evropský sociální fond v ČR
EVROPSKÁ UNIE
MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY
OP Vzdělávání pro konkurenceschopnost
INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

METODOLOGIE NÁVRHU DISTRIBUOVANÉ DATABÁZE

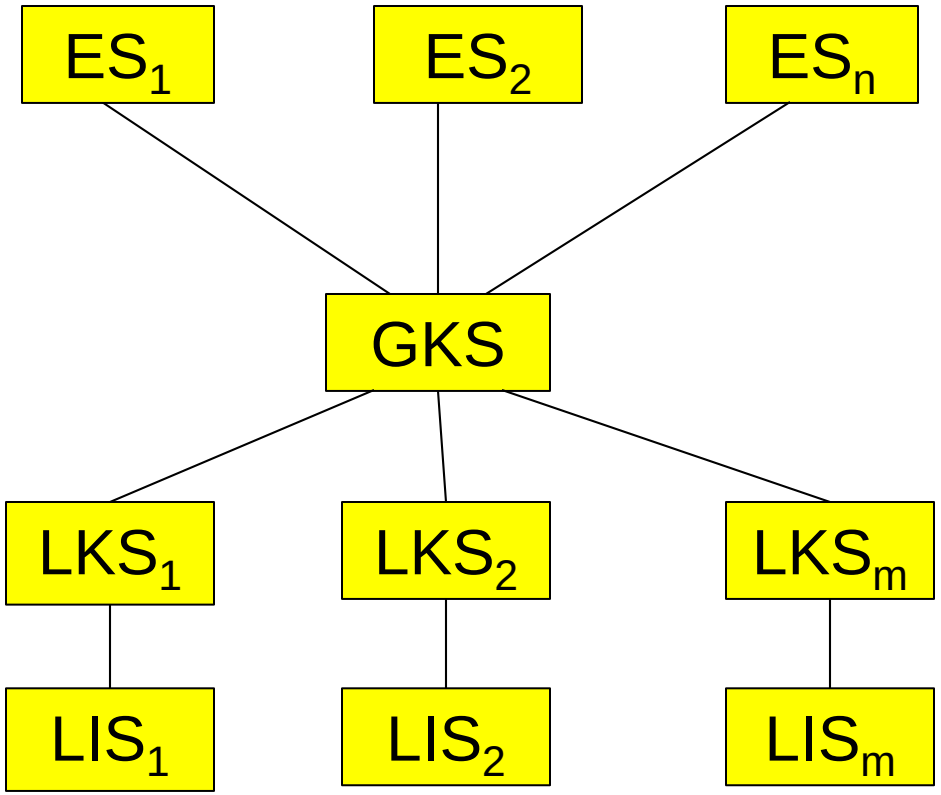
2011/12

J. Mikulecká

Metodologie návrhu schémat v DDBS

- V principu lze postupovat
  - shora–dolů (návrh začíná „na zelené louce“ – databáze se teprve plánují),
  - zdola-nahoru (návrh směřuje k integraci dat uložených v již existujících autonomních databázích).
- Návrh shora-dolů vychází z metodologií používaných v centralizovaném případě - vychází se z datových i funkčních požadavků. Zaměříme-li se na data, jde o to jak zkonstruovat globální konceptuální schéma, odpovídající externím schématům jednotlivých uživatelů a navrhnout, jak rozmístit data na jednotlivé uzly.

Architektura distribuované databáze



ES – externí schéma

GKS – globální konceptuální schéma

LKS – lokální konceptuální schéma

LIS – lokální interní schéma

Popis jednotlivých hladin

- Fyzická organizace dat v každém místě může být rozdílná – na každém místě potřebujeme popis interní schémy dat.
- Celkový pohled na data v celé distribuované databázi popisuje globální konceptuální schéma.
- Organizaci dat na logické úrovni v každém místě popisují lokální konceptuální schémata.
- Pohledy uživatelů na data v celé distribuované databázi popisují externí schémata.

Základní problémy návrhu DDBS

- Fragmentace
  - relaci lze rozdělit na několik sub-relací, které potom distribuujeme
- Alokace
  - každý fragment je uložený v místě s optimální distribucí
- Replikace
  - kopii fragmentu lze udržovat na více místech

Základní otázky

- Proč fragmentovat?
- Jak fragmentovat?
- Jak moc fragmentovat?
- Lze testovat korektnost dekompozice?
- Jak alokovat?
- Jaké informace jsou klíčové při rozhodování o fragmentaci a alokaci?

Fragmentace

- Pro správnou fragmentaci dat je vhodné provést kvantitativní i kvalitativní analýzu nejdůležitějších aplikací.
- Kvantitativní informace mohou zahrnovat
  - frekvenci se kterou je aplikace spouštěna,
  - místo, ve kterém aplikace běží,
  - výkon transakcí a aplikací.
- Kvalitativní informace mohou zahrnovat
  - přístupy k relacím, atributům a řádkům
  - typ přístupu (čtení/zápis).

Proč fragmentovat?

- Aplikace pracují častěji s pohledy než s kompletními relacemi.
- Data jsou uložena v místě, kde se nejčastěji používají.
- Data nepotřebná pro lokální aplikace se v daném místě neukládají.

Proč fragmentovat?

- Paralelizmus - tvoří-li fragmenty jednotku distribuce, tak transakce se mohou rozdělit na několik poddotazů, které mohou pracovat nad fragmenty.
- Bezpečnost - data nepotřebná pro lokální aplikace nejsou v daném místě uložena.

