Oracle-Statistiken

im Data Warehouse effizient nutzen

21.11.2012

Reinhard Mense

CTO, ARETO Consulting

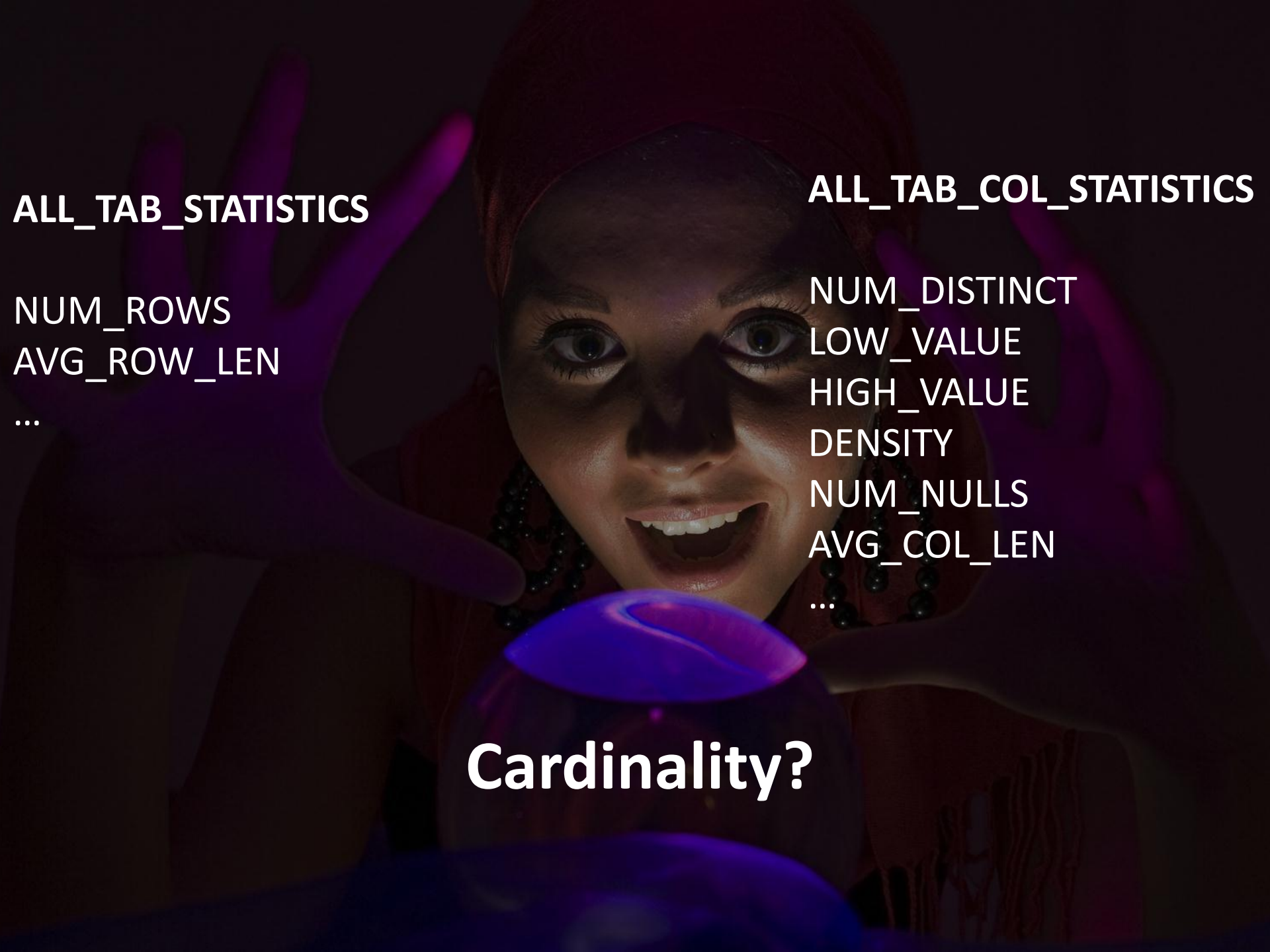
ARETO consulting

Statistiken

welche? wie? wann?

Statistiken

welche? wie? wann?



ALL_TAB_STATISTICS

NUM_ROWS
AVG_ROW_LEN
…

ALL_TAB_COL_STATISTICS

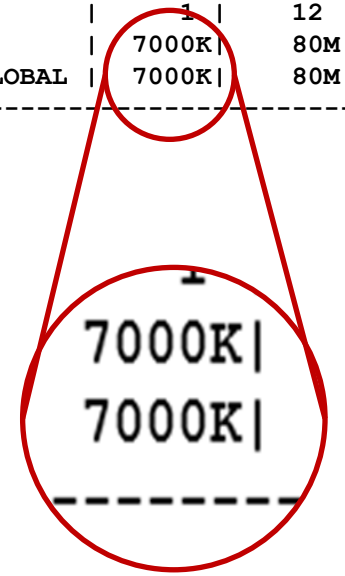
NUM_DISTINCT
LOW_VALUE
HIGH_VALUE
DENSITY
NUM_NULLS
AVG_COL_LEN
…

Cardinality?

