Exploring Interesting New Features in Oracle Database 12.1.0.2

Zoran Pavlović, Oracle Technical Architect, Parallel
Maja Veselica, Security Consultant, Parallel

Contact

Zoran Pavlović, *Oracle Technical Architect*
z.m.pavlovic@gmail.com

Twitter: ChallengeZoran

Maja Veselica, *Security Consultant*
maja.veselica@gmail.com

Twitter: orapassion

Our Websites

www.challengezoran.com – forum
www.orapassion.com – blog
www.parallel.rs - Company

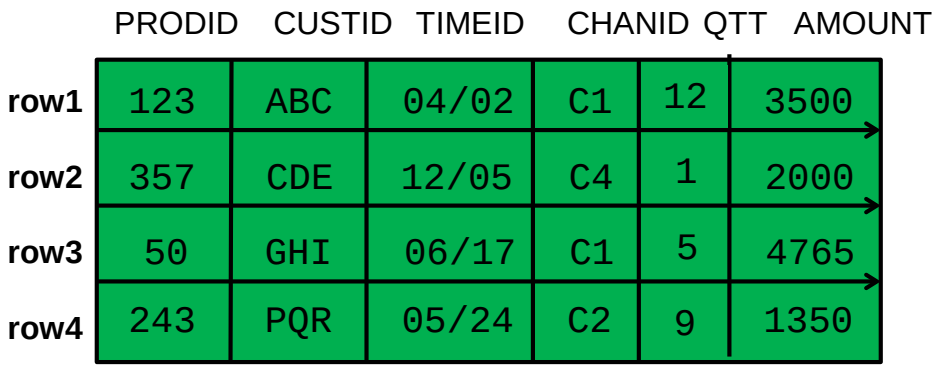
In-Memory Introduction

- In-Memory column store – a new pool in SGA
- Segments are populated into the In-Memory Column Store, and during that process converted to columnar format.
- In-Memory segments are CONSISTENT with the buffer cache
- There is NO columnar format on DISK! All data is still stored on Disk in the row format.

In-Memory Introduction

What is performance benefit of using columnar store instead of traditional row store?

SALES table (aprox 400 million rows)



ROW format

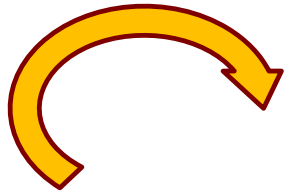
Calculate total amount sold in first quarter of 2012 through channel C4.

- Oracle needs to scan every single row in order to find ones that satisfy this query. Each time Oracle access the row, it reads all columns in that row.

SALES table

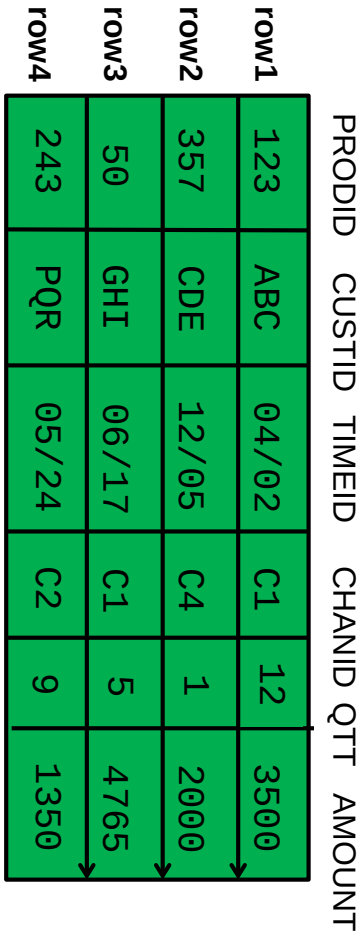
	PRODID	CUSTID	TIMEID	CHANID	QTT	AMOUNT
row1	123	ABC	04/02	C1	12	3500
row2	357	CDE	12/05	C4	1	2000
row3	50	GHI	06/17	C1	5	4765
row4	243	PQR	05/24	C2	9	1350

