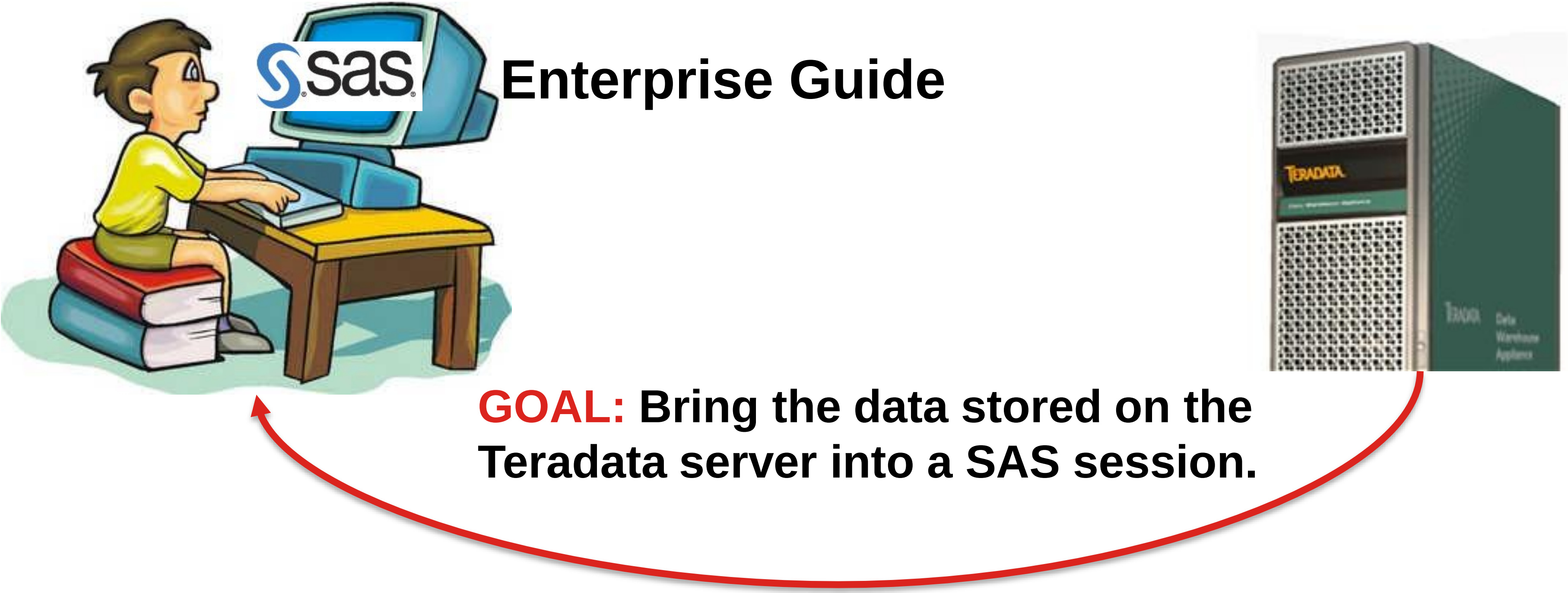
CONNECTING SAS TO A TERADATA SERVER

KEVIN MARK
DATA SCIENTIST, GROUP COLLECTIONS

WEDNESDAY 25 OCTOBER 2017

Problem of data not in SAS format



Enterprise Guide

GOAL: **Bring the data stored on the Teradata server into a SAS session.**

Images from http://www.teradata.com/Resources/Videos/Enabling-in-database-processing-with-SAS-ACCE and http://www.teradata.com/Solutions-and-Industries/sas-optimization/

