Mobile and Heterogeneous databases

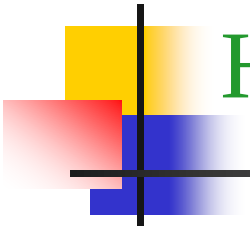
Heterogeneous Distributed Databases

Concept, Data Integration, and Approaches

A.R. Hurson

Computer Science

Missouri Science & Technology



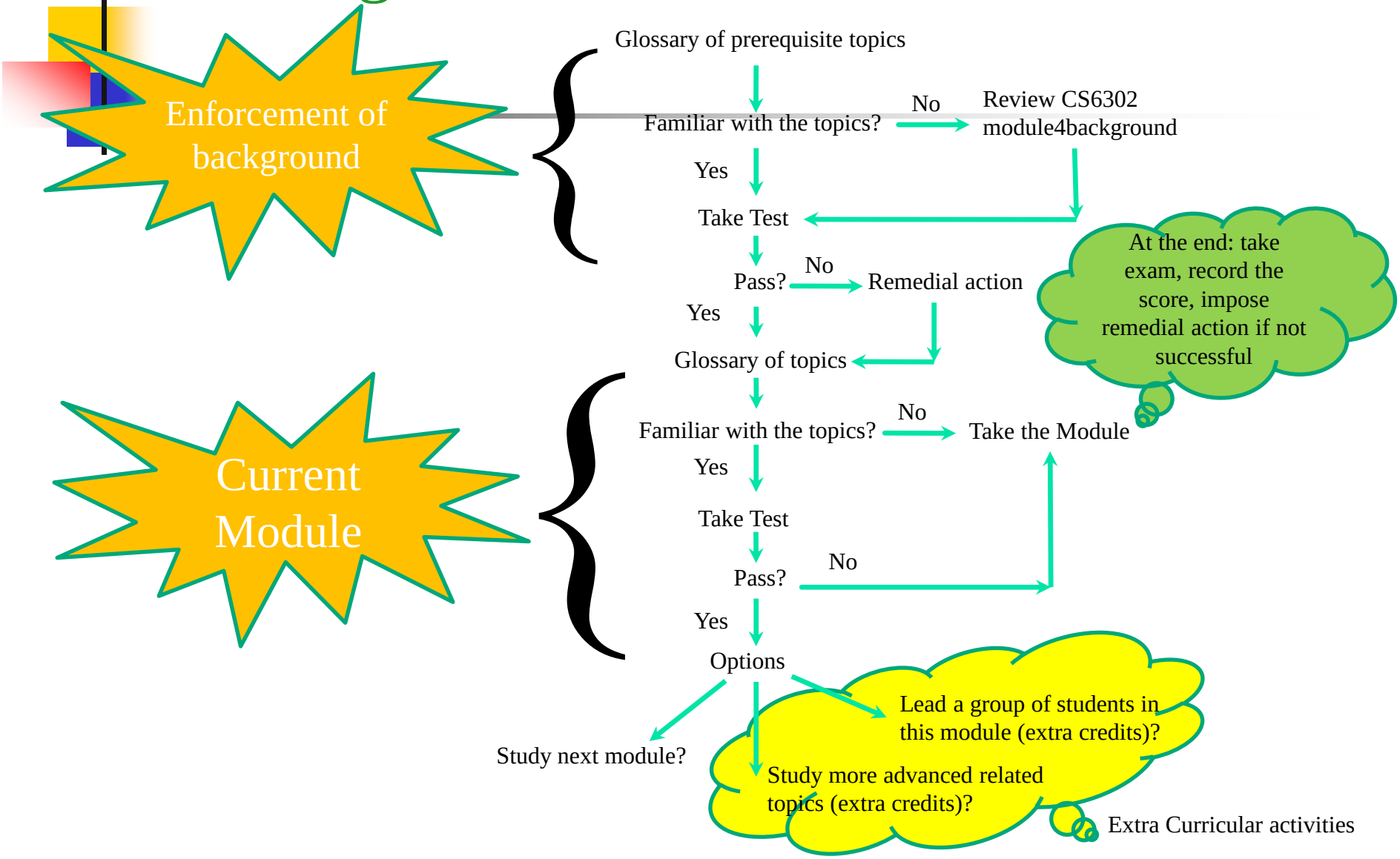
Heterogeneous Distributed Databases

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

1) Take the early test and start CS6302.module5
2) Study the supplement module (supplement CS6302.module4)
3) Act as a helper to help other students in studying CS6302.module4

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

Heterogeneous Distributed Databases



Heterogeneous Distributed Databases

- You are expected to be familiar with:
  - Homogeneous Distributed Databases,
  - Query processing and Transaction processing in homogeneous distributed databases
- If not, you need to study CS6302.module2 and CS6302.module3

Heterogeneous Distributed Databases

- Discussion in the last two modules was concentrated on traditional distributed databases, namely "homogeneous distributed databases". We also focused on two aspects of homogeneous distributed databases: query processing and transaction processing.
- In general, design of homogeneous distributed databases is a top-down process, i.e., during the design phase, global database administrator is well aware of data type, data model, applications, operations, … that the system is designed for.
- Consequently, within the scope of homogeneous distributed databases the main challenge lies in the data distribution: i.e., "how the data should be distributed" to meet various performance constraints.

Heterogeneous Distributed Databases

- The next two modules will look at the so called "heterogeneous distributed databases". In contrast to "homogeneous distributed databases", "heterogeneous distributed databases" design is a bottom up approach: i.e., during the design phase, the global database administrator, in general, neither has any clear notion about the structure and type of data nor has knowledge about the type of application or operations performed on the data.
- Reading between the lines, "heterogeneous distributed databases" is more dynamic than "homogeneous distributed databases" and this dynamic nature needs to be reflected in the design and operational procedures of the system.
- To look further, while data distribution is the main challenge in the design of "homogeneous distributed databases", data integration is the main challenge in the design of "heterogeneous distributed databases" .

Heterogeneous Distributed Databases

- First, we will express the notions of heterogeneity and autonomy and will link it to the definition of "heterogeneous distributed databases", then we will talk about different classes of information processing that fall under the general concept of "heterogeneous distributed databases". Different issues of concern within the scope of "heterogeneous distributed databases" will be enumerated and discussed. Data integration and different methods to implement "heterogeneous distributed databases" will be addressed in this module.

Heterogeneous Distributed Databases

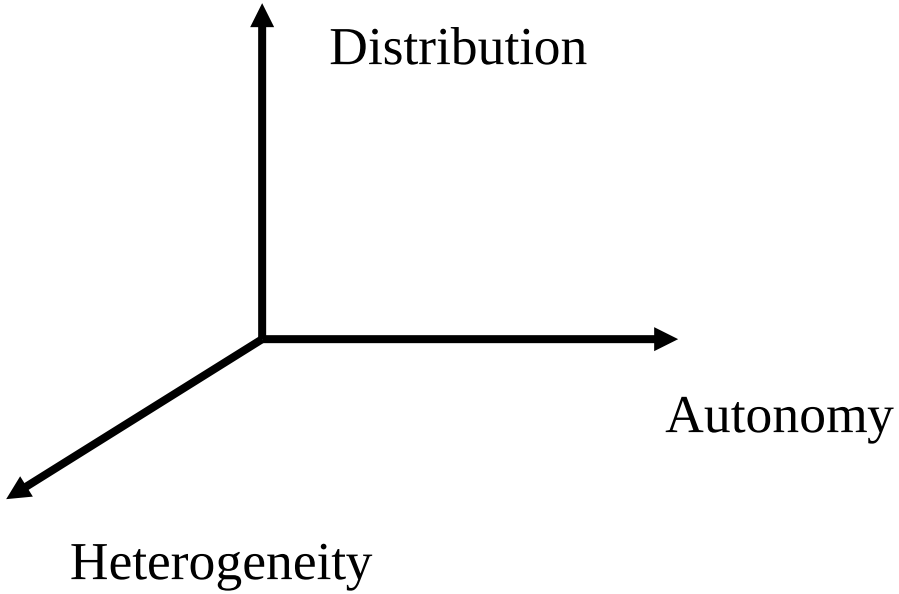
- Contents:
  - Definition of heterogeneous distributed databases
  - Different classes of multidatabases: a taxonomy
  - Database integration
  - Issues in Multidatabases
  - Design choices
  - Summary schemas model
  - Performance analysis of summary schemas model

Heterogeneous Distributed Databases

- MultiDatabase Systems
  - In general systems that allow information sharing among several component data sources can be classified along three dimensions:
    - Autonomy
    - Heterogeneity
    - Distribution

Heterogeneous Distributed Databases

- MultiDatabase Systems



Heterogeneous Distributed Databases

- MultiDatabase Systems – Autonomy
  - Autonomy refers to the distribution of control not data. It indicates the degree to which individual database can operate independently.
  - Autonomy is a function of several factors:
    - Whether the component systems exchange information (communication autonomy),
    - Whether the component systems can independently execute transactions (execution autonomy),
    - Whether a component system is allowed to modify information (design and execution autonomy).

Heterogeneous Distributed Databases

- MultiDatabase Systems – Autonomy
  - Distributed and heterogeneous databases are under separate and independent control.
  - Autonomy comes in different forms and shapes:
    - Design autonomy,
    - Communication autonomy,
    - Execution autonomy,
    - Association autonomy.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Design autonomy
  - Local databases choose their own data model, query language, semantic interpretation of data constraints, functions/operational support, etc.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Communication autonomy
  - Local databases decide when and how to respond to the requests from other databases.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Execution autonomy
  - The execution order of local and global transactions is controlled locally and local databases do not need to inform other database management systems of the execution order of transactions.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Association autonomy
  - Local databases decide how much of their functions/operations and data to be shared globally.
  - Local databases have the right to associate and disassociate themselves from the network of databases.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Local autonomy
  - The literature also has referred to the term of local autonomy that comes in two forms:
    - Operation autonomy requirements – ability of the local database to exercise control over its database.
    - Service autonomy requirements – ability of the local database to exercise control over services it provides to other local databases.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Local autonomy
  - From the global database management system, local autonomy takes several forms:
    - Heterogeneity of local databases (Design autonomy),
    - Refusing to provide services necessary for the global system to do the job or efficiently to do the job (Execution autonomy and Communication autonomy).

Heterogeneous Distributed Databases

- MultiDatabase Systems – Local autonomy
  - From the local database system, local autonomy can be violated in different ways:
    - Being modified,
    - Being forced to do things that it was not originally designed to do,
    - Being prevented from doing things it was originally able to do.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Autonomy
  - Along this dimension one can also talk about different strategies:
    - Tight integration (Traditional (homogeneous) distributed databases),
    - Semi-autonomous, and
    - Total isolation (Multidatabases).

Heterogeneous Distributed Databases

- MultiDatabase Systems – Tight integration
  - A single image of the entire databases is available to any user who wants to share the information which may reside in multiple databases.
  - In this environment, one of the data managers is in control of processing of each user request even if that request is serviced by more than one data manager.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Semi-autonomous
  - The system consists of database management systems that can operate independently but have decided to participate in a federation to make their local data sharable.
  - Each component database can determine what part of their database they will make accessible to the users of other database management systems.
  - The local databases are not fully autonomous systems since they need to be modified to be able to exchange information with one another.

