

# Romancing Your Data: The Getting-to-Know-You Phase

**Carole Jesse**

**Twin Cities Area SAS Users Group**

**November 3, 2010**

# Overview

## Motivation

## Introduction to Oracle® Database Architecture and Data Dictionary

## The SYS Schema and Views

## Review of SAS/ACCESS® Interface to Oracle - Query Types

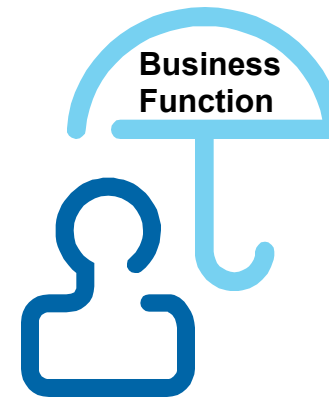
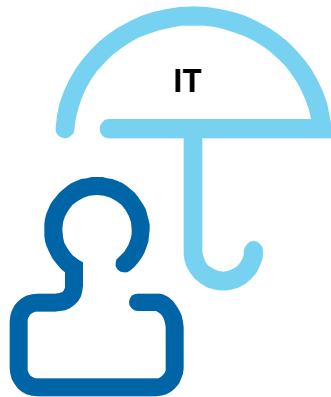
## Macro Variables in the SAS scripts

## Base SAS® Scripts:

- `1_Code_SysAllViews3Fam.sas`
- `2_Code_SysAllTables.sas`
- `3_Code_SysUserRolePrivs.sas`
- `4_Code_SysAllIndColumns.sas`
- `5_Code_Sys_PrimaryKeys.sas`

## Conclusions

# Motivation

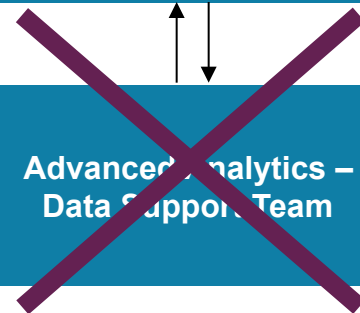


**Database and Server Administration:  
Builders/Maintainers of Oracle DBs  
i.e. DATA!**

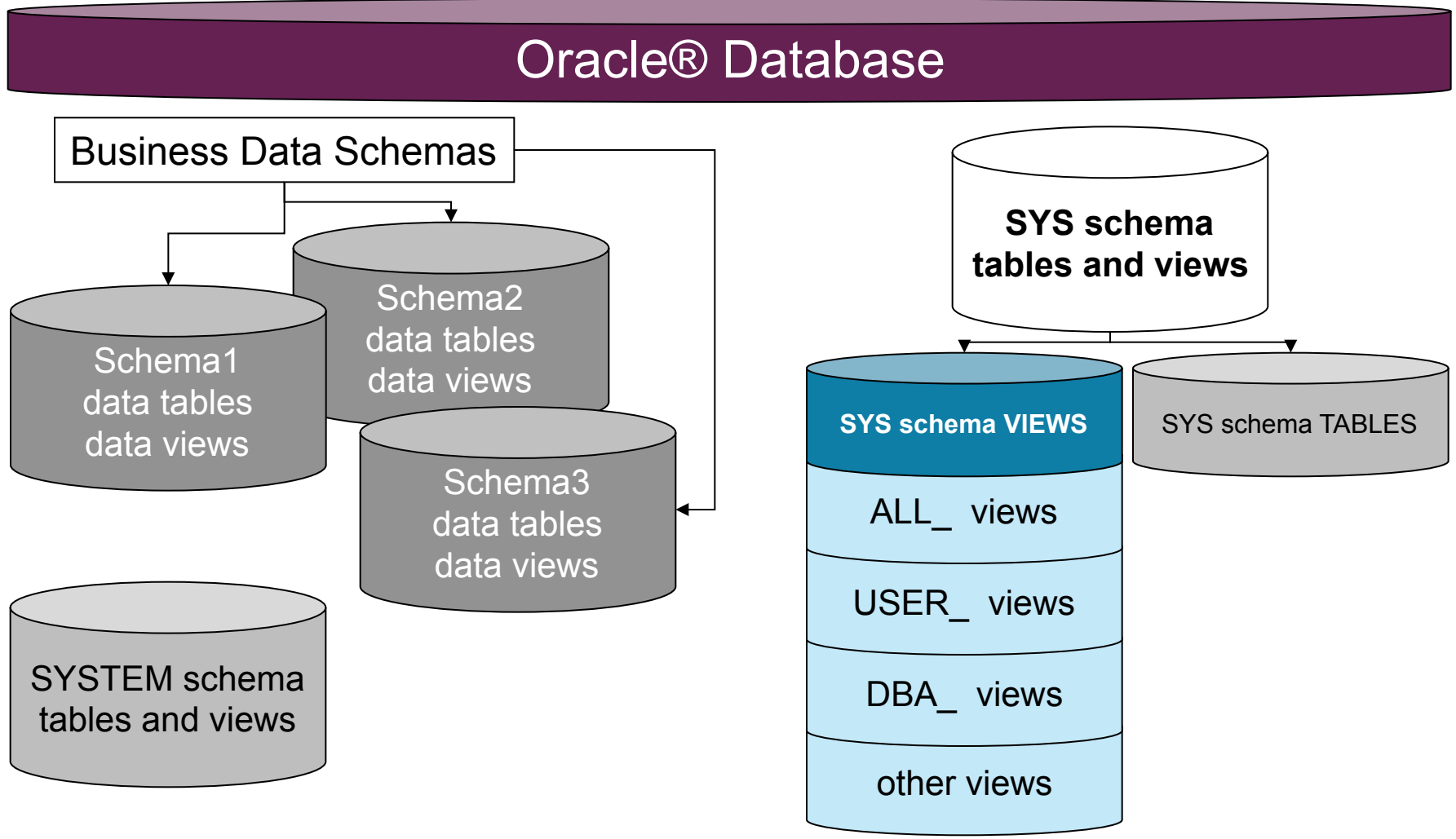
**Consumers of the Oracle DBs:  
Marketing Analysts, Statisticians,  
Econometricians, Report Analysts, etc.  
i.e. Advanced Analytics on the DATA!**

