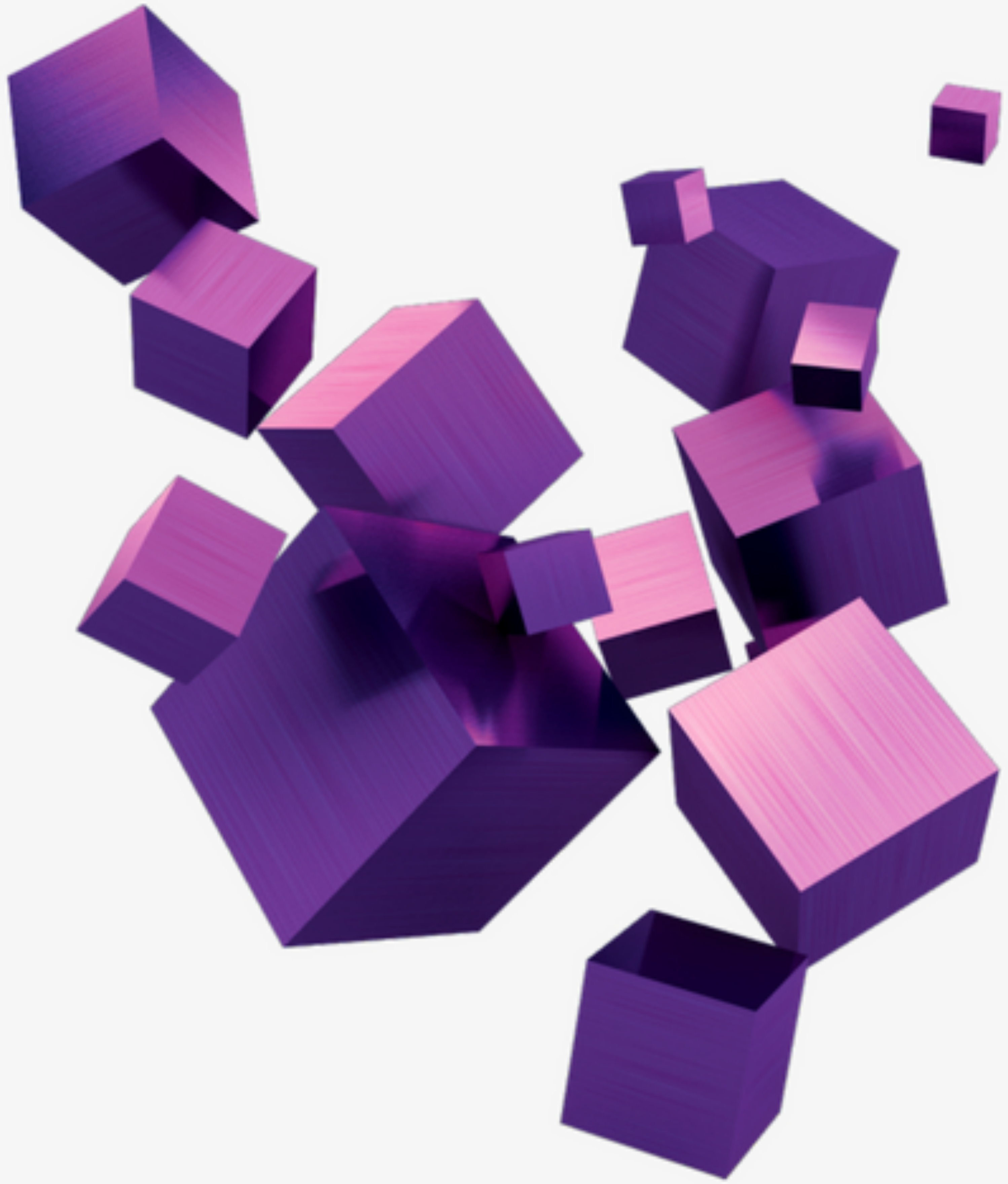


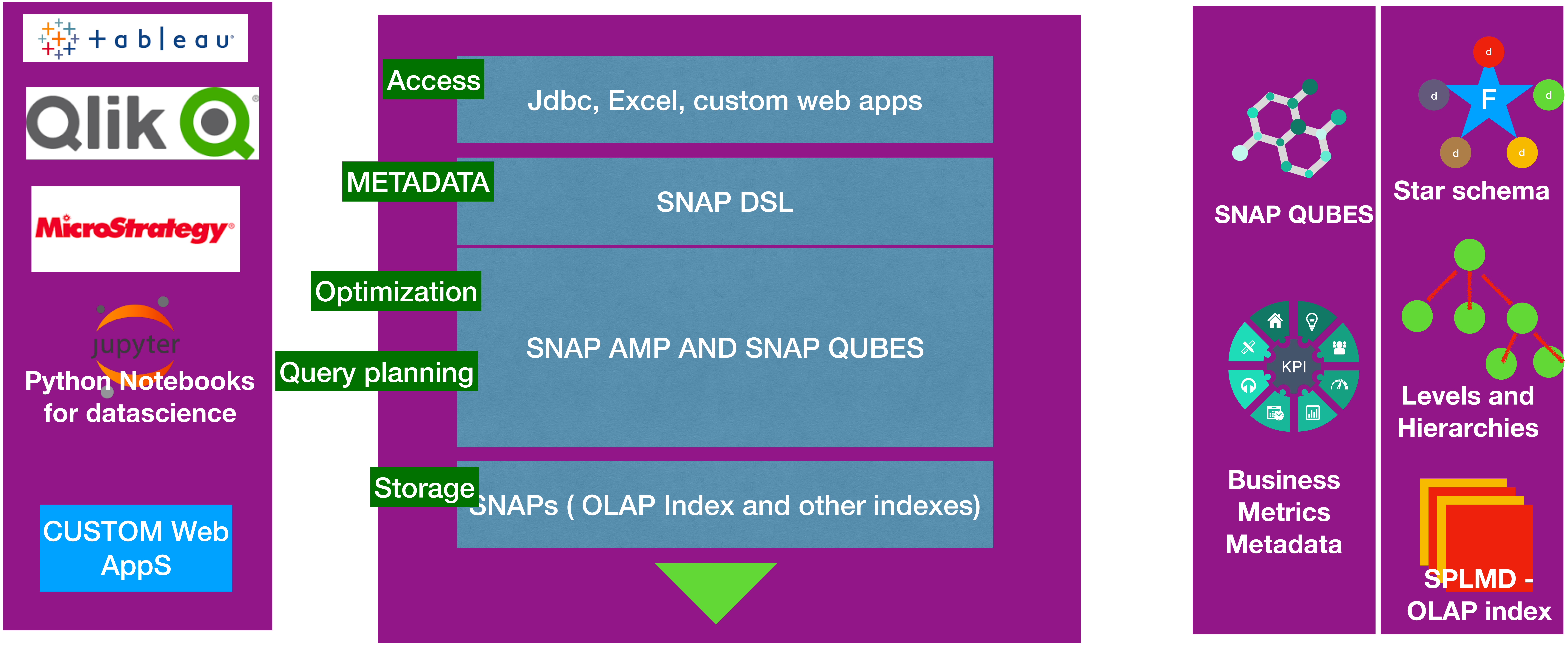
Sparkline SNAP

Enterprise Datamart | Sub second queries

JUNE, 2017 |

Solution Architecture

SPARKLINE
METADATA



Python Notebooks for datascience

CUSTOM Web AppS

Access

Jdbc, Excel, custom web apps

METADATA

SNAP DSL

Optimization

Query planning

SNAP AMP AND SNAP QUBES

Storage

SNAPs (OLAP Index and other indexes)

SNAP QUBES

Star schema

Business Metrics Metadata

Levels and Hierarchies

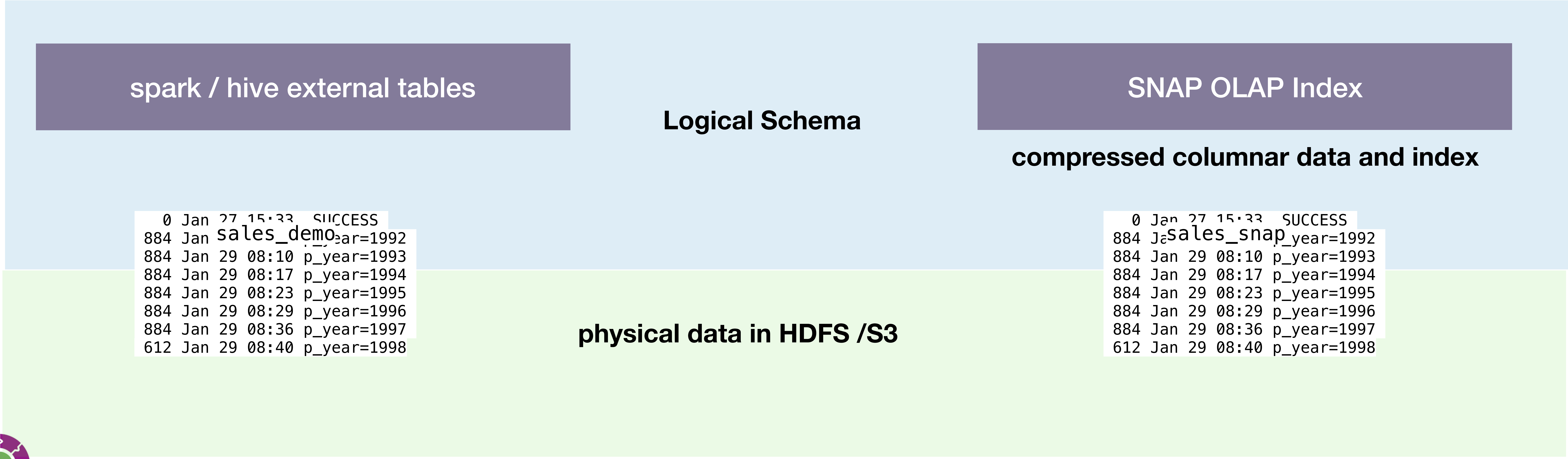
SPLMD - OLAP index

Set up an index

SNAP indexes

create external table ……

create olap index ……



spark / hive external tables
Logical Schema
SNAP OLAP Index
compressed columnar data and index
physical data in HDFS /S3

sales_demo

```
  0 Jan 27 15:33  _SUCCESS
884 Jan           ear=1992
884 Jan 29 08:10 p_year=1993
884 Jan 29 08:17 p_year=1994
884 Jan 29 08:23 p_year=1995
884 Jan 29 08:29 p_year=1996
884 Jan 29 08:36 p_year=1997
612 Jan 29 08:40 p_year=1998
```

sales_snap

```
  0 Jan 27 15:33  _SUCCESS
884 Ja            _year=1992
884 Jan 29 08:10 p_year=1993
884 Jan 29 08:17 p_year=1994
884 Jan 29 08:23 p_year=1995
884 Jan 29 08:29 p_year=1996
884 Jan 29 08:36 p_year=1997
612 Jan 29 08:40 p_year=1998
```

