Distributed Databases

COSC430—Advanced Databases

David Eyers

Learning objectives

- You should be able to
  - define the key concepts in distributed database
  - distinguish between different types of distributed database
  - understand the architectures of distributed database
  - explain how to perform data fragmentation, allocation, and replication
  - explain how to do semi-join in distributed database
  - understand technologies for distributed transactions
- You will get a taste of carrying out scientific research
  - optimisation on data allocation and replication
  - the study of the research paper on Google Spanner

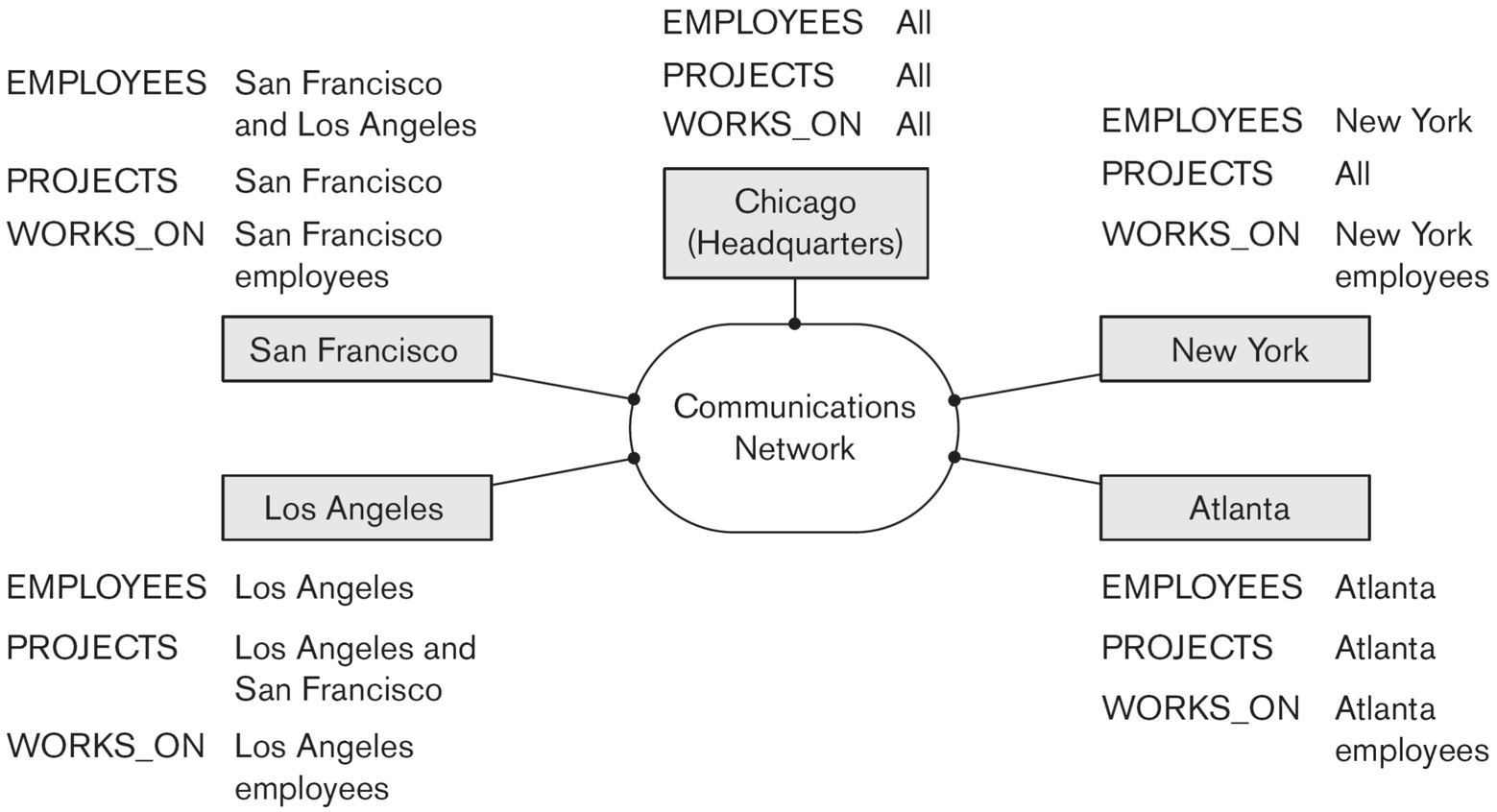
Definitions

- Distributed database
  - a collection of multiple, **logically interrelated** databases distributed over a **network** of computers
- Distributed DBMS
  - a software system that manages a distributed database and provides an access mechanism while making the distribution **transparent to the users**

