Oracle In-Memory Column Store for BI

dbi services

About me

Franck Pachot
Senior consultant
Oracle Technology Leader

Mobile +41 79 963 27 22
franck.pachot@dbi-services.com
www.dbi-services.com

ORACLE Certified Professional
Oracle Database 12c
Administrator

ORACLE Certified Expert
Oracle Database 11g
Performance Tuning

ORACLE Certified Master
Oracle Database 11g
Administrator

ORACLE ACE

@FranckPachot

Developpez.com UK'OUG UK ORACLE USER GROUP DOAG SOUG

Who we are
dbi services

Experts At Your Service

- 40 specialists in IT infrastructure
- Certified, experienced, passionate

Based In Switzerland

- 100% self-financed Swiss company
- Over CHF 6 mio. turnover

Leading In Infrastructure Services

- More than 100 customers in CH, D, & F
- Over 35 SLAs dbi FlexService contracted

Certified Management System SQS ISO/IEC 20000-1

Agenda

1. Analytic queries
2. Row store and Column store
3. In-Memory population
4. Performance
5. How to use it

Analytic Queries
Where are your analytic queries?

Datawarehouse

- Ad-hoc queries aggregating data from lot of rows
- You offload it to another database (designed for full scans, joins, sorts,...)
- You design it for reporting (STAR schema, bitmap indexes, compression)

- Queries reading lot of rows filtered by few dimensions are analytic queries

OLTP

- 3NF data model with primary keys and foreign keys
- You insert rows (customer information, order lines,...)
- You retrieve rows (by primary key, navigating through joins)

- Queries not accessing from PK or selective index are analytic queries

Analytic Queries
Examples in OLTP

On your Order Entry application

- You want to highlight good customers (more than n ordered in prev. days)
- You want to show other articles related with the order

On your CRM, ERP

- You want to search by multicriteria screen
- You want to calculate some real-time indicators

Real-time Business Intelligence

...are you executing BI queries on your OLTP database?

Analytic Queries
Where are you doing analytic queries?

In Oracle, you don't need to offload reporting: it's only better

Oracle READS do not lock anything

- It's the Multi Versioning Consistency Model
- You can QUERY on the operational OLTP database
  - Access through secondary indexes partitioning, full scan

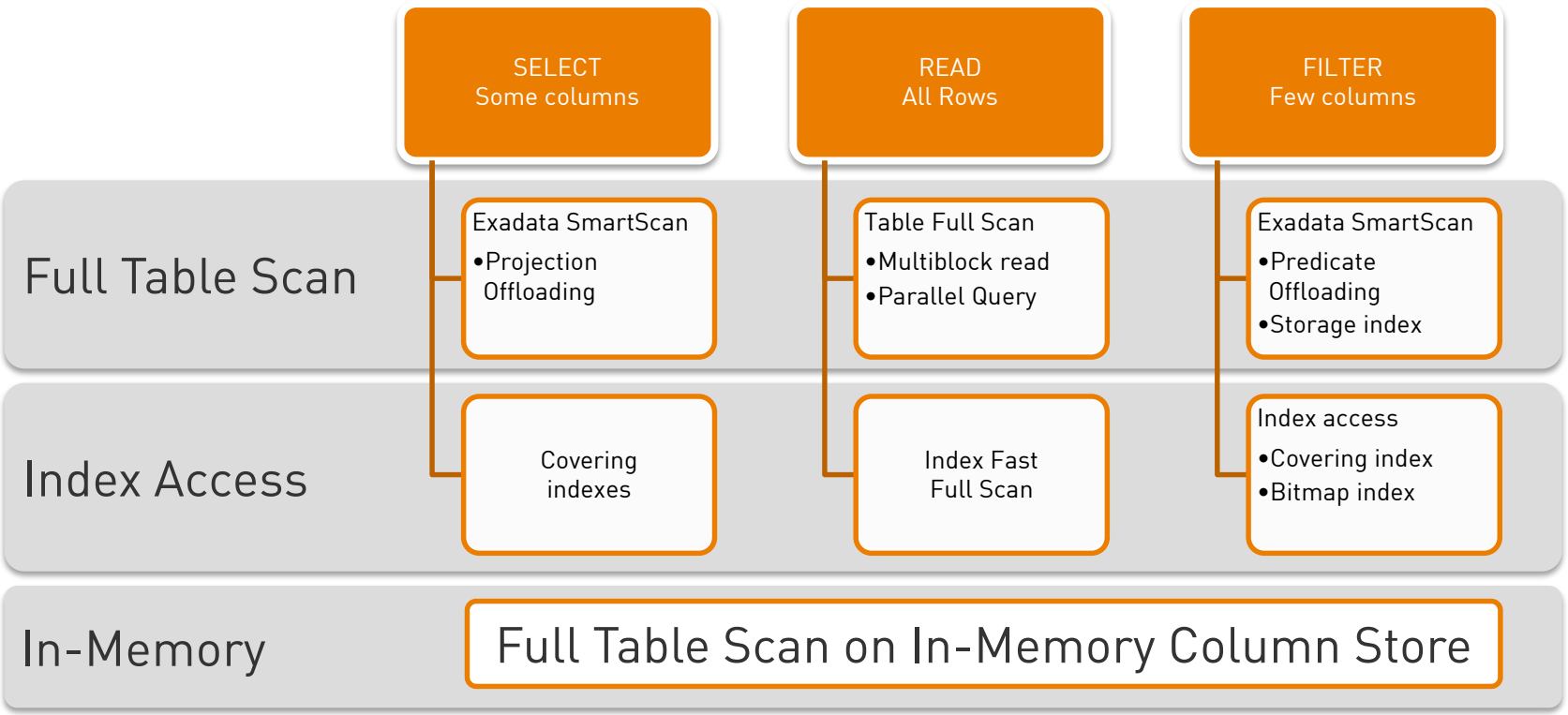
But it's still better to offload to a BI database