Globale und lokale Statistiken

```
select sum (umsatz)
  from fakt_verkauf
 where datum between to_date ('01.01.2012', 'dd.mm.yyyy')
                 and <ende>
```

```
-----------------------------------------------------------------------------------------------------------------------
| Id  | Operation                  | Name                      | Rows  | Bytes | Cost (%CPU)| Time     | Pstart| Pstop |
-----------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT           |                           |     1 |    12 |  6215   (2)| 00:01:15 |       |       |
|   1 |  SORT AGGREGATE            |                           |     1 |    12 |            |          |       |       |
|   2 |   PARTITION RANGE ITERATOR|                           |  7000K|    80M|  6215   (2)| 00:01:15 |     1 |     7 |
|*  3 |    TABLE ACCESS FULL       | TF_VERKAUF_LOCAL_GLOBAL |  7000K|    80M|  6215   (2)| 00:01:15 |     1 |     7 |
-----------------------------------------------------------------------------------------------------------------------
```



Cardinality

```
select sum (umsatz) from fakt_verkauf
where datum between to_date ('01.01.2012', 'dd.mm.yyyy') and <ende>
```

<ende>	#rows	Ohne Stat.	nur lokale Stat. (ohne MAXVALUE)	nur lokale Stat. (mit MAXVALUE)	lokale + globale Stat.
01.01.2012	1.000.000	989.310	1.000.000	1.000.000	1.000.000
02.01.2012	2.000.000	1.912.425	1.999.910	2.000.000	2.000.000
03.01.2012	3.000.000	2.868.793	2.999.910	3.000.000	3.000.000
04.01.2012	4.000.000	4.220.568	3.999.910	4.000.000	4.000.000
05.01.2012	5.000.000	5.878.273	4.999.907	5.000.000	5.000.000
06.01.2012	6.000.000	5.602.747	5.999.909	6.000.000	6.000.000
07.01.2012	7.000.000	8.255.838	6.999.910	7.000.000	7.000.000

Cardinality – out of range condition

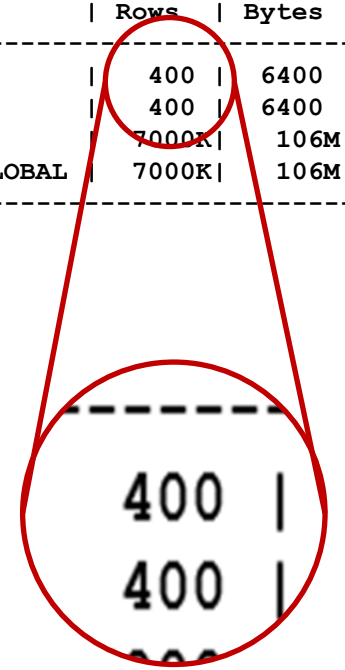
```
select sum (umsatz) from fakt_verkauf
where datum between to_date ('01.01.2012', 'dd.mm.yyyy') and to_date ('08.01.2012', 'dd.mm.yyyy')
```

<ende>	#rows	Ohne Stat.	nur lokale Stat. (ohne MAXVALUE)	nur lokale Stat. (mit MAXVALUE)	lokale + globale Stat.
01.01.2012	1.000.000	889.110	790.371	1.000.000	1.000.000
02.01.2012	2.000.000	2.455.191	1.673.551	2.000.000	2.000.000
03.01.2012	3.000.000	3.309.624	3.672.977	3.000.000	3.000.000
04.01.2012	4.000.000	2.838.101	3.666.418	4.000.000	4.000.000
05.01.2012	5.000.000	5.193.334	4.918.003	5.000.000	5.000.000
06.01.2012	6.000.000	6.776.543	6.988.246	6.000.000	6.000.000
07.01.2012	7.000.000	7.018.066	7.920.052	7.000.000	7.000.000

Globale und lokale Statistiken

```
select produkt_id
     , sum (umsatz)
  from fakt_verkauf
 where datum between to_date ('01.01.2012', 'dd.mm.yyyy')
                 and <ende>
 group by produkt_id
```

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time	Pstart	Pstop
0	SELECT STATEMENT		400	6400	6564 (8)	00:01:19		
1	HASH GROUP BY		400	6400	6564 (8)	00:01:19		
2	PARTITION RANGE ITERATOR		7000K	106M	6216 (2)	00:01:15	1	7
* 3	TABLE ACCESS FULL	TF_VERKAUF_LOCAL_GLOBAL	7000K	106M	6216 (2)	00:01:15	1	7



Cardinality

```
select produkt_id, sum (umsatz) from fakt_verkauf
where datum between to_date ('01.01.2012', 'dd.mm.yyyy') and <ende>
group by produkt_id
```

<ende>	produkt_id	#rows	Ohne Stat.	nur lokale Stat. (ohne MAXVALUE)	nur lokale Stat. (mit MAXVALUE)	lokale + globale Stat.
01.01.2012	1-100	100	989.310	100	100	100
02.01.2012	1-50 101-150	150	1.912.425	134	100	150
03.01.2012	1-50 151-200	200	2.868.793	159	100	200
04.01.2012	1-50 201-250	250	4.220.568	179	100	250
05.01.2012	1-50 251-300	300	5.878.273	195	100	300
06.01.2012	1-50 301-350	350	5.602.747	210	100	350
07.01.2012	1-50 351-400	400	8.255.838	222	100	400

Cardinality – out of range condition

```
select produkt_id, sum (umsatz) from fakt_verkauf
where datum between to_date ('01.01.2012', 'dd.mm.yyyy') and to_date ('08.01.2012', 'dd.mm.yyyy')
group by produkt_id
```

<ende>	produkt_id	#rows	Ohne Stat.	nur lokale Stat. (ohne MAXVALUE)	nur lokale Stat. (mit MAXVALUE)	lokale + globale Stat.
01.01.2012	1-100	100	889.110	790.371	100	100
02.01.2012	1-50 101-150	150	2.455.191	1.673.551	100	150
03.01.2012	1-50 151-200	200	3.309.624	3.672.977	100	200
04.01.2012	1-50 201-250	250	2.838.101	3.666.418	100	250
05.01.2012	1-50 251-300	300	5.193.334	4.918.003	100	300
06.01.2012	1-50 301-350	350	6.776.543	6.988.246	100	350
07.01.2012	1-50 351-400	400	7.018.066	7.920.052	100	400

