Distributed Database Design

Topics:

- What is fragmentation?
- Types of data Fragmentation
- Horizontal data fragmentation
  - Primary Horizontal Fragmentation (PHF)
  - Derived Horizontal Fragmentation (DHF)
- Reconstruction of Horizontal fragmentation
- Correctness rules of fragmentation
- Derived horizontal fragmentation
- Minterm predicate
- Vertical Fragmentation
  - Grouping(Bottom-Up approach)
  - Splitting(Top -Down Approach)
- Reconstruction of vertical fragmentation
- Hybrid/Mixed fragmentation
- Reconstruction of hybrid fragmentation

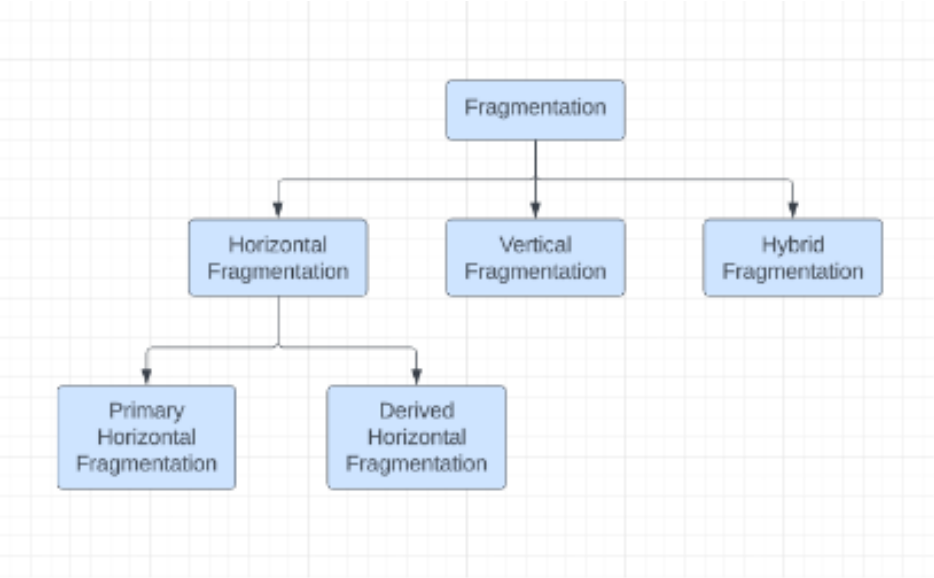
What is fragmentation?

- The process of dividing the database into a smaller multiple parts is called fragmentation.
- These fragments may be stored at different locations/sites.
- The data fragmentation process should be carried out in such a way that the reconstruction of original database from the fragments is possible.

Types of data Fragmentation

There are three types of data fragmentation:

1. Horizontal data fragmentation
2. Vertical Fragmentation
3. Hybrid/Mixed fragmentation



❖ Horizontal Data Fragmentation:

