CSE 4125: Distributed Database Systems
Chapter – 4

Distributed Database Design.

(part – A)

Outline

- Problems in designing DDB.
- Objectives of the Design of Data Distribution.
- Approaches to Design the Data Distribution.
- The Design of Primary Horizontal Fragmentation.

Problems in designing centralized DB

- Designing *conceptual schema* (description of the database).
- Designing *physical database* (mapping conceptual schema to storage area and defining access methods).

Problems in designing DDB

- Designing ~~*conceptual schema*~~ (description of the database).
- Designing ~~*physical database*~~ (mapping conceptual schema to storage area and defining access methods).
- Designing *fragmentation*.
- Designing *allocation of fragments*.

Objectives of the Design of Data Distribution

- **Processing locality.**
  - Placing data as close as possible to the application using them.

- **Availability and reliability.**
  - Multiple copies of data.
  - Recovery.

Objectives of the Design of Data Distribution (cont.)

- **Workload distribution.**
  - Taking advantage of the powers and computer resources at each site.
  - Parallel execution of application.

- **Storage costs and availability.**
  - CPU, I/O and transmission costs.
  - Considering the storage limitation.

Approaches to Design the Data Distribution

- **Top-Down approach.**
  - Have a database.
  - How to partition and allocate to individual sites.
- **Bottom-Up approach.**
  - Have existing databases at different sites.
  - How to integrate them
  - How to deal with heterogeneity and autonomy (i.e. independence).

The Design of Fragmentation

1. Design of Horizontal Fragmentation
    - Primary
    - Derived
2. Design of Vertical Fragmentation
3. Design of Mixed Fragmentation

The Design of Primary Horizontal Fragmentation

Simple Predicate

Given a relation $R(A_1, A_2, \ldots, A_n)$ where A_i has 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 $Value \in D_i$

Simple Predicate (cont.)

Example: Given global relation ***J***.

J

JNO	JNAME	BUDGET	LOC
J1	Instrumental	150,000	Montreal
J2	Database Dev.	135,000	New York
J3	CAD/CAM	250,000	New York
J4	Maintenance	350,000	Orlando

- Simple predicates: $p_j : A_i \theta Value$

p_1: *JNAME = "Maintenance"*

P_2: *BUDGET <= 200,000*

Minterm Predicate

Given a set of simple predicates for relation ***R***:

$$\boldsymbol{P = \{ p_1, p_2, \ldots, p_m \}},$$

The set of minterm predicates: $\boldsymbol{M = \{ m_1, m_2, \ldots, m_n \}}$ is defined as,

$$M = \{ m_i \mid m_i = \bigwedge_{p_j \in P} p_j^* \}$$

where $p_j^* = p_j$ or $p_j^* = \neg(p_j)$.

Provided that, $m_i \neq false$.

Minterm Predicate (cont.)

Example:

TITLE	SAL
Elect. Eng.	40,000
Syst. Analy.	54,000
Mech. Eng.	32,000
Programmer	42,000

Possible simple predicates:

P_1: TITLE="Elect. Eng."
P_2: TITLE="Syst. Analy"
P_3: TITLE="Mech. Eng."
P_4: TITLE="Programmer"
P_5: SAL<=35,000
P_6: SAL > 35,000

Some corresponding minterm predicates:

$$m_1 : TITLE = "Elect.Eng." \wedge SAL \leq 35{,}000$$

$$m_2 : TITLE \neq "Elect.Eng" \wedge SAL > 35{,}000$$

Horizontal Fragments

- A horizontal fragment $\boldsymbol{R_i}$ of relation $\boldsymbol{R}$ consists of all the tuples of $\boldsymbol{R}$ that satisfy a minterm predicate $\boldsymbol{m_i}$.
- There are as many horizontal fragments (also called *minterm fragments*) as there are minterm predicates.

Desirable properties of the set of simple predicates

Which minterm predicate should we use?

- We have to decide on the *set of simple predicates* that are the basis for the minterm predicates.

- Selection of predicates cannot be helped too much by precise rules since usefulness of particular predicates mostly relies on the intuition of the database designer.
- However, there are two properties: ***Complete*** and ***Minimal***.

