1.1 Destributed database: Destributed database is a collection of multiple, logically interrelated databases located at the nodes of a distributed system.

Distributed DBMS is then defined as the software system that permits the management of the distributed database and makes the distribution transparent to the users.

1.3 Data Delivery Alternatives

Three dimentions

① Delivery modes

↳ Pull-only (client requesting data from server) — by posing the query

↳ Push-only (server sends data without request)

↳ Hybrid (the transfer of data first initiated by a pull then subsequent transfer of updated data is initiated by push)

② Frequency

↳ Periodic / regular

(hybrid + Push-only) ↳ Conditional

(Pull-based) ↳ Ad-hoc or irregular or random

③ Communication Methods

↳ Unicast

↳ Multicast } minicast

↳ Broad-cast }

Not possible

↳ Pull + conditions

↳ Push + ad-hoc

↳ Pull broadcast

↳ Pull - multicast

1.4 Promises of Distributed DBMS

1. Transparency (Transparent management of distributed, fragmented, replicated data)

Transparency means hiding the complex details of how a system works from user. This makes it easier to develop and use complex details of how applications. Transparency helps by making it seem like all the data is in one place, even if it's spread across multiple locations. There are four types of transparencies. ① Data independence

② Network transparency / Location / distribution transparency

③ Fragmentation Transparency

④ Replication transparency

① Data independence: It is the idea that user applications shouldn't be affected by changes in how data is defined or organized. There are two types

1. Logical Data independence: Changes to the logical structure don't affect user applications.

2. Physical Data independence: Changes to the storage details don't affect user applications.

② Network Transparency: It ensures that users are unaware of the existence of network connecting different sites. The goal is to make data applications that run on a distributed database feel the same, as those on a centralized database. This abstraction

P-T.O.

allows users to interact with the database. without considering whether it is centralized or distributed.

i) Location Transparency: This command used to perform a task is independent of both the location of the data and the system on which an operation is carried out.

ii) Naming transparency: A unique name is provided for each object in the database. In the absence of naming transparency, users are required to embed the location name as part of the object name.

(3.) Fragmentation Transparency: It means that users are unaware of how a database relation is divided into smaller fragments. The system manages the details of fragmentation, enabling users to interact with the database as if it were a single, unified structure. This is commonly done for reasons of performance, availability, and reliability.

(4) Replication Transparency: For performance, reliability and availability reasons, it is ~~us~~ usually desirable to be able to distribute data & in a replicated fashion across the machines on a network. There are 3 types of replication transparency. i) Full

ii) Partial

iii) Partition.

2. Reliability Through Distributed Transactions:

Distributed DBMS are designed to improve reliability by replicating data across multiple sites. This means that the failure of a single site or a communication link does not bring down the entire system. It means user can access other parts of the database with proper care.

3. Improve Performance: (i) Each site handles only a portion of the database, concention for CPU and I/O services is not as severe as for centralized databases. (ii) Locality reduces remote access delays that are usually involved in wide area networks.

(iii) Parallelism: it is achived when there is no dependency between the operators

that are executed in parallel. Two types of parallelism ① inter parallelism
② intra parallelism

4. Scalability : Handlling larger databases and heavier workloads is easier by adding more processing and storage power across the network. This scalability, known as "scale-out" or horizontal skaling, is a major benefit of distributed DBMS's in cluster and cloud computing setups. Scale-out involves adding more servers in a flaxible manner to increase system capacity.

1.5 Design Issues - The challenges that are faced by a DDB.

1. Distributed Database Design :→ The questions that is being addressed is how the data is placed across the sites.

(i) There are two basic ~~types~~ alternatives to placing data: (I) Replicated (II) Non-replicated

(ii) A related problem is the design and management of system directory.

(iii) The two fundamental design issues are

⇒ Fragmentation: The separation of the database into partitions

⇒ Distribution: The optimum distribution of fragments.

2. Distributed Data control: An important requirement of a DBMS is to maintain data consistency by controlling how data is accessed. This is called data control and involves view management, access control and integrity enforcement

3. Destributed Query Processing: Query processing deals with designing algorithms that analyze queries and convert them into a series of data ~~manu~~ manipulation operations.

