PURDUE UNIVERSITY

CS 542 Distributed Database Systems

Prof. Chris Clifton
Spring 2009

Indiana Center for Database Systems

Text

Principles of Distributed Database Systems
(Second Edition)
Tamer Ozsu and Patrick Valduriez
Prentice Hall, 1999
ISBN 0-13-659707-6
http://www.cs.ualberta.ca/~database/ddbook.html

Course Objective

- This course will cover fundamental concepts and issues of distributed database systems
  - The course is *not* about the use of a distributed database management system
- Students are expected to have an undergraduate-level familiarity with concepts of database systems and distributed computing

Topics

- Distributed transaction processing and concurrency control
- Distributed reliability
- Distributed query processing
- Parallel database systems

Grading

- Midterm Exam (20%)
- Final Exam (Comprehensive, 25%)
- Written assignments and Lab Projects (45%)
- Evaluation of instructor based on in-class contributions, discussions, and overall performance (10%)

Projects

- Largely to be developed, but it is suggested that you:
  - Be comfortable with Java programming
  - can learn a small subset of CORBA (necessary help will be provided in class).
    - CORBA will be available on the CS machines – you do not need to have it on your end.
- Other requirements will be discussed as they arise

PURDUE UNIVERSITY

CS 542: Distributed Database Systems

Introduction

12 January, 2009

Indiana Center for Database Systems

What is a DBMS?

- A very large, integrated collection of data.
- Models real-world *enterprise.*
  - Entities (e.g., students, courses)
  - Relationships (e.g., Madonna is taking CS564)
- A *Database Management System (DBMS)* is a software package designed to store and manage databases.

(C) Ozsu & Valduriez 12

Why Use a DBMS?

- Data independence and efficient access.
- Reduced application development time.
- Data integrity and security.
- Uniform data administration.
- Concurrent access, recovery from crashes.

(C) Ozsu & Valduriez 13

Why use a Distributed DBMS?

- Data is commonly distributed among several sites in the real world
- Multiple copies of data provide improved reliability and availability.
- Performance can be improved through the use of multiple, distributed hardware devices.
- Centralizing the data may not be acceptable.

(C) Ozsu & Valduriez 14

What is distributed?

- Processing Logic
- Functions
- Data
- Control

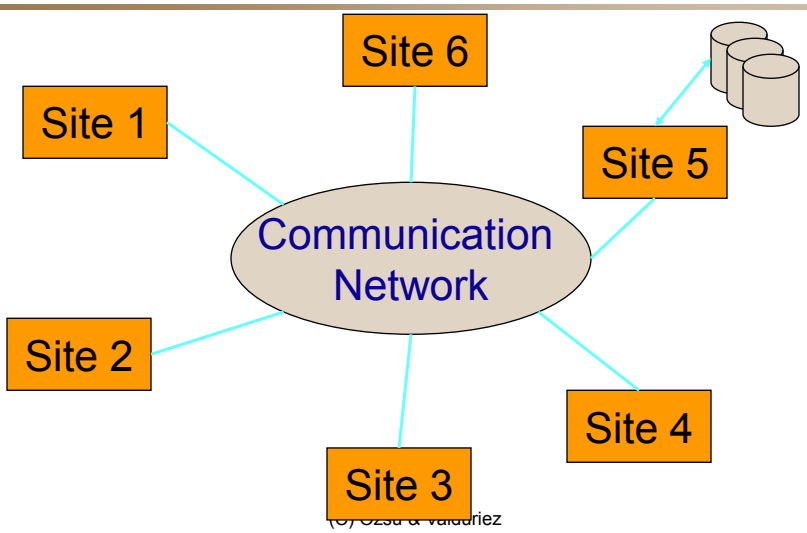
What is a Distributed DBMS?

