



# Nutzung der In Memory Option für SAP BW

Jörn Bartels

Architect - SAP Entwicklung

Oracle Deutschland

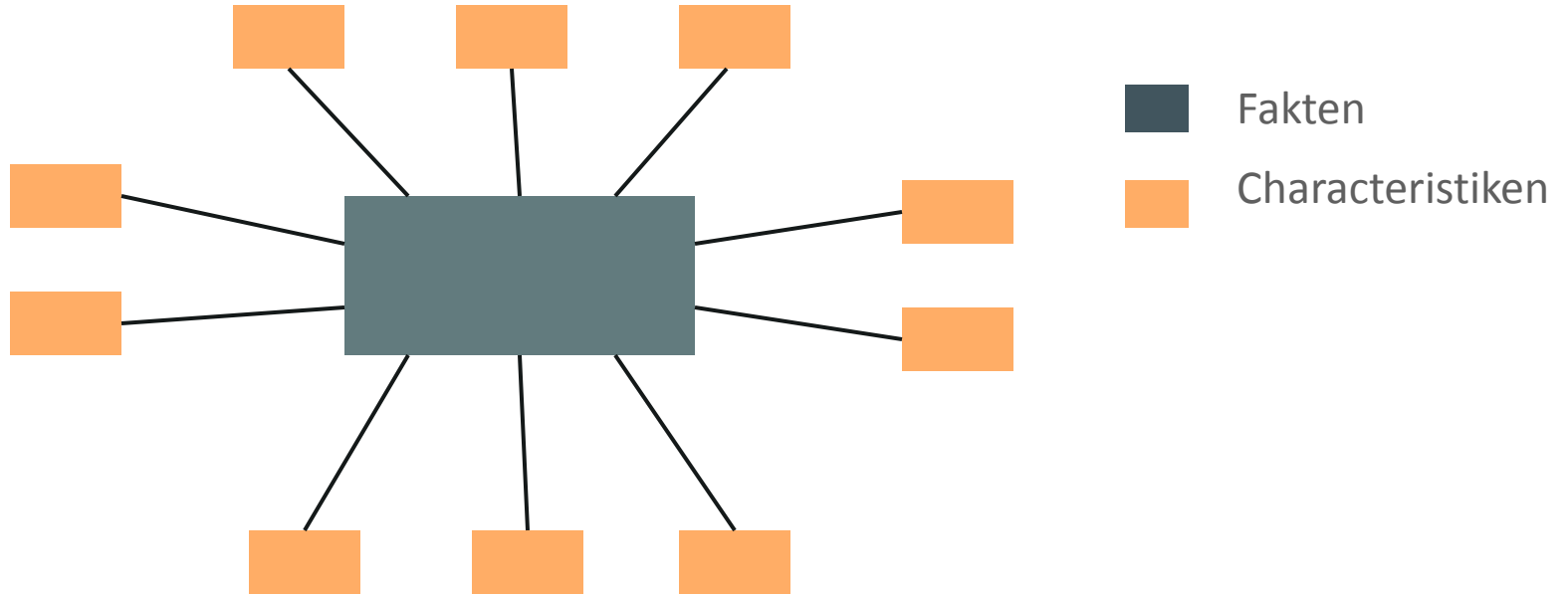
# Agenda

1 Flache Tabelle

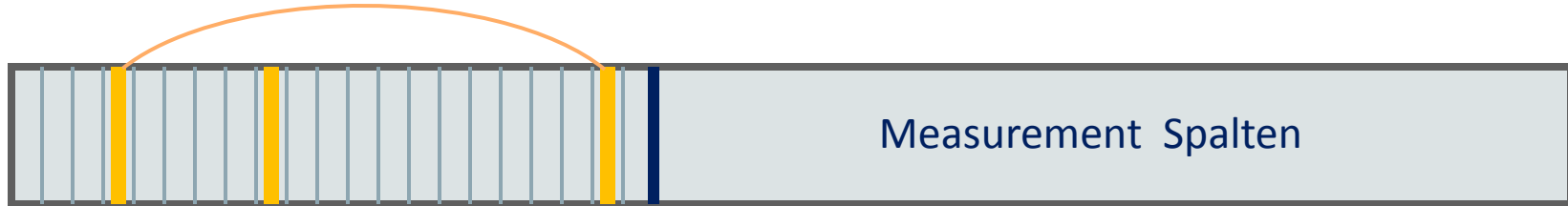
2 Star Cube

3 Flat Cube

# Flache Tabelle



# Flache Tabelle



- Nachteile
  - Viel Platzverbrauch
  - Redundante Schlüsselfelder
  - Viele Indizes
  - Langsames Laden
- Vorteile
  - Einfache Struktur