- Adding too many indexes slows DML
- Bitmap indexes are not suited for DML
- Memory sizing is different (High SGA for OLTP, High PGA for reporting)
- Availability requirements are not the same
- Keep more historical data that is purged on operation database

Analytic Queries
How are you doing analytic queries?

Analytic queries

- It's not: '*give me all columns for few specific rows*'
- It's: '*give some columns from all rows that match filter on few columns*'



Agenda

1. Analytic queries
2. Row store and Column store
3. In-Memory population
4. Performance
5. How to use it

Row store and Column store
Full scan

SELECT ENAME FROM EMP WHERE SAL>=3000

- We read all blocks, with all column values

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		20
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

Row store and Column store
Index scan

SELECT ENAME FROM EMP WHERE SAL>=3000

- We read full blocks, with all column values

SAL
800
950
1100
1250
1250
1300
1500
1600
2450
2850
2975
3000
3000
5000

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		20
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

Row store and Column store
In-Memory scan

SELECT ENAME FROM EMP WHERE SAL>=3000

- We read and process vectors of columns

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		20
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

Row store and Column store
Oracle choices about In-Memory implementation

Column Store is only In Memory

- No persistence of the column store
- Data is already persisted in the row store
  - Main reasons:
    - don't have to design new block format,
    - manage recovery, etc.

In-Memory is focused on analytic queries

- No index, No IOT, no out-of-line LOB,...

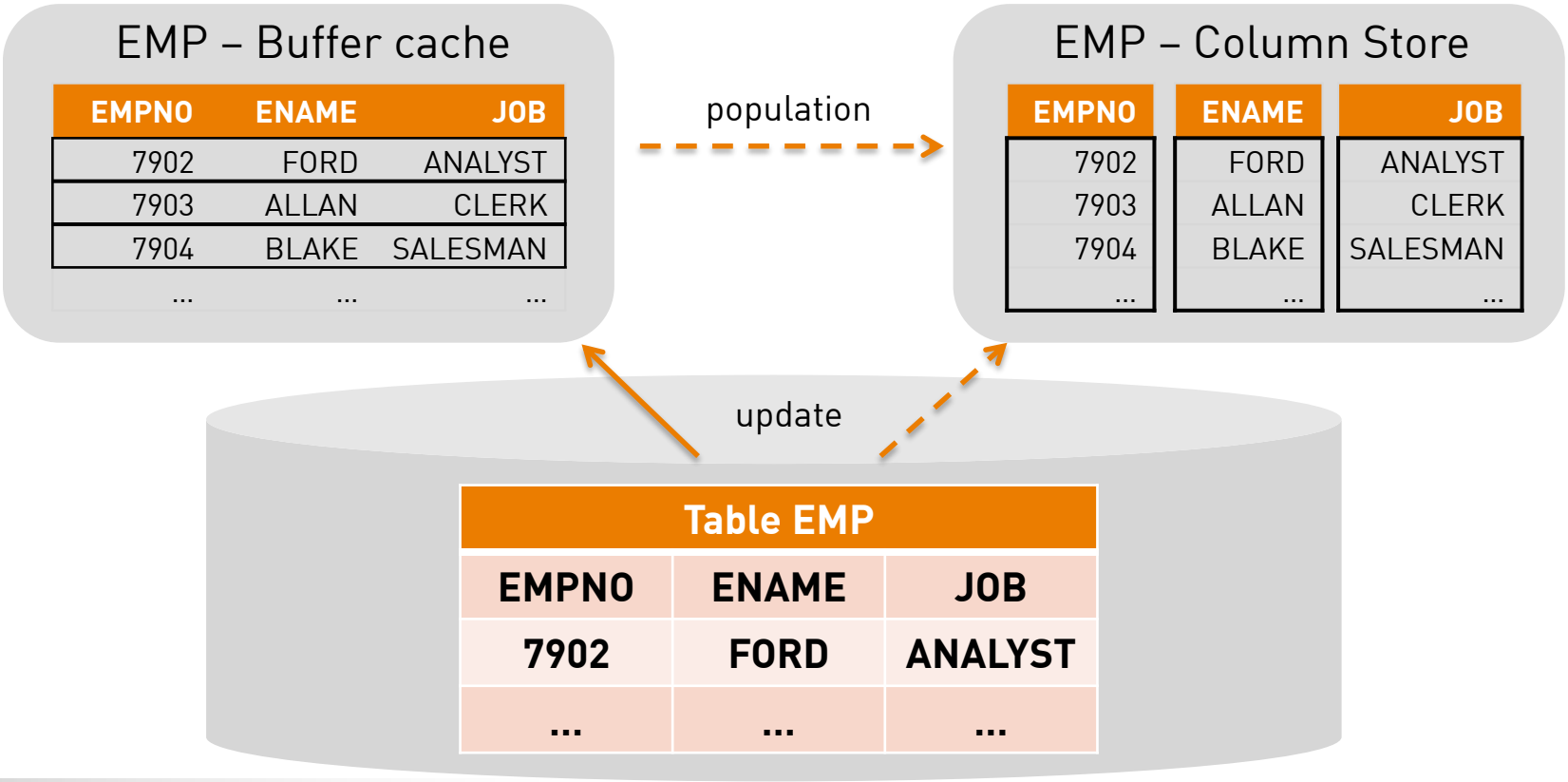
In-Memory is transparent for application

