Advanced Management of Data

Concepts of Distributed Databases (2)

DDBMS-Architecture

Global conceptual schema

- is a logical description of the whole database
- provides physical data independence from the distributed environment

Global external schemas

- provide logical data independence

Fragmentation schema

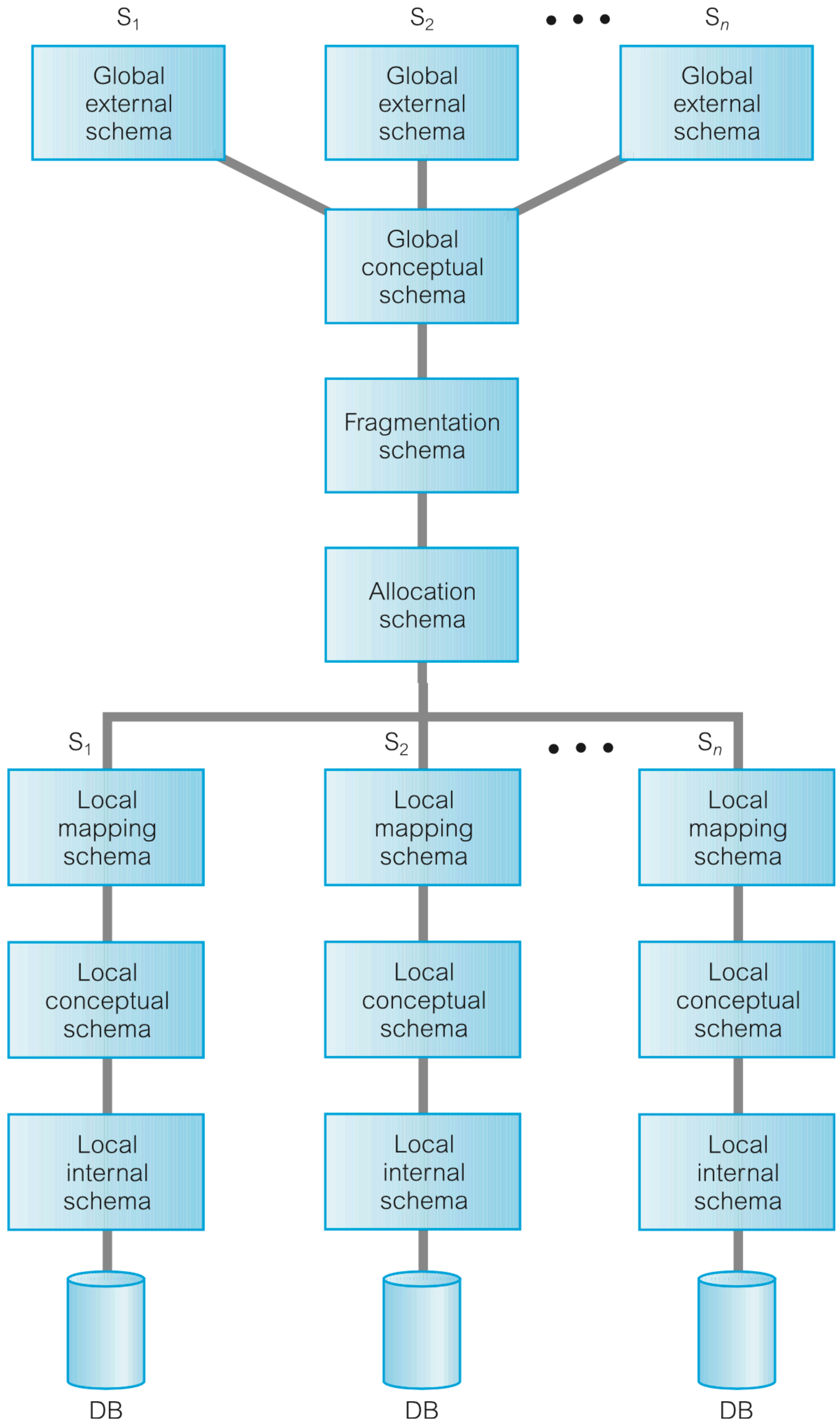
- description of how the data is to be logically partitioned

Allocation schema

- description of where the data is to be located, taking account of any replication

Local mapping schemas

- map fragments in the allocation schema into external objects in the local database



[Connolly & Begg]

