The Educational Conference For Oracle Techology Users

ODTUG Kscope23

aurora, co june 25-29

Welcome

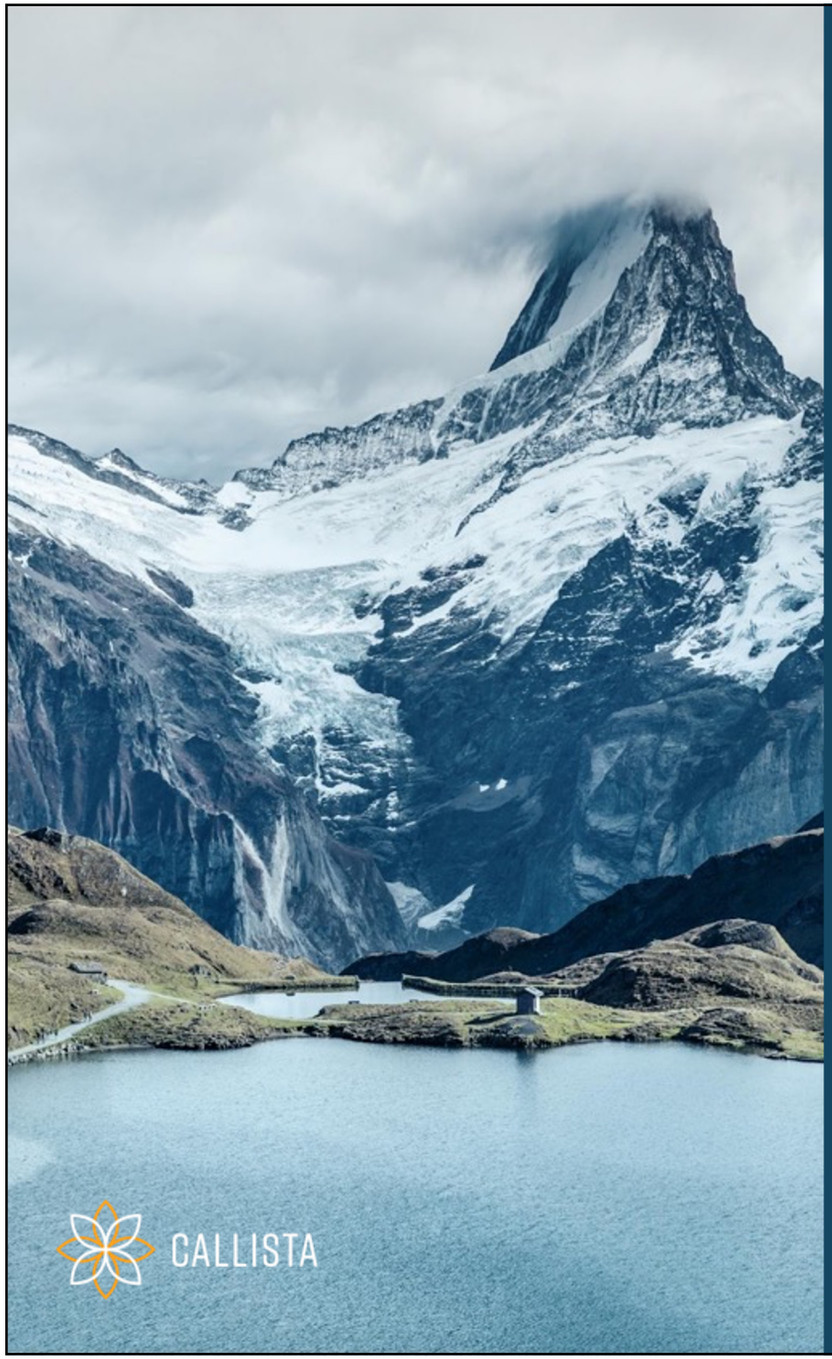

DRIVEN BY EXCELLENCE
Make Your Oracle Data Warehouse Fast:
Comparing Performance Features
Dani Schnider
CALLISTA

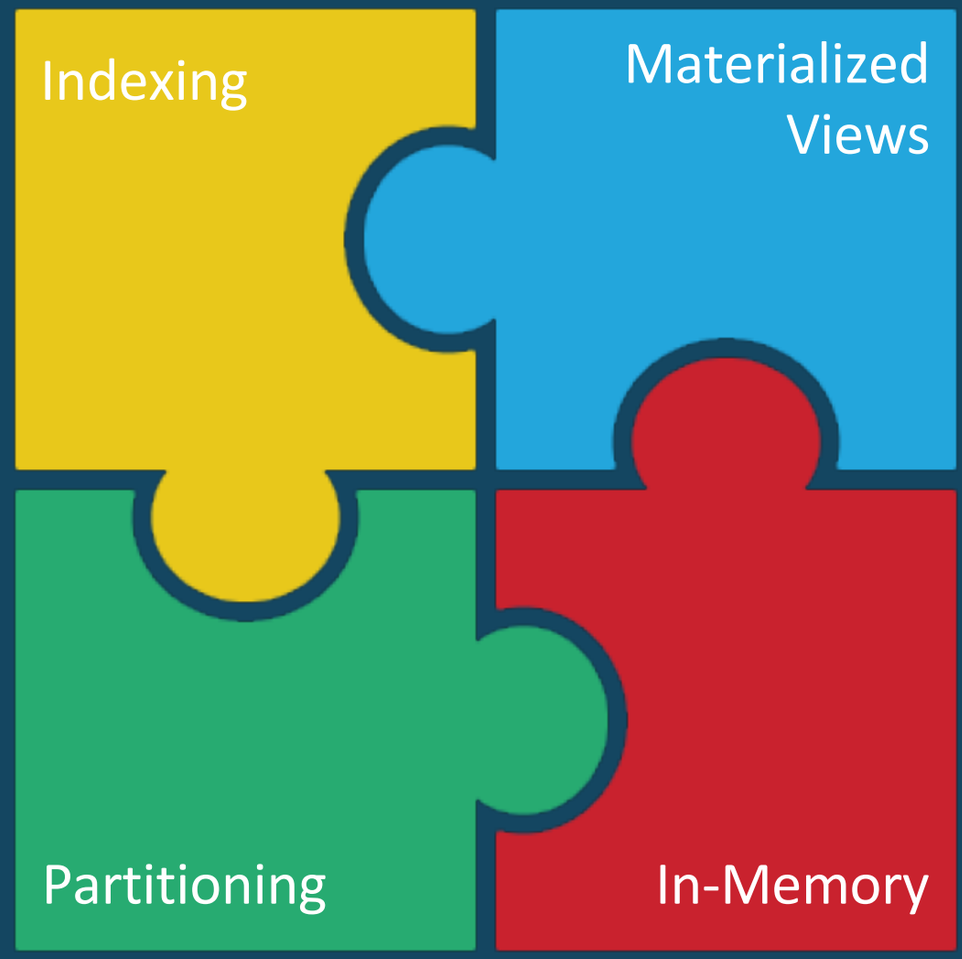
About me

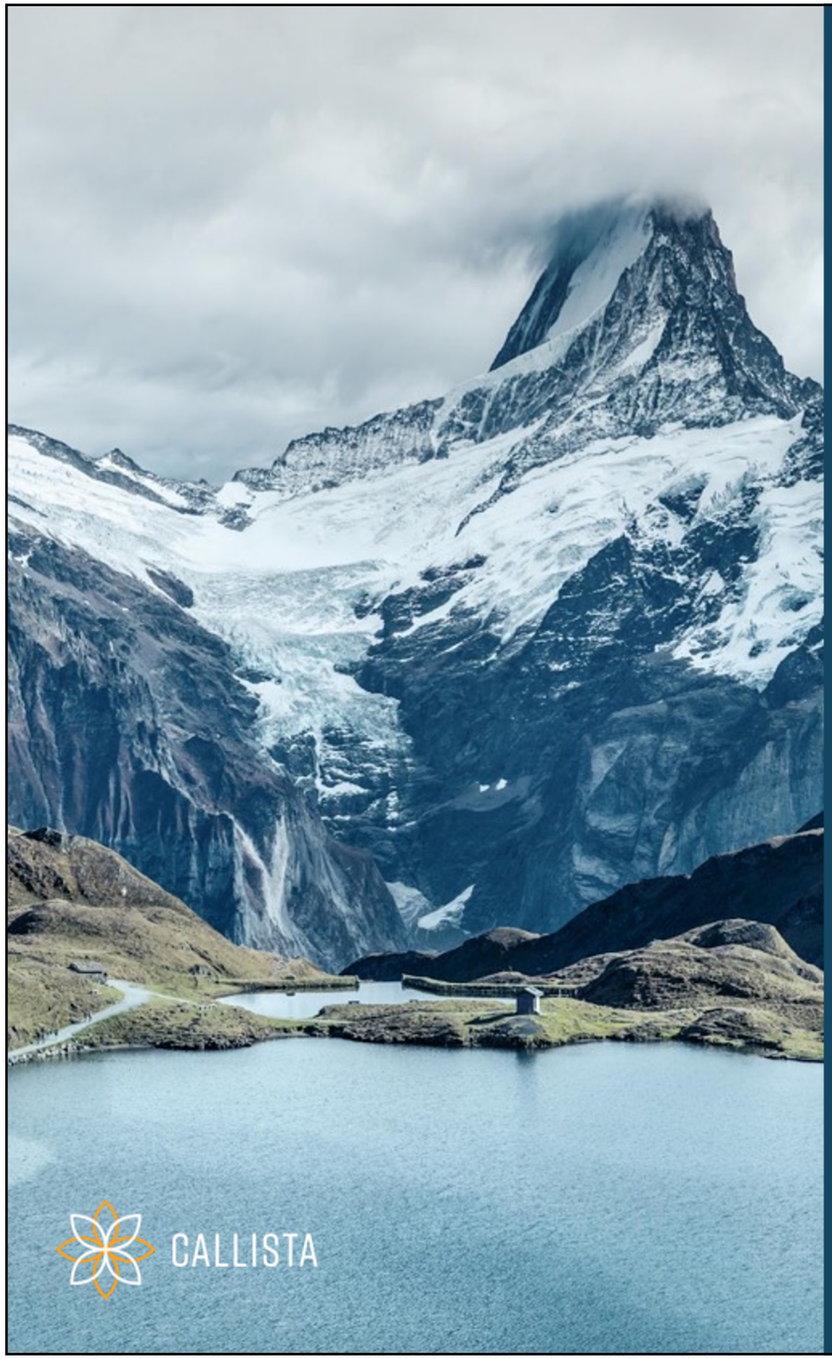
Dani Schnider

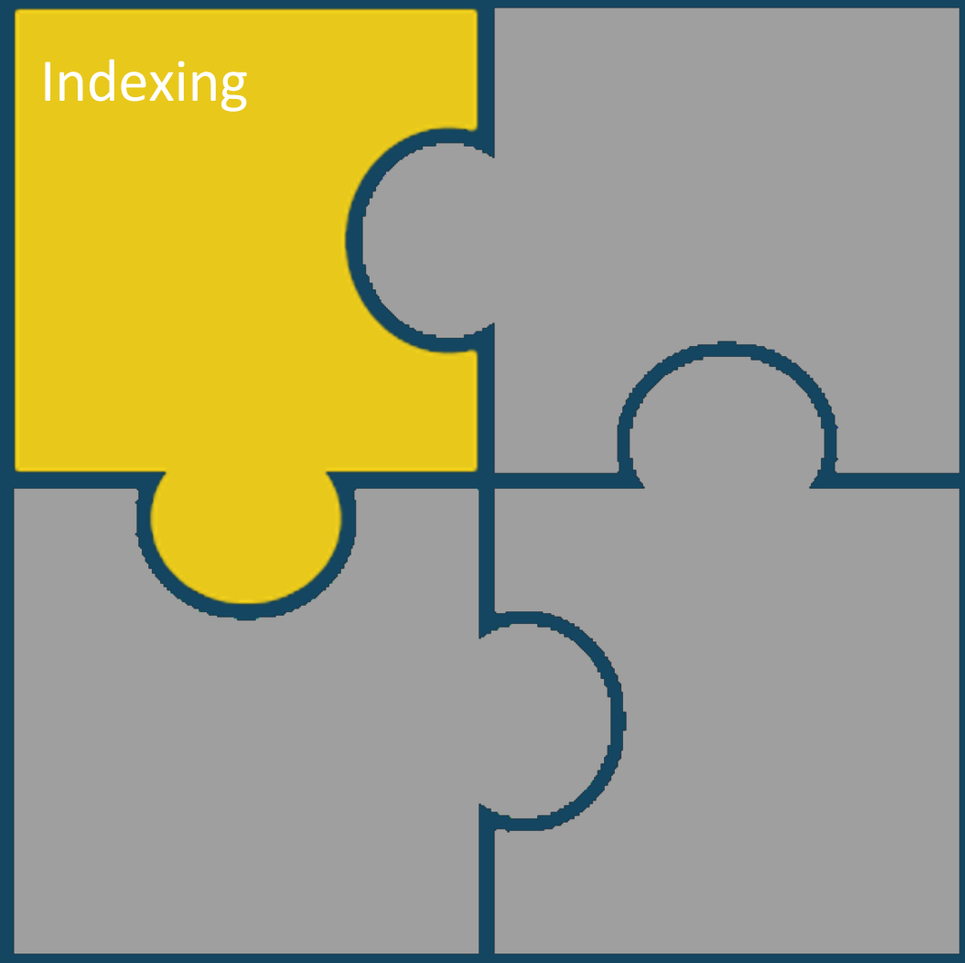
- Working for Callista
- Oracle ACE Director
- Member of Symposium 42
- Hobby: Craft Beer Brewing

