SAS & Databases
Who Does What Where ?!

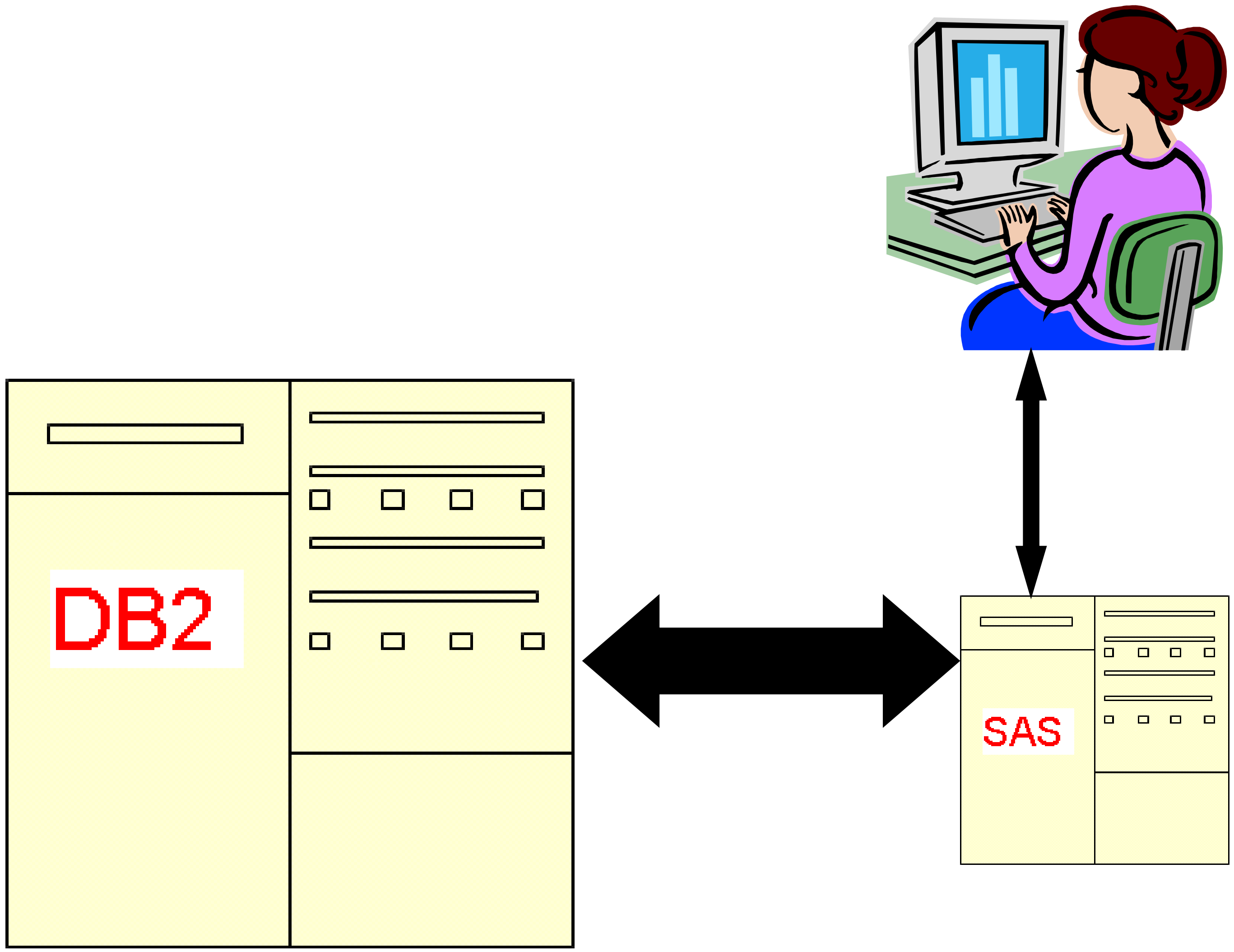
Harry Droogendyk, Stratia Consulting Inc.
GHSUG – 2014-05-09

Big Mountain of Data

Move the Mountain ?

Move the Mountain !

