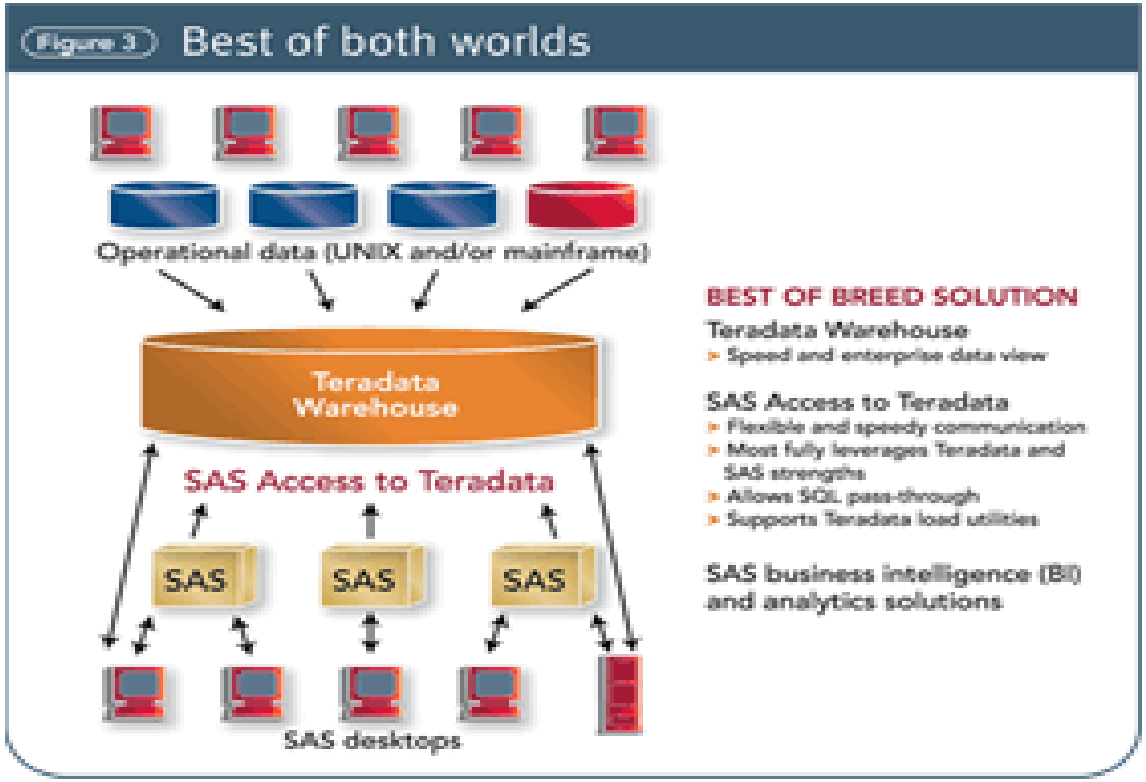
Teradata Can Do SAS Too !

Harry Droogendyk, Stratia Consulting Inc.
2014-05-29

Typical Architecture



Typical Architecture

why is PII dying?!

Principles

- processing data
  - do it where it makes sense
    - don't move data unnecessarily
      - detail vs summary
      - s t r e t c h your brain – learn new functionality
    - use the right tool / environment
      - strengths / weaknesses
  - do it once
  - do not store what you can easily compute
    - Unix space emails ?

Complications

- SQL is "set" oriented
  - remember Relational Algebra ?
  - data normalization
  - Employee set joined to Department set
    - yields "result set"
  - requires abstraction

- we (and SAS) are "row oriented"
  - read a row, deal with it, read another row
  - 1st row ? Last row ?
  - columns ? eg. array structures

Complications

- data step is so friendly
  - granular control
  - explicit control
  - comfort zone
  - bits and bytes

- SQL can be nebulous
  - “sets” are uncomfortable
  - uncontrollable
  - but, preferred by the DB
  - why a cartesian product join ?!
  - why are results sometimes different ?! eg. ORDER BY

