Distributed Databases (Ref: Chapters 20-23 in Silberchatz' Text)

Database System Concepts, 6th Ed.

Chapter 19: Distributed Databases

- Homogeneous and Heterogeneous Distributed Databases
- Distributed Data Storage
- Distributed Transactions
- Commit Protocols
- Concurrency Control in Distributed Databases
- Availability
- Distributed Query Processing
- Heterogeneous Distributed Databases

Homogeneous and Heterogeneous Distributed Databases

Database System Concepts, 6th Ed.

Distributed Database System

- A distributed database: a logically interrelated collection of shared data physically distributed over a computer network
- Distributed DBMS: the software system that permits the management of the distributed database and makes the distribution transparent to users.
- Transactions may access data at one or more sites

Example

- Draw a diagram that shows a distributed database system.
  - Answer:

Types of Distributed Database Systems

- In a homogeneous distributed database system
  - All sites have identical software
  - Are aware of each other and agree to cooperate in processing user requests.
  - Each site surrenders part of its autonomy in terms of right to change schemas or software
  - Appears to user as a single system
- In a heterogeneous distributed database system
  - Different sites may use different schemas and software
    - Difference in schema is a major problem for query processing
    - Difference in software is a major problem for transaction processing
  - Sites may not be aware of each other and may provide only limited facilities for cooperation in transaction processing

Distributed Data Storage

Database System Concepts, 6th Ed.

Distributed Data Storage

- Assume relational data model
- Replication
  - System maintains multiple copies of data, stored in different sites, for faster retrieval and fault tolerance.
- Fragmentation
  - Relation is partitioned into several fragments stored in distinct sites
- Replication and fragmentation can be combined
  - Relation is partitioned into several fragments: system maintains several identical replicas of each such fragment.

Data Replication

- A relation or fragment of a relation is **replicated** if it is stored redundantly in two or more sites.
- **Full replication** of a relation is the case where the relation is stored at all sites.
- Fully redundant databases are those in which every site contains a copy of the entire database.

Data Replication (Cont.)

- What are the Advantages of Replication?

Answer:

- What are the Disadvantages of Replication?

Answer:

Data Fragmentation

- Division of relation r into fragments r_1, r_2, …, r_n which contain sufficient information to reconstruct relation r.
- **Horizontal fragmentation**: each tuple of *r* is assigned to one or more fragments
- **Vertical fragmentation**: the schema for relation *r* is split into several smaller schemas
  - All schemas must contain a common candidate key (or superkey) to ensure lossless join property.
  - A special attribute, the tuple-id attribute may be added to each schema to serve as a candidate key.

Horizontal and Vertical Fragmentation: Examples

- Answer:

Advantages of Fragmentation

- Advantages of Horizontal Fragmentation?
  - Answer:
- Advantages of Vertical Fragmentation?
  - Answer:
- Vertical and horizontal fragmentation can be mixed.
  - Fragments may be successively fragmented to an arbitrary depth.

Data Transparency

- **Data transparency**: Degree to which system user may remain unaware of the details of how and where the data items are stored in a distributed system
- Consider transparency issues in relation to:
  - **Fragmentation transparency**
  - **Replication transparency**
  - **Location transparency**

Naming of Data Items - Criteria

1. Every data item must have a system-wide unique name.
2. It should be possible to find the location of data items efficiently.
3. It should be possible to change the location of data items transparently.
4. Each site should be able to create new data items autonomously.

Centralized Scheme - Name Server

- Structure:
  - name server assigns all names
  - each site maintains a record of local data items
  - sites ask name server to locate non-local data items
- What are the Advantages of this scheme?

Answer:

- What are the Disadvantages of this scheme?

Answer:

Use of Aliases

- Alternative to centralized scheme: each site prefixes its own site identifier to any name that it generates i.e., *site* 17.a*ccount.*
  - Fulfills having a unique identifier, and avoids problems associated with central control.
  - However, fails to achieve network transparency.
- Solution: Create a set of **aliases** for data items; Store the mapping of aliases to the real names at each site.
- What are the advantages of this scheme?