Distributed database system=Database+Communication

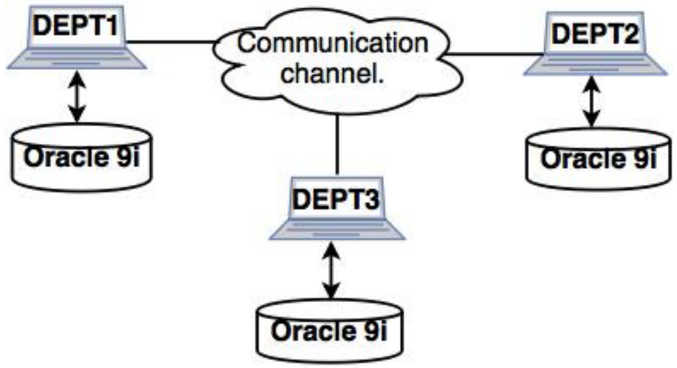
Key concepts for distributed databases

- Data stored at **several locations**
  - **Fragmentation**
  - **Replication**
- **Transparency**: hide implementation for users/developers
  - Location
  - Fragmentation
  - Replication
  - Design
  - Execution

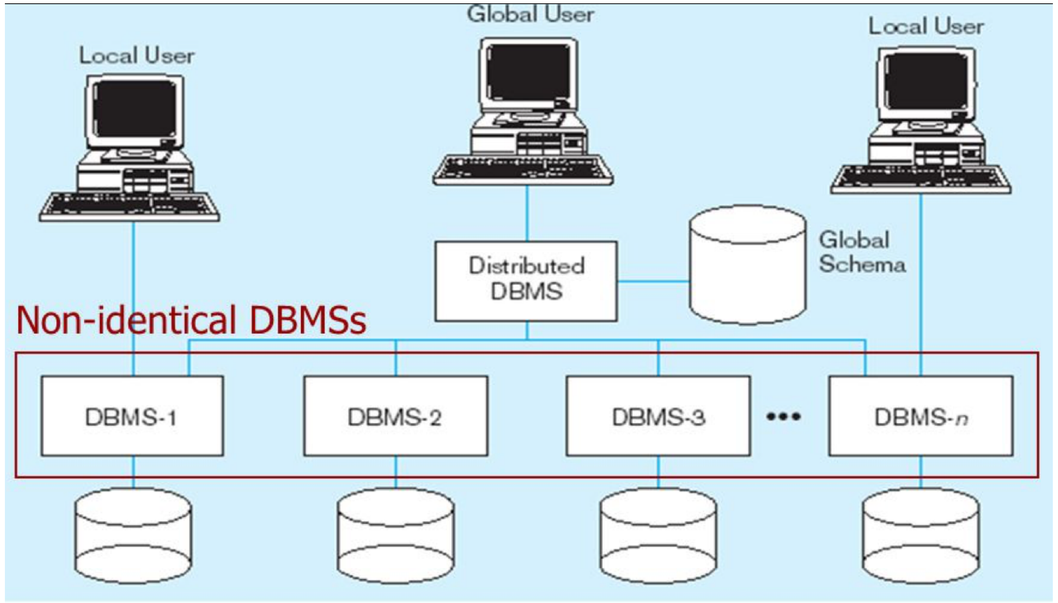


