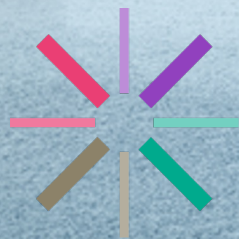
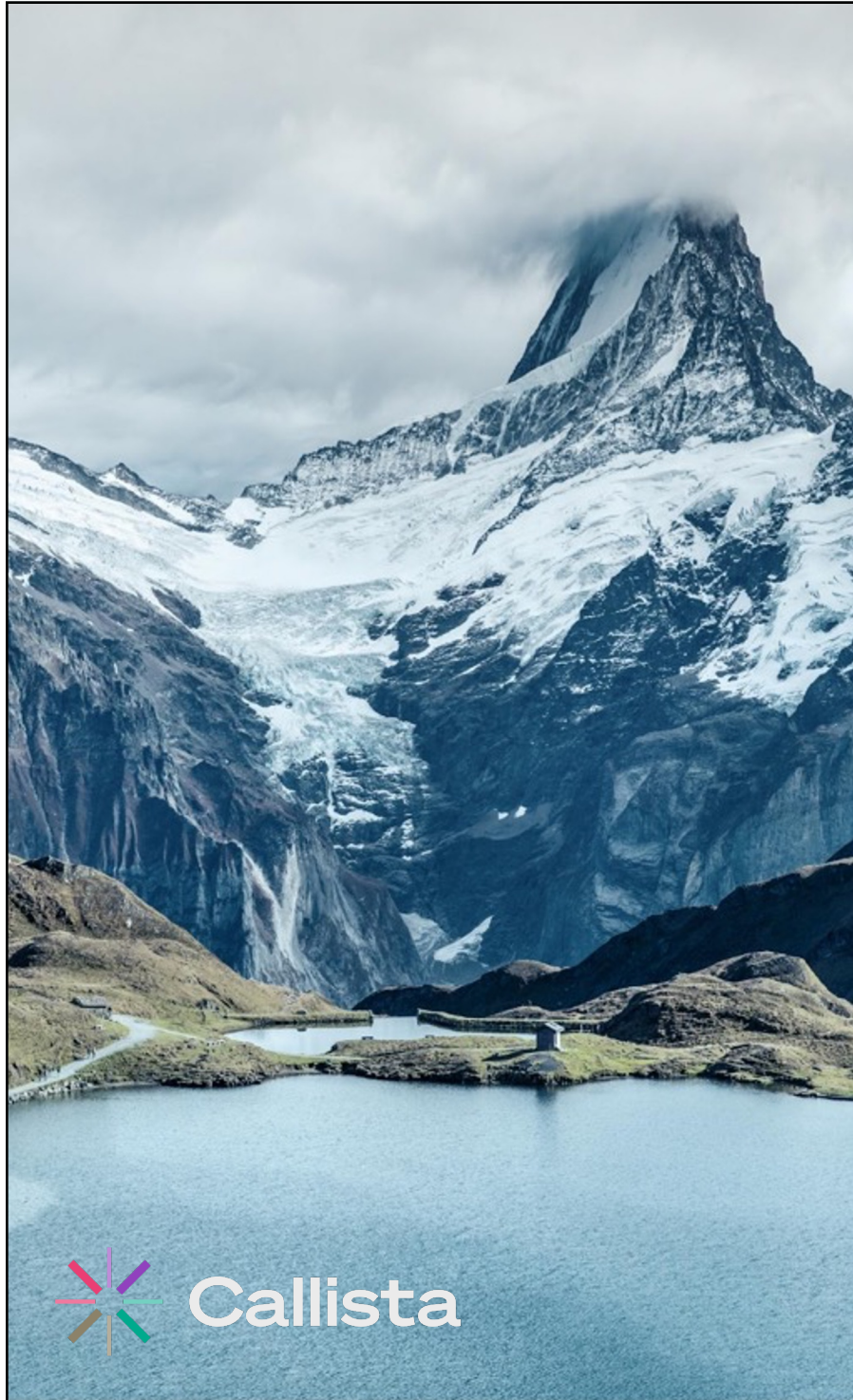


Dani Schneider

Function Calls in SQL – Black Box for the Optimizer?



