Performanter Aufbau von Materialized Views im ETL-Prozess

Reinhard Mense
ARETO Consulting
Köln

Schlüsselworte:

DWH, Data Warehouse, ETL-Prozesse, Performance, Laufzeiten, Materialized Views, Refresh

Einleitung

Materialized Views sind im Data Warehouse (DWH) ein einfaches Mittel um aggregierte Daten für Abfragen und Berichte zu Verfügung zu stellen. Durch die Query Rewrite-Funktionalität für Materialized Views kann der Optimizer der Oracle Datenbank SQL-Abfragen ggf. automatisch umschreiben, so dass z. B. statt auf sehr große Fakttabellen auf weniger umfangreiche, aggregierte Tabellen zugegriffen wird. Für performante Abfragen stellen Materialized Views deshalb häufig ein unverzichtbares Mittel dar.

Werden die Daten der den Materialized Views zugrunde liegenden Tabellen, den Basistabellen, geändert, sind auch die Daten der Materialized View zu aktualisieren. Dieser Vorgang wird als Refresh bezeichnet und kann abhängig von der Datenmenge sehr aufwendig und zeitraubend sein. Der Refresh der Materialized Views erfolgt in der Regel innerhalb des ETL-Prozesses und sollte möglichst effizient sein, um das für den ETL-Prozess zur Verfügung stehende Zeitfenster nicht zu überschreiten. Als größte Herausforderung erweist sich dabei meist, dass die historischen Datenmenge im DWH stetig zunimmt, und somit die Gefahr besteht, dass der Refresh ebenfalls zunehmend mehr Zeit benötigt.

Unterschiedliche Refresh-Verfahren und das Design der Materialized Views haben erheblichen Einfluss auf die Laufzeit des Refresh. Im Folgenden werden Optimierungsmöglichkeiten aber auch Grenzen anhand für Data Warehouse Systeme typischer Szenarien aufgezeigt. Ziel ist, für die geschilderten Szenarien möglichst konstante – im Sinne von unabhängig von der historischen Datenmenge – Laufzeiten für den Refresh zu erzielen.

Refresh-Verfahren

Für den Refresh stellt die Oracle Datenbank die Package-Prozedur DBMS_MVIEW.RERESH zur Verfügung. Diese erlaubt neben einem Complete Refresh, bei dem der Inhalt der Materialized View vollständig neu aufgebaut wird, auch einen inkrementellen Refresh, der von Oracle als Fast Refresh bezeichnet wird. Beim Fast Refresh wird auf einen vollständigen Aufbau der Materialized View verzichtet. Stattdessen werden nur die Änderungen der Basistabellen in die Materialized View übernommen. Damit der Fast Refresh möglich ist, muss eine Vielzahl von Bedingungen erfüllt sein, für die an dieser Stelle auf die Oracle Dokumentation verwiesen sei.

Die Bezeichnung Fast Refresh ist etwas irreführend, da der Fast Refresh nicht immer schneller ist als ein Complete Refresh. Insbesondere bei umfangreichen Änderungen in den Basistabellen der Materialized Views kann ein Fast Refresh mehr Zeit benötigen als ein Complete Refresh.

Complete Refresh

Beim Complete Refresh hängt der optimale Einsatz unter anderem von der richtigen Parametrisierung der Package-Prozedur DBMS_MVIEW.REFRESH ab. Der Parameter atomic_refresh legt z. B. fest, wie die alten Daten der Materialized View gelöscht und die neuen Daten in die Materialized View eingefügt werden.

Während bei atomic_refresh = true zunächst ein DELETE und anschliessend ein konventionelles INSERT … SELECT ausgeführt wird, wird bei atomic_refresh = false ein TRUNCATE TABLE mit anschliessendem INSERT /*+ APPEND */ … SELECT ausgeführt. Wie die folgende Abbildung zeigt, ist das zweite Verfahren bzgl. Der Laufzeit dem ersten Verfahren deutlich überlegen. Bemerkenswert ist, dass atomic_refresh = true seit Oracle 10g als Default-Wert vorgegeben ist. Für einen performanten Complete Refresh ist also explizit der Wert false anzugeben.

