

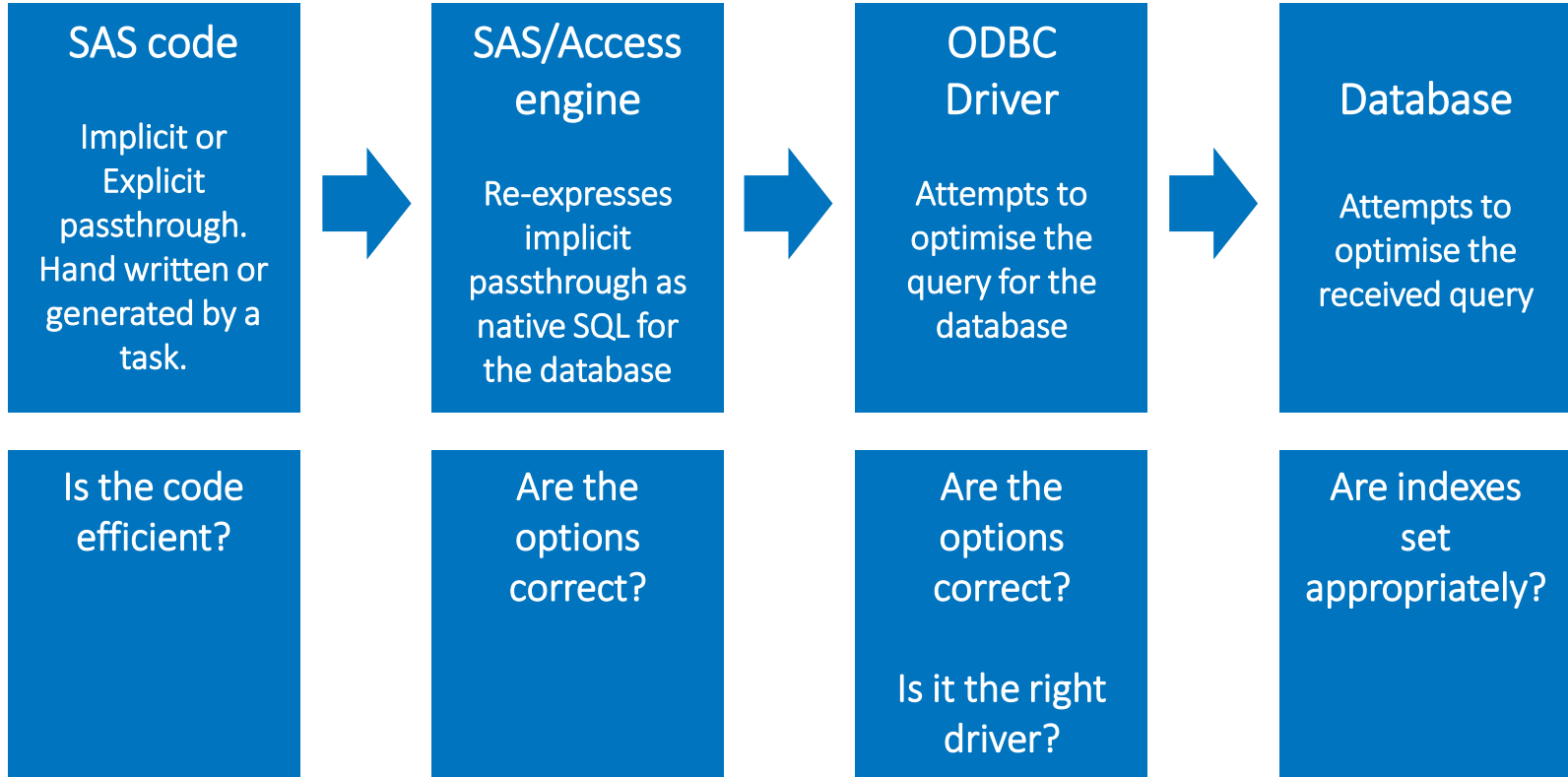


SAS and SQL databases

Understanding and tuning queries and settings

Under the covers

The query undergoes multiple steps



What about metadata libraries?