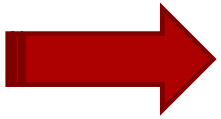
ROW format

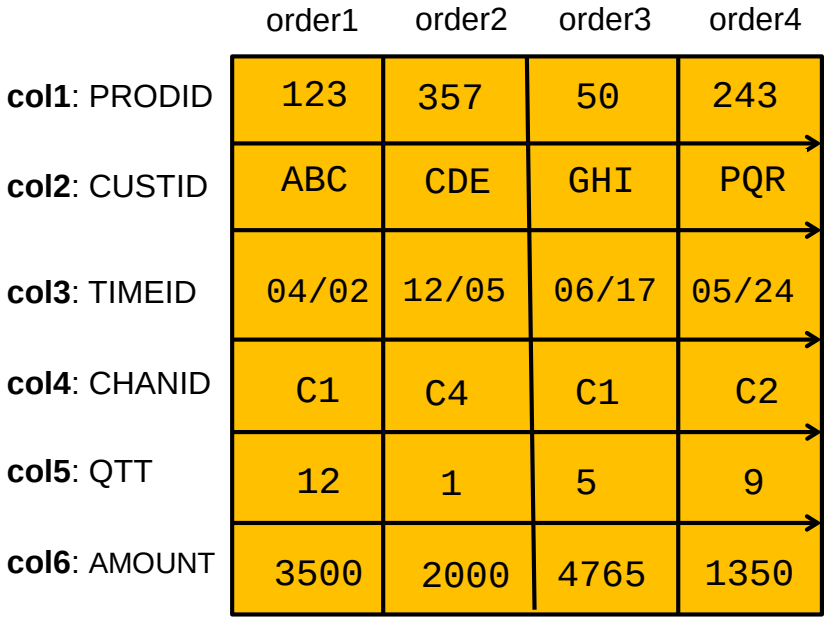


	row1	row2	row3	row4
PRODID	123	357	50	243
CUSTID	ABC	CDE	GHI	PQR
TIMEID	04/02	12/05	06/17	05/24
CHANID	C1	C4	C1	C2
QTT	12	1	5	9
AMOUNT	3500	2000	4765	1350

SALES table







Column store format

In-memory Storage Index

When query arrives, Oracle will compare value(s) specified in where clause to the values of minimum and maximum in IMCU storage index. This way, Oracle will scan only those IMCUs that can contain entries that will satisfy the query.

In-memory Database and Indexes

Until now, the best practice for analytic queries in an OLTP database is to create specific indexes on frequently accessed columns. However, maintenance of these indexes is very expensive and they need additional space.

With In-Memory Database, only indexes needed for referential integrity is needed. More that 70% of the indexes of in-memory table can be droped!

Enabling In-Memory Database

New static initialization parameter:

inmemory_size

- default 0 – in-memory database disabled
- min 100M

```
SQL> ALTER SYSTEM SET INMEMORY_SIZE=32G scope=spfile;
```

Populating Segments in In-Memory Column Store

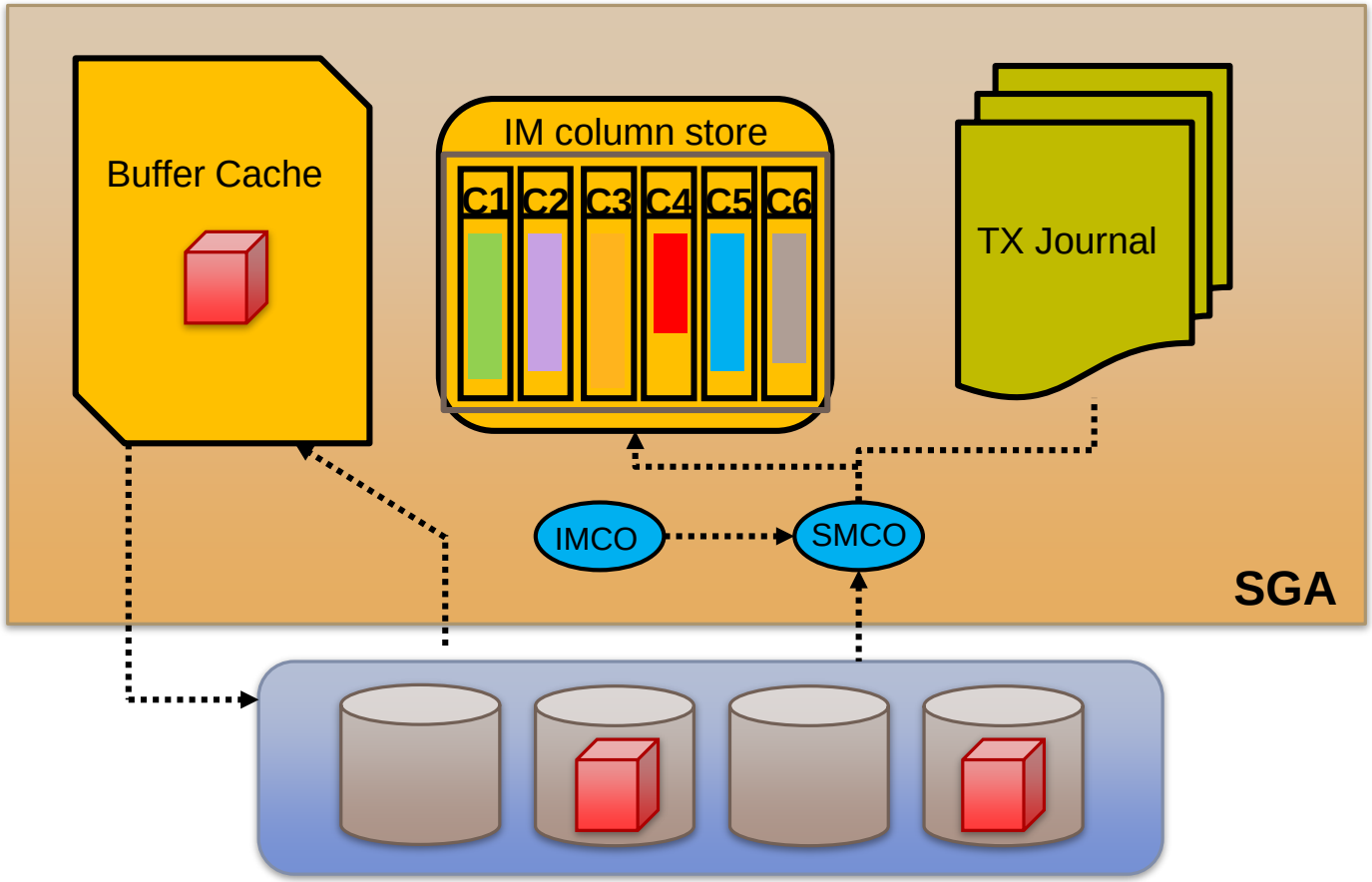
```
SQL> ALTER TABLE ssb.lineorder INMEMORY;
```

