Adatbázisok elmélete

Egy kísérlet az OLTP és az analitikus jellegű adatbázishasználat kombinálására: Oracle Database12c In-Memory

Kerepes Tamás
Webváltó kft.
tamas.kerepes@webvalto.hu

Relációs adatbázisok felhasználási területei

- Üzleti adatokat manapság leginkább adatbázisokban tárolunk
- Ha már adatbázisok, akkor az elmúlt kb. 30 évben a relációs adatbázisok dominálnak
- Az ilyen rendszereket kétféle környezetben használják:
  - OLTP rendszerek
  - Adattárházak, döntéstámogató rendszerek, analitikus jellegű rendszerek
- A valóság sohasem ilyen vegytiszta, hanem e kettő keveréke a gyakori
- A vegyes jellegű használat mindegyik ma használt rendszernél igen komoly kihívásokat jelent

Miért probléma a „vegyes használat”?

- Az OLTP rendszerekben a tranzakciókezelés van fókuszban, a konzisztencia, sok „kicsi” felhasználó egyidejű munkájának a hatékony elvégzése, a megbízható és flexibilis zárolások (pl. row level locking). Rengeteg kicsi tranzakció „nyüzsög” egymás mellett és érint kevés adatot.
- Az adattárházakban kevés felhasználó igen nagy adatmennyiséget kérdez le, vagy esetleg transzformál. A kihívás itt a lekéredzések párhuzamosítása, a hardver erőforrások minél jobb kihasználása, akár csak egy lekérdezés számára is. Kritikus a minél jobb végrehajtási terv.
- A két „kaszt” annyira eltérő, hogy mondhatni: ellehetetlenítik egymást.

Különbségek az Oracle adatbázisoknál OLTP rendszerek és adattárházak között

- **Normalizáltság:**
  - **OLTP: normalizált adatok**
  - **Adattárházak: csillag (star) vagy hópehely (snowflake) modell**
- **Indexek:**
  - **OLTP: B* fa szerkezetű indexek minden elsődleges és egyedi kulcs oszlopon, szinte minden külső kulcs oszlopon és azokon az oszlopokon, amelyek szerint gyakori a szűrés vagy összekapcsolás**
  - **Adattárházak: bittérképes (bitmap) indexek, néha bitmap join indexek**
- **Származtatott értékek:**
  - **OLTP: hacsak lehet, kerüljük a használatukat**
  - **Adattárház: gyakori az összesített értékek tárolása materializált nézetek formájában**

A klasszikus megoldás erre a problémára

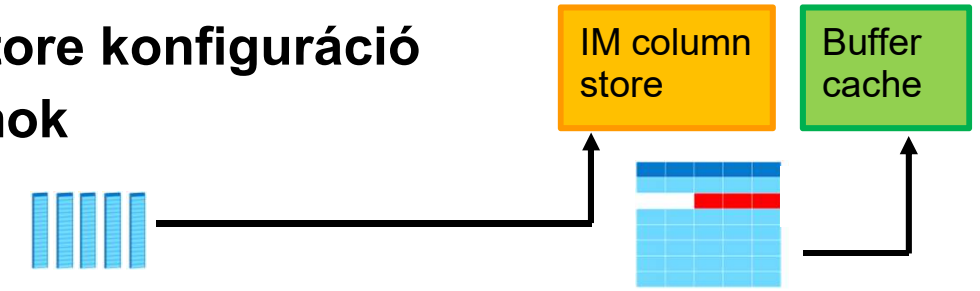
- Külön OLTP adatbázisok és külön adattárház jellegű adatbázisok, noha mindkettő relációs
- Egyes gyártóknál ez külön termék is lehet: specializált adatbáziskezelő rendszer pl. adattárházak céljaira: célszoftver. Ilyen pl. a Terradata
- Másoknál ugyanazt a szoftvert használják mindkét területen, de másként konfigurálják őket. Ilyen pl. az Oracle RDBMS
- Folyamatos adat-áttöltések az OLTP adatbázisból az adattárházba: ETL folyamatok (Extraction, Transformation, Loading)
- Komoly probléma, hogy az adatelemzések nem valós idejű adatokkal történnek

Az Oracle megoldási kísérlete:Oracle Database12c: In-Memory

- Előszóban: az In-Memory opció egy olyan lehetősége az Oracle12c adatbáziskezelő rendszernek, amelyben a táblák adatait az SGA-n belül egy új gyorsítóban, az In-memory Column Store-ban is tároljuk.
- Ott oszloponként és nem soronként.
- Mivel emellett még igen hatékonyan tömörítődnek is az így tárolt adatok, jelentősen lerövidül a lekérdezések válaszideje

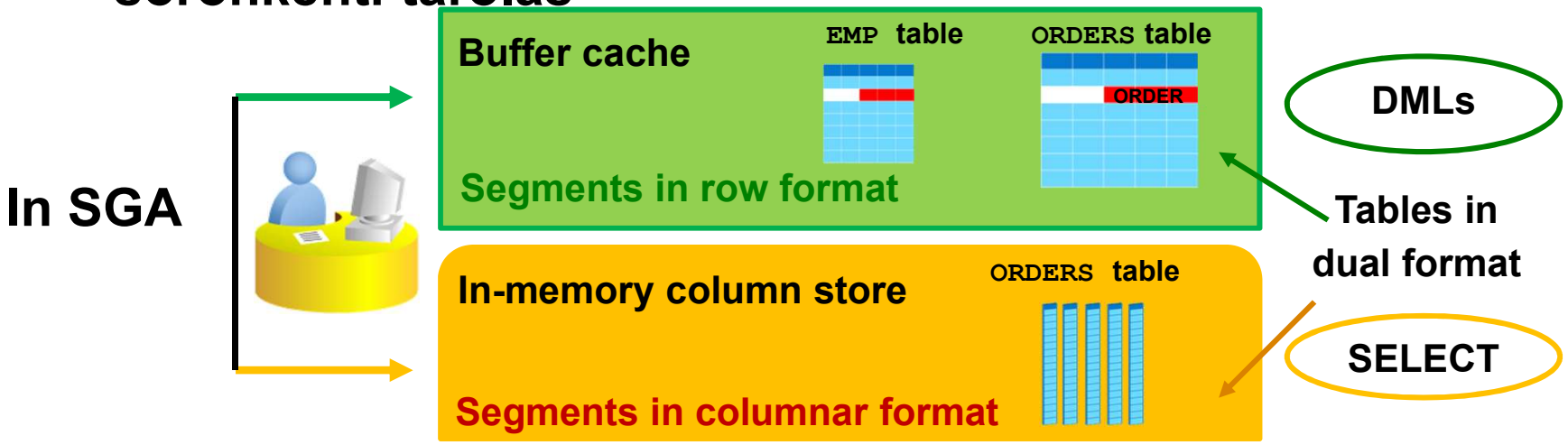
Az In-Memory Column Store célkitűzései

- „Szinte azonnal válaszidő a lekérdezésekre:
  - A nagyon nagy táblák esetén a lekérdezések sokkal gyorsabbal (100x)
  - A „full scan”, a „Join” és az aggregált adatok kiszámítása válnak gyorssá
  - A lekérdezéseknek nem kell az index
  - Analitikus feldolgozásokra ideális:kevés oszlop,sok sor
- Gyorsabb DML: mivel kevesebb az index (3-4x)
- Teljes mértékű transparencia az alkalmazásoknak
- Könnyű setup:
  - In-memory column store konfiguráció
  - Szegmens attribútumok