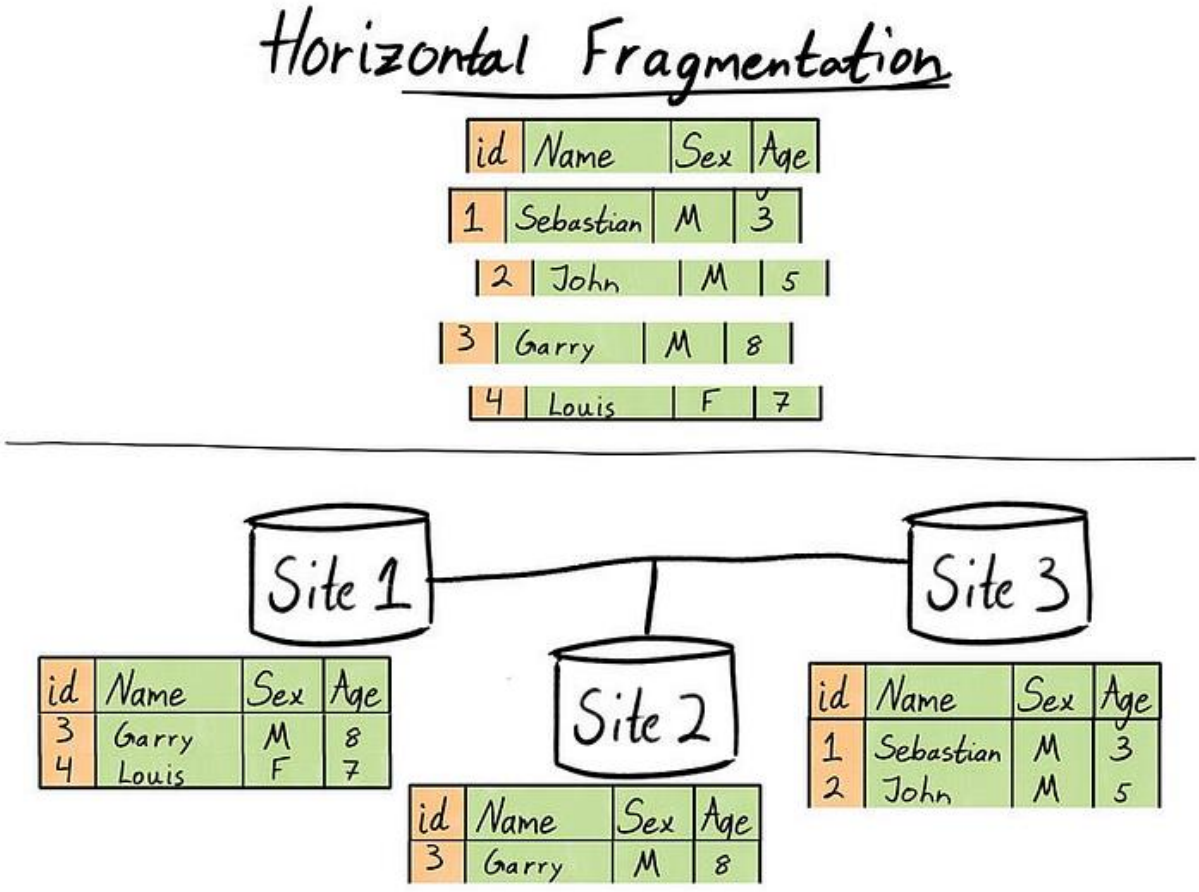
- Horizontal fragmentation divides a relation (table) horizontally into the group of rows to create subsets of tables.

- In horizontal fragmentation, table is partitioned with respect to tuples. It breaks relation 'R' by assigning each tuple of R to one or more fragments

- Every fragment has all the columns. Each fragment is a subset of tuples in original relation 'R'

- All fragments are stored at different sites & each fragment has distinct tuples

- Distribute the pieces to the sites of your DDBMS however you like and reassemble them.



- **Primary Horizontal Fragmentation (PHP)**

Let's assume the following example:

Id	Name	City	Salary
6	Sam	Pune	3500
5	Punit	Pune	4000
7	Kunal	Pune	6900
8	Sonam	Delhi	6900
2	Naitik	Mumbai	6500
3	Sneha	Chennai	6600
4	Pratiksha	Delhi	6200

Employee

Now we will create fragments based on city

Fragment 1: Employee whose city is 'Pune'

Query:

```
create table fragment1 as
select * from  employee
where city = 'Pune'
```

Id	Name	City	Salary
6	Sam	Pune	3500
5	Punit	Pune	4000
7	Kunal	Pune	6900

fragment1

Fragment 2: Employee whose city is 'Mumbai'

Query:

```
create table fragment2 as
select * from  employee
where city = 'Mumbai'
```

Id	Name	City	Salary
2	Naitik	Mumbai	6500

fragment2

Fragment 3: Employee whose city is 'Delhi'

Query:

```
create table fragment3 as
select * from  employee
where city = 'Delhi'
```

