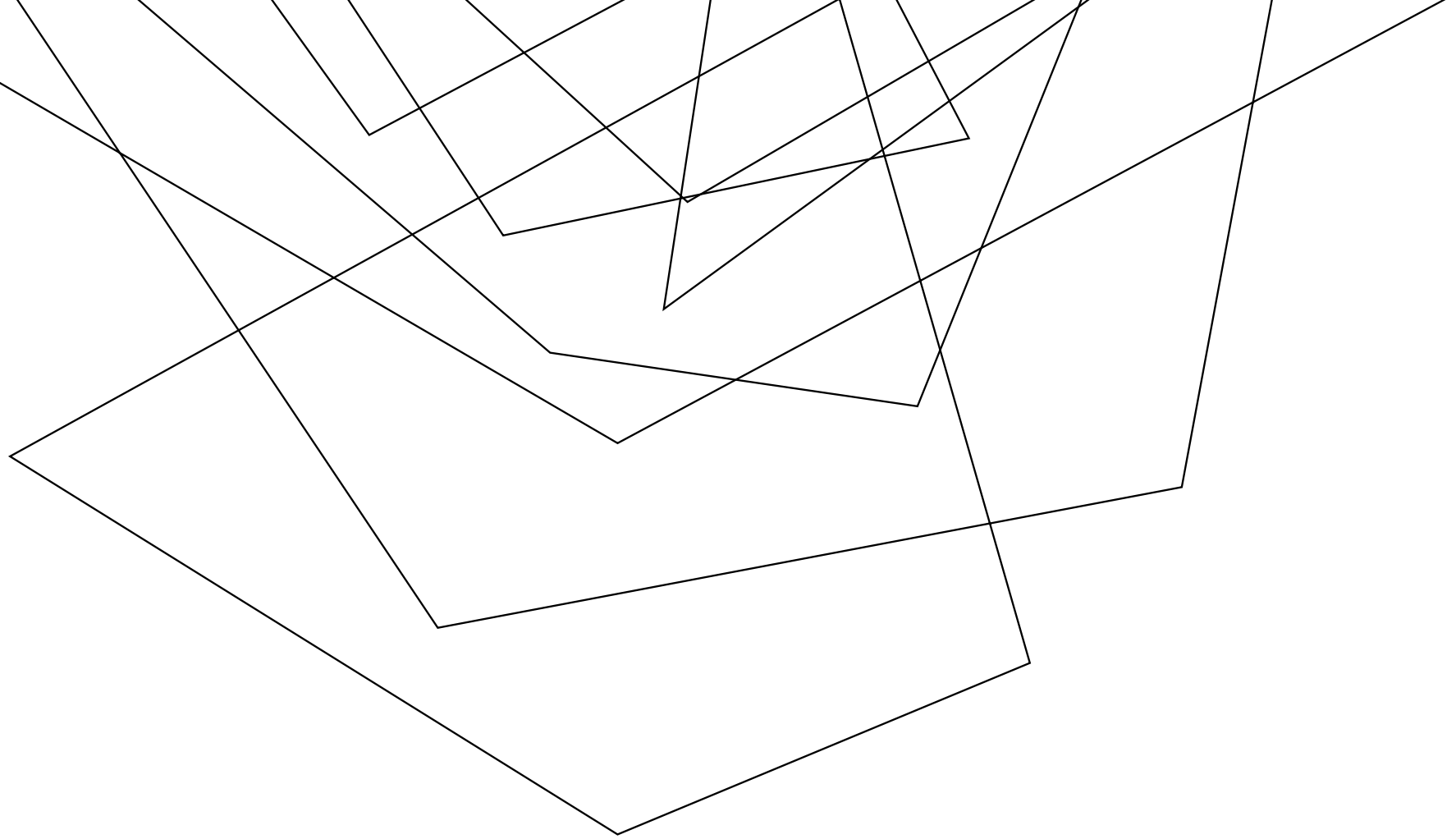


Migrating Data - Duct Tape and Baling Twine

Harry Droogendyk
Stratia Consulting Inc.

AGENDA

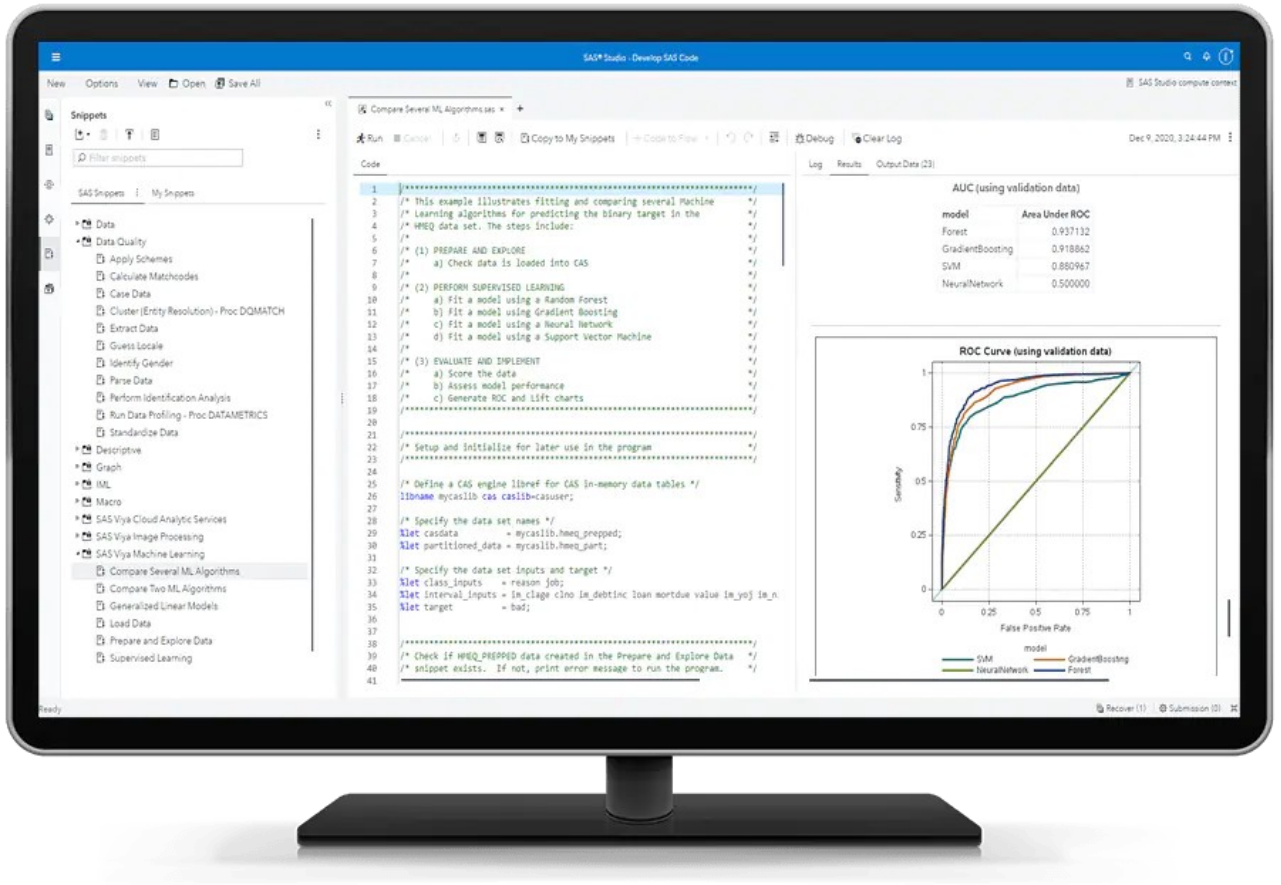
- Introduction
- Constraints
- Process
- Parallel processing
- Lessons Learned

