The MAGIC of AGGREGATES

Michael Wilcke, sumIT AG
AGENDA

- introduction to aggregates
  - logical constraints
- identifying useful aggregates
  - useful?
- building aggregates
  - ETL or script
- using aggregates
  - prerequisites => level counts
- refreshing aggregates
  - fast & safe
- lifecycle considerations
  - which changes change what?
### AGGREGATES

<table>
<thead>
<tr>
<th>PRO</th>
<th>CON</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FAST QUERY RESPONSE</strong></td>
<td>risk =&gt; fast but wrong</td>
</tr>
<tr>
<td></td>
<td>effort to identify</td>
</tr>
<tr>
<td></td>
<td>effort to build/refresh</td>
</tr>
<tr>
<td></td>
<td>effort to use</td>
</tr>
<tr>
<td></td>
<td>time to build/refresh</td>
</tr>
<tr>
<td></td>
<td>effort to maintain</td>
</tr>
<tr>
<td></td>
<td>less agile</td>
</tr>
</tbody>
</table>
Intro

AGGREGATED DATA

- Sales (B, HY): 3*11 = 33
- Sales (PT, M): 11*63 = 693
- Sales: 10'000'000

Time:
- Year: 6
- Half-Year: 11
- Quarter: 21
- Month: 63
- Day: 1673

Product:
- Brand: 3
- LOB: 6
- Product Type: 11
- Product: 20

Sales (B, HY) and Sales (PT, M) are derived from the product categories.
SIMPLE AGG RULES

Intro

data

1  5  5
**Intro**

**SIMPLE AGG RULES**

- **DETAIL**
  - higher
  - agg data
  - data
  - $\text{sum}(1,5,5)$
  - $\text{sum}(1,5)$
  - $\text{sum}(5)$
  - $\text{sum}(6,5)$

- **AGGREGATE**
  - $11 = 11$

- $6$
- $5$

---

Sunday, 19 May 13
Intro

SIMPLE AGG RULES

- $\text{sum(data)} = \text{sum( sum(data) )}$
- $\text{min(data)} = \text{min( min(data) )}$
- $\text{max(data)} = \text{max( max(data) )}$

easy and safe to use
COUNT

DETAIL

agg data

count(1,5)

higher

count(1,5,5)

count(5)

data

3 = 3

AGGREGATE

sum(2,1)

1

5

5

2

1
count(data) = sum( count(data) )

BI Server uses SUM on higher levels
AVERAGE

DETAIL

AGGREGATE

avg(1,5,5)
detail

agg data

higher

data

avg(1,5)

avg(5)

1 5 5

avg(3,5)

3 5

3.6 <> 4
BI Server uses aggregates ONLY on their EXACT grain and requires logical level keys in query.

\[
\text{avg(data)} \not\equiv \text{avg( avg(data) )}
\]

\[
\frac{\text{sum(data)}}{\text{count(data)}} = \frac{\text{sum( sum(data) )}}{\text{sum( count(data) )}}
\]
Intro

OTHER AGG RULES

DETAIL

cntdct(1,5,5)

cntdct(1,5)
cntdct(5)

AGGREGATE

sum(2,1)

higher

agg data

data

2 <-> 3

1 5 5

2 1

Sunday, 19 May 13
BI Server uses aggregates ONLY on their EXACT grain and requires logical level keys in query.

\[
\text{cntdct(data) <> sum( cntdct(data) )}
\]

\[
\text{median(data) <> median( median(data) )}
\]
SUM-LAST

Logical Column - 9- Full Time Empl (End Period)

<table>
<thead>
<tr>
<th>Logical Dimension</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Other</td>
<td>SUM(&quot;01: Sample App&quot;.&quot;F4 Headcount Base Measures&quot;.&quot;9- Full Time Empl (End Period)&quot;)</td>
</tr>
<tr>
<td>H0 Time</td>
<td>LAST(&quot;01: Sample App&quot;.&quot;F4 Headcount Base Measures&quot;.&quot;9- Full Time Empl (End Period)&quot;)</td>
</tr>
</tbody>
</table>

- **year**
- **month**
- **day**
SUM-LAST (2)

**DETAIL**

- `sumlst(1,5,5)`
- `sumlst(1,5)`
- `sumlst(5)`
- `1`
- `5`
- `5`

**AGGREGATE**

- `sumlst(5,5)`
- `5`
- `5`

**Intro**

`higher`

`agg data`

`data`

`5=5`
sum(lst(data)) = sum(lst(sum(lst(data))))

BI Server uses aggregates ONLY on their EXACT grain and requires logical level keys in query.