  Answer:

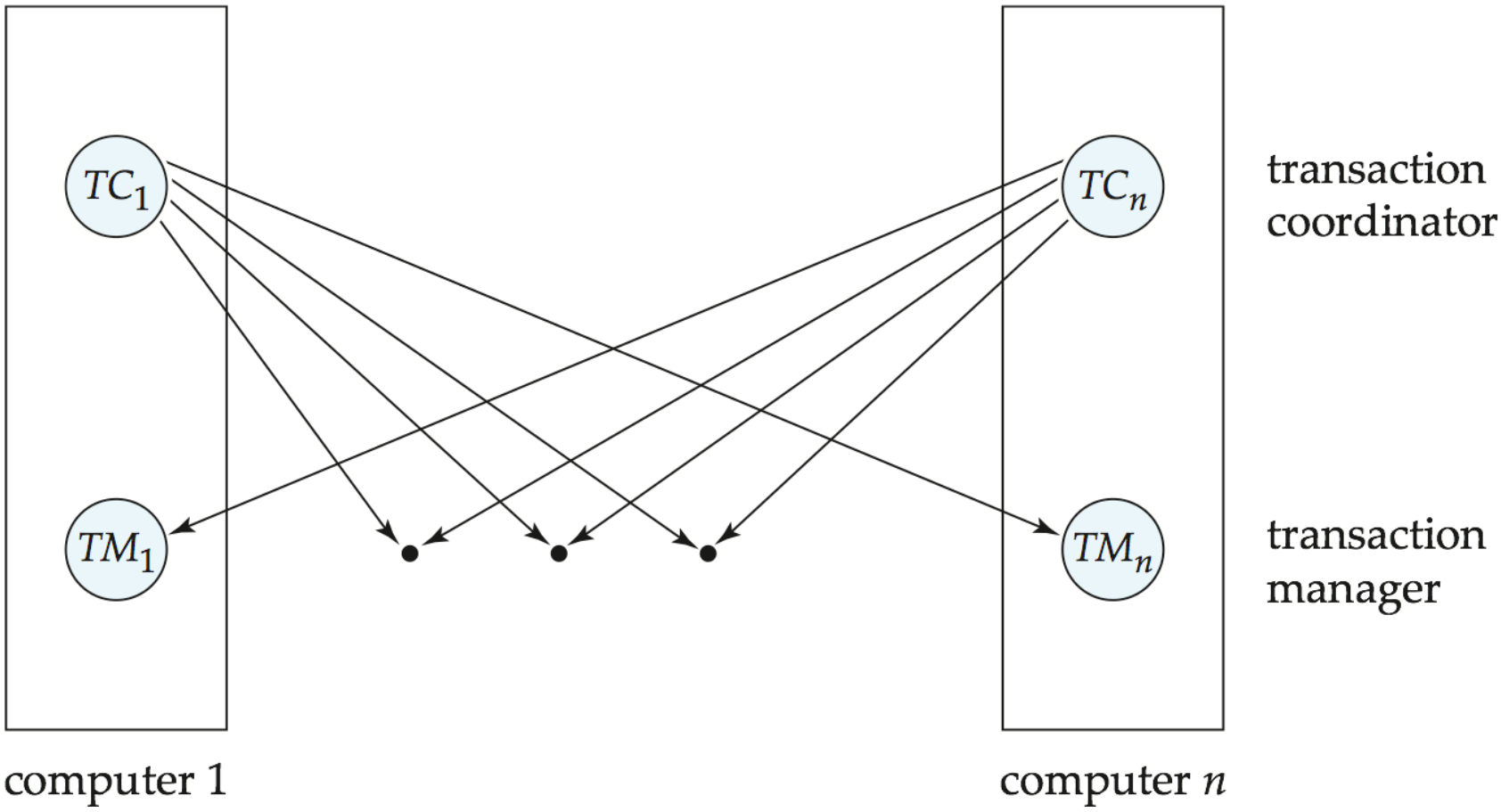
Distributed Transactions

Database System Concepts, 6th Ed.