@dani_schnider danischnider.wordpress.com www.linkedin.com/in/danischnider/

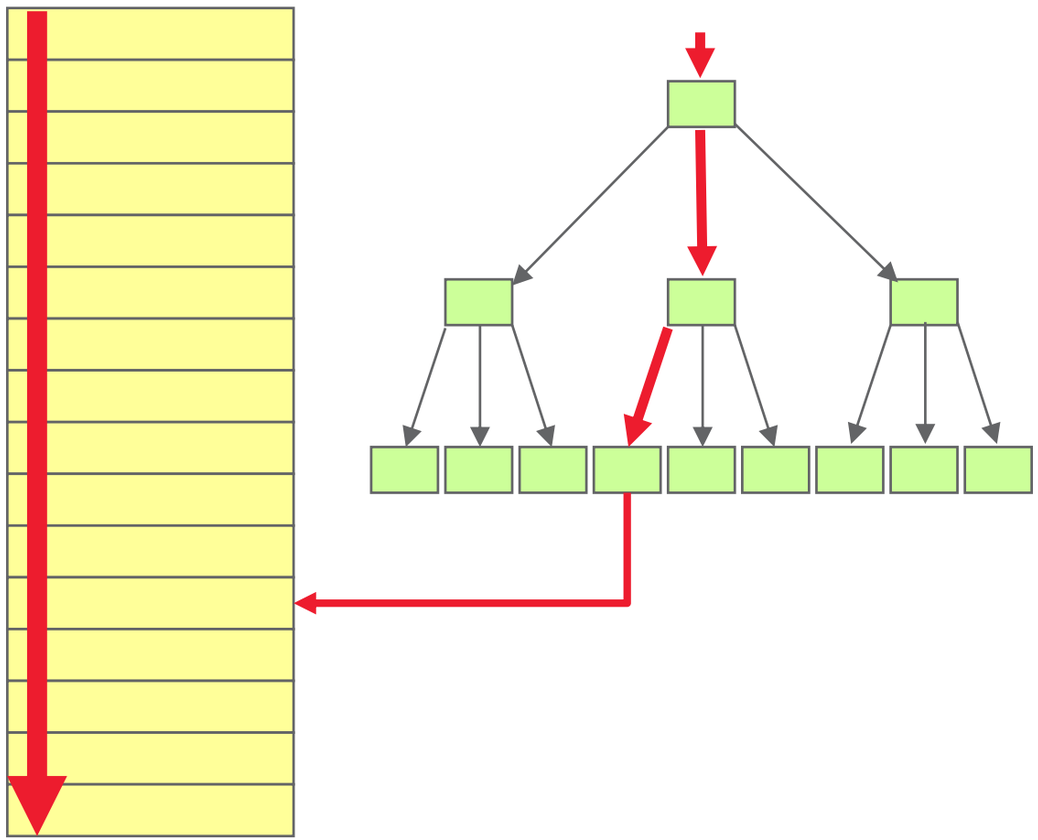

CALLISTA


Indexing
Materialized Views
Partitioning
In-Memory


CALLISTA


Indexing

Full Table Scan vs. Index Scan



Full Table Scan

- Efficient for weak selectivity
- High percentage of data

Index Scan

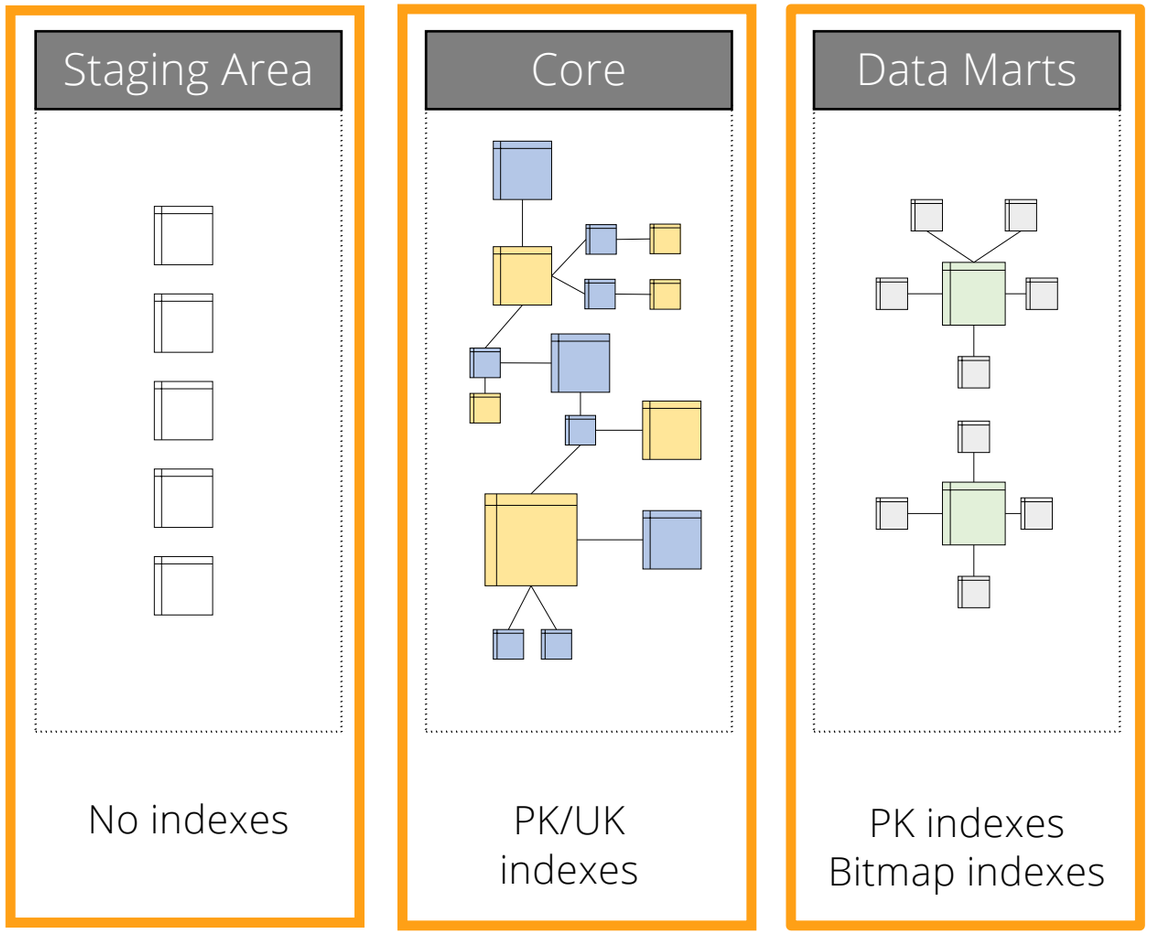
- Efficient for strong selectivity
- Small percentage of data

Typical for Data Warehouse

- High percentage (often 100%) of data in ETL jobs
- Aggregations on high volumes of data, few filter restrictions

CALLISTA

Indexing Data Warehouse Layers



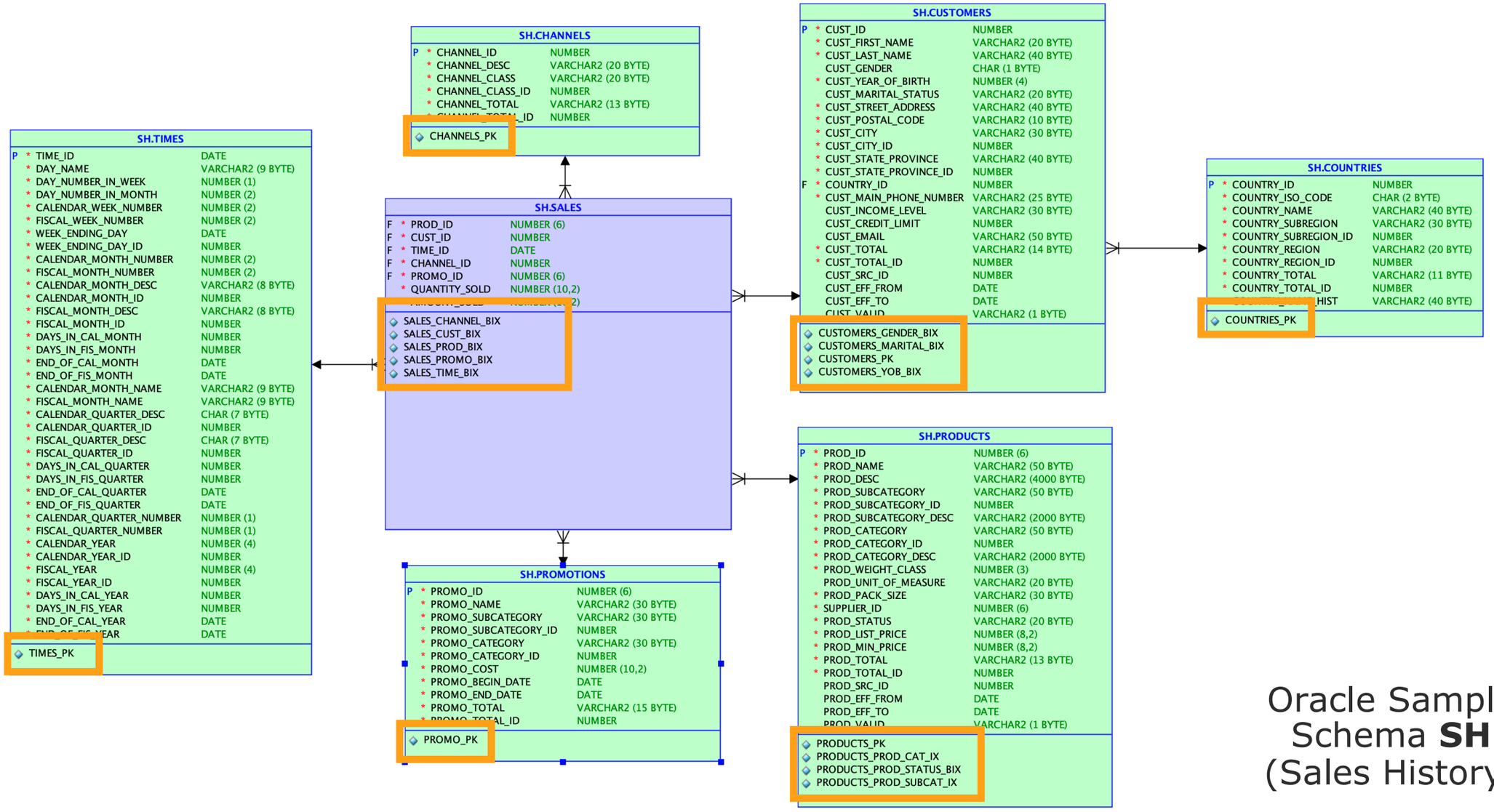
Dimension tables

- Primary key (unique index)
- Bitmap indexes on filter columns (optional)