Default Priority level: NONE, This segment will be populated next time it is queried. If we want to populate it immediately, we need to specify one of the priority levels:

- CRITICAL
- HIGH
- MEDIUM
- LOW

```
SQL> ALTER TABLE ssb.lineorder INMEMORY PRIORITY HIGH;
```

Segments in dual format



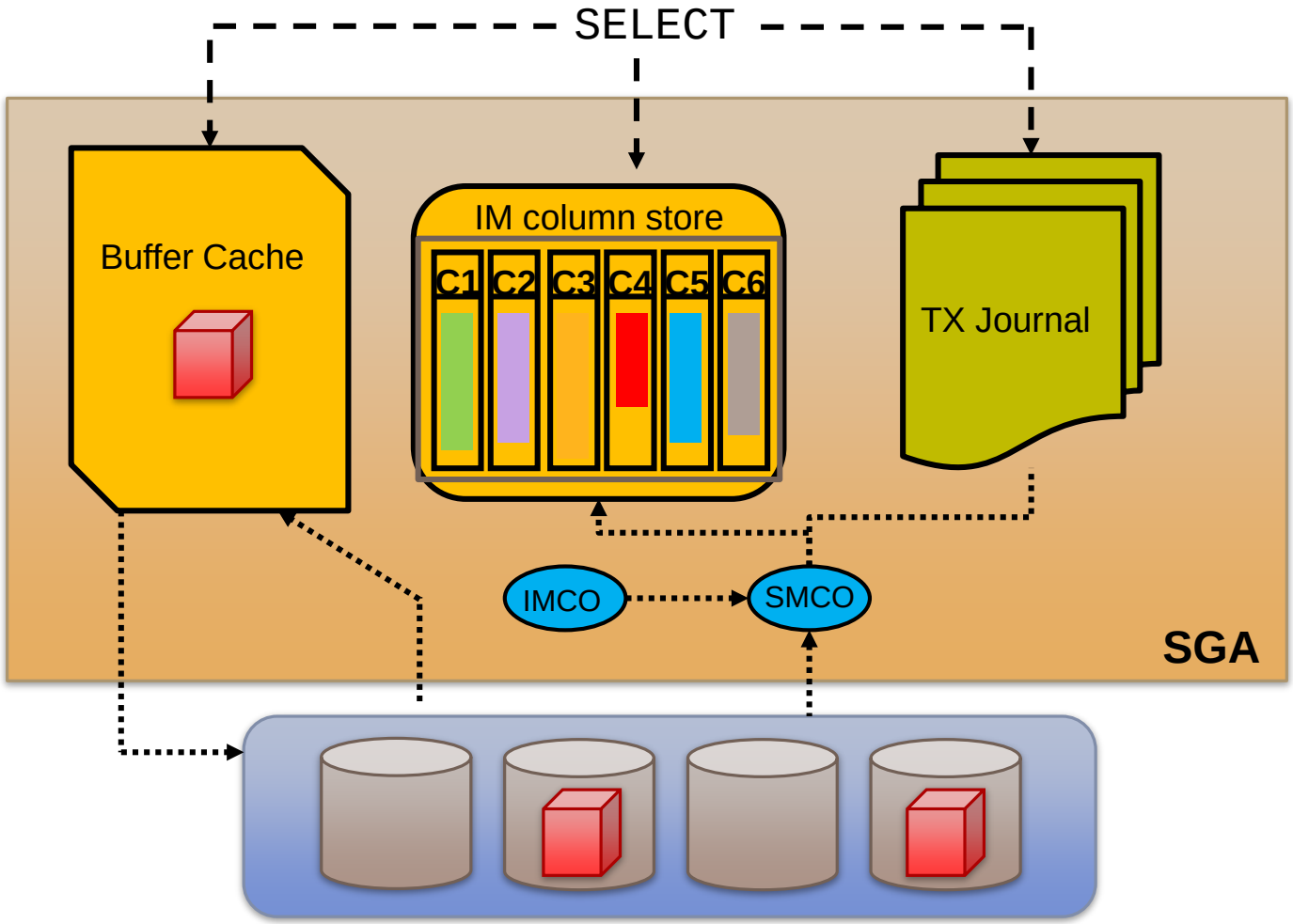
Segments in dual format

Optimizer chooses whether to use segment stored in buffer cache or in in-memory column store.

