PROCESSOR OF TSQL2 ON RELATIONAL DATABASE SYSTEM

Jiří Tomek
Master Degree Programme, FIT BUT
E-mail: xtomek04@stud.fit.vutbr.cz

Supervised by: Marek Rychlý
E-mail: rychly@fit.vutbr.cz

ABSTRACT

This paper is focused on practical usage of temporal databases in current world and on design of TSQL2 language interpreter for SQL. In this time, there is no current usable implementation of temporal database system suitable for normal and transparent database use. Result of this work will be a working TSQL2 interpreter implemented in Java as JDBC adapter.

1. ÚVOD

V dnešní době jsou nejrozšířenějším typem databází databáze relační. Data ukládaná v těchto databázích jsou z hlediska databáze považována za aktuální, a pokud je třeba uchovávat historii dat, je třeba to řešit na úrovni aplikace a k datům přidávat například časová razítka.

Tento nedostatek řeší temporální databáze. Je to druh databáze, která pracuje s daty s ohledem na čas a podporuje operace s časem přímo v databázi. Není tedy třeba při ukládání nebo čtení dat explicitně specifikovat časové údaje, ale temporální databáze toto provádí automaticky. Zároveň uchovává historii dat uložených v databázi.

Použití temporální databáze se příliš neliší od běžné relační databáze. Jako dotazovací jazyk se používá upravená verze jazyka SQL s názvem TSQL2 [1][2]. TSQL2 přidává podporu pro práci s časem, ale zároveň je zpětně kompatibilní s SQL.

Temporální databáze podporuje běžné tabulky relační databáze, které se v terminologii temporálních databází nazývají **snímkové** a neobsahují žádná časově závislá data. Navíc přidává tabulky s podporou **časů platnosti** a **časů transakce**. Čas platnosti je údaj, který udává, kdy jsou daná data platná v reálném světě. Jedná se například o dobu života člověka. Čas transakce je naproti tomu údaj, udávající čas uložení dat v databázi. To je například doba, po kterou jsou data o dané osobě uložena v registru obyvatel.

V dnešní době je třeba ukládat obrovské množství dat a velká část těchto dat obsahuje nějakým způsobem časové údaje. Jedná se například o:

- **Medicínská data** – evidence prodělaných nemocí pacienta, předepsané léky, …
- **Finanční záznamy** – historie transakcí, historie stavu bankovního účtu, …

Cílem tohoto příspěvku je seznámit čtenáře s vlastním řešením temporálních databází. Prezentovaným výsledkem bude vytvoření překladače temporálního jazyka TSQL2, který je možné použít s existující relační databází a přidat tak podporu temporálních dat do jinak netemporální databáze.

2. EXISTUJÍCÍ IMPLEMENTACE TEMPORÁLNÍCH DATABÁZÍ

Nejúplnější implementací temporální databáze je produkt TimeDB [3]. Tento produkt podporuje většinu funkcí temporálních databází a je nejblíže praktickému použití. Bohužel obsahuje několik chyb v implementaci a již dlouho nebyla vydána nová verze. Zároveň se jedná o uzavřený software, takže ani není možné tyto chyby opravit.

Druhou implementací temporální databáze je produkt ChronoLog, který je založený na jazyce Prolog a z toho plyne jeho hlavní nevýhoda. Nepoužívá totiž jazyk SQL ale právě formu jazyka Prolog a je tedy v běžných databázových aplikacích jen těžko použitelný.

Posledním temporálním produktem je databáze Oracle, která ovšem nepodporuje temporální SQL. Obsahuje tzv. funkčnost „versioning“, která přidá k tabulce podporu času platnosti a tato tabulka potom umožňuje uchovávat temporální data. Kontrola této funkčnosti je ovšem řešena pomocí uložených procedur, které řídí nastavovaní časů platnosti a je tak nutno kombinovat SQL s těmito procedurami. Tento přístup je však oproti čistému SQL v praxi zbytečně komplikovaný.

3. ŘEŠENÍ

Snaha o vytvoření interpretu TSQL2 nad běžnou relační databází vede k řešení několika problémů.

3.1. PODPORA TEMPORÁLNÍCH DAT V RELAČNÍ DATABÁZI

Specifikace TSQL2 definuje podporu temporálních dat na úrovni jednotlivých tabulek databáze. Každá tabulka v databázi tedy může mít rozdílnou úroveň temporální podpory a některé tabulky vůbec temporální data podporovat nemusejí. Kromě snímkových tabulek, které neobsahují podporu temporálních dat, vyžadují všechny ostatní tabulky několik přidaných údajů, které popisují jejich úroveň podpory temporálních dat.

