Data-Warehouse-Systeme gehören heute zum Unternehmensalltag. Sie sind strategisch und oft mit wichtigen operativen Anwendungen verzahnt. Aber in kaum einem anderen Bereich gibt es so viele Baustellen, unzufriedene Anwender und klagende Kostenträger.

Konzepte und Methoden im Oracle Data Warehouse

Alfred Schlaucher, ORACLE Deutschland B.V. & Co. KG

Die größten Probleme beim Betrieb von Data Warehouses sind: zu geringe Umsetzungsgeschwindigkeit neuer Analyse-Anforderungen, Komplexität und Kosten für Betrieb und Weiterentwicklung, zu geringe Informationsausbeute für die Benutzer und zu wenig Anwenderkomfort hinsichtlich Performance und Flexibilität. Auf der anderen Seite boomt der Markt für Tools und es lösen sich die Wellen vermeintlicher Innovationen gegenseitig ab: Appliance, Realtime Data Warehouse, In-Memory, Big Data etc. Doch die Ursachen der Probleme sind nicht schlechte Techniken oder Tools, oft werden wichtige Methoden und Architekturgrundsätze nicht angewandt. Deshalb an dieser Stelle eine Zusammenfassung von einfachen Regeln und Methoden, um Anwendung und Betrieb des Oracle Data Warehouse effizienter zu machen – ohne dass es teuer wird.

Theoretische Grundlagen

Seit Einführung des Data-Warehouse-Konzepts in den 1990er Jahren gelten vier Prinzipien für ein erfolgreiches Data Warehouse (DWH):

- Unternehmensweite DWHs halten Informationen an zentraler Stelle in einer integrierten und abgestimmten Form vor
- Die Informationen sind für alle verständlich abgelegt, ohne Fachjargon und mit zusätzlichen fachlichen Erklärungen und Referenzinformationen
- Die Daten sind historisiert, was Vorhersagen für die Zukunft erlaubt
- Die Daten sind aus den OLTP-Systemen herauskopiert und ermöglichen eine neutrale, von operativen Zwängen losgelöste und Zeitpunktbezogene Analyse

Damit OLTP-Daten diesen Zustand erreichen, durchlaufen sie in dem DWH einen notwendigen Informationsbeschaffungsweg, den man in drei Phasen (Schichten / Layer) unterteilen kann:

- Herauslösen der Daten aus den Vorsystemen (OLTP), Integrieren, Harmonisieren und Qualitätssichern (Stage-Schritt, Data Integration Layer)
- Granularisiertes Ablegen der Informationen (nahezu drei NF), Anreichern mit Referenzdaten und Historisieren (Enterprise Information Layer)
- Endbenutzergerechtes, fachthemenorientiertes Aufbereiten (meist multidimensional organisierte Daten, User View Layer)

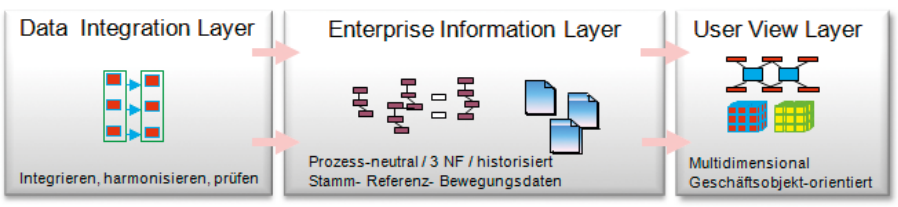

Abbildung 1: Klassisches Drei-Schichten-Modell einer DWH-Architektur

Dieses Drei-Schichten-Modell hat sich als Standard etabliert (siehe Abbildung 1). Auf das Oracle Data Warehouse angewendet gelten nachfolgend beschriebene Konzepte beziehungsweise Best Practices.

Konzepte, Methoden und Best Practices

Die Konzentration des Schichtenmodells auf eine einheitliche, zusammenhängende physische Ablage ist gerade für das Oracle Data Warehouse eine der wichtigsten Forderungen und zielt auf eine Reihe von Vorteilen, die nachfolgend beschrieben sind. Alle oben genannten Layer sind am besten in einer Datenbank auf einem Server (oder Server-Cluster) aufgehoben.

