Primary horizontal fragmentation in distributed database

1. Horizontal Fragmentation:

A relation (table) is partitioned into multiple subsets horizontally using simple conditions.

Account(Acno, Balance, Branch_Name, Type).

If the permitted values for ***Branch_Name*** attribute are 'New Delhi', 'Chennai', and 'Mumbai', then the following SQL query would fragment the bunch of tuples (records) satisfying a simple condition.

```
SELECT * FROM account WHERE branch_name = 'Chennai';
```

Horizontal Fragmentation has two variants as follows;

1. Primary Horizontal Fragmentation (PHF)
2. Derived Horizontal Fragmentation (DHF)

1.1 Primary Horizontal Fragmentation (PHF)

Primary Horizontal Fragmentation is about fragmenting a single table horizontally (row wise) using a set of simple predicates (conditions).

Acno	Balance	Branch_Name
A101	5000	Mumbai
A103	10000	New Delhi
A104	2000	Chennai
A102	12000	Chennai
A110	6000	Mumbai
A115	6000	Mumbai
A120	2500	New Delhi

Figure 1: Account table

For the above table, we could define any simple predicates like, Branch_name = ‘Chennai’, Branch_name= ‘Mumbai’, Balance < 10000 etc using the above expression “Aj =Value”.

SELECT * FROM account WHERE balance < 10000;

$Account_2$

Acno	Balance	Branch Name
A101	5000	Mumbai
A104	2000	Chennai
A120	2500	New Delhi
A110	6000	Mumbai
A115	6000	Mumbai

SELECT * FROM account WHERE balance ≥ 10000;

$Account_3$

Acno	Balance	Branch Name
A103	10000	New Delhi
A102	12000	Chennai

Correctness of Fragmentation

We have chosen set of min-term predicates which would be used to horizontally fragment a relation (table) into pieces. Now, our next step is to validate the chosen fragments for their correctness. We need to verify did we miss anything? We use the following rules to ensure that we have not changed semantic information about the table which we fragment.

1. ***Completeness*** – If a relation R is fragmented into set of fragments, then a tuple (record) of R must be found in any one or more of the fragments. This rule ensures that we have not lost any records during fragmentation.
2. ***Reconstruction*** – After fragmenting a table, we must be able to reconstruct it back to its original form without any data loss through some relational operation. This rule ensures that we can construct a base table back from its fragments without losing any information. That is, we can write any queries involving the join of fragments to get the original relation back.

3. ***Disjointness*** – If a relation R is fragmented into a set of sub-tables R_1, R_2, …, R_n, a record belongs to R_1 is not found in any other sub-tables. This ensures that $R_1 \neq R_2$.

When we use an operation, say ***Union*** between $Account_2$, and $Account_3$ we will be able to get the original relation Account.

(SELECT * FROM account2) Union (SELECT * FROM account3);

The above query will get us *Account* back without loss of any information. Hence, the fragments created can be reconstructed.
Finally, if we write a query as follows, we will get a Null set as output. It ensures that the Disjointness property is satisfied.

(SELECT * FROM account2) Intersect (SELECT * FROM account3);

We get a null set as result for this query because, there is no record common in both relations $Account_2$ and $Account_3$.

Derived Horizontal Fragmentation

Consider an example, where an organization maintains the information about its customers.They store information about the customer in CUSTOMER table and the customer addresses in C_ADDRESS table as follows;

CUSTOMER(<u>CId</u>, CName, Prod_Purchased, Shop_Location)

C_ADDRESS(<u>CId</u>, C_Address)

The table CUSTOMER stores information about the customer, the product purchased from their shop, and the shop location where the product is purchased. C_Address stores information about permanent and present addresses of the customer. Here, CUSTOMER is the owner relation and C_ADDRESS is the member relation.

Table 1: CUSTOMER table

CID	CNAME	PROD_PURCHASED	SHOP_LOCATION
C001	Ram	Air Conditioner	Mumbai
C002	Guru	Television	Chennai
C010	Murugan	Television	Coimbatore
C003	Yuvraj	DVD Player	Pune
C004	Gopinath	Washing machine	Coimbatore

Table 2: C_ADDRESS table

CID	C_ADDRESS
C001	Bandra, Mumbai
C001	XYZ, Pune
C002	T.Nagar, Chennai
C002	Kovil street, Madurai
C003	ABX, Pune
C004	Gandhipuram, Ooty
C004	North street, Erode
C010	Peelamedu, Coimbatore

If the organization would go for fragmenting the relation CUSTOMER on the shop_location attribute, it needs to create 4 fragments using horizontal fragmentation technique as given in Figure 3 below.

