Dieser Artikel beschreibt das WorkbookNG, ein modernes Web-Frontend, und gibt Einblick in die Erfahrungen, die die Autoren während der Entwicklung gemacht haben. Dabei wurde neben SCRUM als Vorgehensmodell auf das Rails-Framework und JRuby gesetzt. Das WorkbookNG ist durch und durch „Next Generation" – Captain Jean-Luc Picard wäre stolz. Der Artikel zeigt, warum für dieses Web-Frontend eine Oracle-Datenbank mit dem IQIMS-Framework Sinn ergibt. Schließlich spielt das Thema „Security" mit neuartigen Erscheinungen wie Wikileaks oder Anonymous eine immer wichtigere Rolle.

WorkbookNG auf IQIMS-Basis

Jürgen Schiefeneder und Michael Schröder, Data-Warehouse GmbH

Als Star-Trek-Fans haben sich die Entwickler der Data-Warehouse GmbH immer schon für fremde Welten und Galaxien interessiert, die nie zuvor ein Mensch gesehen hat. Schnell waren genug motivierte Mitarbeiter für ein Abenteuer in den unendlichen Weiten der Open-Source-Software gefunden. Das Ziel der Crew: Überwinden Sie die Grenzen klassischer Software-Entwicklung, indem Sie ein intuitives Web-Interface des nächsten Jahrhunderts mit den Methoden der agilen Software-Entwicklung erstellen. Als Schutzschilde und Deflektoren konnte dabei voll auf die Features des IQIMS-Frameworks gesetzt werden (IQIMS = Integrated Quality based Information Management System).

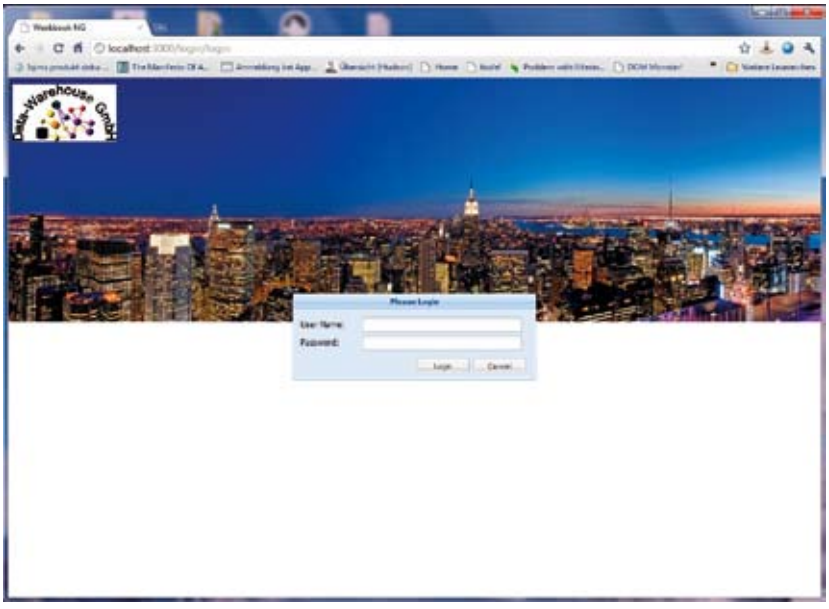

Abbildung 1: Die Login-Aufforderung des Workbooks „America"

Der Weg zum WorkbookNG

Mit dem WorkbookNG-System verhält es sich wie mit jedem Kommunikationsmittel: Eines allein ergibt keinen Sinn. Erst wenn mindestens zwei Gesprächspartner beschließen sich ein Telefon zuzulegen, entsteht ein Mehrwert. Beim Workbook sind die Telefone die unterschiedlichen Instanzen. Jede Instanz ist dabei ein Knotenpunkt im Austausch von Daten. Ein Knotenpunkt besitzt die exakt definierten Kontaktdaten anderer Workbooks, die für ihn Gesprächspartner darstellen. Neue Gesprächspartner können von autorisierten Benutzern angelegt werden. Das WorkbookNG-System kümmert sich dann um den Rest – es überprüft, ob die anderen Instanzen die für sie definierten Daten bekommen haben. Man kann sich das wie ein Telefongespräch vorstellen, wobei ein Workbook das andere anruft. Der Dialog selbst läuft dann in etwa folgendermaßen ab:

