(Wichtige Vorbemerkung:
An dieser Stelle sind Informationen zusammengetragen, wie sie im Rahmen der Oracle Open World im September 2013 bereits über Vorträge und Pressemitteilungen veröffentlicht wurden. Da die InMemory Database Option zu dem Erstellungszeitpunkt dieses Textes noch nicht durch Oracle freigegeben ist, kann man sich auf Features und Eigenschaften, die in diesem Text beschrieben sind, nicht berufen. Dieser Text ist rein informativ und verpflichtet zu keinerlei Ansprüchen. Der Text beinhaltet weiterhin Aussagen über mögliche Konzepte, die mit der InMemory Option nach Meinung des Autors machbar sind).

Oracle Database In-Memory Option

Mit der Ankündigung der InMemory Database Option reagiert Oracle auf Marktbewegungen, die schon seit einiger Zeit zu beobachten waren: Zum einen konnte SAP durch seine HANA – Aktivitäten sehr viel Aufmerksamkeit auf sich ziehen, zum anderen haben sich gerade in dem Segment des Data Warehouse mit sog. analytischen Datenbanken oder Columnar Based Databases neue Produkte ins Spiel gebracht. Auch wenn all diese technischen Entwicklungen in vielen Fällen nicht die nötigen Verbesserungen einbrachten (weil sich durch Tools und Technik konzeptielle oder architektonische Schwächen kaum beheben lassen), gibt es eine Menge Diskussion im Feld der Datenbanken und des Data Warehouse. Wenn jetzt Oracle ebenfalls hier eine Lösung anbietet ist das vor allem für die hohe Anzahl der Oracle-Kunden interessant,

- die ihr zentrales Data Warehouse auf Oracle-Basis aufgebaut haben,
- die technisch ein engeres Zusammenspiel zwischen OLTP und Data Warehouse Systemen erreichen wollen,
- die keine separaten Datenbanken in dem Data Mart-Umfeld, sondern eine integrierte zentrale Lösungen bevorzugen.

Diese Firmen haben vor allem die Chance, ihre bestehende Umgebung ohne aufwendige Änderungen zu einer Best-In-Class-Lösung zu *evolutionieren* ohne ihre gewohnte Basistechnologie zu verlassen.

Anforderungen im Data Warehouse (DWH)

Es klingt wie eine Attitude: Data Warehouse – Systeme müssen in den letzten Jahren in immer kürzerer Zeit Informationen aus zunehmend komplexeren Geschäftsprozessen extrahieren, aufbereiten und verständlich präsentieren. Aber es ist tatsächlich so.
Das hängt vor allem mit der wachsenden Bedeutung der Systeme zusammen. Ohne funktionierende Warehouse-Systeme können Unternehmen vor dem Hintergrund der sich immer rascher wandelnden Rahmenbedingungen nicht mehr gesteuert werden.
Das hat zur Folge:

- Batch-Zeiten werden immer kürzer
- Durchlaufzeiten zur Aufbereitung der auszuwertenden Daten müssen kompakter und schneller sein
- Die Nähe zu operativen Systemen wird immer größer
    - Die Granularität wird feiner
    - Datenmengen wachsen

Nachvollziehbar, dass auch die Performance-Anforderungen der Anwender an Warehouse-Systeme steigen: Aussagen wie „Wir brauchen Antwortzeiten von unter 2 Sekunden“ bei einem Datenvolumen von mehreren Terabyte und mehreren 100 Benutzern auf dem System sind keine Seltenheit.

Parallel dazu haben auf der Hardware-Seite Verschiebungen des Kosten-/Leistungsverhältnissen stattgefunden. Preise sinken bei wachsender Leistung, junge Techniken wie Flash-Speicher aber auch schnelle Netz-Protokolle wie Infiniband entwickeln sich zum Standard.
Die ehemals sehr kostspielige Ressource MEMORY oder RAM ist so günstig wie noch nie. Erwartet wird, dass die Preise für verbaubaren Speicher weiter fallen und die Hauptspeicherausbaugrenzen der Server-Maschinen scchieben sich nach oben.

