INSTITUTO POLITÉCNICO NACIONAL

UNIDAD PROFESIONAL INTERDISCIPLINARIA DE INGENIERÍA EN TECNOLOGÍAS AVANZADAS

Ingeniería Telemática

Bases de Datos Distribuidas

Tarea 3.

- **Docente**: De la Cruz Sosa Carlos
- **Alumnos:** De Jesús Lucio Oscar Daniel
  Pérez Iturbe Carolina
- **Grupo:** 3TM3
- **Fecha de entrega**: 04 de junio 2025
- **Ciclo escolar:** 2025-2

Informacion sobre los algoritmos que puedo aplicar en la fragmentación horizontal primaria en bases de datos relacionales basado en la teoría de Tamer Ozsu:
CHATGPT:

En la **fragmentación horizontal primaria** en bases de datos distribuidas relacionales, el objetivo es dividir una relación (tabla) en subconjuntos de tuplas que serán almacenados en distintos sitios o nodos, basándose en un criterio que maximice la eficiencia del sistema. Según la **teoría y metodología de Tamer Özsu y Patrick Valduriez**, autores reconocidos por su trabajo en bases de datos distribuidas, existen **varios algoritmos y enfoques** que se pueden aplicar para llevar a cabo esta fragmentación. A continuación, te resumo los más relevantes según su libro *Principles of Distributed Database Systems*:

🔷 1. Fragmentación Horizontal Simple (Basada en Predicados)

También conocida como **fragmentación horizontal primaria**, consiste en dividir una relación en fragmentos usando **predicados de selección** que derivan directamente de las consultas más frecuentes.

Algoritmo básico:

1. **Análisis del esquema y las aplicaciones** para determinar predicados relevantes (conjuntos de condiciones que aparecen comúnmente en consultas).
2. Generar fragmentos mediante la **selección** de tuplas que cumplen los predicados.
3. Asegurarse de que los fragmentos sean:
    - **Completos** (la unión de todos los fragmentos reconstruye la relación original).
    - **Disjuntos** (una tupla pertenece solo a un fragmento).
    - **Reconstruibles** (posibilidad de reconstruir la relación con operaciones relacionales como la unión).

Ejemplo:

Para una relación EMPLEADOS con atributo departamento, se pueden usar predicados como:

sql
CopiarEditar
P1: departamento = 'Ventas'
P2: departamento = 'Finanzas'

🔷 2. Algoritmo de Min-Term Predicates

Este es uno de los algoritmos **formales y sistemáticos** propuestos por Özsu para asegurar una fragmentación válida.

Pasos del algoritmo:

1. **Identificar el conjunto de predicados simples** a partir de las consultas del sistema.
2. **Generar los predicados min-term**, que son combinaciones de predicados y sus negaciones.
3. **Eliminar min-terms inconsistentes** (como $x > 10 \land x < 5$).
4. Usar los min-terms resultantes como **definición de los fragmentos**.

Ventajas:

- Garantiza disjunción y completitud.
- Basado en análisis lógico, no solo en heurísticas.

🔷 3. Algoritmo Basado en el Uso (Query Usage-Based)

Este algoritmo fragmenta la relación en función del **patrón de acceso de las consultas**.
Proceso:

1. Analizar un conjunto de consultas representativas.
2. Identificar condiciones frecuentes en la cláusula WHERE.
3. Crear fragmentos que **optimicen las consultas más comunes**, mejorando el rendimiento al minimizar el acceso a datos innecesarios.

◆ 4. Fragmentación derivada de Clustering de Consultas

A veces se utiliza un enfoque más dinámico basado en **análisis estadístico o de clustering** de las consultas.
Técnica:

- Se construye una matriz de acceso (consultas vs. atributos).
- Se aplica un algoritmo de clustering para agrupar consultas similares.
- Luego se definen fragmentos que satisfacen los grupos de consultas.

Consideraciones adicionales según Özsu:

- **Localización de fragmentos:** Se decide en qué sitio físico se almacenará cada fragmento.
- **Balanceo de carga:** Los fragmentos deben distribuirse para no sobrecargar nodos específicos.
- **Seguridad y autonomía local:** A veces se busca fragmentar por razones administrativas o legales.