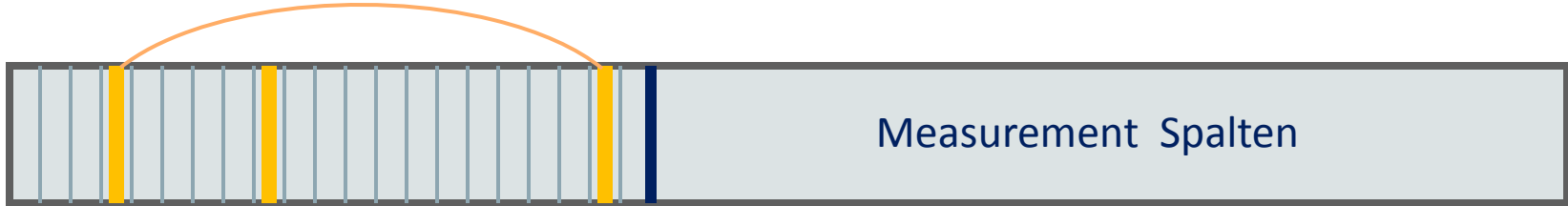
# Agenda

1 Flache Tabelle

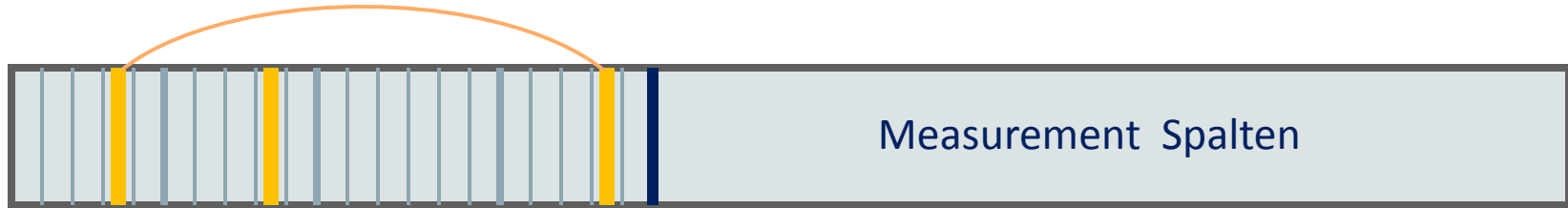
2 Star Cube

3 Flat Cube