Ein wichtiges Prinzip ist das datenzentrierte Vorgehen beim Management der Warehouse-Daten. Das beinhaltet das Nutzen der technischen Mittel der Datenbank als primärer Host der Daten und betrifft Security, ETL, Vorbereiten von Kennzahlen und auch fachspezifische Sichten. Die Vorteile liegen in dem gemeinsamen Nutzen von Hardware und Lizenzen, einmaligem Aufwand und kurzen Wegen.

Es gilt Neutralität des DWH-Systems gegenüber den Vorsystemen, vor allem aber gegenüber den nachfolgenden

Business-Intelligence-Tools zu wahren. Das bedeutet, dass nur die Datenmodell-Formen genutzt werden, die für die Aufgabenstellung in den jeweiligen DWH-Layern am passendsten sind:

- nahezu drei NF im Enterprise-Layer
- kompakte multidimensionale Strukturen (Star-Schema) im End-User-Layer

DWH-Daten sind langlebig (zehn Jahre und länger), Vorsysteme ändern sich meist viel häufiger und BI-Tools werden ebenfalls öfter ausgetauscht oder es befinden sich sogar unterschiedliche BI-Tools gleichzeitig im Einsatz. Wer diese Regel nicht berücksichtigt, transportiert alle Änderungen in den Vorsystemen und bei den BI-Tools automatisch auch in das DWH.

Durchgängigkeit des Schichtenmodells für Benutzerzugriffe

Um Informationen nicht unnötig zwischen den Schichten zu kopieren und zu verdoppeln, sollten Endbenutzern Lesezugriffe bis in den Enterprise-Information-Layer hinein erlaubt sein. Hier sind neben Stammdaten auch viele interessante Referenzdaten abgelegt. Das macht das DWH zu einem echten Information-Repository, wodurch es für die Endanwender noch wertvoller wird. Meist haben die DWH-Administratoren ihre Hände auf dieser zentralen Schicht und sie argumentieren mit Security und Performance. Mit dieser restriktiven Haltung schmälern sie jedoch nur den Nutzen des DWH für die Anwender. Die vorgebrachten, meist technischen Punkte lassen sich jedoch mit anderen Mitteln lösen. Es gilt der Grundsatz: Eine Kopie der Daten im End-User-Layer ist nur sinnvoll, wenn Daten in eine kompaktere Modellform (etwa Star-Schema) oder in eine für Endbenutzer leichter verstehbare Darstellung überführt werden.

Große Bewegungsdaten-Bestände (Fakten-Daten) sollte es nur einmal im gesamten System geben und sie sollten nicht vom Enterprise-Layer in den User-View-Layer (Data Marts) „1:1" kopiert werden. Das bedeutet, dass sie im Enterprise-Layer angesiedelt sind, um sie über den End-User-Layer zu referenzieren.

