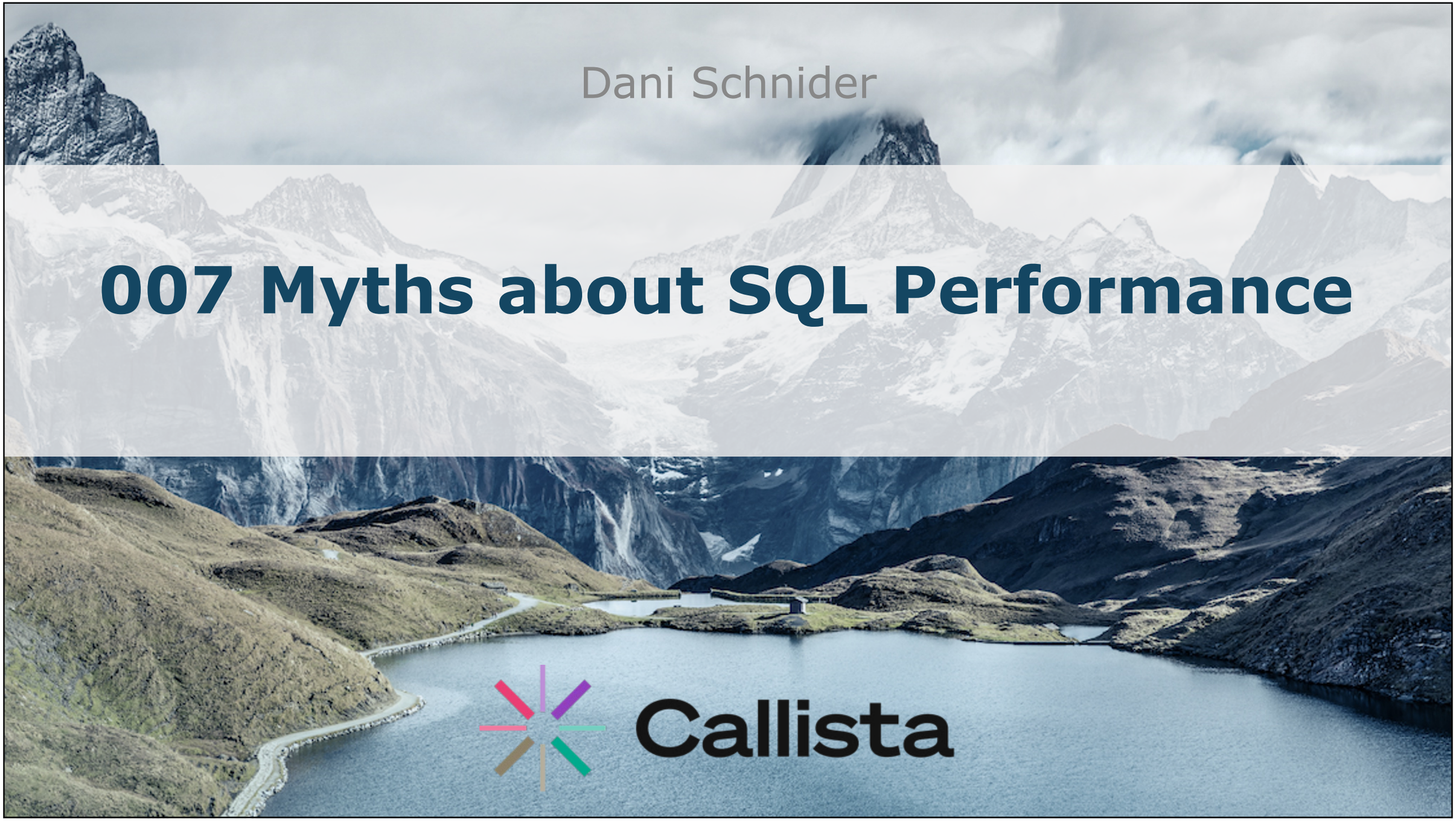

Dani Schnider
007 Myths about SQL Performance
Callista

About me



Dani Schnider

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

@danischnider.bsky.social danischnider.wordpress.com www.linkedin.com/in/danischnider/

Callista

Inspired by James Bond movie titles



Try

«~~Die~~ Another Day»



Look at performance at the end of the project

Performance Tuning Tasks

Performance Planning

- Scalability
- System Architecture
- Application Design
- Data Model
- Testing
- etc.

Instance Tuning

- Memory Allocation
- I/O Balancing
- Database Configuration
- System Configuration
- etc.

SQL Tuning

- Indexing
- Gathering Statistics
- Partitioning
- Rewrite SQL Statements
- Hints
- etc.

