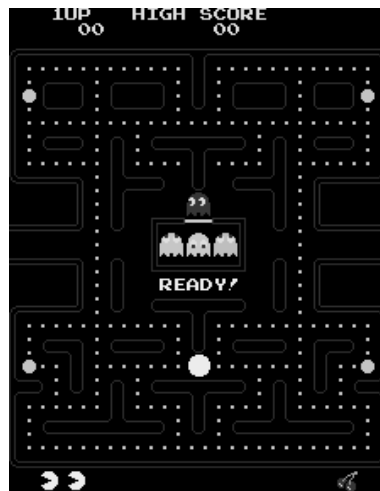


Avoiding Common Traps When Accessing RDBMS Data

Mike Rhoads, Westat

Boston Area SAS Users Group November 9, 2010

The Idea ...

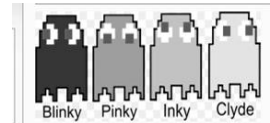


How You Can Play ...

- So, play the role of Pac-Man (or Ms. Pac-Man) ...



- Avoid the ghosts ...



- And reach the top level of the game



Agenda for this talk

- Alert you to some of the more common issues when accessing RDBMS data from SAS® (powerful, easy, transparent, but ...)
- Provide background, symptoms, diagnoses, and treatment for each
- Start with quick intro to SAS/Access®, then move on to the traps

Working with relational databases

- *Connect* to the database
- *Perform tasks* using Structured Query Language (SQL) – SELECT, INSERT, UPDATE, ...
 - SQL is “standard”, but each RDBMS has some variations
 - So does SAS
- *Disconnect*
- No “magic” with SAS – to the RDBMS, looks like any other client software

RDBMS security

- Managed by the database software – SAS can't get around it
- Access to an individual database
- Access to specific database *objects* (tables, views, etc.)
- Type of access
- Often controlled through *roles*
- Can be extremely complex, or extremely simple

Methods in SAS for RDBMS access

- Pass-Through Facility (manual transmission)
 - Statements within PROC SQL
- LIBNAME statement (automatic transmission)
- ACCESS and DBLOAD procs (for backward compatibility only, no longer recommended)

Pass-Through Facility -- CONNECTing

```
PROC SQL;  
Connect to oledb as myTest (  
  provider=sqlOleDb  
  dataSource=YourServerName  
  properties=(  
    "Integrated Security"=SSPI  
    "Initial Catalog"=YourDBName  
  )  
);  
/* More PROC SQL statements ... */
```

CONNECT statement -- interface

```
Connect to oledb as myTest (  
  provider=sqlOleDb  
  dataSource=YourServerName  
  properties=(  
    "Integrated Security"=SSPI  
    "Initial Catalog"=YourDBName  
  )  
);
```

Identifies SAS/ACCESS
interface being used (here
OLE/DB)

CONNECT statement – connection alias

```
Connect to oledb as myTest (  
  provider=sqlOleDb  
  dataSource=YourServerName  
  properties=(  
    "Integrated Security"=SSPI  
    "Initial Catalog"=YourDBName  
  )  
);
```

Identifies optional *alias* for
the connection

CONNECT statement – interface-specific

```
Connect to oledb as myTest (  
  provider=sqlOleDb  
  dataSource=YourServerName  
  properties=(  
    "Integrated Security"=SSPI  
    "Initial Catalog"=YourDBName  
  )  
);
```

All of this will vary, depending on the SAS/ACCESS interface being used (DB/2, Oracle, OLE/DB, ...)

OLE/DB connection -- provider

```
Connect to oledb as myTest (  
  provider=sqlOleDb  
  dataSource=YourServerName  
  properties=(  
    "Integrated Security"=SSPI  
    "Initial Catalog"=YourDBName  
  )  
);
```

Identifies the OLE/DB provider, such as SQL Server

OLE/DB connection – data source (server)

```
Connect to oledb as myTest (  
  provider=sqlOleDb  
  dataSource=YourServerName  
  properties=(  
    "Integrated Security"=SSPI  
    "Initial Catalog"=YourDBName  
  )  
);
```

Identifies the name of your
data source (server)

Items for this specific provider (SQL Server)

```
Connect to oledb as myTest (  
  provider=sqlOleDb  
  dataSource=YourServerName  
  properties=(  
    "Integrated Security"=SSPI  
    "Initial Catalog"=YourDBName  
  )  
);
```

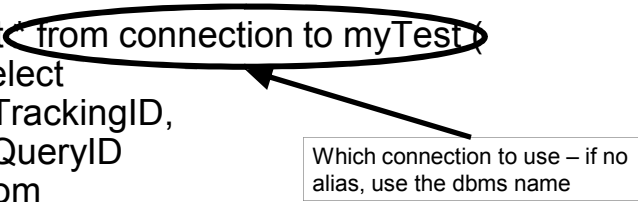
Specific information for this
OLE/DB provider (security,
database name, etc.)