Dialog zwischen WorkbookNG America und WorkbookNG Europe:

*„Hier ist WorkbookNG America, Benutzer: ‚***' Autorisation: ‚***' – Gibt es neue Daten für mich?"*

„Ja, die folgenden neue Daten-Pakete liegen zum Abholen bereit: Paket4711, Paket0815, Paket123."

„Ok, bitte schicke mir Paket4711."

Der Vorteil der kommunizierenden Workbooks liegt darin, dass Datenpakete einfach, sicher und gezielt ausgetauscht werden können. Für die Bedienung kann per Browser auf ein Workbook zugegriffen werden. Benutzer können so neue Daten-Pakete hochladen. Um die Verifizierung der Daten-Pakete muss sich der Benutzer keine Sorgen machen, das System kümmert sich darum. Versucht er doch einmal, eine beschädigte oder nicht zugelassene Datei einzustellen, wird er sofort auf den Fehler hingewiesen. Selbstverständlich können Pakete typisiert und gruppiert werden.

Ein Open-Source-Abenteuer

In der Entwicklungsphase des WorkbookNG-Prototyps im Jahr 2010 er-

lebten die Entwickler plötzlich ein „blaues Open-Source-Wunder“. Vollkommen vertieft in das Rails-Framework und JRruby wurde für die sichere Kommunikation zwischen den Workbooks auf das JRuby-Gem „httpclient“ gesetzt („Gems“ erlauben die einfache Einbindung frei verfügbarer Programm-Bibliotheken). Beim Testen des Interfaces stellten die Entwickler einen Fehler in der Bibliothek fest, und wie es im Open-Source-Umfeld so üblich ist, wurde der Fehlerbericht dazu beim Besitzer des Gems eingestellt. Dieser reagierte prompt und ließ wissen, dass er sich die Sache ansieht. Sechs Wochen lang hörte man nichts mehr vom Gem-Besitzer [nahi] und war schon damit beschäftigt, einen Ersatz für die sehr umfangreiche Bibliothek zu stricken. Als dann plötzlich wieder ein Lebenszeichen von [nahi] gesendet wurde, war man erleichtert und ziemlich überrascht. Seine Botschaft an die Entwickler: „Der von euch entdeckte Fehler stellte eine kritische Sicherheitslücke in der Bibliothek dar, leider durfte ich euch nicht mit Details versorgen, bis wir das Problem behoben hatten. Danke für eure Geduld, das Problem ist in der aktuellen Version gelöst“. Mit der neuen Version war dann tatsächlich alles in Ordnung und die Testläufe waren endlich erfolgreich.