Id	Name	City	Salary
8	Sonam	Delhi	6900
4	Pratiksha	Delhi	6200

fragment3

Fragment 4: Employee whose city is 'Chennai'

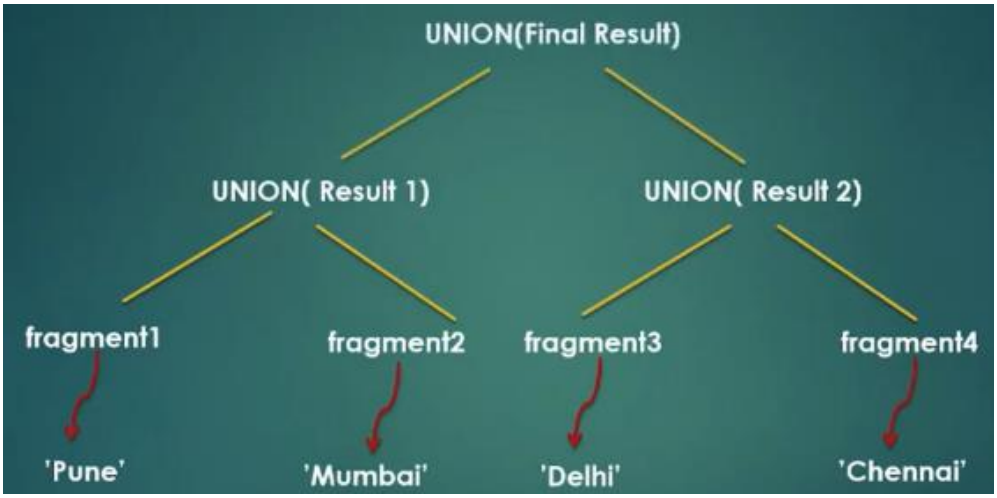
Query:

```
create table fragment4 as
select * from  employee
where city = 'Chennai'
```

Id	Name	City	Salary
3	Sneha	Chennai	6600

fragment4

Reconstruction:



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

What is simple predicate?

Given a table R with set of attributes [A1, A2, ... , An], a simple predicate Pi can be expressed as follows:

$$Pi: Ai\ \theta\ Value$$

Where θ can be any of the symbols in the set $\{=, <, >, \leq, \geq, \neq\}$

Examples:

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

Example 1: for the table Account, if simple conditions are:

Balance < 10000, Balance ≥ 10000, then,

Set of simple predicates P1 = {Balance < 10000, Balance ≥ 10000}

Example 2: if simple conditions are:

Branch_name = 'Chennai', Branch_name= 'Mumbai', Balance < 10000, Balance ≥ 10000, then,

Set of simple predicates P2 = {Branch_name ='Chennai', Branch_name = 'Mumbai', Balance < 10000, Balance ≥ 10000}

Min-term Predicate:
Min-term predicate can be written as follows;

Min-term predicate, Mi = P1 ∧ P2 ∧ P3 ∧ ... ∧ Pn

- Here, P1 means both P1 or ¬ (P1), P2 means both P2 or ¬ (P2), and so on. Using the conjunctive form of various simple predicates in different combination, we can derive many such min-term predicates.

- **For Example 1**: for the table Account, if simple conditions are, Balance < 10000, Balance ≥ 10000, then,
  - We will get 2^n min-term predicates, where n is the number of simple predicates in the given predicate set. For P1, we have 2 simple predicates. Hence, we will get 4 (2^2) possible combinations of min-term predicates as follows:

  ✓ m1 = {Balance < 10000 ∧ Balance ≥ 10000}
  ✓ m2 = {Balance < 10000 ∧ ¬ (Balance ≥ 10000)}
  ✓ m3 = {¬ (Balance < 10000) ∧ Balance ≥ 10000}
  ✓ m4 = {¬ (Balance < 10000) ∧ ¬ (Balance ≥ 10000)}

Our next step is to choose the min-term predicates which can satisfy certain conditions to fragment a table, and eliminate the others which are not useful.