Heterogeneous Distributed Databases

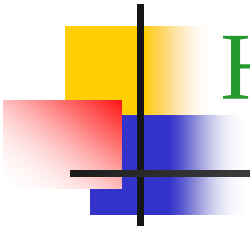
- MultiDatabase Systems – Total isolation
  - In this configuration, individual systems are stand alone which know neither the existence of other systems, nor how to communicate with them.
  - In this configuration, the processing of the user request that spans over several databases is very difficult, since there is no global control over the execution of individual database systems.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Distribution
  - Autonomy refers to the distribution of control, the distribution dimension deals with data and services — physical distribution of data over multiple sites. Along this dimension one can recognize two subclasses:
    - Client/server platform
    - Peer-to-peer platform

Heterogeneous Distributed Databases

- MultiDatabase Systems – Distribution
  - Client/server platform: In this environment data management duties are concentrated at the servers and clients facilitate requirements for the application environment needs such as user interface. In this platform we can talk about:
    - Thin client
    - Fat client
  - Peer-to-peer platform: In this environment, also called fully distributed, each site has full database management functionality and can communicate with other sites in order to execute queries and transactions.



Heterogeneous Distributed Databases

- MultiDatabase Systems – Heterogeneity
  - Along this dimension we can talk about various forms of heterogeneity such as hardware heterogeneity.

Heterogeneous Distributed Databases

- At this point, I am expecting you to fully appreciate the issue of autonomy and its potential effects on database functions such as query processing and transaction processing (refer to our discussion in modules 2 and 3 about query processing and transaction processing in traditional distributed databases) .

Heterogeneous Distributed Databases

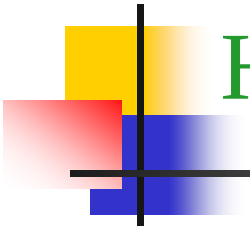
- As noted before, the distributed databases can be classified into two groups:
  - Homogeneous distribute databases,
  - Heterogeneous distributed databases.

Heterogeneous Distributed Databases

- Homogeneous distributed databases are logically integrated data sources to provide a single image of the database though they are physically distributed.
- Heterogeneous distributed databases are intended to provide interoperability among a set of preexisting (and by default, heterogeneous) database management systems.

Heterogeneous Distributed Databases

- MultiDatabase Systems
  - Existing databases are managed by different database management systems running on heterogeneous platforms.
  - The challenge is to give the user the illusion that they are accessing a single database (distribution transparency, heterogeneity transparency) that contains desired information while preserving the integrity and investment of preexisting environment.

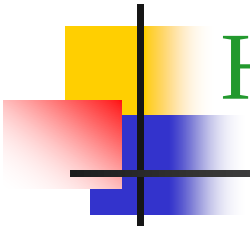


Heterogeneous Distributed Databases

- MultiDatabase Systems
  - In a multidatabase system it is necessary to:
    - Provide uniform access to the data and resources,
    - Allow databases to cooperate by exchanging data and synchronizing their execution seamlessly.

Heterogeneous Distributed Databases

- MultiDatabase Systems
  - In general, any solution to establish interconnection and cooperation among preexisting autonomous and heterogeneous databases has to address two fundamental issues:
    - Autonomy
    - Heterogeneity



Heterogeneous Distributed Databases

- MultiDatabase Systems
  - Various classes of solutions to multidatabase approach (as will be discussed later) are an attempt to compromise between autonomy and heterogeneity.
  - As an example, federated database approach favored heterogeneity over autonomy and multidatabase language approach favored autonomy over heterogeneity

Heterogeneous Distributed Databases

- So far, we referred to several terms such as:
  - Homogeneous distributed databases,
  - Heterogeneous distributed databases,
  - Federated databases,
  - Multidatabases, and
  - Multidatabase language.
- Now we will define a taxonomy in order to distinguish these terms from each other.

Heterogeneous Distributed Databases

- MultiDatabase Systems – A Taxonomy
  - There is a wide range of solutions for global information sharing in a distributed system. Factors such as:
    - User Requirements,
    - Existing Hardware and Software, and
    - The Amount of Investment Available

    Will determine which solution is appropriate in an application domain.

Heterogeneous Distributed Databases

- MultiDatabase Systems – A Taxonomy
  - Terms such as:
    - Distributed Databases,
    - Multidatabases,
    - Federated Databases, and
    - Interoperable Systems

  Have been tossed up in the literature.

Heterogeneous Distributed Databases

- MultiDatabase Systems – A Taxonomy
  - In spite of their differences, these solutions have two common characteristics:
    - They are intended to address a distributed system with two distinguishing components:
      - A global component with access to all globally shared information and
      - Multiple local components that only manage information at that site.
    - They are designed to provide timely and reliable access to the information sources.

Heterogeneous Distributed Databases

- MultiDatabase Systems – A Taxonomy
  - Our taxonomy distinguishes these solutions according to how tightly/loosely the local components are integrated with the global component.
    - In a tightly coupled system, the global functions have access to low level internal functions of the local database systems.
    - In a loosely coupled system, the global functions access local functions through the database management system external user interface.

Heterogeneous Distributed Databases

- MultiDatabase Systems – A Taxonomy

Tightly Coupled ← – – – – – – – – – – – – – – – – – – – → **Loosely Coupled**

Distributed Database	Global-Schema Multidatabase	Federated Database	Multidatabase Language	Homogeneous Multidatabase Language	Interoperable System

Heterogeneous Distributed Databases

- MultiDatabase Systems – A Taxonomy
  - Distributed Databases
    - Global and Local functions share very low level internal interfaces and are so tightly integrated that there is little distinction between them.
    - They are typically designed in a top-down fashion.
    - Local database management systems are typically homogeneous, even though they may be implemented on different hardware systems and/or software platforms.

Heterogeneous Distributed Databases

- MultiDatabase Systems – A Taxonomy
  - Distributed Databases
    - The global system has control over local data and processing.
    - The system typically maintains a global schema by integrating the schemas of all the local databases.
    - It offers the best performance of all the information sharing solutions, at the cost of significant local modification and loss of control.

Heterogeneous Distributed Databases

- MultiDatabase Systems – A Taxonomy
  - Multidatabases
    - Multidatabases are more loosely coupled than distributed databases because global functions access local information through the external user interface of the local database management system.
    - Multidatabases typically integrate the data from pre-existing, heterogeneous local databases.
    - Multidatabases present global users with transparent methods to use the total information in the system.

Heterogeneous Distributed Databases

- MultiDatabase Systems – A Taxonomy
  - Federated Databases
    - Federated databases do not maintain a single common global schema.
    - Each local system maintains its own partial global schema which contains only the global information description that will be used at that node.
    - Each node cooperates closely with the specific nodes it accesses.

Heterogeneous Distributed Databases

- MultiDatabase Systems – A Taxonomy
  - Federated Databases
    - As noted before, federated database is a compromise between no integration and total integration. It allows more local autonomy relative to total integration approach (association autonomy).
    - The association autonomy is made possible through several layers of abstractions and schema generations, as follows:

Heterogeneous Distributed Databases

- MultiDatabase Systems – A Taxonomy
  - Federated Databases
    - Local schema: The conceptual schema of a component database expressed in the data model of component database management system.
    - Component Schema: A local schema is translated to the common data model of the federated database system to alleviate data model heterogeneity. Each local database should store one-to-one mappings between the local data model and conceptual data model objects during the schema translation.

Heterogeneous Distributed Databases

- MultiDatabase Systems – A Taxonomy
  - Federated Databases
    - Export Schema: Each database can specify the sharable objects to other members of the federated database system – Association autonomy is maintained here.
    - Federated schema: This is the global view schema of exported schemas of component databases forming a federation.

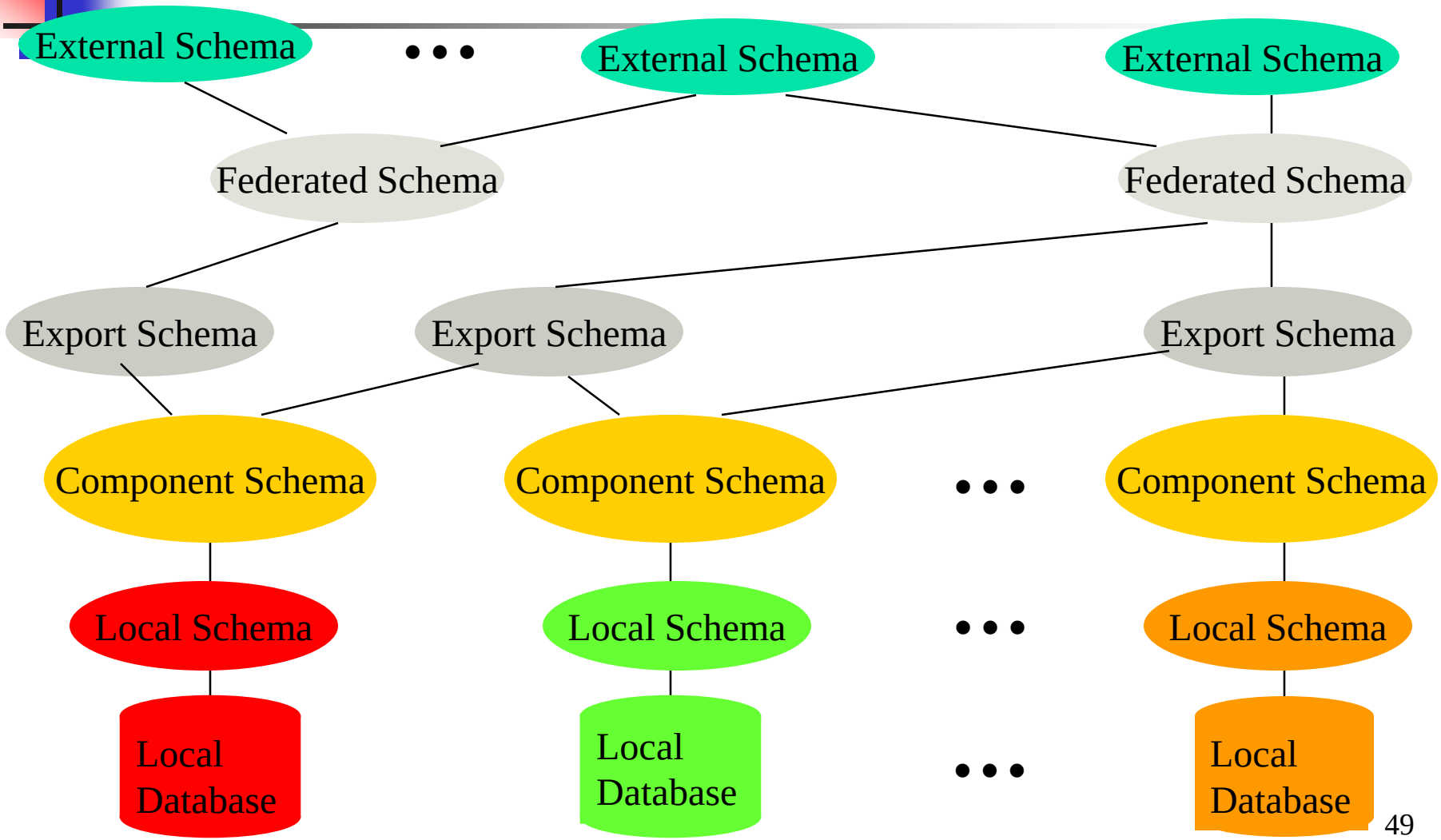
Heterogeneous Distributed Databases

- MultiDatabase Systems – A Taxonomy
  - Federated Databases
    - Federated schema can be statically integrated schemas or a dynamic user view of multiple schemas.
      - The integrated schema is managed and controlled by the federated database system administrator, if the federated database is tightly coupled.
      - The view is managed and controlled by the user if the federated database is loosely coupled. There can be multiple federated schemas, one for each class of federation users.

