Ein Tool-Set zur Datenbank-Analyse und -Normalisierung

Daniel Fesenmeyer, Tobias Rafreider, Jürgen Wäsch *
HTWG Konstanz

Abstract: In diesem Beitrag werden zwei Softwarewerkzeuge zur Datenbank-Analyse und -Normalisierung vorgestellt. *TANE-java* dient zur Extraktion von funktionalen Abhängigkeiten aus relationalen Datenbanken. *DBNormalizer* dient zur Normalisierung relationaler Datenbanken auf Basis funktionaler Abhängigkeiten. Ergebnis ist ein ausführbares SQL-Skript zur Schemamodifikation und Datenmigration. Die Werkzeuge können in der Datenbank-Ausbildung, aber auch in realen Projekten zum Refactoring existierender Datenbanken eingesetzt werden.

1 Einleitung und Motivation

Ein schwierig zu erfassendes Themengebiet im Bereich der relationalen Datenbanken ist die relationale Entwurfstheorie. Kommerzielle Datenbankentwurfswerkzeuge unterstützen den formalen Normalisierungsprozesse nicht oder nur unzureichend [F08, R08]. Die im Rahmen von [F08] analysierten freien "akademischen" Normalisierungswerkzeuge Database Normalizer (TU München [J04]) und Database Normalization Tool (Cornell University [S03]) arbeiten nicht auf realen Datenbanken, sondern nur auf vom Benutzer einzugebenden Relationenschemata. In diesem Beitrag werden zwei Softwarewerkzeuge zur Analyse und Normalisierung von relationalen Datenbanken auf Basis von funktionalen Abhängigkeiten vorgestellt: *TANE-java* und *DBNormalizer*.

DBNormalizer dient zur Normalisierung von relationalen Datenbanken und wird u.a. in der Datenbank-Ausbildung an der HTWG Konstanz verwendet, um den Studierenden die relationale Entwurfstheorie praktisch zu vermitteln. Neben der Nutzung in der Lehre ist aber auch der Einsatz in realen Projekten möglich. So können z.B. das Forward Engineering sowie die Analyse und das Refactoring von existierenden relationalen Datenbanken [AS06] im Rahmen von agilen Prozessen [A03] unterstützt werden.

Ein häufiger Kritikpunkt gegen den praktischen Einsatz der relationalen Entwurfstheorie ist der Aufwand zur Bestimmung von funktionalen Abhängigkeiten. Hier setzt das Werkzeug *TANE-java* an. Es extrahiert automatisch funktionale Abhängigkeiten aus einer relationalen Datenbankausprägung. Diese extrahierten funktionalen Abhängigkeiten können dann die Basis für die Normalisierung mittels *DBNormalizer* bilden. Ergebnis ist ein SQL-Skript für die Schemamodifikation und die Datenmigration, das dann auf der relationalen Datenbank ausgeführt werden kann (siehe Abbildung 1).

*Die Softwarewerkzeuge wurden im Rahmen von Diplomarbeiten an der HTWG Konstanz entwickelt [F08, R08]. Daniel Fesenmeyer arbeitet heute als Software Entwickler bei der Sybit GmbH.
Kontakt: juergen.waesch@htwg-konstanz.de

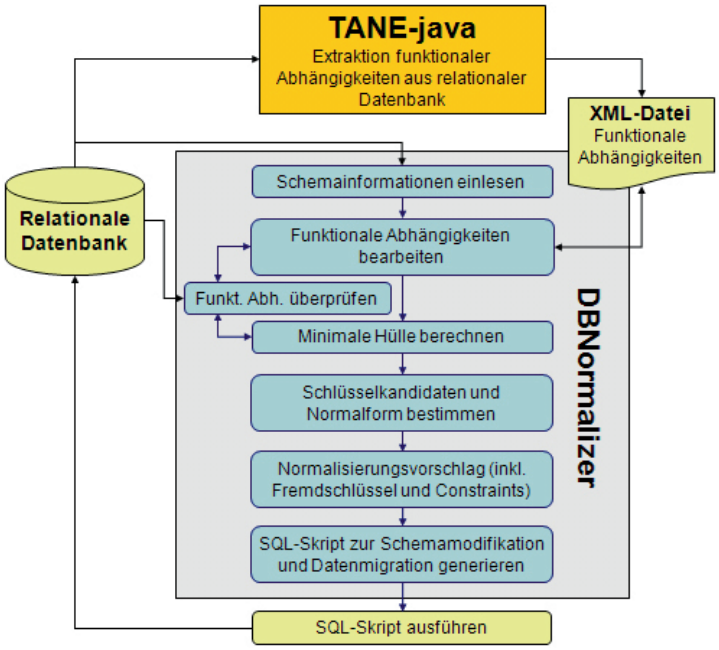

Abbildung 1: Normalisierungsprozess: Zusammenspiel zwischen *TANE-java* und *DBNormalizer*.