Callista



WHERE **f(attr)** = expr

Issues with Function Calls

**Index
Usage**

**Cardinality
Estimation**

**Context
Switch**

**Nested
SQL Calls**

About me



Dani Schnider

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



[@danischnider.bsky.social](https://bsky.social/@danischnider)



danischnider.wordpress.com



www.linkedin.com/in/danischnider/



SQL Function Calls in WHERE Condition



Callista



Live Demo

Simple Use Case

Simple Example

```
SELECT *  
  FROM addresses  
 WHERE ctr_code = 'GB'
```

```
SELECT *  
  FROM addresses  
 WHERE ctr_code = 'DE'
```

- 357 of 23941 rows (1.5%)
- Cardinality = 357
- Index Range Scan is used

- 9344 of 23941 rows (39%)
- Cardinality = 9344
- Full Table Scan is used

Example with Function Calls

```
SELECT *  
  FROM addresses  
 WHERE UPPER(ctr_code) = 'GB'
```

```
SELECT *  
  FROM addresses  
 WHERE UPPER(ctr_code) = 'DE'
```

**Index
Usage**

**Cardinality
Estimation**

- 357 of 23941 rows (1.5%)
 - Cardinality = 239
 - Full Table Scan is used
-
- 9344 of 23941 rows (39%)
 - Cardinality = 239
 - Full Table Scan is used

$$\text{Cardinality} = \text{Selectivity} * \text{TotalNumberOfRows}$$

```
WHERE ctr_code = 'DE'
```

Cardinality is derived from

- Table statistics (total number of rows)
- Column statistics (number of distinct values, histograms)
- WHERE condition

$$\text{Cardinality} = \text{Selectivity} * \text{TotalNumberOfRows}$$

0.01

WHERE



= 'DE'

Function call is a black box for the optimizer

- Selectivity is always 1%

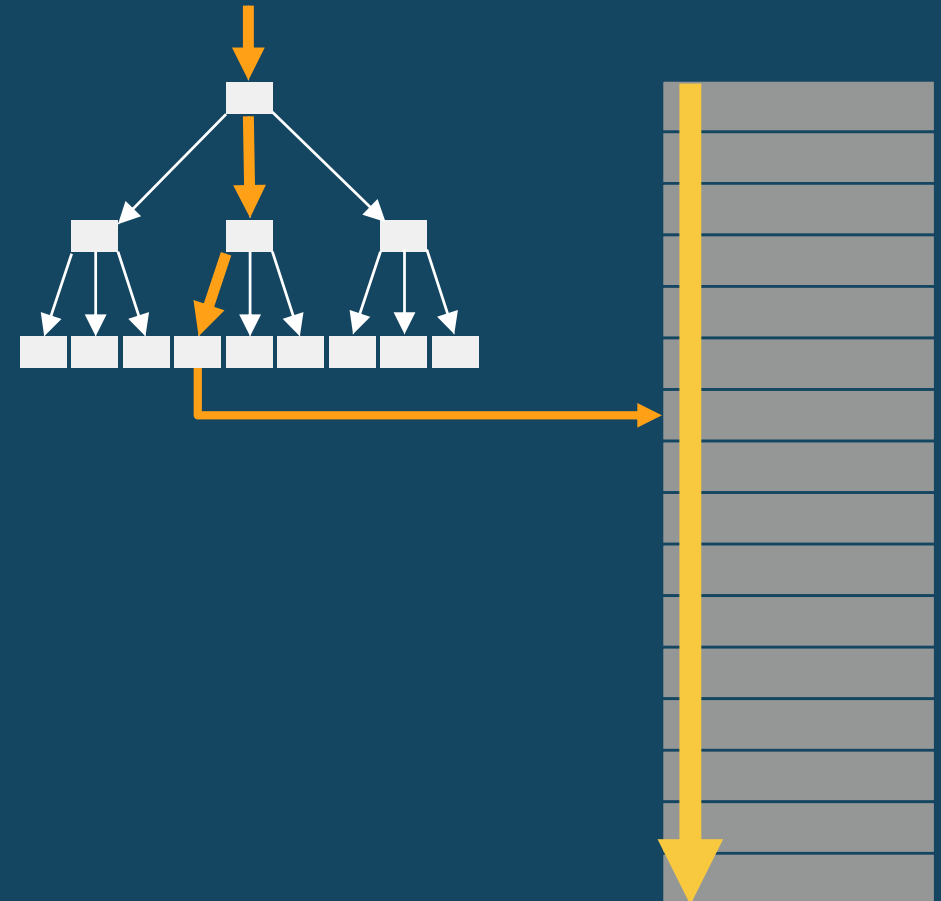
Index Usage

```
WHERE ctr_code = 'GB'
```