- Account1 = $\sigma_{\text{Balance < 10000 ∧ Balance ≥ 10000}}$ (Account)
  SQL:
  SELECT * FROM account
  WHERE balance < 10000 AND balance ≥ 10000

- Account2 = $\sigma_{\text{Balance < 10000 ∧ ¬ (Balance ≥ 10000)}}$ (Account)
  SQL:
  SELECT * FROM account
  WHERE Balance < 10000 ∧ NOT (Balance ≥ 10000)

- Account3 = $\sigma_{\text{¬ (Balance < 10000) ∧ Balance ≥ 10000}}$ (Account)
  SQL:
  SELECT * FROM account
  WHERE NOT Balance < 10000 AND Balance ≥ 10000

- Account4 = $\sigma_{\text{¬ (Balance < 10000) ∧ ¬ (Balance ≥ 10000)}}$ (Account)
  SQL:
  SELECT * FROM account
  WHERE NOT balance < 10000 AND NOT balance ≥ 10000

- The first query for fragment Account1 (min-term predicate m1) is invalid as any record in a table cannot have two values for any attribute in one record. That is, the condition (Balance < 10000 ∧ Balance ≥ 10000) requires that the value for balance must both be less than 10000 and greater and equal to 10000, which is not possible. Hence the condition violates and can be **eliminated**.

- For fragment Account2 (min-term predicate m2), the condition is {Balance < 10000 ∧ ¬ (Balance ≥ 10000)} which ultimately means balance<10000 which is correct.

- Likewise, fragment Account3 is valid and Account4 must be eliminated.

Finally, fragments are:

Show the fragments

Correctness Rules for Fragmentation:

While we perform the fragmentation process, as a result we expect the following as outcomes:

- We should not lose data because of fragmentation
- We should not get redundant data because of fragmentation

Hence, to ensure these properties we need to verify that whether we performed the fragmentation correctly or not. For this verification we use the correctness rules.

■ **Completeness**:

- To ensure that there is no loss of data due to fragmentation.
- Completeness property ensures this by checking whether all the records which were part of a table (before fragmentation) are found in at least one of the fragments after fragmentation.

■ **Reconstruction**:

- This rule ensures the ability to re-construct the original table from the fragments that are created.
- This rule is to check whether the functional dependencies are preserved or not.
- If a table R is partitioned into fragments R1, R2, ..., Rn, then Reconstruction insists the following:

$$R = R1 \cup R2 \cup ... \cup Rn$$

■ **Disjointness**:

- This rule ensures that no record will become a part of two or more different fragments during the fragmentation process.
- If a table R is partitioned into fragments R1, R2, ..., Rn, then Disjointness insists the following:

$$R1 \cap R2 \cap ... \cap Rn = \emptyset$$

Derived Horizontal Fragmentation

- In derived horizontal fragmentation a relation R1 may get fragmented by attribute of some other relation R2
- Relation in a derived horizontal fragmentation are classified as :
    1. Owner relation (has a primary key)
    2. Member relation (uses the owner's primary key as foreign key)
        - Owner relation R1(emp-id,emp_name,emp_sal)
        - Member relation R2(proj_id,emp_id,role)

Example:

Dno	Dname	Budget	Loc
D1	Mgmt	750,000	Mumbai
D2	Payroll	500,000	New Delhi
D3	Production	400,000	New Delhi
D4	Maintainence	300,000	Lucknow

Department (Owner Table)

Pno	Pname	Budget	Dno
P1	DB design	135,000	D2
P2	Maintainence	310,000	D3
P3	CAD/CAM	500,000	D2
P4	Architecture	300,000	D1
P5	Documentation	450,000	D1

Project (Member Table)

Step 1: Create fragments on owner relation

Fragment 1:

```
create table HF1 as
select * from  department
where loc = 'Mumbai'
```

