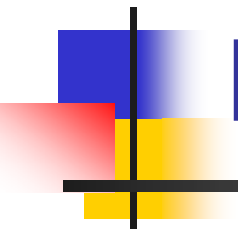
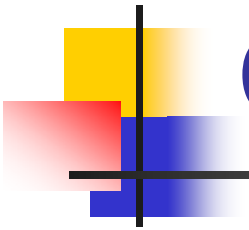
QC Your RDBMS Data Using Dictionary Tables

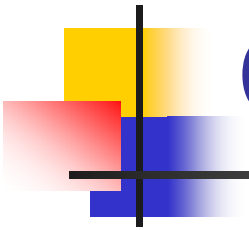


Harry Droogendyk
Stratia Consulting Inc.



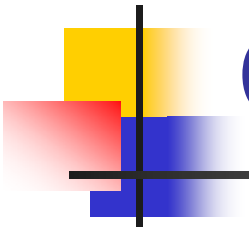
QC Data

- ETL process
- data summarization
- test data creation
- verifying data is a good thing!



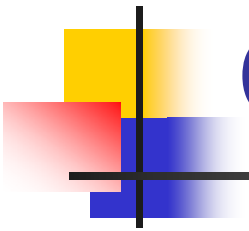
QC Data

- rudimentary data QC
  - continuous
    - numeric variables
      - eg. weight, balance
    - count, min, max, sum, mean, stddev
    - not all numeric data are *really* numbers
  - categorical
    - frequency distributions



QC Data

- what's involved?
  - separate data into three categories
    - continuous
    - categorical
    - junk
  - generate numeric analysis
  - generate frequency distributions



QC Data

- lazy programmers
  - don’t want to hard-code
  - don’t want to code
  - don’t want to think.... too much
  - do want to be productive
- can we let the data drive the process?

Data Driven Code

```
proc summary data = acct_data print sum;
  class state_cd ;

  var chequing_bal savings_bal ;
run;
```

- next month, visa_bal, mortgage_bal
- month after, loc_bal, invest_bal
- month after ….

Data Driven Code

- SAS has “dictionary” data
  - special libname – dictionary
    - only useful in PROC SQL
  - sashelp.v* views
    - available in PROC SQL and data step
- let the data drive the code

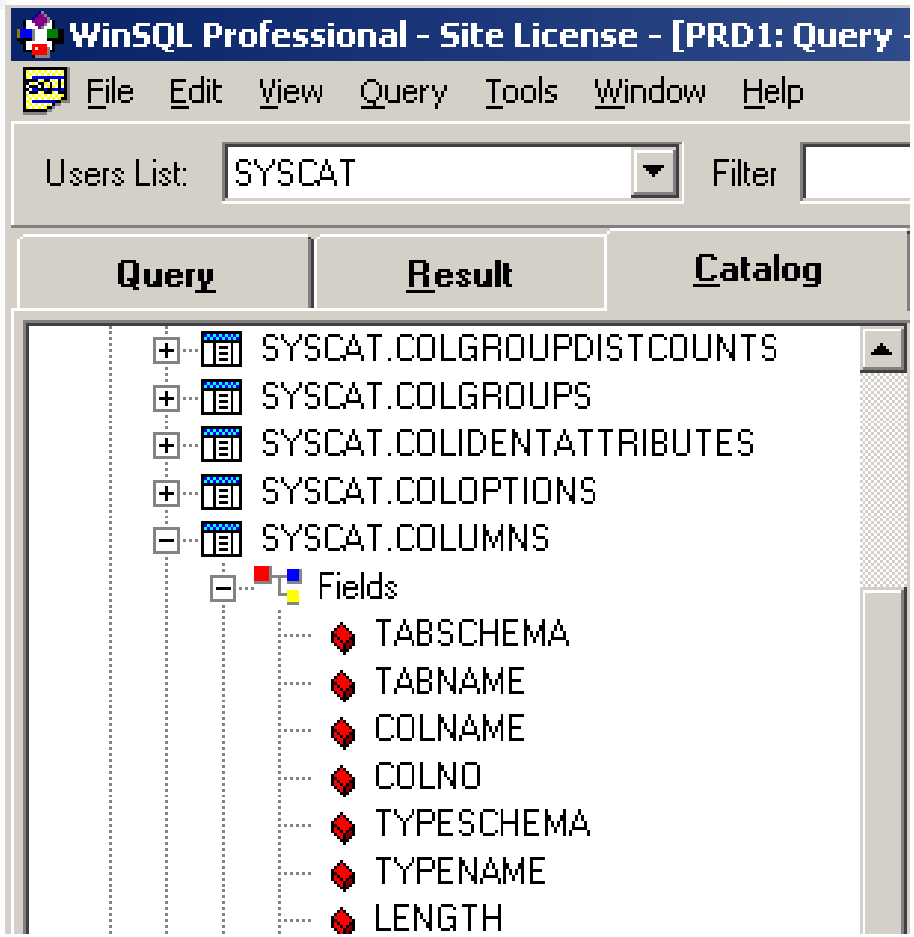
Data Driven Code

```
proc sql;
  select name
    into :bal_vars separated by ' '
    from sashelp.vcolumn
   where libname = 'WORK'
     and memname = 'ACCT_DATA'
     and scan(name,-1,'_') = 'bal' ;
quit;
```

