DataArt

Alexander Tokarev

In-Memory Computing Summit Europe 2018

Speed-of-light faceted search

Oracle In-Memory from trenches

Who am I

- Alexander Tokarev
- Age 38
- Database performance architect at DataArt:
  1. Solution design
  2. Performance tuning
  3. POCs and crazy ideas
- First experience with Oracle - 2001, Oracle 8.1.7
- First experience with In-Memory solutions - 2015
- Lovely In-Memory databases:
  1. Oracle InMemory
  2. Exasol
  3. Tarantool
- Hobbies: spearfishing, drums

Who is my employer

DataArt

Consulting, solution design, re-engineering

20 development centers

>2500 employees

Famous clients: NASDAQ
S&P
Ocado
JacTravel
Maersk
Regus and etc

Overview

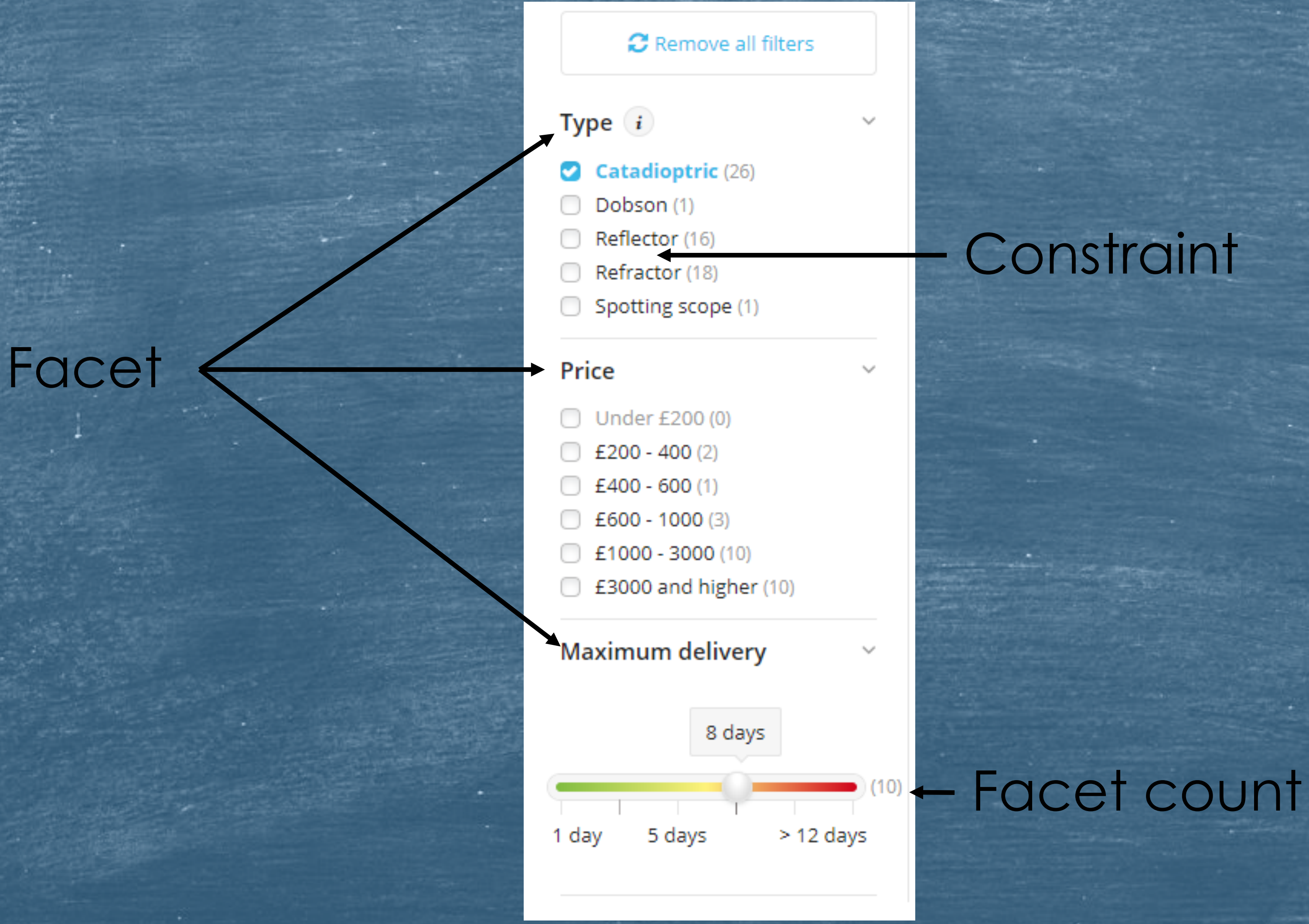
- Faceted search
- Project
  - Architecture
  - Faceted search place
  - Performance issues
- In-memory internals
- Implementation steps and traps
- Key findings
- Conclusion
- Q&A

Safe Harbor Statement

The presentation may include predictions, estimates or other information that might be considered forward-looking. While these forward-looking statements represent our current judgment on what the future holds, they are subject to risks and uncertainties that could cause actual results to differ materially. You are cautioned not to place undue reliance on these forward-looking statements, which reflect our opinions only as of the date of this presentation.

We are not obligating ourselves to revise or publicly release the results of any revision to these statements in light of new information or future events.

Faceted search



Remove all filters

Type i

- ☑ **Catadioptric** (26)
- □ Dobson (1)
- □ Reflector (16)
- □ Refractor (18)
- □ Spotting scope (1)

Price

- □ Under £200 (0)
- □ £200 - 400 (2)
- □ £400 - 600 (1)
- □ £600 - 1000 (3)
- □ £1000 - 3000 (10)
- □ £3000 and higher (10)

Maximum delivery

8 days

(10)

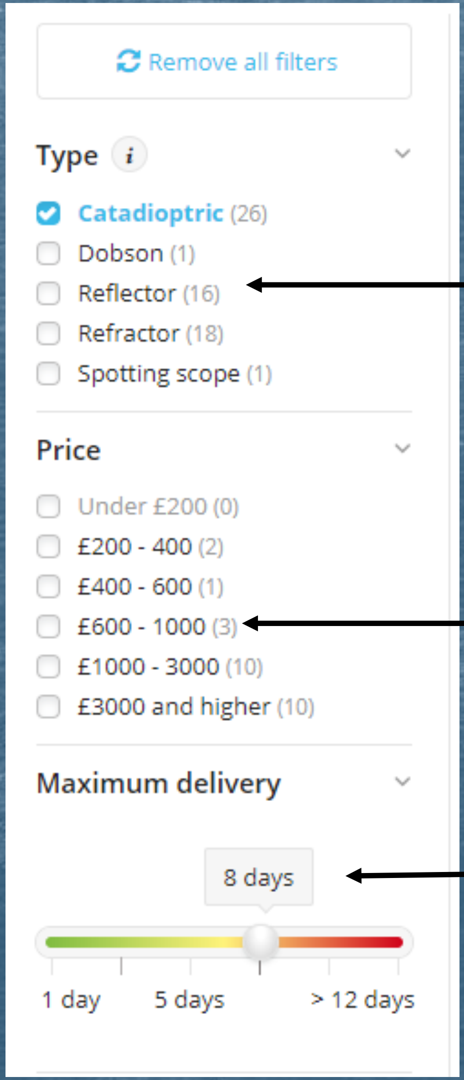
1 day · 5 days · > 12 days

Facet

Constraint

Facet count

Facet types



Remove all filters

Type

☑ Catadioptric (26)
□ Dobson (1)
□ Reflector (16)
□ Refractor (18)
□ Spotting scope (1)

Price

□ Under £200 (0)
□ £200 - 400 (2)
□ £400 - 600 (1)
□ £600 - 1000 (3)
□ £1000 - 3000 (10)
□ £3000 and higher (10)

Maximum delivery

8 days

1 day 5 days > 12 days

Values/Terms

Interval

Range

What for

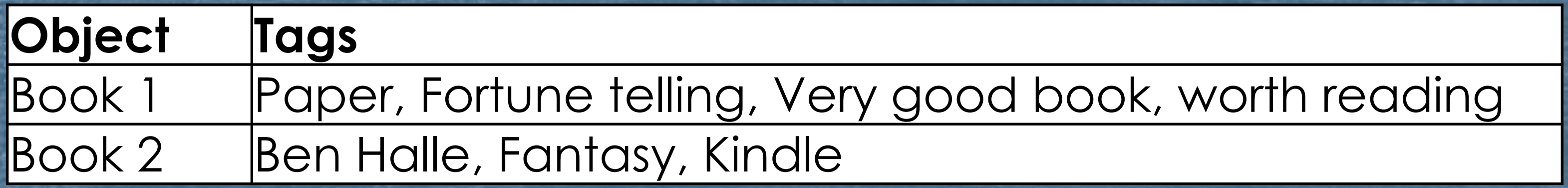
1. Filter by multiple taxonomies
2. Combine text, taxonomy terms and numeric values
3. Discover relationships or trends between objects
4. Make huge items volume navigable
5. Simplify "advanced" search UI

Tags

- Keyword assigned to an object
- Chosen informally and personally by item's creator or viewer
- Assigned by the viewer + unlimited = Folksonomy
- Assigned by the creator + limited = Taxonomy

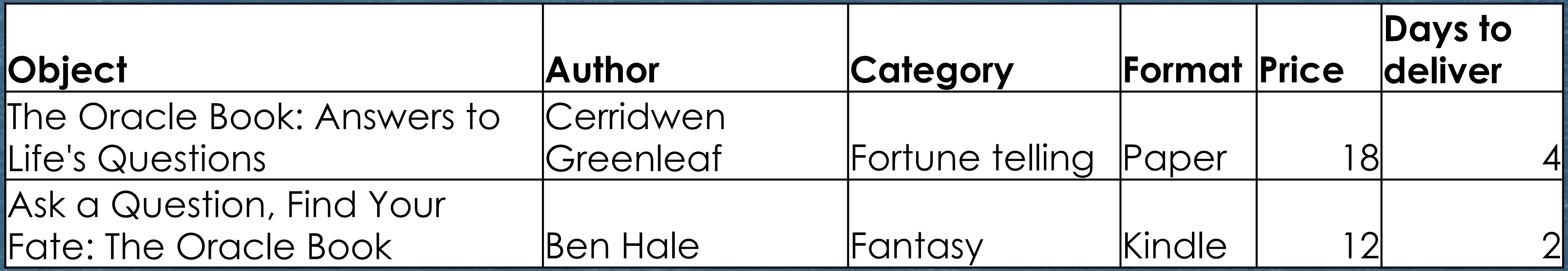
Faceted search base

- Tag-based



Object	Tags
Book 1	Paper, Fortune telling, Very good book, worth reading
Book 2	Ben Halle, Fantasy, Kindle

- Plain-structure based



Object	Author	Category	Format	Price	Days to deliver
The Oracle Book: Answers to Life's Questions	Cerridwen Greenleaf	Fortune telling	Paper	18	4
Ask a Question, Find Your Fate: The Oracle Book	Ben Hale	Fantasy	Kindle	12	2