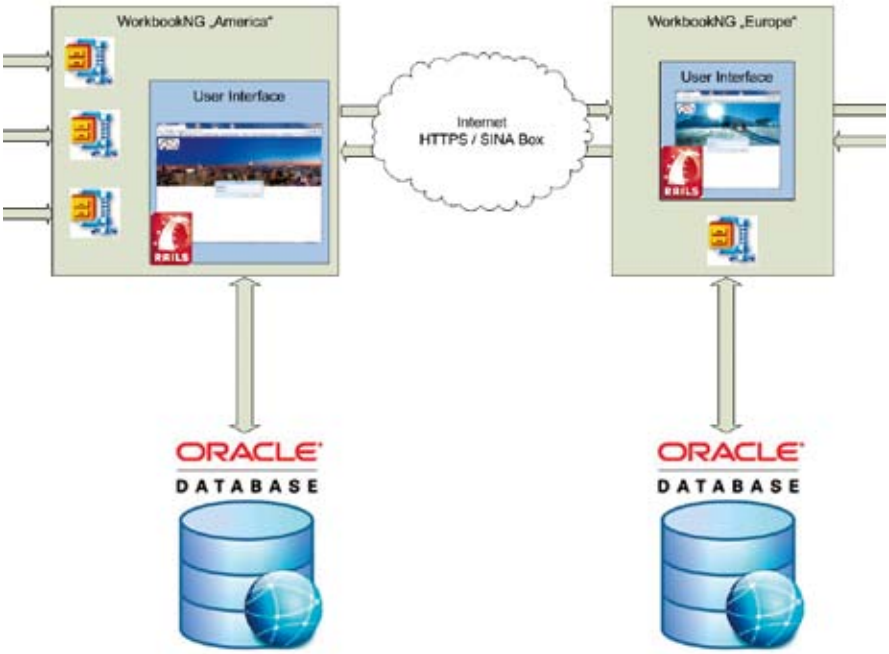

Abbildung 2: Vorhandene Schnittstellen im WorkbookNG

Die IQIMS-Basis

Da sich die Data-Warehouse GmbH zumeist in sehr sensiblen Datenbereichen bewegt, war schnell klar, dass auch das Workbook der nächsten Generation entsprechende Anforderungen erfüllen soll. Wer Datenpakete über das WorkbookNG austauscht, möchte das gezielt tun und verhindern, dass unberechtigte Personen darauf Zugriff bekommen. Um diesen Zugriff selbst im „Worst-Case“-Szenario eines Datenbank-Diebstahls zu unterbinden, wurde auf das IQIMS-Framework gesetzt. Stellen Sie sich einmal das Gesicht eines (Daten-)Bank-Räubers vor, der gerade erfolgreich Ihren IQIMS- (Datenbank-)Tresor gestohlen hat, diesen aufbricht und dann feststellen muss, dass sich im Inneren ein weiterer Tresor befindet, den er jedoch nicht knacken kann, da dieser innere Tresor aus Millionen kleiner Einzelteile besteht. Erst wenn alle Einzelteile zusammengepuzzelt sind, kann der innere Tresor geöffnet werden. Dieser verzweifelte Ausdruck im Gesicht des (Daten-)Bank-Räubers ist das, was die Spezialisten des IQIMS-Frameworks antreibt.

Das Konzept des WorkbookNG nochmals kurz zusammengefasst:

- Asynchrone Kommunikation über WebServices
- Point-to-Point-Kommunikation (keine weitere SOA-Infrastruktur notwendig)
- Schnittstelle, die auch von anderen Systemen genutzt werden kann
- interne Abbildung von Workflows
- definierter Benutzerkreis
- Benachrichtigungen über Ereignisse im Workflow (über tixxle oder E-Mail, SMS)
- Nachvollziehbares Logging aller Workflow-Events

Das logische Datenbank-Framework

Als Basis für das Workbook dient hierbei ein Datenbank-Framework, das unter anderem folgende Erweiterungen zu Oracle hat:

- Metadatengesteuertes, mehrschichtiges, objektorientiertes Data-Warehouse-Repository
- Logisches Datenmodell (Darstellung einer transparenten und historisierten, auf eine Zeitachse bezogenen, bidirektionalen Sicht)
- Automatisches, protokolliertes Auditing
- Trennung von fachlichen und physikalischen Primärschlüsseln (Dies bietet die Möglichkeit einer Normalisierung der Daten, ohne die ursprüngliche Datenform, beispielsweise für Datenexporte, zu verlieren.)
- Abstraktion und dynamische Codegenerierung (kein zusätzlicher Aufwand für die Pflege von Views, Indizes, Primär- und Fremdschlüsseln notwendig)
- Reduzierung inhaltlich gleicher Datentypen durch effektive Referenzierung und Abspeicherung in allgemeine, nicht abgeleitete Datentypen
- Kategorisierung von Datenfeldinhalten, das heißt es können verschiedene Ausprägungen von Dateninhalten abhängig vom gesetzten Kontext gespeichert werden (etwa Sprache oder Währung).