Dno	Dname	Budget	Loc
D1	Mgmt	750,000	Mumbai

Fragment 2:

```
create table HF2 as
select * from  department
where loc = 'NewDelhi'
```

Dno	Dname	Budget	Loc
D2	Payroll	500,000	New Delhi
D3	Production	400,000	New Delhi

Fragment 3:

```
create table HF3 as
select * from  department
where loc = 'Lucknow'
```

Step 2: Perform semi-join operation on Member relation and fragments:

Project ⋉ HF1
Project ⋉ HF2
Project ⋉ HF3

Output 1:

create table project1 as
select Pno, Pname, Budget, Project.Dno from project, HF1
where Project.Dno = HF1.Dno

Pno	Pname	Budget	Dno
P4	Architecture	300,000	D1
P5	Documentation	450,000	D1

Output 2:

create table project2 as
select Pno, Pname, Budget, Project.Dno from project, HF2
where Project.Dno = HF2.Dno

Output 2:

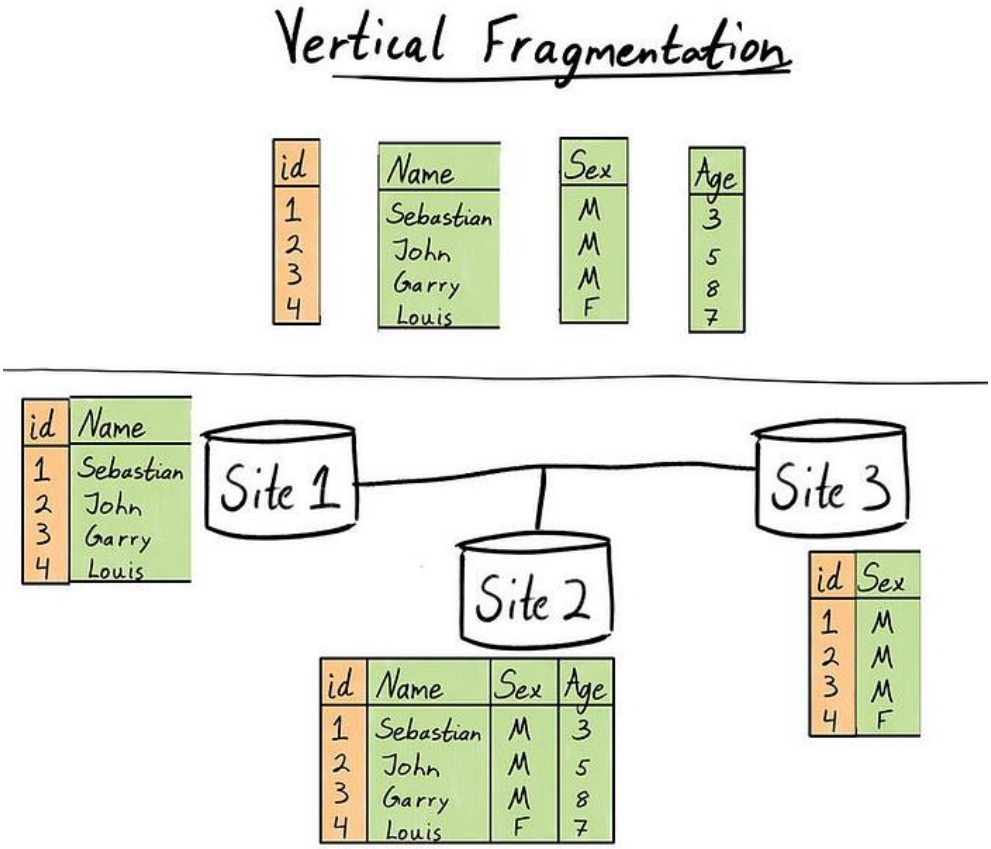
create table project3 as
select Pno, Pname, Budget, Project.Dno from project, HF3
where Project.Dno = HF3.Dno

Vertical Fragmentation:

- Vertical fragmentation divides a relation (table) vertically into groups of columns to create subsets of tables.