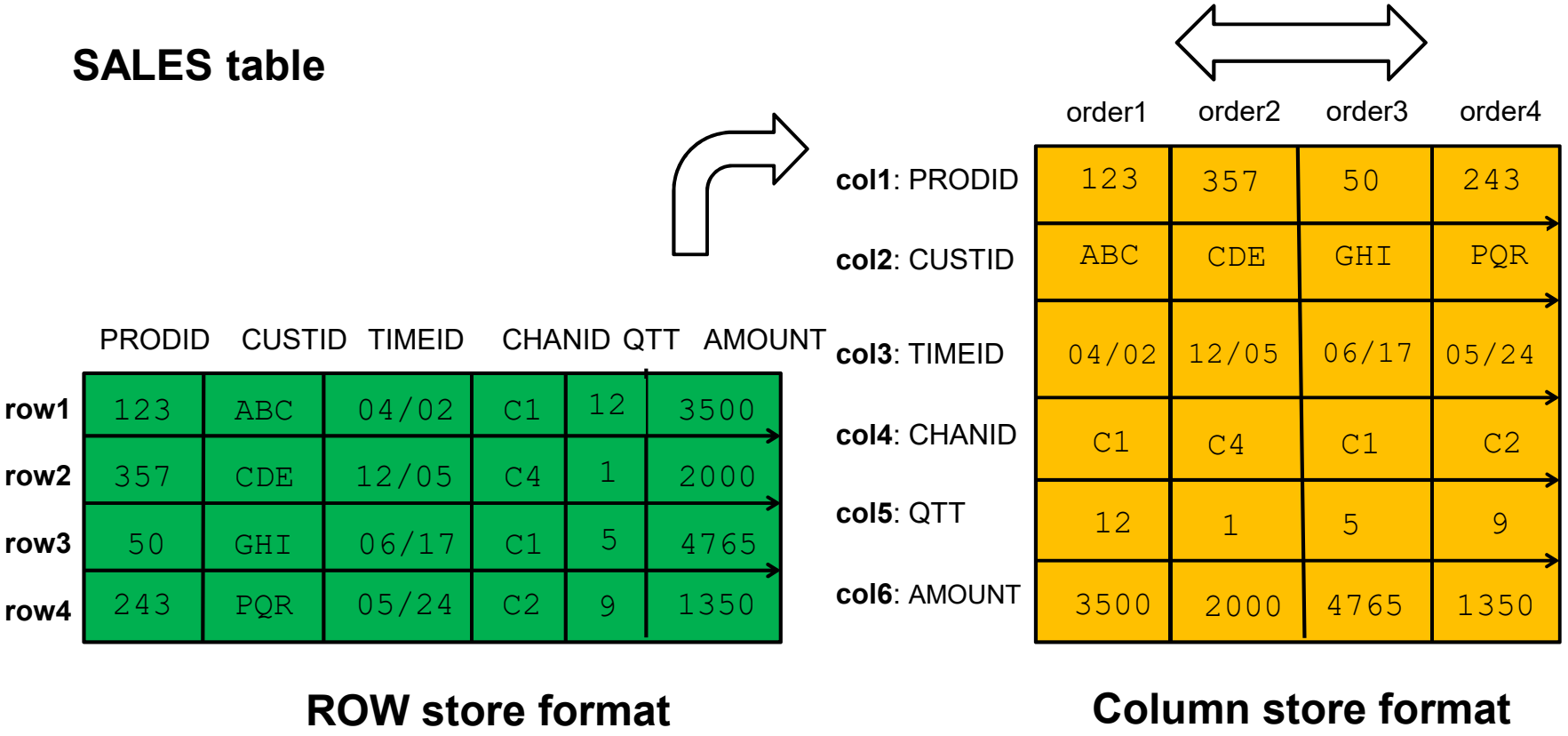
Egy kis architektúrai áttekintés

- Új memóriazóna az SGA-ban: In-Memory column store
  - Azok a szegmensek, amelyek ide is bekerülnek, oszloponkénti formátumba konvertálódnak az IM Column Storeban
  - Az In-Memory szegmensek értékei tranzakcionálisan konzisztensek a buffer cache tartalmával
- A merevlemezen továbbra is csak egy formátum: soronkénti tárolás