**Urban Legend? Analytics Data Support Team**

**Advanced Analytics –  
Data Support Team**



# Oracle® database architecture



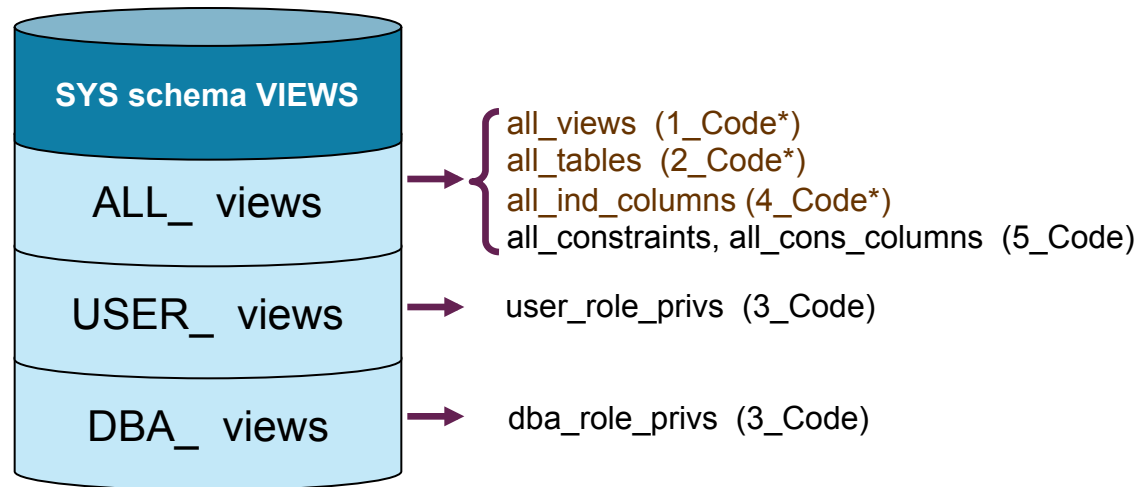
# 10 Oracle Databases... SYS View counts

SYS View counts for each of the three View families, across 10 Oracle databases (for user cjesse):

