Verifying TPC-DS Results
TPC-DS?

- “Decision Support”
- VLDB ’06, Othayoth & Pöss: The Making of TPC-DS
  - “TPC-H is not representative”
- *One* official result, 2018
  - “Transwarp Data Hub v5.1” ?!
  - 12 years later? Why?
TPC-DS Design

• Multiple snowflake schemas with *shared dimensions*
  • 24 tables with an average of 18 columns
  • More representative skewed database content + NULLs
  • Sub-linear scaling of non-fact tables
• Ad-hoc, reporting, iterative and extraction queries
  • 99 103 distinct SQL 99 queries with random substitutions *(ROLLUP & PARTITION)*
    • Substitutes also aggr funcs *and column names*
• ETL-like data maintenance (ignored)
with customer_total_return as
(select sr_customer_sk as ctr_customer_sk,
sr_store_sk as ctr_store_sk,
sum([AGG_FIELD]) as ctr_total_return
from store_returns,
date_dim
where sr_returned_date_sk = d_date_sk
and d_year = [YEAR]
group by sr_customer_sk,
sr_store_sk)
select c_customer_id
from customer_total_return ctr1,
store,
customer
where ctr1.ctr_total_return > (select avg(ctr_total_return)*1.2
from customer_total_return ctr2
where ctr1.ctr_store_sk = ctr2.ctr_store_sk)
and s_store_sk = ctr1.ctr_store_sk
and s_state = '[STATE]'
and ctr1.ctr_customer_sk = c_customer_sk
order by c_customer_id
LIMIT 100;
with year_total as 
(select c_customer_id customer_id,
c_first_name customer_first_name,
c_last_name customer_last_name,
c_preferred_cust_flag customer_preferred_cust_flag,
c_birth_country customer_birth_country,
c_login customer_login,
c_email_address customer_email_address,
d_year dyear,
sum(((ss_ext_list_price-ss_ext_wholesale_cost-ss_ext_discount_amt)+ss_ext_sales_price)/2) year_total,
's' sale_type
from customer,
store_sales,
date_dim
where c_customer_sk = ss_customer_sk
and ss_sold_date_sk = d_date_sk
group by c_customer_id,
c_first_name,
c_last_name,
c_preferred_cust_flag,
c_birth_country,
c_login,
c_email_address,
d_year
union all
select c_customer_id customer_id,
c_first_name customer_first_name,
c_last_name customer_last_name,
c_preferred_cust_flag customer_preferred_cust_flag,
c_birth_country customer_birth_country,
c_login customer_login,
c_email_address customer_email_address,
d_year dyear,
sum((((cs_ext_list_price-cs_ext_wholesale_cost-cs_ext_discount_amt)+cs_ext_sales_price)/2) ) year_total,
'c' sale_type
from customer,
catalog_sales,
date_dim
where c_customer_sk = cs_bill_customer_sk
and cs_sold_date_sk = d_date_sk
group by c_customer_id,
c_first_name,
c_last_name,
c_preferred_cust_flag,
c_birth_country,
c_login,
c_email_address,
d_year
union all
select c_customer_id customer_id,
c_first_name customer_first_name,
c_last_name customer_last_name,
c_preferred_cust_flag customer_preferred_cust_flag,
c_birth_country customer_birth_country,
c_login customer_login,
c_email_address customer_email_address,
d_year dyear,
sum((((ws_ext_list_price-ws_ext_wholesale_cost-ws_ext_discount_amt)+ws_ext_sales_price)/2) ) year_total,
'w' sale_type
from customer,
web_sales,
date_dim
where c_customer_sk = ws_bill_customer_sk
and ws_sold_date_sk = d_date_sk
group by c_customer_id,
c_first_name,
c_last_name,
c_preferred_cust_flag,
c_birth_country,
c_login,
c_email_address,
d_year
)
select
t_s_secyear.customer_id,
t_s_secyear.customer_first_name,
t_s_secyear.customer_last_name,
from year_total t_s_firstyear,
year_total t_s_secyear,
year_total t_c_firstyear,
year_total t_c_secyear,
year_total t_w_firstyear,
year_total t_w_secyear
where t_s_secyear.customer_id = t_s_firstyear.customer_id
and t_s_firstyear.customer_id = t_c_secyear.customer_id
and t_s_firstyear.customer_id = t_c_firstyear.customer_id
and t_s_firstyear.customer_id = t_w_firstyear.customer_id
and t_s_firstyear.customer_id = t_w_secyear.customer_id
and t_s_firstyear.sale_type = 's'
and t_c_firstyear.sale_type = 'c'
and t_w_firstyear.sale_type = 'w'
and t_s_secyear.sale_type = 's'
and t_c_secyear.sale_type = 'c'
and t_w_secyear.sale_type = 'w'
and t_s_firstyear.dyear = [YEAR]
and t_s_secyear.dyear = [YEAR]+1
and t_c_firstyear.dyear = [YEAR]
and t_c_secyear.dyear = [YEAR]+1
and t_w_firstyear.dyear = [YEAR]
and t_w_secyear.dyear = [YEAR]+1
and t_s_firstyear.year_total > 0
and t_c_firstyear.year_total > 0
and t_w_firstyear.year_total > 0
and case when t_c_firstyear.year_total > 0 then t_c_secyear.year_total / t_c_firstyear.year_total else null end
> case when t_s_firstyear.year_total > 0 then t_s_secyear.year_total / t_s_firstyear.year_total else null end
and case when t_c_firstyear.year_total > 0 then t_c_secyear.year_total / t_c_firstyear.year_total else null end
and case when t_s_firstyear.year_total > 0 then t_s_secyear.year_total / t_s_firstyear.year_total else null end
order by t_s_secyear.customer_id,
t_s_secyear.customer_first_name
limit 100;
“Qualification”

