2.4 Data Fragmentation

In this section it is presented the main fragmentation strategies. As mentioned previously, there are two fundamental fragmentation strategies: horizontal and vertical. Furthermore, there is a possibility of nesting fragments in a hybrid fashion.

2.4.1 Horizontal Fragmentation

Horizontal fragmentation partitions a relation along its tuples. Thus each fragment has a subset of the tuples of the relation. There are two versions of horizontal partitioning: primary and derived. *Primary horizontal fragmentation* of a relation is performed using predicates that ate defined on that relation. *Derived horizontal fragmentation*, on the other hand, is partitioning of a relation that results from predicates being defined on another relation.

Figure 3.7 shows the expression of links among the database relations. Note that the direction of the link shows a one-to-many relationship. For example, for each title there are multiple employees with that title; thus there is a link between the PAY and EMP relations. Along the same lines, the many-to-many relationship between the EMP and PROJ relations is expressed with two links to the ASG relation. The relation at the tail link is called the *owner* (source) of the link and the relation at the head is called the *member* (target).

Example. Given link L_1 of Figure 3.7, the *owner and member* functions have the following values:

owner (L_1) = PAY
member (L_1) = EMP

The fundamental qualitative information consists of the predicates used in user queries. At this point we are interested in determining simple predicates. Given a relation $R(A_1, A_2, ..., A_n)$, where A_i is an attribute defined over domain D_i, a simple predicate p_j defined on R has the form

p_j: $A_i \; \theta \; Value$

where $\theta \in \{=, <, \neq, \leq, >, \geq\}$ and *Value* is chosen from the domain of A_i ($Value \in D_i$). We use Pr_i to denote the set of all simple predicates defined on a relation R_i. The members of Pr_i are denoted by p_{ij}.

Example. Given the relation instance PROJ of Figure 3.3

PNAME = “Maintenance”
Is a simple predicate, as well as
BUDGET ≤ 200000

User queries quite often include more complicated predicates, which are Boolean combinations of simple predicates. One combination that we are particularly interested in, called a *minterm predicate*, is the conjunction of simple predicates. Given a set $Pr_i = \{p_{i1}, p_{i2}, ..., p_{im}\}$ of simple predicates for relation R_i, there are a set of *minterm* predicates $M_i = \{m_{i1}, m_{i2}, ..., m_{iz}\}$.

Example. Consider relation PAY of Figure 3.3. The following are some of the possible simple predicates that can be defined on PAY.

p_1: TITLE = "Elect. Eng."
p_2: TITLE = "Syst. Anal."
p_3: TITLE = "Mech. Eng."
p_4: TITLE = "Programmer"
p_5: SAL ≤ 30000

The following are some of the minterm predicates that can be defined base don these simple predicates.

m_1: TITLE = "Elect. Eng." ∧ SAL ≤ 30000
m_2: TITLE = "Elect. Eng." ∧ SAL > 30000
m_3: ¬(TITLE = "Elect. Eng.") ∧ SAL ≤ 30000
m_4: ¬(TITLE = "Elect. Eng.") ∧ SAL > 30000
m_5: TITLE = "Programmer" ∧ SAL ≤ 30000
m_6: TITLE = "Programmer" ∧ SAL > 30000

2.4.1.1 Primary Horizontal Fragmentation

A primary horizontal fragmentation is defined by a selection operation on the owner relations of database schema. Therefore, given relation R, its horizontal fragments are given by

$$R_i = \sigma_{Fi}(R), 1 \le i \le w$$

Where F_i is the selection formula used to obtain fragment R_i (also called *fragmentation predicate*).

Example. Consider relation PROJ of Figure 3.3, we can define the following horizontal fragments based on the project location. The resulting fragments are shown in Figure 3.8.

$PROJ_1 = \sigma_{LOC = \text{"Montreal"}}(PROJ)$
$PROJ_2 = \sigma_{LOC = \text{"New York"}}(PROJ)$
$PROJ_3 = \sigma_{LOC = \text{"Paris"}}(PROJ)$

2.4.1.2 Derived Horizontal Fragmentation

A derived horizontal fragmentation is defined on a member relation of a link according to a selection operation specified on its owner. It is important to remark two points. First, the link between the owner and the member relations is defined by means of semijoins. Accordingly, 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 define on R, and $S_i = \sigma_{Fi}(S)$, where F_i is the formula according to which the primary horizontal fragment S_i is defined.