Pass-Through Facility – retrieving data

```
PROC SQL;  
connect to ... ;  
select * from connection to myTest (  
    select  
        TrackingID,  
        QueryID  
    from  
        dbo.MyDatabaseTableName  
    order by  
        TrackingID ASC,  
        QueryID ASC  
);  
...
```

CONNECTION TO clause

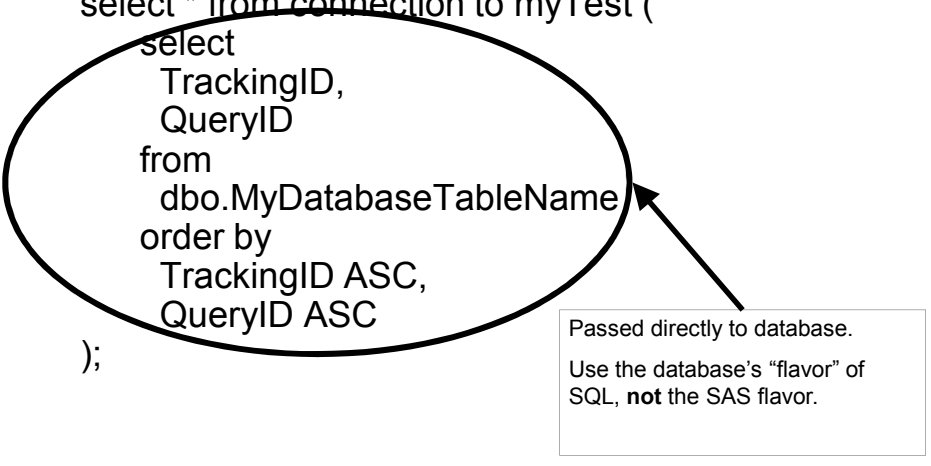
```
select * from connection to myTest (  
    select  
        TrackingID,  
        QueryID  
    from  
        dbo.MyDatabaseTableName  
    order by  
        TrackingID ASC,  
        QueryID ASC  
);
```



Which connection to use – if no alias, use the dbms name

Nested SELECT

```
select * from connection to myTest (  
  select  
    TrackingID,  
    QueryID  
  from  
    dbo.MyDatabaseTableName  
  order by  
    TrackingID ASC,  
    QueryID ASC  
);
```



Passed directly to database.
Use the database's "flavor" of
SQL, **not** the SAS flavor.

Pass-Through Facility – performing other DB tasks

```
PROC SQL;  
connect to ... ;  
execute  
  (insert into dbo.MyDatabaseTableName  
    values (24582,12) )  
  by myTest;  
...
```

Pass-Through Facility – identify connection

```
execute  
  (insert into dbo.MyDatabaseTableName  
   values (24582,12) )  
  by myTest;
```

Which connection to use – if no
alias, use the dbms name

Pass-Through Facility – RDBMS commands

```
execute  
  (insert into dbo.MyDatabaseTableName  
   values (24582,12) )  
  by myTest;
```

Passed directly to database.
Use the database's "flavor" of
SQL, **not** the SAS flavor.

Pass-Through Facility -- DISCONNECTing

```
PROC SQL;  
connect ... ;  
select ... ;  
disconnect from myTest;
```

2nd method for RDBMS access

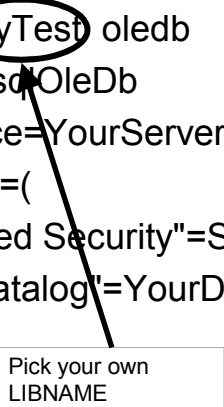
1. Pass-Through Facility
2. LIBNAME statement

LIBNAME statement

```
libname myTest oledb  
  provider=sqlOleDb  
  dataSource=YourServerName  
  properties=(  
    "Integrated Security"=SSPI  
    "Initial Catalog"=YourDBName  
  )  
;
```