Libro:

Before we present a formal algorithm for horizontal fragmentation, we intuitively discuss the process for primary (and derived) horizontal fragmentation. A primary horizontal fragmentation is defined by a selection operation on the owner relations of a database schema. Therefore, given relation R, its horizontal fragments are given by

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

where F_i is the selection formula used to obtain fragment R_i (also called the fragmentation predicate). Note that if F_i is in conjunctive normal form, it is a minterm predicate (m_i). The algorithm we discuss will, in fact, insist that F_i be a minterm predicate.
Example 3.7.
The decomposition of relation PROJ into horizontal fragments PROJ1 and PROJ2 in Example 3.1 is defined as follows:

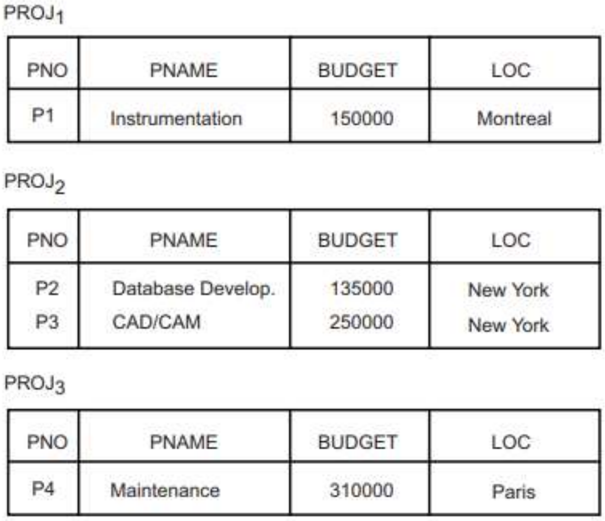
$$PROJ_1 = \sigma_{BUDGET} \leq 200000\ (PROJ)$$
$$PROJ_2 = \sigma_{BUDGET} > 200000\ (PROJ)$$

Example 3.7 demonstrates one of the problems of horizontal partitioning. If the domain of the attributes participating in the selection formulas are continuous and infinite, as in Example 3.7, it is quite difficult to define the set of formulas F = { F_1, F_2, ..., F_n} that would fragment the relation properly. One possible course of action is to define ranges as we have done in Example 3.7. However, there is always the problem of handling the two endpoints. For example, if a new tuple with a BUDGET value of, say, $600,000 were to be inserted into PROJ, one would have had to review the fragmentation to decide if the new tuple is to go into $PROJ_2$ or if the fragments need to be revised and a new fragment needs to be defined as

$$PROJ_2 = \sigma_{200000} < BUDGET \leq 400000\ (PROJ)$$
$$PROJ_3 = \sigma_{BUDGET} > 400000\ (PROJ)$$

Example 3.8. 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.

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



PROJ1

PNO	PNAME	BUDGET	LOC
P1	Instrumentation	150000	Montreal

PROJ2

PNO	PNAME	BUDGET	LOC
P2	Database Develop.	135000	New York
P3	CAD/CAM	250000	New York

PROJ3

PNO	PNAME	BUDGET	LOC
P4	Maintenance	310000	Paris

Fig. 3.8 Primary Horizontal Fragmentation of Relation PROJ

Now we can define a horizontal fragment more carefully. A horizontal fragment R_i of relation R consists of all the tuples of R that satisfy a minterm predicate m_i . Hence, given a set of minterm predicates M, there are as many horizontal fragments of relation R as there are minterm predicates. This set of horizontal fragments is also commonly referred to as the set of *minterm fragments.*

From the foregoing discussion it is obvious that the definition of the horizontal fragments depends on minterm predicates. Therefore, the first step of any fragmentation algorithm is to determine a set of simple predicates that will form the minterm predicates.

An important aspect of simple predicates is their *completeness*; another is their *minimality*. A set of simple predicates P_r is said to be complete if and only if there is an equal probability of access by every application to any tuple belonging to any minterm fragment that is defined according to Pr.