Introduction – Previous Project

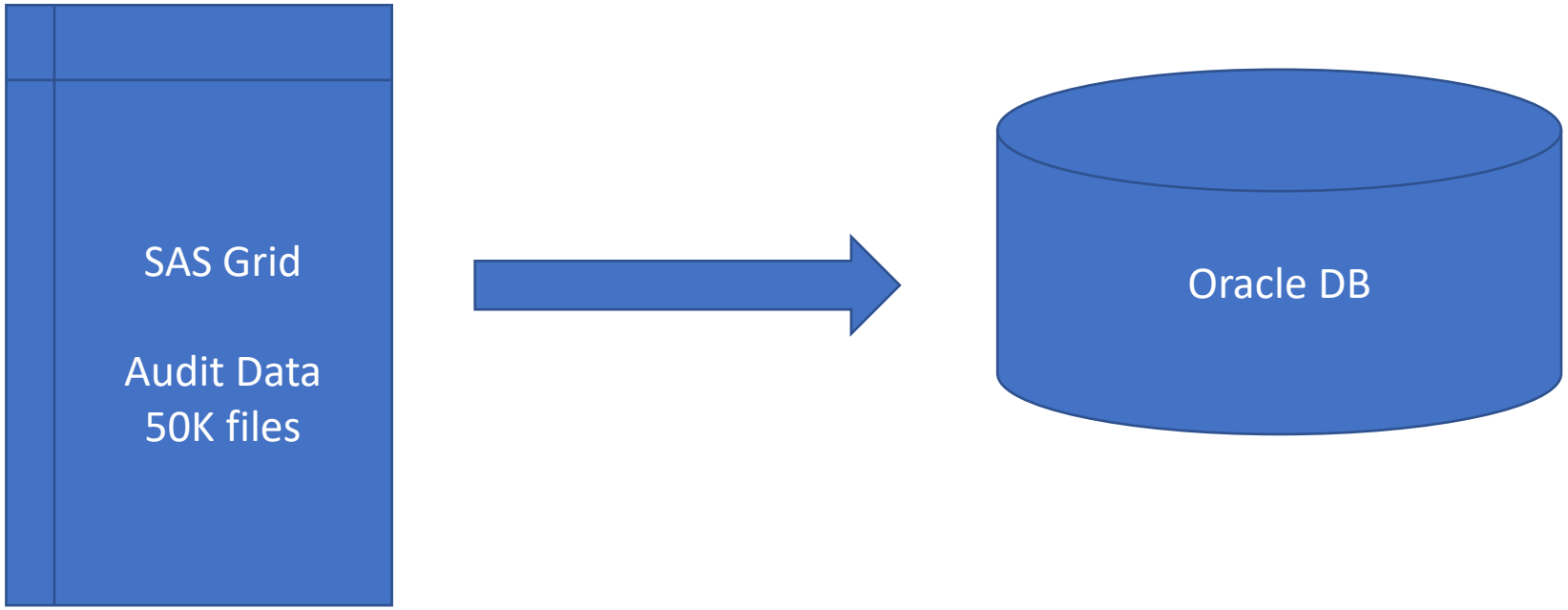
SAS Grid

Process & Data

Day to day use



Introduction - Migrate Audit Data



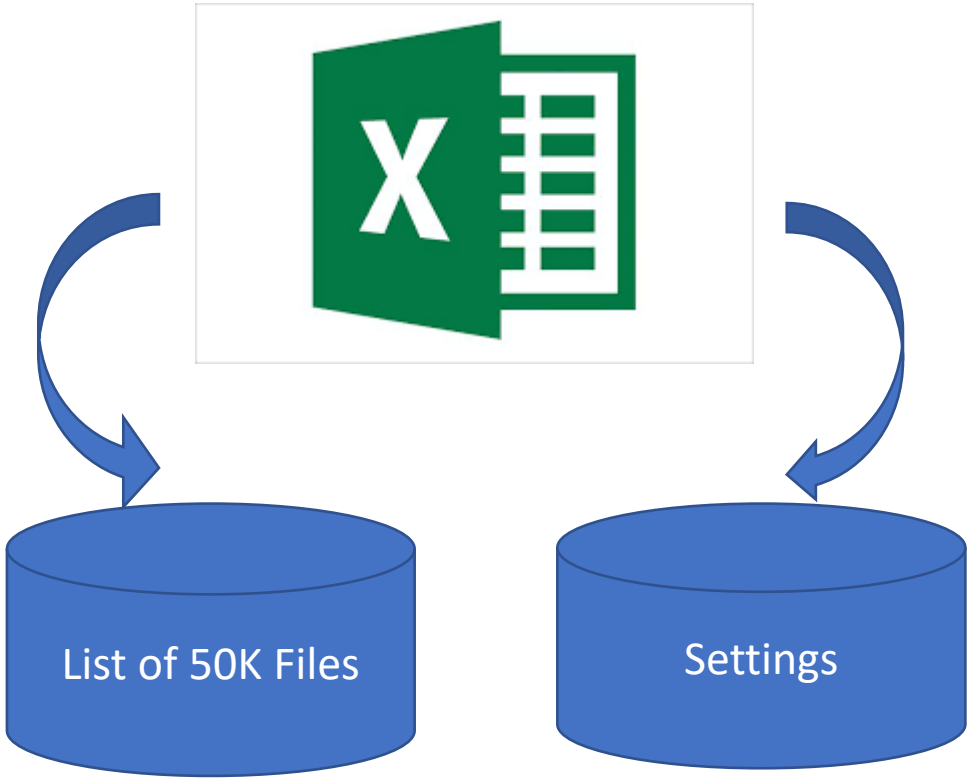
Introduction - Constraints

- SAS software availability
  - SAS Access for Oracle not approved for loading data
  - SAS Share not licensed

- Processing
  - Unix mount point capacity limits
  - Daily HSM recall limit
  - Landing Zone capacity limit
  - Oracle limitations
    - Number of columns, column width
  - No Unix scheduler