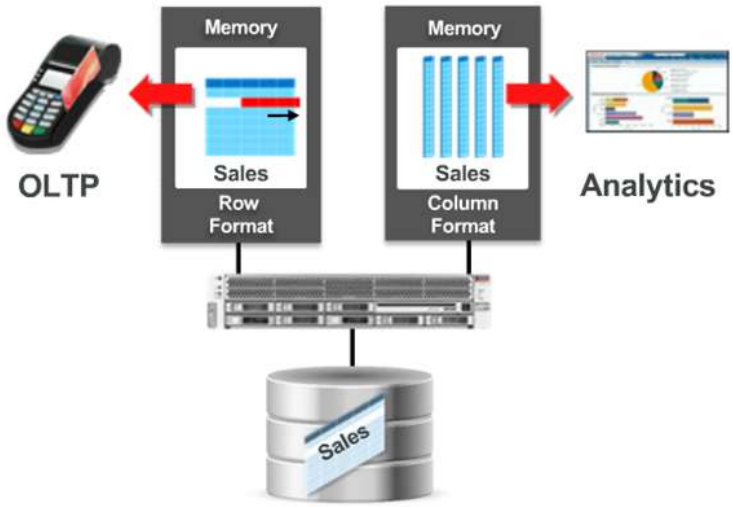
Row-Based und Column-Based Speicherung

In dem Datenbanken-Segment gibt es schon seit längerem zwei alternative Speicherformate, die Vor- und Nachteile für jeweils entgegen gesetzt OLTP- und Warehouse-Anwendungen mit sich bringen:

- **OLTP – Systeme** arbeiten am besten mit der **Satz-bezogenen Speicherung** der Daten, d.h. alle Spalten eines Satzes finden sich in einem bzw. mehreren zusammenhängenden Blöcken wieder. Das Lesen eines einzelnen Satzes erfordert wenig physiche IOs und ist sehr schnell.
  -> ROW Based Operation
  OLTP-Systeme arbeiten mit einer Column-basierten Speicherung langsamer, da für Einzelsatz-Operationen wesentlich mehr Speicherblöcke angefasst werden müssen.
- **DWH- Systeme** ziehen eher Vorteile aus der **Spalten-orientierten Speicherung**. D. h. die Columns der Sätze einer Tabelle sind zusammenhängend in einem bzw. mehreren verketteten Blöcken gespeichert. DWH-Systeme lesen seltener nur einzelne Sätze, sondern abhängig von Abfragekriterien lesen sie komplette Datenbereiche, also viele Sätze einer Tabelle zusammenhängend. Hinzu kommt, dass oft nur einzelne Spalten eher als viele Spalten einer Tabelle benötigt werden.
  -> Column-Based-Operations

Oracle InMemory Option

Vor der Einführung der Oracle Database *In Memory Option* musste man sich zwischen den beiden eben vorgestellten Speicherformaten entscheiden. Die Column-Based-Speicherung stand nur für Exadata-Systeme im Rahmen der Hybrid Columnar –Compression zur Verfügung.
Mit der neuen **InMemory Option** liegen Tabellen sowohl mit Satz-orientierter als auch mit Spalten-orientierter Speicherung vor. So können die Vorteile beider Speichervarianten genutzt und Nachteile umgangen werden.