See Oracle® Database Performance Tuning Guide, Chapter 1, Performance Tuning Overview

One of my first Performance Reviews

One of my first Performance Reviews

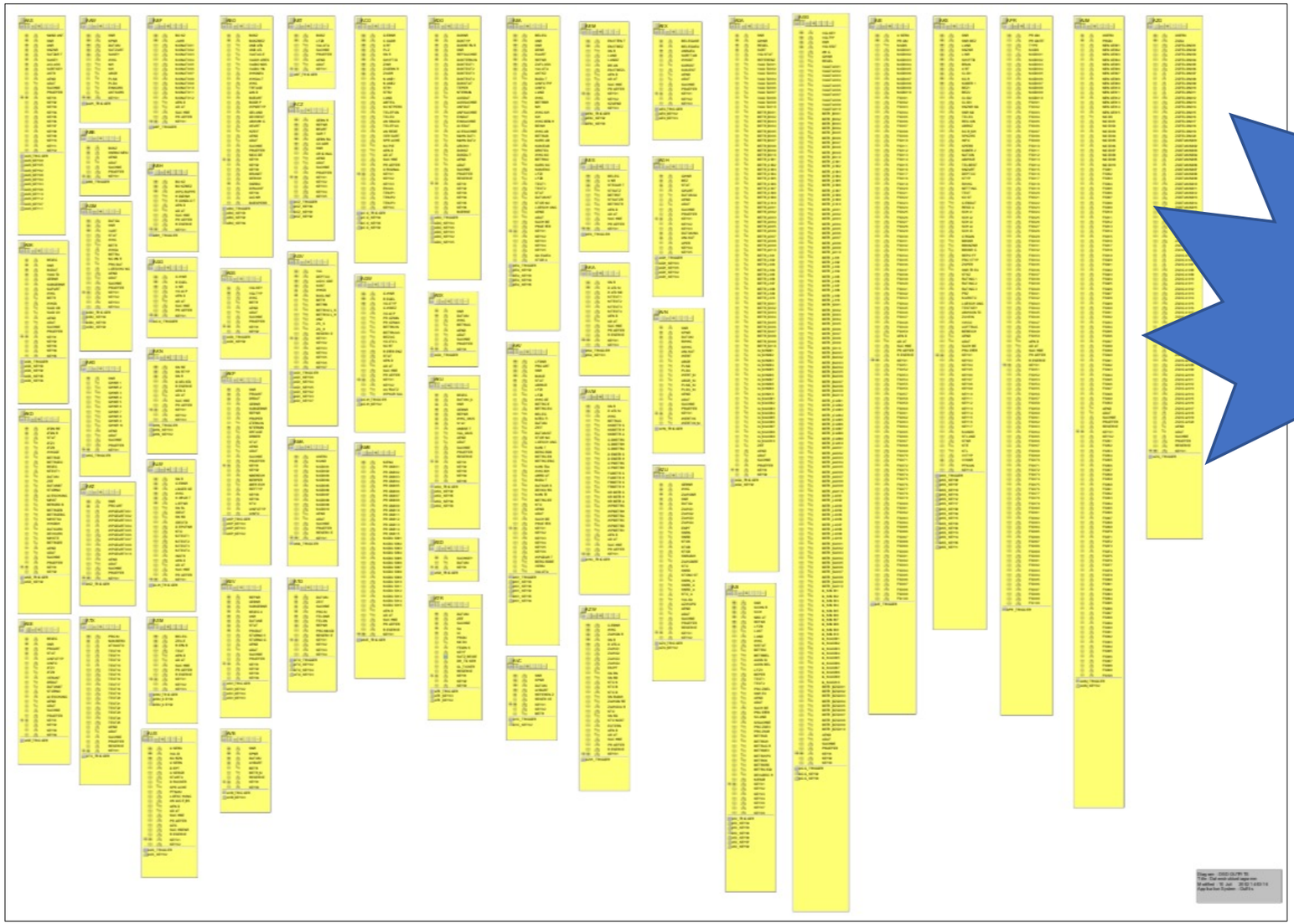
```
select SNR  ,SNRBEZ  ,LAND  ,KNZNR  ,LINR  ,SWIFTID  ,BRAN  ,ORT  ,CLID1  ,
  KGR  ,KUBER1  ,BEZ1  ,BEZ2  ,CLID2  ,CLID3  ,KNZNRMU  ,SNRMU  ,TELEX  ,
  REGION  ,ABRKZ  ,KGR_SW  ,SPKZHS  ,INFO  ,SPERR  ,KUBER2  ,NATION  ,ABW
  TELBEST  ,KNZART  ,DEPTCO  ,KTYP  ,RWHG  ,NETTING  ,TEXT  ,KOST  ,GEBDAT
  RESDA  ,SCHL1  ,SCHL2  ,SCHL3  ,SCHL4  ,SCHL5  ,ORGAN  ,BBSNR  ,BBKNZNR
  BBSNRG  ,REPOFF  ,PROVTYP  ,ZSPER  ,SNRTREU  ,STKZ  ,RATING1  ,RATI
  RATING3  ,PRZ  ,MARKTO  ,LOESCHUNG  ,TESTKEY  ,ABWKONTO  ,ZAVEIN  ,V
  AUFTRAG  ,BEREICH  ,MANDK  ,AEND  ,ADAT  ,SACHBE  ,PRUEFER  ,STLAND
  ,KTE  ,KTL  ,VVTYP  ,VVSNR  ,PFMAN  into :b1,:b2,:b3,:b4,:b5,:b6,:b7
  :b9,:b10,:b11,:b12,:b13,:b14,:b15,:b16,:b17,:b18,:b19,:b20,:b21,:b
  :b24,:b25,:b26,:b27,:b28,:b29,:b30,:b31,:b32,:b33,:b34,:b35,:b36,:b37,
  :b39,:b40,:b41,:b42,:b43,:b44,:b45,:b46,:b47,:b48,:b49,:b50,:b51,:b52,:
  :b54,:b55,:b56,:b57,:b58,:b59,:b60,:b61,:b62,:b63,:b64,:b65,:b66,:b67,
  :b69,:b70,:b71,:b72,:b73,:b74,:b75
from
 AKS where rowid =:b76
```




```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse     9275      2.63       4.30          0          0          0           0
Execute 195234     16.46      17.07          0          0          0           0
Fetch   195230     40.85     165.61      63452     195230          0      195230
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total   399739     59.94     186.98      63452     195230          0      195230

Rows     Execution Plan
-------  ---------------------------------------------------
      0  SELECT STATEMENT   GOAL: CHOOSE
      1   TABLE ACCESS (BY USER ROWID) OF 'AKS'
```