Distributed Transactions

- Transaction may access data at several sites.
- Each site has a local **transaction manager** responsible for:
  - Maintaining a log for recovery purposes
  - Participating in an appropriate concurrency control scheme to coordinate the concurrent execution of the transactions executing at that site.
- Each site has a **transaction coordinator**, which is responsible for:
  - Starting the execution of transactions that originate at the site.
  - Distributing subtransactions at appropriate sites for execution.
  - Coordinating the termination of each transaction that originates at the site, which may result in the transaction being committed at all sites or aborted at all sites.

Transaction System Architecture



System Failure Modes

- What are the failures that are unique to distributed systems?
- Answer:

Commit Protocols

Database System Concepts, 6th Ed.

Commit Protocols

- What are Commit protocols used for?
- Answer:

- The *two-phase commit* (2PC) protocol:

- The *three-phase commit* (3PC) protocol:

Two Phase Commit Protocol (2PC)

- Assumes **fail-stop** model – failed sites simply stop working, and do not cause any other harm, such as sending incorrect messages to other sites.
- Execution of the protocol is initiated by the coordinator after the last step of the transaction has been reached.
- The protocol involves all the local sites at which the transaction executed
- Let T be a transaction initiated at site S_i, and let the transaction coordinator at S_i be C_i

Phase 1: Obtaining a Decision – How Does It Work?

- Answer:

Phase 2: Recording the Decision – How Does It Work?

- Answer:

How to Handle Site Failure (participating site S_k fails)?

- Answer:

How to Handle Coordinator Failure?

- Answer:

How to Handle Network Partition?

- Answer:

Concurrency Control

Database System Concepts, 6th Ed.

Concurrency Control

- Modify concurrency control schemes for use in distributed environment.
- We assume that each site participates in the execution of a commit protocol to ensure global transaction automicity.
- We assume all replicas of any item are updated

Serializability

- What are Local and Global Serializability?
- Why do we need Global Serializability?
- Answer:

Single-Lock-Manager Approach (or Centralized-Lock-Manager)

- System maintains a *single* lock manager that resides in a *single* chosen site, say S_i
- When a transaction needs to lock a data item, it sends a lock request to S_i and lock manager determines whether the lock can be granted immediately
  - If yes, lock manager sends a message to the site which initiated the request
  - If no, request is delayed until it can be granted, at which time a message is sent to the initiating site

Single-Lock-Manager Approach (Cont.)

- The transaction can read the data item from *any* one of the sites at which a replica of the data item resides.
- Writes must be performed on all replicas of a data item
- What are Advantages of this scheme?
  - Answer:

- What are the Disadvantages of this scheme?
  - Answer:

Distributed-Lock-Manager Approach (or Multiple-Coordinator-Lock-Manager)

- In this approach, functionality of locking is implemented by a lock manager at each site
  - Lock managers control access to local data items
    - But special protocols may be used for replicas
- What are the Advantages?
  - Answer:
- What are the Disadvantages?
  - Answer:
- Several variants of this approach
  - Primary copy
  - Majority protocol
  - Biased protocol
  - Quorum consensus

Primary Copy

- For each data item X, choose one replica of X to be the **primary copy** (other replicas of X are secondary copies)
  - Site containing the replica is called the **primary site** for that data item
  - Different data items can have different primary sites
- When a transaction needs to lock a data item *Q*, it requests a lock at the primary site of *Q*.
  - Implicitly gets lock on all replicas of the data item
- Benefit?
  - Answer:
- Drawback?
  - Answer:

Majority Protocol

- Local lock manager at each site administers lock and unlock requests for data items stored at that site.
- When a transaction wishes to lock an unreplicated data item *Q* residing at site S_i, a message is sent to S_i 's lock manager.
  - If *Q* is locked in an incompatible mode, then the request is delayed until it can be granted.
  - When the lock request can be granted, the lock manager sends a message back to the initiator indicating that the lock request has been granted.