Data Driven Code

```
proc summary data = acct_data print sum;
  class state_cd ;

  var &bal_vars ;
run;

SYMBOLGEN:  Macro variable BAL_VARS
  resolves to chequing_bal savings_bal
  visa_bal mortgage_bal
```

RDBMS Dictionary Data



DB2 Dictionary Data

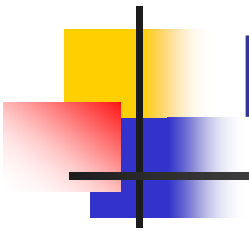
```
select tabschema, tabname, colname,
       typename
  from syscat.columns
 where tabschema = 'DROOGH2'
   and tabname   = 'QC_TEST'
order by typename, colname
    ;
```

DB2 Dictionary Data

TABSCHEMA	TABNAME	COLNAME	TYPENAME
DROOGH2	QC_TEST	ACCT_ID	BIGINT
DROOGH2	QC_TEST	CTD_CREDIT_AM	DECIMAL
DROOGH2	QC_TEST	CTD_DEBIT_AM	DECIMAL
DROOGH2	QC_TEST	DISPUT_AM	DECIMAL
DROOGH2	QC_TEST	CTD_CREDIT_CT	INTEGER
DROOGH2	QC_TEST	CTD_DEBIT_CT	INTEGER
DROOGH2	QC_TEST	ACCT_FAMILY_CD	SMALLINT
DROOGH2	QC_TEST	ACCT_SUBFAM_CD	SMALLINT
DROOGH2	QC_TEST	ACCT_TYPE_ID	SMALLINT
DROOGH2	QC_TEST	APPL_SUFFIX_NO	SMALLINT
DROOGH2	QC_TEST	CLIENT_PRODCT_CD	SMALLINT
DROOGH2	QC_TEST	TBAL_CD	SMALLINT
DROOGH2	QC_TEST	ACCT_TYPE_MN	VARCHAR
DROOGH2	QC_TEST	ACCT_TYPE_NA	VARCHAR

Teradata Dictionary Data

```
select tablename, columnname,
        columntype
  from dbc.columns
 where databasename = ‘SANDBOX'
   and tablename    = 'QC_TEST'
order by columntype, columnname
    ;
```



RDBMS Dictionary Data

- RDBMS metadata extraction
  - database specific syntax
  - argh...
- SAS rules !
  - RDBMS libnames
  - proc contents

RDBMS Dictionary Data

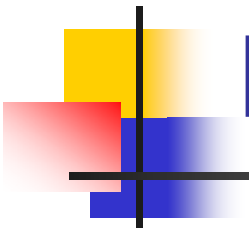
```
options sastrace=',,,d'
  sastraceloc=saslog nostsuffix;

libname _db2 db2 database=test
  schema=droogh2;

proc contents data = _db2.qc_test;
run;
```

RDBMS Dictionary Data

```
DB2: AUTOCOMMIT is NO for connection 0
516  options sastrace=',,,d'
  sastraceloc=saslog nostsuffix;
518  libname _db2 db2 database=test
  schema=droogh2;
NOTE: Libref _DB2 was successfully
  assigned as follows:
      Engine:          DB2
      Physical Name: test
```