2 DBNormalizer

DBNormalizer ermöglicht die Normalisierung von Datenbankschemata bis zur Boyce-Codd-Normalform (BCNF). Das Werkzeug arbeitet auf realen Datenbanken, berücksichtigt im Gegensatz zu [J04, S03] auch Fremdschlüsselabhängigkeiten zwischen Relationen bei der Normalisierung und generiert einen Vorschlag zur Schemamodifikation als ausführbares SQL-Skript. Abbildung 2 zeigt die graphische Benutzungsoberfläche von *DBNormalizer*. Das Werkzeug basiert primär auf einem Synthesealgorithmus; weitere Details zur Realisierung sind in [WFR08, F08] zu finden. Im Augenblick bietet *DBNormalizer* folgende Funktionalität:

- Einlesen von Schemainformationen aus einer gegebenen relationalen Datenbank bzw. Eingabe und Bearbeiten von Datenbankschemata durch den Benutzer, falls keine relationale Datenbank verfügbar ist.
- Eingabe und Verwaltung von funktionalen Abhängigkeiten für die Relationen durch den Benutzer sowie Überprüfung der Gültigkeit von funktionalen Abhängigkeiten auf Basis der augenblicklichen Datenbankausprägung (falls vorhanden).
- Berechnung einer minimalen Hülle für funktionale Abhängigkeiten von Relationen, Berechnung von Attributhüllen, sowie Bestimmung *aller* Schlüsselkandidaten.
- Bestimmung der Normalform von Relationen (1NF, 2NF, 3NF oder BCNF). Dies beinhaltet die Berechnung, welche funktionalen Abhängigkeiten in einer Relation eine bestimmte Normalform verletzen.

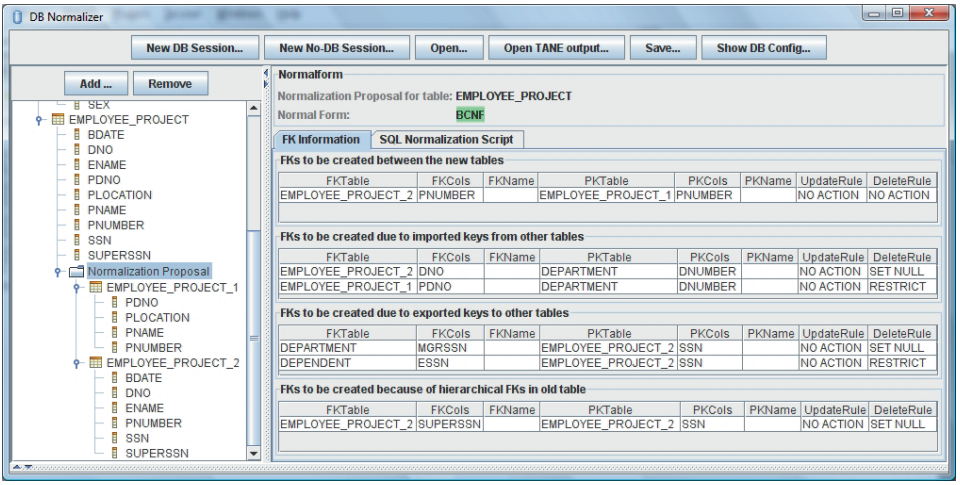

Abbildung 2: Graphische Benutzungsoberfläche von *DBNormalizer*.

- Erzeugen eines Normalisierungsvorschlags für ein Datenbankschema. Dies beinhaltet die Berechnung *aller* Schlüsselkandidaten für die neuen Relationen, die Berechnung der Fremdschlüsselbeziehungen für die Relationen des Normalisierungsvorschlags und die Bestimmung der Normalformen für die neuen Relationen (3NF oder BCNF).
- Erzeugung eines ausführbaren SQL-Skripts zur Transformation der Datenbank. Das SQL-Skript beinhaltet die Erzeugung der neuen Tabellen, die Datenmigration, das Löschen der alten Tabellen und das Erzeugen der Fremdschlüsselconstraints für die neuen Tabellen. Außerdem werden Views erzeugt, um existierende Anwendungen mit dem neuen Datenbankschema weiterhin nutzen zu können bzw. um die Portierung der Datenbankanwendungen zu vereinfachen. Im Falle von Oracle könnten für nicht-aktualisierbare Views auch INSTEAD-OF Trigger erzeugt werden.
- Import und Export von funktionalen Abhängigkeiten und Datenbankschemata als XML-Datei sowie Import der mittels *TANE-java* automatisch ermittelten funktionalen Abhängigkeiten einer relationalen Datenbankausprägung.