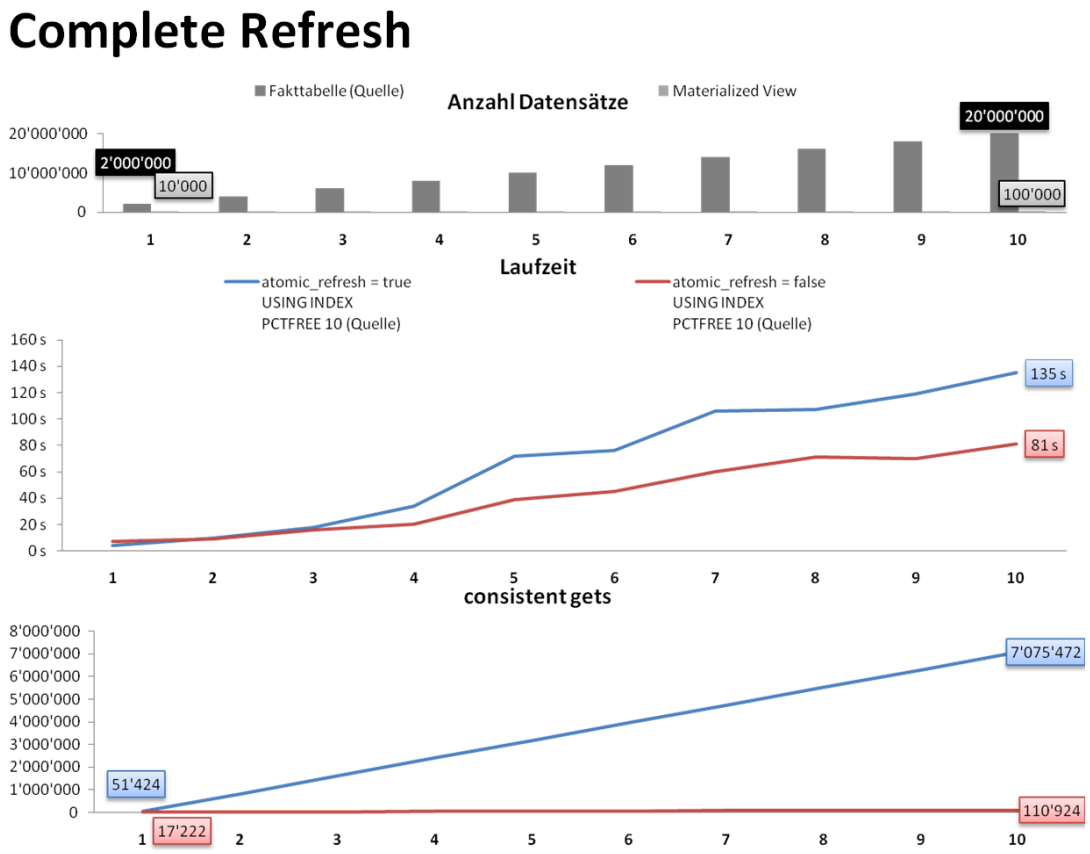

Abb. 1: Auswirkungen des Parameters atomic_refresh beim Complete Refresh

Beim Erstellen der Materialized View wird standardmäßig automatisch auch ein Function Based Index für die Materialized View erzeugt. Dieser dient zu Beschleunigung des Fast Refresh. Wird für die Materialized View ausschließlich ein Complete Refresh durchgeführt, kann auf den Function Based Index verzichtet werden. Damit wird die relativ aufwendige Pflege des Function Based Index vermieden und Laufzeit beim Refresh eingespart. Die Erzeugung des Function Based Index wird durch die Angabe der Klausel USING NO INDEX beim Erstellen der Materialized View verhindert.