Globale und lokale Statistiken

1 Partition

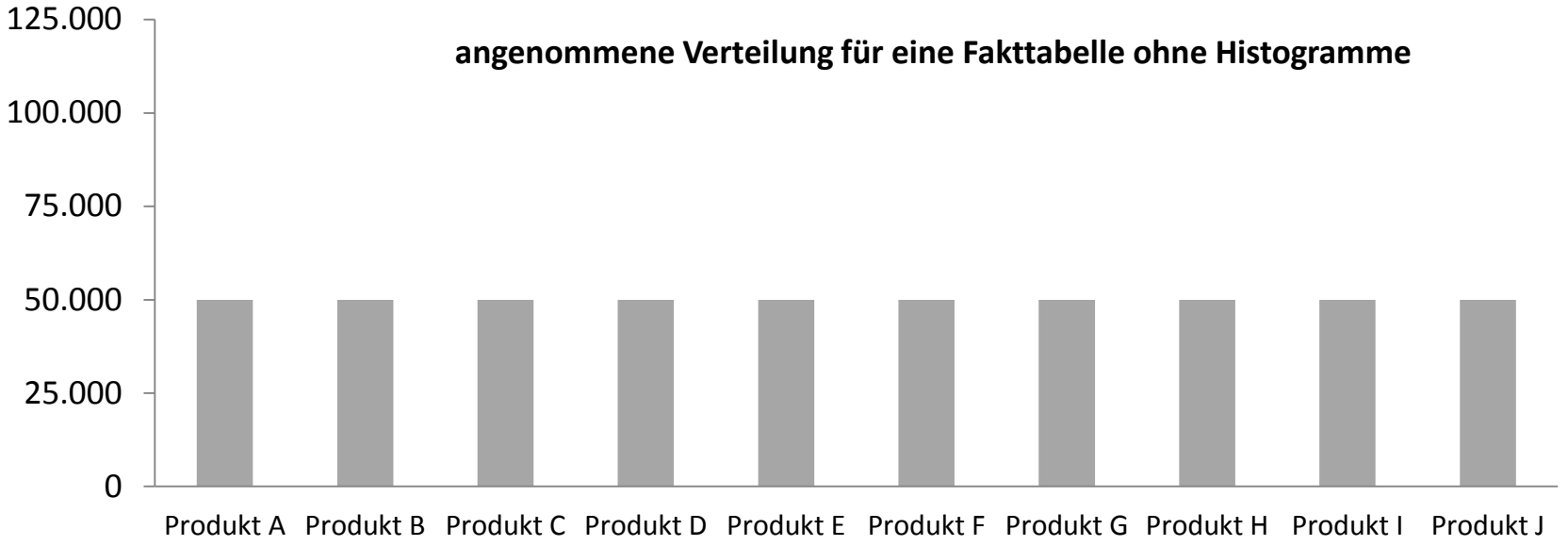
→ lokale Statistiken

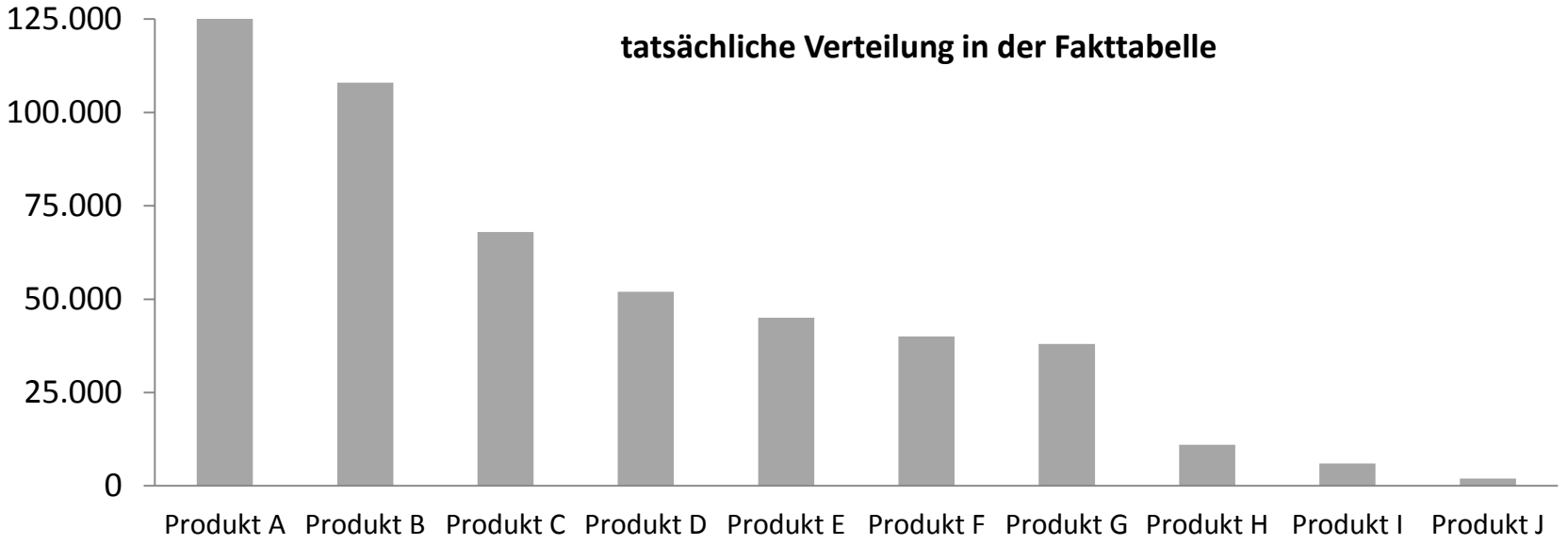
>1 Partition

→ lokale und globale Statistiken

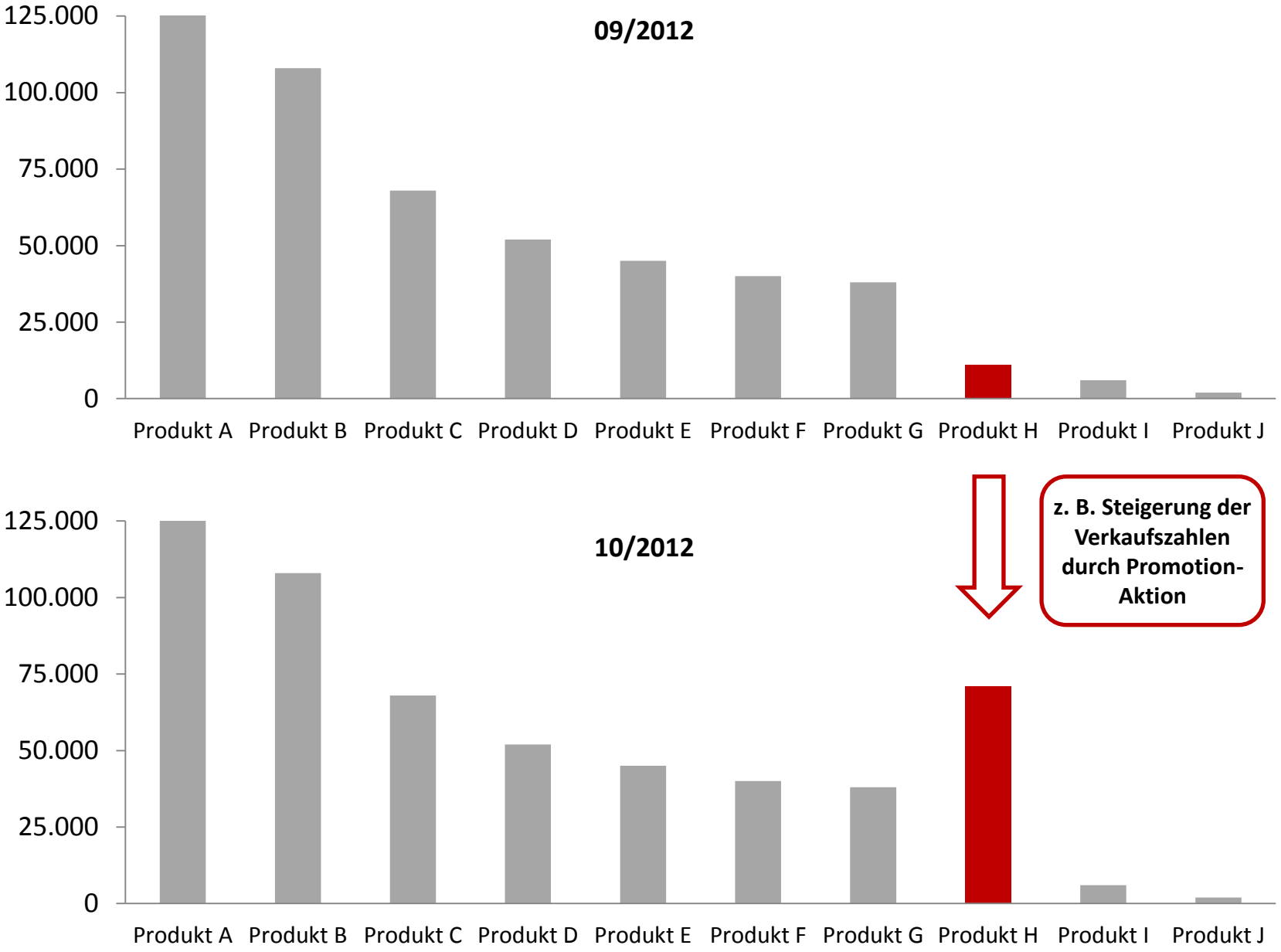

Histogramme

Verteilung der Daten

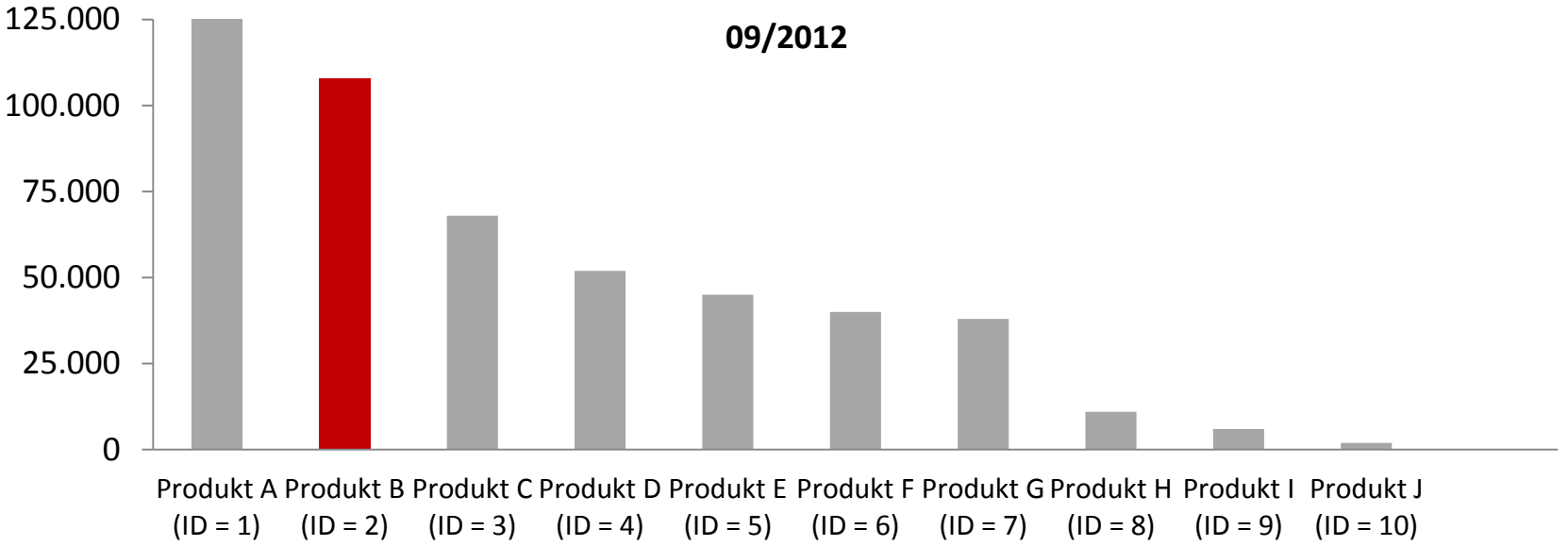


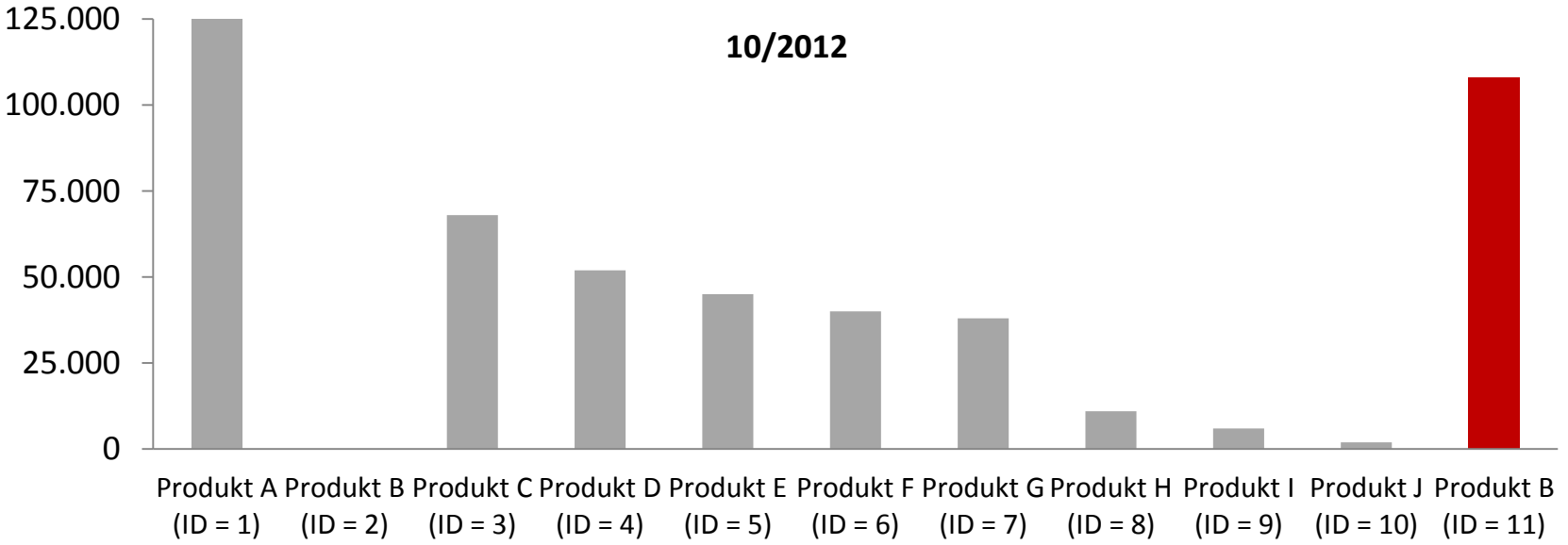


Daten verändern sich



Surrogate Keys





Histogramme und Bind Variablen

Oracle 10g

Bind Variable Peeking legt den Ausführungsplan fest.

Oracle 11g

Optimizer nutzt Adaptive Cursor Sharing.
(bind sensitive, bind-aware)

DWH-Systeme

Front End Tools verwenden oft keine Bind Variablen.

Korrelierende Spalten

Produktdimension
25.000 Produkte – 500 Produktgruppen – 50 Produktkategorien

```
select *
  from dim_produkt
 where produktgruppe = 'Obst'
   and produktkategorie = 'Lebensmittel'
```

```
ALL_TABLES

TABLE_NAME        NUM_ROWS