One of my first Performance Reviews



The data model

One of my first Performance Reviews



The best solution* for the application

(I wrote it a little more diplomatically in the review document)*

Performance Tuning Tasks

Performance Planning

- Scalability
- System Architecture
- Application Design
- Data Model
- Testing
- etc.

Instance Tuning

- Memory Allocation
- I/O Balancing
- Database Configuration
- System Configuration
- etc.

SQL Tuning

- Indexing
- Gathering Statistics
- Partitioning
- Rewrite SQL Statements
- Hints
- etc.

See Oracle® Database Performance Tuning Guide, Chapter 1, Performance Tuning Overview

Performance must be considered in every phase of the project

«From Index Russia With Love»



Index is good, full table scan is bad

Which Execution Plan is Better?

OPERATION	OBJECT_NAME	CARDINALITY	COST
SELECT STATEMENT		1	274
NESTED LOOPS		1	274
NESTED LOOPS		1	274
NESTED LOOPS		1	273
NESTED LOOPS		1	272
NESTED LOOPS		114	272
TABLE ACCESS (BY INDEX ROWID BATCHED)	PERSONS	69	65
INDEX (RANGE SCAN)	PRS_BIRTHDAY	69	2
TABLE ACCESS (BY INDEX ROWID BATCHED)	ADDRESSES	2	3
INDEX (RANGE SCAN)	ADR_PRS_ID	2	1
INDEX (UNIQUE SCAN)	CTR_PK	1	0
TABLE ACCESS (BY INDEX ROWID)	FORMSOFADDRESS	1	1
INDEX (UNIQUE SCAN)	FOA_PK	1	0
INDEX (UNIQUE SCAN)	ATP_PK	1	0
TABLE ACCESS (BY INDEX ROWID)	ADR_TYPES	1	1

A

OPERATION	OBJECT_NAME	CARDINALITY	COST
SELECT STATEMENT		1	899
HASH JOIN		1	899
HASH JOIN		1	896
HASH JOIN		1	893
INDEX (FULL SCAN)	CTR_PK	1	1
HASH JOIN		114	892
TABLE ACCESS (FULL)	PERSONS	69	278
TABLE ACCESS (FULL)	ADDRESSES	255691	613
TABLE ACCESS (FULL)	ADR_TYPES	1	3
TABLE ACCESS (FULL)	FORMSOFADDRESS	3	3

B

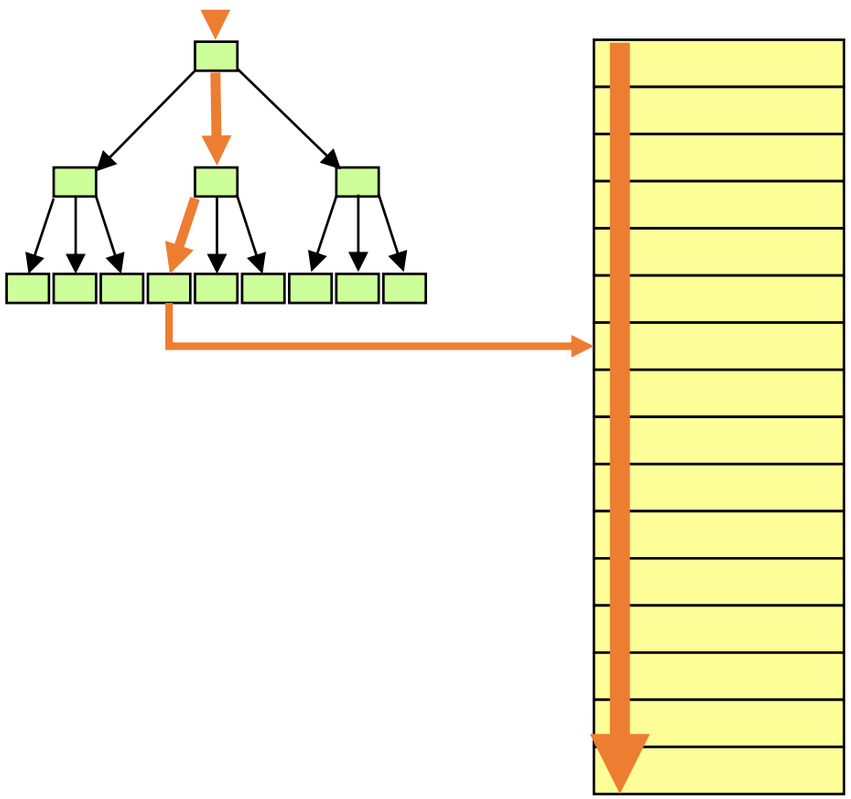
Is this a Good Execution Plan?

