High Speed DWH mithilfe der Oracle 12c In-Memory Option

Mag. Dr. Thomas Petrik
Sphinx IT Consulting
Wien

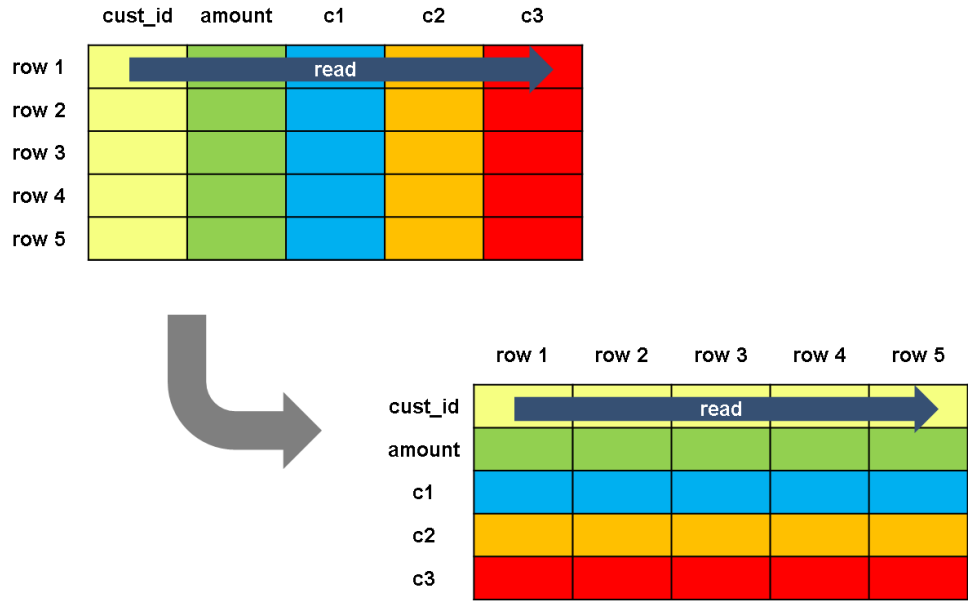
Schlüsselworte

Oracle 12c, DWH, In-Memory, Performance, Tuning, Business Intelligence, ROLAP, MDX, Open Source, Mondrian

Einleitung

Die In-Memory Option der Version 12.1.0.2 stellt einen weiteren Schritt in Richtung wartungsarmes DWH dar. Durch das duale Speicherformat (row based im Buffer Cache und column based im In-Memory Cache) hat der Optimizer zur Laufzeit die Möglichkeit, den optimalen Plan zu wählen, ohne dass ein Eingriff in die Applikation erforderlich wäre. Die abfragespezifische Indizierung kann nahezu vollständig entfallen und speziell BI-Systeme (mit ihren nicht vorhersehbaren, zumeist automatisch generierten Statements) erfahren mit dieser Option eine dramatische Performancesteigerung ohne jeden Tuningaufwand.

Grundlegende Funktionsweise der In-Memory Option



In einem eigenen Cache Bereich der SGA werden ausgewählte Tabellen (und wahlweise auch nur ein Subset an Columns) column based abgelegt. Dabei wird die Datenmenge durch Deduplizierungs- und Kompressionsverfahren einerseits reduziert, andererseits werden die Daten in In-Memory Column