- The DBA assigns physical memory to the column store
- The data modeler defines which tables are populated in memory (and when – on first read or at instance startup, with priority)
- The CBO chooses the right store to access
- No need for denormalization, new indexes, materialized views,...

Row store and Column store
Dual format architecture

Oracle 12.1.0.2 introduces new columnar format

- New format does NOT replace existing row format
- Standard memory pools keeps rows format and Oracle block size structure



Row store and Column store
Dual format architecture

Row store is still there

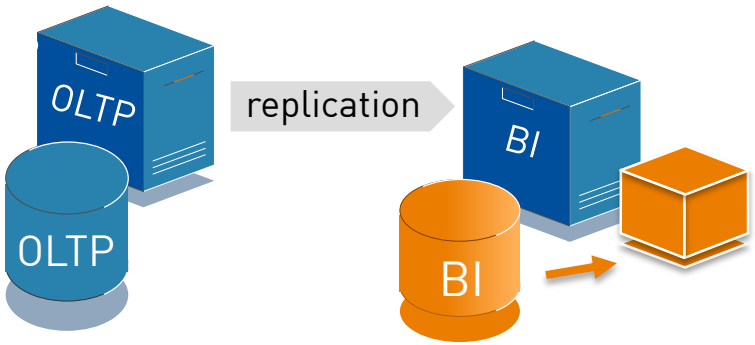
- Well proven mecanisms for:
  - Buffer cache algorithms, cache fusion RAC
  - **A**CID: transaction tables, ITL, global cache, ...
  - A**CI**D: undo, multi versioning concurency system, locks
  - ACI**D**: redo log, log writer, well known recovery mecanism

Column Store comes in addition to it

- Is populated in background
- Is maintained asynchronously
- But improves performance when populated
- Can fallback to buffer cache if not available/up-to-date
- Is a shared nothing architecture when in RAC
- Is not persistent: has to be populated when instance starts

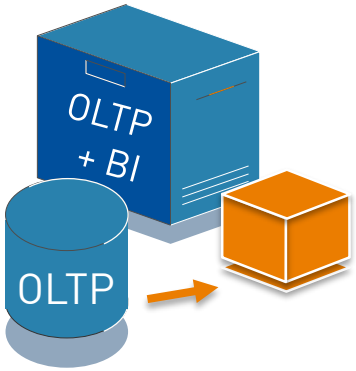
Row store and Column store
Dual format architecture

You can use In-Memory On your BI database



- Real-time replication
  - Golden Gate
  - Dbvisit replicate
- Add In-Memory to the BI database
  - Because bitmap indexes not possible with DML

You can use In-Memory On your OLTP database



- Add In-Memory directly to OLTP
  - It is a second store
- No need to
  - Replicate
  - Denormalize
  - Index differently