Use fact alias join to last day of period

=> eliminates LAST, leaves simple SUM

=> represents an agg on period grain

requires a fact alias for every higher level
Intro

EXAMPLE

THEN

#Employees (year)

year

IF

#Employees (month)

month

day
Identify

BRAIN

ETL

Facts & their grain

Script

Domain Knowledge

Usage Tracking

Web Catalog

BI-Model

Calculator

get value from data
RE-MODEL

Identify

Sunday, 19 May 13
Identify

create an “artificial” level when dim is flat
=> logical level key has to be mapped physically
avoid NULL values
Identify

RE-MODEL (2)

levels from alternate hierarchies prevent aggregates from being used
re-model into separate dimensions
Which AGGREGATES would accelerate most considering the USAGE of the BI-Model in a given TIMEFRAME?

- Script
- Usage Tracking
- BI-Model

*only available on Exalytics*
<table>
<thead>
<tr>
<th><strong>Brain</strong></th>
<th><strong>Summary Advisor</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>just do it</td>
<td>buy Exalytics</td>
</tr>
<tr>
<td>takes time</td>
<td>runs in minutes</td>
</tr>
<tr>
<td>identifies useful aggregates to build</td>
<td>identifies useful aggregates to build</td>
</tr>
<tr>
<td>identifies useless aggregates to remove</td>
<td>never recommends aggregates to remove</td>
</tr>
<tr>
<td>may change web-cat / model</td>
<td>will never do so</td>
</tr>
<tr>
<td>re-thinks with or without aggregates</td>
<td>re-runs with aggregates</td>
</tr>
<tr>
<td>knows politics</td>
<td>doesn’t know politics</td>
</tr>
</tbody>
</table>
1. develop ETL & build
2. map into BI model
OPTION: SCRIPT

1. create a script
2. run the script

Oracle BI Server

DWH

aggregate builder

Interactive Dashboards
Answers
BI Publisher
## COMPARISON

<table>
<thead>
<tr>
<th>ETL</th>
<th>Hybrid</th>
<th>Script</th>
</tr>
</thead>
<tbody>
<tr>
<td>optimal build performance</td>
<td><img src="image.png" alt="Green Circle" /></td>
<td>bad build performance</td>
</tr>
<tr>
<td>incremental refresh</td>
<td><img src="image.png" alt="Green Circle" /></td>
<td>full refresh only</td>
</tr>
<tr>
<td>optimal query performance</td>
<td><img src="image.png" alt="Green Circle" /></td>
<td>good query performance</td>
</tr>
<tr>
<td>high development effort</td>
<td><img src="image.png" alt="Green Circle" /></td>
<td>low development effort</td>
</tr>
<tr>
<td>use of data model (only)</td>
<td><img src="image.png" alt="Green Circle" /></td>
<td>use of BI model</td>
</tr>
<tr>
<td>manual map to BI</td>
<td><img src="image.png" alt="Green Circle" /></td>
<td>automatic map to BI</td>
</tr>
<tr>
<td>high maintenance effort</td>
<td><img src="image.png" alt="Green Circle" /></td>
<td>low maintenance effort</td>
</tr>
<tr>
<td>only physical changes to be handled</td>
<td><img src="image.png" alt="Green Circle" /></td>
<td>also logical changes to be handled</td>
</tr>
</tbody>
</table>
## COMPARISON

<table>
<thead>
<tr>
<th>ETL</th>
<th>Hybrid</th>
<th>Script</th>
</tr>
</thead>
<tbody>
<tr>
<td>optimal build performance</td>
<td>bad build performance</td>
<td></td>
</tr>
<tr>
<td>incremental refresh</td>
<td></td>
<td>full refresh only</td>
</tr>
<tr>
<td>high development effort</td>
<td></td>
<td></td>
</tr>
<tr>
<td>use of data model</td>
<td></td>
<td></td>
</tr>
<tr>
<td>manual map to BI</td>
<td></td>
<td></td>
</tr>
<tr>
<td>high maintenance effort</td>
<td>low maintenance effort</td>
<td></td>
</tr>
<tr>
<td>only physical changes to be handled</td>
<td>also logical changes to be handled</td>
<td></td>
</tr>
</tbody>
</table>

---

Look at Robin’s EXCELLENT blog entry:

**Incremental refresh of Exalytics aggregates using native BI Server capabilities**
WHAT?

Physical Table: "SampleApp". "BISAMPLE". "SAMP_REVENUE_F"

Logical Column - Month Ago Revenue

Data:
- Type: DOUBLE
- Length: 
- Nullable: 
- Derives from: 
  Ago(sum(SAMP_REVENUE_F.REVENUE), [Level Month], 1)

Column Source Type:
- Derived from physical mappings
  Show all logical sources
- Derived from existing columns using an expression
  AGO("SampleApp":"F0 Base Measures":"Revenue", "SampleApp":"H0 Time":"Month", 1)
WHAT?
only physically mapped columns can be used very often there is no choice (e.g. time series)

if you have the choice, it is a choice between FAST or AGILE
no session variables
no VPD
no parent-child
no skip-level or ragged

CLEAN model!

check model on each hierarchy (NOT globally!)
!
runs only in online mode
?
logical level key columns are physically mapped
?
logical level keys are really KEYS
?
all dimension attributes got a level assigned
Watch LTSs, not tables!