4. Distributed concurrency control: It involves the synchronization of accesses to the distributed database, such that the integrity of the database is maintained.

5. Reliability of Distributed DBMS: Improving reliability and availability in distributed systems requires mechanisms to ensure database consistency, detect failures and recover from them. Operational sites must ~~b~~ remain consistent and up-to-date during failures.

6. Replication: In distributed database with partial or full replication, protocol must ensure replica consistency, meaning all copies of the same data item have the same value.

7. Parallel DBMS: Distributed and parallel databases are closely related. While distributed databases assume each sites is a single logical computer, many are parallel clusters. The differences are single-site distribution and geo-distribution. Parallel DBMS aim for high scalability and performance.

8. <u>Database Intregation</u>: The shift towards loosely federated, heterogeneous data sources has led to multidatabase systems, integrating distributed databases for easier access. This bottom-up approach is vital in today's distributed environments.

9. <u>Alternative Distributed Approaches</u>: The ~~Int~~ internet's growth has raised questions about distributed database systems, focusing on peer-to-peer computing and world wide web. Both improve data sharing but pose different data management challenges.

10. Big Data Processing and NoSQL: The last decade's "big data" ~~boo~~ explosion marked by high volumn, velocity and veracity, has led to new data management systems. These includes scale-out computing platforms and NoSQL systems, addressing the limitations of traditions of traditional relational DBMSs.

Architecture

1. Client - server systems
2. Multidatabase systems
3. Cloud computing

1. Client - server system : It offers a structured way to handle complex modern systems and distributed environments. The key idea is to divide the workload between server, which manages data, and the client, which handles user interaction and applications. The Server is responsible for core operations like query processing, transaction management and storage management, ~~and~~ while client handles applications, user interface, and possibly cached data and transaction looks.

The client sends SQL queries to the server, ~~and~~ which processes and optimizes them before sending results back.

Types of client - Server:

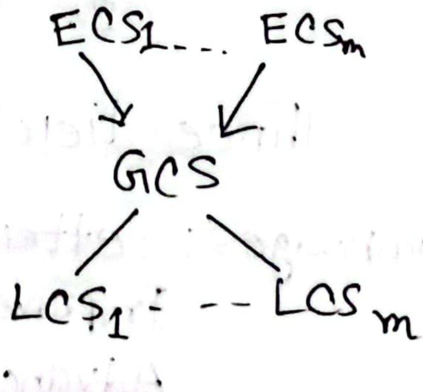
1. Multiple client Single server
2. Multiple client multiple server

Three tier Architecture → 1. Client
2. Application server
3. Database server

Advantages: Better Data Management
Improve performance
Advanced hardware

2. Multidatabase System - is a system where different database are independent and don't necessarily interact or even know each other. — Database integration

Multidatabase	Distributed
1. It is a type of DBMS that allows an application to use multiple types of databases simultaneously.	1. It is a type of database management system that stores data across multiple computers or sites that are connected by a network.
2. Bottom-up approach.	2. Top-down approach.
3. Heterogenious	3. Homogenious
4. Non-cooperative	4. Co-operative and known to each other.
5. Integration	5. Fragmentation.
6. Service from local and Global conceptual Schema.	6. Service from only Global.



The mediator/wrapper approach is a common way to build a multidatabase system by connecting different databases.

Mediator: It is a software component that collects data from various databases and presents it to users in a useful way.

Wrapper: The wrapper acts as a translator between the mediator and the database.

Together, they form a middleware layer

Cloud computing: It refers to the delivery of computing services over internet, including storage, database, software and analyties.

It provides various levels:

1. Infrastructure-as-a-service (Iaas): Provides computing infrastructure (servers, storage, networking)
2. Platform-as-a-service (Paas): Offers a Platform with development tools and APIs

3. Software-as-a-Service (SaaS): Delivers application software.

4. Database-as-a-Service (DaaS): offers database management as a service.

Advantages:

1. Cost-effective: Users pay only for resources they use.
2. Ease of access: Services are available anywhere via the internet.
3. Quality of service: Specialized providers ensure high performance and reliability
4. Elasticity: Easily scale resources up or down as needed.

