

# Session 10

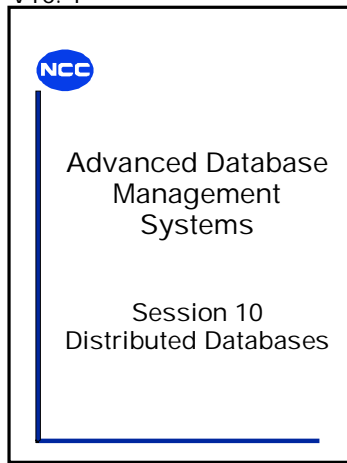
## Distributed Databases

---

### 1 Introduction

(5 minutes)

V10.1



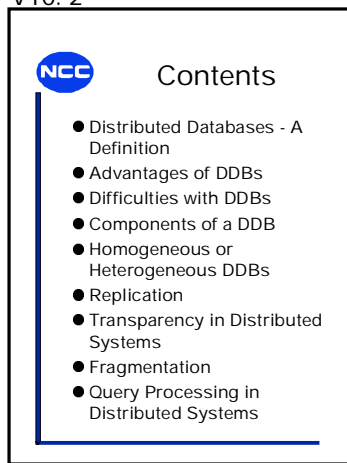
This is a very broad topic and it will only be possible in this session to present some of the fundamental ideas and concepts.

Distributed databases (DDBs) bring many potential advantages but due to their size and complexity there can be severe problems with this technology.

*Inform students that a handout containing a full set of visuals will be provided to them at the end of this lecture.*

### 1.1 Summary of Topics to be Covered

V10.2



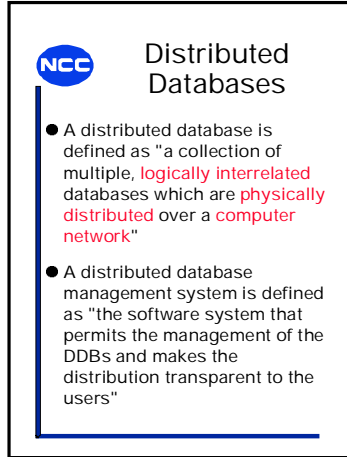
The topics detailed in the visual will be discussed during this session.

## 2 Distributed Databases - A Definition

(5 minutes)

“A *distributed database* is a collection of multiple, logically interrelated databases which are physically distributed over a computer network.”

V10.3



**NCC** Distributed Databases

- A distributed database is defined as "a collection of multiple, **logically interrelated** databases which are **physically distributed** over a **computer network**"
- A distributed database management system is defined as "the software system that permits the management of the DDBs and makes the distribution transparent to the users"

In visual V10.3, the words *logically interrelated* are highlighted to emphasise the fact that it only makes sense to group together data in a distributed fashion if there is some relationship between them.

The words *physically distributed* are highlighted to emphasise the fact that two or more separate sites is usual. This will of course have advantages and disadvantages.

The words *computer network* are highlighted to emphasise the fact that obviously some type of network technology will have to be utilised in this area. Again there will be advantages and disadvantages to this.

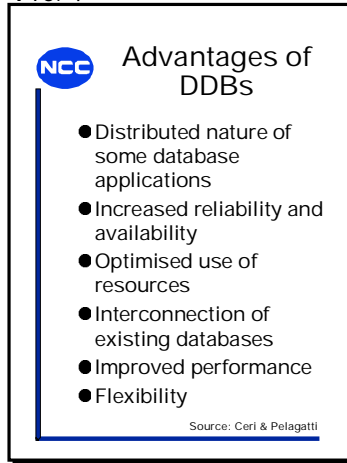
“A *distributed database management system* is defined as the software system that permits the management of the DDBs and makes the distribution transparent to the users.”

This is simply to clarify that even in the distributed environment a DBMS is necessary for management purposes. It shares many of the aspects of a centralised DBMS, but has to have additional facilities such as making the use of the DDB transparent to the users.