Completeness

A set of simple predicate $\boldsymbol{P_r}$ is said to be **complete** if and only if –

Any two tuples in the same fragment (defined by $\boldsymbol{P_r}$) are referenced (accessed) with the same probability by any application (i.e. query).

Example of Completeness*

Pr= { LOC="Montreal", LOC="New York", LOC="Orlando"

J

JNO	JNAME	BUDGET	LOC
J1	Instrumental	150,000	Montreal
J2	Database Dev.	135,000	New York
J3	CAD/CAM	250,000	New York
J4	Maintenance	350,000	Orlando

JP1

JNO	JNAME	BUDGET	LOC
J1	Instrumental	150,000	Montreal

JP2

JNO	JNAME	BUDGET	LOC
J2	GUI	135,000	New York
J3	CAD/CAM	250,000	New York

JP3

JNO	JNAME	BUDGET	LOC
J4	Database Dev.	310,000	Orlando

(Here, $JP_i = SL_{Pi}\ J$)

Example of Completeness* (cont.)

- Case 1: An application (app_1) is issued in three sites. It wants to access the tuples according to the location (any location).

JP1

JNO	JNAME	BUDGET	LOC
J1	Instrumental	150,000	Montreal

JP2

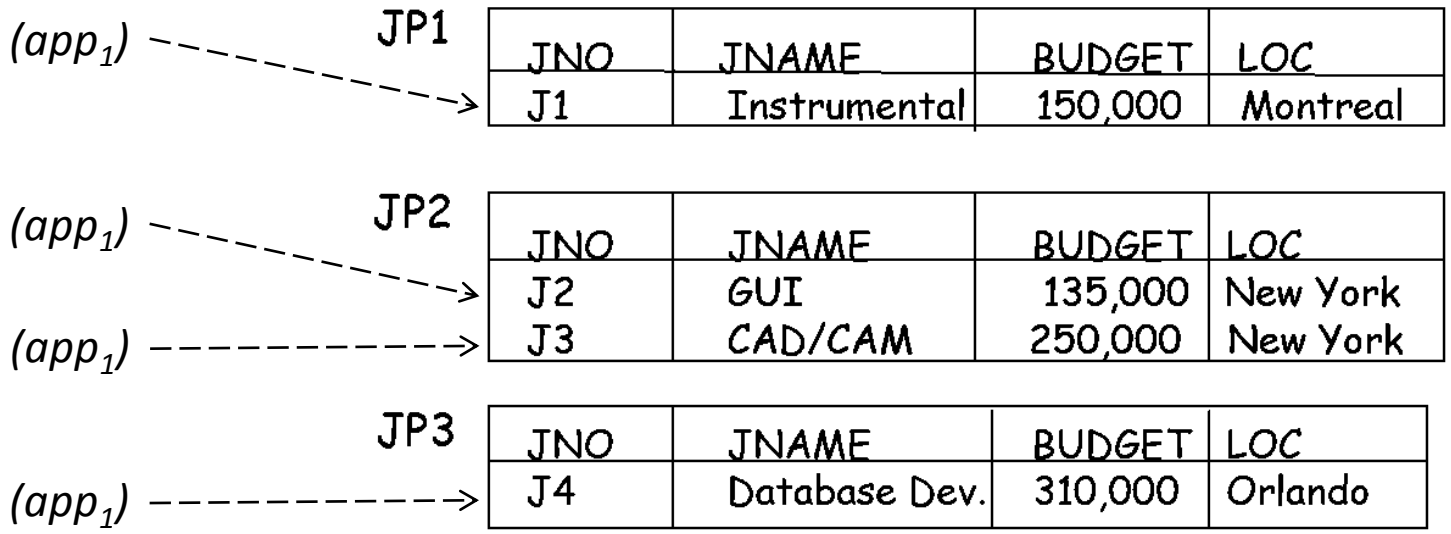
JNO	JNAME	BUDGET	LOC
J2	GUI	135,000	New York
J3	CAD/CAM	250,000	New York

JP3

JNO	JNAME	BUDGET	LOC
J4	Database Dev.	310,000	Orlando

Example of Completeness* (cont.)

