High Performance BI mit In-Memory-Datenbanken

Reinhard Mense
areto consulting gmbh
Köln

Schlüsselworte
In-Memory-Datenbank, Oracle In-memory-Option, Performance, Data Warehouse, Business Intelligence.

Einleitung

Die In-Memory-Option für Oracle 12c wird großen Einfluss auf die Entwicklung von BI/DWH-Systemen haben. Sie kann deshalb sicherlich als eine der spannendsten Neuerungen der letzten Jahre im BI-Umfeld angesehen werden.

Der Vortrag wird sich mit der Funktionsweise von spaltenorientierten In-Memory-Datenbanken und deren Einsatzszenarien befassen. Die Arbeitsweise der Oracle In-Memory-Option wird anhand von Beispielen erläutert. Es werden die Auswirkungen der Oracle In-Memory-Option auf die Architektur und Modellierung von bestehenden und zukünftigen BI/-DWH-Systemen dargestellt.

Abschließend werden Veränderungen und Chancen, die sich durch die Einführung der In-Memory-Technologie für BI/DWH-Projekte ergeben, erläutert.

Agenda

- Funktionsweise der Columnar In Memory Technologie
- Einsatzgebiete in BI-Projekten
- Chancen für BI/DWH-Projekte
- Vorteile und Nutzen der Oracle In Memory Option

Funktionsweise der Columnar In Memory Technologie

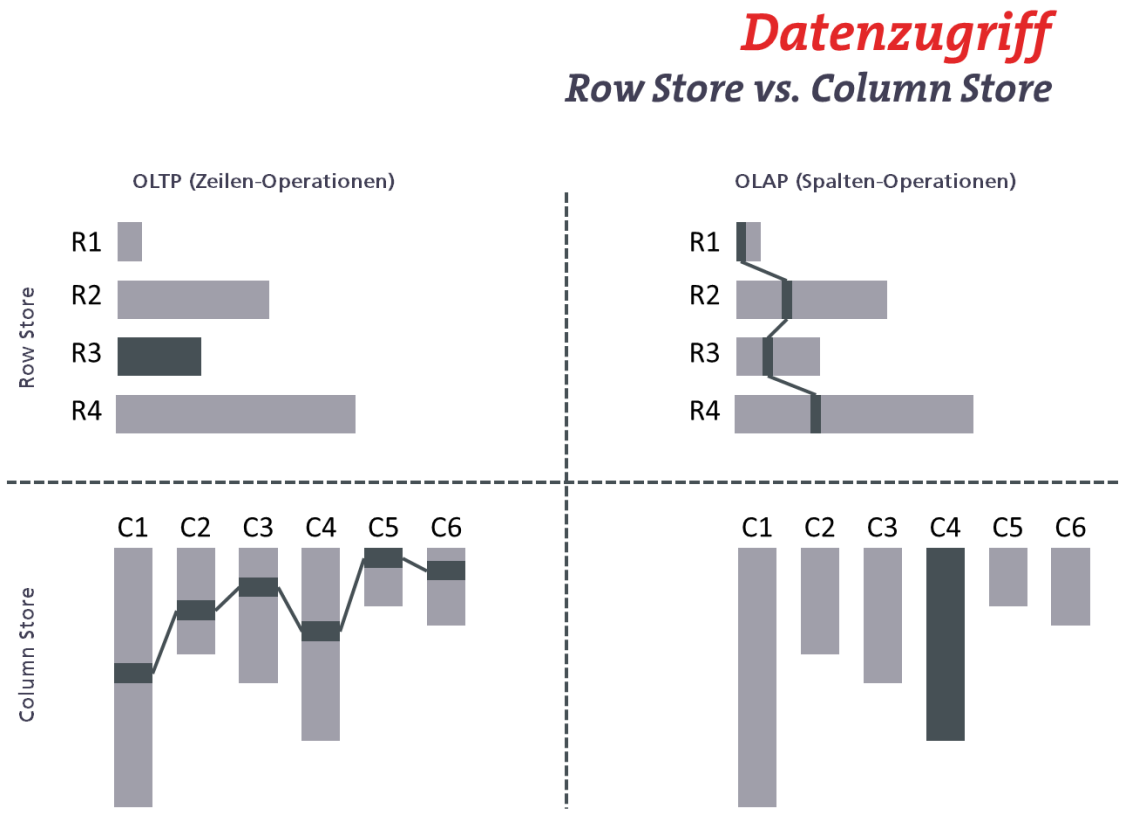

Abb. 1: Row Store vs. Column Store

Die hohe Abfrage-Performance von Columnar Stores beruht im Wesentlichen auf vier Techniken:

1. Leichtgewichtige Komprimierung (Dictionary, RLE)
2. Blockweise Vektor-Verarbeitung
3. Späte Materialisierung der Ergebnis-Rows
4. Bloom-Filter

Dictionary Compression

Durch den Einsatz der Dictionary Compression werden die Werte eine Spalte durch nummerische Schlüssel ersetzt.