Our World



SAS and Databases

- SAS Access software
  - allows direct access to back-end DB
  - SAS/Access handles interface
    - available for all major DBs, eg. Oracle, Teradata, DB2
    - if all else fails -- ODBC
    - essentially APIs to communicate with the DB

- SQL "Pass Through"
  - explicit
  - implicit

SAS and Databases – Pass Through

- Explicit
  - SAS program connects to DB
  - SQL is "passed through" to DB
    - SQL must be native to DB

- Implicit
  - LIBNAME with DB specific "engine"
  - write SAS code
  - SAS decides and takes care of it for you
  - EG generates implicit pass-through queries

Explicit Pass-Through

- open a program window in EG

```
proc sql;
  connect to teradata ( user=&user pass="&pass"
                                                server=&server);
  create table pp_sum_cube as
    select * from connection to teradata
             (
        select s.brand_name, m.*
        from mi_analytics.v_bm_f_pp_migration_sum_cube m
             left join
            mi_data.bm_d_segment                      s
          on m.segment_cd = s.segment_cd
       where m.period_dt in ( '2014-03-01', '2014-03-02' )
       order by m.period_dt, s.brand_name
             );
quit;
```

Implicit Pass-Through

- You write SAS
- SAS interprets your code and it writes Teradata SQL

    - functions
    - … if possible …

- In Database processing
    - SAS Procedures
    - SAS formats

Implicit Pass-Through

```
libname mitddata teradata user=&user pass="&pass"
                 server=&server sql_functions=all ;

proc sql;
      select segment_tier_0, count(*) as cnt
        from mitddata.BM_D_Segment
       group by segment_tier_0
       order by segment_tier_0
       ;
quit;
```

- SAS LIBNAME statement – Teradata engine
- SAS SQL

- summary query runs entirely in Teradata
    - small result set returned to SAS

Implicit Pass-Through

```
libname mitddata teradata user=&user pass="&pass"
             server=&server sql_functions=all;

proc freq data = mitddata.BM_D_Segment;
      tables segment_tier_0 /missing nopercent ;
run;
```

```
514  proc freq data = mitddata.BM_D_Segment;
515      tables segment_tier_0 /missing nopercent ;
516  run;

NOTE: SQL generation will be used to construct frequency
and crosstabulation tables.
```

Implicit Pass-Through – Uh oh…

```
proc sql;
      select adj_type, adj_reason, count(*) as cnt

        from mitddata.BM_D_DA_Adjustments

       where substr(adj_reason,1,3) = 'M2M'
         and intck('month',eff_dt,exp_dt) > 1

       group by adj_type, adj_reason
       order by adj_type, adj_reason
       ;
quit;
```

- runs *forevvvvvvvvvver*

Implicit Pass-Through – Uh oh…

- SUBSTR and INTCK are SAS functions…
- SQL_Functions = ALL

- finite list
  - SUBSTR is there
  - not INTCK
  - what did SAS do ?

- better than you might imagine

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

Implicit Pass-Through – Uh oh…

```
SAS_SQL:  Unable to convert the query to a DBMS specific
SQL statement due to an error.
ACCESS ENGINE:  SQL statement was not passed to the
DBMS, SAS will do the processing.

TERADATA: trqacol- No casting. Raw row size=66, Casted size=74, CAST_OVERHEAD_MAXPERCENT=20%
TERADATA_8: Prepared: on connection 12

SELECT "ADJ_TYPE","ADJ_REASON","EFF_DT","EXP_DT"
        FROM mi_data."BM_D_DA_Adjustments"

WHERE  ( (SUBSTR("ADJ_REASON", 1, 3) = 'M2M' ) )

TERADATA_9: Executed: on connection 12
SELECT "ADJ_TYPE","ADJ_REASON","EFF_DT","EXP_DT" FROM mi_data."BM_D_DA_Adjustments"  WHERE  ( (
SUBSTR("ADJ_REASON", 1, 3) = 'M2M' ) )

TERADATA: trget - rows to fetch: 30

TERADATA: trforc: COMMIT WORK
```

Implicit Pass-Through – Caveat

Due to incompatibility in date and time functions between Teradata and SAS, Teradata might not process them correctly.