Disadvantage:

1. Provider lock-in : It can be hard to switch cloud providers due to special software or high costs for moving data.

2. Less of control: You may lose control over important things like system updates ~~and~~ or downtime.

3. Security concerns: Cloud data can be exposed to hackers. While there are security options, they can be tricky to set up.

4. Unexpected costs: Modifying apps to work on the cloud can lead to extra costs.

Chap-2

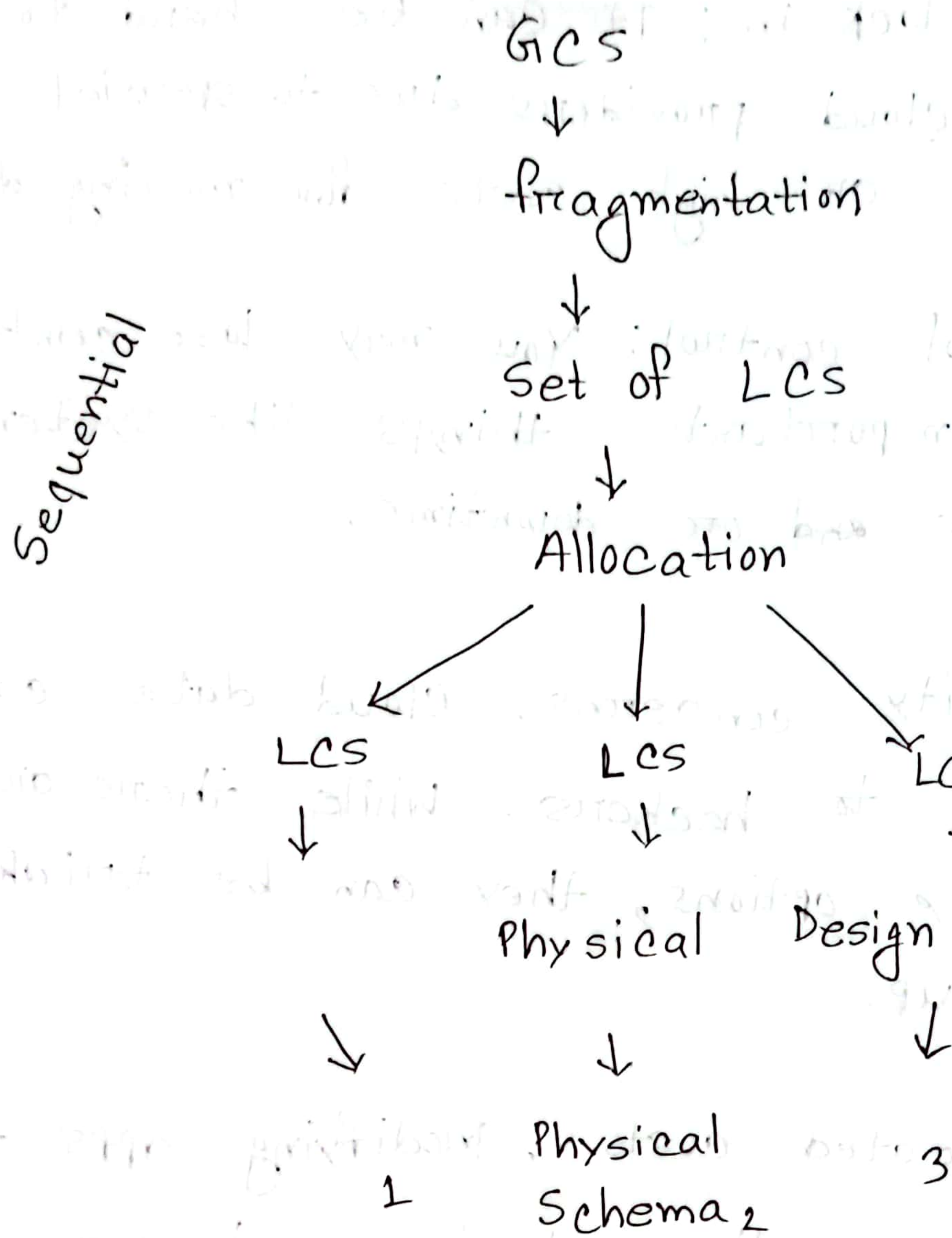

GCS
Fragmentation
Set of LCS
Allocation
Sequential
LCS
LCS
LCS
Physical Design
Physical Schema
1
2
3

