Outline

- Introduction
- Background
- Distributed DBMS Architecture
- Distributed Database Design
  - Fragmentation
  - Data Location
- Distributed Query Processing (Briefly)
- Distributed Transaction Management (Extensive)
- Building Distributed Database Systems (RAID)
- Mobile Database Systems
- Privacy, Trust, and Authentication
- Peer to Peer Systems
Useful References

- Textbook *Principles of Distributed Database Systems*,
  Chapter 3.1, 3.2
Design Problem

- In the general setting:

  Making decisions about the placement of *data* and *programs* across the sites of a computer network as well as possibly designing the network itself.

- In Distributed DBMS, the placement of applications entails
  - placement of the distributed DBMS software; and
  - placement of the applications that run on the database.
Dimensions of the Problem

- **Level of sharing**
  - static
  - data
  - data + program

- **Level of knowledge**
  - partial information
  - complete information

- **Access pattern behavior**
  - dynamic
Distribution Design

- **Top-down**
  - mostly in designing systems from scratch
  - mostly in homogeneous systems

- **Bottom-up**
  - when the databases already exist at a number of sites
Distribution Design Issues

- Why fragment at all?
- How to fragment?
- How much to fragment?
- How to test correctness?
- How to allocate?
- Information requirements?
Fragmentation

- Can't we just distribute relations?
- What is a reasonable unit of distribution?
  - relation
    - views are subsets of relations
    - extra communication
  - fragments of relations (sub-relations)
    - concurrent execution of a number of transactions that access different portions of a relation
    - views that cannot be defined on a single fragment will require extra processing
    - semantic data control (especially integrity enforcement) more difficult
Fragmentation Alternatives – Horizontal

PROJ₁ : projects with budgets less than $200,000

PROJ₂ : projects with budgets greater than or equal to $200,000

<table>
<thead>
<tr>
<th>PNO</th>
<th>PNAME</th>
<th>BUDGET</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>Instrumentation</td>
<td>150000</td>
<td>Montreal</td>
</tr>
<tr>
<td>P2</td>
<td>Database Develop.</td>
<td>135000</td>
<td>New York</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PNO</th>
<th>PNAME</th>
<th>BUDGET</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>P3</td>
<td>CAD/CAM</td>
<td>250000</td>
<td>New York</td>
</tr>
<tr>
<td>P4</td>
<td>Maintenance</td>
<td>310000</td>
<td>Paris</td>
</tr>
<tr>
<td>P5</td>
<td>CAD/CAM</td>
<td>500000</td>
<td>Boston</td>
</tr>
</tbody>
</table>
### Fragmentation Alternatives – Vertical

**PROJ$_1$:** information about project budgets

**PROJ$_2$:** information about project names and locations

#### PROJ$_1$

<table>
<thead>
<tr>
<th>PNO</th>
<th>BUDGET</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>150000</td>
</tr>
<tr>
<td>P2</td>
<td>135000</td>
</tr>
<tr>
<td>P3</td>
<td>250000</td>
</tr>
<tr>
<td>P4</td>
<td>310000</td>
</tr>
<tr>
<td>P5</td>
<td>500000</td>
</tr>
</tbody>
</table>

#### PROJ$_2$

<table>
<thead>
<tr>
<th>PNO</th>
<th>PNAME</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>Instrumentation</td>
<td>Montreal</td>
</tr>
<tr>
<td>P2</td>
<td>Database Develop.</td>
<td>New York</td>
</tr>
<tr>
<td>P3</td>
<td>CAD/CAM</td>
<td>New York</td>
</tr>
<tr>
<td>P4</td>
<td>Maintenance</td>
<td>Paris</td>
</tr>
<tr>
<td>P5</td>
<td>CAD/CAM</td>
<td>Boston</td>
</tr>
</tbody>
</table>
Degree of Fragmentation

finite number of alternatives

tuples or attributes

Finding the suitable level of partitioning within this range

relations
Correctness of Fragmentation

- Completeness
  - Decomposition of relation $R$ into fragments $R_1, R_2, ..., R_n$ is complete if and only if each data item in $R$ can also be found in some $R_i$

- Reconstruction
  - If relation $R$ is decomposed into fragments $R_1, R_2, ..., R_n$, then there should exist some relational operator $\nabla$ such that
    \[ R = \nabla_{1 \leq i \leq n} R_i \]

- Disjointness
  - If relation $R$ is decomposed into fragments $R_1, R_2, ..., R_n$, and data item $d_i$ is in $R_j$, then $d_i$ should not be in any other fragment $R_k$ ($k \neq j$).
Other Fragmentation Issues

- Privacy
- Security
- Bandwidth of Connection
- Reliability
- Replication Consistency
- Local User Needs