Check your results to determine whether these functions are working as expected

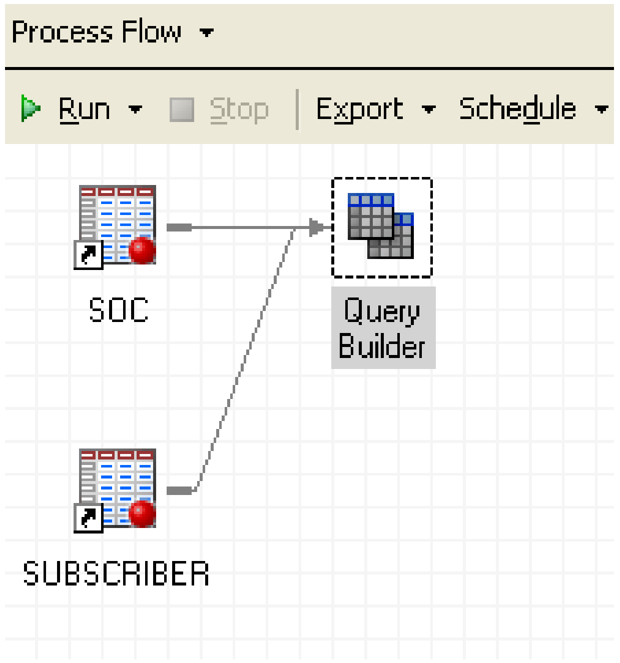
- how to be certain ?
    - explicit pass-through

Implicit Pass-Through – Ummmm

- Oracle DB
  - o two schemas BMLAPPO BMLREF

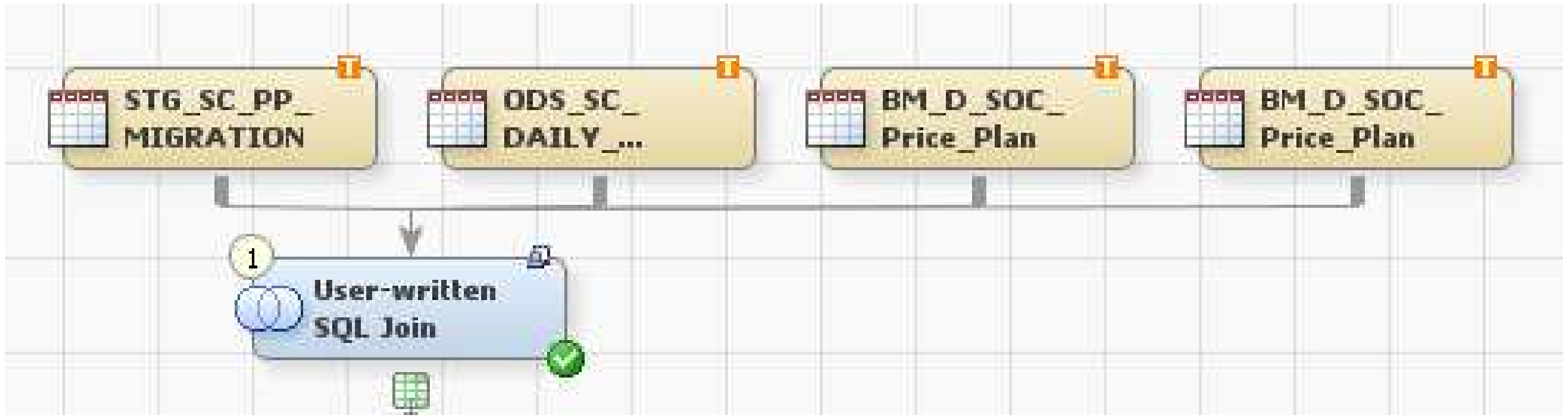
- SAS will happily run in Oracle
- turns SQL into Oracle query

- much happiness



Implicit Pass-Through – Ummmm

- Teradata DB
    - o two schemas MI_DATA MI_STAGE



- 15,000 rows left joined to 120,000,000
    - o 576 amps !??!
- 48 minutes later…

Implicit Pass-Through – NOT !

```
TERADATA_7: Prepared: on connection 6
SELECT * FROM mi_data."BM_D_SOC_Price_Plan“
```

```
ERROR: This SQL statement will not be
passed to the DBMS for processing because
it involves a join across librefs with
different connection properties.
```

```
TERADATA: trqacol- Casting decimals. Raw row size=38, Casted size=38, CAST_OVERHEAD_MAXPERCENT=20%

TERADATA_8: Prepared: on connection 4
SELECT "OLD_PP_SOC_CD","NEW_PP_SOC_CD",CAST("SERVICE_ACTIVITY_KEY" AS
FLOAT),"PERIOD_DT",CAST("SERVICE_ACTIVITY_KEY_PREV" AS FLOAT)
FROM mi_stage."STG_SC_PP_MIGRATION"
```