Fact tables

- Bitmap indexes on dimension keys
- Typically no primary key

CALLISTA

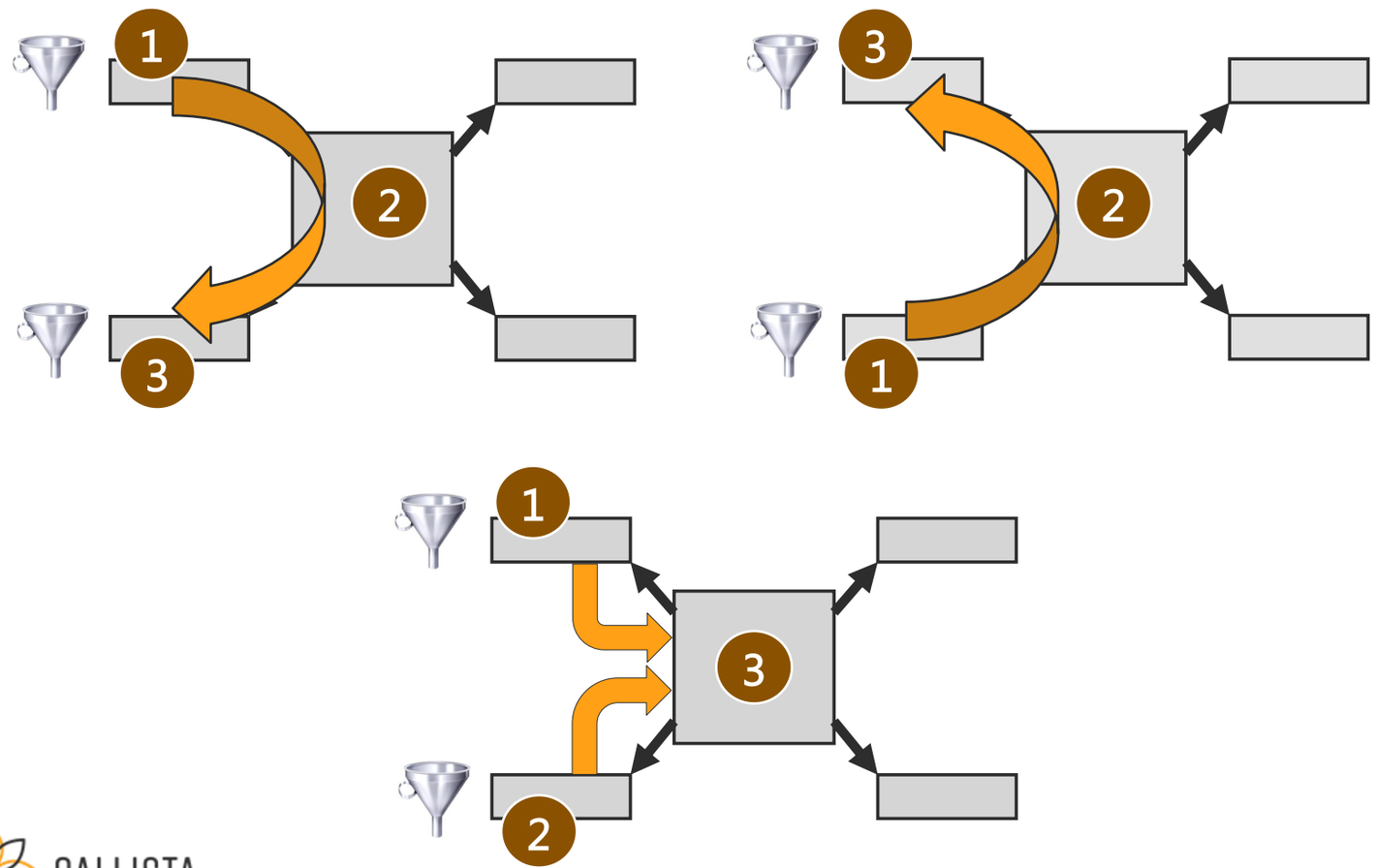


Oracle Sample Schema **SH** (Sales History)

CALLISTA

Download: https://github.com/oracle-samples/db-sample-schemas/releases

Joins in Star Schema



Typical queries on star schema

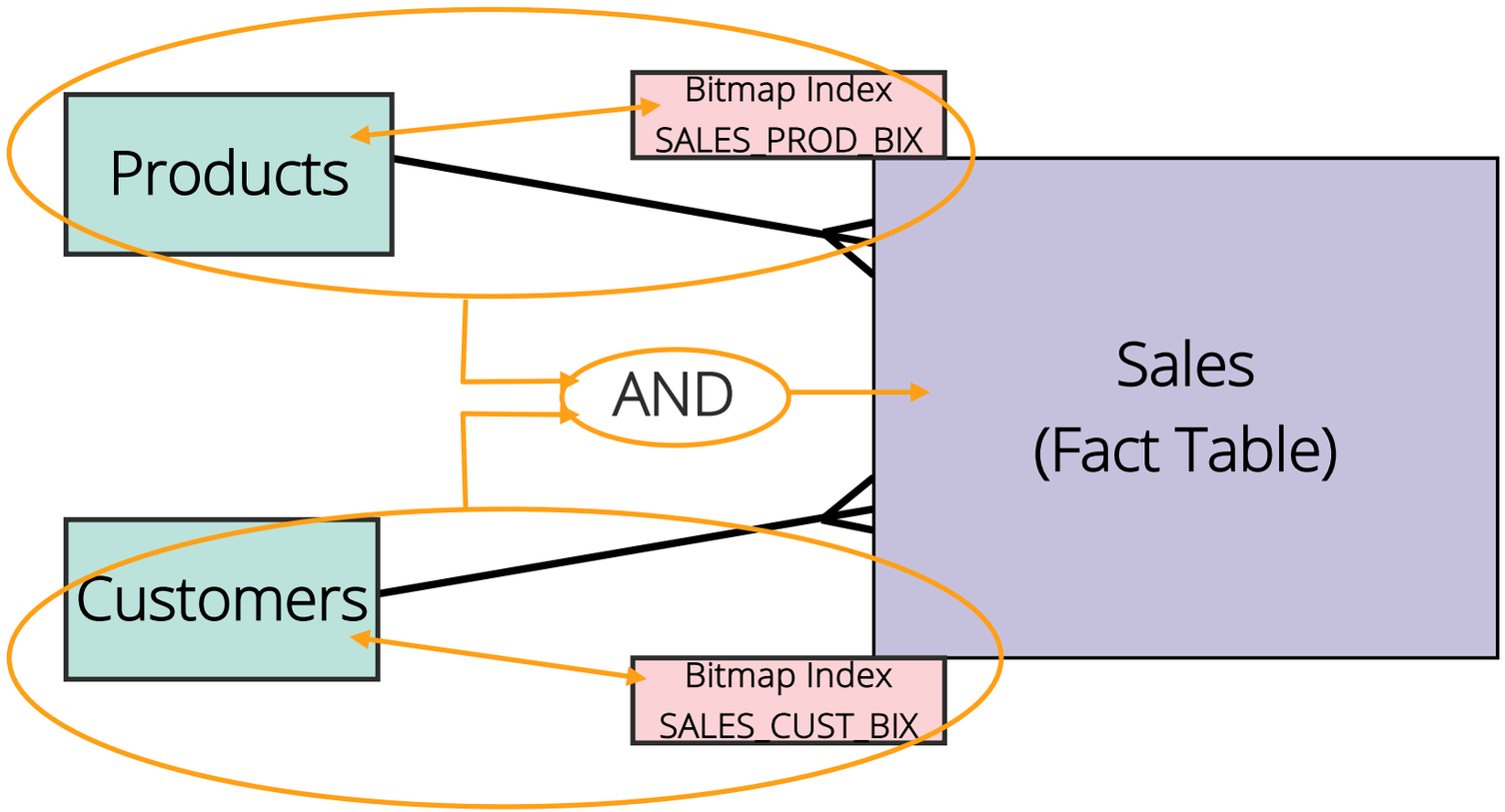
- Filter on multiple dimensions
- Facts are joined with all required dimensions

Challenge for optimizer

- Dimensions (with filters) should be read first
- No relationships between dimensions => fact table is joined too early