----------------- ------------
DIM_PRODUKT              25000

ALL_TAB_COL_STATISTICS

COLUMN_NAME       NUM_DISTINCT
----------------- ------------
PRODUKTGRUPPE             500
PRODUKTKATEGORIE           50
```

25.000 * (1 / 500) * (1 / 50)

```
-------------------------------------------------------------------------------
| Id  | Operation          | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |              |     1 |    63 |    90   (0)| 00:00:02 |
|*  1 |  TABLE ACCESS FULL| DIM_PRODUKT  |     1 |    63 |    90   (0)| 00:00:02 |
-------------------------------------------------------------------------------
```

Extended Statistics

```
declare
  vResult varchar2 (4000);
begin
  vResult := dbms_stats.create_extended_stats
              (ownname => 'MART'
             , tabname => 'DIM_PRODUKT'
             , extension => '(produktgruppe, produktkategorie)');

  dbms_stats.gather_table_stats
   (ownname => 'MART'
  , tabname => 'DIM_PRODUKT'
  , method_opt => 'for columns (produktgruppe, produktkategorie)');
end;
/
```

```
ALL_TAB_COL_STATISTICS

COLUMN_NAME                    NUM_DISTINCT
------------------------------ ------------
PRODUKTGRUPPE                           500
PRODUKTKATEGORIE                         50
SYS_STUBJ8KAS865OF3VM3Z95_NB5X          500
```

Extended Statistics

Produktdimension
25.000 Produkte – 500 Produktgruppen – 50 Produktkategorien

```
select *
  from dim_produkt
 where produktgruppe = 'Obst'
   and produktkategorie = 'Lebensmittel'
```

25.000 * (1 / 500)

ALL_TABLES

TABLE_NAME	NUM_ROWS
DIM_PRODUKT	25000

ALL_TAB_COL_STATISTICS

COLUMN_NAME	NUM_DISTINCT
PRODUKTGRUPPE	500
PRODUKTKATEGORIE	50
SYS_STUBJ8KAS…	500

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		50	3150	90 (0)	00:00:02
*1	TABLE ACCESS FULL	DIM_PRODUKT	50	3150	90 (0)	00:00:02