Fig-2.2

EMP

EMPNO	EName	Title

ASG

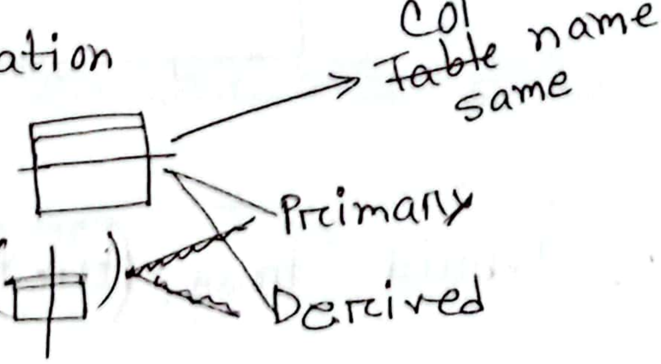
ENO	PNo	RESP	DUR

PNO	PName	Budget	LOC

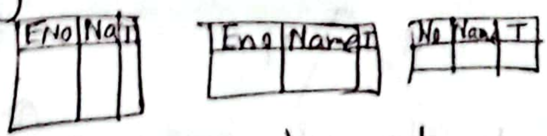
Pay

Title	Sal

3 types of fragmentation



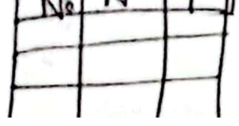
1. Horizontal frag
2. vertical frag
3. Hybrid frag



1. Horizontal frag : reconstract = Union

$Emp_1 \cup Emp_2 \cup Emp_3$

Structure same থাকে



H. fragmentation based on given condition.

Condition – Example : fragment based on Title.

2. Vertical frag: reconstruct = Join

$Proj = Proj_1 \bowtie Proj_2$

frag

PN	PName

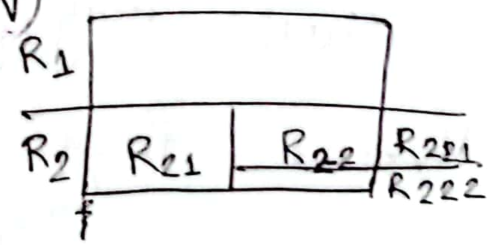
$\bowtie$

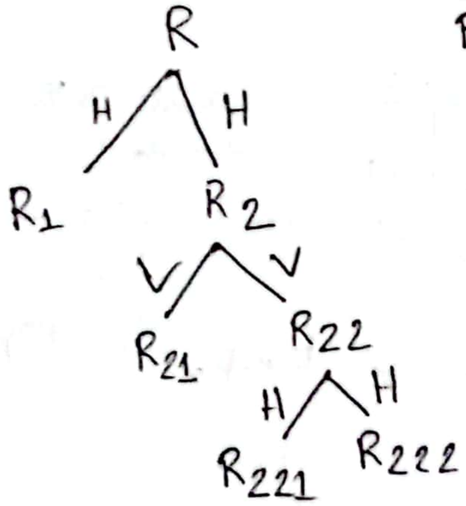
PN	Bud	LOC

reconstruct

PN	PName	Bud	LOC
1		20	
2		25	
3		30	

3. Hybrid frag: (H & V)





reconstruct ⇒

$R_{221} \cup R_{222} = R_{22}$

$R_{21} \bowtie R_{22} = R_2$

$R_1 \cup R_2 = R$

* Properties of fragmentation:

1. Completeness / Lossless Decomposition

2. Reconstruction:

$$R = \nabla R_n$$

3. Disjointness: Uncommon in Horizontal but not in vertical (Primary key same)

Horizontal

Primary Horizontal: Condition in same table. Ex: based on Title, frag-EMP

Derived Horizontal: condition is not in the same table. (condition in source table)

Ex: based on sal, frag- EMP

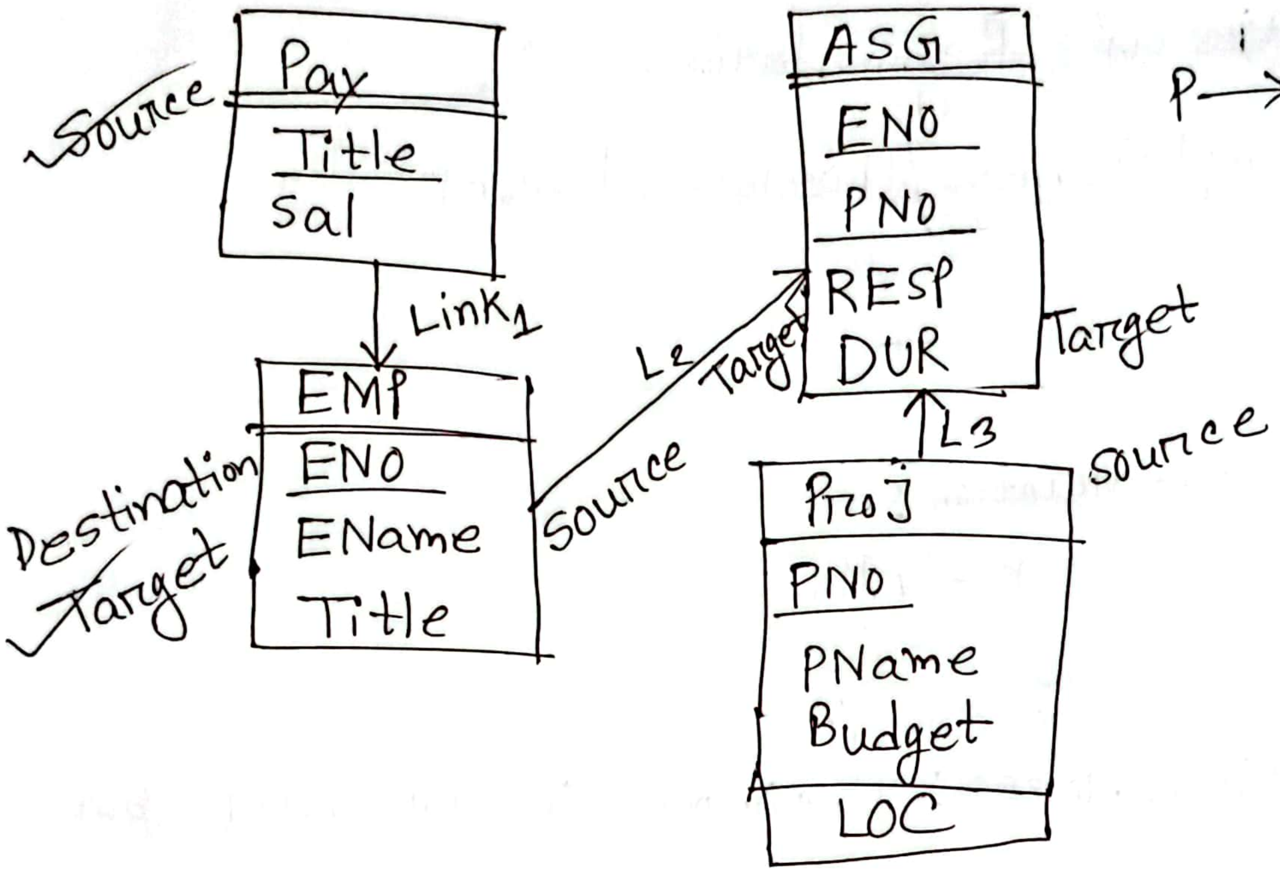


fig: Join graph (2.5)

