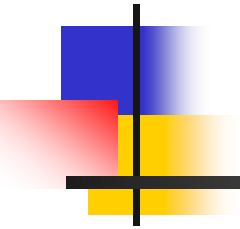
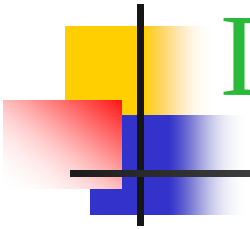


Mobile and Heterogeneous databases
Distributed Database System
Query Processing

A.R. Hurson

Computer Science

Missouri Science & Technology



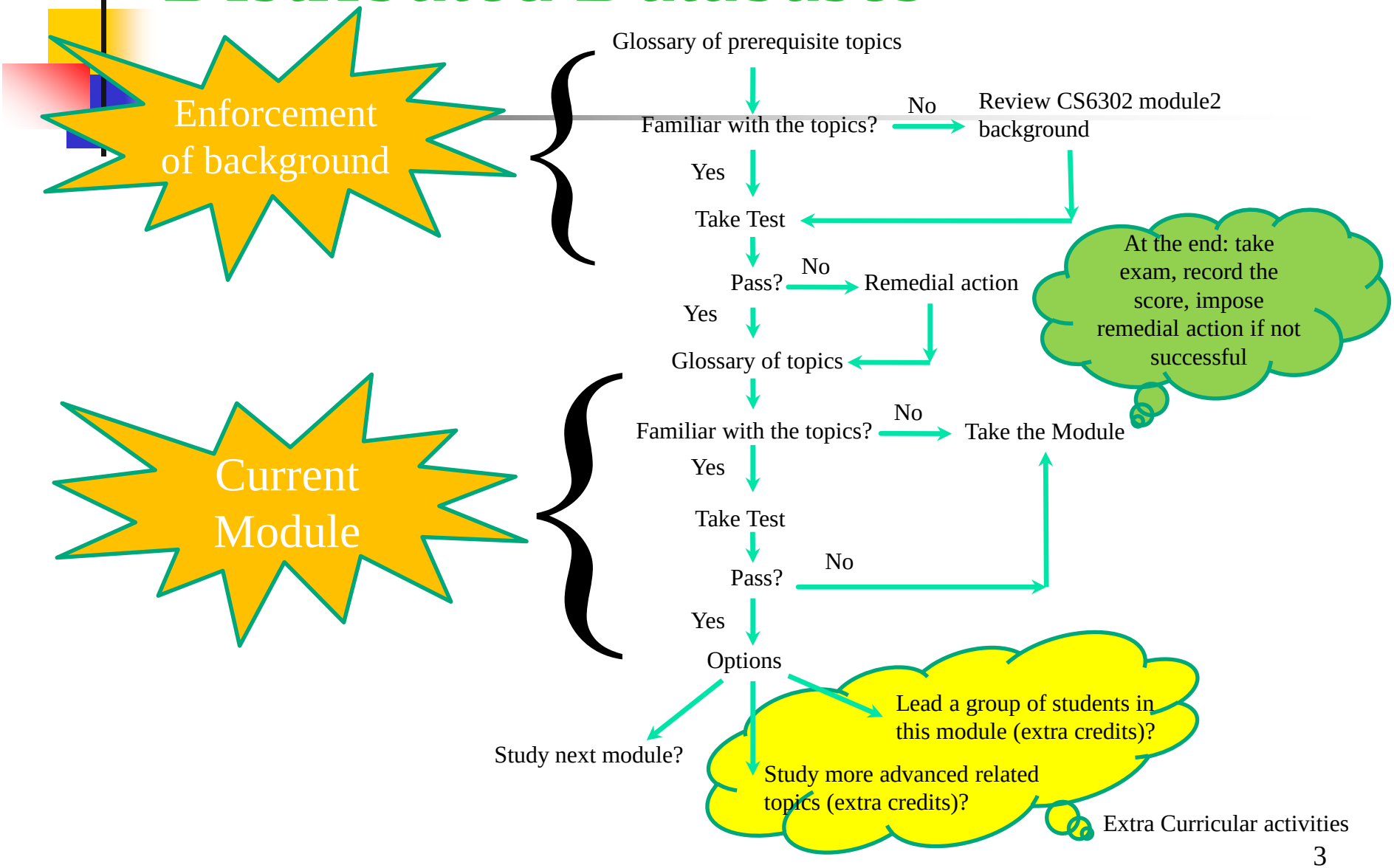
Distributed Databases

Note, this unit will be covered in four lectures. In case you finish it earlier, then you have the following options:

1) Take the early test and start CS6302.module3
2) Study the supplement module (supplement CS6302.module2)
3) Act as a helper to help other students in studying CS6302.module2

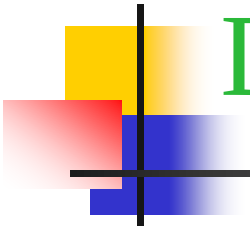
Note, options 2 and 3 have extra credits as noted in course outline.

Distributed Databases



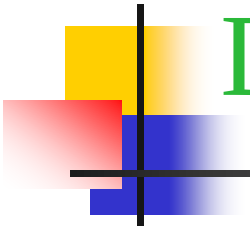
Distributed Databases

- You are expected to be familiar with:
  - Centralized database configuration and query processing and query optimization in a centralized database environment.
- If not, you need to study CS6302.module2.background



Distributed Databases

- In this module, we define:
  - Distributed Databases
  - Issues in distributed databases,
  - Query processing in general
  - Query optimization
    - Optimization process
    - Query transformation (equivalence rules)
  - Distributed query processing



Distributed Databases

- From our discussion in the last module, one could conclude that “distributed databases” is one form of database organization to represent and process the data. It is a natural extension of client-server and peer-to-peer paradigms.
- We will look at the major issues in a distributed database organization and then concentrate on query processing.

