Oracle In-Memory & Data Warehouse: Die perfekte Kombination?

Dani Schnider
Trivadis AG
Zürich/Glattbrugg, Schweiz

Einleitung

Als Larry Ellison in einer Keynote im Juni 2014 die Oracle In-Memory Option vorstellte, sprach er von der grossen Innovation, dass die gleichen Daten nun in zwei Formaten zur Verfügung stehen: im bisherigen Row-Format für OLTP und neu im Column-Format der In-Memory Option für analytische Abfragen. Diese Aussage wurde von vielen Leuten dann so interpretiert, dass durch den Einsatz von Oracle In-Memory kein Data Warehouse mehr notwendig sei – schliesslich können die analytischen Abfragen ja nun direkt auf den OLTP-Systemen ausgeführt werden.

Dem ist natürlich nicht so. Typische Aufgaben eines Data Warehouses wie die Integration von Daten aus unterschiedlichen Quellen und die Historisierung und Versionierung aller Datenänderungen sind nach wie vor notwendig und setzen weiterhin eine geeignete Architektur voraus, in welcher operative und analytische Anforderungen sauber getrennt werden. Durch den Einsatz von Oracle In-Memory können jedoch analytische Abfragen im Data Warehouse stark beschleunigt werden. Doch für eine effiziente Nutzung dieser Features braucht es etwas mehr als nur das Aktivieren der In-Memory Option. Was sind die Auswirkungen auf Architektur, Datenmodell und physischem Design eines Oracle Data Warehouses, um die bestmöglichen Resultate zu erhalten?

Oracle Database In-Memory

Oracle Database In-Memory wurde mit Oracle 12c Release 1 (12.1.0.2) eingeführt und ist als Option der Enterprise Edititon verfügbar. Es handelt sich hierbei nicht um eine reine In-Memory-Datenbank, sondern um eine Erweiterung der relationalen Datenbank von Oracle. Die Daten werden zusätzlich zur bisherigen Speicherung in der Datenbank auch im Arbeitsspeicher des Datenbankservers gehalten, und zwar im sogenannten „Column Store“. Dabei werden die Daten mittels einer speziellen Komprimierungstechnik (Columnar Compression) stark komprimiert. Je nach Konfiguration sind Kompressionsraten zwischen 2 und 20 möglich.

Dass die Daten im Memory gehalten werden, um Abfragen zu beschleunigen, ist nicht neu. Auch im Buffer Cache werden Daten im Arbeitsspeicher zwischengespeichert, um die Anzahl der Diskzugriffe zu minimieren. Dabei werden die zuletzt verwendeten Blöcke im Memory gehalten, sodass bei mehrmaligen Zugriffen auf die gleichen Datensätze diese aus dem Cache gelesen werden können. Der entscheidende Unterschied beim Column Store liegt darin, dass hier die Daten nicht zeilenweise (bzw. als Tabellenblöcke) in den Arbeitsspeicher geladen werden, sondern spaltenweise komprimiert im Memory gehalten werden. Diese Speichermethode erlaubt sehr effiziente Abfragen auf einzelne oder wenige Spalten von grossen Tabellen mit vielen Rows, die mittels Filtern eingeschränkt und ggf. aggregiert werden. Diese Art von Abfragen kommen typischerweise in Data Warehouses vor.
Vereinfacht gesagt: Der Buffer Cache dient der Performanceoptimierung von selektiven Abfragen, wie sie typischerweise in OLTP-Systemen ausgeführt werden. Der In-Memory Column Store hingegen dient zur Optimierung von nicht-selektiven Abfragen, wie sie häufig in Data Warehouses vorkommen.

Ist die In-Memory Option aktiviert, stehen die Daten sowohl zeilenweise sowie spaltenweise (im In-Memory Column Store) zur Verfügung. Je nach Art der Abfrage wird auf die einzelnen Tabellenblöcke (auf Disk oder via Buffer Cache) oder auf den Column Store zugegriffen. Änderungen werden im Column Store nachgeführt, sodass die In-Memory Option ohne applikatorische Anpassungen verwendet werden kann.

