Flexible Schnittstelle für Flat Files in das DWH

Thomas Mauch
Trivadis GmbH
Stuttgart

Schlüsselworte
DWH, Flat Files, External Tables, Schnittstelle, Versionierung

Einleitung

In DWH's werden oft Flat Files als Quelle verwendet. Anhand der Erfahrungen in einem Projekt wird ein Beispiel für eine dynamische DWH Schnittstelle vorgestellt, die sich während des Ladeprozesses an Strukturänderungen der Flat Files anpassen kann. Dabei werden die Strukurinformationen, die z.B. über ein zusätzliches File geliefert werden, benutzt, um im Rahmen konfigurierbarer Grenzen die External Tables und die Ladetabellen in der Datenbank so anzupassen, dass dieselbe Schnittstelle unterschiedliche Versionen einer Datei im selben Zeitraum verarbeiten kann. Zusätzlich wird gezeigt, wie mit der Datenbank die Vollständigkeit der Files kontrolliert werden kann, die Files entpackt und nach dem Laden archiviert werden können.
Diese Aufgaben werden komplett mit der Oracle Datenbank ausgeführt, d.h. die Logik ist in PL/SQL Packages und in Java Stored Procedures in der Datenbank abgelegt. Die Konfiguration und die Ergebnisprotokollierung des Ladeprozesses erfolgt in Datenbanktabellen. Damit kann der Ladeprozess sehr gut in die weitere Verarbeitung der Daten in der Datenbank eingebunden werden.

Vorbereitung der Files

Bevor die Files geladen werden können, muss überprüft werden, ob alle Files vorhanden sind, gepackte Files müssen entpackt werden und die Files evtl. in ein Ladeverzeichnis kopiert werden. Diese Aufgaben werden in vielen Systemen außerhalb der DB erledigt, z.B. mit Hilfe von Shell-Skripten. Dies hat jedoch folgende Nachteile:

- Skripte sind oft schlecht wartbar.
- Konfiguration und Verantwortung für Skripte oft außerhalb des DWH-Bereichs, deshalb sind Anpassungen aufwendig
- Ergebnisprotokollierung außerhalb der Datenbank z.B. in Files schlecht verfügbar und auswertbar im DWH

Diese Nachteile fallen weg mit einer Verarbeitung der Files über PL/SQL und Java in der Datenbank. Tab. 1 gibt einen Überblick, wie die Aufgaben umgesetzt werden können.
Zusätzlich gibt es die Möglichkeit über Java in der DB einen Host-Call zu machen und damit Shell-Skripte einzubinden.

Aufgabe	Umgesetzt mit
Kopieren und Löschen von File	PL/SQL (utl_file Package)
Lesen und Schreiben von Files	PL/SQL (utl_file Package)
Überprüfung der Existenz und Größe eines Files	PL/SQL (utl_file Package)
Liste der Namen und Eigenschaften (Größe, Erstellungsdatum) von Files in einem Verzeichnis	Java in der DB
Packen und Entpacken von Winzip und Gzip Archiven	Java in der DB
Packen und Entpacken von Tar Archiven	Java Library (z.B. JTAR) in DB importiert

Tab. 1: Überblick wie mit der DB Files verarbeitet werden können

Beim Vorbereiten der Files wird folgendes Szenario betrachtet: Die Files, die geladen werden sollen, werden in ein Source Verzeichnis geliefert, evtl. auch gepackt als zip, gz, tar oder tar.gz File. Die Files werden dann in ein Ladeverzeichnis kopiert und dabei entpackt. Optional können die Files nach dem Laden wieder gepackt in ein Archivverzeichnis kopiert werden (siehe Abb. 1)