Example 3.9. Consider the fragmentation of relation PROJ given in Example 3.8. If the only application that accesses PROJ wants to access the tuples according to the location, the set is complete since each tuple of each fragment PROJi (Example 3.8) has the same probability of being accessed. If, however, there is a second application which accesses only those project tuples where the budget is less than or equal to $200,000, then Pr is not complete. Some of the tuples within each PROJi have a higher probability of being accessed due to this second application. To make the set of predicates complete, we need to add (BUDGET ≤ 200000, BUDGET > 200000) to Pr:

$$Pr = \{LOC = \text{“}Montreal\text{”}, LOC = \text{“}New\ York\text{”}, LOC = \text{“}Paris\text{”},$$
$$BUDGET \leq 200000, BUDGET > 200000\}$$

The reason completeness is a desirable property is because fragments obtained according to a complete set of predicates are logically uniform since they all satisfy the minterm predicate. They are also statistically homogeneous in the way applications access them. These characteristics ensure that the resulting fragmentation results in a balanced load (with respect to the given workload) across all the fragments.

Therefore, we will use a complete set of predicates as the basis of primary horizontal fragmentation.

It is possible to define completeness more formally so that a complete set of predicates can be obtained automatically. However, this would require the designer to specify the access probabilities for each tuple of a relation for each application under consideration. This is considerably more work than appealing to the common sense and experience of the designer to come up with a complete set. Shortly, we will present an algorithmic way of obtaining this set.

The second desirable property of the set of predicates, according to which minterm redicates and, in turn, fragments are to be defined, is minimality, which is very intuitive. It simply states that if a predicate influence how fragmentation is performed (i.e., causes a fragment f to be further fragmented into, say, f_i and f_j), there should be at least one application that accesses fi and fj differently. In other words, the simple predicate should be relevant in determining a fragmentation. If all the predicates of a set P_r are relevant, P_r is *minimal.*

A formal definition of relevance can be given as follows [Ceri et al., 1982b]. Let m_i and m_j be two minterm predicates that are identical in their definition, except that m_i contains the simple predicate p_i in its natural form while m_j contains $\neg p_i$. Also, let f_i and f_j be two fragments defined according to m_i and m_j, respectively. Then p_i is *relevant* if and only if

$$\frac{\text{acc}(m_i)}{car(f_i)} \neq \frac{\text{acc}(m_j)}{car(f_j)}$$

Example 3.10. The set P_r defined in Example 3.9 is complete and minimal. If, however, we were to add the predicate

PNAME = “Instrumentation”

to P_r, the resulting set would not be minimal since the new predicate is not relevant with respect to P_r – there is no application that would access the resulting fragments any differently.

We can now present an iterative algorithm that would generate a complete and minimal set of predicates P_r' given a set of simple predicates P_r. This algorithm, called COM_ MIN, is given in Algorithm 3.1. To avoid lengthy wording, we have adopted the following notation:

Rule 1: each fragment is accessed differently by at least one application.’

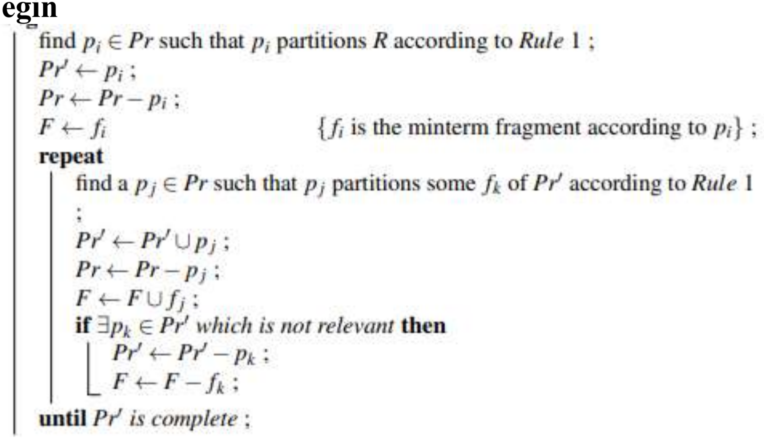
f_i of P_r': fragment fi defined according to a minterm predicate defined over the predicates of P_r'.

Algorithm 3.1: COM MIN Algorithm

Input: R: relation; P_r: set of simple predicates
Output: P_r': 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

The algorithm begins by finding a predicate that is relevant and that partitions the input relation. The repeat-until loop iteratively adds predicates to this set, ensuring minimality at each step. Therefore, at the end the set $P_r{}'$ is both minimal and complete.