Metadata adds another layer of abstraction

Using Base v9 Engine

```
Libname mktg v9 "C:\marketing";
```

Using Meta engine

```
Libname demo meta library="Marketing data";
```

Retrieve assignment information
and construct libname statement

Library	Libname	Engine	Path
Marketing data	Mktg	V9	C:\marketing

```
Libname mktg v9 "C:\marketing";
```

SAS

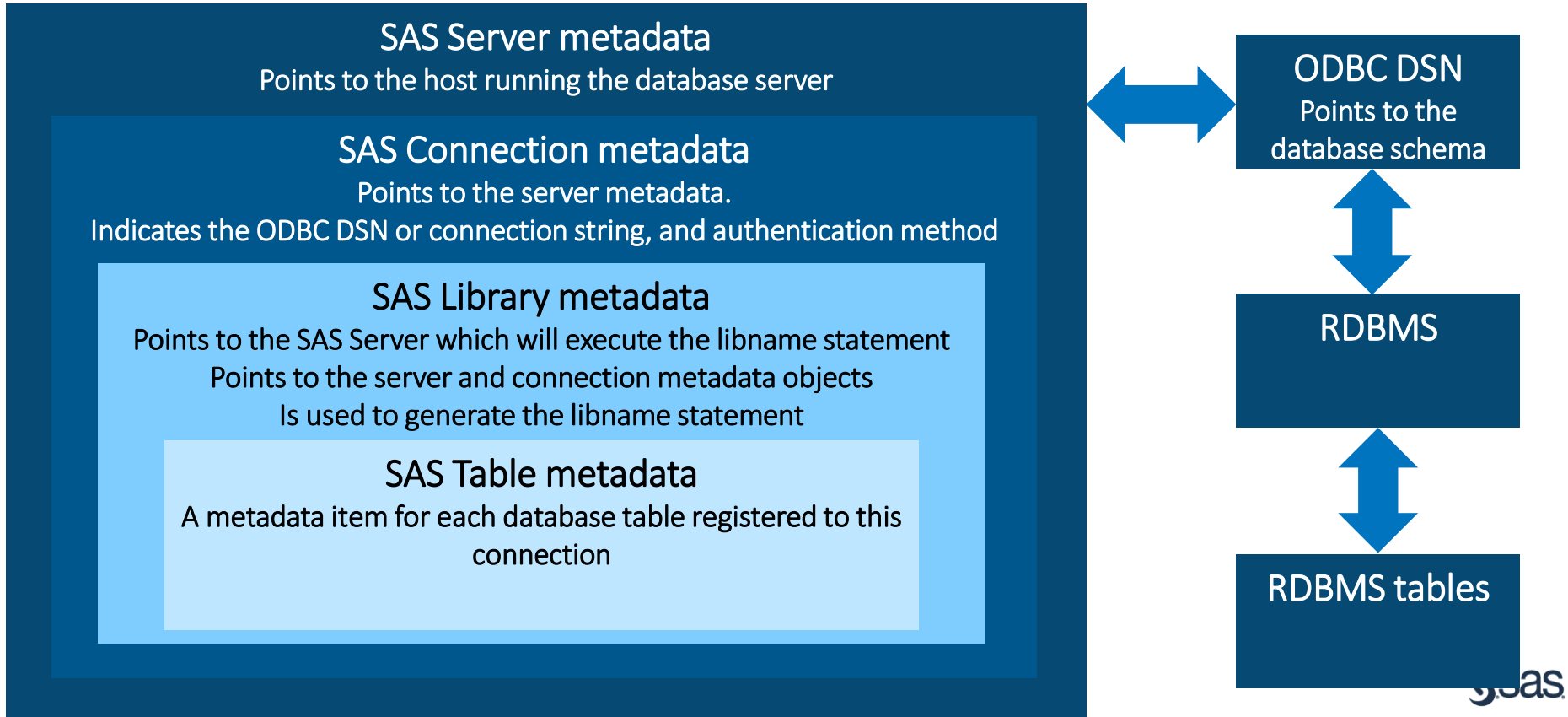
Assign library

V9 engine

Data

Under the covers

There is a nesting of metadata leading to the data



Assignment

Initialisation delays

- Libraries are often pre-assigned as a convenience feature
 - Making multiple connections will slow down server initialisation. Consider setting deferred connections (defer=yes) Defer= can be set on the **Connection** tab
- Engine options
 - Native
 - When using the native engine no SAS metadata security is applied
 - Meta
 - When using the META engine, metadata security is applied
 - External
 - Uses an autoexec to define the connection
 - Metadata does not generate a connection