Overview

Connecting SAS with Teradata

Two interfaces to connect SAS with Teradata

Concealing your Teradata password

Importing Teradata data to SAS

Joining a small SAS dataset with Teradata data

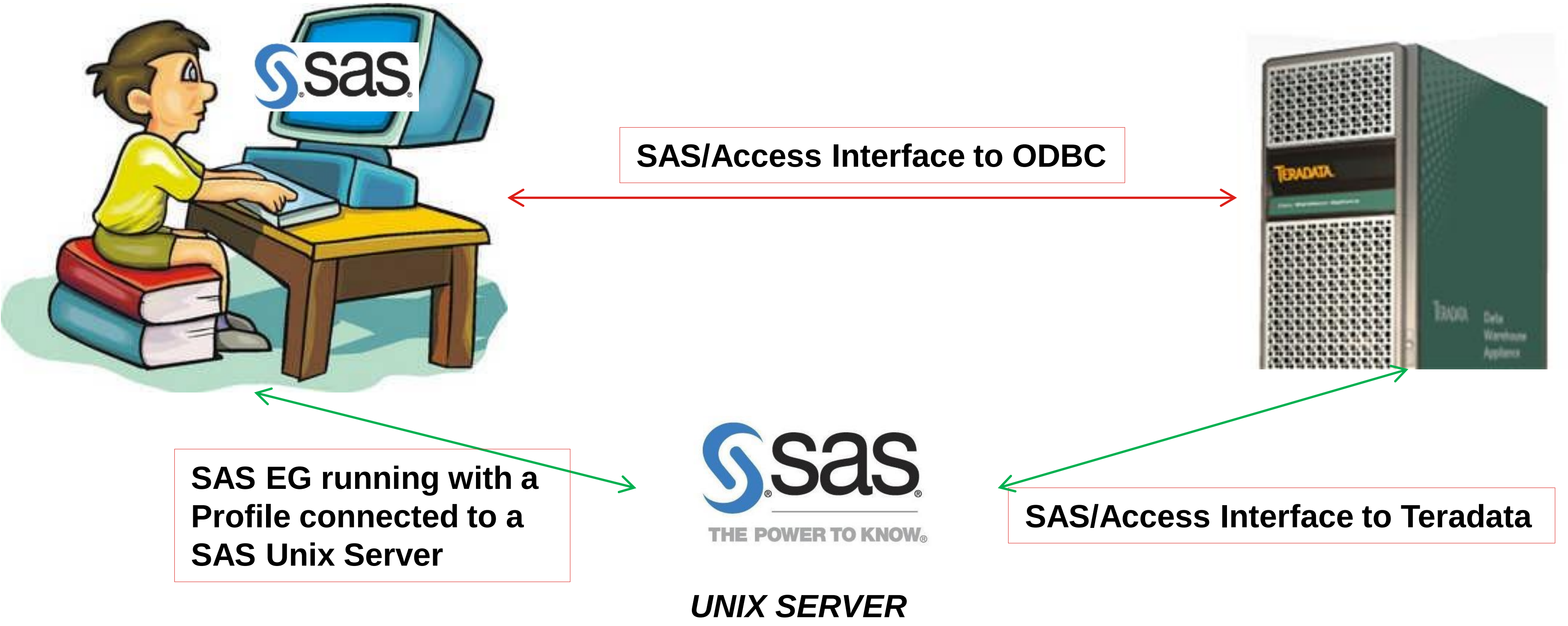
Questions

Disclaimer: The presentation are the views of the presenter and not that of the Westpac Group.

CONNECTING SAS WITH TERADATA

Two interfaces

Two interfaces to connect SAS with Teradata



SAS/Access Interface to ODBC

SAS EG running with a Profile connected to a SAS Unix Server

SAS/Access Interface to Teradata

UNIX SERVER

Images from http://www.teradata.com/Resources/Videos/Enabling-in-database-processing-with-SAS-ACCE and http://www.teradata.com/Solutions-and-Industries/sas-optimization/

Battle of the Interfaces

Access Interface to ODBC	Access Interface to Teradata
Connects from local PC Server	Connects from SAS Unix Server • Requires SAS EG to be connected to SAS Unix Server
Implicit connection via a libname statement	**Implicit** connection via a libname statement
Explicit connection via an ODBC connection in a SQL procedure	**Explicit** connection via a Teradata connection in a SQL procedure
	Ability to **execute** Teradata commands from SAS • Including ability to create and use temporary (volatile) tables.