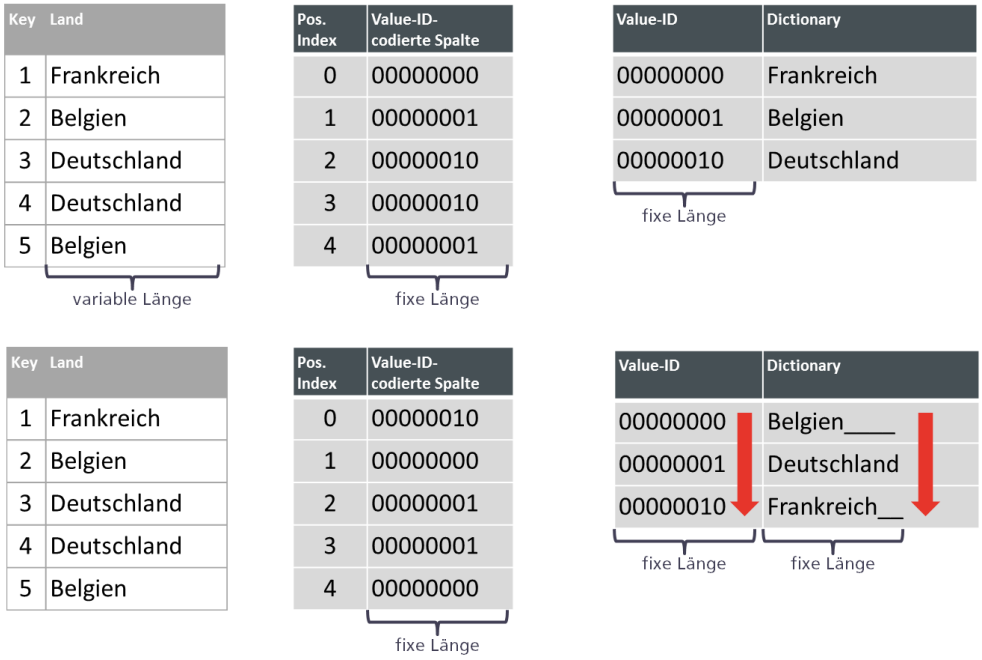

Abb. 2: Beispiel für Dictionary Compression

Bei der Dictionary Compression werden nummerische Schlüssel (Value-ID) mit einer fixen Länge verwendet. Dadurch wird der direkte Zugriff auf Elemente im Hauptspeicher mit Hilfe eines Positionsindex möglich. Die Berechnung der Speicheradresse erfolgt dabei wie folgt:

Speicheradresse = Basisadresse + Positionsindex * Länge der Value-ID.

Werden ausserdem Dictionary-Werte mit fixer Länge verwendet, erlauben auch Value-IDs den direkten Zugriff auf Dictionary-Elemente mit Hilfe der Value-ID. Die Berechnung der Speicheradresse erfolgt dabei wie folgt:

Speicheradresse = Basisadresse + Value-ID * Länge der Werte.

Nebem dem direkten Hauptspeicherzugriff ist ein weiterer Vorteil der Dictionary Compression die Reduzierung des Speicherbedarfs. Dazu eine Beispielrechnung: Angenommen eine Tabelle enthalte 999.997 Datensätze und die drei Spalten KEY, LAND (drei unterschiedliche Werte) und GEBURTSJAHR. Die Spalte KEY sei die Schlüsselspalte und die Werte benötigen stets drei Bytes, die Spalte LAND enthalte drei unterschiedliche Werte, die im Durchschnitt neun Bytes jedoch maximal elf Bytes benötigen, und die Spalte GEBURTSJAHR enthalte 39 verschiedene Werte, die stets zwei Bytes benötigen. Damit ergibt sich für einen Row Store in etwa folgender Speicherbedarf:

Speicherbedarf im Row Store
= 999.997 x (3 Bytes + 9 Bytes + 2 Bytes) = 13.999.958 Bytes = ca. 13,4 Mbytes

Für einen Column Store ergibt sich hingegen folgende Rechnung:

Speicherbedarf im Column Store
= 999.997 x (3 Bytes + 1 Byte + 1 Byte)

+ 999.997 x 3 Byte + 3 x 11 Byte + 39 x 2 Bytes
= 8.000.087 Bytes = ca. 7,6 MBytes

Somit benötigt in diesem Beispiel die gleiche Datenmenge deutlich weniger Speicher im Column Store, so dass der Hauptspeicher deutlich effizienter genutzt werden kann.

Bit Packing

Beim Bit-Packing werden die Value-IDs auf die minimale Anzahl Bits reduziert, so dass mehrere Werte in einem Byte gespeichert werden können. Die folgende Abbildung illustriert das Verfahren.

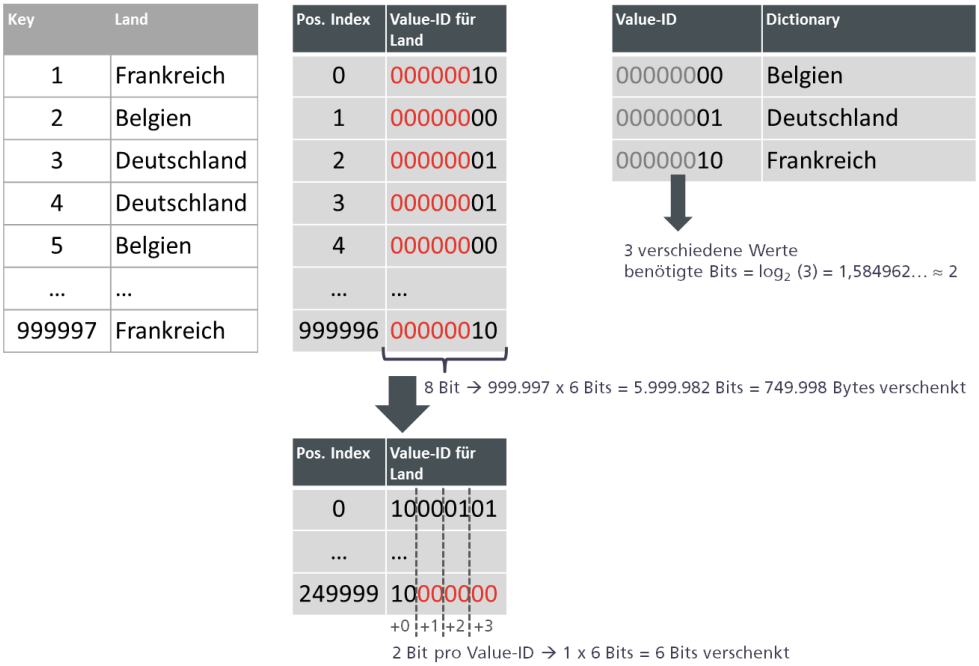

Abb.3: Durch Bit-Packing den Speicherplatz optimal nutzen

Die Kalkulation des Speicherbedarfs verändert sich beim Einsatz von Bit-Packing für das obige Beispiel wie folgt:

Speicherbedarf im Column Store
= 999.997 x 3 Bytes
+ 250.000 x 1 Byte
+ 750.000 x 1 Byte
+ 999.997 x 3 Byte + 3 x 11 Byte + 39 x 2 Bytes
= 7.000.093 Bytes = ca. 6,7 Mbytes

Die Effizienz der Speichernutzung kann somit noch einmal deutlich verbessert werden.

Bit-Packing bietet also den Vorteil der effizienteren Speichernutzung, erfordert jedoch auf der anderen Seite auch die Extraktion der gepackten Bits. Die Extraktion kann jedoch ausschliesslich mit Hilfe einfacher Bit-Operationen erfolgen, so dass der Performance-Verlust sehr gering ausfällt.

Oracle In Memory Option
Die Oracle In Memory Option ist sehr gut in die Oracle-Datenbank integriert, und kann sehr einfach genutzt werden:

```
alter system set inmemory_size = <size> GB;
alter table <table name> inmemory;
alter table <table name> partition <partition name> inmemory;
```

Aufgrund dieser einfachen Integration bleibt das bestehende Oracle Know How weiterhin von Nutzen. Ausserdem können bewährte Techniken der Oracle Datenbank, wie z. B. Materialized Views, ggf. zusätzlich eingesetzt werden.

Kontaktadresse:
Reinhard Mense
areto consulting gmbh
Schanzenstr. 6-20
D-51063 Köln

Telefon:	+49 (0) 221-66 95 75-0
Fax:	+49 (0) 221-66 95 75-99
E-Mail	reinhard.mense@areto-consulting.de
Internet:	www.areto-consulting.de