Library

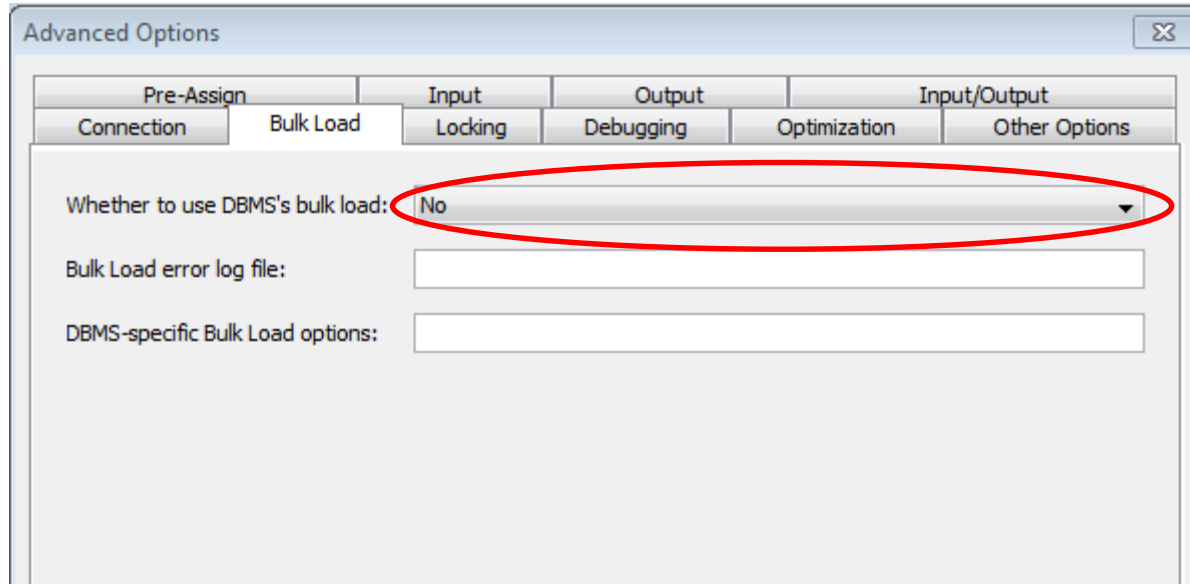
Advanced options - Connection

Advanced Options

Pre-Assign		Input		Output		Input/Output	
Connection	Bulk Load	Locking	Debugging	Optimization	Other Options		
Type of connection:	<input type="text" value="SHAREDREAD"/>						
Connection group name:	<input type="text"/>						
User-defined connection initialization command:	<input type="text"/>						
User-defined connection termination command:	<input type="text"/>						
User-defined library initialization command:	<input type="text"/>						
User-defined library termination command:	<input type="text"/>						
Whether to defer a connection until needed:	<input type="text" value="No"/>						
Whether to automatically drop utility connections:	<input type="text" value="No"/>						
Login timeout, in seconds:	<input type="text"/>						

Library

Advanced options - Bulkload



Advanced Options

Pre-Assign		Input	Output	Input/Output	
Connection	Bulk Load	Locking	Debugging	Optimization	Other Options

Whether to use DBMS's bulk load: **No**

Bulk Load error log file:

DBMS-specific Bulk Load options:

Library

Advanced options - optimisation

Advanced Options

Pre-Assign		Input		Output		Input/Output	
Connection	Bulk Load	Locking	Debugging	Optimization	Other Options		
Block insert buffer size:				1			
Block read buffer size:				0			
Pass functions to the DBMS that match those supported by SAS:							
Pass DELETE to the DBMS:				Blank (no value)			
Whether to use indexes:				Yes			
Whether to check for null keys when generating where clauses:				Yes			
Multi data source optimization:				NONE			
Whether to use data source's SQL insert:				Blank (no value)			
Whether to use Current-of-Cursor SQL to update/delete rows:				Blank (no value)			
Whether to create a spool file for two-pass processing:				Yes			
Threaded DBMS access:				(THREADED_APPS,3)			