SNAP Qubes - Step 1
Set up Spark datasource table

```
CREATE TABLE IF NOT EXISTS sales_demo
  (
     o_orderkey      INTEGER,
     o_orderstatus   STRING,
     o_totalprice    DOUBLE,
     o_orderdate     STRING,
     o_orderpriority STRING,
     o_shippriority  INTEGER,
     l_linenumber    INTEGER,
     l_quantity      DOUBLE,
     l_extendedprice DOUBLE,
     l_discount      DOUBLE,
     l_tax           DOUBLE,
     l_returnflag    STRING,
     l_linestatus    STRING,
     l_shipdate      STRING,
     l_commitdate    STRING,
     l_receiptdate   STRING,
     l_shipmode      STRING,
     order_year      STRING,
     ps_availqty     INTEGER,
     ps_supplycost   DOUBLE,
     s_name          STRING,
     s_acctbal       DOUBLE,
     s_nation        STRING,
     s_region        STRING,
     p_name          STRING,
     p_mfgr          STRING,
     p_brand         STRING,
     p_type          STRING,
     p_size          INTEGER,
     p_container     STRING,
     p_retailprice   DOUBLE,
     c_name          STRING,
     c_phone         STRING,
     c_acctbal       DOUBLE,
     c_mktsegment    STRING,
     c_nation        STRING,
     c_region        STRING,
     p_year          STRING,
     p_month         STRING
  )

using csv
OPTIONS (path “s3://SNAP/samples/sales_demo_par”)
partitioned by ( p_year, p_month )
```

A. Your Data is in HDFS /S3 or S3 compatible object storage or an external datawarehouse.

B. Connect to SNAP (using jdbc:hive://<ip address where snap server is running: port and any authentication you may have configured. (see hive URL format for example of authentication)

C. create an external table pointing at the data

    A. See example to the left.

D. if your data is not partitioned, partition the data.

A simple Qube - step 2

```
o_orderkey      STRING,
o_orderstatus   STRING,
o_orderpriority STRING,
o_shippriority  STRING,
l_linenumber    STRING,
l_returnflag    STRING,
l_linestatus    STRING,
l_shipmode      STRING,
order_year      STRING,
s_name          STRING,
s_nation        STRING,
s_region        STRING,
p_name          STRING,
p_mfgr          STRING,
p_brand         STRING,
p_type          STRING,
p_size          STRING,
p_container     STRING,
c_name          STRING,
c_phone         STRING,
c_acctbal       DOUBLE,
c_mktsegment    STRING,
c_nation        STRING,
c_region        STRING
o_orderdate     STRING,
l_shipdate      STRING,
l_commitdate    STRING,
l_receiptdate   STRING,
o_totalprice    DOUBLE,
l_quantity      DOUBLE,
l_extendedprice DOUBLE,
l_discount      DOUBLE,
l_tax           DOUBLE,
s_acctbal       DOUBLE,
ps_availqty     INTEGER,
ps_supplycost   DOUBLE,
p_retailprice   DOUBLE,
```

Source table we defined in the previous step

Define columns As either metrics Or dimensions

Dimensions

Metrics

```
create olap index sales_snap on sales_demo
dimension p_name is not nullable
timestamp dimension l_shipdate spark
timestamp dimension o_orderdate
timestamp dimension l_commitdate
          is nullable nullvalue
"1992-01-01T00:00:00.000"
timestamp dimension l_receiptdate
          is not nullable
dimensions
"o_orderkey,o_orderstatus,o_orderpriority,o_
shippriority,l_linenumber,l_returnflag,l_lin
estatus,l_shipmode,s_name,s_nation,s_region,
p_mfgr,p_brand,p_type,p_size,p_container,c_n
ame,c_phone,c_mktsegment,c_nation,c_region"
metrics
"o_totalprice,l_quantity,l_extendedprice,l_d
iscount,l_tax,ps_availqty,ps_supplycost,s_ac
ctbal,p_retailprice,c_acctbal"
OPTIONS ( path "/SNAP/samples/
sales_demo_index", rowFlushBoundary "10000",
rowsPerSegment "50000",
nonAggregateQueryHandling
"push_project_and_filters")
partition by order_year
```

Qube sections

```
create olap index sales_snap on sales_demo
dimension p_name is not nullable
timestamp dimension l_shipdate spark is nullable nullvalue
"1992-01-01T00:00:00.000"
timestamp dimension o_orderdate
timestamp dimension l_commitdate
          is nullable nullvalue "1992-01-01T00:00:00.000"
dimension o_orderstatus is nullable nullvalue “NA”
dimensions
“o_orderkey,o_orderpriority,o_shippriority,l_linenumber,l_
returnflag,…”
metrics “o_totalprice,l_quantity,l_extendedprice”

OPTIONS ( path "/SNAP/samples/sales_demo_index",,
 avgSizePerPartition "100mb",
               preferredSegmentSize "100mb",
        rowFlushBoundary "100000")
partition by order_year
```

Name of the index

Name of external hive or spark table

define rules on dimensions if any - Default nullValue

Define dimensions and metrics(exclude those you have already defined earlier as timestamps, in this section

Define index options - see docs for detailed explanation of options.

Partitioning indexes

```
create olap index sales_snap on sales_demo
dimension p_name is not nullable
timestamp dimension l_shipdate is nullable nullvalue "1992-01-01T00:00:00.000"
timestamp dimension o_orderdate
timestamp dimension l_commitdate
          is nullable nullvalue "1992-01-01T00:00:00.000"

dimensions
“o_orderkey,o_orderstatus,o_orderpriority,o_shippriority,l_linenumber,l_returnflag,…”
metrics “o_totalprice,l_quantity,l_extendedprice,…”

OPTIONS ( path "/SNAP/samples/sales_demo_index", rowFlushBoundary "10000",
rowsPerSegment “50000",
 avgSizePerPartition "100mb",
        avgNumRowsPerPartition "1660000",
        preferredSegmentSize "100mb",
        rowFlushBoundary "100000")
partition by order_year
```

Indexes can be partitioned even if the source table is not

If your source data is already partitioned use the same partition scheme for the indexes as well.

when new data is added to your source partition insert data into the index as well

Source table is a flat table

Index is partitioned on “order_year”

adding data to snap indexes

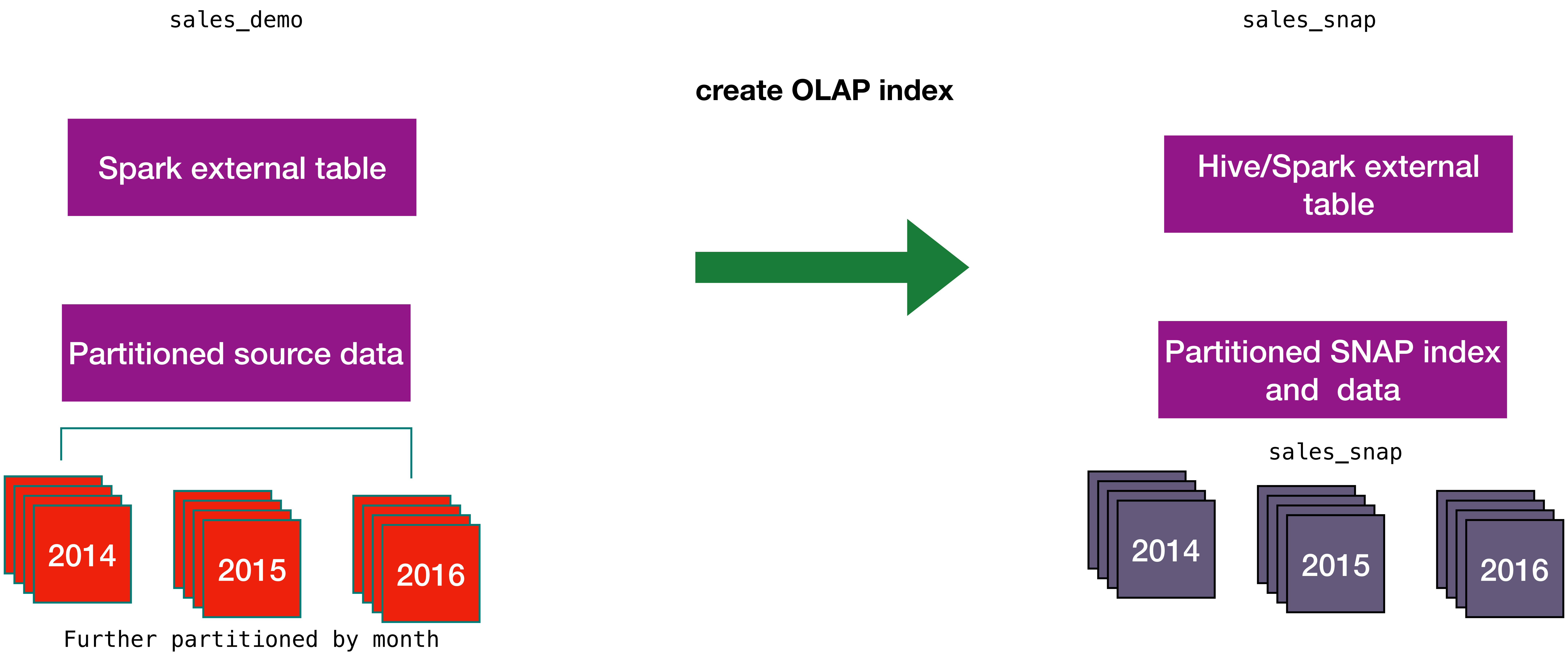
Insert olap index *sales_snap* of *sales_demo*