Heterogeneous Distributed Databases

- MultiDatabase Systems – A Taxonomy
  - Federated Databases
    - External Schema: It is mainly for customization when the federated database is large and complicated.

Heterogeneous Distributed Databases



Heterogeneous Distributed Databases

- MultiDatabase Systems – A Taxonomy
  - Federated Databases
    - Federated database systems can be further classified as:
      - Loosely coupled, and
      - Tightly coupled

Heterogeneous Distributed Databases

- MultiDatabase Systems – A Taxonomy
  - Loosely coupled
    - It is the user's responsibility to maintain and create the federation schema.
    - No control is enforced by the federation system or federation administrator.
    - Each user is expected to be knowledgeable about the information and the structure of the relevant export schemas.
    - Federated schema is dynamic and can be created or dropped on the fly.
    - This class of systems are intended for highly autonomous read-only databases and cannot support view updates

Heterogeneous Distributed Databases

- MultiDatabase Systems – A Taxonomy
  - Tightly coupled
    - Federation administrators have full control over the creation and maintenance of federated schemas and access to the export schemas.
    - As a result, this class of systems provides location, replication, distribution, and heterogeneity transparency.
    - It supports one or more federated schemas.

Heterogeneous Distributed Databases

- MultiDatabase Systems – A Taxonomy
  - Interoperable Systems
    - Global function is limited to single message passing and does not support full database functions.
    - Global system is not database-oriented, hence local systems may include other types of information repositories such as: Expert systems, Knowledge-based systems, ...

Heterogeneous Distributed Databases

- MultiDatabase Systems – A Taxonomy

	Tightly <----------> Loosely coupled					
	Distributed Database	Global Schema Multidatabase	Federated Database	Multidatabase Language System	Homogeneous Multidatabase Language System	Interoperable System
Global system has access to local ...	Internal DBMS function	DBMS user interface	DBMS user interface	DBMS user interface	DBMS user interface, plus some internal functions	Application on top of DBMS
Local nodes typically are ...	Homogeneous databases	Heterogeneous databases	Heterogeneous databases	Heterogeneous databases	Heterogeneous databases	Any data source that meets the interface protocol
Full global database function	Yes	Yes	Yes	Yes	Yes	No

Heterogeneous Distributed Databases

- MultiDatabase Systems – Some terminologies
  - Effective sharing, and use of data and functions can be achieved in the forms of integration, interoperability, interdependency, and exchange.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Some terminologies
  - Interoperability is the ability to request and receive services among the interoperating systems in order to use each others' functionalities.
  - In another words, information systems are interoperable if:
    - They can exchange messages and requests,
    - They can receive services and operate as a unit in solving a common problem.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Some terminologies
  - Interdependency means functions and data in different systems are related or dependent on each other.
  - Integration implies uniform and transparent access to data managed by multiple databases.

Heterogeneous Distributed Databases

- Within the scope of multidatabases, data integration has two general aspects:
  - Schema integration: to provide a uniform global view of the shared data and a means to locate and access distributed and heterogeneous data sources.
  - Result integration: to fuse partial and heterogeneous results and present it in a uniform format to the user.
  - In the following discussion, we concentrate on schema integration since it is a more challenging issue.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Database integration
  - Database integration involves the process by which information from participating databases can be conceptually integrated (fused) to form a single comprehensive definition of a multidatabase – designing the global conceptual schema (schema integration).

Heterogeneous Distributed Databases

- MultiDatabase Systems – Database integration
  - Database integration is based on complete integration of multiple databases to provide a single and uniform view (global schema) of the shared data. This process should provide:
    - Distribution transparency,
    - Uniform view of data,
    - Consistent and uniform access to data, and
    - Heterogeneity transparency.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Database integration
  - In short, schema integration refers to the methodology that facilitates integration of schemas in order to:
    - Hide any heterogeneity and
    - Represent the semantics of the database.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Database integration
  - The integrated schema is used to formulate the global queries and global transactions that may possibly span multiple databases.
  - It should be noted that the schema integration differs from view integration in several ways:

Heterogeneous Distributed Databases

- MultiDatabase Systems – Database integration
  - View integration is used in a top-down database design. It is the process of generating a single integrated schema from multiple user views.
  - Schema integration is a bottom-up approach, since it attempts to integrate existing databases.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Database integration
  - In view integration, users define views using a single data model – Homogeneity.
  - In schema integration, schemas may be represented using multiple data models.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Database integration
  - At the time view integration is defined, user views do not reflect existing data in a database – views represent abstract objects.
  - In schema integration, we integrate schemas that represent existing databases – the schema integration process cannot violate the semantics of the existing databases.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Database integration
  - Schema integration is not a one time process and changes in databases may force changes at integrated schema.
  - Integrated schema should be dynamic because of:
    - Changes in the database structure (may result changes in the local schema).
    - Changes in constraints on the underlying databases.
    - Changes to database values due to addition, deletion, and modification operations, and
    - Addition/departure of data sources to/from global information sharing environment.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Database integration
  - A three layered schema framework can be used to facilitate the integration of heterogeneous systems:
    - The first layer is represented by the native schemas of the local databases,
    - The second layer is represented by imported schemas that are direct translation of the native schemas to a common and global data model.
    - The third layer is the conceptual unified schema that defines a consistent and unified set of rules and constructs that facilitate the global information sharing process.

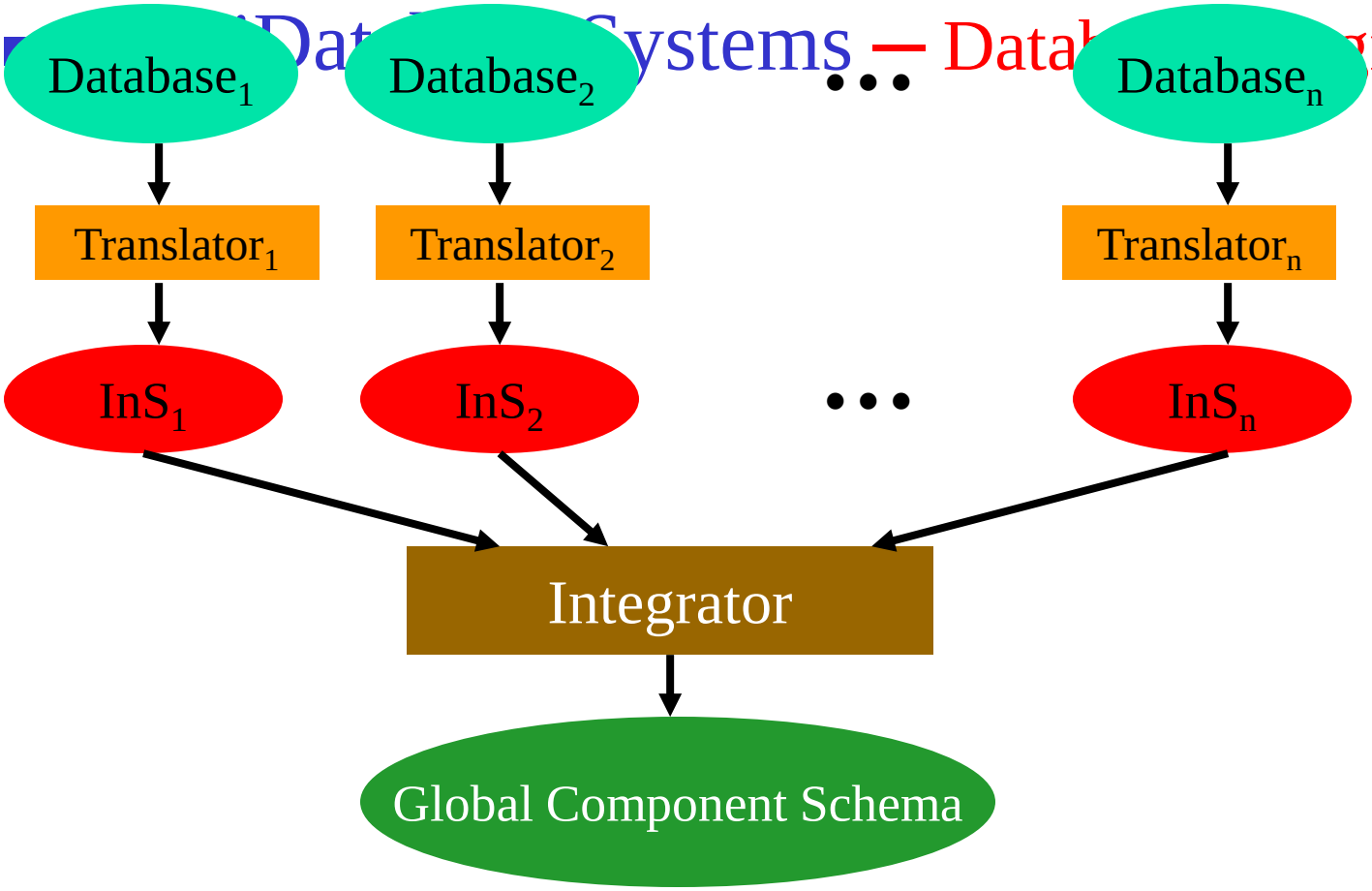
Heterogeneous Distributed Databases

- MultiDatabase Systems – Database integration
  - Database integration requires two steps:
    - Schema translation and
    - Schema integration

Heterogeneous Distributed Databases

- MultiDatabase Systems – Database integration
  - Schema translation: The component databases schemas are translated to a common intermediate canonical representation. This step is necessary if we are dealing with heterogeneous local components.
  - Schema integration: Intermediate schemas are integrated into a global conceptual schema.