Our case facets

1. Facet source: taxonomy + folksonomy
2. Facet types: terms mostly
3. Implementation type: tag-based

Our case statistics

Extracted entities: ***Objects, Tags, Tags of objects, Facet types***

Date: ***2016 to 2017***

Tagged objects: ***3 000 000***

Applied tags: ***42 000 000***

Unique tags count: ***100 000***

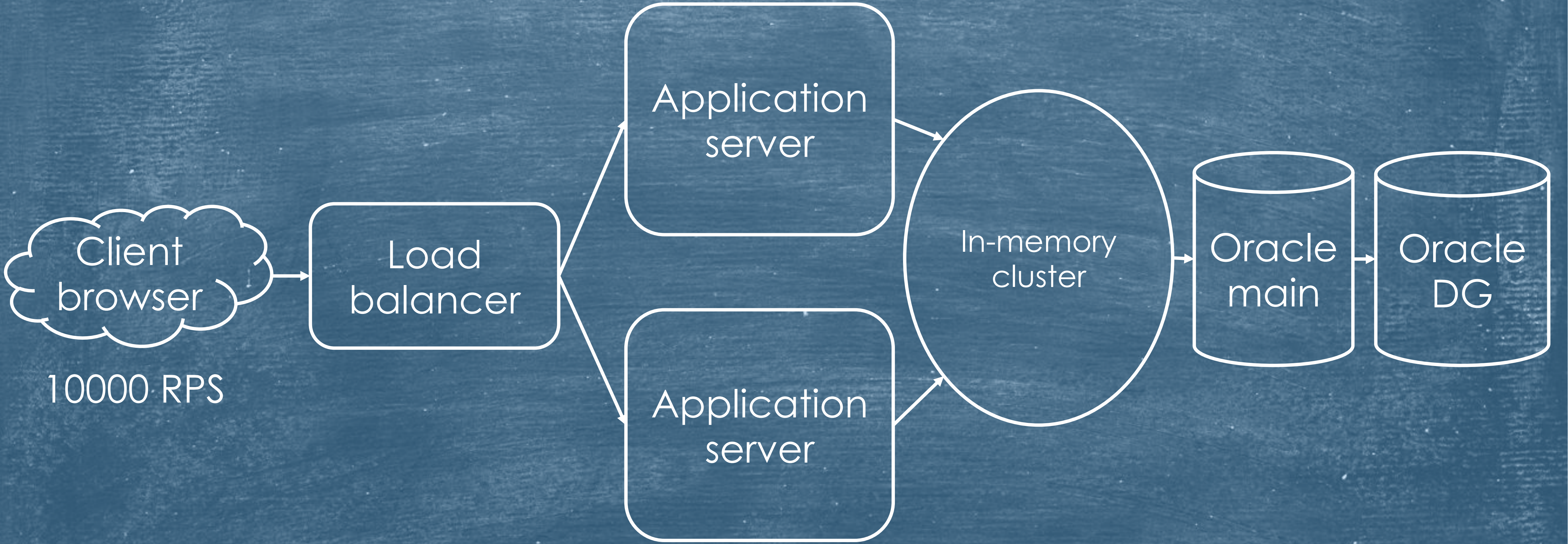
Max tags count for an object: ***15***

Max tag length: ***50***

Facets count: 150

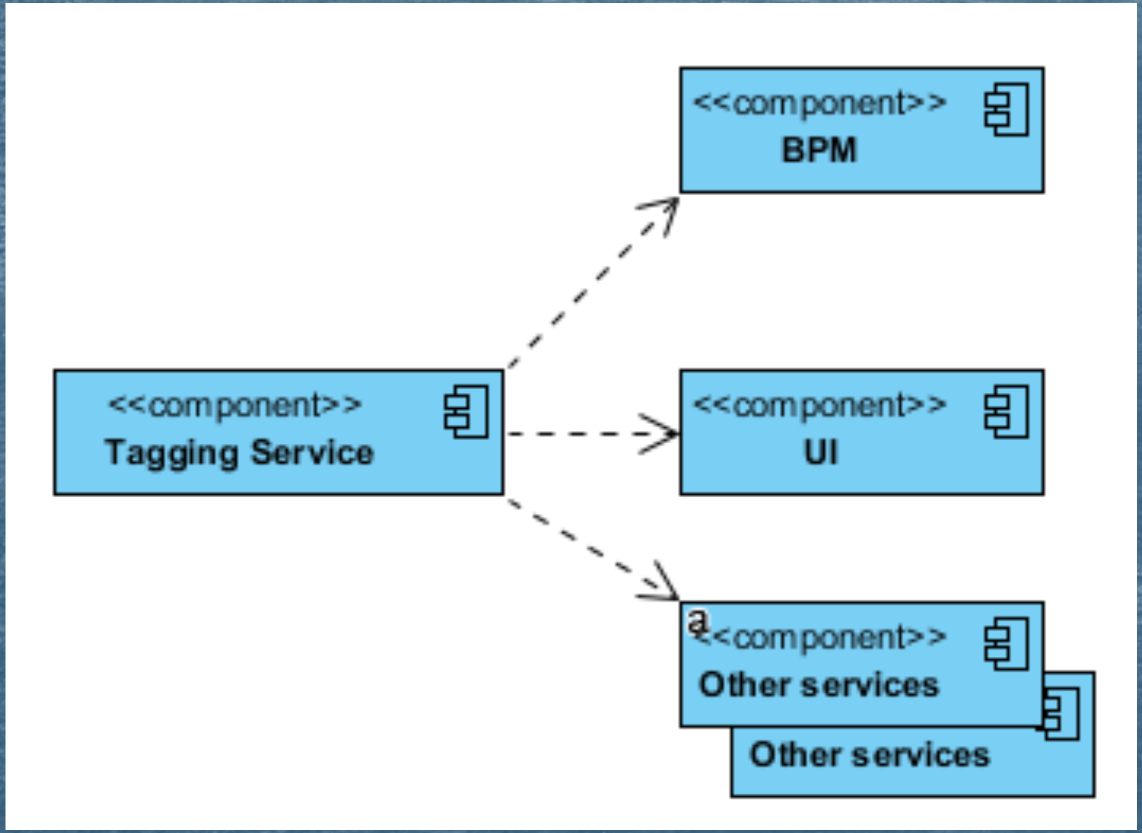
Data volume = StackOveflow x 3!

Architecture



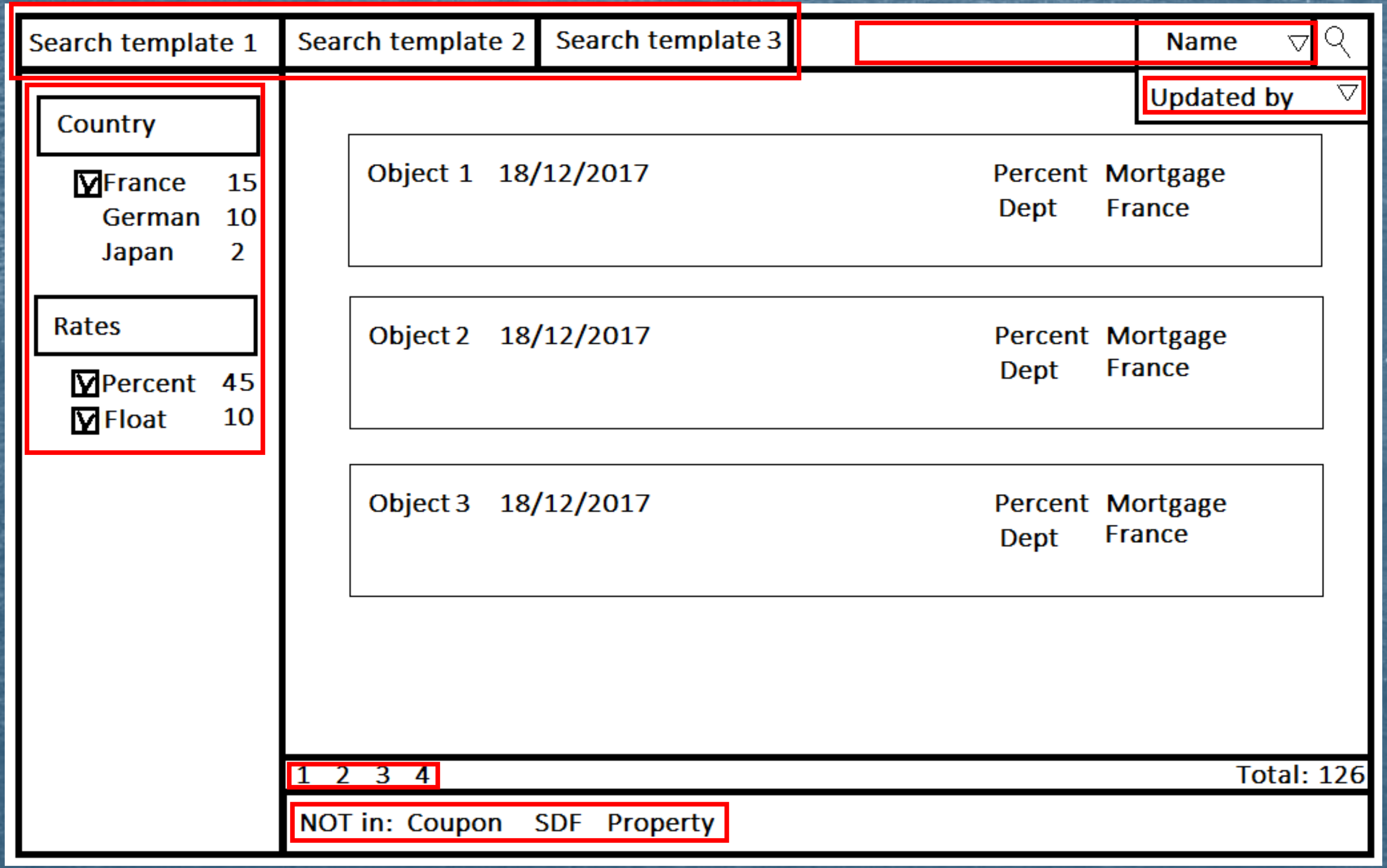
10000 RPS

Architecture



~20 fine-tuned SQL queries

UI



☑France 15
German 10
Japan 2

☑Percent 45
☑Float 10

Object 1 18/12/2017 Percent Mortgage
Dept France

Object 2 18/12/2017 Percent Mortgage
Dept France

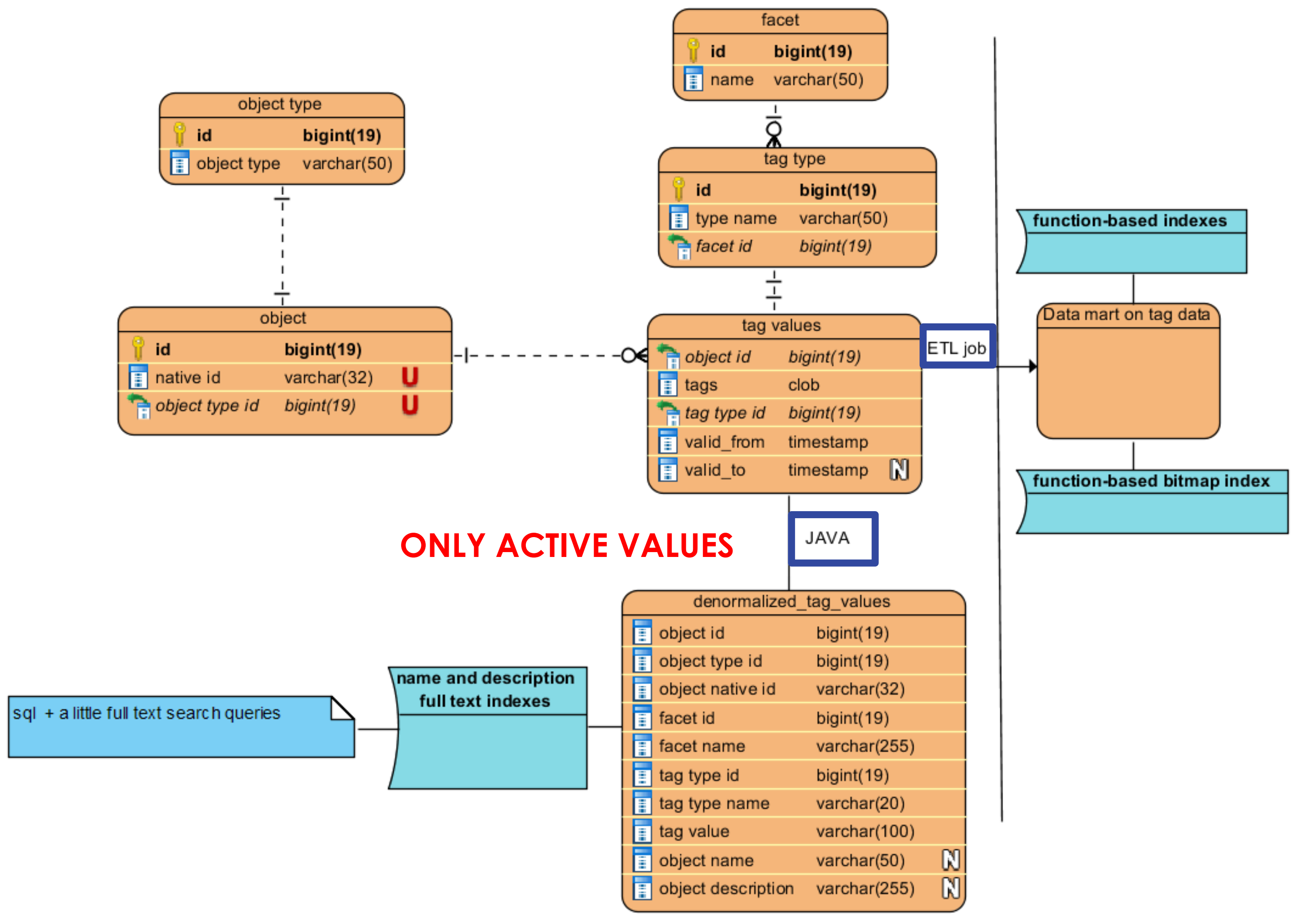
Object 3 18/12/2017 Percent Mortgage
Dept France

1 2 3 4 Total: 126

NOT in: Coupon SDF Property

Database structure

OLTP **DWH**



facet	
id	**bigint(19)**
name	varchar(50)

object type	
id	**bigint(19)**
object type	varchar(50)

tag type	
id	**bigint(19)**
type name	varchar(50)
facet id	*bigint(19)*

object		
id	**bigint(19)**	
native id	varchar(32)	U
object type id	*bigint(19)*	U

tag values		
object id	*bigint(19)*	
tags	clob	
tag type id	*bigint(19)*	
valid_from	timestamp	
valid_to	timestamp	N

denormalized_tag_values		
object id	bigint(19)	
object type id	bigint(19)	
object native id	varchar(32)	
facet id	bigint(19)	
facet name	varchar(255)	
tag type id	bigint(19)	
tag type name	varchar(20)	
tag value	varchar(100)	
object name	varchar(50)	N
object description	varchar(255)	N