partitions   *order_year = “2017”*

Insert overwrite olap index *sales_snap* of *sales_demo*

partitions   *order_year = “2016”*

Putting it all together



sales_demo

Spark external table

Partitioned source data

2014

2015

2016

Further partitioned by month

create OLAP index

sales_snap

Hive/Spark external table

Partitioned SNAP index and data

sales_snap

2014

2015

2016

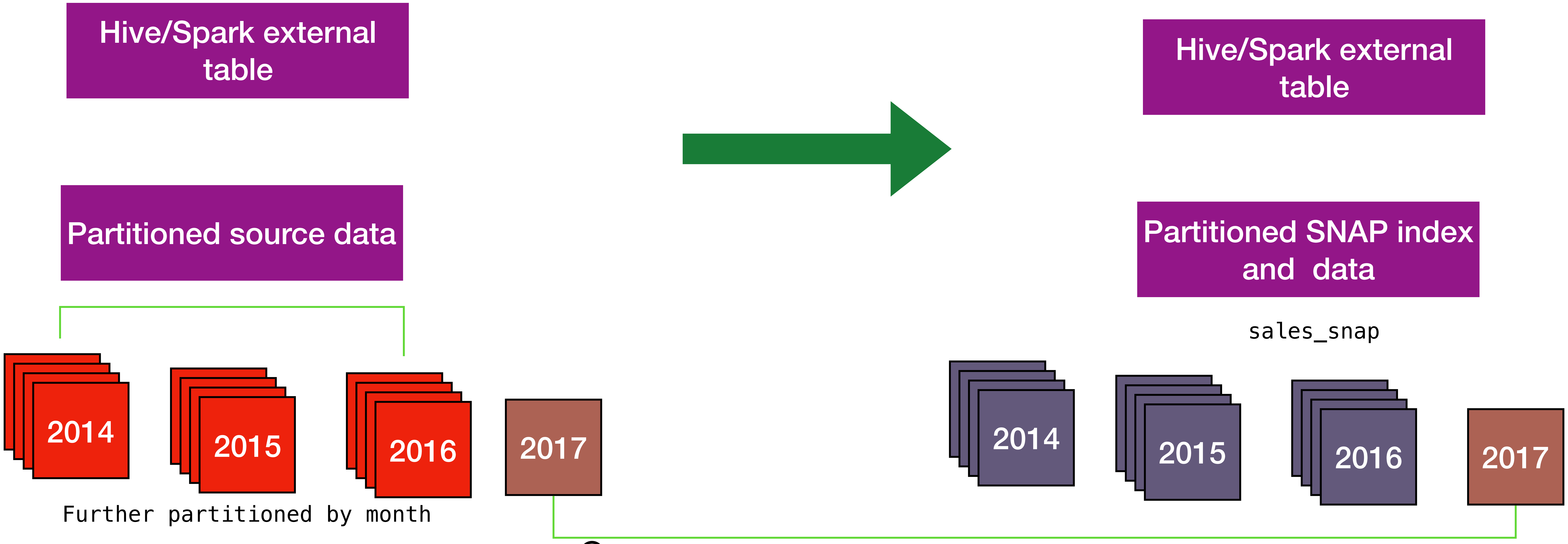
Putting it all together

Insert olap index *sales_snap* of *sales_demo*

sales_demo

partitions *order_year = “2017” order_month = “01”*

sales_snap



Hive/Spark external table

Partitioned source data

Further partitioned by month

Hive/Spark external table

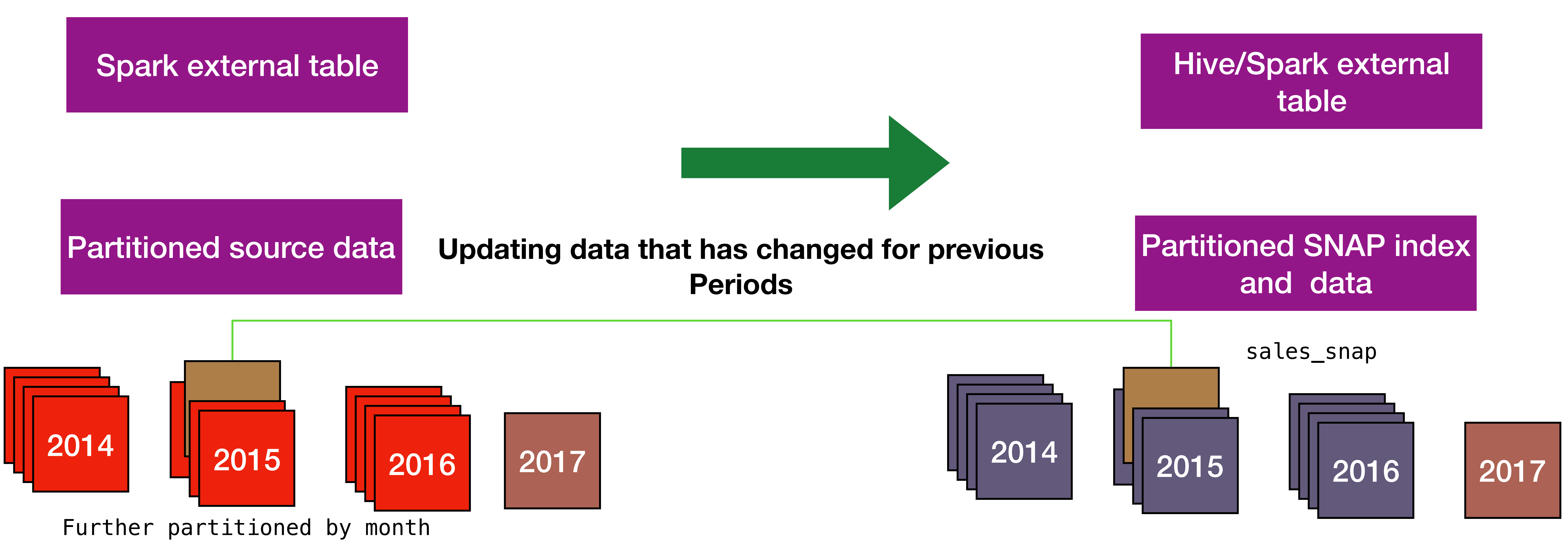
Partitioned SNAP index and data

sales_snap

Putting it all together

Insert overwrite olap index *sales_snap* of *sales_demo*

partitions *order_year = “2016” order_month = “03”*



sales_demo

sales_snap

Spark external table

Partitioned source data

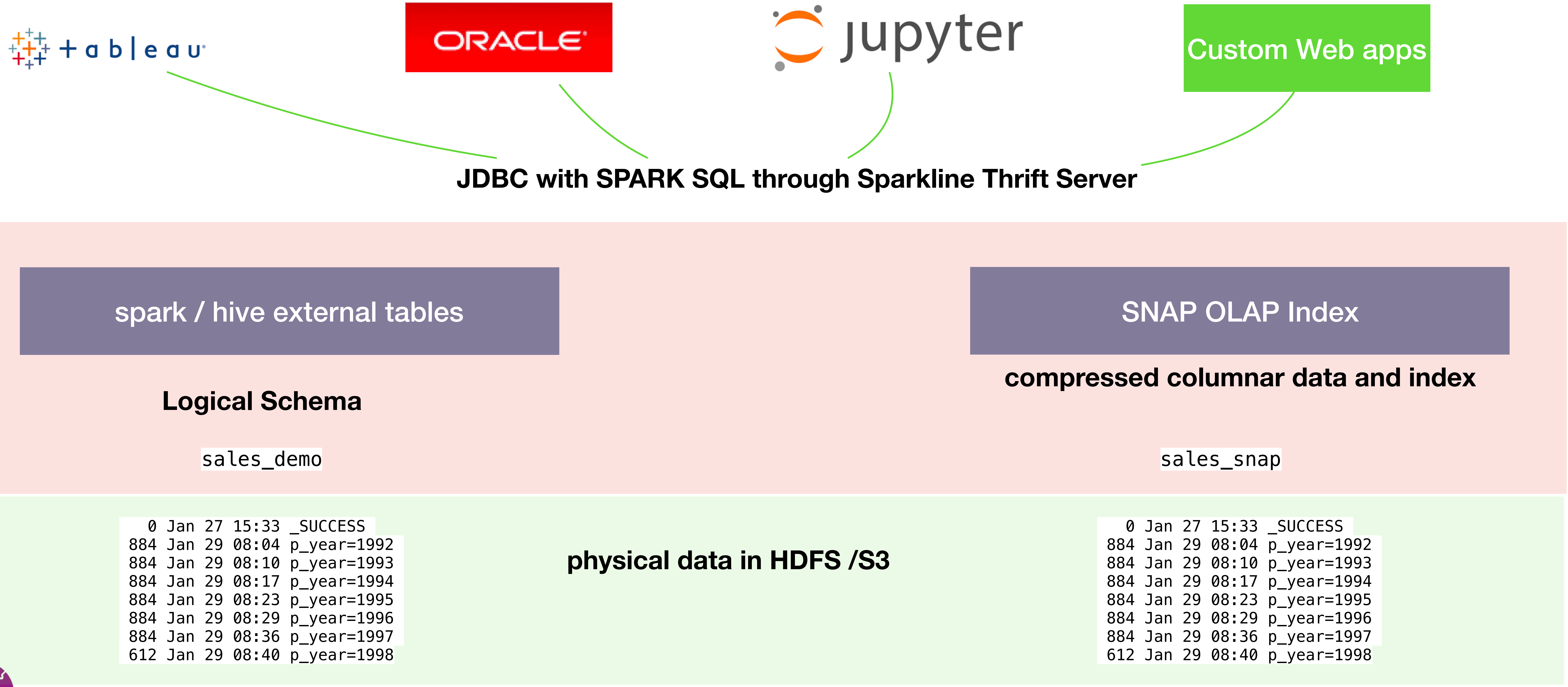
Hive/Spark external table

Partitioned SNAP index and data

Updating data that has changed for previous Periods

Further partitioned by month

SNAP QUERIES



JDBC with SPARK SQL through Sparkline Thrift Server

spark / hive external tables

Logical Schema

`sales_demo`

SNAP OLAP Index

compressed columnar data and index

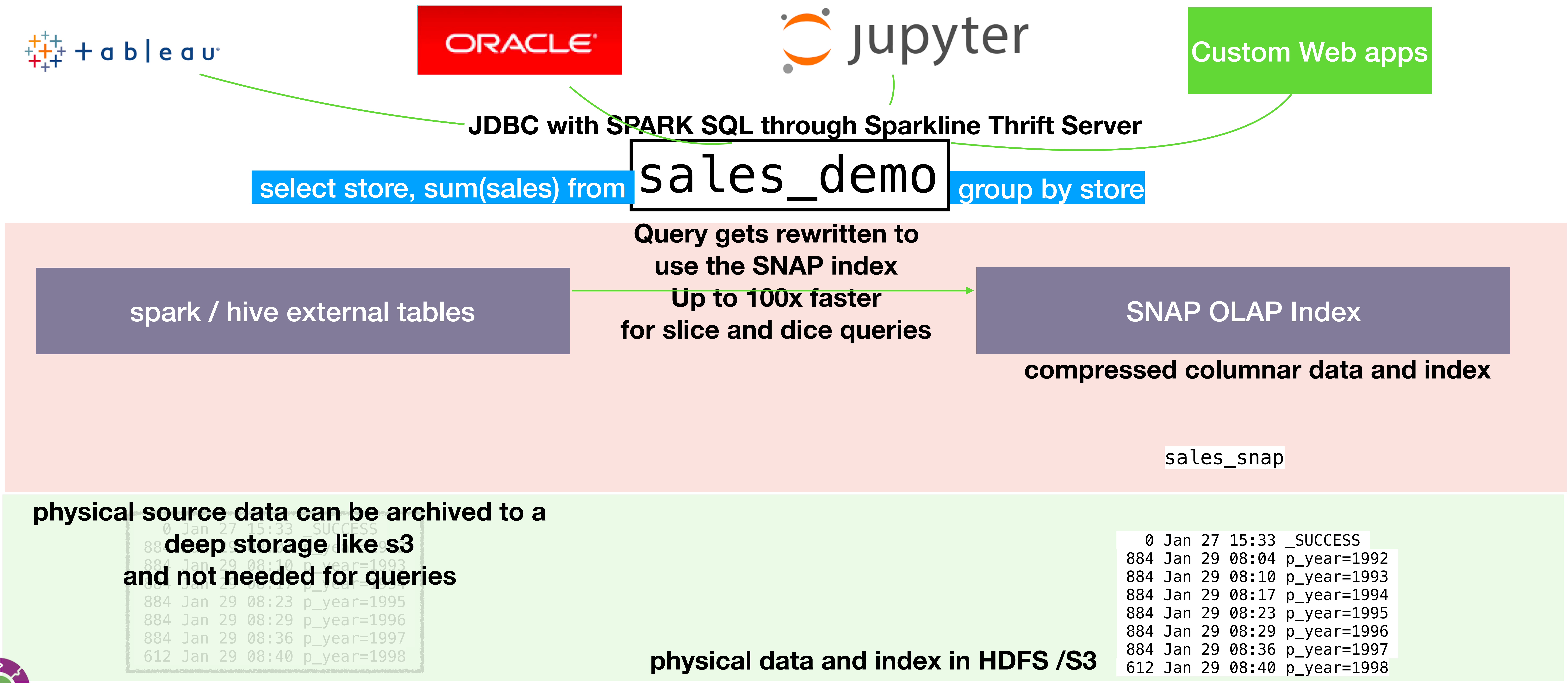
`sales_snap`

```
  0 Jan 27 15:33 _SUCCESS
884 Jan 29 08:04 p_year=1992
884 Jan 29 08:10 p_year=1993
884 Jan 29 08:17 p_year=1994
884 Jan 29 08:23 p_year=1995
884 Jan 29 08:29 p_year=1996
884 Jan 29 08:36 p_year=1997
612 Jan 29 08:40 p_year=1998