CONCEALING YOUR PASSWORD

Concealing your Teradata password

Situation

User names and passwords to a Teradata server need to be specified in SAS code.

We want to write, save and run code without divulging passwords, especially in LOGs.

The PWENCODE Procedure will encrypt a password which you can use.

Not perfect. See SAS Documentation on the procedure for how to use this securely.

Example SAS Code:

```
proc pwencode in=‘my_password’;
run;
```

The SAS Log outputs the following:

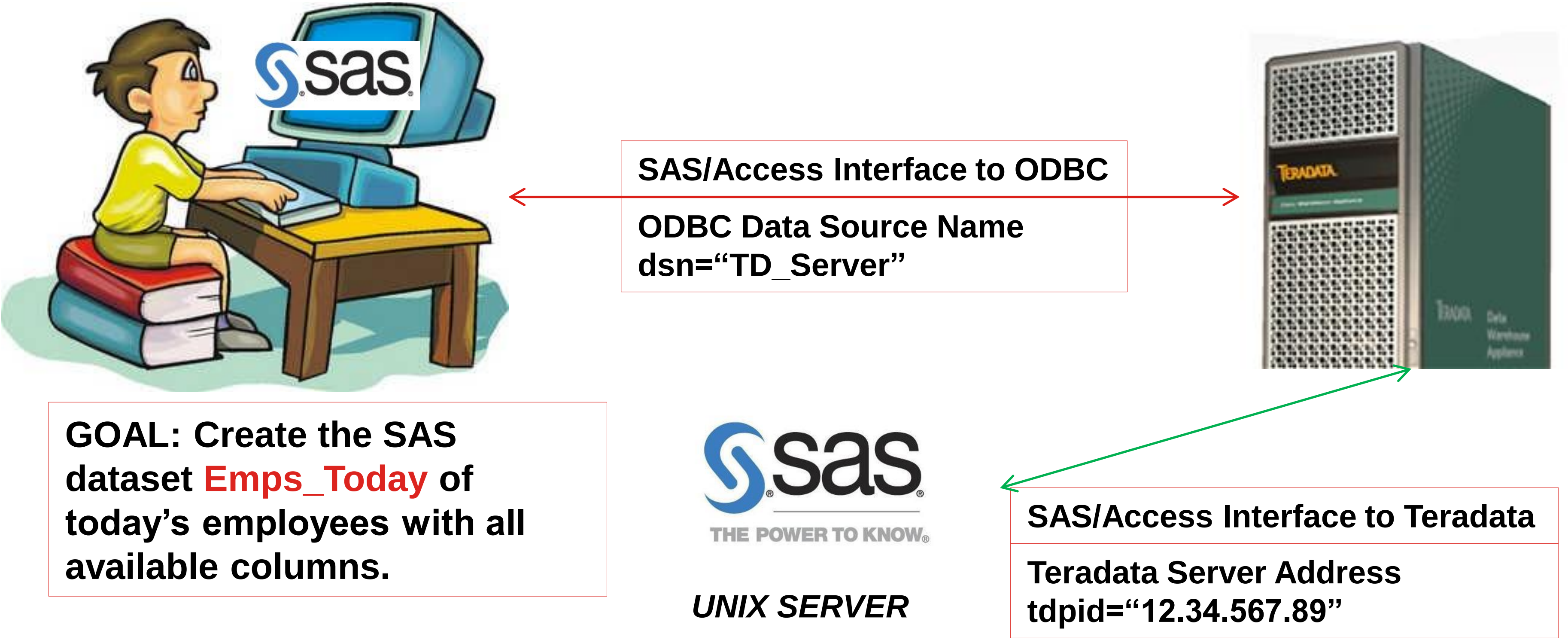
```
{sas002}
DBCC5712369DE1C65B19864C1564FB850F39
8DCF
```

```
%let password={sas002}
DBCC5712369DE1C65B19864C1564FB850F39
8DCF;
```

IMPORTING TERADATA DATA TO SAS

Importing Teradata data to SAS

Database=DB1 Table=Employees
Columns: employee_id, date,…



SAS/Access Interface to ODBC

ODBC Data Source Name
dsn="TD_Server"

GOAL: Create the SAS dataset Emps_Today of today's employees with all available columns.

