Distributed Object Oriented Database

Distributed Data Processing(DDP)

- A number of autonomous processing elements (not necessarily homogeneous) that are interconnected by a computer network and that cooperate in performing their assigned tasks.
- A method of organizing data processing that uses a central computer in combination with smaller local computers which communicate with central as well as with one another.
- Arrangements of networked computers in which data processing capabilities are spread across the network.

Distributed Data Processing(DDP)

- Distributed data processing involves distributed databases(a database in which the data is stored across two or more computers.)

- Distributed data processing is a computer networking method in which multiple computers across different locations share computer processing capability. Computers that provide distributed data processing network are located at different locations but interconnected by means of wireless or satellite links.

Advantages of DDP

1. Lower Cost.
2. Reliable.
3. Improved performance and reduced processing time.
4. Flexible

Concept of Distributed Database

What is distributed????

- Processing logic
- Functions
- Data
- Control

Concept of Distributed Database

- A distributed database (DDB) is a collection of multiple logically interconnected database distributed over a computer network.
- A database that consists of two or more data files located at different sites on a computer network. Because the database is distributed users can access it without interfering with one another.

Concept of Distributed Database

- Homogeneous distributed databases
  - Same software/schema on all sites, data may be partitioned among sites
  - Goal: provide a view of a single database, hiding details of distribution
- Heterogeneous distributed databases
  - Different software/schema on different sites
  - Goal: integrate existing databases to provide useful functionality
- Differentiate between *local* and *global* transactions
  - A local transaction accesses data in the *single* site at which the transaction was initiated.
  - A global transaction either accesses data in a site different from the one at which the transaction was initiated or accesses data in several different sites.

