Versionierung in relationalen Datenbanken

Stephan Rieche, Ulf Leser

14. Januar 2005

Zusammenfassung

Für viele wissenschaftliche Anwendungen ist es wünschenswert, gleichzeitig auf verschiedene zeitliche Zustände einer Datenbank zuzugreifen, also verschiedene *Versionen* der Datenbank im Zugriff zu halten. Heutige Datenbankmamagementsysteme unterstützen dies primär nicht. In der Arbeit werden Strategien zur Versionierung relationaler Datenbanken untersucht. Aufbauend auf einer Klassifikation möglicher Definitionen des Begriffs Versionierung werden aussichtsreiche Vorgehensweisen ausgewählt. Diese wurden implementiert und einer detailierten Analyse bzgl. ihrer Performanz bei unterschiedlichen Operationen untersucht. Die Messungen zeigen, dass die gewählen Strategien in nahezu allen Belangen deutlich performanter als ein kommerzielles Versionierungssystem sind.

1 Einleitung

Daten, die man in einer Datenbank hält, sind in der Regel nicht statisch. Im Laufe der Zeit verändert man diese Daten, weil sie überholt sind bzw. die Anwendung es so verlangt. In heutigen kommerziellen Datenbankmanagementsystemen ist die Arbeit mit alten Daten meistens nur möglich, wenn man diese aufwendig vom Log einspielt. Es gibt aber viele Anwendungen, in denen es wünschenswert ist, auf alte und neue Daten gleichzeitig zugreifen zu können. Im Bereich der Naturwissenschaften basieren beispielsweise Analysen immer auf dem aktuellsten Zustand einer Datenbank. Dieser Zustand hat sich zum Zeitpunkt der Veröffentlichung der Ergebnisse oft schon geändert, wodurch die Ergebnisse nicht mehr ohne weiteres nachvollziehbar sind. Das wird ermöglicht, wenn man die Daten in mehreren Versionen vorhält.

Als Version einer Datenbank bezeichnen wir eine konkrete Instanz der Datenbank mit allen enthaltenen Tupeln. Eine wichtige Anforderung an ein Versionierungssystem ist seine Transparenz für Altanwendungen. Prinzipiell gibt es zwei Möglichkeiten, dies zu erreichen: Durch Modifikation des DBMS, oder durch (für die Anwendungen unsichtbare) Änderungen des Schemas. In dieser Arbeit wird der letztere Ansatz verfolgt.

Versionierung von Daten wurde in letzten Jahren vor allem für XML und für objektorientierte Datenbanken untersucht. Versionierung in relationalen Datenbanken fällt in den Bereich der temporalen Datenbanken[1, 2], wurde dort bisher aber nur qualititativ analysiert. Ein Tool zur transparenten Versionierung mit entsprechenden Untersuchungen zur Performanz gibt es nicht. Es gibt ein kommerzielles Datenbankmanagementsystem, dass ein Werkzeug enthält, mit dem hierarchische Versionierung möglich ist (im Folgenden Workstate-Manager, WSM, genannt). Die von uns vorgeschlagenen Lösungen wurden durch Messungen untereinander und mit dem WSM verglichen. Details der Ergebnisse kann man in [3] finden.

2 Versionierungsstrategien

Bei einer linearen Versionierung, im Unterschied zur hierarchischen, hat jede Version nur eine Folgeversion. Nur dieses Szenario wurde untersucht, da es ausreichend für naturwissenschaftli-

che Anwendungen ist. Weiterhin kann man zwischen kontinuierlicher, d.h., jede Änderung eines Tupels erzeugt eine neue Version, und diskreter Versionierung, bei der das Anlegen einer neuen Version explizit vom Benutzer veranlasst wird, unterscheiden. Da wir die Forderung nach Transparenz für Altanwendungen gestellt haben, wird in der Arbeit nur kontinuierliche Versionierung betrachtet. Wir unterscheiden des weiteren zwischen Delta- und Zeitstempelmethoden. Der Zeitstempel besteht aus einem Intervall, dass die Start- und Endzeitpunkte für die Gültigkeit eines Tupel angibt. In theoretischen Überlegungen zeigten sich Delta- der Zeitstempelstrategie von vorneherein als unterlegen und wurden nicht weiter betrachtet. Zur Sicherung der Versionsinformation haben wir zwei Varianten untersucht. In der Strategie „mit Schattentabelle“ (DBmS) werden alle alten Versionen eines Tupels in einer separaten Tabelle gehalten, während in der Strategie „ohne Schattentabelle“ (DBoS) die Originaltabelle um Zeitstempelattribute erweitert wird. Auswirkungen auf Integritätsconstraints werden in [3] diskutiert.