Majority Protocol (Cont.)

- In case of replicated data
  - If Q is replicated at n sites, then a lock request message must be sent to more than half of the n sites in which Q is stored.
  - The transaction does not operate on Q until it has obtained a lock on a majority of the replicas of Q.
  - When writing the data item, transaction performs writes on *all* replicas.
- Benefit?
  - Answer:
- Drawback?
  - Answer:

Biased Protocol

- Local lock manager at each site as in majority protocol, however, requests for shared locks are handled differently than requests for exclusive locks.
- **Shared locks**. When a transaction needs to lock data item *Q*, it simply requests a lock on *Q* from the lock manager at one site containing a replica of *Q*.
- **Exclusive locks**. When transaction needs to lock data item *Q*, it requests a lock on *Q* from the lock manager at all sites containing a replica of *Q*.
- Advantage?
  - Answer:
- Disadvantage?
  - Answer:

Quorum Consensus Protocol

- A generalization of both majority and biased protocols
- Each site is assigned a non-negative weight.
  - Let S be the total of all site weights where data item X resides
- For each data item X, choose two values **read quorum** Q_r and **write quorum** Q_w
  - Such that $Q_r + Q_w > S$ and $2 * Q_w > S$
  - Quorums can be chosen (and S computed) separately for each item
- Each read must lock enough replicas that the sum of the site weights is >= Q_r
- Each write must lock enough replicas that the sum of the site weights is >= Q_w
- For now we assume all replicas are written
  - Extensions to allow some sites to be unavailable described later

Quorum Consensus Protocol (cont.)

- What are the benefits of this protocol?
  - Answer:

Replication with Weak Consistency

- Many commercial databases support replication of data with weak degrees of consistency (i.e., without a guarantee of serializabiliy)
- E.g., **master-slave replication**: updates are performed at a single "master" site, and propagated to "slave" sites.
  - Propagation is not part of the update transaction: its is decoupled
    - May be immediately after transaction commits
    - May be periodic
  - Data may only be read at slave sites, not updated
    - No need to obtain locks at any remote site
  - Particularly useful for distributing information
    - E.g., from central office to branch-office
  - Also useful for running read-only queries offline from the main database

Replication with Weak Consistency (Cont.)

- Replicas should see a **transaction-consistent snapshot** of the database
  - That is, a state of the database reflecting all effects of all transactions up to some point in the serialization order, and no effects of any later transactions.
- E.g., Oracle provides a **create snapshot** statement to create a snapshot of a relation or a set of relations at a remote site
  - snapshot refresh either by recomputation or by incremental update
  - Automatic refresh (continuous or periodic) or manual refresh

Deadlock Handling

Consider the following two transactions and history, with item X and transaction T_1 at site 1, and item Y and transaction T_2 at site 2:

T_1: write (X)
write (Y)

T_2: write (Y)
write (X)

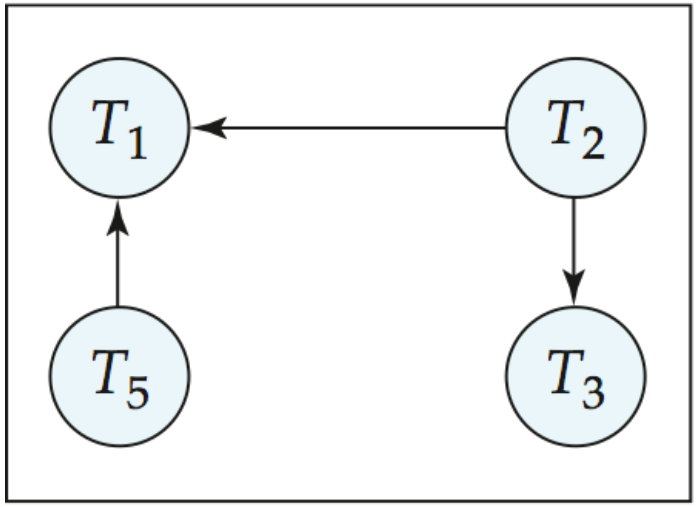
X-lock on X write (X)	
	X-lock on Y write (Y) wait for X-lock on X
Wait for X-lock on Y	