Während für OLTP-Systeme eine Tabelle nach wie vor im ROW-Based-Format klassisch gespeichert und bearbeitet wird, steht gleichzeit eine Kopie der Daten im Hauptspeicher (InMemory) spaltenorientiert und komprimiert bereit.
Die spaltenorientierte InMemory-Version der Tabelle entsteht entweder während der Startphase der Datenbank oder alternativ nachdem eine Tabelle zum ersten Mal durch einen Benutzer gelesen wird.
Für die spaltenorientierte Version gibt es kein Logging. Änderungen an den Daten werden über den klassischen Änderungsvorgang der Datenbank vorgenommen. D. h. INSERTS und UPDATES finden wie bisher in der SGA statt, Schreibprozesse persistieren die Änderungen im Hintergrund und synchronisieren jetzt zusätzlich die InMemory-Version der Tabelle.
Der Zeitaufwand für diese zusätzliche Synchronisierung ist kaum messbar, da die Operation ohne Netzlatenz und ohne Platten-IO stattfindet. Oracle nutzt hier die SGA-Mechanik, bei der zu ändernden und zu lesenden Daten sich sowieso schon im Hauptspeicher befinden.
Das primäre Ziel „schnell und InMemory“ lesen zu können ist somit erfüllt. Update-fähig und persistiert sind die Daten automatisch. Damit handelt es sich um eine vollwertige InMemory-Datenbank, plus alle technischen Features und Möglichkeiten der klassischen Oracle Datenbank.

Abfragebeispiele

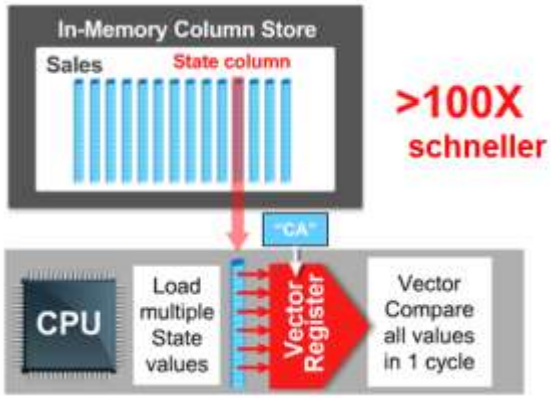
1. Abfrage über Werteeinschränkung pro Spalte in einer Tabelle

Das erste Beispiel ist eine Abfrage wie die folgende:

-> Liefere den Umsatz in dem Land „XX“,

Die Abfrage beschränkt die Werte einer Spalte einer Tabelle.
Das Merkmal einer solchen Abfrage ist ihr Bereichs-bezogener Charakter. In einer Länder-Spalte befinden sich denormalisierte Informationen. Es wird eine gewisse Ergebnismenge pro WHERE-Bedingung zurück geliefert (im Gegensatz zu einzelnen wenigen Sätzen). Hier handelt es sich um eine DWH-typische Abfrage.
In der klassischen Datenbank-Technologie würde man einen Bitmap-Index als präferiertes Hilfsmittel für eine Spalte mit denormaliserten Werten (wie hier „Land“) wählen. Die Abarbeitung der Spaltenwerte einer InMemory-Tabelle ähnelt diesem Vorgehen, da auch die InMemory-Ablage der Spaltenwerte optimiert ist, d. h. nicht jeder Spaltenwert wird einzeln „InMemory“ gespeichert, sondern über einen

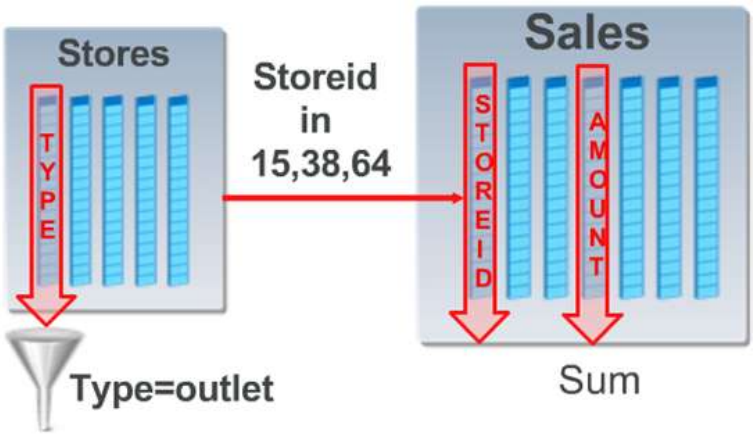
Pointer-Algorithmus im Speicher abgelegt. Im Ergebnis bedeutet dies, dass stark denormalisierte Spaltenwerte besonders von der Technologie profitieren.
-> Daher erreicht man auch diese Art einer Abrage eine Performance-Verbesserung von über 100% gegenüber der Nicht-InMemory Technik.