# Star Cube

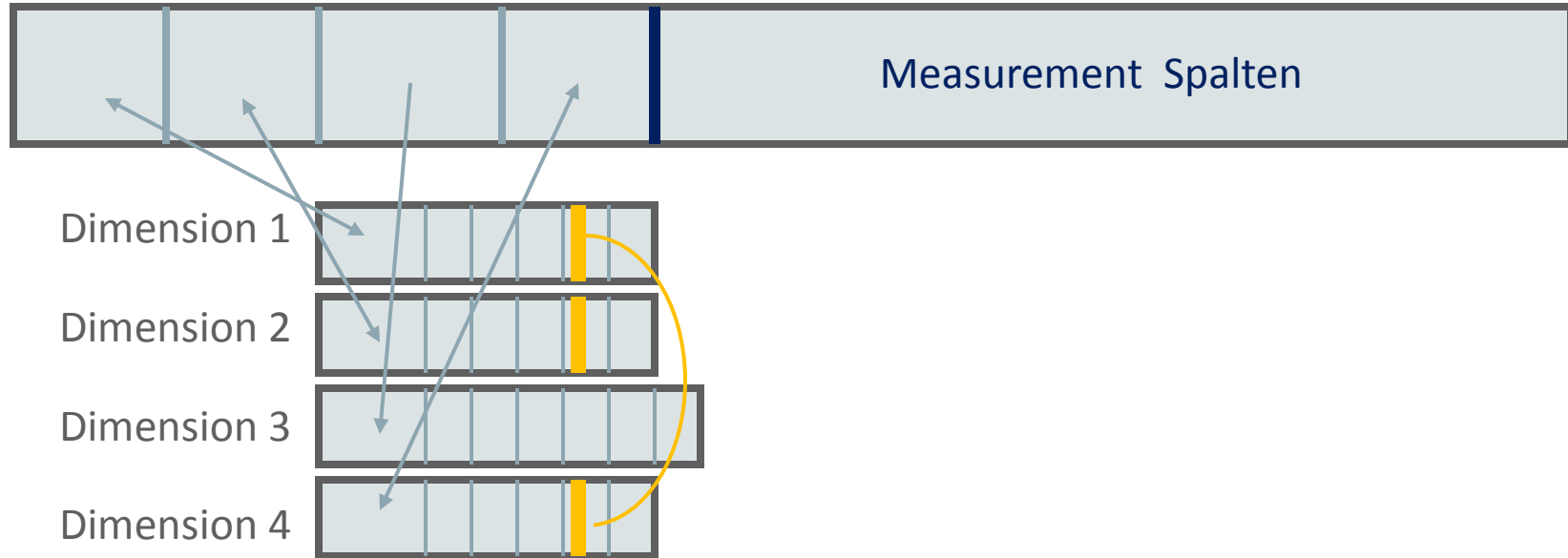


# Star Cube



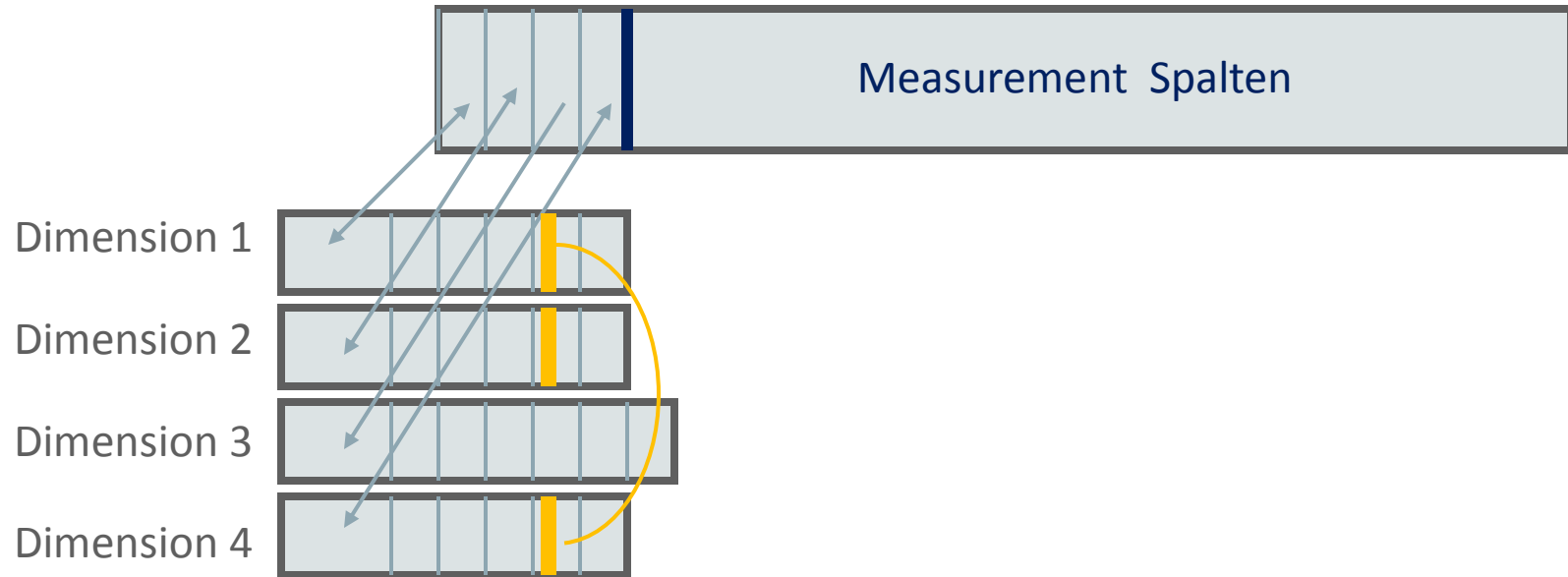
- Definition der Dimensionen

# Star Cube