ETL job

JAVA

ONLY ACTIVE VALUES

sql + a little full text search queries

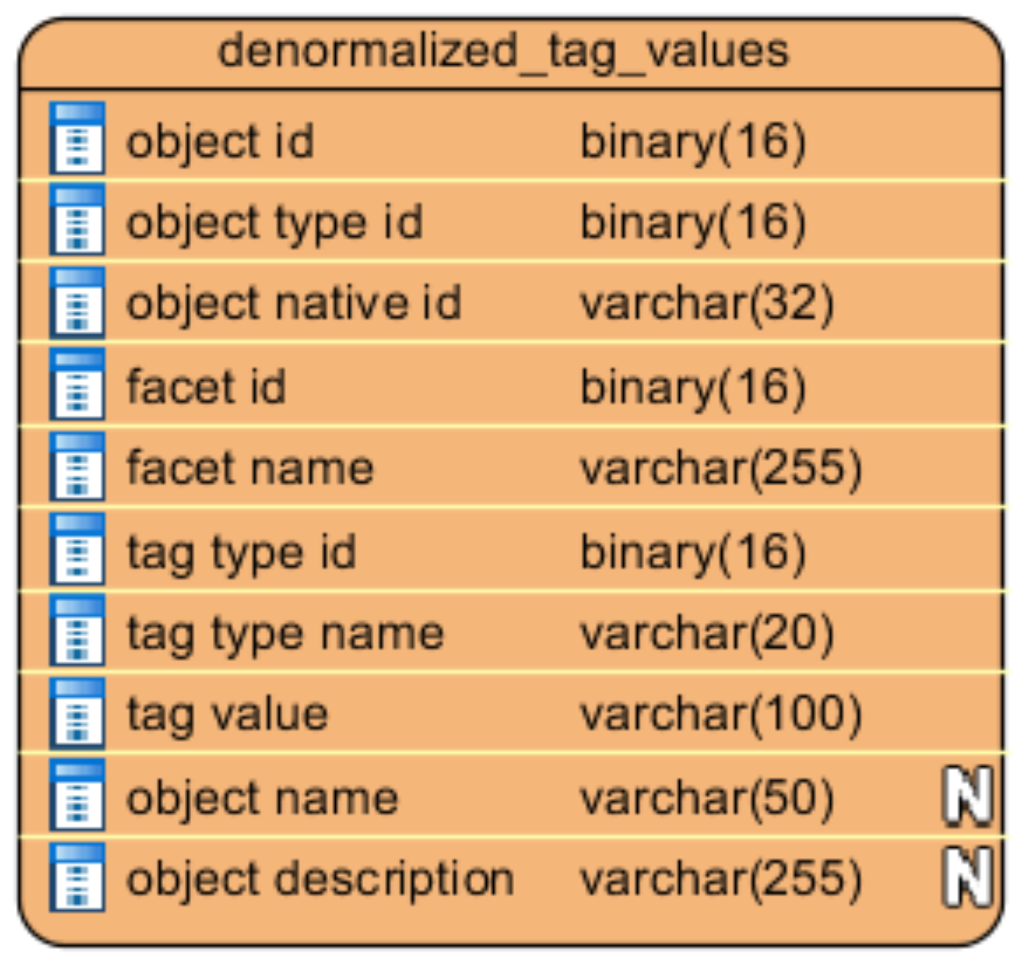
name and description full text indexes

function-based indexes

Data mart on tag data

function-based bitmap index

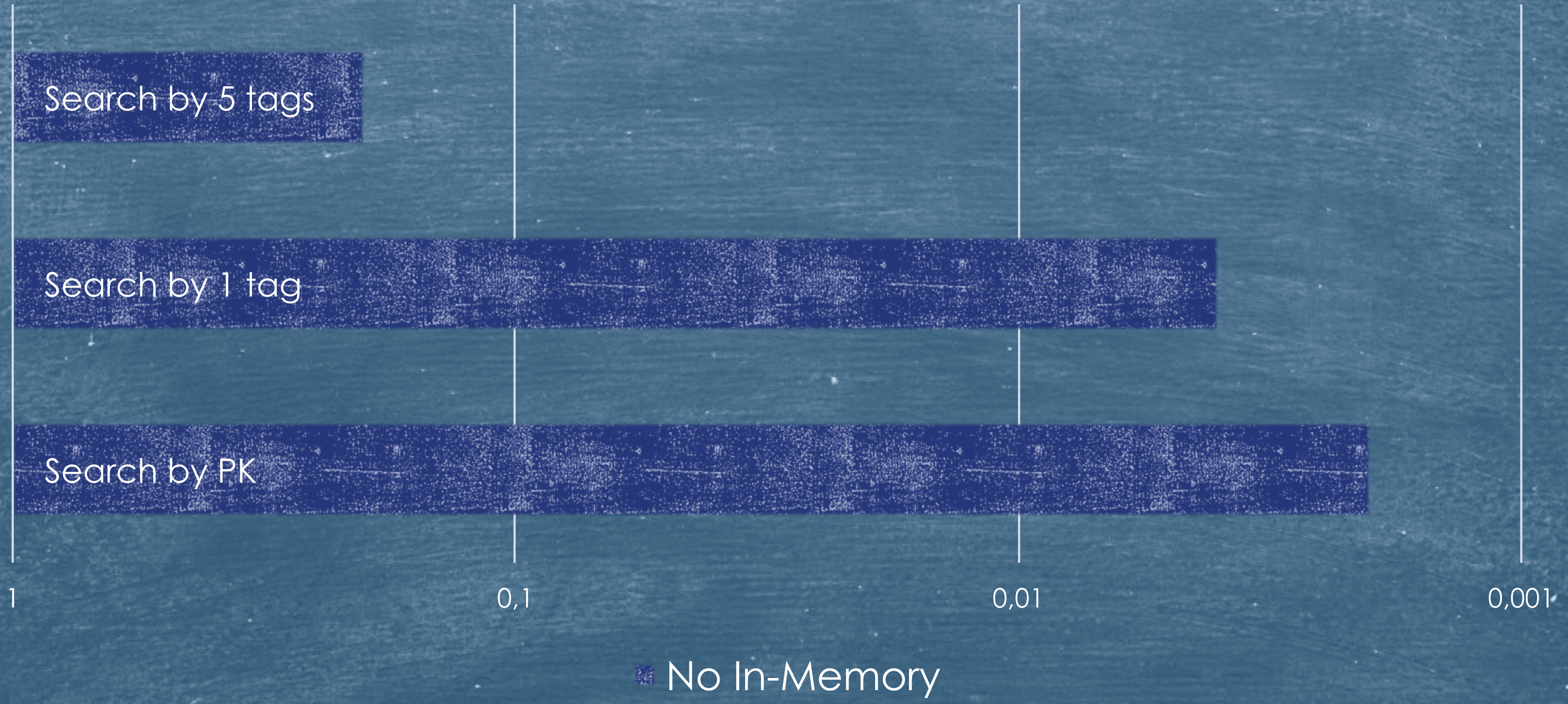
Search table structure



denormalized_tag_values

object id	binary(16)	
object type id	binary(16)	
object native id	varchar(32)	
facet id	binary(16)	
facet name	varchar(255)	
tag type id	binary(16)	
tag type name	varchar(20)	
tag value	varchar(100)	
object name	varchar(50)	N
object description	varchar(255)	N

Performance



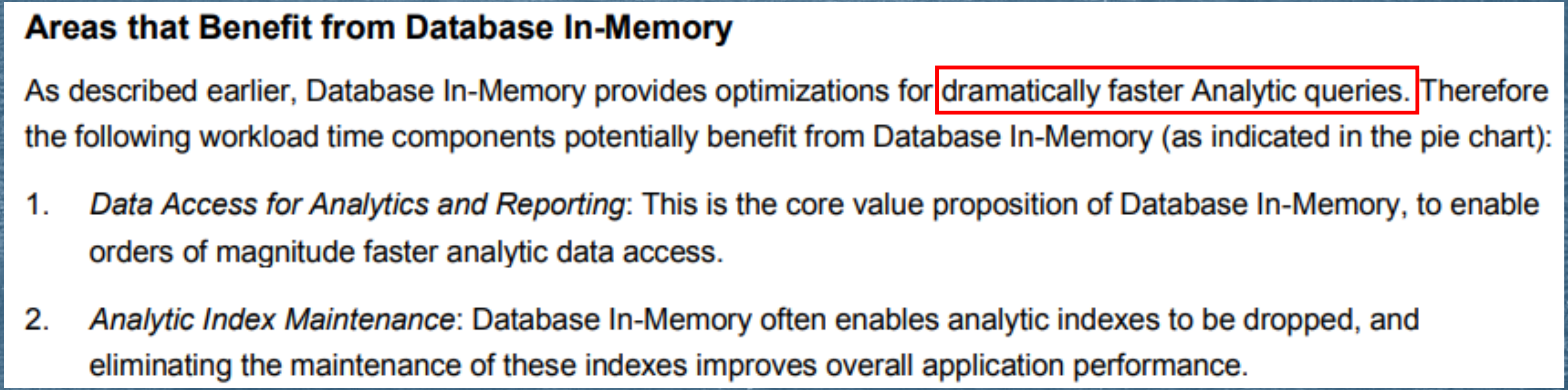
Solution design

Full Text Search server

Act III

To FTS, or not to FTS, that is the question

Implementation



Areas that Benefit from Database In-Memory

As described earlier, Database In-Memory provides optimizations for dramatically faster Analytic queries. Therefore the following workload time components potentially benefit from Database In-Memory (as indicated in the pie chart):

1. *Data Access for Analytics and Reporting*: This is the core value proposition of Database In-Memory, to enable orders of magnitude faster analytic data access.

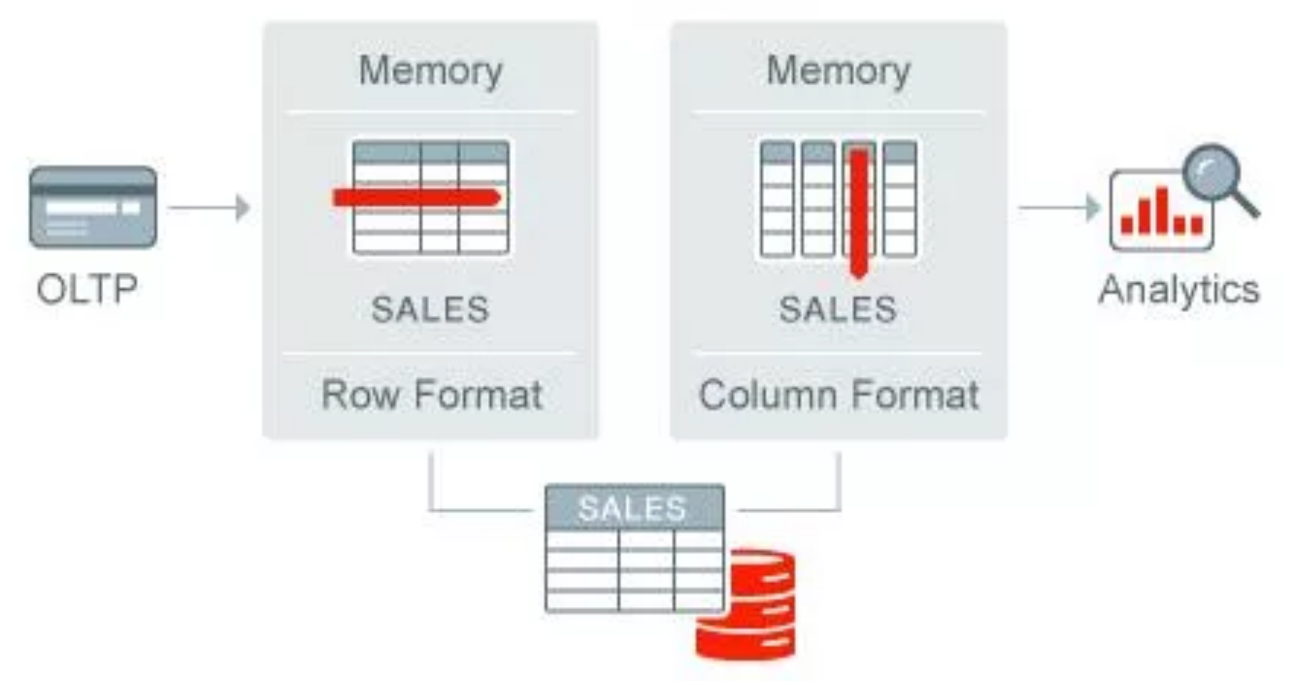
2. *Analytic Index Maintenance*: Database In-Memory often enables analytic indexes to be dropped, and eliminating the maintenance of these indexes improves overall application performance.

It isn't our case completely!

Why to try

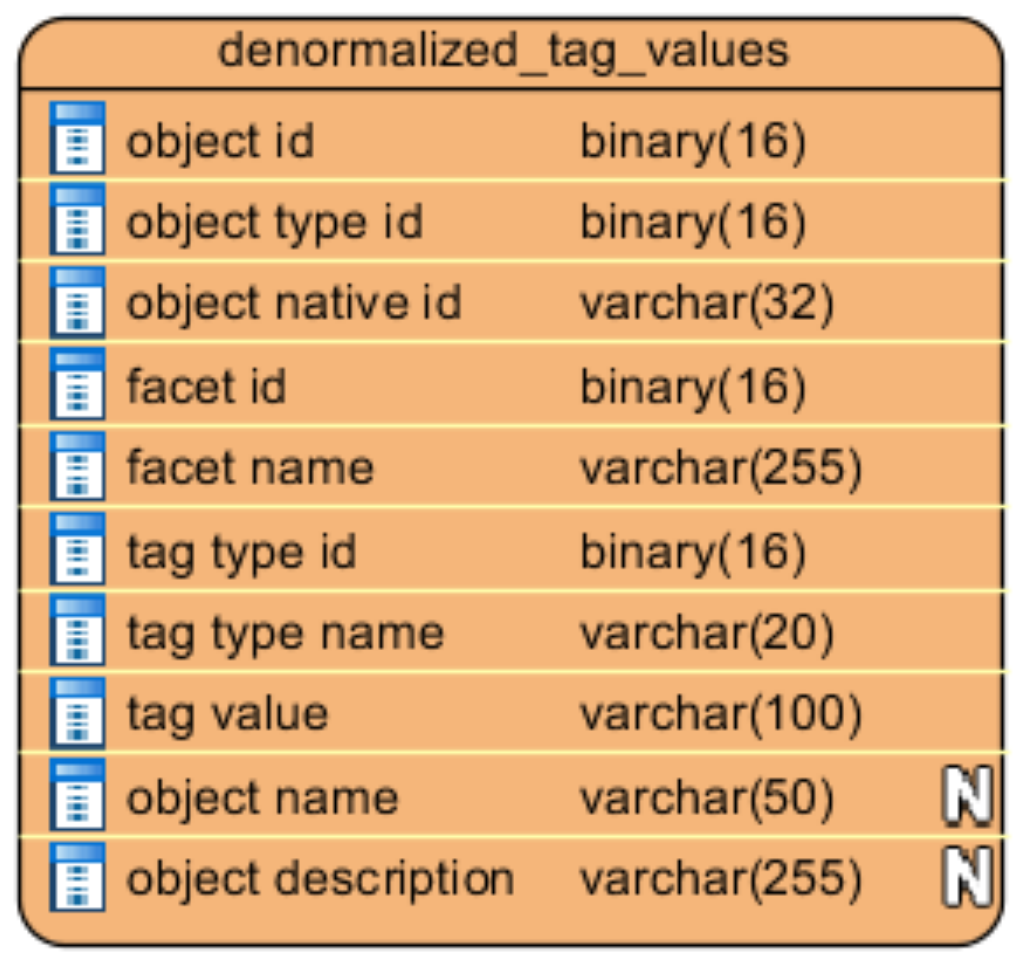
1. Limited POC resources: people + time
2. Customer has a license
3. Wide search table
4. A lot of rows
5. A lot of equal values:
   - Object types
   - Facet types
   - Tag names
6. Size is fine for InMemory
7. Queries use a lot of aggregate functions

Internals



Dual storage format!