Soronkénti /oszloponkénti tárolás: két dimenzióban mutatva

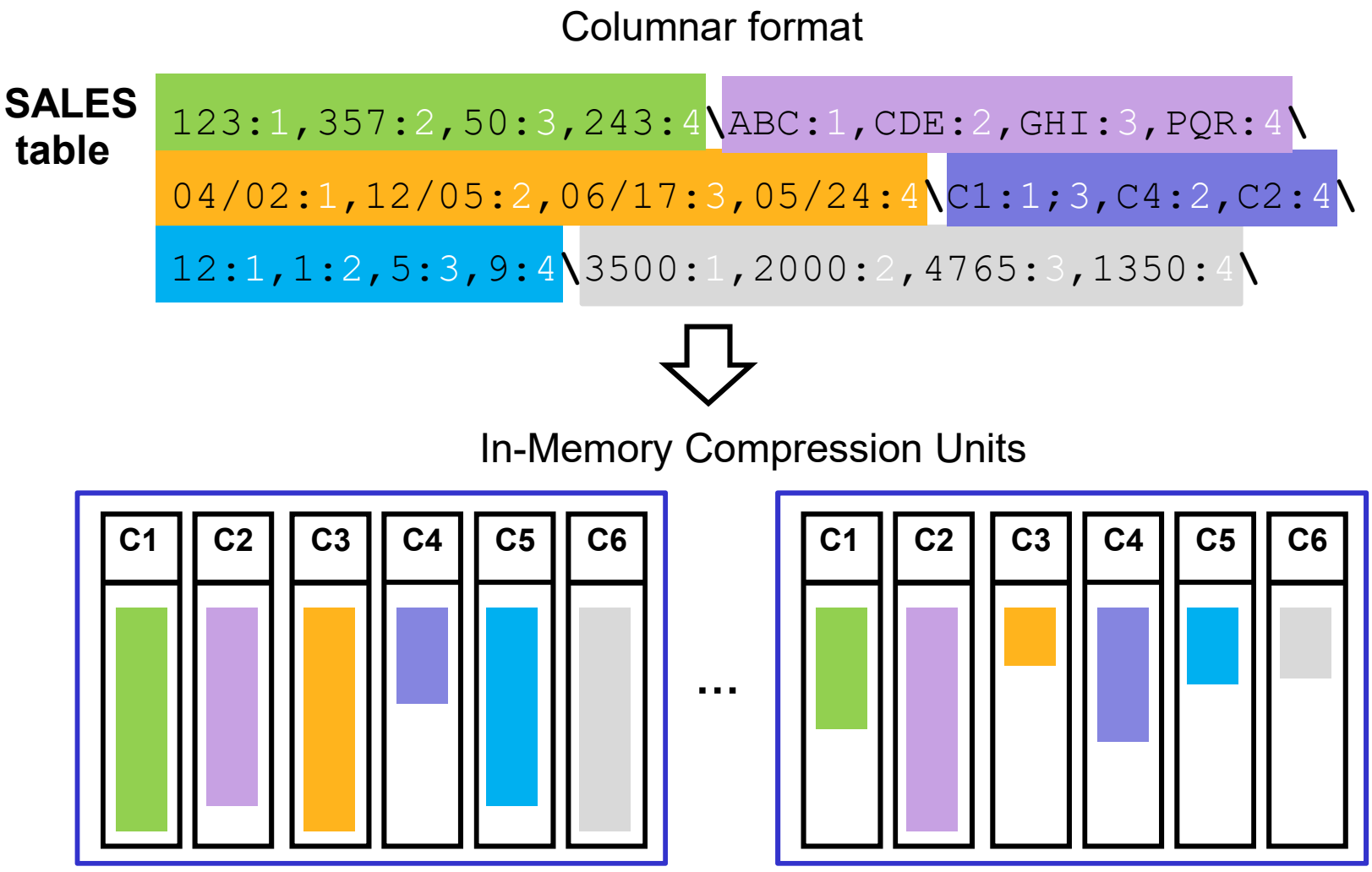
SALES table



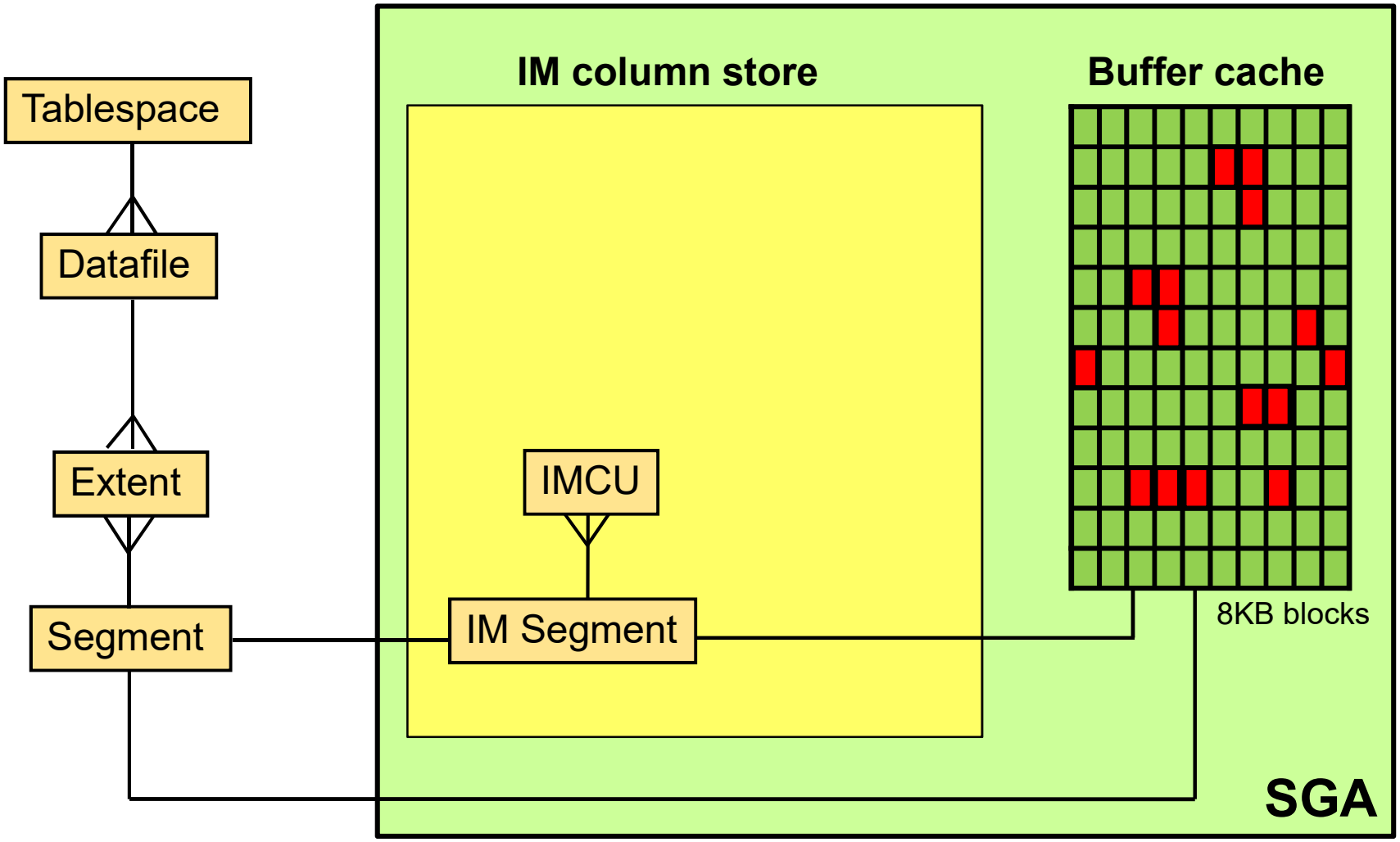
ROW store format

Column store format

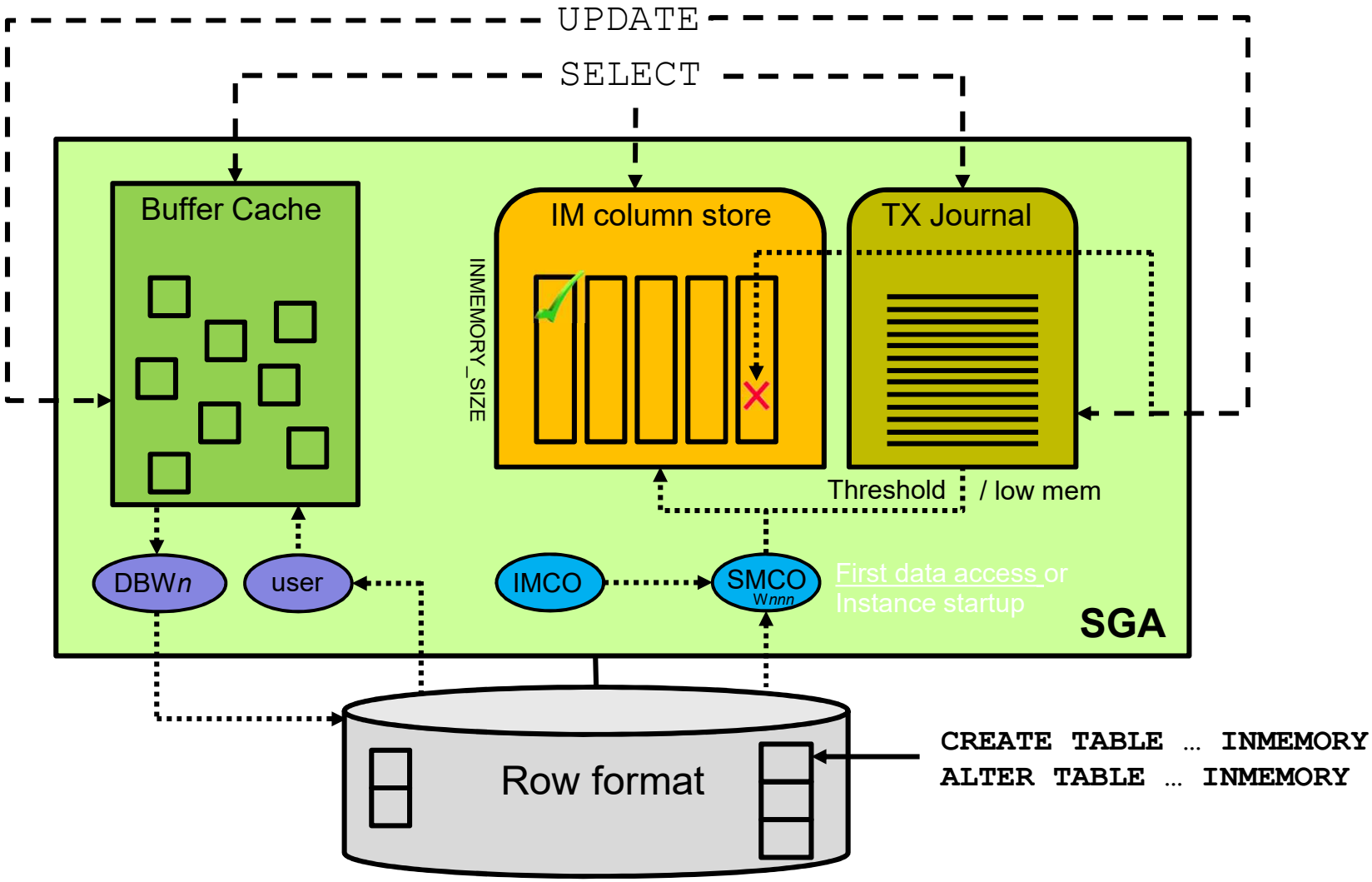
IMCU: In-Memory Compression Unit



In-Memory Column Store Cache illetve Database Buffer Cache



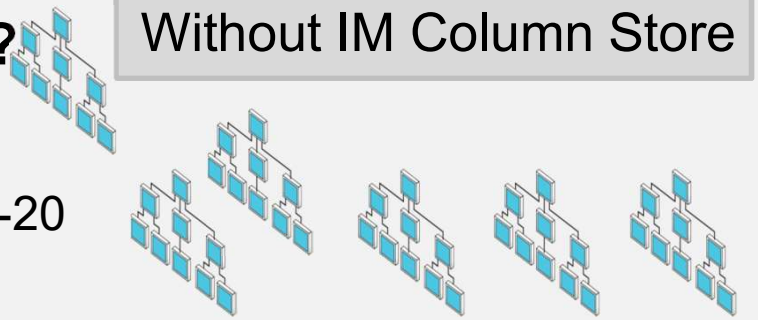
Dual Format In Memory



Indexek

- **Mely oszlopokat indexeljük?**

OLTP-ben: 1 - 3
Analitikus rendszerekben:10-20



- **Lassú lekérdezés ott, ahol nincs index**
- **Nehéz/lassú karbantartás a DML-ek során**
- **Nagy tárterület szükséges**

With IM Column Store

- Nincsenek többé analitikus célú indexek az in-memory táblákon
- Nemcsak az előre definiált, hanem az ad-hoc analitikus lekérdezések is gyorsak
- OLTP & batch összességében akár háromszor gyorsabb

Az IM Column Store technológia bevezetése (I)

1. **Az adatbáziskezelő rendszer kompatibilitási verziója**

```
COMPATIBLE = 12.1.0.0.0
```

2. **Az IM column store méretének a konfigurálása**

```
INMEMORY_SIZE = 100G
```

Az IM Column Store technológia bevezetése (2): objektumok beállításai

3. Lehetővé tenni az objektumok betöltődését:
   - Egy egész szegmens szintjén:

     Az IMCU-k jellemzően a lekérdezések végrehajtásakor inicializálódnak és töltődnek fel.

```
SQL> CREATE TABLE large_tab (c1 …) INMEMORY;
```
→ Dual format

```
SQL> ALTER TABLE t1 INMEMORY ;
```
→ Dual format

```
SQL> ALTER TABLE sales NO INMEMORY;
```
→ Row format only

```
SQL> CREATE TABLE countries_part ... PARTITION BY LIST ..
      ( PARTITION p1 .. INMEMORY, PARTITION p2 .. NO INMEMORY);
```

IMCU-k inicializálódhatnak adatbázis megnyitáskor is

```
SQL> CREATE TABLE test (…) INMEMORY PRIORITY CRITICAL;
```

