DATABÁZOVÉ SYSTÉMY 2

Temporální databáze

Temporální databáze

Motivace

- Potřebujeme v databázích čas?

- Studijní informační systém
- Skladová evidence
- Účetní a bankovní systémy
- Docházkové systémy
- ... a mnoho dalších

Klasické vs. temporální databáze

- Klasický databázový systém
  - Zachycen stav systému v aktuálním časovém okamžiku
  - Problém: co dělat se starými daty
- Temporální databázový systém
  - Databáze určitým způsobem podporující čas
  - Jednodušší dotazy
  - Jednodušší udržování aplikací

Studie konkrétního případu

- Příklad v SQL:

  Zaměstnanec(
  Jméno, Plat, Funkce)

  - Jaký plat má Pepa?

```
SELECT Plat
  FROM Zaměstnanec
  WHERE Jméno = 'Pepa'
```

Studie konkrétního případu

- Příklad v SQL:

  Zaměstnanec(Jméno, Plat, Funkce,
  Datum_narození DATE)

  - Kdy se Pepa narodil?

```
SELECT Datum_narození
  FROM Zaměstnanec
  WHERE Jméno = 'Pepa'
```

Konkrétní případ

- Příklad v SQL:

```
Zaměstnanec(Jméno, Plat, Funkce,
        Datum_narození,
        Platí_od DATE, Platí_do DATE)
```

  - Jaký je Pepův aktuální plat?

```
SELECT Plat
  FROM Zaměstnanec
  WHERE Jméno = 'Pepa'
      AND Platí_od <= CURRENT_DATE      AND
  CURRENT_DATE <= Platí_do
```

Temporální projekce

- Jako projekce v klasických databázích
- Navíc bere v úvahu čas
  - Srůstání
- Nedostatečná podpora v klasických databázových systémech

Temporální projekce – příklad

Jméno	Plat	Funkce	Datum narození	Platí_od	Platí_do
Pepa	60000	Vrátný	1945-04-09	1995-01-01	1995-06-01
Pepa	70000	Vrátný	1945-04-09	1995-06-01	1995-10-01
Pepa	70000	Vrchní vrátný	1945-04-09	1995-10-01	1995-02-01
Pepa	70000	Ředitel bezpečnosti	1945-04-09	1996-02-01	1997-01-01

- Příklad:
  - Jaká Pepova platová historie?

Jméno	Plat	Platí_od	Platí_do
Pepa	60000	1995-01-01	1995-06-01
Pepa	70000	1995-06-01	1997-01-01

Temporální projekce – příklad

```
CREATE TABLE Temp(Plat, Od, Do)
    AS SELECT Plat, Platí_od AS Od, Platí_do AS Do
    FROM Zaměstnanci
    WHERE Jméno = 'Pepa'
SELECT DISTINCT F.Plat, F.Od, L.Do
FROM Temp AS F, Temp AS L
WHERE F.Od < L.Do
    AND F.Plat = L.Plat
    AND NOT EXISTS (SELECT *
        FROM Temp AS M
        WHERE M.Plat = F.Plat
            AND F.Od < M.Od AND M.Od < L.Do
            AND NOT EXISTS (SELECT * FROM Temp AS T1
                WHERE T1.Plat = F.Plat
                    AND T1.Od < M.Od AND M.Od <= T1.Do
    AND NOT EXISTS (SELECT *
        FROM Temp AS T2
        WHERE T2.Plat = F.Plat
            AND ((T2.Od < F.Od AND F.Od <= T2.Do) OR
                (T2.Od < L.Do AND L.Do < T2.Do)))
```

Temporální spojení

- Stejné jako spojení (join) v klasických databázích
- Navíc bere v úvahu čas události
- Příklad v SQL:

```
Zaměstnanec1(Jméno, Plat, Platí_od, Platí_do)
  Zaměstnanec2(Jméno, Funkce, Platí_od, Platí_do)
```

  - Pro každého zaměstnance najděte historii platů a funkcí
  - Myšlenka: Pro každou dvojici (Plat, Funkce) najít průnik intervalů (Platí_od, Platí_do) z obou relací.
  - Realizace: Rozbor případů