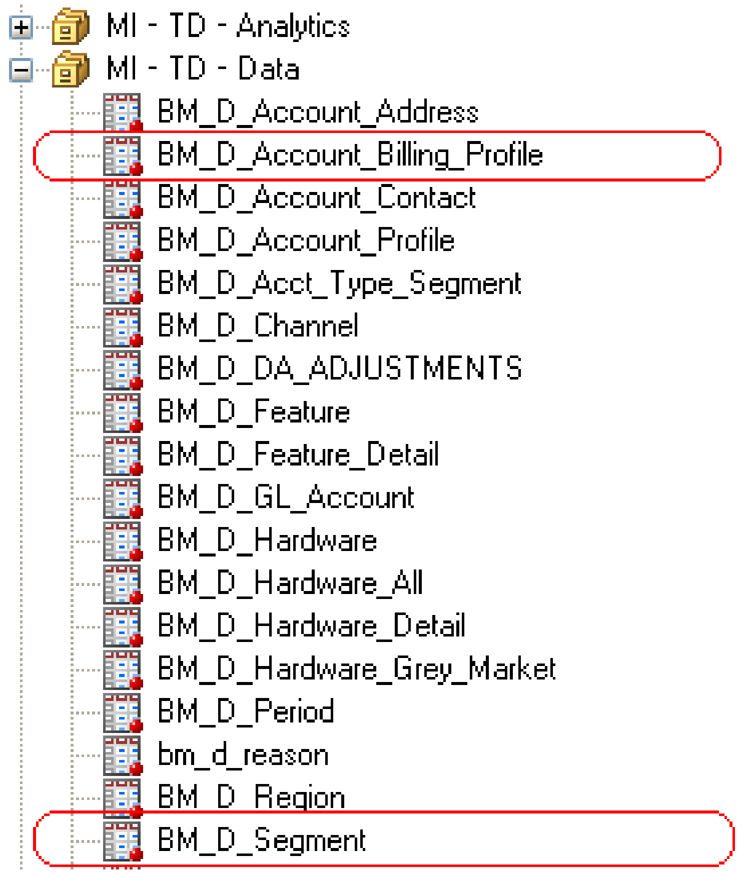
```
NOTE: PROCEDURE SQL used (Total process time):
      real time           48:12.89
      user cpu time       30:29.85
      system cpu time     6:40.33
```

Explicit Pass-Through – Joy

```
select * connection to teradata (
 select ..
  from MI_STAGE.ODS_SC_DAILY_ACTIVITY,
       MI_STAGE.STG_SC_PP_MIGRATION left join
       MI_DATA.BM_D_SOC_Price_Plan as BM_D_SOC_Price_Plan_prev
    on STG_SC_PP_MIGRATION.OLD_PP_SOC_CD =
        BM_D_SOC_Price_Plan_prev.PP_SOC_CD
 where STG_SC_PP_MIGRATION.SERVICE_ACTIVITY_KEY =
        ODS_SC_DAILY_ACTIVITY.SERVICE_ACTIVITY_KEY
   and STG_SC_PP_MIGRATION.PERIOD_DT =
        ODS_SC_DAILY_ACTIVITY.PERIOD_DT
   and ODS_SC_DAILY_ACTIVITY.ADJ_KEY = 0
                );
```

```
NOTE: PROCEDURE SQL used (Total process time):
      real time           4.66 seconds
      user cpu time       0.22 seconds
      system cpu time     0.04 seconds
```

EG – Who Does What Where ?