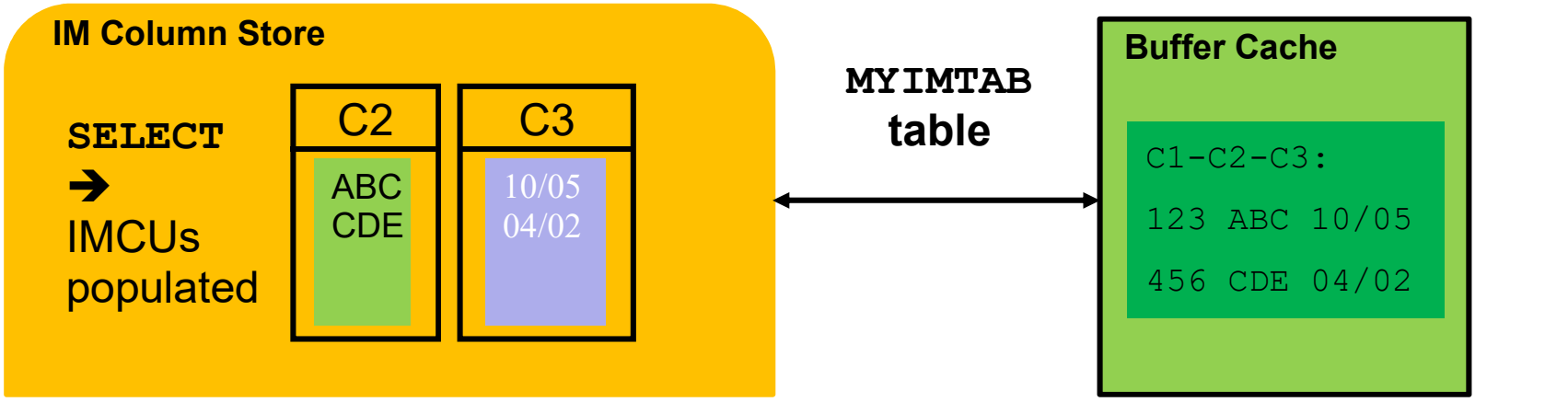
Deploying IM Column Store: Columns Setting

- Enable/disable a subset of columns:

```
SQL> CREATE TABLE myimtab (c1 NUMBER, c2 CHAR(2), c3 DATE)
                    INMEMORY NO INMEMORY (c1);
```

```
SQL> ALTER TABLE myimtab NO INMEMORY (c2);
```

```
SQL> CREATE TABLE t (c1 NUMBER, c2 CHAR(2)) NO INMEMORY INMEMORY (c2);
ORA-64361: column INMEMORY clause may only be specified for an inmemory
           table
```



Mely objektumok lehetnek az IM Column Store-ban

A DBA_TABLES nézet ezt is mutatja:

```
SQL> SELECT table_name tab, inmemory_compression Comp,
       inmemory_priority Priority,
       inmemory_distribute RAC, inmemory_duplicate DUP
     FROM  dba_tables;
```

```
TABLE  COMP                PRIORITY   RAC           DUP
------ ----------------- ---------- ------------ --------------
TEST1  FOR QUERY HIGH     NONE       AUTO         DUPLICATE ALL
TEST2  NO MEMCOMPRESS     CRITICAL   AUTO         DUPLICATE
EMP    FOR DML            LOW        BY PARTITION NO DUPLICATE
```

- **NO MEMCOMPRESS**: Not compressed
- **FOR QUERY / FOR CAPACITY**: Compressed

- **NONE**: Populated on demand
- Other values: Populated according to priority level

- **AUTO**: Object distributed by Oracle among the IM column stores of the RAC nodes
- **BY PARTITION**: The object is distributed by partitions among the RAC nodes

- DUPLICATE ALL
- DUPLICATE
- NO DUPLICATE

Mely oszlopok lehetnek az IM Column Store-ban

A V$IM_COLUMN_LEVEL nézet mutatja ezt:

```
SQL> SELECT obj_num, segment_column_id, inmemory_compression
     FROM   v$im_column_level;
```

In-memory columns overriding compression clause of the parent table

```
   obj_num segment_co      _id inmemory_compression
---------- ------------------- ----------------
     98202                   1 DEFAULT
     98202                   2 NO MEMCOMPRESS
     98202                   3 DEFAULT
     98202                   4 NO INMEMORY
```

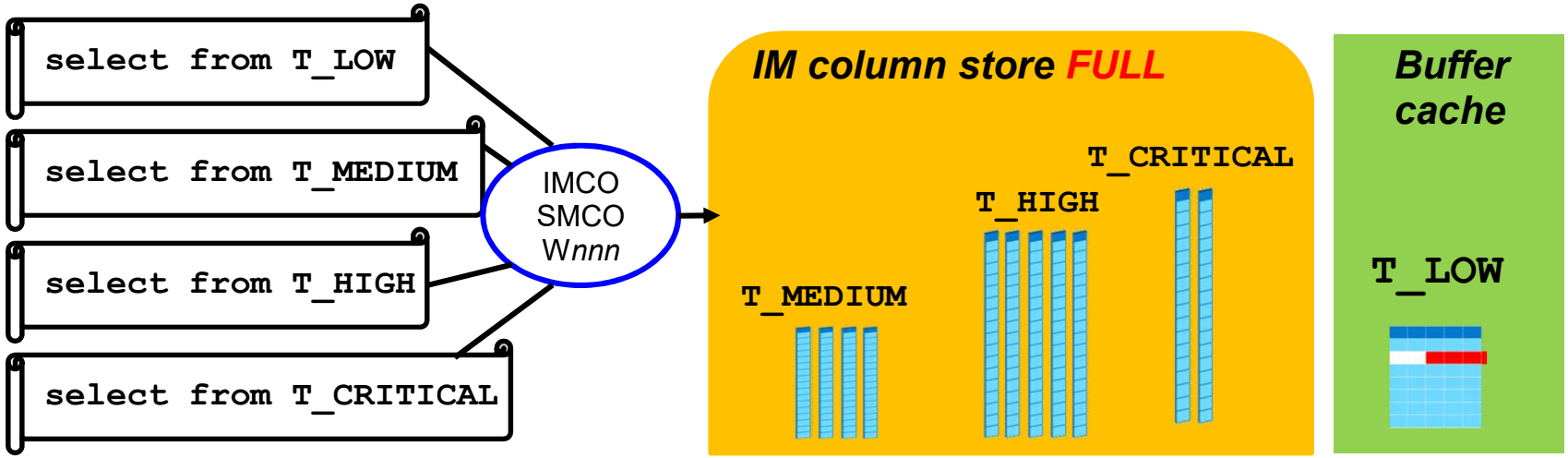
In-memory columns of in-memory tables inheriting compression clause of the parent table