Table 3: Horizontal fragments of Figure 1 on Shop_Location attribute

Select *from CUSTOMER where SHOP_LOCATION=”Mumbai”

$CUSTOMER_1$

CID	CNAME	PROD_PURCHASED	SHOP_LOCATION
C001	Ram	Air Conditioner	Mumbai

Select *from CUSTOMER where SHOP_LOCATION=”Chennai”

$CUSTOMER_2$

CID	CNAME	PROD_PURCHASED	SHOP_LOCATION
C002	Guru	Television	Chennai

Select *from CUSTOMER where SHOP_LOCATION=”Coimbatore**”**

$CUSTOMER_3$

CID	CNAME	PROD_PURCHASED	SHOP_LOCATION
C010	Murugan	Television	Coimbatore

C004	Gopinath	Washing machine	Coimbatore

Select *from CUSTOMER where SHOP_LOCATION=”Pune**”**

$CUSTOMER_4$

CID	**CNAME**	**PROD_PURCHASED**	**SHOP_LOCATION**
C003	Yuvraj	DVD Player	Pune

Now, it is necessary to fragment the second relation C_ADDRESS based on the fragment created on CUSTOMER relation. Because, in any other way, if we fragment the relation C_ADDRESS, then it may end in different location for different data.

Figure 3. This type of fragmentation based on owner relation is called Derived Horizontal Fragmentation. This will work for relations where an equi-join is required for joining two relations. Because, an equi-join can be represented as set of semi-joins.

The fragmentation of C_ADDRESS is done as follow as set of semi-joins as follows.

$$\mathbf{C\_ADDRESS_1 = C\_ADDRESS \ltimes CUSTOMER_1}$$

$$\mathbf{C\_ADDRESS_2 = C\_ADDRESS \ltimes CUSTOMER_2}$$

$$\mathbf{C\_ADDRESS_3 = C\_ADDRESS \ltimes CUSTOMER_3}$$

$$\mathbf{C\_ADDRESS_4 = C\_ADDRESS \ltimes CUSTOMER_4}$$

$\ltimes$ is a semi join operator

- A Semi-join returns rows from the left table(C_ADDRESS) for which there are corresponding matching rows in the right table(CUSTOMER fragments).
- Unlike regular joins which include the matching rows from both tables, a semi-join only includes columns from the left table in the result.

This will result in four fragments of C_ADDRESS where the customer address of all customers of fragment $CUSTOMER_1$ will go into $C\_ADDRESS_1$, and the customer address of all customers of fragment $CUSTOMER_2$ will go into $C\_ADDRESS_2$, and so on. The resultant fragment of C_ADDRESS will be the following.

Figure 4: Derived Horizontal fragments of Figure 2 as a member relation of the owner relation's fragments from Figure 3

$C\_ADDRESS_1$

CID	C_ADDRESS
C001	Bandra, Mumbai
C001	XYZ, Pune

$C\_ADDRESS_2$

CID	C_ADDRESS
C002	T.Nagar, Chennai

C002	Kovil street, Madurai

$C\_ADDRESS_3$

CID	C_ADDRESS
C004	Gandhipuram, Ooty
C004	North street, Erode
C010	Peelamedu, Coimbatore

$C\_ADDRESS_4$

CID	C_ADDRESS
C003	ABX, Pune

Checking for correctness

Completeness: The completeness of a derived horizontal fragmentation is more difficult than primary horizontal fragmentation. Because, the predicates used are determining the fragmentation of two relations. Formally, for fragmentation of two relations R and S, such as $\{R_1, R_2, \ldots, R_3\}$ and $\{S_1, S_2, \ldots, S_3\}$, there should be one common attribute such as A. Then, for each tuple t of R_i, there should be a tuple S_i which have a common value for A. This is known as *referential integrity*.

The derived fragmentation of C_ADDRESS is complete. Because, the value of the common attributes CID for the fragments $CUSTOMER_i$ and $C\_ADDRESS_i$ are the same. For example, the value present in CID of $CUSTOMER_1$ is also and only present in $C\_ADDRESS_1$, etc.

Reconstruction: Reconstruction of a relation from its fragments is performed by the union operator in both the primary and the derived horizontal fragmentation.

Disjointness: If the minterm predicates are mutually exclusive then the disjointness rule is satisfied for Primary Horizontal Fragmentation. For derived horizontal fragmentation, we state that the fragments are disjoint if the fragments were created using the mutually exclusive simple predicates of the base relation. Hence, in our example, as the simple predicates Shop_Location='Mumbai', etc are mutually exclusive, the derived fragments are also disjoint.