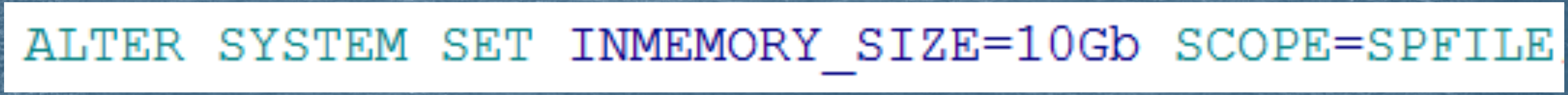
Search table structure



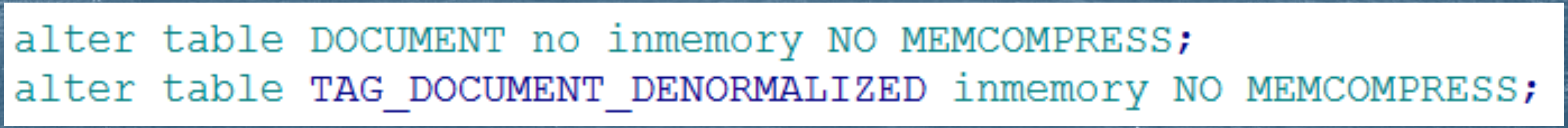
denormalized_tag_values

object id	binary(16)	
object type id	binary(16)	
object native id	varchar(32)	
facet id	binary(16)	
facet name	varchar(255)	
tag type id	binary(16)	
tag type name	varchar(20)	
tag value	varchar(100)	
object name	varchar(50)	N
object description	varchar(255)	N

Naive implementation

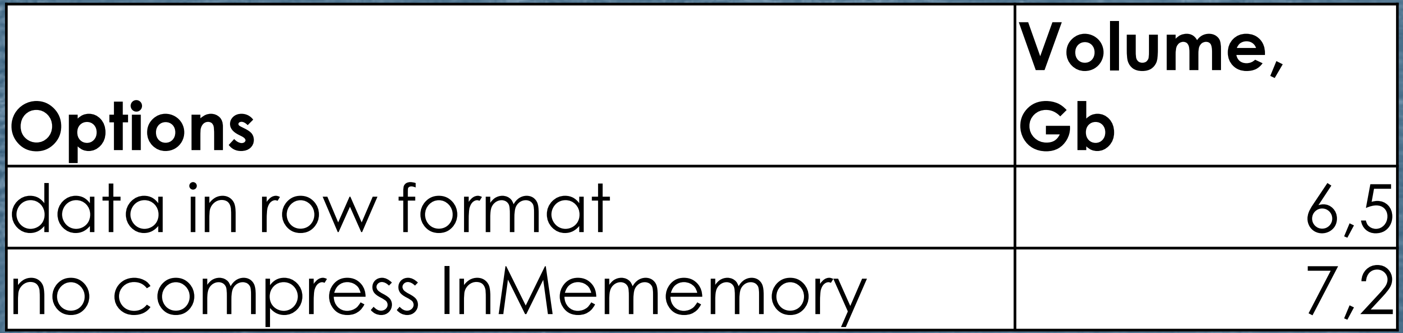


```
ALTER SYSTEM SET INMEMORY_SIZE=10Gb SCOPE=SPFILE
```



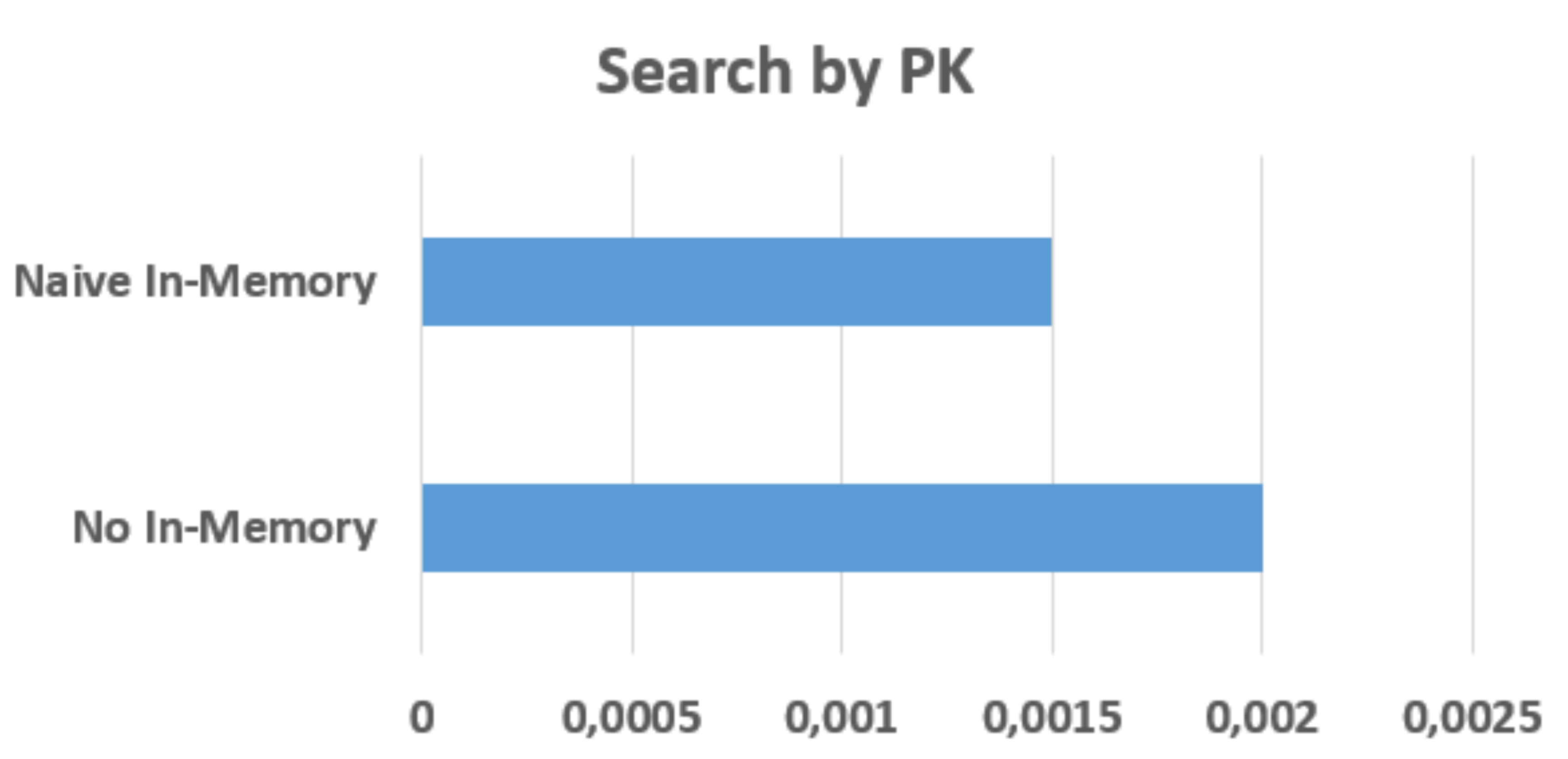
```
alter table DOCUMENT no inmemory NO MEMCOMPRESS;
alter table TAG_DOCUMENT_DENORMALIZED inmemory NO MEMCOMPRESS;
```