Konkrétní případ

- Historie platů a funkcí v TSQL

```
SELECT    Zaměstnanec1.Plat,
          Zaměstnanec2.Funkce
FROM Zaměstnanec1, Zaměstnanec2
WHERE     Zaměstnanec1.Jméno =
          Zaměstnanec2.Jméno
```

Modely času

- Temporální logika: čas je libovolná množina okamžiků s daným uspořádáním
- Modely času podle uspořádání
  - Lineární
  - Větvený (čas možných budoucností)
  - Cyklický
- Modely času podle hustoty
  - Diskrétní
  - Hustý
  - Spojitý
- Omezenost času
- Absolutní / relativní čas

Datové typy pro čas

- Časový okamžik (instant) (SQL-92)
  - DATE
  - TIME
  - TIMESTAMP
- Časový úsek (time period)
  - Doba mezi dvěma časovými okamžiky
  - 15:30 – 15:50
- Časový interval (interval)
  - Doba o specifikované délce, ale bez konkrétních krajních bodů
  - 30 minut
- Množina časových okamžiků (instant set)
- Množina časových úseků (temporal elements)

Vztah událostí a času

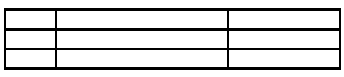
- Čas platnosti (valid time)
  - Čas, kdy byla událost pravdivá v reálném světě
  - Může být v minulosti, přítomnosti i budoucnosti
- Transakční čas (transaction time)
  - Čas, kdy byl fakt reprezentován v databázi
  - Nabývá **pouze** aktuální hodnoty
  - Monotónně roste
- Čas platnosti a transakční čas jsou ortogonální

Vztah událostí a času

- snapshot
  - Datový model nepodporující čas platnosti ani transakční čas
- valid-time
  - Datový model podporující pouze čas platnosti
- transaction-time
  - Datový model podporující pouze transakční čas
- bitemporální
  - Datový model podporující čas platnosti i transakční čas
- temporální
  - Datový model podporující čas platnosti nebo transakční čas
- Obvykle založeno na relačním datovém modelu nebo objektově orientovaném datovém modelu.

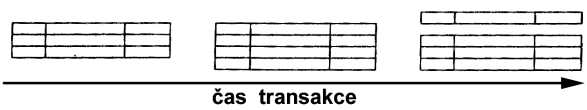
snapshot relace

- Datový model nepodporující čas platnosti ani transakční čas
- Klasický relační model
- Každá n-tice je fakt platný v reálném světě
- Při změně reálného světa jsou do relace prvky přidávány nebo z ní odebírány



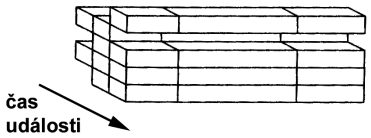
transaction-time relace

- Datový model podporující pouze transakční čas
- Posloupnost snapshot-ů indexované transakčním časem
- Umožňuje získat informaci ze stavu databáze v nějakém okamžiku minulosti
- Je možné uvažovat i větvení



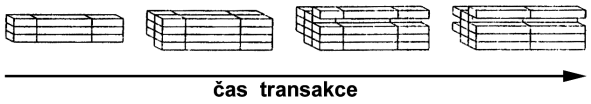
valid-time relace

- Datový model podporující pouze čas platnosti
- Cokoliv v relaci může být upraveno
  - Hodnoty n-tic
  - Čas události (začátek i konec)
- Umožňuje klást dotazy o faktech platných v minulosti i budoucnosti



bitemporální relace

- Datový model podporující čas platnosti a transakční čas
- append-only



Reprezentace „času platnosti"

- Reprezentace času platnosti
  - časový okamžik
  - doba
  - časový úsek
  - množina okamžiků
- Čas platnosti může být přidružen k
  - atributu
  - množině atributů
  - celé n-tici nebo objektu
- Rozšíření operací relační algebry

Reprezentace „transakčního času"

- Reprezentace transakčního času
  - časový okamžik
    - nová n-tice se stejným klíčem => logické odstranění původní
  - časový úsek
    - (teď, *dokud nezměněno*)
  - tři časové okamžiky
    - čas zaznamenání začátku události v reálném světě
    - čas zaznamenání konce události v reálném světě
    - čas logického odstranění události z databáze
  - Množina časových úseků
- Někdy podporován i versioning