~~Othr~~ Only Primary Horizontal ⇒ Pay, PROJ

because no source table

Q: Type

6 - sigma - condition

$P_1 : 6_{Sal > 5000} (Pay)$

condition Table

Condition → Multiple ~~cond~~ combination

~~১~~ একটা condition থাকলে ২টা ।।

2^n

Minterm - All possible combination of Predicates

P_1 এর জন্য $2^1 = 2$

P_1

$\neg P_1$

Q:

$P_1 : \sigma_{sal > 5000}$ (Pay)

$P_2 : \sigma_{title}$ = "Prog" (Pay)

Pair combination

$P_3 : \sigma_{title}$ = "Me ~~Prog~~" (Pay)

$2^3 = 8$ combination

Same predicates এর মধ্যে pair বানাতে ~~হবে~~। পারবো না।

$P = \{P_1, P_2, P_3\}$

same title

So (P_1, P_2) (P_1, P_3)

M1: $P_1 \wedge P_2 \rightarrow Sal > 5000$ and prog

M2: $P_1 \wedge \neg P_2 \rightarrow$ " and not prog

M3: $\neg P_1 \wedge P_2$

M4: $\neg P_1 \wedge \neg P_2$

└→ and

M5: $P_1 \wedge P_3$ — $Sal > 5000$ and Me. Eng

M6: $\neg P_1 \wedge P_3$

M7: $P_1 \wedge \neg P_3$

M8: $\neg P_1 \wedge \neg P_3$.

যেগুলোতে মান নাই সেগুলো Empty.

Algorithm: 2.2 PHorizontal

input : R: relation, P_r = set of simple Predicates

Relation = R, Simple Predicates = P_r

Note: Com-Min algorithm → Condition test
↳ Complete - Minimal

$P_r' \leftarrow$ Com-Min (R, P_r)

Next step: Minterm (All possible)

Next step: Simplification (Conflict, বাদ confusion)

Next step: Final output

Completeness = যদি, < থাকে তবে => বসাবো
" = " তবে বাকিসব গুলো উল্লেখ করতে হবে

Minimal = বাদ দিয়ে দিবো

Example: R : Proj

Q

$Pr = \{P_1, P_2, P_3\}$

$P_1 = \sigma_{Budget > 20,0000}$

$P_2 = \sigma_{Loc = \text{"Montreal"}}$

$P_3 = \sigma_{Loc = \text{"Paris"}}$

Sol:

$P_4 = \sigma_{Budget \leq 200000}$ → Less than greater than

$P_5 = \sigma_{Loc = \text{"New york"}}$

$Pr' = \{P_1, P_2, P_3, P_4, P_5\}$ $2^5 = 32$

P_1 and P_4 same so P_1, P_4 not Possible

P_2, P_3, P_5 are same so not possible

Pairs:

$M1: P_1 \wedge P_2$	$M5. P_1 \wedge P_3$	$M9. P_1 \wedge P_5$
$\times M2: \neg P_1 \wedge P_2$	$M6. \neg P_1 \wedge P_3$	$M10. \neg P_1 \wedge P_5$
$\times M3: P_1 \wedge \neg P_2$	$M7. P_1 \wedge \neg P_3$	$M11. P_1 \wedge \neg P_5$
$M4: \neg P_1 \wedge \neg P_2$	$M8. \neg P_1 \wedge \neg P_3$	$M12. \neg P_1 \wedge \neg P_5$

M13. $P_4 \wedge P_2$	M17. $P_4 \wedge P_3$	M21. $P_4 \wedge P_5$
×M14. $\neg P_4 \wedge P_2$	M18. $\neg P_4 \wedge P_3$	M22. $\neg P_4 \wedge P_5$
×M15. $P_4 \wedge \neg P_2$	M19. $P_4 \wedge \neg P_3$	M23. $P_4 \wedge \neg P_5$
M16. $\neg P_4 \wedge \neg P_2$	M20. $\neg P_4 \wedge \neg P_3$	M24. $\neg P_4 \wedge \neg P_5$

Note:

$\geq$ এর মধ্যে conflict | = Confusion

$\leq$ " " " | P_1 $\neg P_4$ same — not equal এর বাদ $\neg P_2$

$\sum$ " " " | $\neg P_1$ P_4 same

$\sum$ " " "

$P_1 \wedge P_2 = \neg P_4 \wedge P_2$

M1 M14 conflict so, M14 Eliminated

M2 M13 conflict so, M2 "

M3 M16 " so, M3 "

~~M4 M15 " So,~~

M4 M15 ~~confusion~~ so, ~~M4~~, M15

M1 and M13 are not eliminated.

