Automatisierte Migration von Legacy-Dateien in relationale Datenbanken

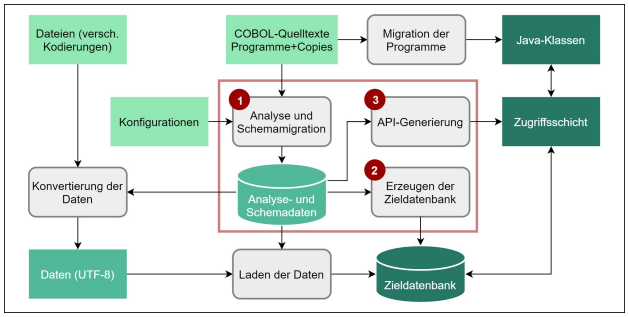
Felix Graßler
pro et con Innovative Informatikanwendungen GmbH, Reichenhainer Straße 29a, 09126 Chemnitz
felix.grassler@proetcon.de

Abstract

Dieses Paper basiert auf der Masterarbeit des Autors [5]: Große Unternehmen setzen noch vielfach auf historisch gewachsene Legacy-Systeme mit dateibasierter Datenhaltung. Oft besteht der Wunsch, diese durch moderne Systeme abzulösen. Das Ziel dieser Arbeit war die Entwicklung einer allgemeinen Technologie für die werkzeuggestützte Migration einer dateiorientierten Datenhaltung zu einer relationalen Datenbank (DB). Dies wird im Kontext einer zeitgleichen COBOL-zu-Java-Programmmigration des umgebenden Softwaresystems betrachtet. Um auch die Migration großer Datenbestände zu ermöglichen, wird eine hohe Automatisierung angestrebt. Der Fokus dieser Arbeit liegt auf zwei Aspekten: Der Migration des Datenschemas (dateiorientiert vs. relationale DB) sowie einer Migration der Datenzugriffe des umgebenden Softwaresystems. Bei letzterem sind die Unterschiede in den Programmiersprachen (COBOL vs. Java) sowie die abweichenden Arbeitsweisen (prozedural vs. SQL) zu beachten.

1 Überblick

Im Bereich der Datenmigration existieren bereits einige theoretische Ansätze sowie erfolgreich absolvierte Projekte [1][2][3]. Das Ziel dieser Arbeit war eine Vertiefung der bestehenden Ansätze und Technologien. Der Fokus lag besonders auf einer offenen, konfigurierbaren Vorgehensweise sowie einem hohen Grad an Automatisierung. Als Alternative zu einer Eigenentwicklung wurden auch existierende ETL-Technologien untersucht. Damit können Datenbestände aus einem Quellsystem extrahiert, transformiert und in ein Zielsystem geladen werden [6][4]. Es kann jedoch festgehalten werden, dass diese Werkzeuge für eine Datenmigration im Sinne dieser Arbeit nicht geeignet sind. Daher wurde stattdessen der nachfolgende Ansatz gewählt.



Wie dargestellt, umfasst eine Migration der Datenhaltung mehrere Aspekte. Es ist das Datenschema des Legacy-Systems zu ermitteln. In COBOL sind die Informationen über die verwendeten Dateien und die Strukturierung der Datensätze nicht Bestandteil der Datenhaltung (also der Dateien). Stattdessen sind sie im umgebenden Programmsystem enthalten. Sie werden in einer Analysephase extrahiert und sind die Grundlage für die Schemamigration (1). Bei dieser entsteht eine Menge von Tabellen- und Spaltenbeschreibungen, mit denen die Datenbestände des Legacy-Systems in einer relationalen DB abgebildet werden können. Darauf basierend erfolgt die Erzeugung der Ziel-DB (2). Im Zielsystem unterscheidet sich die Art der Zugriffe auf die Daten in mehreren Aspekten. Daher ist die Generierung einer Zugriffsschicht notwendig (3).

2 Analyse des Programmsystems

In COBOL existieren verschiedene Organisations- und Zugriffsarten von bzw. auf Dateien. In einer Datei können unterschiedliche Arten von Datensätzen gespeichert werden, welche sich in Länge, Strukturierung und Datentypen unterscheiden. Diese und weitere Informationen stellen die Grundlage der weiteren Schritte dar. Sie sind Bestandteil der COBOL-Programme und müssen in einer Analysephase extrahiert werden. Dafür stehen bei pro et con ausgereifte Werkzeuge zur Verfügung [2].
Die Programmiersprache COBOL bietet einige Features, welche kein Äquivalent in Java besitzen, aber Einfluss auf die Datenhaltung haben. Diese sind insbesondere Redefinitionen und Filler. Mit Redefinitionen können Teilbereiche eines Datensatzes unterschiedlich interpretiert werden. Filler kommen i. d. R. zum Einsatz, wenn mehrere Strukturbeschreibungen jeweils nur einen Teil des Datenbereiches beschreiben. Alle genannten Informationen sind in einem Analyseprozess aus den Programmen zu gewinnen. Dadurch entsteht das Datenschema des Legacy-Systems.