Příklady datových modelů (1)

- Segevův datový model
  - Podporuje pouze čas platnosti
  - Razítko označuje, kdy fakt začal platit

Jméno	*Oddělení*	*Čas*
Erik	Obuv	1
Erik	Knihy	6
Erik	Obuv	11
Erik	Null	13

Příklady datových modelů (2)

- Sardův datový model
  - Podporuje jen čas platnosti
  - Razítko má podobu intervalu

Jméno	*Oddělení*	*Čas*
Erik	Obuv	[1-5]
Erik	Knihy	[6-10]
Erik	Obuv	[11-12]

Příklady datových modelů (3)

- Datový model HRDM
  - Podporuje jen čas platnosti
  - Razítkují se hodnoty atributů

Jméno	*Oddělení*
1 → Erik … 12 → Erik	1 → Obuv … 5 → Obuv 6 → Knihy … 10 → Knihy 11 → Obuv 12 → Obuv

Příklady datových modelů (4)

- Extensionální datový model

Jméno	*Odd*	*VT*	*TT*
Erik	Obuv	1	1
Erik	Obuv	2	1
..	..	..	..
Erik	Obuv	1	2
Erik	Obuv	2	2
..	..	..	..
Erik	Obuv	1	8
..	..	..	..
Erik	Obuv	5	8
Erik	Knihy	6	8
..	..	..	..
Erik	Obuv	1	11
..	..	..	..
Erik	Obuv	5	11

Jméno	*Odd*	*VT*	*TT*
Erik	Knihy	6	11
..	..	..	..
Erik	Knihy	10	11
Erik	Obuv	11	11
..	..	..	..
Erik	Obuv	1	13
..	..	..	..
Erik	Obuv	5	13
Erik	Knihy	6	13
..	..	..	..
Erik	Knihy	10	13
Erik	Obuv	11	13
Erik	Obuv	12	13

Příklady datových modelů (5)

- Bhargavův datový model

Jméno	Oddělení
[1, 12] x [1, ∝] Erik [13, →] x [1, 12] Erik	[1, 7] x [1, ∝] Obuv [8, 10] x [1, 5] Obuv [8, 10] x [6, ∝] Knihy [11, 12] x [1, 5] Obuv [11, 12] x [6, 10] Knihy [11, 12] x [11, ∝] Obuv [13, →] x [1, 5] Obuv [13, →] x [6, 10] Knihy [13, →] x [11, 12] Obuv

Temporální dotazovací jazyky

- Temporální datový model
  - Objekty se přesně danou strukturou
  - Omezení pro dané objekty
  - Operace na daných objektech
    - Temporální dotazovací jazyky

Temporální dotazovací jazyky

- Velké množství
- Nejčastěji založeny na SQL
- Typy
  - Relační
    - př.: HQL, HSQL, TDM, TQuel, TSQL, TSQL2
  - Objektově orientované
    - př.: MATISSE, OSQL, OQL, TMQL

Shrnutí

- Ideální temporální datový model
  - Minimální rozšíření existujícího DM
  - Souvislá prezentace chování měnícího se v čase
  - Snadná implementace
  - Vysoký výkon
- Dosažení ideálu je prakticky nemožné

Shrnutí

- Hlavní cíl temporálního DM by měl být zachytit sémantiku dat měnící se v čase
  - Ale často se dostává do pozadí
- Mnoho nekompatibilních datových modelů s mnoha dotazovacími jazyky

TSQL2

Obsah

- Čas v TSQL2
- Datový model
- Vytvoření relace
- Příkaz SELECT
- Modifikační příkazy
- Shrnutí

TSQL2

- Temporal Structrured Query Language 2
- Měl sjednotit přístupy k temporálním datovým modelům
- Nadmnožina SQL-92

Pojetí času

- Časová osa TSQL2 je na obou koncích omezena, ale dostatečně daleko (18 miliard let)
- U časových údajů jsou možné různé granularity
- Časové typy
  - DATE, TIME, TIMESTAMP, INTERVAL, PERIOD