Non in-memory columns in in-memory tables

IM Column Store prioritások

A PRIORITY szabályozza, hogy kik kerüljenek be:

```
SQL> CREATE TABLE t_low (code NUMBER) INMEMORY PRIORITY LOW;
```




```
SQL> CREATE TABLE countries_part ... PARTITION BY LIST ..
      ( PARTITION p1 .. INMEMORY PRIORITY HIGH,
        PARTITION p2 .. INMEMORY MEMCOMPRESS FOR CAPACITY LOW,
        PARTITION p3 .. , .. );
```

Mely szegmensek kerültek be:

- **Először még üres az IM column store:**

```
SQL> SELECT owner, segment_name name FROM v$im_segments
     WHERE  segment_name = 'SALES';
```

Priority NONE ➔
No rows populated yet

```
SQL> SELECT /*+ full(s) noparallel (s)*/ count(*) FROM sales s;
```

- **A fenti lekérdezés után:**

```
SQL> SELECT segment_name, bytes, inmemory_size,
            bytes_not_populated
     FROM   v$im_segments;

SEGMENT_NAME         BYTES   INMEMORY_SIZE BYTES_NOT_POPULATED
------------ ------------ --------------- ------------------
SALES              228231           36299                  0
```

BYTES

BYTES in IM column store

IM Column Store tömörítés

- A MEMCOMPRESS szabályozza a tömörítési szintet

```
SQL> CREATE MATERIALIZED VIEW mv1 INMEMORY NO MEMCOMPRESS;
```

```
SQL> CREATE TABLE large_tab (c1 …) INMEMORY
                                   MEMCOMPRESS FOR QUERY;
```

- A merevlemezen több helyet foglal el:

- Magasabb tömörítési szint:

```
SQL> ALTER TABLE t1 INMEMORY MEMCOMPRESS FOR CAPACITY HIGH;
```

```
SEGMENT_NAME         BYTES   INMEMORY_SIZE
------------ ------------ --------------
T1             11475615744     4664262656
```

No automatic repopulation

IM Column Store Compression Advisor

Egyik „Advisor” a sokból. Azt elemzi, hogy mennyi memóriára van szükségünk a különböző MEMCOMRESS értékek esetén-

```
BEGIN
   DBMS_COMPRESSION.GET_COMPRESSION_RATIO (
      'TS_DATA', 'SSB', 'LINEORDER', NULL,
       DBMS_COMPRESSION.COMP_INMEMORY_QUERY_LOW,
      blkcnt_cmp, blkcnt_uncmp, row_cmp, row_uncmp, cmp_ratio,
      comptype_str,10000,1);
 DBMS_OUTPUT.PUT_LINE('Block count uncompressed = ' || blkcnt_uncmp);
 DBMS_OUTPUT.PUT_LINE('Row count per block uncompressed=' || row_uncmp);
 DBMS_OUTPUT.PUT_LINE('Compression type = ' || comptype_str);
 DBMS_OUTPUT.PUT_LINE('Comp ratio= '|| cmp_ratio );
 DBMS_OUTPUT.PUT_LINE('    ');
 DBMS_OUTPUT.PUT_LINE('The IM column store space needed is about 1 byte
in IM column store for '|| cmp_ratio  ||' bytes on disk.');
END;
/
```

A tömörítési arány kiszámolása

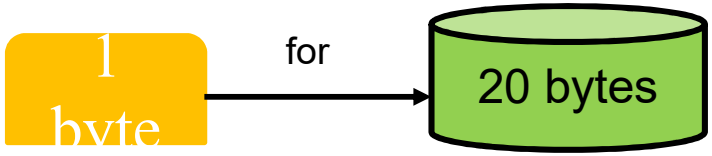
A V$IM_SEGMENTS nézetből nyerhető ki:

```
SQL> SELECT segment_name, bytes Disk, inmemory_size,
            inmemory_compression COMPRESSION,
            bytes / inmemory_size comp_ratio
     FROM   v$im_segments;

OBJECT_NAME       DISK INMEMORY_SIZE COMPRESSION         COMP_RATIO
----------- --------- ------------- ------------------ ----------
SALES       212284915     200104115 FOR QUERY LOW            1.06
EMPLOYEES     3456956         17984 FOR CAPACITY HIGH        192.22
```