Eine weitere Möglichkeit zur Beschleunigung des Refresh besteht darin, die Zahl der zu lesenden Blöcke aus den Basistabellen zu reduzieren. Eine einfache Möglichkeit ist den Wert für PCTFREE für die Basistabelle auf 0 zu setzen. Damit wird in den Datenblöcken der Basistabelle kein Platz für etwaige Updates reserviert, so dass beim Refresh der Materialized View weniger Datenblöcke der Basistabelle gelesen werden müssen. Da für Fakttabellen typischerweise keine Updates durchgeführt werden, gleichzeitig aber aus diesen die größten Datenmengen für den Aufbau der Materialized Views gelesen werden, stellen diese oft ideale Kandidaten für Tabellen mit PCTFREE 0 dar. Abbildung 2 zeigt die Auswirkungen von USING NO INDEX und PCTFREE 0.

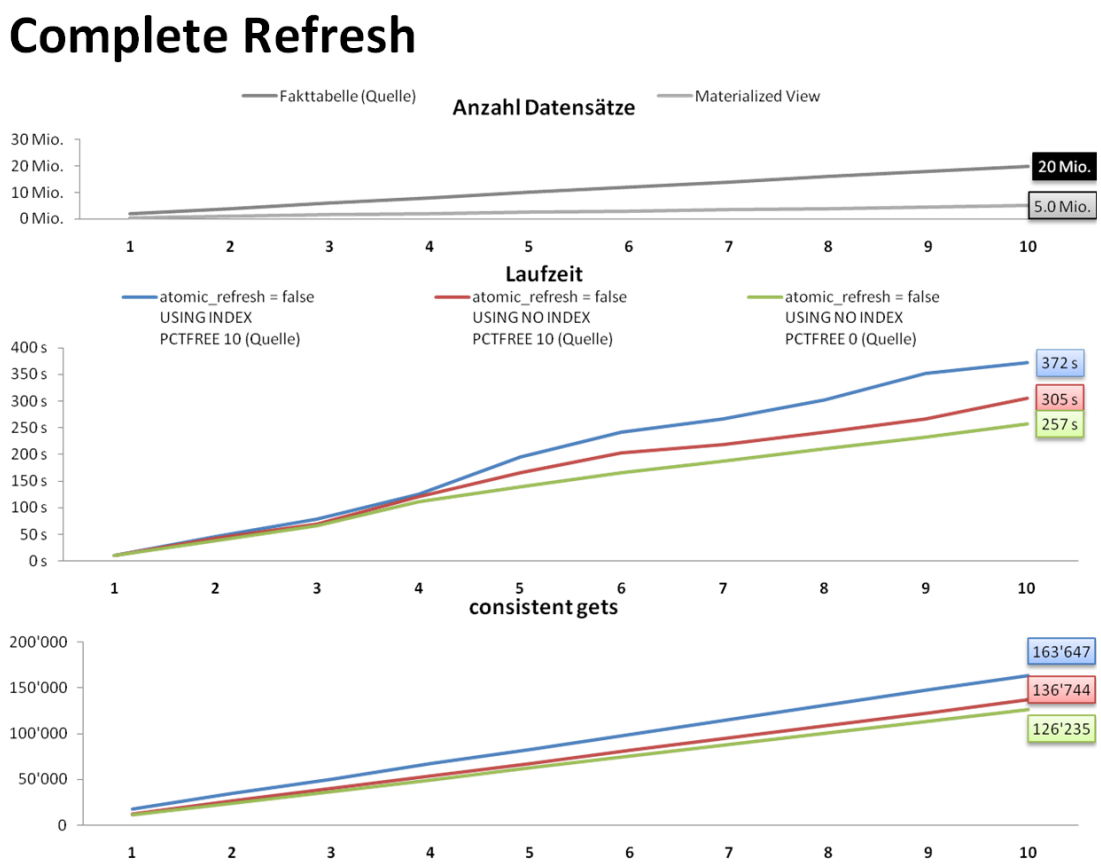

Abb. 2: Auswirkungen von USING NO INDEX und PCTFREE 0 beim Complete Refresh

Sowohl in Abbildung 1 als auch in Abbildung 2 ist zu erkennen, dass die Laufzeit des Complete Refresh mit steigenden Datenvolumen in den Basistabellen zunimmt. Um konstante Laufzeiten für den Materialized View Refresh zu erzielen, ist deshalb ein Complete Refresh zu vermeiden, da dabei die kompletten historischen Daten der Basistabellen gelesen werden müssen. Aber auch bei einem Fast Refresh kann ein unerwünschter Zugriff auf die historischen Daten erfolgen.