```

physical data in HDFS /S3

```
  0 Jan 27 15:33 _SUCCESS
884 Jan 29 08:04 p_year=1992
884 Jan 29 08:10 p_year=1993
884 Jan 29 08:17 p_year=1994
884 Jan 29 08:23 p_year=1995
884 Jan 29 08:29 p_year=1996
884 Jan 29 08:36 p_year=1997
612 Jan 29 08:40 p_year=1998
```

SNAP QUERIES



JDBC with SPARK SQL through Sparkline Thrift Server

select store, sum(sales) from `sales_demo` group by store

spark / hive external tables

Query gets rewritten to
use the SNAP index
Up to 100x faster
for slice and dice queries

SNAP OLAP Index

compressed columnar data and index

`sales_snap`

physical source data can be archived to a
deep storage like s3
and not needed for queries

```
  0 Jan 27 15:33 _SUCCESS
884 Jan 29 08:10 p_year=1993
884 Jan 29 08:23 p_year=1995
884 Jan 29 08:29 p_year=1996
884 Jan 29 08:36 p_year=1997
612 Jan 29 08:40 p_year=1998
```

physical data and index in HDFS /S3

```
  0 Jan 27 15:33 _SUCCESS
884 Jan 29 08:04 p_year=1992
884 Jan 29 08:10 p_year=1993
884 Jan 29 08:17 p_year=1994
884 Jan 29 08:23 p_year=1995
884 Jan 29 08:29 p_year=1996
884 Jan 29 08:36 p_year=1997
612 Jan 29 08:40 p_year=1998
```

SNAP with Tableau

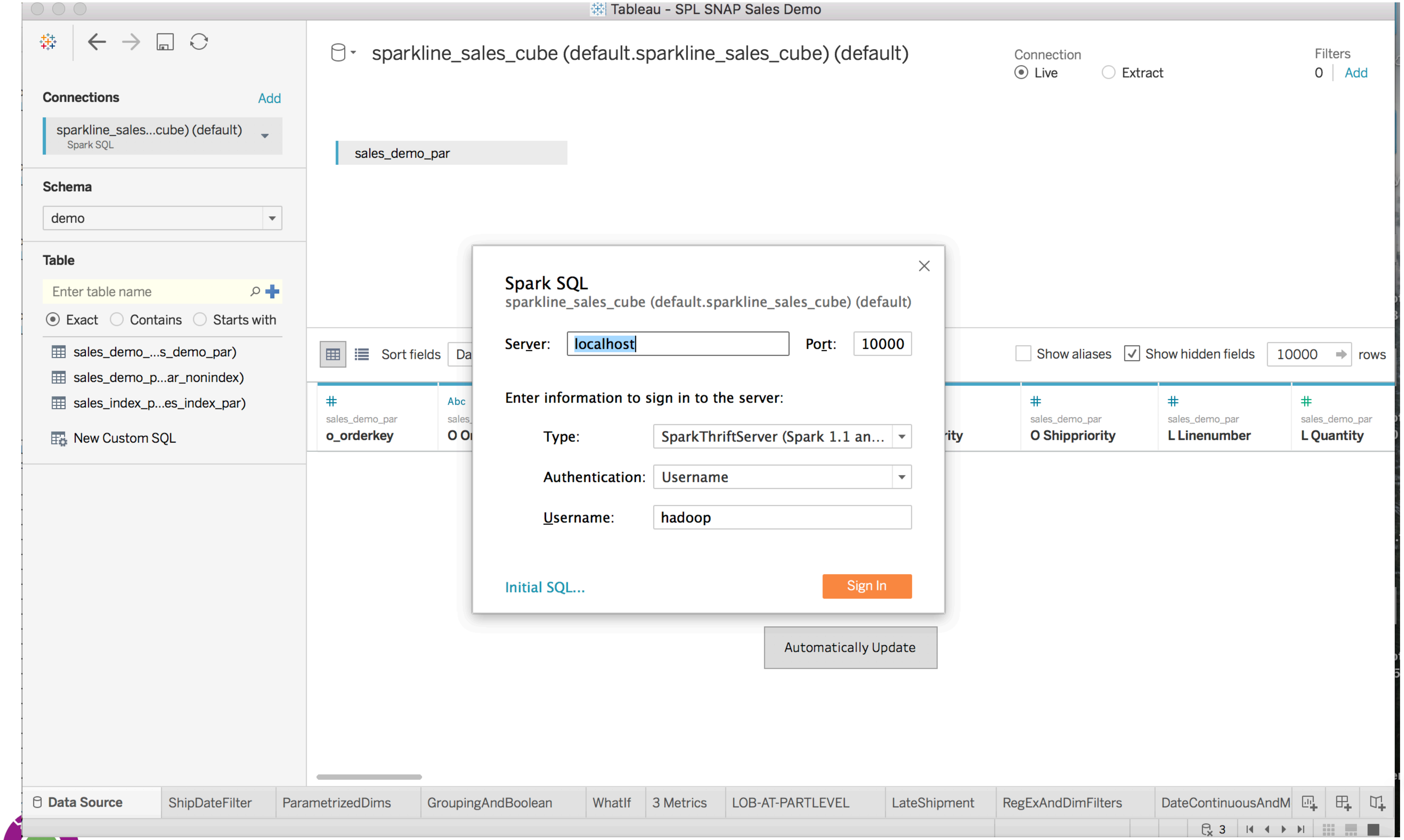


Tableau - SPL SNAP Sales Demo

sparkline_sales_cube (default.sparkline_sales_cube) (default)

Connection: ● Live ○ Extract

Filters: 0 Add

Connections: sparkline_sales...cube) (default) Spark SQL

Schema: demo

Table: ● Exact ○ Contains ○ Starts with

Spark SQL
sparkline_sales_cube (default.sparkline_sales_cube) (default)

Server: localhost

Port: 10000

Enter information to sign in to the server:

Type: SparkThriftServer (Spark 1.1 an...