3 Migration des Datenschemas

Bei der Transformation des Datenschemas sind verschiedene Aspekte zu beachten:
(1) Die Abbildung von Dateien zu Tabellen. Aus einer Datei können eine oder mehrere Tabellen hervorgehen, bspw. in Abhängigkeit der verwendeten Strukturbeschreibungen sowie der Anzahl unterschiedlicher Datensatzarten, welche in derselben Datei gespeichert werden. Zudem sind Spalten für Verwaltungsinformationen vorzusehen. Damit wird Verhalten explizit im Zielsystem nachgebildet, welches im Legacy-System implizit umgesetzt wurde. Das betrifft z. B. die garantierte Reihenfolge der Datensätze bei sequentiellen Dateien sowie die Schlüsselverwaltung bei indexsequentiellen und relativen Dateien. Das entwickelte Werkzeug unterstützt die Definition allgemeiner und dateispezifischer Regeln für die Abbildung.
(2) Große Datenfelder können in Untertabellen ausgelagert werden. Das erhöht die Übersicht und Wartbarkeit des Zielschemas. Eine Auslagerung ist z. T. technisch notwendig. Auch hier können Regeln für spezifische Datenfelder definiert werden, sowie allgemeine Regeln in Abhängigkeit der Anzahl der Datenelemente, der Größe in Zeichen und der Verschachtelungstiefe der entstehende Tabelle.
(3) Redefinitionen beschreiben denselben Speicherbereich unterschiedlich. Es ist jeweils zu entscheiden, welche Beschreibung(en) in das Zielschema übernommen wird/werden. Das kann für spezifische Redefinitionen erfolgen, so-

wie durch allgemeine Regeln. So kann bspw. automatisch stets die detaillierteste Redefinition gewählt werden.
(4) Die Datentypen des Zielsystems müssen mit den Datenbeständen des Legacy-Systems zusammenpassen. Dafür sind Informationen aus der Analysephase sowie eine Kenntnis des exakten Ziel-DBS notwendig. Daher kann letzteres für die Migration konfiguriert werden.
(5) Bezeichner können nicht immer 1:1 übernommen werden. Das liegt z. B. an unterschiedlichen Namenskonventionen in Legacy- und Zielsystem. Zudem stellen unterschiedliche DBS eigene Restriktionen an mögliche Bezeichner, bspw. bei der Länge und der Auswahl der erlaubten (Start)Zeichen. Für die Namenskonvertierung können allgemeine und spezifische Regeln definiert werden.

Die genannten Konfigurationen werden in separaten Dateien vorgenommen. Dies ermöglicht eine Wiederholung der Schemamigration mit verschiedenen Konfigurationen. Zudem können diese in einer Versionsverwaltung abgelegt werden, um durchgeführte Schemamigrationen reproduzieren zu können. Beide Punkte sind im Kontext von komplexen Migrationsprojekten relevant. Das Ergebnis der Schemamigration ist eine Menge von Tabellen- und Spaltenbeschreibungen (das Datenschema des Zielsystems).

4 Datenzugriffe im Zielsystem

4.1 Erzeugen der Datenbank

Aus dem gewonnenen Zielschema entsteht im nächsten Schritt die DB des Zielsystems. Dafür werden DDL-Skripte mit *create table*-Statements generiert. Diese müssen den SQL-Dialekt des konkreten Ziel-DBS berücksichtigen. Sie übersetzen damit das Schema in konkrete Anweisungen für das spezifische DBS des Zielsystems. Um eine hohe Performanz bei Zugriffen auf ehemals index-sequentiell oder relativ organisierte Datenbestände zu gewährleisten, werden zusätzlich Indizes generiert.

