Aufbau eines Data Warehouse mit den Open-Source-Komponenten JPivot, Mondrian und MySQL — Eine Alternative zu kommerziellen Produkten?

Olaf Herden
Berufsakademie Stuttgart
Florianstr. 15
D-72160 Horb/Neckar
o.herden@ba-horb.de

Benjamin Pfrommer
Spirit/21 AG
Otto-Lilienthal-Straße 36
D-71034 Böblingen
bpfrommer@spirit21.de

Abstract: In diesem Beitrag wird der Aufbau eines Data-Warehouse-Systems auf Basis von Open-Source-Komponenten realisiert, wobei MySQL als Datenbank, Mondrian als OLAP-Server und JPivot als GUI-System zum Einsatz kommen. Neben einer Beschreibung der Architektur und der Funktionsweise, wird eine Evaluation des Systems vorgenommen und die Praxistauglichkeit des Ansatzes im Vergleich mit entsprechenden kommerziellen Werkzeugen untersucht.

1 Einleitung

Data-Warehouse-Systeme (DWS) [4, 5, 8] haben sich als technische Plattform entscheidungsunterstützender Systeme etabliert. Unter einem Data Warehouse (DWH) wird dabei eine Datenbank (DB) verstanden, auf der alle Datenanalysen eines Unternehmens durchgeführt werden. Besondere Bedeutung besitzt die als Online Analytical Processing (OLAP) bezeichnete Form der Datenanalyse, die inzwischen eine Vielzahl kommerzieller Produkte, wie z.B. Microstrategy [9] oder Cognos Powerplay [3] anbieten. Seit kurzem stehen als Alternative dazu auch einige Open-Source-Komponenten zur Verfügung. In diesem Beitrag soll der Aufbau eines DWS mit MySQL [11] als Datenbank, Mondrian [10] als OLAP-Server und JPivot [7] als OLAP-Front-End-Komponente untersucht werden.
Der Beitrag ist folgendermaßen aufgebaut: Abschnitt 2 stellt die Architektur und das Zusammenspiel der Open-Source-Komponenten dar, Abschnitt 3 die Vorgehensweise zur Untersuchung der Praxistauglichkeit und wichtige Resultate dieser Evaluation. Abschnitt 4 gibt eine Zusammenfassung und einen Ausblick über weitere mögliche Folgearbeiten.

2 Architektur und Konzepte

Die grundlegende Architektur und das Zusammenspiel der Komponenten JPivot, Mondrian und MySQL ist in Abbildung 1 dargestellt: Als Front-End-Werkzeug wird ein Web-

Browser verwendet, JPivot läuft in einem Tomcat Web-Server, der Mondrian OLAP-Server und die MySQL-Datenbank laufen als separate Prozesse. Zusätzlich benötigt Mondrian eine XML (Extensible Markup Language)-Datei, in der die multidimensionalen OLAP-Strukturen definiert werden.

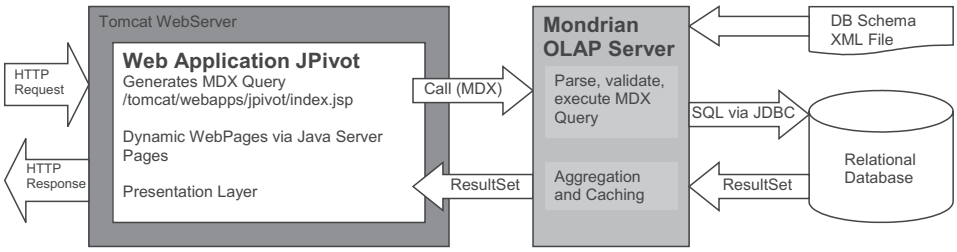

Abbildung 1: Architektur des Open Source DWS [13]

Eine multidimensionale Anfrage läuft folgendermaßen ab: Der Benutzer wählt diese zunächst im Browser aus. Die multidimensionale Anfrage wird als HTTP (Hypertext Transfer Protocol)-Anfrage an den Web-Server übergeben. Die im Web-Server laufende JPivot-Komponente nimmt die Anfrage entgegen und wandelt diese in eine MDX (Multidimensional Expressions)-Anfrage um. Die MDX-Anweisung wird anschließend an den OLAP-Server Mondrian gesendet. Dieser nimmt die Anfrage entgegen, parst und validiert sie. Danach führt Mondrian die MDX-Anfrage aus, indem eine Folge von SQL-Anweisungen generiert wird. Diese werden per JDBC (Java Database Connectivity) an die relationale Datenbasis gesendet und hier bearbeitet. Das Resultat wird an Mondrian zurückgegeben, der die Einzelresultate zu einem Ergebnis zusammenfasst. Ebenso führt Mondrian ein Caching durch, so dass (Zwischen)resultate bei späteren Anfragen wiederverwendet werden können. Das von Mondrian ermittelte Resultat wird dann an den Web-Server zurückgeliefert und hier von JPivot graphisch aufbereitet, indem mittels JSP (Java Server Pages)-Technologie dynamisch eine Webseite generiert wird. Diese wird an den Browser zurückgesendet und dort dem Benutzer angezeigt.