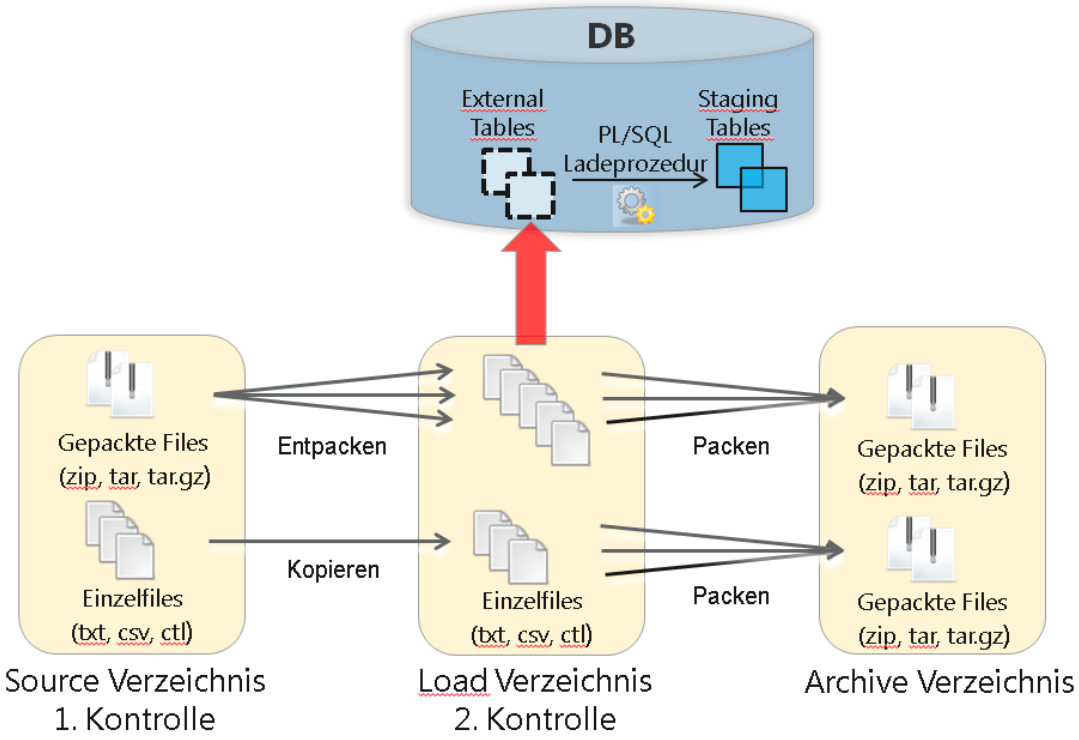

Abb. 1: Überblick über den gesamten Ladeprozess

Konfiguration:
Ein wichtiger Punkt beim Vorbereiten der Files ist die Kontrolle, ob alle Files vorhanden sind. Deshalb werden die Namen der Files im Vorfeld in Datenbanktabellen eingetragen (Datenmodell siehe Abb. 2). In der Tabelle TFH_SOURCE_DATAFILES werden die Files eingetragen, die im Source Verzeichnis angeliefert werden. In der Tabelle TFH_FILE_GROUPS können die Files zu Gruppen zusammengefasst werden. Die Files werden immer pro Filegruppe verarbeitet, d.h. die Files werden erst dann verarbeitet, wenn alle Files einer Filegruppe vorhanden sind. In der Tabelle TFH_SUPPLIER werden die Lieferanten von Files eingetragen, z.B. die Niederlassungen oder Applikationen. Über die Tabellen TFH_SUPPLIER_GROUPS kann jeder Eintrag in TFH_SOURCE_DATAFILES mehreren Lieferanten zugeordnet werden. Falls das gelieferte File im Source Verzeichnis ein gepacktes File ist, kann es mehrere Einzelfiles enthalten. Diese Files werden in TFH_DATAFILES eingetragen.

Ablauf Vorbereitung der Files:
Die Prozedur zum Vorbereiten der Files wird mit dem Filedatum als Parameter gestartet. Dieses Datum wird als Bestandteil des Filenamen überprüft. Falls nicht alle Source Files einer Filegruppe (eingetragen in TFH_SOURCE_DATAFILES) im Source Verzeichnis vorhanden sind, wird ein Scheduler Job in der Datenbank erstellt, der nach einer parametrisierbaren Zeitspanne die Prozedur

zum Vorbereiten der Files für diese Filegruppe wieder startet. Dadurch wird periodisch überprüft, ob die Files eingetroffen sind (1. Kontrolle).
Falls alle Source Files einer Filegruppe vorhanden sind, werden die gepackten Files entpackt und alle Files in das Load Verzeichnis kopiert. Dann überprüft die Prozedur, ob alle Einzelfiles (eingetragen in TFH_DATAFILES) im Load Verzeichnis vorhanden sind (2. Kontrolle). Falls nicht, wird der weitere Ladeprozess für die Filegruppe abgebrochen und ein Fehler protokolliert.

Ergebnisprotokollierung:
Falls alle Files einer Filegruppe vorhanden sind, werden die Filename in die Tabelle TFH_LOG_FOUND_DATAFILES eingetragen. Damit kann der weitere Ladeprozess die Einträge in der Tabelle verwenden, um die Files zu laden
Das Ergebnis der Prozedur wird in der Tabelle TFH_LOG_FILEGRP_LOADS festgehalten, so dass in dieser Tabelle für jeden Ladelauf protokolliert ist, ob eine Filegruppe vollständig vorhanden ist, welche Files noch fehlen usw. Falls Files fehlen oder Fehler aufgetreten sind kann optional eine Mail mit entsprechender Nachricht gesendet werden.
Sobald alle Files einer Filegruppe vorhanden sind, wird der Ladeprozess für die Filegruppe gestartet.