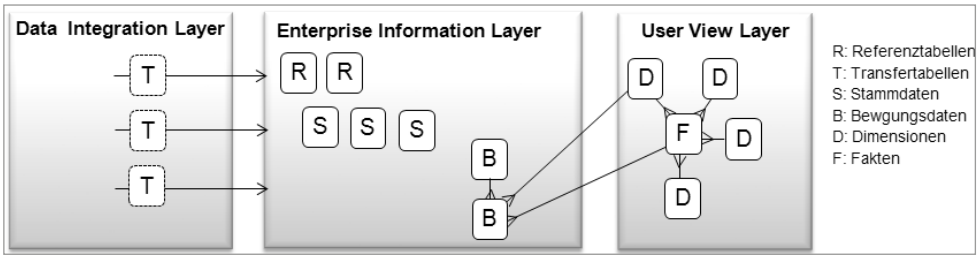

Abbildung 2: Umgang mit großen redundanten Tabellen

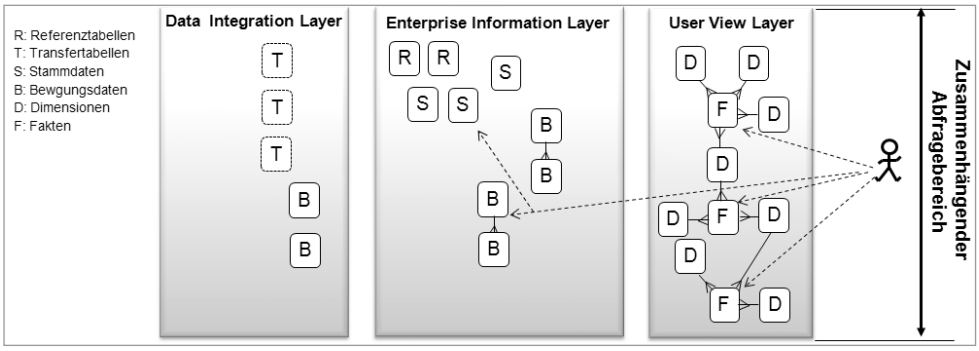

Abbildung 3: Verbundende Kennzahlen

Eventuell können die Bestände sogar mehrfach wiederverwendet werden. In Star-Schemen referenzieren bei dieser Vorgehensweise die Dimensions-Tabellen im User-View-Layer (Data Mart) als Parent Table ihre Fakten-Tabellen im Enterprise-Layer. Ohne technische Einschränkungen hinnehmen zu müssen (wie das Star-Transformation-Feature), funktioniert das jedoch nur, wenn alle Objekte in derselben Datenbank liegen (siehe Abbildung 2).

Verbundene Kennzahlen

Im User-View-Layer sind Verknüpfungen in der multidimensionalen Struktur zu schaffen, um Sachverhalte aus unterschiedlichen Kontexten und Sichten zusammenhängend abfragen zu können (siehe Abbildung 3). Die verbindenden Elemente sind die Dimensionen (Geschäftsobjekte), über die die Kennzahlen (Fakten, Measures) in einen Bezug gebracht werden können. Das verhindert sogenannte „Äpfel/Birnen-Vergleiche" schon beim Design des Datenmodells, macht sachübergreifende Abfragen gefahrlos möglich und schafft zusätzliche Flexibilität für die Anwender. Extrem formuliert: Es sollte auf alle Fakten-Tabellen mit passenden Joins über ihre Dimensionen in einer Abfrage zugegriffen werden können. Das ist in der Praxis nicht immer machbar, weil Sachgebiete unter Umständen zu unterschiedlich sind.

Vorgedachte Aggregate und Kennzahlen

Die meisten Kennzahlen sind bereits bekannt. Man bereitet sie schon in der DWH-Datenbank als vorgedachte Aggregate vor und nicht erst im BI-Tool. Das mindert den Aufwand in den aufsetzenden BI-Tools und schafft zusätzliche Flexibilität beziehungsweise mehr Angebote für die Anwender. Hier helfen Kennzahlen-Bäume, die technisch als Nested Materialized Views (aufeinander aufbauende Materialized Views) realisiert sind. Die Verwaltung erfolgt durch die Datenbank transparent, völlig automatisiert und ressourcenschonend. Der Wiederverwendungseffekt ist beachtlich. Die klassischen Aggregationstabellen sind out. Man realisiert sie heute grundsätzlich

mit Materialized Views. Das spart Verwaltungs- und ETL-Aufwand.

Generell wird man den Feinheitsgrad der Daten in User- und Enterprise-Layer so granular wie möglich halten, denn aggregieren kann man später immer noch, aber etwas Aggregiertes lässt sich im Nachhinein nicht mehr granularer gestalten. Granulare Daten bieten mehr Auswerte-Optionen und Flexibilität für die Anwender. Datenbanken und Hardware sollten so ausgelegt sein, dass auch große Datenbestände „on the Fly" aggregiert werden können. Die Technologie hierzu ist längst vorhanden.