Fast PCT Refresh

Materialized Views im DWH basieren in der Regel auf einer Fakttabelle und gegebenenfalls per Join verknüpfter Dimensionstabellen. Den einfachsten Fall stellt eine Materialized View dar, die nur auf eine Fakttabelle basiert. In diesem Fall wird die Aggregation der Daten durch das Weglassen der Fremdschlüssel für einzelne Dimensionen erreicht. Wird eine solche Materialized View analog zu der

Fakttabelle partitioniert, kann der Refresh per Fast PCT durchgeführt werden. PCT steht dabei für Partition Change Tracking und bedeutet, dass die Datenbank anhand von internen Log-Informationen automatisch die Partitionen der Fakttabelle erkennt für die die Daten geändert wurden. (Wichtig: Dafür ist kein Materialized View Log auf der Fakttabelle notwendig!) Da die Materialized View analog zur Fakttabelle partitioniert ist, werden beim Refresh auch nur die betroffenen Partitionen geändert. Geht man davon aus, dass beim ETL-Prozess keine historischen Daten in der Fakttabelle geändert werden, werden beim Refresh immer nur die Daten einer Partition erneuert. Somit können, wie in Abbildung 3 gezeigt, konstante Laufzeiten für den Refresh einer solchen Materialized View erzielt werden.