- In vertical fragmentation the fragments are created by dividing the attributes

- In order to maintain re-constructiveness, each fragment should contain the primary key field(s) of the table



There are two approaches:

1. Grouping(Bottom-Up approach):
    - Starts by assigning each attribute to one fragment, and at each step, joins some of the fragments until some criteria is satisfied.
    - Attributes are progressively aggregated to constitute fragments.
2. Splitting(Top -Down Approach):
    - Starts with a relation and decides on beneficial partitioning based on the access behavior of applications to the attributes.
    - Global relations are progressively split into fragments.

Example:

Eid	Ename	Salary	Bonus	Project	Experience
1	ABC	50,000	5000	P_1	5
2	DEF	40,000	7000	P_2	3
3	GHI	45,000	8000	P_3	2

Fragment 1: Basic Employee Details(Eid, Ename, Salary)

Eid	Ename	Salary
1	ABC	50,000
2	DEF	40,000
3	GHI	45,000

Fragment 1

Fragment 1: Bonus on Projects(Eid, Project, Bonus)

Eid	Bonus	Project
1	5000	P_1
2	7000	P_2
3	8000	P_3

Fragment 2

Reconstruction:
Reconstruction of vertical fragmentation is performed by using **Full Outer Join** operation on fragments.

```
select * from
fragment 1
NATURAL JOIN
fragment 2
```

Hybrid/Mixed fragmentation:

- In hybrid fragmentation, a combination of horizontal and vertical fragmentation techniques are used.

- Mixed fragmentation is group of rows & columns in relation

- However, reconstruction of the original table is often an expensive task. Original relation is obtained by combination of join and union operations.

- Hybrid fragmentation can be done in two alternative ways:
  - At first, generate a set of horizontal fragments; then generate vertical fragments from one or more of the horizontal fragments.
  - At first, generate a set of vertical fragments; then generate horizontal fragments from one or more of the vertical fragments.

- Fragmentation is defined using the selection and projection operations of relational algebra

Project

Proj-name	Pno	Location	Dept-no
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computing	10	Stafford	4
Web sales	20	Houston	1
Benefits	30	Stafford	4

Pno	Proj-name
1	ProductX
2	ProductY
3	ProductZ

Site 1

Pno	Location	Dept-no
1	Bellaire	5
2	Sugarland	5
3	Houston	5

Site 2

Proj-name	Pno	Location	Dept-no
Web sales	20	Houston	1

Site 3

Proj-name	Pno	Location
Computing	10	Stafford
Benefits	30	Stafford

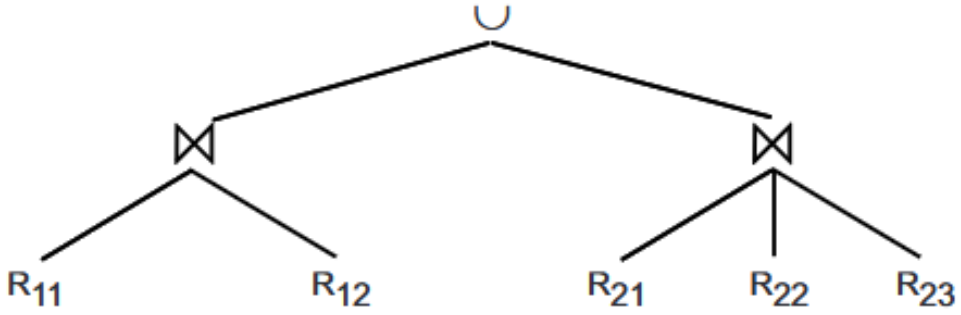
Site 4

Pno	Dept-no
10	4
30	4

Site 5

Reconstruction of Hybrid Fragmentation:

The original relation in hybrid fragmentation is reconstructed by performing UNION and FULL OUTER JOIN.



Question: *Discuss advantages and disadvantages of various fragmentations*