II. FRAGMENTAREA RELAŢIILOR

1. Fragmentarea orizontală primară

Se consideră relaţia COUNTRIES_ALL.

Fie p_1 şi p_2 două predicate:

p_1: region_name= 'Europe'

p_2: region_name= 'Americas'

Observaţie:

Se presupune că:

- aceste predicate sunt cel mai des utilizate în aplicaţii;
- probabilităţile de acces ale fiecărei aplicaţii la orice tuplu din fragmentele orizontale ce sunt definite pe baza acestor predicate sunt egale;
- există cel puţin o aplicaţie care să acceseze în mod diferit fragmentele determinate de aceste predicate.

Exerciţiu:

a) să se fragmenteze orizontal relaţia COUNTRIES_ALL pe baza celor două predicate;

Observaţie: Fragmentul COUNTRIES_AM va fi stocat pe staţia 1 (*server1*), iar fragmentul COUNTRIES_EU va fi stocat pe staţia 2 (*server2*).

- conectaţi-vă la *bd_am* şi creaţi tabelul *countries_am_**** folosind informaţiile din tabelul *countries_all* pentru care numele regiunii este *Americas*.
- Încercaţi comanda:

```
CREATE TABLE countries_eu_***@bd_eu AS
       SELECT * FROM countries_all
       WHERE region_name='Europe';
```

Ce observaţi?

- conectaţi-vă la *bd_eu* şi creaţi tabelul *countries_eu_**** folosind informaţiile din tabelul distant *countries_all* (numele legăturii este *bd_am*) pentru care numele regiunii este *Europe*.

b) să se verifice că fragmentarea realizată este corectă;

- $Pr = \{p_1, p_2\}$este mulţimea de predicate simple, deoarece predicatele simple p_1 şi p_2 sunt cele mai des utilizate predicate în aplicaţii.
- P_r este completă şi minimală.
  - Deoarece există probabilităţi de acces egale ale fiecărei aplicaţii la orice tuplu care aparţine oricărui fragment orizontal ce este definit pe baza acestei mulţimi, rezultă că mulţimea *Pr* este completă.
  - Mulţimea *Pr* este minimală, deoarece toate predicatele mulţimii *Pr* sunt relevante (un predicat simplu determină divizarea unui fragment *f* în fragmentele f_i şi f_j, dacă există cel puţin o aplicaţie care să acceseze

fragmentele f_i şi f_j în mod diferit; acest predicat simplu va fi relevant pentru fragmentare).

- Se determină mulţimea *M* de predicate compuse pe baza mulţimii de predicate simple *Pr*.

  $Pr = \{p_1, p_2\}$, unde

  p_1: region_name = 'Europe'

  p_2: region_name = 'Americas'

  Prin interogarea tabelului COUNTRIES_ALL se deduce că

  region_name $\in$ {'Europe', 'Americas'}

  Se construieşte mulţimea de implicaţii *I*:

  i_1: (region_name= 'Europe') $\Rightarrow \neg$ (region_name= 'Americas')
  i_2: $\neg$ (region_name= 'Europe') $\Rightarrow$ (region_name= 'Americas')

  Se construieşte mulţimea M care are patru predicate compuse:

  m_1: (region_name= 'Europe') Λ (region_name= 'Americas')
  m_2: (region_name= 'Europe') $\Lambda \neg$ (region_name= 'Americas')
  m_3: $\neg$ (region_name= 'Europe') Λ (region_name= 'Americas')
  m_4: $\neg$ (region_name= 'Europe') $\Lambda \neg$ (region_name= 'Americas')

  Se observă că:

  - predicatele m_1 şi m_4 nu au sens în raport cu implicaţiile mulţimii *I* şi de aceea trebuie eliminate din mulţimea *M*;
  - $m_2 = p_1$ şi $m_1 = p_2$.

  Deci mulţimea de predicate care determină fragmentele este $M = Pr = \{p_1, p_2\}$

- Fragmentarea este corectă dacă sunt îndeplinite regulile de completitudine, reconstrucţie şi disjuncţie.

  Completitudinea: M este completă, deci proprietatea de completitudine este asigurată.