Distributed Databases

- Distributed Database Systems
  - A distributed database in an environment in which related data sources:
    - Are residing on different geographically distributed sites – data is closed to the application domain (s) that uses it.
    - Might be replicated – to improve performance.
    - Are slit (Fragmented), horizontally and/or vertically, and distributed among the sites – to balance the load and improve performance.

Distributed Databases

- Data Distribution
  - Data might be distributed to minimize communication cost and/or response time.
  - Data might be kept at the site where it was created to allow higher control and security.

Distributed Databases

- Data Replication
  - Data replication allows more availability (in the event of failure — if one site is down, data can be accessed from another site),
  - Data replication increases the degree of parallelism (reduce response time),
  - Data replication increases overhead on update.

Distributed Databases

- Data Replication
  - In general, data replication increases the performance of read operations and increases the availability of the data to read-only transactions.
  - However, data replication incurs overhead for update transactions, since all copies of the data must be synchronized. In addition, controlling concurrent updates by several transactions on replicated data is becoming very complex.

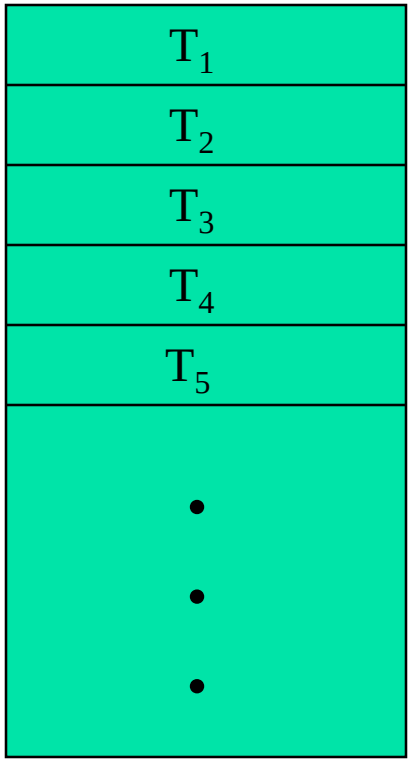
Distributed Databases

- Data Fragmentation – Horizontal fragmentation
  - In horizontal fragmentation, data set is partitioned (broken) into subsets with the same structure – original data set can be constructed by taking the union of the subsets.
  - Horizontal fragmentation keeps the subsets at the sites where they are used the most.
  - Horizontal fragmentation is lossless.

Distributed Databases

- Data Fragmentation – Horizontal fragmentation

$$T = \cup T_i \qquad 1 \leq i \leq n$$



Distributed Databases

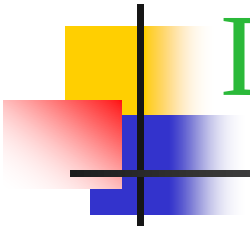
- Data Fragmentation – Vertical fragmentation
  - In vertical fragmentation, data set is decomposed (broken) into subsets with different attributes – Original data set is constructed by performing join on the fragments.
  - Vertical fragmentation is also lossless.

Distributed Databases

- Data Fragmentation – Vertical fragmentation

$$T_i = \Pi_{Li}(T) \qquad 1 \leq i \leq n$$

$$T = T_1 \bowtie T_2 \bowtie \ldots \bowtie T_n$$

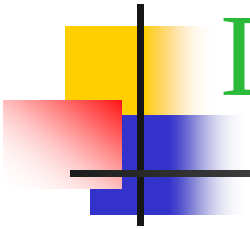


Distributed Databases

- Two types of applications access distributed databases:
  - Application that accesses data at the level of SQL statements.
  - Application that accesses data using only stored procedures provided by the database management system.
  - Only applications of the first type can access data directly and hence subject to employ query optimization, concurrency control, … strategies.

Distributed Databases

- Issues in Distributed Database Systems – Data Distribution
  - How should a distributed database be designed?
  - How should the data be distributed?



Distributed Databases

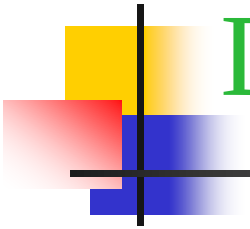
- Issues in Distributed Database Systems – Transparency
  - Distributed data should be managed transparently ─ details regarding the location of the data must be hidden. Transparency comes in different types:
    - Network transparency
    - Replication transparency
    - Fragmentation transparency

Distributed Databases

- Issues in Distributed Database Systems – Transparency
  - Location transparency (Network transparency) – Freedom from operational details of the network.
  - Replication transparency – Unaware from the existence of duplicated copies.
  - Fragmentation transparency – Unaware from the existence of fragments.

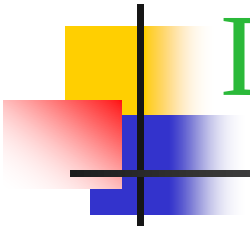
Distributed Databases

- Issues in Distributed Database Systems – Transparency
  - The degree of transparency is a compromise between:
    - Ease of use and the difficulty, and
    - The overhead cost of providing high levels of performance.



Distributed Databases

- Issues in Distributed Database Systems – Keeping track of data
  - Ability to keep track of the data distribution, fragmentation, and replication.
  - Application of catalog, meta-data,…



Distributed Databases

- Issues in Distributed Database Systems – Distributed Query processing
  - The ability to access remote sites and transmitting queries (sub-queries) and data among different sites via the network:
    - How should queries that access multiple databases be processed?
    - How do we improve query processing?

Distributed Databases

- Issues in Distributed Database Systems – Distributed Transaction management
  - The ability to devise execution strategies for queries and transactions that access data from more than one site and to synchronize the access to distributed data (efficiently) and maintain integrity of the whole database.

Distributed Databases

- Issues in Distributed Database Systems –

  Replicated data management

  - The ability to decide which copy of a replicated data set to access, to maintain the consistency of the replicated data, and to control number of replica.