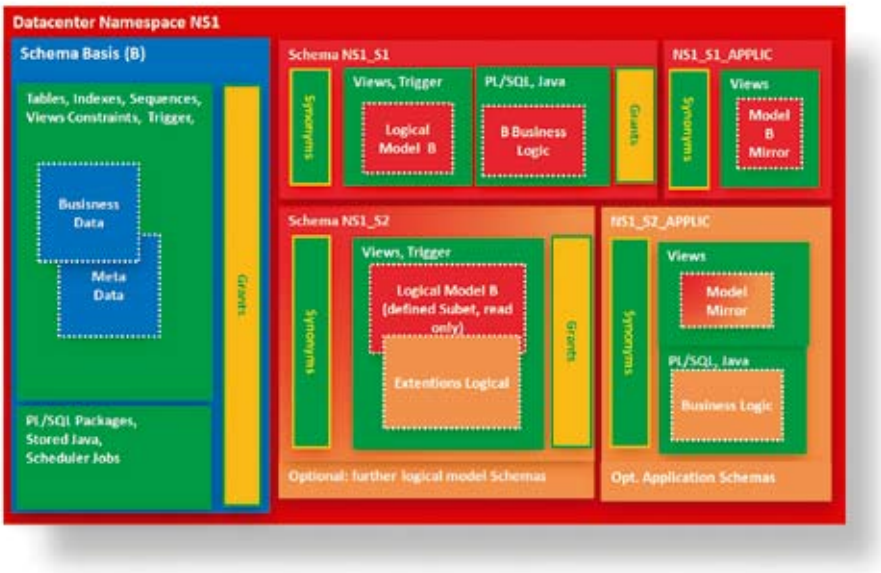

Abbildung 3: Schemamodell

Technischer Hintergrund des Datenbank-Frameworks

Basierend auf einer Oracle Enterprise Edition (aktuell 10.2.0.5, 11.2.0.1) bietet das Framework die Grundlage für eine automatisierte, historisierte Speicherung, die dem objektorientieren Ansatz verschrieben ist. Realisiert wird dies durch ein physikalisches Datenmodell, das aus folgenden Tabellen besteht:

- Objektklassen: repräsentiert die Tabelle im Repository
- Objektattribute: gruppiert die Spalten der Objektklasse
- Objekthierarchien: Vererbung von Objektattribut-Eigenschaften
- Objektprofiltexte, / -werte, / -zeiten, / -dateien: Speicherung der Daten
- Objektprofilreferenzen: Verwaltung der Referenzen zwischen Objektklassen

Die technische Basis des Frameworks bilden neben diesen Tabellen Indizes, Sequences, Trigger, PL/SQL-Packages und Java-Stored-Procedures. Die Grundidee dieser Architektur ist, dass über dem normalen Oracle-Schema (Schema-Basis) mit seinen physikalischen Tabellen (inklusive Framework-Logik) ein weiteres Schema (Layer) liegt. In diesem werden ausschließlich Views angezeigt, deren Daten aus dem Basis-Schema abgeleitet werden. Jegliche Datenänderung von angemeldeten Benutzern wird über Views gesteuert. Diese repräsentieren die üblichen Tabellen, mit denen der Benutzer arbeitet. Die Struktur (Spalten, Datentypen) der Views wird im Repository (Schema-Basis) historisiert gespeichert und dient einem Codegenerator (PL/SQL-Packages) als Bauplan der View-Erzeugung.