Example. Consider link L_1 in Figure 3.7 where *owner* (L_1) = PAY and *member* (L_1) = EMP. Then we can group engineers into two groups according to their salary: those making less than or equal to $30,000, and those making more than $30,000. The two fragments EMP1 and EMP2 are defined as follows:

EMP1 = EMP $\ltimes$ PAY1
EMP2 = EMP $\ltimes$ PAY2

Where

PAY1 = $\sigma_{SAL \leq 30000}$(PAY)
PAY2 = $\sigma_{SAL > 30000}$(PAY)

The result of this fragmentation is depicted in Figure 3.11. To carry out derived horizontal fragmentation, three inputs are needed: the set of partitions of the owner relation (e.g., PAY1 and PAY2), the member relation, and the set of semijoin predicates between the owner and the member (e.g. EMP.TITLE = PAY.TITLE).

2.4.2 Vertical Fragmentation

Vertical fragmentation of a relation R produces fragments R_1, R_2, …, R_r, each of which contains a subset of R's attributes as well as the primary key of R. The objective of vertical fragmentation is to partition a relation into a set of smaller relations so that many of the user applications will run on only one fragment. In this context, an "optimal" fragmentation is one that produces a fragmentation scheme, which minimizes the execution time user applications that run on these fragments.

Vertical partitioning is inherently more complicated than horizontal partitioning. This is due the total number of alternatives that are available. For example, in horizontal partitioning, if the total of simple predicates in P_r is n, there are 2^n possible minterm predicates that can be defined on it. In addition, we know that some of these will contradict the existing implications, further reducing the candidate fragments that need to be considered. In case of vertical partitioning,

however, if a relation has m non-primary key attributes, the number of possible fragments is equal to B(m), which is the mth Bell number. For large values of m, $B(m) \approx m^m$; for example, for m=10, B(m) ≈ 115,000, for m = 15, B(m) ≈ 10^9, for m=30, B(m) = 10^{23}.

These values indicate that it is futile to attempt to obtain optimal solutions to the vertical partitioning problem; one has to resort to heuristics. Two types of heuristic approaches exist for the vertical fragmentation of global relations:

a) *Grouping:* starts by assigning each attribute to one fragment, and at each step, joins some of the fragments until some criteria is satisfied. Grouping was first suggested for centralized databases and was used later for distributed databases.
b) *Splitting:* starts with the relation and decides on beneficial partitionings based on the access behavior of applications to the attributes. The technique was also first discussed for centralized database design. It was then extended to the distributed environment.

The splitting technique fits more naturally within the top-down design methodology, since the "optimal" solution is probably closer to the full relation than to a set of fragments each of which consist of a single attribute. Furthermore, splitting generates non-overlapping fragments whereas grouping typically results in overlapping fragments. We prefer non-overlapping fragments for disjointness which refers to non-primary key attributes. In addition, the replication of the global relation's key into the fragments is a characteristic of vertical fragmentation that allows the reconstruction of the global relation. Therefore, splitting is considered only for those attributes that do not participate in the primary key.

2.4.3 Hybrid Fragmentation

In most cases a simple horizontal or vertical fragmentation of a database scheme will not be sufficient to satisfy the requirements of user applications. In this case a vertical fragmentation may be followed by a horizontal one, or vice versa, producing a tree structured partitioning (Figure 3.19). Since two types of partitioning strategies are applied one after the other, this alternative is called hybrid fragmentation. It has also been named *mixed* fragmentation or *nested* fragmentation.

A good example for the necessity of hybrid fragmentation is relation PROJ. We can partition PROJ into six horizontal fragments based on two applications, and vertically into two. What we have, therefore, is a set of horizontal fragments, each of which is further partitioned into two vertical fragments. The number of levels of nesting can be larger, but it is certainly finite. In the case of horizontal fragmentation, one has to stop when each fragment consist of only one tuple, whereas the termination point for vertical fragmentation is one attribute per fragment.