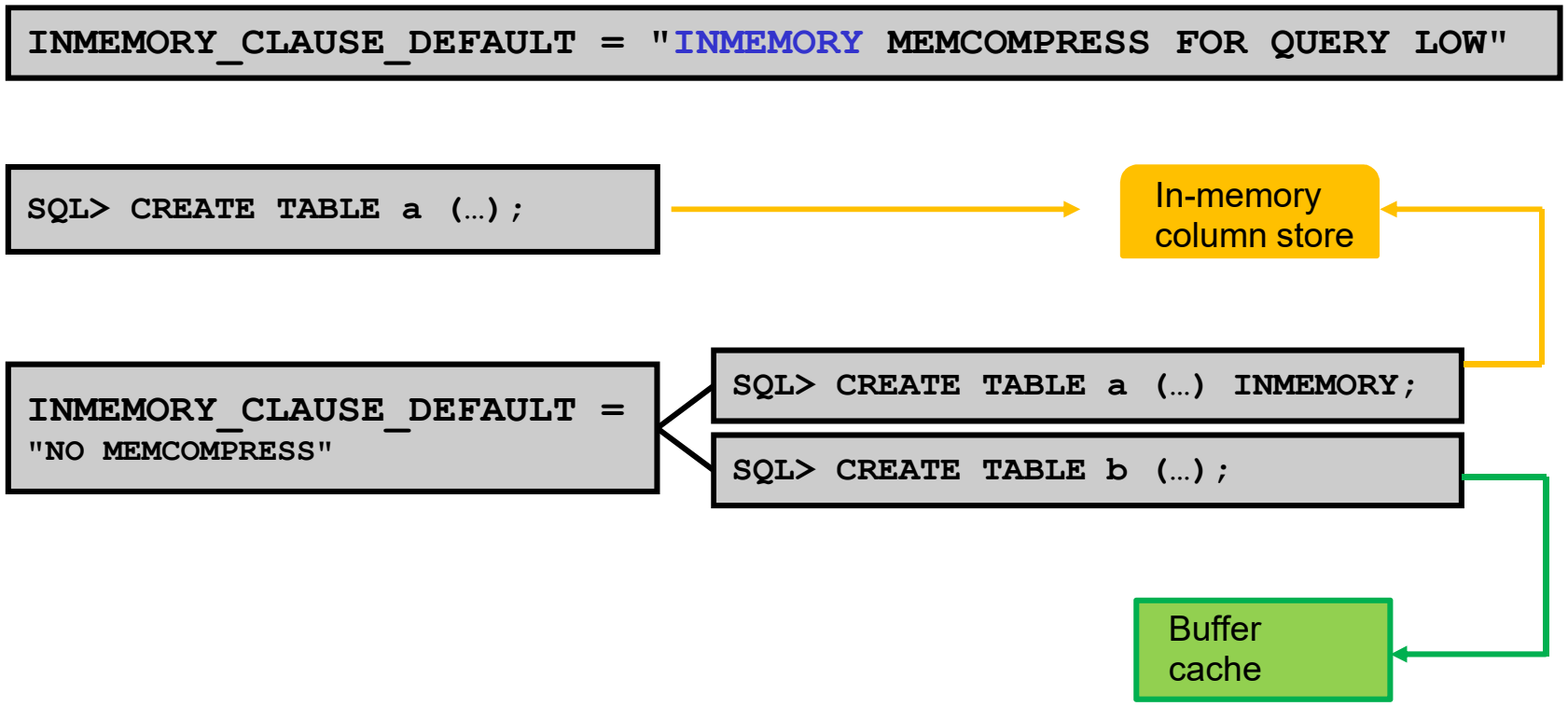
The highest, the best: 20

The lowest, the worse: 1.06

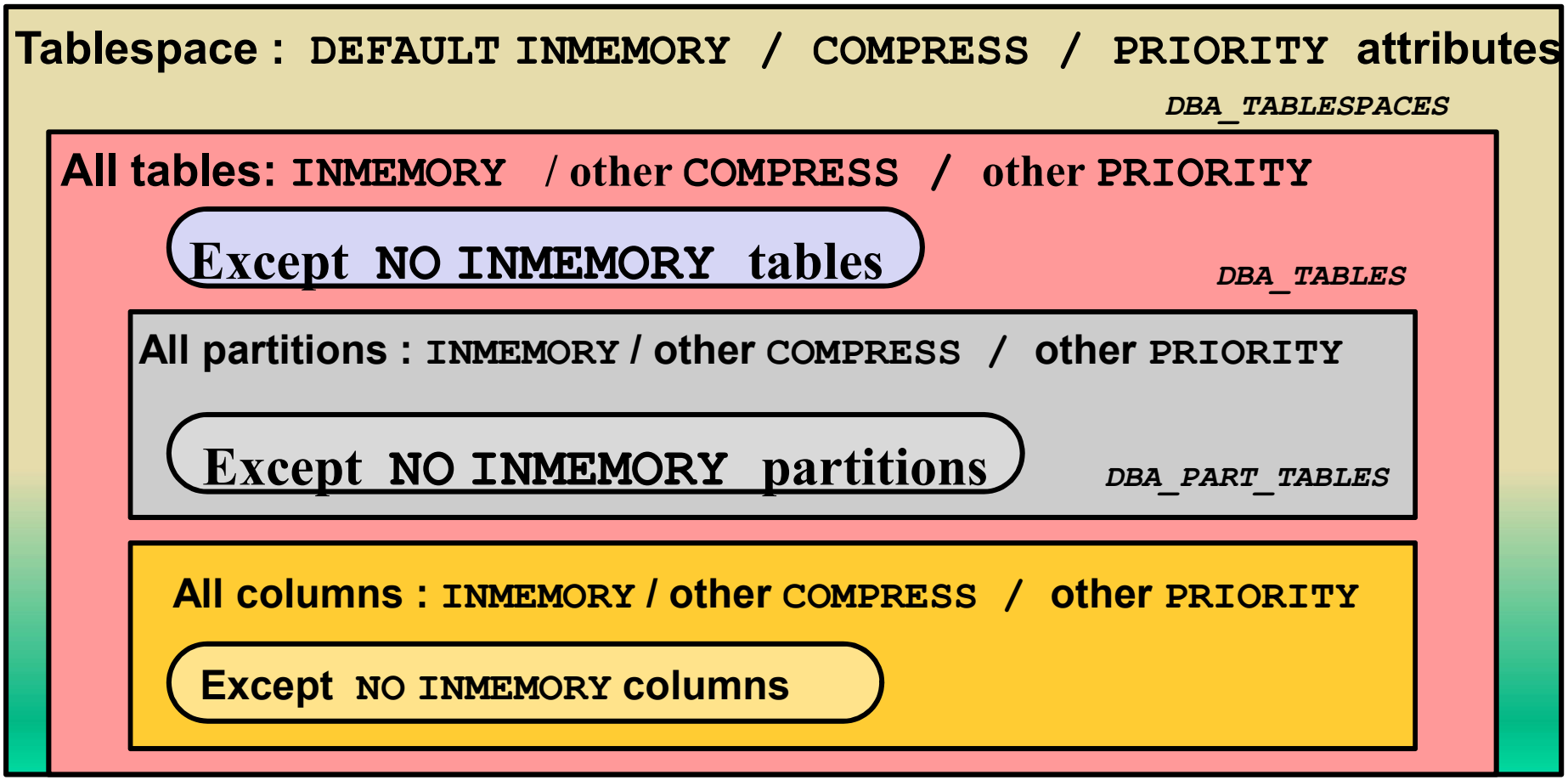


Default In-Memory beállítások

Az új táblák milyen attribútumokkal születnek:



INMEMORY öröklődési szisztéma




```
SQL> CREATE TABLESPACE IMCS … DEFAULT INMEMORY
            MEMCOMPRESS FOR CAPACITY HIGH PRIORITY LOW;
```

Tennivalók a bevezetés után

Eldobni az analitikus indexeket

```
SQL> DROP INDEX i_sales_timeid PURGE;
```

```
SQL> DROP INDEX i_sales_chanid PURGE;
```

Kipróbáljuk a lekérdezések felgyorsulását

- **Ha az IM column store-ból fut:**

```
SQL> SET timing ON
SQL> SELECT max(lo_ordtotalprice) most_expensive_order
     FROM   lineorder;
```

