UNIT 16

Distributed Database

Contents

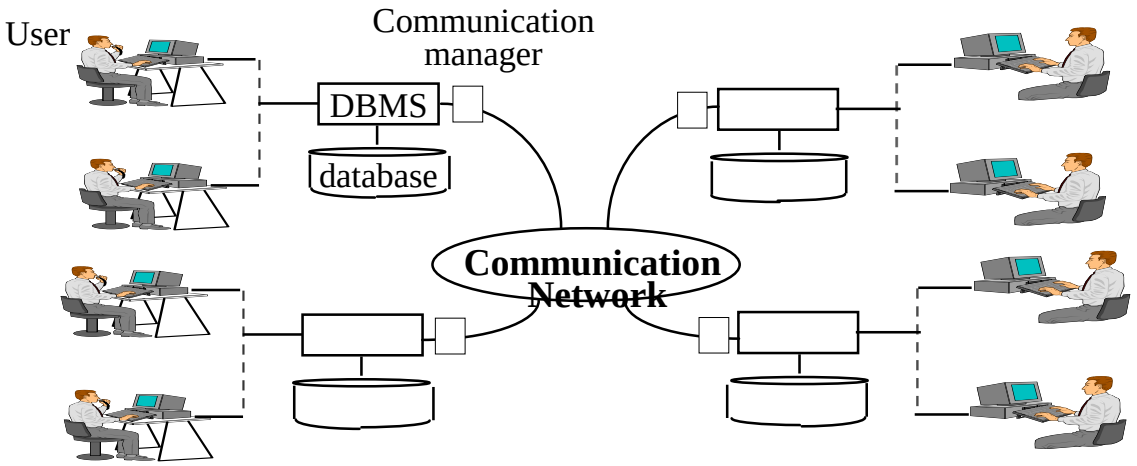


Fig 16.1: A typical distributed database system

16.1 Introduction

Distributed Database System

- A system involving multiple sites connected together via communication network.
- User at any site can access data stored at any site.
- Each site is a database system in its own right: its own local database, local users, local DBMS, local DC manager.

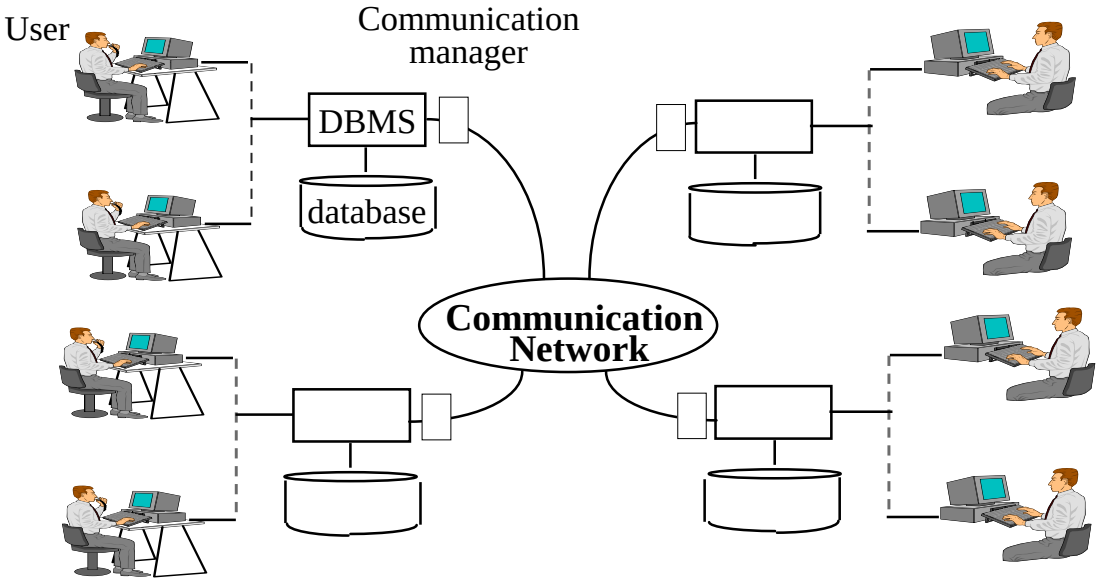


Fig 16.1: A typical distributed database system

Distributed Database System (cont.)