Authentication: Username

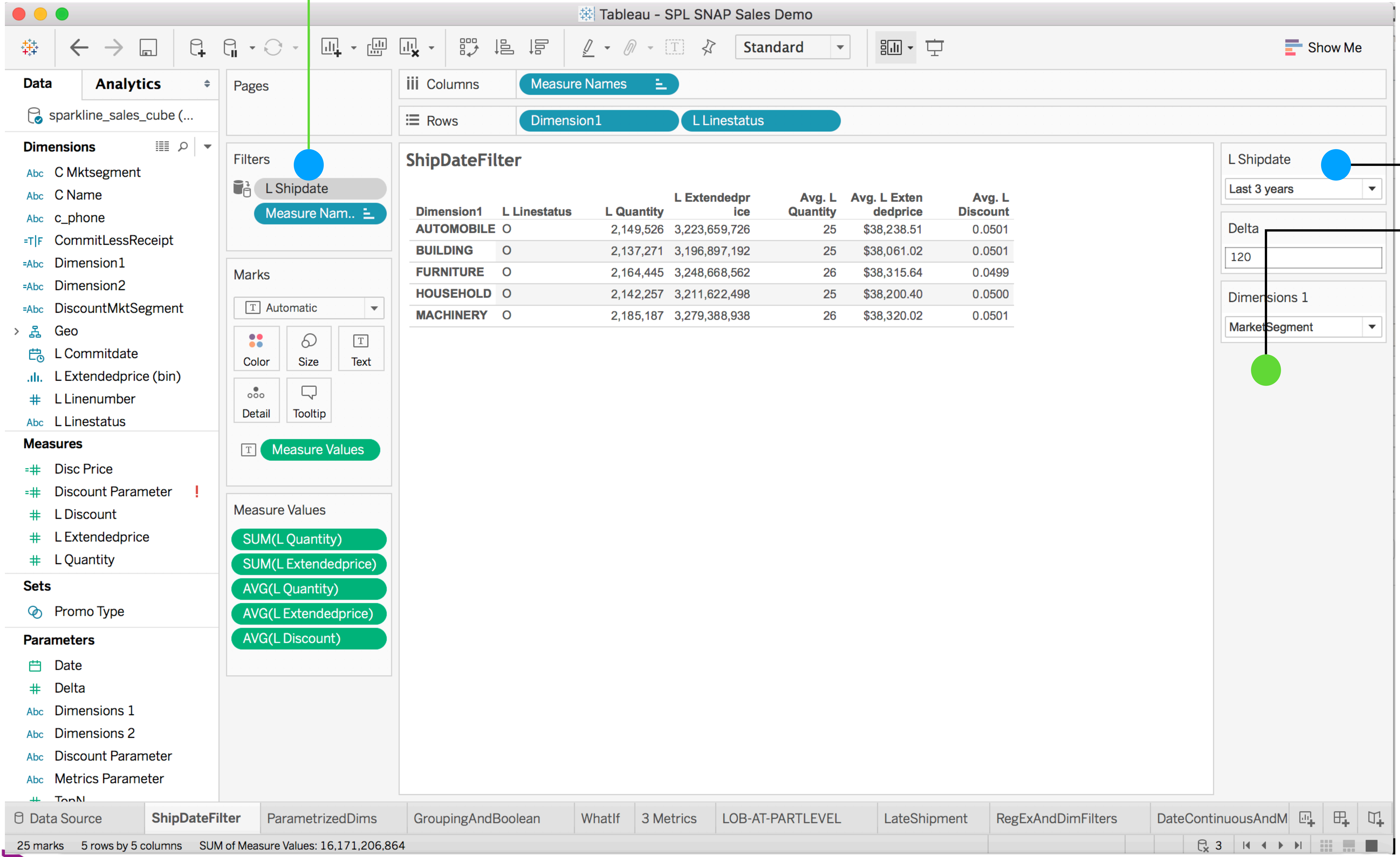
Username: hadoop

☐ Show aliases ☑ Show hidden fields 10000 rows

Start tableau and connect to the spark-sql connector

Choose the host where your started the thrift server and the corresponding port

types of queries in tableau



ShipDateFilter

Dimension1	L Linestatus	L Quantity	L Extendedprice	Avg. L Quantity	Avg. L Extendedprice	Avg. L Discount
AUTOMOBILE	O	2,149,526	3,223,659,726	25	$38,238.51	0.0501
BUILDING	O	2,137,271	3,196,897,192	25	$38,061.02	0.0501
FURNITURE	O	2,164,445	3,248,668,562	26	$38,315.64	0.0499
HOUSEHOLD	O	2,142,257	3,211,622,498	25	$38,200.40	0.0500
MACHINERY	O	2,185,187	3,279,388,938	26	$38,320.02	0.0501

L Shipdate: Last 3 years

Delta: 120

Dimensions 1: MarketSegment

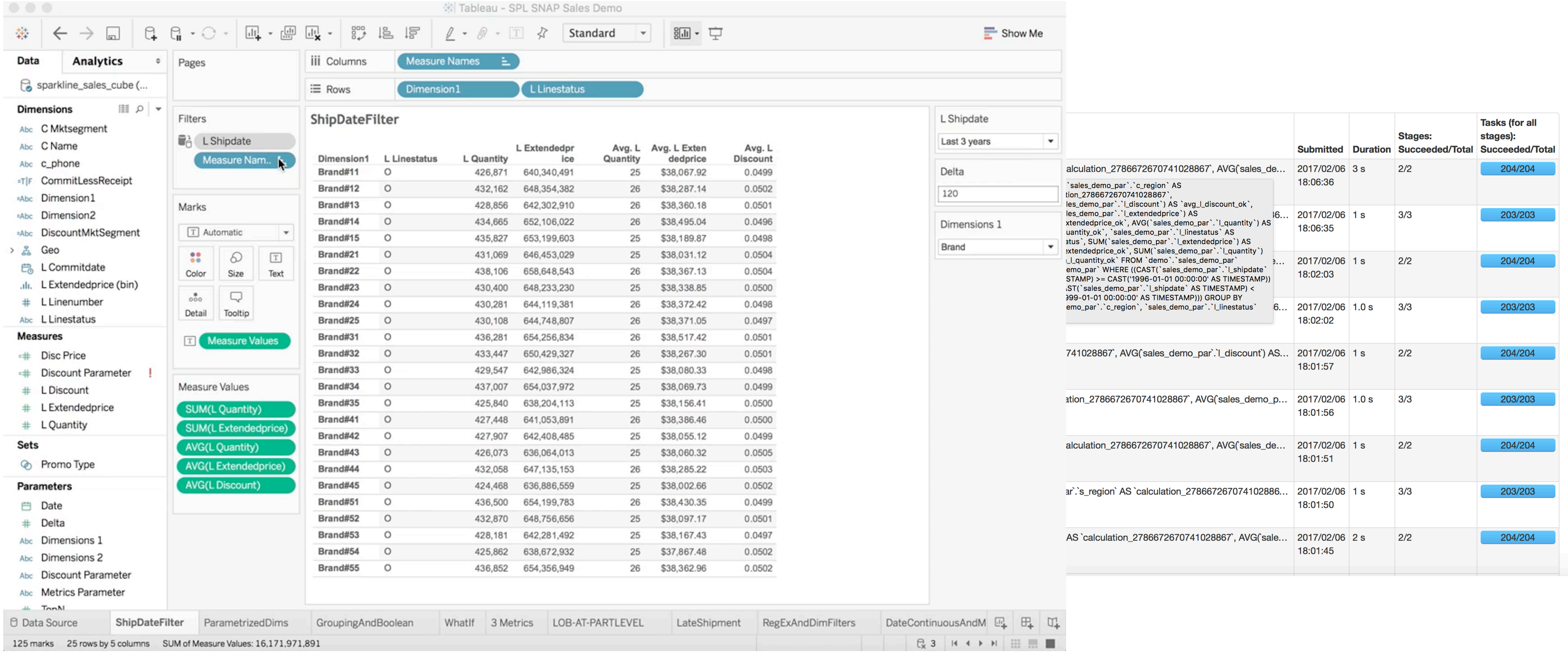
25 marks 5 rows by 5 columns SUM of Measure Values: 16,171,206,864

Timestamp filter

Dimension filter

choose drop down to dynamically change the

Example



Data source: sparkline_sales_cube (...

Columns: Measure Names

Rows: Dimension1 | L Linestatus

Filters: L Shipdate | Measure Nam..