Agenda

- Teradata can do SAS !!!
  - don't move the data !!
- start simple – all in Teradata
  - sorting correctly for SAS
  - conditional processing
  - useful intermediate tables
- OLAP / Analytical functions
  - first.
  - lag & lead
  - cumulative / specialized sums

Sorting

```
create table tables as
   select * from connection to teradata (
       select databasename, tablename
         from dbc.tables
        where databasename in ( 'DDWV01','DDWV04I' )
        order by tablename );
```

- how many tables are in both views?
  - could use SQL EXCEPT
  - using data step MERGE

Sorting

```
data idm;
  merge tables ( where = ( databasename = 'DDWV01' )
                    in = ddwv01  )
        tables ( where = ( databasename = 'DDWV04I' )
                    in = ddwv04i ) ;
      by tablename;
  flag = ddwv01 + ddwv04i * 2 ;
run;
```

```
ERROR: BY variables are not properly sorted on data set
  WORK.TABLES.
ddwv01=1 ddwv04i=0 DatabaseName=DDWV01 TableName=tln_loans
  FIRST.TableName=1 LAST.TableName=1 flag=. _ERROR_=1 _N_=2199
NOTE: The SAS System stopped processing this step because of
  errors.
```

Sorting

VIEWTABLE: Rmtwork.Tables

	DatabaseName	TableName
2180	DDWV01	TELECOM_DLY
2181	DDWV01	TELECOM_DLY_DELTA
2182	DDWV01	TEL_BKG_PYMNT
2183	DDWV01	TEL_BKG_TXN
2184	DDWV01	TFC_MSG_DTL
2185	DDWV01	tln_loans
2186	DDWV01	TLN_LOANS_HIST
2187	DDWV01	TM_ZNE

- ASCII collation ?
- Teradata ignores case by default
- data step MERGE / BY does not
  - add a PROC SORT step ?!

Sorting

```
order by tablename (casespecific)
```

2245	DDWV01	WEB_REQ_RCBASE
2246	DDWV01	WIRE_PYMT_ADT
2247	DDWV01	WLTH_SRC_DLY
2248	DDWV01	WLTH_SRC_DLY_DELTA
2249	DDWV01	WLTH_SRC_TYP
2250	DDWV01	tln_loans

- mode=ANSI / Teradata – no effect
- MERGE / BY now happy

```
                              Cumulative
flag     Frequency            Frequency
-------------------------------------
 1          1351                 1351
 2           891                 2242
 3             4                 2246
```

Conditional Logic

- call data
  - queue_cd determines call centre category
    - 'HA', 'H' Home / Auto
    - 'CR', 'C' Creditor
    - 'GN','GEN' General line
  - records call metrics
    - inbound / outbound calls
    - offered, abandoned
- need wide table of call data by date
  - transpose by queue
- data step "IF" or "SELECT" ?
  - nooooooo

Conditional Logic

```
select c.call_dt, c.operator_id, n.operator_nm
   ,   sum(case when queue_cd in ( 'HA', 'H' )
                then ib_cnt
                else 0        end)   as ha_ib_cnt
```

```
, sum(case when queue_cd in ('CR','C')          then ib_cnt else 0 end) as cr_ib_cnt
, sum(case when queue_cd in ('GN','GEN')        then ib_cnt else 0 end) as gen_ib_cnt
, sum(case when queue_cd in ('HA','H')          then ob_cnt else 0 end) as ha_ob_cnt
, sum(case when queue_cd in ('CR','C')          then ob_cnt else 0 end) as cr_ob_cnt
, sum(case when queue_cd in ('GN','GEN')        then ob_cnt else 0 end) as gen_ob_cnt
        ...
 from  db.call_data c
        …
group  by 1,2,3
```