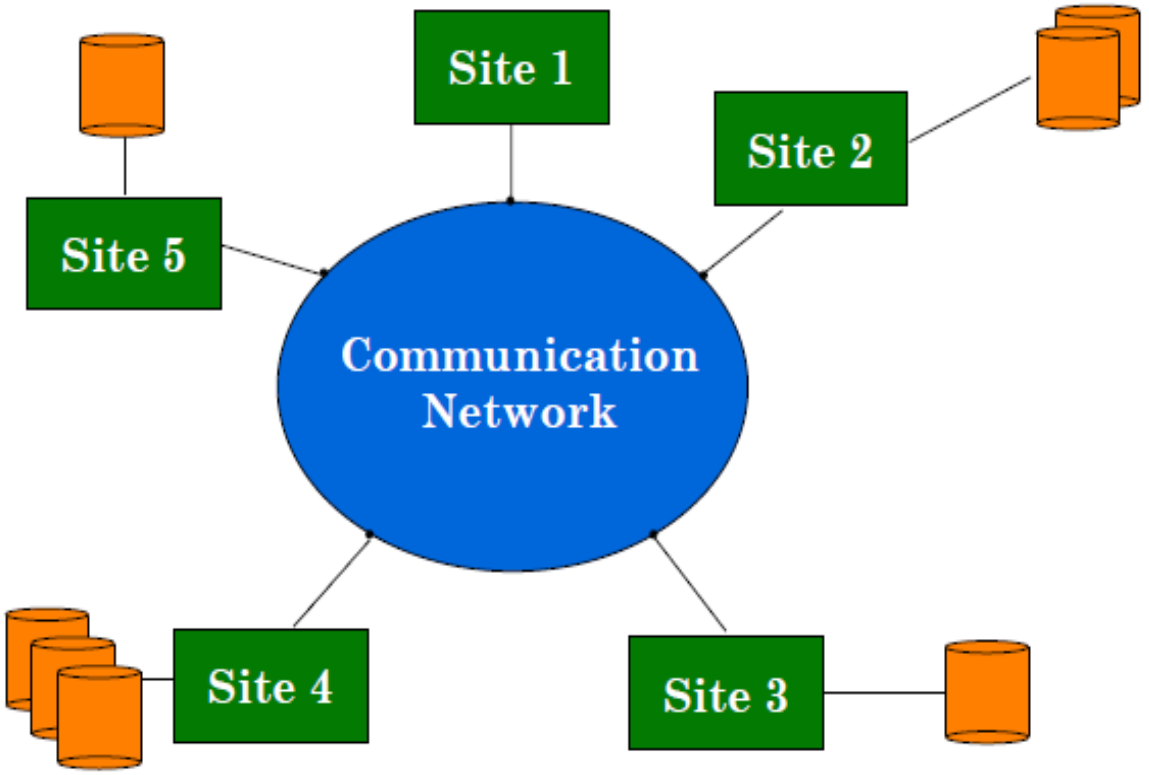
Distributed Database management System

- A distributed database management system(D-DBMS) is a software that manages the distributed database(DDB) and provides an access mechanism that makes the distribution transparent to users.

- Distributed Database System= DDB +(D-DBMS)

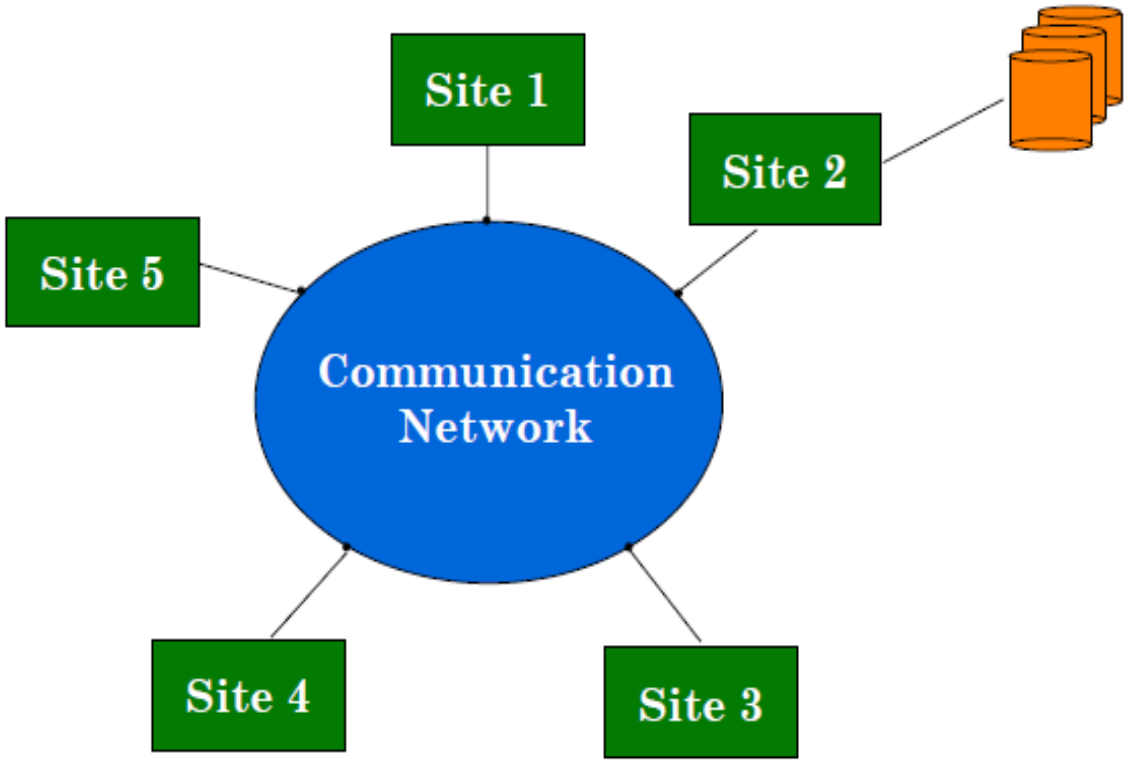
Distributed Database System

Distributed DBMS Environment



Centralized Database System

Centralized DBMS on a Network



Centralized Database System

- A database system that resides at one of the nodes on a network of computers is called as centralized database system or client server database systems.