Result: deadlock which cannot be detected locally at either site

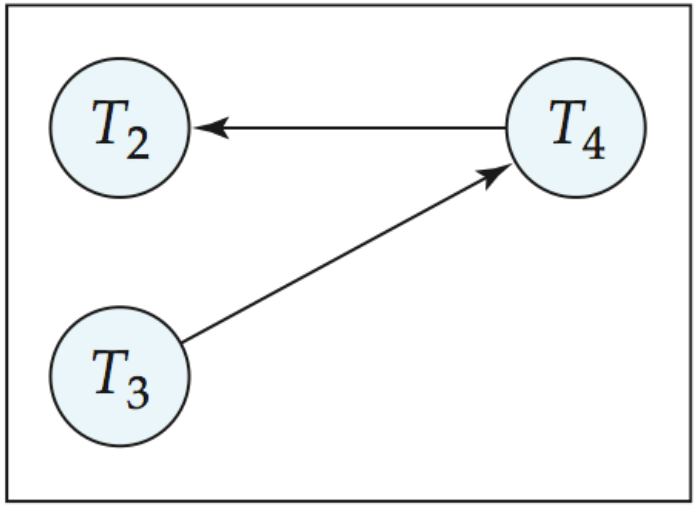
Centralized Approach

- A global wait-for graph is constructed and maintained in a *single* site; the deadlock-detection coordinator
  - *Real graph*: Real state of the system.
  - *Constructed graph*: Approximation generated by the controller during the execution of its algorithm.
- the global wait-for graph can be constructed when:
  - a new edge is inserted in or removed from one of the local wait-for graphs.
  - a number of changes have occurred in a local wait-for graph.
  - the coordinator needs to invoke cycle-detection.
- If the coordinator finds a cycle, it selects a victim and notifies all sites. The sites roll back the victim transaction.

Local and Global Wait-For Graphs

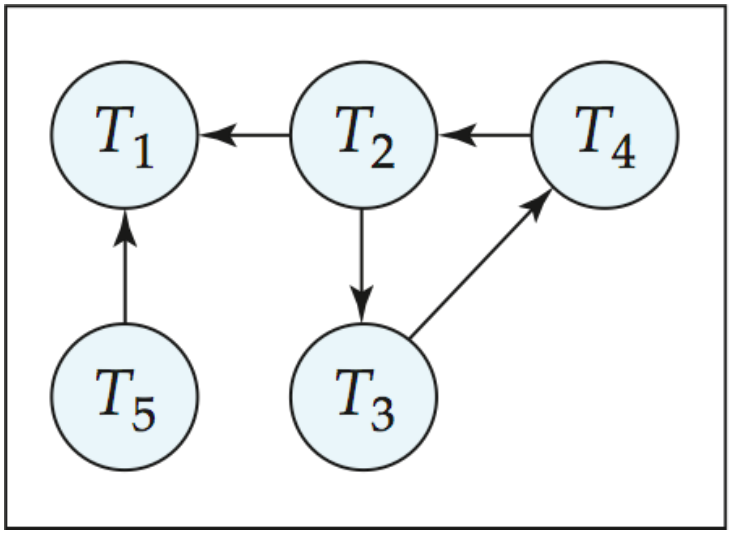


site S_1



site S_2

Local



Global

Availability

Availability

- High availability: time for which system is not fully usable should be extremely low (e.g., 99.99% availability)
- Robustness: ability of system to function spite of failures of components
- Failures are more likely in large distributed systems
- To be robust, a distributed system must
  - Detect failures
  - Reconfigure the system so computation may continue
  - Recovery/reintegration when a site or link is repaired
- Failure detection: distinguishing link failure from site failure is hard
  - (partial) solution: have multiple links, multiple link failure is likely a site failure

Reconfiguration

- Reconfiguration:
  - Abort all transactions that were active at a failed site
    - Making them wait could interfere with other transactions since they may hold locks on other sites
    - However, in case only some replicas of a data item failed, it may be possible to continue transactions that had accessed data at a failed site (more on this later)
  - If replicated data items were at failed site, update system catalog to remove them from the list of replicas.
    - This should be reversed when failed site recovers, but additional care needs to be taken to bring values up to date
  - If a failed site was a central server for some subsystem, an **election** must be held to determine the new server
    - E.g., name server, concurrency coordinator, global deadlock detector

Reconfiguration (Cont.)