Extended Statistics

```
select v.*
  from dim_produkt p
     , fakt_verkauf v
 where p.produkt_id = v.produkt_id
   and p.produktgruppe = 'Obst'
   and p.produktkategorie = 'Lebensmittel';
```

ohne Extended Statistics

```
-------------------------------------------------------------------------------------------
| Id  | Operation                     | Name           | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |                |   100 |  2600 |     111 (0)| 00:00:02 |
|   1 |  NESTED LOOPS                 |                |       |       |            |          |
|   2 |   NESTED LOOPS                |                |   100 |  2600 |     111 (0)| 00:00:02 |
|*  3 |    TABLE ACCESS FULL          | DIM_PRODUKT    |     1 |    11 |      45 (0)| 00:00:01 |
|   4 |    BITMAP CONVERSION TO ROWIDS|                |       |       |            |          |
|*  5 |     BITMAP INDEX SINGLE VALUE | BX_PRODUKT_ID  |       |       |            |          |
|   6 |   TABLE ACCESS BY INDEX ROWID | FAKT_VERKAUF   |   100 |  1500 |     111 (0)| 00:00:02 |
-------------------------------------------------------------------------------------------
```

mit Extended Statistics

```
-----------------------------------------------------------------------------------------
| Id  | Operation           | Name           | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |                |     1 |    19 |   939   (2)| 00:00:12 |
|   1 |  SORT AGGREGATE     |                |     1 |    19 |            |          |
|*  2 |   HASH JOIN         |                |  5000 | 95000 |   939   (2)| 00:00:12 |
|*  3 |    TABLE ACCESS FULL| DIM_PRODUKT    |    50 |   600 |    90   (0)| 00:00:02 |
|   4 |    TABLE ACCESS FULL| FAKT_VERKAUF   |  1000K|  6835K|   843   (2)| 00:00:11 |
-----------------------------------------------------------------------------------------
```

Dynamic Sampling

Level	Dynamic Sampling	Stichprobe
0	kein Dynamic Sampling	
1	für alle nicht analysierten Tabellen	32 Blöcke
2	für alle nicht analysierten Tabellen	64 Blöcke
3	wie Level 2 zusätzlich alle Tabellen bei der die Standard-Selektivitäts-Schätzung bei einem Prädikat eine Vermutung treffen musste, das ein potentieller Kandidat für Dynamic Sampling ist	64 Blöcke für nicht analysierte Tabellen, 32 Blöcke für den Rest
4	wie Level 3 zusätzlich alle Tabellen mit Prädikaten, die **zwei oder mehr Spalten** in der gleichen Tabelle referenzieren	64 Blöcke für nicht analysierte Tabellen, 32 Blöcke für den Rest
5	wie Level 4	64 Blöcke
6	wie Level 4	128 Blöcke
7	wie Level 4	256 Blöcke
8	wie Level 4	1024 Blöcke
9	wie Level 4	4096 Blöcke
10	wie Level 4	alle Blöcke

Dynamic Sampling

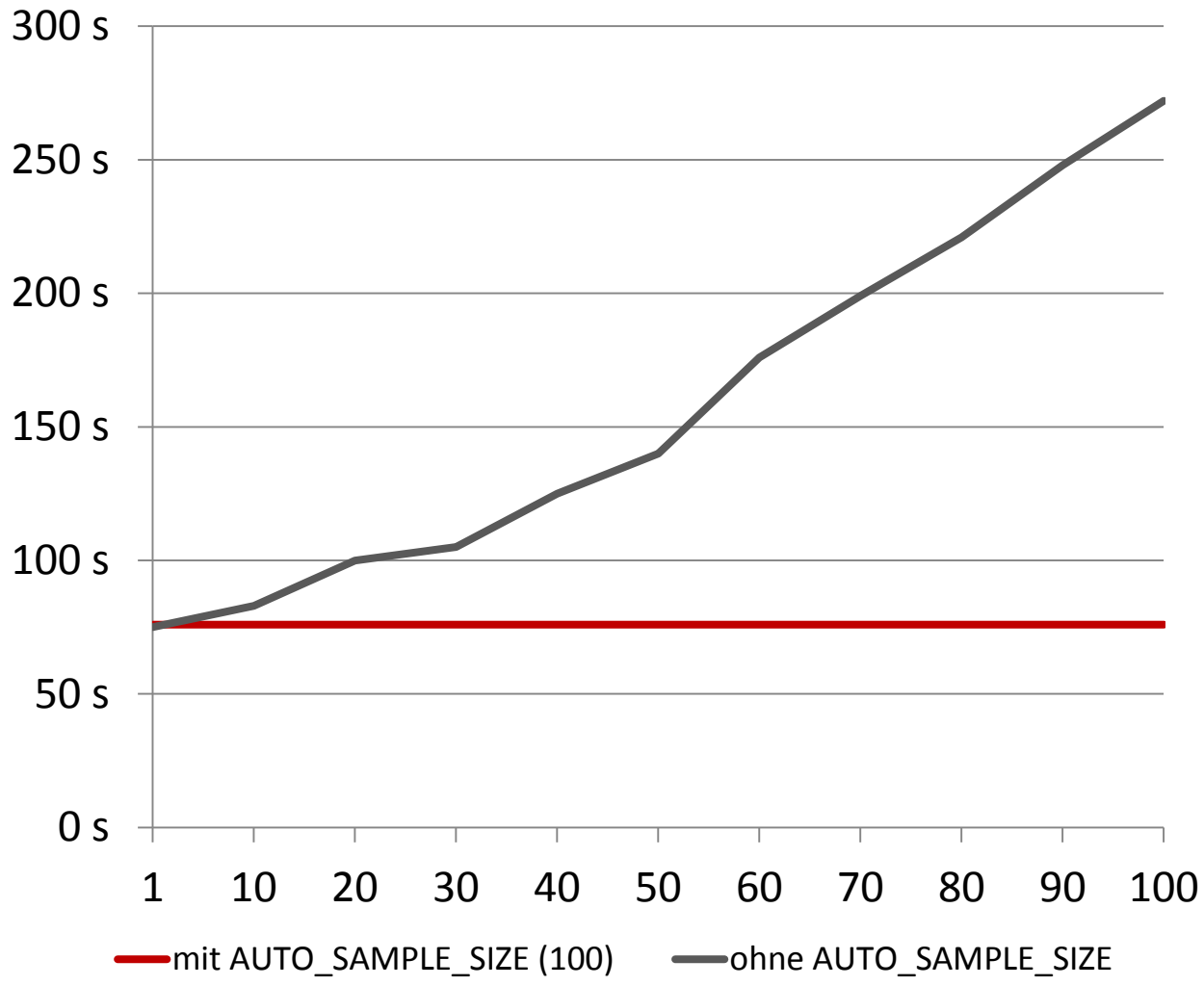
```
alter session set optimizer_dynamic_sampling = 8
```

```
-------------------------------------------------------------------------------------
| Id  | Operation           | Name           | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------

PLAN_TABLE_OUTPUT
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |                |     1 |    19 |   939   (2)| 00:00:12 |
|   1 |  SORT AGGREGATE     |                |     1 |    19 |            |          |
|*  2 |   HASH JOIN         |                |  5000 | 95000 |   939   (2)| 00:00:12 |
|*  3 |    TABLE ACCESS FULL| DIM_PRODUKT    |    50 |   600 |    90   (0)| 00:00:02 |
|   4 |    TABLE ACCESS FULL| FAKT_VERKAUF   |  1000K|  6835K|   843   (2)| 00:00:11 |
-------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------
   2 - access("P"."PRODUKT_ID"="V"."PRODUKT_ID")
   3 - filter("P"."PRODUKTGRUPPE"='Obst' AND "P"."PRODUKTKATEGORIE"='Lebensmittel')

Note
-----
   - dynamic sampling used for this statement (level=8)
```