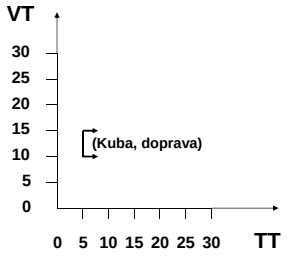
Datový model

- Je použit BCDM (Bitemporal Conceptual Data Model)
- Řádek relace je orazítkován množinou bitemporálních chrononů
- Bitemporální chronon je dvojice (chronon transakčního času, chronon času platnosti)

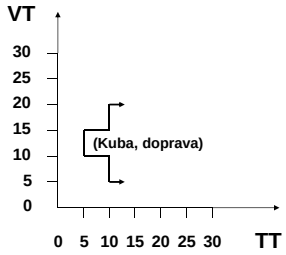
Příklad bitemporální relace (1)

- Relace Zaměstananec – umístění lidí v odděleních určitého podniku
- Schéma (Jméno, Oddělení) + časové razítko
- Předpokládejme granularitu 1 den u času platnosti i u transakčního času

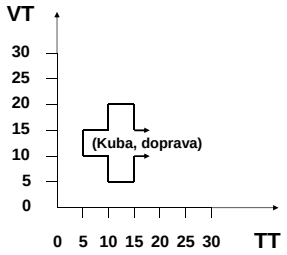
Příklad bitemporální relace (2)



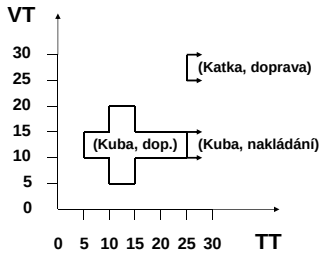
Příklad bitemporální relace (3)



Příklad bitemporální relace (4)



Příklad bitemporální relace (5)



Příklad bitemporální relace (6)

Jméno	*Oddělení*	*Časové razítko*
Kuba	Doprava	{(5, 10),…,(5,15),…(9, 10),…,(9, 15), (10,5),…,(10,20),…,(14,5),…,(14,20), (15,10),…,(15,15),…,(24,10),…,(24,15)}
Kuba	Nakládání	{(U.C., 10), …, (U.C., 15)}
Katka	Doprava	{(U.C., 25), …, (U.C., 30)}

Definice schématu

- Příklad

```
CREATE TABLE Předpis (Jméno CHAR(30),
 Lékař CHAR (30), Lék CHAR (30), Dávka CHAR (30),
 Frekvence INTERVAL MINUTE)
 AS VALID STATE DAY AND TRANSACTION
```

  - Čas platnosti má granularitu 1 den
  - Transakční čas má granularitu 1 ms nebo menší

Příkaz SELECT

- Komu byl předepsán nějaký lék?

```
SELECT SNAPSHOT Jméno
FROM Předpis
```

- Kdo bral (bere) lék Glyvenol?

```
SELECT SNAPSHOT Jméno
FROM Předpis
WHERE Lék = 'Glyvenol'
```

Příkaz SELECT

- Komu byl předepsán nějaký lék a kdy?

```
SELECT Jméno
FROM Předpis
```

Příkaz SELECT

- Které léky byly užívány současně s lékem Glyvenol?

```
SELECT P1.Jméno, P2.Lék
FROM Předpis AS P1, Předpis AS P2
WHERE P1.Lék = 'Glyvenol' AND
   P2.Lék <> 'Glyvenol' AND
   P1.Jméno = P2.Jméno
```

Restrukturalizace

- Na úrovni klauzule FROM se provede projekce a slévání
- Kdo měl (má) brát nějaký lék déle než 6 měsíců (v souhrnu)?

```
SELECT Jméno, Lék
FROM Předpis(Jméno, Lék) AS P
WHERE CAST(VALID(P) AS INTERVAL
  MONTH) > INTERVAL '6' MONTH
```

Restrukturalizace

- Kdo po celou dobu léčby bere Glyvenol?

```
SELECT SNAPSHOT P1.Jméno
FROM Předpis(Jméno) AS P1, P1(Lék) AS
  P2
WHERE P2.Lék = 'Glyvenol' AND
   VALID(P2) = VALID(P1)
```