Diese Ansichten zeigen die Dateninhalte abhängig von einer gesetzten Referenzzeit an. Hierbei wird – gegenüber der klassischen Data-Warehouse-Idee, bei der zu jedem Datensatz in einer Tabelle die Gültigkeit (von / bis sowie eventuell eigene-ID) mit abgespeichert wird – die Gültigkeit separat und explizit, das heißt in Abhängigkeit von ihrer Objekteigenschaft (zeilen- und spaltenweise), intern abgespeichert. Das ermöglicht, dass nicht nur ein kompletter Datensatz einen Gültigkeitszeitraum besitzt, sondern dass einzelne Werte (Objektwerte) mit einem Gültigkeitszeitraum abgespeichert werden (siehe Abbildung 4). Der Benutzer sieht einfache, klassische Views (eines Schemas), bei denen die Abarbeitungslogik in Triggern versteckt wird. Die Referenzzeit wird Datenbank-Session-spezifisch gespeichert und erlaubt somit mehreren Benutzern gleichzeitige Lese- und Schreibzugriffe bei unterschiedlicher Referenzzeit. Ein über dem Oracle-Schema liegendes Benutzermanagement erlaubt dem Framework-Administrator, weitere Rechte (wie etwa auf Spaltenebene Select-, Insert-, oder Update-Recht) einzustellen.

Diese Views können auch noch mit weiterer Business-Logik erweitert werden (siehe Abbildung 3: Schema NS1_S2). Diese werden ebenfalls, wie die Standard-Views (NS1_S1), an das Schema NS1_S1_Applic beziehungsweise NS1_S2_Applic weitergereicht. Hier kann nun zum Beispiel mit Drittanbieter-Software auf das Schema zugegriffen werden, ohne die Business-Logik zu sehen und ohne explizit zugewiesene Rechte auf die Verarbeitung zu besitzen. Auch werden hier Trigger, Grants etc. verborgen und sind durch eine View-on-View-Methodik etwa einem SQL*Plus-Benutzer nicht sichtbar.

Die technische Idee hinter dem Applic-Schema basiert darauf, dass auch einem versierten DB-Benutzer die Views als Schnittstelle weitergereicht werden können, ohne dass die Oracle-Trigger-Logik offen dargelegt wird. Auch bietet Oracle ab der Version 10*g* einen Optimizer, der die View-on-View-Architektur in Verbindung mit einer logischen Modelldefinition besser unterstützt und in akzeptabler Performance abarbeitet. Standardmä-

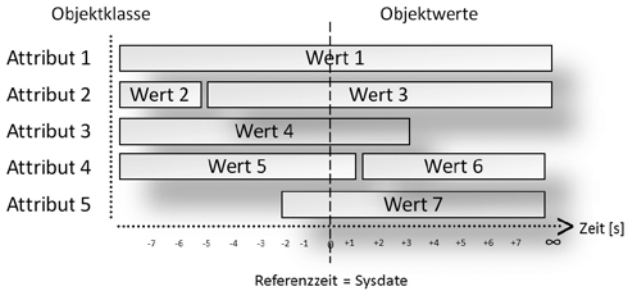

Abbildung 4: Zusammenhang zwischen Attributen und Objektwerten

ßig wird mit Realtime-/Echtzeit-Daten gearbeitet.

Durch Verändern der Referenzzeit werden, wie auf einer Zeitachse, eventuell andere Daten dargestellt (siehe Abbildung 4). Automatisch durch das Framework setzt der angemeldete Benutzer bei Datenänderungen seinen Zeitstempel (Benutzername, Zeitpunkt der Änderung, referenzzeitabhängiger Gültigkeitswert). Dieser Zeitstempel bezieht sich auf jede Attributebene, das heißt jegliche Datenfeldänderung wird mit dem Benutzernamen, der Uhrzeit und der gesetzten Referenzzeit protokolliert. Wird eine frühere Zeit eingestellt, so wird die Anzeige der Daten, wie auf einer Zeitachse, rückwirkend dargestellt und die Änderungen werden anhand der gesetzten Referenzzeit durchgeführt (siehe Abbildung 5). Ebenfalls kann hierüber das Datenmodell (Gültigkeitszeitraum von Objektklassen, Objektattributen) per Schema-Codegenerator konform mit den Dateninhalten erzeugt werden, falls im Laufe der Zeit Datenmodelländerungen stattgefunden haben.