Distributed Databases

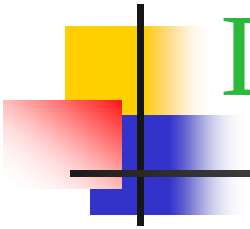
- Issues in Distributed Database Systems – Distributed database recovery
  - The ability to recover from individual site crashes and system failures.

Distributed Databases

- Issues in Distributed Database Systems – Security
  - The ability to authenticate users, and enforce authorization and access control.

Distributed Databases

- To appreciate the issue of query processing in a distributed environment, one has to have a good appreciation and understanding of query processing in a centralized database environment. This also brings out the issue of query optimization, equivalency between queries, and equivalence rules.

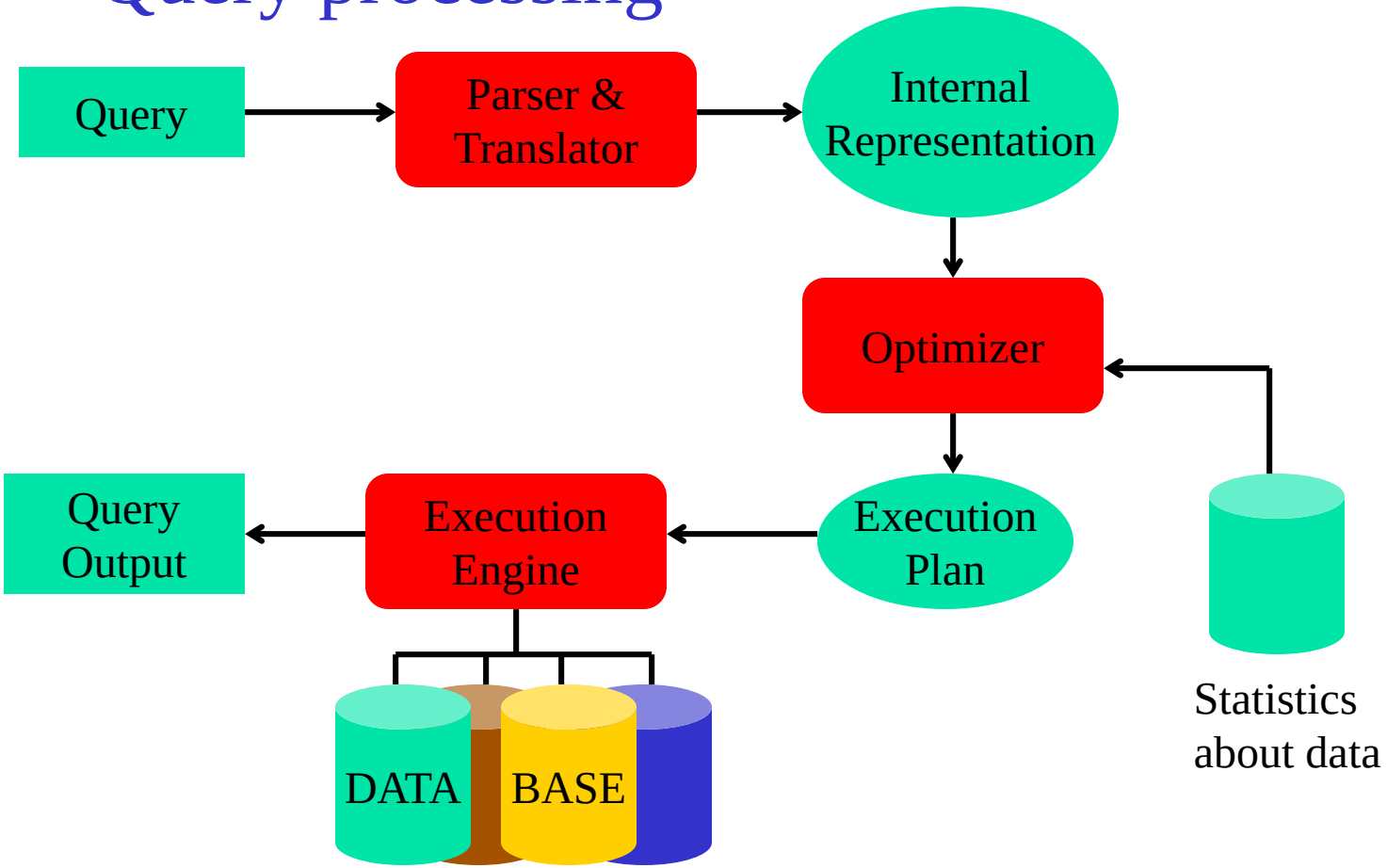


Distributed Databases

- Query processing
  - A query processing involves three steps:
    - Parsing and Translation
    - Optimization
    - Evaluation

Distributed Databases

- Query processing



Distributed Databases

- Query Optimization — A Simple Example
  - Get names of suppliers who supply part P_2:

```
SELECT    DISTINCT   Sname
FROM      S, SP
WHERE     S.S# = SP.S#
AND       SP.P# = 'P2';
```

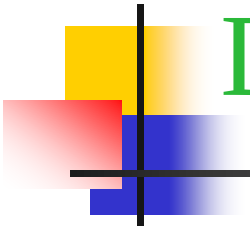
  - Suppose that the cardinality of *S* and *SP* are 100 and 10,000, respectively. Furthermore, assume 50 tuples in *SP* are for part P_2.

Distributed Databases

- Query Optimization — A Simple Example
  - Without an optimizer, the system will:
    - Generates Cartesian product of *S* and *SP*. This will generate a relation of size 1,000,000 tuples — Too large to be kept in the main memory.
    - Restricts results of previous step as specified by WHERE clause. This means reading 1,000,000 tuples of which 50 will be selected.
    - Projects the result of previous step over *Sname* to produce the final result.