	Oracle Database alias	ALL_	DBA_	USER_
1	DWDB	288	3	290
2	DWDX	288	5	290
3	DWIN	288	3	290
4	GMINFP	270	8	270
5	PROD	291	562	287
6	PROD02	291	562	287
7	PLPS	280	548	275
8	WLTP	287	2	289
9	GMEDAPP	290	4	290
10	NEVP	288	10	290

# SYS Views of Interest (and Why)

Five SAS scripts that work with six of the SYS Views:



SYS View(s)	Question(s) about the database	Why should you care?
all_views	Where do I start?	Allows exploration of SYS Views.
all_tables	Is this table partitioned? How big is this table?	Determine pass-through SQL requirements and estimate expected run times.
user_role_privs dba_role_privs	What do I have access to?	Ensure proper access to objects in the database and transferability of code.
all_ind_columns	Is this table indexed? If so, on what?	Write more efficient query logic.
all_constraints all_cons_columns	What uniquely defines a record in this table?	Understand table merges (joins).

# SAS/ACCESS® Interface to Oracle: Libname Query

## Generic Syntax of the LIBNAME statement to an Oracle schema:

```
LIBNAME libref oracle USER='ORACLE-user-name' PASSWORD='ORACLE-password'  
PATH="ORACLE-database-specification" SCHEMA=schema-name;
```

```
/* Example: LIBNAME query in SAS/ACCESS Interface to ORACLE */  
  
/* Establish connection to the Oracle schema with LIBNAME statement */  
LIBNAME db1_sch1 oracle USER='cjesse' PASSWORD='bWFLA$01e'  
PATH="db1.server1.com" SCHEMA=sch1;  
  
/* LIBNAME query to table TBL1 in the SCH1 schema of the DB1 ORACLE database */  
PROC SQL;  
  create table WORK.TBL1 as  
  select col1, col2, col3  
  from db1_sch1.TBL1;  
QUIT;  
  
/* Alternatively, a DATA step. */  
DATA WORK.TBL1;  
  set db1_sch1.TBL1;  
  keep col1 col2 col3;  
RUN;
```

## The libname query:

- is simple! and
- the syntax is familiar to all SAS programmers, even the Newbs.

# SAS/ACCESS® Interface to Oracle: Pass-through Facility

## Generic Syntax of the CONNECT statement to an Oracle database within PROC SQL:

```
CONNECT TO oracle <AS Alias>
      (USER='ORACLE-user-name'   PASSWORD='ORACLE-password'
       PATH="ORACLE-database-specification");
```

```
/* Example: Pass-Through Facility query in SAS/ACCESS Interface to ORACLE */

/* Establish connection to Oracle database within PROC SQL */
/* Pass-through query to table TBL1 in the SCH1 schema of the DB1 ORACLE database */
PROC SQL;
CONNECT TO oracle AS db1 (USER='cjesse'   PASSWORD='bWFLA$01e'   PATH="db1.server1.com");
  CREATE table WORK.TBL1 as
  SELECT *           ← SAS SQL, SELECT
  FROM connection to db1 ← connection based on CONNECT statement
    (SELECT col1, col2, col3 from sch1.tbl1) ← Oracle pass-through SQL, SELECT
  ;
  DISCONNECT FROM db1;
QUIT;
```

## The pass-through facility query:

- allows the blending of the power of SAS with the power of Oracle SQL !
- allows communication from the Oracle side to SAS in the log (Oracle error messages)
- is a method that's likely required by the DBA for partitioned Oracle tables.

# Macro Variables in the SAS Scripts

The five SAS Scripts utilize up to six macro variables:

```

/*****/
/* Begin USER INPUTS */
/*****/
%let ODBcred= user='cjesse' pw='bWFLA$01e'; /* credentials to the DB, Case Sensitive */
%let ODBlong= database1.server1.com; /* path to the DB */
%let ODBshrt= db1sv1; /* alias for the DB */
%let unixpath= /usrv1/grp1/cjesse/; /* UNIX path for output, Case Sensitive */
%let OWNlong= 'OWNER1'; /* Schema name, ALL CAPS, in single quotes */
%let TBL= 'TABLE1'; /* Table name, ALL CAPS, in single quotes */
/*****/
/* End USER INPUTS */
/*****/

