DISTRIBUTED DATABASE SYSTEMS

Dr. Reda M. Hussien

COM_MIN Algorithm

- Given: a relation R and a set of simple predicates p_r
- Output: a complete and minimal set of simple predicates $P_{\acute{r}}$ for P_r
- Rule 1: a relation or fragment is partitioned into at least two parts which are accessed differently by at least one application.

Algorithm 3.1: COM_MIN Algorithm

Input: R: relation; Pr: set of simple predicates
Output: Pr': set of simple predicates
Declare: F: set of minterm fragments
begin
  find $p_i \in Pr$ such that p_i partitions R according to *Rule* 1 ;
  $Pr' \leftarrow p_i$;
  $Pr \leftarrow Pr - p_i$;
  $F \leftarrow f_i$ $\{f_i$ is the minterm fragment according to $p_i\}$;
  repeat
    find a $p_j \in Pr$ such that p_j partitions some f_k of Pr' according to *Rule* 1 ;
    $Pr' \leftarrow Pr' \cup p_j$;
    $Pr \leftarrow Pr - p_j$;
    $F \leftarrow F \cup f_j$;
    if $\exists p_k \in Pr'$ *which is not relevant* **then**
      $Pr' \leftarrow Pr' - p_k$;
      $F \leftarrow F - f_k$;
  until Pr' *is complete* ;
end

COM_MIN Algorithm

PHORIZONTAL Algorithm

Algorithm 3.2: PHORIZONTAL Algorithm

Input: R: relation; Pr: set of simple predicates
Output: M: set of minterm fragments
begin
  $Pr' \leftarrow$COM_MIN(R, Pr) ;
  determine the set M of minterm predicates ;
  determine the set I of implications among $p_i \in Pr'$;
  foreach $m_i \in M$ **do**
    if m_i *is contradictory according to* I **then**
      $M \leftarrow M - m_i$
end

PHF – Example

- Two candidate relations : PAY and PROJ.
- Fragmentation of relation PAY
  - Application: Check the salary info and determine raise.
  - Employee records kept at two sites ▫ application run at two sites
  - Simple predicates
    - p1 : SAL ≤ 30000
    - p2 : SAL > 30000
    - Pr = {p1,p2} which is complete and minimal Pr'=Pr
  - Minterm predicates
    - m1 : (SAL ≤ 30000)
    - m2 : NOT(SAL ≤ 30000) = (SAL > 30000)

PHF – Example

PAY_1

TITLE	SAL
Mech. Eng.	27000
Programmer	24000

PAY_2

TITLE	SAL
Elect. Eng.	40000
Syst. Anal.	34000

PHF – Example

- Fragmentation of relation PROJ
  - Applications:
    - Find the name and budget of projects given their no.
      - Issued at three sites
    - Access project information according to budget
      - one site accesses ≤200000 other accesses >200000

PHF – Example

- Fragmentation of relation PROJ
  - Simple predicates
  - For application (1)
    - p1 : LOC = “Montreal”
    - p2 : LOC = “New York”
    - p3 : LOC = “Paris”
  - For application (2)
    - p4 : BUDGET ≤ 200000
    - p5 : BUDGET > 200000
  - $P_r = P_{r'} = \{p1, p2, p3, p4, p5\}$

PHF – Example

- Fragmentation of relation PROJ continued
  - Minterm fragments left after elimination
    - m1 : (LOC = “Montreal”) $\wedge$ (BUDGET $\leq$ 200000)
    - m2 : (LOC = “Montreal”) $\wedge$ (BUDGET > 200000)
    - m3 : (LOC = “New York”) $\wedge$ (BUDGET $\leq$ 200000)
    - m4 : (LOC = “New York”) $\wedge$ (BUDGET > 200000)
    - m5 : (LOC = “Paris”) $\wedge$ (BUDGET $\leq$ 200000)
    - m6 : (LOC = “Paris”) $\wedge$ (BUDGET > 200000)