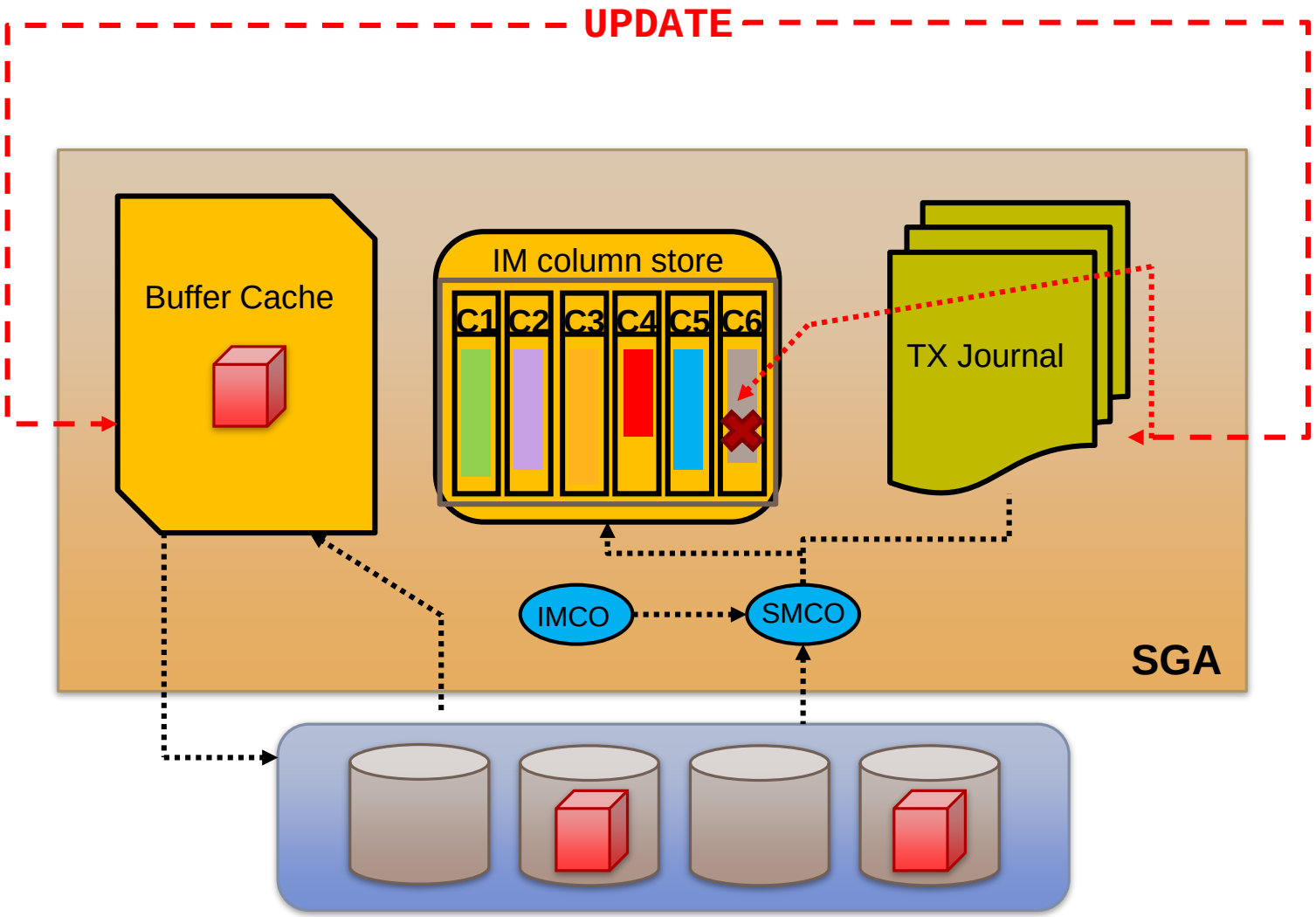
New Initialization parameter:

optimizer_inmemory_aware = TRUE (default)

