TEMPORÁLNÍ DATABÁZE A TSQL2

Zdeněk Vilušínský

Temporální databáze

- Příklady: SIS, analýzy reklamy
- Konvenční databáze reprezentují stav
- Změny stav upravují
- Temporální databáze podporují prvek času
- Dotazy přes časové období

Případová studie

- Databáze zaměstnanců

  Zamestnanec(Jmeno, Plat, Titul)

- Jaký je Davidův plat?

  SELECT Plat
  FROM Zamestnanec
  WHERE Jmeno = ‘David’

Případová studie

- Přidáme do záznamu datum narození

  Zamestnanec(Jmeno, Plat, Titul, DatumNarozeni **DATE**)

- Kdy se David narodil?

  SELECT DatumNarozeni
  FROM Zamestnanec
  WHERE Jmeno = 'David'

- Tedy v SQL je (omezená) temporální podpora

Případová studie

- Nyní chceme přidat záznam o vývoji v zaměstnání

  Zamestnanec(Jmeno, Plat, Titul, DatumNarozeni, Start **DATE**, Stop **DATE**)

- Datový typ je stejný jako Datum Narození, ale dopad mnohem větší

Případová studie

- Temporální projekce
- Jaký je Davidův současný plat?

```
SELECT Plat
FROM Zamestnanec
WHERE Jmeno = 'David' AND Start <= CURRENT_DATE AND
      CURRENT_DATE <= Stop
```

- Díky novým sloupcům je dotaz složitější

Případová studie

- Chtěli bychom zaměstnancům poskytnout jejich historii platů
- Maximální dobu kdy měli stejný plat
- V SQL velmi obtížné
- Koalescence

Případová studie

Jméno	Plat	Titul	Datum Narození	Start	Stop
David	60000	Magistr	1945-04-09	1995-01-01	1995-06-01
David	70000	Magistr	1945-04-09	1995-06-01	1995-10-01
David	70000	Doktor	1945-04-09	1995-10-01	1996-02-01
David	70000	Profesor	1945-04-09	1996-02-01	1997-01-01

Jméno	Plat	Start	Stop
David	60000	1995-01-01	1995-06-01
David	70000	1995-06-01	1997-01-01

```
CREATE TABLE Temp(Plat, Start, Stop)
AS   SELECT Plat, Start, Stop
     FROM Zamestnanec
     WHERE Jmeno = 'David'
repeat
     UPDATE Temp T1
     SET (T1.Stop) = (SELECT MAX(T2.Stop)
     FROM Temp AS T2
     WHERE T1.Salary = T2.Salary AND T1.Start < T2.Start
     AND T1.Stop >= T2.Start AND T1.Stop < T2.Stop)
     WHERE EXISTS (SELECT *
     FROM Temp AS T2
     WHERE T1.Salary = T2.Salary AND T1.Start < T2.Start
     AND T1.Stop >= T2.Start AND T1.Stop < T2.Stop)
until no updates
DELETE FROM Temp T1
WHERE EXISTS (SELECT *
     FROM Temp AS T2
     WHERE T1.Salary = T2.Salary
       AND ((T1.Start > T2.Start AND T1.Stop <= T2.Start)
         OR (T1.Start >= T2.Start AND T1.Stop < T2.Stop))
```

Případová studie

- Problém s tímto řešením – repeat-until
- SQL řešení existuje – pomocí zahnízděných **NOT EXISTS**
- SQL nemá prostředky pro práci s „timestampy“
- Řešení v TSQL2 (Davidova historie platů)

```
SELECT Plat
FROM Zamestnanec
WHERE Jmeno = 'David'
```

Případová studie

- Temporální join
- Reorganizujeme schéma, čímž se vyhneme problémům v SQL

  Zamestnanec1(Jmeno, Plat, Start **DATE**, Stop **DATE**)

  Zamestnanec2(Jmeno, Titul, Start **DATE**, Stop **DATE**)
- Jaká je Davidova historie platů?

```
SELECT Plat, Start, Stop
FROM Zamestnanec1
WHERE Jmeno = 'David'
```

Případová studie

- Ale co když poté chceme vztah mezi obdobími platu a titulu?
- SQL dotaz musí zjistit, jak se překrývá řádek z tabulky Zamestnanec1 s řádky ze Zamestnanec2

Případová studie

- Najdi historii platů a titulů pro všechny zaměstnance

```
SELECT Zamestnanec1 .Jmeno, Plat, Titul, Zamestnanec1.Start,
    Zamestnanec1.Stop
FROM Zamestnanec1, Zamestnanec2
WHERE Zamestnanec1.Jmeno = Zamestnanec2.Jmeno
    AND Zamestnanec2.Start <= Zamestnanec1.Start
    AND Zamestnanec1.Stop <= Zamestnanec2.Stop
UNION
SELECT Zamestnanec1 .Jmeno, Plat, Titul, Zamestnanec1.Start,
    Zamestnanec1.Stop
...
```