InMemory size

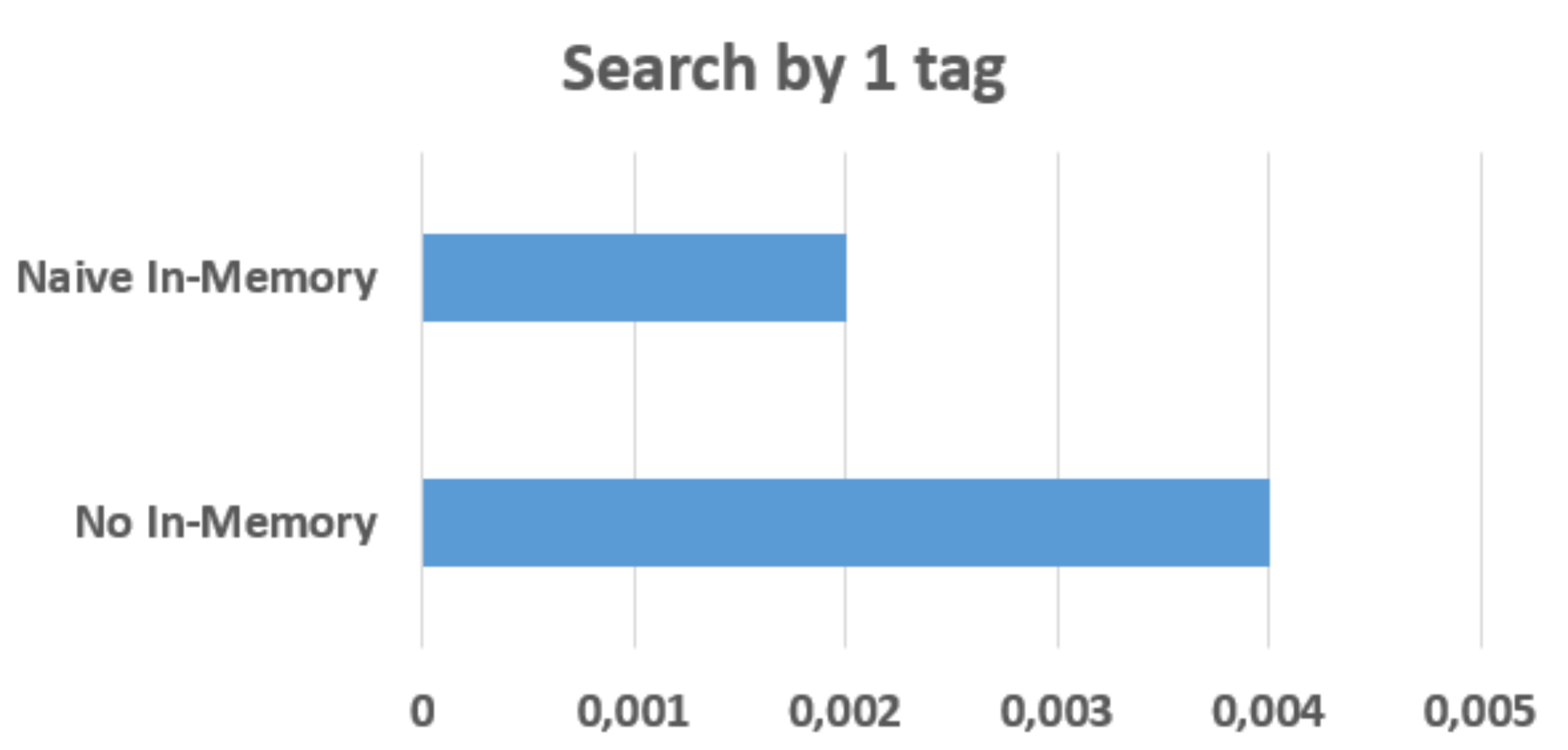


Options	Volume, Gb
data in row format	6,5
no compress InMememory	7,2

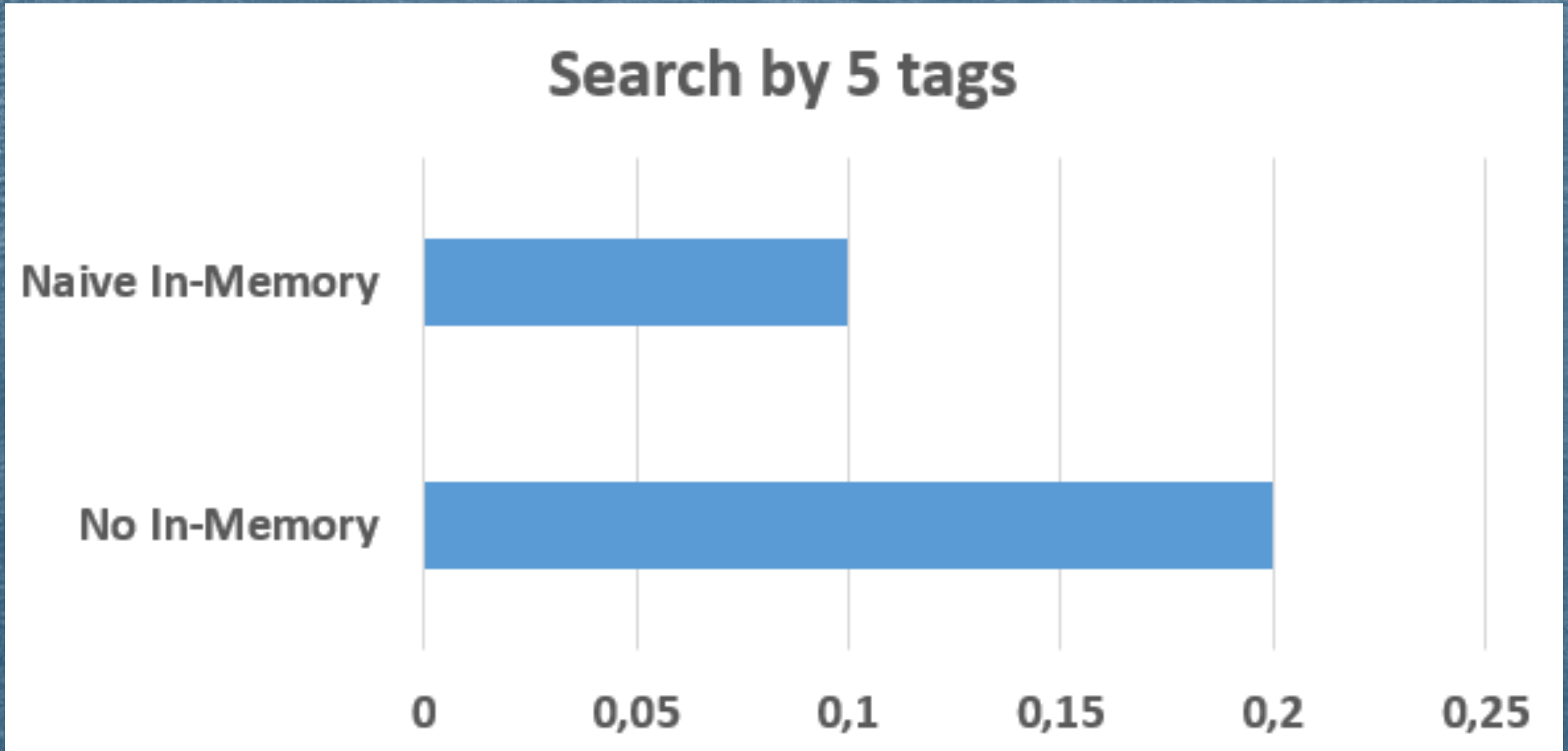
Performance



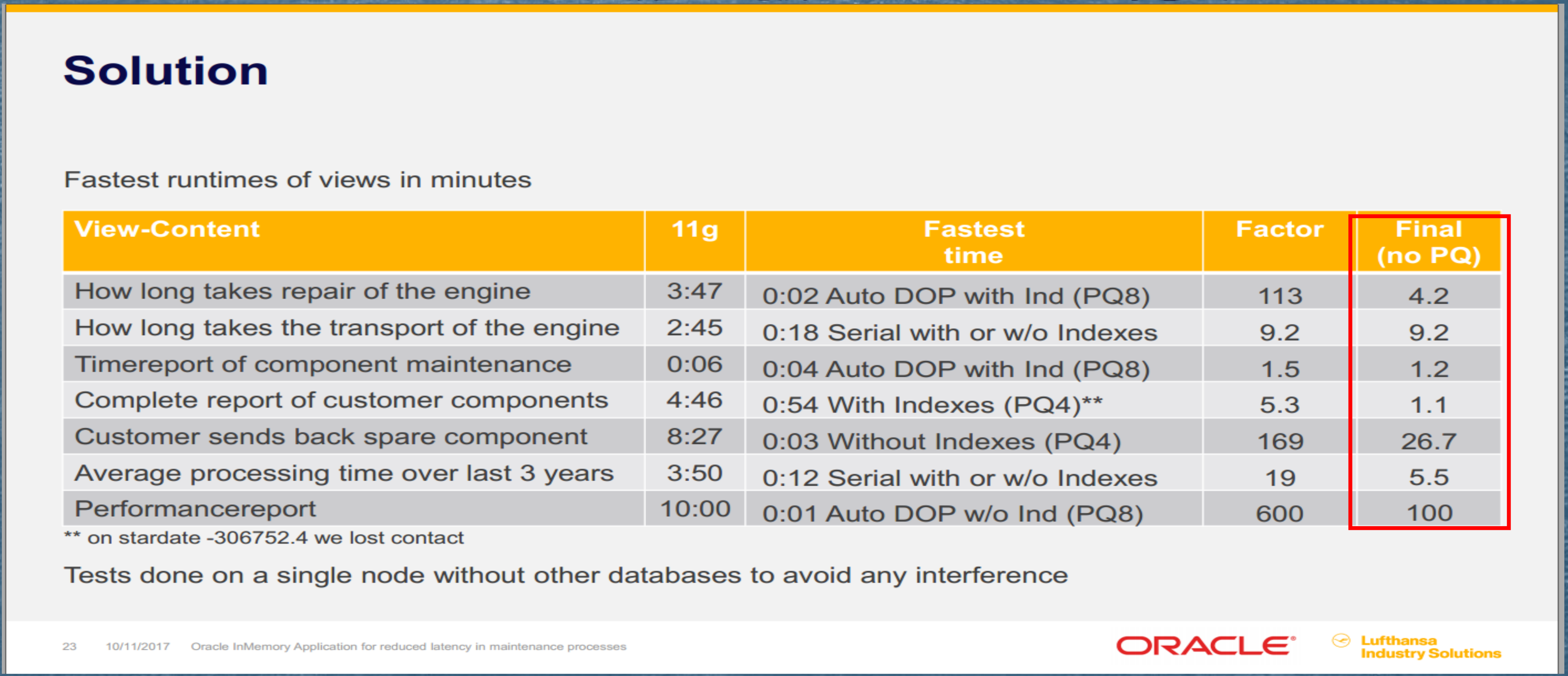
Performance



Performance



Performance profit



Solution

Fastest runtimes of views in minutes

View-Content	11g	Fastest time	Factor	Final (no PQ)
How long takes repair of the engine	3:47	0:02 Auto DOP with Ind (PQ8)	113	4.2
How long takes the transport of the engine	2:45	0:18 Serial with or w/o Indexes	9.2	9.2
Timereport of component maintenance	0:06	0:04 Auto DOP with Ind (PQ8)	1.5	1.2
Complete report of customer components	4:46	0:54 With Indexes (PQ4)**	5.3	1.1
Customer sends back spare component	8:27	0:03 Without Indexes (PQ4)	169	26.7
Average processing time over last 3 years	3:50	0:12 Serial with or w/o Indexes	19	5.5
Performancereport	10:00	0:01 Auto DOP w/o Ind (PQ8)	600	100

** on stardate -306752.4 we lost contact

Tests done on a single node without other databases to avoid any interference

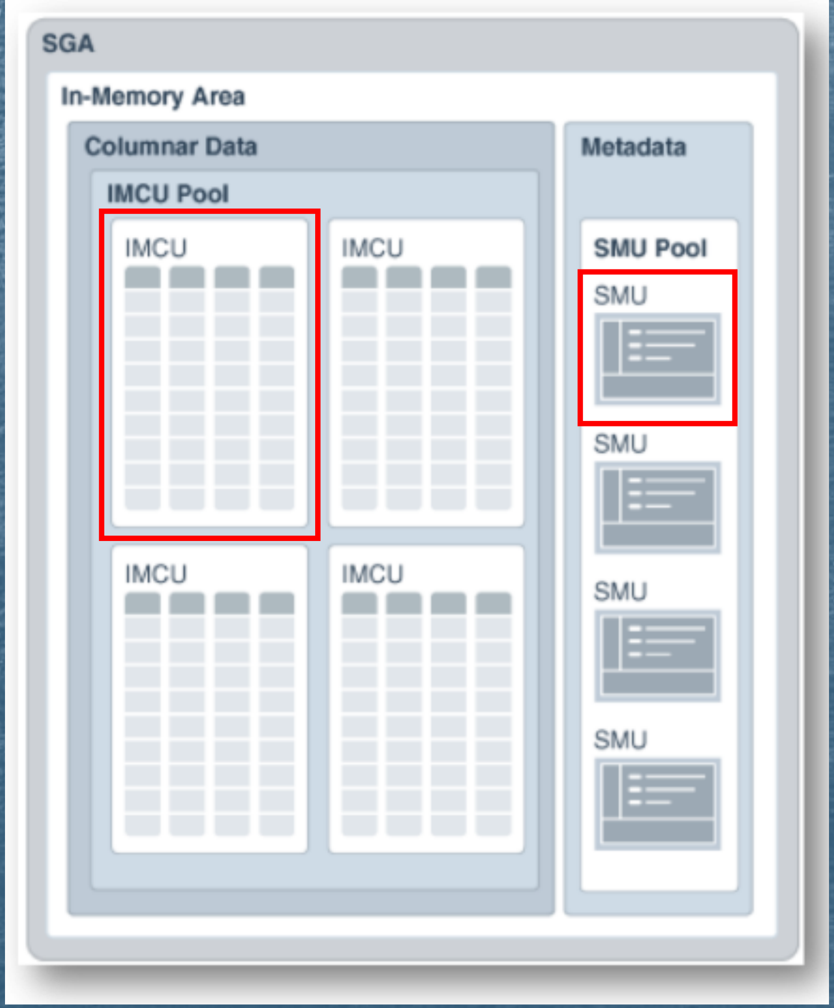
23 10/11/2017 Oracle InMemory Application for reduced latency in maintenance processes

ORACLE®

Lufthansa Industry Solutions

2x <> 21x!
Where is our performance?!

InMemory internals

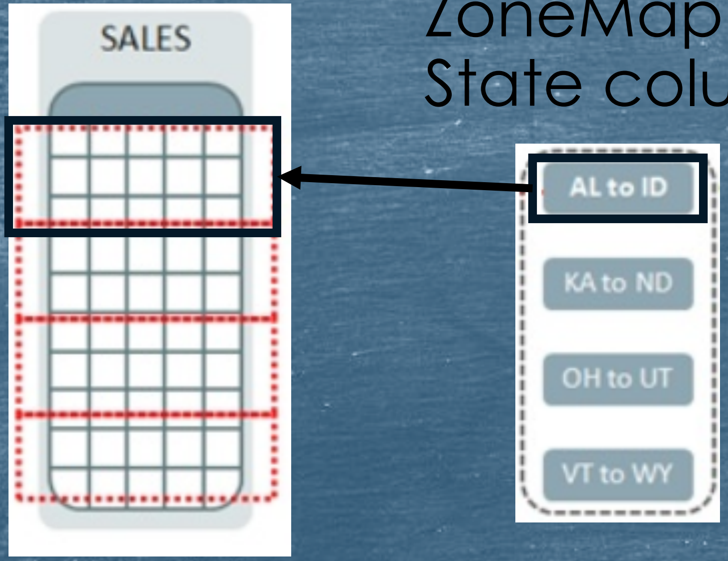


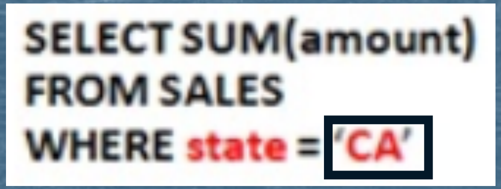
1. IMCU – InMemory Compression Unit
   Size = 1 Mb
   Columnar Data

2. SMU – Snapshot Metadata Unit
   Size = 64 Kb
   Zone Map Based Index

Zone Maps

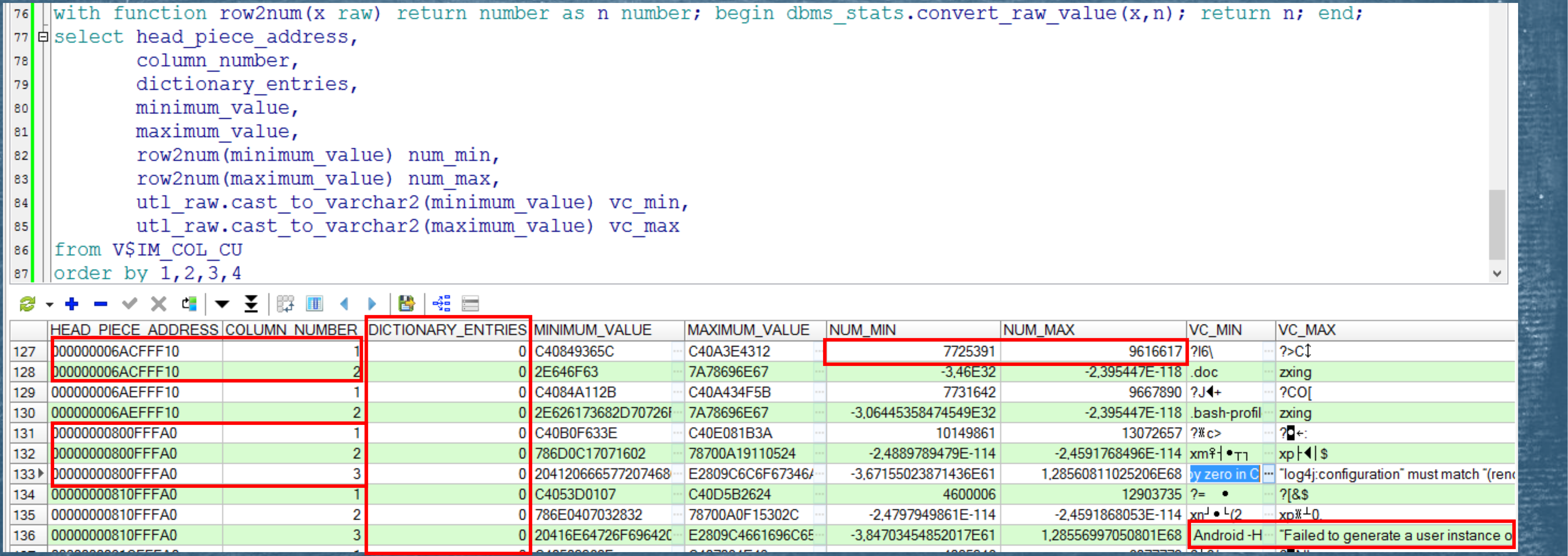
ZoneMap on State column





```
SELECT SUM(amount)
FROM SALES
WHERE state = 'CA'
```

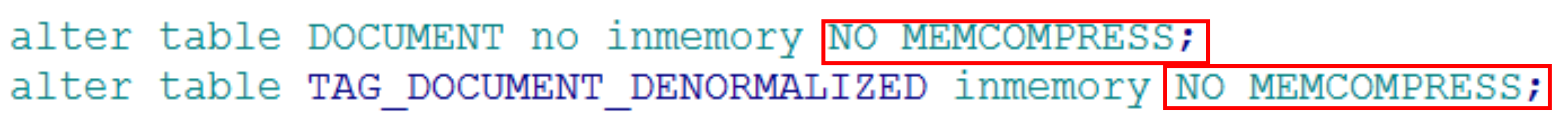
InMemory Zone Maps



```
with function row2num(x raw) return number as n number; begin dbms_stats.convert_raw_value(x,n); return n; end;
select head_piece_address,
       column_number,
       dictionary_entries,
       minimum_value,
       maximum_value,
       row2num(minimum_value) num_min,
       row2num(maximum_value) num_max,
       utl_raw.cast_to_varchar2(minimum_value) vc_min,
       utl_raw.cast_to_varchar2(maximum_value) vc_max
from V$IM_COL_CU
order by 1,2,3,4
```

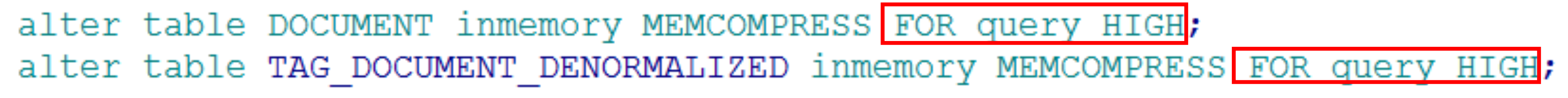
	HEAD_PIECE_ADDRESS	COLUMN_NUMBER	DICTIONARY_ENTRIES	MINIMUM_VALUE	MAXIMUM_VALUE	NUM_MIN	NUM_MAX	VC_MIN	VC_MAX
127	000000006ACFFF10	1	0	C40849365C	C40A3E4312	7725391	9616617	?I6\	?>C↕
128	000000006ACFFF10	2	0	2E646F63	7A78696E67	-3,46E32	-2,395447E-118	.doc	zxing
129	000000006AEFFF10	1	0	C4084A112B	C40A434F5B	7731642	9667890	?J◀+	?CO[
130	000000006AEFFF10	2	0	2E626173682D70726	7A78696E67	-3,06445358474549E32	-2,395447E-118	.bash-profil	zxing
131	00000000800FFFA0	1	0	C40B0F633E	C40E081B3A	10149861	13072657	?※c>	?◘←:
132	00000000800FFFA0	2	0	786D0C17071602	78700A19110524	-2,4889789479E-114	-2,4591768496E-114	xm♀┤•┬┐	xp├◀│$
133	00000000800FFFA0	3	0	204120666577207468	E2809C6C6F67346	-3,67155023871436E61	1,28560811025206E68	y zero in C	"log4j:configuration" must match "(ren
134	00000000810FFFA0	1	0	C4053D0107	C40D5B2624	4600006	12903735	?= •	?[&$
135	00000000810FFFA0	2	0	786E0407032832	78700A0F15302C	-2,4797949861E-114	-2,4591868053E-114	xn┘•└(2	xp※┴0
136	00000000810FFFA0	3	0	20416E64726F69642	E2809C4661696C65	-3,84703454852017E61	1,28556997050801E68	Android -H	"Failed to generate a user instance o