  Reconstrucţia: Se observă că:

  COUNTRIES_ALL = COUNTRIES_AM $\cup$ COUNTRIES_EU

  Disjuncţia:

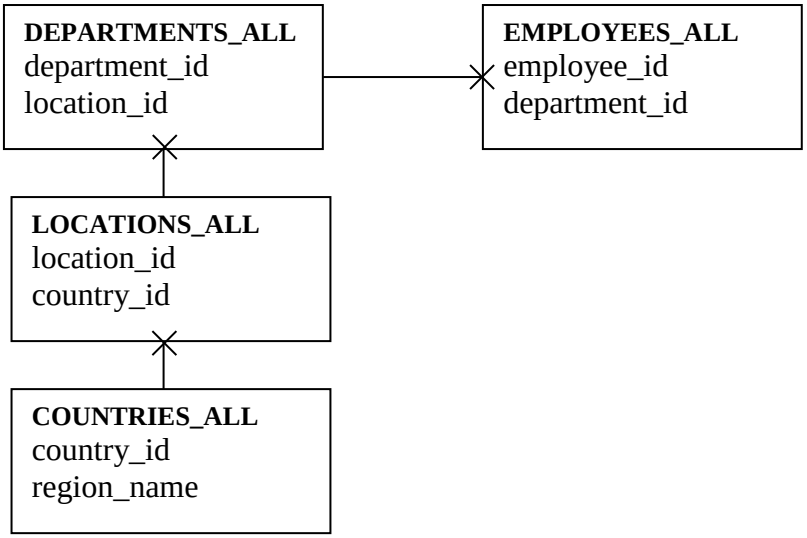
  COUNTRIES_AM $\cap$ COUNTRIES_EU = $\varnothing$

c) să se afişeze relaţia globală (toate ţările), utilizând cele două fragmente.

d) să se verifice cele 3 reguli de corectitudine a fragmentării, folosind comenzi SQL**.**

2. Fragmentarea orizontală derivată

Se consideră următoarele relaţii:



Aceste relaţii vor fi fragmentate aceste relaţii, pornind de la predicatele:

p_1: region_name= 'Europe'

p_2: region_name= 'Americas'

Fragmentele vor fi stocate astfel:

- pe staţia 1
  LOCATIONS_AM, DEPARTMENTS_AM, EMPLOYEES_AM
- pe staţia 2
  LOCATIONS_EU, DEPARTMENTS_EU, EMPLOYEES_EU

Exerciţiu:

a) creaţi fragmentul *locations_am_**** utilizând următoarele comenzi.

```
CONNECT magi_bdd/parola@bd_am;
CREATE TABLE locations_am_*** AS
     SELECT a.*
     FROM   locations_all a, countries_am_*** b
     WHERE  a.country_id=b.country_id;
```

b) creaţi fragmentele *departments_am_**** şi *employees_am_****.

c) creaţi fragmentul *locations_eu_**** utilizând următoarele comenzi.

```
CONNECT magi_bdd/parola@bd_eu;
CREATE TABLE locations_eu_*** AS
  SELECT a.*
  FROM   locations_all@bd_am a, countries_eu_*** b
  WHERE  a.country_id=b.country_id;
```

d) creaţi fragmentele *departments_eu_**** şi *employees_eu_****.

e) verificarea corectitudinii fragmentării orizontale derivate;

- LOCATIONS_ALL

Completitudinea: Este asigurată, deoarece fiecărui tuplu din cele două fragmente ale relaţiei LOCATIONS_ALL îi corespunde un tuplu din relaţia COUNTRIES_ALL.

Reconstrucţia: Se observă că:

LOCATIONS_ALL = LOCATIONS_AM ∪ LOCATIONS_EU

Disjuncţia: Este asigurată, deoarece graful ce reprezintă diagrama relaţiilor (COUNTRIES_ALL, LOCATIONS_ALL) şi a legăturilor dintre acestea este un graf simplu (există o singură legătură ce pleacă sau vine pentru orice fragment).

- DEPARTMENTS_ALL similar cu LOCATIONS_ALL;
- EMPLOYEES_ALL

Completitudinea: similar cu LOCATIONS_ALL.
Reconstrucţia: similar cu LOCATIONS_ALL.