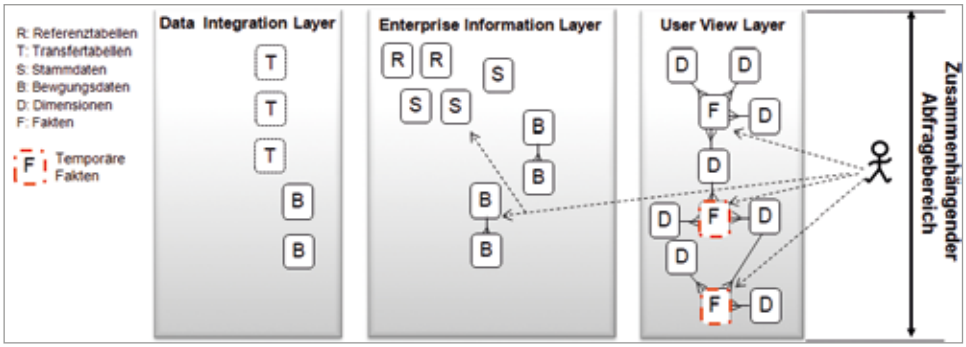

Abbildung 4: Dynamisch erstellte Auswerte-Strukturen

Dynamische Auswerte-Strukturen

Die Strukturen im User View Layer (Data Marts) sollten so gewählt werden, dass man ihre Inhalte jederzeit dynamisch aus den Daten der Vorschicht wieder herleiten kann, wenn diese durch die Anwender gezielt angefordert werden (siehe Abbildung 4). So muss man nicht permanent alle User-View-Daten bereithalten oder aktualisieren. Das spart vor allem Plattenplatz sowie ETL-Aufwand und auch ein Backup der Data-Mart-Daten ist nicht mehr nötig.

Das Security-Konzept des Analyse-Systems sollte in der DWH-Datenbank gelöst sein und nicht in dem nachgelagerten BI-Tool. Damit ist eine Offenheit für flexible BI-Strategien gegeben. Unterschiedliche BI-Tools, aber auch beispielsweise Excel- oder sonstige OCI-Anwendungen partizipieren von dem „Einmal-Security"-Aufwand in der Datenbank. Außerdem vermeidet man peinliche Security-Pannen. Security- oder Mandanten-Anforderungen dürfen nicht zu einer Verdoppelung von Strukturen und Daten führen.

ETL-Prozesse sollten innerhalb der Datenbank stattfinden, um unnötigen Datentransport zwischen Datenbank- und ETL-Server zu vermeiden. Zudem nutzt man die Hardware-Ressource der Datenbank sowie die Datenbank-Lizenzen besser aus. Hinzu kommen die vielen technischen Mittel der Datenbank wie Table Functions, Partitioning und die Load-Features, die extrem schnelles Laden möglich machen. „1:1"-Transport-Vorgänge, also Datenbewegungen ohne Mehrwert-liefernde Transformationen, sind zu vermeiden. Alle ETL-Prozesse sollte man Set-Based, also mengenbasiert und nicht satzweise durchführen. Constraints und Indizes sind auszuschalten. Syntaktische Datenqualität wird nicht mit Datenbank-Constraints, sondern aufgrund der besseren Performance mit SQL-Mitteln gelöst.

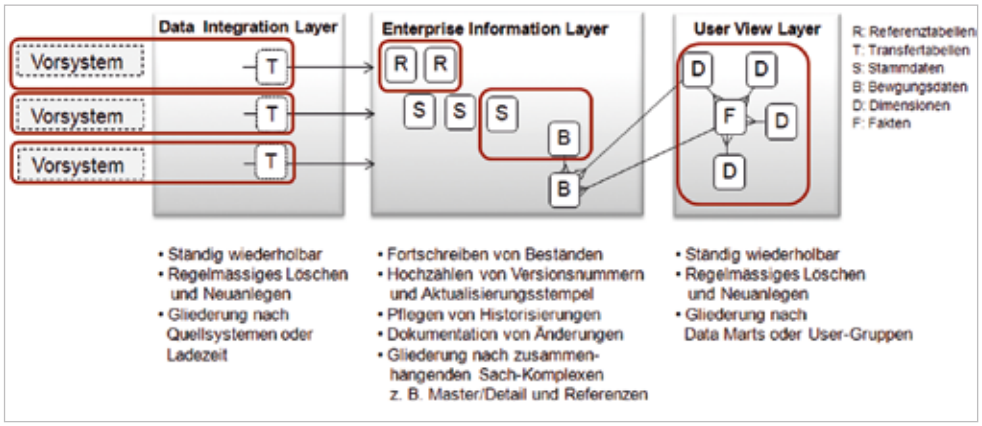

Abbildung 5: Strukturierung des ETL-Prozesses in Wiederholer- und Zusammenhangs-Gruppen