The second step in the primary horizontal design process is to derive the set of minterm predicates that can be defined on the predicates in set $P_r{}'$. These minterm predicates determine the fragments that are used as candidates in the allocation step. Determination of individual minterm predicates is trivial; the difficulty is that the set of minterm predicates may be quite large (in fact, exponential on the number of simple predicates). We look at ways of reducing the number of minterm predicates that need to be considered in fragmentation.

This reduction can be achieved by eliminating some of the minterm fragments that may be meaningless. This elimination is performed by identifying those minterms that might be contradictory to a set of implications I. For example, if Pr0 = {p1, p2}, where

$p1 : att = value\_1$

$p2 : att = value\_2$

and the domain of att is {value_1, value_2}, it is obvious that *I* contains two implications:

$i1 : (att = value\ 1) \Rightarrow \neg(att = value\ 2)$

$i2 : \neg(att = value1) \Rightarrow (att = value\ 2)$

The following four minterm predicates are defined according to $P_r{}'$:

$m1 : (att = value\ 1) \wedge (att = value\ 2)$

$m2 : (att = value\ 1) \wedge \neg(att = value\ 2)$

$m3 : \neg(att = value\ 1) \wedge (att = value\ 2)$

$m4 : \neg(att = value\ 1) \wedge \neg(att = value\ 2)$

In this case the minterm predicates m1 and m4 are contradictory to the implications I and can therefore be eliminated from M.

The algorithm for primary horizontal fragmentation is given in Algorithm 3.2. The input to the algorithm PHORIZONTAL is a relation *R* that is subject to primary horizontal fragmentation, and P_r, which is the set of simple predicates that have been determined according to applications defined on relation *R*.

Example 3.11. We now consider the design of the database scheme given in Figure 3.7. The first thing to note is that there are two relations that are the subject of primary horizontal fragmentation: PAY and PROJ.

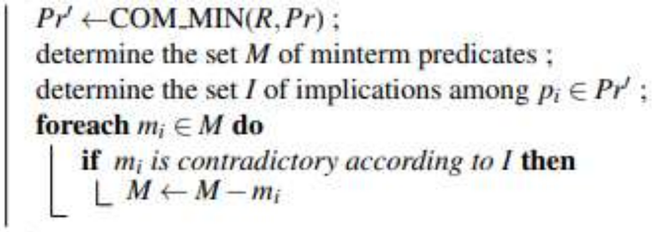
Suppose that there is only one application that accesses PAY, which checks the salary information and determines a raise accordingly. Assume that employee records are managed in two places, one handling the records of those with salaries less than or equal to \$30,000, and the other handling the records of those who earn more than \$30,000. Therefore, the query is issued at two sites.

Algorithm 3.2: PHORIZONTAL Algorithm

Input: R: relation; P_r: 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

The simple predicates that would be used to partition relation PAY are

$p1: SAL \leq 30000$

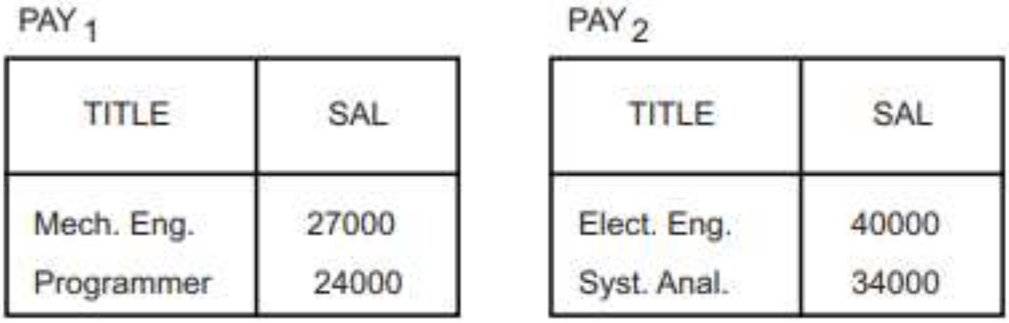
$p2: SAL > 30000$

thus giving the initial set of simple predicates $P_r = \{p1, p2\}$. Applying the COM_MIN algorithm with i = 1 as initial value results in $P_r{}' = \{p1\}$. This is complete and minimal since p_2 would not partition f_1 (which is the minterm fragment formed with respect to p_1) according to Rule 1. We can form the following minterm predicates as members of M:

$m_2: (SAL < 30000)$

$m_2: \neg(SAL \leq 30000) = SAL > 30000$

Therefore, we define two fragments $F_S = \{S1, S2\}$ according to M (Figure 3.9).