```


Code	&ODBCred	&ODBlong	&ODBshrt	&unixpath	&OWNlong	&TBL
1_Code_SysAllViews3Fam.sas	*	*	*	optional	NA	NA
2_Code_SysAllTables.sas	*	*	*	optional	*	NA
3_Code_SYSUserRolePrivs.sas	*	*	*	optional	NA	NA
4_Code_SysAllIndColumns.sas	*	*	*	optional	*	*
5_Code_Sys_PrimaryKeys.sas	*	*	*	optional	NA	optional

# Typical Layout of #\_Code... .sas

```
ODS HTML body="&unixpath.<filename>.html";
  Title1 "<Text>"; Title2 "<Text>";

  PROC SQL;
  CONNECT to oracle as &ODBshrt. (path="&ODBlong" &ODBcred. );
  /* CREATE table <tblname> as */
  SELECT
    < SAS SQL >
  FROM connection to &ODBshrt.
    (
      SELECT
        < Oracle pass-through SQL >
    )
    < SAS SQL >
  ;
  DISCONNECT FROM &ODBshrt.;
  QUIT;

  Title1; Title2;
ODS HTML close;
```



The Meat of the Query

# PROC SQL: 1\_Code\_SysAllViews3Fam.sas, part 1

```
PROC SQL;
CONNECT to oracle as &ODBshrt. (path="&ODBlong" &ODBCred. );
/* CREATE table SYSallviews as */

SELECT
SCANQ(VIEW_NAME,1,"_") as FAMILY,
*
FROM connection to &ODBshrt.
(
  SELECT
  VIEW_NAME
  FROM SYS.all_views
  WHERE OWNER='SYS'
)
WHERE SCANQ(VIEW_NAME,1,"_") in ('ALL','USER','DBA')
ORDER FAMILY, VIEW_NAME
;

DISCONNECT FROM &ODBshrt.;
QUIT;
```

} SAS SQL

} Oracle pass-through SQL

} SAS SQL

# SYS.all\_views

## 1\_Code\_SysAllViews3Fam.sas, Part 1 Results

SYS.all\_views: 11 columns

View CONTAINS:

Information on all the Views in the database, including those in SYS, as well as the Business data schemas.

Most important columns:

- OWNER (schema name)
- VIEW\_NAME

Punchline:

- a report of View names, by Family
- a starting point for Part 2 of this program.
- Part 2 generates a print of the first 15 rows of each of the View names in a user defined list.

Breakdown of SYS.ALL\_VIEWS in ALL\_, USER\_, DBA\_ For ORACLE database database1.server2.com

FAMILY	VIEW_NAME
ALL	ALL ALL TABLES
ALL	ALL APPLY
ALL	ALL APPLY CONFLICT COLUMNS
.	.
.	.
.	.
ALL	ALL WARNING SETTINGS
DBA	DBA AUTO SEGADV CTL
DBA	DBA AUTO SEGADV SUMMARY
DBA	DBA DATA FILES
.	.
.	.
.	.
DBA	DBA TABLESPACES
USER	USER ADVISOR ACTIONS
USER	USER ADVISOR DIRECTIVES
USER	USER ADVISOR FINDINGS
.	.
.	.
.	.
USER	USER WARNING SETTINGS

# PROC SQL: 2\_Code\_SysAllTables.sas

```
PROC SQL;  
CONNECT to oracle as &ODBshrt. (path="&ODBlong." &ODBcred. );  
/* CREATE table allparttables as */
```

```
SELECT  
TABLE_NAME, NUM_ROWS format=comma19.0, PARTITIONED  
FROM connection to &ODBshrt.
```

} SAS SQL

```
(  
  SELECT  
  TABLE_NAME, NUM_ROWS, PARTITIONED  
  FROM SYS.all_tables  
  WHERE OWNER=&OWNlong. and NUM_ROWS > 0 and PARTITIONED='YES'  
  ORDER by NUM_ROWS DESC, TABLE_NAME  
)  
;
```

} Oracle  
pass-through  
SQL

```
DISCONNECT from &ODBshrt.;  
QUIT;
```

# SYS.all\_tables

## 2\_Code\_SysAllTables.sas, Results

SYS.all\_tables: 49 columns (this is a very information rich SYS View)

View CONTAINS:

Information on all the Tables in the database.

The tables of most interest reside in the Business Data schemas.