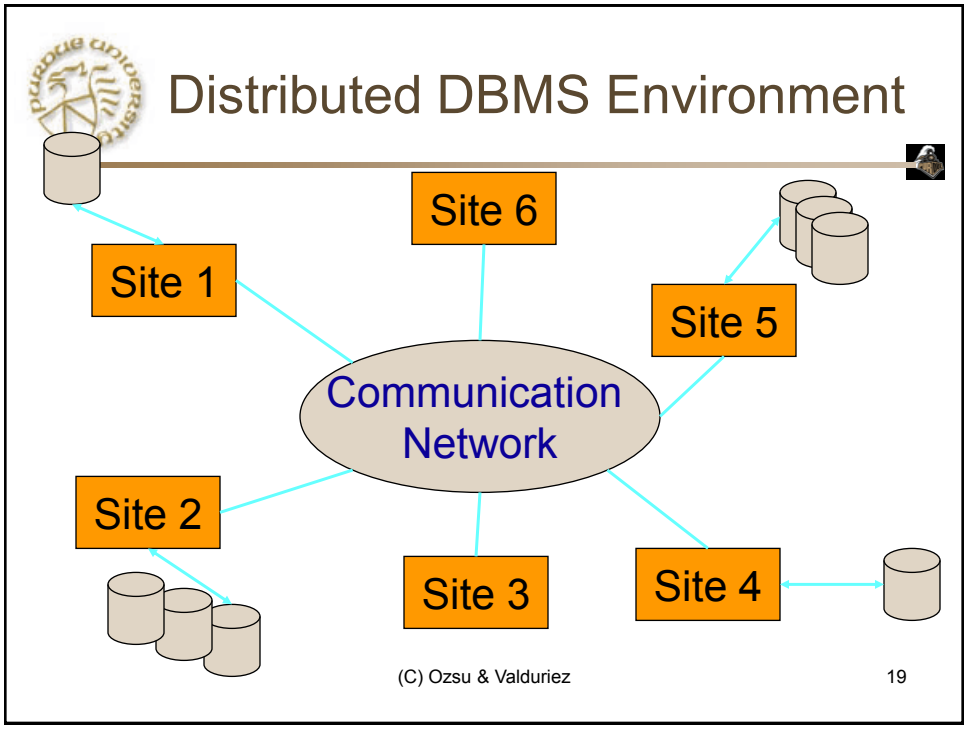
- A distributed database (DDB) is a collection of multiple, *logically interrelated* databases distributed over a *computer network*.
- A distributed DBMS (DDBMS) is the software that manages the DDB and provides an access mechanism that makes this distribution transparent.

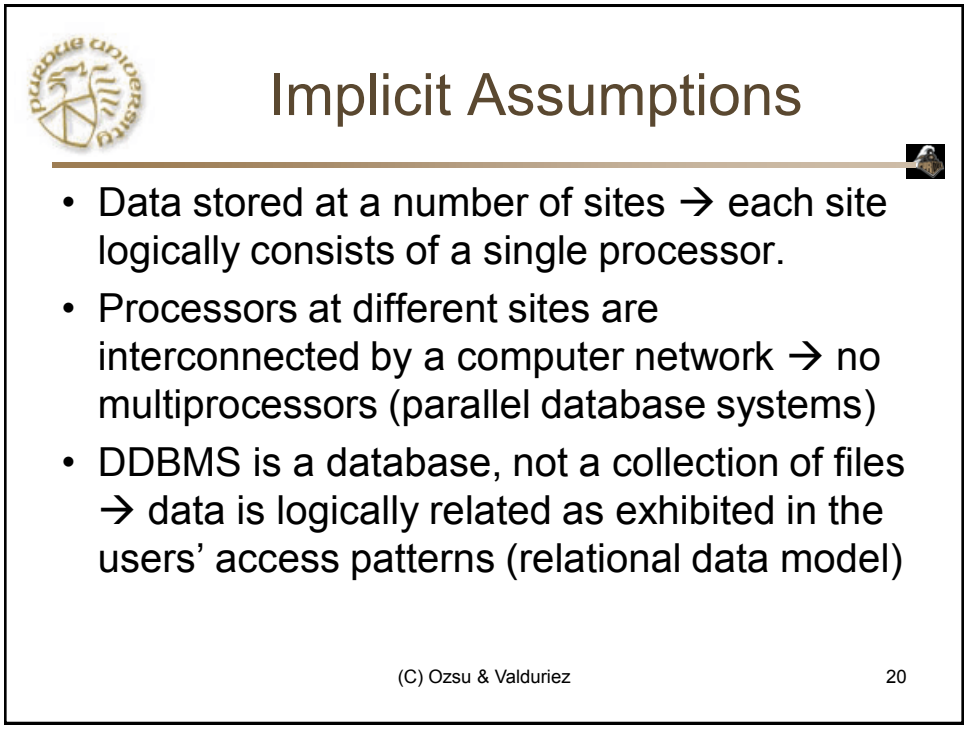
What is not a DDBMS?

- A timesharing computer system.
- A loosely or tightly coupled multiprocessor system.
- A database system which resides at one of the nodes of a network of computers – this is a centralized database on a network node
- Distributed application servers connected to a centralized database

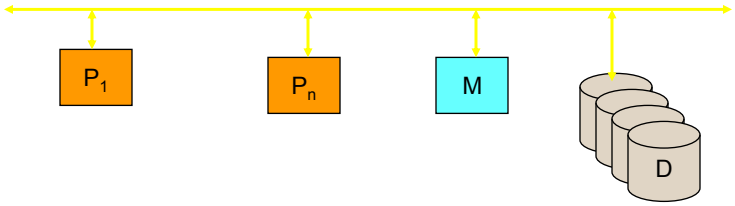
Centralized DBMS on a network




Distributed DBMS Environment
Site 6
Site 1
Site 5
Communication Network
Site 2
Site 3
Site 4
(C) Ozsu & Valduriez
19

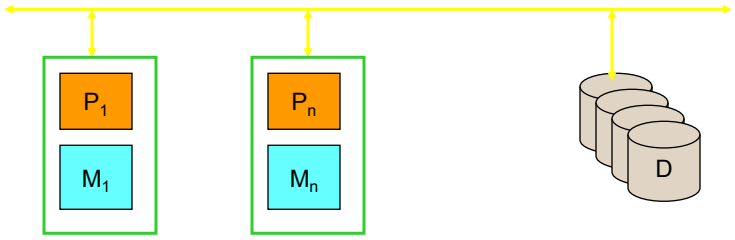

Implicit Assumptions
• Data stored at a number of sites → each site logically consists of a single processor.
• Processors at different sites are interconnected by a computer network → no multiprocessors (parallel database systems)
• DDBMS is a database, not a collection of files → data is logically related as exhibited in the users' access patterns (relational data model)
(C) Ozsu & Valduriez
20

Shared-Memory Architecture



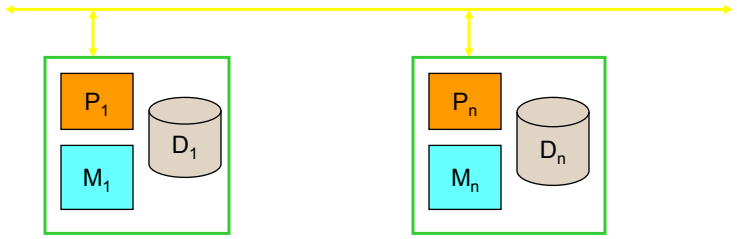
- Examples: symmetric multiprocessors (Sequent, Encore) and some mainframes (IBM3090, Bull's DPS8)

Shared-Disk Architecture



- Examples: DEC's VAX Cluster, IBM's IMS/VS Data Sharing

Shared-Nothing Architecture



- Examples: Teradata's DBC, Tandem, Intel's Paragon, NCR's 3600 and 3700

Applications

- Manufacturing – especially multi-plant
- Military command and control
- Airlines
- Hotel Chains
- Any organization which has a decentralized organization structure.

Distributed DBMS Promises

- Transparent management of distributed, fragmented, and replicated data.
- Improved reliability/availability through distributed transactions.
- Improved performance.
- Easier and more economical system expansion.

Transparency

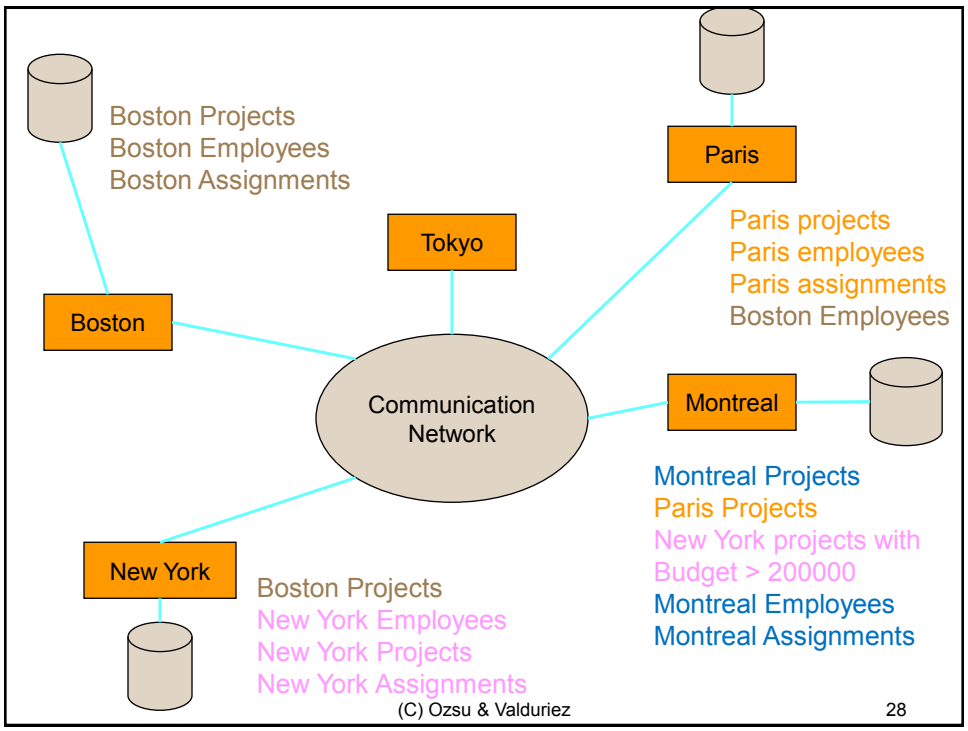
- Transparency is the separation of the higher level semantics of a system from the lower level implementation issues.
- Fundamental issue is to provide DATA INDEPENDENCE in the distributed environment.
  - Network (distribution) transparency
  - Replication transparency
  - Fragmentation transparency

Example

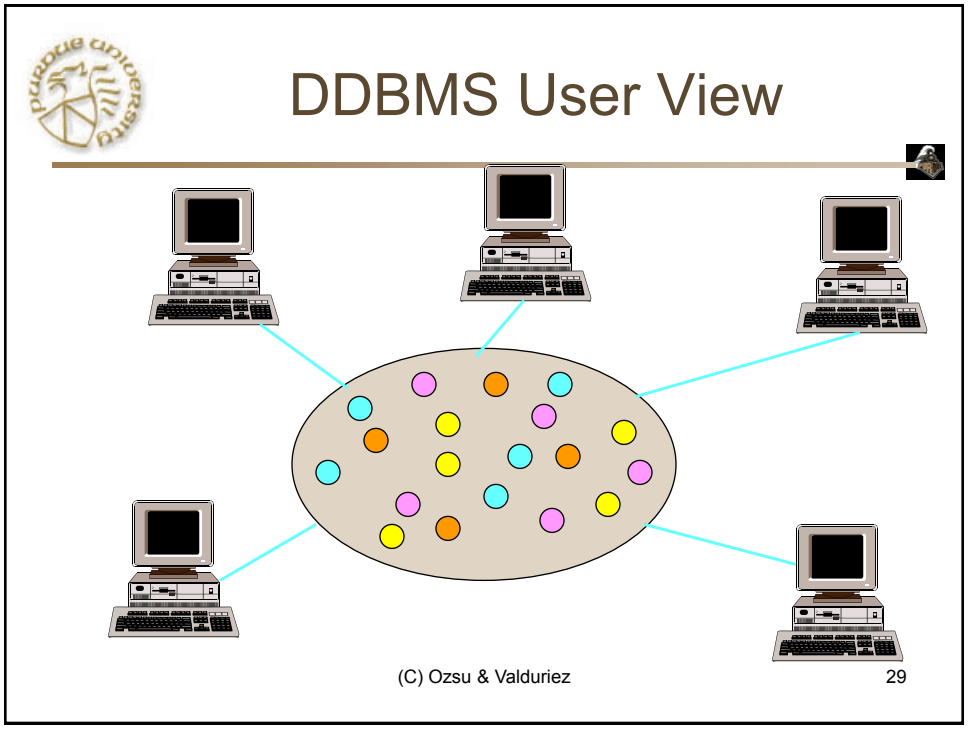
EMP: (ENO, ENAME, ETITLE)
ASG: (ENO, PNO, RESP, DUR)
PROJ: (PNO, PNAME, BUDGET, LOC)
PAY: (TITLE, SAL)

