Outline

- Introduction
- Background
  - Distributed DBMS Architecture
  - Datalogical Architecture
  - Implementation Alternatives
  - Component Architecture
- Distributed DBMS Architecture
- Distributed Database Design
- Semantic Data Control
- Distributed Query Processing
- Distributed Transaction Management
- Parallel Database Systems
- Distributed Object DBMS
- Database Interoperability
- Current Issues

Architecture

- Defines the structure of the system
  - components identified
  - functions of each component defined
  - interrelationships and interactions between components defined
ANSI/SPARC Architecture

Reference Model
- A conceptual framework whose purpose is to divide standardization work into manageable pieces and to show at a general level how these pieces are related to one another.

Approaches
- **Component-based**
  - Components of the system are defined together with the interrelationships between components.
  - Good for design and implementation of the system.
- **Function-based**
  - Classes of users are identified together with the functionality that the system will provide for each class.
  - The objectives of the system are clearly identified. But how do you achieve these objectives?
- **Data-based**
  - Identify the different types of describing data and specify the functional units that will realize and/or use data according to these views.
Conceptual Schema Definition

RELATION EMP [
    KEY = {ENO}
    ATTRIBUTES = {
        ENO : CHARACTER(9)
        ENAME : CHARACTER(15)
        TITLE : CHARACTER(10)
    }
]

RELATION PAY [
    KEY = {TITLE}
    ATTRIBUTES = {
        TITLE : CHARACTER(10)
        SAL : NUMERIC(6)
    }
]

RELATION PROJ [
    KEY = {PNO}
    ATTRIBUTES = {
        PNO : CHARACTER(7)
        PNAME : CHARACTER(20)
        BUDGET : NUMERIC(7)
        LOC : CHARACTER(15)
    }
]

RELATION ASG [
    KEY = {ENO,PNO}
    ATTRIBUTES = {
        ENO : CHARACTER(9)
        PNO : CHARACTER(7)
        RESP : CHARACTER(10)
        DUR : NUMERIC(3)
    }
]
Internal Schema Definition

RELATION EMP [  
   KEY = {ENO}  
   ATTRIBUTES =  
      ENO : CHARACTER(9)  
      ENAME : CHARACTER(15)  
      TITLE : CHARACTER(10)  
   ]  

INTERNAL_REL_E [  
   INDEX ON E# CALL EMINX  
   FIELD =  
      E# : BYTE(9)  
      ENAME : BYTE(15)  
      TIT : BYTE(10)  
   ]

External View Definition – Example 1

Create a BUDGET view from the PROJ relation

CREATE VIEW BUDGET(PNAME, BUD) AS SELECT PNAME, BUDGET FROM PROJ
Create a Payroll view from relations EMP and TITLE_SALARY

```
CREATE VIEW PAYROLL (EMP_NO, EMP_NAME, SAL)
AS SELECT EMP.ENO, EMP.ENAME, PAY.SAL
FROM EMP, PAY
WHERE EMP.TITLE = PAY.TITLE
```
Dimensions of the Problem

■ Distribution
  ➔ Whether the components of the system are located on the same machine or not

■ Heterogeneity
  ➔ Various levels (hardware, communications, operating system)
  ➔ DBMS important one
    ♦ data model, query language, transaction management algorithms

■ Autonomy
  ➔ Not well understood and most troublesome
  ➔ Various versions [Veijalainen and Pepescu-Zeletin, 1988]
    ♦ Design autonomy: Ability of a component DBMS to decide on issues related to its own design.
    ♦ Communication autonomy: Ability of a component DBMS to decide whether and how to communicate with other DBMSs.
    ♦ Execution autonomy: Ability of a component DBMS to execute local operations in any manner it wants to.

Datalogical Distributed DBMS Architecture
Datalogical Multi-DBMS Architecture

Timesharing Access to a Central Database

- No data storage
- Host running all software

Terminals or PC terminal emulators

Batch requests

Response

Network

Communications

Application Software

DBMS Services

Database
Multiple Clients/Single Server

Communications

Client Services

Applications

Client Services

Communications

Applications

Client Services

Communications

DBMS Services

LAN

High-level requests

Filtered data only

Communications

Task Distribution

Application

QL Interface

... Programmatic Interface

Communications Manager

SQL query result table

Communications Manager

Query Optimizer

Lock Manager

Storage Manager

Page & Cache Manager

Database
Advantages of Client-Server Architectures

- More efficient division of labor
- Horizontal and vertical scaling of resources
- Better price/performance on client machines
- Ability to use familiar tools on client machines
- Client access to remote data (via standards)
- Full DBMS functionality provided to client workstations
- Overall better system price/performance

Problems With Multiple-Client/Single Server

- Server forms bottleneck
- Server forms single point of failure
- Database scaling difficult
Multiple Clients/Multiple Servers

- directory
- caching
- query decomposition
- commit protocols

Client Services
- Communications

Applications

LAN

Communications

DBMS Services

Database

Server-to-Server

- SQL interface
- programmatic interface
- other application support environments

Communications

DBMS Services

Database
Peer-to-Peer Component Architecture

Components of a Multi-DBMS
Directory Issues

- **Type**
  - Global & central & non-replicated
  - Local & central & replicated (?)
  - Global & distributed & replicated

- **Location**
  - Local & distributed & non-replicated
  - Global & distributed & non-replicated (?)
  - Local & distributed & replicated
  - Global & distributed & replicated (?)