LIBNAME statement

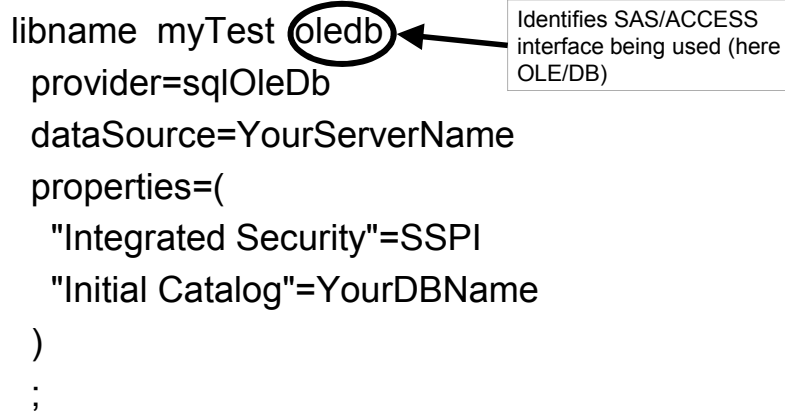
```
libname myTest oledb  
  provider=sqlOleDb  
  dataSource=YourServerName  
  properties=(  
    "Integrated Security"=SSPI  
    "Initial Catalog"=YourDBName  
  )  
;
```



Pick your own
LIBNAME

LIBNAME statement

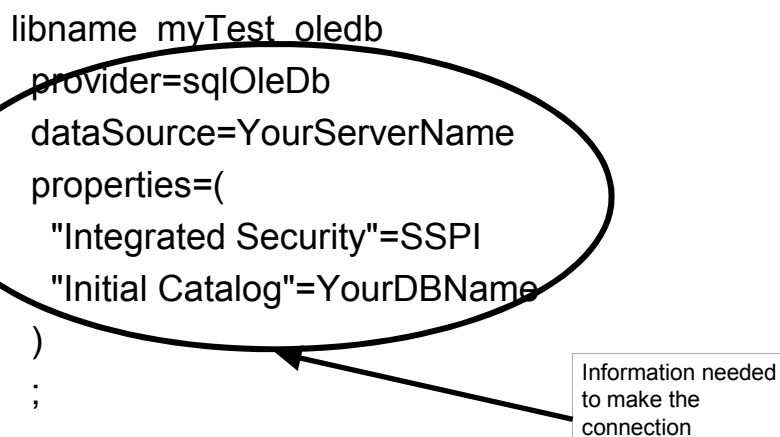
```
libname myTest oledb  
  provider=sqlOleDb  
  dataSource=YourServerName  
  properties=(  
    "Integrated Security"=SSPI  
    "Initial Catalog"=YourDBName  
  )  
;
```



Identifies SAS/ACCESS interface being used (here OLE/DB)

LIBNAME statement – connection information

```
libname myTest oledb  
  provider=sqlOleDb  
  dataSource=YourServerName  
  properties=(  
    "Integrated Security"=SSPI  
    "Initial Catalog"=YourDBName  
  )  
;
```



Information needed to make the connection

Now just use like any other libref

```
proc print data=myTest.MyDatabaseTableName;  
var TrackingID QueryID;  
run;
```

The LIBNAME you
just set up

The name of the object in the
database (table or view)

Comparing the two access methods

Pass-Through Facility

Everything happens within
PROC SQL

RDBMS “flavor” of SQL

Performance???

LIBNAME statement

Easy to access RDBMS
tables from DATA step or any
procedure

SAS “flavor” of SQL (when
used at all)

Performance???

A valuable debugging tool

- Make SAS reveal the exact instructions that it's sending to the database
- You will get detailed information in your SAS log whenever SAS communicates with the RDBMS

```
options sastrace=',,,d' sastraceloc=saslog nostsuffix;  
OLEDB_40: Executed:  
SELECT * FROM "dbo"."class"
```

Now – on to the traps!



Trap #1 – Where are my tables????

- Background – apparently successful LIBNAME connection, trying to look at your data
- Symptoms:
 - You run PROC CONTENTS and see
WARNING: No matching members in directory.
(or, don't see all of the tables you expect)
 - Or, you try to read a particular table and see
ERROR: File MYDB.MyTable.DATA does not exist.

Trap #1 -- Diagnosis

- Probably need to specify a schema for the RDBMS
- A what???

Trap #1 -- Treatment

- Need to specify RDBMS schema

```
libname mydb oledb  
  provider=sqloledb  
  datasource=myservername  
  properties=( "initial catalog"=mydbname )  
  prompt=yes  
  schema=myschemaname;
```

- Or specify alternate schema name as dataset option
 proc print data=mydb.SpecialTable (**SCHEMA=schema2**);

Trap #1 – Alternate diagnosis and treatment

- Your RDBMS table or view name may not follow SAS rules
- SAS will translate invalid characters
- But ... bigger problem if name is > 32 characters
- Pass-Through Facility may be your only solution

Trap #2 – But I can do that with SAS data sets!