Distributed Databases

- Query Optimization — A Simple Example
  - An Optimizer on the other hand:
    - Restricts *SP* to just the tuples for part P_2. This will involve reading 10,000 tuples, but produces a relation with 50 tuples.
    - Joins the result of the previous step with *S* relation over *S#*. This involves the retrieval of only 100 tuples and the generation of a relation with at most 50 tuples.
    - Projects the result of the last operation over *Sname*.



Distributed Databases

- Query Optimization — A Simple Example
  - If the number of tuples I/O's is used as the performance measure, then it is clear that the second approach is far faster that the first approach. In the first case we read about 3,000,000 tuples and in the second case we read about 10,000 tuples.
  - So a simple policy — doing restriction and then join instead of doing product and then a restriction sounds a good heuristic.

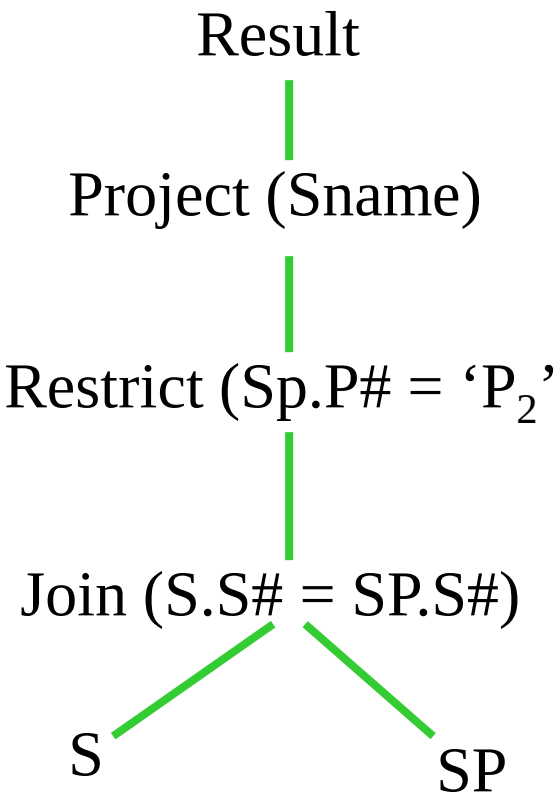
Distributed Databases

- Optimization Process
  - Cast the query into some internal representation — Convert the query to some internal representation that is more suitable for machine manipulation, relational algebra.
  - Now we can build a query tree very easily.

$$\Pi_{(Sname)}(\sigma_{P\# = \text{“P2”}}(S \bowtie_{S.S\# = SP.S\#} SP))$$

Distributed Databases

- Optimization Process



Distributed Databases

- Optimization Process
  - Choose candidate low-level procedure — After transferring the query into more desirable form, the optimizer must then decide how to evaluate the transformed query. At this stage issues such as:
    - existence of indexes or other access paths ─ To reduce I/O cost,
    - physical clustering of records ─ To reduce I/O cost,
    - size of base relations,
    - Selectivity factors,
    -
    -
    -

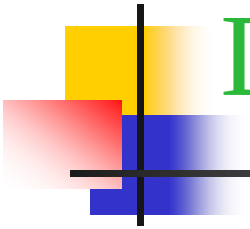
  comes into play.

Distributed Databases

- Optimization Process
  - Choosing the cheapest plan, naturally, requires a method for assigning a cost to any given plan — This cost formula should estimate the number of disk accesses, CPU utilization and execution time, space utilization,….

Distributed Databases

- Query processing – An Example
  - Factors such as number of accesses to the disks, CPU time, Communication cost must be taken into consideration to estimate cost of a plan.
  - Techniques such as pipelining and parallelism can be applied to execute basic operations.
  - Different algorithms can be developed to execute basic operations.



Distributed Databases

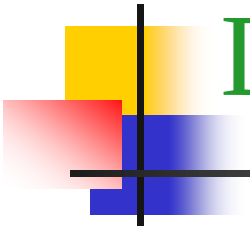
- Optimization Process
  - There are two main techniques for query optimization:
    - Heuristic rules
    - Systematically estimate
  - In this course, we will talk about the heuristic rules.

Distributed Databases

- Optimization Process – heuristic rules
  - Reduce the search space,
  - Reduce the size of intermediate results,
  - Perform selection operations as early as possible,
  - Perform projections early,
  - It is usually better to perform selections earlier than projections,
  - Convert Cartesian product to join.

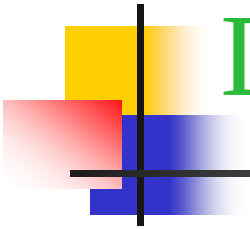
Distributed Databases

- Optimization Process – heuristic rules
  - Based on heuristic rules the optimizer uses equivalence relationships to reorder operations in a query for execution.



Distributed Databases

- Cost of Plan
  - The cost associated with each plan needs to be estimated. This will be accomplished by estimating the cost of each operation.
  - Factors such as: size of relation (s), underlying architecture, buffer size, size of the memory, “reduction factor” for each operation, … need to be taken into consideration.



Distributed Databases

- Assume the following relations:
  - Department (Dname, Dnumber, Mgr-ssn, …)
  - Project (Pname, Pnumber, Plocation, Dnum)
  - Employee (Fname, Lname, Ssn, Bdate, address, Dno, …)

Distributed Databases

- Query Optimization — An Example

```
SELECT    Pnumber, Dnum, Lname, Bdate,
          Address
FROM      Project, Department, Employee
WHERE     Dnum = Dnumber
AND       MGRSSN = SSN
AND       Plocation = 'California';
```