Marks: Measure Values

Dimensions
- C Mktsegment
- C Name
- c_phone
- CommitLessReceipt
- Dimension1
- Dimension2
- DiscountMktSegment
- Geo
- L Commitdate
- L Extendedprice (bin)
- L Linenumber
- L Linestatus

Measures
- Disc Price
- Discount Parameter !
- L Discount
- L Extendedprice
- L Quantity

Sets
- Promo Type

Parameters
- Date
- Delta
- Dimensions 1
- Dimensions 2
- Discount Parameter
- Metrics Parameter
- TopN

Measure Values
- SUM(L Quantity)
- SUM(L Extendedprice)
- AVG(L Quantity)
- AVG(L Extendedprice)
- AVG(L Discount)

L Shipdate: Last 3 years

Delta: 120

Dimensions 1: Brand

ShipDateFilter

Dimension1	L Linestatus	L Quantity	L Extendedprice	Avg. L Quantity	Avg. L Extendedprice	Avg. L Discount
Brand#11	O	426,871	640,340,491	25	$38,067.92	0.0499
Brand#12	O	432,162	648,354,382	26	$38,287.14	0.0502
Brand#13	O	428,856	642,302,910	26	$38,360.18	0.0501
Brand#14	O	434,665	652,106,022	26	$38,495.04	0.0496
Brand#15	O	435,827	653,199,603	25	$38,189.87	0.0498
Brand#21	O	431,069	646,453,029	25	$38,031.12	0.0504
Brand#22	O	438,106	658,648,543	26	$38,367.13	0.0504
Brand#23	O	430,400	648,233,230	25	$38,338.85	0.0500
Brand#24	O	430,281	644,119,381	26	$38,372.42	0.0498
Brand#25	O	430,108	644,748,807	26	$38,371.05	0.0497
Brand#31	O	436,281	654,256,834	26	$38,517.42	0.0501
Brand#32	O	433,447	650,429,327	26	$38,267.30	0.0501
Brand#33	O	429,547	642,986,324	25	$38,080.33	0.0498
Brand#34	O	437,007	654,037,972	25	$38,069.73	0.0499
Brand#35	O	425,840	638,204,113	25	$38,156.41	0.0500
Brand#41	O	427,448	641,053,891	26	$38,386.46	0.0500
Brand#42	O	427,907	642,408,485	25	$38,055.12	0.0499
Brand#43	O	426,073	636,064,013	25	$38,060.32	0.0505
Brand#44	O	432,058	647,135,153	26	$38,285.22	0.0503
Brand#45	O	424,468	636,886,559	25	$38,002.66	0.0502
Brand#51	O	436,500	654,199,783	26	$38,430.35	0.0499
Brand#52	O	432,870	648,756,656	25	$38,097.17	0.0501
Brand#53	O	428,181	642,281,492	25	$38,167.43	0.0497
Brand#54	O	425,862	638,672,932	25	$37,867.48	0.0502
Brand#55	O	436,852	654,356,949	26	$38,362.96	0.0502

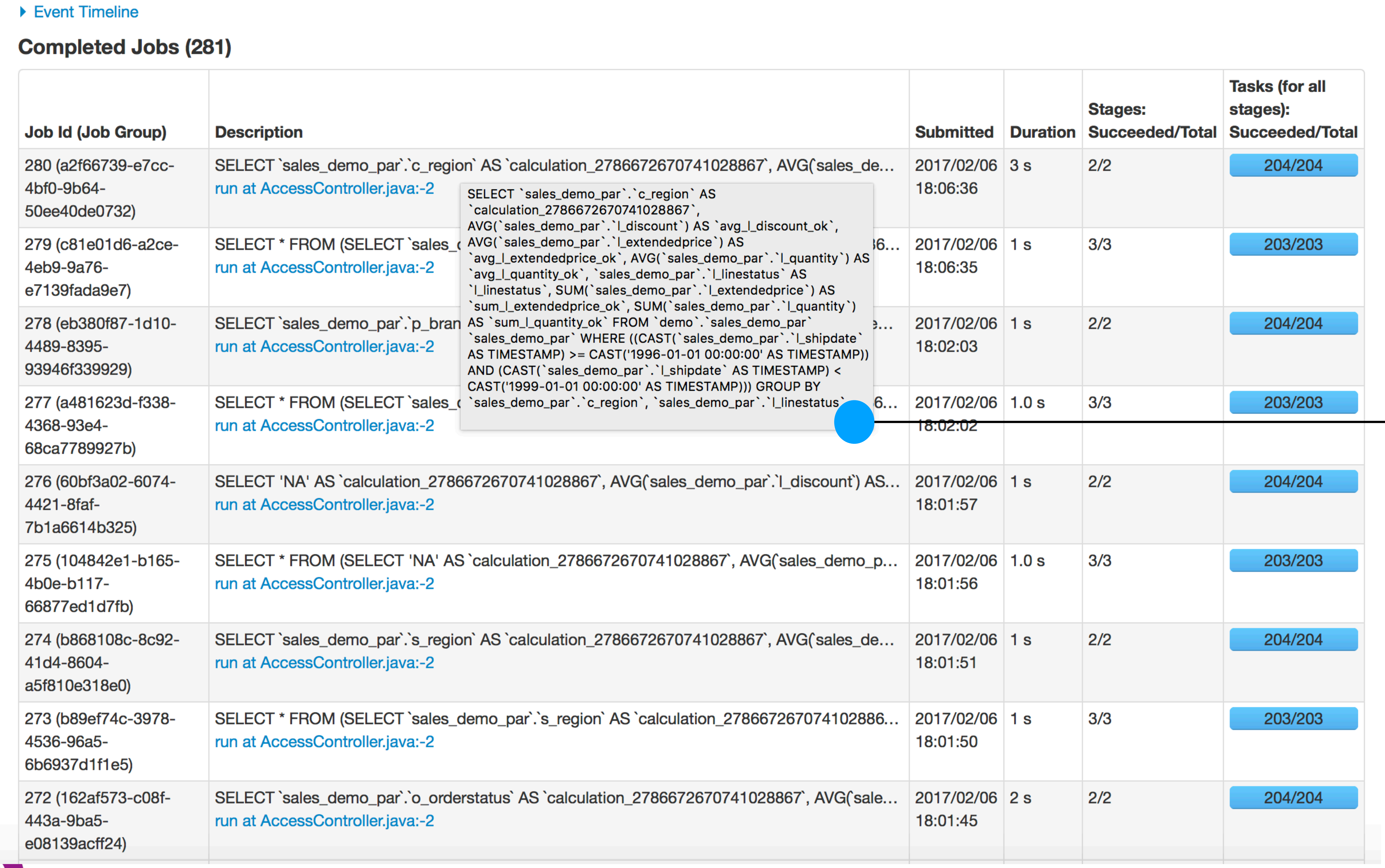
Data Source | ShipDateFilter | ParametrizedDims | GroupingAndBoolean | WhatIf | 3 Metrics | LOB-AT-PARTLEVEL | LateShipment | RegExAndDimFilters | DateContinuousAndM

125 marks | 25 rows by 5 columns | SUM of Measure Values: 16,171,971,891

	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
alculation_2786672670741028867`, AVG(`sales_de...	2017/02/06 18:06:36	3 s	2/2	204/204
6...	2017/02/06 18:06:35	1 s	3/3	203/203
e...	2017/02/06 18:02:03	1 s	2/2	204/204
6...	2017/02/06 18:02:02	1.0 s	3/3	203/203
741028867`, AVG(`sales_demo_par`.`l_discount`) AS...	2017/02/06 18:01:57	1 s	2/2	204/204
ation_2786672670741028867`, AVG(`sales_demo_p...	2017/02/06 18:01:56	1.0 s	3/3	203/203
alculation_2786672670741028867`, AVG(`sales_de...	2017/02/06 18:01:51	1 s	2/2	204/204
ar`.`s_region` AS `calculation_278667267074102886...	2017/02/06 18:01:50	1 s	3/3	203/203
AS `calculation_2786672670741028867`, AVG(`sale...	2017/02/06 18:01:45	2 s	2/2	204/204

```
`sales_demo_par`.`c_region` AS
tion_2786672670741028867`,
les_demo_par`.`l_discount`) AS `avg_l_discount_ok`,
les_demo_par`.`l_extendedprice`) AS
xtendedprice_ok`, AVG(`sales_demo_par`.`l_quantity`) AS
uantity_ok`, `sales_demo_par`.`l_linestatus` AS
atus`, SUM(`sales_demo_par`.`l_extendedprice`) AS
xtendedprice_ok`, SUM(`sales_demo_par`.`l_quantity`)
_l_quantity_ok` FROM `demo`.`sales_demo_par`
emo_par` WHERE ((CAST(`sales_demo_par`.`l_shipdate`
STAMP) >= CAST('1996-01-01 00:00:00' AS TIMESTAMP))
ST(`sales_demo_par`.`l_shipdate` AS TIMESTAMP) <
999-01-01 00:00:00' AS TIMESTAMP))) GROUP BY
emo_par`.`c_region`, `sales_demo_par`.`l_linestatus`
```

Behind the scenes



‣ Event Timeline

Completed Jobs (281)