- In this case, P_r is *complete* because each tuple of each fragment has the same probability of being accessed.



Example of Completeness* (cont.)

- Case 2: There is a second application (app_2) which is also issued in three sites. It accesses only those tuples where *budget is less than $200,000.*

JP1

JNO	JNAME	BUDGET	LOC
J1	Instrumental	150,000	Montreal

JP2

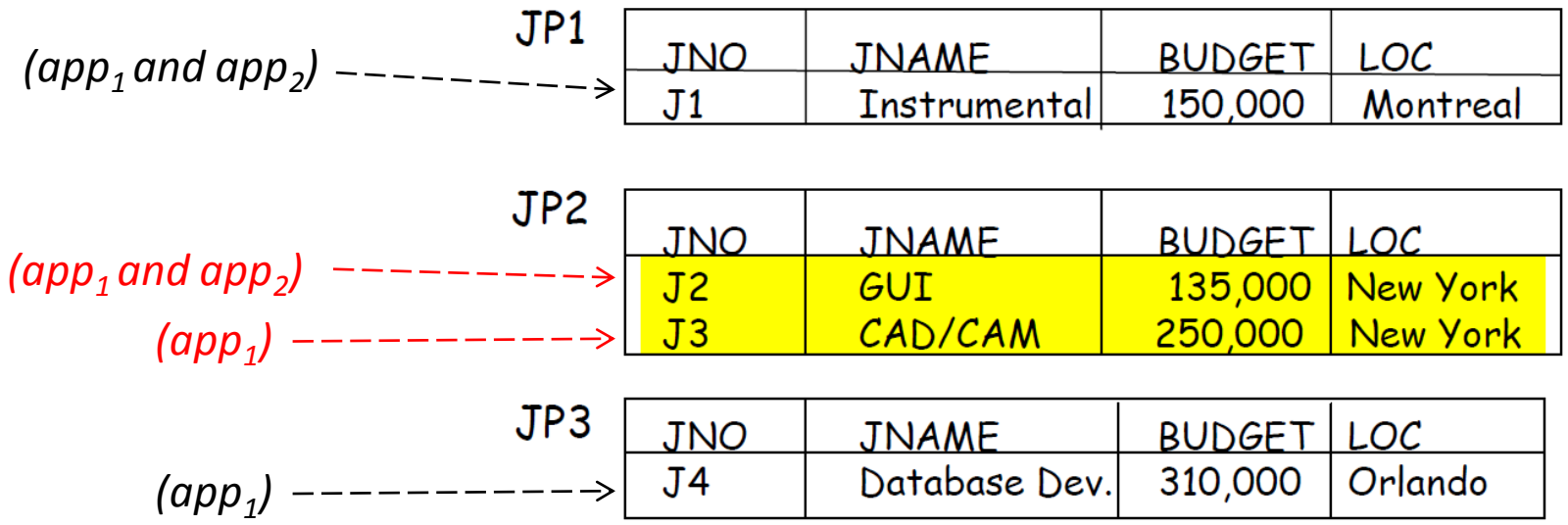
JNO	JNAME	BUDGET	LOC
J2	GUI	135,000	New York
J3	CAD/CAM	250,000	New York

JP3

JNO	JNAME	BUDGET	LOC
J4	Database Dev.	310,000	Orlando

Example of Completeness* (cont.)