Partitioning

- Kdo měl předepsaný jeden lék déle než 6 měsíců nepřerušovaně?

```
SELECT SNAPSHOT Jméno, Lék, VALID(P)
  FROM Předpis(Jméno, Lék)(PERIOD) AS P
  WHERE CAST(VALID(P) AS INTERVAL
        MONTH) > INTERVAL '6' MONTH;
```

Jiný způsob

```
SELECT Jméno, Lék
  FROM Předpis(Jméno, Lék)(PERIOD) AS
  P
  WHERE CAST(VALID(P) AS INTERVAL
        MONTH) > INTERVAL '6'
  MONTH;
```

- VALID(P) lze užít v SELECT pouze při použití PERIOD

Klauzule VALID

- Jaké léky měla Eva předepsané v roce 2006?

```
SELECT Lék
  VALID INTERSECT (VALID(Předpis),
                   PERIOD '[2006]'
  DAY)
  FROM Předpis
  WHERE Jméno='Eva'
```

Modifikace údajů v tabulce

- **INSERT**
- **DELETE**
- **UPDATE**

INSERT

- Vlož nový předpis.

```
INSERT INTO Předpis
  VALUES ( 'Eva', 'Dr. Novák', 'Glyvenol',
  '100mg', INTERVAL '8:00' MINUTE)
```

- Defaultní hodnota

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

INSERT

- Vlož nový předpis s omezenou dobou platnosti

```
INSERT INTO Předpis
  VALUES ( 'Eva', 'Dr. Novák', 'Glyvenol',
  '100mg', INTERVAL '8:00' MINUTE)
  VALID PERIOD '[2006-01-01 – 2006-06-
  30]'
```

DELETE

- Eva neměla v červnu 2006 předepsaný žádný lék.

```
DELETE FROM Předpis
    WHERE Jméno = 'Eva'
    VALID PERIOD '[2006-06-01 – 2006-06-30]'
```

UPDATE

- Změň dávkování léku Glyvenol na 50 mg.

```
UPDATE Předpis
    SET Dávkování TO '50mg'
    WHERE Jméno = 'Eva' AND
          Lék = 'Glyvenol'
```

UPDATE

- Změň dávkování léku od března do května.

```
UPDATE Předpis
    SET Dávkování TO '50mg'
    VALID PERIOD '[2006-03-01 – 2006-05-31]'
    WHERE Jméno = 'Eva' AND
          Lék = 'Glyvenol'
```

Zaznamenání událostí

- Definice tabulky laboratorních testů.

```
CREATE TABLE LabTest ( Jméno
  CHAR(30),  Lékař CHAR(30), ČísloTestu
  INTEGER)
  AS VALID EVENT HOUR AND
   TRANSACTION
  ( x AS VALID STATE DAY AND TRANSACTION )
```

Restrukturalizace

- Který lékař objednával testy jedinému pacientovi a současně tento lékař byl jediný, kdo danému pacientovi testy objednával?
  - **SELECT** L1.Jméno, L2.Lékař
    FROM LabTest(Jméno) **AS** L1,
     L1(Lékař) **AS** L2, LabTest(Lékař) **AS** L3
    WHERE VALID(L1) = **VALID**(L2) **AND**
          L2.Lékař = L3.Lékař **AND**
          VALID(L1) = **VALID**(L3)

Transakční čas

- Jaké léky měla Eva předepsány?

```
SELECT Lék
    FROM Předpis
    WHERE Jméno = 'Eva'
```

Transakční čas

- Jaké záznamy o lécích, které měla předepsány Eva, byly uvedeny v databázi 1. června 2006?

```
SELECT Lék
    FROM Předpis
    WHERE Jméno = 'Eva' AND
    TRANSACTION(P) OVERLAPS
        DATE '2006-06-01'
```

Transakční čas

Kdy byly naposledy změněny záznamy, které se vztahují ke dni 1. 6. 2006?

```
SELECT SNAPSHOT BEGIN(TRANSACTION(P2))
    FROM Předpis AS P1, P2
    WHERE P1.Jméno = 'Eva' AND
          P2.Jméno = 'Eva' AND
    VALID(P1)  OVERLAPS DATE '2006-06-01'
AND
    VALID(P2) OVERLAPS DATE '2006-06-01'
AND
    TRANSACTION(P1) MEETS
TRANSACTION(P2)
```