Případová studie

- 4 případy jak se záznamy překrývají
- Najdi historii platů a titulů pro všechny zaměstnance (TSQL2)

```
SELECT Zamestnanec1 .Jmeno, Plat, Titul,
FROM Zamestnanec1, Zamestnanec2
WHERE Zamestnanec1.Jmeno = Zamestnanec2.Jmeno
```

Shrnutí

- Data závislá na čase jsou bežná
- Netemporální databáze nemají dostatečnou podporu pro práci s nimi
- Temporální databáze umožňuje jednodušší dotazy

Časová doména

- Modelování a reprezentace času
- Čas v temporální logice = libovolná množina okamžiků s částečným uspořádáním
- Další axiomy model upřesňují
  - Lineární – uspořádání celé množiny
  - Větvení – lineární do teď, pak možné budoucnosti
  - Cyklický – rekurentní proces (týden)

Časová doména

- Axiomy mohou charakterizovat *hustotu*
- *Diskrétní* modely – isomorfní přirozeným číslům
  - každý bod v čase má jednoho předchůdce
- *Husté* modely – isomorfní racionálním nebo reálným číslům
  - mezi každými dvěma momenty existuje další
- *Průběžné* modely – isomorfní reálným číslům

Časová doména

- V průběžném modelu každé reálné číslo odpovídá bodě v čase
- V diskrétním modelu každé přirozené číslo odpovídá nedělitelné jednotce času s libovolným trváním - *chronon*.
- Chronon není bod, ale úsečka

Časová doména

- Obvykle se používá diskrétní model
  - nepřesnost měření
  - přirozenost v jazyce
  - přirozená modelace událostí co trvají

Časová doména

- Další axiomy:
- *Omezení*
- Koncept *vzdálenosti*
- *Relativní* a *Absolutní* čas

Datové typy času

- *okamžik* – specifický chronon
- *událost* – něco co se stalo v okamžiku
- *doba události* – okamžik kdy se událost stala ve skutečnosti
- SQL92 – DATE, TIME, TIMESTAMP
- *časová perioda* – čas mezi dvěma okamžiky
  - neplést s typem INTERVAL

Datové typy času

- *časový interval* – známý časový úsek, ale nemá specifické hraniční okamžiky
- *množina okamžiků*
- *temporální elementy* – konečné sjednocení period

Asociování faktů s časem

- *Valid time* – čas po který fakt byl/je/bude pravdivý.
- *Transaction time* – období po které je fakt uložený v databázi
- *snímek* – nepodporuje ani jeden model, okamžik v databázi
- *bitemporální* – podporuje oba modely

Snímek

- Zachycuje, co je aktuálně pravda

Transaction-time relace

- Snímek se neupravuje
- Dojde ke změně aktuálního snímku, který se poté přidá do relace
- Pro dotazy na stav databáze v minulosti

Valid-time relace

- Uchovává platnost dat
- Kterákoliv část se dá upravit
- Nedá se určit předchozí stav databáze

Bitemporální relace

- „append only“ jako transaction-time
- Platnost dat jako valid-time

- S rozvojem databáze transaction-time roste monotóně, ale valid-time může mít velký rozptyl

Shrnutí datového modelu

- Temporální datový model by měl splňovat mnoho požadavků
- jasná sémantika aplikace
- konzistentní, minimální rozšíření existujícího modelu
- souvislé chování faktů v čase
- snadná implementace, vysoký výkon
  - tyto požadavky se zdají protichůdné

Shrnutí datového modelu

- Simultánní zaměření na prezentaci dat, uložení dat a efektivní vyhodnocování dotazů komplikuje zachycení časově proměnných dat
- Mnoho nekompatibilních datových modelů s mnoha dotazovacími jazyky

TSQL2

- Temporal Structured Query Language
- cílem sjednotit přístup k temporálním datovým modelům a dotazovacím jazykům
- rozšíření k SQL92
- částečně obsažené v SQL3

Časová ontologie

- Lineární časová struktura, omezená z obou stran (+- 18 miliard let)
- diskrétní reprezentace reálného času, která může být považována za diskrétní, hustou nebo průběžnou
- granule – seskupení po sobě jdoucích chrononů – různá granularita
- nevyžaduje výběr mezi diskrétní, hustou nebo průběžnou ontologií