CALLISTA

Star Transformation



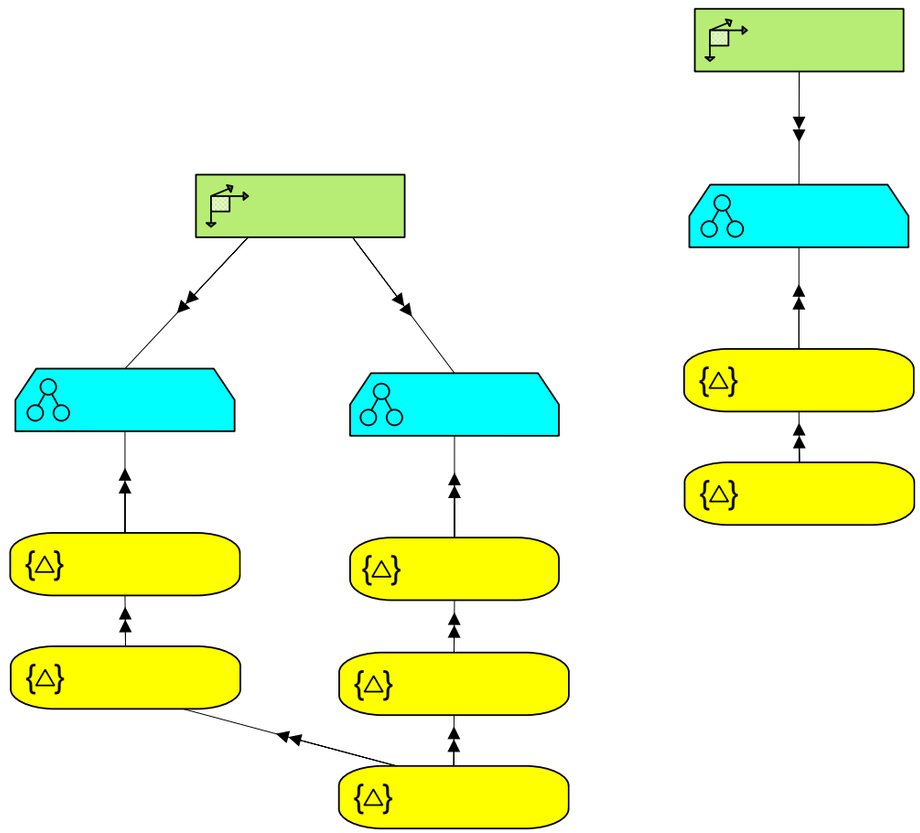
CALLISTA

Live Demo

Star Transformation

CALLISTA

Drill-Up on Star Schemas



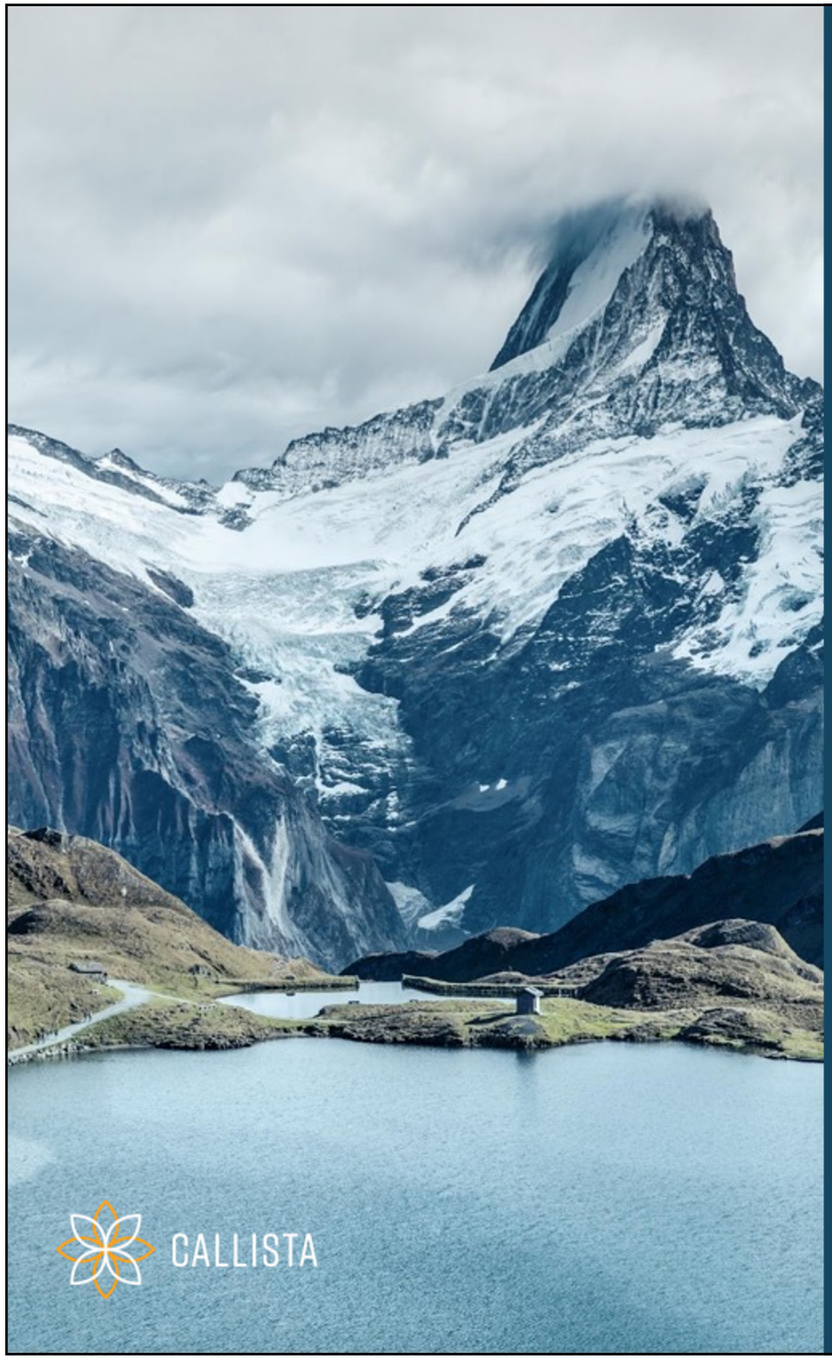
Drill-up on dimensions

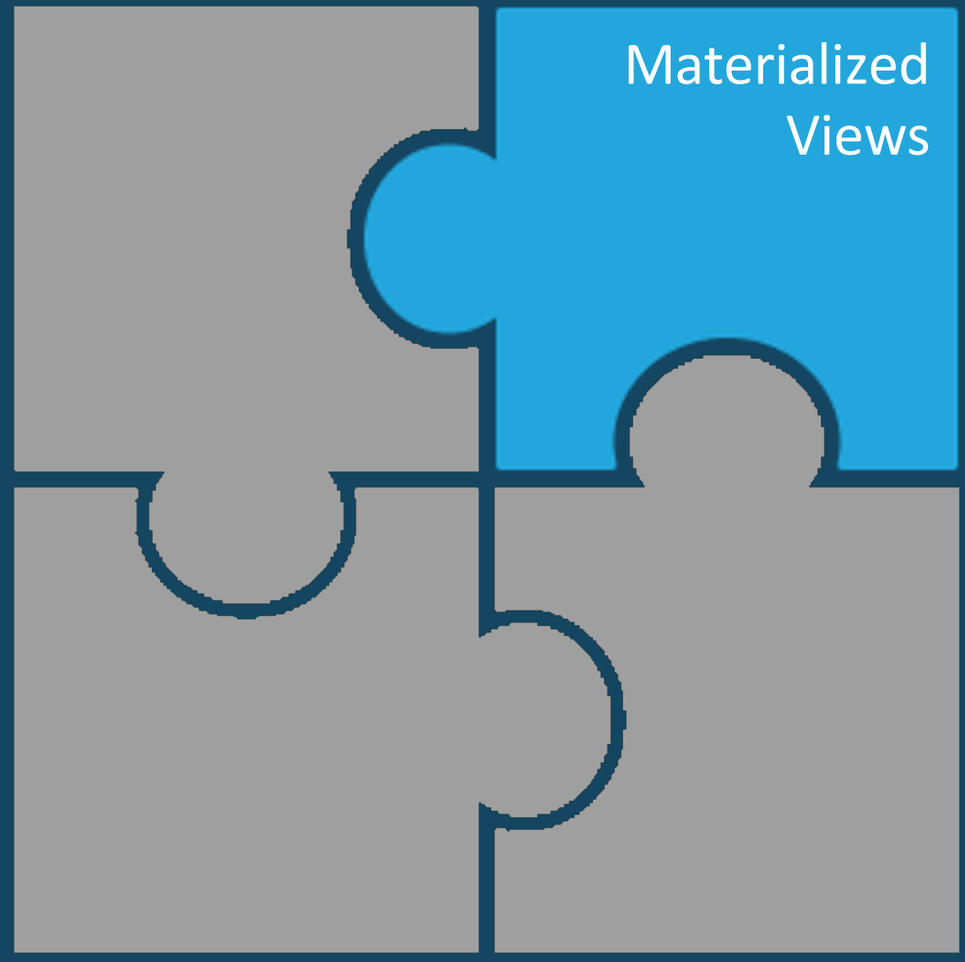
- Grouping on higher hierarchy levels of dimension
- Aggregation of facts along dimensions

Challenge for optimizer

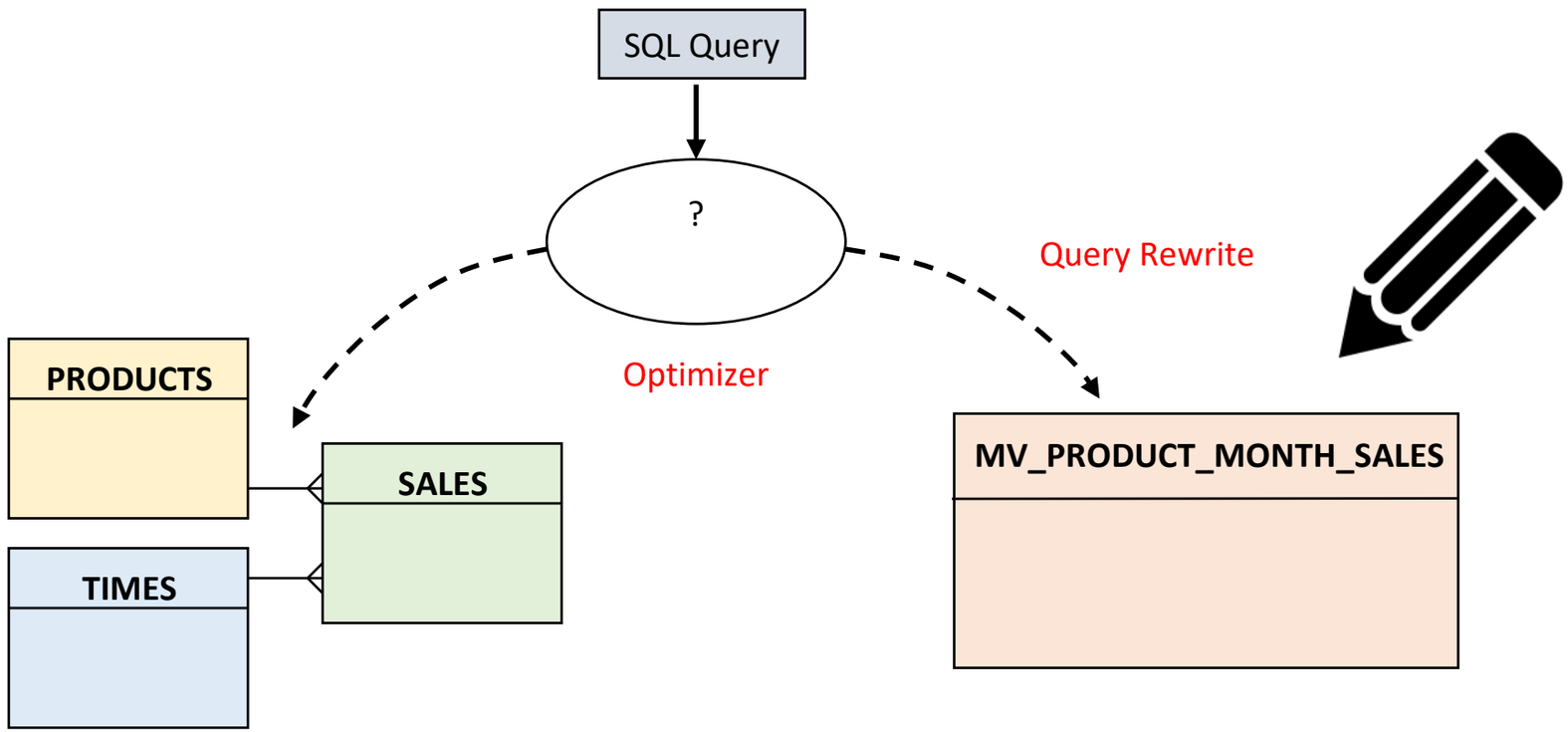
- For aggregation, many (or all) rows in fact table must be read
- How can a full table scan on entire fact table be avoided?
- Indexing does not help here

CALLISTA


CALLISTA


Materialized
Views

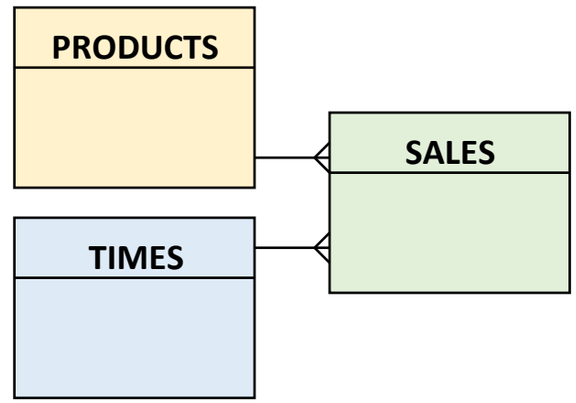
Materialized View and Query Rewrite



CALLISTA

Create Materialized View

```
CREATE MATERIALIZED VIEW mv_product_month_sales
ENABLE QUERY REWRITE
AS
SELECT t.calendar_month_desc
     , p.prod_name
     , SUM(s.amount_sold)
  FROM sales s
     , times t
     , products p
 WHERE t.time_id = s.time_id
   AND p.prod_id = s.prod_id
GROUP BY t.calendar_month_desc, p.prod_name
```

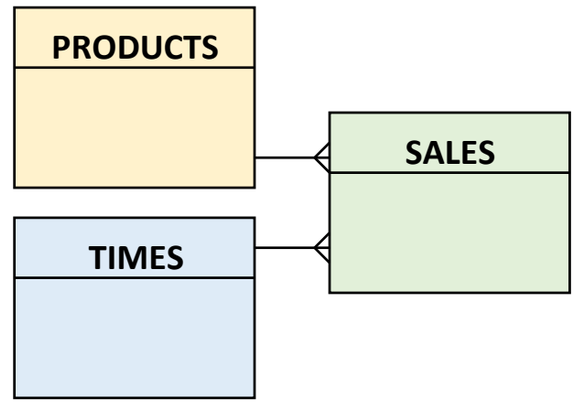


CALLISTA

Query Rewrite

Query Rewrite with Full Text Match

```
SELECT t.calendar_month_desc
     , p.prod_name
     , SUM(s.amount_sold)
  FROM sales s
     , times t
     , products p
 WHERE t.time_id = s.time_id
   AND p.prod_id = s.prod_id
GROUP BY t.calendar_month_desc, p.prod_name
```




```
-------------------------------------------------------------------
| Id  | Operation                          | Name                   |
-------------------------------------------------------------------
|   0 | SELECT STATEMENT                   |                        |
|   1 |  MAT_VIEW REWRITE ACCESS FULL| MV_PRODUCT_MONTH_SALES |
-------------------------------------------------------------------
```

MV_PRODUCT_MONTH_SALES

CALLISTA

Materialized Views in DWH

Materialized Views on Star Schema

- Joins between fact table and dimension tables
- Pre-aggregated facts on often used hierarchy levels
- Query rewrite on higher hierarchy levels
- Refresh at the end of ETL job

Further information:

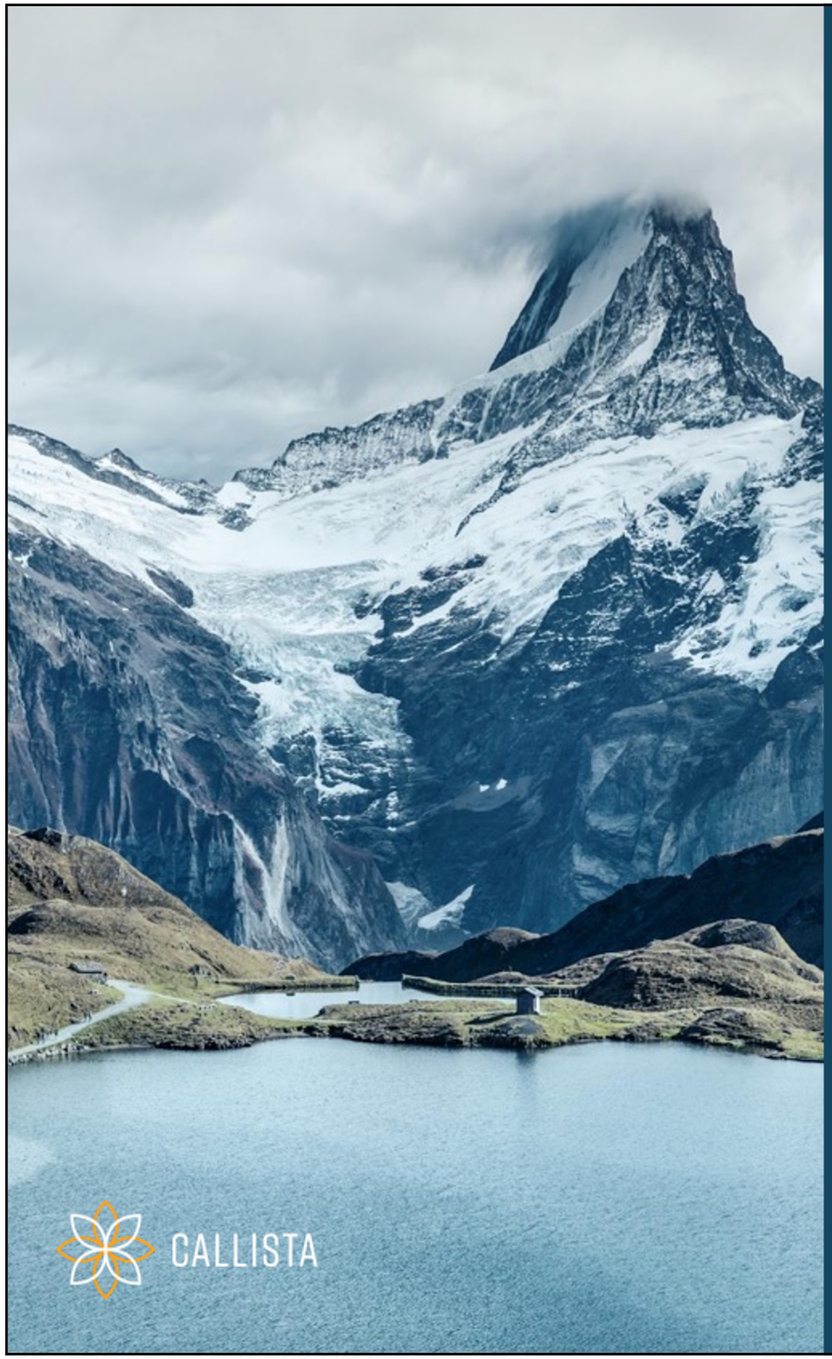
- Query Rewrite: The Supreme League of Materialized Views
- Design Tips for Query Rewrite

