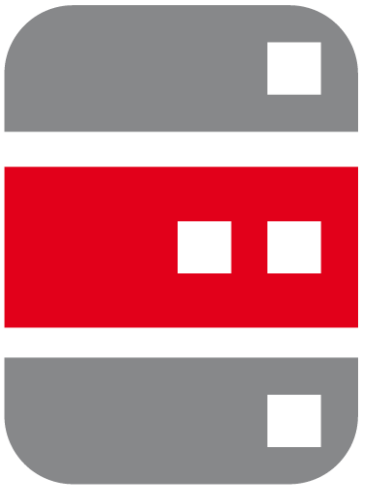
performing databases



Your reliability. Our concern.

performing databases
Your reliability. Our concern.

#1290
Oracle Core: InMemory Column Store

Martin Klier

Performing Databases GmbH
Mitterteich / Germany

Speaker

performing databases
Your reliability. Our concern.

- Martin Klier
- Solution Architect and Database Expert

- My focus
  - Performance Optimization
  - High Availability
  - Architecture DBMS

- Linux since 1997
- Oracle Database since 2003

Speaker

performing databases
Your reliability. Our concern.

- Meet & Greet

DOAG
Regionalgruppen
Fachkonferenzen

ORACLE OPEN WORLD

IOUG
independent oracle users group

COLLABORATE
April 2016

- Contact: martin.klier@performing-db.com
- Weblog: http://www.usn-it.de (English)

performing databases
Your reliability. Our concern.

DOAG

Conference + Exhibition
#DOAG2016

2000+ Visitors

Nuremberg, Germany
November 15.-18. 2016

Performing Databases

performing databases
Your reliability. Our concern.

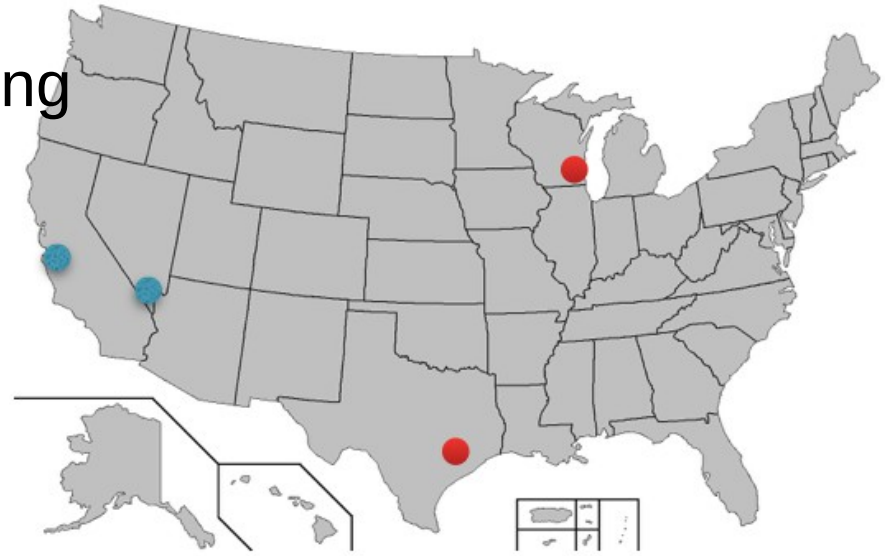
- Experts for Database Technology
  - Concept
  - Planning & Sizing
  - Licensing
  - Implementation and Troubleshooting
- Get in touch
  - Performing Databases GmbH
    Wiesauer Straße 27
    95666 Mitterteich, GERMANY
  - Web: http://www.performing-databases.com
  - Twitter: @PerformingDB

International

performing databases
Your reliability. Our concern.

ORACLE Gold Partner

- Design
- Licensing
- Implementation
- Tuning
- Troubleshooting
- Service
- Upgrade
- Migration



http://www.performing-databases.com

Agenda

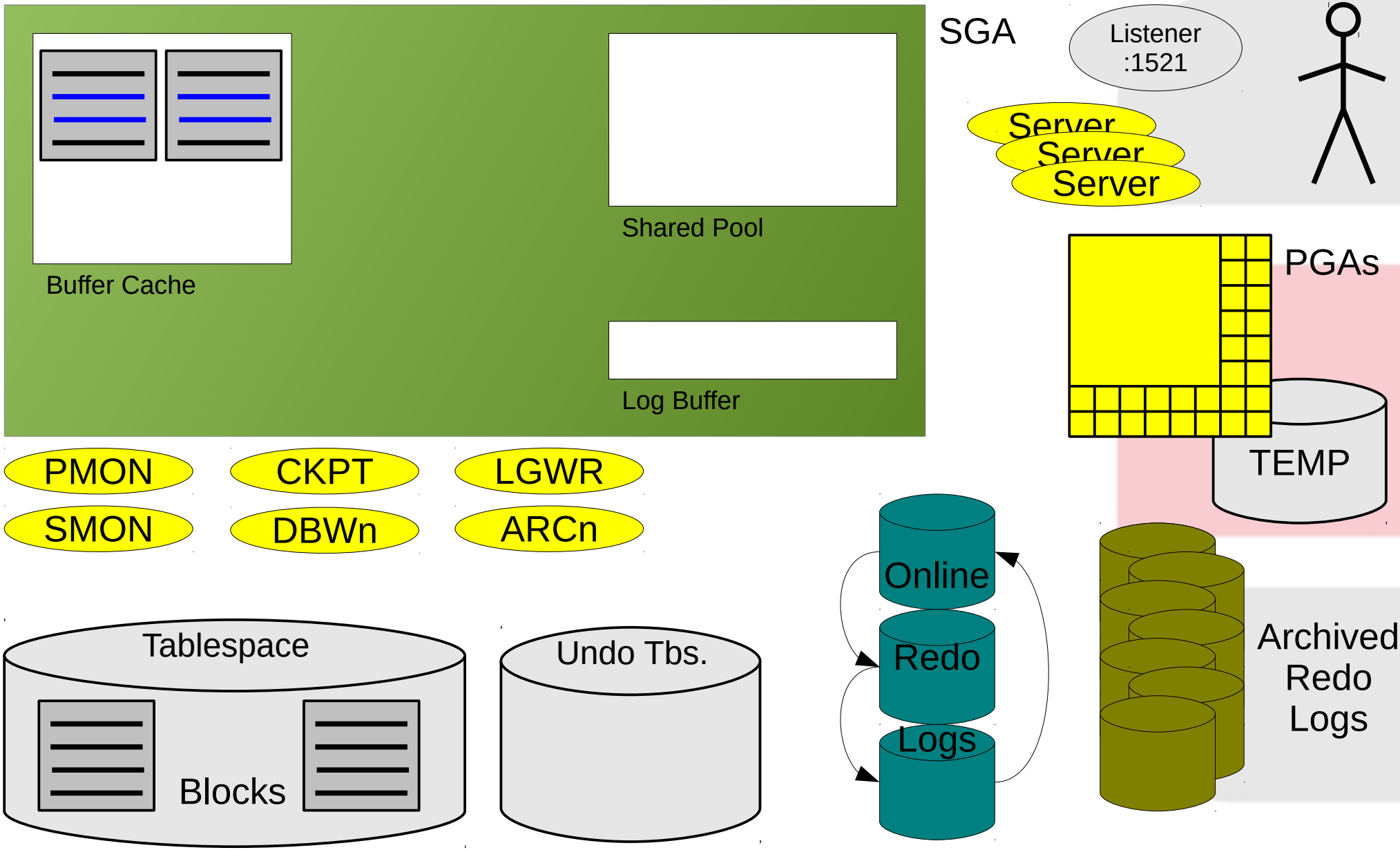
performing databases
Your reliability. Our concern.