Disjuncţia: Graful ce reprezintă diagrama relaţiilor (DEPARTMENTS_ALL, EMPLOYEES_ALL) şi a legăturilor dintre acestea nu este un graf simplu. În aceste condiţii se verifică valorile fiecărui tuplu.

f) verificaţi cele 3 reguli de corectitudine a fragmentării orizontale derivate utilizând comenzi *SQL*;

g) folosind fragmentele create anterior, să se listeze codul angajatului, numele departamentului, oraşul, ţara şi regiunea.

3. Fragmentarea verticală

Exerciţiu:

a) Să se aplice algoritmii de fragmentare verticală pentru relaţia JOBS_ALL (job_id, job_title, min_salary, max_salary), ştiind că:

- cele mai des utilizate cereri sunt următoarele:
  - q_1: SELECT min_salary FROM jobs_all WHERE max_salary = 10000;
  - q_2: SELECT min_salary FROM jobs_all WHERE min_salary > 10000;
  - q_3: SELECT max_salary FROM jobs_all WHERE min_salary > 10000;
  - q_4: SELECT job_title FROM jobs_all WHERE max_salary = 40000;
- $nr\_acc_l(q_k)$ reprezintă numărul de accesări ale atributelor (A_i, A_j) pentru fiecare execuţie a cererii q_k pe staţia S_l; (presupunem că $nr\_acc_l(q_k) = 1 \ \ \forall k \in \overline{1,4}, \ \forall l \in \overline{1,2}$.)
- $fr\_acc_l(q_k)$ reprezintă frecvenţa de acces a cererii q_k pe staţia S_l.

Staţia 1	***Staţia 2***
$fr\_acc_1(q_1) = 20$	$fr\_acc_2(q_1) = 0$
$fr\_acc_1(q_2) = 0$	$fr\_acc_2(q_2) = 10$
$fr\_acc_1(q_3) = 20$	$fr\_acc_2(q_3) = 0$
$fr\_acc_1(q_4) = 10$	$fr\_acc_2(q_4) = 5$

Soluţie:

Fie A_0 = job_id, A_1 = job_title, A_2 = min_salary, A_3 = max_salary.

$$ref(q_i, A_j) = \begin{cases} 1, & dacă\ A_j\ este\ referit\ de\ q_i \\ 0, & altfel \end{cases}$$

- Valorile folosirii atributelor sunt definite de matricea *VA*, $VA(i, j) = ref(q_i, A_j)$.

VA	**A1**	**A2**	**A3**
q1	0	1	1
q2	0	1	0
q3	0	1	1
q4	1	0	1

- Matricea afinităţii atributelor:

$$af(A_i, A_j) = \sum_{k \in K} \sum_{\forall S_l} nr\_acc_l(q_k)\, f\_acc_l(q_k),$$

unde $K = \{k \mid ref(q_k, A_i) = 1 \wedge ref(q_k, A_j) = 1\}$;

$af(A_1, A_2) = 0$

$af(A_1, A_3) = fr\_acc_1(q_4) + fr\_acc_2(q_4) = 10+5 = 15$

$af(A_2, A_3) = fr\_acc_1(q_1) + fr\_acc_2(q_1) + fr\_acc_1(q_3) + fr\_acc_2(q_3)$

$= 20 + 0 + 10 + 0 = 40$

$af(A_1, A_1) = fr\_acc_1(q_4) + fr\_acc_2(q_4) = 10 + 5 = 15$

$af(A_2, A_2) = fr\_acc_1(q_1) + fr\_acc_2(q_1) + fr\_acc_1(q_2) + fr\_acc_2(q_2) + fr\_acc_1(q_3)$

$+ fr\_acc_2(q_3) = 50$

$af(A_3, A_3) = fr\_acc_1(q_1) + fr\_acc_2(q_1) + fr\_acc_1(q_3) + fr\_acc_2(q_3) + fr\_acc_1(q_4)$

$+ fr\_acc_2(q_4) = 55$

AA	**A1**	**A2**	**A3**
A1	15	0	15
A2	0	50	40
A3	15	40	55

Matricea afinităţii atributelor este utilizată pentru realizarea fragmentării verticale. Acest proces implică două etape:

- gruparea atributelor cu afinităţi mari;
- divizarea relaţiei în raport cu grupurile de atribute formate.