“check model” uses the first matching LTS to generate queries to perform checks

“aggregate builder” uses the first matching LTS of a dimension table to create agg-dimension

have a proper (complete) LTS first
PREREQUISITES (2)

Watch LTSs, not tables!

"check model" uses the first matching LTS to generate queries to perform checks.
"aggregate builder" uses the first matching LTS of a dimension table to create agg-dimension.

Have a proper (complete) LTS first.

Sunday, 19 May 13
create aggregates

"ag_872832912"
for "SampleApp"."F0 Base Measures"("Revenue","Billed Quantity","Discount Amount")
at levels ("SampleApp"."H0 Time"."Year" using_surrogate_key, "SampleApp"."H1 Products"."Products Type")
using connection pool "TimesTen"."TT Connection"
in "TimesTen"."EXALYTICS",

"ag_2964341664"
for "SampleApp"."F0 Base Measures"("Revenue","Billed Quantity","Discount Amount")
at levels ("SampleApp"."H0 Time"."Year", "SampleApp"."H2 Offices"."Offices Detail")
using connection pool "TimesTen"."TT Connection"
in "TimesTen"."EXALYTICS";

💡 using surrogate keys makes queries FASTER but makes refresh SLOWER
💡 use surrogate keys NEVER or ALWAYS (for a level)
create aggregates

"ag_872832912"
for "SampleApp"."F0 Base Measures"("Revenue","Billed Quantity","Discount Amount")
at levels ("SampleApp"."H0 Time"."Year" using_surrogate_key, "SampleApp"."H1 Products"."Products Type")
using connection pool "TimesTen"."TT Connection"
in "TimesTen"."EXALYTICS".

"ag_2964341664"
for "SampleApp"."F0 Base Measures"("Revenue","Billed Quantity","Discount Amount")
at levels ("SampleApp"."H0 Time"."Year", "SampleApp"."H2 Offices"."Offices Detail")
using connection pool "TimesTen"."TT Connection"
in "TimesTen"."EXALYTICS";

within a statement the sequence of build will be optimized
next statement uses BI model with these aggregates
if sequence is important, create 1 aggregate per statement and order statements as required
EXAMPLE

THEN #Employees (dept, year)  build FIRST

IF #Employees (dept, month)  build NEXT

day  

month  

year
CREATE SCRIPT

Utilities
Replace Column or Table in Logical Table Source
Oracle BI Event Tables
Externalize Strings
Rename Wizard
Update Physical Layer
Repository Documentation
Remove Unused Physical Objects
Aggregate Persistence
Generate Deployment File
Convert
Oracle BI

Utilities
Replace Column or Table in Logical Table Sources
Oracle BI Event Tables
Externalize Strings
Rename Wizard
Update Physical Layer
Repository Documentation
Remove Unused Physical Objects
Aggregate Persistence
Generate Deployment File
Convert Presentation Folders
Oracle BI Summary Advisor

```
create aggregates

"ag_87832912"
for "SampleApp"."FO Base Measures" at levels ("SampleApp"."HO Time") using connection pool "TimesTen"."EXALYTICS",

"ag_296431664"
for "SampleApp"."FO Base Measures" at levels ("SampleApp"."HO Time") using connection pool "TimesTen"."EXALYTICS";
```

Utilities
Select File Location
Select Business Measures
Select Levels
Select Connection Pool
Finish

Utilities
Filter Logs - Logical Fact Table
Filter Logs - Time Window
Filter Logs - Execution Time Threshold
Targets
Select File Location
Stopping Criteria
Miscellaneous
Run
Filter Aggregates
Finish Script
DECOUPLING MULTI-SOURCING