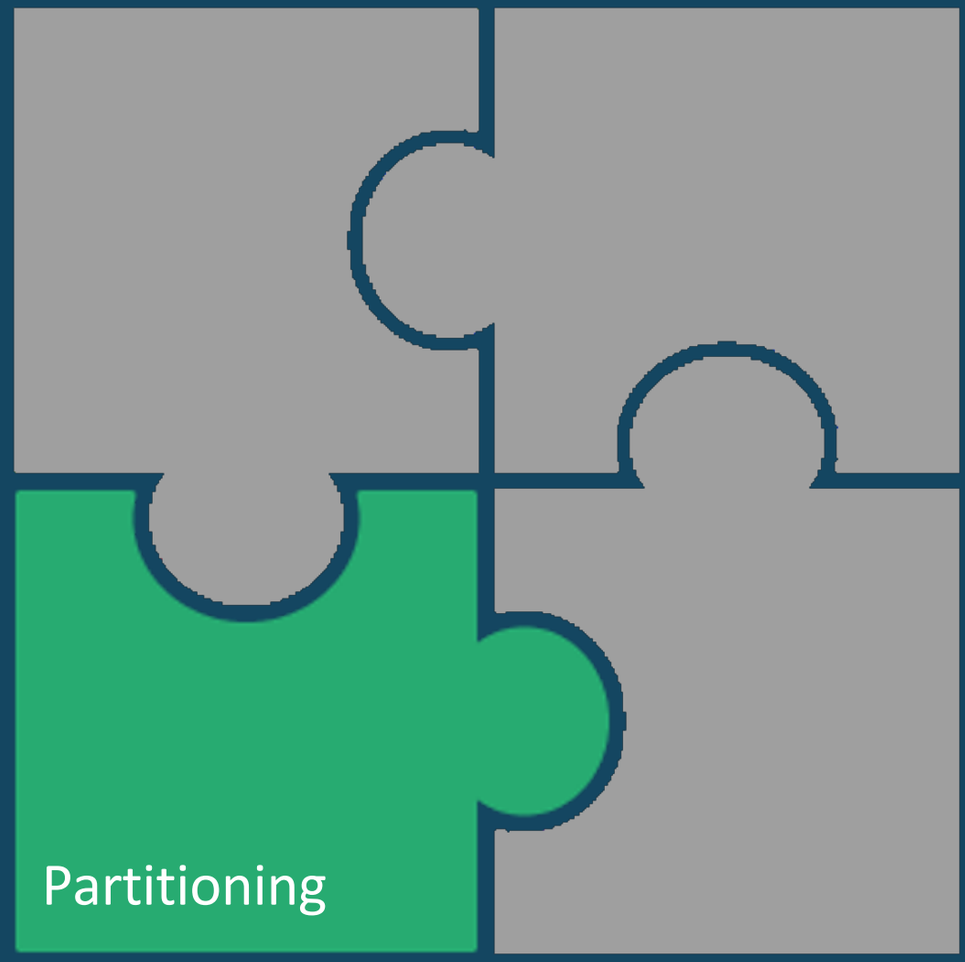
CALLISTA

Live Demo

Query Rewrite

CALLISTA

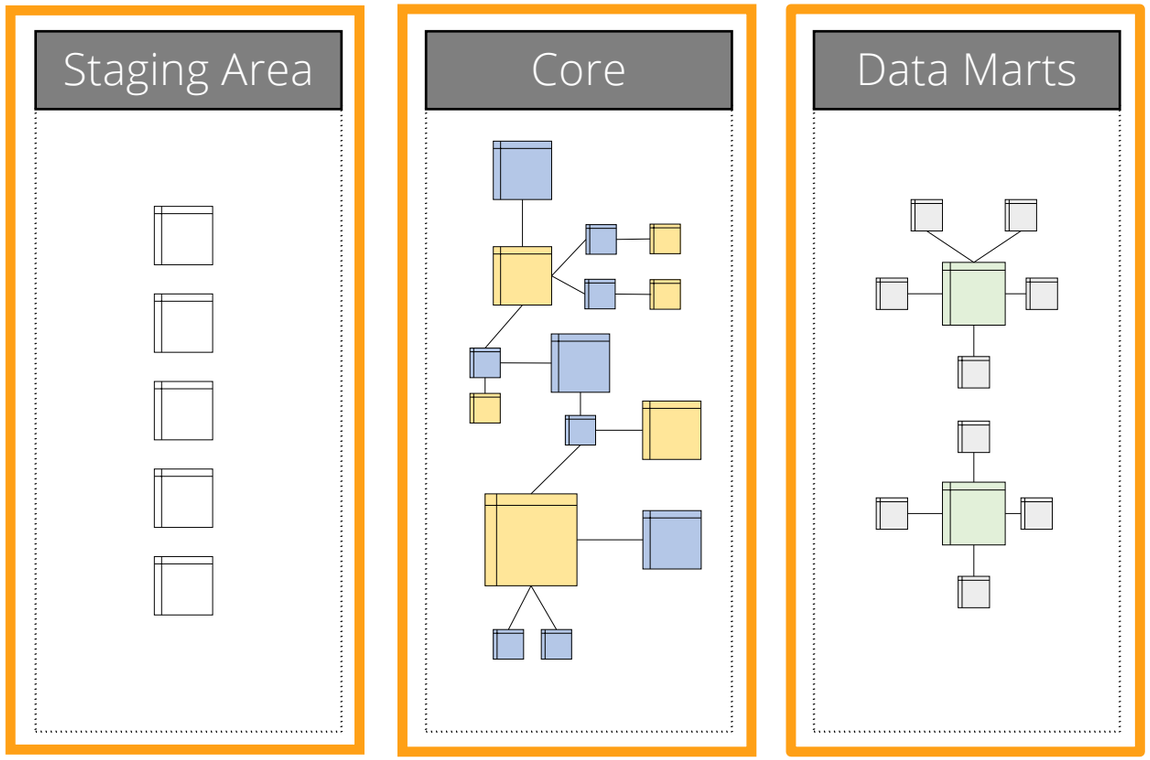

CALLISTA


Partitioning

Advantages of Partitioning

- Query Performance
  - Partition pruning
  - Partition-wise join
- ETL Performance
  - Parallelization of load jobs
  - Partition exchange loading
- Information Lifecyle Management
  - Moving time windows
  - Delete / archive outdated data
- Database Administration
  - Backups current partitions only
  - Compress historical partitions

Partitioning in Data Warehouse



Staging Area

- Partitioning can be useful if multiple versions of loads must be archived

Core

- Different partitioning strategies, mainly to improve ETL performance

Data Marts

- Partitioning fact tables on time dimension key

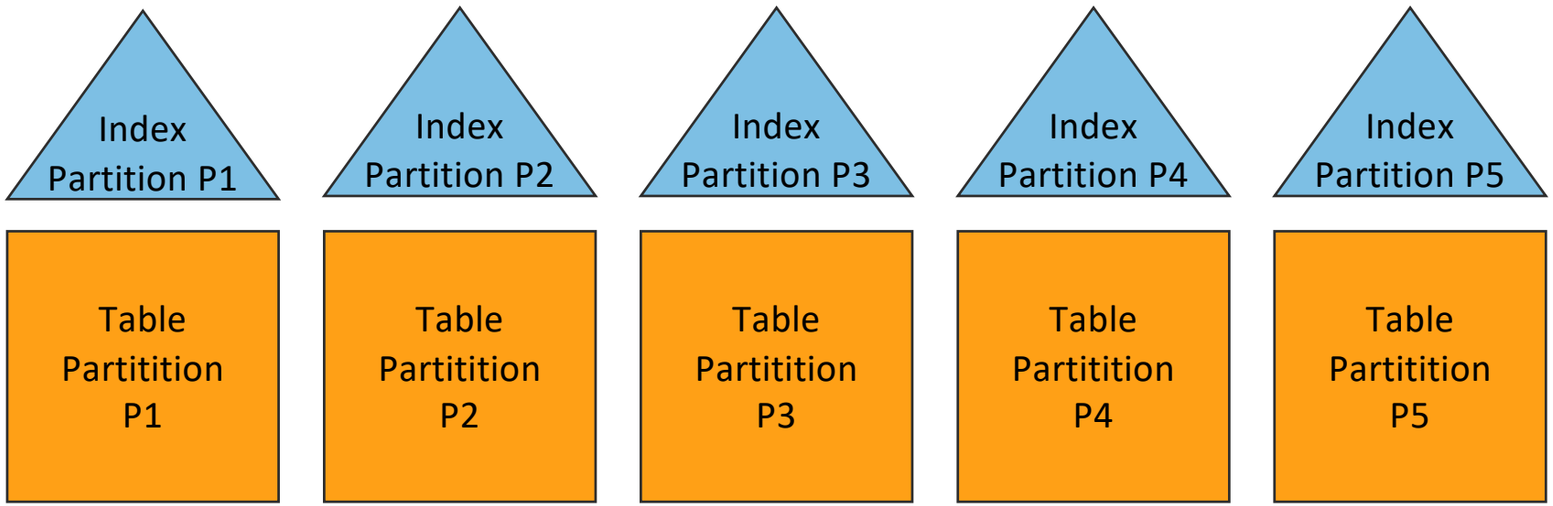
CALLISTA

Information Lifecycle Management

- RANGE or INTERVAL partitioning by date
- Outdated data can be archived/deleted easily