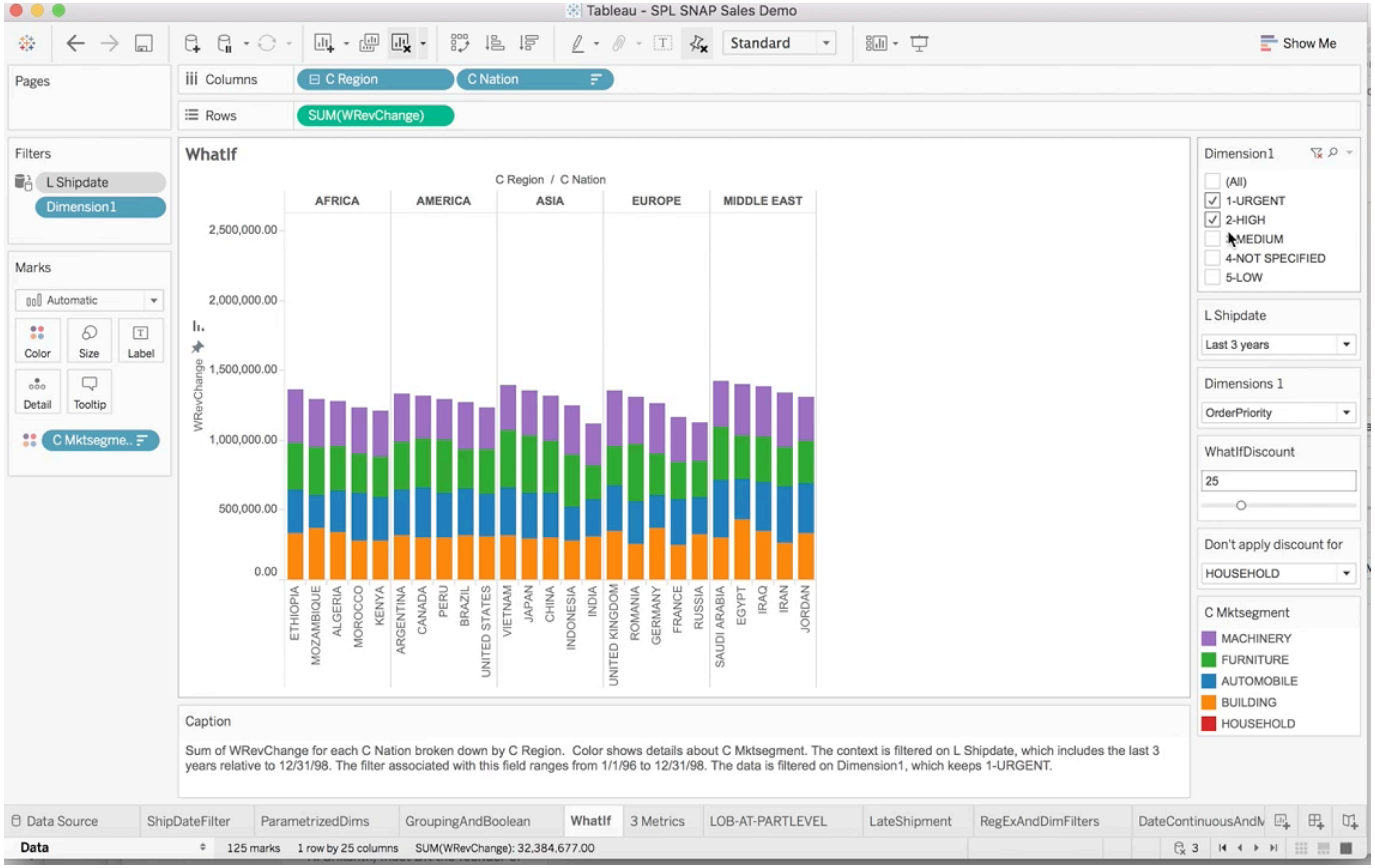
Job Id (Job Group)	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
280 (a2f66739-e7cc-4bf0-9b64-50ee40de0732)	SELECT `sales_demo_par`.`c_region` AS `calculation_2786672670741028867`, AVG(`sales_de… run at AccessController.java:-2	2017/02/06 18:06:36	3 s	2/2	204/204
279 (c81e01d6-a2ce-4eb9-9a76-e7139fada9e7)	SELECT * FROM (SELECT `sales_ 36… run at AccessController.java:-2	2017/02/06 18:06:35	1 s	3/3	203/203
278 (eb380f87-1d10-4489-8395-93946f339929)	SELECT `sales_demo_par`.`p_bran e… run at AccessController.java:-2	2017/02/06 18:02:03	1 s	2/2	204/204
277 (a481623d-f338-4368-93e4-68ca7789927b)	SELECT * FROM (SELECT `sales_ 6… run at AccessController.java:-2	2017/02/06 18:02:02	1.0 s	3/3	203/203
276 (60bf3a02-6074-4421-8faf-7b1a6614b325)	SELECT 'NA' AS `calculation_2786672670741028867`, AVG(`sales_demo_par`.`l_discount`) AS… run at AccessController.java:-2	2017/02/06 18:01:57	1 s	2/2	204/204
275 (104842e1-b165-4b0e-b117-66877ed1d7fb)	SELECT * FROM (SELECT 'NA' AS `calculation_2786672670741028867`, AVG(`sales_demo_p… run at AccessController.java:-2	2017/02/06 18:01:56	1.0 s	3/3	203/203
274 (b868108c-8c92-41d4-8604-a5f810e318e0)	SELECT `sales_demo_par`.`s_region` AS `calculation_2786672670741028867`, AVG(`sales_de… run at AccessController.java:-2	2017/02/06 18:01:51	1 s	2/2	204/204
273 (b89ef74c-3978-4536-96a5-6b6937d1f1e5)	SELECT * FROM (SELECT `sales_demo_par`.`s_region` AS `calculation_278667267074102886… run at AccessController.java:-2	2017/02/06 18:01:50	1 s	3/3	203/203
272 (162af573-c08f-443a-9ba5-e08139acff24)	SELECT `sales_demo_par`.`o_orderstatus` AS `calculation_2786672670741028867`, AVG(`sale… run at AccessController.java:-2	2017/02/06 18:01:45	2 s	2/2	204/204

```
SELECT `sales_demo_par`.`c_region` AS
`calculation_2786672670741028867`,
AVG(`sales_demo_par`.`l_discount`) AS `avg_l_discount_ok`,
AVG(`sales_demo_par`.`l_extendedprice`) AS
`avg_l_extendedprice_ok`, AVG(`sales_demo_par`.`l_quantity`) AS
`avg_l_quantity_ok`, `sales_demo_par`.`l_linestatus` AS
`l_linestatus`, SUM(`sales_demo_par`.`l_extendedprice`) AS
`sum_l_extendedprice_ok`, SUM(`sales_demo_par`.`l_quantity`)
AS `sum_l_quantity_ok` FROM `demo`.`sales_demo_par`
`sales_demo_par` WHERE ((CAST(`sales_demo_par`.`l_shipdate`
AS TIMESTAMP) >= CAST('1996-01-01 00:00:00' AS TIMESTAMP))
AND (CAST(`sales_demo_par`.`l_shipdate` AS TIMESTAMP) <
CAST('1999-01-01 00:00:00' AS TIMESTAMP))) GROUP BY
`sales_demo_par`.`c_region`, `sales_demo_par`.`l_linestatus`
```

Each query in seconds even when scanning for millions of rows

demo running on macbook pro with 8 GB ram and several apps running - tableau, quicktime, keynote etc

What if analysis



Columns: C Region, C Nation

Rows: SUM(WRevChange)

Filters: L Shipdate, Dimension1

Marks: Automatic; C Mktsegme..

Dimension1
□ (All)
☑ 1-URGENT
☑ 2-HIGH
□ 3-MEDIUM
□ 4-NOT SPECIFIED
□ 5-LOW

L Shipdate: Last 3 years

Dimensions 1: OrderPriority

WhatIfDiscount: 25

Don't apply discount for: HOUSEHOLD

C Mktsegment
🟪 MACHINERY
🟩 FURNITURE
🟦 AUTOMOBILE
🟧 BUILDING
🟥 HOUSEHOLD

Caption

Sum of WRevChange for each C Nation broken down by C Region. Color shows details about C Mktsegment. The context is filtered on L Shipdate, which includes the last 3 years relative to 12/31/98. The filter associated with this field ranges from 1/1/96 to 12/31/98. The data is filtered on Dimension1, which keeps 1-URGENT.

125 marks   1 row by 25 columns   SUM(WRevChange): 32,384,677.00

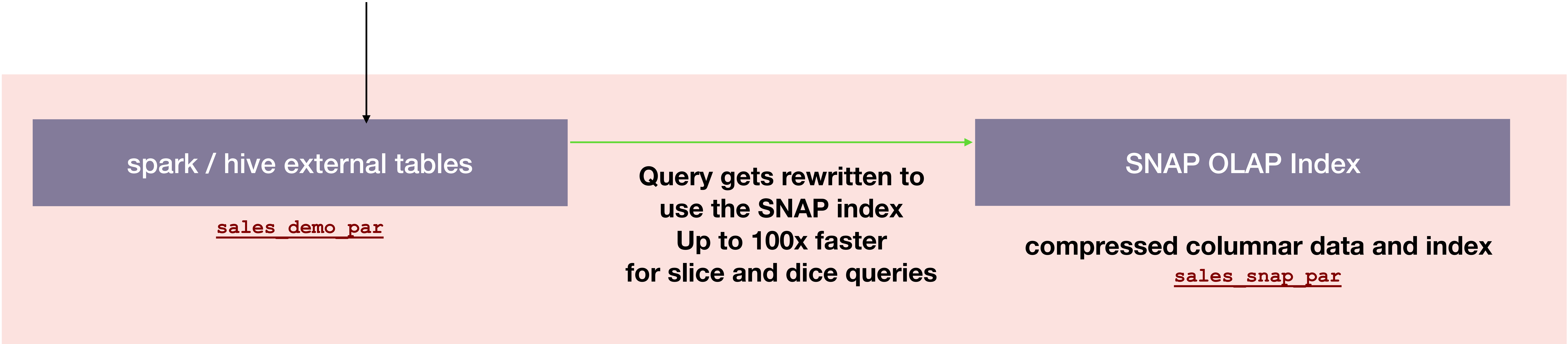
adjust discount to simulate how much would revenue have changed if discount was X% less

X is adjustable through the slider

additional filters are added 0 example - for this what-if analysis exclude a given market segment(drop down param)

What IF

```
SELECT    `sales_demo_par`.`c_mktsegment` AS `c_mktsegment`,
          `sales_demo_par`.`c_nation`     AS `c_nation`,
          `sales_demo_par`.`c_region`     AS `c_region`,
         sum(IF((`sales_demo_par`.`c_mktsegment` = 'AUTOMOBILE'),0,((`sales_demo_par`.`l_extendedprice` * (1 -
 (`sales_demo_par`.`l_discount` * 0.63))) - (`sales_demo_par`.`l_extendedprice` * (1 -
 `sales_demo_par`.`l_discount`))))) AS `sum_revenue_change__copy__ok`
FROM      `demo`.`sales_demo_par` `sales_demo_par`
WHERE    (((cast(`sales_demo_par`.`l_shipdate` AS timestamp) >= cast('1996-01-01 00:00:00' AS timestamp))
         AND      (cast(`sales_demo_par`.`l_shipdate` AS timestamp) < cast('1999-01-01 00:00:00' AS timestamp)))
         AND      (sales_demo_par`.`c_mktsegment` IN ('AUTOMOBILE','FURNITURE')))
GROUP BY `sales_demo_par`.`c_mktsegment`,
          `sales_demo_par`.`c_nation`,
          `sales_demo_par`.`c_region`
```



spark / hive external tables
sales_demo_par
SNAP OLAP Index
compressed columnar data and index
sales_snap_par