Types of distributed databases

- Homogeneous
  - All sites run the same DBMS

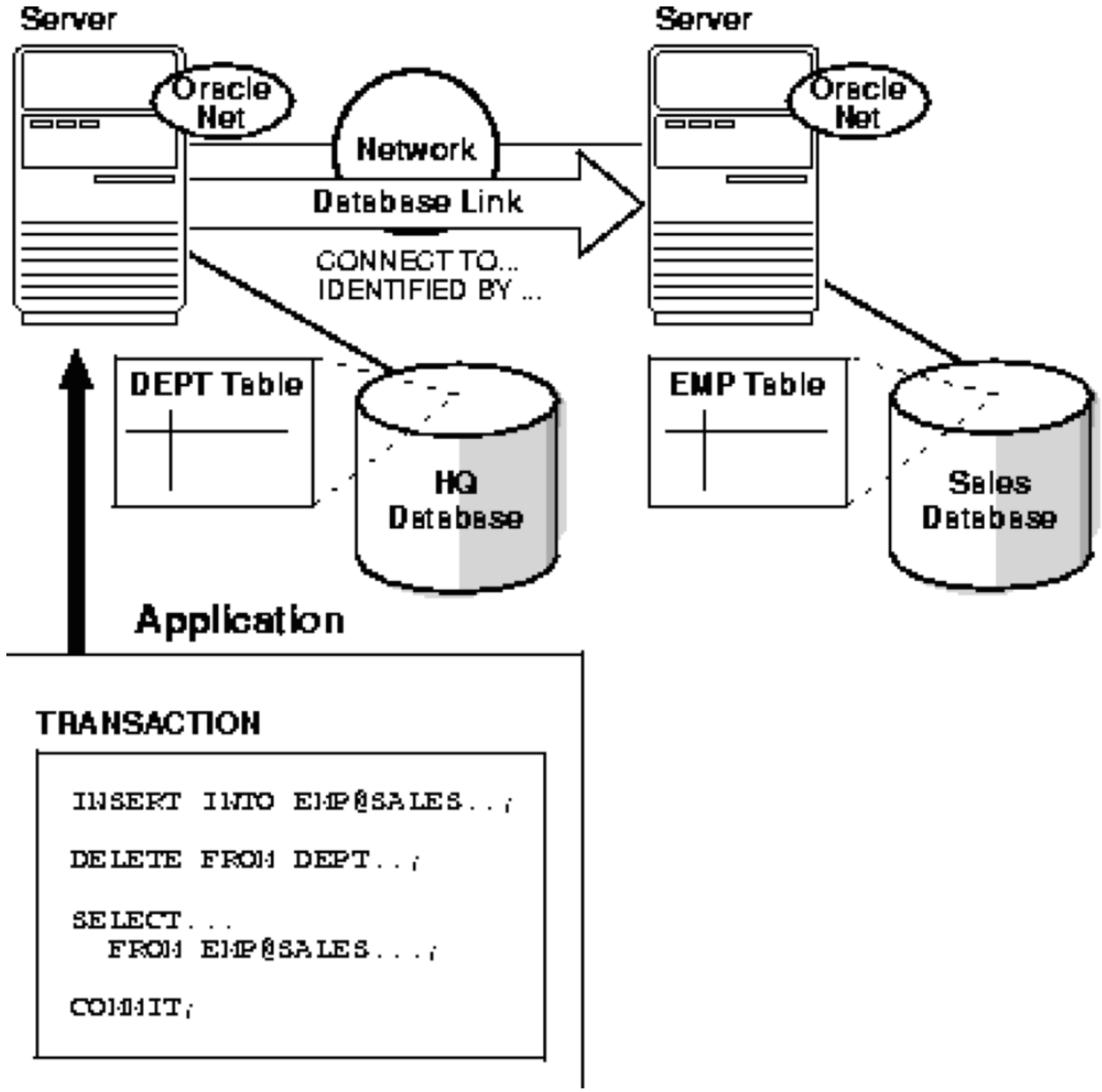


- Heterogeneous
  - Different sites can run different DBMSs



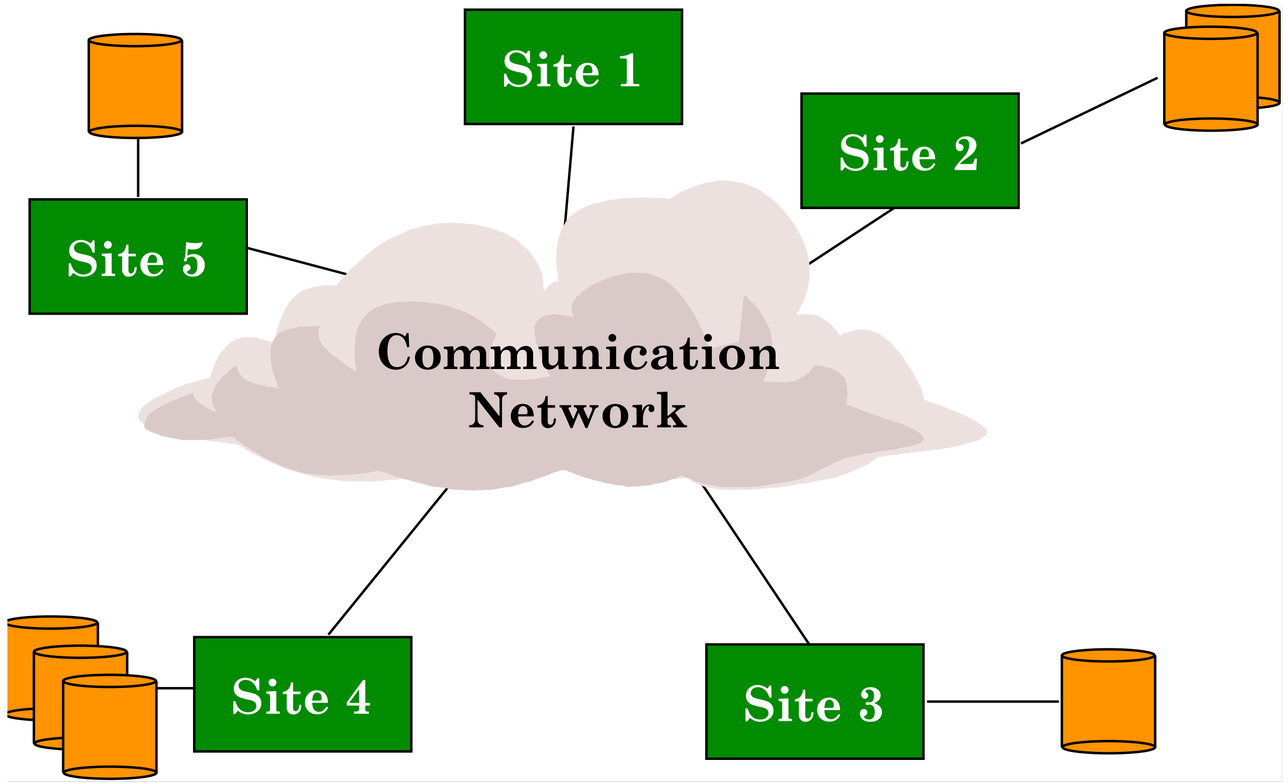
Distributed database architecture

- Client-Server
  - a two-level architecture based on the client-server concept (database centric)
  - A client can directly or indirectly connect to a server