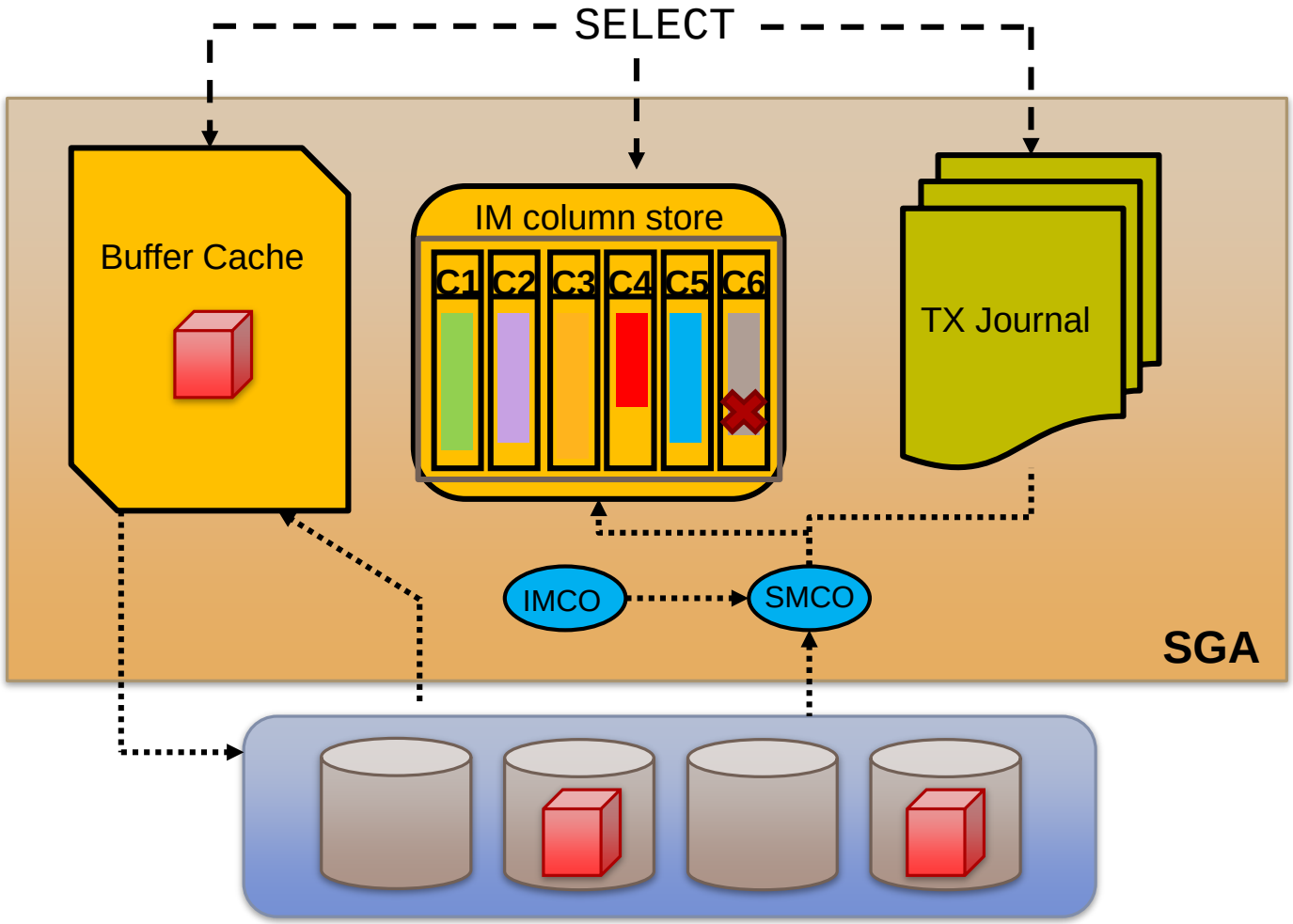
Segments in dual format



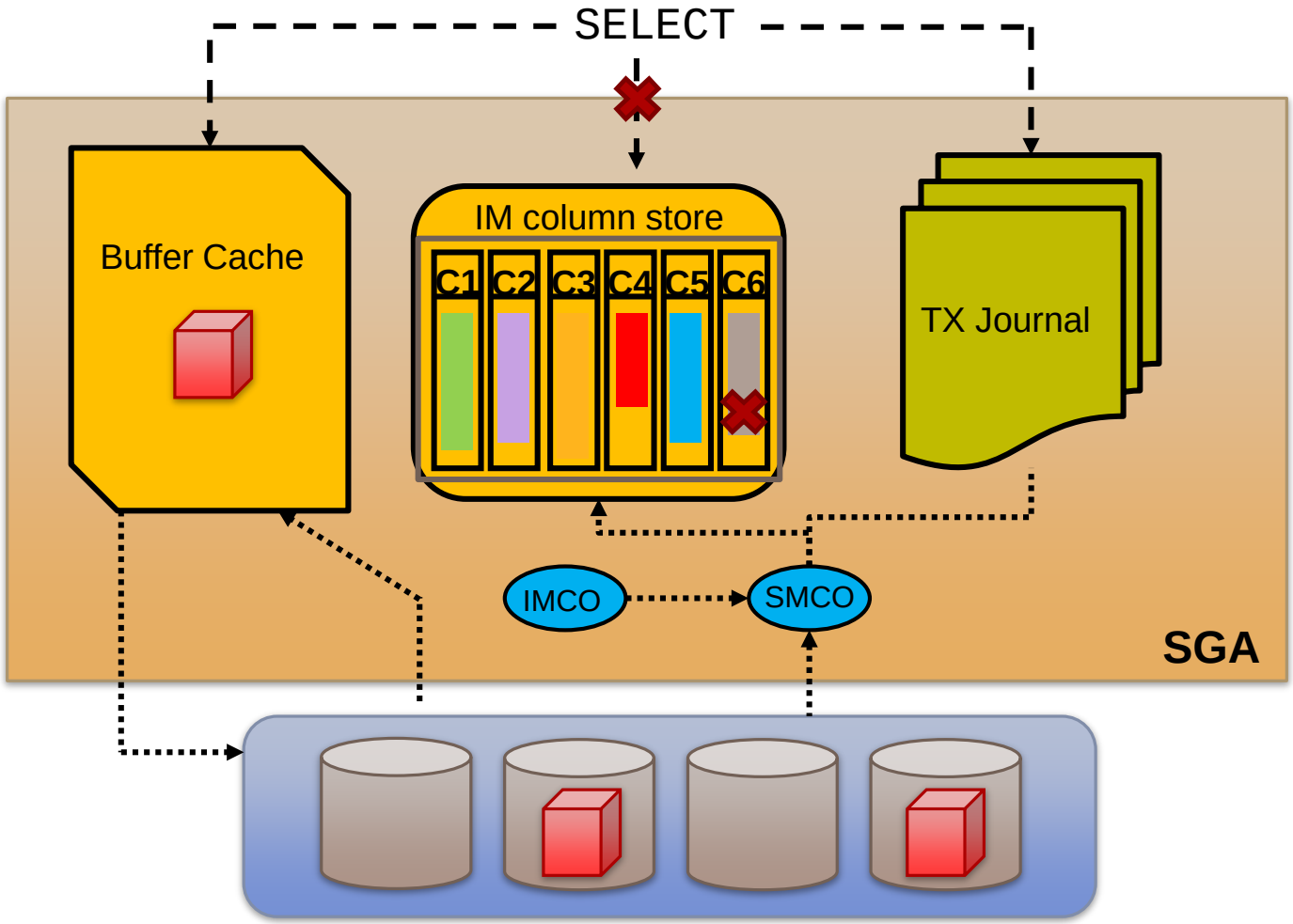
Update


UPDATE
Buffer Cache
IM column store
C1
C2
C3
C4
C5
C6
TX Journal
IMCO
SMCO
SGA

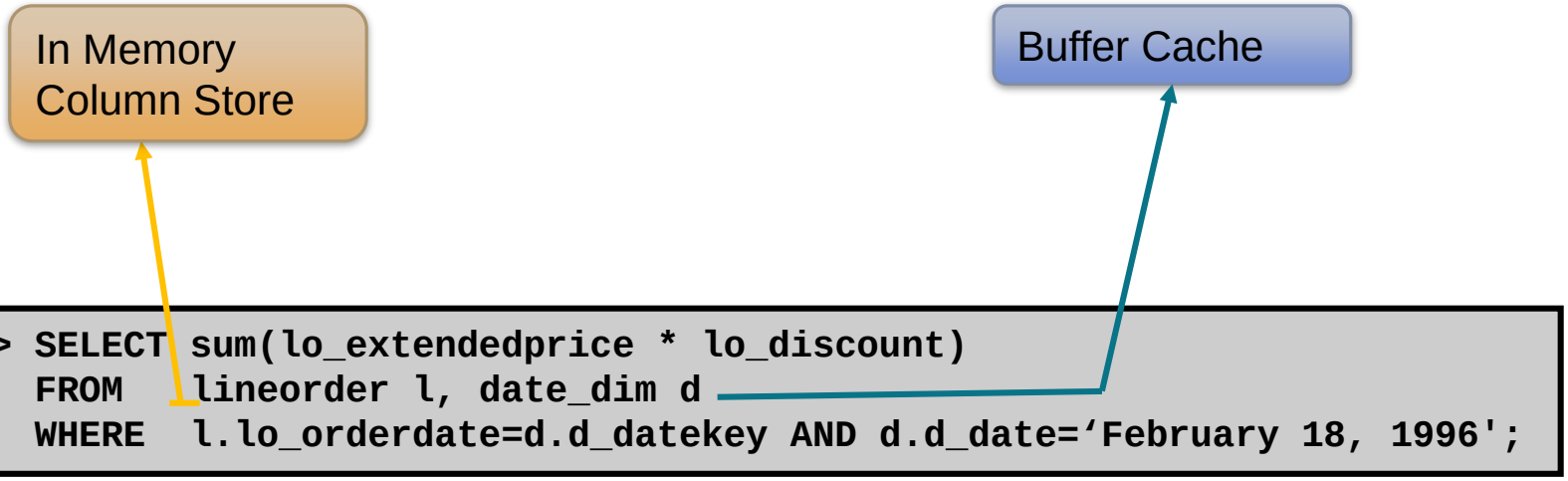
Select on non-stale objects



Select on stale objects



Select on In-Memory and non In-Memory Tables




```
-------------------------------------------------------------------
|Id  | Operation                                | Name              |
-------------------------------------------------------------------
|   0 | SELECT STATEMENT                        |                   |
|*  1 |  HASH JOIN                              |                   |
|   2 |   JOIN FILTER CREATE                    | :BF0000           |
|*  3 |    TABLE ACCESS FULL                    |  DATE_DIM         |
|   4 |     JOIN FILTER USE                     | :BF0000           |
|*  5 |      TABLE ACCESS INMEMORY FULL         |  LINEORDER        |
-------------------------------------------------------------------
```