- Let's refresh Oracle Architecture
- Row- and Column based data
- System architecture w/ InMemory Column Store
- How it works
  - Reading
  - Transactions and consistency
- Usability and Limitations

Basics

Oracle Architecture
(simplified)

performing databases
Your reliability. Our concern.



performing databases
Your reliability. Our concern.

Rows & Columns

Row Data

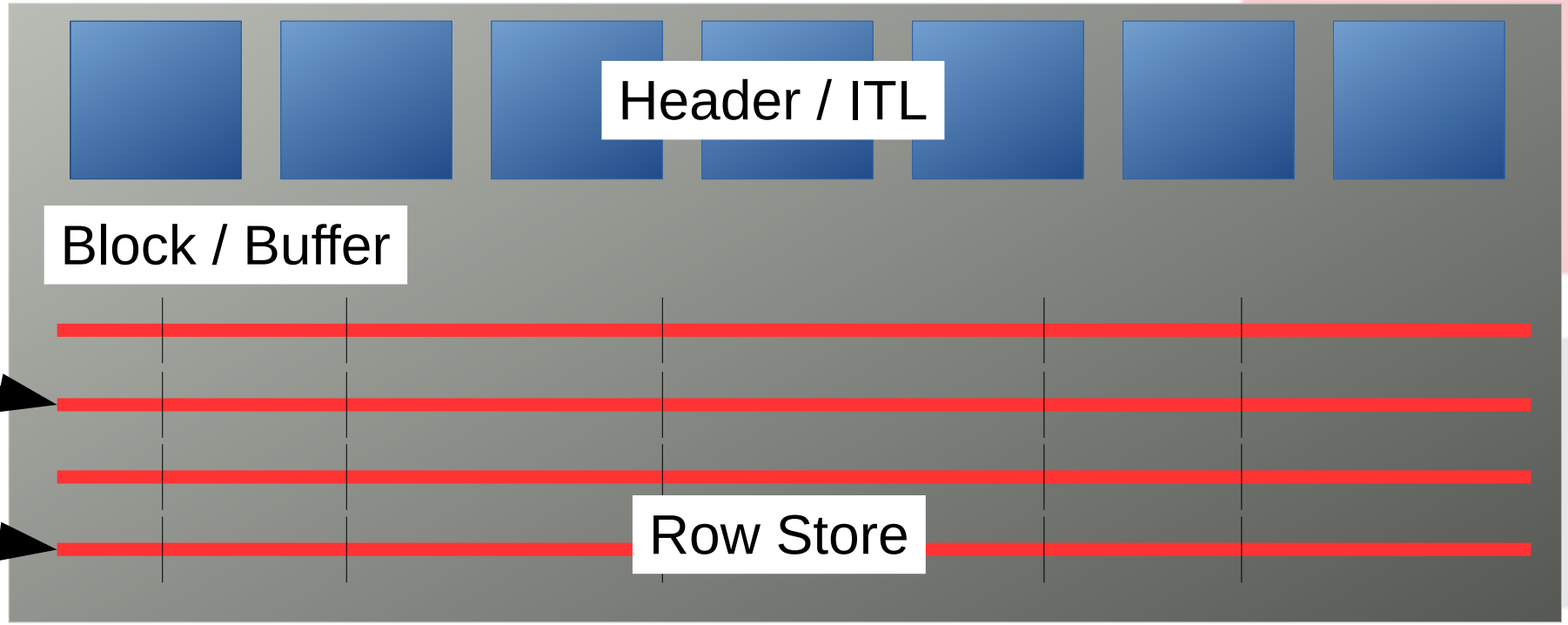
SELECT * … WHERE ID IN (3,36) ….

ID	NAME1	NAME2	STREET	ZIP	CITY
2	John	Doe	2 Wilbury Way	12345	Wesley
3	Alice	Allison	108 Agora Alley	23456	Amberg
4	Sally	Salinger	17 Samson St.	34567	Salisbury
..	..	..	..	..	..
..	..	..	..	..	..
..	..	..	..	..	..
36	..	..	..	..	..
..	..	..	..	..	..
57	Bob	Bobson	42 Century Ct.	56789	Taccanooga

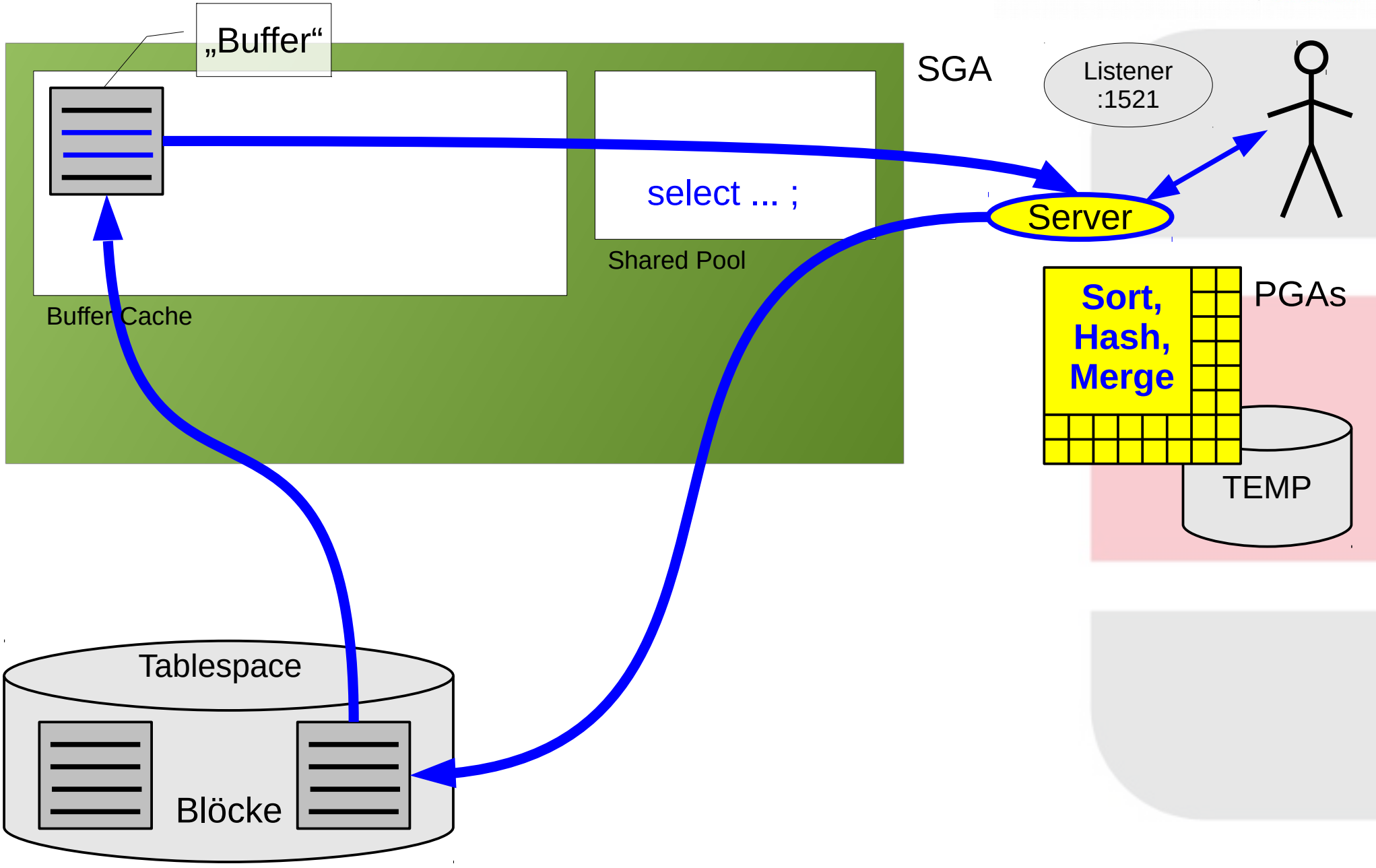
Row Data

performing databases
Your reliability. Our concern.