OPERATION	OBJECT_NAME	CARDINALITY	COST
SELECT STATEMENT		56	1886
SORT (ORDER BY)		56	1886
HASH (GROUP BY)		56	1886
HASH JOIN		578243	1854
TABLE ACCESS (STORAGE FULL)	MONSTER_DV.PIT_BEER	6	2
HASH JOIN (RIGHT OUTER)		341482	1850
TABLE ACCESS (STORAGE FULL)	MONSTER_DV.S_ORDER_ITEM	458527	19
HASH JOIN		341482	1416
TABLE ACCESS (STORAGE FULL)	MONSTER_DV.PIT_ORDER_ITEM	458527	17
HASH JOIN (RIGHT OUTER)		341482	1013
TABLE ACCESS (STORAGE FULL)	MONSTER_DV.S_CUSTOMER_ADDRESS	33406	6
HASH JOIN		341482	1006
TABLE ACCESS (STORAGE FULL)	MONSTER_DV.PIT_CUSTOMER	20000	3
HASH JOIN		255719	1001
TABLE ACCESS (STORAGE FULL)	MONSTER_DV.PIT_CUSTOMER	20000	3
HASH JOIN		248981	997
TABLE ACCESS (STORAGE FULL)	MONSTER_DV.L_CUSTOMER_ORDER	357729	34
HASH JOIN		245184	761
TABLE ACCESS (STORAGE FULL)	MONSTER_DV.S_INVOICE	233084	35
HASH JOIN		376300	497
TABLE ACCESS (STORAGE FULL)	MONSTER_DV.PIT_ORDER	357729	20
HASH JOIN		370562	269
TABLE ACCESS (STORAGE FULL)	MONSTER_DV.L_ORDER_ITEM	458527	40
HASH JOIN		370562	40
JOIN FILTER (CREATE)	SYS.:BF0000	6	4
HASH JOIN (OUTER)		6	4
JOIN FILTER (CREATE)	SYS.:BF0001	6	2
TABLE ACCESS (STORAGE FULL)	MONSTER_DV.PIT_BEER	6	2
JOIN FILTER (USE)	SYS.:BF0001	19	2
TABLE ACCESS (STORAGE FULL)	MONSTER_DV.S_BEER_DESCRIPTION	19	2
JOIN FILTER (USE)	SYS.:BF0000	458527	34
TABLE ACCESS (STORAGE FULL)	MONSTER_DV.L_BEER_ORDER_ITEM	458527	34

No

Yes

Index Scan or Full Table Scan?



It depends...

... on the selectivity of the query

Index Scan is efficent for a *strong* selectivity (i.e. small percentage of data)

Full Table Scan is efficent for a *weak* selectivity (i.e. high percentage of data)

LIVE DEMO

«For Your Eyes Only»



Count

«The ~~World~~ Is Not Enough»



COUNT(1) is faster than COUNT(*)

Which is the Fastest Way to Count the Rows?

A

```
SELECT COUNT(*) FROM employees
```

B

```
SELECT COUNT(1) FROM employees
```

C

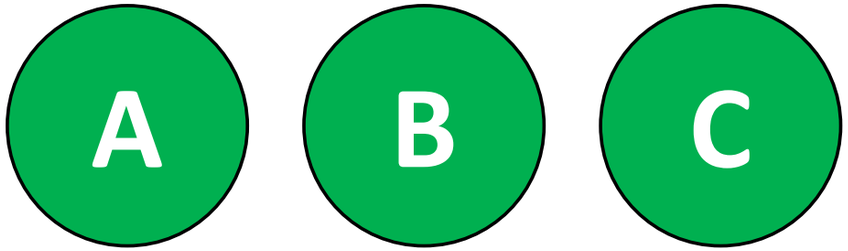
```
SELECT COUNT(emp_id) FROM employees
```

LIVE DEMO

«For Your Eyes Only»



Which is the Fastest Way to Count the Rows?




```
SELECT COUNT(*) FROM employees
```

```
SELECT COUNT(1) FROM employees
```

```
SELECT COUNT(emp_id) FROM employees
```

The optimizer does an index fast full scan on the smallest index on a NOT NULL columns

→ **All queries use the same execution plan**

```
-----------------------------------------------------
| Id  | Operation                | Name          | Rows  |
-----------------------------------------------------
|   0 | SELECT STATEMENT         |               |     1 |
|   1 |  SORT AGGREGATE          |               |     1 |
|   2 |   INDEX FAST FULL SCAN| EMP_LIC_KILL |  6453 |
-----------------------------------------------------
```

«Live and Let ~~Die~~ Buy»



Buy more hardware will give us good performance

Add More Hardware...