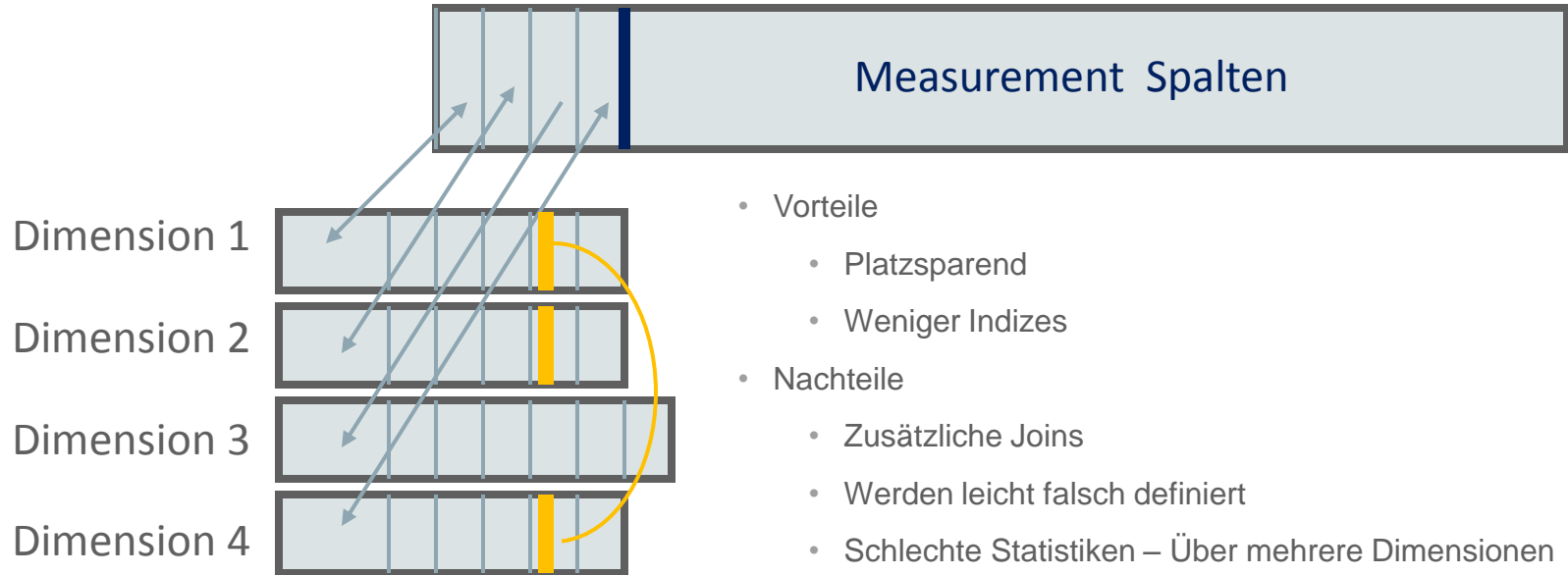




# Star Cube

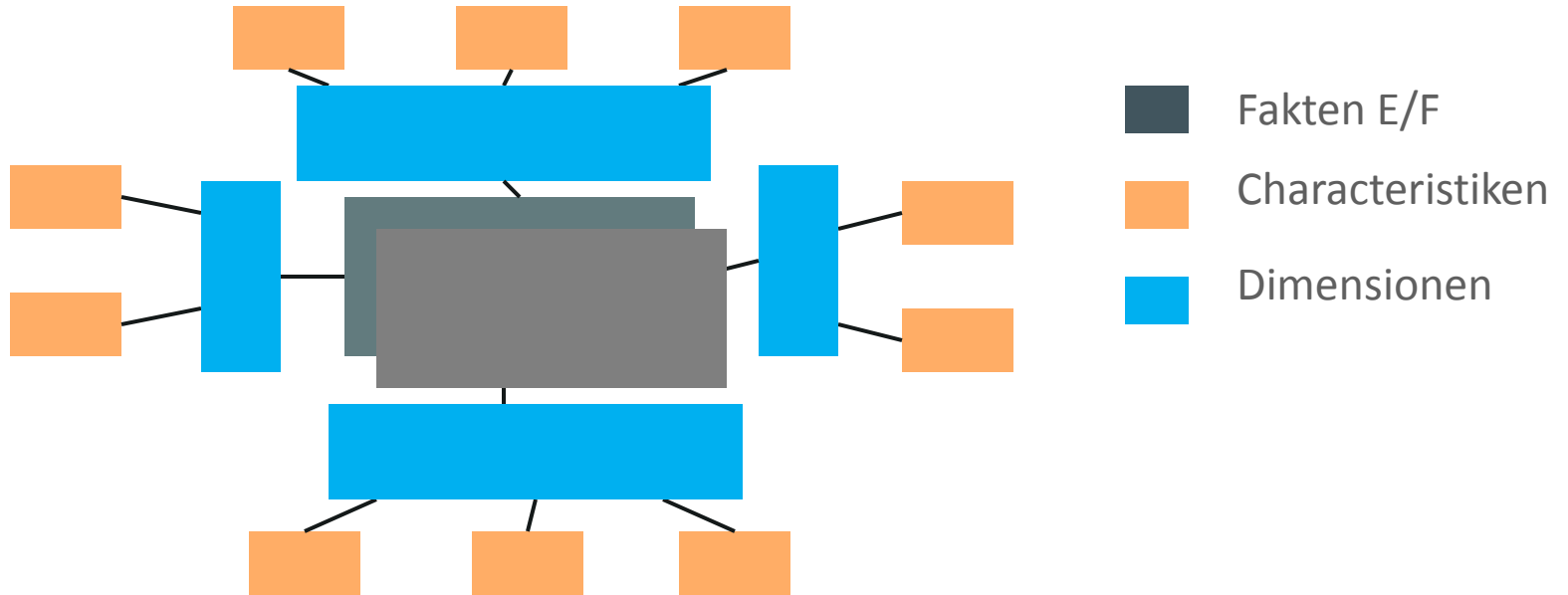


# Star Cube



- Vorteile
  - Platzsparend
  - Weniger Indizes
- Nachteile
  - Zusätzliche Joins
  - Werden leicht falsch definiert
  - Schlechte Statistiken – Über mehrere Dimensionen
  - Langsames Laden – Dimensions Ermittlung