Časová ontologie

- Dokonce nedovoluje otázky, které by nutili rozhodnout mezi modely
- Nelze se ptát, zda okamžik A předchází okamžik B – pouze v rámci zvolené granularity (vteřiny, dny,...)
- Přidává datový typ **PERIOD**

Datový model

- jednoduchý
- zachovává obecnost a jednoduchost relačního modelu
- separátní modely pro prezentaci dat, ukládání dat a vyhodnocování dotazů
- Více koordinovaných datových modelů zvládne co by jeden nedokázal

Stavba jazyka

- striktní nadmnožina SQL92
- pro příklad temporálních relací budeme používat databázi pacientů

```
CREATE TABLE Predpis(Jmeno CHAR(30), Lekar CHAR(30), Lek CHAR(30), Davka
    CHAR(30), Frekvence INTERVAL MINUTE)
AS VALID STATE DAY AND TRANSACTION
```

- frekvence je počet minut mezi dávkami
- valid time – na kdy je lék předepsán
- transaction time – příchod záznamu do databáze

Druhy relací

- snímková – žádná temporální podpora
- valid-time state **AS VALID STATE**
- valid-time event **AS VALID EVENT**
- transaction-time **AS TRANSACTION**
- bitemporal state **AS VALID STATE AND TRANSACTION**
- bitemporal event **AS VALID EVENT AND TRANSACTION**
- typ relace se může změnit **ALTER TABLE**

SELECT

- Novým klíčovým slovem **SNAPSHOT** získáme snímek z temporální relace
- Kdo někdy měl předepsané léky?

```
SELECT SNAPSHOT Jmeno
FROM Predpis
```

- Kdo někdy měl předepsaný aspirin?

```
SELECT SNAPSHOT Jmeno
FROM Predpis
WHERE Lek = 'Aspirin'
```

SELECT

- Kdo měl předepsané léky a kdy?

```
SELECT Jmeno
FROM Predpis
```

- Defaultní chování vrací historii
- TSQL2 automaticky provádí koalescenci
- Výsledkem je množina řádků, každý s periodou, kdy pacient bral jeden či více léků

Přeorganizování (Restructuring)

- Jeden z nejsilnějších prostředků
- Koalescence se automaticky provádí na výsledek dotazu – toto umožňuje provést ji na řádky v klauzuli **FROM**

- Kdo bral lék celkem déle než 6 měsíců?

Přeorganizování

```
SELECT Jmeno, Lek
FROM Predpis(Jmeno, Lek) AS P
WHERE CAST(VALID(P) AS INTERVAL MONTH)
                > INTERVAL '6' MONTH
```

- Přeorganizování na Jmene a Leku, výsledkem je maximální doba kdy byl lék předepsán
- **VALID**(P) vrací valid-time prvky z P
- operátor **CAST** konvertuje co vyjde z valid

Přeorganizování

- Kdo užíval Aspirin?

  SELECT SNAPSHOT P1.Jmeno
  FROM Predpis(Jmeno) **AS** P1, P1(Lek) **AS** P2
  WHERE P2.Lek = 'Aspirin' **AND VALID**(P2) = **VALID**(P1)

- Spárování
- Jak přeorganizování, tak spárování je „syntaktické cukrátko," dá se přepsat pomocí vnořených selectů

Štěpení (Partitioning)

- Často chceme zkoumat maximální periody timestamp
- klíčové slovo **PERIOD**
- Kdo bral stejný lék déle než 6 měsíců v kuse?

```
SELECT SNAPSHOT Jmeno, Lek, VALID(P)
FROM Predpis(Jmeno, Lek)(PERIOD) AS P
WHERE CAST(VALID(P) AS INTERVAL MONTH)
        > INTERVAL '6' MONTH
```

Štěpení

- Alternativa

```
SELECT Jmeno, Lek
FROM Predpis(Jmeno, Lek)(PERIOD) AS P
WHERE CAST(VALID(P) AS INTERVAL MONTH)
        > INTERVAL '6' MONTH
```

- Pro každý pár lék-jméno pouze jeden výsledek s maximální délkou užívání.
- štěpení není „syntaktické cukrátko“

VALID

- Jaké léky měla Michaela předepsány v roce 1996?

```
SELECT Lek
VALID INTERSECT(VALID(Predpis), PERIOD '[1996]' DAY)
FROM Predpis
WHERE Name = 'Michaela'
```

- Výsledkem je seznam léků společně s časem, kdy byl předepsán.