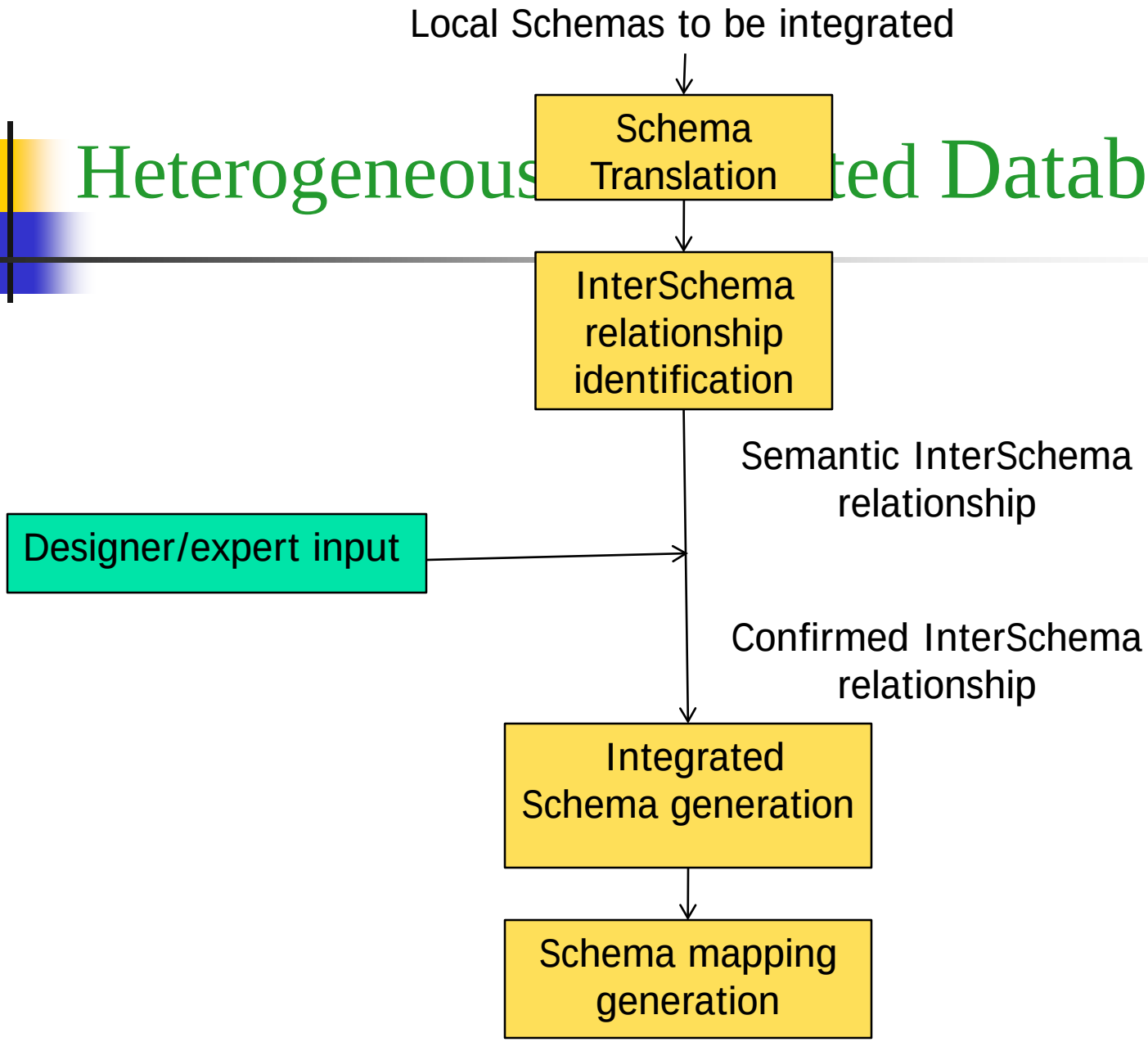
Heterogeneous Distributed Databases



Heterogeneous Distributed Databases

- MultiDatabase Systems – Database integration
  - In more detail, schema integration is a four-phase process:

Heterogeneous ted Databases



Heterogeneous Distributed Databases

- MultiDatabase Systems – Database integration
  - Schema translation: Local schemas are translated into schema using a common model.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Database integration
  - Schematic interschema relationship generation: To identify objects in the local components that may be related and to categorize the relationships among them.
  - In this process, we need to examine the semantics of the objects in different databases and identify the relationship among objects based on their semantics. The final goal of this step is to generate a reliable and accurate set of relationships among database objects, i.e., semantic differences and semantic similarities must be resolved.

Heterogeneous Distributed Databases

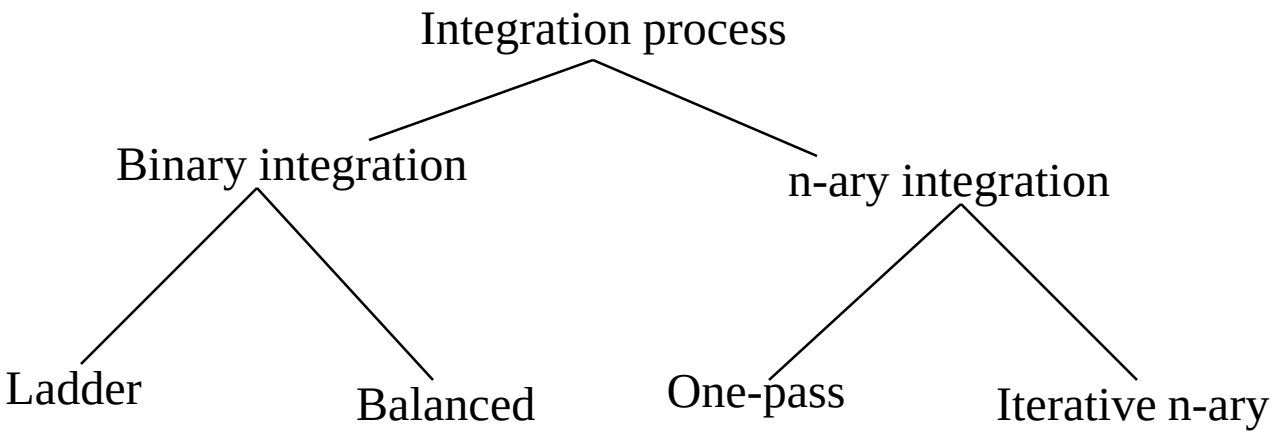
- MultiDatabase Systems – Database integration
  - Integrated schema generation: Various forms of existing heterogeneity that might exist must be resolved.
  - Schema mapping generation: This phase maps entities in the integrated schema to the objects in local schemas.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Database integration
  - Integration methodologies have been classified as, binary and n-ary mechanisms.
    - Binary integration involves the manipulation of two schemas at a time. Binary integration methods can be further subgroups into:
      - Ladder pattern
      - Balanced pattern
    - n-ary integration involves more than two schemes at a time. n-ary integration methods can be further classified as:
      - One pass integration
      - Iterative n-ary integration

Heterogeneous Distributed Databases

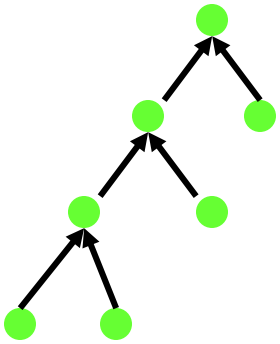
- MultiDatabase Systems – Database integration



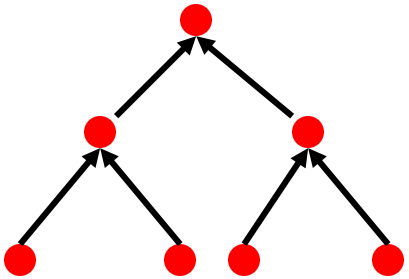
Heterogeneous Distributed Databases

- MultiDatabase Systems – Database integration

Ladder integration method



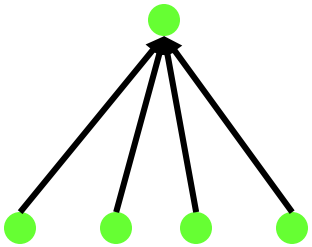
Balanced integration method



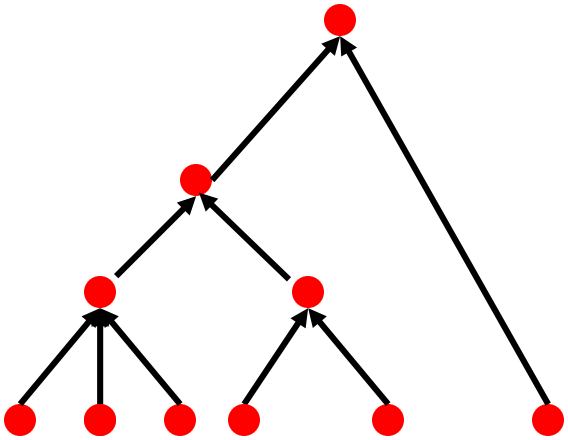
Heterogeneous Distributed Databases

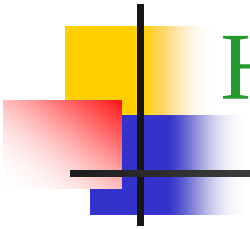
- MultiDatabase Systems – Database integration

One-step integration method



Iterative n-ary method





Heterogeneous Distributed Databases

- Questions
  - Compare and contrast loosely coupled and tightly coupled federated database systems against each other.
  - Compare and contrast different classes of integration methodologies against each other.

Heterogeneous Distributed Databases

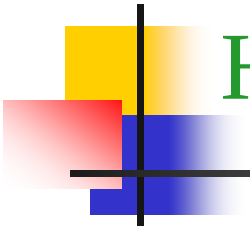
- MultiDatabase Systems – Database integration
  - In general, schema integration:
    - Is hard to automate,
    - Violates local autonomy,
    - Is computation intensive,
    - Has the potential of missing some of the embedded semantic knowledge,
    - Is not dynamic — Hard to expand/shrink multidatabases.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Database integration
  - As an example, federated databases are a compromise between no integration and total integration. It is aimed to ease static nature of global schema integration approach while allowing more local autonomy — Association autonomy.

Heterogeneous Distributed Databases

- MultiDatabase Systems
  - Corporate mainframes are sources of global administrative functions, departmental systems run specific technical functions, and workstations and personal computers provide individual services and immediate access to remote locations.
  - Databases in each of these environments have developed independently to meet specific requirements and incompatible database management systems have evolved to meet the varying needs in these independent environments.



Heterogeneous Distributed Databases

- MultiDatabase Systems
  - Separate, autonomous data sources — islands of information — are no longer able to meet increasingly sophisticated user needs in today's networked world.
  - Related information important to a global application may exist in multiple incompatible databases.
  - Users cannot be expected to manage system details of sending multiple requests in different languages, possibly different data models, to multiple information sources.