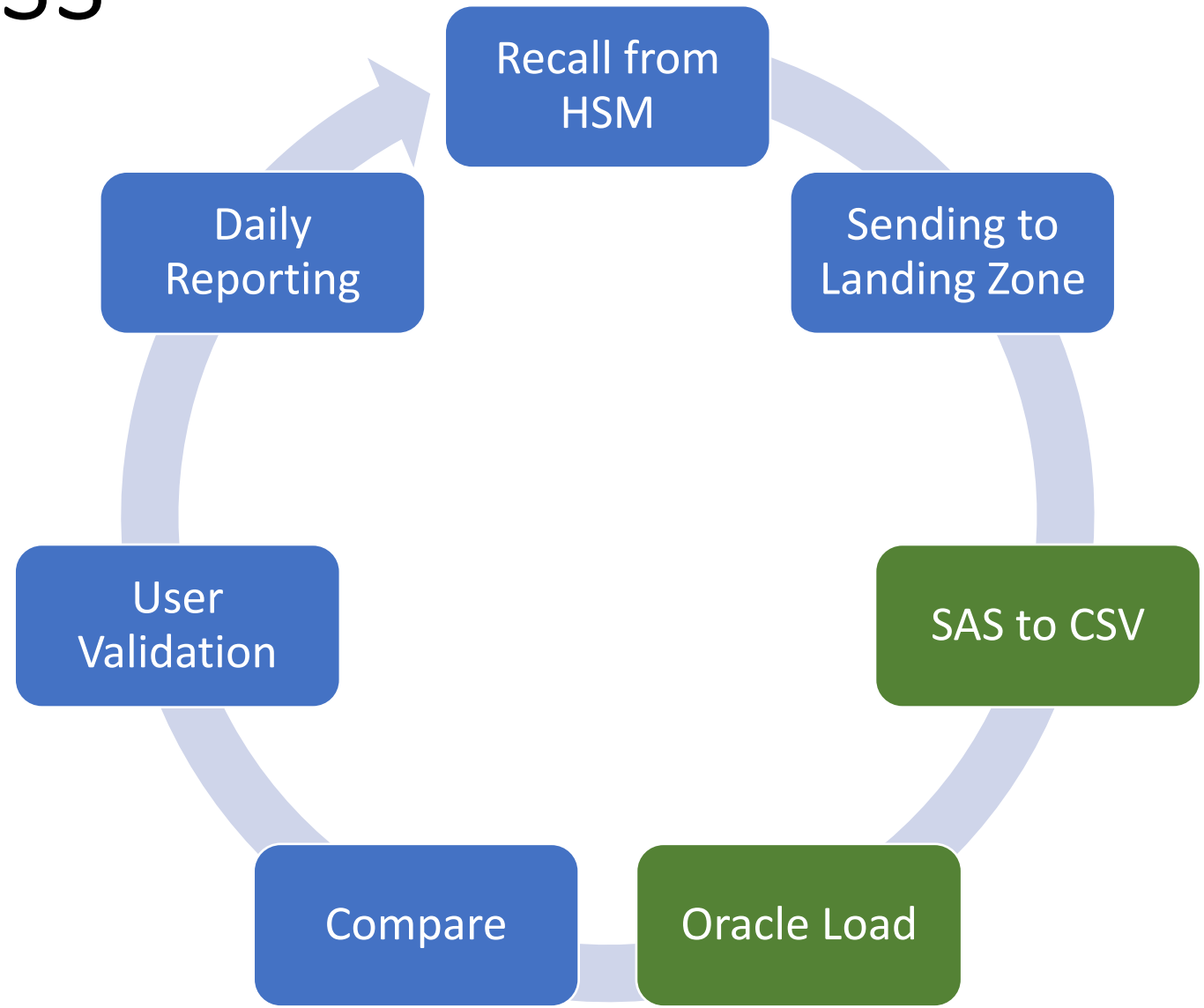
- Minimize user involvement

Introduction



Item	Qualifier	Value
recall	daily limit	1000
SAS Grid capacity	modeler	65000
SAS Grid capacity	other	8000
SAS Grid capacity	reporter	30000
SAS Grid capacity	risk	18000
recall	concurrency	5
scp	concurrency	3
scp	daily limit	1024
landing capacity		2048
small file		61
partial user data	validate	Y
partial user data	delete	Y
interval	delete	3
interval	scp resend	7
interval	recall	2
char var max length		4000
file name max length		100
max columns		879
Oracle loading alert min		30
Oracle loading alert email		harry.droogendyk@comp.com
compare_csv	concurrency	3

Process



No Concurrent Updates

- No SAS/Share
- Each process wrote CSV status files
- Create a dynamic view to read all CSV status files
- Sort by directory / filename, descending entry_dtm
  - dedup on directory / filename

```
libname usmle_o "&dir/output";

data usmle_o.v_file_status / view = usmle_o.v_file_status;

      length  entry_dtm          8
              task               $64
              dir_fn             $1024
              status             $64
              complete_dtm       8
              elapsed_sec        8
              csv_fn
              fn                 $128
              ;
```

No Concurrent Updates

- No SAS/Share
- Each process wrote CSV status files
- Create a dynamic view to read all CSV status files
- Sort by directory / filename, descending entry_dtm
  - dedup on directory / filename

```
format  entry_dtm
        complete_dtm            datetime19.
        elapsed_sec             comma16.
        ;

infile "&dir/output/csv/*.csv" firstobs=2
        dlm=',' filename=fn;

input @;

* firstobs only applies to the first file,
  because we have wildcard subsequent header
  rows will be read;

if _infile_ =: 'entry' then delete;
```

No Concurrent Updates

- No SAS/Share
- Each process wrote CSV status files
- Create a dynamic view to read all CSV status files
- Sort by directory / filename, descending entry_dtm
  - dedup on directory / filename

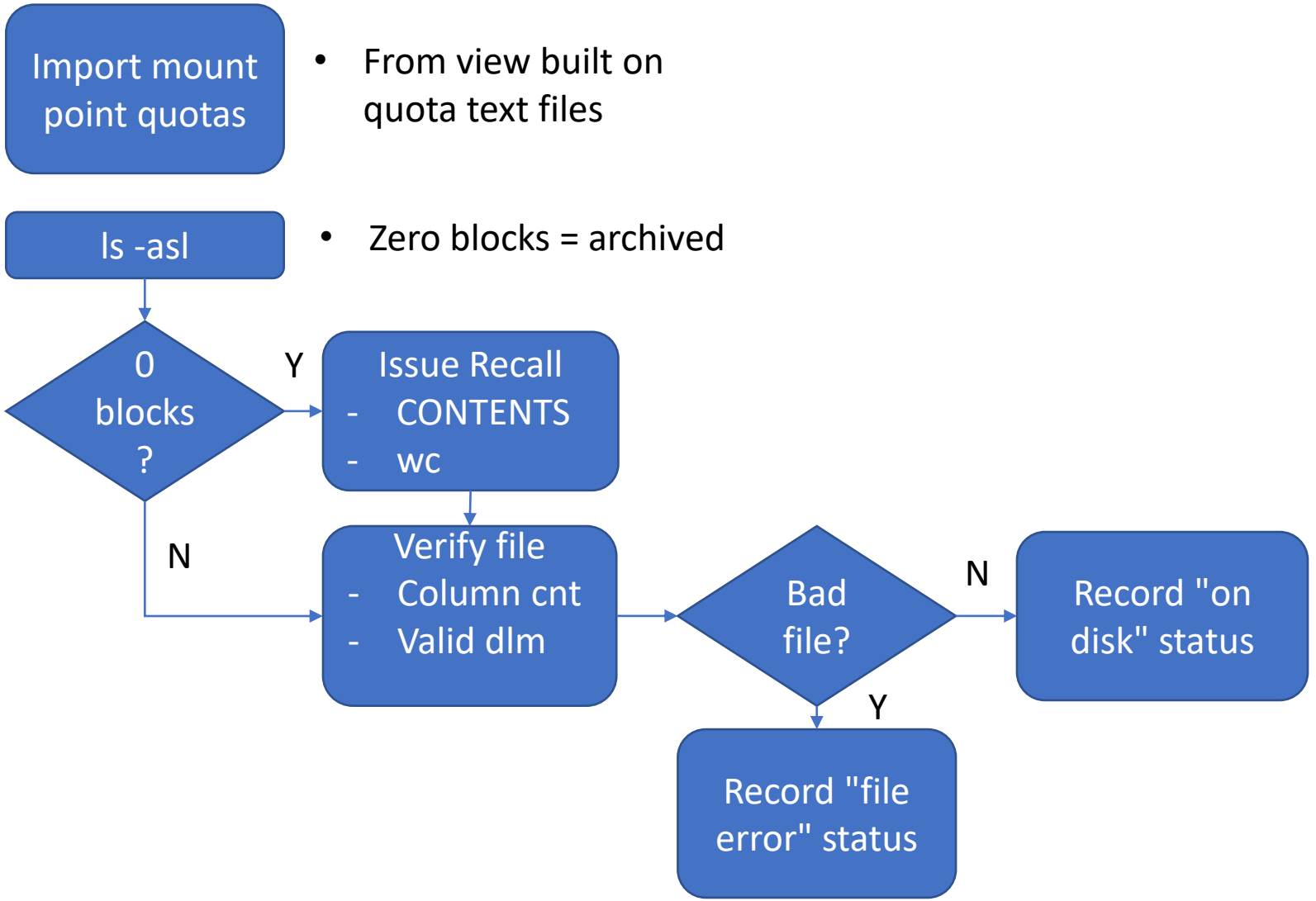
```
        input   entry_dtm
                task
                dir_fn
                status
                complete_dtm
                elapsed_sec
                ;
        csv_fn = scan(fn,-1,'/');
run;
```

No Concurrent Updates

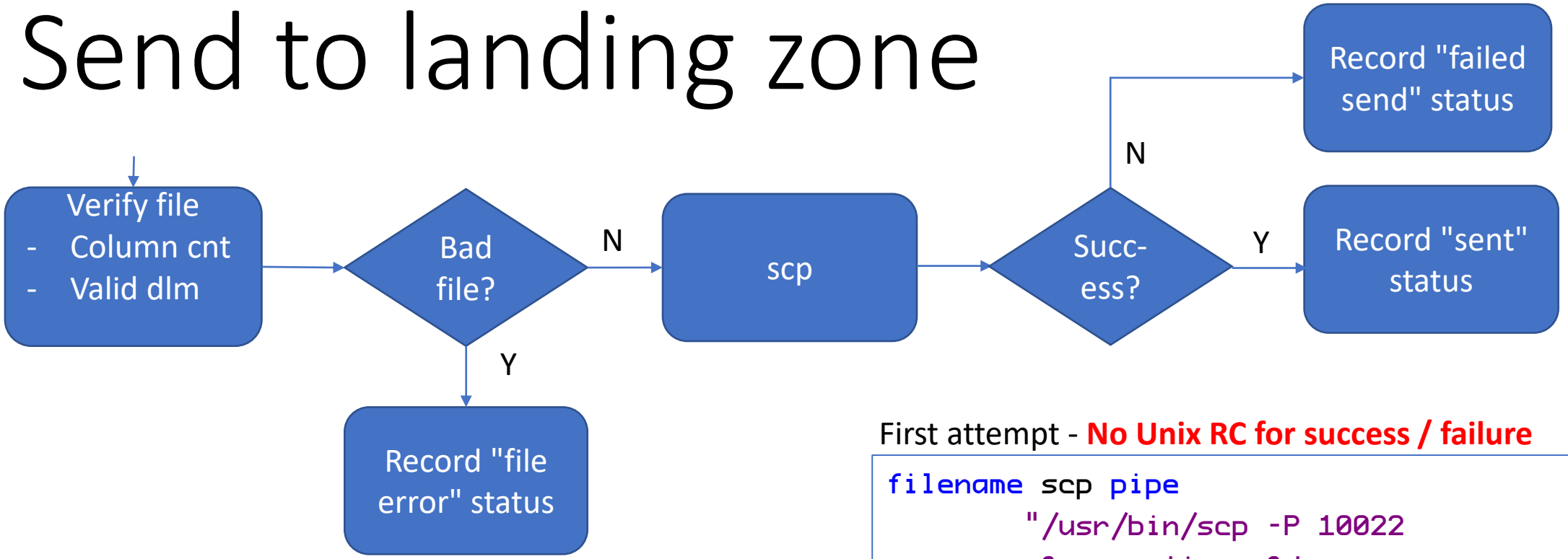
- No SAS/Share
- Each process wrote CSV status files
- Create a dynamic view to read all CSV status files
- Sort by directory / filename, descending entry_dtm
  - dedup on directory / filename

```
options nonotes;          * to prevent zillions of log messages
                            for raw file read;

proc sort data = usmle_o.v_file_status
             out = file_status;
          by dir_fn descending entry_dtm;
run;

options notes;

data file_status_last;
          set file_status;
           by dir_fn;

          if first.dir_fn;
run;
```

Recalling data



Send to landing zone



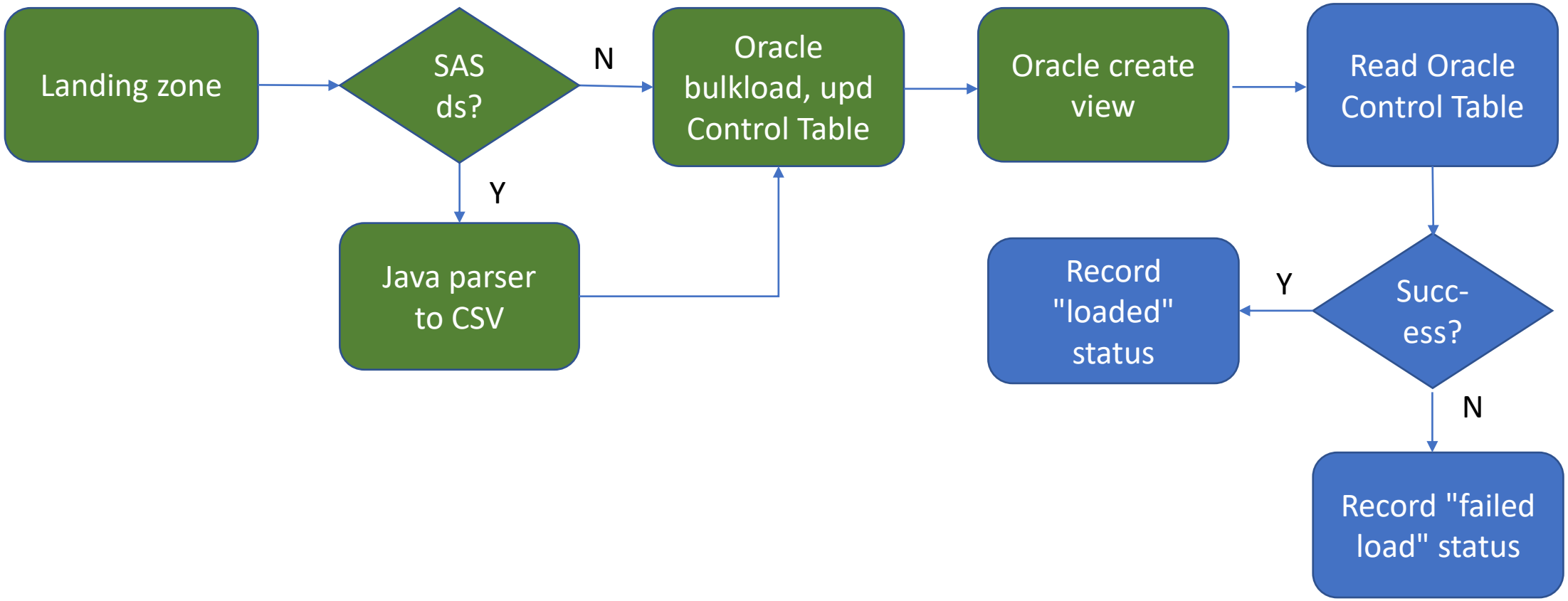
Unix RC returned

```
%let sfUnixCommand = /usr/bin/scp
        -P 10022 &scp_options "&ds"
        &scp/%trim(&fn_landing_zone) ;

%sysexec(&sfUnixCommand) ;
%if &SysRC = 0 %then …
```