2. Abfragen über Tabellengrenzen hinweg (JOINs)

Sind mehrere Tabellen von der Abfrage betroffen , findet ein mehrstufiges Columns-Scan statt.
Wenn in einer Umsatztabelle (z. B. Fakten-Tabelle im Star Schema) neben der Umsatzspalte lediglich eine Schlüsselspalte für „Stores“ (Filialen, Niederlassungen) vorhanden ist, aber weitere Informationen zu dem Store (z. B. Filialname, Ort etc.) in einer weiteren Tabelle (Dimension) zu finden sind, werden InMemory separate Scans nacheinander über die jeweiligen Tabellen und deren Spalten durchgeführt.

InMemory wird in dem Beispiel zunächst die Stores-Tabelle gefiltert, um die relevanten Stores festzuhalten. Dieses Zwischenergebnis wird dann über das Bloom-Filtering-Verfahren InMemory auf die i. d. R. wesentlich größere Sales-Tabelle (Fakten-Tabelle) als Filter angewendet.



Solche mehrstufigen und komplexeren Join-Scans können bis zu 10 mal schneller abgearbeitet werden, als mit der klassischen Technik.

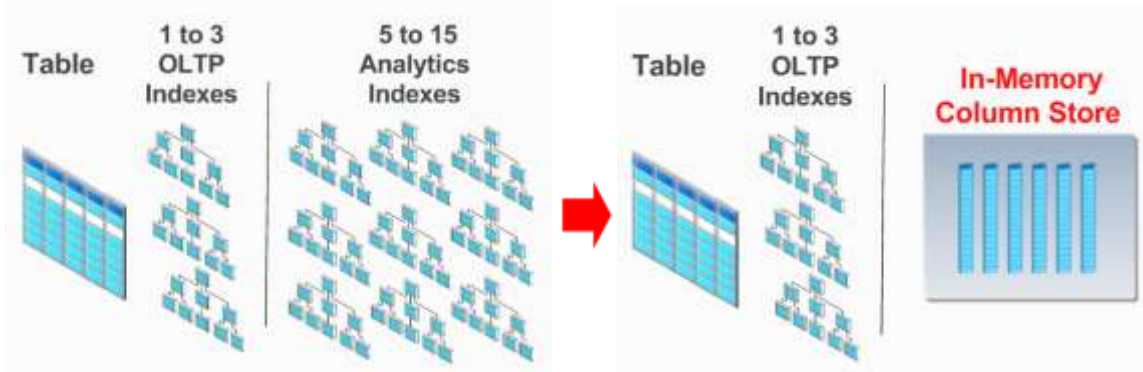
3. Aggregationen und vorkalkulierte Reporting-Strukturen

Sollen bekannte Abfragen (Standard-Reports) beschleunigt werden, so kann das System dynamisch InMemory Report – Strukturen pflegen und verwalten. Meist enthalten solche Abfragen Aggregationen in

Form von „sum() group by“ Konstrukten. Der Optimizer der Datenbank erkennt solche Konstrukte und kann im laufenden Betrieb InMemory-Arrays mit den aggregierten Werten bilden.
Entsprechende Abfragen werden um bis zu Faktor 20 schneller.

Weniger Indexe

Viele der zur Beschleunigung bekannter Abfragen eingeführten Indexe sind nicht mehr nötig. Das minimiert den Verwaltungsaufwand und spart Platz. Hatte man bislang für Abfragen, für die noch kein Index definiert war, mit längeren Antwortzeiten zu rechnen, so sind auch diese „unvorbereiteten“ Abfragen schneller.
Für Mischsysteme, d. h. für Systeme mit denen ein OLTP und ein DWH – Betrieb in einem praktiziert wird, profitieren die OLTP-Anwendungen von dem Wegfall vieler Index-Definitionen, die für die DWH-Optimierung notwendig waren.