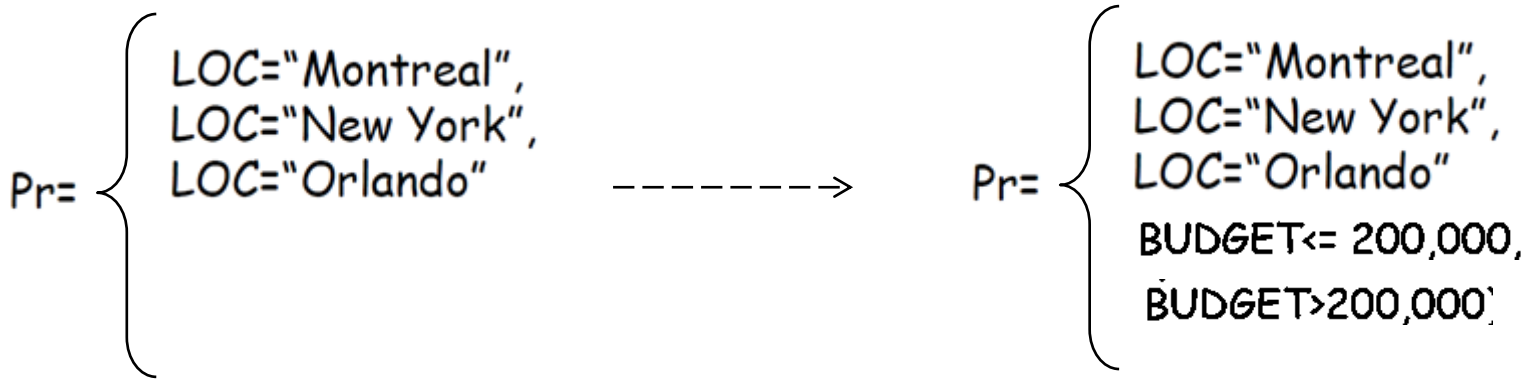
- Tuple J_2 has higher access probability than tuple J_3 in JP_2. In this case, P_r is *not complete* since some tuples (J_i) in JP_i has higher access probability.



Example of Completeness* (cont.)

- To make the set complete, we need to add –

  BUDGET<= 200,000, *BUDGET>200,000*) to $\boldsymbol{P_r}$.



Example of Completeness* (cont.)

Pr= {
LOC="Montreal",
LOC="New York",
LOC="Orlando"
BUDGET<= 200,000,
BUDGET>200,000}

$SL_{LOC = 'Montreal'}(J)$

JP1

JNO	JNAME	BUDGET	LOC
J1	Instrumental	150,000	Montreal

$SL_{LOC = 'New York' \text{ and } budget <= 200,000}(J)$

JP21

JNO	JNAME	BUDGET	LOC
J2	GUI	135,000	New York

$SL_{LOC = 'New York' \text{ and } budget > 200,000}(J)$

JP22

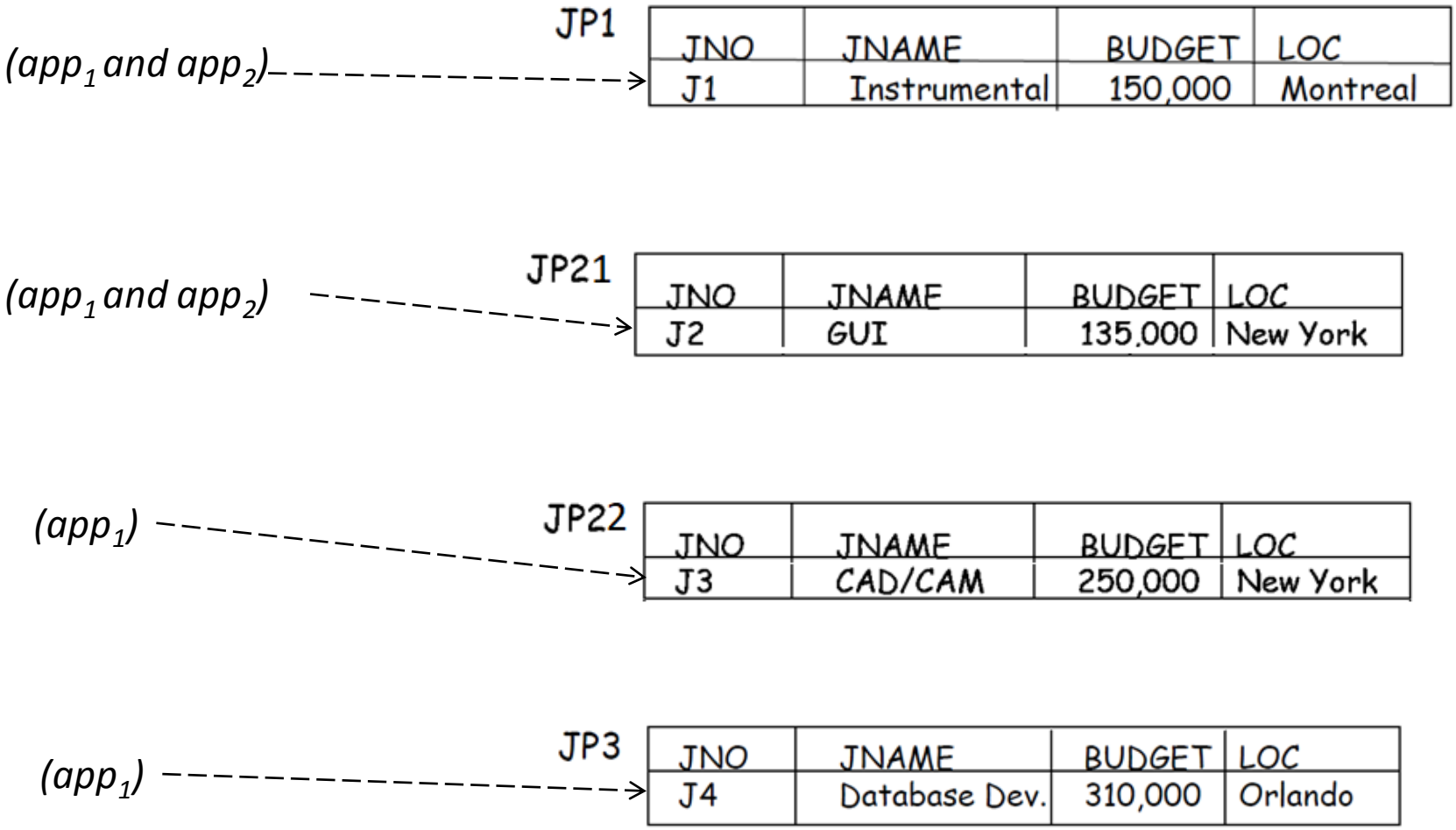
JNO	JNAME	BUDGET	LOC
J3	CAD/CAM	250,000	New York