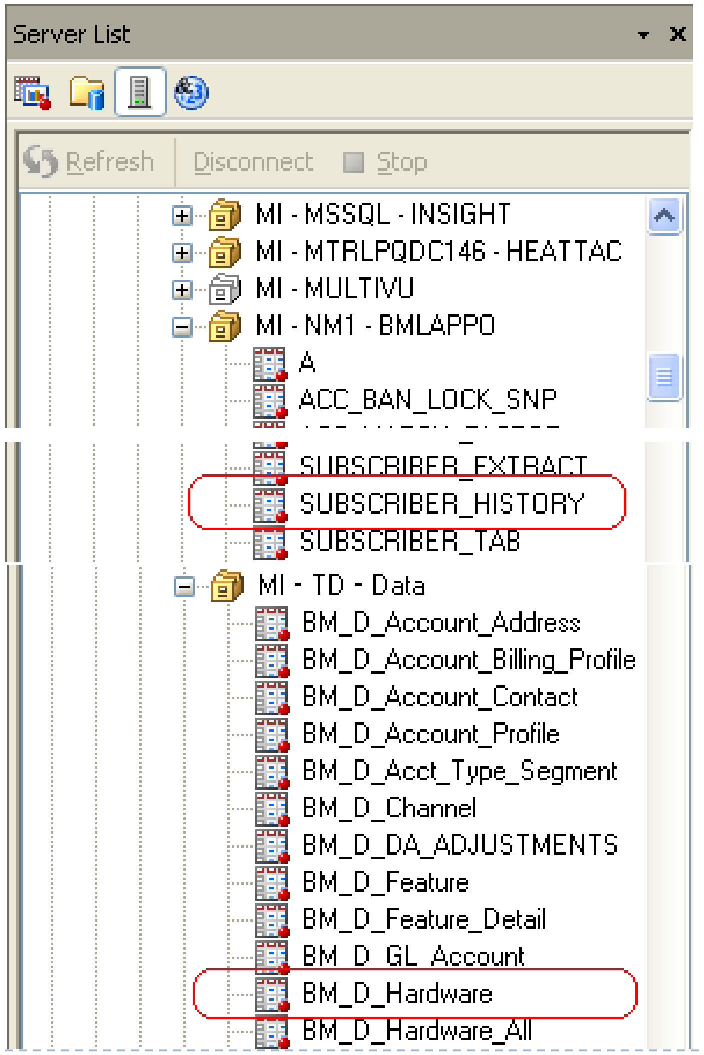


MI - TD - Analytics
MI - TD - Data
BM_D_Account_Address
BM_D_Account_Billing_Profile
BM_D_Account_Contact
BM_D_Account_Profile
BM_D_Acct_Type_Segment
BM_D_Channel
BM_D_DA_ADJUSTMENTS
BM_D_Feature
BM_D_Feature_Detail
BM_D_GL_Account
BM_D_Hardware
BM_D_Hardware_All
BM_D_Hardware_Detail
BM_D_Hardware_Grey_Market
BM_D_Period
bm_d_reason
BM_D_Region
BM_D_Segment

- same library
- same database

- candidate for pass-through

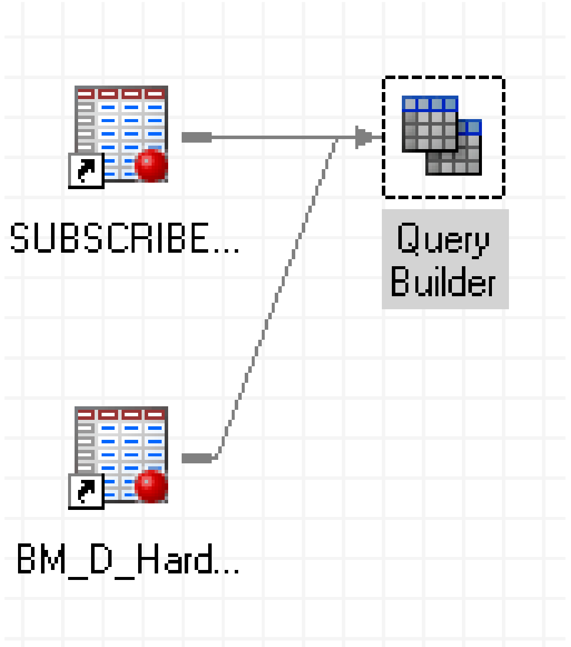
EG – Who Does What Where ?