Agenda

1. Analytic queries
2. Row store and Column store
3. In-Memory population
4. Performance
5. How to use it

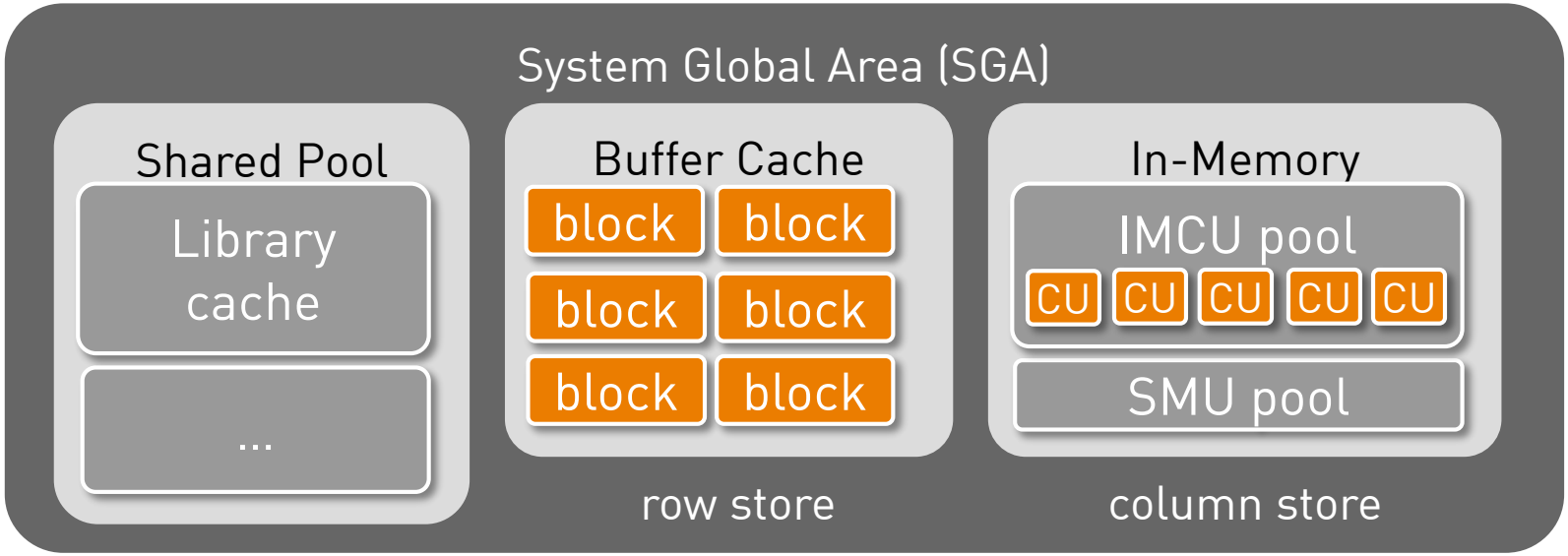
In-Memory Column Store in the SGA
Columnar storage – Dedicated pool

In-Memory Column Store is a new static pool in SGA

- In addition to the buffer cache (not a substitute)
- Populated and maintained asynchronously

It is not a cache

- Once populated it stays there (no LRU algorithms)



In-Memory Column Store Population
On demand or on startup

Population is done in background

```
  PID USER      VIRT    RES  S %CPU %MEM     TIME+ COMMAND
 2933 oracle 1419732  92376  R 97.0  9.1   0:09.72 ora_w001_DEMO
 7349 oracle 1413504 171016  S  1.0 16.8   0:11.31 oracleDEMO (LOCAL=NO)
 2143 oracle 1395876   9644  S  0.0  0.9   0:02.45 ora_pmon_DEMO
 2165 oracle 1403104 257476  S  0.0 25.3   0:13.00 ora_dbw0_DEMO
 2167 oracle 1396520   9720  S  0.0  1.0   3:25.30 ora_lgwr_DEMO
 2171 oracle 1397784  57776  S  0.0  5.7   0:01.15 ora_smon_DEMO
 2203 oracle 1409236  90868  S  0.0  8.9   0:05.72 ora_w000_DEMO
 2207 oracle 1402960 133628  S  0.0 13.1   0:11.61 ora_imco_DEMO
```

Population on demand (at first access):

- `INMEMORY PRIORITY NONE`

Population on startup (starting with most critical)

- `INMEMORY PRIORITY LOW/MEDIUM/HIGH/CRITICAL`

In-Memory Column Store Population
Population in RAC

By default, rows are DISTRIBUTE to all instances

- By partition (supposes equal distribution), by rowid

147
258
369

On Oracle engineered systems, can be DUPLICATE

- Ensures high availiability of IMCS in case of node failure
- Exadata or ODA X-5 (infiniband)
- No cache fusion: sends only messages to trigger repopulation

13467
124578
235689

In-Memory in RAC

- Makes sense only with Parallel Query and Auto DOP
- It is supposed to be service aware
  - Currently only for on demand population
  - on startup population probably in 12.2

13579
2468

In-Memory Column Store Population
Population

IM population needs CPU

- Reads blocks from disk or buffer cache
- Need to read rows in order -> follow chained rows

DML makes IM become stale, and IM scan is slower

- Trickle repopulates slowly and constantly
- Full repopulation is done when IMCU staleness reaches threshold

```
NAME                                       VALUE
------------------------------------------ -----
inmemory_max_populate_servers              2      default=CPU_COUNT/2
inmemory_trickle_repopulate_servers_percent 1
```

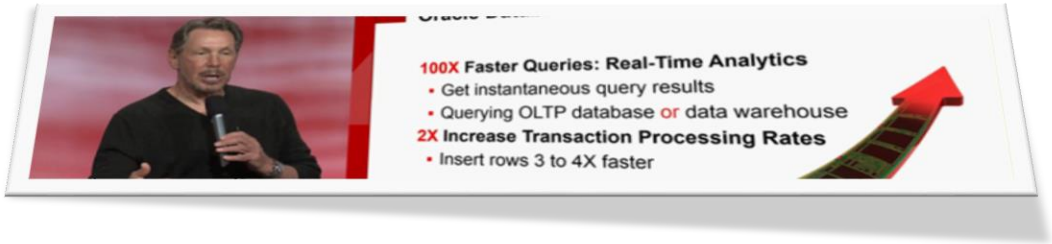
- Default probably too high. Do you want to:
  - give all resources to population in order to have IMCS quickly or
  - let it populate slowly in the background keeping CPU for non-IM activity?