M4 and M16 are confusion because $\neg P_2$ same. So, M4 and M16 eliminated

So, M1 and M13 are not eliminated.

Pg. 29

For $P_1 \wedge P_3$, similand $P_4 \wedge P_3$, similarly,

M_5 and M_{17} are remain.

For $P_1 \wedge P_5$ and $P_4 \wedge P_5$, ",

M_9 and M_{21} ~~or~~ remaining pair.

Proj1 Empty

Proj2

PNO	PNAME	BUDGET	LOC

Proj3

2.1.1.3 Derived Horizontal

Q: Emp দুই ভাগে ভাগ Salary অনুযায়ী

Emp1 Emp2

$Emp1 = Emp \ltimes Pay1$ 30000 < sal

$Emp2 = Emp \ltimes Pay2$

where, $Pay1 = \sigma_{sal \nleq 30000}$

$Pay2 = \sigma_{sal \leq 30000}$

** Source -কে 2 ভাগ করবো

Pay 1 , Pay2

Pay 1 → Elect. Eng, Syst. "

Pay2 → Mech, Programming

EMP1

EMO	ENAME	TITLE
E1	J Doe	Elect Eng
E2	M. Smith	Syst
E5	R.	Syst
E6	L. Chu	Elect
E8	J. Jones	Syst

EMP2

<u>ENO</u>	EName	Title
E3		
E4		
E7		

Q: ASG Table fragment by Location.

$PROJ1 = \sigma_{LOC = \text{"Montreal"}} (PROJ)$

$PROJ2 = \sigma_{LOC = \text{"New York"}} (PROJ)$

$PROJ3 = \sigma_{LOC = \text{"Paris"}} (PROJ)$

$ASG_1 = ASG \ltimes Proj1$

$ASG_2 = ASG \ltimes Proj2$

$ASG_3 = ASG \ltimes Proj3$

Proj 1

<u>ENO</u>	<u>PNO</u>	RESP	DUR
E1	P1		
E2			

Proj 2

ENO	PNO	
E2	P2	
E3	P3	
E4	P2	
E5	P2	
E7	P3	
E8	P3	

Proj 3

ENO	PNO	
E3	P4	
E6	P4	

Q: ASG Table fragment by Title

$EMP_1 = \sigma_{EMP = \text{"Electr ENG"}}(EMP)$ E1, E6

$EMP_2 = \sigma_{EMP = \text{"Syst.A"}}(EMP)$ E2 E5, E8

$EMP_3 = \sigma_{EMP = \text{"Mach"}}(EMP)$ E3 E7

$EMP_4 = \sigma_{EMP = \text{"Programmig"}}(EMP)$ E4

$ASG_1 = ASG \ltimes EMP_1$

$ASG_2 = ASG \ltimes EMP_2$

$ASG_3 = ASG \ltimes EMP_3$

$ASG_4 = ASG \ltimes EMP_4$

ASG1

ENO	PNO	
E1	P1	
E6	P4	

ASG2

ENO	PNO
E_2	P1
E_5	P2
E8	P3
E2	P2

ASG3

ENO	PNO
E_3	P_3
E_3	P_4
E7	P_3

ASG4

ENO	PNO
E4	P_2

2.1.1.9 Checking for correctness

Completeness

The completeness of a PHF is based on the selection predicates used. As long as the selection predicates are complete the resulting frag is complete. Since the basis of the frag algorithm is a set of complete and minimal

52P Reconstruction:

Disjoinness: Mutually exclusive

Fig 2.16 Hybrid frag tree and reconstru