Leicht implementierbar

Die neue Option integriert sich nahtlos mit der bestehenden Oracle Datenbank. Das bedeutet als Konsequenz:

- Datenbankanwendungen müssen nicht umgeschrieben werden
- OLTP- und Data Warehouse-Anwendungen können von der neuen Technik profitieren
- Die Implementierung der Option ist sehr einfach

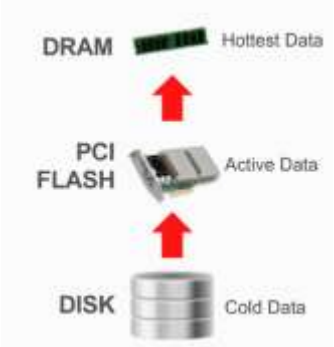
Über einen Initialisierungsparameter legt man die Größe des „InMemory-Hauptspeicherbereichs“ fest:

Inmemory_size = XXX GB

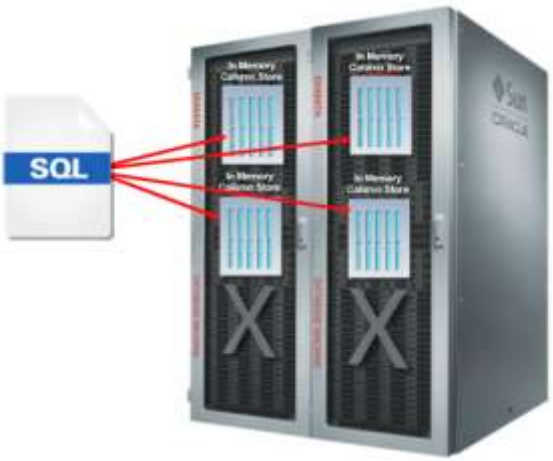
Dann werden die Objekte markiert, die InMemory vorzuhalten sind. Man kann Tabellen aber auch einzelne Partitionen „InMemory-fähig“ manchen mit:
alter table | partition *name* ***inmemory;***

Skalierung und Memory-Grenzen

Bei der Größe heutiger Warehouse-Datenhaltungen ist trotz Komprimierung eine Memory-Limitierung zu vermuten. Die Oracle-Datenbank kann in solchen Fällen jedoch weniger häufig genutzte Daten auf weitere Datenträger ihrer sog. Storage-Hierarchie auslagern und dieses Auslagern dynamisch verwalten. Zur Storage-Hierarchie gehören nebem dem RAM (InMemory) auch PCI-Flash-Karten und wenn diese nicht ausreichen sollten klassicher Spindel-Storage, der über ein schnelles Netzwerk angeschlossen ist. Dieses Technologie gibt es bereits seit dem Datenbankrelease 11.



In einem RAC-Environement steht nicht nur der Hauptspeicher der einzelnen Server separat zur Verfügung. Passt eine sehr große Tabelle nicht in den Speicher eines Serves, so kann die Datenmenge in die Hauptspeicher aller weiteren am RAC-System beteiligter Server verteilt werden. Das System erlaubt es, diese so verteilten Daten zusätzlich noch parallelisiert über auf alle Server verteilte Abfrage-Prozesse abzufragen.



Oracle nutzt hierfür das bereits unter 11.2 eingeführte InMemory-Parallel-Execution-Feature. Unterstützt werden alle DML-Operationen, die auch bislang schon parallelisierbar waren.