ID	NAME1	NAME2	STREET	ZIP	CITY
2	John	Doe	2 Wilbury Way	12345	Wesley
3	Alice	Allison	108 Agora Alley	23456	Amberg
4	Sally	Salinger	17 Samson St.	34567	Salisbury
8	Bob	Bobson	42 Century Ct.	56789	Taccanooga

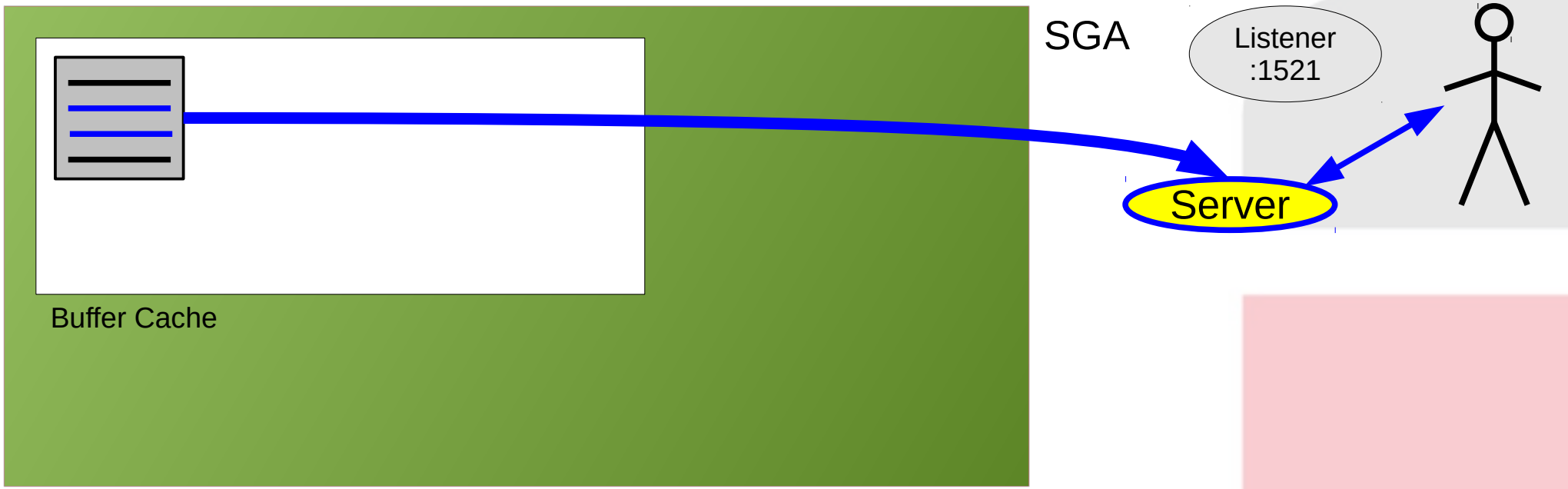


Reading a Row



Reading a Row

performing databases
Your reliability. Our concern.



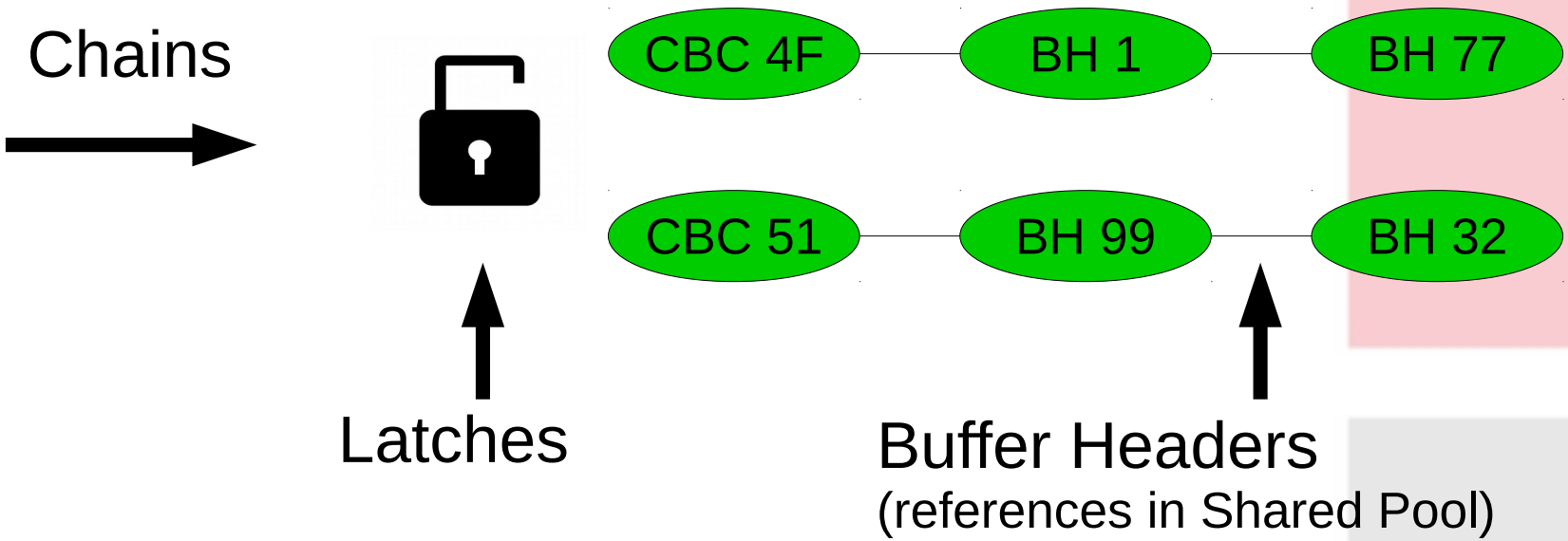
Maximum
1,000,000 rows per second
by design.

(Tirthankar Lahiri, Oracle VP)