Implementation



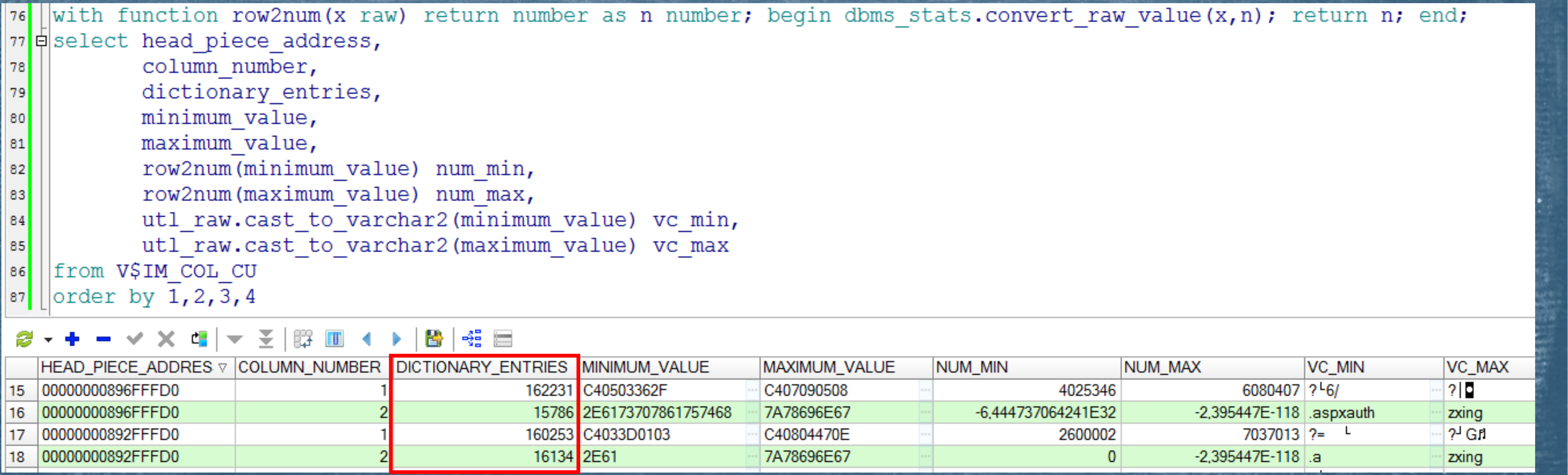
```
alter table DOCUMENT no inmemory NO MEMCOMPRESS;
alter table TAG_DOCUMENT_DENORMALIZED inmemory NO MEMCOMPRESS;
```

Implementation



```
alter table DOCUMENT inmemory MEMCOMPRESS FOR query HIGH;
alter table TAG_DOCUMENT_DENORMALIZED inmemory MEMCOMPRESS FOR query HIGH;
```

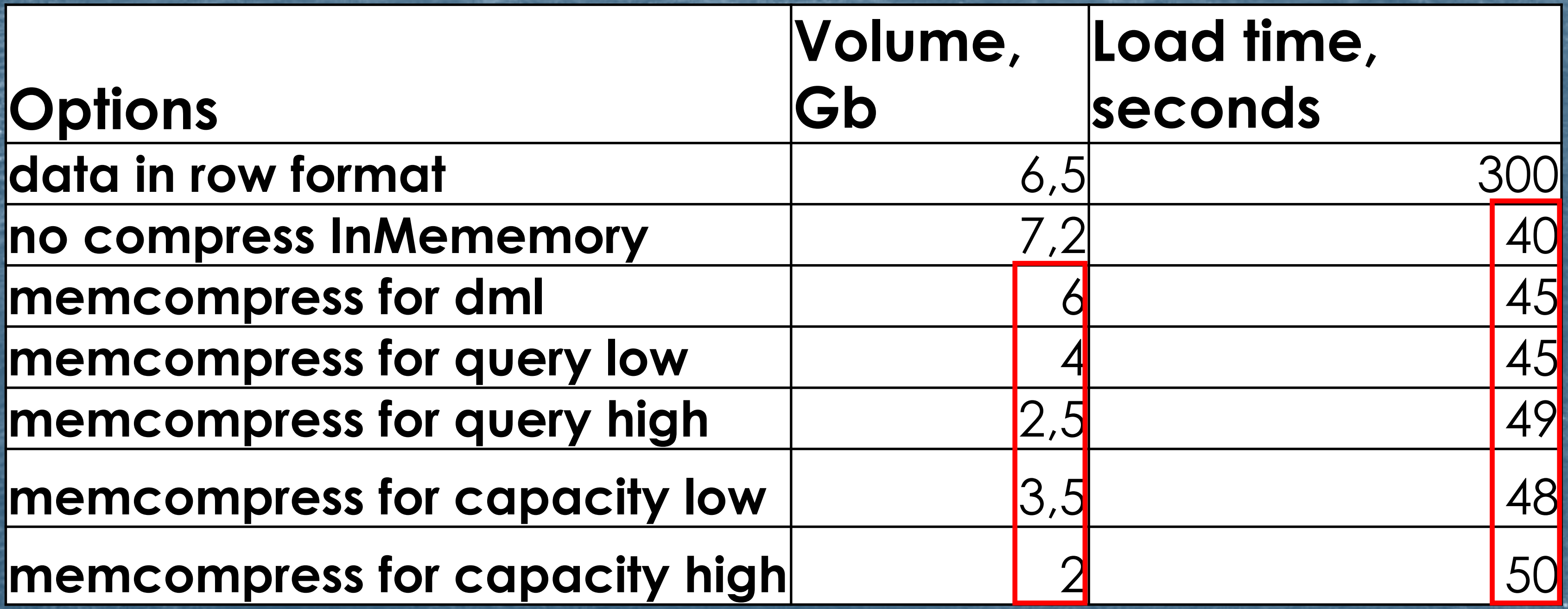
New InMemory Zone Maps



```
with function row2num(x raw) return number as n number; begin dbms_stats.convert_raw_value(x,n); return n; end;
select head_piece_address,
       column_number,
       dictionary_entries,
       minimum_value,
       maximum_value,
       row2num(minimum_value) num_min,
       row2num(maximum_value) num_max,
       utl_raw.cast_to_varchar2(minimum_value) vc_min,
       utl_raw.cast_to_varchar2(maximum_value) vc_max
from V$IM_COL_CU
order by 1,2,3,4
```

	HEAD_PIECE_ADDRES	COLUMN_NUMBER	DICTIONARY_ENTRIES	MINIMUM_VALUE	MAXIMUM_VALUE	NUM_MIN	NUM_MAX	VC_MIN	VC_MAX
15	00000000896FFFD0	1	162231	C40503362F	C407090508	4025346	6080407	?└6/	?│◘
16	00000000896FFFD0	2	15786	2E6173707861757468	7A78696E67	-6,444737064241E32	-2,395447E-118	.aspxauth	zxing
17	00000000892FFFD0	1	160253	C4033D0103	C40804470E	2600002	7037013	?= └	?┘G♬
18	00000000892FFFD0	2	16134	2E61	7A78696E67	0	-2,395447E-118	.a	zxing

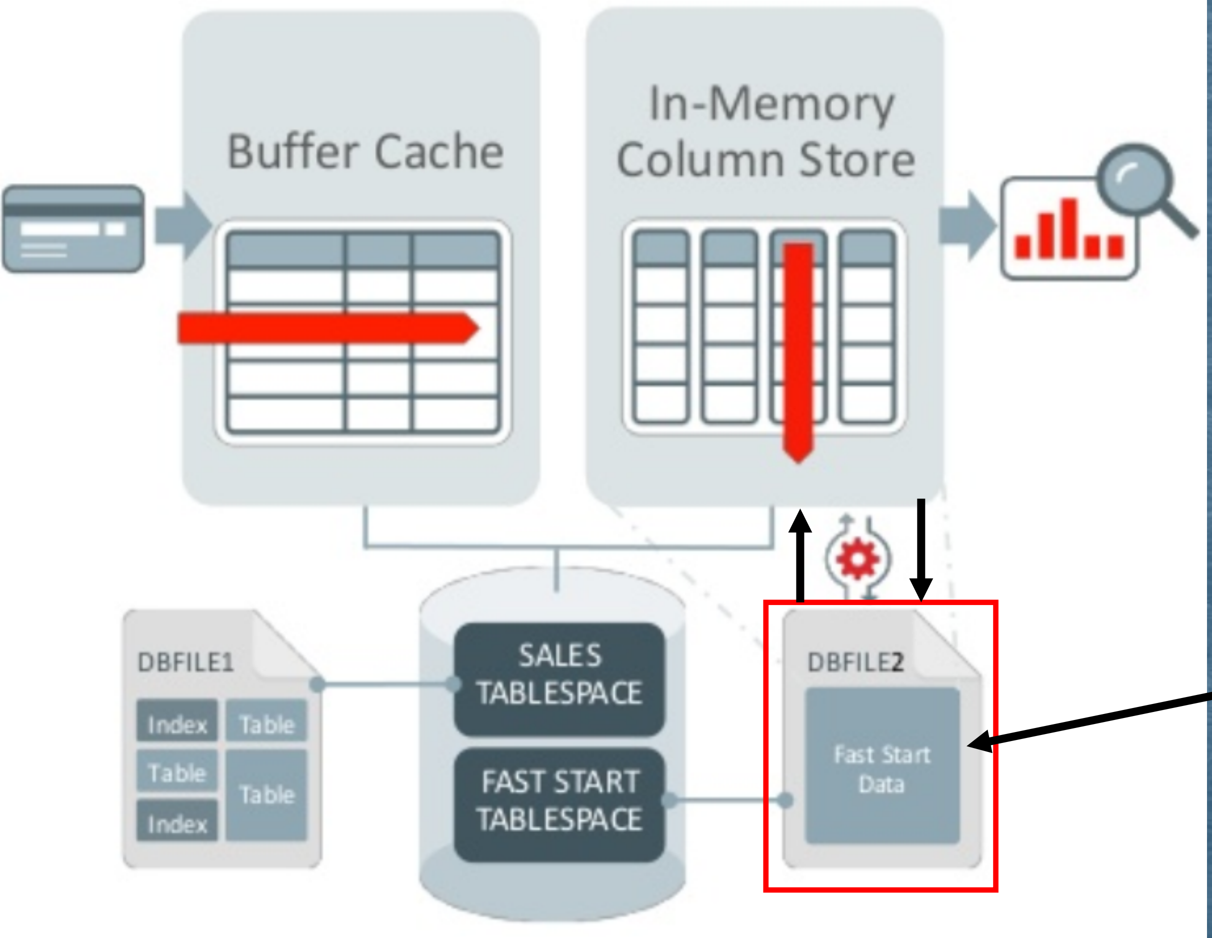
InMemory size



Options	Volume, Gb	Load time, seconds
data in row format	6,5	300
no compress InMemememory	7,2	40
memcompress for dml	6	45
memcompress for query low	4	45
memcompress for query high	2,5	49
memcompress for capacity low	3,5	48
memcompress for capacity high	2	50

No significant differences for loading!

FastStart



Columnar format

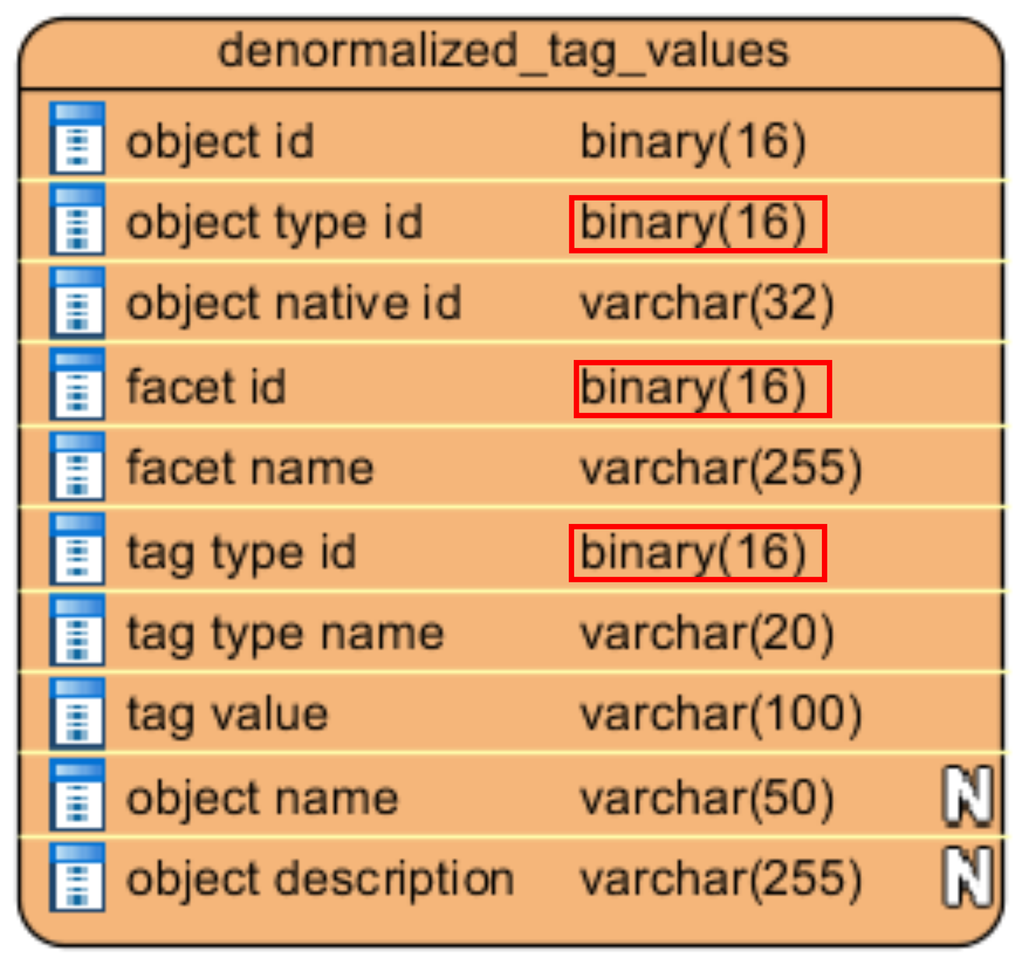
3x loading speed boost!

Test rerun

Same metrics!

Where is our performance?!