PAY_1

TITLE	SAL
Mech. Eng.	27000
Programmer	24000

PAY_2

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

Fig. 3.9 Horizontal Fragmentation of Relation PAY

Let us next consider relation PROJ. Assume that there are two applications. The first is issued at three sites and finds the names and budgets of projects given their location. In SQL notation, the query is

```
SELECT PNAME, BUDGET
FROM PROJ
WHERE LOC=Value
```

For this application, the simple predicates that would be used are the following:

$p_1: LOC\ =\ \text{“}Montreal\text{”}$
$p_2: LOC\ =\ \text{“}New\ York\text{”}$
$p_3: LOC\ =\ \text{“}Paris\text{”}$

The second application is issued at two sites and has to do with the management of the projects. Those projects that have a budget of less than or equal to $200,000 are managed at one site, whereas those with larger budgets are managed at a second site. Thus, the simple predicates that should be used to fragment according to the second application are

$p_4: BUDGET\ \leq\ 200000$
$p_5: BUDGET\ >\ 200000$

If the COM MIN algorithm is followed, the set $P_r{}' = \{p_1, p_2, p_4\}$ is obviously complete and minimal. Actually COM_MIN would add any two of p_1, p_2, p_3 to $P_r{}'$; in this example we have selected to include p_1, p_2.
Based on $P_r{}'$, the following six minterm predicates that form M can be defined:

m_1: (LOC = “Montreal”) $\wedge$ (BUDGET $\leq$ 200000)
m_2: (LOC = “Montreal”) $\wedge$ (BUDGET $>$ 200000)
m_3: (LOC = “New York”) $\wedge$ (BUDGET $\leq$ 200000)
m_4: (LOC = “New York”) $\wedge$ (BUDGET $>$ 200000)
m_5: (LOC = “Paris”) $\wedge$ (BUDGET $\leq$ 200000)
m_6: (LOC = “Paris”) $\wedge$ (BUDGET $>$ 200000)

As noted in Example 3.6, these are not the only minterm predicates that can be generated. It is, for example, possible to specify predicates of the form

$$p1\ \wedge\ p2\ \wedge\ p3\ \wedge\ p4\ \wedge\ p5$$

However, the obvious implications

$i_1: p_1 \Rightarrow \neg p_2 \wedge \neg p_3$

$i_2: p_2 \Rightarrow \neg p_1 \wedge \neg p_3$

$i_3: p_3 \Rightarrow \neg p_1 \wedge \neg p_2$

$i_4: p_4 \Rightarrow \neg p_5$

$i_5: p_5 \Rightarrow \neg p_4$

$i_6: \neg p_4 \Rightarrow p_5$

$i_7: \neg p_5 \Rightarrow p_4$

eliminate these minterm predicates and we are left with m_1 to m_6.
Looking at the database instance in Figure 3.3, one may be tempted to claim that the following implications hold:

i_8: LOC = “Montreal” $\Rightarrow \neg$ (BUDGET $>$ 200000)
i_9: LOC = “Paris” $\Rightarrow \neg$ (BUDGET $\leq$ 200000)
i_{10}: $\neg$ (LOC = “Montreal”) $\Rightarrow$ BUDGET $\leq$ 200000
i_{11}: $\neg$ (LOC = “Paris”) $\Rightarrow$ BUDGET $>$ 200000

However, remember that implications should be defined according to the semantics of the database, not according to the current values. There is nothing in the database semantics that suggest that the implicationsi8 through i11 hold. Some of the fragments defined according to M = $\{m_1, \ldots, m_6\}$ may be empty, but they are, nevertheless, fragments.

The result of the primary horizontal fragmentation of PROJ is to form six fragments FPROJ = {PROJ1, PROJ2, PROJ3, PROJ4, PROJ5, PROJ6} of relation PROJ according to the minterm predicates M (Figure 3.10). Since fragments PROJ2, and PROJ5 are empty, they are not depicted in Figure 3.10.

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	250000	New York

PROJ$_6$

PNO	PNAME	BUDGET	LOC
P4	Maintenance	310000	Paris

Fig. 3.10 Horizontal Partitioning of Relation PROJ