Scale Up/Down cancel

CPU CORE COUNT

8

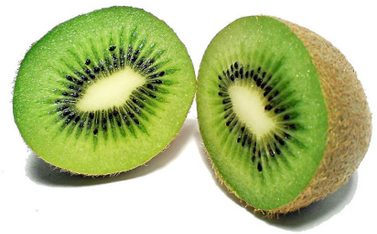
The number of CPU cores to enable. Maximum cores per database: 128. Available cores are subject to your tenancy's service limits.

STORAGE (TB)

2

The available storage, up to 128 TB.

Update



KIWI

«Kill It With Iron»



```
WITH a AS (SELECT … )
```

```
   , b AS (SELECT … )
```

```
   , c AS (SELECT … )
```

```
   , d AS (SELECT … )
```

```
   , e AS (SELECT … )
```

```
   , f AS (SELECT … )
```

```
SELECT DISTINCT …
  FROM a
  JOIN b
  JOIN c
  JOIN d
  JOIN e
  JOIN f
```

ORA-01652: unable to extend temp segment by 128 in tablespace TEMP

```
WITH a AS (SELECT … )
   , b AS (SELECT … )
   , c AS (SELECT … )
   , d AS (SELECT … )
```

Join condition was missing

```
SELECT DISTINCT …
  FROM a
  JOIN b
  JOIN c
  JOIN d
```

A Increase TEMP tablespace

B Add missing join condition

«The Man with the Golden ~~Gun~~ Hint»