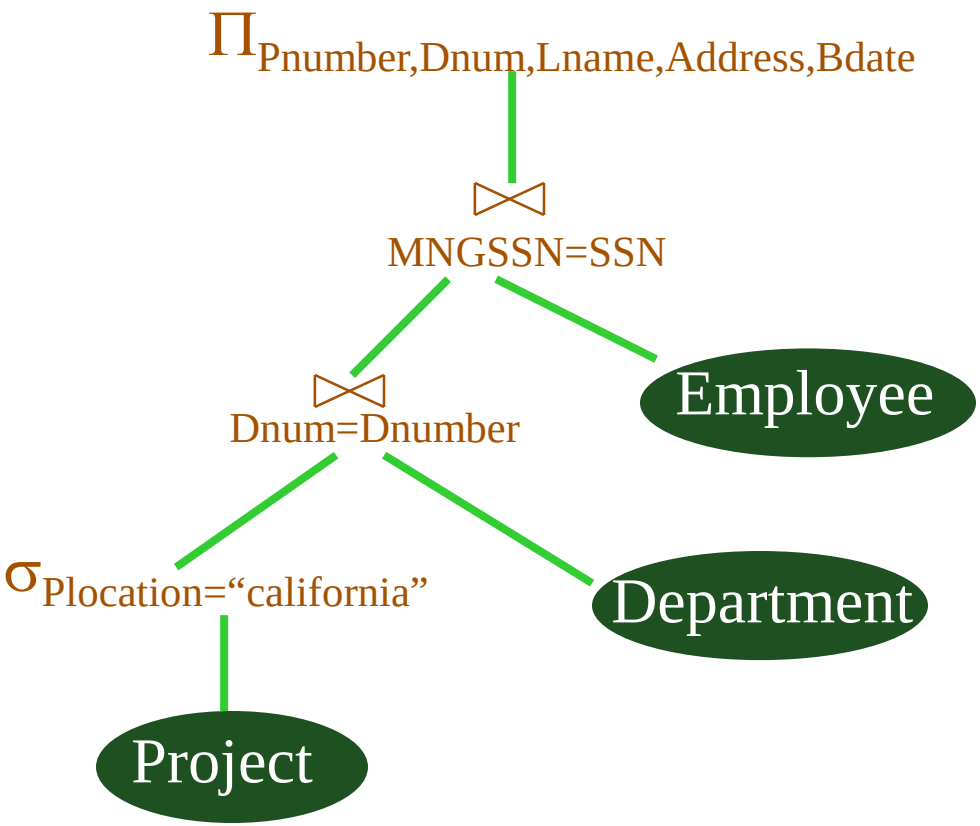
Distributed Databases

- Query Optimization — An Example
  - The above query based on the schemas of the relations can be translated into:

$$\Pi_{Pnumber,Dnum,Lname,Address,Bdate}(((\sigma_{Plocation=\text{“california”}}(Project)) \bowtie_{Dnum=Dnumber} (Department)) \bowtie_{MNGSSN=SSN} (Employee))$$

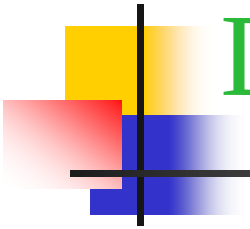
Distributed Databases

- Query Optimization — An Example



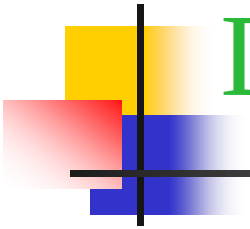
Distributed Databases

- Distributed Query processing
  - In a distributed database system, relative to the centralized database, several additional factors complicate query processing. These includes:
    - Data Transfer cost – size of data, communication cost, communication reliability,
    - The potential performance gain – data replication and fragmentation,
    - Potential parallelism – data replication and fragmentation,
    - Security, and
    - Processing capability and speed.



Distributed Databases

- Distributed Query processing
  - Query translation must be done correctly (same semantic as the original query) and efficiently.
  - In a centralized context, query execution strategies can be well expressed in an extension of relational algebra.



Distributed Databases

- Distributed Query processing
  - In a distributed system, relational algebra is not enough to express execution strategy. It must be supplemented with operations for exchanging data between sites. In addition, the distributed query processor must also select the best sites to process data and possibly the way data should be transferred.

Distributed Databases

- Example
  - Assume the following relations:

    EMP(ENO, ENAME, TITLE) and

    ASG(ENO, PNO, RESP, DUR)
  - Find the names of employees who are managing a project

```
SELECT   ENAME
FROM     EMP, ASG
WHERE    EMP.ENO = ASG.ENO ∧ RESP = “manager”
```

$$\Pi_{(ENAME)}(EMP \bowtie_{ENO} (\sigma_{RESP = \text{“Manager”}} (ASG)))$$

Distributed Databases

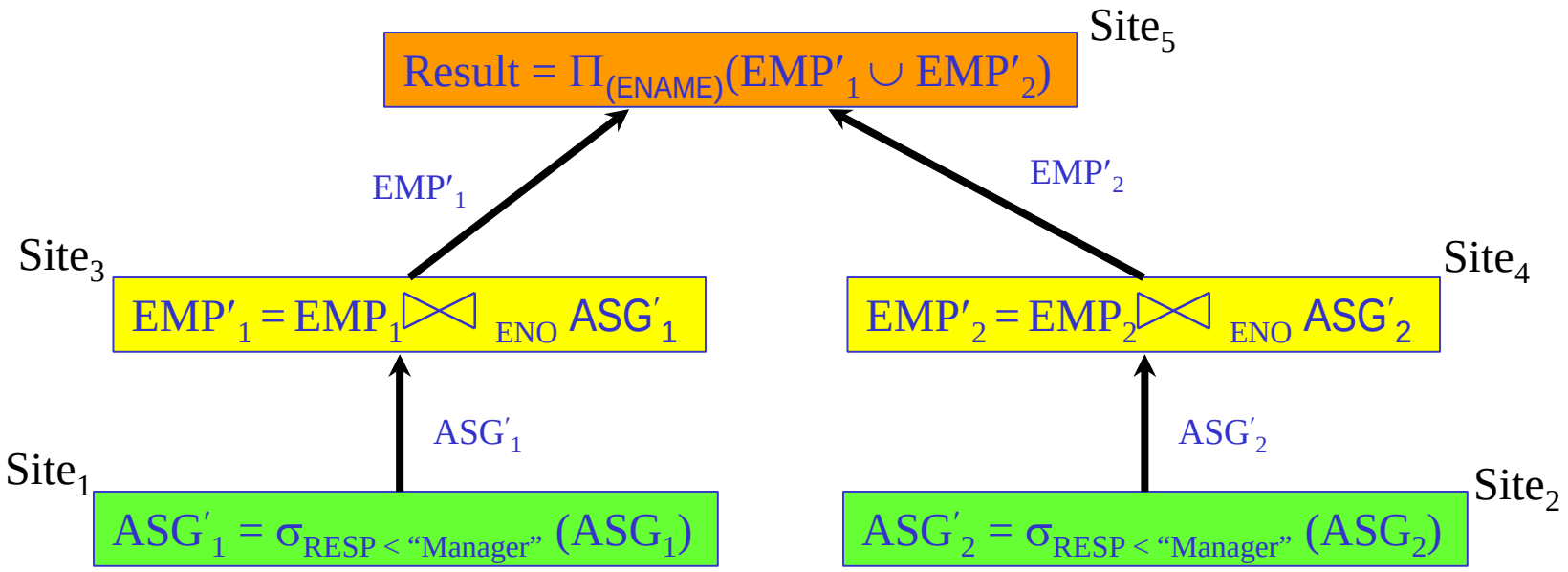
- Example
  - Assume that EMP and ASG are horizontally fragmented as follows:

    $EMP_1 = \sigma_{ENO \leq \text{“E3”}}(EMP)$ $\quad\quad$ $ASG_1 = \sigma_{ENO < \text{“E3”}}(ASG)$

    $EMP_2 = \sigma_{ENO < \text{“E3”}}(EMP)$ $\quad\quad$ $ASG_2 = \sigma_{ENO > \text{“E3”}}(ASG)$
  - Fragments of ASG_1, ASG_2, EMP_1, and EMP_2 are stored at sites 1, 2, 3, 4, respectively and the result is expected at site 5.

Distributed Databases

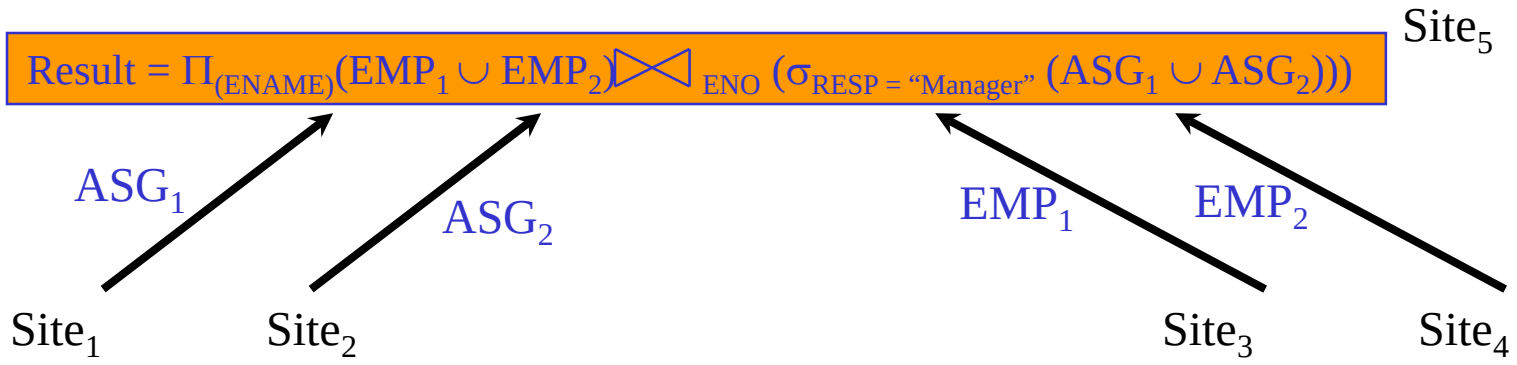
- Example



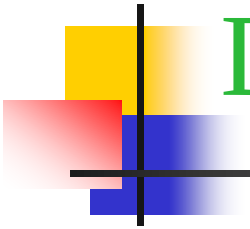
Strategy1

Distributed Databases

- Example



Strategy2



Distributed Databases

- Example
  - Assume tuple access (tupacc) is 1 unit and tuple transfer (tuptrans) is 10 units, relations EMP and ASG have 400 and 1000 tuples, respectively, and there are 20 managers. Further assume data is distributed uniformly among sites and ASG and EMP are locally clustered on attributes RESP and ENO.

Distributed Databases

- Example

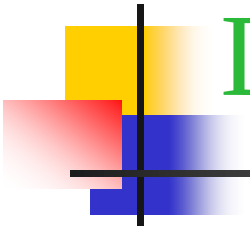
Total cost of strategy1

1) Produce ASG′ (10 + 10)*tupacc = 20
2) Transfer ASG′ (10 + 10)*tuptrans = 200
3) Produce EMP′ (10 + 10)*tupacc*2 = 40
4) Transfer EMP′ (10 + 10)*tuptrans = 200

Total cost of strategy1

1) Transfer EMP 400*tuptrans = 4000
2) Transfer ASG 1000*tuptrans = 10000
3) Produce ASG′ 1000*tupacc = 1000
4) Join EMP and ASG′ 400*20*tupacc = 8000

Note: in strategy2, the access methods to EMP and ASG are lost because of the data transfer.