Heterogeneous Distributed Databases

- MultiDatabase Systems
  - A multidatabase system is a particular type of distributed system that allows global user to easily access information from multiple local databases.
  - key features that distinguish a multidatabase system from a distributed database system are:
    - Local autonomy, and
    - Heterogeneity

Heterogeneous Distributed Databases

- MultiDatabase Systems
  - By preserving the autonomy of local databases, existing organizational investment in local applications and user training is preserved while significant new functions of global data access is provided.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Issues
  - Multidatabases inherit many of the problems associated with distributed systems in general and distributed databases in particular.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Issues
  - Site Autonomy
  - Differences in Data Representation
    - Name Differences
    - Structural Differences
    - Abstraction Differences
    - Missing or Conflicting Data
  - Heterogeneous Local Databases
  - Global Constraints
  - Global Query Processing
  - Concurrency Control
  - Security
  - Local Node Requirements

Heterogeneous Distributed Databases

- MultiDatabase Systems – Site Autonomy
  - Each local database management system retains complete control over local data and processing. Each site independently determines what information it will share with the global system, what global requests it will service, when it will join the multidatabase, and when it will stop participating in the multidatabase.
  - Therefore, the database management system itself is not modified and global changes — i.e., addition and deletion of other sites, global optimization — do not effect the local database management system.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Site Autonomy
  - Site autonomy may be desirable for a number of reasons:
    - Some local databases may have critical roles in an organization, and economically it may be impossible to change them.
    - Capital invested in existing hardware, software, and user training is preserved.
    - A site can protect information by not including it in the local schema that is shared with the global schema.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Site Autonomy
  - Factors such as economical issues and security measures motivate site autonomy. However, despite these desirable aspects, site autonomy and lack of control over local resources introduce global performance degradation relative to a tightly coupled distributed database system.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Site Autonomy
  - Local autonomy can be handled in two ways:
    - Imposing limitations and restrictions on global function implementation and/or capability,
    - Compromising local autonomy to some extent and handling any local autonomy violations as they occur.
  - To read between the lines, it is impossible to support global applications without compromising local autonomy in one way or another.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Differences in Data Representation
  - Since local databases are developed independently with differing local requirements, it is possible for the same real-world objects to be modeled and represented differently.
  - The same object in different local databases should be mapped to a single global representation and naturally, semantically different objects should be mapped to different global representations.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Differences in Data Representation
  - Name Differences: Local databases may have different conventions for naming objects. This leads to the problems of:
    - Synonyms: The same data item has different names in different databases, and
    - Homonyms: Different data items have the same name in different databases.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Differences in Data Representation
  - Format Differences: These include differences in:
    - Data type — part number is defined as an integer in one database and as an alphanumeric string in another,
    - Domain — temperatures in one database may be rounded off, while another keeps exact readings,
    - Scale — in one database the area is measured in square feet and acres in another,
    - Precision — one database may use single-precision floating-point numbers for a given quantity while another uses double-precision, and
    - Item combinations — dates can be kept as a single string, such as 012290, or as separate quantities for month, day, and year

Heterogeneous Distributed Databases

- MultiDatabase Systems – Differences in Data Representation
  - Structural Differences:
    - Objects may be structured differently in separate local databases — different data models,
    - A data item may have a single item in one database and multiple values in another,
    - The same item may be a data value in one place, an attribute in another place, and a relation in a third place,
    - The relationships between objects may differ from database to database,
    - Dependencies between objects may differ in databases.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Differences in Data Representation
  - Recognizing semantically equivalent objects despite their structural differences can be a difficult task and is almost always a manual process for the global Data Base Administrator or user.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Differences in Data Representation
  - Abstraction Differences: Different local users may be interested in different levels of detail about the same object.
  - Different levels of abstraction can be integrated through the use of generalization hierarchies — an object at one level of the hierarchy represents the collection of the common attributes of its immediate descendant at the next level of the hierarchy.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Differences in Data Representation
  - Missing or Conflicting Data:
    - Local databases may have embedded information that is not explicitly recorded. Embedded data is information that is assumed by local users, so it does not have to be spelled out with explicit data value — company's name.
    - Databases that model the same real-world object may have conflicts in the actual data values recorded. One site may not have some information recorded due to the incomplete updates, system error, or insufficient demand to maintain such data.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Local Heterogeneity
  - Multidatabases claim to support heterogeneous data models at the local level.
  - The support mainly consists of providing local translation capability from the local model to the common global model.
  - A problem with supporting local database management system heterogeneity is making the trade-off between writing translation code and limiting participation. Moreover, any local functional deficiencies must be compensated for with global system software.

Heterogeneous Distributed Databases

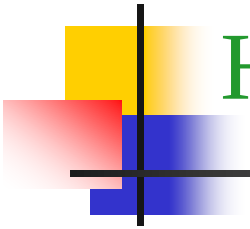
- MultiDatabase Systems – Global Constraints
  - Since different local databases may represent semantically equivalent data or semantically related data, the global system needs some method for specifying and enforcing integrity constraints on inter-database dependencies and relationships.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Global Constraints
  - The enforcement of global constraints should address issues such as:
    - Inter-database update dependency — updating an object in one database should cause an equivalent object in another to be updated, and
    - Aggregate privacy dependency — combining independent data from several sources may reveal more information than the simple sum of the individual data items.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Global Query Processing
  - Query decomposition and optimization in a distributed database environment was discussed in detail.
    - The query is decomposed into a set of sub-queries. The query optimizer creates an access strategy that specifies which local databases are to be involved, what each will do, how the intermediate results will be combined, and where the global processing will occur.
    - Global constraints must also be checked and enforced during the query execution.



Heterogeneous Distributed Databases

- MultiDatabase Systems – Global Query Processing
  - Multidatabase systems must also handle inter-database dependencies, manage global resources, and support additional language features.
  - All these demands on the query processor must be handled in an efficient manner, despite its dynamic, distributed nature and the lack of control over local database management systems.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Global Query Optimization
  - In distributed database, a global query optimizer deals with parameters such as:
    - Capabilities of individual nodes,
    - Communication link costs,
    - Data and processing requirements.
  - The optimizer applies a cost function that weighs these factors and produces an efficient strategy to handle the query.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Global Query Optimization
  - In a multidatabase system, global query optimization is becoming more complicated due to the lack of control over local database management systems and possible conflicts between global and local optimization.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Concurrency Control
  - The traditional concept of a transaction as short lived and atomic is unsuited to the multidatabase environment.
  - Multidatabase transactions will typically involve multiple, separate local databases and several layers of data/query translations.
  - While the global system has information and control over global transactions, it does not have information about local transactions. Furthermore, site autonomy prevents global control from enforcing local serializability orders.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Concurrency Control
  - In a multidatabase environment semantic knowledge about the transaction may be used to break it up into atomic sub-transactions.
  - The quest for global concurrency control may be a strong enough incentive to violate site autonomy.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Security
  - Providing security in any distributed system is a difficult task at best. Problems include non-secure communication links and varying levels of security provided at different nodes.
  - Site autonomy provides some measure of local security.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Local Node Requirements
  - Multidatabases require global data structures and global software modules to implement global functions. Although, site autonomy guarantees that local DBMSs will be unchanged by joining a multidatabase, the local machine will have to share some of the global storage and processing requirements.
  - Maintaining a full complement of global data and software modules on a personal computer or a heavily loaded mainframe may be difficult or impossible.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Design choices
  - There are two major approaches to designing a multidatabase system:
    - The global schema approach, and
    - The multidatabase language approach.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Design choices
  - Global Schema Approach
    - This approach, known as the Schema (view) integration, provides a conceptual layer that represents a single and integrated Schema (view) of the data available in the multidatabase system.
    - For the very same reasons discussed earlier, Global schema design is much more difficult than just taking a union of the input schemas of the local nodes.
    - It should take the independently developed local schemas, resolves semantic and syntactic heterogeneity between them, and creates an integrated summary of all the information from the union of the local schemas.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Design choices
  - Global Schema Approach
    - There are a number of common techniques for integrating multiple, distinct schemas:
      - Analyzing similarities and conflicts between objects and relationships in separate schemas must be done before they can be integrated.
      - Generalization hierarchies — the process of taking similar objects and creating a new, generic object that has all the common properties of the original objects,
    - Can be used to develop the global schema.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Design choices
  - Global Schema Approach
    - This approach is a direct outgrowth of distributed databases.
    - The global schema is just another layer, above the local external schemas, which provides additional data independence.
    - The global schema makes global access quite user friendly.
    - The global interface is independent of all the heterogeneity in local database management systems and data representations.
    - The global schema is usually replicated at each node.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Design choices
  - Global Schema Approach
    - The amount of global knowledge required about what is being integrated and how to integrate it is a major problem with the global schema approach.
    - Despite the methodologies, algorithms and heuristics that have been defined to help automate parts of the schema-integration process, this process is still very human-labor-intensive.
    - The sheer size of a global schema can make it a problem to be replicated at nodes with limited storage capabilities.
    - Global schema must also be maintained in the face of arbitrary changes to local schemas.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Design choices
  - Global Schema Approach
    - The literature is largely silent on how to maintain the global schema. Changes to local schemas, addition and deletion of a local node means massive computation and synchronization that should be reflected in all replicated copies.
    - Local changes may force the global DBA to reconsider many design decisions made during the initial integration process — with wide-reaching consequences.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Design choices
  - Multidatabase Language Approach
    - It provides local data integration dynamically through global query language features.
    - It is an attempt to resolve some of the problems associated with global schemas, such as up-front knowledge, up-front development cost, large maintenance requirements, and processing/storage requirements placed on local nodes.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Design choices
  - Multidatabase Language Approach
    - A basic requirement for a multidatabase language is to define a common name space across all participating schemas. The most straightforward way to accomplish this is to allow data-item names to be qualified with the associated database name and node identifier.
    - A common name space can still provide some measure of location independence in the face of data-item movement if object names are independent of the node they currently reside at.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Design choices
  - Multidatabase Language Approach
    - A multidatabase language system puts most of the integration responsibility on the user, but alleviates the problem by giving the user many functions to ease this task.
    - Multidatabase language system users must have a means to display what information is available from various sources.
    - The user is assumed to have well-defined ideas about what information is required and where it probably resides.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Design choices
  - Multidatabase Language Approach
    - Multidatabase language approach shifts the burden of integration from global database administrators to users and local database administrators. User queries may have to contain some programming to achieve the desired results. However, the results and processing methods can be individually tailored.
    - Multidatabase language systems trade a level of data independence for a more dynamic system and greater control over system information.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Design choices
  - Multidatabase designers have created methods to integrate semantically similar, but syntactically different data entities.
  - These methods all assume that database designers or users can identify semantically similar entities despite the representation and naming differences. Without intimate knowledge of the structure of all local databases, this assumption in invalid.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Design choices
  - The Summary Schemas Model has been developed as an extension to multidatabase systems to provide linguistic support to automatically identify semantically similar entities with different access terms.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Design choices
  - A Summary Schema is a concise, abstract description of the semantic contents of a group of input schemas.
  - Summary Schemas Model uses specific linguistic relationships between schema terms to build a hierarchical global meta data which describes the information available in all local databases in an increasingly abstract form.