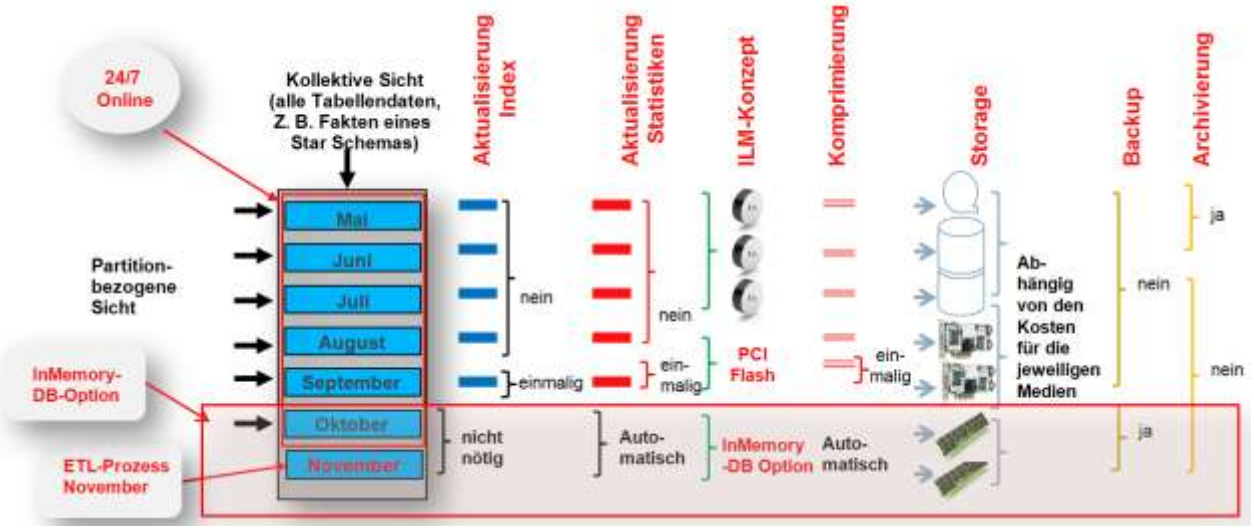
Dadurch, dass man das InMemory-Feature sowohl auf Tabellen- als auch auf Partitionsebene anwenden kann, ziehen auch bisher praktizierte „Information Lifecycle-Konzepte“.
Das bedeutet, man wird z. B. große Fakten-Tabellen nicht komplett in den Hauptspeicher legen, sondern nur die am häufigsten genutzten Daten, also meist die jüngsten Partitionen. Damit ist dieses Konzept von Oracle den übrigen bereits bestehenden InMemory- DBs anderer Hersteller überlegen. Denn das Oracle-Konzept ist auch bei sehr großen Datenmengen praktizierbar. Viele einmalige Features, die die Oracle-Datenbank zur führenden Datenbank gemacht haben, stehen auch InMemory zur Verfügung.

Verwaltungsprozesse im Data Warehouse

Es gibt bereits seit längerem Empfehlungen für die optimale Verwaltung eines Oracle Data Warehouse. Diese bauen sehr stark auf das Partitioning-Feature. Durch die InMemory-Option können diese Empfehlungen noch erweitert werden.
Verwaltungsaktivitäten, die bislang für aktive Partitionen durchzuführen waren, verschieben sich jetzt auf einmalige Massnahme im Rahmen des ILM (Information Life Cycle Managements).
Das sind z. B:

- Aktualisierung von Indexen
- Aktualisieren von Statistiken
- Komprimieren

Diese Massnahmen sind nur dann nötig, wenn man klassische Storage-Medien verwendet. Das sind Spindel-Platten und PCI-Flash-Karten für den Fall, dass man eine expliziete Speicherzuordnung für Partitionen vornimmt und nicht die Datenbank dynamisch entscheiden lässt.