CALLISTA

Partitioning Indexes

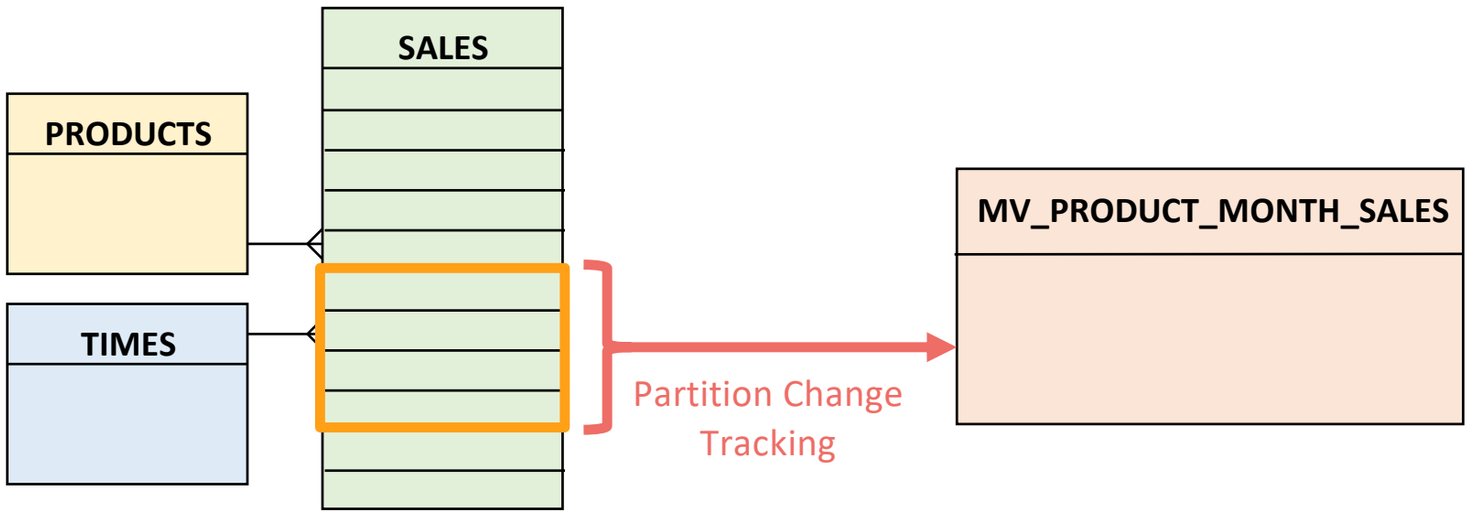


Indexes

- Use local indexes whenever possible
- Bitmap indexes are always local indexes

CALLISTA

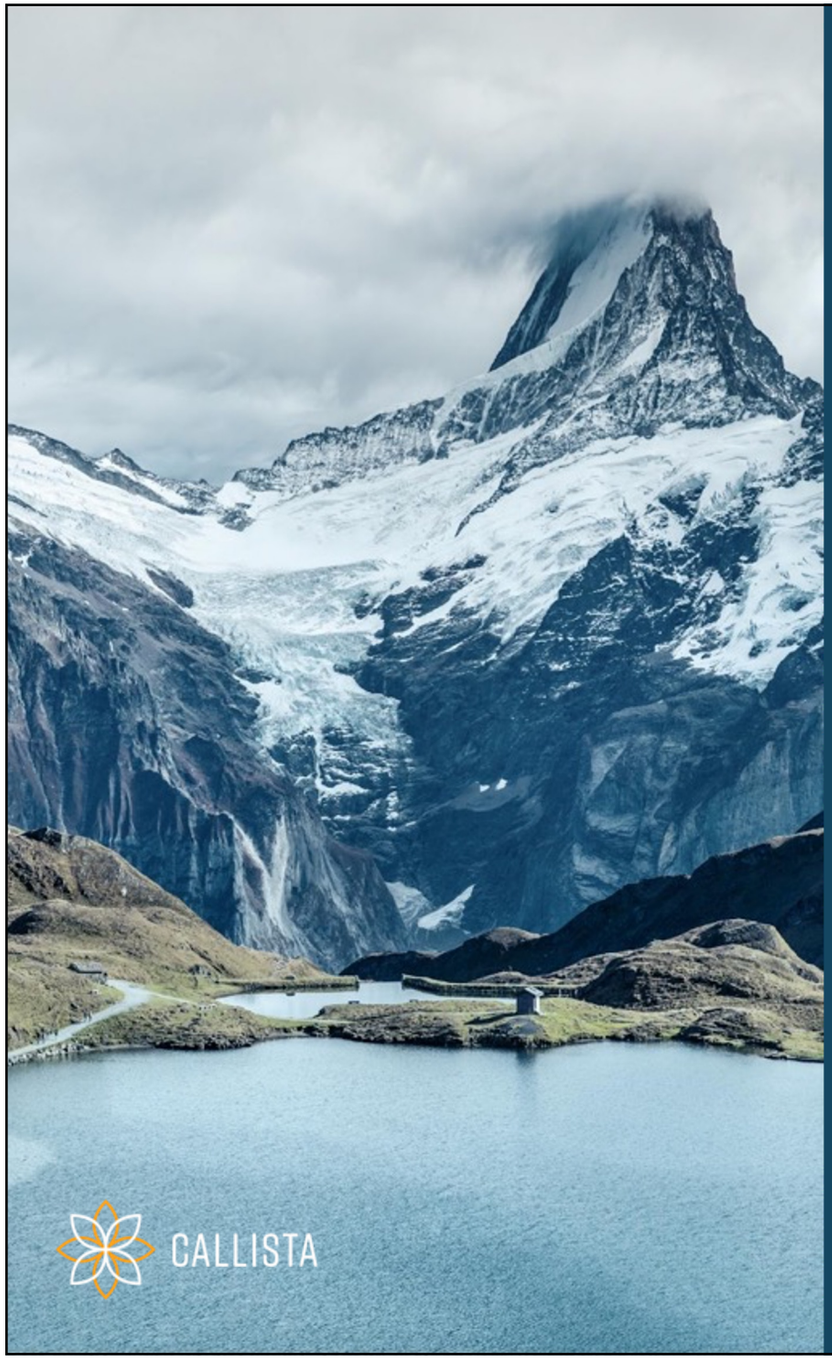
Partitioning Materialized Views

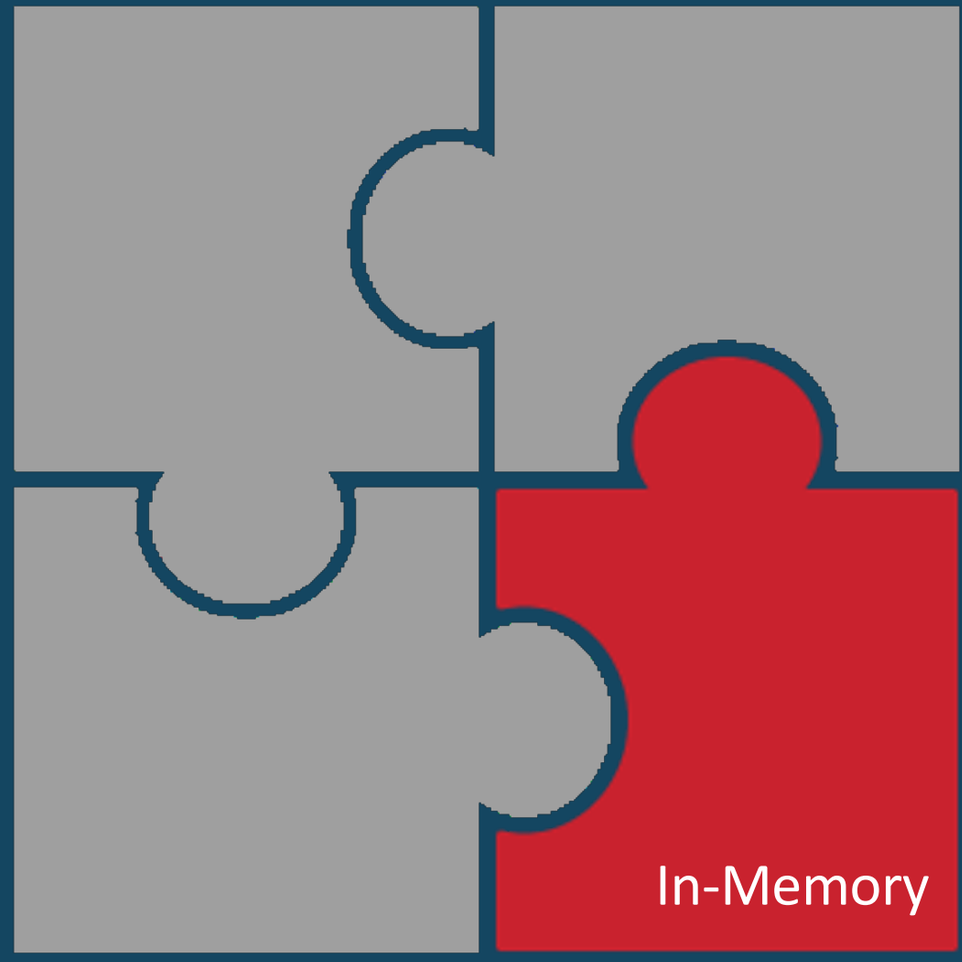


Materialized Views

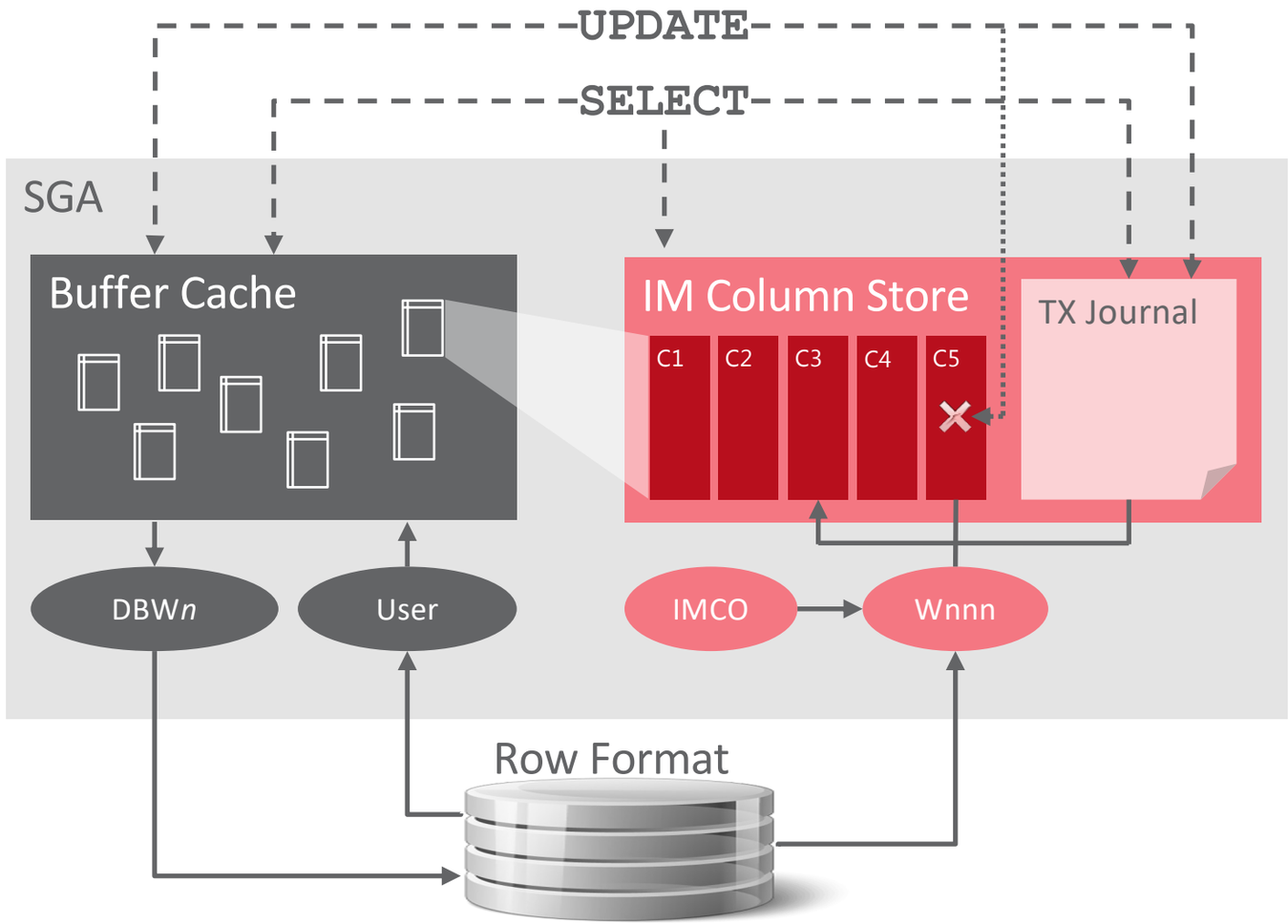
- Materialized views can be partitioned, too
- Partition Change Tracking (PCT) for refresh of materialized views on partitioned tables

CALLISTA


CALLISTA


In-Memory

Oracle Database In-Memory Architecture



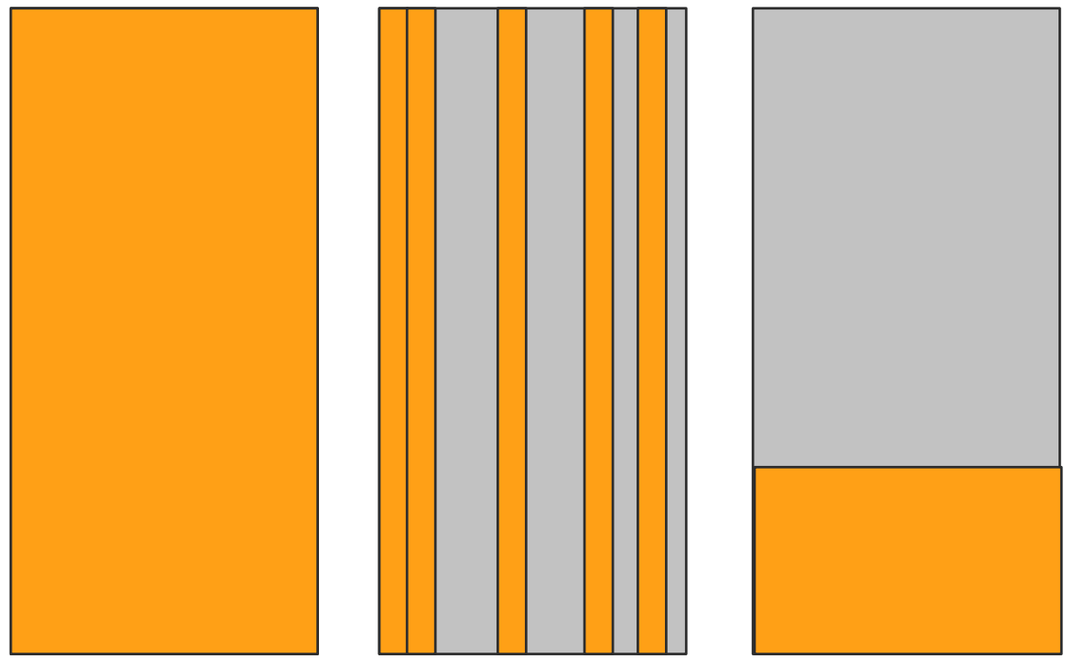
CALLISTA

In-Memory Configuration Options

```
ALTER TABLE sales
INMEMORY
```

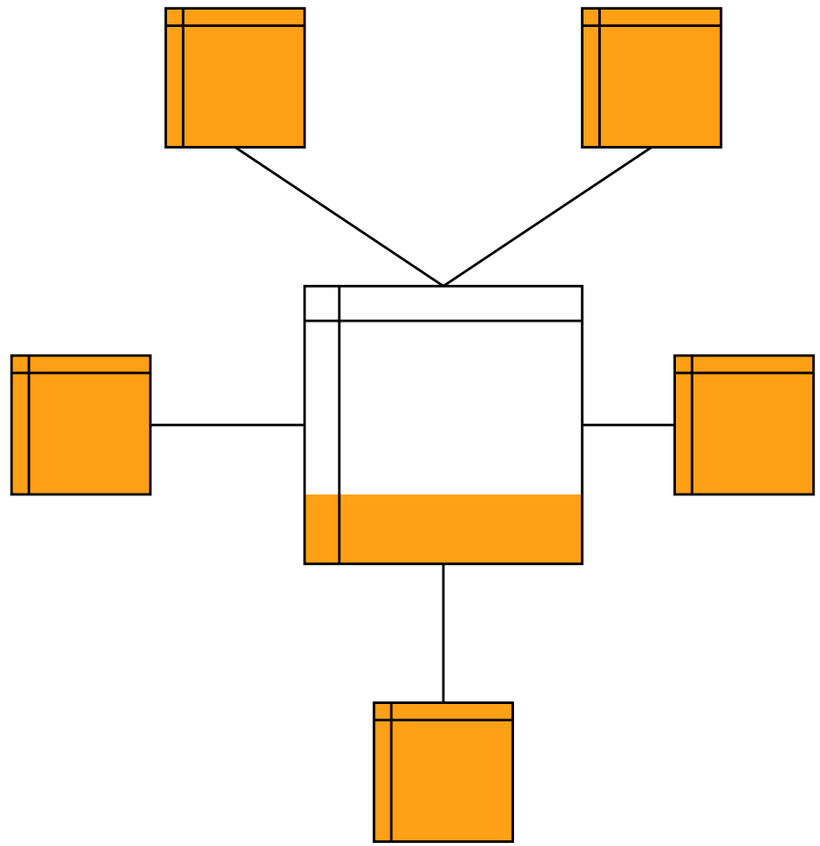
```
ALTER TABLE sales
INMEMORY NO INMEMORY
(cust_id,time_id)
```

```
ALTER TABLE sales MODIFY
PARTITION p_2020 INMEMORY
```