In-Memory Column Store Population
And the DML ?

DML must update the column store

- In addition to the row store (buffer cache + redo logging)
- IMCS is updated asyncronously
  - Updates are going to transaction log

Isn't it an overhead for OLTP ?



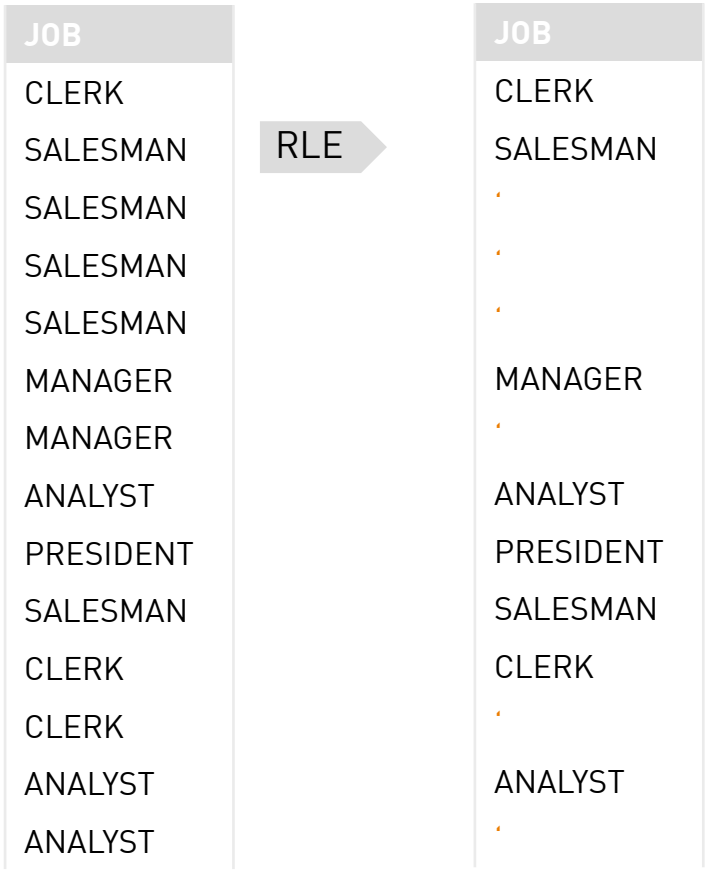
No, OLTP is faster with In-Memory:

- You can drop some indexes used only for analytics
- And maintaining IMCS is faster than maintaining 2 or 3 indexes
- But keep indexes on PK, FK,
  and those that you need before IMCS population is completed
- and have enough CPU for re-population

Agenda

1. Analytic queries
2. Row store and Column store
3. In-Memory population
4. Performance
5. How to use it

In-Memory performance
Columnar Compression



Run-Length Encoding (RLE)

- Columnar storage is the best suited for compression of repeated values (like HCC)
- CPU can work in compressed values
- Compression Unit are indexed (like Exadata Storage Index)

Columnar Storage offers efficient ways to lower the RAM to CPU transfer which is now the new bottleneck (as we have no disk I/O)

In-Memory performance
Columnar Compression

Compression Unit are compressed by default

- Filtering is done on compressed values.
- Decompression occurs only for returned values
- Compression occurs at population

Levels of compression

- NO MEMCOMPRESS – Compression disabled
- MEMCOMPRESS FOR DML
- **MEMCOMPRESS FOR QUERY LOW**
- MEMCOMPRESS FOR QUERY HIGH
- MEMCOMPRESS FOR CAPACITY LOW
- MEMCOMPRESS FOR CAPACITY HIGH – Save more space