Distributed Databases

- Distributed Query processing
  - Types of optimizations:
    - Exhaustive approach
    - Heuristic approach
      - Reduce search space at each site
      - Reduce size of intermediate results
      - Reduce communication cost
  - Optimization Overhead
    - Static optimization
    - Dynamic optimization
    - Hybrid

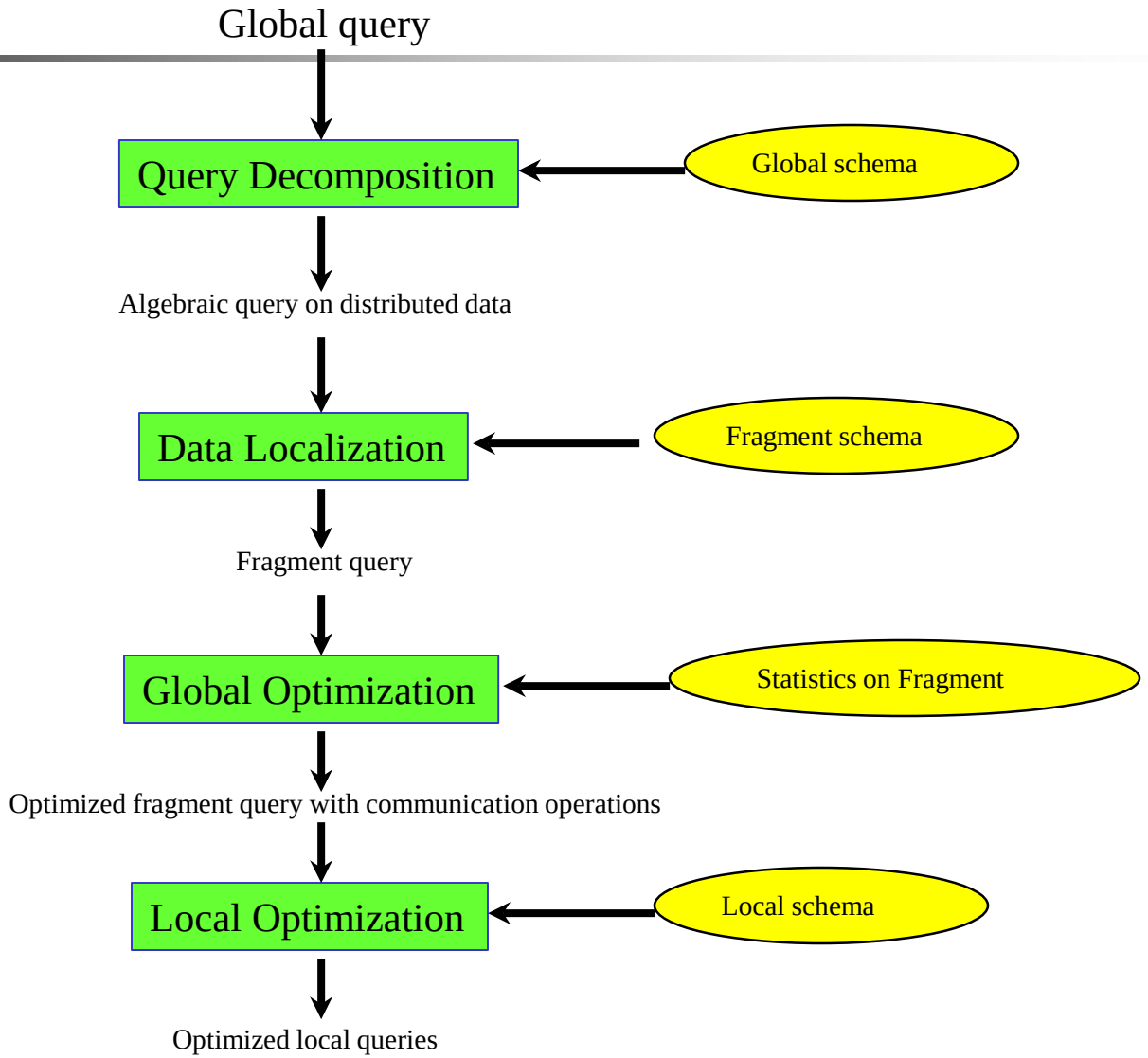
Distributed Databases

- Distributed Query processing
  - Statistics: Effectiveness of optimization (specially static optimization) relies on statistics on the databases.
  - Decision sites: When static optimization is employed, either a single site or multiple sites may participate in selection of the strategy for query processing. Most system use the centralized decision approach. Naturally, the centralized approach is easier to implement, however, it requires knowledge of the entire distributed databases. The distributed approach requires only local information. One can employ a hybrid approach where one site makes the major decisions and other sites can make local decisions.

Distributed Databases

- Distributed Query processing
  - Exploitation of the Network Topology
    - Local area network
    - Wide area network
  - Exploitation of Replicated Fragments: Global query expressed on global relations must be mapped into queries on physical fragments of relations (localization)
  - Use of Semijoins: A semijoin is of particular importance since it improves the processing of distributed join operations by reducing the size of data exchanged between sites. However, semijoins may result in an increase in the number of messages and local processing time.

Distributed Databases



Distributed Databases

- Distributed Query processing
  - Cost of an execution strategy can be expressed in terms of either total execution time or the response time.

Total-exec.time = T_{CPU} * #insts + $T_{I/O}$ * #I/Os + T_{MGS} * #MSGs + T_{TR} * #bytes

  - T_{CPU} * #insts + $T_{I/O}$ * #I/Os measures the local processing time and T_{MGS} * #MSGs + T_{TR} * #bytes represents the cost of communications.

Distributed Databases

- Distributed Query processing
  - Size of intermediate result(s) is one of the major factors affecting the cost:

    Length(A_i) denotes length of attribute A_i in bytes

    Card(Π_{Ai} (*R*)) denotes the number of distinct values of A_i *in R.*

    Card(*dom* [A_i]) denotes the cardinality of the domain A_i.

    Card(*R*) denotes number the entities (tuples) in *R*.

    Size(*R*) = *Card*(*R*) * *Length*(*R*) where *Length*(*R*) is the length of each entity (tuple) in *R*.

Distributed Databases

- Distributed Query processing

  $Card(\sigma_F(R)) = SF_S(F) * card(R)$ denotes the number of data entities (tuples) responding to a selection condition.