Heterogeneous Distributed Databases

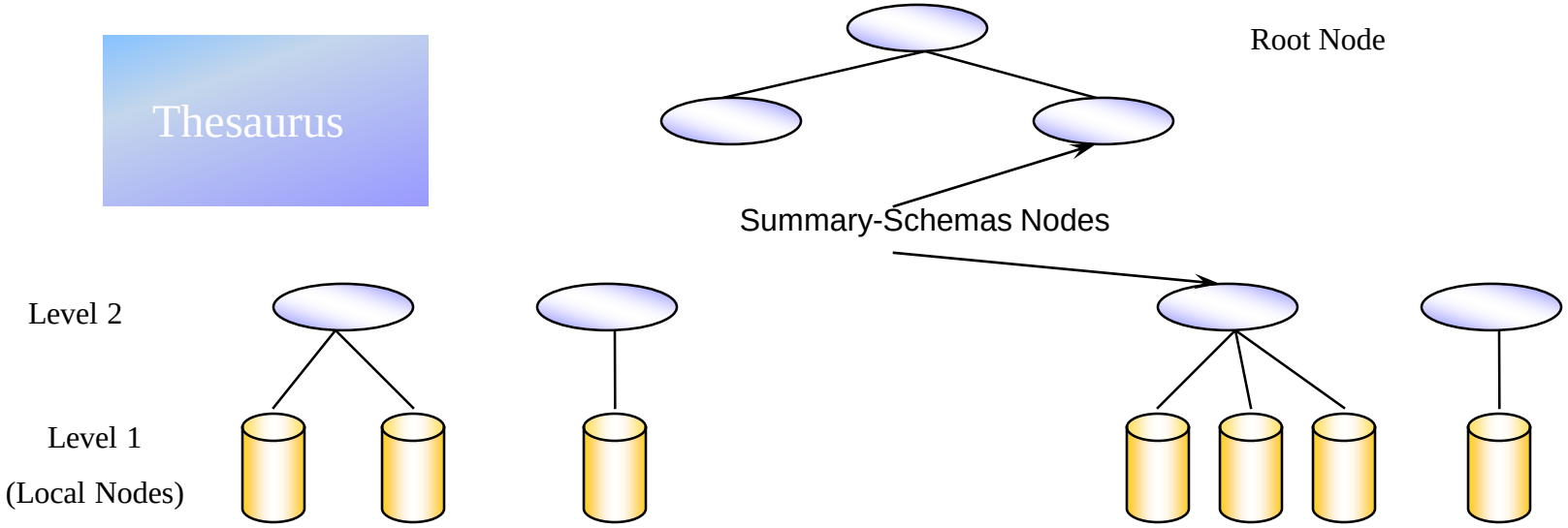
- MultiDatabase Systems – Design choices
  - Summary Schemas Model provides intelligent, user friendly access to multidatabase systems.
  - Summary Schemas Model global integration process is largely automatic.
  - Summary Schemas Model meta data is relatively much smaller and easier to create, maintain, and store — the hierarchy is short and bushy.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Design choices
  - Summary Schemas Model allows multidatabase users to submit global database queries that describe requested data in terms that are meaningful to the user — imprecise query.
  - Summary Schemas Model uses the global meta data and an on-line linguistic tools to interpret the user's imprecise query and associate it with the precise local system access terms that are semantically closest to the user's terms.
  - Summary Schemas Model allows the users to iteratively identify multiple sources of data that match an imprecise query — query Refinement Facility.

Heterogeneous Distributed Databases

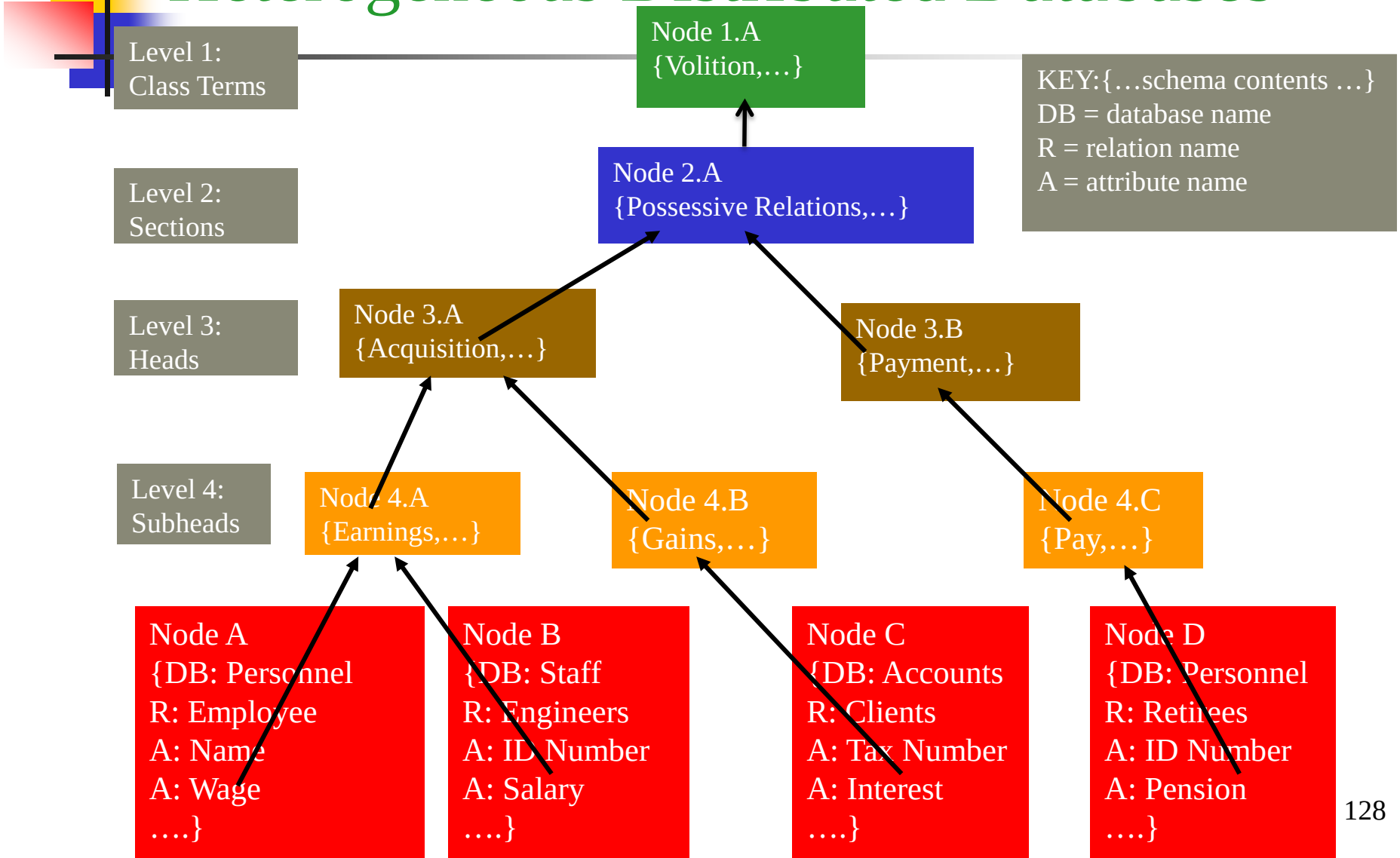
- MultiDatabase Systems – Design choices



Heterogeneous Distributed Databases

- MultiDatabase Systems – Design choices
  - In SSM hierarchical structure leaf nodes are the local databases and Internal nodes are responsible for the summary schemas structure.
  - Each leaf node contributes a database schema.
  - Each access term in a leaf schema is associated with an entry level term in the system taxonomy. Once these terms are linked to the taxonomy hierarchy, creating the summary schema at the internal nodes is automatic.
  - Summary schemas at the internal nodes are lists of hypernyms from the taxonomy, where each hypernym keeps a list of all child nodes with corresponding hyponyms.

Heterogeneous Distributed Databases



Heterogeneous Distributed Databases

- MultiDatabase Systems – Design choices
  - Summary Schemas Model – Advantages
    - The meta-data is by orders of magnitude smaller than the meta-data generated by the Global-schema approach.
    - Preserves local DBMS autonomy.
    - Provides good system scalability.
    - Reduces average search time.
    - Resolves imprecise queries.

Heterogeneous Distributed Databases

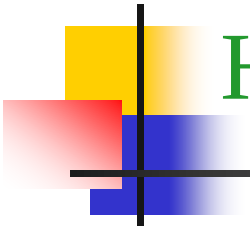
- MultiDatabase Systems – Summary Schemas Model
  - Global Data Structure Costs
    - Using Roget's Taxonomy, Summary Schemas Model must maintain a taxonomy structure of 7300 entries, their hypernym/hyponym links, and 30,000 cross reference links.
    - For linking local database schemas into the Summary Schema hierarchy, the taxonomy structure must include an additional 215,000 entry level terms and their associated links.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Summary Schemas Model
  - Precise Query Processing
    - Traditional multidatabase language systems queries precisely reference data to be accessed and how that data is to be manipulated.
    - A data reference includes location, a local access term, and how the local name and structure are to be mapped with similar data to a common name and structure for global processing.
    - The origin node parses the query, sends requests to remote sources, and combines the partial results.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Summary Schemas Model
  - Imprecise Query Processing
    - Summary Schemas Model performs the same step, but adds a reference resolution phase between parsing the query and sending the remote access requests.
    - If the user is unsure of the existence, location, or local access terms, she/he can initiate an imprecise query processing procedure.



Heterogeneous Distributed Databases

- MultiDatabase Systems – Summary Schemas Model
  - Imprecise Query Processing
    - Summary Schemas Model also allows searching for more distant semantic matches, by using the Semantic Distant Metric.
    - The Semantic Distant Metric is a weighted Count of the number of semantic links in the path between two terms in the taxonomy.

Heterogeneous Distributed Databases

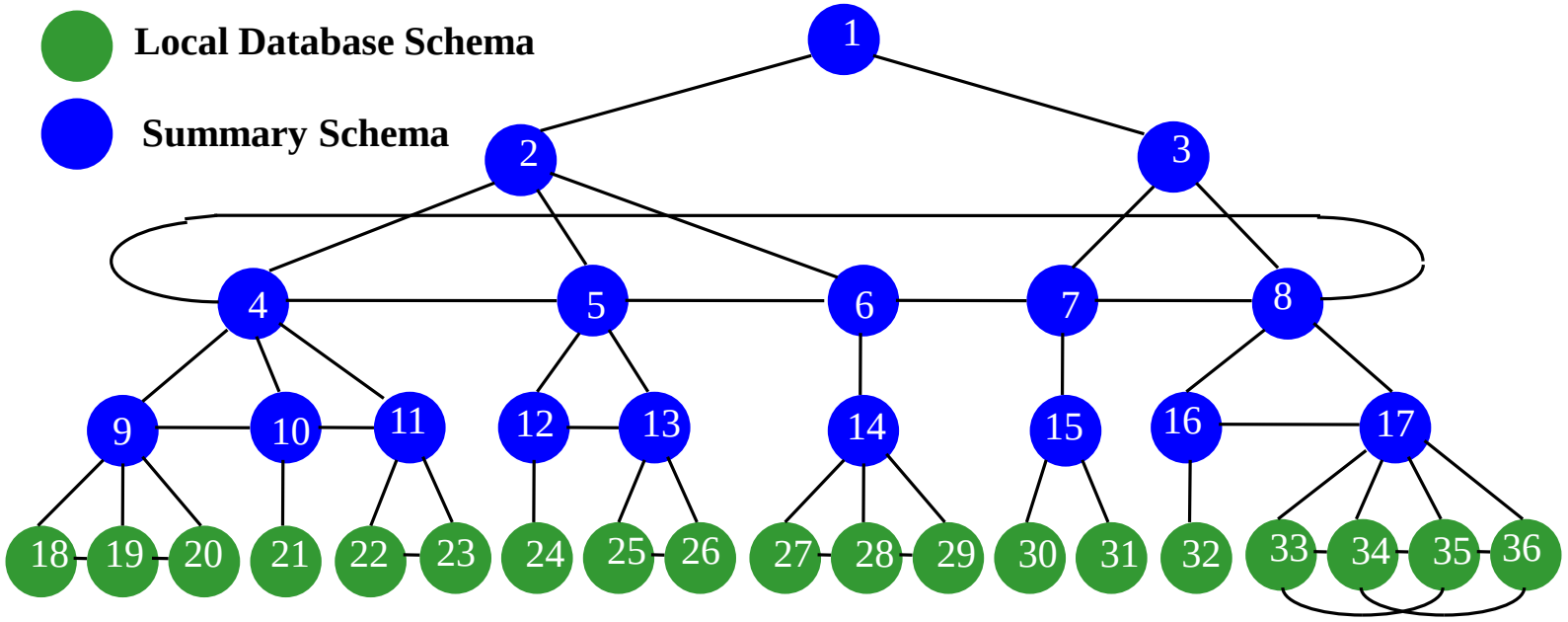
- MultiDatabase Systems – Summary Schemas Model
  - Global Data Structure Costs
    - On a sample of ≈ 1000 schema access terms, there was an 80% reduction in the number of terms when they were mapped to the corresponding hypernyms. The next level of mapping produced a 31% reduction and a final mapping to the highest level hypernyms produces a 79% reduction.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Summary Schemas Model
  - Global Data Structure Costs
    - In another experience, the space requirements across the whole system for the Summary Schemas Model was 4% of the total space required by the global schemas.