- Assumptions
  - The system is **homogeneous**, in the sense that each site is running its own copy of the same DBMS.
    (can be relaxed => **heterogeneous** DBMS)
  - The communication network is **slow**: component sites are geographically dispersed.
- Advantages
  - Enables the structure of database to mirror that of the enterprise.
  - Combines efficiency of processing with increased accessibility.
- Disadvantage
  - Complex for implementation

Distributed Database System (cont.)

- Sample Systems
  - Prototypes
    - SDD-1: Computer Corporation of America (CCA), late 1970s and early 1980s.
    - R*: IBM, early 1980s
    - Distributed INGRES: Berkeley, early 1980s.
  - Commercial Products :
    - INGRES / STAR: Relational Technology Inc.
    - SQL*STAR: Oracle Corp.
    - DB2 version 2 release 2: IBM
    - SQL server: Microsoft

- A Fundamental Principle

  To the user, a **distributed system** should look exactly like a **non-distributed system**.

16.2 The Twelve Objectives

The Twelve Objectives

1. Local Autonomy

- all operations at a given site are controlled by that site, should not depend on other sites.
- local data is locally owned and managed.
- Not wholly achievable => sites should be autonomous to the maximum extend possible.

2. No Reliance on a Central Site

- all sites must be treated as equals.
- the central site may be bottleneck.

3. Continuous Operation

- Reliability
- Availability
- Never require the system to be shutdown to perform some function: e.g. add a new site.

The Twelve Objectives (cont.)

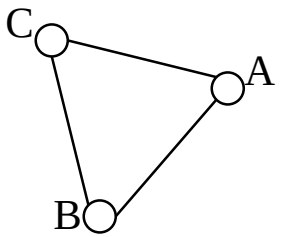
4. Location Independence (Location Transparency)

- user should not need to know at which site the data is stored, but should be able to behave as if the entire database were stored at their own local site.
- a request for some remote data => system should find the data automatically.
- Advantages

<1> Simplify user programs and activities

<e.g.>

```
SELECT  S#
FROM    S
AT SITE A
WHERE   SNAME = 'John'
```



<2> allow data to be moved from one site to another at any time without invalidating any program or activities.

The Twelve Objectives (cont.)

5. Fragmentation Independence (Fragmentation Transparency)

- Data Fragmentation
  - a given local object can be divided up into pieces (**fragments**) for physical storage purpose.