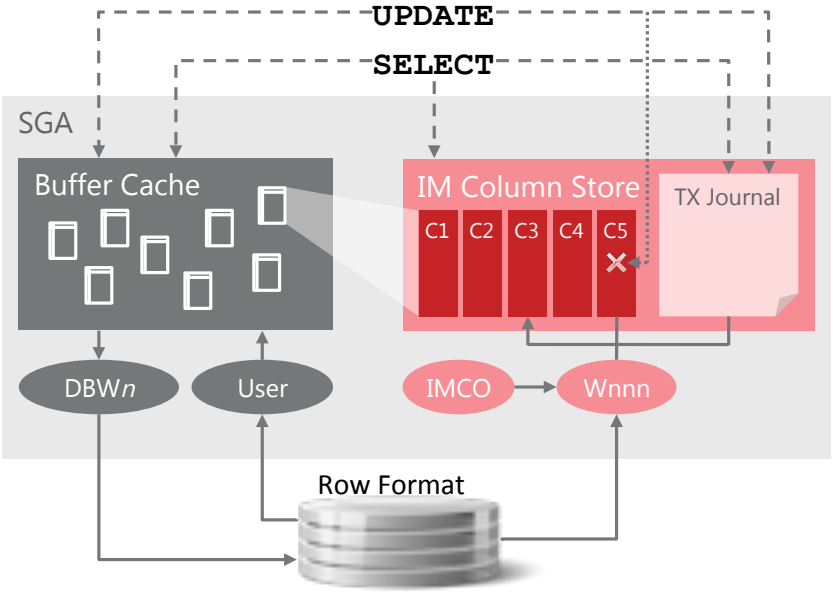

Abbildung 1: Grundprinzip des In-Memory Column Store

Die In-Memory Option wird über den Konfigurationsparameter *inmemory_size* aktiviert. Der Wert des Parameters bestimmt den Speicherplatz, welcher für den In-Memory Column Store reserviert wird (mind. 100 MB). Standardmässig ist der Parameter auf 0 gesetzt, d.h. die In-Memory Option ist nicht aktiviert.

Das Aktivieren des Column Store ist jedoch nur der erste Schritt. Erst durch die Konfiguration, welche Daten in den Column Store geladen werden, kann die In-Memory Option sinnvoll eingesetzt werden. Für diese Aufgabe müssen wir unser Data Warehouse und die Abfragen darauf genauer untersuchen.

Welche Daten gehören in den In-Memory Column Store?

Hätten wir beliebig viel Memory auf dem Datenbankserver zur Verfügung, könnten wir unser gesamtes Data Warehouse in den In-Memory Column Store laden. Dies ist aber weder realistisch noch sinnvoll. Deshalb kann und muss detailliert festgelegt werden, welche Tabellen in den Column Store geladen werden soll. Es ist auch möglich, nur einzelne Spalten oder Partitionen einer Tabelle zu laden oder diese explizit auszuschliessen.

Für die meisten Bereiche eines Data Warehouses bringt die In-Memory Option keinen oder nur einen geringen Nutzen. Tabellen der Staging Area (und eventuell Cleansing Area, falls vorhanden) werden durch die ETL-Prozesse gefüllt und danach von den nachfolgenden ETL-Prozessen einmal gelesen. Diese Daten zusätzlich in den Column Store zu laden, würde keinen Nutzen bringen. Dies gilt auch für die Tabellen im Core, sofern keine direkten Abfragen darauf ausgeführt werden (Ausnahmen werden am Vortrag erläutert). Die In-Memory Option kommt somit hauptsächlich in den Data Marts zum

Einsatz, auf welche die Benutzer mittels BI-Tools zugreifen. Für die nachfolgenden Betrachtungen gehen wir von Data Marts mit einem dimensionalen Datenmodell (also Star Schemas mit Fakten- und Dimensionstabellen) aus.

Effiziente Abfragen auf Star Schemas mit In-Memory

Um die Möglichkeiten der In-Memory Option optimal nutzen zu können, werden idealerweise alle Tabellen eines Star Schemas in den Column Store geladen. Weil dabei die einzelnen Spalten separat komprimiert werden können, sind hohe Kompressionsraten möglich, sodass auch grosse Faktentabellen im Column Store Platz finden. Die Dimensionstabellen werden ebenfalls ins Memory geladen – sie sind in der Regel vom Platzbedarf her vernachlässigbar.