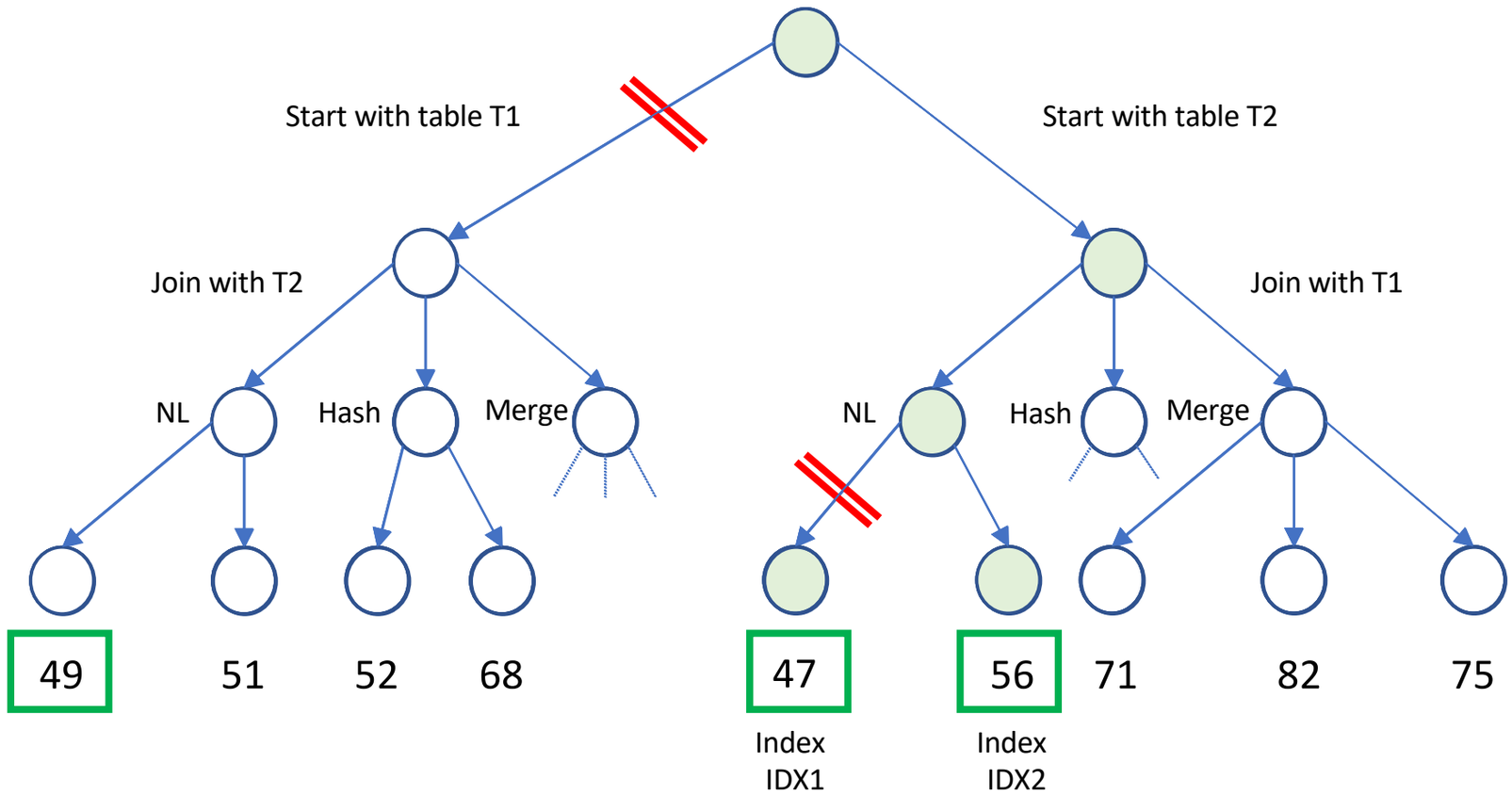


Hints reduce the costs of an execution plan

Concept of Hints

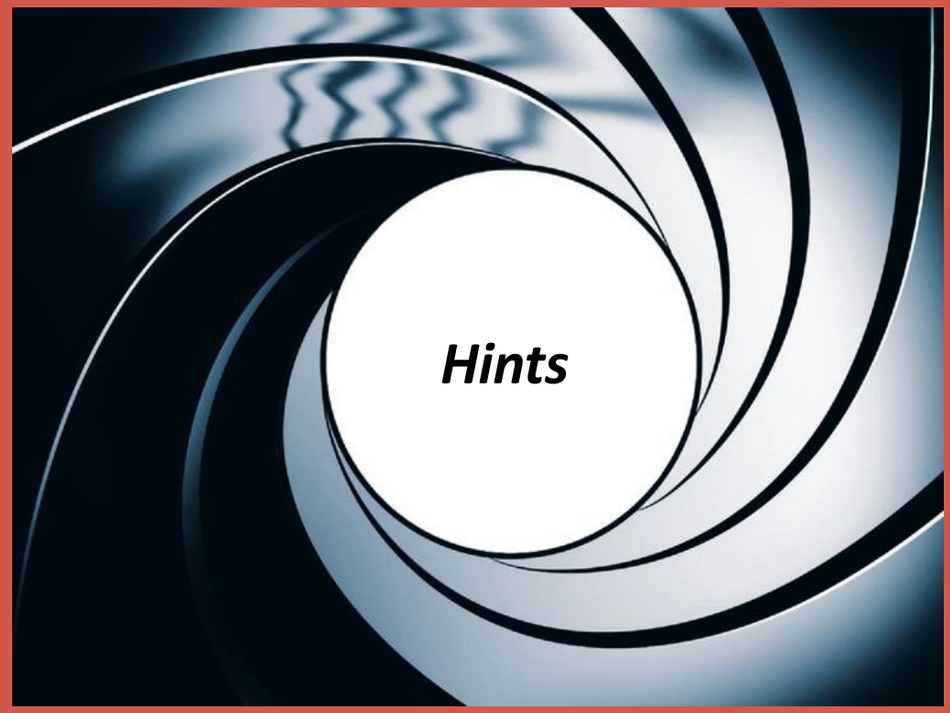
Hints “cut” branches of the optimizer decision tree

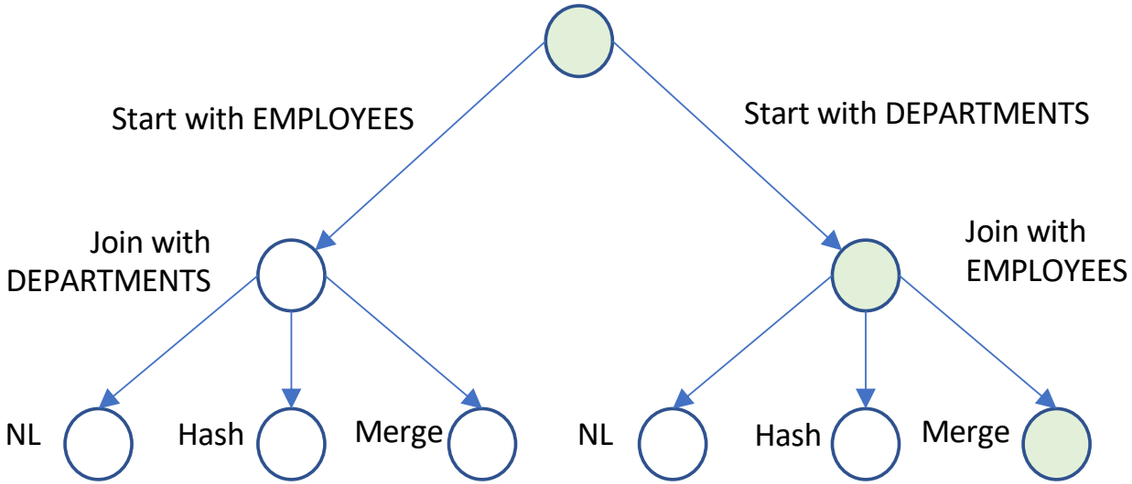
```
/*+ leading(T2) index(T1 IDX2) */
```