# Star Cube (heutiger Oracle - SAP BW Standard)



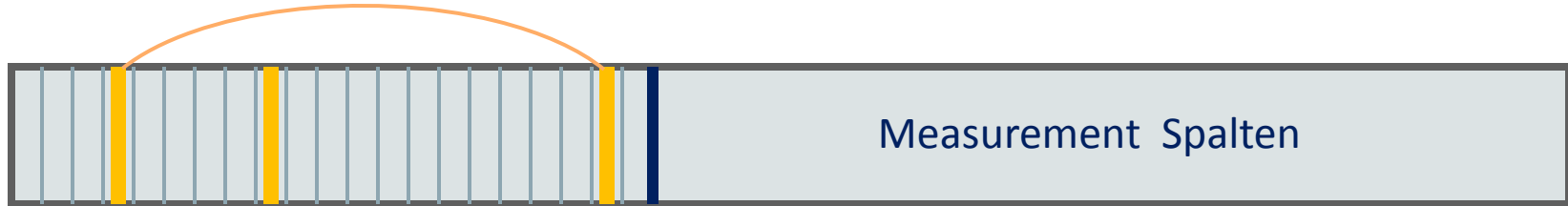
# Agenda

1 Flache Tabelle

2 Star Cube

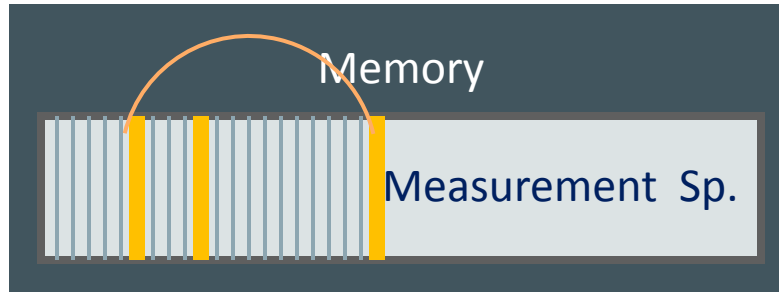
3 Flat Cube

# Flat Cube ohne In-Memory (nicht für Oracle – SAP BW)



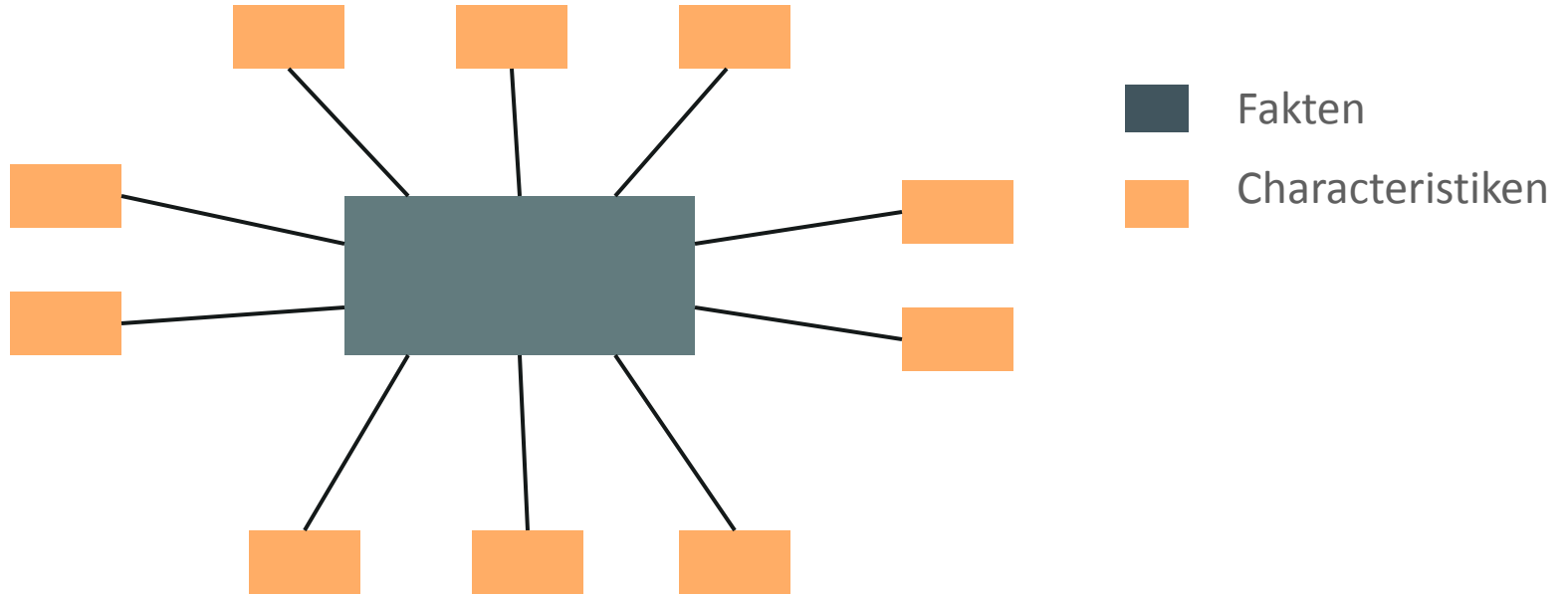
- Viel Platzverbrauch
  - Redundante Schlüsselfelder
  - Viele Indizes
  - Langsames Laden
  - Schlechte Statistiken