RDBMS Dictionary Data

```
DB2: AUTOCOMMIT turned ON for connection
  id 0
DB2_1: Prepared:
SELECT * FROM droogh2.QC_TEST FOR READ
  ONLY
DB2: COMMIT performed on connection 0.
520  proc contents data = _db2.qc_test;
521  run;
NOTE: PROCEDURE CONTENTS used :
```



RDBMS Dictionary Data

- SQLNumResultCols
  - number of columns in table
- SQLDescribeCol
  - column name, type, length etc.
- SQLColAttribute
  - type specific column attributes

RDBMS Dictionary Data

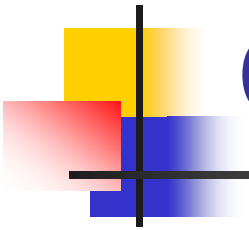
The CONTENTS Procedure

Data Set Name	_DB2.QC_TEST	Observations	.
Member Type	DATA	Variables	14
Engine	DB2	Indexes	0
Created	.	Observation Length	0
Last Modified	.	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO

RDBMS Dictionary Data

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format	Informat	Label
12	ACCT_FAMILY_CD	Num	8	11.	11.	ACCT_FAMILY_CD
1	ACCT_ID	Num	8	20.	20.	ACCT_ID
13	ACCT_SUBFAM_CD	Num	8	11.	11.	ACCT_SUBFAM_CD
10	ACCT_TYPE_ID	Num	8	11.	11.	ACCT_TYPE_ID
11	ACCT_TYPE_MN	Char	15	$15.	$15.	ACCT_TYPE_MN
14	ACCT_TYPE_NA	Char	23	$23.	$23.	ACCT_TYPE_NA
3	APPL_SUFFIX_NO	Num	8	11.	11.	APPL_SUFFIX_NO
2	CLIENT_PRODCT_CD	Num	8	11.	11.	CLIENT_PRODCT_CD
9	CTD_CREDIT_AM	Num	8	15.2	15.2	CTD_CREDIT_AM
8	CTD_CREDIT_CT	Num	8	11.	11.	CTD_CREDIT_CT
7	CTD_DEBIT_AM	Num	8	15.2	15.2	CTD_DEBIT_AM
6	CTD_DEBIT_CT	Num	8	11.	11.	CTD_DEBIT_CT
5	DISPUT_AM	Num	8	15.2	15.2	DISPUT_AM
4	TBAL_CD	Num	8	11.	11.	TBAL_CD



QC Data

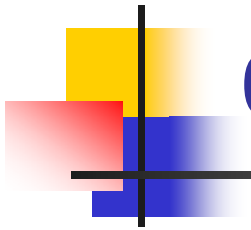
- define input table
- identify variables requiring numerical analysis
- identify variables requiring frequency distributions
- do the deed!

qc_db_data.sas

- macro is self-documenting

  `%qc_db_data(?)`

- generates documentation in log
  - purpose
  - parms and their values
  - output



qc_db_data.sas

```
================================================================================
%qc_db_data( help, lib=, table=, drop_columns=, keep_columns=, by_vars=, where=, freq_limit = 100)

QC / analyze the RDBMS table specified, creating frequency distributions or min, max, mean,
stddev and sum depending on the column type and granularity of the data in the table.

Parms:
   help             any value in the sole positional parameter provides this help text
   lib              SAS libref via RDBMS engine for schema that contains &table
   table            RDBMS table to be analyzed, MUST be sorted by &by_vars ( if specified )
   drop_columns     comma-delimited, single-quoted column names to be IGNORED in analysis,
                    - must use %str('col1','col2') when specifying multiple column names
                    - always specify 'acct_id', 'cust_id' type fields in this parm
   keep_columns     comma-delimited, single-quoted column names to be considered for analysis,
                    - must use %str('col1','col2') when specifying multiple column names
   by_vars          comma-delimited, single-quoted column names for BY groups
                    - must use %str('col1','col2') when specifying multiple column names
   where            WHERE clause to apply to input &schema..&table to focus analysis
   freq_limit       upper limit of number of distinct values used to decide which vars generate
                    frequence distributions, default is 100 distinct values
                    - all columns with <= &freq_limit distinct values will generate freq dist
                    - num columns with >  &freq_limit distinct values will generate num analysis
```

qc_db_data.sas

Macro logic outlined below:

1. Derive table columns using PROC CONTENTS data=&lib..&table, incorporate &drop_column and &keep_column criteria
2. count distinct values for all selected fields
3. numeric fields where count of distinct values > &freq_limit, create min/max/stddev/sum stats
4. run frequency distribution on any fields that have <= &freq_limit distinct values
5. if &by_vars are specified, all stats will be created with the BY groups specified
6. create datasets of final results in remwork._qc_continuous_data and remwork._qc_categorical_data

Sample Invocation:

```
libname rdbms <RDBMS engine> <RDBMS connection particulars>;

%qc_db_data(lib             = rdbms,
            table           = qc_test,
            drop_columns    = %str('acct_id'),
            by_vars         = %str('acct_type_na'),
            where           = %str(acct_type_na like 'SAV%'),
            freq_limit      = 50
            )
```

qc_db_data.sas

```
%qc_db_data(
lib               = _db2,
table             = qc_test,
drop_columns      = %str('acct_id'),
by_vars           = %str('acct_type_na'),
where             =
     %str(acct_type_na like '%Visa%'),
 freq_limit       = 100
           );
```

qc_db_data.sas

```
proc contents data = &lib..&table
                  out = _qc_db_columns_all
             ( keep =   name type formatl
             rename = ( name = colname )
             ) noprint;
run;
```

qc_db_data.sas

```
data _qc_db_columns;
      set _qc_db_columns_all;
      %if &drop_columns > %then %do;
            if colname not in ( %upcase(&drop_columns) );
      %end;

      %if &keep_columns > %then %do;
            if colname in ( %upcase(&keep_columns) );
      %end;

      if type = 1 then coltype = 'N'; else coltype = 'C';

      drop type;
  run;
```

qc_db_data.sas

```
/*
 Create the count(distinct x) as x phrases.  The
 results of these will determine whether we do
 freq distribution on the variables
*/

select

'count (distinct(' || trim(colname) ||
      ')) as ' || trim(column_name)

 into :_qc_count_distinct separated by ','

 from _db_columns
```

qc_db_data.sas

```
/*
   Count distinct values of each variable,
   these counts used to decide if
   min/max/etc.. or freqs to be done
*/

   create table _qc_count_distinct as
       select &_qc_count_distinct
         from &lib..&table
       %if &where ne %then %do;
        where &where
       %end;
   ;
```

qc_db_data.sas

VIEWTABLE: Remwork._qc_count_distinct_x

	colname	cnt
1	ACCT_FAMILY_CD	1
2	ACCT_SUBFAM_CD	3
3	ACCT_TYPE_ID	10
4	ACCT_TYPE_MN	10
5	APPL_SUFFIX_NO	1
6	CLIENT_PRODCT_CD	11
7	CTD_CREDIT_AM	1979
8	CTD_CREDIT_CT	14
9	CTD_DEBIT_AM	19950
10	CTD_DEBIT_CT	65
11	DISPUT_AM	1
12	TBAL_CD	9

qc_db_data.sas

```
/* Numeric columns will be run through proc summary */

   select d.colname
     into :numeric_cols separated by ' '

     from _db_columns                    d,
          _qc_count_distinct_xpose       c

    where d.colname  = c.colname
      and d.coltype  = 'N'
      and c.cnt      > &freq_limit
   ;

   %let numeric_fld_cnt    = &sqlobs;
```

qc_db_data.sas

```
/*
   Any column with < &freq_limit distinct values is freqqed.
   This means that some character columns will have no
   analysis performed on them, eg. name fields.
*/
   select d.colname, d.colname
     into :char_col1 - :char_col&sysmaxlong ,
          :char_cols   separated by ' '

     from _db_columns                  d,
          _qc_count_distinct_xpose     c

    where d.colname    = c.colname
      and c.cnt        <= &freq_limit ;

   %let char_fld_cnt   = &sqlobs;
```