Most important columns:

- OWNER (schema name)
- TABLE\_NAME
- NUM\_ROWS
- PARTITIONED

**Partitioned Tables on Schema 'OWNER5' in Database: database1.server2.com**

TABLE_NAME	NUM_ROWS	PARTITIONED
TABLE236	232,942,060	YES
TABLE285	36,673,534	YES

Punchline:

Must use Pass-Through queries on tables TABLE236 and TABLE285 in the OWNER5 schema of this Oracle database.

# PROC SQL: 3\_Code\_SysUserRolePrivs.sas

```
PROC SQL;
CONNECT to oracle as &ODBshrt. (path="&ODBlong" &ODBCred. );
/* CREATE table myroles_&ODBshrt. as*/

SELECT
*
FROM connection to &ODBshrt.
(
  SELECT
  USERNAME, GRANTED_ROLE /* GRANTEE, GRANTED_ROLE */
  FROM SYS.user_role_privs /* SYS.dba_role_privs */
)
;

DISCONNECT from &ODBshrt.;
QUIT;
```

} SAS SQL

} Oracle  
pass-through  
SQL

# SYS.user\_role\_privs (SYS.dba\_role\_privs)

## 3\_Code\_SysUserRolePrivs.sas, Results

SYS.user\_role\_privs: 5 columns

View CONTAINS:

Information related to how database Roles and Privileges are assigned to the users of the database.

Most important columns:

- USERNAME (GRANTEE in DBA\_)
- GRANTED\_ROLE

Roles/Privs granted for cjesse on Database: database1.server2.com

USERNAME	GRANTED_ROLE
CJESSE	ROLE_OWNER5_READ
CJESSE	ROLE_OWNER8_READ

Punchline:

- Report of granted roles for the user on a specific database.
- Compare results to that for other users to determine transferability of code.

# PROC SQL: 4\_Code\_SysAllIndColumns.sas

```
PROC SQL;
CONNECT to oracle as &ODBshrt. (path="&ODBlong" &ODBcred. );
/* CREATE table &TBL.indexes as */

SELECT
*
FROM connection to &ODBshrt.
(
  SELECT
  INDEX_NAME, COLUMN_POSITION, COLUMN_NAME
  FROM SYS.all_ind_columns
  WHERE TABLE_OWNER=&OWNlong. and TABLE_NAME=&TBL.
  ORDER by INDEX_NAME, COLUMN_POSITION
)
;

DISCONNECT from &odbsqrt.;
QUIT;
```

} SAS SQL

} Oracle  
pass-through  
SQL

## SYS.all\_ind\_columns

### 4\_Code\_SysAllIndColumns.sas, Results TABLE11

SYS.all\_ind\_columns: 9 columns

View CONTAINS:

Information related to how Tables are indexed.

Most important columns:

- INDEX\_NAME
- COLUMN\_POSITION
- COLUMN\_NAME

Indexes on Schema 'OWNER5', Table 'TABLE11' in Database: database1.server2.com

INDEX_NAME	COLUMN_POSITION	COLUMN_NAME
TABLE11_PK	1	FIPS_STATE_CODE
TABLE11_PK	2	FIPS_COUNTY_CODE

Punchline:

- Table TABLE11 has one composite index created by the DBA, called TABLE11\_PK.
- The column FIPS\_STATE\_CODE is in the first position of the index.
- The column FIPS\_COUNTY\_CODE is in the second position.

# SYS.all\_ind\_columns

## 4\_Code\_SysAllIndColumns.sas, Results TABLE236

Indexes on Schema 'OWNER5', Table 'TABLE236' in Database: database1.server2.com

INDEX_NAME	COLUMN_POSITION	COLUMN_NAME
NI1_TABLE236	1	BATCH_DATE
NI1_TABLE236	2	ACCOUNT_NUMBER
NI1_TABLE236	3	BAL_PRIN
NI1_TABLE236	4	LATE_FEE_UNCOLL
NI2_TABLE236	1	BATCH_DATE
NI2_TABLE236	2	ACCOUNT_NUMBER
NI2_TABLE236	3	STRAT_COLLECTIONS
NI2_TABLE236	4	LOAN_STATUS_CODE
NI2_TABLE236	5	COLLECTION_RESPONSE_CODE
NI2_TABLE236	6	COLLECTOR_ID
NI2_TABLE236	7	SUB_SERVICED_IND
NI3_TABLE236	1	BATCH_DATE
NI3_TABLE236	2	ACCOUNT_NUMBER
NI3_TABLE236	3	LOAN_STATUS_CODE
NI3_TABLE236	4	CREDIT_MAX
NI3_TABLE236	5	BLOCK_NUMBER
NI3_TABLE236	6	INVESTOR_NUMBER
NI3_TABLE236	7	BAL_PRIN_PRIOR
NI3_TABLE236	8	REOPENED_IND