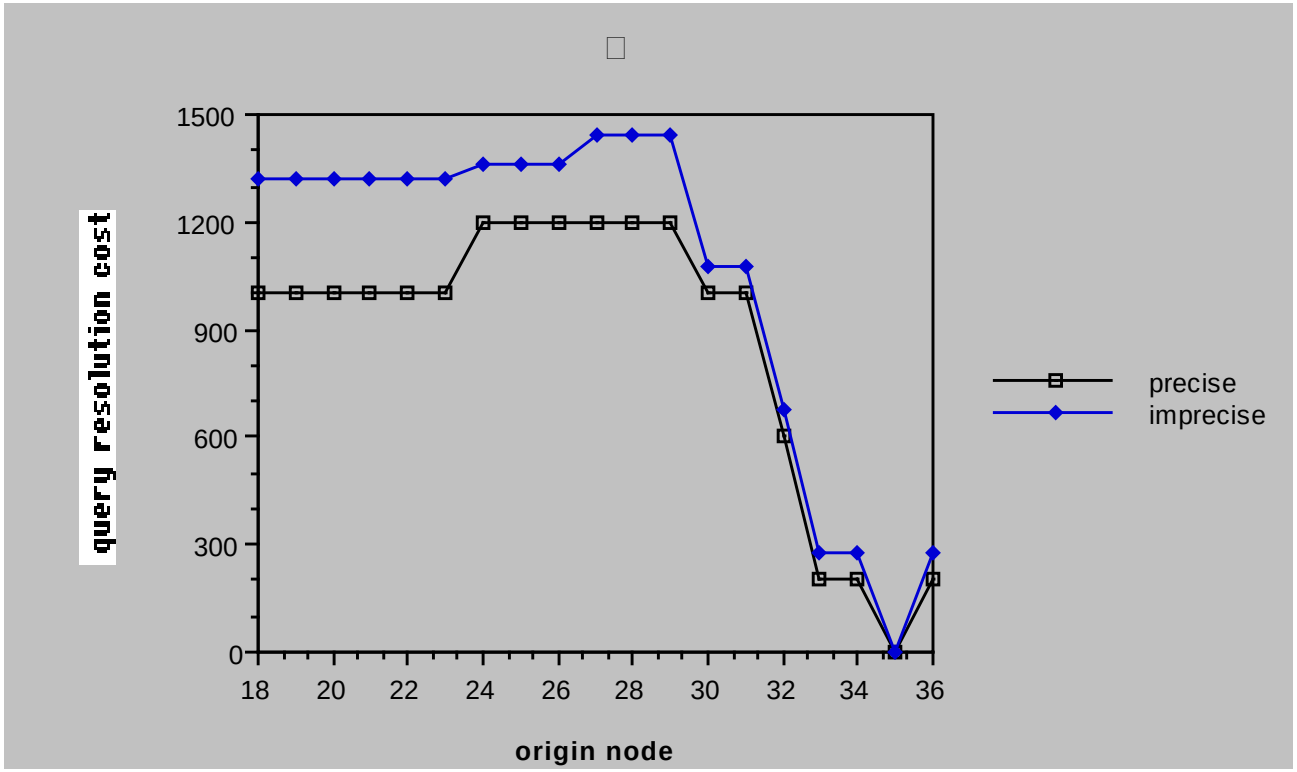
Heterogeneous Distributed Databases

- MultiDatabase Systems – Summary Schemas Model
  - Network Topology



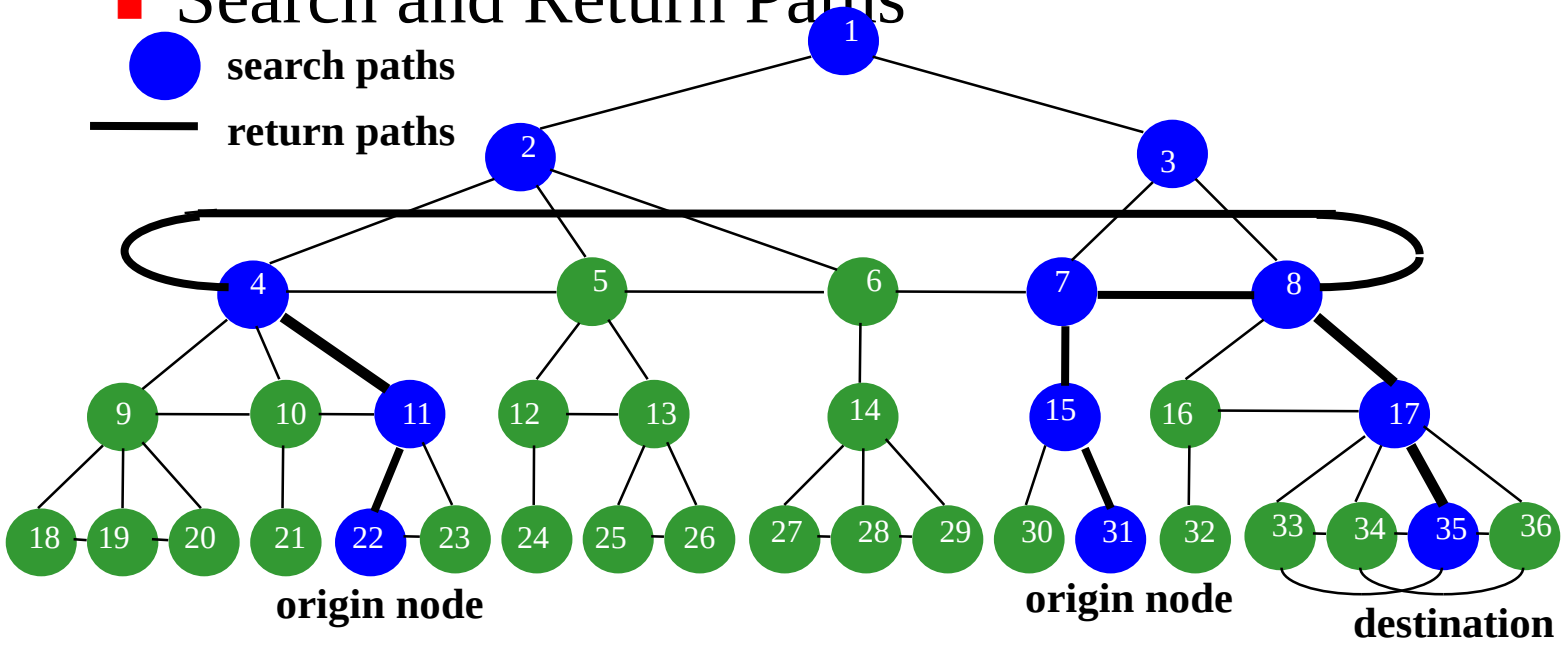
Heterogeneous Distributed Databases

- MultiDatabase Systems – Summary Schemas Model
  - Precise *vs.* Imprecise query