- Index is used for strong selectivity
- Full table scan is used for weak selectivity

```
WHERE UPPER(ctr_code) = 'GB'
```

- Index cannot be used at all
- Full table scan is the only option





Live Demo

Function-based Index

Solution 1: Function-Based Index

```
CREATE INDEX adr_upper_ctr_code_fbi  
ON addresses(UPPER(ctr_code))
```



Cardinality
Estimation



Index
Usage

- Index can be used for function calls
- Correct cardinality estimation after gathering new statistics



Live Demo

Virtual Column

Solution 2: Virtual Column

```
ALTER TABLE addresses  
ADD (upper_ctr_code VARCHAR2(30)  
    AS (UPPER(ctr_code)) VIRTUAL)
```



Cardinality
Estimation



Index
Usage

- Better cardinality estimation due to column statistics on virtual column
- Index on virtual column can be created (= function-based index)
- WHERE condition can use virtual column name or expression/function call



Live Demo

Extended Statistics

Solution 3: Extended Statistics

```
SELECT dbms_stats.create_extended_stats  
       (USER, 'addresses', '(UPPER(ctr_code))')  
FROM dual
```



Cardinality
Estimation



Index
Usage

- Additional statistics for expressions, function calls or column correlations
- Better cardinality estimation due to additional statistics
- Function-based index possible in Oracle 23c

Features Overview

	Function-based Index	Virtual Column	Extended Statistics
Cardinality Estimation			
Index Usage			
Virtual Column	Yes	Yes	Yes
Hidden Column	Yes	No	Yes

PL/SQL Function Calls



Callista

Example for PL/SQL Function

```
CREATE OR REPLACE FUNCTION f_price_category
  (in_quantity order_items.quantity%TYPE,
   in_price order_items.price_per_unit%TYPE) RETURN VARCHAR2
IS
  v_total_price order_items.price_per_unit%TYPE;
  v_category VARCHAR2(6);
BEGIN
  v_total_price := in_quantity * in_price;
  IF v_total_price < 100 THEN
    v_category := 'low';
  ELSIF v_total_price < 1000 THEN
    v_category := 'medium';
  ELSE
    v_category := 'high';
  END IF;
  RETURN v_category;
END f_price_category;
```



Live Demo

PL/SQL Function Call

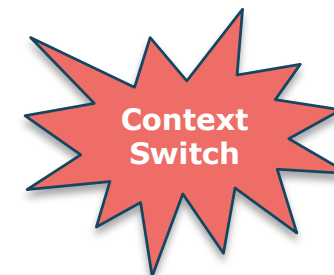
PL/SQL Function Call in SQL Query

```
SELECT COUNT(*), SUM(quantity)
  FROM order_items
 WHERE f_price_category(quantity, price_per_unit) = 'high'
```