PHF – Example

$PROJ_1$

PNO	PNAME	BUDGET	LOC
P1	Instrumentation	150000	Montreal

$PROJ_3$

PNO	PNAME	BUDGET	LOC
P2	Database Develop.	135000	New York

$PROJ_4$

PNO	PNAME	BUDGET	LOC
P3	CAD/CAM	255000	New York

$PROJ_6$

PNO	PNAME	BUDGET	LOC
P4	Maintenance	310000	Paris

PHF – Correctness

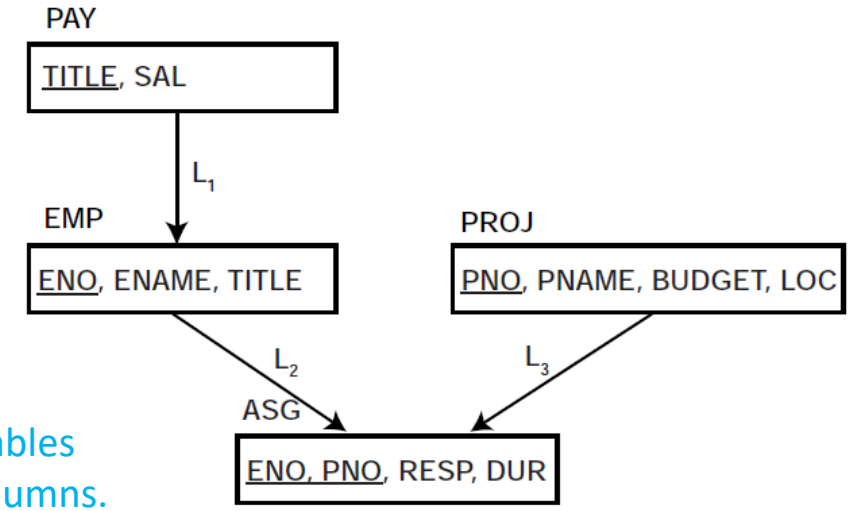
- Completeness
  - Since $P_{\acute{r}}$ is complete and minimal, the selection predicates are complete
- Reconstruction
  - If relation R is fragmented into $F_R = \{R_1, R_2, \ldots, R_r\}$
    - $R = \cup_{\forall R_i \in FR} R_i$
- Disjointness
  - Minterm predicates that form the basis of fragmentation should be mutually exclusive.

DERIVED HORIZONTAL FRAGMENTATION

Derived Horizontal Fragmentation

- Defined on a member relation of a link according to a selection operation specified on its owner.
  - Each link is an equijoin.
  - Equijoin can be implemented by means of semijoins.

links between database objects (i.e., relations in our case)
Link owner: The relation at the tail of a link
Link member: relation at the head



equijoin is a type of join that combines tables based on matching values in specified columns.

DHF – Definition

- Given a link L where $owner(L) = S$ and $member(L) = R$, the derived horizontal fragments of R are defined as

$$R_i = R \ltimes S_i, 1 \leq i \leq w$$

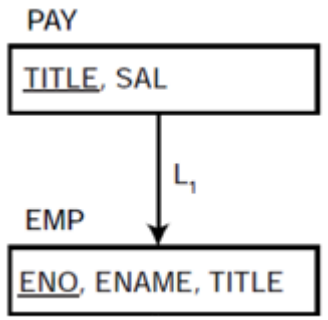
- where w is the maximum number of fragments that will be defined on R and

$$S_i = \sigma F_i(S)$$

- where F_i is the formula according to which the primary horizontal fragment S_i is defined.

DHF – Example

- Given link L_1 where
  - $owner(L_1) = PAY$
  - $member(L_1) = EMP$
- Then we can group engineers into two groups according to their salary
  - $EMP_1 = EMP \ltimes PAY_1$
  - $EMP_2 = EMP \ltimes PAY_2$
  - Where
    - $PAY_1 = \sigma_{SAL \leq 30000}(PAY)$
    - $PAY_2 = \sigma_{SAL > 30000}(PAY)$



EMP

ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng
E2	M. Smith	Syst. Anal.
E3	A. Lee	Mech. Eng.
E4	J. Miller	Programmer
E5	B. Casey	Syst. Anal.
E6	L. Chu	Elect. Eng.
E7	R. Davis	Mech. Eng.
E8	J. Jones	Syst. Anal.

PAY

TITLE	SAL
Elect. Eng.	40000
Syst. Anal.	34000
Mech. Eng.	27000
Programmer	24000

EMP_1

ENO	ENAME	TITLE
E3	A. Lee	Mech. Eng.
E4	J. Miller	Programmer
E7	R. Davis	Mech. Eng.

EMP_2

ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng.
E2	M. Smith	Syst. Anal.
E5	B. Casey	Syst. Anal.
E6	L. Chu	Elect. Eng.
E8	J. Jones	Syst. Anal.

PAY1

TITLE	SAL
Mech. Eng.	27000
Programmer	24000

PAY2

TITLE	SAL
Elect. Eng.	40000
Syst. Anal.	34000

DHF – Correctness

- Completeness
  - Referential integrity
    - Let R be the member relation of a link whose owner is relation S which is fragmented as

$$F_S = \{S_1, S_2, \dots, S_n\}$$

    - Furthermore, let A be the join attribute between R and S. Then, for each tuple t of R, there should be a tuple $t'$of S such that

$$t[A] = t'[A]$$

  - Reconstruction
    - Same as primary horizontal fragmentation.
  - Disjointness
    - disjointness is guaranteed as long as the minterm predicates determining the fragmentation are mutually exclusive.

VERTICAL FRAGMENTATION

Vertical fragmentation

- Vertical fragmentation (VF) will group the columns of a table into fragments.
- VF must be done in such a way that the original table can be reconstructed from the fragments.
- This fragmentation requirement is called “reconstructiveness.”
- each VF fragment must contain the primary key column(s) of the table.
  - Because each fragment contains a subset of the total set of columns in the table
- To create a vertical fragment from a table:
  - a select statement is used in which “Column_list” is a list of columns from R that includes the primary key.