Statistiken

welche? wie? wann?

Sample Rate



40.000.000 rows, degree 8, keine Histogramme

Sample Rate

Ergebnisse vom TPC-H Benchmark
230 GB Tabelle

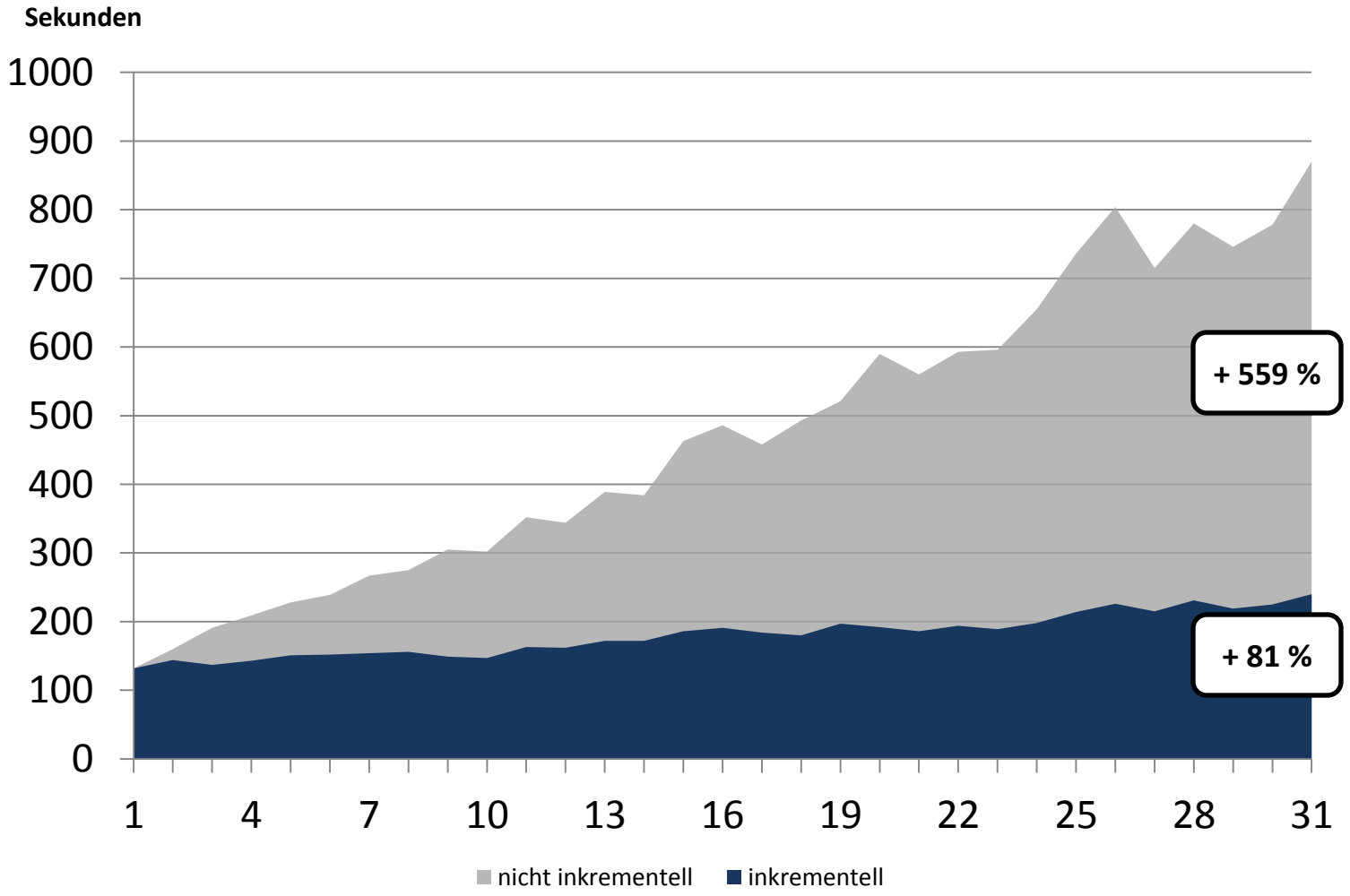
	1% SAMPLE	AUTO_SAMPLE_SIZE	100% SAMPLE
Laufzeit (Sekunden)	797	1.908	18.772
NUM_DISTINCT für Spalte L_ORDERKEY	225.000.000	450.000.000	450.000.000
NUM_DISTINCT für Spalte L_COMMENT_COLUMN	7.244.885	177.499.684	181.122.127

Quelle: Best Practises for Gathering Optimizer Statistics, Oracle

Inkrementelle globale Statistiken

```
dbms_stats.set_table_prefs
 ('MART', 'FAKT_VERKAUF', 'INCREMENTAL','TRUE');

dbms_stats.gather_table_stats
 ('MART','FAKT_VERKAUF');
```

Inkrementelle globale Statistiken



Statistiken

welche? wie? wann?