Stehen nun alle Tabellen – spaltenweise komprimiert – im Column Store zur Verfügung, hat der Optimizer verschiedene Möglichkeiten, die benötigten Daten effizient zu lesen. Wie üblich wird dabei die günstigste Variante, also der Ausführungsplan mit den geringsten Kosten, verwendet. Folgende Features werden von Oracle Database In-Memory zur Verfügung gestellt:

- **In-Memory Scan**: Typische Abfragen auf ein Star Schema lesen viele Datensätze aus einer Faktentabelle, eventuell eingeschränkt durch Filterkriterien auf einer oder mehreren Dimensionen, und aggregieren die Kennzahlen auf die gewünschte Granularität (z.B. Summe aller Verkaufszahlen pro Bundesland und Monat). Das Lesen einzelner Attribute einer grossen Faktentabelle kann durch In-Memory Scans stark beschleunigt werden.
- **In-Memory Join**: Eine effiziente Art, um in Oracle mehrere Tabellen zu joinen, besteht in der Verwendung sogenannten „Bloom Filters“. Dieses Verfahren, das bereits mit Oracle 10g eingeführt wurde, kann auch für Spalten von Dimensionen und Fakten im Column Store verwendet werden und wird als In-Memory Join bezeichnet.
- **In-Memory Aggregation**: Für die Aggregation der Fakten auf die erforderliche Granularität der verwendeten Dimensionsattribute kann die „Vector Transformation“ verwendet werden. Vom Prinzip her ist sie vergleichbar mit der Star Transformation, verwendet aber keine Bitmap Indizes, sondern sogenannte „Key Vectors“, die zur Laufzeit einer Abfrage aufgebaut werden.

Die einzelnen Verfahren werden im Rahmen der Präsentation anhand von Live-Demos gezeigt.

Auswirkungen auf das physische Datenbankdesign

Durch die Verwendung von Oracle Database In-Memory ändert sich grundsätzlich nichts an Architektur und Datenmodellen eines Data Warehouses. Nach wie vor haben wir verschiedene DWH-Schichten, und die Star Schemas im der Mart-Schicht bestehen immer noch aus Dimensions- und Faktentabellen. Werden diese jedoch in den In-Memory Column Store geladen, so hat dies Auswirkungen auf das physische Design des Data Marts. Die wichtigsten Punkte dabei sind:

- **Indexierung**: Indizes werden nur noch für die Primary Key Constraints der Dimensionstabellen verwendet. Der sonst wichtige Grundsatz, für jeden Dimensionsschlüssel der Faktentabelle einen Bitmap Index zu erstellen, ist mit der In-Memory Option nicht mehr relevant. Die Kombination von In-Memory Scans und In-Memory Joins ist viel schneller als der Zugriff über Indizes, und anstatt der Star Transformation wird die In-Memory Aggregation mittels Vector Transformation verwendet. Somit müssen auf den Faktentabellen keine Indizes mehr erstellt werden.

- **Materialized Views**: Häufig werden in Data Marts mit grossen Faktentabellen die Kennzahlen voraggregiert in Materialized Views gespeichert, um Abfragen mittels Query Rewrite zu beschleunigen. Steht die Faktentabelle vollständig im Column Store zur Verfügung, ist dies nicht mehr notwendig. In-Memory Scans können die Kennzahlen sehr rasch lesen und aggregieren, ohne dass dazu vorberechnete Aggregationen notwendig sind. Auch die Definition von „Hierarchie-Constrains“ (CREATE DIMENSION) auf den einzelnen Dimensionstabellen entfällt.
- **Constraints**: Primary Key Constrains auf den Dimensionstabellen sowie Foreign Key Constraints auf den Faktentabellen sollten weiterhin angelegt werden. Für die Abfragefeatures der In-Memory Option werden sie zwar nicht benötigt, aber für das Verständnis und die Lesbarkeit des Datenmodells sind Constraints weiterhin empfohlen.
- **Partitionierung**: Grosse Faktentabellen sollten weiterhin partitioniert werden, denn Partition Pruning wird auch auf den Column Store angewendet. Ausserdem ergibt sich dadurch die Möglichkeit, nur die häufig verwendeten Partitionen in den Column Store zu laden (falls zu wenig Memory für die gesamte Faktentabelle vorhanden ist).
- **Statistiken**: Das regelmässige Aktualisieren der Statistiken auf Tabellen und Partitionen, idealerweise jeweils nach dem Laden des Data Marts, ist weiterhin sehr wichtig. Der Optimizer benötigt die Statistiken für die Ermittlung der Selektivitäten jeder Dimension und entscheidet sich aufgrund dieser Informationen, ob und wie die Daten aus dem Column Store gelesen und gejoined werden.