  $SF_S(F)$ is the selection factor which is dependent on selection formula and can be determined as follows:

  $SF_S(A = value) = 1/(Card(\Pi_{Ai}(R)))$

  $SF_S(A > value) = (max(A) - value)/(max(A) - min(A))$

  $SF_S(A < value) = (value - min(A))/(max(A) - min(A))$

  $SF_S(p(A_i) \wedge p(A_j)) = SF_S(p(A_i)) * SF_S(p(A_j))$

  $SF_S(p(A_i) \vee p(A_j)) = SF_S(p(A_i)) + SF_S(p(A_j)) - SF_S(p(A_i)) * SF_S(p(A_j))$

  $SF_S(A \varepsilon \{value\}) = SF_S(A = value) * Card(\{value\})$

  Where $p(A_i)$ and $p(A_j)$ are the predicates over respective attributes.

Distributed Databases

- Distributed Query processing

  Join selectivity factor is defined as:

  $SF_J(R, S) = Card(R \bowtie S) / (Card(R) * Card(S))$

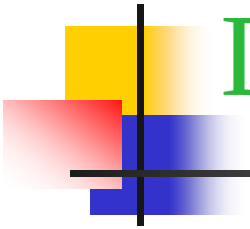
  And hence $Card(R \bowtie S) = SF_J(R, S) * (Card(R) * Card(S))$

  For Cartesian product $Card(R \times S) = Card(R) * Card(S)$

  For projection $Card(\Pi_A(R)) = card(R)$ if A contains a key

Distributed Databases

- Distributed Query processing – Query transformation
  - Based on the query, location of data sets, size of the data sets, communication cost, processing capability, … a dynamic strategy should be laid out.
  - According to the strategy, then the query is decomposed into sub-queries.
  - Sub-queries are sent to the designated sites for execution.



Distributed Databases

- Distributed Query Processing
  - In distributed systems, several additional factors further complicate query processing:
    - Cost of data transmission – this includes intermediate data and the final result.
    - Hence, the query optimization algorithm must attempt to reduce the amount of data transfer.

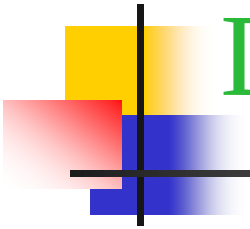
Distributed Databases

- Distributed Query Processing
  - Use of semijoin – the idea is to reduce the number of tuples in a relation before transferring it to another site.
  - If A and B are domain compatible attributes of R and S then we have $R \ltimes_{A=B} S$
  - Note semijoin is not commutative:

$$R \ltimes S \neq S \ltimes R$$

Distributed Databases

- Questions
  - Calculate selectivity factor of semijoin operation and cardinality of semijoin operation?
  - Calculate cardinality of Union and Difference operations?



Distributed Databases

- Distributed Query Processing
  - A query is decomposed into a set of sub-queries that can be executed at the individual sites.
  - A strategy for combining the results of the sub-queries to form the query result must be generated:
    - An estimate on the size of data transmission must be made to minimize communication cost:
      - In case of data replication and fragmentation attempt must be made to choose the closest replica and/or fragments.
      - If possible, semijoin operation should be performed to reduce data transfer size.
    - Where ever possible attempt must be made to enforce heuristics and equivalence rules, at each site, to reduce the execution cost.

Distributed Databases

- Distributed Query Processing – An example
  - Assume the following queries on two relations:

$$\Pi_{FNAME,LNAME,DNAME}((EMPLOYEE) \bowtie_{DNO=DNUMBER} (DEPARTMENT))$$

Site1: EMPLOYEE

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	• • •	DNO

10,000 records
Each record is 100 bytes
SSN field is 9 bytesFNAME field is 15 bytes
DNO field is 4 bytes LNAME field is 15 bytes

Distributed Databases

- Distributed Query Processing – An example

Site2: DEPARTMENT

DNAME	DNUMBER	MGRSSN	MGRSTARTDATE	• • •

100 records
Each record is 35 bytes
DNUMBER field is 4 bytes DNAME field is 10 bytes
MGRSSN field is 9bytes

Distributed Databases

- Distributed Query Processing – An example
  - The result of this query include 10,000 records, each 40 bytes long.
  - The query is generated at site3.

Distributed Databases

- Distributed Query Processing – An example
  - Transfer both relations to site3 and process the request there. Here we have to transfer 1,000,000+3500 bytes.
  - Transfer employee relation to site2, process the request at site2, and transfer the result to site3. Here we have to transfer 400,000+1,000,000 bytes.
  - Transfer department relation to site1, process the request at site1, and transfer the result to site3. Here we have to transfer 400,000+3500 bytes.

Distributed Databases

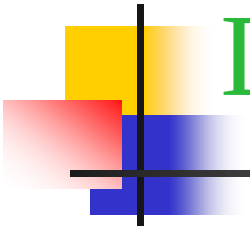
- Distributed Query Processing – An example
  - Suppose we have the following query initiated at site3:

$$\Pi_{FNAME,LNAME,DNAME}((DEPARTMENT) \bowtie_{MGRSSN=SSN} (EMPLOYEE))$$

  - Note that this query generate 100 records.

Distributed Databases

- Distributed Query Processing – An example
  - Transfer both relations to site3 and process the request there. Here we have to transfer 1,000,000+3500 bytes.
  - Transfer employee relation to site2, process the request at site2, and transfer the result to site3. Here we have to transfer 4000+1,000,000 bytes.
  - Transfer department relation to site1, process the request at site1, and transfer the result to site3. Here we have to transfer 4000+3500 bytes.



Distributed Databases

- Question
  - What is a semijoin Strategy?
  - Apply equivalence relations to the aforementioned examples, enumerate different strategies, and calculate the data transfer accordingly.
  - Apply semijoin operation to carry out the aforementioned queries and calculate the data transfer size.
  - How can we perform database operations on a parallel system (assume relational model)?