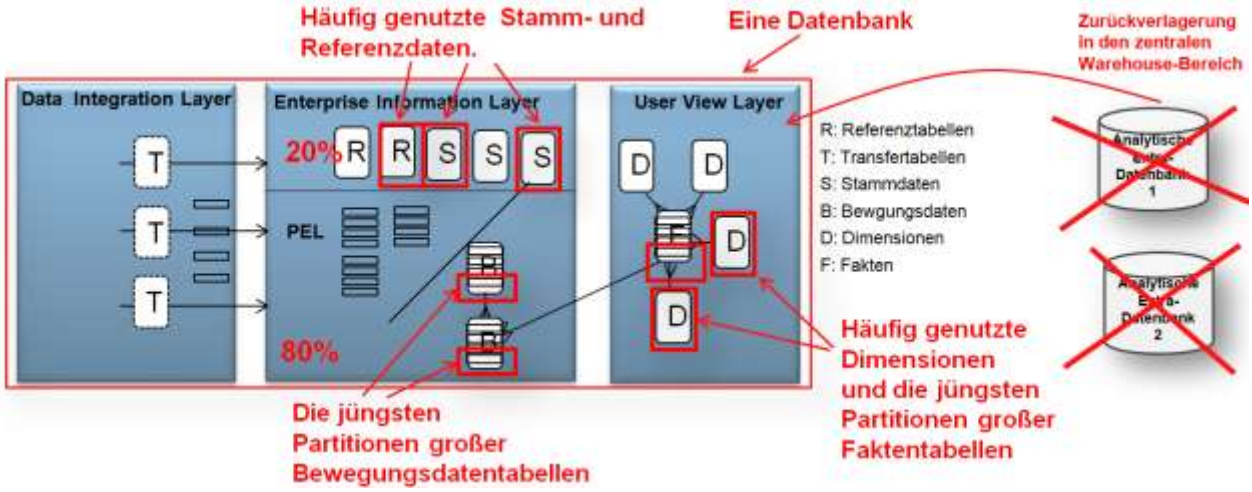
Brauchen wir dann noch Star Schemen?

Star Schemen werden oft als Hilfsmittel zur Performance – Optimierung missverstanden. Waren sie Mitte der 1990er Jahre u. a. auch zur Performance-Optimierung gedacht, so werden sie heute längst primär dazu verwendet, um Sachzusammenhänge in einer multidimensionalen Form innerhalb einer relationalen Datenbank darzustellen. Dimensionen eines Star Schemas repräsentieren Geschäftsobjekte wie sie von Fachanwendern verstanden werden können. Dimensionen strukturieren potentielle Abfragen indem sie

z. B. Drillpfade und Aggregationslevel aufzeigen. Technisch kann man performante Abfragen heute auch anders erreichen.
Star Schemen profitieren in besonderem Mass von der InMemory Option. Ihre Dimensions-Column-Werte sind z. T. stark denormalisiert und Joins und Aggregationsabfragen kommen in Star Schemen besonders häufig vor. Hier passen die oben beschriebenen Abfragebeispiele mit dort genannten Beschleunigungsfaktoren.

Auswirkungen auf das Schichtenmodell im Data Warehouse

Das klassiche 3-Schichten-Schema muss an dieser Stelle nicht noch einmal dargestellt werden und wird als bekannt vorausgesetzt. Die InMemory-Option kommt diesem Architektur-Konzept besonders entgegen.
Bei Oracle gab es immer die Empfehlung, alle Schichten eines Data Warehouse innerhalb einer einzigen Datenbank zu realisieren. Einige Oracle-Kunden haben sich jedoch gegen dieses Konzept entschieden, und den User View Layer in separate Datenbanken verlagert, mit all den Konsequenzen wie zusätzliches Kopieren der Daten, separate Verwaltung und Zusatz-Hardware, Komplexität der Systeme usw.
Begründet hat man diese Vorgehensweise mit der fehlenden Abfrageperformance im User View Layer. bzw. dem Wunsch, das Kern-Data Warehouse vor potentiellen „ungeschickt“ formulierten Benutzerabfragen „zu schützen“.
Nutzt man die InMemory- Option, so fallen diese Argumente weg.