First attempt - **No Unix RC for success / failure**

```
filename scp pipe
        "/usr/bin/scp -P 10022
         &scp_options &ds
         &scp/%trim(&fn_landing_zone)";

data scp_results;
        infile scp end = eof;
        do until (eof);
                input;
                if _infile_ =: '…'
```

Load Oracle table



Load Oracle table

VIEWTABLE: Work.Trans

	acct_no	tran_dt	tran_loc	tran_amt	tran_cat
1	4321432143214321	2023-04-12	Retailer	321.67	Clothes
2	4356564786967893	2023-05-22	Gas Station	56.88	Fuel
3	4519523454357455	2023-02-08	Restaurant	200.00	Dining
4	4499876429934334	2023-11-11	Restaurant	33.42	Takeout
5	4873838438348458	2023-01-25	Retailer	17.45	Hardware

```
create table schema.generic_sas (
       SAS_dataset_name varchar(100),
       field1 varchar(4000),
       field2 varchar(4000),
       field3 varchar(4000),
       field4 varchar(4000),
       field5 varchar(4000), …
```

VIEWTABLE: Work.Acct

	acct_no	cust_name	acct_bal	open_dt	credit_score	address
1	4321432143214321	ABC Enterprises	123.45	2017-12-12	679	123 Main St
2	4356564786967893	Stratia Consulting Inc	891.12	2013-03-22	701	3928 Governors Rd
3	4519523454357455	Ernie's Bait Shop	4,367.01	2012-02-08	639	512 River St
4	4499876429934334	Smith Roofing	2,534.65	2012-11-11	677	451 Lynden Rd
5	4873838438348458	Co-op Feeds	23,675.19	2024-01-25	714	5 Barnabas St

```
create view schema.v_orig_SAS_dataset_name (
       field1 as orig_SAS_col1,
       field2 as orig_SAS_col2,
       field3 as orig_SAS_col3, …
  from schema.generic_sas
  where SAS_dataset_name = 'orig_SAS_dataset_name'
```

Load Oracle table

Java parser used to convert SAS datasets to CSV

- Open source – you often get what you pay for
- Selected by non-SAS person

- Examined SAS column metadata
  - Applied SAS formats to values
    - date value of 2017-04-13 with YYMMn6. format
    - numeric value of 23.56823 with PERCENT 7.2 format
    - SAS datetime value in a column with DATE9. format
    - User-defined formats

Oracle bulkloader allowed two delimiters: | and ,

- SAS dataset contained Customer_Name ABC Enterprises|Toronto

SAS dataset character values with embedded '0a'x

Non-standard SAS column names, e.g. Customer Name, Total Chargeoff %

Load Oracle table

Java parser changes

- Fixed bug changing numeric values of E10 to E1 and E-10 to E-1
- Provided table of SAS date, time and datetime formats
  - date: YYMMDDd10.
  - time: TIME15.6
  - datetime: DATETIME26.6
  - ignore all other formats, ingest raw value

Dual delimiter issue

- Oracle change to only consider , - load all SAS datastets, comma-delimited files
- Oracle change to only consider | - load all pipe-delimited files

Have users remediate SAS character values with embedded '0a'x

Non-standard SAS column names, e.g. "Customer Name", "Total Chargeoff %"

- Change column names in Oracle views

Validation

Oracle control table

- Rows in = rows loaded equaled success
- Input data was not compared to the Oracle data
- View creation failures caused by non-standard names were not recorded

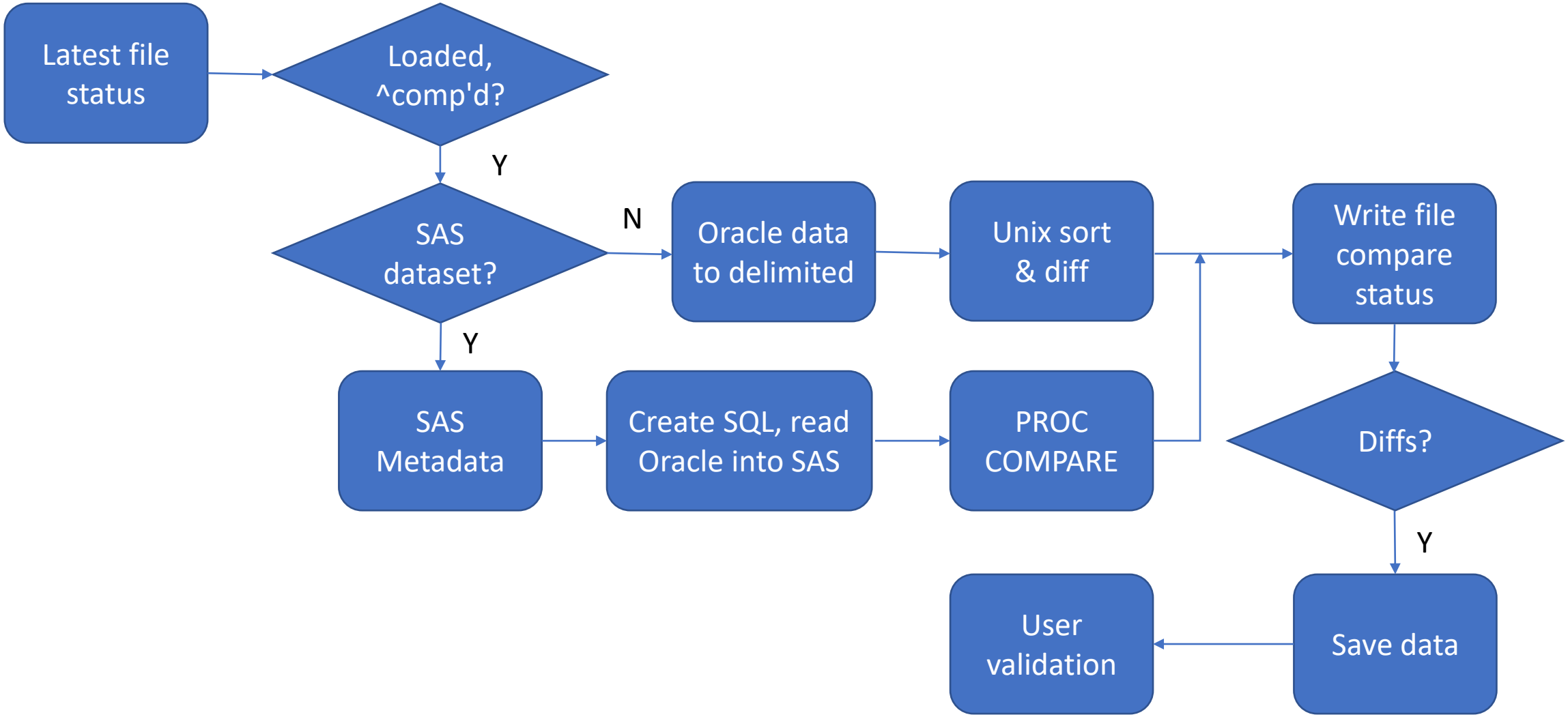
Oracle data was all varchar(4000)

- View only renamed fieldN to the original SAS dataset names
- Did not convert data back to original SAS data types & formats

How to validate accurate loads ?