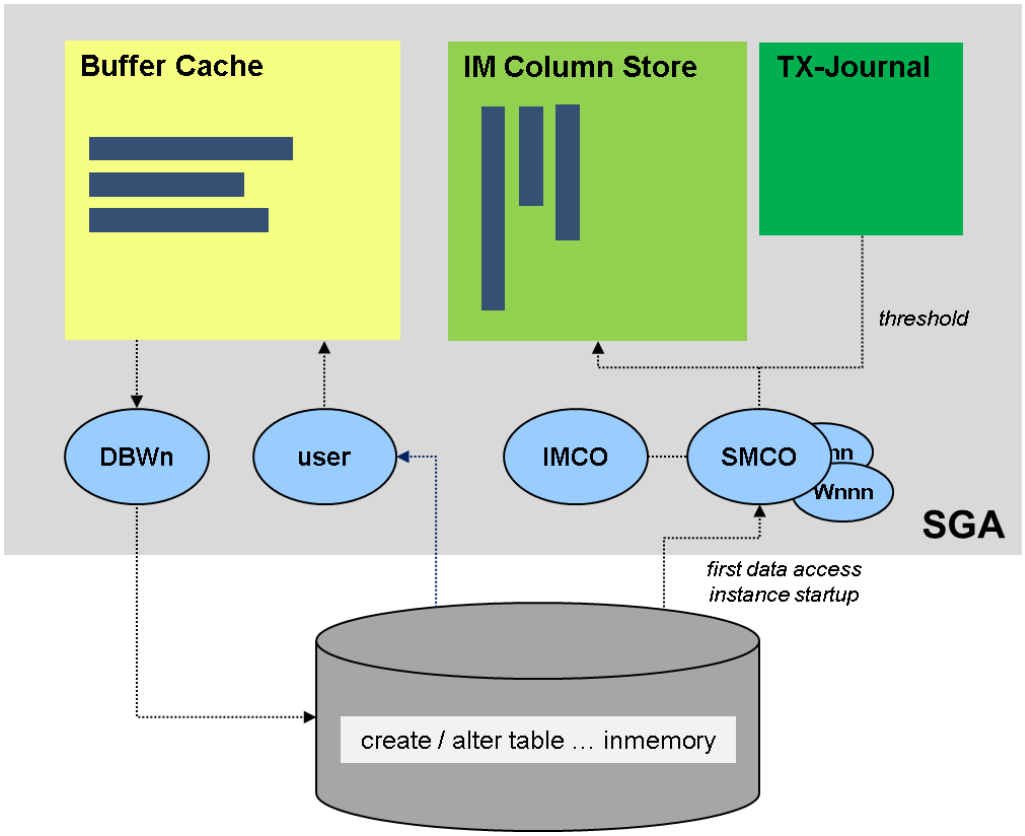
Units (IMCUs) – also einzelne Verwaltungseinheiten – aufgeteilt, für die implizit eigene Storage-Indices mit Min/Max-Werten und Distinct Values angelegt werden.

Dadurch ergibt sich ein dramatischer Performancegewinn einerseits durch die Datenlage beim Scan auf eine Column, andererseits ist die Datenmenge im Vergleich zum Buffer Cache bereits reduziert. Mithilfe des Storage Index kommt es in vielen Fällen zum IMCU Pruning und somit zu einer weiteren Reduktion der zu durchsuchenden Datenmenge.

Der eigentliche Geschwindigkeitsvorteil der IM-Technologie resultiert also aus der neuartigen Speicherarchitektur und nicht so sehr aus der Ablage der Daten im Memory – dies war mit dem Buffer Cache und Technologien wie In-Memory Parallel Query bereits seit Version 11.2 möglich.

Die Aktivierung des IM Column Stores erfolgt lediglich durch (derzeit statische) Definition des Cache Bereichs und entsprechende Attribut-Settings auf Tabellenebene:

```
alter system set compatible=12.1.0.0.0 scope=spfile;
alter system set inmemory_size=2G scope=spfile;
startup force
alter table … inmemory;
```



Die Read Consistency ist zu jeder Zeit garantiert, asynchrone Refreshes (wie bei Materialized Views) sind nicht erforderlich. Ein neuer Background-Prozess IMCO (In-Memory Coordinator) weist den SMCO (Space Management Coordinator) an, mittels mehrerer Worker-Prozesse (Wnnn) den IM

Column Store zu befüllen, wobei dies je nach Load Priority erstmalig nach dem Instance Start oder beim ersten Tabellenzugriff erfolgen kann, z.B.:

```
alter table … inmemory priority high;
```

Level	Beschreibung
critical	IM-Cache Beladung startet nach Instanzstart
high	IM-Cache Beladung startet nach Instanzstart aber erst nach Fertigstellung der CRITICAL Objekte
medium	IM-Cache Beladung startet nach Instanzstart aber erst nach Fertigstellung der HIGH Objekte
low	IM-Cache Beladung startet nach Instanzstart aber erst nach Fertigstellung der MEDIUM Objekte
none	IM-Cache Beladung startet beim 1. Select

Die Definition erfolgt segment based, also für Tabellen, Partitionen, Subpartitionen oder Materialized Views. Dictionary Objekte, Segmente aus dem SYSTEM oder SYSAUX Tablespace sowie Clustered Tables, IOTs, LONGs oder out of line LOBs sind ausgenommen.