qc_db_data.sas

```
proc summary data = _&lib..&table
   ( keep = &numeric_cols &by_vars_stmt ) nway missing ;

   var &numeric_cols;

 * RDBMS sort order for mixed-case character columns differs;
   %if &by_vars_stmt ne %then %do;
     by &by_vars_stmt notsorted;
   %end;

   output out = metrics_num_min    ( rename=(_freq=count)) min= ;
   output out = metrics_num_max    ( rename=(_freq=count)) max= ;
   output out = metrics_num_mean   ( rename=(_freq=count)) mean= ;
   output out = metrics_num_stddev ( rename=(_freq=count)) stddev=;
   output out = metrics_num_sum    ( rename=(_freq=count)) sum= ;
run;
```

qc_db_data.sas

```
proc freq data = &lib..&table
   ( keep = &char_cols &by_vars_stmt );

 * DB2 sort order for mixed-case character columns differs;
   %if &by_vars_stmt ne %then %do;
       by &by_vars_stmt notsorted;
   %end;

   %do i = 1 %to &char_fld_cnt;
       tables &&char_col&i / missing
         out = &&char_col&i
         ( rename = ( &&char_col&i = value )) ;
   %end;
run;
```

Results

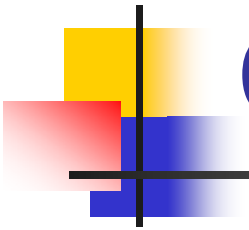
VIEWTABLE: Remwork._qc_categorical_data

	colname	value	ACCT_TYPE_NA	COUNT	PERCENT
56	CLIENT_PRODCT_CD	10223	Mellow Yellow Visa Card	6246	22.839799612
57	CLIENT_PRODCT_CD	10224	Mellow Yellow Visa Card	21101	77.160200388
58	CLIENT_PRODCT_CD	12271	Platinum Visa Card	255	100
59	CLIENT_PRODCT_CD	10226	Student Visa Card	21557	100
60	CLIENT_PRODCT_CD	10225	US Visa Card	1624	100
61	CLIENT_PRODCT_CD	10227	Visa Check Card run	1235	100
62	CTD_CREDIT_CT	0	Dividend Visa Card	17267	97.719298246
63	CTD_CREDIT_CT	1	Dividend Visa Card	308	1.7430673458
64	CTD_CREDIT_CT	2	Dividend Visa Card	61	0.3452178834
65	CTD_CREDIT_CT	3	Dividend Visa Card	15	0.0848896435
66	CTD_CREDIT_CT	4	Dividend Visa Card	11	0.0622524052
67	CTD_CREDIT_CT	5	Dividend Visa Card	3	0.0169779287

Results

VIEWTABLE: Remwork._qc_continuous_data

	ACCT_TYPE_NA	colname	min	max	mean	stddev	sum
1	Dividend Visa Card	CTD_CREDIT_AM	0.00	2513.36	2.86	42.89	50562.80
2	Dividend Visa Card	CTD_DEBIT_AM	0.00	10500.00	84.04	372.10	1485008.04
3	Dividend Visa Card	count	17670.00	17670.00	17670.00	17670.00	17670.00
4	Gold Plus Visa Card	CTD_CREDIT_AM	0.00	11992.72	11.25	190.59	68000.08
5	Gold Plus Visa Card	CTD_DEBIT_AM	0.00	84070.80	286.87	1605.78	1733842.93
6	Gold Plus Visa Card	count	6044.00	6044.00	6044.00	6044.00	6044.00
7	Gold Premium Visa Card	CTD_CREDIT_AM	0.00	4569.57	5.02	78.28	47838.08
8	Gold Premium Visa Card	CTD_DEBIT_AM	0.00	19400.00	148.88	613.30	1417760.22
9	Gold Premium Visa Card	count	9523.00	9523.00	9523.00	9523.00	9523.00



Conclusion

- leveraging metadata allows data driven code
  - SAS/Access LIBNAME engine
  - PROC CONTENTS
- data driven code = no maintenance
- no maintenance = happy programmer

Contact

Harry Droogendyk, SAS Consultant

conf@stratia.ca

Phone: 905-512-3827

Web: www.stratia.ca/papers