Conditional Logic

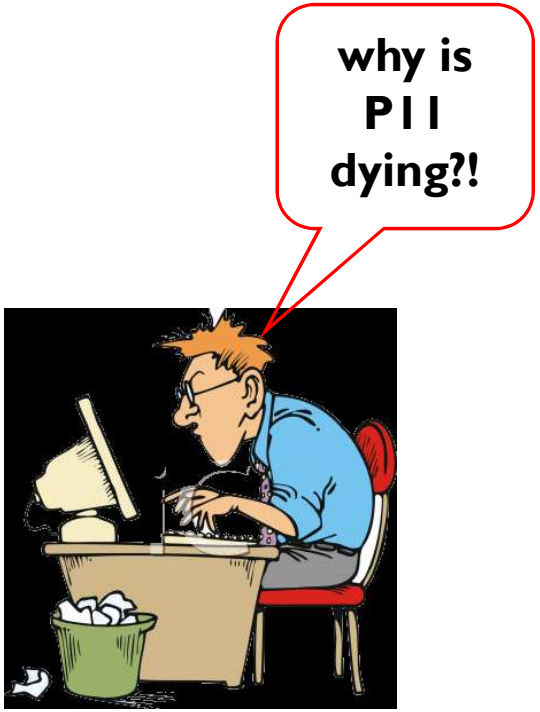
- CASE statements can be used anywhere
  - SELECT
    - within functions, e.g. SUM ( )
  - WHERE
  - HAVING
  - etc…
- functions like data step SELECT / WHEN
- granular control

WORK tables

- intermediate tables can be helpful
  - reduce query complexity
    - outer joins - left, right, full, inner
  - verify intermediate results
- SAS WORK lib
  - no need to define, allocate space, cleanup
- Teradata temporary tables
  - disconnect – gonzo

WORK tables

- need data for 2,000 accounts supplied in Excel
  - import
  - create macro variable of IDs using SQL into:
  - acct_no in (&mother_of_all_macro_vars)
    - > 64K bytes ?
    - IN () performance ?
- pull entire account table down to SAS
  - subset
- use volatile table
  - define Teradata libname
  - proc append
  - pass-thru inner-join query



Teradata Temporary Tables

- CREATE GLOBAL TEMPORARY TABLE
  - cannot be created WITH DATA
  - CREATE, subsequent INSERT
  - toooo lazy to get column definitions

- CREATE VOLATILE TABLE
  - create WITH DATA
  - define a PRIMARY INDEX
  - COLLECT STATS

Teradata Temporary Tables

```
connect to teradata ( &password database = ddwv04i
                      mode=teradata connection=global);

execute(
   create volatile table travel_case_active as (
     select distinct r.clm_bnft_cse_id
       from ddwv04i.ins_clm_fncl_evnt                e,
            ddwv04i.ins_clm_bnft_cse_evnt_reltn      r

      where e.evnt_sys_src_id          = 75
            <snip>
     group by r.clm_bnft_cse_id
        )  with data primary index ( clm_bnft_cse_id )

              on commit preserve rows ) by teradata;
```

Teradata Temporary Tables

- temporary table had a primary index
  - align indexes as much as possible – think AMPS
    - show view *teradata_view.view_name;* → get table name
    - show table *teradata_db.table_name;* → get PI

```
with data primary index ( clm_bnft_cse_id )
```

- assist optimizer in formulating query plan

```
execute (
  collect statistics
   column clm_bnft_cse_id
       on travel_case_active
   ) by teradata;
```

Teradata Temporary Tables

- volatile tables are …. temporary !

```
disconnect from teradata;
quit;
```

→ say b'bye

- to persist across SQL Connections

```
libname tdtemp teradata &password
        database = ddwv04i mode=teradata
        connection=global dbmstemp=yes ;
```

... multiple SQL connects / disconnects / quits ...

```
libname tdtemp clear;
```

→ say b'bye

Teradata Temporary Tables

- Teradata v14

```
execute ( create table claim_master as (
    with claim_status_int as (
          select i_ocactvty_changes
           ,     max(i)              as i_x
            from  fineos..vocstagechange
           group by 1
                 )
          select a.*
            from fineos..voccase  a,
                 claim_status_int b
            where  a.i = b.i_x
          ) by netezza;      * TD v14 ;
```

first.*by_var*

```
proc sort data = sashelp.class
          out = class;
           by sex age;
run;

data unique_class;
  set class;
   by sex;
  if first.sex;
run;
```

```
19 observations read from WORK.CLASS.
WORK.UNIQUE_CLASS has 2 observations
```

first.*by_var*

- requires use of special database functions
  - Windowing
  - OLAP or Analytical

- QUALIFY
  - limits result set, analogous to HAVING
- OVER
  - defines grouping criteria
- PARTITION
  - similar to GROUP BY
- ORDER BY
  - sorts result set *before* QUALIFY is applied

first.*by_var*

- move SAS data to Teradata volatile table
  - remember the 2,000 account Excel file ?