Elapsed: 00:**00:01.80**

Transparent for the application

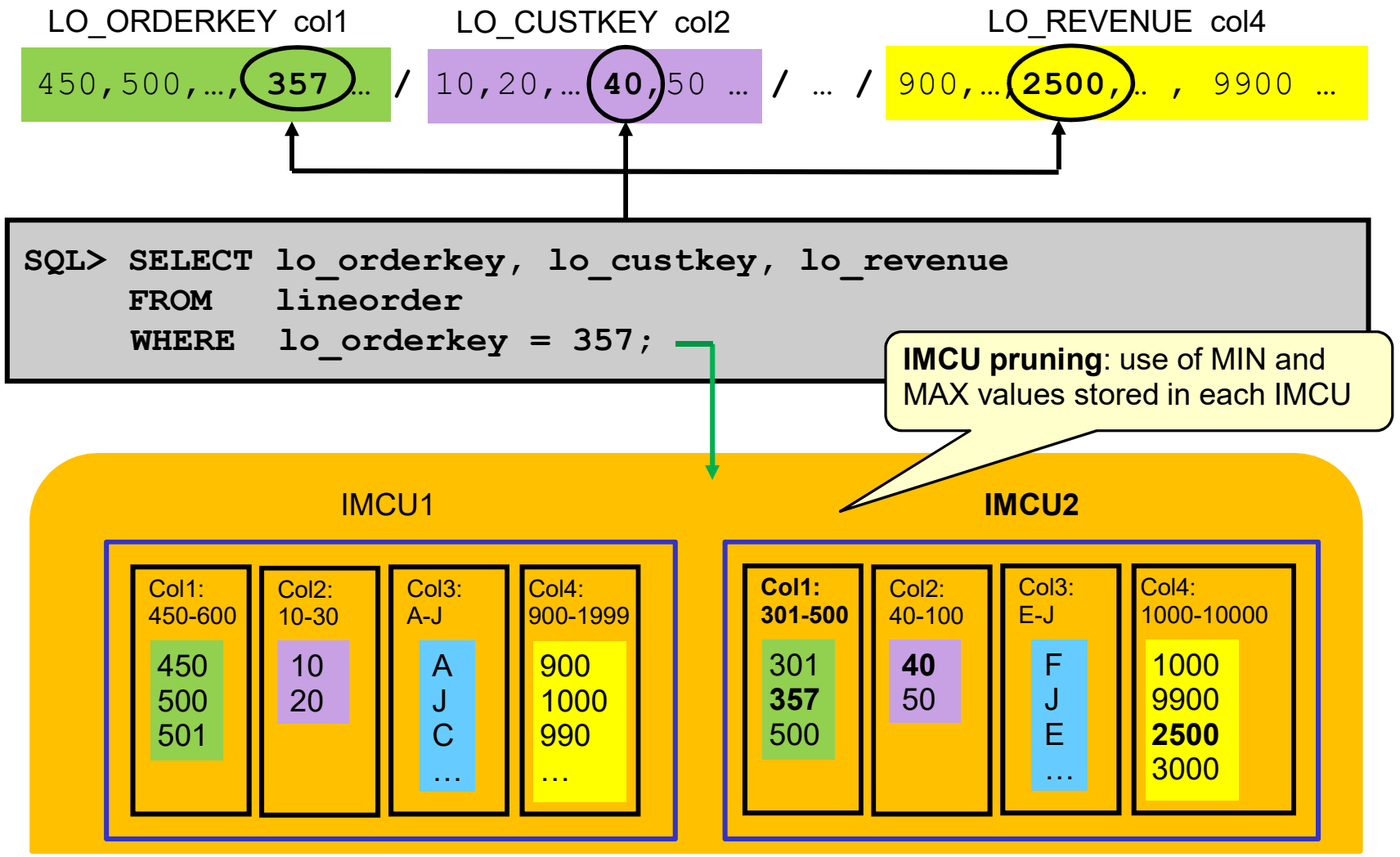
- **Ugyanez az adatblokk gyorsítóból:**

```
SQL> ALTER SESSION SET INMEMORY_QUERY="DISABLE";
SQL> SELECT max(lo_ordtotalprice) most_expensive_order
     FROM   lineorder;
```

Elapsed: 00:**12:21.90**

Use the **INMEMORY_QUERY** session parameter to execute the query against the buffer cache.

Lekérdezések egyszerű szűrési feltételekkel



„MINMAX Pruning” statisztikák

MINMAX pruning történhet különböző `WHERE` feltételek esetén:

```
SQL> SELECT display_name, value
     FROM   v$mystat m, v$statname n
     WHERE  m.statistic# = n.statistic#
     AND    display_name IN (
                    'IM scan segments minmax eligible',
                    'IM scan CUs pruned',
                    'IM scan CUs optimized read',
                    'IM scan CUs predicates optimized');

DISPLAY_NAME                                                     VALUE
---------------------------------------- ----------
IM scan segments minmax eligible                                250
IM scan CUs pruned                                              249
IM scan CUs optimized read                                        0
IM scan CUs predicates optimized                                249
```

Végrehajtási terv:
TABLE ACCESS IN MEMORY FULL

Executed in IM column store or in buffer cache?

```
SQL> SELECT * FROM table(dbms_xplan.display_cursor());
-----------------------------------------------------
| Id  | Operation
-----------------
...
|*  4 |  TABLE ACCESS INMEMORY FULL     | LINEORDER   |
-----------------------------------------------------

    Predicate Infor                (identified by operation id):
-----------------  ------------------------------------
   4 - inmemory("LO_ORDERKEY"=357) filter("LO_ORDERKEY"=357)
```

Executed in IM column store

Executed in IM column store

```
...
|*  4 |  TABLE ACCESS FULL
-----------------------------------------------------
4 - access (:Z>=:Z AND :Z<=:Z) filter("LO_ORDERKEY"=357)
```

Executed against buffer cache

In-Memory táblák join műveletei

Nagyságrendi különbségek a lekérdezések idejében:

```
SQL> SELECT SUM(lo_extendedprice * lo_discount) revenue
     FROM   lineorder l, date_dim d
     WHERE  l.lo_orderdate = d.d_datekey
     AND    l.lo_discount BETWEEN 2 AND 3
     AND    l.lo_quantity < 24
     AND    d.d_date='December 24, 1996';
```

Elapsed: 00:**00:00.28**

```
SQL> SELECT SUM(lo_extendedprice * lo_discount) revenue
     FROM   lineorder l, date_dim d
     WHERE  l.lo_orderdate = d.d_datekey
     AND    l.lo_discount BETWEEN 2 AND 3
     AND    l.lo_quantity < 24
     AND    d.d_date='December 24, 1996';
```

Elapsed: 00:**02:28.85**

DML-k és In-Memory Column Store

- **Az In-Memory column store tartalma frissül a tranzakció végén**

```
SQL> SELECT display_name, value FROM   v$mystat m, v$statname n
     WHERE  m.statistic# = n.statistic#
     AND    display_name like 'IM transactions%';

DISPLAY_NAME                                                     VALUE
---------------------------------------- ----------------------------
IM transactions                                                      1
IM transactions rows journaled                                   10224
…
```

- **Teljesen kompatibilis az OLTP-vel**

Köszönöm a figyelmet

A téma iránt érdeklődőket a következő címen várjuk:

tamas.kerepes@webvalto.hu