UNIX SERVER

SAS/Access Interface to Teradata

Teradata Server Address
tdpid="12.34.567.89"

Images from http://www.teradata.com/Resources/Videos/Enabling-in-database-processing-with-SAS-ACCE and http://www.teradata.com/Solutions-and-Industries/sas-optimization/

Implicit connection via a libname statement

Access Interface to ODBC

```
libname TD_DB1
   odbc
   dsn=“TD_Server”
   user=“user-id”
   password=“&password.”
   schema=“DB1”;

data work.Emps_Today;
   set TD_DB1.Employees;
   where date=‘25OCT2017’d;
run;
```

Access Interface to Teradata

```
libname TD_DB1
   teradata
   tdpid=“12.34.567.89”
   user=“user-id”
   password=“&password.”
   database=“DB1”;

data work.Emps_Today;
   set TD_DB1.Employees;
   where date=‘25OCT2017’d;
run;
```

Explicit connection via a SQL Procedure

Access Interface to ODBC

```
proc sql;
   connect to ODBC (
       dsn=“TD_Server”
       user=“user-id”
       password=“&password.”);

   create table work.Emps_Today as
select * from connection to ODBC
    (select * from DB1.Employees
   where date=date ‘2017-10-25’);
quit;
```

Access Interface to Teradata

```
proc sql;
   connect to teradata (
       tdpid=“12.34.567.89”
       user=“user-id”
       password=“&password.”);

   create table work.Emps_Today as
select * from connection to teradata
    (select * from DB1.Employees
   where date=date ‘2017-10-25’);
quit;
```

JOINING A SMALL SAS DATASET

WITH TERADATA DATA

SAS Dataset=EMPS_Output
Columns: employee_id, date,…
Has only 100 rows.

Database=DB1 Table=Employees
Columns: employee_id, date,…
Has millions of rows

SAS Dataset=EMPS
Columns: employee_id, date
Has only 100 rows.

Upload SAS
Dataset to
Teradata

Create a temporary (volatile)
table in Teradata called
EMPS_TEMP

Join data in
Teradata

Images from http://www.teradata.com/Resources/Videos/Enabling-in-database-processing-with-SAS-ACCE and http://www.teradata.com/Solutions-and-Industries/sas-optimization/

Joining a small SAS dataset with Teradata data

Connect globally to temporary Teradata

```
libname temp teradata
    tdpid=“12.34.567.89”
    user=“user-id”
    password=“&password.”
    connection=GLOBAL
    dbmstemp=YES;
proc sql;
    connect to teradata (
        tdpid=“12.34.567.89”
        user=“user-id”
        password=“&password.”
        connection=GLOBAL);
```

Create a volatile table EMPS_TEMP

```
  execute(
create multiset volatile table
EMPS_TEMP (
employee_id VARCHAR(10)
date        DATE)
primary index (employee_id)
on commit preserve rows;
  ) by teradata;

  execute (commit work) by teradata;
quit;
```

Joining a small SAS dataset with Teradata data

Upload SAS data to a Teradata volatile table

```
proc append
 base=temp.EMPS_TEMP (MULTISTMT=yes)
 data=EMPS (keep=employee_id date);
run;
```

Joining a small SAS dataset with Teradata data

Create the resultant SAS dataset with …

```
proc sql;
   connect to teradata (
      tdpid=“12.34.567.89”
      user=“user-id”
      password=“&password.”
      connection=GLOBAL);
   create table work.EMPS_OUTPUT as
select * from connection to teradata
(
```

…an explicit Teradata SQL select statement

```
select
   b.*
from EMPS_TEMP a
   inner join Employees b
      on b.employee_id=a.employee_id
      and b.date=a.date

);

quit;
```

Thank you and Questions

kevin.mark@westpac.com.au