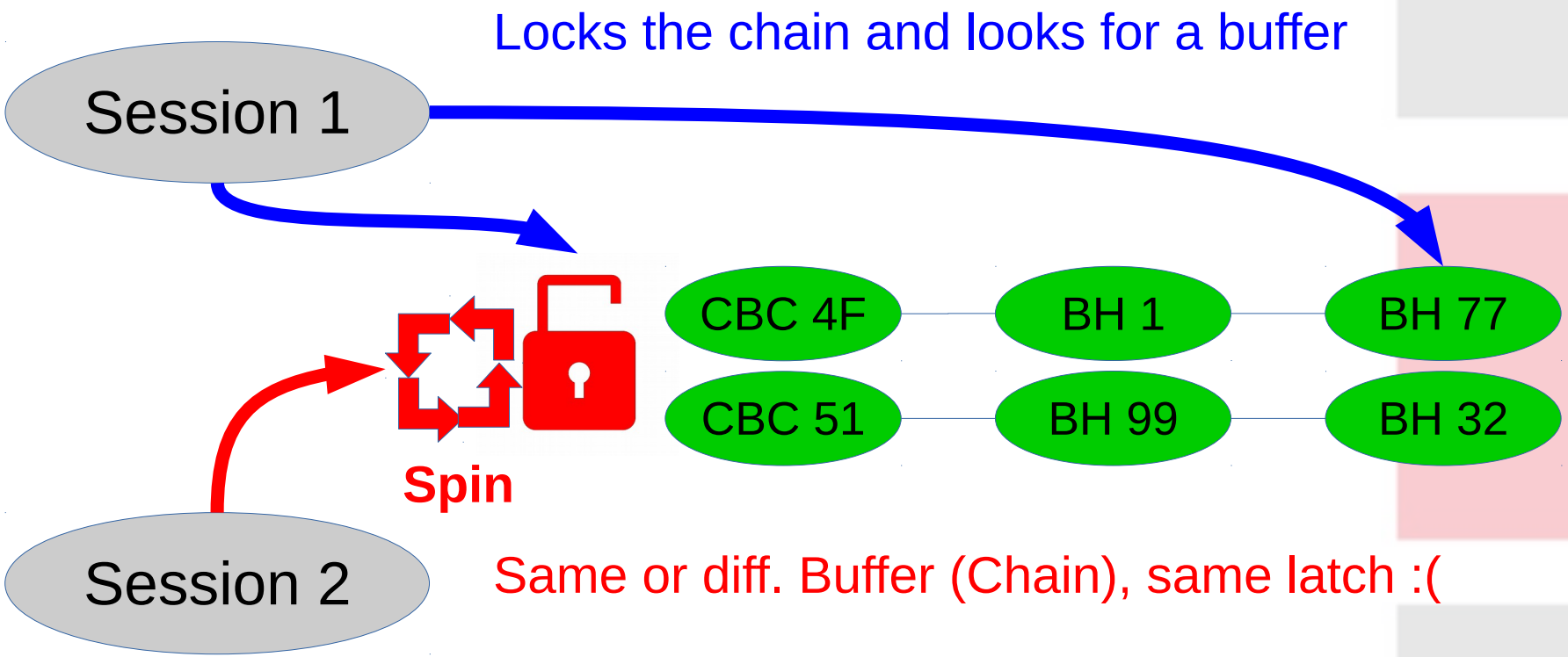
Buffer Cache Access

performing databases
Your reliability. Our concern.

Cache Buffer Chains: Is this block in the BC?



Buffer Cache Access



performing databases
Your reliability. Our concern.

Rows & Columns

Columnar Data

SELECT ID …. WHERE ZIP ….;

SELECT COUNT(*) …. WHERE ZIP ….;

ID	NAME1	NAME2	STREET	ZIP	CITY
2	John	Doe	2 Wilbury Way	12345	Wesley
3	Alice	Allison	108 Agora Alley	23456	Amberg
4	Sally	Salinger	17 Samson St.	34567	Salisbury
..	..	..	..	..	..
..	..	..	..	..	..
..	..	..	..	..	..
36	..	..	..	..	..
..	..	..	..	..	..
57	Bob	Bobson	42 Century Ct.	56789	Taccanooga

Columnar Data

performing databases
Your reliability. Our concern.

ID	NAME1	NAME2	STREET	ZIP	CITY
2	John	Doe	2 Wilbury Way	12345	Wesley
3	Alice	Allison	108 Agora Alley	23456	Amberg
4	Sally	Salinger	17 Samson St.	34567	Salisbury
..	..	..	..	..	..
..	..	..	..	..	..

IMCU

ROWID
AABBCC
BBCCDD
DDEEFF
..
..

ID
2
3
4
..
..

ZIP
12345
23456
34567
..
..

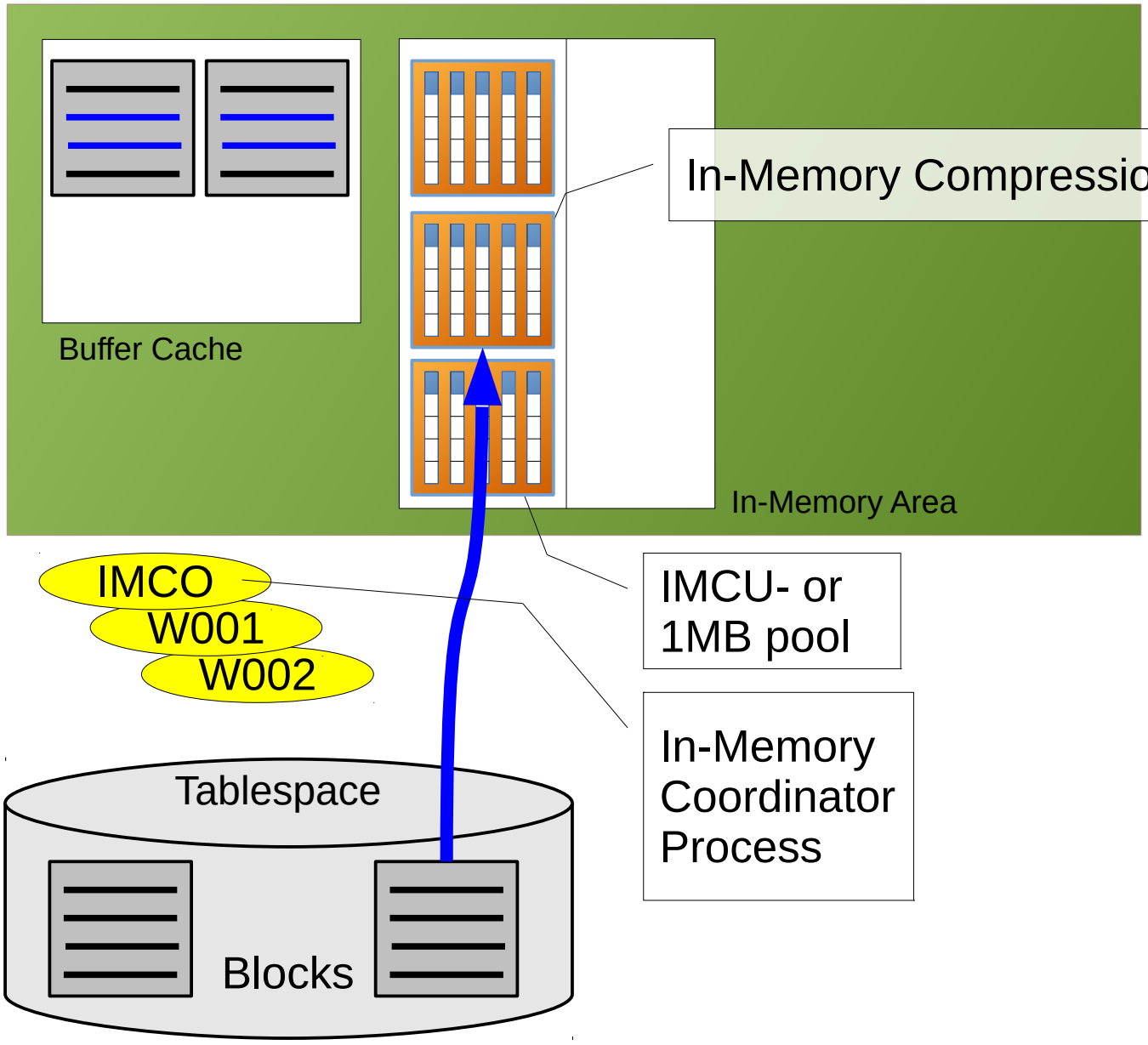
City
Wesley
Amberg
Salisbury
..
..

Column Compression Units (CU)

performing databases
Your reliability. Our concern.