Sicherheit

Wenn in einem sehr sensiblen Umfeld gearbeitet wird, muss auf die akkurate Einhaltung von Sicherheitsrichtlinien geachtet werden. Hier spielen unter anderem ein sicherer Datentransfer (dafür wird HTTPS oder SINA verwendet), verlässliche, historisierte und protokollierte Datenspeicherung sowie Sensibilisierung der Mitarbeiter eine entscheidende Rolle. Jede Anmeldung am Framework erfolgt in zwei Schritten. Der erste Schritt ist eine Oracle-Anmeldung, die noch keinen Datenzugriff erlaubt. Erst im zweiten Schritt wird durch eine Anmeldeprozedur des Frameworks der Zugriff auf die Daten möglich. Dadurch wird eine hervorragende Daten-Sicherheit gewährleistet. Selbstverständlich wird durch die Applikation ein sehr feines Rechtekonzept verfolgt, sodass nur berechtigte Personen Datenänderungen durchführen bzw. Dateninhalte sehen dürfen. Das sind unsere Schutzschilde und Deflektoren im Kampf gegen Daten-Räuber. Schließlich muss die Cyber-Kavallerie für jeden verhinderten Daten-Diebstahl einmal weniger ausreiten.

Fazit

Das WorkbookNG wurde als unterstützendes Tool entwickelt, um dem Benutzer ein Werkzeug zur Verfügung zu stellen, das für sicheren Datentransfer, das Abbilden und Anbinden des Geschäfts-Workflows sowie für die Benachrichtigung über Ereignisse im Workflow sorgt. Es ist gelungen, mit Open-Source-Frameworks eine Web-2.0-Oberfläche zu erstellen, die dem Benutzer ein angenehmes und übersichtliches Arbeiten ermöglicht und ihn dabei mit allen notwendigen Informationen versorgt. Durch das erweiterte Framework konnte auf eine sichere und flexible Datenbasis aufgebaut werden, die für die verlässliche und historisierte Datenspeicherung für das WorkbookNG sorgt.

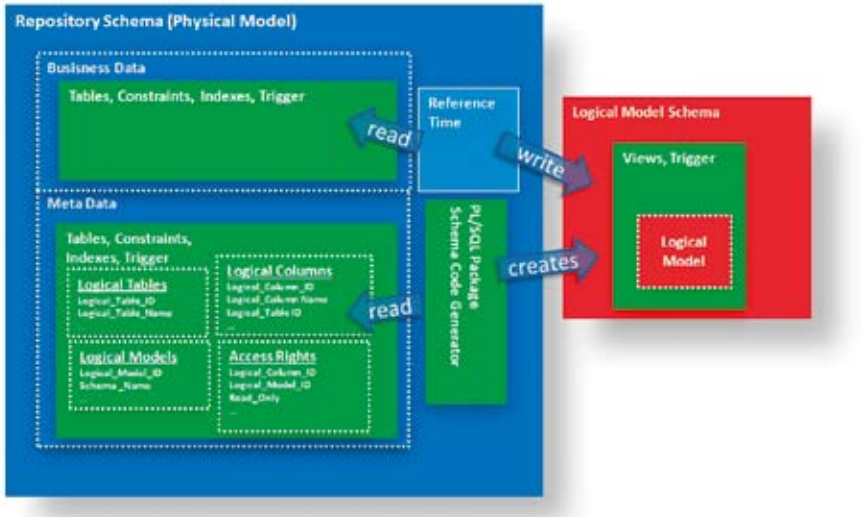

Abbildung 5: Schematische Darstellung der technischen Realisierung zu den historisierten Ansichten

Jürgen Schiefeneder (rechts)
und Michael Schröder
Data-Warehouse GmbH
j.schiefeneder@datawh.de
m.schroeder@datawh.de