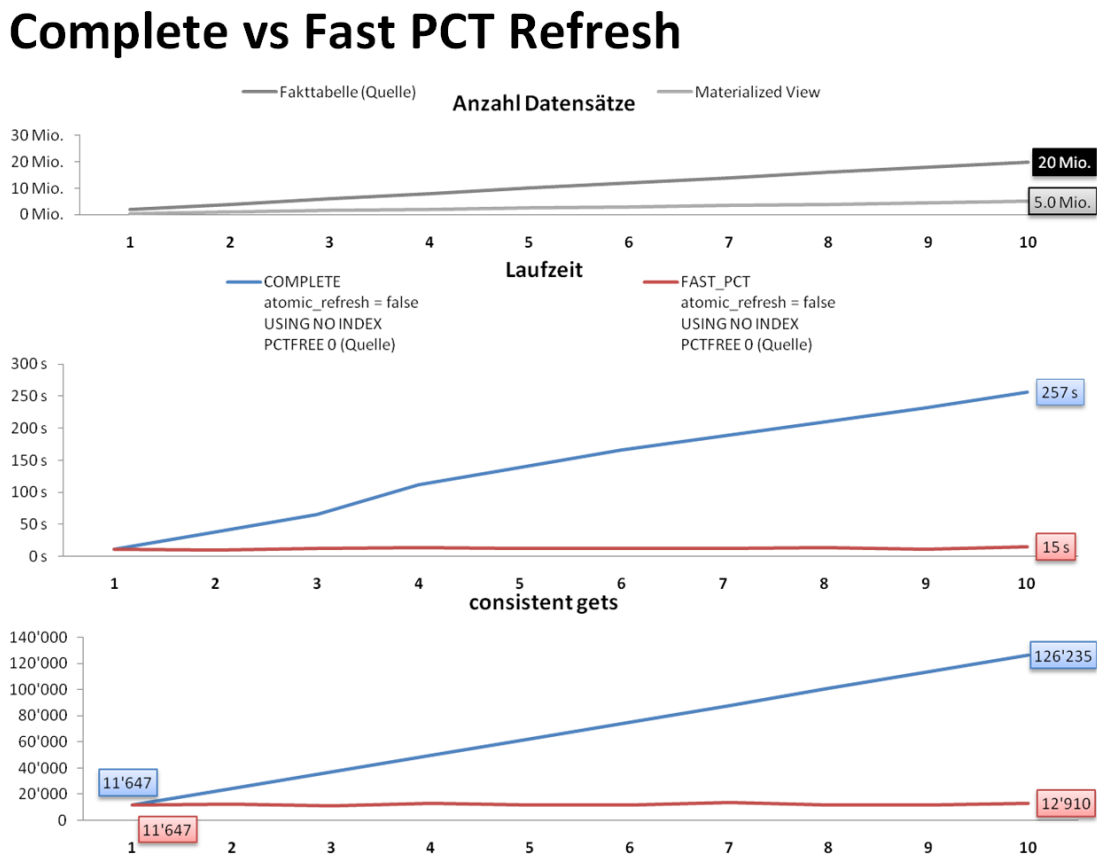

Abb. 3: Vergleich von Complete Refresh und Fast PCT Refresh

Fast Refresh

Schwieriger wird es, wenn die Materialized View neben der Fakttabelle auch auf per Join verknüpfte Dimensionstabellen basiert. Grundsätzlich kann man in diesem Fall zwar einen Fast Refresh basierend auf Materialized View Logs einsetzen. Allerdings garantiert dies keine konstanten Laufzeiten, da beim Refresh in allen Partitionen der Materialized View nach den zu aktualisierenden Datensätzen gesucht wird. Der Fast Refresh hat aber, sofern die geänderte Datenmenge nicht zu groß ist, deutliche Performance-Vorteile gegenüber einem Complete Refresh.

Für den Fast Refresh kann es für die Laufzeit von Vorteil sein den standardmäßig von der Datenbank für die Materialized View erzeugten Function Based Index zu verwenden (Klausel USING INDEX

beim Erstellen der Materialized View). Dabei zeigt sich jedoch häufig, dass der Nutzen nur bei sehr geringen Datenänderungen gegeben ist.