Distributed Database Architecture

- Peer-to-Peer
  - All nodes have the same role and functionality
  - High scalability and flexibility
  - Servers are **autonomous** and **loosely coupled**
  - Management layer determine whether such infrastructure is easier or harder to manage



Main database functionality to consider

- **Data** layout
  - Data fragmentation
  - Data allocation and replication
- **Query** processing and optimisation
  - Data transfer cost
  - Semi-join
  - Query and update decomposition
- Distributed **transactions**
  - Transaction atomicity using two-phase commit
  - Transaction serialisability using distributed locking

Data fragmentation

- Break up the database into logical units, called **fragments**, which may be assigned for storage at various sites
- What is a reasonable unit of data distribution?
  - Relation
    - Can increase communications
    - Less parallelism
  - Sub-relations
    - Extra processing for views that cannot be defined on a single fragment
    - Better parallelism
    - Difficult to enforce integrity constraints

Data Fragmentation

- Horizontal fragmentation
  - Grouping rows to create subsets of tuples
  - Each subset has a certain logical meaning

PNAME	PNUMBER	PLOCATION	DNUM
Reorganisation	20	Houston	1
Computerisation	10	Stafford	4
Newbenefits	30	Stafford	4
NonProject	40		4
ProductX	1	Bellaire	5
ProductZ	3	Houston	5
ProductY	2	Sugarland	5

PNAME	PNUMBER	PLOCATION	DNUM
Reorganisation	20	Houston	1

PNAME	PNUMBER	PLOCATION	DNUM
Computerisation	10	Stafford	4
Newbenefits	30	Stafford	4
NonProject	40		4

PNAME	PNUMBER	PLOCATION	DNUM
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5

Data Fragmentation

- Vertical fragmentation—divide relation by columns
  - Each fragment has the primary key or some candidate key so that the full relation can be reconstructed

FNAME	M	LNAME	SSN	BDATE	ADDRESS	S	SALARY	SUPERSSN	DNO
John	B	Smith	123456789	09-JAN-65	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	08-DEC-55	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	19-JUL-68	3321 Castle, Spring,TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	20-JUN-41	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	15-SEP-62	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	31-JUL-72	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	29-MAR-69	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	10-NOV-37	450 Stone, Houston, TX	M	55000		1

FNAME	M	LNAME	SSN	BDATE	ADDRESS	S
John	B	Smith	123456789	09-JAN-65	731 Fondren, Houston, TX	M
Franklin	T	Wong	333445555	08-DEC-55	638 Voss, Houston, TX	M
Alicia	J	Zelaya	999887777	19-JUL-68	3321 Castle, Spring,TX	F
Jennifer	S	Wallace	987654321	20-JUN-41	291 Berry, Bellaire, TX	F
Ramesh	K	Narayan	666884444	15-SEP-62	975 Fire Oak, Humble, TX	M
Joyce	A	English	453453453	31-JUL-72	5631 Rice, Houston, TX	F
Ahmad	V	Jabbar	987987987	29-MAR-69	980 Dallas, Houston, TX	M
James	E	Borg	888665555	10-NOV-37	450 Stone, Houston, TX	M

SSN	SALARY	SUPERSSN	DNO
123456789	30000	333445555	5
333445555	40000	888665555	5
999887777	25000	987654321	4
987654321	43000	888665555	4
666884444	38000	333445555	5
453453453	25000	333445555	5
987987987	25000	987654321	4
888665555	55000		1

Data Fragmentation

- Mixed (Hybrid) fragmentation
  - Intermix the two types of fragmentation
- Correctness of fragmentation
  - **Completeness**: each data item can be found in one fragment
  - **Reconstruction** (lossless): the full relation can be reconstructed from all fragments
  - **Disjointness** (non-overlapping): each data item except the key should not be in more than one fragment

Data Replication

- Non-replicated
  - Each fragment resides at only one site
- Replicated
  - Fully replicated: each fragment at each site
  - Partially replicated: each fragments at some sites
- Pros & cons
  - Improve availability, distribute load, cheaper reads
  - Complexity on update and storage
- Rule of thumb
  - If (read-only queries)/(update queries) ≥ 1, replication is advantageous, otherwise replication may cause problems

Comparison of replication alternatives

	Full replication	Partial replication	Partitioning
Query processing	Easy	Some difficulty	
Directory management	Easy or non-existent	Some difficulty	
Concurrency control	Moderate	Difficult	Easy
Reliability	Very high	High	Low