Optimization Tab - ODBC Libraries

options to optimize the access to the database table

- Block insert buffer size
 - specifies the number of rows in a single Insert operation. Corresponds to the INSERTBUFF= option in the LIBNAME statement.
- Block read buffer size
 - specifies the number of rows of DBMS data to read into the buffer. Corresponds to the READBUFF= option in the LIBNAME statement.
- Pass functions to the DBMS that match those supported by SAS
 - specifies whether functions are passed to ODBC that are normally not passed. Use of this option can cause unexpected results. Corresponds to the SQL_FUNCTIONS= option in the SAS/ACCESS LIBNAME statement.
- Pass DELETE to the DBMS
 - specifies that an SQL delete statement is passed directly to the DBMS for processing. Selecting this option improves performance, because SAS does not have to read the entire result set and delete one row at a time. Corresponds to the DIRECT_EXE= option in the SAS/ACCESS LIBNAME statement.



Efficient SAS/SQL code

Efficient SAS/SQL code

- Optimisation has four competing factors
 - CPU
 - Memory
 - I/O (disk and network)
 - Disk space

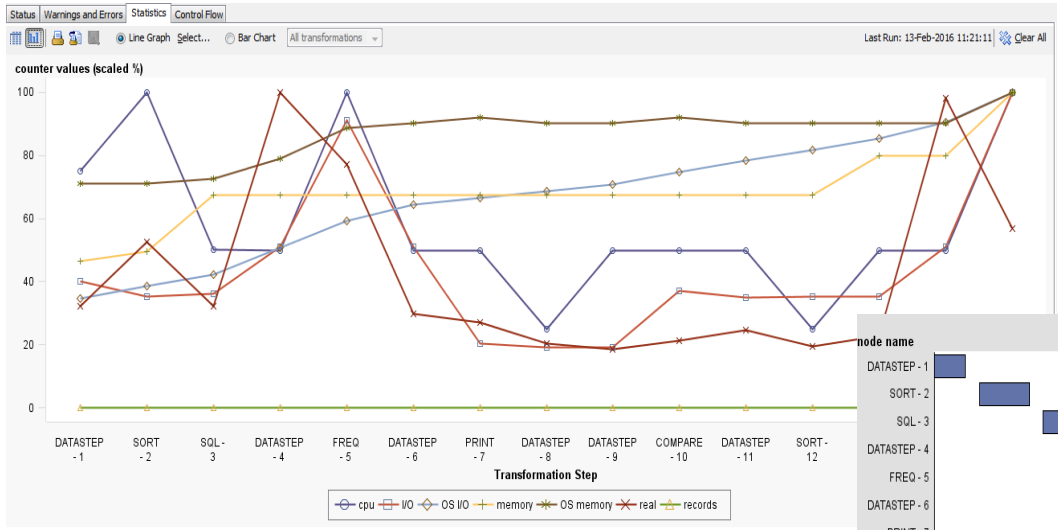
Efficient SAS/SQL code

Overview

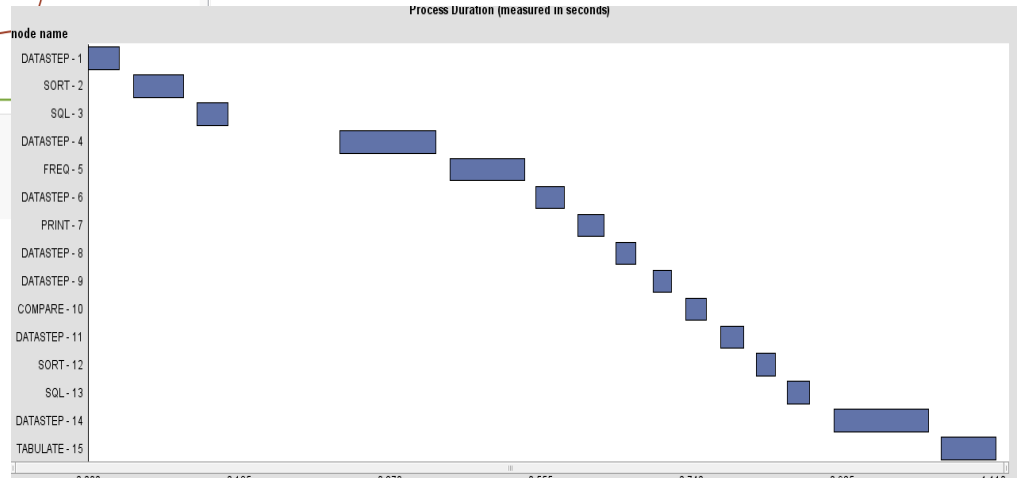
- Basic principles:
 - Don't do more work than you need to
 - Optimise for your environment
 - Go for the quick wins first
 - Jobs which take the most time
 - Jobs which are run most often
 - Benchmark after each change