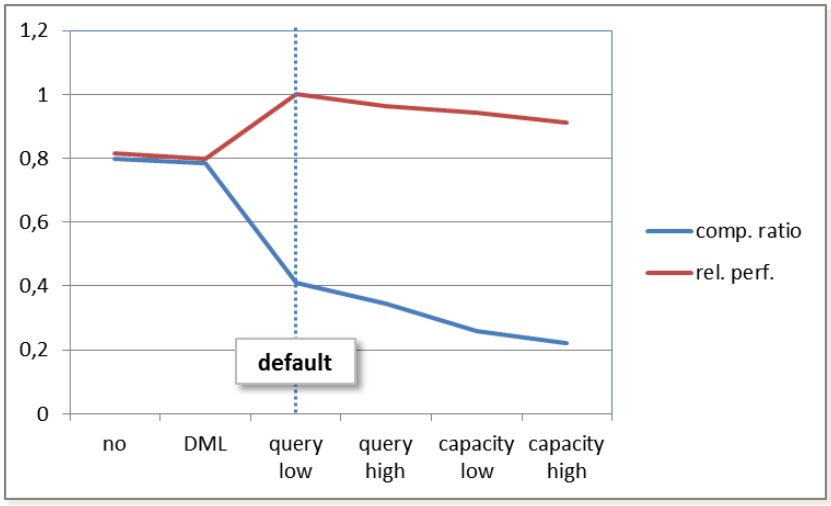
Die Kompression in den IMCUs wirkt sich einerseits auf den Platzverbrauch im Cache und andererseits auf die Performance aus, wobei das Ausmaß der Effekte von der jeweiligen Datenlage abhängt.

Beispiel:

```
alter table … inmemory memcompress for query high;
```

Option	Anmerkung
no memcompress	keine Kompression
memcompress for dml	empfohlen für DML
memcompress for query low	Default
memcompress for query high	
memcompress for capacity low	
memcompress for capacity high	

In der folgenden Abbildung zeigt die blaue Kurve das Kompressionsverhältnis eines typischen Star-Schemas (Swingbench SH-Schema) in Relation zum Platzverbrauch auf Disk (ausgehend von unkomprimierten Daten), während die rote Kurve die Performance-Gewinne bzw. Verluste in Relation zur Default-Einstellung (memcompress for query low) darstellt.



Neue Optimizer Methoden

3 Methoden tragen wesentlich zur Performance bei:

1. SIMD – Single Instruction Multiple Data
   Damit wird ein (aus der Gaming-Technologie) bekanntes Prozessor Feature genutzt, um ein Array von Werten einer Column in einem Takt zu verarbeiten.
2. In-Memory Joins / Bloom Filter
   Bloom Filter gibt es in der Oracle DB intern bereits seit 10.2, in Optimizer Plänen tauchen sie aber erst jetzt auf. Mithilfe dieser Technologie werden Filterkriterien auf Basis kleiner Dimensionstabellen in Form von Bitmap-Vektoren abgebildet ("Join Filter Create"), die dann in einem 2. Schritt unmittelbar beim Scan der Faktentabelle angewandt werden ("Join Filter Use").
3. In-Memory Aggregation / Vector Group By
   Dabei handelt es sich ebenfalls um eine mehrphasige Methode, bei der zunächst Filter-Bitmaps und temporäre Tabellen aus den Dimensionen aufgebaut werden und es dann in einer Aggregator Structure in der PGA bereits während des Scans der Faktentabelle zur Aggregation kommt.

In-Memory im RAC

Im RAC lässt sich zunächst die Verteilung der Rows auf die pro Instanz vorhandenen Column Stores bestimmen (DISTRIBUTE BY Clause). Diese kann auf Rowids, Partitions oder Subpartitions basieren.

Weiters können IMCUs nur in einem Cache liegen (NO DUPLICATE), redundant in 2 Caches (DUPLICATE) oder alle Caches sollen den gleichen (kompletten) Inhalt haben (DUPLICATE ALL).

Letztere Variante entspricht einer Single Node Konfiguration für jede Instanz und harmoniert sehr gut mit der Einstellung PARALLEL_FORCE_LOCAL=TRUE, während die zweifach redundante Variante im Wesentlichen beim Ausfall eines Knotens vor einem Performanceeinbruch durch das Nachladen des Column Stores schützen soll.

Bei verteilten Parallel Queries (PARALLEL_FORCE_LOCAL=FALSE) werden remote IMCUs via Cache Fusion abgefragt.

Einsatz im DWH

Im DWH wird der maximale Performancegewinn im Business Layer (Datamart) erzielt. Dabei geht es gar nicht so sehr um die absolute Größe der beteiligten Tabellen, vielmehr ist die Relation der Tabellengrößen zueinander sowie die Zahl der selektierten Spalten und Aggregationen relevant.