In-Memory performance
Storage Index

Storage indexes allows IMCU pruning

- Check predicate against min/max
- Skip IMCU that doesn't match

Many predicate can be filtered

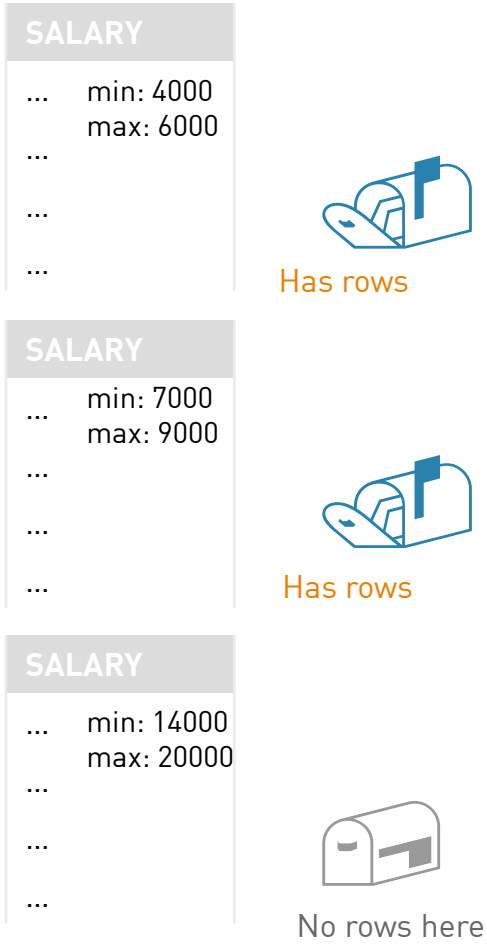
- Equality, range, in-list for example

Reduce again volume of data to process

- Eliminate unnecessary IMCUs

Example:

Find employees with salary between 5000 and 8000



In-Memory performance
Storage Index

Storage indexes can skip filtering

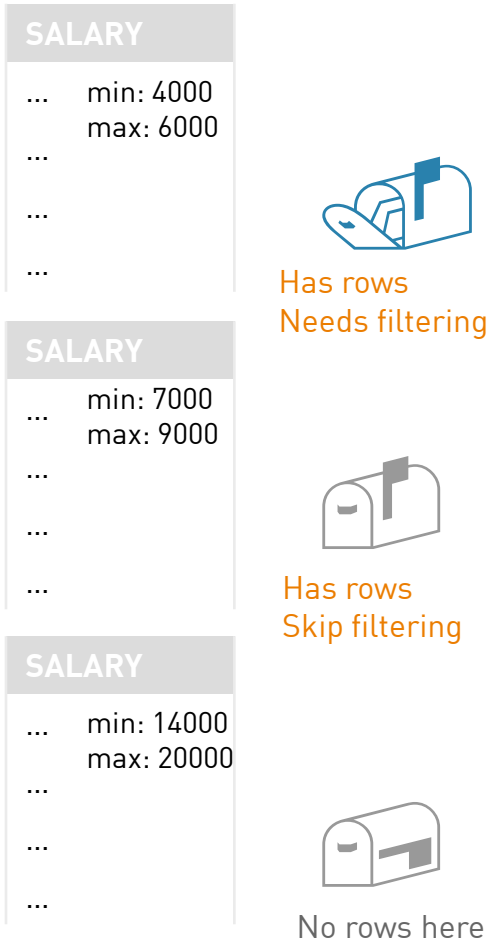
- Skipped IMCUs are not evaluated
- No need to evaluate filter when all rows match

Not only min/max

- Also hash of distinct values

Example:

Find employees with salary between 5000 and 9000



In-Memory performance
Joins ?

In-Memory enables Bloom Filter on serial statements

- Join optimization introduced in Oracle 10g for Parallel Query

Operation	Name
SELECT STATEMENT	
HASH JOIN	
JOIN FILTER CREATE	:BF0000
MERGE JOIN CARTESIAN	
MERGE JOIN CARTESIAN	
TABLE ACCESS INMEMORY FULL	DIM2
BUFFER SORT	
TABLE ACCESS INMEMORY FULL	DIM3
BUFFER SORT	
TABLE ACCESS INMEMORY FULL	DIM1
JOIN FILTER USE	:BF0000
TABLE ACCESS INMEMORY FULL	FACT