Optimisation

- Best fragmentation, replication and allocation?
  - A generic algorithm for fragment allocation in a distributed database system (Corcoran, SAC'94)
  - Set of m sites S, each has capacity c_i

$$S = \{c_1, c_2, c_3, \ldots, c_i, \ldots, c_m\}$$

  - Set F of n fragments, each fragment j has size s_j

$$F = \{s_1, s_2, s_3, \ldots, s_j, \ldots, s_n\}$$

  - Site requirement matrix R:
    - $r_{i,j}$ is requirement by site i for fragment j

$$R = \begin{bmatrix} r_{1,1} & r_{1,2} & \cdots & r_{1,n} \\ r_{2,1} & r_{2,2} & \cdots & r_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ r_{m,1} & r_{m,2} & \cdots & r_{m,n} \end{bmatrix}$$

Formalising optimisation of allocation

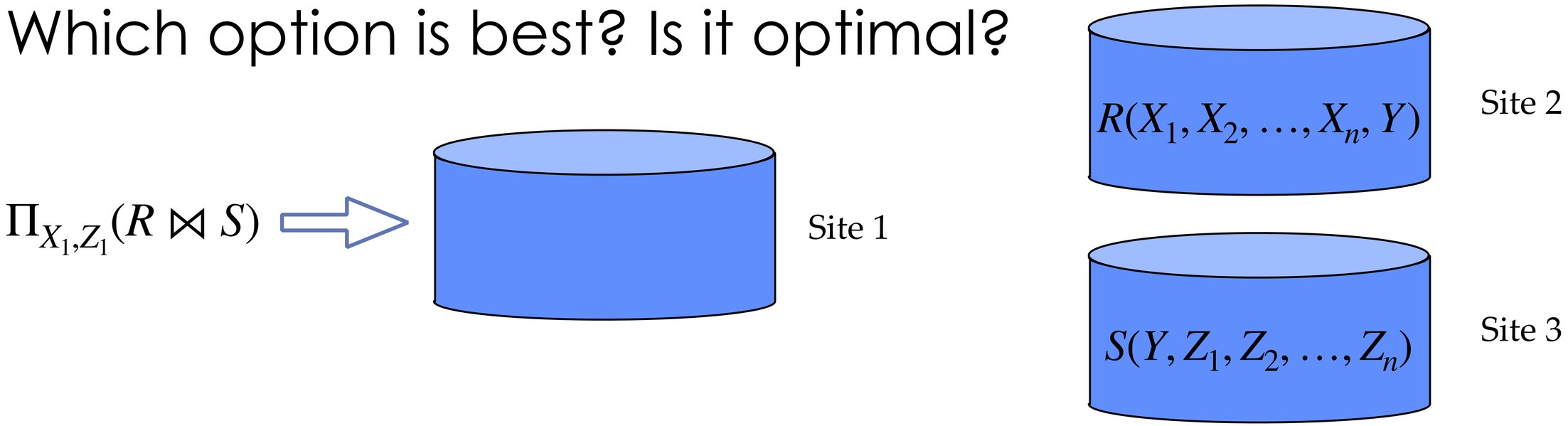
- Transmission cost matrix T
  - $t_{i,j}$ is cost for site i to access a fragment of data on site j

$$T = \begin{bmatrix} t_{1,1} & t_{1,2} & \cdots & t_{1,m} \\ t_{2,1} & t_{2,2} & \cdots & t_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ t_{m,1} & t_{m,2} & \cdots & t_{m,m} \end{bmatrix}$$

- Fragment placement vector $P = \{p_1, p_2, p_3, \ldots, p_j, \ldots, p_n\}$
  - p_j=i indicates fragment j is at site i
- Objective: minimise total transmission cost: $\sum_{i=1}^{m} \sum_{j=1}^{n} r_{i,j} t_{i,p_j}$
- … but subject to: $\forall i, 1 \leq i \leq m, \sum_{j=1}^{n} r_{i,j} s_j \leq c_i$