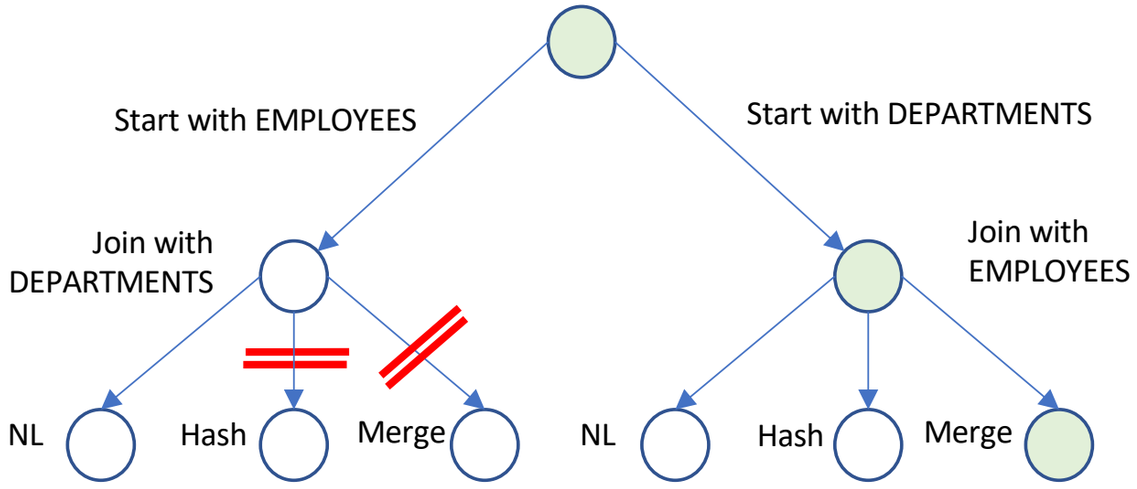
LIVE DEMO

«For Your Eyes Only»

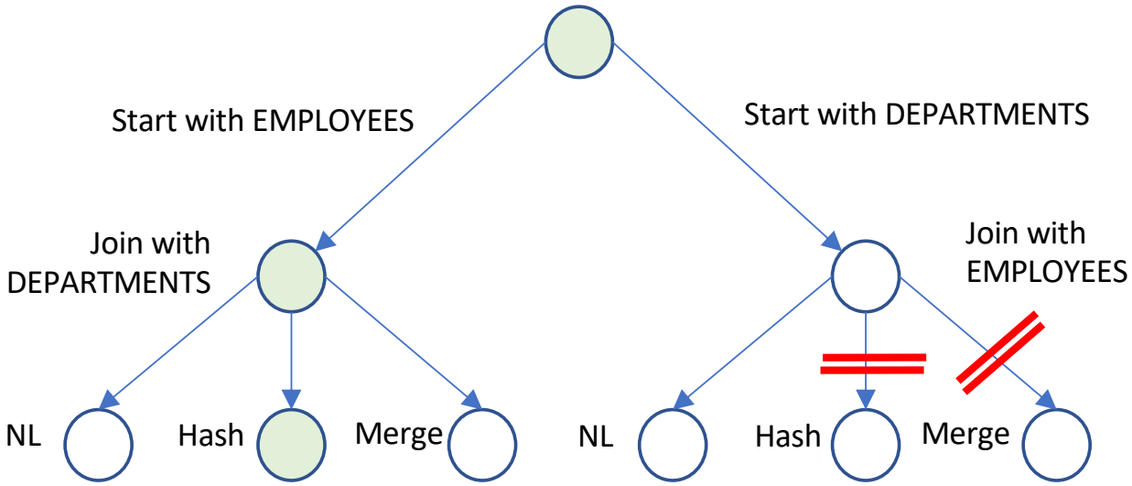




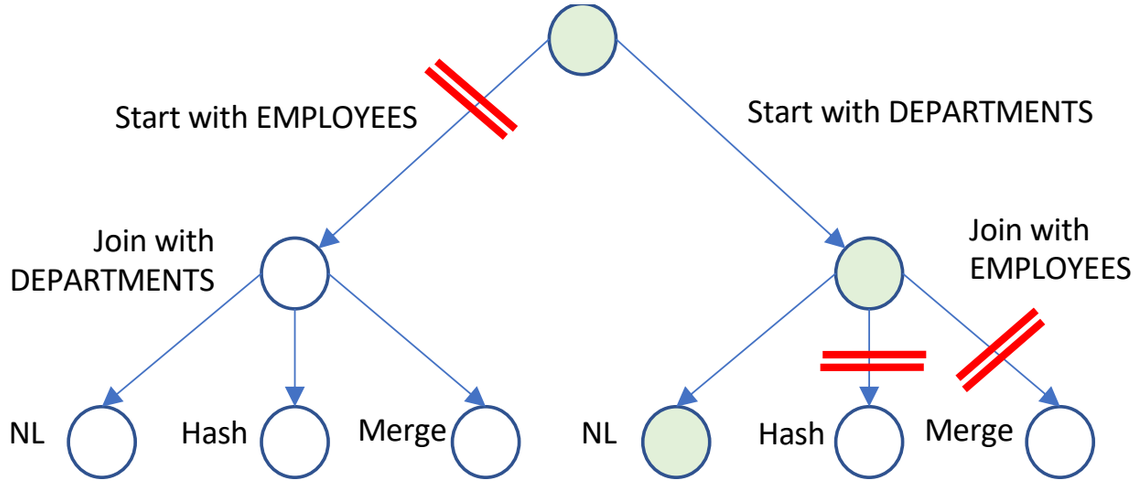

```
/*+ use_nl(d) */
```




```
/*+ use_nl(e) */
```




```
/*+ leading(d) use_nl(e) */
```



