Versionierung in relationalen Datenbanken

Exposee einer Diplomarbeit

Stephan Rieche

24. Mai 2004

Betreuer: Prof. Dr. Ulf Leser,
Institut für Informatik, Humboldt-Universität zu Berlin

Zeitraum: 24.05.2004-23.11.2004

Motivation

Die Nutzer von Datenbanken stehen oft vor dem folgenden Problem. Die Daten repräsentieren nur den aktuellen Zustand der Welt. Da sich die Welt ständig ändert, unterliegen auch die Daten einem stetigen Wandel. Die alten Zustände werden heute in kommerziellen Datenbankmanagementsystemen in der Regel nur auf einem Log gespeichert, weswegen es schwierig ist, sich in einem wissenschaftlichen Papier auf einen bestimmten Datenbestand zu beziehen. Alte Zustände können nur wieder hergestellt werden, indem man die alten Daten zurückschreibt und damit den aktuellen Datenbestand abgesehen vom Log verliert.

Eine lineare Versionierung liegt vor, wenn die Umweltzustände linear voneinander abhängen, was bedeutet, dass ein neuer Umweltzustand immer aus einer Änderung des aktuellsten Zustands resultiert.

Ein allgemeinere Variante ergibt sich, wenn man z.B. im Data Warehousing verschiedene Szenarien aufstellen und analysieren möchte, ohne die für das Unternehmen wichtigen Originaldaten zu verändern. Ein gleichzeitiger Zugriff auf die Szenario- und die Originaldaten für Vergleiche ist bisher nur möglich, wenn in einer eigenständigen Datenbank die Originaldaten und in einer anderen die Szenariodaten liegen. In dieser Variante kann es eine nichtlineare Hierarchie zwischen dem Originalbestand und den Szenariodaten geben.

Um die oben genannten Probleme zu lösen, ist es sinnvoll, die Daten in mehreren Versionen vorzuhalten. Die Versionierung von Daten wurde schon für XML-Dokumente ([1]), objektorientierte Datenbanken ([4]) und Dateisysteme ([6])

untersucht. Leider wird die Versionierung von Daten in relationalen Datenbanken in heutigen Systemen nicht oder nur eingeschränkt unterstützt.

Es gibt zwei Ansätze, wie eine relationale Datenbank Versionierung ermöglichen kann. Einmal könnte man die Versionierung direkt in das Datenbankmanagementsystem implementieren ([2, 5]). Auf der anderen Seite könnte die Versionierung durch die Erweiterung des Datenbankschemas und Speicherung der Versionierungsinformationen in einem Table der Datenbank erfolgen - also auf der Ebene der Datenbank. Besonders zum zweiten Ansatz sind nur wenige Untersuchungen bekannt ([2, 3]), weswegen der zweite Ansatz in dieser Diplomarbeit verfolgt wird.

Zielsetzung

Ziel dieser Diplomarbeit ist es, Möglichkeiten zu finden, die in einer relationalen Datenbank Versionierung erlauben. Die Versionierung soll transparent für Altanwendungen sein. Verschiedene Möglichkeiten sollen beleuchtet, analysiert und qualitativ wie quantitativ bewertet werden. Als Schwerpunkt wird dabei lineare Versionierung betrachtet.

Vorgehen

Um das Ziel der Diplomarbeit zu erreichen, ist folgendes Vorgehen geplant. Die Versionierung soll durch eine Erweiterung des vorhandenen Datenbankschemas, die Definition von Triggern und Views erreicht werden. Dafür werden zunächst die Anforderungen an eine Versionierung der Ausprägung in relationalen Datenbanken erhoben. Anschließend werden in der Literatur bereits vorhandene Versionierungstechniken bezüglich ihrer Einsatzfähigkeit geprüft. Zu berücksichtigen sind dabei unter Anderen die folgenden Kriterien:

- Redundanz des Versionierungsmodells
- Komplexität für das Wiederherstellen einer alten Version
- Komplexität für das Löschen einer alten Version
- Komplexität für den Vergleich zweier Versionen
- Komplexität für das Lesen der aktuellen Version
- Komplexität für das Lesen einer alten Version
- Komplexität für das Einfügen, Update und Löschen eines Tupels einer bestimmten Version
- Verträglichkeit mit Mehrbenutzerbetrieb
- Umgang mit Referenzen zwischen versionierten Tupeln

Auf der Basis der Bewertung werden einige erfolgversprechende Ansätze auf einem Oracle Datenbanksystem implementiert. Die Verfahren werden an einer

echten Datenbank wie Swiss-Prot angewendet und gemessen. Ein Schemaumwandlungstool wird das Schema einer Datenbank für die Versionierung anpassen. Zusätzlich werden physikalische Optimierungsmöglichkeiten wie z.B. Indizes untersucht.

Literatur

[1] Chien, Shu-Yao ; Tsotras, Vassilis J. ; Zaniolo, Carlo: Version Management of XML Documents. In: *WebDB(Selected Papers)* (2000), S. 184–200

[2] Dadam, P. ; lum, V. ; Werner, H.-D.: Integration of Time Versions into a Relational Database System. In: Dayal, Umeshwar (Hrsg.) ; Schlageter, Gunter (Hrsg.) ; Seng, Lim H. (Hrsg.): *Tenth International Conference on Very Large Data Bases, August 27-31, 1984, Singapore, Proceedings*, Morgan Kaufmann, 1984. – ISBN 0–934613–16–8, S. 509–522

[3] Date, C.J. ; Darwen, Hugh ; Lorentzos, Nikos A.: *Temporal Data and the Relational Model*. San Francisco : Morgan Kaufmann Publishers - An imprint of Elsevier Science, 2003. – ISBN 1–55860–855–9

[4] Gançarski, Stéphane ; Jomier, Geneviève: A framework for programming multiversion databases. In: *Data & Knowledge Engineering* 36 (2000), Juni, S. 29–53

[5] Salzberg, Betty ; Jiang, Linan ; Lomet, David ; Kanoulas, Evangelos ; Barrena, Manuel ; Shan, Jing: A Framework for Access Methods for Versioned Data. In: *Conference on Extending Database Technology* (2004)

[6] Zhu, Ningning: *Data Versioning Systems*. New York: Experimental Computer Systems Lab, Februar 2003