- Read Oracle data, compare to:
  - SAS datasets
    - PROC COMPARE
    - Data types matter
    - SAS datasets were not indexed, unique primary keys unknown
  - Delimited files
    - IMPORT (guessingrows=max)
    - Or push Oracle to delimited file, use Unix diff ?

Validation

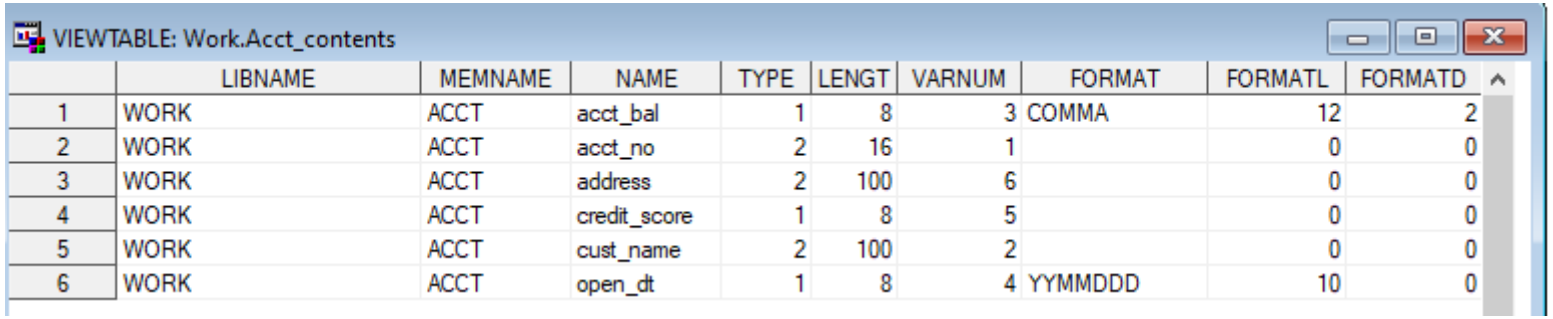


Validation

Oracle data was all varchar(4000)

- How do convert it back to original SAS data types & formats for PROC COMPARE?

```
PROC CONTENTS data = acct
              out = acct_contents noprint;
run;
```



VIEWTABLE: Work.Acct_contents

	LIBNAME	MEMNAME	NAME	TYPE	LENGT	VARNUM	FORMAT	FORMATL	FORMATD
1	WORK	ACCT	acct_bal	1	8	3	COMMA	12	2
2	WORK	ACCT	acct_no	2	16	1		0	0
3	WORK	ACCT	address	2	100	6		0	0
4	WORK	ACCT	credit_score	1	8	5		0	0
5	WORK	ACCT	cust_name	2	100	2		0	0
6	WORK	ACCT	open_dt	1	8	4	YYMMDDD	10	0

Validation

Construct Oracle SQL to transform all character data to reasonableness

- Sort SAS column metadata by Varnum
- Do most transformation in big Oracle DB, complete in SAS where necessary

- Build appropriate Oracle SELECT clauses based on:
  - SAS column type, length, format
  - Use various Oracle functions:
    - TO_DATE
    - TO_NUMBER
    - TO_TIMESTAMP
    - CAST – to appropriate length for character column

- Double quote Oracle column names for non-standard names
- Use RegEx to test Oracle values before transforming
  - E.g. deal with period for missing numeric values

Validation

Oracle character data values okay? Examine for invalid characters that choke Oracle functions

```
if type = 1 then do;
        case_stmt = catx(' ', 'case when regexp_like(', _oracle_col,
                                 ",'^\.|\+') then null else ");
        case_end        = 'end';
end; else do;
        case_stmt       = '';
        case_end        = '';
end;
```

Validation

Creating Oracle transformation SQL from SAS metadata

```
select;
  when ( type = 2 )
    line = catx(' ',_comma,'CAST(',_oracle_col,'as varchar2(',put(length,3.-l),'))' );
  when ( category = 'TIMESTAMP' )
    line = catx(' ',_comma, case_stmt, 'TO_TIMESTAMP(', _oracle_col, cats(",'",
                          informat_to_use,"')"), case_end        );
  when ( category = 'DATE' )
    line = catx(' ',_comma, case_stmt, 'TO_DATE(', _oracle_col, cats(",'",
                          informat_to_use,"')"), case_end );
  when ( category = 'TIME' )
    line = catx(' ',_comma, case_stmt, 'TO_TIME(', _oracle_col, cats(",'",
                          informat_to_use,"')"), case_end );
  otherwise
    line = catx(' ',_comma, case_stmt, 'TO_NUMBER(', _oracle_col,')', case_end  );
end;
```

Validation

Oracle transformation SQL – written to a Unix file

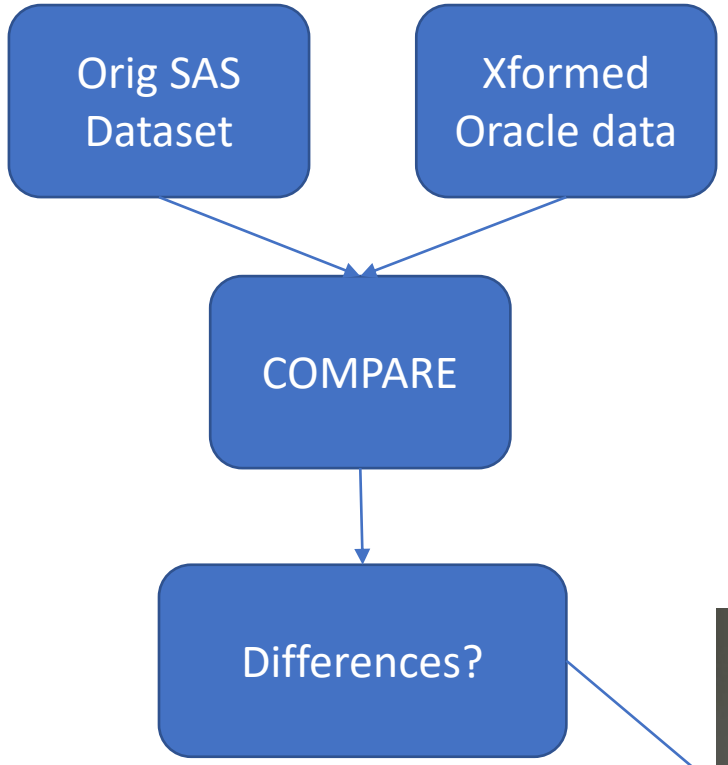
```
SELECT
case when regexp_like( PROCESS_PERIOD ,'^\.|\+') then null else TO_DATE(
PROCESS_PERIOD ,'YYYY-MM-DD') end as "PROCESS PERIOD"
, CAST( CURRENCY_CODE as varchar2( 9 )) as "CURRENCY CODE"
, case when regexp_like( ASSESSMENT_DATE ,'^\.|\+') then null else TO_DATE(
ASSESSMENT_DATE ,'YYYY-MM-DD') end as "ASSESSMENT DATE"
, case when regexp_like( RISK_STAGE ,'^\.|\+') then null else TO_NUMBER( RISK_STAGE
) end as "RISK_STAGE"
<snip>
, case when regexp_like( LOAN_OUT_AMT ,'^\.|\+') then null else TO_NUMBER(
LOAN_OUT_AMT ) end as "LOAN OUT AMT"
, CAST( CUSTOMER_NAME as varchar2( 36 )) as "CUSTOMER_NAME"
from VIEW_SCHEMA.v_ORIG_SAS_DS_1983631585_4959_9502
order by 2 nulls first
, 3 nulls first
, 4 nulls first
<snip>
```