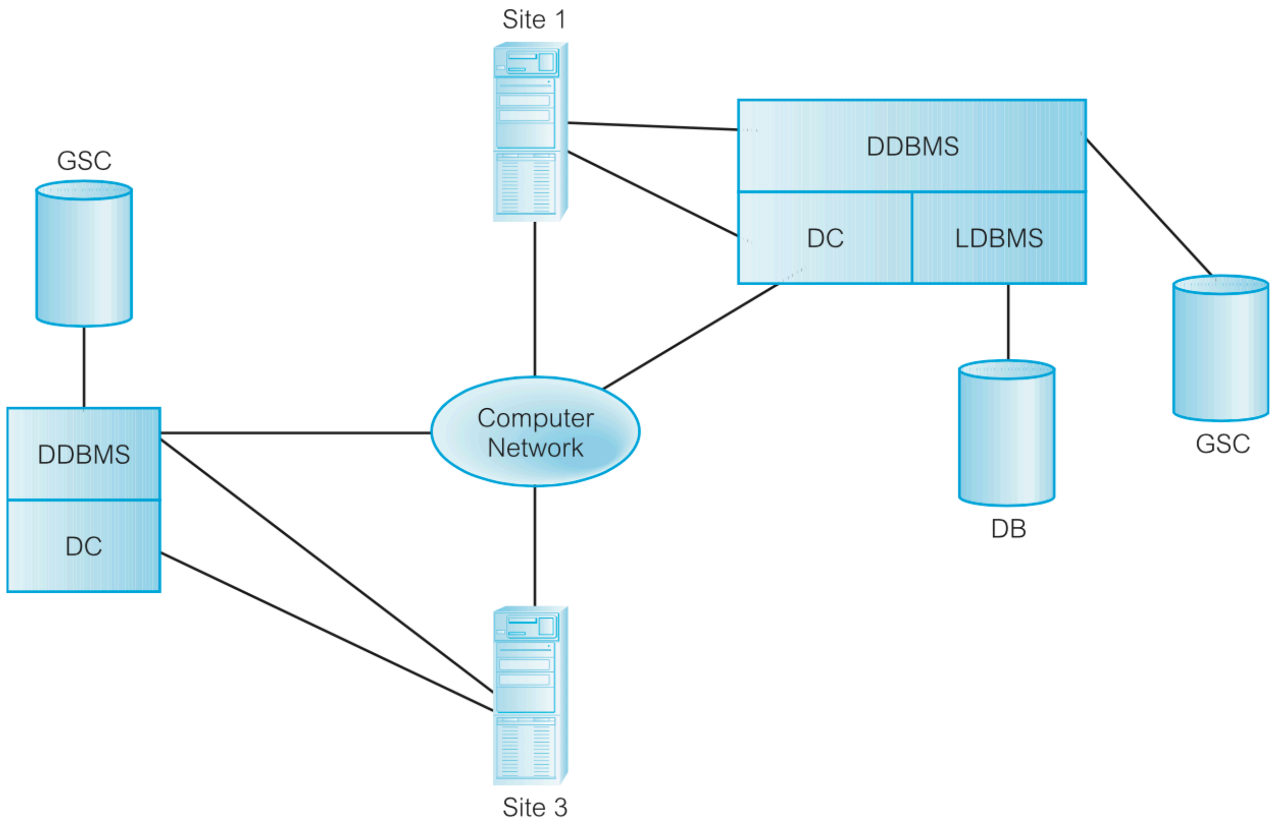
DDBMS Component Architecture

Local DBMS (LDBMS) component

- standard DBMS, responsible for controlling the local data at each site that has a database
- has its own local system catalog that stores information about the data held at that site.

Data communications (DC) component

- software that enables all sites to communicate with each other
- contains information about the sites and the links



[Connolly & Begg]