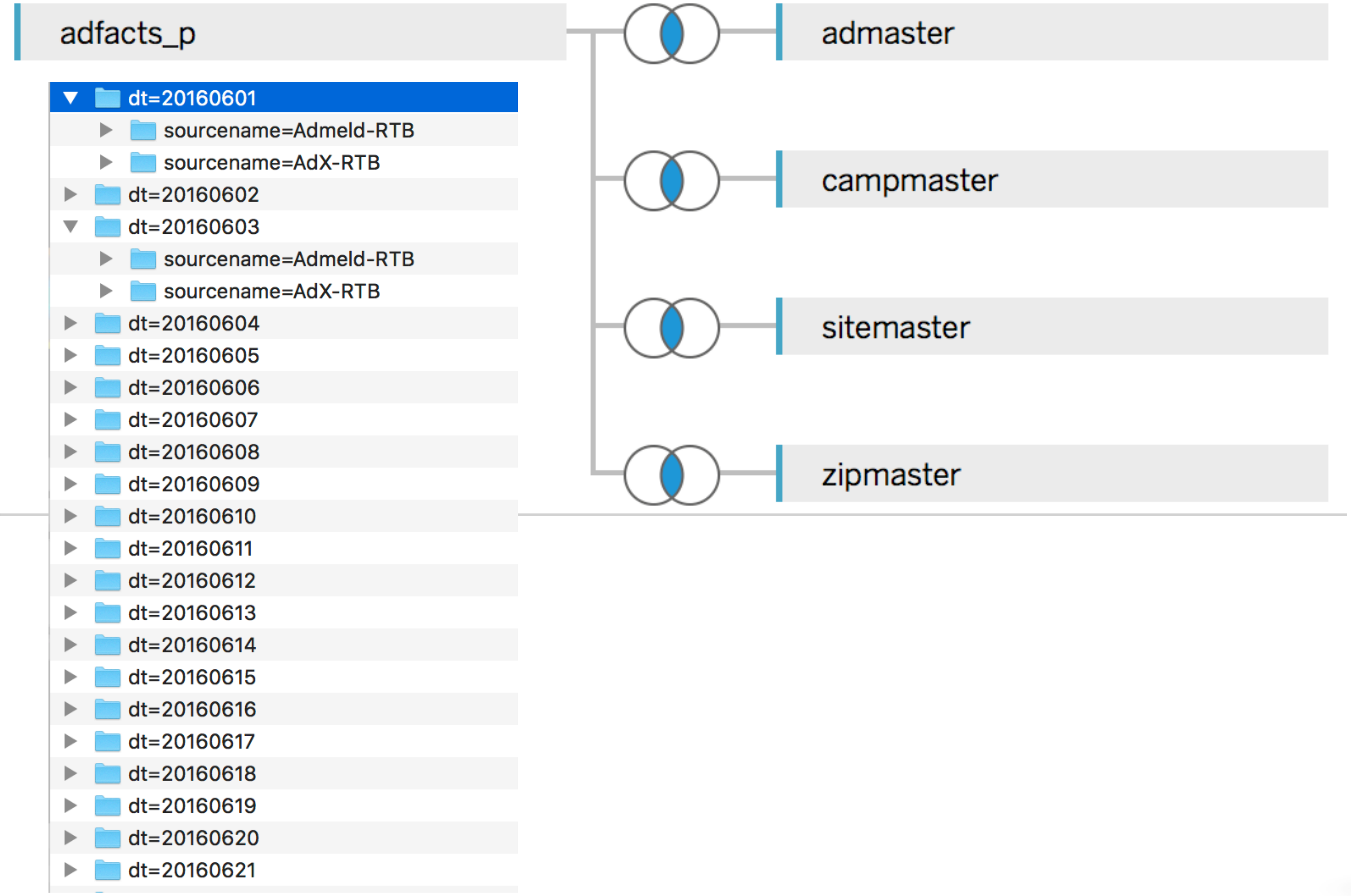
Query gets rewritten to use the SNAP index Up to 100x faster for slice and dice queries

physical data and index in HDFS /S3

```
  0 Jan 27 15:33 _SUCCESS
884 Jan 29 08:04 p_year=1992
884 Jan 29 08:10 p_year=1993
884 Jan 29 08:17 p_year=1994
884 Jan 29 08:23 p_year=1995
884 Jan 29 08:29 p_year=1996
884 Jan 29 08:36 p_year=1997
612 Jan 29 08:40 p_year=1998
```

defining star schemas

star schema indexing and querying



adfacts_p

- dt=20160601
  - sourcename=Admeld-RTB
  - sourcename=AdX-RTB
- dt=20160602
- dt=20160603
  - sourcename=Admeld-RTB
  - sourcename=AdX-RTB
- dt=20160604
- dt=20160605
- dt=20160606
- dt=20160607
- dt=20160608
- dt=20160609
- dt=20160610
- dt=20160611
- dt=20160612
- dt=20160613
- dt=20160614
- dt=20160615
- dt=20160616
- dt=20160617
- dt=20160618
- dt=20160619
- dt=20160620
- dt=20160621

admaster

campmaster

sitemaster

zipmaster

1

```
create star schema on adfacts_p as
many_to_one join of adfacts_p
with admaster on advertiser_id = advi
many_to_one join of adfacts_p
with sitemaster on site_id = siteidmaster
many_to_one join of adfacts_p
with campmaster on campaign_id = campid
many_to_one join of adfacts_p
with zipmaster on zip = zipcode
```

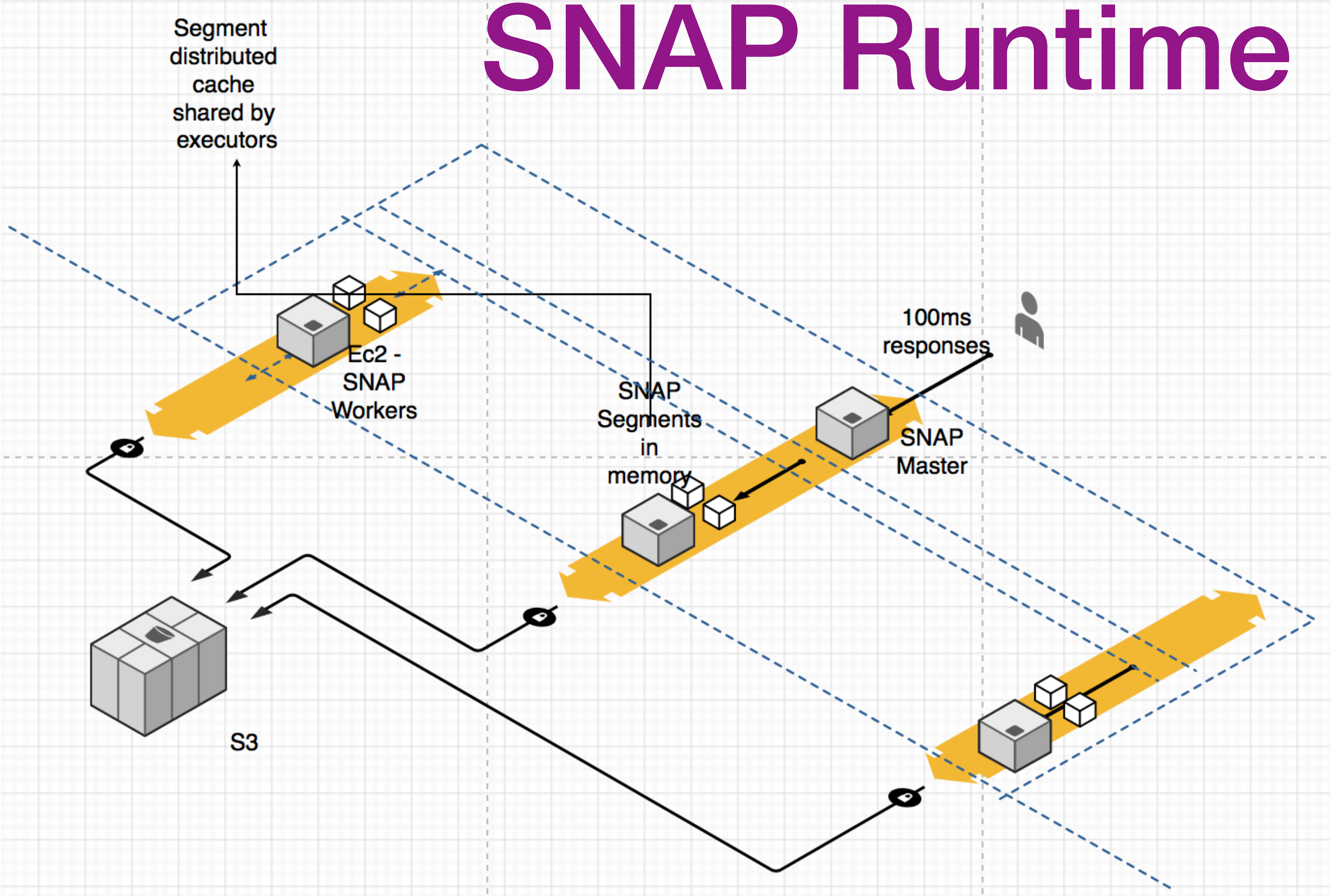
2

```
create olap index adstarindexp on adfacts_p
dimension acode is nullable nullvalue "NA"
dimensions
"hr,media_type_id,mobile_platform,model,sitenam
e,advid,advname,campid,campname,zipcode,type,
primarycity,state,county,timezone,acode,country,e
stpop"
metrics "revenue,numimpr,numclicks"
OPTIONS (path "/SNAP/adstarindexP"
nonAggregateQueryHandling
"push_project_and_filters",avgSizePerPartition
"40mb",
avgNumRowsPerPartition "10000",
preferredSegmentSize "20mb",
rowFlushBoundary "1000")
partition by dt, sourcename;
```

Star Schema

- Once the star schema is defined the process is the same as before for creating an olap index
    - Choose dimensions and metrics across all table columns in the join graph
    - Choose partitioning scheme if needed based on the fact table partitioning
    - Insert into the olap index

SNAP Runtime



- Example - SNAP on S3
- No Hadoop/YARN
- Standalone can support billions of rows and terabytes of data
- SNAP Segments downloaded from s3 to local node distributed cache
- Queries served from segments in-memory
- As new data arrived in S3 SNAP downloads the new segments on request