Tento problém je možné řešit jednoduchou úpravou informačního schématu databáze. Přidáním několika atributů do informačního schématu je možné rozlišovat mezi běžnými a temporálními tabulkami. Hlavní údaje, které je třeba u každé tabulky temporální databáze evidovat, jsou podpora času platnosti a podpora transakčního času. Interpret TSQL2 potom může jednoduše z upraveného schématu zjišťovat temporální vlastnosti tabulek a podle toho transparentně upravovat dotazy.

V případě, že není možné provést úpravu informačního schématu například z důvodu omezených oprávnění, je třeba vytvořit samostatnou tabulku obsahující nutná metadata pro ostatní tabulky temporální databáze.

3.2. UKLÁDÁNÍ TEMPORÁLNÍCH DAT V RELAČNÍ DATABÁZI

Podle typu tabulky může být třeba ke každému záznamu evidovat až dva temporální údaje. Jedná se o čas platnosti a čas transakce. Oba tyto časy jsou ve většině případů ve formě intervalu.

Nejvhodnějším způsobem uložení časové informace je tedy přidání dalších dvou atributů relace pro každý z ukládaných časů. Tyto atributy budou obsahovat hodnoty počátku a konce daného časového intervalu a podle použitého relačního databázového systému mohou mít tyto atributy různé datové typy. Takto je možné snadno za použití dostupných funkcí relační databáze ukládat interní časové údaje pro temporální data.

3.3. KONTROLA REFERENČNÍ INTEGRITY V TEMPORÁLNÍ DATABÁZI

Při použití referencí v temporální databázi narazíme na problém udržení referenční integrity, je-li odkazovaná tabulka s podporou času platnosti. Taková tabulka může pro jeden identifikátor obsahovat několik záznamů platných v různých časech a odkaz na tento identifikátor tak není jednoznačný. Tento problém je možné řešit pomocí databázových triggerů.

Pokud na temporální záznam odkazují jiné záznamy, trigger po změně odkazovaného temporálního záznamu a vytvoření nové verze automaticky aktualizuje odkazy v odkazujících tabulkách. Pokud tím vznikne situace, kdy čas platnosti odkazujícího záznamu není uvnitř času platnosti odkazovaného záznamu, musí se navíc odkazující záznam rozdělit na dva časy platnosti, aby zůstala zachována časová integrita.

V opačném případě, tedy při vkládání nového záznamu s odkazem do temporální tabulky, je nutné před samotným vložením provést kontrolu, zda je odkazovaný záznam platný ve stejném čase platnosti jako má vkládaný záznam. Pokud odkazovaná tabulka obsahuje více verzí daného záznamu, je třeba vybrat tu verzi, která odpovídá požadovanému času platnosti a odkaz směřovat na tuto konkrétní verzi.

4. ZÁVĚR

Tento příspěvek uvedl čtenáře do situace v oboru implementací temporálních databází a seznámil ho s vlastním řešením implementace temporální databáze.

Dnes neexistuje běžně použitelná a plně funkční implementace temporální databáze a cílem této práce je tedy vytvoření použitelného produktu kompatibilního se standardní technologií JDBC.

V současné době je implementace překladače temporálního TSQL2 ve fázi teoretického návrhu a je rozpracován parser jazyka TSQL2. Implementace nebude zaměřena pouze na jeden databázový systém, ale bude podporovat většinu dnes běžně užívaných databázových systémů. Vzhledem k použití jazyka Java bude tato implementace zároveň multiplatformní.

LITERATURA

[1] SNODGRASS, Richard T., et al. TSQL2 Language Specification [online]. 1994 [cit. 2008-12-17]. Dostupný z WWW: <ftp://ftp.cs.arizona.edu/tsql/tsql2/spec.pdf>.

[2] SNODGRASS, Richard T.. TSQL2 Temporal Query Language [online]. - [cit. 2008-12-17]. Dostupný z WWW: <http://www.cs.arizona.edu/~rts/tsql2.html>.

[3] STEINER, Andreas. A Generalisation Approach to Temporal Data Models and their Implementations. [s.l.], 1998. 163 s. SWISS FEDERAL INSTITUTE OF TECHNOLOGY ZURICH. Dizertační práce. Dostupný z WWW: <http://www.timeconsult.com/Publications/diss.pdf>.