Server List

Refresh Disconnect Stop

- MI - MSSQL - INSIGHT
- MI - MTRLPQDC146 - HEATTAC
- MI - MULTIVU
- MI - NM1 - BMLAPPO
  - A
  - ACC_BAN_LOCK_SNP
  - SUBSCRIBER_EXTRACT
  - SUBSCRIBER_HISTORY
  - SUBSCRIBER_TAB
- MI - TD - Data
  - BM_D_Account_Address
  - BM_D_Account_Billing_Profile
  - BM_D_Account_Contact
  - BM_D_Account_Profile
  - BM_D_Acct_Type_Segment
  - BM_D_Channel
  - BM_D_DA_ADJUSTMENTS
  - BM_D_Feature
  - BM_D_Feature_Detail
  - BM_D_GL_Account
  - BM_D_Hardware
  - BM_D_Hardware_All



- different libraries
- different databases
- millions of rows
- SY tomorrow !

Expensive Data Pulls

```
create table visa_bal as
     select * from connection to db2 (
       select acct_id, client_product_ds,
               current_balance_am
         from edw.visa_acct
        where effective_dt = '2014-03-31'
          and lifecycle_cd in ( 114,116,117 )
     );
```

- 5,000,000 rows come through the pipe to SAS

Expensive Data Pulls

```
proc summary data = visa_bal;
  class client_product_ds;
  var current_balance_am;
  output out = visa_bal_sum sum=;
run;
```

- 18 rows in summary data set

Efficient Data Pulls

```
select * from connection to db2 (
   select  client_product_ds,
           sum(current_balance_am)  as
                current_balance_am
     from  edw.visa_acct
    where  effective_dt = '2014-03-31'
      and  lifecycle_cd in ( 114,116,117 )
    group by client_product_ds
    order by client_product_ds      );
```

- let DB2 do the heavy lifting
  - query optimizer
- 18 rows through the pipe to SAS

Temporary Tables

- intermediate results
    - o subsetting exercise
    - o deep dive analysis

- need historical data for these 10,000 subscribers

1. Pull subscriber history down to SAS, join
2. where subscriber_no in (‘123’,’345’,’567’ x 10K)
3. Push 10,000 subscriber numbers up
    - o do the join in DB
    - o pull only what you need down
    - o minimize slow data transfers

Temporary Tables

- Oracle – “global temp” tables
- Teradata – volatile tables

- exist for the duration of the session
  - like SAS WORK datasets

- requires
  - LIBNAME engine
  - explicit pass-thru
    - EXECUTE (DDL statement) by teradata;
  - implicit pass-thru to load subset
  - explicit pass-thru for final result

Temporary Tables

```
libname td_volt teradata server="&server"
      user=&user password="&pass" dbmstemp=yes
      connection=global dbcommit=0;

proc sql;
      connect to teradata ( server="&server"
            user=&user password="&pass"
            connection=global mode=teradata );

      /* create an empty volatile table */
      execute ( create multiset volatile table
            test_vol ... ( .. columns .. )
                  ) by teradata;
quit;
```

Temporary Tables

```
/*
Put rows into the volatile table via the libname -
if lots of records, additional options necessary,
e.g. fastload
*/

proc sql;
 insert into td_volt.test_vol
   select subscriber_number as subscriber_no
     from midata.dim_subscriber ( obs = 100 )
 ;
quit;
```

Temporary Tables

```
/* Execute pass-thru query to join Teradata table
to volatile table */

proc sql;
  connect to teradata (server="&server" user=&user
            password="&pass" connection=global);

  create table sas_datastet_of_results as
   select * from connection to teradata (
    select s.*
      from  bmbi_view.vb_subscriber       s,
            test_vol                      v
      where v.subscriber_no = s.sub_no
                        );
quit;
```

Intermediate Tables

- use “WITH” to create temporary tables on the fly
  - exists only for duration of query

- allows division of tasks

- complex joins can be simplified
  - left, right, inner, full
  - how do I combine them ?!

- can lower query cost

Intermediate Tables

```
create table visa_bal_sum as
  select * from connection to db2 (
      with intermediate_acct as (
     select a.acct_id, a.acct_type,
            b.current_balance_am
       from edw.acct      a    left join
            edw.acct_bal b
         on a.acct_id = b.acct_id       )

        select a.*, c.cust_id
         from intermediate_acct a …
```

Conclusion

- do stuff where it makes sense
    - o use power of DB
    - o summarize, subset, sort in DB

o don’t move data unnecessarily

- not all implicit pass-thru is
    - o use option SASTRACE

- use EXPLAIN

Contact

Harry Droogendyk
harry@stratia.ca

Phone: 905-512-3827
Web: www.stratia.ca/papers