DDBMS Component Architecture

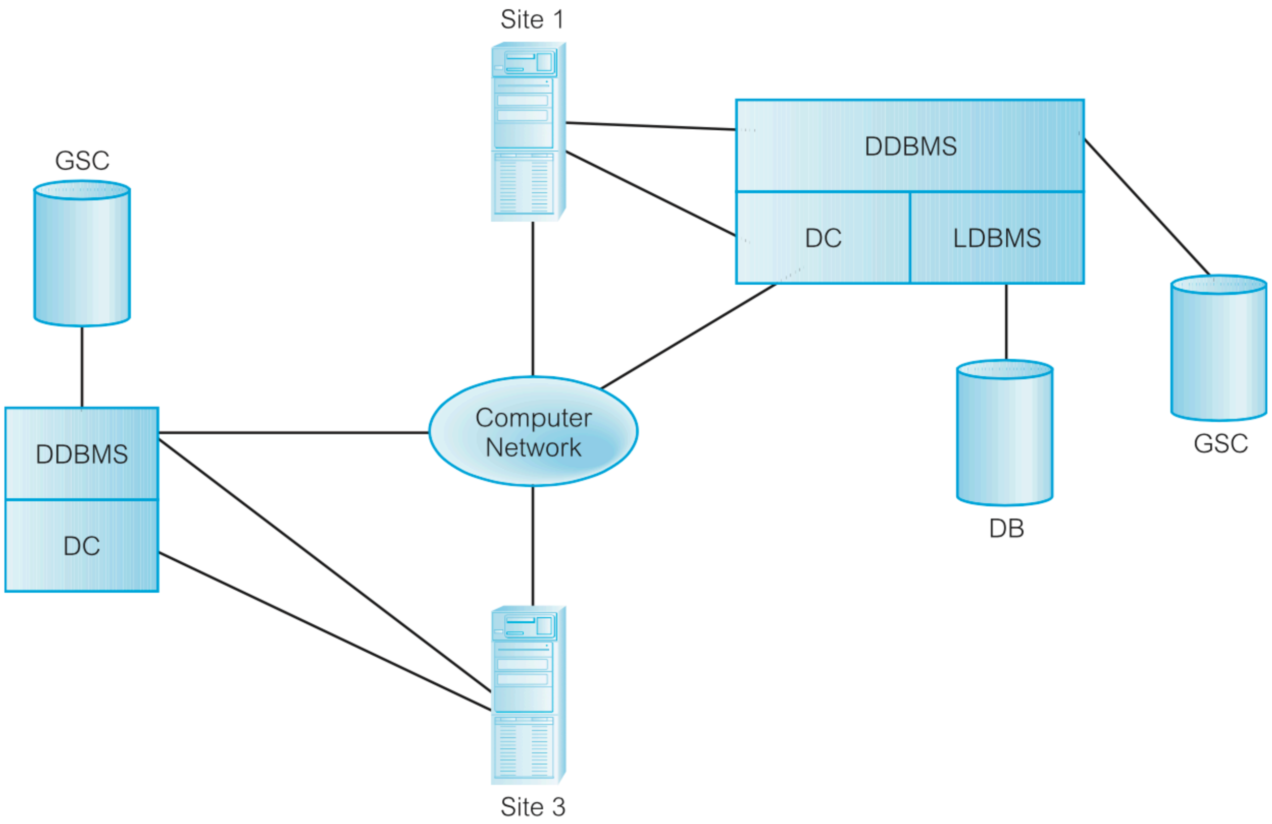
Global system catalog (GSC)

- holds information specific to the distributed nature of the system, such as the fragmentation, replication, and allocation schemas

- can itself be managed as a distributed database (showing similar advantages and disadvantages)

DDBMS component

- controlling unit of the entire system



[Connolly & Begg]

Distributed Queries / Updates

DDBMS with no distribution transparency

- users phrase a query by specifying the location of needed fragments directly

DDBMS with no replication transparency

- users are responsible for maintaining consistency of replicated data items when updating

DDBMS that supports full distribution, fragmentation, and replication transparency

- users specify a query just as in a non-distributed DBMS
    1. a query decomposition module decomposes a query into subqueries that can be performed at the individual sites
    2. a subquery composition module combines the results of the subqueries
- for updates, the DDBMS is responsible for maintaining consistency among replicated items

Query and Update Decomposition

Whenever the DDBMS determines that an referenced item is replicated, it must choose a particular replica during query execution. The DDBMS catalog stores information about

- replication
- distribution
- fragmentation
    - for vertical fragmentation, the attribute list for each fragment is kept in the catalog
    - for horizontal fragmentation, a selection condition called guard is kept for each fragment
    - for mixed fragments, both the attribute list and the guard condition are kept in the catalog

EMPD_4

Fname	Minit	Lname	Ssn	Salary	Super_ssn	Dno
Alicia	J	Zelaya	999887777	25000	987654321	4
Jennifer	S	Wallace	987654321	43000	888665555	4
Ahmad	V	Jabbar	987987987	25000	987654321	4

Example

DEP_4

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Administration	4	987654321	1995-01-01

DEP_4_LOCS

Dnumber	Location
4	Stafford

WORKS_ON_4

Essn	Pno	Hours
333445555	10	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0

PROJS_4

Pname	Pnumber	Plocation	Dnum
Computerization	10	Stafford	4
New_benefits	30	Stafford	4

[Elmasri & Navathe]

Fragmentation information stored in DDBMS catalog:

A * symbol specifies all attributes of a relation

EMPD4 attribute list: Fname, Minit, Lname, Ssn, Salary, Super_ssn, Dno
EMPD4 guard: Dno = 4

DEP4 attribute list: *
DEP4 guard: Dnumber = 4

DEP4_LOCS attr: *
DEP4_LOCS guard: Dnumber = 4

PROJS4 attribute list: *
PROJS4 guard: Dnum = 4

WORKS_ON4 attr.: *
WORKS_ON4 guard: Essn IN (π_{Ssn}(EMPD4)) OR Pno IN ($\pi_{Pnumber}$(PROJS4))

Query Processing

A distributed database query is processed in four stages:

1. **Query Mapping** The input query on distributed data is specified formally using a query language. It is then translated into an algebraic query on global relations using the global conceptual schema.

2. **Localization** The distributed query is mapped on the global schema to separate queries on individual fragments using data distribution and replication information.

3. **Global Query Optimization** Selecting a strategy from a list of candidates that is closest to optimal. A list of candidate queries can be obtained by permuting the ordering of operations within a fragment query generated by the previous stage. The total cost is a weighted combination of CPU cost, I/O costs, and communication costs.

4. **Local Query Optimization** The techniques are similar to those used in centralized systems.

Query Processing

Network Data Transfer

- relational data needs to be transferred to other sites for further processing
  - single tuples
  - intermediate files (the result of a partial query)
  - entire relations
- the final query result may be needed at a different site than it has been computed

Distributed query optimization

Network data transfer is an important cost factor in DDBs, because in most cases it is relatively slow compared to CPU or I/O transfer speeds ➔ reducing the amount of network data transfer is an important optimization criterion in DDBMS.

Example 1

Site 1:

EMPLOYEE

Fname	Minit	Lname	<u>Ssn</u>	Bdate	Address	Sex	Salary	Super_ssn	Dno

10,000 records
each record is 100 bytes long
Ssn field is 9 bytes long
Dno field is 4 bytes long
Fname field is 15 bytes long
Lname field is 15 bytes long

Site 2:

DEPARTMENT

Dname	<u>Dnumber</u>	Mgr_ssn	Mgr_start_date

100 records
each record is 35 bytes long
Dnumber field is 4 bytes long
Mgr_ssn field is 9 bytes long
Dname field is 10 bytes long

[Elmasri & Navathe]

Query Q1

For each employee, retrieve the employee name and the name of the department for which the employee works (we assume that every employee is related to a department).

Result side

The query is submitted at a distinct site 3

Example 2

Site 1:

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

10,000 records
each record is 100 bytes long
Ssn field is 9 bytes long
Dno field is 4 bytes long
Fname field is 15 bytes long
Lname field is 15 bytes long

Site 2:

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

100 records
each record is 35 bytes long
Dnumber field is 4 bytes long
Mgr_ssn field is 9 bytes long
Dname field is 10 bytes long

[Elmasri & Navathe]

Query Q2

For each department, retrieve the department name and the name of the department manager (we assume that each department has a manager)

Result side

The query is submitted at a distinct site 3.

Query Processing Using Semijoin

Semijoin

$$\mathbf{R} \triangleright_{F} \mathbf{S}$$

The Semijoin operation defines a relation that contains the tuples of **R** that participate in the join of **R** with **S** satisfying the predicate ***F***.