Aktualizace dat

```
INSER INTO Predpis
VALUES('Michaela', 'Dr. Sova', 'Aspirin', '100mg',
        INTERVAL '8:00' MINUTE)
```

- Nespecifikovali jsme timestamp, default:

```
VALID PERIOD(CURRENT_TIMESTAMP,
        NOBIND(CURRENT_TIMESTAMP))
```

- Otevřený konec (konec je aktualní čas)

Aktualizace dat

INSER INTO Predpis
VALUES('Michaela', 'Dr. Sova', 'Aspirin', '100mg',
INTERVAL '8:00' **MINUTE**)
VALID PERIOD '[1996-01-01 – 1996-06-30]'

- automatická koalescence
- transaction time je roven **CURRENT_TIMESTAMP**
- **VALID** je takto použitelné i v **DELETE** a **UPDATE**

Aktualizace dat

- **DELETE** může změnit více záznamů kvůli překrývání timestampů – režii řeší TSQL2

```
UPDATE Predpis
SET Davka TO '50mg'
WHERE Name = 'Melanie' AND Lek = 'Aspirin'
```

- Dojde ke změně všech současných a budoucích! dávek

Události (Event Relations)

- Doteď jsme se zabývali jen stavem, který je po nějaký čas pravdivý
- Eventy zaznamenávají okamžité události

  CREATE TABLE LabTest (Jmeno **CHAR**(30), Lekar **CHAR**(30), TestID **INTEGER**)
  AS VALID EVENT HOUR AND TRANSACTION

- Události se také dají Štěpit a Přeorganizovávat

Události (Event Relations)

- Byl nějaký pacient jediný, kdo šel na testy od konkrétního lékaře?

```
SELECT L1.Jmeno, L2.Lekar
FROM LabTest(Jmeno) AS L1, L1(Lekar) AS L2,
     LabTest(Lekar) AS L3
WHERE VALID(L1) = VALID(L2) AND L2.Lekar = L3.Lekar
     AND VALID(L1) = VALID(L3)
```

Podpora Transaction time

- Doteď jsme neřešili, že tabulka Predpis podporuje transaction time
- Jaké předpisy Michaela měla?

```
SELECT Lek
FROM Predpis
WHERE Jmeno = 'Michaela'
```

- Vrací historii jak je nejlépe známá, včetně oprav

Podpora Transaction time

- Můžeme udělat v databázi rollback
- Kdyby bylo 1.6.1996, jaké předpisy by Michaela měla?

```
SELECT Lek
FROM Predpis AS P
WHERE Jmeno = 'Michaela'
    AND TRANSACTION(P) OVERLAPS DATE '1996-06-01'
```

- Default je **TRANSACTION**(P) **OVERLAPS CURRENT_TIMESTAMP**

Podpora Transaction time

- Kdy byla Michaelina data, validní k 1.6.1996 naposledy opravována?

```
SELECT SNAPSHOT BEGIN(TRANSACTION(P2))
FROM Predpis AS P1P2
WHERE P1.Jmeno = 'Michaela' AND P2.Jmeno = 'Michaela'
   AND VALID(P1) OVERLAPS DATE '1996-06-01'
   AND VALID(P2) OVERLAPS DATE '1996-06-01'
   AND TRANSACTION(P1) MEETS TRANSACTION(P2)
```

Agregační funkce

```
SELECT COUNT *
FROM Predpis
WHERE Jmeno = 'Michaela'
```

- vrací valid-time state relaci
- jak se měnil počet předpisů v libovolném bodě v čase

Agregační funkce

- TSQL2 přidává funkci **RISING**
- nejdelší období, kdy atribut monotónně rostl

```
SELECT SNAPSHOT RISING (Davka)
FROM Predpis
WHERE Jmeno = 'Michaela' AND Lek = 'Aspirin'
```

- dotaz vrátí množinu období, kdy atribut rostl

Vývoj a verzování schématu

- SQL dovoluje schéma měnit pomocí **ALTER** - vývoj.
- Pokud má relace podporu transaction-time tak se schéma pro tuto relaci verzuje
- V praxi se celé schéma stane množinou relací transacion-time
- Když chci jinou verzi: **SET SCHEMA DATE** ‘1996-08-19’

Některé další konstrukce

- *surrogate* – unikátní hodnota, vhodná k porovnání na shodu; TSQL2 přidává sloupec **SURROGATE** a unární funkci **NEW**
- *vacuuming* – odstranění zastaralých dat
  - s podporou transaction-time data nemizí z databáze, ale přidá se timestamp o smazání

Shrnutí (TSQL2)

- přidává práci s prvky které se mění časem
- lze používat i konvenční relace
- periody jsou nový typ s daným trváním v čase

Zdroje

- Carlo Zaniolo: Advanced database systems
  - kapitoly 5 a 6
- www.wikipedia.org