<e.g.> user perception

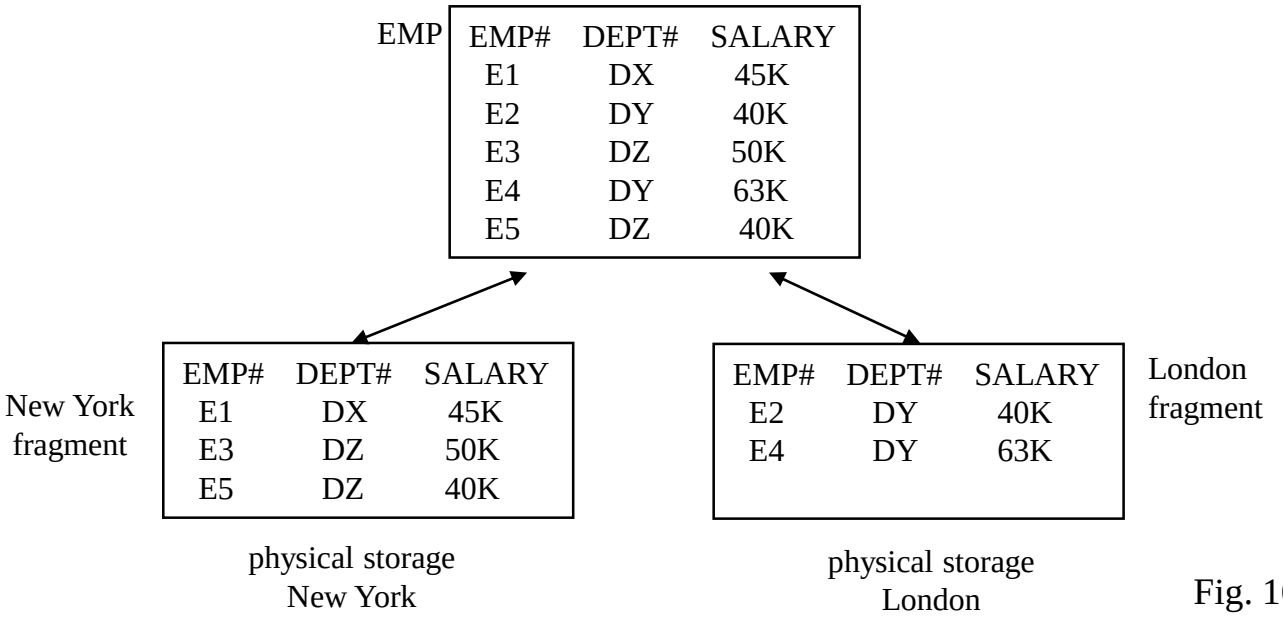


Fig. 16.2: An example of fragmentation.

The Twelve Objectives (cont.)

- Data Fragmentation
  - A fragmentation can be any subrelation derivable via restriction and projection (with primary key).
  - Advantage: data can stored at the location where it is most frequently used.
- **Fragmentation independence**
  - user should be able to behave as if the relations were not fragmented at all.
  - one reasons why relational technology is suitable for DBMS.
  - user should be presented with a view of data.

    => system must support updates against join and union views.
  - Advantages

    (1) simplify user program and activity.

    (2) allow data to be re-fragmented at any time.

The Twelve Objectives (cont.)

6. Replication Independence (Replication Transparency)

- Data Replication

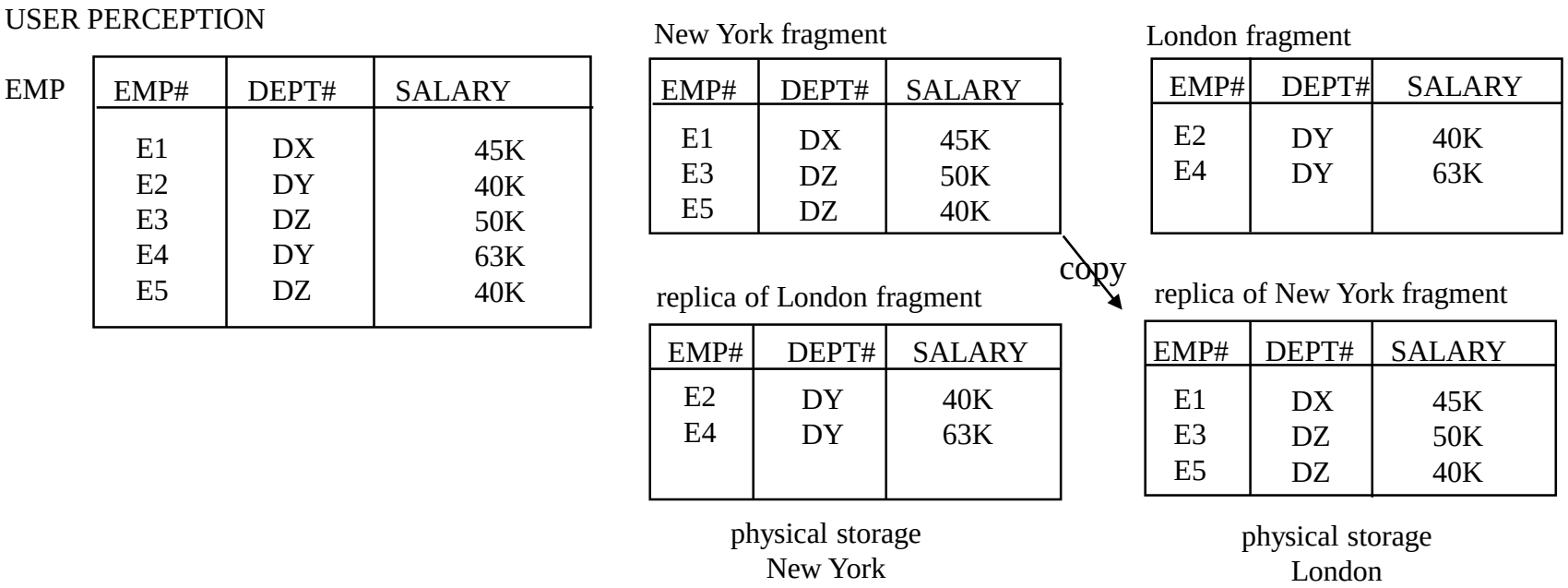


Fig. 16.3: An example of replication.

The Twelve Objectives (cont.)

- Data Replication
  - A given fragment of relation can be represented at the physical level by many distinct copies of the same object at many distinct sites.
  - Unit of replication: fragment (may not a complete relation)
  - Advantage: better performance and availability
  - Disadvantage: update propagation problem.
- **Replication Independence**
  - User should be able to behave as if the data is not replicated at all.
  - Advantages

    (1) simplify user programs and activities.

    (2) allow replicas to be created and destroyed dynamically.

The Twelve Objectives (cont.)

7. **Distributed Query Processing**
   - message transfer cost
   - optimization
8. **Distributed Transaction Management**
   - concurrency control
   - recovery control
9. **Hardware Independence:** IBM, DEC, HP, PC, ...
10. **Operating System Independence**: VMS, UNIX, ...
11. **Network Independence**: BITNET, INTERNET, ARPANET, ...
12. **DBMS Independence:** Relational, hierarchical, network, ...
    - distributed system may be heterogeneous.

16.3 Problems of Distributed Database Systems

Basic Point: Network are slow !

Basic point: network are slow !

⇩

Overriding Objective : *minimize the number and volume of messages.*

⇩

Give rise to the following problem

- Query Processing
- Update Propagation
- Concurrency
- Recovery
- Catalog Management

⋮

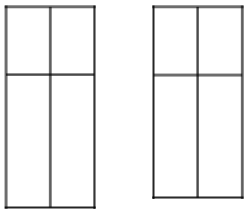
Query Processing: Example

- Query Optimization is more important in a distributed system.
- Example (Date, Vol.2 p.303)
  - Database:

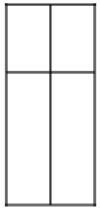
S (S#, CITY) 10,000 tuples, stored at site A.
P (P#, COLOR) 100,000 tuples, stored at site B.
SP (S#, P#) 1,000,000 tuples, stored at site A.

Assume each tuple is 100 bits long.

Site A: S SP



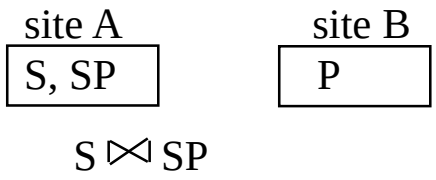
Site B: P



Query Processing: Example (cont.)

- Query: "Select S# for London suppliers of Red Parts"

```
SELECT    S.S#
FROM      S, P, SP
WHERE     S.CITY =  "London"
            AND  S.S# = SP.S#
            AND  SP.P# = P.P#
            AND P.COLOR =  'Red'
```



- Estimates

  # of Red Parts = 10

  # of Shipments by London Supplier = 100,000

- Communication Assumption :

  Data Rate = 10,000 bits per second

  Access Delay = 1 second

- T[***i***] = total communication time for strategy ***i***

  = total access delay + total data volume / data rate

  = (# of messages * 1 sec) + (total # of bits / 10,000) sec.

Query Processing: Example (cont.)

- Strategy 1
  1. Join S and SP at site A
  2. Select tuples from (S ⋈ SP) for which city is 'London' (100,000 tuples)
  3. For each of those tuple, check site B to see if the part is red. (2 messages: 1 query, 1 response)

  T[1] = (100,000 * 2) * 1 = 2.3 days

- Strategy 2

  Move relations S and SP to site B and process the query at B.

  T[2] = 2+(10,000+1,000,000)*100/10,000 = 28 hours

- Strategy 3

  Move relation P to site A and process the query at A

  T[3] =1+(100,000*100) /10,000 = 16.7 min

Query Processing: Example (cont.)

- Strategy 4
  1. Select tuples from P where color is red. (10 tuples)
  2. Check site A to see if there exists a shipment relating the part to a London Supplier. (2*10 messages)

     T[4] = 2*10*1 = 20 sec

- Strategy 5
  1. Select tuples from P where color is red (10 tuples)
  2. Move the result to site A and complete the processing at A.

     T[5] = 1 + (10*100) / 100,000 = 1.01 sec

- *Note*: Each of the five strategies represents a plausible solution, but the variation in communication time is enormous.

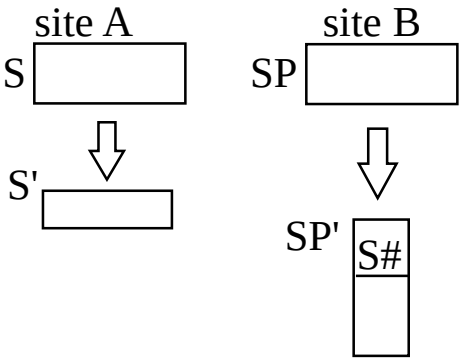
Query Processing: Semijoin

- Semijoin: (used in SDD - 1) Ref. p.529 [18.15] p.626 [21.26]

  $A \ltimes B$

  $A \ltimes B \ltimes \ldots$

<e.g.>



- Database :

  S: 1,000 tuples, at site A
  SP: 2,000 tuples, at site B
  # of tuples in S where S.S#=SP.S#: 100,
  length of a S tuple: 100 bit
  length of a SP tuple: 100 bit
  length of the S# field: 10 bit

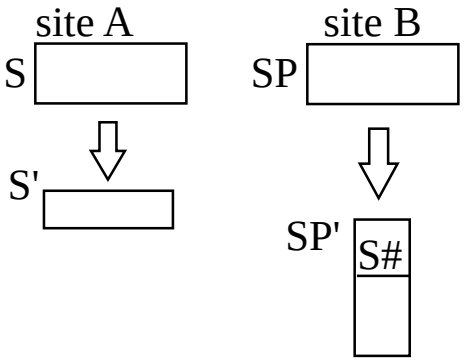
- Regular Join:

  <1> Ship S to site B (1000 * 100 bits)
  <2> Join S and SP at site B

  communication time = 1 + 1000*100/10000 = 11 sec

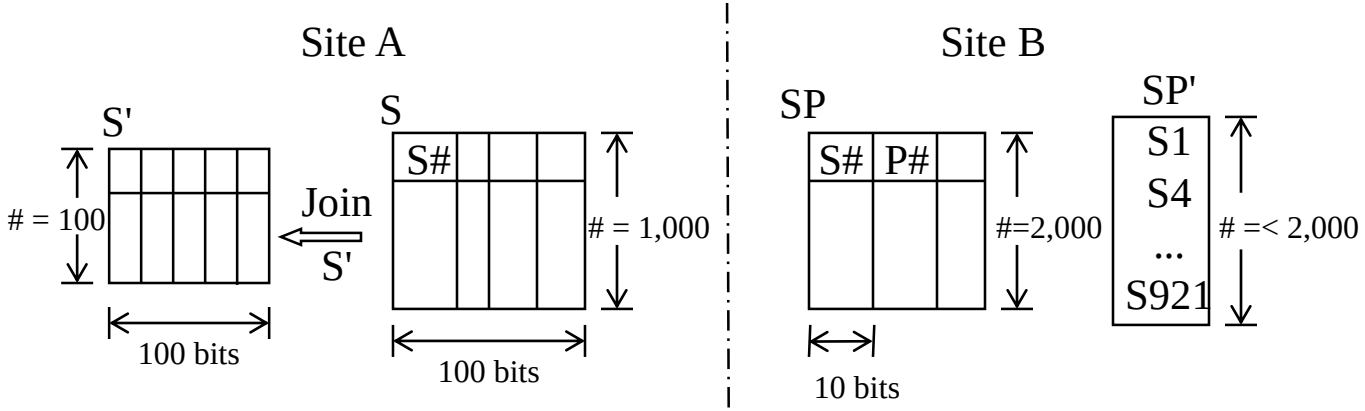
Query Processing: Semijoin (cont.)

- Semijoin



<1> site B: step 1. Project SP on S# (get SP')
step 2. ship to site A
<2> site A: step 3. Join the projection of SP' on S# with S
step 4. The result S‘, ship to site B
<3> site B: step 5. Join S' with SP

communication time = 1+10*2000/10000+1+100*100/10000
= 1+2+1+1= 5 sec



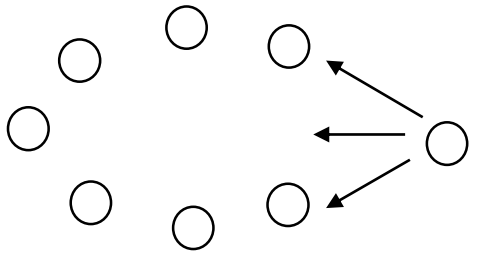
Update Propagation

- Basic problem with data replication
  - An update to any given logical data object must be propagated to all stored copies of that object.
  - some sites may be unavailable (because of site or network failure) at the time of the update
    => Data is less available !
- A possible Solution: Primary Copy (used in distributed INGRES)
  - one copy of each object is designated as the *primary copy*.
  - primary copies of different objects will generally be at different sites.
  - Update Operation
    1. Complete as soon as the primary copy has been updated.
    2. Control is returned and the transaction can continue execute.
    3. The site holding the primary copy broadcasts the update to all other sites.
  - Further Problem: violation of the local autonomy objective.

Concurrency

- Most distributed systems are based on locking .
- Requests to *test*, *set* and *release* locks are *messages* ⟶ overhead.

  <e.g.>: Consider a transaction **T** that needs to update an object for which there exists replicas at n remote sites.



5n :

n lock requests
n lock grants
n update messages
n acknowledgments
n unlock requests

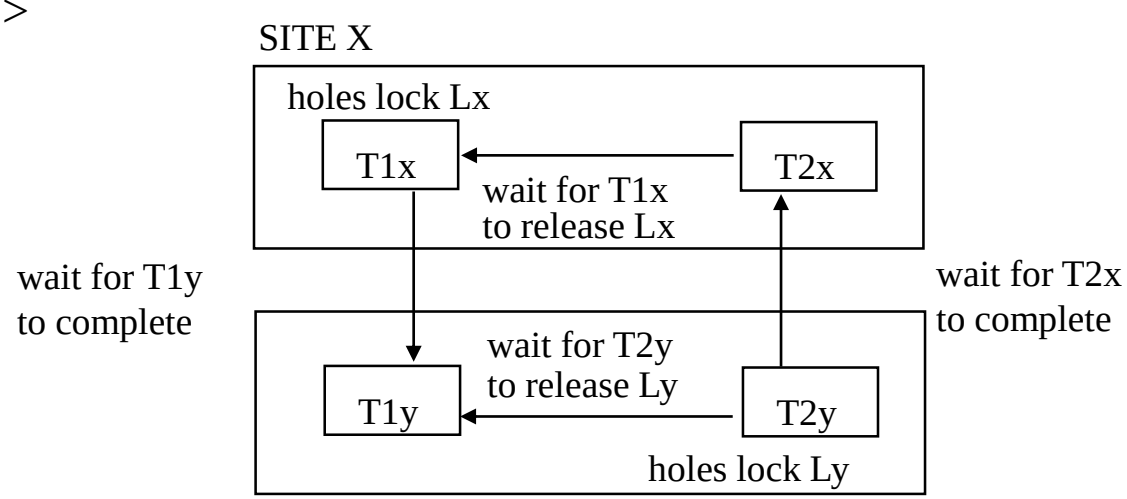
  - Several orders of magnitude greater than in a centralized system.
- Solution
  - adopt the primary copy strategy.
  - the site holding the primary copy of X handles all locking operations involving X.
  - 1 lock request, 1 lock grant, n updates, n ack, and 1 unlock request (2n+3<=5n).
  - Problem : loss of local autonomy.

Concurrency (cont.)

- **Global Deadlock Problem**
  - Neither site can detect it using only information that is internal to that site.

    i.e. no cycles in two local wait-for-graph, but a cycle in the global.

<e.g.>



  - Global deadlock detection needs further communication overhead.

Catalog Management

- Contents of catalog:
  - not only data regarding relations, indexes, users, etc,
  - but also all the necessary control information for independence.
- Where and How ?
  1. **Centralized:** violate no reliance on a central site.
  2. **Fully replicated**: loss of local autonomy.
  3. **Partitioned**: nonlocal operations are very expensive.
  4. **Combination** of (1) and (3): violate no reliance on a central site.