- Background – trying to replace a data set "on the fly"

```
data mydb2.class;  
set mydb2.class;  
wh_ratio = weight / height;  
run;
```

- Symptom – message similar to the following:
 - ERROR: The OLEDB table class has been opened for OUTPUT. This table already exists, or there is a name conflict with an existing object. This table will not be replaced. **This engine does not support the REPLACE option.**

Trap #2 -- Diagnosis

- You can't just replace RDBMS tables on the fly
- Good reasons for that:
 - More structured environment ("long-term commitment")
 - Don't want to disrupt other users

Trap #2 – Treatment options

- Update in-place: modify / add / drop records using PROC SQL (UPDATE, INSERT, DELETE) or DATA step (MODIFY, REMOVE, REPLACE, OUTPUT)
- You can use ALTER TABLE to first modify the structure if necessary (and you have permission)
- Or, explicitly drop and recreate the table (again, if you have permission)

Trap #3 – I just sorted that (blasted) file!

- Background – you create a sorted data set from an RDBMS table, then go to use it

```
proc sort data=dbmslib.DBTable out=SortedSASDataSet;  
  by CharVar;  
run;  
data _null_;  
  set SortedSASDataSet end=eof;  
  by CharVar;  
  if eof then put 'Everything worked';  
run;
```

- Symptom:

```
ERROR: BY variables are not properly sorted on data set  
WORK.SORTEDSASDATASET.
```

Trap #3 -- Diagnosis

- SAS told the database to do the sort, and they don't agree on all of the details of what the final order should be
 - Handling of SAS missing values / RDBMS nulls: sort "low" or "high"
 - Handling of upper and lower case, accented characters, etc. ("linguistic sorting")

Trap #3 -- Treatment

- Maybe exact sort order doesn't matter – can you use NOTSORTED when reading "sorted" file?
- Make sure SAS does the sorting itself:
 - `OPTIONS SORTPGM=SAS;`
 - May take longer to run
- Use a DBMS-specific workaround
 - May be able to use pass-through SQL
 - Or SAS may provide a special "trick" for your database and situation

Trap #4 – What's not true, not false, and ...

- Background – You are comparing two values in a WHERE statement or SQL expression that is executed by the RDBMS.
 - WHERE Profit < 0
 - WHERE Profit ^= 0
- Symptom – You don't get the results you expect.

Trap #4 -- Diagnosis

- In SQL (the language of RDBMS), comparisons involving NULL values are neither TRUE nor FALSE!
 - Three-valued logic
- Conundrum – When Profit is NULL, NONE of the following is true:
 - Profit = 0, Profit ^= 0, Profit < 0, NOT(Profit < 0)
- And, even when A and B are both NULL, A = B is not true

Trap #4 -- Treatment

- If your comparison may be executed by the database, specify how NULLs should be handled explicitly to get the results you expect
 - where Profit < 0 **and Profit is not null**
 - where Profit < 0 **or Profit is null**
- See Fred Levine paper for more info

Trap #5 – Forgetting about efficiency

- Symptoms
 - Seemingly simple jobs take hours to run
 - You get frequent frantic calls from your database administrator
 - Other database users get together and slash your tires

Trap #5 continued

- An ounce of prevention ...
 - Don't extract unnecessary columns
 - Avoid repeated large extracts
 - Make the database do the summarization
(Special bonus slides next!)
 - Be careful of SAS-specific functions
SAS may have to pull in all of the data to filter
 - Be careful of "heterogeneous joins"
Filtering a huge DB table based on IDs that you have in SAS
Make the RDBMS do the work – see paper for techniques
 - Compress SAS data sets with wide text fields

In-Database Processing

- Relatively new capability, initially supported only for the Teradata database
- Expanded in 9.2M3 to DB2 (Unix and PC hosts) and Oracle
- Enhances certain PROCs so they have RDBMS do summarization and return results, rather than asking for all of the individual records
- Summarization functions supported by FREQ, SUMMARY/MEANS, REPORT, TABULATE

In-Database Processing (slide 2)

- SQLGENERATION=DBMS (default is NONE)
 - System option or LIBNAME statement option
- Can be **significant** improvements in throughput
- Recommended system options: SASTRACE, MSGLEVEL=I
- Some program options prevent its use
- “SAS® Presents In-Database Base Procedures in Practice” – SAS Global Forum 2010, Mebust & Ray

Conclusion

- You can certainly get your data out of your RDBMS and into SAS
- There are some traps to watch out for ...
- ... but, if you avoid these, you can keep yourself, your boss, other database users, and your DBA happy



Acknowledgements and Disclaimer

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration. Other brand and product names are registered trademarks or trademarks of their respective companies.

The contents of this paper are the work of the author and do not necessarily represent the opinions, recommendations, or practices of Westat.

Questions?

Mike Rhoads
Westat
1650 Research Blvd.
Rockville, MD 20850
MikeRhoads@Westat.com

<http://support.sas.com/resources/papers/proceedings09/141-2009.pdf>