- no Teradata fastload options available for volatile tables ☹

```
libname tdtemp teradata &password database = ddwv04i
            mode=teradata
            connection=global dbmstemp=yes ;

proc append       base = tdtemp.sashelp_class
                  data = sashelp.class;
run;

%drop_td_table(lib=tdtemp, table=unique_class);
```

first.*by_var*

```
execute ( create volatile table unique_class as (
     select * from sashelp_class

       qualify row_number() over
            ( partition by sex
                 order by age desc, name) = 1

            ) with data  on commit preserve rows
     ) by teradata;
```

- QUALIFY — row_number = 1
- OVER — defines grouping criteria
  - PARTITION
    - sex — first.sex, specified row_number = 1
  - ORDER BY descending age and name
    - SAS default is EQUALS
    - Teradata parallel processing
      - **explicitly define order**

first.*by_var* - results

first. - SAS

Obs	Name	Sex	Age	Height	Weight
1	Janet	F	15	62.5	112.5
2	Philip	M	16	72.0	150.0

first. - Teradata

Obs	Name	Sex	Age	Height	Weight
1	Janet	F	15	62.5	112.5
2	Philip	M	16	72.0	150.0

- **could we use RANK instead of ROW_NUMBER ?**
- **what if we want last.sex ?**

%drop_td_table macro

```
%macro drop_td_table(lib=, table=);

	%if %sysfunc(exist(&lib..&table)) %then %do;

		proc sql;
			drop table &lib..&table;
		quit;

	%end;

%mend drop_td_table;
```

Lag / Lead Functionality

- SAS has LAG() function
  - found in some databases as well

- Analyical / OLAP functionality
  - MIN / MAX / AVG with OVER
  - rows between 1 following and 1 following - lead
  - rows between 1 preceding and 1 preceding - lag
  - rows unbounded preceding - all before
  - etc…

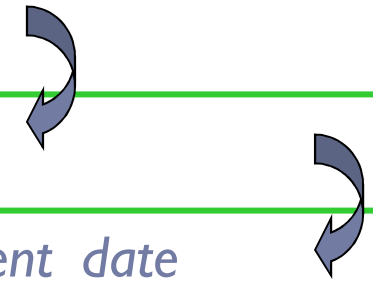
- PERIOD data type
  - range of date values
  - use EXPAND ON to generate rows

Lag / Lead Functionality

```
period ( captr_dt,
  coalesce (min (captr_dt)
              over ( partition by ip_rol_id, alt_no
                           order by captr_dt
        /*
            return the next row, i.e. next highest value of
            captr_dt, if there isn't a next row, return the
            Teradata current_date value
        */
              rows  between 1 following and 1 following
              ), current_date
            )
        ) as period_dt
```

Lag / Lead Functionality

Partition by	Captr_Dt
1	2014-03-01
1	2014-03-12
	Teradata *current_date*
2	2014-05-02
2	2014-05-11
2	2014-05-29
	Teradata *current_date*



PERIOD data type – EXPAND ON

```
/*
  Now that we have the period() data value, create a row
  for each date between the beginning / ending date value
  in the period_dt field.
  We're only interested in dates that have a range from
  the MOR Start date - 6 months to the current IDM snap dt.
*/
expand on period_dt as captr_dt2
  by interval '1' day
  for period ( cast ('2011-04-30' as date),
               cast( %single(&idm_snap_dt) as date) )
……….
select begin(captr_dt2) as captr_dt
```

Conditional Logic Anywhere

- PERIOD requires begin date > end date
- data isn't always pretty
  - but you can do conditional logic in SQL to deal with it

```
period( coalesce(req_received_dt,req_created_dt),
   case when app_sts_cd = 'CLOSED'
         and app_sts_dt > coalesce(req_received_dt,req_created_dt)
        and ( req_closed_dt is null or app_sts_dt < req_closed_dt )
               then app_sts_dt
   when req_closed_dt is null
     or req_closed_dt >= cast ( %single(&idm_snap_dt) as date )
               then cast ( %single(&idm_snap_dt) as date )
               else req_closed_dt end
  ) as period_dt
```