In-Memory Architecture

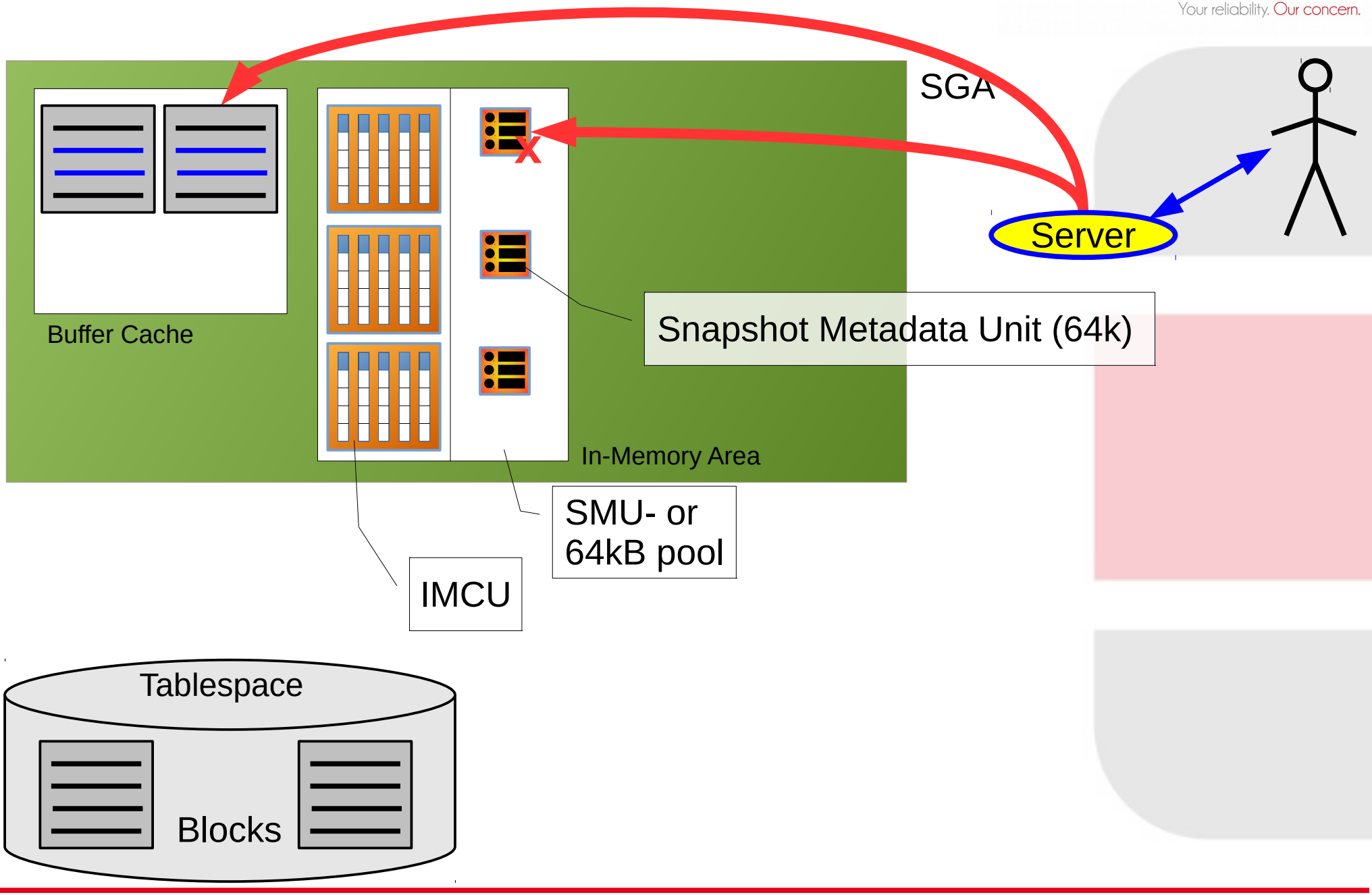
In-Memory Population



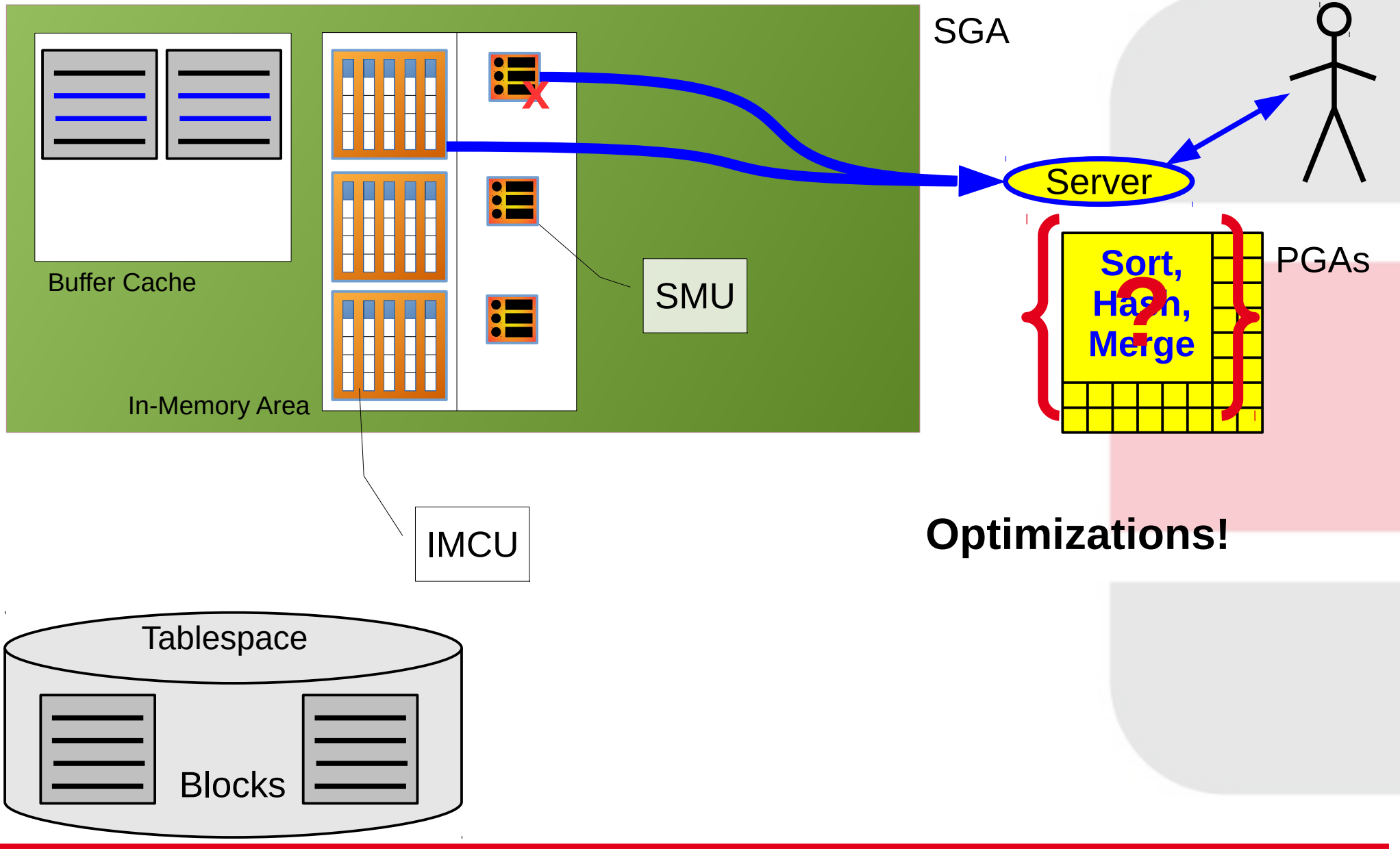
Order-Priority per table

- critical
- high
- medium
- low
- none (=on demand ?)