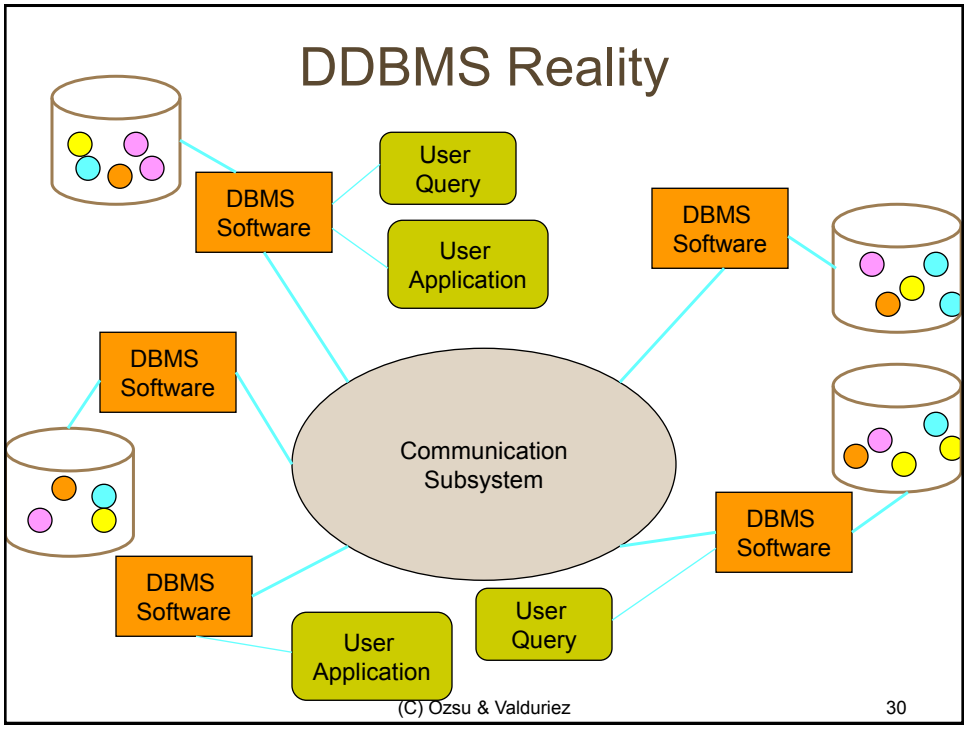
```
SELECT  ENAME, SAL
FROM EMP, ASG, PAY
WHERE DUR > 12
AND EMP.ENO=ASG.ENO
AND EMP.TITLE=PAY.TITLE
```

(C) Ozsu & Valduriez 27



(C) Ozsu & Valduriez 28


DDBMS User View
(C) Ozsu & Valduriez
29


DDBMS Reality
User Query
DBMS Software
User Application
DBMS Software
DBMS Software
Communication Subsystem
DBMS Software
DBMS Software
User Application
User Query
(C) Ozsu & Valduriez
30

Potentially Improved Performance

- Proximity of data to its point of use (Requires some support for fragmentation and replication)
- Parallelism in execution
  - Inter-query parallelism
  - Intra-query parallelism

Parallelism Requirements

- Have as much of the data required by each application at the site where the application executes
  - Full replication
- How about updates?
  - Updates to replicated data requires implementation of distributed concurrency control and commit protocols.

Distributed DBMS Issues

- Distributed Database Design
  - How to distribute the database
  - Replicated and non-replicated distribution.
  - Directory management
- Query Processing
  - Convert user transactions to data manipulation instructions
  - Optimization problem
  - Generally it is NP-hard.

Distributed DBMS Issues

- Concurrency Control
  - Synchronization of concurrent accesses
  - Consistency and isolation of transactions' effects
  - Deadlock Management
- Reliability
  - How to make the system resilient to failures
  - Atomicity and durability.