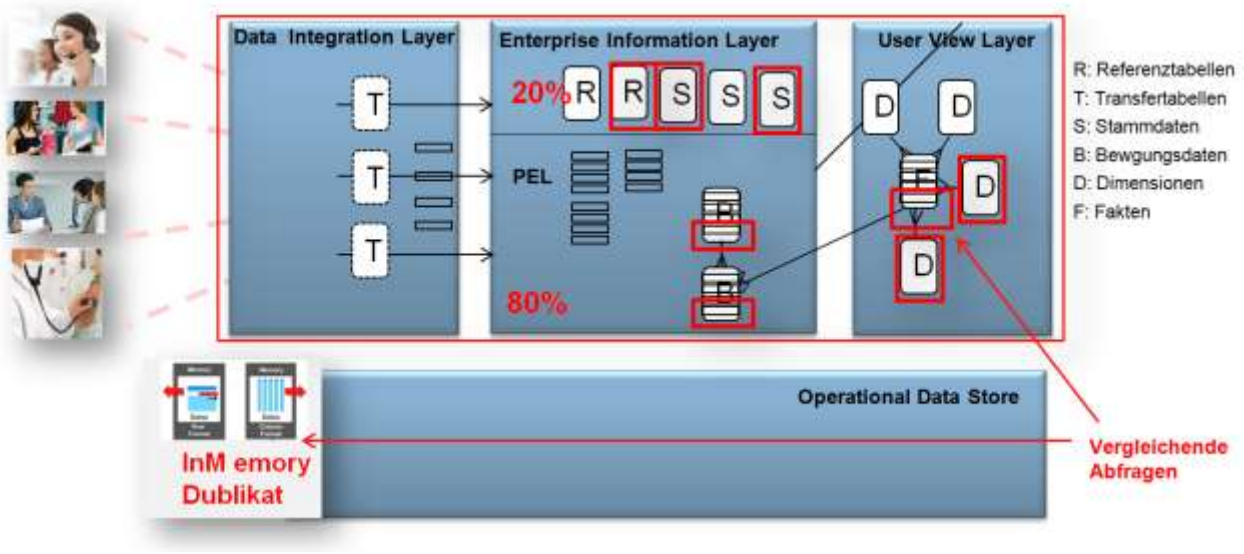


Das ideale Konzept alle Auswertungs-relevanten Daten zentral und an einer Stelle vorzuhalten kann mit der InMemory-Option erst recht angewendet werden. Die InMemory-Option erreicht in dem Data Warehouse den größten Nutzen je dichter die Daten für unterschiedliche Zwecke beieinander liegen.
Das bedeutet:

- Gleichbleibende und durchgängig hohe Performancewerte für alle (!) Benutzer und alle Datenbereiche
- Einheitliche und stimmige Kennzahlen über Sachgebietsgrenzen hinaus (-> unternehmensweit)
- Einsparung an Kosten (Hardware und Verwaltung)
- Einmalig gesichertes System
- Reduzierte ETL-Aufwände

Rücken OLTP und DWH Systeme noch enger zusammen?

Warehouse Systeme sind in den vergangenen Jahren immer mehr auch operativ eingesetzt worden. Zudem rückten OLTP und DWH-Systeme oft sehr eng zusammen und der Wunsch nach einer gleichzeitigen, parallelen und sich ergänzenden Auswertung in OLTP und DWH Systemen ist von Anwendern regelmäßig formuliert worden.
Bei der Umsetzung dieser Ziele blieben in der Vergangenheit leider auch einige sinnvollen Konzepte, z. B. das Schichtenmodell im Data Warehouse, auf der Strecke.
Mit der InMemory Option kann man jetzt zu 100 % Warehousing-Konzepte fahren und gleichzeitig Zugriff auf operative Tabellen der Vorsysteme ermöglichen. Operative Tabellen in den Vorsystemen können „InMemory“ markiert werden und stehen als eine Art Clone-Tabelle im Hauptspeicher für das Data Warehouse zur Verfügung. Man kann sie für Analysezwecke abfragen, ohne die operativen Anwendungen zu beeinflussen.
Nimmt man das Konzept des Operational Data Store hinzu, so erhält man ein sehr praktisches und zukunftsorientiertes Szenario für klassisches Data Warehousing und (Near-) Realtime Auswertungen mit der größt möglichen Nähe zu den Vorsystemen. In dem letzten Fall sogar ohne ETL-Prozess.



Autor: Alfred Schlaucher, Oracle Deutschland, Oktober 2013