Vorteile des In-Memory Column Store gegenüber einer klassischen Row Store-Datenbank

Bei der Beurteilung der Leistungsfähigkeit von In-Memory-Column-Store-Datenbanken im Vergleich zu klassischen Row-Store-Datenbanken werden gelegentlich die zwei untenstehenden Aussagen getätigt, die zunächst logisch erscheinen, jedoch bei näherer Betrachtung nicht haltbar sind. Ich möchte explizit darauf hinweisen, dass ich nachfolgend nur die pauschale Gültigkeit dieser Aussage widerlegen möchte, auch im Bewusstsein, dass die Vorteile des In-Memory-Column-Store nicht stets ihre volle Wirkung zeigen. Gleichwohl sind sie meiner Erfahrung nach bei den meisten typischen DWH-/BI-Abfragen deutlich spürbar.

1. „Wenn man bei einer klassischen Row-Store-Datenbank den Cache groß genug dimensioniert, so dass alle Tabellen im Hauptspeicher gehalten werden können, wird die Datenbanken ähnlich oder gleich schnell wie eine In-Memory-Datenbank sein“

Bei dieser Aussage werden gleich mehrere Vorteile des In-Memory-Column-Store außer Acht gelassen:

- Durch die Speicherung im Column Store werden nur die tatsächlich selektierten Spalten gelesen. Beim Row Store hingegen, müssen stets alle Spalten eines Datensatzes gelesen werden.
- Im Column Store werden typischerweise leichtgewichtige Komprimierungsverfahren (z. B. Dictionary Compression) eingesetzt, so dass im Vergleich zu einem nicht komprimierten Row Store mehr Daten in den Hauptspeicher geladen werden können.
- Wird der Row Store ebenfalls komprimiert, findet die Komprimierung meist auf Datenblock-Ebene statt. Für den Zugriff auf die benötigten Spalten ist dann in der Regel zunächst eine Dekomprimierung erforderlich. Im Column Store kann aufgrund der leichtgewichtigen Komprimierung die Selektion der Daten größtenteils im komprimierten Zustand erfolgen. Im Idealfall wird erst die finale Ergebnismenge dekomprimiert. (Prinzip der späten Materialisierung)

Fazit: Die Speicherung als Column Store und die leichtgewichtige Komprimierung führen bei typischen DWH-/BI-Abfragen dazu, dass weniger Daten gelesen werden müssen. Durch das Prinzip der späten Materialisierung werden unnötige Dekomprimierungen stark reduziert. Beim Column Store erfolgt somit der Zugriff auf den Hauptspeicher effizienter.

2. „Durch die spaltenbasierte Speicherung erfordert der Zugriff auf mehrere Spalten einer Tabelle aufwendige Join-Operationen, so dass solche Abfragen langsam sind.“

Bei dieser Aussage werden Column Stores mit einzelnen Tabellen gleich gesetzt, und es wird davon ausgegangen, dass für diese die bei Row Stores bekannten Mechanismen angewendet werden. Die Struktur der Column Stores ermöglicht jedoch den Einsatz anderer sehr effizienter Mechanismen.
Bei der Selektion mehrerer Spalten einer Tabelle profitiert der Column Store wieder von den Eigenschaften einer leichtgewichtigen Komprimierung, der Dictionary Compression. Bei der Dictionary Compression werden für die unterschiedlichen Werte einer Spalte jeweils numerische Werte (Value-IDs) vergeben und in einem Dictionary gespeichert.
In den Spalten werden statt der Originalwerte die Value-IDs gespeichert. Die Value-IDs besitzen dabei eine fixe Breite. Jedem Spaltenwert wird außerdem ein Positionsindex zugeordnet. Soll nun z. B. in einer Tabelle über eine Spalte nach einem Wert gefiltert werden, können die Werte aller weiteren Spalten der Tabelle mit einem einfachen Positions-Lookup ermittelt werden.
Ein Beispiel für eine einfache SQL-Abfrage verdeutlicht das. *(siehe Listing 1)*

Das Ermitteln der Ergebnismenge wird dabei in vier Schritten durchgeführt. *(siehe Abbildung 1)*

Schritt 1: Im Dictionary für die Land-Spalte wird nach der Value-ID für „Deutschland“ gesucht.

Schritt 2: Im Column Store für die Land-Spalte werden alle Einträge ermittelt, deren Value-ID der in Schritt 1 ermittelten Value-ID (00000001) entspricht. Die Positionsindizes der ermittelten Einträge werden in einer Positionsliste gespeichert.

Vorteile des In-Memory Column Store gegenüber einer klassischen Row Store-Datenbank

Schritt 3: Mit Hilfe der Positionsindizes wird die Hauptspeicheradresse für die fehlenden Spaltenwerte der Column Stores für Ort und Geburtsjahr ermittelt. Dabei erfolgt die Berechnung der Adresse nach der Formel: Zieladresse = Basisadresse des Column Stores + Positionsindex * Breite der Value-ID. Dadurch reduziert sich das Auslesen der Value-IDs der weiteren Spalten auf jeweils einzelne sehr effiziente direkte Hauptspeicherzugriffe.

Schritt 4: Abschließend werden für die Value-IDs der Spalten Ort und Geburtsjahr die Werte im jeweiligen Dictionary ausgelesen (Dictionary Lookup). Sind die Werte im Dictionary mit einer fixen Breite gespeichert (z. B. durch Auffüllen mit Null-Bytes bei Zeichenketten), kann auch hier ein direkter Hauptspeicherzugriff erfolgen. Die Berechnung der Adresse im Hauptspeicher wird dann wie folgt durchgeführt: Zieladresse = Basisadresse des Dictionary + Value-ID * Breite der Werte im Dictionary

Fazit

Beim Zugriff auf mehrere Spalten einer Tabelle müssen diese also nicht mit aufwendigen Join-Operationen miteinander verknüpft werden. Die Dictionary Compression ermöglicht den sehr effizienten direkten Zugriff im Hauptspeicher per Positions-Lookup und Dictionary Lookup.

Vorteile des In-Memory Column Store gegenüber einer klassischen Row Store-Datenbank

Listing 1:

```
select land
     , ort
     , geburtsjahr
  from d_kunde
 where land = 'Deutschland';
```

Abbildung 1: