Adapted for a textbook by Date C. J.

An Introduction to Database Systems
Pearson Addison Wesley, 2004

Introduction to Distributed Databases

Remigijus GUSTAS
Phone: +46-54 700 17 65
E-mail: Remigijus.Gustas@kau.se
http://www.cs.kau.se/~gustas/
Distributed Database System

✓ One portion of data is stored on one computer and another portion on a different machine, occasional access to data on the other computer.

A client is able to access many servers simultaneously.
The fundamental principle of distributed database

✔ Principle: To the user, a distributed system should look exactly like a non-distributed system

- Users (who perform data manipulation operations) in a distributed system should be able to behave as if the system is not distributed.

- Data definition operations, by contrast will require some extensions in a distributed system. For instance, a user at a site X can specify that a given base relvar is divided into fragments that are stored at sites Y and Z.
The Twelve Rules
(of the fundamental principle)

1. Local autonomy
2. No reliance on a central site
3. Continuous operation
4. Location independence
5. Fragmentation independence
6. Replication independence
7. Distributed query processing
8. Distributed transaction management
9. Hardware independence
10. Operating system independence
11. Network independence
12. DBMS independence
Local Autonomy

- All operations at a given site are controlled by that site.
- No site should depend on some other site for its successful operation (able to run if other site is down).
- Local data is locally owned and managed (even if data base if accessible from other sites).
- Integrity, security and a way of physical storage representation of local data remain under control and jurisdiction of the local site.
No reliance on a central site

✓ All sites must be treated as equals
✓ No reliance on a central master site for some service
✓ ‘No reliance on a central site’ is desirable objective even if full ‘local autonomy’ is not achieved
✓ Reliance on a central site would be undesirable for the following reasons:
  ■ Central site might be a bottleneck
  ■ The system is vulnerable, if central site went down
Continuous Operation

✓ **Reliability** is the probability that system is up and running at any moment of time.

✓ Reliability is improved in distributed systems, because they continue to operate (at reduced level) in case some database component fails.

✓ **Availability** is the probability that system is up and running continuously through the specified time period.

✓ Availability is improved in distributed systems, because they continue to operate in case of some component failure (partially because of data replication.)
Location independence

✓ Users should not have to know where data is physically stored

✓ Location independence is just an extension to the distributed case of property of physical data independence

✓ Location independence allows data to migrate from site to site without invalidating any programs

✓ Migratability is desirable, because it allows data to be moved around the network in response to changing performance requirements
Fragmentation independence

✓ System supports data fragmentation if a given base relvar can be divided into fragments and stored at different sites
✓ Data can be stored at the location where it is most frequently used, so that most operations are local and network traffic is reduced
✓ Two kinds of fragmentation:
  ■ Horizontal. Restrictions must constitute an orthogonal decomposition
  ■ Vertical. Projections must constitute a non loss decomposition
Horizontal Fragmentation

EMPL

<table>
<thead>
<tr>
<th>EMP#</th>
<th>DEPT#</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>D1</td>
<td>19000</td>
</tr>
<tr>
<td>E2</td>
<td>D2</td>
<td>22000</td>
</tr>
<tr>
<td>E3</td>
<td>D3</td>
<td>35000</td>
</tr>
<tr>
<td>E4</td>
<td>D2</td>
<td>20000</td>
</tr>
</tbody>
</table>

Perceived by User

Site = 'London'
L_EMPL

<table>
<thead>
<tr>
<th>EMP#</th>
<th>DEPT#</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>D1</td>
<td>19000</td>
</tr>
<tr>
<td>E2</td>
<td>D2</td>
<td>22000</td>
</tr>
<tr>
<td>E4</td>
<td>D2</td>
<td>20000</td>
</tr>
</tbody>
</table>

Site = 'Paris'
P_EMPL

<table>
<thead>
<tr>
<th>EMP#</th>
<th>DEPT#</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>E3</td>
<td>D3</td>
<td>35000</td>
</tr>
</tbody>
</table>
Replication independence

Users should be able to behave as if the data were not replicated

✓ A system supports data replication if a given fragment of a given base relvar can be represented by distinct copies at many sites

✓ Desirable for two reasons:
  better performance and better availability
  - Applications can operate on local copies instead of having to communicate with remote sites
  - Replicated object remains available as long as at least one copy remains available

✓ Disadvantage: Update propagation problem
Responsibility of optimizer is to determine which replicas need to be accessed.

<table>
<thead>
<tr>
<th>Site = 'London'</th>
<th>Site = 'Paris'</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>L_EMPL</strong></td>
<td><strong>P_EMPL</strong></td>
</tr>
<tr>
<td>EMP#</td>
<td>DEPT#</td>
</tr>
<tr>
<td>E1</td>
<td>D1</td>
</tr>
<tr>
<td>E2</td>
<td>D2</td>
</tr>
<tr>
<td>E4</td>
<td>D2</td>
</tr>
</tbody>
</table>

**REPLICATE** L_EMPL **AS** LN_EMP **AT SITE** 'London'

**REPLICATE** P_EMPL **AS** PR_EMP **AT SITE** 'Paris'

<table>
<thead>
<tr>
<th>EMP#</th>
<th>DEPT#</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>D1</td>
<td>19000</td>
</tr>
<tr>
<td>E2</td>
<td>D2</td>
<td>22000</td>
</tr>
<tr>
<td>E4</td>
<td>D2</td>
<td>20000</td>
</tr>
<tr>
<td>E3</td>
<td>D3</td>
<td>35000</td>
</tr>
</tbody>
</table>
Distributed Query Processing

✓ Optimization in a distributed system:
  ■ There are many ways of moving data around in the system to satisfy the request
  ■ Efficient optimization strategy is crucial

✓ Relational systems are likely to outperform nonrelational ones by orders of magnitude
  ■ Variation of response time to the same query could be from 1/10 of second to several hours
  ■ N requests for N for records (in return) may be sent
  ■ One request and one result (consisting of N tuples)
Distributed transaction management

✓ Recovery: Transaction is atomic
  - In distributed environment the set of agents for one transaction either all commit or all roll back
    - Transaction consists of several agents, if it involves execution of code at many sites
    - An agent is a process performed on behalf of a given transaction at a given site (update process)

✓ Concurrency
  - System needs to know when two agents are part of the same transaction
  - Such two agents are not allowed to deadlock with each other
Hardware Independence
Operating System Independence
Network Independence

✓ It is desirable to run the same DBMS on different hardware platforms
✓ ... Including different OS on the same hardware
✓ It is desirable to support a variety of disparate communication networks
DBMS independence

- Different sites do not necessarily all have to be copies of the DBMS system software
- Many different operating systems quite often run different DBMS
  - If INGRES and ORACLE DBMS both support the SQL standard, then it might be possible an INGRES site to talk to ORACLE site
- Heterogeneity support is definitely desirable
- Ideal distributed system should provide DBMS independence