Partitioning with in-memory

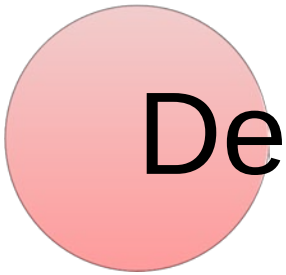
Divide and conquer principle

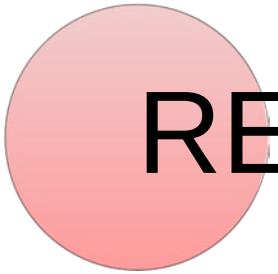
Partitions can have different in-memory attributes

```
SQL> CREATE TABLE SALES …
PARTITION BY RANGE …
  (PARTITION SALES_P1 …
   INMEMORY MEMCOMPRESS FOR DML
   PARTITION SALES_P2 …
   INMEMORY MEMCOMPRESS FOR QUERY LOW
   .
   .
   PARTITION SALES_p150 …
   INMEMORY MEMCOMPRESS FOR CAPACITY HIGH
   .
   .
   PARTITION SALES_P250 …
   NO INMEMORY);
```

In-memory on RAC environment

- Every node in cluster has it's own In-memory column store
- Objects can be distributed across RAC nodes automatically (AUTO), BY ROWID RANGE or BY PARTITION
- On Exadata and Super-Cluster, objects can be duplicated with following statements:
  - DUPLICATE – make 2 copies of object (on 2 nodes)
  - DUPLICATE ALL – duplicate object on all RAC nodes


Demo



READ Privilege

READ Privilege

In Oracle 12.1.0.2 in addition to SELECT privilege, there is a new READ privilege

Why having two privileges for the same purpose?

READ Privilege

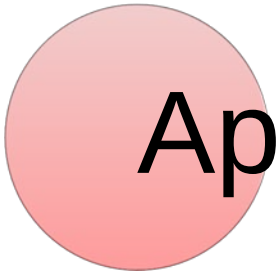
```
zoran@DBM1> create user seltest identified by oracle;
zoran@DBM1> grant create session to seltest;
zoran@DBM1> grant select on Zoran.tbl to seltest;
```

```
seltest@DBM1> LOCK TABLE zoran.tbl IN EXCLUSIVE MODE;

Table(s) Locked.

seltest@DBM1> SELECT * FROM zoran.tbl FOR UPDATE;

         A
----------
         1
         4
         2
```

READ Privilege

```
zoran@DBM1> create user readtest identified by oracle;
zoran@DBM1> grant create session to readtest;
zoran@DBM1> grant read on zoran.tbl to readtest;
```

```
readtest@DBM1> LOCK TABLE zoran.tbl IN EXCLUSIVE MODE;
LOCK TABLE zoran.tbl IN EXCLUSIVE MODE
                  *
ERROR at line 1:
ORA-01031: insufficient privileges


readtest@DBM1> SELECT * FROM zoran.tbl FOR UPDATE;
SELECT * FROM zoran.tbl FOR UPDATE
                    *
ERROR at line 1:
ORA-01031: insufficient privileges
```



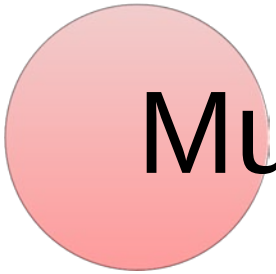
Approximate count distinct

Approximate count distinct

In situations when we don’t need to get exact number (for instance number of visitors of our website, or number of customers for specific product).