- Since network partition may not be distinguishable from site failure, the following situations must be avoided
  - Two or more central servers elected in distinct partitions
  - More than one partition updates a replicated data item
- Updates must be able to continue even if some sites are down
- Solution: majority based approach
  - Alternative of “read one write all available” is tantalizing but causes problems

Majority-Based Approach

- The majority protocol for distributed concurrency control can be modified to work even if some sites are unavailable
  - Each replica of each item has a **version number** which is updated when the replica is updated, as outlined below
  - A lock request is sent to at least ½ the sites at which item replicas are stored and operation continues only when a lock is obtained on a majority of the sites
  - Read operations look at all replicas locked, and read the value from the replica with largest version number
    - May write this value and version number back to replicas with lower version numbers (no need to obtain locks on all replicas for this task)

Majority-Based Approach

- Majority protocol (Cont.)
  - Write operations
    - find highest version number like reads, and set new version number to old highest version + 1
    - Writes are then performed on all locked replicas and version number on these replicas is set to new version number
  - Failures (network and site) cause no problems as long as
    - Sites at commit contain a majority of replicas of any updated data items
    - During reads a majority of replicas are available to find version numbers
    - Subject to above, 2 phase commit can be used to update replicas
  - Note: reads are guaranteed to see latest version of data item
  - Reintegration is trivial: nothing needs to be done
- Quorum consensus algorithm can be similarly extended

Site Reintegration

- When failed site recovers, it must catch up with all updates that it missed while it was down
  - Problem: updates may be happening to items whose replica is stored at the site while the site is recovering
  - Solution 1: halt all updates on system while reintegrating a site
    - Unacceptable disruption
  - Solution 2: lock all replicas of all data items at the site, update to latest version, then release locks
    - Other solutions with better concurrency also available

Coordinator Selection

- **Backup coordinators**
  - site which maintains enough information locally to assume the role of coordinator if the actual coordinator fails
  - executes the same algorithms and maintains the same internal state information as the actual coordinator fails executes state information as the actual coordinator
  - allows fast recovery from coordinator failure but involves overhead during normal processing.
- **Election algorithms**
  - used to elect a new coordinator in case of failures
  - Example: Bully Algorithm - applicable to systems where every site can send a message to every other site.

Bully Algorithm

- If site S_i sends a request that is not answered by the coordinator within a time interval *T*, assume that the coordinator has failed S_i tries to elect itself as the new coordinator.
- S_i sends an election message to every site with a higher identification number, S_i then waits for any of these processes to answer within *T*.
- If no response within *T*, assume that all sites with number greater than *i* have failed, S_i elects itself the new coordinator.
- If answer is received S_i begins time interval *T*', waiting to receive a message that a site with a higher identification number has been elected.

Bully Algorithm (Cont.)

- If no message is sent within *T*', assume the site with a higher number has failed; S_i restarts the algorithm.
- After a failed site recovers, it immediately begins execution of the same algorithm.
- If there are no active sites with higher numbers, the recovered site forces all processes with lower numbers to let it become the coordinator site, even if there is a currently active coordinator with a lower number.

Distributed Query Processing

Database System Concepts, 6th Ed.

Distributed Query Processing

- For centralized systems, the primary criterion for measuring the cost of a particular strategy is the number of disk accesses.
- In a distributed system, other issues must be taken into account:
  - The cost of a data transmission over the network.
  - The potential gain in performance from having several sites process parts of the query in parallel.