16.4 Gateways

Gateways

- DBMS independence
- Suppose INGRES provides a "gateway", that "making ORACLE look like INGRES"

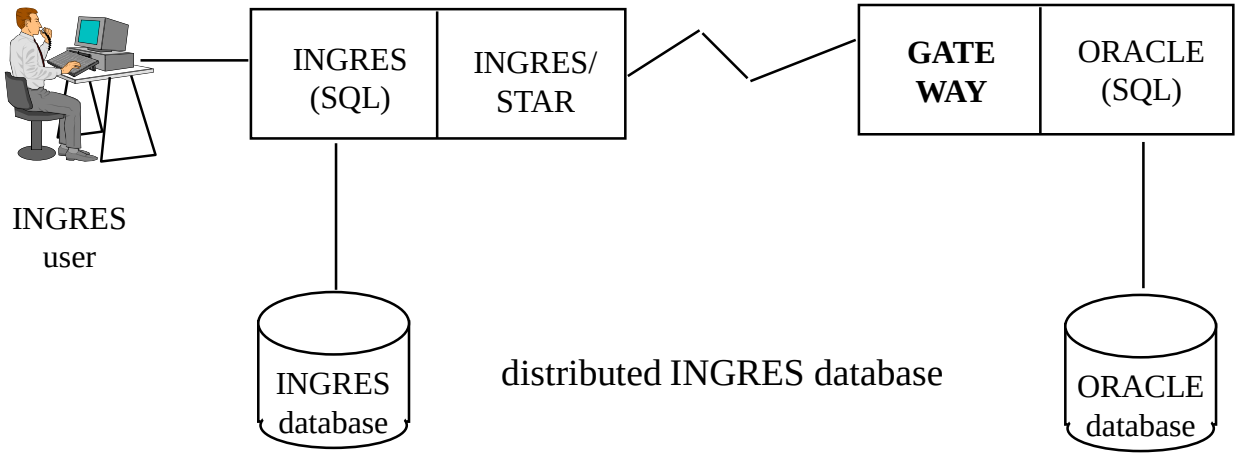


Fig. 12.4: A hypothetical INGRES-provided gateway to ORACLE

16.5 Client/Server Systems

Client/Server Architecture

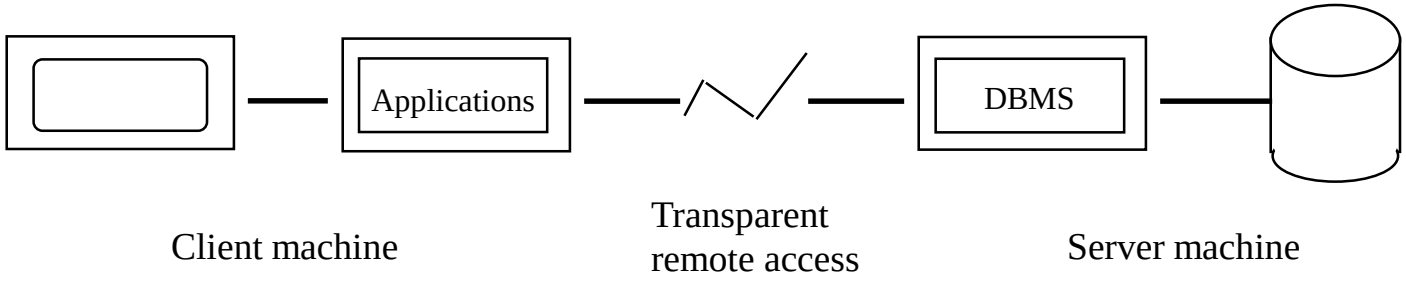


Fig. 16.5: A client/server system

- some sites are client, and others are server sites
- a great deal of commercial products
- little in "true" general-purpose distributed system (but long-term trend might be important)
- client: application or front-end
- server: DBMS or backend
- Several variations

Client/Server Systems

- **Client/Server Standards**
  - SQL/92 standard
  - ISO (International Organization for Standardization)
  - Remote Data Access standard: RDA.
  - Distributed Relational Database Architecture standard: DRDA.
- **Client/Server Application Programming**
  - Stored procedure: a precompiled program that is stored at the server site.
  - invoked from the client by a remote procedure call (RPC).
  - one stored procedure can be shared by many clients.
  - optimization can be done at the precompiled time instead of at run time.
  - provide better security.
  - but, no standards in this area.

end of unit 16