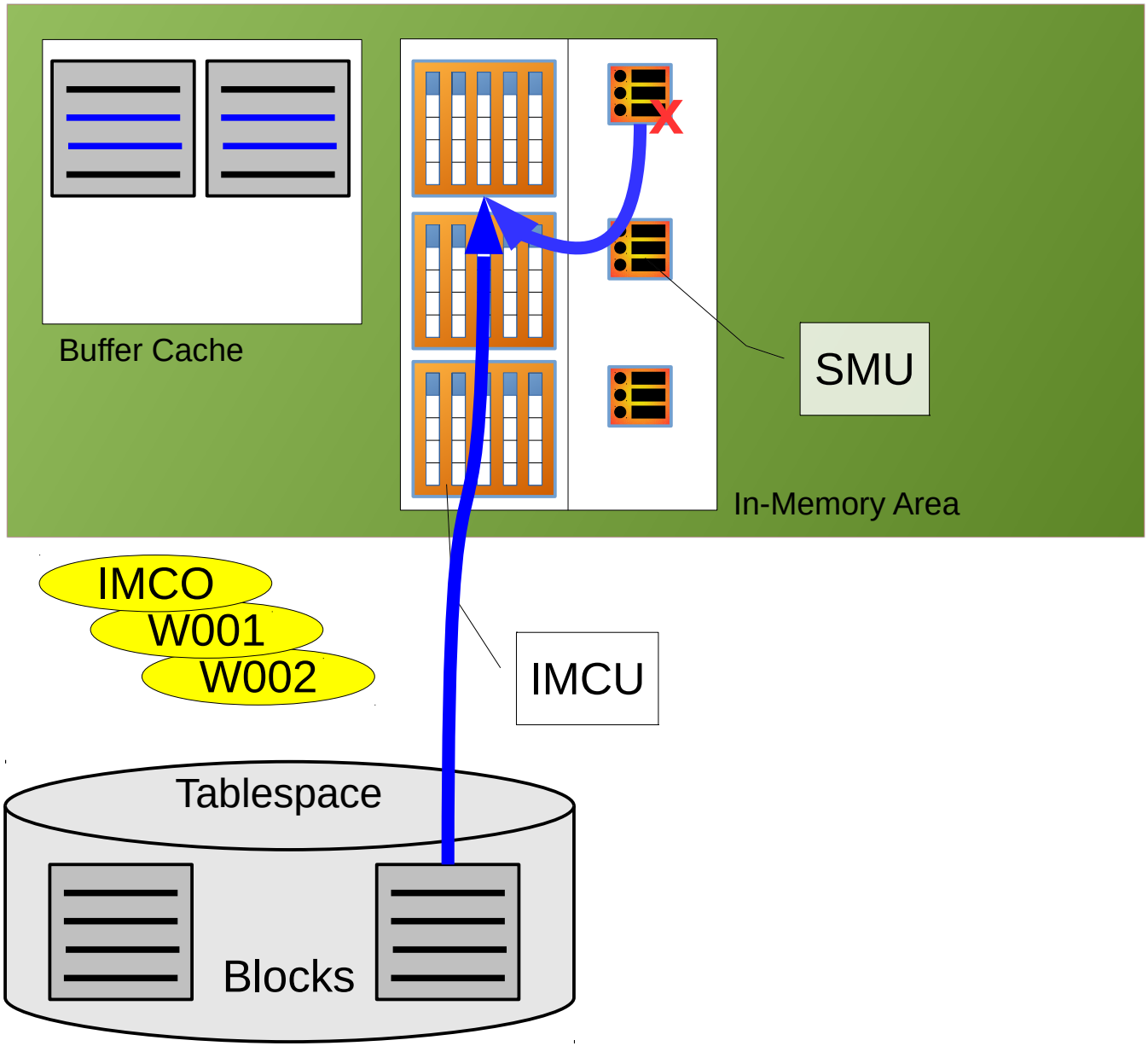
In-Memory Data Change



Lesevorgang Columnar



In-Memory RePopulation

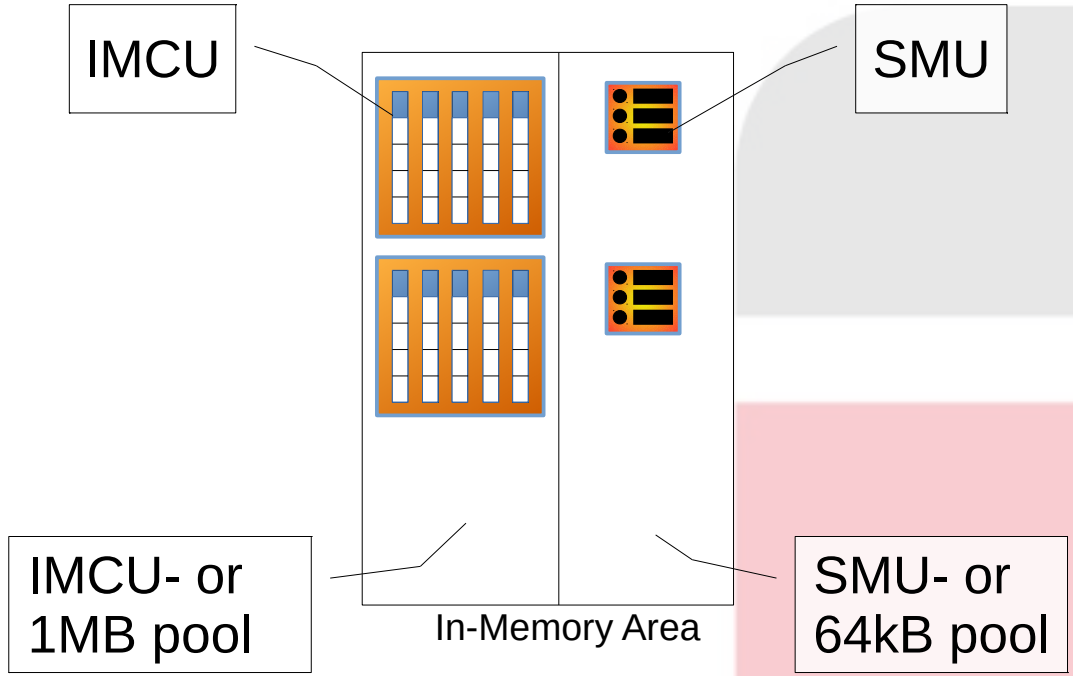


When?

- Threshold per IMCU
- Trickle (constant)

InMemory Area

- Segregated in
  - 1MB-Pool (IMCUs)
  - 64k-Pool (SMUs)



- **NO** Least-Recently-Used-Mechanism (LRU)

  First come, first serve!

  => Think**: „Tablespace“** not ~~„Cache“~~

performing databases
Your reliability. Our concern.