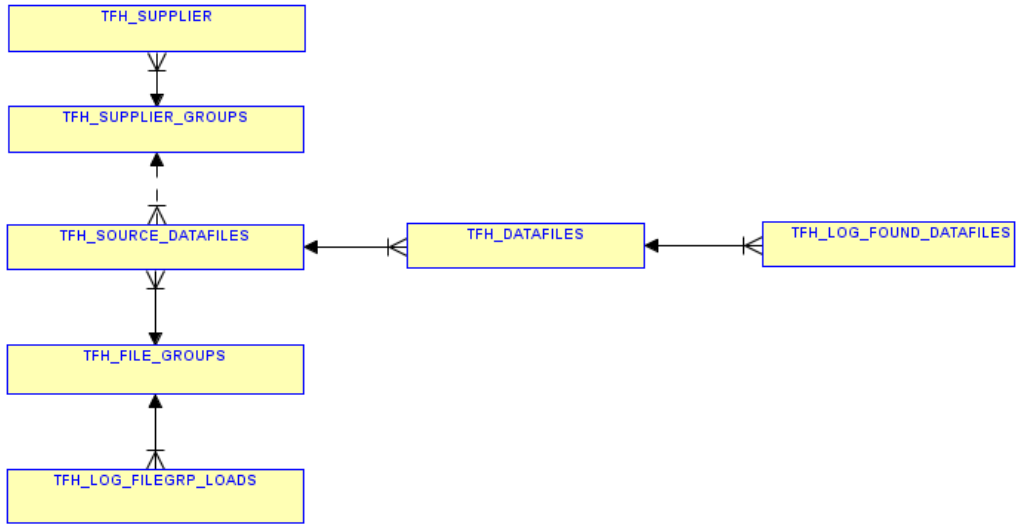

Abb. 2: Datenmodell für die Vorbereitung der Files

Laden der Files
Im zweiten Schritt werden die Files über External Tables in Tabellen in der Datenbank (die Staging Tables im DWH) geladen (siehe Abb. 1). Das besondere an der im Folgenden gezeigte Methode ist, dass dabei die External Tables und die Staging Tables nicht fest vorgegeben ist, sondern diese während des Ladeprozesses erstellt oder angepasst werden. Dies ist dann wichtig, wenn bei einem Ladevorgang Files mit unterschiedlichem Format geladen werden müssen, weil z.B. die Quellsysteme einen unterschiedlichen Versionsstand haben.
Damit dies funktioniert, muss mit dem Datenfile ein Metadatenfile geliefert werden, in dem die Struktur des Datenfile so beschrieben ist, dass das Datenfile geladen werden kann. Als Beispiel wird im Folgenden dafür ein SQLLoader Control File verwendet. Mit diesem Control File wird während des Ladeprozesses eine External Table erzeugt oder angepasst, dann die Staging Table erzeugt oder angepasst und dann die Daten von der External Table in die Staging Table geladen.
Wird kein Control File geliefert, kann das File auch geladen werden aber nur in der Struktur, die zuletzt geladen wurde.

Konfiguration:
Bevor der Ladeprozess gestartet wird, kann er konfiguriert werden, um den Ladeprozess zu kontrollieren. (Datenmodell siehe Abb. 3).