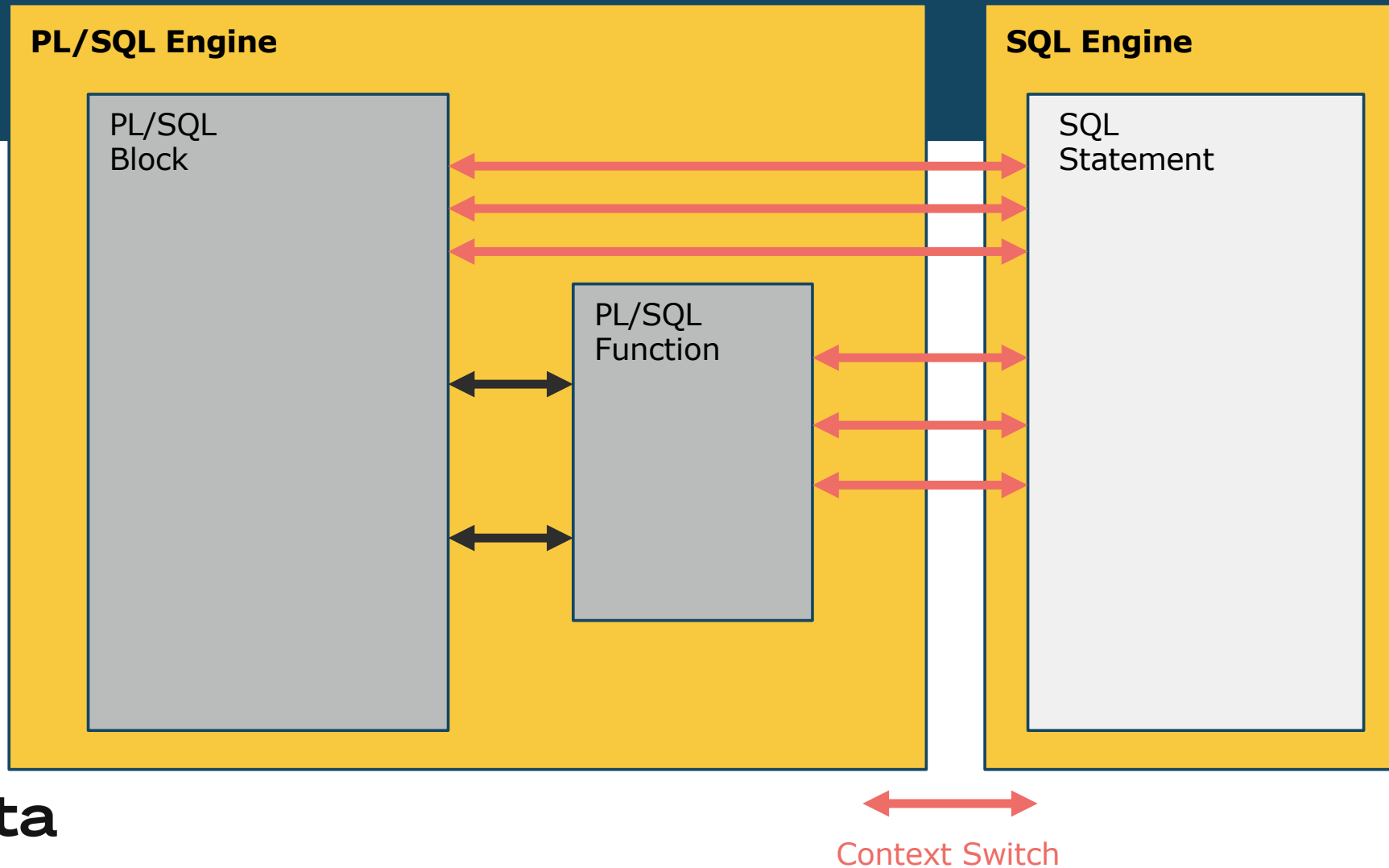


How can we Improve the Query Performance?