$SL_{LOC = 'Orlando'}(J)$

JP3

JNO	JNAME	BUDGET	LOC
J4	Database Dev.	310,000	Orlando

Example of Completeness* (cont.)



Minimal

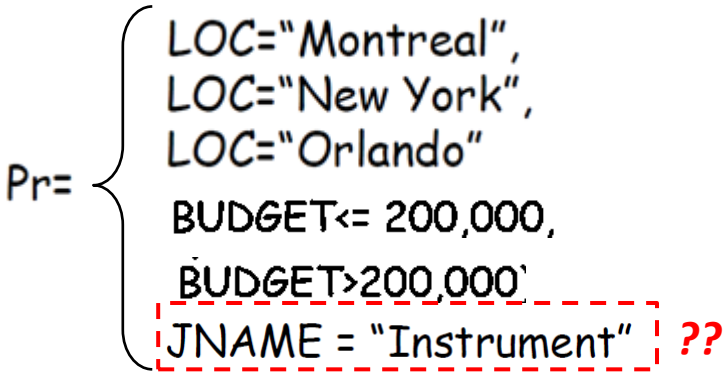
- The set of predicates P_r is **minimal** if and only if there is at least one application (i.e. query) that accesses the fragment.

$$Pr = \{ LOC=\text{"Montreal"},\ LOC=\text{"New York"},\ LOC=\text{"Orlando"},\ BUDGET <= 200{,}000,\ BUDGET > 200{,}000 \}$$

Example of Minimality*

Considering the previous app_1 and app_2 , and the set $\boldsymbol{P_r}$.

- If we add the predicate *JNAME = "Instrument"* to $\boldsymbol{P_r}$.
- Resulting $\boldsymbol{P_r}$ is *not minimal* since the new predicate is not contributing to least one of the applications.



Additional Reading

- Advantages and disadvantages of –
  - Top-down approach.
  - Bottom-up approach.

Practice Problems/ Questions

- Create your own scenario with –
  - Global relation and fragments (like ***J***, $\boldsymbol{JP}_i$ etc. in the lecture slides).
  - Different sets simple predicates (like *LOC = 'Montreal'* in the lecture slides)
  - Different applications (like app_1 and app_2 in the lecture slides)

Now, determine if the sets are complete.