Cardinality – out of range condition

```
select sum (umsatz) from fakt_verkauf
where datum between <start> and <ende>
```

<start>	<ende>	#rows	Lokale + globale Stat. vom 07.01.2012	lokale + akt. globale Stat.
01.01.2012	07.01.2012	7.000.000	7.000.000	7.000.000
02.01.2012	08.01.2012	7.000.000	5.935.199	7.000.000
03.01.2012	09.01.2012	7.000.000	4.970.561	7.000.000
04.01.2012	10.01.2012	7.000.000	3.938.297	7.000.000
05.01.2012	11.01.2012	7.000.000	2.963.452	7.000.000
06.01.2012	12.01.2012	7.000.000	1.943.948	7.000.000
07.01.2012	13.01.2012	7.000.000	978.035	7.000.000
08.01.2012	14.01.2012	7.000.000	482.319	7.000.000

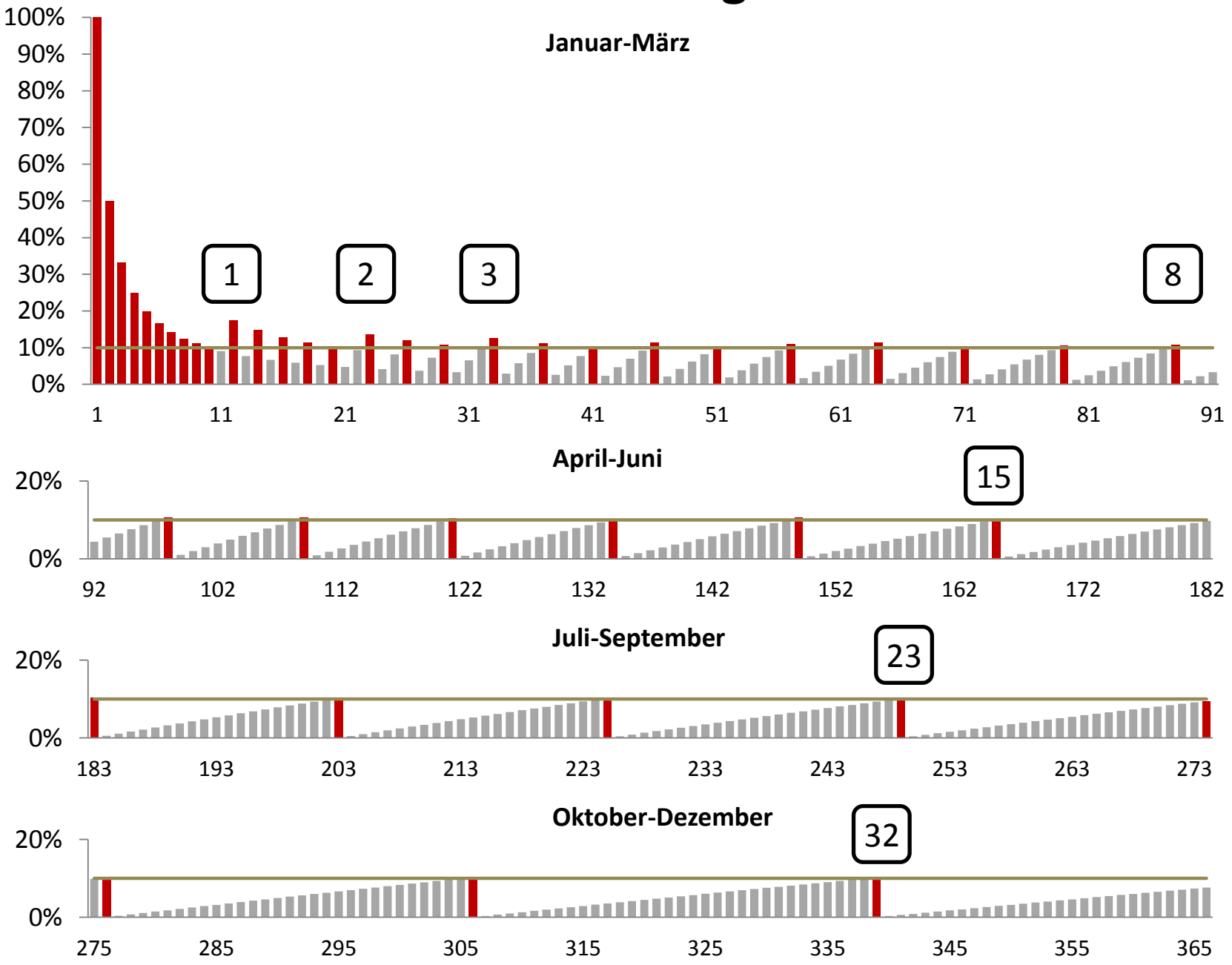
Ist der automatische Statistik-Job geeignet?

```
DBMS_STATS.GATHER_DATABASE_STATS_JOB_PROC
```

STALE_PERCENT = 10% Default

```
DBMS_STATS.SET_GLOBAL_PREFS
('STALE_PERCENT', '5')

DBMS_STATS.SET_TABLE_PREFS
(<ownname>, <tabname>, 'STALE_PERCENT', '5')
```

Monitoring


Januar-März
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
1
2
3
8
1
11
21
31
41
51
61
71
81
91
April-Juni
20%
0%
15
92
102
112
122
132
142
152
162
172
182
Juli-September
20%
0%
23
183
193
203
213
223
233
243
253
263
273
Oktober-Dezember
20%
0%
32
275
285
295
305
315
325
335
345
355
365

Fazit

- Globale + lokale Statistiken
- Histogramme
- Extended Statistics
- Dynamic Sampling
- AUTO_SAMPLE_SIZE
- Inkrementelle globale Statistiken
- Statistiken im ETL-Prozess aktualisieren

ARETO Consulting GmbH
51063 Köln
www.areto-consulting.de

Reinhard Mense
CTO, ARETO Consulting
reinhard.mense@areto-consulting.de