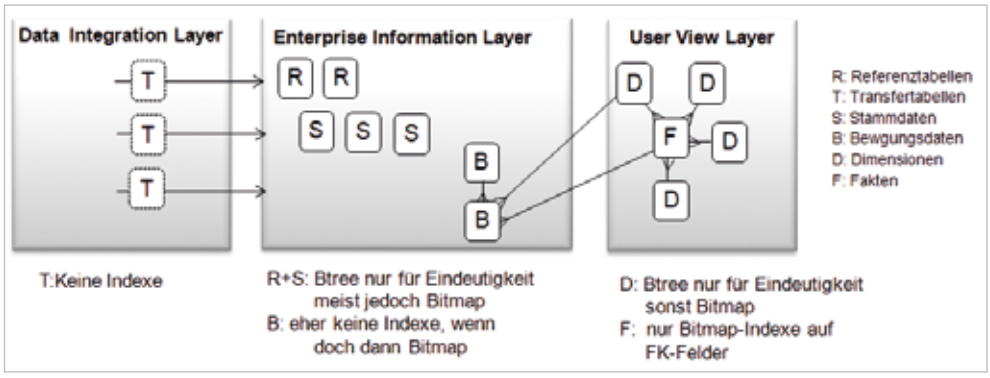

Abbildung 6: Indizierung im DWH

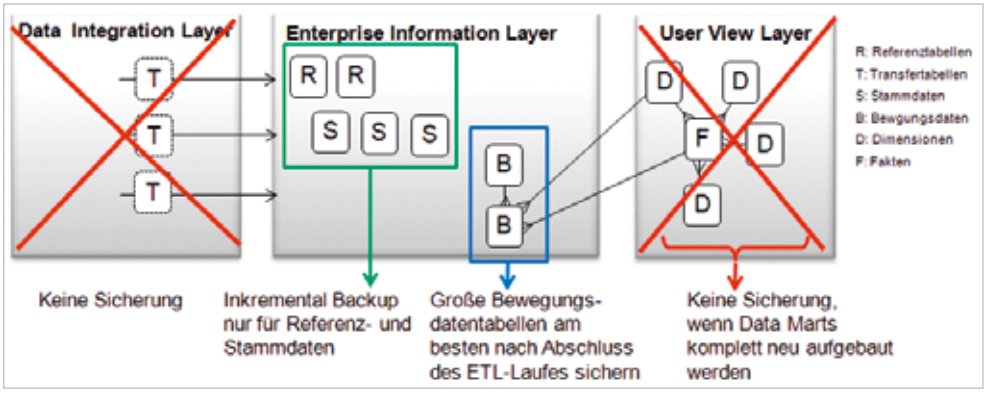

Abbildung 7: Was wird wie gesichert

Die Parallelisierung (parallele Datenbank-Prozesse) wird im ETL-Prozess nicht pauschal, sondern gezielt für einzelne Objekte gesteuert. ETL-Prozesse sind kontrollierte Vorgänge (siehe Abbildung 5). Hier weiß man in der Regel, was passiert, und kann bezogen auf Parallelisierung, aber auch bezüglich Re-Indizierung, Verwenden von Partitioning oder Aktualisierung der Statistiken kontrolliert vorgehen. Künstliches, also manuell programmiertes Parallelisieren außerhalb der Datenbank ist zu aufwändig, führt zu Lock-Situationen und sollte vermieden werden.

Kommen dennoch Engine-gestützte ETL-Werkzeuge zur Anwendung, die Transformations-Prozesse nicht in der Datenbank durchführen, sollte man die Datenleitung zwischen ETL- und Datenbank-Server möglichst leistungsstark wählen (10 GB). Einfache Datenkopien sollte man nicht mit dem ETL-Werkzeug, sondern mit Datenbank-Mitteln lösen und stattdessen einen grafischen Platzhalter für diese Datenbank-Operation in das grafische Modell einfügen.

Dokumentation, Metadaten und Kontrolle

Kaum ein anderer Bereich wird im DWH stärker vernachlässigt, als der der Metadaten. Das liegt wahrscheinlich daran, dass die Verantwortlichen hier keinen unmittelbaren Nutzen sehen. Aber die Folgen fehlender Metadaten-Dokumentation entstehen einige Zeit später umso drastischer, wenn Chaos Einzug hält: Sachverhalte sind mehrfach und auch noch unterschiedlich gespeichert. Die meisten Nutzer (und Administratoren) wissen nur zum Teil, was gespeichert ist. Große Datenmengen liegen nutzlos brach.