3 TANE-java

TANE-java ist ein Software-Werkzeug zur Extraktion von funktionalen Abhängigkeiten aus (großen) Datenbankausprägungen. Die extrahierten funktionalen Abhängigkeiten können als Eingabe für den Normalisierungsprozess in *DBNormalizer* dienen. *TANE-java* implementiert u.a. die in [H+99] vorgestellten Algorithmen zur Extraktion von funktionalen Abhängigkeiten. Diese Algorithmen sind durch das zur Suchraumeinschränkung verwendete Partitionierungsprinzip sehr effizient und ermöglichen auch die Extraktion von approximativen funktionalen Abhängigkeiten [H+99, R08].

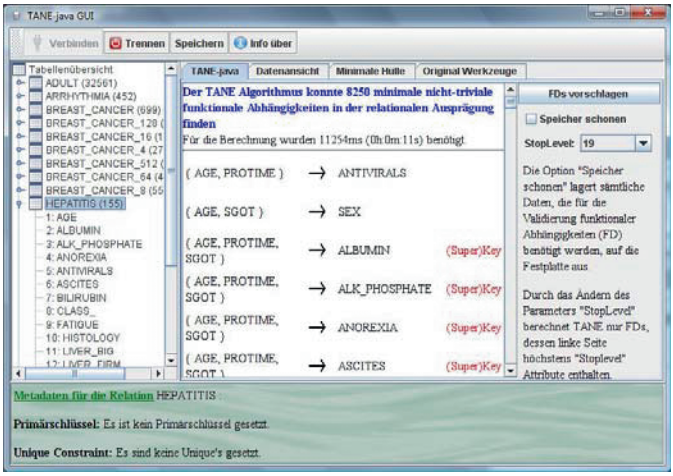

Abbildung 3: Graphische Benutzungsoberfläche von *TANE-java*.

Im Gegensatz zu der ursprünglichen C-Implementierung von TANE [TH] arbeitet *TANE-java* auf relationalen Datenbanken (nicht auf CSV-Dateien) und bietet eine graphische Benutzungsoberfläche (siehe Abbildung 3). Neben der Java-Implementierung der TANE-Algorithmen integriert das Werkzeug aber auch die ursprüngliche C-Implementierung. Hierzu werden zuerst die Daten aus der Datenbank in CSV-Dateien exportiert und dann die C-Programme gestartet. Diese wurden so modifiziert, dass sie nun ihre Ergebnisse in einem XML-Format liefern, das wiederum in *DBNormalizer* weiterverarbeitet werden kann (siehe Abbildung 1). Weitere Details zur Realisierung von *TANE-java* sind in [R08] zu finden.

Literatur

[AS06] S.W. Ambler, P.J. Sadalage: Refactoring Databases - Evolutionary Database Design. Addison-Wesley, 2006.

[A03] S.W. Ambler: Agile Database Techniques. Wiley, 2003.

[F08] D. Fesenmeyer: Design und Implementierung eines Software-Tools zur Normalisierung relationaler Datenbanken. Diplomarbeit, HTWG Konstanz, 2008.

[H+99] Y. Huhtala, Y. Kärkkainen, P. Porkka, H. Toivonen: TANE – An Efficient Algorithm for Discovering Functional and Approximate Dependencies. The Computer Journal, Vol. 42, No. 2, 1999.

[J04] E. Jürgens: Database Normalizer. Technische Universität München. http://home.in.tum.de/juergens/DatabaseNormalizer/index.htm/

[R08] T. Rafreider: Data-Mining Verfahren und Tools zur Unterstützung der Datenbankanalyse und Schemaoptimierung. Diplomarbeit, HTWG Konstanz, 2008.

[S03] S. Selikoff: The Database Normalization Tool. Educational Database Tools, Cornell University, 2003. http://dbtools.cs.cornell.edu/norm_index.html

[TH] TANE Homepage. http://www.cs.helsinki.fi/research/fdk/datamining/tane/

[WFR08] J. Wäsch, D. Fesenmeyer, T. Rafreider: Ein Normalisierungswerkzeug für die Datenbank-Ausbildung. Herbsttreffen der GI-Fachgruppe Datenbanken, Düsseldorf, 2008.