- Function-based Index
 - Function must be DETERMINISTIC
 - Index creation takes time and disk space
 - Index is not optimal choice for all type of queries
- What is the root cause for the bad response time?



Context Switch between PL/SQL and SQL





Live Demo

Avoid Context Switch

Solution 1: User Defined Function

```
CREATE OR REPLACE FUNCTION f_price_category
(in_quantity order_items.quantity%TYPE,
 in_price order_items.price_per_unit%TYPE) RETURN VARCHAR2
IS
  PRAGMA UDF;
  v_total_price order_items.price_per_unit%TYPE;
  v_category VARCHAR2(6);
BEGIN
  v_total_price := in_quantity * in_price;
  IF v_total_price < 100 THEN
    v_category := 'low';
  ELSIF v_total_price < 1000 THEN
    v_category := 'medium';
  ELSE
    v_category := 'high';
  END IF;
  RETURN v_category;
END f_price_category;
```

- PRAGMA UDF in function declaration
- Useful for PL/SQL functions mainly used in SQL statements

Solution 2: SQL Macro



```
CREATE OR REPLACE FUNCTION f_price_category_macro
(in_quantity VARCHAR2,
 in_price VARCHAR2) RETURN VARCHAR2 SQL_MACRO (SCALAR)
IS
  v_category VARCHAR2(4000);
BEGIN
  v_category := 'CASE
                WHEN in_quantity * in_price < 100 THEN
''low''
                WHEN in_quantity * in_price < 1000 THEN
''medium''
                ELSE ''high''
                END';
  RETURN v_category;
END f_price_category_macro;
```

- Scalar expression that can be used in SELECT, WHERE, GROUP BY and ORDER BY clauses
- Processed in SQL statement directly, no context switch to PL/SQL

PL/SQL Function Calls with Embedded SQL



Callista

PL/SQL Function with SQL Calls

```
CREATE OR REPLACE FUNCTION f_price_category
(in_quantity order_items.quantity%TYPE,
 in_price order_items.price_per_unit%TYPE) RETURN VARCHAR2
IS
  v_total_price order_items.price_per_unit%TYPE;
  v_category VARCHAR2(6);
BEGIN
  v_total_price := in_quantity * in_price;

  SELECT category
  INTO v_category
  FROM categories
  WHERE v_total_price >= min_price
        AND v_total_price < max_price;

  RETURN v_category;
END f_price_category;
```

SQL Query

CATEGORY	MIN_PRICE	MAX_PRICE
low	0	100
medium	100	1000
high	1000	999999



Live Demo

PL/SQL Function
with Embedded SQL

How can we Improve the Query Performance?



What is the Problem?



```
SELECT COUNT(*), SUM(quantity)
  FROM order_items
 WHERE f_price_category(quantity, price_per_unit) = 'high'
```

call	count	cpu	elapsed	disk	query	current	rows
Parse	1	0.04	0.28	0	2	0	0
Execute	1	0.00	0.00	0	0	0	0
Fetch	1	140.51	536.93	0	1728	0	0
total	3	140.56	537.21	0	1730	0	0

```
Rows (1st) Rows (avg) Rows (max) Row Source Operation
-----
0 0 0 SORT AGGREGATE (cr=0 pr=0 pw=0 time=23 us starts=1)
33132 33132 33132 TABLE ACCESS FULL ORDER_ITEMS (cr=2015895 pr=2 pw=0 time=559816988 us starts=1 cost=1
250 size=66264 card=8283)
```

```
SELECT CATEGORY
FROM
  CATEGORIES WHERE :B1 >= MIN_PRICE AND :B1 < MAX_PRICE
```

call	count	cpu	elapsed	disk	query	current	rows
Parse	1	0.00	0.00	0	0	0	0
Execute	335796	4.26	10.93	0	0	0	0
Fetch	335796	8.52	22.10	0	2014772	0	335795
total	671593	12.78	33.04	0	2014772	0	335795

```
Parsing user id: 90 (recursive depth: 1)
```

```
Rows (1st) Rows (avg) Rows (max) Row Source Operation
-----
1 1 1 TABLE ACCESS FULL CATEGORIES (cr=6 pr=0 pw=0 time=85 us starts=1 cost=3 size=12 card=1)
```