Zone Maps nuances

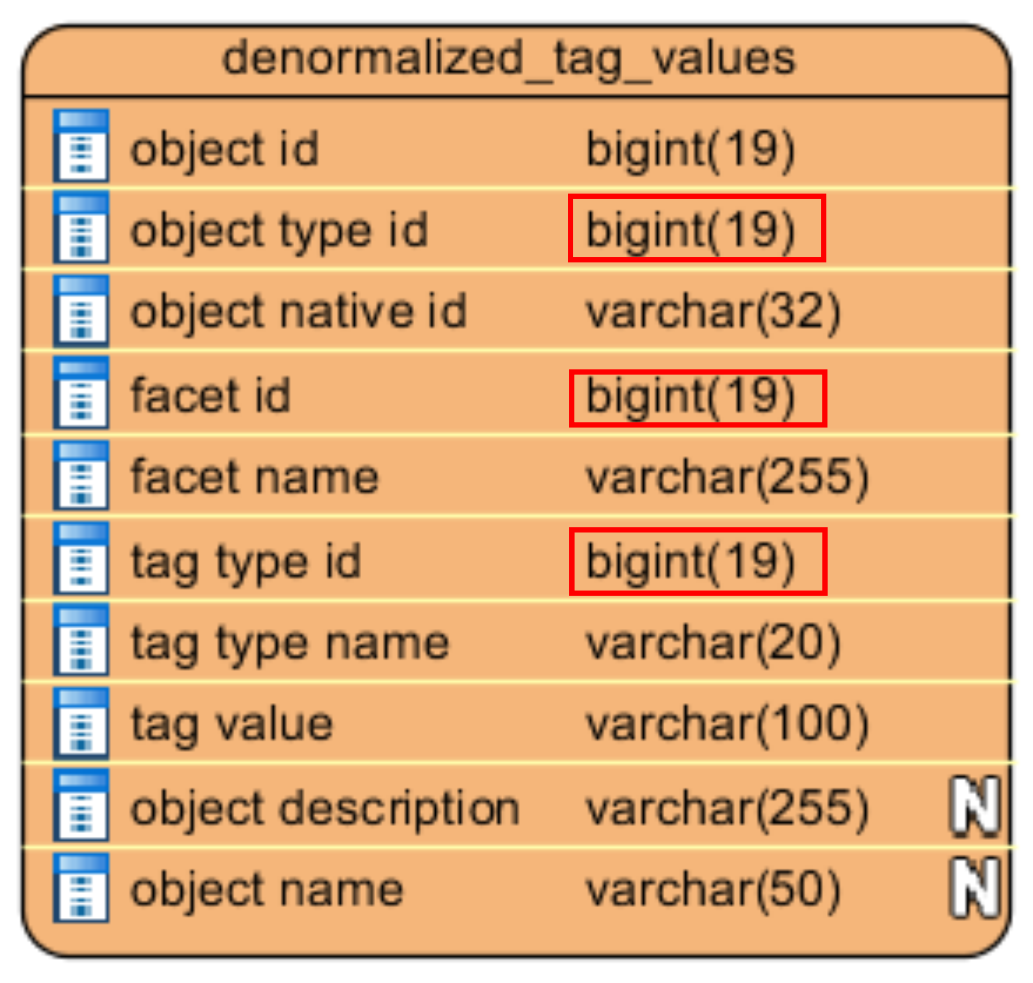


denormalized_tag_values

object id	binary(16)	
object type id	binary(16)	
object native id	varchar(32)	
facet id	binary(16)	
facet name	varchar(255)	
tag type id	binary(16)	
tag type name	varchar(20)	
tag value	varchar(100)	
object name	varchar(50)	N
object description	varchar(255)	N

Zone Maps not efficient!

Table structure

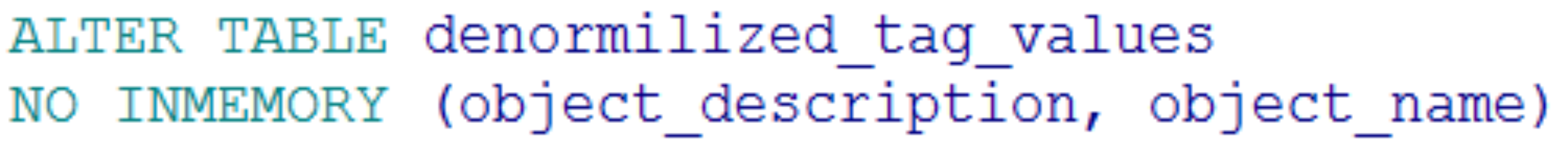


denormalized_tag_values

object id	bigint(19)	
object type id	bigint(19)	
object native id	varchar(32)	
facet id	bigint(19)	
facet name	varchar(255)	
tag type id	bigint(19)	
tag type name	varchar(20)	
tag value	varchar(100)	
object description	varchar(255)	N
object name	varchar(50)	N

+ all indexes are dropped

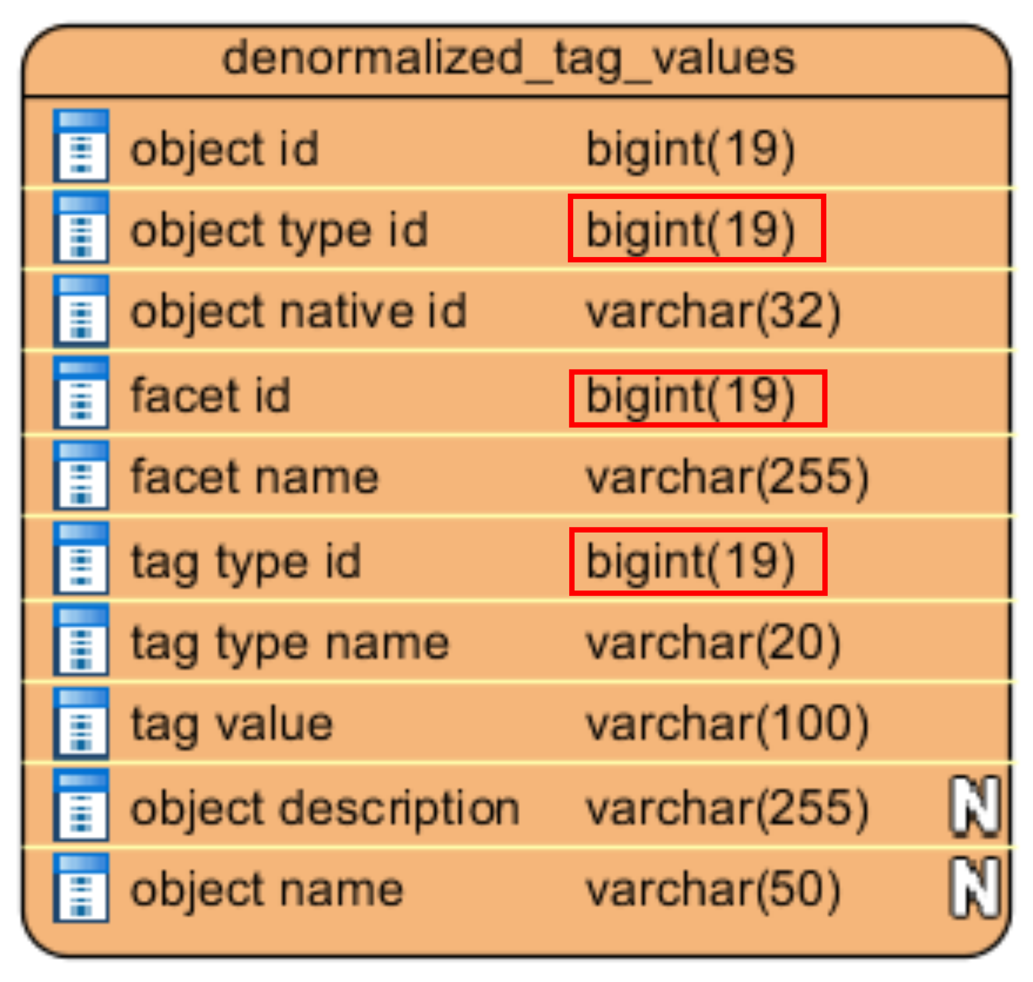
Final table structure



```
ALTER TABLE denormilized_tag_values
NO INMEMORY (object_description, object_name)
```

Searchable via Full Text Search indexes!

Final table structure



denormalized_tag_values

object id	bigint(19)	
object type id	bigint(19)	
object native id	varchar(32)	
facet id	bigint(19)	
facet name	varchar(255)	
tag type id	bigint(19)	
tag type name	varchar(20)	
tag value	varchar(100)	
object description	varchar(255)	N
object name	varchar(50)	N

+ all indexes are dropped

No InMemory

Performance spikes

- Sporadic search degradation – 5-10%
- Happens when a lot of records inserted

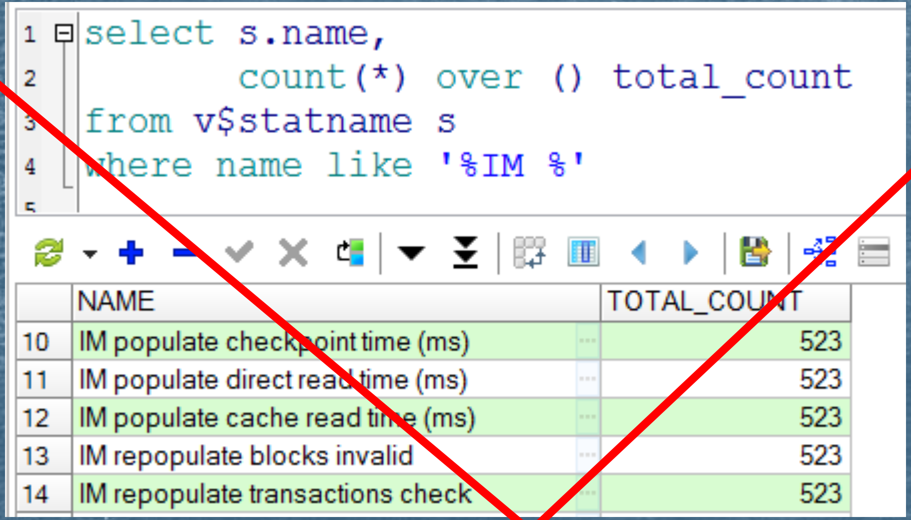
Transaction processing

1. Changed records -> Mark as stale
2. Stale records -> Read from row storage
   - buffer cache
   - disk
3. Stale records -> repopulate IMCU:
   - Staleness threshold
   - Trickle repopulation process - each 2 minutes
   - processes count - ***INMEMORY_MAX_REPOPULATE_SERVERS***
   - processes utilization - ***INMEMORY_TRICKLE_REPOPULATE_SERVERS_PERCENT***

Performance spikes elimination

1. *INMEMORY_MAX_REPOPULATE_SERVERS = 4*
2. *INMEMORY_TRICKLE_REPOPULATE_SERVERS_PERCENT = 8*

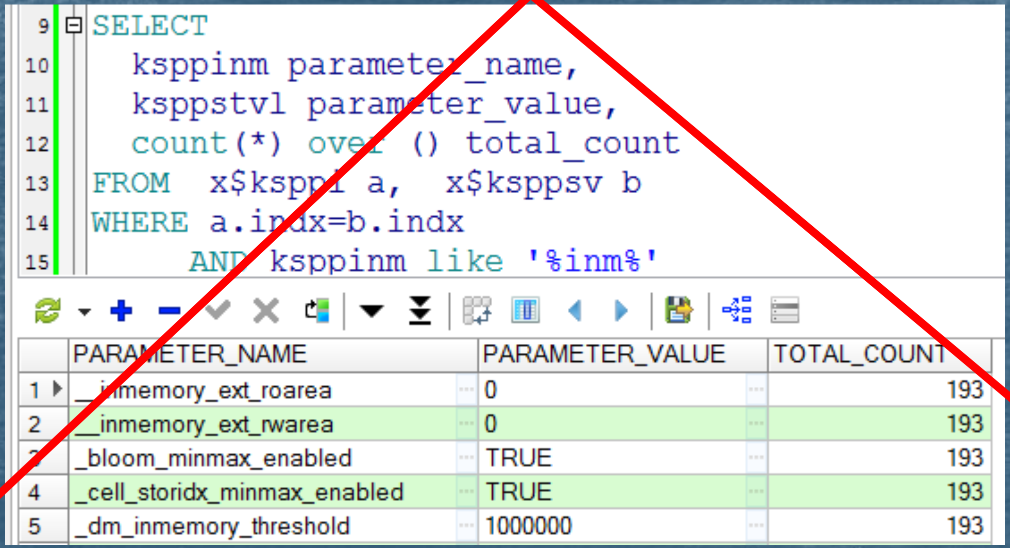
TOP SECRET



```
select s.name,
       count(*) over () total_count
from v$statname s
where name like '%IM %'
```

	NAME	TOTAL_COUNT
10	IM populate checkpoint time (ms)	523
11	IM populate direct read time (ms)	523
12	IM populate cache read time (ms)	523
13	IM repopulate blocks invalid	523
14	IM repopulate transactions check	523

More than 500 statistics!

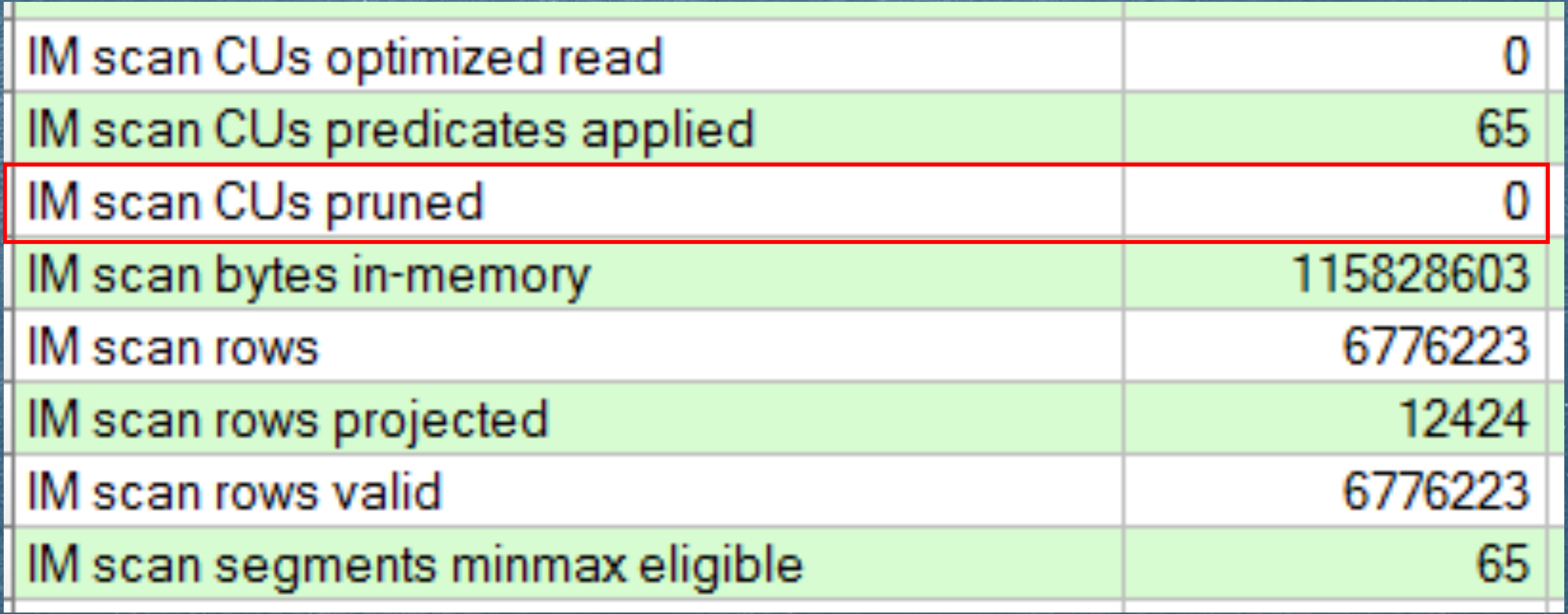


```
SELECT
  ksppinm parameter_name,
  ksppstvl parameter_value,
  count(*) over () total_count
FROM  x$ksppi a,  x$ksppsv b
WHERE a.indx=b.indx
     AND ksppinm like '%inm%'
```

	PARAMETER_NAME	PARAMETER_VALUE	TOTAL_COUNT
1	__inmemory_ext_roarea	0	193
2	__inmemory_ext_rwarea	0	193
3	_bloom_minmax_enabled	TRUE	193
4	_cell_storidx_minmax_enabled	TRUE	193
5	_dm_inmemory_threshold	1000000	193

More than 190 parameters!