Zu wenig Memory – was nun?

Die Grösse des In-Memory Column Stores kann bei Bedarf erweitert werden, aber bei vielen grossen Tabellen mit vielen Attributen tritt irgendwann die Situation auf, dass nicht mehr der ganze Data Mart ins Memory geladen werden kann. Was nun?

Wie bereits erwähnt, können nicht nur ganze Tabellen in den Column Store geladen werden. Stattdessen bestseht die Möglichkeit, nur die häufig verwendeten Attribute zu laden. Bei einer Faktentabelle heisst dies typischerweise, dass nur die regelmässig abgefragten Kennzahlen sowie häufig verwendete Dimensionsschlüssel geladen werden. Zusätzliche Attribute und technische Columns (z.B. Audit-Columns, Load-ID, etc.) können weggelassen werden, wenn sie in den Abfragen nicht vorkommen.

Für partitionierte Faktentabellen besteht ausserdem die Möglichkeit, nur einzelnen Partitionen in den Column Store zu laden. Ist beispielsweise eine Tabelle monatsweise partitioniert und werden vor allem Abfragen auf das aktuelle Kalenderjahr ausgeführt, kann es zweckmässig sein, nur die Monatspartitionen des laufenden Jahres zu laden. Dies hat aber wiederum Auswirkungen auf das physische Design. Um trotzdem effiziente Abfragen auf historische Daten ausführen zu können, sind Bitmap Indizes auf die nicht im Memory geladenen Partitionen zweckmässig. Hier bietet sich eine weitere Möglichkeit von Oracle 12c an: Die Definition von Partial Indexes, d.h. Indizes, die nur auf bestimmten Partitionen erstellt werden. Durch die Kombination der In-Memory Option für aktuelle Daten und Partial Indexes für historische Daten lassen sich die Vorteile der verschiedenen Verfahren kombinieren.

Fazit

Oracle Database In-Memory ist ein mächtiges Feature von Oracle 12c, das sich sehr gut für den Einsatz in Data Warehouses eignet. Durch den Einsatz des In-Memory Column Store in den Data Marts können Abfragen massiv beschleunigt werden. Allerdings ist dazu mehr notwendig als nur das Einschalten der In-Memory Option. Vor allem auf das physische Design der Data Marts hat der Einsatz der In-Memory Option grosse Auswirkungen. Vieles wird vereinfacht, aber neue Fragen in bezug auf Konfiguration und Design der Datenbank müssen geklärt und je nach Konstellation der Daten entsprechend umgesetzt werden. Die notwendigen Werkzeuge dafür stellt Oracle zur Verfügung, aber anwenden müssen wir sie weiterhin selber.

Quellen und weitere Informationen

- Oracle Database Online Documentation 12c Release 1 (12.1) Database SQL Tuning Guide, Chapter 5: Query Transformations
  https://docs.oracle.com/database/121/TGSQL/tgsql_transform.htm#TGSQL95256
- Oracle Database In-Memory: In-Memory Aggregation Oracle White Paper, January 2015 (William Endress)
  http://www.oracle.com/technetwork/database/bi-datawarehousing/inmemory-aggregation-twp-01282015-2412192.pdf
- Oracle Database In-Memory, Oracle White Paper, July 2015 (Maria Colgan)
  http://www.oracle.com/technetwork/database/in-memory/overview/twp-oracle-database-in-memory-2245633.html
- Oracle Scratchpad, In-memory Aggregation, August 2014 (Jonathan Lewis)
  https://jonathanlewis.wordpress.com/2014/08/24/in-memory-aggregation/
- Oracle Database In-Memory – A game-changer for Data Warehousing?
  Präsentation DOAG Konferenz, November 2014 (Maria Colgan, Hermann Baer)
- Oracle Database In-Memory Option, Trivadis Work Bench, November 2014
  (Christian Antognini, Robert Bialek, Daniele Massimi, Peter Welker)

Kontaktadresse:

Dani Schnider
Trivadis AG
Sägereistrasse 29
CH-8152 Glattbrugg

Telefon: +41 58 459 50 81
Fax: +41 58 459 56 95
E-Mail dani.schnider@trivadis.com
Internet: www.trivadis.com