Dani Schnider
@dani_schnider

Calling #PLSQL functions in a #SQL query, and the functions run other #SQL queries with #PLSQL functions than call #SQL statements with #PLSQL functions 🙄

#Performance tuning issues on a Friday afternoon - and another example for SQL #HorrorShow



1:09 PM - 24 May 2019

6 Retweets 46 Likes



5 6 46



Steven Feuerstein
@sfonplsql

Following

Let this be a warning to us all:

When you call your own #PLSQL functions in a #SQL statement, it should NOT have SQL in it. If it does review closely for read consistency issues, performance, ridiculous nesting of context switches.



Dani Schnider @dani_schnider

Calling #PLSQL functions in a #SQL query, and the functions run other #SQL queries with #PLSQL functions than call #SQL statements with #PLSQL functions 🙄

2:14 PM - 24 May 2019

11 Retweets 44 Likes



5 11 44

Solution: Avoid PL/SQL Function

```
SELECT COUNT(*), SUM(quantity)
  FROM order_items
 WHERE (SELECT category
        FROM categories
       WHERE quantity * price_per_unit >= min_price
        AND quantity * price_per_unit < max_price) = 'high'
```

```
SELECT COUNT(*), SUM(quantity)
  FROM order_items oi, categories cat
 WHERE cat.category = 'high'
    AND oi.quantity * oi.price_per_unit >= cat.min_price
    AND oi.quantity * oi.price_per_unit < cat.max_price
```

SOLUTION 1

- Replace function call with subquery

SOLUTION 2

- Join with lookup table

Conclusion

SQL Functions

No problem at all in SELECT list

Try to avoid them in WHERE condition

Solve cardinality estimation issues:

- Virtual Columns
- Extended Statistics

For queries with strong selectivity:

- create Function-based Index
- create index on virtual column

PL/SQL Functions

Same rules as for SQL functions

Try to avoid them in SELECT list and WHERE condition

Solve context switch issues:

- PRAGMA UDF
- SQL MACRO (SCALAR)

Never use PL/SQL function calls with embedded SQL queries!

- Try to rewrite the SQL statement



danischnider.wordpress.com

<https://danischnider.wordpress.com/2022/02/28/performance-tips-function-calls-in-where-conditions/>

<https://danischnider.wordpress.com/2022/05/15/performance-tips-pl-sql-functions-in-sql-queries/>



Data Warehousing with Oracle

Dani Schneider's Blog



[Home](#) [Contact](#) [About](#) [Publications](#) [Presentations](#)

SEARCH RESULTS FOR: FUNCTION CALLS

Performance Tips: Function Calls in WHERE Conditions

Posted on 28. February 2022

A typical reason of many performance issues with SQL are function calls in the WHERE condition. They make it difficult for the query optimizer to determine the cardinality. Fortunately, there are several ways to solve this problem.

Posted in [Oracle](#), [Performance](#), [SQL](#) | 1 Reply

1

Performance Tips: PL/SQL Functions in SQL Queries

Posted on 15. May 2022

PL/SQL functions in SQL statements may cause major performance problems. If it is not possible to avoid these calls, there are several solutions to improve the performance. But not for all situations.

Posted in [Oracle](#), [Performance](#), [PL/SQL](#), [SQL](#) | 1 Reply

1

**Oracle ACE Director**

**SYMPOSIUM L2**

**CALLISTA**

CATEGORIES

- ADWC (6)
- Books (8)
- Cloud (5)
- Conferences (19)
- Data Vault Modeling (20)
- Dimensional Modeling (11)
- DWH Architecture (16)
- DWH Automation (4)
- ETL (27)
- In-Memory (5)
- Materialized Views (6)
- Oracle (79)
- Oracle 12c (27)
- Oracle 19c (3)
- Oracle 20c (1)
- Oracle Data Integrator (1)

 **ASK TOM**

Search Questions...