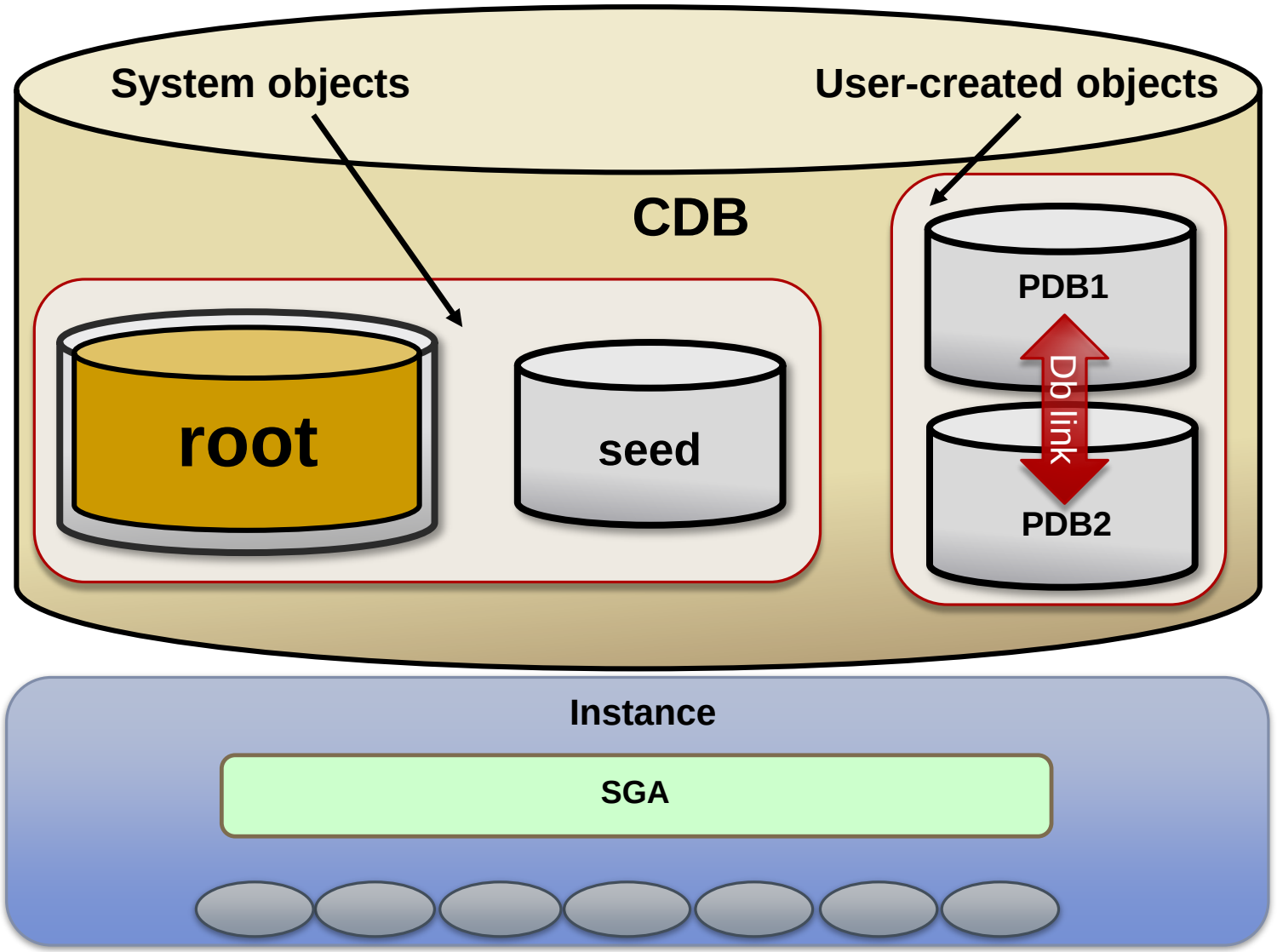
It represents an alternative to *COUNT(DISTINCT(something)*, which returns exact number. On contrary, approximate count distinct returns approximate number, but significantly faster.

```
SQL> SELECT prod_id, APPROX_COUNT_DISTINCT(cust_id)
  FROM sales;
```



Multitenant new features

Oracle Multitenant Architecture



Things that exist on CDB level (same for all PDBs)

- *Instance*
- *Control files, Redo logs and Archive logs*
- *System metadata (system dictionary)*
- *System data*
- *UNDO tablespace*
- *Default TEMP tablespace*
- *Common users and roles*
- *Encryption wallet*
- *SPFILE / PFILE*
- *Root SYSTEM and SYSAUX Tablespaces*

Things that exist on PDB level

- *User metadata (user dictionary)*
- *User data*
- *Local TEMP tablespace*
- *Local users and roles*
- *Parameters that have is_pdbmodifiable = true*
- *Encryption master key*
- *PDB SYSTEM and SYSAUX Tablespaces*

Multitenant New features in 12.1.0.2

- New STANDBYS clause for creation of PDBs in Data Guard environment
- New CONTAINERS clause to query user data across multiple PDBs from root.
- Saving PDB state across restarts
- PDB Remote Clone
- PDB Metadata Clone
- PDB Subset Clone
- Logging Clause at PDB level
- FDA Support for PDBs

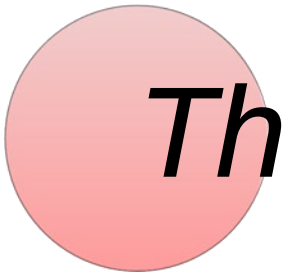
CONTAINERS Clause

It enables queries issued in root container to access data from multiple PDBs in the same CDB.

```
c##zoran@PDB1> CREATE OR REPLACE VIEW emp AS SELECT * FROM
SCOTT.EMP;
```

```
c##zoran@PDB2> CREATE OR REPLACE VIEW emp AS SELECT * FROM
SCOTT.EMP;
```

```
c##zoran@CDB1> SELECT * FROM CONTAINERS(emp) WHERE CON_ID
IN(3,4);
```



Thank you!