Je-li relace malá a nepříliš často upravovaná, tak může být vhodnější nefragmentovat.

Korektnost fragmentace

Fragmentace by měla být

- úplná,
- rekonstruovatelná,
- disjunktní.

Úplnost

Je-li relace R dekomponovaná na fragmenty R_1, R_2, ... R_n, tak každý element dat který se nachází v R se musí nacházet alespoň v jednom fragmentu.

Rekonstruovatelnost

- Musí být možné definovat relační operaci, pomocí které lze z fragmentů rekonstruovat původní relaci R.
- Horizontální fragmentace - Union
- Vertikální fragmentace - Join.

Disjunktnost

- Je-li prvek dat d_j ve fragmentu R_i, potom by se neměl objevit v žádném jiném fragmentu.
- Výjimka: vertikální fragmentace, kde je nutné opakovat primární klíč z důvodu rekonstruovatelnosti.

Horizontální fragmentace

- Sestává z podmnožiny řádků relace: $\sigma_p(R)$

Příklad:

$N_1 = \sigma_{type='RodDům'}(Nemovitost)$

$N_2 = \sigma_{type='Byt'}(Nemovitost)$

$N_3 = \sigma_{type='Chata'}(Nemovitost)$

Vertikální fragmentace

Sestává z podmnožiny atributů relace. Je definovaná pomocí operátoru projekce: $\Pi_{a1, \ldots, an(R)}$

- Příklad:

$$Z1 = \Pi_{cZam,\ pracZar,\ datumNarozeni,\ plat}\ (Zamestnanec)$$

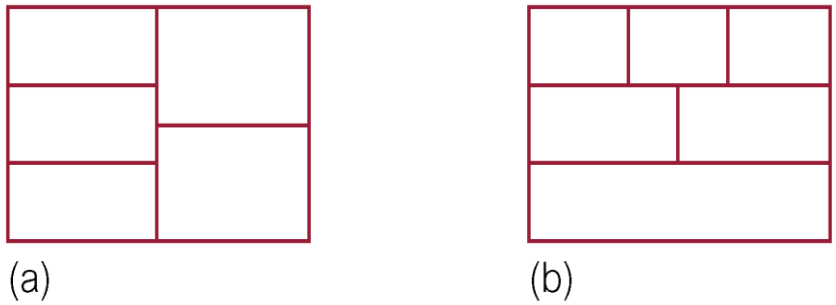
$$Z2 = \Pi_{cZam,\ Jméno\ Prijmeni,\ cPob}\ (Zamestnanec)$$

Původní relaci dostaneme pomocí operátoru přirozené spojení

Smíšená fragmentace

- Sestává z horizontálního fragmentu, který je vertikálně fragmentovaný, nebo vertikálního fragmentu, který je horizontálně fragmentovaný.
- Je definovaná pomocí operátorů selekce a projekce:

$$\sigma_p(\Pi_{a1, \ldots, an}(R))$$
$$\Pi_{a1, \ldots, an}(\sigma_p(R))$$



Příklad - smíšená fragmentace

Máme vertikálně fragmentovanou tabulku Zaměstnanec

$$Z_1 = \Pi_{cZam,\ pracZar,\ datumNaroz,\ plat}(Zaměstnanec)$$

$$Z_2 = \Pi_{cZam,\ Jmeno,\ prijmeni,\ cPob}(Zaměstnanec)$$

Tabulku Z_2 můžeme dále horizontálně fragmentovat dle příslušnosti k pobočce:

$$Z_{21} = \sigma_{cPob='P01'}(Z_2)$$

$$Z_{22} = \sigma_{cPob='P02'}(Z_2)$$

$$Z_{23} = \sigma_{cPob='P03'}(Z_2)$$

Odvozená horizontální fragmentace

Mnoho aplikací používá přirozené spojení relací. Pokud jsou relace uložené na různých místech, spojení je náročná operace. Je proto vhodné, aby spojované relace nebo jejich fragmenty byly na témže místě.
K tomu lze použít odvozenou horizontální fragmentaci.

- Odvozený horizontální fragment je založený na horizontální fragmentaci rodičovské relace.
- Zajišťuje, že fragmenty, které se používají často společně jsou uložené na stejném místě
- Jsou definované použitím polospojení :

$$R_i = R \textit{ semijoin } S_i, \qquad 1 \leq i \leq w$$

Příklad – odvozená horizontální fragmentace

Předpokládejme, že máme aplikaci, která používá přirozené spojení tabulek Zaměstnanec a Nemovitost, tabulka Zaměstnanec je horizontálně fragmentovaná.

$$Z_3 = \sigma_{cPob='P03'}(Zamestnanec)$$
$$Z_4 = \sigma_{cPob='P04'}(Zamestnanec)$$
$$Z_5 = \sigma_{cPob='P05'}(Zamestnanec)$$

Můžeme použít odvozenou fragmentaci pro Nemovitost

N_i = Nemovitost *semijoin* $_{cZam}$ Z_i, $3 \leq i \leq 5$

N_i obsahuje ty řádky tabulky Nemovitost, o které se starají zaměstnanci pobočky P_i

Alokace dat

Alternativní strategie vzhledem k umístění dat:

- centralizované,
- fragmentované,
- plně replikované,
- selektivně replikované.