Validation

SAS SQL, finishing the transformation task – written to a Unix file

```
SELECT
case when missing( 'Process Period'n ) then . else datepart ( 'Process Period'n )
end as 'Process Period'n format=MMDDYY10.0
, 'Segment Name'n format=$9.
, 'Entity Name'n format=$84.
, 'Currency Code'n format=$9.
, case when missing( 'Assessment Date'n ) then . else datepart ( 'Assessment Date'n
) end as 'Assessment Date'n format=MMDDYY10.0
, 'risk_stage'n
<snip>
, 'Loan Out Amt'n
, 'Adjusted Loan Outstanding Amt'n
, 'Unfunded Amt1'n
, 'Customer_Name'n format=$36.
from connection to oracle (
<snip>
```

Validation



- **No unique primary key**
  - Sort by all variables

VIEWTABLE: Work.Sas

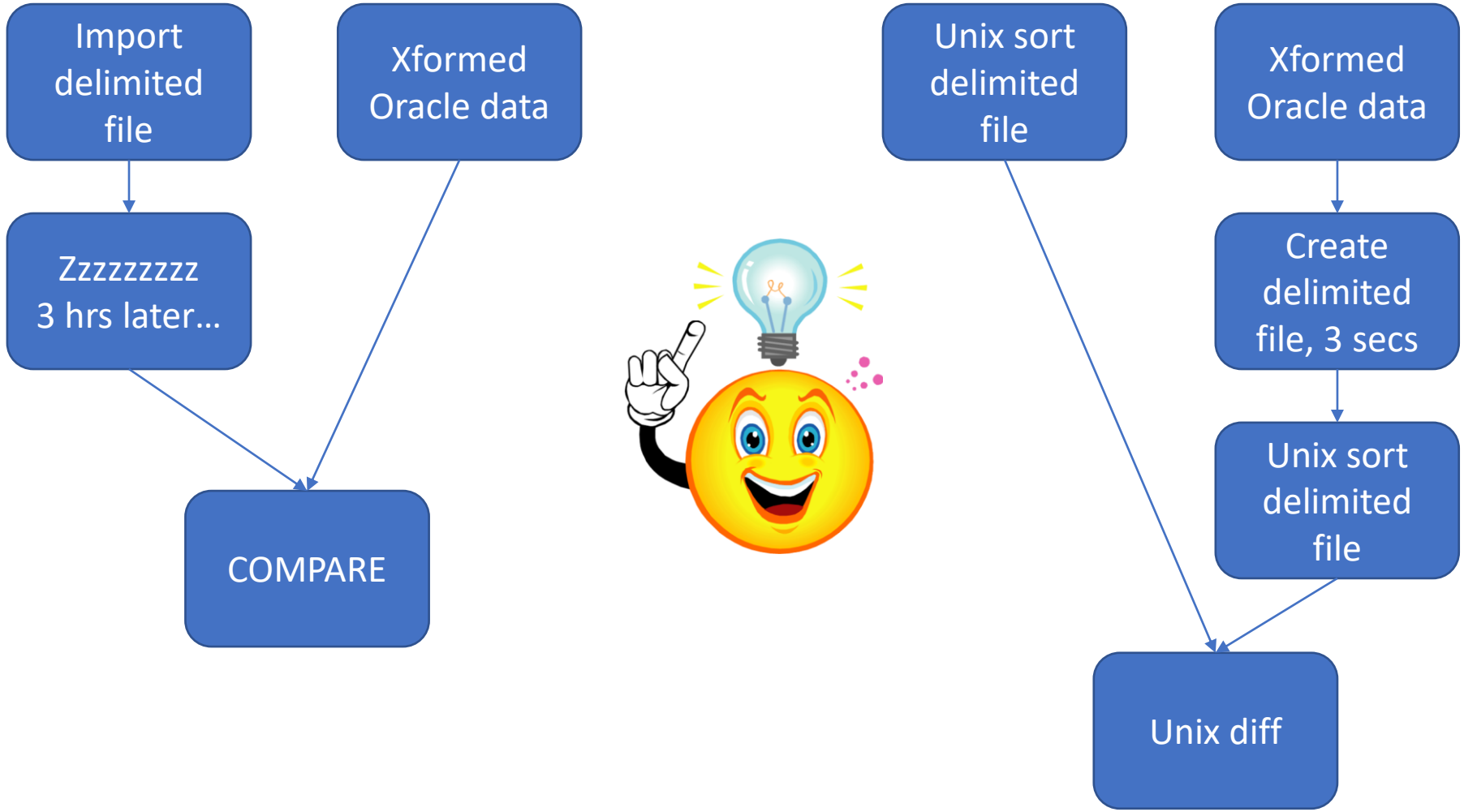
	fld1	fld2	fld3
1	23.56789	0.0023	87.3
2	23.56789	0.0023	66.22

VIEWTABLE: Work.Oracle

	fld1	fld2	fld3
1	23.56789	0.0023	66.22
2	23.56789	0.0023	87.3

- **Floating point weirdness**
  - best32.16
  - Looks identical, but it ain't
  - fld2 value in 2nd row had rogue floating point value in Nth decimal place
    - meant it was GT 1st row value

Validation



Parallel Processing

Constrained by capacity and daily limits

- latitude in terms of number of parallel processes
  - Recalling archive files
  - Sending files to the Landing Zone

How to do this?