Ein DWH braucht neben einer sauberen Datenmodellierung auch ein neutrales Metadaten-Repository, in dem neben den technischen Objekten der Datenbank (Tabellen, Spalten, Views,

Mappings) auch fachliche Beschreibungen der Inhalte, eine Dokumentation über Herkunft (Transformationen) und Verwendung (Nutzungsstatistik) der Daten und eine Dokumentation über Benutzer und ihre Erwartungen vorhanden sind. Es muss jederzeit bekannt sein, wer welche Daten benutzt und wo welche Daten wie gespeichert sind. Diese Information sollte für alle Beteiligten beispielsweise in einer Web-Oberfläche zur Verfügung stehen, und nicht nur dem Administrator.

Indizierung im DWH

Im DWH nutzt man eher Bitmap-Indizierung als Btrees (siehe Abbildung 6). Bitmaps benötigen weniger Platz und sind bei mengenorientierten Abfragen schneller. Btrees nutzt man oft nur zur Verwaltung der Eindeutigkeit bei Stamm-, Referenz- und Dimensions-Tabellen (Primary-Key-Funktion).

Daten-Backup des Data Warehouse

DWH-Systeme benötigen ein eigenes, also von OLTP-Systemen unabhängiges Backup-Verfahren, denn nicht alles muss man regelmäßig sichern. Integration und User-View-Layer sind nicht zu sichern. Referenz- und Stammdaten muss man nur dann sichern, wenn sie sich geändert haben (Inkrementelles Backup des RMAN). Große Bewegungsdaten-Tabellen sichert man gekoppelt an den ETL-Prozess, weil man hier die geänderten Datenbestände kennt, etwa in Verbindung mit Partitioning (siehe Abbildung 7). Backups nur über das Storage-System (Image-Copy) sind zu vermeiden (Gefahr von Block-Corruption und zu teuer).

Regeln aus Hardware-Sicht

Data-Warehouse-Daten sollten in einem separaten DWH-Storage-System liegen. Eine gemeinsame Storage-Nutzung, zum Beispiel mit OLTP-Systemen, ist zu vermeiden. Das DWH liest meist größere und zusammenhängende Datenmengen, oft sogar über einen Full Table Scan, OLTP-Systeme dagegen eher in kleineren Daten-Häppchen und dazu oft noch über einen Index. Das gilt analog auch für das Netzwerk zwischen DWH-Storage und Datenbank-Server. Direkt angeschlossener Datenspeicher ist sicher das beste (Exadata – Appliance-Prinzip).

Bei der Wahl des Storage-Systems setzt man eher auf mehr und dafür kleine Platten. Ideal ist es, wenn diese nur zur Hälfte beschrieben werden (Datenablage nur auf den äußeren Spindeln). Der Hauptspeicher sollte mindestens mit 8 GB pro Core dimensioniert sein, um aufwändige Star-Join-Operationen und analytische Funktionen besser zu unterstützen.

Die Menge der CPU-Cores richtet sich nach der zu lesenden Datenmenge pro Sekunde (MB/sec). Man geht bei heutigen CPUs (>2GHz) von etwa 200 MB/sec Verarbeitungskapazität pro Core aus. Bei einem optimierten 30TB-DWH mit etwa 5000 MB/sec Datendurchsatz (5GB/sec) sind das ungefähr 25 Cores und etwa 100 Platten (ohne Spiegelung). Wer weniger als 5 GB/sec Lese-Performance in einem solchen DWH hat, sollte sich damit nicht zufrieden geben und Alternativen suchen.

Die Kosten für Storage-Systeme variieren sehr stark. Hat man privaten Storage für das DWH gewählt, so kann man unterschiedlich teure Platten für unterschiedlich wichtige/aktuelle Daten im DWH einsetzen. Meist wird nur ein geringer Teil der DWH-Daten intensiv genutzt und der große Rest liegt für Monate brach.

Alfred Schlaucher
alfred.schlaucher@oracle.com