Implicit Assumptions for Distributed System

- Data are stored at multiple number of sites Each sites logically consists of single processor.
- Processors at different sites are interconnected by a computer network no multiprocessors i.e. parallel database systems.
- Distributed database is a database, not a collection of files data logically related as exhibited in the users' access patterns.(relational data model).
- D-DBMS is a full-fledged DBMS not remote file system

Applications

1. Manufacturing - especially multi-plant manufacturing
2. Military command and control
3. EFT (Electronic Fund Transfer)
4. Corporate MIS
5. Airlines
6. Hotel chains
7. Any organization which has a decentralized organization structure

Advantages

1. Transparent management of distributed, fragmented, and replicated data
2. Improved reliability through distributed transactions
3. Improved performance
4. Easier and more economical system expansion
5. Sharing data – users at one site able to access the data residing at some other sites.
6. Autonomy – each site is able to retain a degree of control over data stored locally.
7. Higher system availability through redundancy — data can be replicated at remote sites, and system can function even if a site fails.

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 i.e.

➢ Network (distribution) transparency

➢ Replication transparency

➢ Fragmentation transparency

- ❑ horizontal fragmentation: selection
- ❑ vertical fragmentation: projection
- ❑ hybrid

Transparency

Example

EMP

ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng.
E2	M. Smith	Syst. Anal.
E3	A. Lee	Mech. Eng.
E4	J. Miller	Programmer
E5	B. Casey	Syst. Anal.
E6	L. Chu	Elect. Eng.
E7	R. Davis	Mech. Eng.
E8	J. Jones	Syst. Anal.

ASG

ENO	PNO	RESP	DUR
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E3	P3	Consultant	10
E3	P4	Engineer	48
E4	P2	Programmer	18
E5	P2	Manager	24
E6	P4	Manager	48
E7	P3	Engineer	36
E7	P5	Engineer	23
E8	P3	Manager	40

PROJ

PNO	PNAME	BUDGET
P1	Instrumentation	150000
P2	Database Develop.	135000
P3	CAD/CAM	250000
P4	Maintenance	310000

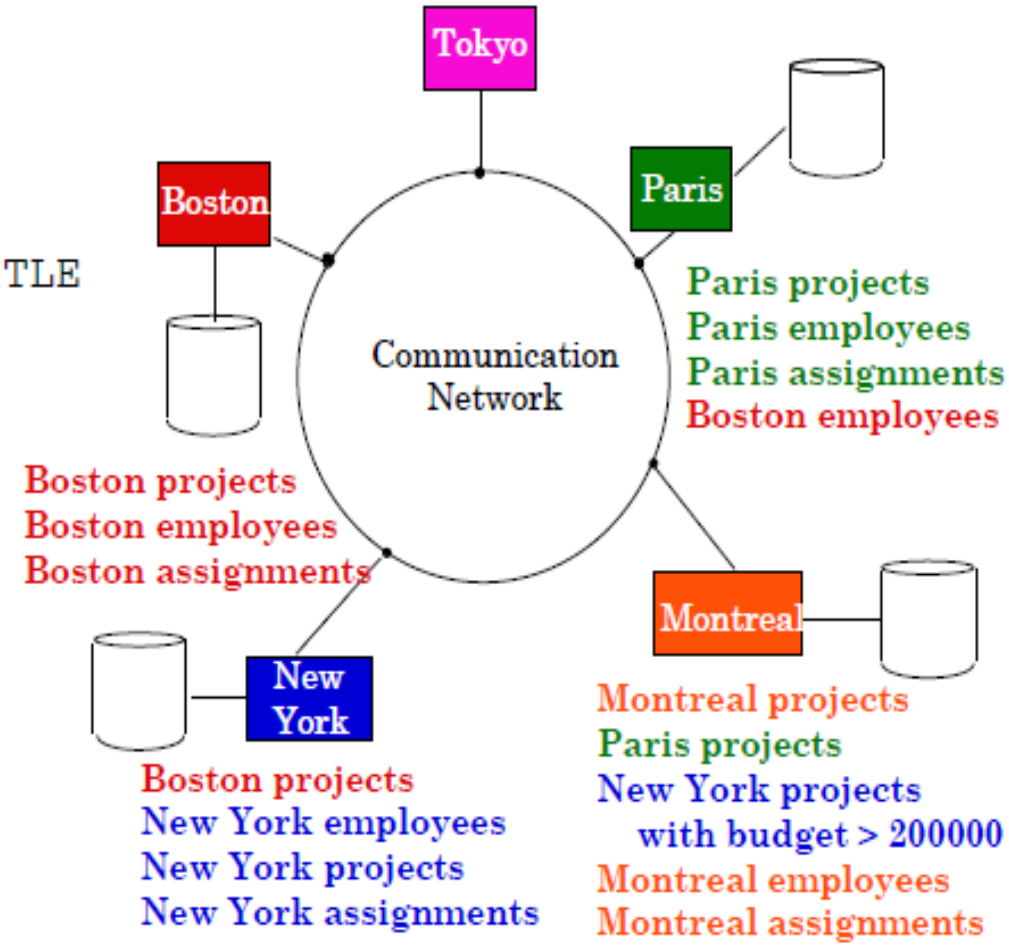
PAY

TITLE	SAL
Elect. Eng.	40000
Syst. Anal.	34000
Mech. Eng.	27000
Programmer	24000

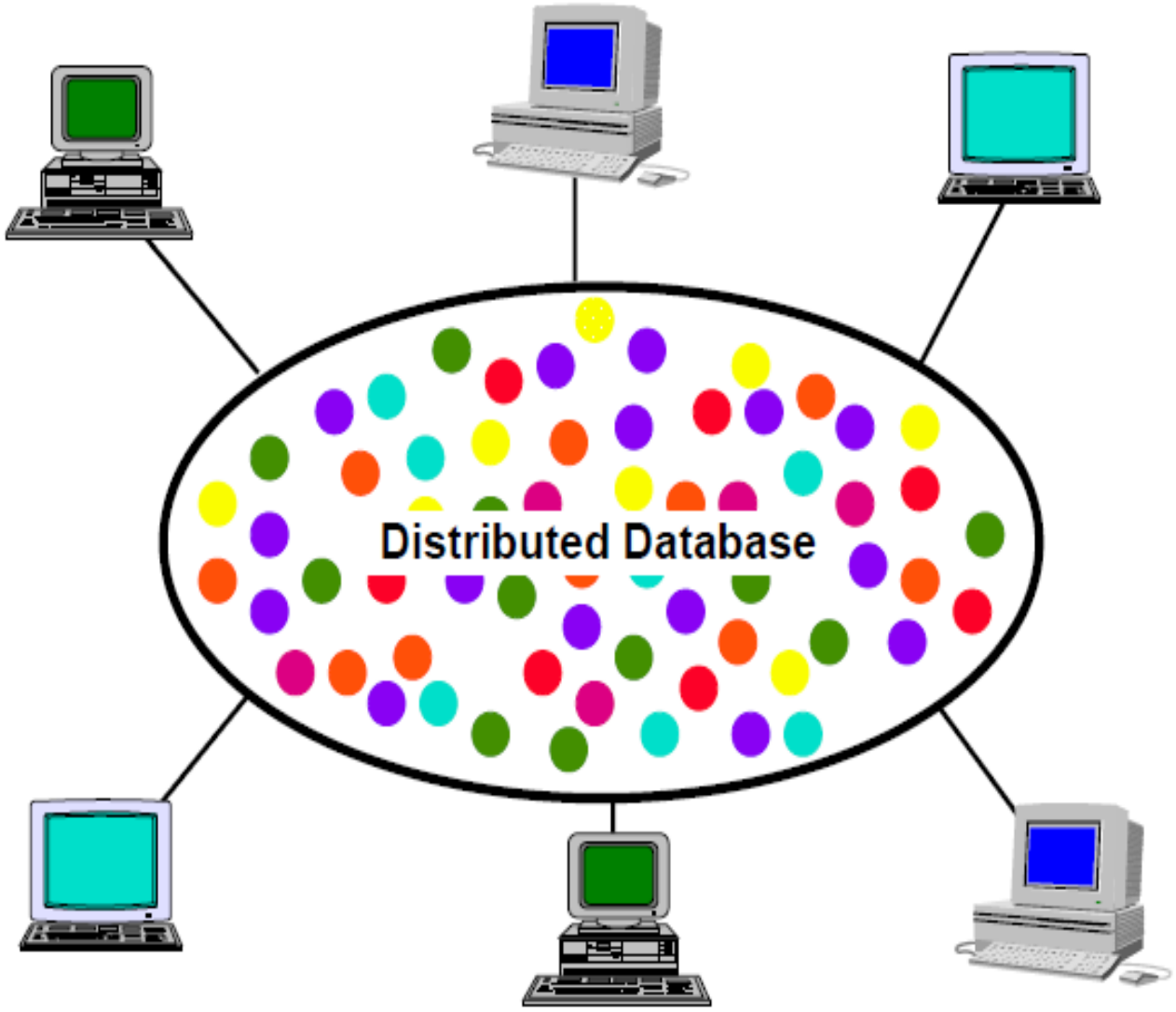
Transparency

Transparent Access

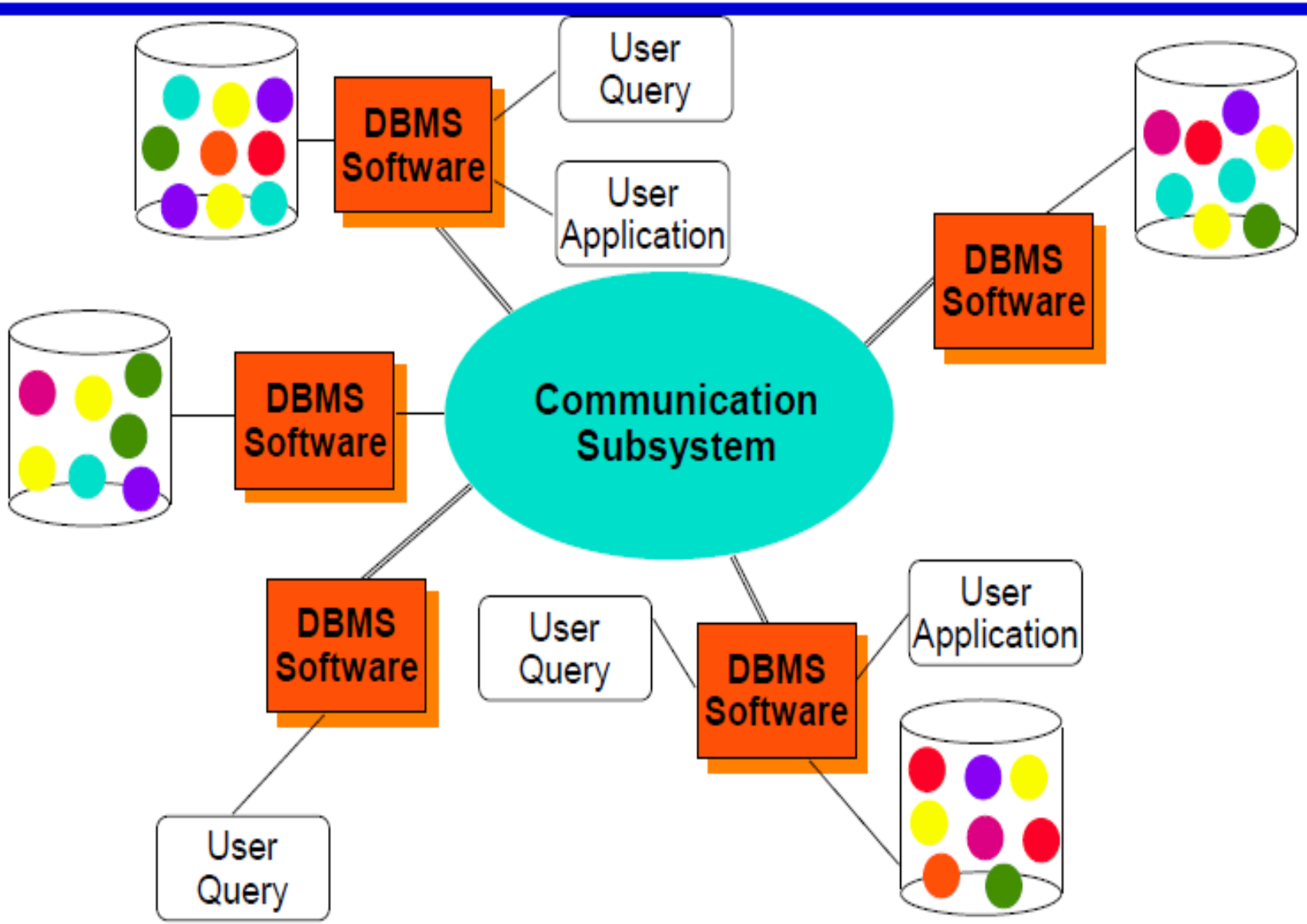
```
SELECT ENAME,SAL
FROM   EMP,ASG,PAY
WHERE  DUR > 12
AND    EMP.ENO = ASG.ENO
AND    PAY.TITLE = EMP.TITLE
```



Distributed Database - User View



Distributed DBMS - Reality



Performance

- **Proximity of data to its points of use**

  Requires some support for fragmentation and replication

- **Parallelism in execution**
  - ✓Inter-query parallelism
  - ✓Intra-query parallelism
- **Requirements for parallelism**
  - ✓Full replication
  - ✓Updates

System Expansion

- Issue is database scaling
- Emergence of microprocessor and workstation technologies
  - ✓Client-server model of computing
- Data communication cost versus telecommunication cost

Problem areas of Distributed Database

1. **Distributed Database Design**

- how to distribute the database
- replicated & non-replicated database distribution
- a related problem in directory management

2. **Query Processing**

- convert user transactions to data manipulation instructions
- optimization problem
- min{cost = data transmission + local processing}

Problem areas of Distributed Database

3. Concurrency Control

- synchronization of concurrent accesses
- consistency and isolation of transactions' effects
- deadlock management

4. **Reliability**

- how to make the system resilient to failures
- atomicity and durability

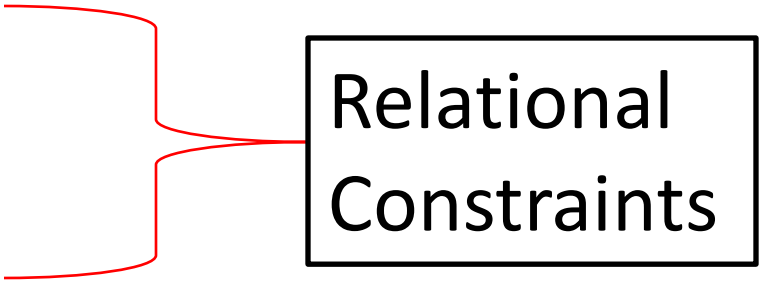
Problem areas of Distributed Database

5. Software development cost.
6. Greater potential for bugs.
7. Increased processing overhead.

Integrity Constraints in Distributed Database

A distributed database supports basically four types of integrity constraints .They are

1. Data type constraints
2. Relation constraints
3. Referential constraints
4. Explicit constraints



1. Data type Constraints

- Data type constraints are semantic integrity rules that specify the data type for columns of relational tables.
- Some relational database systems may allow specification of user-defined data types.
- A data type constrains the range of values and the type of operations that we can apply to the column to which the data type is attached.

1. Data type Constraints(Domain)

- System defined data type

  Date, Integer, Decimal (X,Y), Float, Char(X),

  Varchar(X), Enumerated data types such as ('M', 'F') or ('Sat', 'Sun', 'Mon', 'Tue','Wed', 'Thu', 'Fri'),

- User defined data type

```
create type salary as object(salary number(8,2));
create table employee
                    (name varchar2(20) not null,
                     sal salary not null,
                     age  number(3) );
```

- Create table Employee (

  Empid integer not null,

  Name varchar(20) not null,

  Emp_Nickname char(10),

  DOB date check (DOB LIKE ‘MM-DD-YYYY’),

  Sex char(1) in (‘M’, ‘F’),

  Primary key (Empid),

  Unique Emp_Nickname);

2. Relation Constraints

- The table name “Employee” is unique within a database. This is called a relation constraint and is inherited from the fact that in the ERM there is a unique entity type called “Employee.”
- The Employee table has five columns: Empid, Name, Emp−Nickname, DOB, and Sex. Each column has a **data type constraint**, may have a **null or not null constraint**, may have an **enumeration constraint**, and may have a **range constraint**.

2. Relation Constraints

- The **Empid** column must be an integer and must have a value(domain constraint).
- The **Name** column is of type varchar2(20) and must have a value. Since “varchar2(20)” is used as the data type, the Name cannot be longer than 20 characters. If the Name is shorter than 20 characters, then the actual length of the name will be used.
- The **DOB** column is of type Date. This column may be left as null. If the column has a value for date it has to follow the format ‘MM-DD-YYYY’. Otherwise, the row is not validated.
- The **Sex** column is a one-character field and can be left as null. The Sex column has an enumeration constraint on it. This constraint limits the value of the column Sex to either character ‘M’ or ‘F’. Any other value is a violation of this constraint.

2. Relation Constraints

- The primary key constraint forces the **Empid** column to be unique. If the value of this column is not unique, the employee is not validated.
- In addition to the **Empid**, which is used as a primary key and is unique, each Employee row's nickname is also unique. Since a table cannot have two primary keys, the uniqueness of Emp−Nickname is enforced by a different type of constraint in the table, known as a unique constraint.

3. Referential Integrity Constraints

- Referential integrity (RI) constraints restrict the values stored in the database so that they reflect the way the objects of the real world are related to each other.
- Referential Integrity is implemented by controlling the way the rows in one table relate to the row(s) in another table. Referential Integrity in relational systems forces the foreign key values in one table to correspond to a primary key value in another table.

3. Referential Integrity Constraints

- ✓ Create table department (

  Deptid number(5) primary key,

  Name varchar2(30) not null,

  Budget number(10,2) not null);

- ✓ Create table employee (

  Empid number(5) primary key,

  Name varchar2(50) not null,

  Deptid number(5),

  Foreign key(**deptid**) references department(**deptid**)

  );

4.Explicit Constraints

- Some commercial databases, like Oracle, do not support the concept of **assertions**. (Assertions are constraints that are not associated with any individual table).
- For these products, explicit semantic integrity constraints have to be used.
- Explicit constraints are not inherited from the ERM like the other three constraints.
- These constraints are coded in the application programs or database using stored procedures or triggers.

4.Explicit Constraints

- **Stored procedures and triggers** are written programs that use SQL and extensions to the standard SQL.
- These programs are stored in the database as opposed to being compiled and stored as object code outside the database.
- The only difference between a **stored procedure** and other procedures is that stored procedures cannot request any user interaction once they have started.
- Create Procedure Proc_Name (Param_list)

  AS

  Declaration section;

  Begin

  Actions;

  End Proc_Name;

4.Explicit Constraints

- A **trigger** is a special kind of stored procedure that is stored in the database. Triggers are an extension to the standard SQL.
- The only difference between a trigger and a stored procedure is that users cannot call a trigger—only the DBMS can do this.
- Triggers are run automatically (also known as fired) by the DBMS when certain changes (such as insert, delete, and update operations) are performed on tables of the database.

4.Explicit Constraints

```
Create TRIGGER Trigger_Name ON Table FOR operation AS
        Declaration section;
Begin
        If Trigger_Condition Exec Proc_Name (Param_list);
        . . .
End Trigger_Name;
```