### Punchline:

- Table TABLE236 has many composite indexes created by the DBA.
- Most have BATCH\_DATE or ACCOUNT\_NUMBER in the first position of the composite.
- Use this knowledge in the variable order of subset 'where' logic in the Oracle SQL!

# Use the results for better coding!

## Pass-through SQL BEFORE SYS View

```
(  
  SELECT ACCOUNT_NUMBER, BAL_PRIN  
  from OWNER5.TABLE236  
  where  
  580 < FICO_SCORE_CURR  
  and  
  FICO_SCORE_CURR < 620  
  and  
  BATCH_DATE = '28-FEB-2010'  
)
```

## Pass-through SQL AFTER SYS View

```
(  
  SELECT ACCOUNT_NUMBER, BAL_PRIN  
  from OWNER5.TABLE236  
  where  
  BATCH_DATE = '28-FEB-2010'  
  and  
  580 < FICO_SCORE_CURR  
  and  
  FICO_SCORE_CURR < 620  
)
```

### Punchline:

- Using the knowledge about table indexing for the variable order in 'where' logic on the Oracle side yields a 21-31% reduction in run time! (depending on server and traffic)
- You are more productive and you get a bigger bonus!

# PROC SQL: 5\_Code\_Sys\_PrimaryKeys.sas

```
PROC SQL;
CONNECT to oracle as &ODBshrt. (path="&ODBlong." &ODBCred. );
/* CREATE table TblPriKeys as */

SELECT
*
FROM connection to &ODBshrt.
(
  SELECT
  b.OWNER,
  a.TABLE_NAME, a.COLUMN_NAME, a.POSITION,
  b.STATUS
  FROM SYS.all_constraints b, SYS.all_cons_columns a
  WHERE
    a.TABLE_NAME = &TBL.
  AND b.CONSTRAINT_TYPE = 'P'
  AND b.CONSTRAINT_NAME = a.CONSTRAINT_NAME
  AND b.OWNER = a.OWNER
  ORDER BY b.OWNER, a.TABLE_NAME, a.POSITION
)
;

DISCONNECT from &ODBshrt.;
QUIT;
```

} SAS SQL

} Oracle  
pass-through  
SQL

# SYS.all\_constraints & SYS.all\_cons\_columns

## 5\_Code\_Sys\_PrimaryKeys.sas, Results

SYS.all\_constraints: 20 columns

View CONTAINS:

Descriptions of table constraints on tables.

Most important columns:

- OWNER (schema)
- STATUS
- CONSTRAINT\_NAME
- CONSTRAINT\_TYPE

SYS.all\_cons\_columns: 5 columns

View CONTAINS:

Descriptions of the columns that are specified in table constraints.

Most important columns:

- TABLE\_NAME
- COLUMN\_NAME
- POSITION
- CONSTRAINT\_NAME
- OWNER (schema)

**All Primary Keys in Table 'TABLE1' in the ORACLE database: database1.server1.com**

OWNER	TABLE_NAME	COLUMN_NAME	POSITION	STATUS
OWNER1	TABLE1	ASSET_SEQ_ID	1	ENABLED
OWNER1	TABLE1	PAYOFF_TYPE_CODE	2	ENABLED

Punchline:

- Report which identifies the primary key structure for a table.
- These are the potential merge variables for this table.

## Conclusions and Contact Information

---

Utilizing SAS/Access Interface to Oracle to explore the Oracle Data Dictionary is the Getting-to-Know-You Phase of Romancing Your Data!

Carole Jesse

LinkedIn: <http://www.Linkedin.com/in/CaroleJesse>

SASCommunity: <http://www.sascommunity.org/wiki/User:CaroleJesse>

Twitter: <http://www.twitter.com/CaroleJesse>

The full MWSUG10 paper can be downloaded:

<http://preview.tinyurl.com/MWSUG10-015>