We can rewrite the Semijoin using the Projection and Join operations:

$$R \triangleright_F S = \Pi_A(R \bowtie_F S)$$

(A is the set of all attributes for R)

T

A	B
a	1
b	2

U

B	C
1	x
1	y
3	z

$T \triangleright_{\mathrm{B}} U$

A	B
a	1

Query Processing Using Semijoin

Idea

Reduce the number and size of tuples before transferring them to another site

Steps

1. send the joining column(s) *jc* of one relation *R* to the site where the other relation *S* is located
2. join *jc* with *S*
3. project the *jc* and attributes required in the result from S and transfer them to the site of *R*
4. join the transferred columns with *R*

Realization of distributed Semijoin

1. Project the join attributes of *S* and transfer them to the site where *R* resides
2. Join the transferred attributes with *R.*

Example 1

Site 1:

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

10,000 records
each record is 100 bytes long
Ssn field is 9 bytes long
Dno field is 4 bytes long

Fname field is 15 bytes long
Lname field is 15 bytes long

Site 2:

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

100 records
each record is 35 bytes long
Dnumber field is 4 bytes long
Mgr_ssn field is 9 bytes long

Dname field is 10 bytes long

[Elmasri & Navathe]

Using Semijoin Strategy for Query Q1

```
SELECT Fname, Lname, Dname
FROM EMPLOYEE JOIN DEPARTMENT
WHERE Dno=Dnumber
```

1. Project the join attributes of DEPARTMENT (Dnumber) at site 2, and transfer them to site 1:

$$4 * 100 = 400 \text{ bytes}$$

2. Join the transferred file with the EMPLOYEE relation at site 1, and transfer the required attributes from the resulting file (Dno, Fname, Lname) to site 2:

$$34 * 10{,}000 = 340{,}000 \text{ bytes}$$

3. Perform the query by joining the transferred file with DEPARTMENT

- in total, we transferred 400 + 340,000 = 340,400 (vs. 403,500 without semijoin) bytes

Example 2

Site 1:

EMPLOYEE

Fname	Minit	Lname	<u>Ssn</u>	Bdate	Address	Sex	Salary	Super_ssn	Dno

10,000 records
each record is 100 bytes long
Ssn field is 9 bytes long
Dno field is 4 bytes long

Fname field is 15 bytes long
Lname field is 15 bytes long

Site 2:

DEPARTMENT

Dname	<u>Dnumber</u>	Mgr_ssn	Mgr_start_date

100 records
each record is 35 bytes long
Dnumber field is 4 bytes long
Mgr_ssn field is 9 bytes long

Dname field is 10 bytes long

[Elmasri & Navathe]

Using Semijoin Strategy for Query Q2

```
SELECT Fname, Lname, Dname
FROM DEPARTMENT JOIN EMPLOYEE
WHERE Mgr_ssn=Ssn
```

1. Project the join attributes of DEPARTMENT (Mgr_ssn) at site 2, and transfer them to site 1:

   9 * 100 = 900 bytes

2. Join the transferred file with the EMPLOYEE relation at site 1, and transfer the required attributes (Mgr_ssn, Fname, Lname) from the resulting file to site 2:

   39 * 100 = 3,900 bytes

3. Perform the query by joining the transferred file with DEPARTMENT

   - in total, we transferred 900 + 3,900 = 4,800 (vs. 7,500 without semijoin) bytes

Query Processing and Optimization

Communication time

The time taken to send a message depends upon the length of the message and the type of network being used. It can be calculated using the following formula:

communication_time = access_delay + (no_of_bits_in_message / transmission_rate)

Examples

Using access_delay = 1 second, transmission_rate = 10 000 bits per second, record_size = 100 bits, we can calculate the time to send 100 000 records as a whole as:

communication_time = 1 + (100 000 * 100 / 10 000) = 1001 seconds

To transfer 100 000 records one at a time, we need:

communication_time = 100 000 * [1 + (100 / 10 000)] = 101 000 seconds

Example

Consider the following query over three relations P, C, and V:

```
SELECT P.pKey
FROM P INNER JOIN (C INNER JOIN V ON C.cKey = V.cKey) ON P.pKey = V.pKey
WHERE pValue = 1 AND cValue = 2;
```

P(pKey, pValue)	location: site 1	tuple number: 10 000	tuple size: 100 bits
C(cKey, cValue)	location: site 2	tuple number: 100 000	tuple size: 100 bits
V(pKey, cKey)	location: site 1	tuple number: 1 000 000	tuple size: 100 bits

tuples in (P INNER JOIN V) that fulfill the condition pValue = 1: 100 000

tuples in C that fulfill the condition cValue = 2: 10

data transmission rate: 10 000 bits / second

access delay: 1 second

In this example we consider computation time to be negligible compared with communication time.