3 Vorgehen und Resultate

Nach der Installation und Konfiguration des Systems haben wir eine Reihe von OLAP-Berichten entworfen [13]. Diese Auswahl repräsentiert typische OLAP-Anfragen von Benutzern, die in anderen Projekten mit dem kommerziellen Werkzeug Microstrategy realisiert wurden [1, 14]. Im Folgenden werden anhand wichtiger Kriterien die untersuchten Open Source Produkte (JPivot/Mondrian/MySQL) mit den Eigenschaften der entsprechenden Klasse kommerzieller Produkte verglichen. Beispiele kommerzieller Vertreter sind Microstrategy und Cognos PowerPlay als OLAP-Werkzeuge sowie Oracle, IBM DB2 und SQL Server als Datenbanken.

Die erste Gruppe von Vergleichskriterien bezieht sich auf den OLAP-Client, d.h. auf die

Sichtweise des Endbenutzers:

- Als *Client* kann aufgrund der eingesetzten JSP-Technologie bei den Open-Source-Komponenten nur ein Browser dienen. Herkömmlich bieten kommerzielle Systeme einen Windows-Client, in den neuesten Versionen trifft man jedoch auch hier aufgrund des reduzierten Administrationsaufwandes meistens Browser an.
- Mit Drilling, Rotation und Slice and Dice stehen die *Standard OLAP-Operationen* sowohl in den Open-Source- als auch in den kommerziellen Produkten zur Verfügung.
- Darüber hinaus bieten kommerzielle Werkzeuge eine Reihe *erweiterter OLAP-Funktionalitäten*, die den Bedienkomfort für den Benutzer erhöhen. Hierzu zählen z.B. komfortable Filteroptionen oder Drag-and-Drop-Möglichkeiten. Diese stellt JPivot nicht zur Verfügung.
- Beide Klassen von Systemen bieten *Exportschnittstellen* im Excel-, PDF- und Textformat an.
- Die Aufbereitung *großer Ergebnismengen* ist bei den Open-Source-Komponenten bei gleicher Hardware und Datenmenge merklich langsamer als bei den kommerziellen Werkzeugen.

Eine Gegenüberstellung der Eigenschaften von Mondrian mit kommerziellen Servern liefert folgende Ergebnisse:

- Die *Installation* kommerzieller Systeme läuft mit einer mitgelieferten Setup-Routine ab, möglicherweise einzustellende Parameter oder Optionen werden im Dialog erfragt. Die Installation von Mondrian (und auch von JPivot) hingegen erwies sich als äußerst zeitaufwändig, insbesondere traten viele Abhängigkeiten auf und die für ein problemloses Zusammenspiel zu verwendenden Versionen des Tomcat-Servers bzw. der MySQL-DB waren nur unzureichend dokumentiert.
- Die *Definition der multidimensionalen Strukturen* erfolgt bei Mondrian über eine XML-Datei. Kommerzielle Werkzeuge bieten zur Festlegung Wizards an, die bereits initiale Strukturen vorschlagen und die Definition mit graphischer Unterstützung erlauben.
- Als *Schematyp* werden von Mondrian Schneeflocken- und Sternschema akzeptiert. Kommerzielle Systeme ermöglichen meistens beliebige relationale Schemaformen, wobei jedoch meistens auch hier nur eine Form optimal bezgl. Performanz ist.
- *Zugriffsschutz* ist in der Open-Source-Realisierung auf Ebene der DB realisierbar, ebenso kann der Zugriffsschutz pro JSP-Seite realisiert werden. Kommerzielle Systeme hingegen bieten eine detaillierte Zugriffsrechteverwaltung auf verschiedenen Ebenen (Bericht, einzelne Kennzahlen, einzelne Hierarchie-Ebenen, DB).
- Während die kommerziellen OLAP-Server vielfältige *Konfigurationsmöglichkeiten* bieten (z.B. Caching oder Lastverteilung), können bei Mondrian keine besonderen Einstellungsmöglichkeiten vorgenommen werden.

- Neben der Datenanalyse durch Benutzer gewinnt in Unternehmen zunehmend auch die *aktive Informationsversorgung* an Bedeutung. Kommerzielle Werkzeuge bieten hierzu Möglichkeiten, wie z.B. das automatisierte Versenden eines Berichts zu bestimmten Zeitpunkten. Die Open-Source-Produkte liefern hierzu keine eingebauten Funktionalitäten mit.