Agregační funkce

- Kolik léků Eva brala?

```
SELECT COUNT(*)
    FROM Předpis
    WHERE Jméno = 'Eva'
```

- Kolik lidí má předepsané jednotlivé léky?

```
SELECT Lék, COUNT(*)
    FROM Předpis
    GROUP BY Lék
```

RISING

- Jaké bylo nejdelší období, kdy dávka Glyvenolu pro Evu rostla?

```
SELECT SNAPSHOT RISING(Dávka)
    FROM Předpis
    WHERE Jméno = 'Eva' AND
          Lék = 'Glyvenol'
```

Změna schématu tabulky

- 20. 1. 2007 přidáme k tabulce nový sloupec IdČíslo.
  - **ALTER TABLE** Předpis
    ADD COLUMN IdČíslo **INTEGER**
- Zobrazení dat ve starém schématu
  - **SET SCHEMA DATE** '2007-01-19'

Převody času

- **CAST** vždy vrací přesný výsledek
- **SCALE** může vrátit „nepřesný" výsledek

CAST(**TIMESTAMP** '04-19-2006 15:24:00' **AS DAY**)
SCALE(**TIMESTAMP** '04-19-2006 15:24:00' **AS DAY**)
U obou výsledek '04-19-2006'
CAST(**DAY** '04-19-2006' **AS SECOND**)
Výsledkem je 1. sekunda tohoto dne.
SCALE(**DAY** '04-19-2006' **AS SECOND**)
Výsledkem je nějaká sekunda tohoto dne.

Shrnutí

- Výsledek bez podpory času **SELECT SNAPSHOT**
- **Restrukturalizace** – provede projekci na vybrané sloupce a provede slití času platnosti ekvivalentních řádků
- **Partitioning** – vybere nejdelší interval(y) času platnosti pro každou řádku

Další zdroje informací

- ODBC
  - http://www.microsoft.com/data/odbc/
  - http://www.unixodbc.org/
- JDBC
  - java.sun.com/products/jdbc

Distribuované databáze

Distribuované databáze

= databábázové systémy + počítačové sítě

DBS = integrace
PS = decentralizace

DDBS= integrace dat bez centralizace (je variantou DVS)

Distribuované databáze

Distribuovaný výpočetní systém má vlastnosti:

- Je rozložen do uzlů sítě spojených komunikačními kanály
- V uzlech je možné autonomně ukládat a zpracovávat data
- Prostředky v uzlech mohou být nehomogenní
- Uživatel nemusí vědět o existenci ostatních uzlů (tzv.transparentnost)
-

Distribuované databáze- důvody

Proč DDBS?

- lokální autonomie (odpovídají struktuře decentralizovaných organizací. Data uložena v místě nejčastějšího využití a zpracování - zlevnění provozu). V centralizované DB je nutné připojovat se ke vzdálené databázi = přídavná režie, cena komunikace, zatížení sítě
- zvýšení výkonu (inherentní paralelismus rozdělením zátěže na více počítačů)

Distribuované databáze

Proč DDBS?

- spolehlivost (replikace dat, degradace služeb při výpadku uzlu, přesunutí výpočtů na jiný uzel)
- lepší rozšiřitelnost konfigurace (přidání procesorů, uzlů)
- větší schopnost sdílet informace integrací podnikových zdrojů
- uzly mohou zachovat autonomní zpracování a současně virtuálně zabezpečovat globální zpracování
- agregace informací (z více bází dat lze získat informace nového typu)

Problémy

složitost (distribuce databáze, distrib. zpracování dotazu a jeho optimalizace, složité globální transakční zpracování, distribuce katalogu, paralelismus a uvíznutí, případná integrace heterogenních dat do odpovídajících schémat, složité zotavování z chyb)

- cena (komunikace je navíc)
- bezpečnost

Problémy

obtížný přechod (neexistence automatického konverzního prostředku z centralizovaných DB na DDB)