Query Transformation

- Translating algebraic queries on fragments.
  - It must be possible to construct relation *r* from its fragments
  - Replace relation *r* by the expression to construct relation *r* from its fragments
- Show an example of a query and the application of query transformation on it for query processing:
  - Answer:

Example Query (Cont.)

Simple Join Processing

- Example: Consider the following relational algebra expression in which the three relations are neither replicated nor fragmented:

  $account \bowtie depositor \bowtie branch$

- *account* is stored at site S_1
- *depositor* at S_2
- *branch* at S_3
- For a query issued at site S_I, the system needs to produce the result at site S_I

Possible Query Processing Strategies

- What are the possible query processing strategies to process the given example query?
- Answer:

- What factors do we need to consider when choosing a query processing strategy?
- Answer:

Semijoin Strategy

- Let r_1 be a relation with schema R_1 stores at site S_1

  Let r_2 be a relation with schema R_2 stores at site S_2
- Evaluate the expression $r_1 \bowtie r_2$.
- Suppose we want to obtain the result at S_1
- Suppose there are many tuples in r_2 that do not join with any tuple in r_1
- A possible strategy:
  - Answer:

Verification of Correctness of the Above Example

- Answer:

Formal Definition

- The **semijoin** of r_1 with r_2, is denoted by:

$$r_1 \ltimes r_2$$

- It is defined by:

$$\Pi_{R1} (r_1 \bowtie r_2)$$

- Thus, $r_1 \ltimes r_2$ selects those tuples of r_1 that contributed to $r_1 \bowtie r_2$.
- In step 3 above, $temp_2 = r_2 \ltimes r_1$.
- For joins of several relations, the above strategy can be extended to a series of semijoin steps.

Join Strategies that Exploit Parallelism

- Example: Consider $r_1 \bowtie r_2 \bowtie r_3 \bowtie r_4$ where relation r_i is stored at site S_i. The result must be presented at site S_1.
- Show a join strategy that exploits parallelism to process the above join query.
  - Answer:

Heterogeneous Distributed Databases

Database System Concepts, 6th Ed.

Heterogeneous Distributed Databases

- Many database applications require data from a variety of preexisting databases located in a heterogeneous collection of hardware and software platforms
- Data models may differ (hierarchical, relational, etc.)
- Transaction commit protocols may be incompatible
- Concurrency control may be based on different techniques (locking, timestamping, etc.)
- System-level details almost certainly are totally incompatible.
- A **multidatabase system** is a software layer on top of existing database systems, which is designed to manipulate information in heterogeneous databases
  - Creates an illusion of logical database integration without any physical database integration

What are the Advantages of Heterogeneous Distributed Databases?

- **Answer:**

Unified View of Data

- Agreement on a common data model
  - Typically the relational model
- Agreement on a common conceptual schema
  - Different names for same relation/attribute
  - Same relation/attribute name means different things
- Agreement on a single representation of shared data
  - E.g., data types, precision,
  - Character sets
    - ASCII vs EBCDIC
    - Sort order variations
- Agreement on units of measure
- Variations in names
  - E.g., Köln vs Cologne, Mumbai vs Bombay

Query Processing

- Several issues in query processing in a heterogeneous database
- Schema translation
  - Write a **wrapper** for each data source to translate data to a global schema
  - Wrappers must also translate updates on global schema to updates on local schema
- Limited query capabilities
  - Some data sources allow only restricted forms of selections
    - E.g., web forms, flat file data sources
  - Queries have to be broken up and processed partly at the source and partly at a different site
- Removal of duplicate information when sites have overlapping information
  - Decide which sites to execute query
- Global query optimization

Mediator Systems

- **Mediator** systems are systems that integrate multiple heterogeneous data sources by providing an integrated global view, and providing query facilities on global view
  - Unlike full fledged multidatabase systems, mediators generally do not bother about transaction processing
  - But the terms mediator and multidatabase are sometimes used interchangeably
  - The term **virtual database** is also used to refer to mediator/multidatabase systems

End of Topic 4