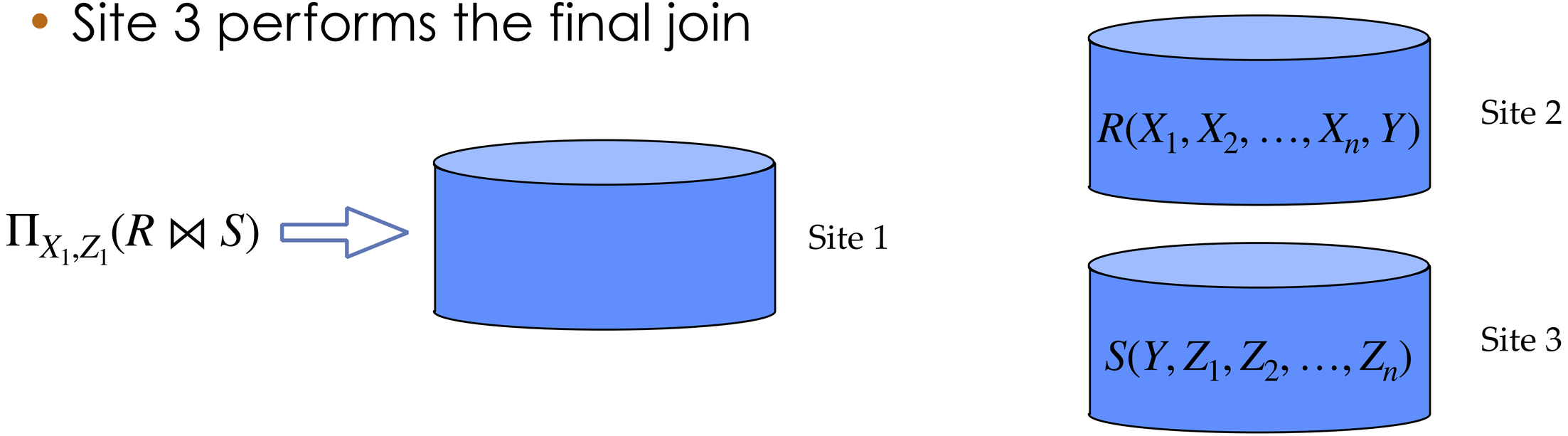
Distributed query processing

- Optimisation goal: reduce the amount of data transfer
  - Option 1: Send both R and S to Site 1 for join
  - Option 2: Send R to Site 3 to join, send join results back to site 1
  - Option 3: Send S to Site 2 to join, send join results back to site 1
- Which option is best? Is it optimal?

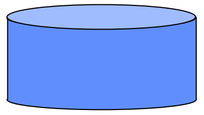


Semi-joins can improve efficiency of queries

- Reduce number of tuples before transfer to other site
  - Site 3 sends only S.Y column to Site 2
  - Site 2 does the join based on R's Y column; sends the records of R that will join (without duplicates) back to Site 3
  - Site 3 performs the final join



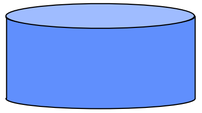
Semi-join example



Site 2
Employee

FNAME	M	LNAME	SSN	BDATE	ADDRESS	S	SALARY	SUPERSSN	DNO
John	B	Smith	123456789	09-JAN-65	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	08-DEC-55	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	19-JUL-68	3321 Castle, Spring,TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	20-JUN-41	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	15-SEP-62	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	31-JUL-72	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	29-MAR-69	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	10-NOV-37	450 Stone, Houston, TX	M	55000		1

Site 3
Department



DNAME	DNUMBER	MGRSSN	MGRSTARTD	TOTAL_SALARY
Research	5	333445555	22-MAY-88	0
Administration	4	987654321	01-JAN-95	0
Headquarters	1	123456789	19-JUN-81	0
Dummies	0	666884444	31-DEC-04	0

Query at site 1:

$\Pi_{Fname,Lname,Dname}(Employee \bowtie_{Dno=Dnumber} Department)$

$F = \Pi_{Dnumber}(Employee)$

DNO
5
4

$Q = \Pi_{Dname,Dnumber}(F \bowtie_{Dno=Dnumber} Department)$

DNAME	DNUMBER
Administration	4
Research	5

$\Pi_{Fname,Lname,Dname}(Q \bowtie_{Dno=Dnumber} Employee)$

FNAME	LNAME	DNAME
Franklin	Wong	Research
John	Smith	Research
Jennifer	Wallace	Administration
Alicia	Zelaya	Administration
Joyce	English	Research
Ramesh	Narayan	Research
Ahmad	Jabbar	Administration

Query or Update Decomposition

- Idea: decompose query or update into sequence of queries or updates executed at the individual sites

$$\Pi_{Fname,Lname,hours}\sigma_{dno=5}(Employee \bowtie_{Dno=Dnumber} Project \bowtie_{Pno=Pnumber} Works\_on)$$

- Suppose a site stores all information about projects controlled by department 5 (Projs5) and employees working on these projects (Works_on5):

$$T_1 \leftarrow \Pi_{Essn,pno,hours}\sigma_{dno=5}(Projs5 \bowtie_{Pno=Pnumber} Works\_on5)$$

$$T_2 \leftarrow \Pi_{Fname,Lname,hours}(T_1 \bowtie_{Ssn=Essn} Employee)$$

Transactions

- A transaction is an atomic sequence of actions (reads and writes)
- ACID properties covered earlier
  - Atomicity
  - Consistency
  - Isolation
  - Durability

Success!