CALLISTA

In-Memory for Star Schema



Dimension tables

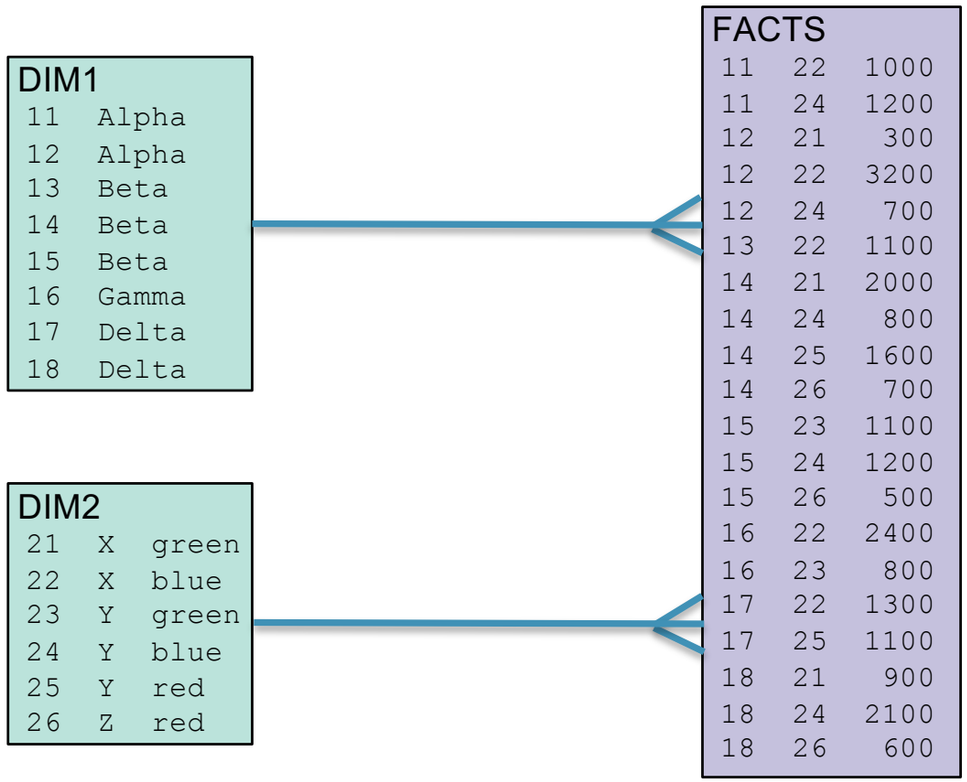
- All dimensions are completely populated to In-Memory column store

Fact tables

- Partitioned by date
- Newest (or often used) partitions populated to In-Memory column store
- Historical partitions not in In-Memory column store

CALLISTA

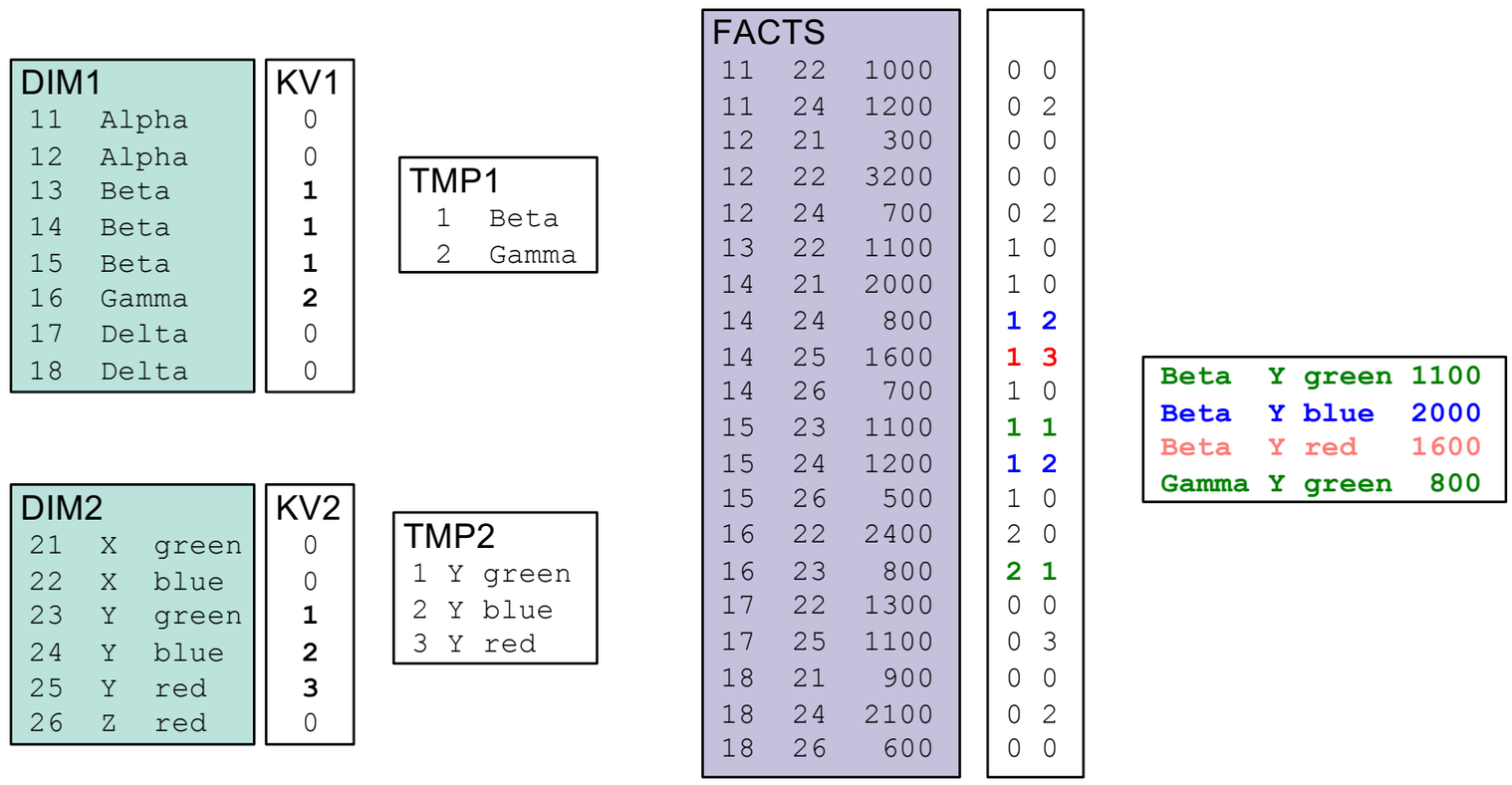
Vector Transformation




```
SELECT D1, D21, D22
     , SUM(FACTS.F)
  FROM FACTS
  JOIN DIM1 ON (...)
  JOIN DIM2 ON (...)
 WHERE D1 IN ('Beta',
              'Gamma')
   AND D21 = 'Y'
GROUP BY D1, D21, D22
```