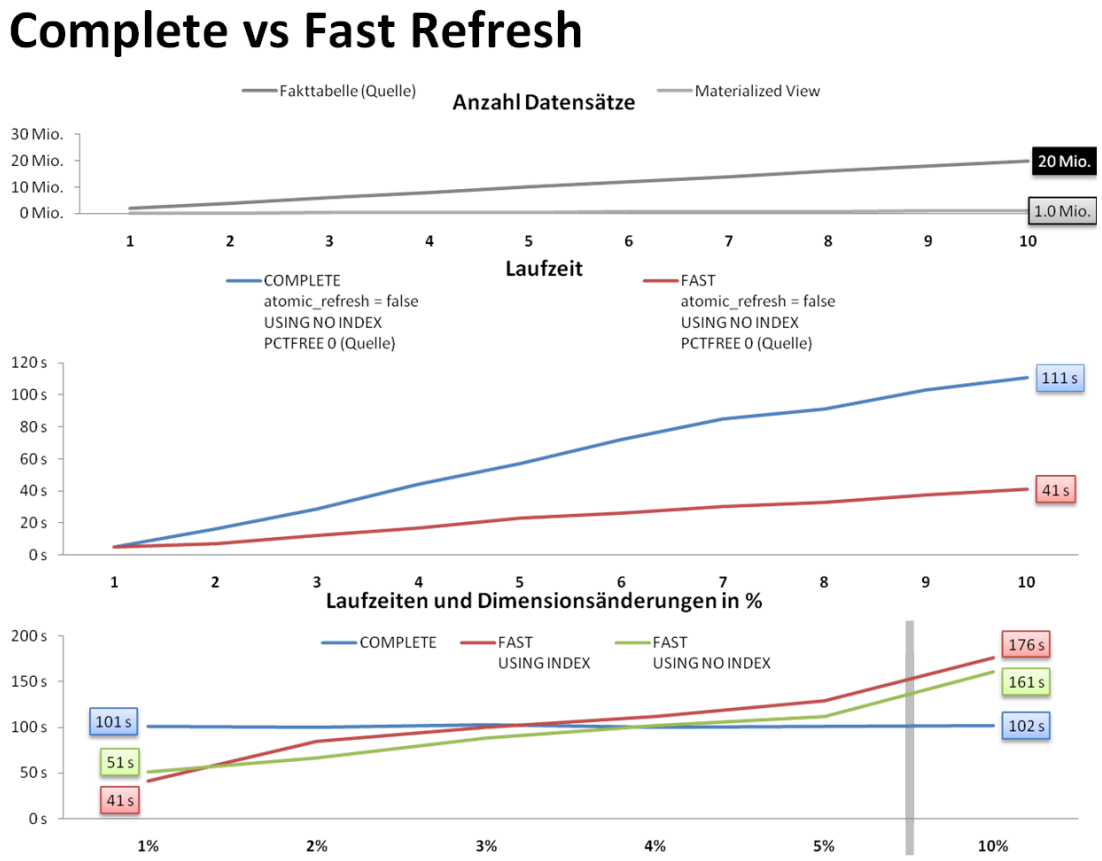

Abb. 4: Vergleich von Complete Refresh und Fast Refresh

Manueller Refresh

Die Datenbank unterscheidet beim Refresh nicht, ob es sich bei den beteiligten Dimensionen um historisierte Dimensionen (SCD Typ 2) oder um nicht historisierte Dimensionen (SCD Typ 1) handelt. Sind ausschließlich historisierte Dimensionen beteiligt, werden bei Änderungen in der SCD Typ 2 stets neue Datensätze erzeugt und bestehende Datensätze bleiben unverändert. Bei den bestehenden Datensätzen wird lediglich das Ende des Gültigkeitszeitraums verändert.

Sowohl neu eingefügte als auch die im Gültigkeitszeitraum angepassten Datensätze werden im Materialized View Log aufgenommen. Beim Fast Refresh der Oracle Datenbank wird für alle im Materialized View Log vorhandenen Einträge über die gesamte Materialized View nach veränderten Datensätzen gesucht. Mit zunehmenden historischen Datenvolumen nimmt dieser Vorgang immer mehr Zeit in Anspruch, so dass keine konstanten Laufzeiten für den Refresh erzielt werden können.

Durch ein manuelles Refresh-Verfahren kann man konstante Laufzeiten für den Refresh erzielen, sofern neben den Fakttabellen nur SCD Typ 2 als Basistabellen für die Materialized View dienen. Geht man davon aus, dass der ETL-Prozess keine historische Daten verarbeitet, werden weder historische Daten in der Fakttabelle noch in den historisierten Dimensionen geändert. Damit ist beim Refresh der Materialized View auch lediglich die aktuelle Partition zu aktualisieren. Erstellt man die Materialized View auf einer Prebuilt Table, kann man diese im ETL-Prozess manuell mit einem

Partitionsaustausch aktualisieren. Für die kurze Dauer des Partitionsaustauschs muss die Materialized View gelöscht werden und anschließend wieder neu erstellt werden. Da die Materialized View für eine Prebuilt Table erzeugt wird, bleibt beim Löschen der Materialized View die Tabelle erhalten und benötigt das neue Erstellen der Materialized View keine nennenswerte Zeit.