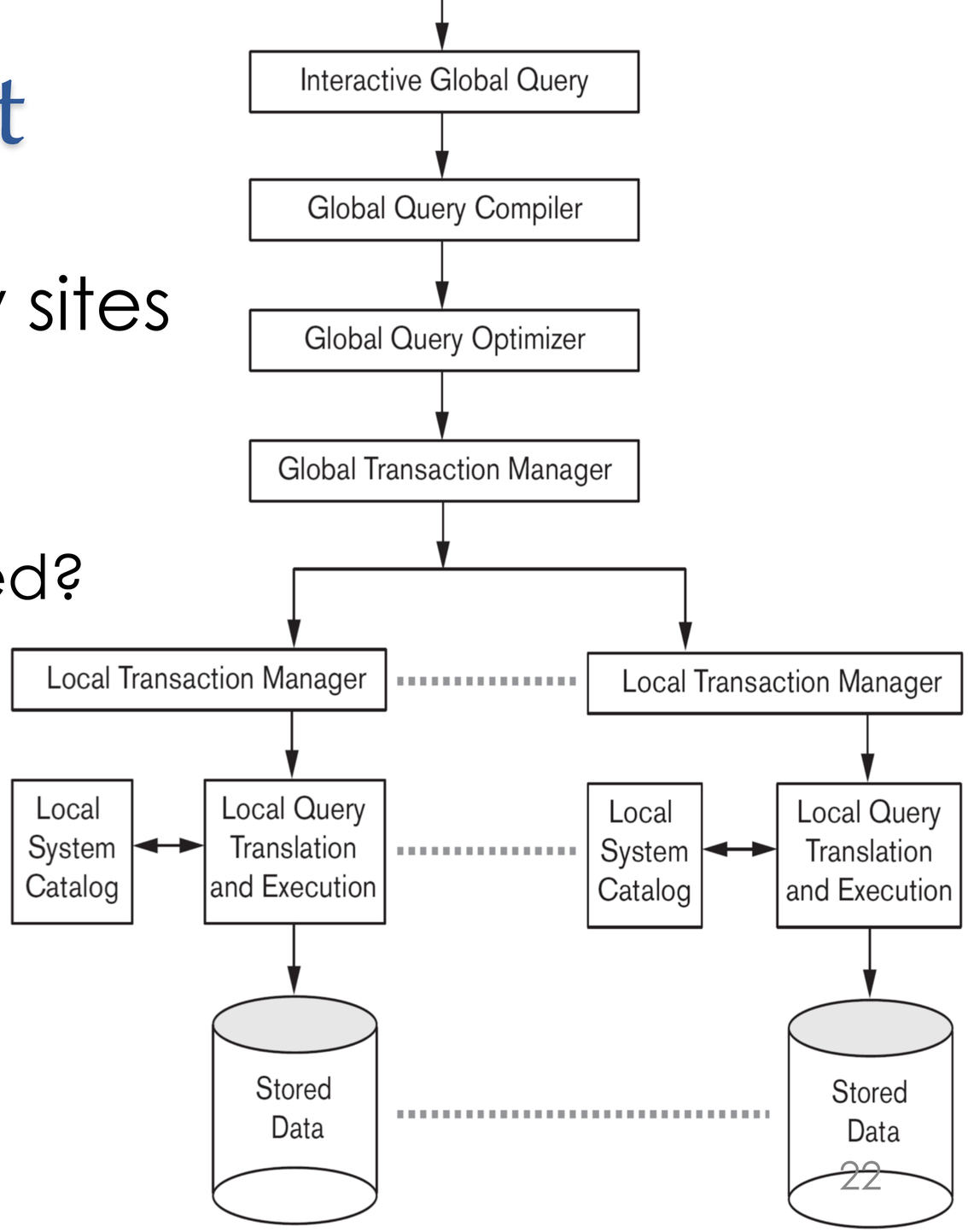
```
Begin()
  action
  action
  action
  action
Commit()
```