1. HASH JOIN reads the first input to build the hash table

2. Bloom Filter :BF0000 is created (filters but allows false positive)

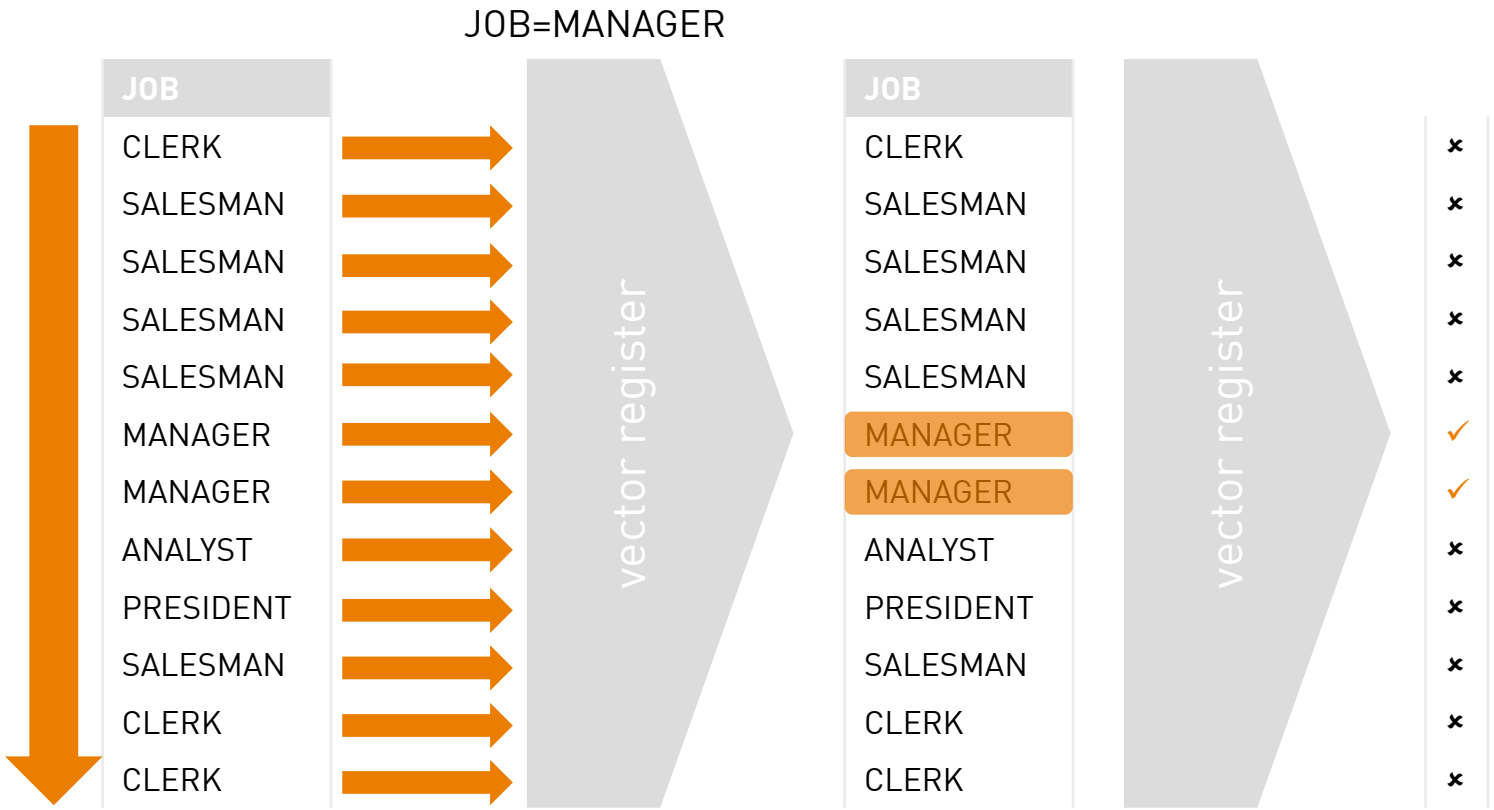
3. Bloom filter used to filter while reading second input

```
11 - inmemory(SYS_OP_BLOOM_FILTER(:BF0000
       filter(SYS_OP_BLOOM_FILTER(:BF0000
```

In-Memory performance
Vector processing

Columnar format allows sending vector to CPU

- SIMD vector used to compare multiple values in one CPU cycle



Agenda

1. Analytic queries
2. Row store and Column store
3. In-Memory population
4. Performance
5. How to use it

How to use In-Memory
Basics

One parameter to enable In-Memory

- Pool is static, database restart is required

```
alter system set INMEMORY_SIZE=7G scope=spfile;
shutdown immediate
startup
```

  - Don't forget to increase SGA_TARGET

How to put table in memory?

```
alter table SH.SALES INMEMORY;
```

- Exclude some columns:

```
alter table SH.SALES INMEMORY NO INMEMORY(tax_region);
```

- Choose compression

```
alter table SH.SALES INMEMORY MEMCOMPRESS FOR CAPACITY LOW;
```

- Choose population

```
alter table SH.SALES INMEMORY PRIORITY CRITICAL;
```

How to use In-Memory
Monitor In-Memory

Check In-Memory usage

```
SQL> select * from V$INMEMORY_AREA;

POOL      ALLOC_BYTES USED_BYTES POPULATE_STATUS CON_ID
--------- ----------- ---------- --------------- ----------
1MB POOL   5149556736   11534336 DONE                     0
64KB POOL  1275068416          0 DONE                     0
```

Check segment stored in Colum Store

- When not enough memory Oracle just stop loading objects

```
SQL> select owner, segment_name name, inmemory_size,
populate_status pop_status, bytes, bytes_not_populated
from V$IM_SEGMENTS

OWNER  NAME  INMEMORY_SIZE POP_STATUS BYTES       BYTES_NOT_POPULATED
------ ----- ------------- ---------- ----------- -------------------
SH     SALES    5858787328 COMPLETED   18990759936        2151686144
```

In-Memory advisor

- Need Diagnostic Pack + Tuning Pack (SQL Performance Analyzer)