Die Benutzung des Systems aus Sicht eines Entwicklers lässt sich folgendermaßen festhalten:

- Das *Erstellen eines Berichts* geschieht in den kommerziellen Werkzeugen durch graphische Wizards, die eine übersichtliche Verwaltung und Wiederverwendung der angelegten Objekte ermöglichen. Bei den Open Source Produkten hingegen wird ein Bericht durch die Definition einer MDX-Abfrage und eines JSP-Template relativ unkomfortabel angelegt.
- Von den *Gestaltungsmöglichkeiten* ließen sich die am Anfang dieses Abschnitts genannten OLAP-Berichte (bis auf wenige Ausnahmen mit spezieller Funktionalität) wie in den kommerziellen Systemen nachbilden.
- Zum Gestalten von Berichten sind in kommerziellen OLAP-Servern eine Vielzahl *eingebauter Funktionen*, wie z.B. besondere Durchschnitte, Verteilungen und Rang-Funktionen, implementiert. Mondrian besitzt solche Funktionen hingegen nicht.

Schließlich gibt es noch folgende allgemeine bzw. auf die DB bezogene Vergleichskriterien:

- Hersteller kommerzieller Produkte stellen vielfältige *technische Dokumentationen* zur Verfügung, insb. auch für die Administration im laufenden Betrieb. Die hier untersuchten Open Source Produkte sind in diesem Punkt weitgehend undokumentiert.
- Die *Datenbank* wird in der Open-Source-Architektur „ganz normal“ verwendet, d.h. sie bietet keine besonderen Erweiterungen für den Einsatz als DWH. Kommerzielle DBen besitzen hingegen eine ganze Reihe von Erweiterungen für den Einsatz als DWH, z.B. für das performante Einfügen, erweiterte Indextypen (z.B. Bitmap-Index) oder SQL-Erweiterungen für die Abfrage (z.B. CUBE-Operator) [4].

4 Zusammenfassung und Ausblick

In diesem Beitrag wurde der Aufbau eines DWS auf Basis der Open-Source-Produkte JPivot, Mondrian und MySQL behandelt. Dabei wurde auf die Architektur und die Funktionsweise eingegangen. Bei der Installation und Konfiguration des Systems sowie beim Nachbau einer Reihe von aus anderen Projekten bekannten Berichten wurden Erfahrungen gesammelt, die in einem Vergleich mit kommerziellen Systemen festgehalten wurden. Als Fazit lässt sich an dieser Stelle sagen, dass die untersuchten Systeme aufgrund der

gemachten Erfahrungen (insb. Schwierigkeiten bei der Installation, Komplexität des Erstellens neuer Berichte, mangelnde Performanz bei großen Datenmengen) momentan für den praktischen Einsatz keine Alternative zu den kommerziellen Systemen darstellen.

Als weitere Aufgaben im Themenfeld Open-Source-DWS sehen wir u.a. folgendes:

- Als Alternative zu Mondrian ist für Mitte des Jahres 2005 das Linux-basierte System Jedox [6] angekündigt. Eine Gegenüberstellung der beiden Systeme ist hier von Interesse.
- Auf der DB-Schicht können noch weitere Untersuchungen durchgeführt werden. Bisher wurde lediglich eine MySQL-Standardinstallation verwendet. Denkbar sind hier die Untersuchung von Konfigurationen und Optimierungsmaßnahmen sowie Performance-Vergleiche mit kommerziellen Systemen oder anderen Open-Source-Produkten, wie z.B. PostgreSQL.
- Schließlich wurde bisher der ETL (Extraktion-Transformation-Laden)-Bereich, der für die Überführung der Daten aus ihren Quellsystemen in das DWH zuständig ist, gar nicht betrachtet. Hier ist vor allem eine Untersuchung der Projekte clover.ETL [2] und Octopus [12] interessant.

Literatur

[1] Althaus, Michaela: *Auswirkungen der Schemagestaltung auf das Antwortzeitverhalten in einem Data Warehouse*. Berufsakademie Stuttgart, Außenstelle Horb, Studienarbeit, 2004.

[2] Homepage clover.ETL-Projekt. http://cloveretl.berlios.de/.

[3] Homepage Cognos. http://www.cognos.com/.

[4] Bauer, Andreas und Holger Günzel (Hrg.): *Data Warehouse-Systeme – Architektur, Entwicklung, Anwendung*. Zweite Auflage, dpunkt-Verlag, 2004.

[5] Devlin, Barry: *Data Warehouse*. Addison-Wesley Professional, 1996.

[6] Homepage Jedox Open Source-Projekt. http://www.opensourceolap.org/.

[7] Homepage JPivot. http://jpivot.sourceforge.net/.

[8] Kimball, Ralph: *The Data Warehouse Toolkit*. Zweite Auflage, John Wiley & Sons Inc., 2004.

[9] Homepage Microstrategy. http://www.microstrategy.com/.

[10] Homepage Mondrian. http://mondrian.sourceforge.net/.

[11] Homepage MySQL. http://www.mysql.com/.

[12] Homepage Octopus-Projekt. http://octopus.objectweb.org/.

[13] Pfrommer, Benjamin: *Konzeption und Entwicklung eines Data Warehouse auf Basis der Open Source Produkte JPivot und Mondrian*. Berufsakademie Stuttgart, Studienarbeit, 2005.

[14] Schnaidt, Benno: *Tuningmaßnahmen in einem Data Warehouse*. Berufsakademie Stuttgart, Außenstelle Horb, Studienarbeit, 2005.