- Reference results “provided” for SF 1
  - *Completely borked* format, every single one had different encoding (grrr)
- NULL first/last difference, hence 2 sets of results for many queries
- Query placeholder values only in spec PDF (grrr)
B.1  query1.tpl

Find customers who have returned items more than 20% more often than the average customer returns for a store in a given state for a given year.

Qualification Substitution Parameters:

- YEAR.01=2000
- STATE.01=TN
- AGG_FIELD.01 = SR_RETURN_AMT
SQL> Connected.
SQL> SQL> SQL> SQL>

<table>
<thead>
<tr>
<th>C_LAST_NAME</th>
<th>C_FIRST_NAME</th>
<th>C_SALUTATI</th>
<th>SS_TICKET_NUMBER</th>
<th>CNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abbott</td>
<td>Harriet</td>
<td>Ms.</td>
<td>N</td>
<td>15</td>
</tr>
<tr>
<td>Acosta</td>
<td>Victor</td>
<td>Mr.</td>
<td>N</td>
<td>15</td>
</tr>
<tr>
<td>Adams</td>
<td>Jerry</td>
<td>Sir</td>
<td>N</td>
<td>15</td>
</tr>
<tr>
<td>Woody</td>
<td>Philip</td>
<td>Mr.</td>
<td>N</td>
<td>16</td>
</tr>
<tr>
<td>Wright</td>
<td>Amanda</td>
<td>Y</td>
<td>105395</td>
<td>15</td>
</tr>
<tr>
<td>Yates</td>
<td>Kimberly</td>
<td>Ms.</td>
<td>Y</td>
<td>15</td>
</tr>
<tr>
<td>Grace</td>
<td></td>
<td></td>
<td>147095</td>
<td>16</td>
</tr>
<tr>
<td>Joshua</td>
<td>Sir</td>
<td>N</td>
<td>108357</td>
<td>15</td>
</tr>
<tr>
<td>Judy</td>
<td>Dr.</td>
<td></td>
<td>234350</td>
<td>16</td>
</tr>
<tr>
<td>Roy</td>
<td></td>
<td></td>
<td>200033</td>
<td>15</td>
</tr>
<tr>
<td>Mr.</td>
<td>Y</td>
<td></td>
<td>214003</td>
<td>15</td>
</tr>
<tr>
<td>Ms.</td>
<td></td>
<td></td>
<td>54536</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>148695</td>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>151297</td>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td></td>
<td>16278</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td></td>
<td>146557</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td></td>
<td>55640</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td></td>
<td>58614</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td></td>
<td>96514</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td></td>
<td>220211</td>
<td>15</td>
</tr>
</tbody>
</table>

451 rows selected.

Elapsed: 00:00:03.36
SQL> Disconnected from Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit Production
With the Partitioning, Real Application Clusters, Automatic Storage Management, OLAP,
Advanced Analytics and Real Application Testing options
Challenge: Reproduce Results

- What are the correct answers to the qualification queries on SF1?
- Very useful for testing!
  - check correctness before bragging about performance
- Steps:
  - Clean up reference results (manually)
  - Create tool to check reference results (numeric diffs)
  - Create qualification queries from PDF (manually)
  - Run queries on bunch of systems to see whether they agree
Systems tested

- Postgres 11.0
- HyPer 20182.18.1009.2120
- SQL Server on Docker
- MonetDB 11.31.11
- SQLite 3.24.0
  - (dropped because could not run enough queries)
Results

- Reference results for Queries 21, 34, 66, 71, 73, 79 and 98 were wrong (missing/additional rows, values, spacing)
  - Determined by consensus of 2 or more systems
- Queries 78 and 83 are probably broken (int div/rounding)
- Postgres/SQL server can run all queries
- HyPer misses ROLLUP, thus failing ~ 10 queries
- MonetDB misses ROLLUP and PARTITION, thus failing ~ 20 queries
  - Analytics branch can’t do PARTITION queries yet (checked 2018-10-29)
- All but three queries have reproduced results
  - Need additional systems to double-check reference results
https://github.com/cwida/tpcds-kit

- Cleaned and checked reference (results NULL first+last)
- Qualification queries
- Data generator
  - Copy of SF1 at https://homepages.cwi.nl/~hannes/tpcds-sf1-xz/
- Load scripts for various DBs
- Result diff tool (1% numeric drift and spaces allowed)
- Plotting script (R)
- Some more cleanup and documentation to come
Next steps

- Cleanup repo and document
- Verify last three inconsistent results
  - Which system? Oracle? Vectorwise? Redshift
- Write paper for TPCTC?
- Actually run a performance comparison?
- Create test cases for systems / file bug reports
- Questions?