### 3 Advantages of DDBs

(10 minutes)

V10. 4



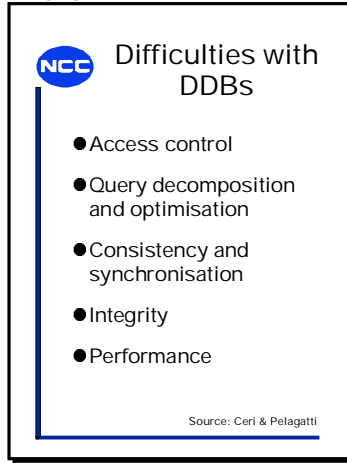
From Ceri & Pelagatti:

- *Distributed nature of some database applications:* Many database applications are naturally distributed over some different locations. For example, a company may have locations in different cities.
- *Increased reliability and availability:* These are two of the most common potential advantages cited for DDBs. Reliability is broadly defined as the probability that a system is up at a particular moment in time, whereas availability is the probability that the system is continuously available during a time interval. In a distributed environment, one site may fail while other sites continue in operation - this improves both reliability and availability.
- *Optimised use of resources:* Load sharing has commonly been quoted as a major advantage of resource-sharing computer networks. DDBs with replicated data offer the opportunity for sharing the immediate processing load across a number of sites.
- *Interconnection of existing databases:* DDBs are the natural solution when several databases already exist in an organisation and the necessity of performing global applications arises. In this case, the distributed database is created bottom-up from the pre-existing local databases.
- *Improved performance:* By distributing a large database over multiple sites, smaller databases will exist at each site. Local queries and transactions accessing data at a single site will potentially demonstrate better performance because of having to process smaller local databases.
- *Flexibility/Deployability:* Implementation timescales often can be minimised by installing computer systems in parallel rather than having to rely on the implementation of one or two large-scale computer centres.

## 4 Difficulties with DDBs

(10 minutes)

V10.5



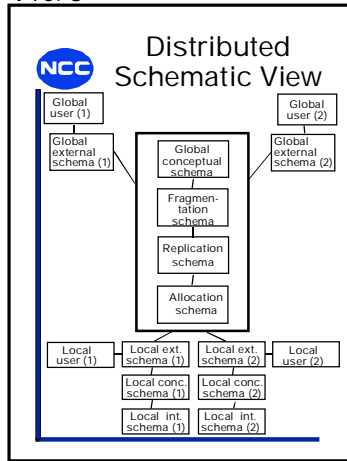
From Ceri & Pelagatti:

- *Access control:* This is more difficult to implement because of the necessity of deciding where to locate the control mechanism. The vulnerability of dispersed data may become an issue and it may be necessary to locate access control with each distributed component which may be costly.
- *Query decomposition and optimisation:* Users may often express access requests which require the synthesis of data which has been retrieved from many local DBMSs. Because of the communication costs in assembling data from many remote sites the aim of a distributed DBMS should be to minimise these internode accesses.
- *Consistency and synchronisation:* Because of the inherent nature of communication networks it is important that it can be asserted that transactions cause consistent updates when dealing with multiple node accesses for example. Means have to be found of synchronising the accesses and of detecting when inconsistency may result.
- *Integrity:* An integral part of a DBMS's objective is the ability to have continuous operation. This implies provisions for the recovery of data and the restart of operations. In a distributed environment the task becomes more difficult when applied to a partitioned, non-redundant distribution and when considered in conjunction with the synchronisation issue.
- *Performance:* If the implemented system cannot perform efficiently enough to meet response requirements, the service goals are not met and the system fails. There is always the risk that the communication overhead incurred in the interests of system integration and control will degrade performance sufficiently for the user to question the responsiveness of the system as a whole.



## 5.2 Distributed Schematic View

V10. 8



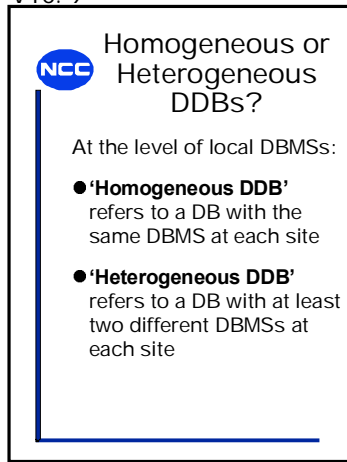
From Ceri &amp; Pelagatti:

The global schema defines all the data which are contained in the distributed database as if the database were not distributed at all. So, at the top level the distributed database acts conceptually as a single centralised database, and the global conceptual schema, therefore, represents to the end user a unified view of data for the complete distributed system. This schema description is the union of each local external schema description. A class of end users can be envisaged which interact with this global data model through a global schema. External schemas use subsets of conceptual schemas and a global external schema is, therefore, a particular global user of the global conceptual schema. Note that in a non-distributed database, external schemas use subsets of the conceptual schema whilst in the distributed environment local external schemas describe subsets of the global conceptual schema. Thus these mappings are in opposite directions.