SAS/Connect

- PC SAS running on the desktop
- signon and rsubmit to the remote server to do the heavy lifting

MP Connect

- spawn parallel processes on the same machine
  - Starts up a new SAS session
  - Log / output returned to the original SAS session
  - When new SAS session(s) finish, control returns to original SAS session

Parallel Processing

Calc allowable processing

- Daily limits and capacity – set by Settings Excel sheet
  - Recalling archive files
  - Sending files to the Landing Zone

Split among X parallel process using MP Connect

- Split the "to be processed" data X ways
- Spawn X processes
- Execute X processes

Parallel Processing

Split "to be processed" X ways

- Dorfman, aka "Hash Man", smartest SAS programmer on the planet
- Hash of hashes
- https://support.sas.com/resources/papers/proceedings/proceedings/sugi30/236-30.pdf
- Grab the code
  - Insert the link and a comment
    - "I don't understand this, you don't either, don't touch it, it works"
- Write X permanent datasets

SAS macro code looping X times:

signon
rsubmit

Each sessions writes results to X permanent datasets

Parallel Processing

```
proc sql noprint;
        select quote(trim(name))
          into :hash_columns separated by ','
          from sashelp.vcolumn
         where upcase(libname)       = 'WORK'
           and lowcase(memname) = 'files_to_scp_m_a_s'
         order by name
               ;
quit;

%put Hash columns are: &hash_columns;
```

Parallel Processing

```
data _null_;

                declare hash hoh  ( ordered: 'A' ); * hoh = hash of hashes;
                declare hiter hih ( 'hoh' );                        * hash iter ;

                hoh.defineKey     ( '_rem'                          );              * hoh is keyed by the _rem values, i.e. 1 - &scp_concurrency ;
                hoh.defineData    ( '_rem', 'hh'    );                              * data in hoh contains hh, the hash that actually contains the data ;
                hoh.defineDone    ();

                declare hash hh   ();

                do _n_ = 1 by 1 until ( eof ) ;

                                set files_to_scp_m_a_s end = eof;

                                /*  If we have not seen this _rem value before, define a hash for it  */

                                if hoh.find () ne 0 then do;
                                                hh = _new_ hash ( ordered: 'a' );
                                                hh.defineKey      ( '_rem','_n_' );
                                                hh.definedata     ( &hash_columns ) ;
                                                hh.defineDone     ();
                                                hoh.replace       ();
                                end;

                                /*  Jam the row's data into the _rem value's hash  */

                                hh.replace ( );
                end ;

                /*  Iterate over hih, dumping the hh hash, creating a separate dataset for each value of _rem, magic  */

                do rc = hih.next () by 0 while ( rc = 0 ) ;
                                hh.output (dataset: 'inter.scp'|| put (_rem, best.-L))  ;
                                rc = hih.next() ;
                end;

                stop;

run ;
```

Parallel Processing

```
options signonwait autosignon;

%do i = 1 %to &scp_concurrency;

        %put Parallel scp: Signing on scp&i, starting at %sysfunc(datetime(),datetime26.6);

        %let waitfor = &waitfor scp&i;

        signon scp&i sascmd="!sascmd -nosyntaxcheck -noterminal";

        %syslput unix_dir       = &unix_dir             / remote=scp&i;
        %syslput i              = &i                    / remote=scp&i;
        %syslput scp            = &scp                  / remote=scp&i;
        %syslput scp_options    = &scp_options  / remote=scp&i;

%end;
```

Parallel Processing

```
%do i = 1 %to &scp_concurrency;

        %put Parallel scp: Testing scp&i, pushing macro vars at %sysfunc(datetime(),datetime26.6)

        rsubmit scp&i wait=no;

                libname inter            "&unix_dir/output/intermediate";

                %put Parallel scp: Remote scp&i;
```

Lessons Learned

Don't trust anyone

- Sorry, not sorry

Pilots

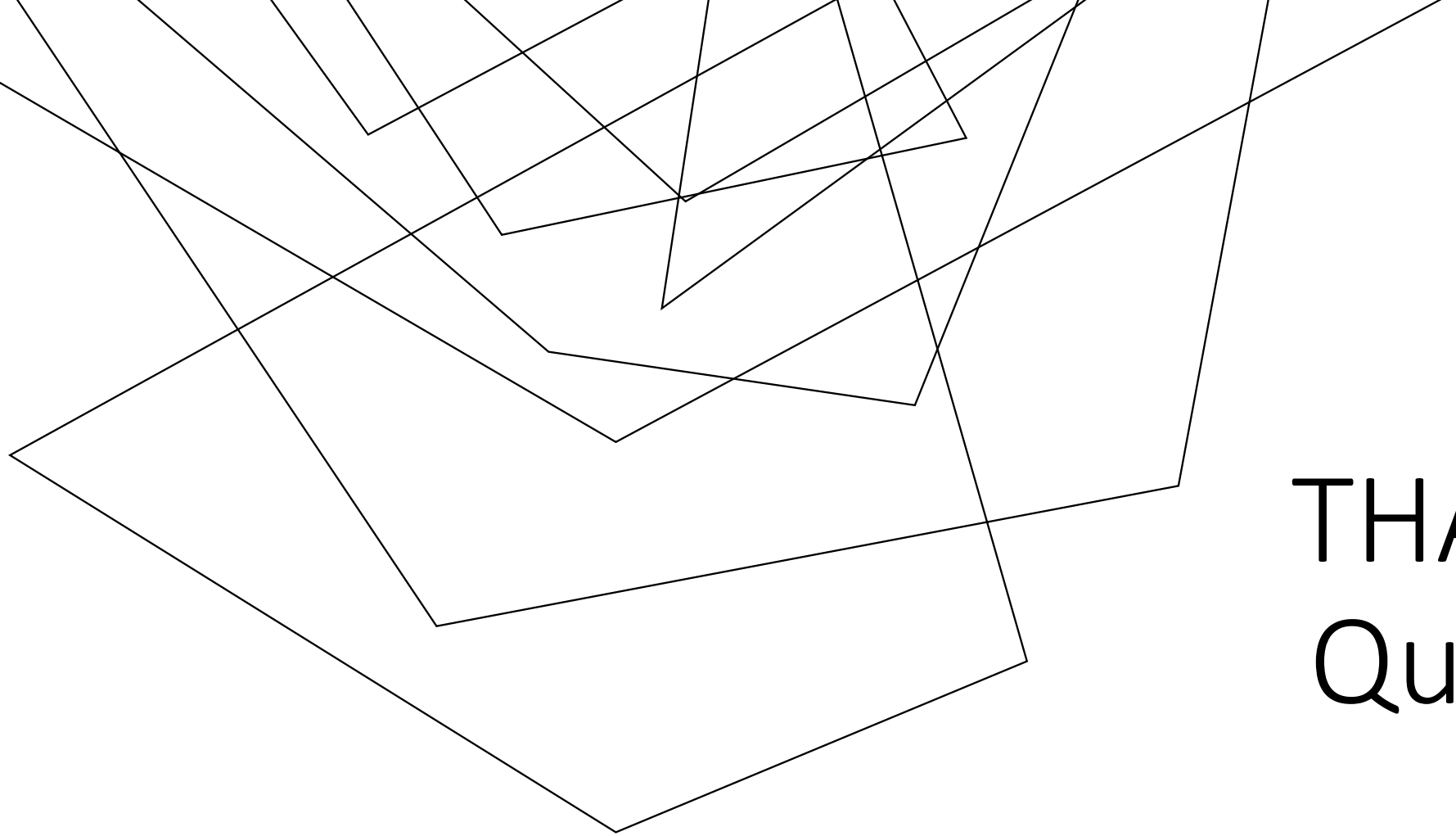
- Be imaginative
  - Every scenario
  - Every data type
  - Every data value (ranges)

Think outside the box

- e.g. SAS/Share vs CSV status files

Stand for right

- "close enough" only works in horseshoes and hand grenades
- are your users happy?



THANK YOU
Questions?

Harry Droogendyk

harry@stratia.ca

www.stratia.ca