Vertical fragmentation of customer table:

CID	CNAME	PROD_PURCHASED	SHOP_LOCATION
C001	Ram	Air Conditioner	Mumbai
C002	Guru	Television	Chennai
C010	Murugan	Television	Coimbatore
C003	Yuvraj	DVD Player	Pune
C004	Gopinath	Washing machine	Coimbatore

Select CID, CNAME, SHOP_LOCATION from CUSTOMER;

Cust_vertical_1

CID	CNAME	SHOP_LOCATION

C001	Ram	Mumbai
C002	Guru	Chennai
C010	Murugan	Coimbatore
C003	Yuvraj	Pune
C004	Gopinath	Coimbatore

Select CID,PROD_PURCHASED from CUSTOMER;

Cust_vertical_2

CID	PROD_PURCHASED
C001	Air Conditioner
C002	Television
C010	Television
C003	DVD Player

C004	Washing machine

Example 2

STUDENT(RollNo, Name, Marks, Country)

ADDRESS(RollNo, Address)

RollNo	NAME	MARKS	COUNTRY
01	Fazal	22	IRAQ
02	Abdul	66	Italy
03	Sameed	77	UK
04	Shahzeb	90	China

05	Mumraiz	66	China

Table 1: STUDENT table

RollNo	Address
01	City A, IRAQ
01	City A, UK
02	City D, Italy
02	City A, Pakistan
03	City D, IRAQ
04	City D, Iraq
04	City A, Pakistan
05	City B, China

Table 2: Address Table

If the organization would go for fragmenting the relation STUDENT on the Country attribute, it needs to create 4 fragments using horizontal fragmentation as mentioned in table below;

	ROLLNO	NAME	MARKS	COUNTRY
$STUDENT_1$	C001	Fazal	22	IRAQ
$STUDENT_2$	ROLLNO	NAME	MARKS	COUNTRY
	C002	Abdul	66	Italy

	ROLLNO	NAME	MARKS	COUNTRY
$STUDENT_3$	C010	Mumraiz	66	China
	C004	Shahzeb	90	China
$STUDENT_4$	ROLLNO	NAME	MARKS	COUNTRY
	C003	Sameed	77	UK

Table 3: Horizontal fragments of Table 1 on Country attribute

Now, it is necessary to fragment the second relation ADDRESS based on the fragment created in STUDENT relation. The fragmentation of ADDRESS is done as follow as a set of semi-joins as follows.

$$\mathbf{ADDRESS_1} = \text{ADDRESS} \ltimes \text{STUDENT}_1$$

$$\mathbf{ADDRESS_2} = \text{ADDRESS} \ltimes \text{STUDENT}_2$$

$$\mathbf{ADDRESS_3} = \text{ADDRESS} \ltimes \text{STUDENT}_3$$

$$\mathbf{ADDRESS_4} = \text{ADDRESS} \ltimes \text{STUDENT}_4$$

This will result in four fragments of ADDRESS where the STUDENT address of all STUDENTs of fragment $STUDENT_1$ will go into $ADDRESS_1$, and the STUDENT address of all STUDENTs of fragment $STUDENT_2$ will go into $ADDRESS_2$, and so on.

The resultant fragment of ADDRESS will be the following;

RollNo	Address
01	City A, IRAQ
01	City A, UK

Table 4: Showing fragment 1 of “address” table

RollNo	Address

02	City D, Italy
02	City A, Pakistan

Table 5: Showing fragment 2 of “address” table

RollNo	Address
04	City D, Iraq
04	City A, Pakistan
05	City B, China

Table 6: Showing fragment 3 of “address” table

RollNo	Address
03	City D, IRAQ

Table 7: Showing fragment 4 of “address” table

Checking the fragments for correctness in Derived Horizontal Fragmentation

Completeness: The completeness of a derived horizontal fragmentation is complex than primary horizontal fragmentation. The reason for this complexity is because the predicates used are determining the fragmentation of two different tables/relations. Formally, for fragmentation of two relations R and T, such as {R_1, R_2, …, R_n} and {T_1, T_2, … , Tn}, there should be one common attribute such as A. Then, for each tuple of relation R_i, there should be a tuple T_i which has a common value for A. This concept is called referential integrity concept.

The derived fragmentation of Address is complete. Because the value of the common attributes RollNo for the fragments $STUDENT_i$ and $Address_i$ are the same. For example, the value present in RollNo of $STUDENT_1$ is also and only present in $Address_1$, etc.

Reconstruction: Reconstruction of the pre-existing tables is possible by a union operation.

Disjointness: If the minterm predicates are mutually exclusive then the disjointness rule is satisfied for Primary Horizontal fragmentation.