Use

query on DETAIL level

select the only matching source

Presentation layer

Business layer

Physical layer

BI Server

Oracle TT

Oracle DB

AGGR

“physical” SQL

get value from data

Sunday, 19 May 13
Use

DECOUPLING MULTI-SOURCING

query on AGGR level

select most efficient source

Physical layer

Business layer

Presentation layer

Oracle TT

Oracle DB

“physical” SQL


Detail

AGGR

BI Server

get value from data

DECOUPLING MULTI-SOURCING

Physical layer

Business layer

Presentation layer

Oracle TT

Oracle DB

“physical” SQL

query on AGGR level

select most efficient source

Detail

AGGR

BI Server

get value from data
LEVEL COUNTS

Use

Product

3

Brand

6

LOB

11

Product Type

20

Product

Business Model and Mapping

Logical Level - Products Type

Name: Products Type

Number of elements at this level: 11

- Brand total level
- Supports rollup to higher level of aggregation

Oracle BI Administration Tool

Estimate Levels

- SampleApp Slow”, “H1 Products”, “Products Brand” = 3
- SampleApp Slow”, “H1 Products”, “Products LOB” = 6
- SampleApp Slow”, “H1 Products”, “Products Type” = 11
- SampleApp Slow”, “H1 Products”, “Products Detail” = 20

Sunday, 19 May 13
Use LEVEL COUNTS

"Golden rule"
For each hierarchy (and LTS)
check consistency
check model
estimate levels

Sales (PT, M)

Sales (B, HY)
Refresh AGGS

Complete

Dimension

Fact

Complete

Increment last period

Complete

Complete

Complete

Complete

using_surrogate_key
refresh facts incrementally
- unless there are late arriving facts
刷新维度完全
- 刷新依赖的事实完全，当键被更新的时候
- 刷新依赖的事实完全，当使用代理键的时候

有一个QA-仪表板，检查逻辑级键
=> 汇总和详细数据的比较
使用变量INACTIVE_SCHEMAS
REBUILD AGGS

Maintain

Script

BI Server

Presentation layer

Business layer

Physical layer

get value from data

Sunday, 19 May 13
REBUILD AGGS

Maintain

Script

Expression Builder - Expression

CASE
  WHEN "SampleApp"."BISAMPLE"."SAMP_REVENUE_F"."REVENUE" < 1000 THEN 'Below 1k'
  WHEN "SampleApp"."BISAMPLE"."SAMP_REVENUE_F"."REVENUE" < 2000 THEN 'Between 1k and 2k'
  WHEN "SampleApp"."BISAMPLE"."SAMP_REVENUE_F"."REVENUE" < 3000 THEN 'Between 2k and 3k'
  WHEN "SampleApp"."BISAMPLE"."SAMP_REVENUE_F"."REVENUE" < 5000 THEN 'Between 3k and 5k'
  ELSE 'Over 5k'
END
Maintain Script

REBUILD AGGS

Logical Column - Order Month

Data:
Type: INT Length: Nullable

Dervives from:
extract(month from SAMP_REVENUE_F_ORDER_DAY_DT)

Column Source Type
- Derived from physical mappings
  - Show all logical sources
    - Logical Table Source: SAMP_REVENUE_F
      - Mapped as: Month("SampleApp"."BISAMPLE".)

Logical Column - Billed Month

Data:
Type: INT Length: Nullable

Dervives from:
extract(month from SAMP_REVENUE_F_BILL_DAY_DT)

Column Source Type
- Derived from physical mappings
  - Show all logical sources
    - Logical Table Source: SAMP_REVENUE_F

- Derived from existing columns using an expression
  - EXPACT(MONTH FROM "SampleApp"."D3 Orders (Facts Attributes)"."Billed Date")
REBUILD AGGS

Maintain

Script

Sunday, 19 May 13
Maintain

Script

REBUILD AGGS

H0 Time
- Total Time
  - Fiscal Year
    - Per Name Fiscal Year
    - Fiscal Month
      - Per Name Fiscal Month
      - Day Detail
    - Calendar Date
  - Year
    - Per Name Year
  - Month
    - Per Name Month
  - Day Detail
    - Calendar Date

H01 Fiscal Time
- Total Time
  - Fiscal Year
    - Per Name Fiscal Year
    - Fiscal Month
      - Per Name Fiscal Month
      - Day Detail
    - Calendar Date

H00 Time
- Total Time
  - Year
    - Per Name Year
  - Month
    - Per Name Month
  - Day Detail
    - Calendar Date
consider rebuilding aggregates (i.e. delete and re-create with updated script)

=> WHEN physical mapping has changed
=> WHEN hierarchies are changed
SUMMARY

- Introduction to aggregates
  - Logical constraints
- Identifying useful aggregates
  - Useful?
- Building aggregates
  - ETL or script
- Using aggregates
  - Prerequisites => Level counts
- Refreshing aggregates
  - Fast & safe
- Lifecycle considerations
  - Which changes change what?
The MAGIC of AGGREGATES

get value from data