Durch den Einsatz der IM-Technologie werden ROLAP (relationales OLAP) Anwendungen erstmals auch für den professionellen Einsatz mit größeren Datenmengen interessant. Das Open Source Projekt Mondrian (das beispielsweise im Pentaho BI-Server Verwendung findet) setzt auf diese Technologie. Dabei werden Cubes nur logisch definiert (kein zusätzlicher ETL-Prozess), MDX wird direkt auf SQL umgesetzt und gegen die DB gefeuert. Dahinter liegen Star-Schemata, sodass diese Vorgehensweise massiv von der IM-Technologie profitiert.

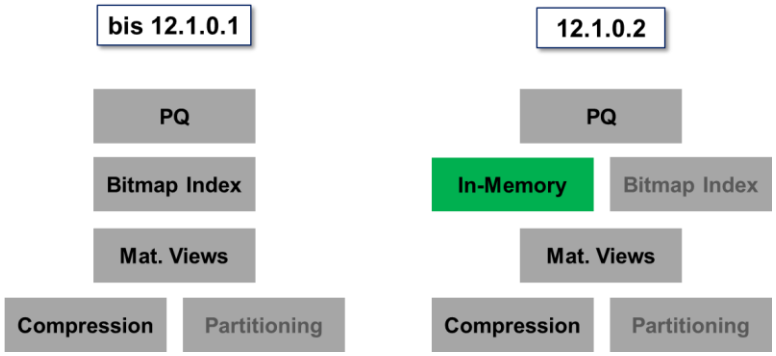
Da im Allgemeinen nicht alle Tabellen in den Column Store passen (von teilweisen Beladungen ist aus Performancegründen dringend abzuraten), muss eine sinnvolle Auswahl (u.U. auch auf Spaltenebene) getroffen werden.

Durch dem Einsatz des IM Column Stores werden Performanceindices weitgehend überflüssig. Eine unnötige Indizierung (Überindizierung) kann den Optimizer zu falschen Plänen verleiten, daher sind diese Indices zu entfernen.

Parallel Query und Materialized Views (die ihrerseits wiederum In-Memory abgelegt werden sollten) ergänzen den Tuningansatz optimal. Partitioning kann vor allem bei historisierten Datamarts in Betracht gezogen werden, da ja auch der IM Column Store auf Segmentebene arbeitet.

Für die Dimensionierung des IM-Cache ist nicht immer zusätzliches Memory erforderlich. In bestehenden DWHs mit großem Buffer Cache kann durchaus ein Teil davon für den Column Store verwendet werden. Bei der Durchführung von Benchmarks ist aber in jedem Fall auf die Vergleichbarkeit der Ergebnisse zu achten, d.h. auch ohne IM-Cache sollten die Blöcke der abgefragten Tabellen vollständig im Buffer Cache liegen. Performancegewinne mithilfe der IM-Option um Faktoren von 1000 oder mehr deuten zumeist auf unzureichendes Caching im Altsystem hin, sodass durch Einsatz der IM-Technologie primär die Vermeidung von I/O eine Rolle spielt.

Building Blocks für ein erfolgreiches DWH:



Zusammenfassung

Die IM-Option der 12c Datenbank eröffnet vollkommen neue Möglichkeiten für Business Analytics Systeme. Dabei profitieren bereits scheinbar kleine Schemata (einige Millionen Rows in der Faktentabelle) massiv – die Betrachtung ausschließlich großer DWHs greift hier viel zu kurz. ROLAP Systeme ohne zusätzlichen ETL-Aufwand für den Cube-Aufbau werden attraktiv und das mühsame Statement-Tuning entfällt weitgehend. Speziell in BI-Umgebungen werden Abfragen oftmals

automatisch generiert und sind vorab unbekannt. Performancetuning mittels Indizierung war daher immer schon ein unzureichendes Hilfsmittel in diesem Bereich und kann nun durch die sehr wartungsarme IM-Technologie ersetzt werden. Die klare Strukturierung des DWH in ein Core-DWH und einen Datamart-Teil erlangt dadurch zusätzliche Bedeutung.

Kontaktadresse:

Mag. Dr. Thomas Petrik
Sphinx IT Consulting
Aspernbrückengasse 2
A-1020 Wien

Telefon:	+43 664 155 8304
Fax:	+43 (1) 599 31-99
E-Mail	Thomas.Petrik@sphinx.at
Internet:	www.sphinx.at