Algoritmul de grupare al atributelor determină grupurile de atribute similare din punct de vedere al afinităţii.

Matricea legăturilor de afinitate se generează în trei paşi:

- iniţializarea – se plasează şi se fixează o coloană arbitrară a matricei *AA* în matricea *LA*;
- iteraţia – se alege fiecare coloană, neutilizată încă, din *AA* (se va alege acea coloană care aduce cea mai mare contribuţie la măsura afinităţii globale) şi se plasează în *LA* pe una dintre poziţiile disponibile; acest pas se repetă până când au fost alese toate coloanele matricei *AA*;
- ordonarea liniilor – se permută liniile matricei *LA* astfel încât să fie păstrată simetria.

Se consideră că $LA(0, j) = LA(i, 0) = LA(n + 1, j) = LA(i, n + 1) = 0$.

Contribuţia globală a afinităţii atributelor obţinută prin plasarea atributului A_k între atributele A_i şi A_j va fi dată prin formula:

$$cont(A_i, A_k, A_j) = 2leg(A_i, A_k) + 2leg(A_k, A_j) - 2leg(A_i, A_j),$$

unde $leg(A_x, A_y) = \sum_{i=1}^{n} af(A_i, A_x) af(A_i, A_y)$.

CA	**1 (A_1)**	**2 (A_2)**	**3**
1	15	0	
2	0	50	
3	15	40	

$index \leftarrow 3$

$$cont(A_0, A_3, A_1) = 2*leg(A_0, A_3) + 2*leg(A_3, A_1) - 2*leg(A_0, A_1) =$$
$$= 2*\sum_{i=1}^{4} af(A_i, A_0) af(A_i, A_3) + 2*\sum_{i=1}^{4} af(A_i, A_3) af(A_i, A_1) -$$
$$- 2*\sum_{i=1}^{4} af(A_i, A_0) af(A_i, A_1) = 2100$$

$$cont(A_1, A_3, A_2) = 2*leg(A_1, A_3) + 2*leg(A_3, A_2) - 2*leg(A_1, A_2) =$$
$$= 2*\sum_{i=1}^{4} af(A_i, A_1) af(A_i, A_3) + 2*\sum_{i=1}^{4} af(A_i, A_3) af(A_i, A_2) -$$
$$- 2*\sum_{i=1}^{4} aff(A_i, A_1) aff(A_i, A_2) = 9300$$

$$cont(A_2, A_3, A_0) = 2*leg(A_2, A_3) + 2*leg(A_3, A_0) - 2*leg(A_2, A_0) =$$
$$= 2*\sum_{i=1}^{4} af(A_i, A_2) af(A_i, A_3) + 2*\sum_{i=1}^{4} af(A_i, A_3) af(A_i, A_0) -$$
$$- 2*\sum_{i=1}^{4} af(A_i, A_2) af(A_i, A_0) = 8400$$

loc = 2 (A_1, A_3, A_2)

LA	1 (A_1)	2 (A_3)	3 (A_2)
1	15	15	0
2	0	40	50
3	15	55	40

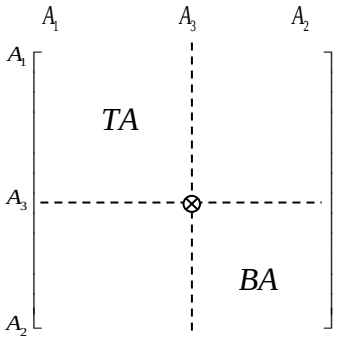
Se permută liniile, corespunzător permutării realizate la coloane (astfel matricea rămâne simetrică).

LA	1 (A_1)	2 (A_3)	3 (A_2)
1 (A_1)	15	15	0
2 (A_3)	15	55	40
3 (A_2)	0	40	50

Algoritmul de divizare a unei relaţii presupune găsirea mulţimilor de atribute care sunt accesate independent sau sunt accesate des de mai multe mulţimi distincte de aplicaţii.