Efficient SAS/SQL code

Overview



Data Integration Studio produces detailed information on resources used and time taken by each step



In base SAS and SAS Enterprise Guide use **options fullstimer** to get more information on the log



Efficient SAS/SQL code

Overview

- NOTE: The data set WORK.SAMPLE2 has 317223 observations and 27 variables.

- NOTE: DATA statement used (Total process time):

- real time 2.10 seconds
- cpu time 0.28 seconds

- NOTE: The data set WORK.SAMPLE2 has 317223 observations and 27 variables.

- NOTE: DATA statement used (Total process time):

- real time 0.27 seconds
- cpu time 0.28 seconds

- NOTE: The data set WORK.SAMPLE2 has 317223 observations and 27 variables.

- NOTE: DATA statement used (Total process time):

- real time 0.27 seconds
- cpu time 0.28 seconds

On the second and third runs the input data set was cached so the runtime was much lower. On a larger data set this effect will be reduced

Proc SQL options

- `_TREE_METHOD`
 - Displays a diagram of the SQL query plan
 - Useful in optimising a query
- `BUFFERSIZE / UBUFSIZE=1000000`
 - Specifies the number of rows SAS reads in one IO
 - Ideally, should fit the smallest table in a query entirely into memory
- `Readbuff=` and `Writebuff=`

SQL Passthrough

Implicit vs. Explicit

```
libname demo odbc (dsn=demo);  
proc sql;  
  select *  
  from demo.customers  
  where upcase(LN) ="SMITH";  
quit;
```

```
proc sql;  
  connect to odbc (dsn=demo);  
  select *  
  from connection to odbc  
  (  
    select *  
    from customers  
    where upper(LN)='SMITH'  
  )  
;  
quit
```


Implicit SQL Passthrough

Implicit SQL before and after passthrough

```
proc sql;  
  select *  
  from demo.customers  
  where upcase(LN) = "SMITH";  
quit;
```

```
select *  
from customers  
where upper(LN) = 'SMITH'
```

Use the SASTRACE option to see the code passed through by the library engine
options sastrace=',,,'d' sastraceloc=saslog nostsuffix;

Passthrough on "Non - SQL" SAS code

- Any data step or proc step in SAS will attempt to pass through as much code as possible
 - Where clauses
 - Calculations
 - Functions
 - Summarisation
 - Sorting
- For example a **proc sort** may be automatically **re-expressed** as SQL

```
select * from..  
order by..
```



SAS/Access engine



ODBC Driver considerations

ODBC Driver

- The correct ODBC driver can make a 10x difference in performance
- The tuning options will vary with each driver
- The DataBase Administrator should be involved in tuning, to show the code which is actually running and advise on tuning.



Database considerations

Indexes

Efficient subsetting of large data

- Index tables which are:
 - Large
 - Read multiple times between updates
 - Usually sampled or subset
- Index variables which are:
 - Used in filters
 - Have high cardinality

Use the MSGLEVEL option to see the information on index usage

options msglevel=i;

Sensible starting values for SAS options

Option	Value
bufno	128 then tune up or down
bufsize	0 (zero) or 128k
memsize	Depends on concurrency, data size, physical memory. ~ 4G?
sortsize	1G, or 75% of memsize - whichever is largest
maxmemquery	0 (zero)

THANK YOU

Wrap-up

- Ideas for future sessions, SAS Forum UK, any speaker volunteers
- Re-vamped [Customer Loyalty website](#)
- Any other business

Thank you for joining us today, we appreciate your time and participation



LET'S DO
LUNCH