# Flat Cube mit In-Memory (neuer Oracle - SAP BW Standard)

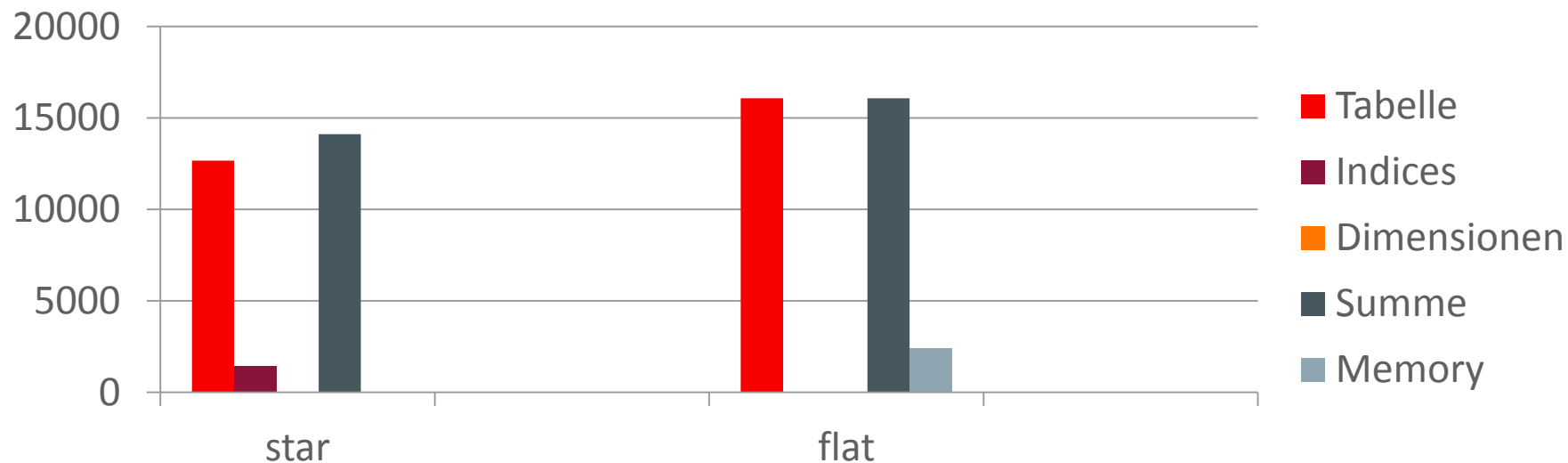


- InMemory
  - Keine Indizes notwendig
  - Keine Joins für Dimensions Tabellen
  - Schnelles Laden
  - Komprimiert
  - Optimierter InMemory Join (Vector Join)

# Flat Cube



# Größenvergleich Star - Flat



8 Dimensionen 188 Werte



# Plan Startransformation

Id	Operation	Name
0	SELECT STATEMENT	
1	SORT GROUP BY	
* 2	HASH JOIN	
* 3	TABLE ACCESS STORAGE FULL	/BIO/YVC_PROD2
* 4	HASH JOIN	
5	TABLE ACCESS STORAGE FULL	/BIC/DZMK_TCSC2
* 6	HASH JOIN	
7	TABLE ACCESS STORAGE FULL	/BIC/DZMK_TCSC4
* 8	HASH JOIN	
9	TABLE ACCESS STORAGE FULL	/BIC/DZMK_TCSCU
* 10	HASH JOIN	
11	TABLE ACCESS BY INDEX ROWID BATCHED	/BIC/DZMK_TCSCP
* 12	INDEX RANGE SCAN	/BIC/DZMK_TCSCP~01
* 13	HASH JOIN	
14	TABLE ACCESS STORAGE FULL	/BIC/DZMK_TCSC5
* 15	HASH JOIN	
* 16	TABLE ACCESS STORAGE FULL	/BIC/DZMK_TCSC7
17	PARTITION RANGE ITERATOR	
* 18	TABLE ACCESS BY LOCAL INDEX ROWID BATCHED	/BIC/EZMK_TCSC
19	BITMAP CONVERSION TO ROWIDS	
20	BITMAP AND	
21	BITMAP MERGE	
22	BITMAP KEY ITERATION	
23	BUFFER SORT	
* 24	TABLE ACCESS STORAGE FULL	/BIC/DZMK_TCSC7
* 25	BITMAP INDEX RANGE SCAN	/BIC/EZMK_TCSC~020
26	BITMAP MERGE	
27	BITMAP KEY ITERATION	
28	BUFFER SORT	
29	INDEX STORAGE FAST FULL SCAN	/BIC/DZMK_TCSC4~0
* 30	BITMAP INDEX RANGE SCAN	/BIC/EZMK_TCSC~070
31	BITMAP MERGE	
32	BITMAP KEY ITERATION	
33	BUFFER SORT	
* 34	TABLE ACCESS STORAGE FULL	/BIC/DZMK_TCSC3
* 35	BITMAP INDEX RANGE SCAN	/BIC/EZMK_TCSC~060