## 6 Homogeneous or Heterogeneous DDBs

(10 minutes)

V10. 9



From Draffon &amp; Poole:

An important property of DDBs is whether they are homogeneous or heterogeneous. Homogeneity and heterogeneity can be considered at different levels in a distributed database: the hardware, the operating system, and the local DBMSs.

However, the important distinction for us is at the level of local DBMSs, because differences at lower levels are managed by the communication software.

Therefore, in the following, the term *homogeneous DDB* refers to a DB with the same DBMS at each site, even if the computers and/or the operating systems are not the same. Note that this situation is not uncommon, because the same vendor often provides the same DBMS on several computers and because software houses produce DBMSs which run on different computers of different vendors.


A *heterogeneous DDB* uses instead at least two different DBMSs. Heterogeneous DDBs add the problem of translating between the different data models of the different local DBMSs to the complexity of homogeneous systems. For this reason, if the development of a distributed database is performed top-down without a pre-existing system, it is convenient to develop a homogeneous system. However, in some cases the motivation for creating a distributed database is the necessity of integrating several pre-existing databases; in which case it is necessary to develop a heterogeneous DDB, capable of building a global view of the database.

## 7 Replication

(15 minutes)

Among the desirable properties of DDBs is the ability to have a local repository of frequently used data, while still being able to access data stored at other network sites.

V10. 10

 Replication

- Stored data is duplicated at various sites
- Full replication occurs when all occurrences are allocated to all sites
- Has a different effect on read-only and update operations

*Replicated database* - is a distributed database where stored data is duplicated at various sites.

*Fully replicated database* - a replicated database where all occurrences are allocated to all sites.


Replication has a different effect on read-only and update applications. Read-only applications take advantage of replication, because it is more likely that they can reference data locally.

For update applications replication is not convenient since they must update all copies in order to preserve consistency.

Replicated data enhances locality of reference by satisfying more read requests locally. This reduces the response time for a request and reduces traffic on the communication network. Furthermore, replicated data provides greater reliability through backup copies from which data can be recovered in the event of media failures.

## 7.1 Replication - Update Strategies

V10. 11



**Replication Update Strategies**

- Unanimous Agreement Update Strategy
  - updates refused unless they have unanimous acceptance
- Single Primary Update Strategy
  - update requests issued to primary replica, which serialises all updates

From Ozsu & Valduriez:

*Unanimous Agreement Update Strategy* - So far, only read requests have been discussed. It is with write requests that having replicas becomes expensive. Most proposals about replicated data have as their goal the presentation to the end user of a single copy image of data. One of the easiest designs to reflect a single copy image of replicated files is to propagate updates to all replicas immediately. Unanimous acceptance of the proposed update by all sites having replicas is necessary in order to make a modification, and all of those sites must be available for this to happen. Updates are refused unless they have unanimous acceptance.


*Single Primary Update Strategy* - Many solutions differ from the above and attempt to improve the success of an update taking place.

The first such solution is to designate one replica as *primary* and the remaining replicas as *secondaries*. Update requests are issued to the primary replica, which serves to serialise updates and thereby preserve data consistency. Under this scheme, the secondaries diverge temporarily from the primary. After having performed the update, the primary will broadcast it to all the secondaries at some later time. There are different proposals for this broadcast, some proposals call for the update requests to be sent to the secondaries *immediately*, others package updates and broadcast them at the end of the transaction. Still others broadcast updates at only specified intervals - once an hour, overnight, *etc.*

## 8 Transparency in Distributed Systems

(10 minutes)

V10. 12



**Transparency in Distributed Systems**

- Important to hide the details of data distribution from the user
- Types of transparency:
  - Distribution
  - Replication
  - Fragmentation

From Ceri & Pelagatti:

An important aspect of a distributed environment is to hide the details of data distribution from the user.

*Distribution Transparency*: the user should write global queries and transactions as though the database were centralised, without having to specify the sites at which the data referenced in the query or transaction resides. This property is called distribution transparency.



*Replication Transparency:* Assuming that data is replicated, the issue related to transparency that needs to be addressed is whether the users should be aware of the existence of copies or whether the system should handle the management of copies and the user should act as if there is a single copy of the data (note that we are not referring to the placement of copies, only their existence).

*Fragmentation Transparency:* When database objects are fragmented, we have to deal with the problem of handling user queries that were specified on entire relations but now have to be performed on subrelations. In other words, the issue is one of finding a query processing strategy based on the fragments rather than the relations, even though the queries are specified on the latter.

## 9 Fragmentation

(30 minutes)

V10. 13

**NCC** Fragmentation

... is basically the dividing of relations for distribution

- Advantages:
  - improves access
  - permits concurrent transaction processing
  - results in parallel execution of queries
- Disadvantages:
  - performance degradation
  - inhibits integrity checking

From Ullman:

Before we decide how to distribute the data, we must determine the logical units of the database that are to be distributed. The simplest of these are the relations themselves. However, in many cases a relation can be divided into smaller logical units for distribution. To do this we need to partition each relation using a technique called fragmentation.

### 9.1 Advantages

- Since application views are usually based on only parts of an entire relation, it makes sense in a distributed environment to only deal with subrelations.
- If the relation can be decomposed into fragments, it is possible to allow a number of transactions to execute concurrently.
- Parallel execution can be utilised whereby a single query can be split into a set of subqueries that operate on fragments.

### 9.2 Disadvantages

- In the event that applications have conflicting requirements, it may not be possible to generate appropriate fragments, which could lead to a performance degradation.

- Integrity checking can be made more complex because checking for attribute dependencies (where the attributes in question are scattered) may mean having to chase after data.

### 9.3 Fragmentation Rules

V10. 14

**NCC** Fragmentation Rules

- Completeness
- Reconstruction
- Disjointness

From Ullman:

*Completeness* - All the data of the global relation must be mapped into the fragments, *i.e.* it must not happen that a data item which belongs to a global relation does not belong to any fragment.

*Reconstruction* - It must always be possible to reconstruct each global relation from its fragments. The necessity of this condition is obvious: in fact, only fragments are stored in the distributed database, and global relations have to be built through this reconstruction operation if necessary.

*Disjointness* - It is convenient that fragments be disjoint, so that the replication of data can be controlled explicitly at the allocation level.

### 9.4 Horizontal Fragmentation

V10. 15

**NCC** Horizontal Fragmentation - An Example

ENAME	ENO	BDATE	ADDRESS	SALARY	DNO
Taylor	1890	17/03/61	29 Fork Way	30000	10
May	1772	30/04/55	334 Mill Hill	34000	10
Smith	4327	02/08/60	45 High St	28000	30
Porter	3357	12/11/57	301 Oak Rd	30000	20
Kline	2095	28/03/58	99 Stone Vale	31000	10
Downs	3056	04/09/60	101 Holden Rd	45000	30
Landon	5557	15/09/61	33 Wood St	29000	10

↓

ENAME	ENO	BDATE	ADDRESS	SALARY	DNO
Taylor	1890	17/03/61	29 Fork Way	30000	10
May	1772	30/04/55	334 Mill Hill	34000	10
Kline	2095	28/03/58	99 Stone Vale	31000	10
Landon	5557	15/09/61	33 Wood St	29000	10

ENAME	ENO	BDATE	ADDRESS	SALARY	DNO
Smith	4327	02/08/60	45 High St	28000	30
Downs	3056	04/09/60	101 Holden Rd	45000	30

ENAME	ENO	BDATE	ADDRESS	SALARY	DNO
Porter	3357	12/11/57	301 Oak Rd	30000	20

From Ullman:

A horizontal fragment of a relation is a subset of the tuples in that relation.

The tuples that belong to the horizontal fragment are specified by a condition on one or more attributes of the relation. Often only a single attribute is involved. For example, we may define three horizontal fragments on the EMPLOYEE relation with the following conditions: (DNO = 10), (DNO = 30) and (DNO = 20); each fragment contains the employee tuples working for a particular department.

As we can see, horizontal fragmentation divides a relation *horizontally* by grouping rows or creating subsets of tuples, where each subset has a certain logical meaning. These fragments can then be assigned to different sites in the distributed system.

A set of horizontal fragments whose conditions include all tuples in the base relation is called a *complete horizontal fragmentation*. In many cases a complete horizontal fragmentation is also disjoint - that is there are no overlapping tuples.

Our example of horizontal fragmentation for the EMPLOYEE relation was both complete and disjoint.

In order to reconstruct the base relation from a complete horizontal fragmentation, we need to apply the UNION operation to the fragments.

## 9.5 Vertical Fragmentation

V10. 16

**Vertical Fragmentation - An Example**

ENAME	ENO	BDATE	ADDRESS	SALARY	DNO
Taylor	1890	17/03/61	29 Fork Way	30000	10
May	1772	30/04/55	334 Mill Hill	34000	10
Smith	4327	02/08/60	45 High St	28000	30
Porter	3357	12/11/57	301 Oak Rd	30000	20
Kline	2095	28/03/58	99 Stone Vale	31000	10
Downs	3056	04/09/60	101 Holden Rd	45000	30
London	5557	15/09/61	33 Wood St	29000	10

ENAME	BDATE	ADDRESS	ENO
Taylor	17/03/61	29 Fork Way	1890
May	30/04/55	334 Mill Hill	1772
Smith	02/08/60	45 High St	4327
Porter	12/11/57	301 Oak Rd	3357
Kline	28/03/58	99 Stone Vale	2095
Downs	04/09/60	101 Holden Rd	3056
London	15/09/61	33 Wood St	5557

ENO	SALARY	DNO
1890	30000	10
1772	34000	10
4327	28000	30
3357	30000	20
2095	31000	10
3056	45000	30
5557	29000	10

From Ullman:

Another type of fragmentation is called *vertical fragmentation*. A vertical fragment of a relation keeps only certain attributes in the relation that are related together in some way.

For example, we may want to fragment the EMPLOYEE relation into two vertical fragments where the first fragment includes personal information - ENAME, BDATE and ADDRESS - and the second includes work-related information - ENO, SALARY, DNO.

This vertical fragmentation is not quite proper because if the two fragments are stored separately we cannot put the original employee tuples back together as there is no common attribute between the two fragments. To be able to do this, it is necessary to include the primary key attribute in any vertical fragment so that the full relation can be reconstructed. Hence, we must add the ENO attribute to the personal fragment.

A set of vertical fragments whose projection lists L1, L2,.... include all the attributes in R but share only the primary key attribute of R is called a complete vertical fragmentation of R. To reconstruct the relation R from a complete vertical fragmentation, we apply the natural join operation to the fragments.

## 9.6 Mixed Fragmentation

V10. 17

**Mixed Fragmentation - An Example**

ENAME	BDATE	ADDRESS	ENO	ENO	SALARY	DNO
Taylor	17/03/61	29 Fork Way	1890	1890	30000	10
May	30/04/55	334 Mill Hill	1772	1772	34000	10
Kline	28/03/58	99 Stone Vale	2095	2095	31000	10
Landon	19/09/61	33 Wood St	5557	5557	29000	10
ENAME	BDATE	ADDRESS	ENO	ENO	SALARY	DNO
Smith	02/08/60	45 High Street	4327	4327	28000	30
Downs	04/09/60	101 Holden Rd	3056	3056	45000	30
ENAME	BDATE	ADDRESS	ENO	ENO	SALARY	DNO
Porter	12/11/57	301 Oak Rd	3357	3357	30000	20

From Ullman:

We can intermix the two types of fragmentation, yielding a mixed fragmentation. For example, we may combine the horizontal and vertical fragmentations of the EMPLOYEE relation given earlier into a mixed fragmentation that includes six fragments. In this case the original operation can be reconstructed by applying UNION and JOIN operations in the appropriate order.

## 10 Query Processing in Distributed Systems

*(30 minutes)*

From Ullman:

We have discussed the issues involved in processing and optimising a query in a centralised DBMS. In a distributed system there are several additional factors that must be taken into account, which further complicate query processing. The first and most important additional factor to consider is the cost of transferring data over the network. This data includes intermediate files that are transferred to other sites for further processing, as well as the final result files that may need to be transferred to the site where the query result is needed.

Although these costs may not be very high if the sites are connected *via* a high-performance local area network, they become quite significant in other types of network. Hence, many DDBMS query optimisation algorithms consider the goal of reducing the amount of data transfer as the main optimisation criterion in choosing a distributed query execution strategy.

## 10.1 Distributed Query Processing - An Example

V10. 18

**NCC** Distributed Processing Query - An Example

Site 1  
EMPLOYEE  
ENAME ENO BDATE ADDRESS SALARY MGR DNO

10,000 records  
each record is 100 bytes long  
ENO field is 9 bytes long  
DNO field is 4 bytes long  
ENAME field is 15 bytes long

Site 2  
DEPARTMENT  
DNAME DNO MGR DLOCATION

100 records  
each record is 35 bytes long  
DNO field is 4 bytes long  
MGR field is 9 bytes long  
DNAME is 10 bytes long

Site 3 (Result site)  
NO RELATIONS HELD HERE

*Example modified slightly from Ullman.*

We illustrate this with a simple query. Suppose the EMPLOYEE and DEPARTMENT relations are distributed as shown. We will assume in this example that neither relation is fragmented. The size of the EMPLOYEE relation is  $100 \times 10,000 = 10,00000$  bytes, and the size of the DEPARTMENT relation is  $35 \times 100 = 3500$  bytes.

V10. 19

**NCC** Distributed Processing Query, An Example - 1

"For each department, retrieve the department name and the name of the department manager"

```
SELECT DNAME, ENAME
FROM DEPARTMENT, EMPLOYEE
WHERE MGR = ENO;
```

Consider the query: "For each department, retrieve the department name and the name of the department manager." This can be stated as:

```
SELECT DNAME, ENAME
FROM DEPARTMENT, EMPLOYEE
WHERE MGR = ENO;
```

Suppose the query is submitted at site 3. Note that the result of the query will include only 100 records, assuming each department has a manager.

V10. 20

**NCC** Distributed Processing Query, An Example - 2

Three possible strategies:

- Transfer both the EMPLOYEE and DEPARTMENT relations to the result site (3) and perform the query there  
Transfer 1003500 bytes (1000000 + 3500)
- Transfer the EMPLOYEE relation to site 2, execute the query there and send the result to site 3  
Transfer 1004000 bytes (1000000 + 4000)
- Transfer the DEPARTMENT relation to site 1, execute the query there and send the result to site 3  
Transfer 7500 bytes (4000 + 3500)

There are three possible strategies for executing this distributed query:

- Transfer both the EMPLOYEE and DEPARTMENT relations to the result site and perform the query at site 3. In this case we need to transfer a total of  $1,000,000 + 3500 = 1,003,500$  bytes.
- Transfer the EMPLOYEE relation to site 2, execute the query at site 2, and send the result to site 3. The size of the query result is  $40 \times 100 = 4000$  bytes, so we transfer  $4000 + 1,000,000 = 1,004,000$  bytes.

- Transfer the DEPARTMENT relation to site 1, execute the query at site 1, and send the result to site 3. In this case we transfer  $4000 + 3500 = 7500$  bytes.

If minimising the amount of data transfer is the optimising criterion, we would choose strategy 3.

## 10.2 Distributed Query Processing Using Semijoin

From Ullman:

The idea behind distributed query processing using the *semijoin* operation is to reduce the number of tuples in a relation before transferring it to another site. Intuitively, the idea is to send the joining column of one relation R to the site where the other relation S is located. This column is then joined with S, and the join attributes and attributes required in the result are projected out and shipped back to the original site and joined with R.

Hence only the joining column of R is transferred in one direction, and a subset of S with no extraneous tuples is transferred in the other directions. If only a small fraction of the tuples in S participate in the join, this could be quite an efficient solution to minimising data transfer.

V10. 21

**Distributed Processing Query Using Semijoin**

This strategy has three steps:

- Project the join attributes of DEPARTMENT at site 2 and then transfer those attributes to site 1  
Transfer <MGR> i.e. 900 bytes ( $9 \times 100$ )
- Join the transferred file with the EMPLOYEE relation at site 1, and transfer the required attributes from the resulting file to site 3  
Transfer <MGR,ENAME> i.e. 3900 bytes ( $39 \times 100$ )
- Execute the query by joining the transferred file with DEPARTMENT, and present the result to the user at site 3

To illustrate this, consider the following strategy for executing the sample query:

- Project the join attributes of DEPARTMENT at site 2 and then transfer those attributes to site 1. We transfer MGR whose size is  $9 \times 100 = 900$  bytes.
- Join the transferred file with the EMPLOYEE relation at site 1, and transfer the required attributes from the resulting file to site 3. We transfer <MGR,ENAME>, whose size is  $39 \times 100 = 3900$  bytes.
- Execute the query by joining the transferred file with DEPARTMENT, and present the result to the user at site 3.

Using this strategy, we transfer only 4800 bytes. The reason is that we limited the EMPLOYEE tuples transmitted to site 2 in step 2 to only those tuples that will actually be joined with a DEPARTMENT tuple in step 3.


## 11 Summary

(10 minutes)

Distributed database systems are important because they can potentially offer many benefits in terms of the way in which a company wishes to organise itself.

They do however pose many problems, in particular they add complexity to the standard database system functionality.

V10. 22



### Summary

Distributed concepts which were dealt with:

- Replication
- Transparency
- Fragmentation
- Distributed query processing

Most DBMS vendors now have many distribution facilities available within their packages but a truly distributed database system is still very much an ideal.

Distributed concepts which were dealt with in some detail in this session include:

- replication;
- transparency;
- fragmentation;
- distributed query processing.