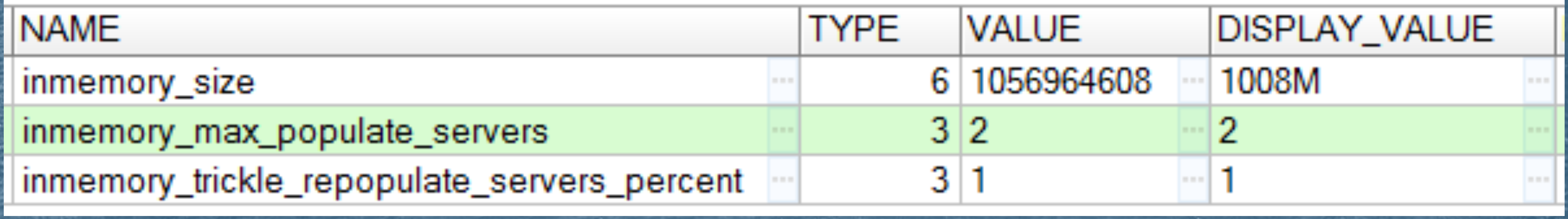
TOP SECRET



IM scan CUs optimized read	0
IM scan CUs predicates applied	65
IM scan CUs pruned	0
IM scan bytes in-memory	115828603
IM scan rows	6776223
IM scan rows projected	12424
IM scan rows valid	6776223
IM scan segments minmax eligible	65

8 statistics covers 90% cases!

Troubleshooting



NAME	TYPE	VALUE	DISPLAY_VALUE
inmemory_size	6	1056964608	1008M
inmemory_max_populate_servers	3	2	2
inmemory_trickle_repopulate_servers_percent	3	1	1

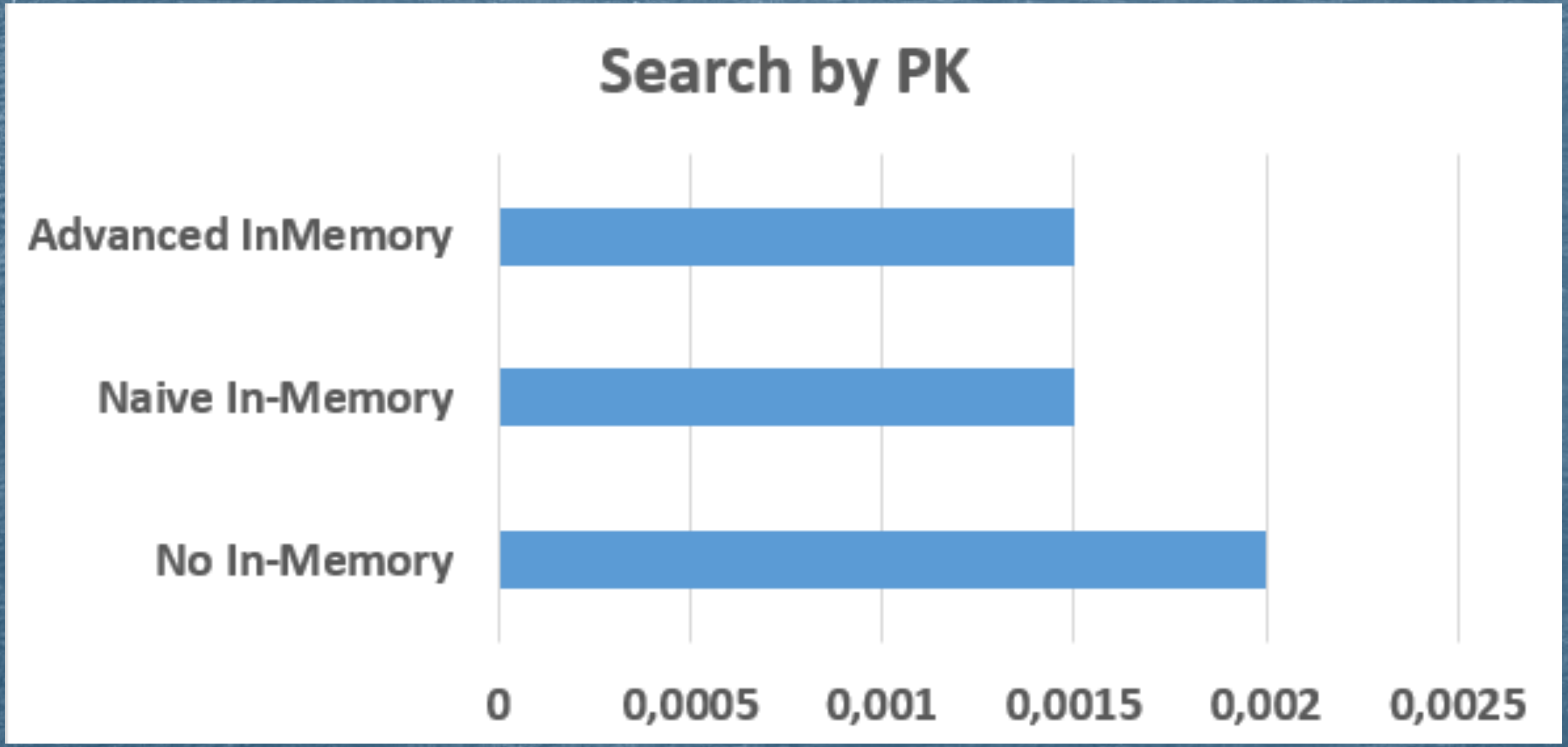
3 parameters covers 90% cases!

Troubleshooting

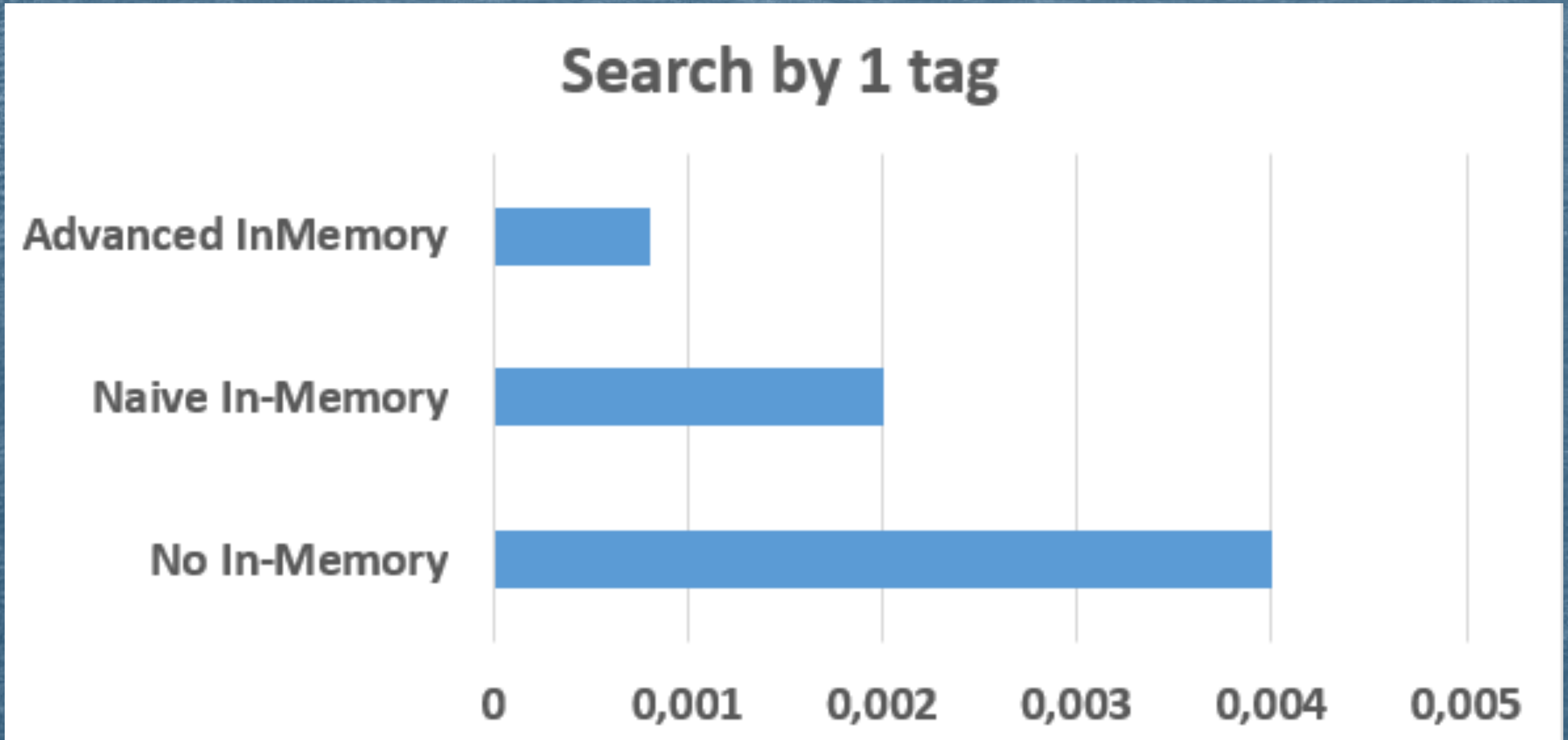
View name	Description
V$IM_COL_CU	SMU detailed information per column
V$IM_SMU_HEAD	SMU header statistics
v$im_segments	Inmemory segment parameters

3 views covers 90% cases!

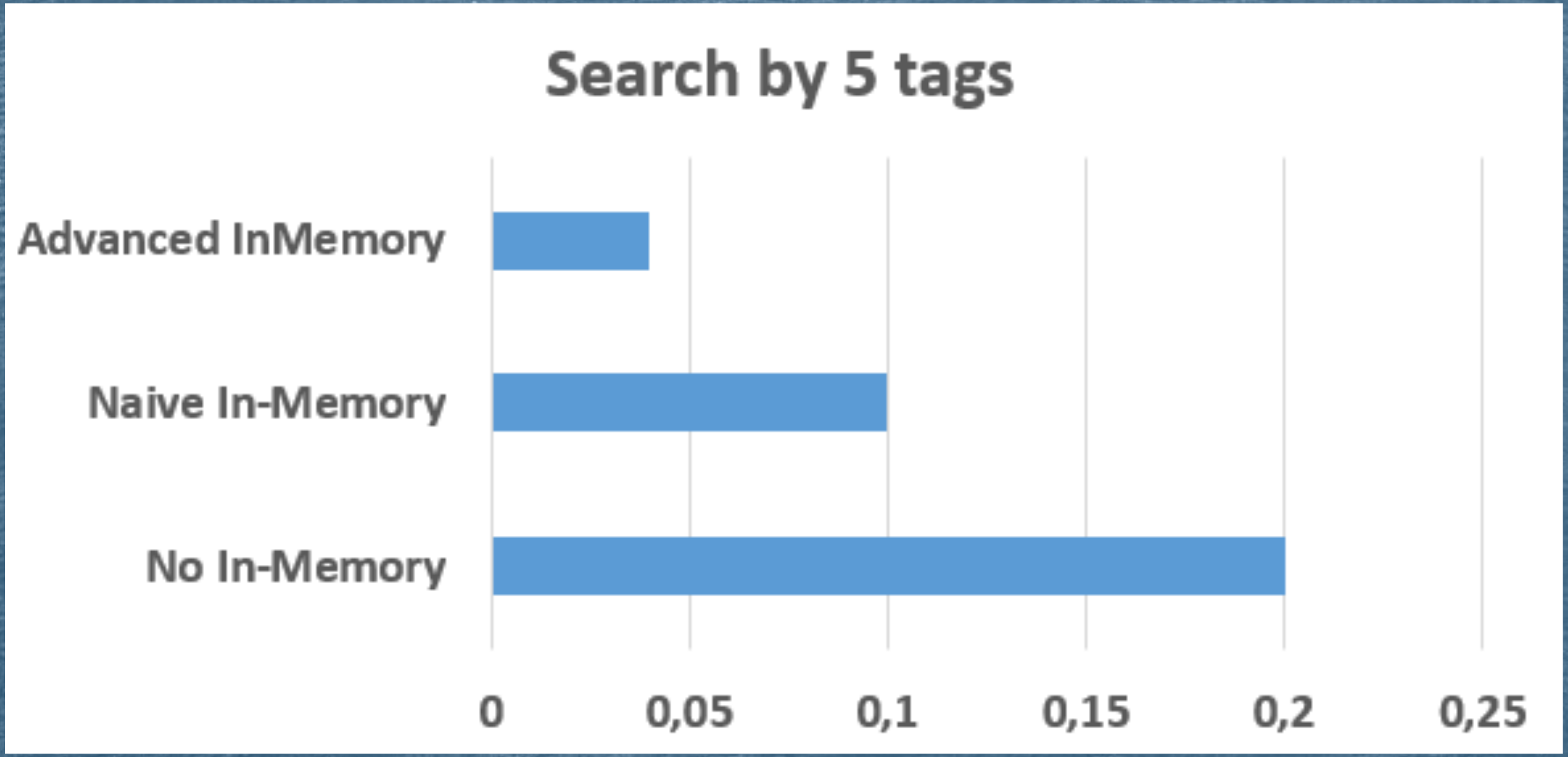
Performance with In-Memory final



Performance with In-Memory final



Performance with In-Memory final



DBAs findings

- InMemory size <> table data size
- All data InMemory <> High performance
- Decent time to be loaded
- 8 IM statistics is enough
- Trickle parameters relevant to workload

Developers finding

- Advanced IM features <> significant profit our case
- Zone maps = ***`numeric`*** and ***`date`*** data type only
- Dictionary pruning – not in Oracle InMemory
- Simple data types = High perfomance
- High compression <> Slow ingestion

Furthers plans

- Add extra memory ☺
- Implement a POC with IM DWH
- Understand all 523 IM statistics + 190 parameters
- Use FastStart to speedup database wakeup
- Play with Oracle 18c features

Conclusion

- Always try and measure
- IM works for short queries as well
- Understanding of IM internals is a must
- Application changes are required
- No extra software/hardware introduced
- Fast POC followed by production deploy
- 5x times performance boost

Thank you for your time!

Alexander Tokarev
Database expert
DataArt
shtock@mail.ru
https://github.com/shtock
https://www.linkedin.com/in/alexander-tokarev-14bab230