Using Column Store

With Or Without You

performing databases
Your reliability. Our concern.

```
select /*+ noparallel no_inmemory */ count(*) from imtable

Plan hash value: 1601702119

-------------------------------------------------------------------------
| Id  | Operation          | Name    | Rows  | Cost (%CPU)| Time     |
-------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |         |     1 |  26650   (1)| 00:00:02 |
|   1 |  SORT AGGREGATE    |         |     1 |             |          |
|   2 |   TABLE ACCESS FULL| IMTABLE |  5810K|  26650   (1)| 00:00:02 |
-------------------------------------------------------------------------
```

+

```
select /*+ noparallel */ count(*) from imtable

Plan hash value: 1601702119

-------------------------------------------------------------------------------------
| Id  | Operation                   | Name    | Rows  | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT            |         |     1 |   1140   (3)| 00:00:01 |
|   1 |  SORT AGGREGATE             |         |     1 |             |          |
|   2 |   TABLE ACCESS INMEMORY FULL| IMTABLE |  5810K|   1140   (3)| 00:00:01 |
-------------------------------------------------------------------------------------
```

When and When Not

performing databases
Your reliability. Our concern.

```
SQL_ID  a202cw0jn1trn, child number 0
-------------------------------------
select /*+ noparallel */ object_id, count(*) x from imtable group by
object_id

Plan hash value: 186096447

-------------------------------------------------------------------------------------------
| Id  | Operation                   | Name    | E-Rows |E-Bytes|E-Temp | Cost (%CPU)| E-Time   |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT            |         |        |       |       |  7741 (100)|          |
|   1 |  HASH GROUP BY              |         |  90895 |   443K|    66M|  7741   (4)| 00:00:01 |
|   2 |   TABLE ACCESS INMEMORY FULL| IMTABLE |   5810K|    27M|       |  1174   (5)| 00:00:01 |
-------------------------------------------------------------------------------------------
```

+

```
SQL_ID  9n0h8a6c4jn5d, child number 0
-------------------------------------
select /*+ noparallel */ object_id, count(*) x  from imtable where
object_id=82345 group by object_id

Plan hash value: 1486712282

-----------------------------------------------------------------------------------------
| Id  | Operation             | Name    | E-Rows |E-Bytes| Cost (%CPU)| E-Time   |
-----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |         |        |       |     3 (100)|          |
|   1 |  SORT GROUP BY NOSORT |         |     64 |   320 |     3   (0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN    | I_OID_1 |     64 |   320 |     3   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------
```

Operation Pushdown I


```
SQL_ID  91063q5ku7j07, child number 0
-------------------------------------
select /*+ noparallel */ object_id, count(*) x  from imtable where
object_id>70000 and object_id<85000 group by object_id

Plan hash value: 186096447

-------------------------------------------------------------------------------------------
| Id  | Operation                   | Name    | E-Rows |E-Bytes| Cost (%CPU)| E-Time   |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT            |         |        |       |  1207 (100)|          |
|   1 |  HASH GROUP BY              |         |  14854 | 74270 |  1207   (8)| 00:00:01 |
|*  2 |   TABLE ACCESS INMEMORY FULL| IMTABLE |    949K|  4636K|  1175   (6)| 00:00:01 |
-------------------------------------------------------------------------------------------

Query Block Name / Object Alias (identified by operation id):
-------------------------------------------------------------

   1 - SEL$1
   2 - SEL$1 / IMTABLE@SEL$1

Outline Data
-------------

  /*+
      BEGIN_OUTLINE_DATA
      IGNORE_OPTIM_EMBEDDED_HINTS
      OPTIMIZER_FEATURES_ENABLE('12.1.0.2')
      DB_VERSION('12.1.0.2')
      ALL_ROWS
      OUTLINE_LEAF(@"SEL$1")
      FULL(@"SEL$1" "IMTABLE"@"SEL$1")
      USE_HASH_AGGREGATION(@"SEL$1")
      END_OUTLINE_DATA
  */

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - inmemory(("OBJECT_ID">70000 AND "OBJECT_ID"<85000))
       filter(("OBJECT_ID">70000 AND "OBJECT_ID"<85000))
```

Operation Pushdown II

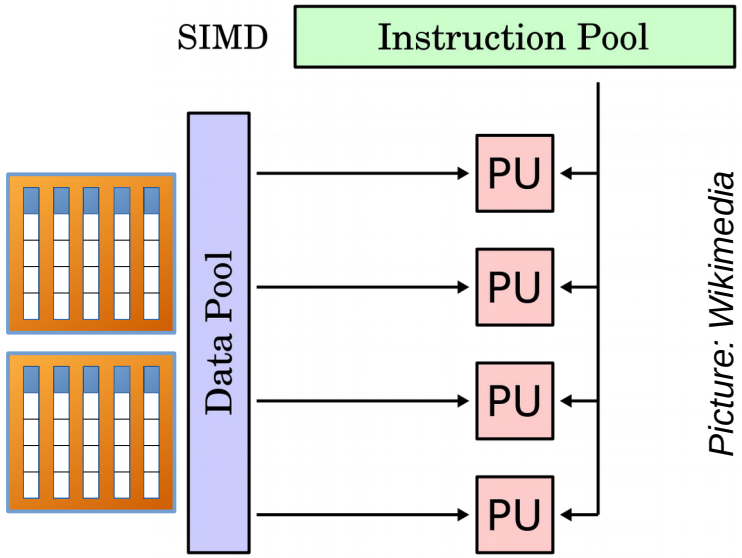

```
select /*+ noparallel */ count(*)
from bigthing bt, imtable imt
where bt.OBJECT_ID=imt.OBJECT_ID
  and imt.OBJECT_ID between 80000 and 85000
```

```
Plan hash value: 2908376589

-------------------------------------------------------------------------------------------------
| Id  | Operation                          | Name     | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                   |          |     1 |    10 |       | 11873   (5)| 00:00:01 |
|   1 |  SORT AGGREGATE                    |          |     1 |    10 |       |            |          |
|*  2 |   HASH JOIN                        |          |  8835K|    84M|  5264K| 11873   (5)| 00:00:01 |
|   3 |    JOIN FILTER CREATE              | :BF0000  |   316K|  1546K|       |  1173   (5)| 00:00:01 |
|*  4 |     TABLE ACCESS INMEMORY FULL     | IMTABLE  |   316K|  1546K|       |  1173   (5)| 00:00:01 |
|   5 |    JOIN FILTER USE                 | :BF0000  |  2537K|    12M|       |  8362   (6)| 00:00:01 |
|*  6 |     TABLE ACCESS INMEMORY FULL     | BIGTHING |  2537K|    12M|       |  8362   (6)| 00:00:01 |
-------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("BT"."OBJECT_ID"="IMT"."OBJECT_ID")
   4 - inmemory("IMT"."OBJECT_ID">=80000 AND "IMT"."OBJECT_ID"<=85000)
       filter("IMT"."OBJECT_ID">=80000 AND "IMT"."OBJECT_ID"<=85000)
   6 - inmemory("BT"."OBJECT_ID">=80000 AND "BT"."OBJECT_ID"<=85000 AND
              SYS_OP_BLOOM_FILTER(:BF0000,"BT"."OBJECT_ID"))
       filter("BT"."OBJECT_ID">=80000 AND "BT"."OBJECT_ID"<=85000 AND
              SYS_OP_BLOOM_FILTER(:BF0000,"BT"."OBJECT_ID"))
```

Optimizations

performing databases
Your reliability. Our concern.

- Vector Processing (Single Instruction Multiple Data SIMD)
  z.B. Intel AVX / 256bit Register



Picture: Wikimedia

>1,000,000,000
rows per second
(Tirthankar Lahiri, Oracle VP)

Optimizations

performing databases
Your reliability. Our concern.