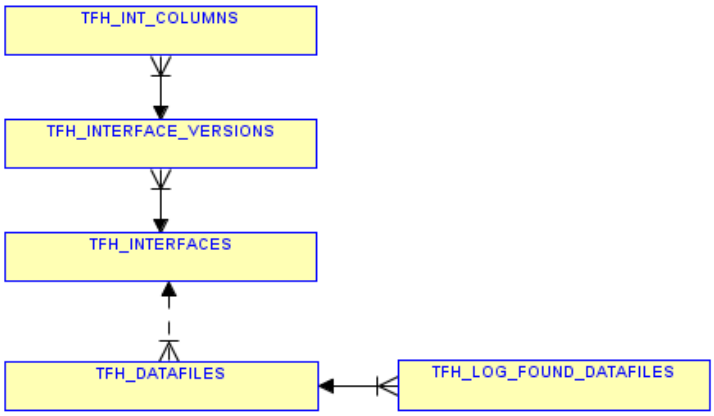

Abb. 3: Datenmodell für das Laden der Files

In der Tabelle TFH_INTERFACES ist die Schnittstelle definiert, die zum Laden benutzt wird. Alle Files, die einer Schnittstelle zugeordnet sind, werden in die Staging Table geladen, die in der Tabelle abgelegt ist. Wenn ein File in einer geänderten Struktur geladen wird, wird eine neue Version der Schnittstelle erzeugt, die in der Tabelle TFH_INTERFACE_VERSIONS abgelegt wird. Die Spaltendefinition der einzelnen Versionen sind in TFH_INT_COLUMNS gespeichert.
Die Einträge in den gerade beschriebenen Tabellen können vor dem Start der Ladeprozedur angelegt werden. Über Parameter gesteuert kann die Ladeprozedur dann so konfiguriert werden, dass nur die eingetragenen Versionen akzeptiert werden. Damit wird verhindert, dass nicht vorgesehene Strukturänderungen stattfinden.
Alternativ kann die Ladeprozedur so konfiguriert werden, dass alle neuen Versionen akzeptiert werden. Dann kann die Ladeprozedur selber beim Ladevorgang die Einträge für die Schnittstellen und Versionen vornehmen, z.B. wird dann der Name der Staging Table aus dem Control File geholt. Das ist flexibler und leicht zu konfigurieren, wenn keine Kontrolle der Versionen notwendig ist

Ablauf Laden der Files:
Die Ladeprozedur wird mit dem Filedatum gestartet und lädt alle Files, die in TFH_LOG_FOUND_DATAFILES eingetragen sind mit diesem Filedatum.
Im ersten Schritt überprüft die Prozedur, ob das Datenfile eine neue geänderte Struktur besitzt. Dies wird mit Hilfe des Control Files gemacht, das den gleichen Namen wie das zugehörige Datenfile hat, aber mit der Extension ‚ctl'. Dazu wird der SQLLoader gestartet, der mit dem Control File ein Skript für die passende External Table generiert. Die Spaltendefinition dieser External Table wird abgeglichen mit den Werten in der Tabelle TFH_INT_COLUMNS.
Falls dort keine passende Version gefunden wird, liegt eine neue Version des Datenfiles vor. Dann passt die Ladeprozedur die Staging Table so an, dass die Daten in der neuen Struktur abgespeichert werden können oder erstellt diese neu falls noch nicht vorhanden. Die Informationen über die neue Version wird von der Prozedur in den Tabellen TFH_INT_COLUMNS und TFH_INTERFACE_VERSIONS eingetragen.
Falls eine passende Version gefunden wird, kann die Staging Table unverändert für das Laden der Daten verwendet werden.

Für den eigentlichen Ladevorgang wird von der Ladeprozedur ein Insert-Statement generiert, das die Daten über einen direct Path Load von der External Table in die Staging Table kopiert. Über die Kontrolle von evtl. erzeugten Badfile oder Errorlogtabelle wird sichergestellt, dass alle Datensätze verarbeitet wurden.

Ergebnisprotokollierung:
In der Tabelle TFH_LOG_FOUND_DATAFILES sind alle Datenfiles eingetragen. Dort wird vermerkt, ob der Ladevorgang erfolgreich war, wie viele Datensätze geladen wurden und welche Version für den Ladvorgang verwendet wurde. Ebenso wird dort eine eindeutige file_id abgelegt. Diese id wird in den Staging Tables mit abgespeichert, so dass später nachverfolgt werden kann, aus welchem File die Datensätze geladen wurden.
Falls der Ladevorgang nicht erfolgreich war, wird dies protokolliert und der Name der Errorlogtabelle oder des Badfiles abgelegt und es wird optional ein Mail mit entsprechenden Nachricht versendet.
Die weitere Verarbeitung der Daten (ETL-Prozess) wird nur angestossen, falls alle Daten erfolgreich geladen wurden.

Fazit
Mit Hilfe der vorgestellten Methode Methode wurde gezeigt, wie eine Schnittstelle für Flat Files in die Oracle Datenbank realisiert werden kann, die dynamisch auf Strukturänderungen der Files reagieren kann. Der Ladeprozess kann durch Tabelleneinträge konfiguriert und so an die jeweiligen Anforderungen angepasst werden. Durch die Kontrolle der Files und des Ladevorgangs ist sichergestellt, dass keine Daten verloren gehen. Bei Problemen können per Mail direkt die Verantwortlichen benachrichtigt werden. Dass Ergebnis des Ladeprozesses ist vollständig in Tabellen protokolliert und kann deshalb jederzeit kontrolliert werden. Und nicht zuletzt kann der Ladeprozess sehr gut in den nachfolgenden ETL-Prozess in der Datenbank integriert werden.

Kontaktadresse:
Thomas Mauch
Trivadis GmbH
Industriestr. 4
D-70565 Stuttgart

Telefon:	+49 (0) 711 - 90 36 32 30
Fax:	+49 (0) 711 - 90 36 32 59
E-Mail	thomas.mauch@trivadis.com
Internet:	www.trivadis.com