How to use In-Memory
Demo: Population and Query

Without In-Memory

- Bitmap indexes

ALTER TABLE INMEMORY

- New execution plan: TABLE ACCESS INMEMORY FULL
- Population triggered at first query
- Not optimal until fully populated

Once populated

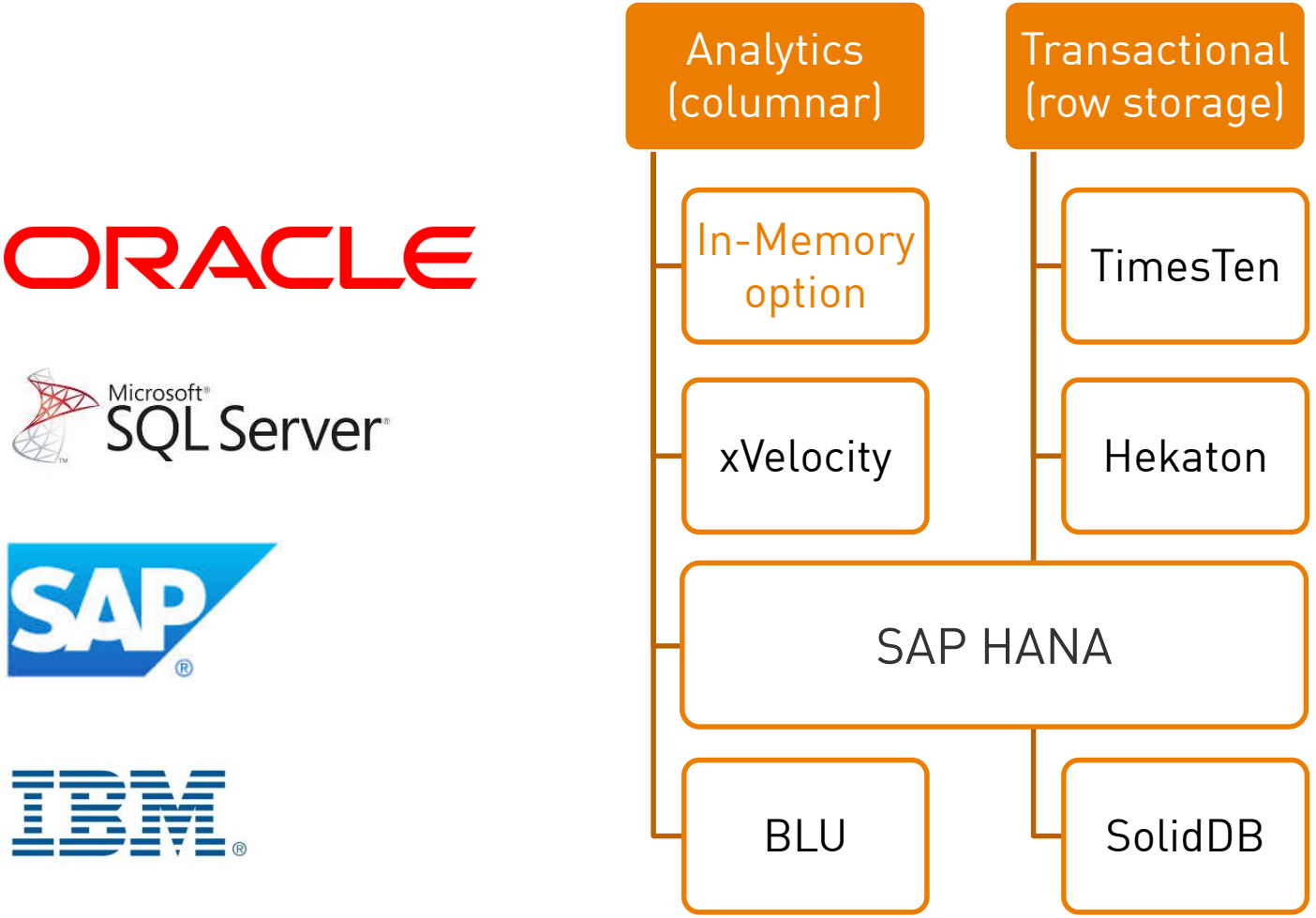
- Very fast access for all queries

ALTER TABLE NO INMEMORY

- Only row store available

- Take care of partially populated table: full table scan through buffer cache

Oracle In-Memory Column Store
RDBMS In-Memory Technologies competitors



Oracle In-Memory Column Store
Core Message

In-Memory goal

- Execute analytic query in operational database or in BI database

Oracle implementation strengths

- Doesn't require any change
- Works on any 12.1.0.2 supported architecture
- Is very efficient to filter data without any additional index

Oracle implementation caveats

- Performance depends on Column Store population (CPU intensive)
  - Do you want to drop your indexes?

It's an option on Enterprise Edition (+50% on public price list)

Any questions? Please do ask.

Franck Pachot
Senior consultant
Oracle Technology Leader

Mobile +41 79 963 27 22
franck.pachot@dbi-services.com
www.dbi-services.com

We look forward to working with you!