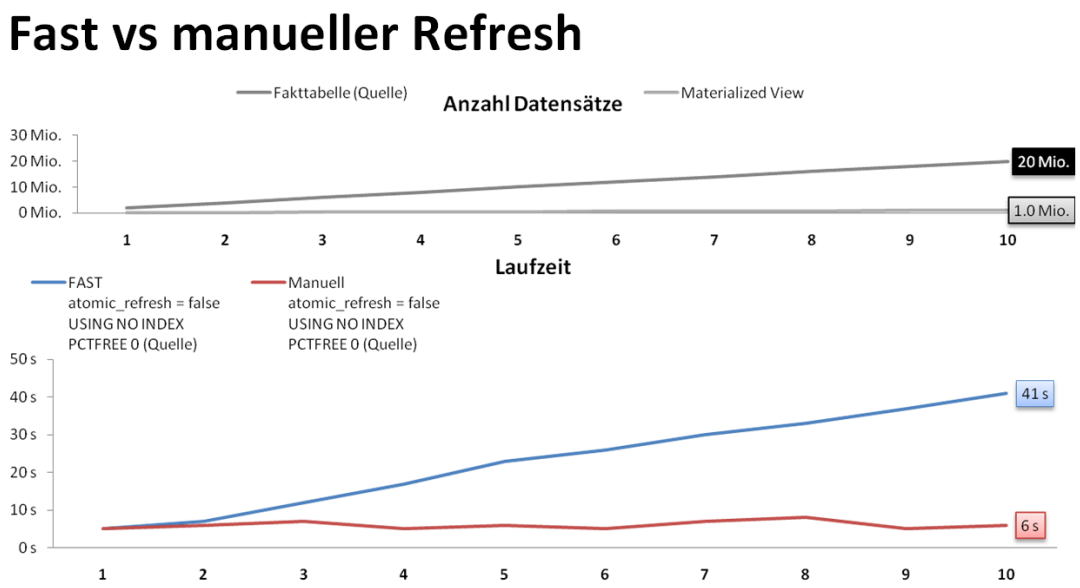

Abb. 5: Vergleich von Fast Refresh und manuellem Refresh

Ein Beispiel aus der Praxis

Die Bedeutung eines gut durchdachten Aufbaus und Refresh der Materialized Views wird an einem Beispiel aus der Praxis deutlich:

Bei einem Kunden hat der Refresh der Materialized Views des DWH stetig mehr Zeit benötigt. Dabei wurde schnell klar, dass die Laufzeit vom historischen Datenvolumen des DWH abhängt. Um die Laufzeit des Refresh zu reduzieren und möglichst konstant zu halten, wurden die oben beschriebenen Maßnahmen konsequent umgesetzt. Für einen Teil der Materialized Views konnte der Einsatz des Fast PCT Refresh ermöglicht werden, so dass eine erste deutliche Reduzierung der Laufzeit erreicht wurde.

Um eine durchgängige Konstanz der Refresh-Laufzeit zu erzielen, wurde außerdem ein manueller Refresh für Materialized Views mit SCD Typ 2-Joins durchgeführt. Da beim Kunden der Oracle Warehosue Builder (OWB) als ETL-Tool zum Einsatz kommt, wurde ein Framework für den manuellen Refresh entwickelt, so dass das manuelle Verfahren auch für zukünftige Materialized Views schnell eingesetzt werden kann.

Insgesamt konnte so die Laufzeit für den Refresh der Materialized Views von 5 Stunden dauerhaft auf nur noch 15 Minuten reduziert werden.

Fazit

Für einen effizienten Refresh ist es wichtig, die von Oracle zur Verfügung gestellten Refresh-Verfahren zu verstehen und möglichst effizient einzusetzen. Durch die richtige Parametrisierung der

Refresh-Methoden und durch ein durchdachtes Design der Materailized Views kann der Refresh deutlich beschleunigt werden. Insbesondere bei Fast PCT Refresh kann mit wenig Aufwand eine effiziente Aktualisierung der Daten der Materialized Views erreicht werden. Schließlich bietet die Datenbank auch die Möglichkeit an, den Refresh der Materialized Views manuell durchzuführen. Auch dieses Verfahren ist speziell im DWH-Umfeld sehr interessant, da somit für viele Materialized Views aus der Praxis eine konstante Laufzeit für den Refresh erzielt werden kann.

Kontaktadresse:

Reinhard Mense
ARETO Consulting GmbH
Julius-Bau-Str. 2
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