Se consideră matricea legăturilor de afinitate *LA*.

$Q = \{q_1, q_2, q_3, q_4\}$



TQ – aplicaţii ce accesează doar atributele din *TA*;
BQ – aplicaţii ce accesează doar atributele din *BA*;

OQ – aplicaţii ce accesează atât atribute din *TA* cât şi atribute din *BA*

$CQ = \sum_{q_i \in Q} \sum_{\forall S_j} fr\_acc_j(q_i)$, costul total al accesării oricărui atribut *A*;

$CTQ = \sum_{q_i \in TQ} \sum_{\forall S_j} fr\_acc_j(q_i)$, costul accesării unui atribut din *TA* de pe orice staţie;

$CBQ = \sum_{q_i \in BQ} \sum_{\forall S_j} fr\_acc_j(q_i)$, costul accesării unui atribut din *BA* de pe orice staţie;

$COQ = \sum_{q_i \in OQ} \sum_{\forall S_j} fr\_acc_j(q_i)$, costul accesării unui atribut din *TA* sau din *BA* de pe orice staţie.

n = 1
$TQ_1 = \varnothing$ (aplicaţiile care accesează doar atributul A_1)
$BQ_1 = \{q_1, q_2, q_3\}$ (aplicaţiile care accesează doar atributele A_3 şi A_2)
$OQ_1 = \{q_4\}$

$CTQ_1 = 0$

$$CBQ_1 = \sum_{q_i \in BQ_1} \sum_{\forall S_j} fr\_acc_j(q_i) = fr\_acc_1(q_1) + fr\_acc_2(q_1) + fr\_acc_1(q_2) + fr\_acc_2(q_2) + fr\_acc_1(q_3) + fr\_acc_2(q_3) = 50$$

$$COQ_1 = \sum_{q_i \in OQ_1} \sum_{\forall S_j} fr\_acc_j(q_i) = fr\_acc_1(q_4) + fr\_acc_2(q_4) = 15$$

$z_1 = CTQ_1 * CBQ_1 - COQ_1^2 = -225$

n = 2

$TQ_2 = \{q_4\}$ (aplicaţiile care accesează doar atributele A_1 şi A_3)
$BQ_2 = \{q_2\}$ (aplicaţiile care accesează doar atributul A_2)
$OQ_2 = \{q_1, q_3\}$

$$CTQ_2 = \sum_{q_i \in TQ_2} \sum_{\forall S_j} fr\_acc_j(q_i) = fr\_acc_1(q_4) + fr\_acc_2(q_4) = 15$$

$$CBQ_2 = \sum_{q_i \in BQ_2} \sum_{\forall S_j} fr\_acc_j(q_i) = fr\_acc_1(q_2) + fr\_acc_2(q_2) = 10$$

$$COQ_2 = \sum_{q_i \in OQ_2} \sum_{\forall S_j} fr\_acc_j(q_i) = fr\_acc_1(q_1) + fr\_acc_2(q_1) + fr\_acc_1(q_3) + fr\_acc_2(q_3) = 40$$

$z_2 = CTQ_2 * CBQ_2 - COQ_2^2 = -1450$

Rezultă că optim este $z = z_1 = -225$, deci punctul de partiţionare este 1.

A_0 este cheia primară şi va fi conţinută de ambele fragmente. Deci cele două fragmente sunt:

$\{A_0, A_1\}$ şi $\{A_0, A_2, A_3\}$.

b) Să se creeze cele două fragmente obţinute la punctul a).
- *jobs_am_**** pe *server1* (*bd_am*);
- *jobs_eu_**** pe *server2* (*bd_eu*).

c) Utilizând comenzi *SQL*, să se verifice corectitudinea fragmentării verticale realizate.

În urma fragmentărilor s-au obţinut:

STAŢIA 1	STAŢIA 2
COUNTRIES_AM	**COUNTRIES_EU**
DEPARTMENTS_AM	**DEPARTMENTS_EU**
EMPLOYEES_AM	**EMPLOYEES_EU**
JOBS_AM	**JOBS_EU**
LOCATIONS_AM	**LOCATIONS_EU**
JOB_GRADES_ALL	