[Questions](#)

[Office Hours](#)

[Videos](#)

[Resources](#)

Write Great SQL Office Hours

Free tips and training every month! Subscribe for reminders and more from Office Hours. [FAQ](#)

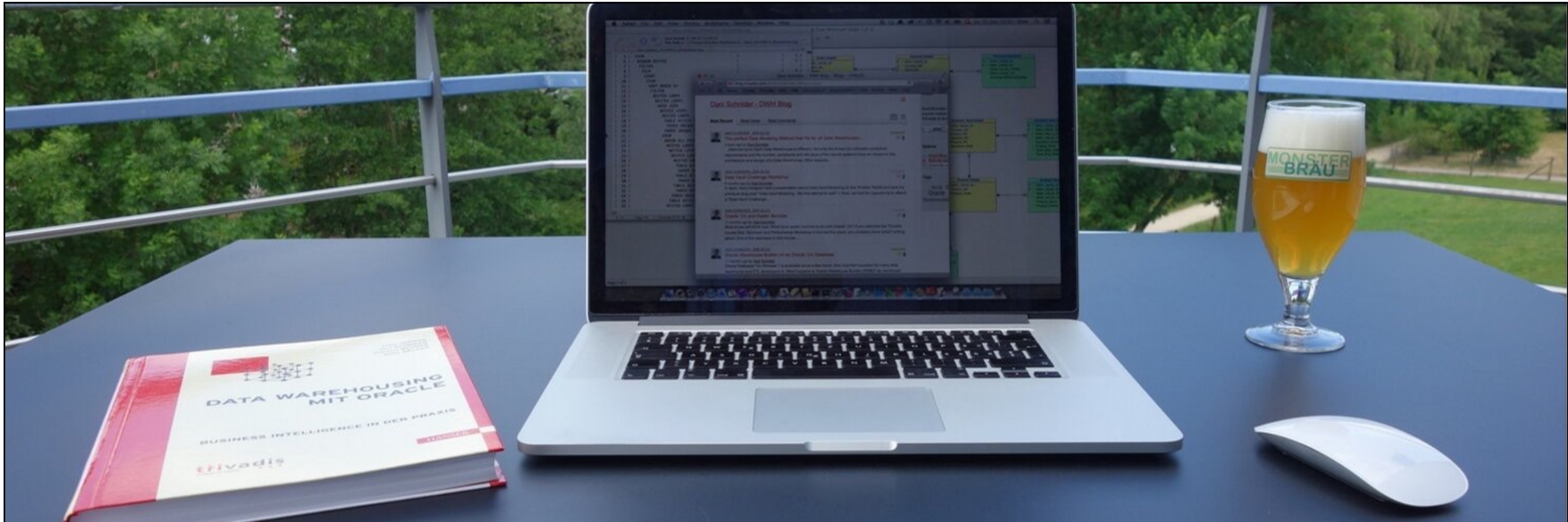
April 19, 2022 13:00 UTC [Start Times Around the World](#)

Subscribe to be notified of changes to sessions and give us feedback!



Having trouble watching the video on this page? Open [the video](#) in your browser.

https://asktom.oracle.com/pls/apex/f?p=100:551::::551:P551_CLASS_ID,P551_INVITED:17183