Failure!

```
Begin()
  action
  action
  action
  action
Rollback()
```

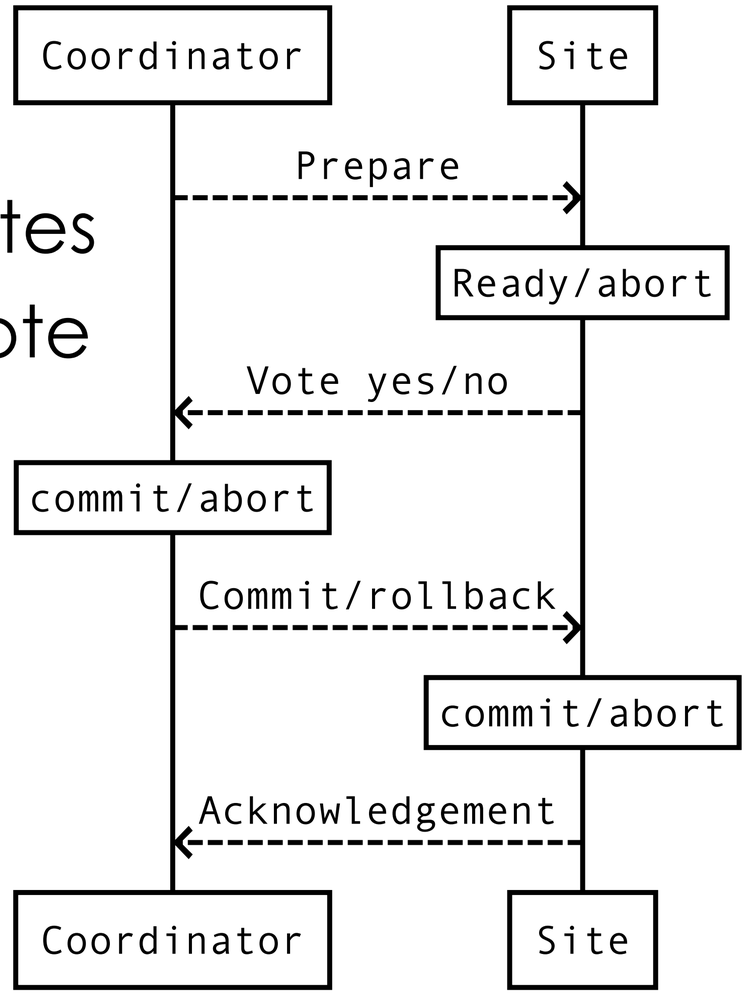
Transaction management

- Transaction T may touch many sites
  - $T=\{T_1, T_2, \ldots, T_n\}$
  - T_k runs at site k
  - How can atomicity be guaranteed?
- Two-Phase Commit protocol
  - Global transaction coordinator
  - e.g., the site that initiated the transaction can be the coordinator



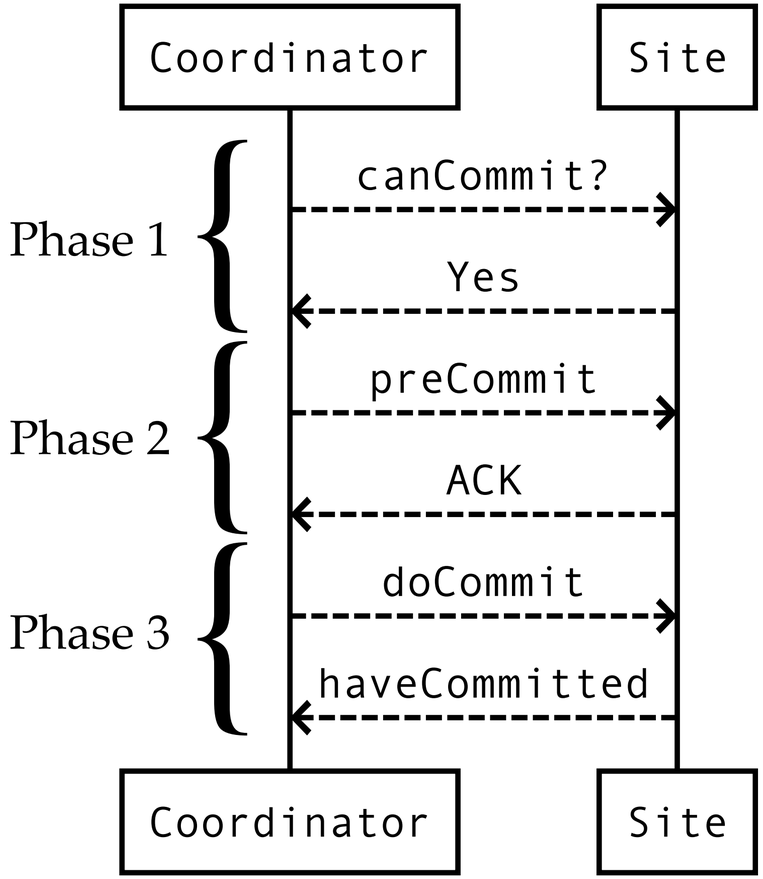
Two-Phase Commit (2PC) protocol

- Request phase
  - Coordinator sends prepare message to all sites
  - Sites reply with “Ready” or “Abort” in their vote
- Commit phase
  - Coordinate sends a commit message if all sites vote “Ready”, otherwise sends an abort message
  - Each site either commits or aborts, and replies with an acknowledge message



Three-Phase Commit (3PC) protocol

- 2PC can't recover from failure of both the coordinator and a site during commit phase
  - 3PC solves by introducing the Prepared to commit state (phase 2)
  - A site receiving `preCommit` knows that all other sites said `Yes` to `canCommit?`



Concurrency control in distributed DBs

- More complicated than that in a centralised DB environment
  - Dealing with multiple copies of the data item
  - Failure of individual site
  - Failure of communication links
  - Distributed commit
  - Distributed deadlock

Distributed locking

T_3	T_4
lock-X(B)	
read(B)	
$B := B - 50$	
write(B)	
	lock-S(A)
	read(A)
	lock-S(B)
lock-X(A)	

- Centralised approach
  - One dedicated site manages all locks
  - Bottleneck, unsalable, single point failure
- Primary-copy approach
  - Each data item has a primary site.
  - Each transaction asks the primary site for lock on the data item.
- Full distributed (voting) approach
  - A lock request is sent to all sites with a copy of the data item
  - Each copy maintains its own lock and can grant or deny the request for it
  - Use timeout to resolve deadlock

Summary

- Data fragmentation, allocation, and replication are key considerations that can significantly affect the performance of a distributed database
- Queries in distributed databases can (sometimes) be optimised through the use of semi-joins
- Transaction management and concurrency control in distributed databases are more complex than the equivalent techniques within centralised databases