Statistics

«~~Diamonds~~ Are Forever»



Optimizer statistics are automatically gathered

Automatic Statistics Gathering

Automatic Statistics Gathering Job

- Gathering missing statistics
- Re-gathering stale statistics
- Runs every night (maintenance window)

Online Statistics Gathering

- Automatic Gathering for CREATE TABLE AS SELECT
- Automatic Gathering for Direct-Path INSERT after TRUNCATE
- Introduced with Oracle 12.1

High-Frequency Automatic Statistics Collection

- Lightweight gathering for stale statistics every 15 minutes
- Introduced with Oracle 19c (Exadata only)

Real-Time Statistics

- Online statistics gathering during DML
- Introduced with Oracle 19c (Exadata only)

Why do I still have to deal with statistics?

LIVE DEMO

«For Your Eyes Only»



«No Time to ~~Die~~ Lie»



The /*+ golden_hint */ makes every query faster

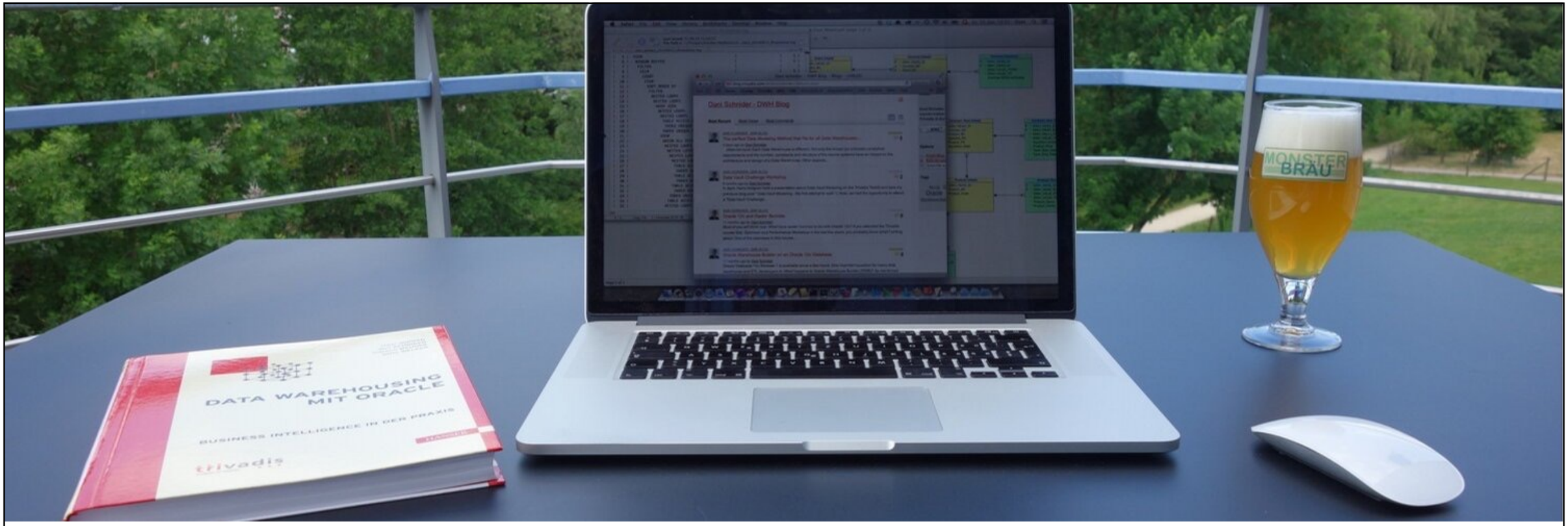
LIVE DEMO

«For Your Eyes Only»



Which Myths are true?

1	Look at performance at the end of the project	
2	Index is good, full table scan is bad	
3	COUNT(1) is faster than COUNT(*)	
4	Buy more hardware will give us good performance	
5	Hints reduce the costs of an execution plan	
6	Optimizer statistics are automatically gathered	
7	The /*+ golden_hint */ makes every query faster	



https://danischnider.wordpress.com

https://bsky.app/profile/danischnider.bsky.social

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

Q&A


DRIVEN BY EXCELLENCE
Callista