4.2 Zugriffsschicht

Das umgebende Programmsystem wird von COBOL nach Java migriert und die dateibasierte Datenhaltung in eine relationale DB überführt. Dadurch unterscheiden sich bei Legacy- und Zielsystem sowohl die Organisation der Datenbestände, als auch die Datenzugriffe: (1) Im Legacy-System greifen die Programme direkt auf die Dateien zu. Im Zielsystem wird ein DBMS als Zwischenschicht genutzt. (2) Die Persistierung eines Datensatzes erfolgt im Legacy-System als zusammenhängender Speicherbereich, während im Zielsystem die Datenelemente separat gespeichert werden. (3) Zur Laufzeit werden im Zielsystem Datenklassen genutzt, wohingegen im Legacy-System ein Speicherbereich interpretiert wird. (4) Das Legacy-System und die migrierten Programme nutzen (logische) Dateioperationen wie READ, WRITE und START, um auf die Daten zuzugreifen. Im Zielsystem werden DB-Operationen genutzt.
Aus diesen Gründen ist eine Zwischenschicht für den Zugriff der migrierten Programme auf die Datenbestände erforderlich. Sie übersetzt logische Dateioperationen in DB-Operationen und übernimmt eine objektrelationale Abbildung. Damit werden DB-Spalten und Attribute der Java-Datenklassen einander zugeordnet und somit Datensätze und Java-Objekte ineinander übersetzt. Dabei ist eine hohe Performanz erforderlich. Aus diesem Grund wird für die Zugriffsschicht ein hybrider Ansatz gewählt: Eine Menge abstrakter Klassen setzt allgemeine Funktionalität um und Interfaces stellen den Programmen logische Dateioperationen zur Verfügung. Zusätzlich wird für jede Tabelle eine separate Hilfsklasse generiert. Diese übernehmen die objektrelationale Abbildung. Besonderheiten wie die Verwendung von Redefines und das Auslagern von Datenfeldern sind darüber abgebildet. Es wurden auch alternative Vorgehensweisen untersucht, wie z. B. die Nutzung von Reflection und Annotation Processing. Ersteres ist jedoch aufgrund der Performanz nicht geeignet und letzteres stellte große Herausforderungen dar, wenn im Legacy-System keine einfache 1:1-Zuordnung von einer Strukturbeschreibung zu einer Datei existiert.

5 Test der Datenmigration

Der Test des Gesamtprozesses erfolgt im direkten Vergleich COBOL vs. Java. Zunächst werden COBOL-Programme mit Dateiarbeiten ausgeführt. Dabei entstehen Referenz-Datenbestände. Anschließend werden die Programme nach Java übersetzt und anhand des hier beschriebenen Prozesses werden eine DB und die Zugriffsschicht generiert. Eine Ausführung der migrierten Java-Programme ergibt nun ebenfalls Datenbestände (in der DB). Ein Vergleich mit den Referenz-Datenbeständen stellt sicher, dass Schreiboperationen korrekt funktionieren. Für den Vergleich von Leseoperationen lesen die COBOL- und Java-Programme jeweils Datensätze aus den Dateien bzw. der DB ein und geben die Ergebnisse aus. Die so entstehenden Ausgaben können automatisiert verglichen werden.

6 Zusammenfassung

In der Arbeit wurden existierende Ansätze und Technologien für die Datenmigration erweitert. Es entstand ein automatisierter, konfigurierbarer Prozess, mit welchem Datenbestände aus Legacy-Systemen modernisiert werden können. Die Automatisierung ermöglicht eine praktikable Migration umfangreicher Systeme. Durch die offene Gestaltung können diverse Besonderheiten und Individualitäten abgebildet werden. Für eine vollständige Datenmigration ist, zusätzlich zu den hier beschriebenen Prozessen, eine Überführung der reinen Datenbestände notwendig. Diese war nicht Gegenstand der Arbeit.

Literaturverzeichnis

[1] Becker, C.; Kaiser, U.: Toolbasierte Software-Migration nach Plan. WSRE 2016

[2] Erdmenger, U.; Prof. Dr. Kaiser, U; Loos, A.; Uhlig, D.: Methoden und Werkzeuge für die Software Migration. WSRE 2008

[3] De Marco, A.; Iancu, V.; Asinofsky, I.: COBOL to Java and Newspapers Still Get Delivered. 2018.

[4] Biplob, Md. B.; Sheraji, G. A.; Khan, S. I.: Comparison of Different Extraction Transformation and Loading Tools for Data Warehousing. 2018.

[5] Graßler, F.: Untersuchung und Implementierung von Verfahren für die automatisierte Migration von Dateien in relationale Datenbanken. Masterarbeit, 2021. An TU Chemnitz Fakultät für Informatik.

[6] Vassiliadis, P.: A Survey of Extract-Transform-Load Technology. 2009.