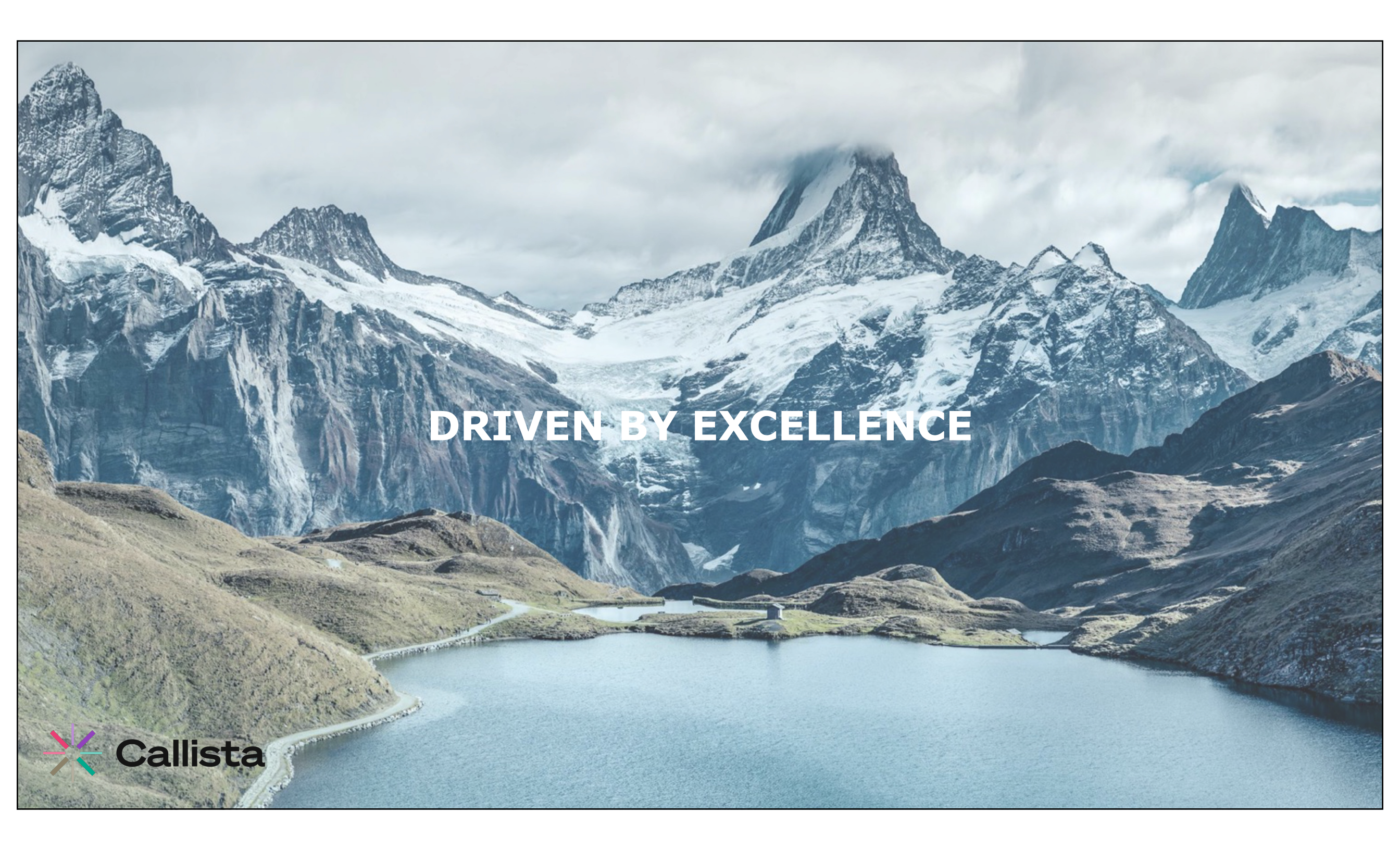


 <https://danischnider.wordpress.com>

 https://twitter.com/dani_schnider

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

Q&A

A wide-angle photograph of a mountain range. In the foreground, a calm, blue lake reflects the sky. The middle ground shows rolling hills with patches of green and brown vegetation. In the background, several sharp, rocky mountain peaks are visible, some covered in snow and partially shrouded in mist or clouds. The overall scene is serene and majestic.

DRIVEN BY EXCELLENCE



Callista