CALLISTA

Vector Transformation

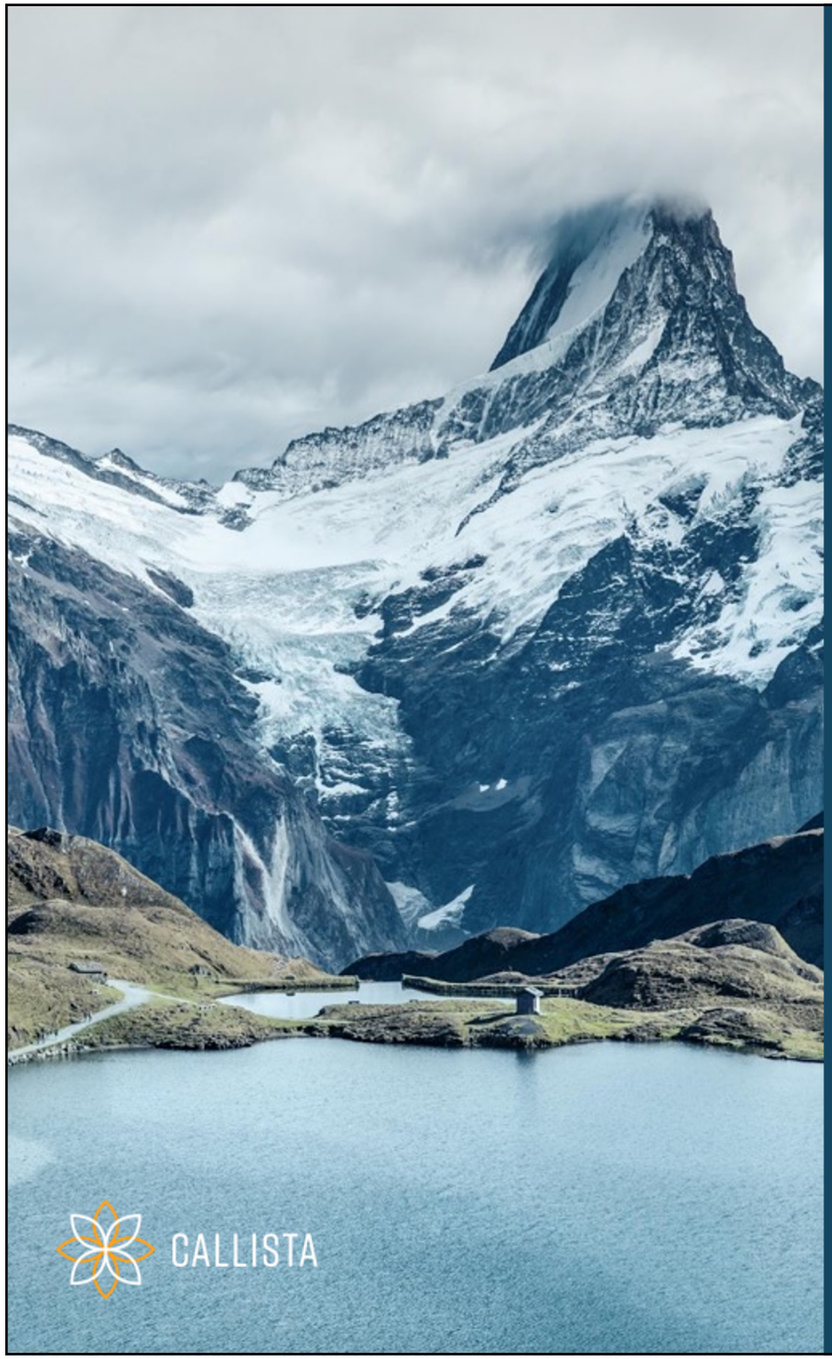


CALLISTA

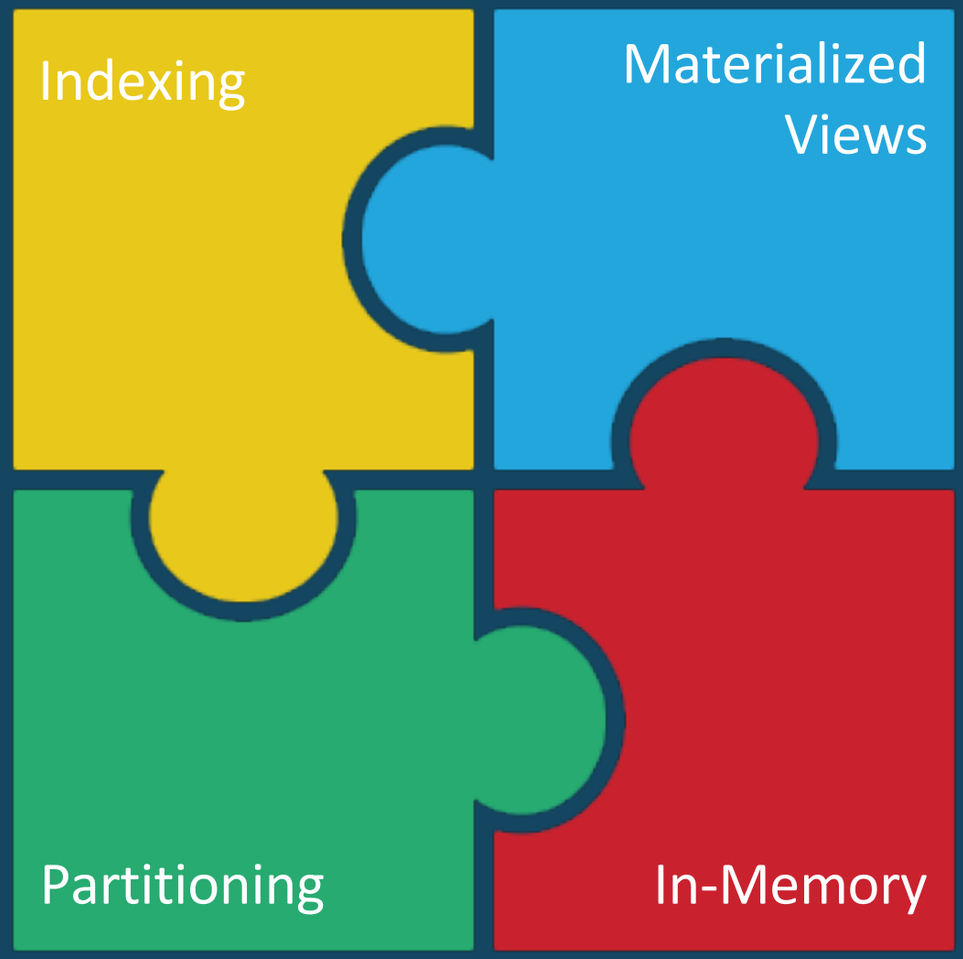
Live Demo

Vector Transformation

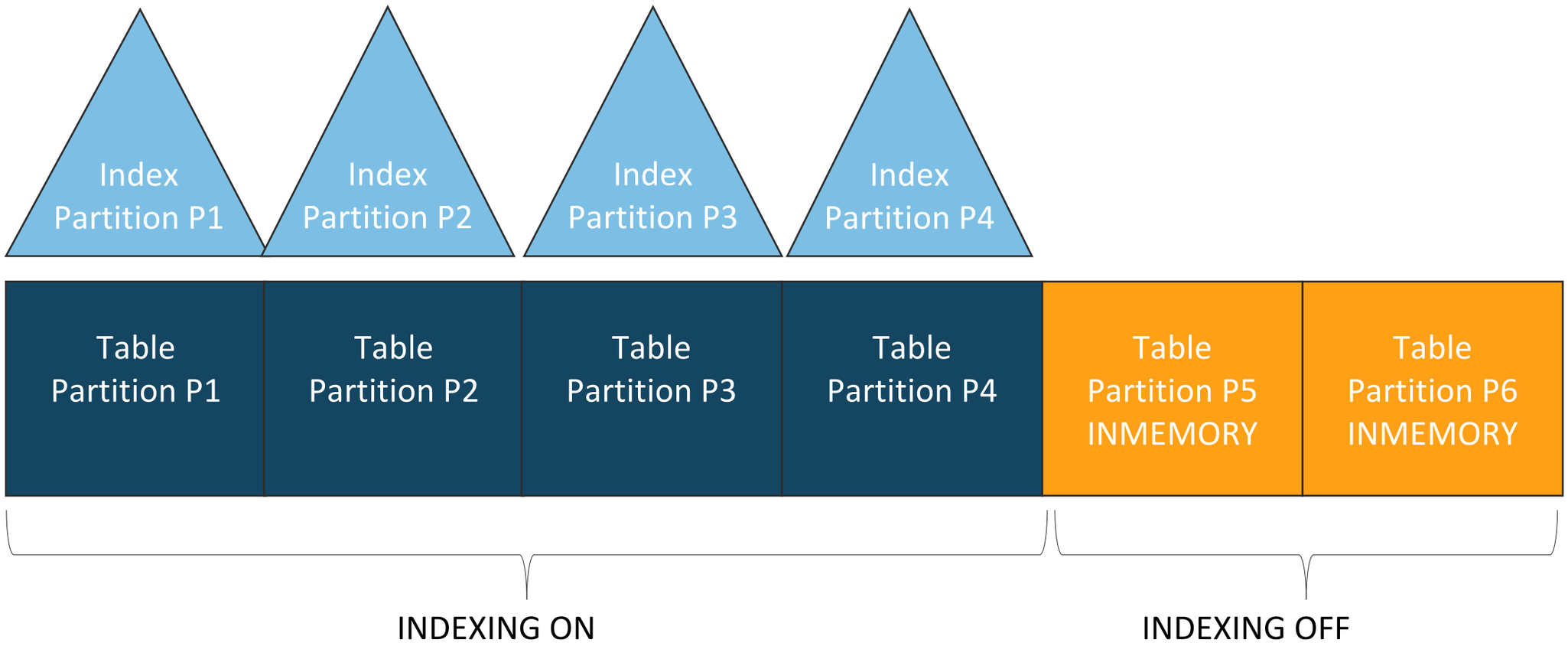
CALLISTA



Combining Performance Features



In-Memory & Partial Indexes



CALLISTA

In-Memory & Partial Indexes

```
SELECT partition_position pos, partition_name, indexing, inmemory
  FROM user_tab_partitions WHERE table_name = 'SALES'
ORDER BY partition_position;


       POS PARTITION_NAME       INDEXING INMEMORY
---------- -------------------- -------- --------
         1 SALES_2018           ON       DISABLED
         2 SALES_H1_2019        ON       DISABLED
         3 SALES_H2_2019        ON       DISABLED
         4 SALES_Q1_2020        ON       DISABLED
         5 SALES_Q2_2020        ON       DISABLED
         6 SALES_Q3_2020        ON       DISABLED
         7 SALES_Q4_2020        ON       DISABLED
         8 SALES_Q1_2021        ON       DISABLED
         9 SALES_Q2_2021        ON       DISABLED
        10 SALES_Q3_2021        ON       DISABLED
        11 SALES_Q4_2021        ON       DISABLED
        12 SALES_Q1_2022        OFF      ENABLED
        13 SALES_Q2_2022        OFF      ENABLED
        14 SALES_Q3_2022        OFF      ENABLED
        15 SALES_Q4_2022        OFF      ENABLED
```

CALLISTA

Live Demo

In-Memory &
Partial Local Indexes

CALLISTA

Indexing

- B-tree indexes for primary key / unique constraints
- Bitmap indexes on dimension keys of fact tables

Materialized Views & Query Rewrite

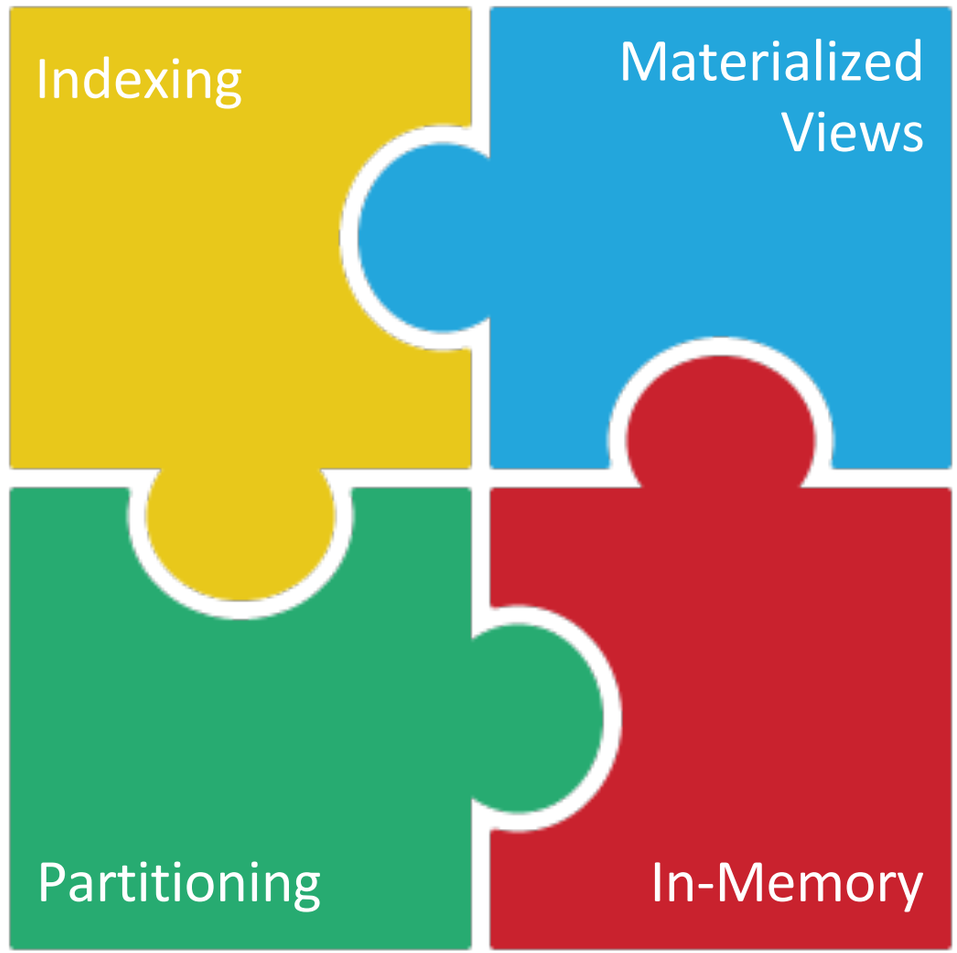
- Useful for drill-up on higher hierarchy levels
- Query Rewrite for queries on star schemas

Partitioning

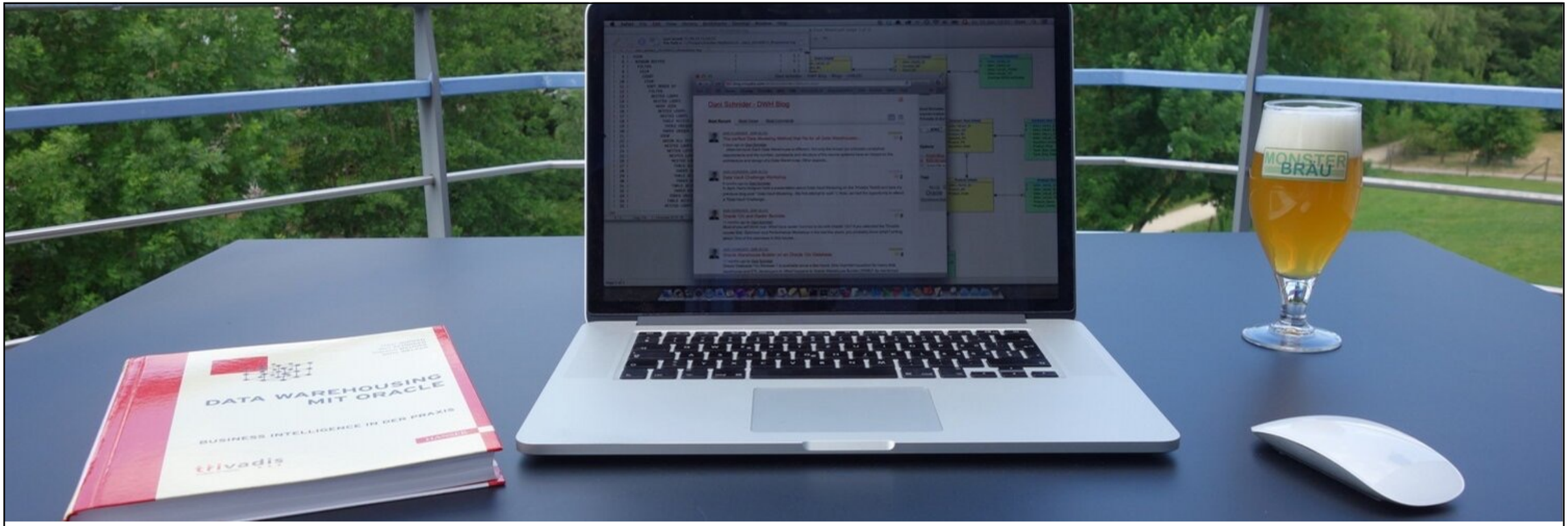
- RANGE/INTERVAL partitioning on date columns
- Mainly used for fact tables in star schemas

Database In-Memory

- High performance improvements in star schemas
- No additional indexes (except PK/UK) required anymore



CALLISTA



https://danischnider.wordpress.com

https://twitter.com/dani_schnider

https://www.linkedin.com/in/danischnider/

Q&A

The Educational Conference For Oracle Techology Users

ODTUG Kscope23

aurora, co june 25-29

Interested in volunteering?
Scan the code to sign up

Don't Forget To
Fill Out Your Evals


DRIVEN BY EXCELLENCE
CALLISTA