Summing Data

- SAS is easy going

```
create table class_sum as
    select name, sex, age, weight, height,
           sum(weight) as wgt_sum
      from sashelp.class
      group by sex;
```

```
NOTE: The query requires remerging summary
statistics back with the original data
```

	Name	Sex	Age	Weight	Height	wgt_sum
9	Janet	F	15	112.5	62.5	811
10	Philip	M	16	150	72	1089.5

Summing Data

- databases are not so tolerant

```
ERROR: Teradata prepare: Selected non-aggregate
values must be part of the associated group
```

- cumulative sums
  - using OLAP / Analytical functions
  - using CSUM
- other similar functions available, "moving"
  - MAVG
  - MDIFF
  - MSUM

Summing Data

- cumulative claim reserves
  - adjuster sets reserve at claim open
  - reserve transactions occur as time goes on
    - increase if new costs come to light
    - decrease as payments are made, or severity lessens

- outstanding reserves are a liability
  - need to know outstanding reserves by day

1. calculate cumulative reserves
2. generate daily reserve totals

Summing Data

Claim No	Trans Dt	Reserve Amt	Pymt Amt	Note	Accum Reserve
1	2014-04-10	+500		Open	500
1	2014-04-12	-350	350	Payment	150
1	2014-04-13	+600		Adding	750
2	2014-04-09	+1,200		Open	1,200
2	2014-04-11	-800	800	Payment	400
2	2014-04-12	-400		Close	0

outstanding reserves on Apr 11?

Summing Data - ANSI

```
select clm_bnft_cse_id, event_dt

      ,sum ( evnt_amt )
        over ( partition by clm_bnft_cse_id
                     order by event_dt
               rows unbounded preceding )
                                as os_reserve_amt

  from travel_reserves;
```

- what's missing ?

- partition by – reset sum on claim case ID change
- order by – regulates order of rows into sum
- rows … – include this row and all rows before it

Summing Data - Teradata

```
select clm_bnft_cse_id
     , event_dt
     , csum(evnt_amt, event_dt) as os_reserve_amt

  from travel_reserves
 group by clm_bnft_cse_id
```

- what's missing ?
- CSUM – Teradata only, not ANSI
  - evnt_amt – summed column
  - event_dt – sort column(s)
- GROUP BY – specifies "reset" column(s)

- GROUP BY is equivalent to PARTITION BY in previous query

Generate Daily Reserve Rows

```
select clm_bnft_cse_id
     , os_reserve_amt
     , period(event_dt,

     coalesce( min(event_dt)
       over ( partition by clm_bnft_cse_id
                    order by event_dt
            rows between 1 following and 1 following
             ), current_date ))  as period_dt
  from travel_cum_os_reserves
```

- create PERIOD with adjacent rows by EVENT_DT

Generate Daily Reserve Rows

```
select clm_bnft_cse_id
     ,  begin(event_dt2)        as event_dt
     ,  os_reserve_amt
     from (  select …
                    coalesce( … ))  as period_dt
           )
expand on period_dt as event_dt2
  by interval '1' day
  for period ( cast(%single(&MORStartDate) as date),
               cast(%single(&idm_snap_dt) as date) )
```

- voila, daily outstanding reserves

Conclusion

- do stuff where it makes sense
  - use Teradata's power
  - summarize, subset, sort in DB
  - don't move data unnecessarily
  - rarely, if ever, move detail data
- be concerned with efficiency
  - coding, execution & storage
- be inquisitive
  - new releases bring new functionality
  - exploit the strengths of your tools

Conclusion

- web resources

http://teradatafaqs.blogspot.ca/

http://teradata.weizheng.net/

http://developer.teradata.com/

Dieter Noeth

http://stackoverflow.com/users/25279 05/dnoeth

0 Questions

This user has not asked any questions

Contact

Harry Droogendyk
harry@stratia.ca

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