# Plan Flat Cube

Id	Operation	Name
0	SELECT STATEMENT	
1	SORT GROUP BY	
* 2	HASH JOIN	
3	JOIN FILTER CREATE	:BF0001
4	PART JOIN FILTER CREATE	:BF0000
5	NESTED LOOPS	
6	TABLE ACCESS BY INDEX ROWID BATCHED	/BIC/DZMK_TCSCP
* 7	INDEX RANGE SCAN	/BIC/DZMK_TCSCP~01
* 8	TABLE ACCESS STORAGE FULL	/BI0/YVC_PROD2
9	JOIN FILTER USE	:BF0001
10	PARTITION LIST JOIN-FILTER	
11	PARTITION RANGE ITERATOR	
* 12	TABLE ACCESS INMEMORY FULL	/BIC/FZMK_TCFL

# Oracle 12cR1 In-Memory Real World POC and what's coming up with 12cR2

Christian Pfundtner

DB Masters ([www.dbmasters.at](http://www.dbmasters.at))

# SAP BW/BI POC

- Customer provides a bunch of „often used slow queries“
- This queries where accessing mainly about 10 tables
- We put this tables into Column Store using different levels of compression

Objekt	DISK	FOR QUERY LOW	FOR QUERY HIGH	Remark
RSRWBSTORE	17.3 GB	14.4GB	14.2GB	4 Spalten + BLOB
/BIC/ECOPA_01	6.9G	1.7G	1.2G	Many columns mostly numbers, Rows 300 Bytes
/BIC/EEG_BCO_12	1.8G	0.97G	0.65G	Many columns mostly numbers, Rows 160 Bytes

- Queries should run as they are (no changes to statement and no changes to degree of parallelism)

# How to compare the results?

- Over all test queries
  - Original runtime production environment (still 11.2.0.x) and 20GB Buffer Cache 2746 Sec
  - Oracle 12c (only test queries), 20 GB Buffer Cache 1500 Sec
  - Oracle 12c using a 256GB Buffer Cache 380 Sec
  - Oracle 12c using 20GB BC and 80GB In-Memory 100 Sec
- So having sufficient Buffer Cache will also speed it up quite well but using the memory with the In-Memory functionality does a quite good job here!
  - One Query got slower by using In-Memory
  - All others where faster
  - Out of POC testing with using parallel (2, 4 and 8) shows also speeding up

## SAP BW/BI POC - conclusion

- SAP BW/BI definitely is a very good candidate for Oracle In-Memory
- SAP certified Oracle In-Memory a few weeks ago (and Oracle was able to run the SAP BW-EML Benchmark more than 2 times faster compared to SAP HANA on the same hardware platform)
- With In-Memory you are able to speed up most queries within an SAP BW/BI Environment – maybe not by 100 times but mostly by at least 5-10 times.

# Examples (not an original query)

- Example for just doing aggregation

```
select /*+ FULL(s) */ sum(KEY_EG_BCO_12P), sum(KEY_EG_BCO_127)
from SAPEBI."/BIC/EEG_BCO_12" s;
```

- Execution Time:
  - Buffer Cache: about 7 seconds
  - In-Memory: about 70 ms ... which is 100 times faster

# Q & A



**Fragen werden gerne gesehen!**  
**Und es werden Pilot Kunden gesucht!**



ORACLE®