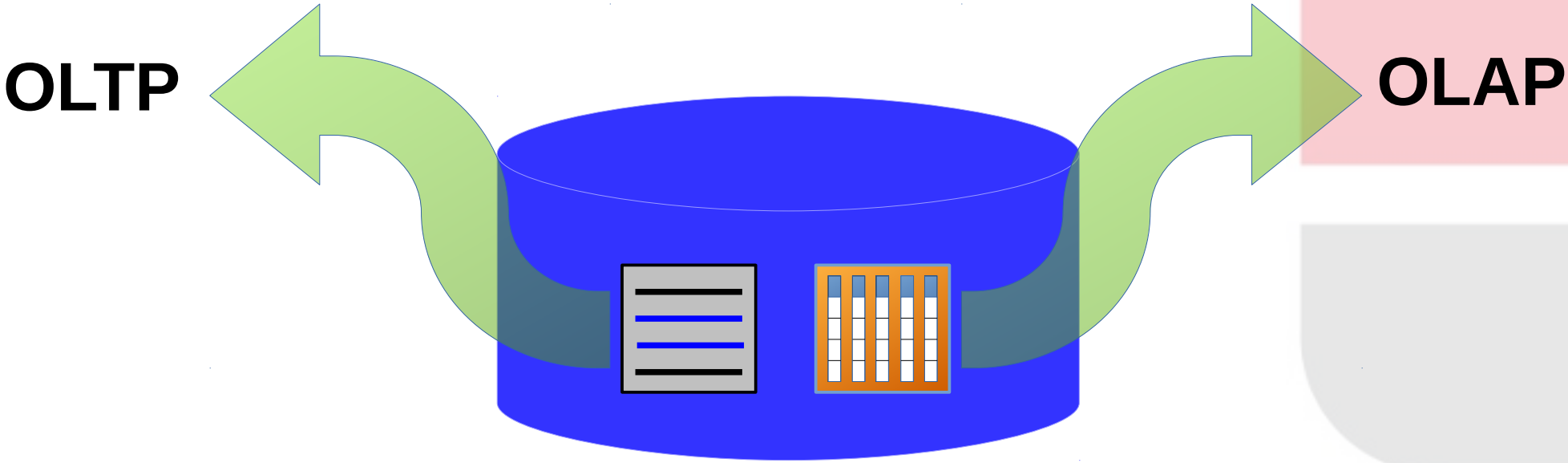
- Operation Pushdown: InMemory Scan (*)
  (like Exadata Smart Scan)
  - Predicate check in Column Store (*)
  - Aggregation during Scan
  - Bloom Filter in IMCU Scan (*)
- In-Memory Storage Index
  (Skip unneeded IMCUs during In-Memory Scan)
- Predicate Optimization
  (if an IMCU is needed as a whole, the predicate is not checked for each row)

Finish run

Advantages IM Column St.

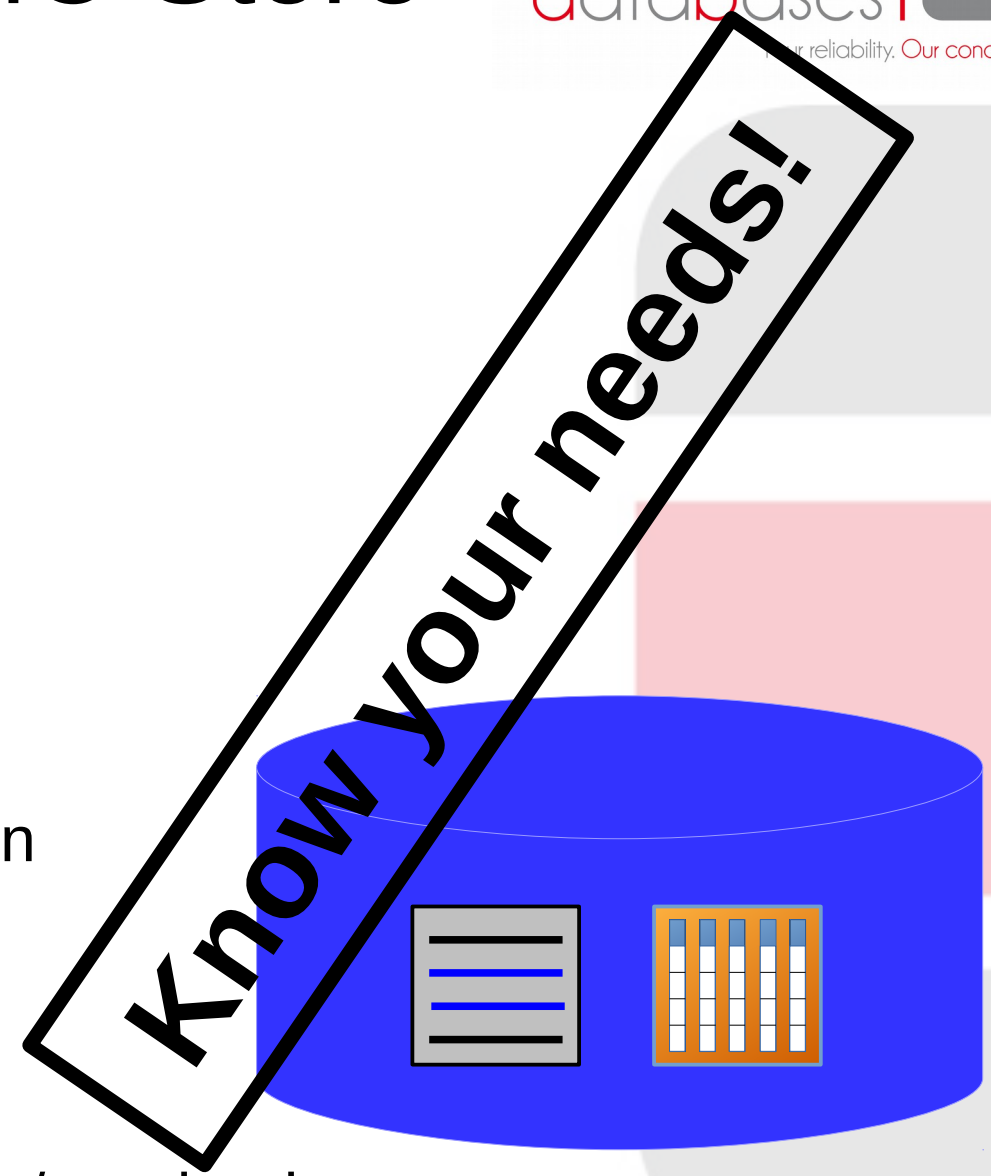
performing databases
Your reliability. Our concern.

- Faster for columnar operations (obviously)
- Optimizer uses „Shortcuts“ in the architecture
- Fully transparent, „mixed“ operations are normal



Disadvantages IMC Store

- Ressources
  - RAM
  - CPU for Compression
- Content maintenance
  => Advisor!
- Not fully deterministic:
  Advantage depends heavily on
  - Workload
  - Content
  - Condition of data
- Scalability in RAC crazy and w/o redundancy
  (available in Exadata only)



InMemory ist cool.
Cool is not enough.
You need a UseCase.

#FiveWordTechHorrors „InMemory solves all performance problems“

performing databases
Your reliability. Our concern.

Q & A

Download Presentation and Whitepaper
http://www.performing-databases.com

performing databases
Your reliability. Our concern.

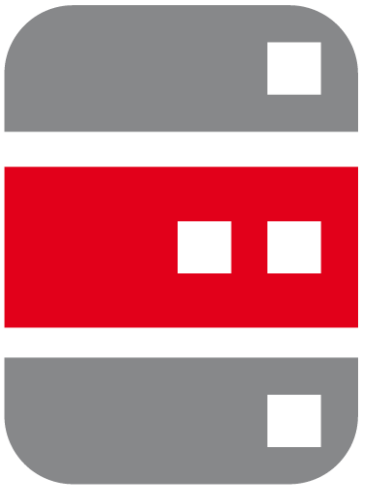
DOAG

Conference + Exhibition
#DOAG2016

2000+ Visitors

Nuremberg, Germany
November 15.-18. 2016

performing databases



Your reliability. Our concern.