3 Ergebnisse

Zum Vergleich der Strategien DBoS, DBmS, WSM und einer Datenbank ohne Versionierung (DB) wurden Daten von insgesamt sieben Releases von SWISS-PROT [4] benutzt. Es wurden die folgenden Operationen gemessen: Einspielen eines Releases, Punkt- und Bereichsanfragen und Anfragen über ganze Relationen in alten und in der aktuellen Version, Vergleich zweier Releases und das Löschen alter Versionen.

Die Messungen zeigten, dass der WSM konsistent langsamer als die selbst entwickelten Versionierungssysteme ist. Dabei muss aber beachtet werden, dass der WSM immer eine hierarchische Versionierung vornimmt. In Abbildung 1 ist beispielhaft das Ergebnis für die Messung einer Bereichsanfrage mit einem Join im aktuellen Release abgebildet. Die Varianten DB und DBmS erreichen praktisch identische Performanz, da beide Systeme das aktuelle Release in exakt den gleichen Tabellen halten. DBmS fällt ab, weil hier für jede Anfrage auf Tabellen zugegriffen wird, die mehrere Releases enthalten und entsprechend größer sind. Das schlechte Abschneiden des WSM tritt bei allen gemessenen Operationen zu Tage, wobei der Unterschied umso größer wird, je komplexer eine Anfrage ist.

Beim Import liegt DB und DBmS ungefähr gleichauf und deutlich vor DBmS. Der WSM ist um bis zu einen Faktor 7 (größtes SWISS-PROT Release) langsamer. Für den Zugriff auf die aktuelle Version ist DBmS am schnellsten, beim Zugriff auf alte Versionen war DBoS vorne. Die einzige Operation, bei der der WSM nicht das Schlusslicht bildet, ist der Vergleich unmittelbar benachbarter Versionen; liegen Versionen weiter auseinander, wird der WSM wieder deutlich von DBoS und DBmS überholt. In der Arbeit wurde ausserdem gezeigt, dass weitere Performanzsteigerungen durch Möglichkeiten moderner kommerzieller Datenbanken (spezielle Indexierung, Partitionierung) möglich sind.

In der Arbeit wurde damit gezeigt, dass unterschiedliche Versionierungsstrategien ihre Vorteile bei unterschiedlichen Anforderungen ausspielen. Zusammen mit der Analyse möglicher Versionierungssysteme bildet dies eine wichtige Grundlage zur Auswahl einer Strategie für konkrete Anwendungen.

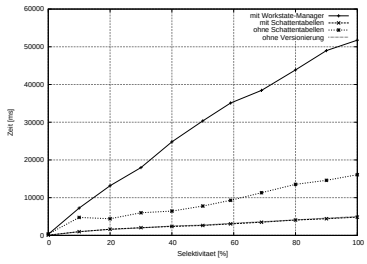

Abbildung 1: Messung einer Anfrage mit einem Join

Literatur

[1] 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

[2] 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

[3] Stephan Rieche, Betreuer: Leser U.: *Diplomarbeit: Versionierung in relationalen Datenbanken*. Berlin: Lehrstuhl für Wissensmanagement in der Bioinformatik, Mathematisch-Naturwissenschaftliche Fakultät II, Institut für Informatik, Humboldt-Universität zu Berlin, Dezember 2004

[4] Swiss Institute of Bioinformatics - University of Geneva, EMBL Outstation - EBI: *Swiss-Prot, Protein knowledgebase*. –
URL: http://www.expasy.org/sprot/
Zugriff am 01.07.2004