Heterogeneous Distributed Databases

- MultiDatabase Systems – Summary Schemas Model
  - Search and Return Paths

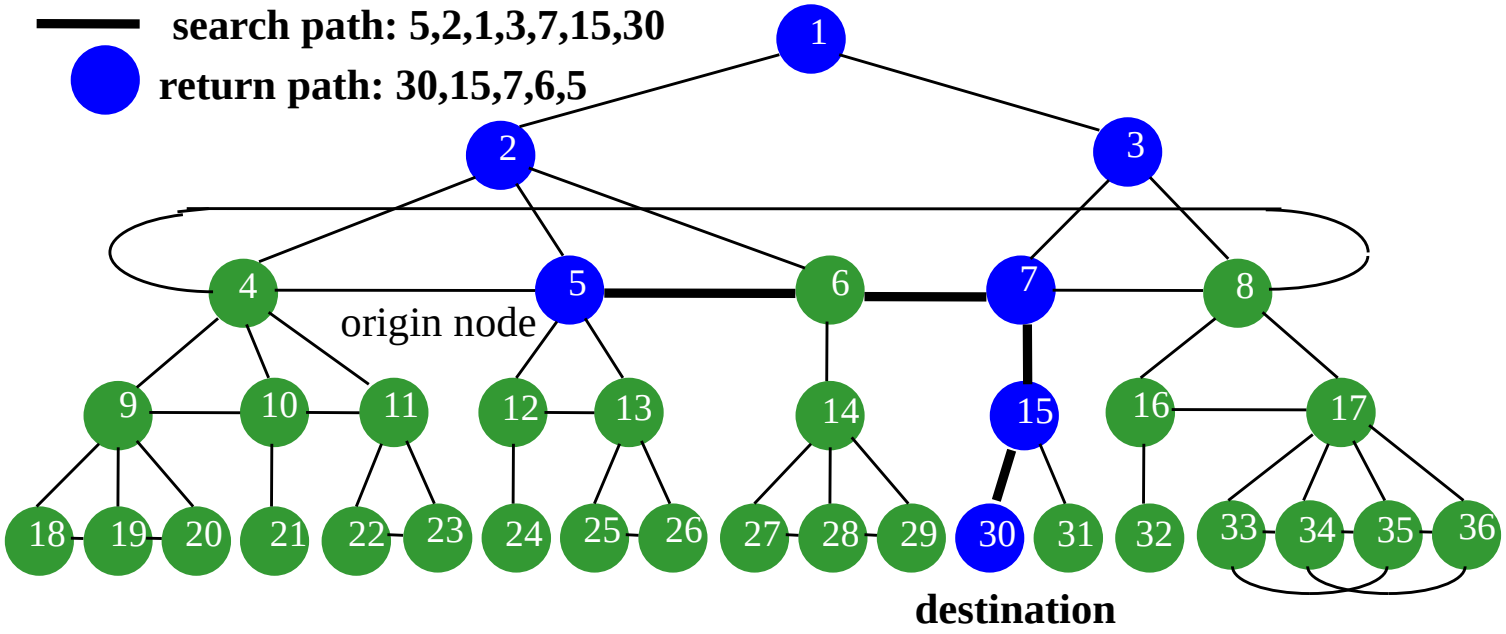


search path: 22,11,4,2,1,3,8,17,35
return path: 35,17,8,4,11,22

search path: 31,15,7,3,8,17,35
return path: 35,17,8,7,15,31

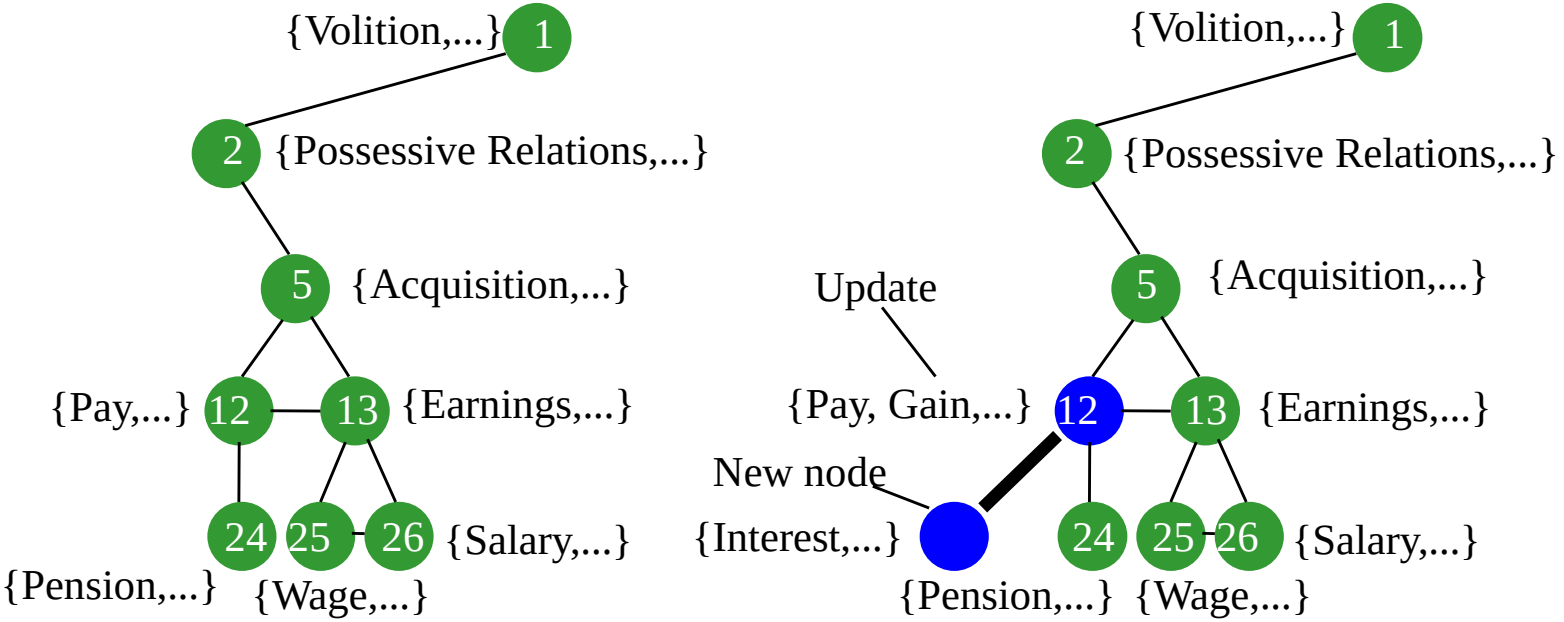
Heterogeneous Distributed Databases

- MultiDatabase Systems – Summary Schemas Model
  - Querying from Summary Schema Node



Heterogeneous Distributed Databases

- MultiDatabase Systems – Summary Schemas Model
  - Incremental Expansion

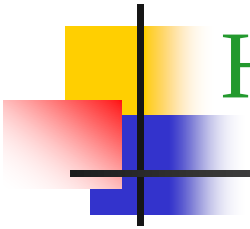


Heterogeneous Distributed Databases

- MultiDatabase Systems – Summary Schemas Model
  - The meta-data is by orders of magnitude smaller than the meta-data generated by the Global-schema approach.
  - Preserves local DBMS autonomy.
  - Provides good system scalability.
  - Reduces average search time.
  - Resolves imprecise queries.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Summary Schemas Model
  - Prototyped a Client/Server based SSM
  - Access Control and Security in SSM
  - Transaction management in SSM
  - Query processing in SSM



Heterogeneous Distributed Databases

- MultiDatabase Systems – Summary Schemas Model
  - Lessons learned
    - Lack of portability,
    - Lack of stability,
    - Relying on network connectivity.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Summary Schemas Model
  - Current distributed and multidatabase systems are designed to allow timely and reliable access to large amounts of autonomous and heterogeneous data sources — Sometimes, somewhere access environment.

Heterogeneous Distributed Databases

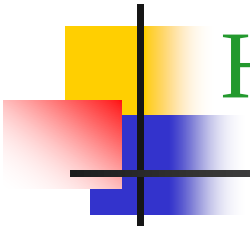
- MultiDatabase Systems – Summary Schemas Model
  - Adding mobility and wireless connection to the traditional multidatabase environments allow anytime, anywhere access to the information sources.
  - However, this advantage comes at the expense of additional complexities due to the network bandwidth, frequent disconnections, and limited processing power and resources.

Heterogeneous Distributed Databases

- MultiDatabase Systems – Extended SSM
  - Add mobility to the Summary Schemas Model (MDAS)
  - Transaction management in MDAS
  - Power management
  - Application of Software Agent
  - Multimedia support
  - Community of local ad-hoc nodes/sensor networks

Heterogeneous Distributed Databases

- It was decided to use agent technology to overcome technological constraints and shortcomings of the first prototype.
- Mobile agent-based computation paradigm responds to the:
  - Limited resources of mobile devices by migrating tasks to more powerful servers on the network, and
  - Intermittent connectivity by supporting disconnected operations.



Heterogeneous Distributed Databases

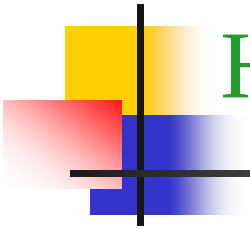
- Mobile users only need to maintain the connection during agent submission and retraction:
  - Handle intermittent network connectivity,
  - Reduce bandwidth consumption,
  - Reduce Power Consumption.

Heterogeneous Distributed Databases

	SSM Prototype	MAMDAS
Design Model	Client/Server Model	Agent-Based Model
Language	C & Java	Java
Communication Protocol	UNIX Sockets	Agent transportation & communication Protocols

Heterogeneous Distributed Databases

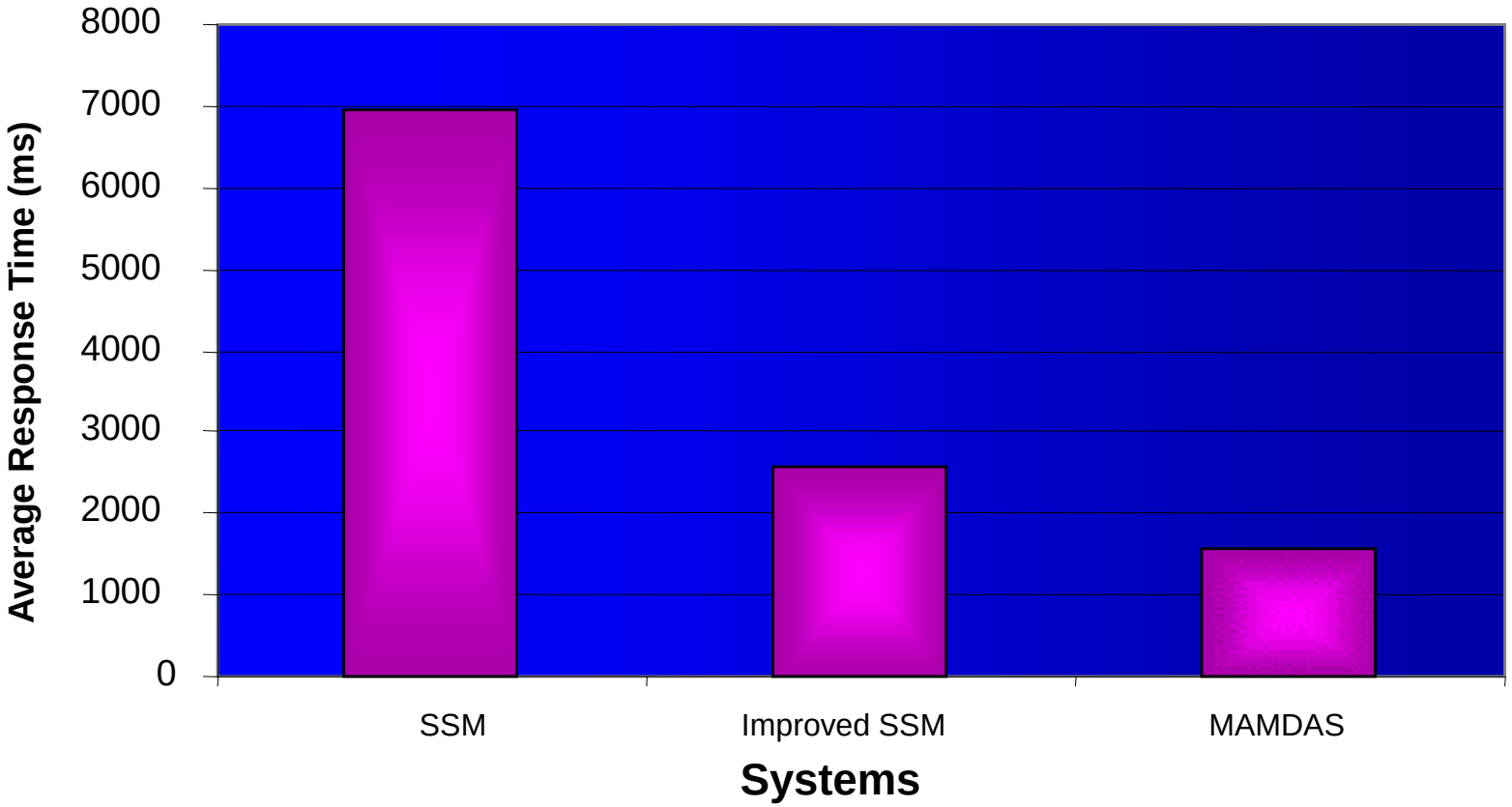
	SSM Prototype	MAMDAS
Average Response Time	Good	Six times faster
Scalability	Not Reported	Good
Portability	Poor	Good
Robustness	Poor	Good



Heterogeneous Distributed Databases

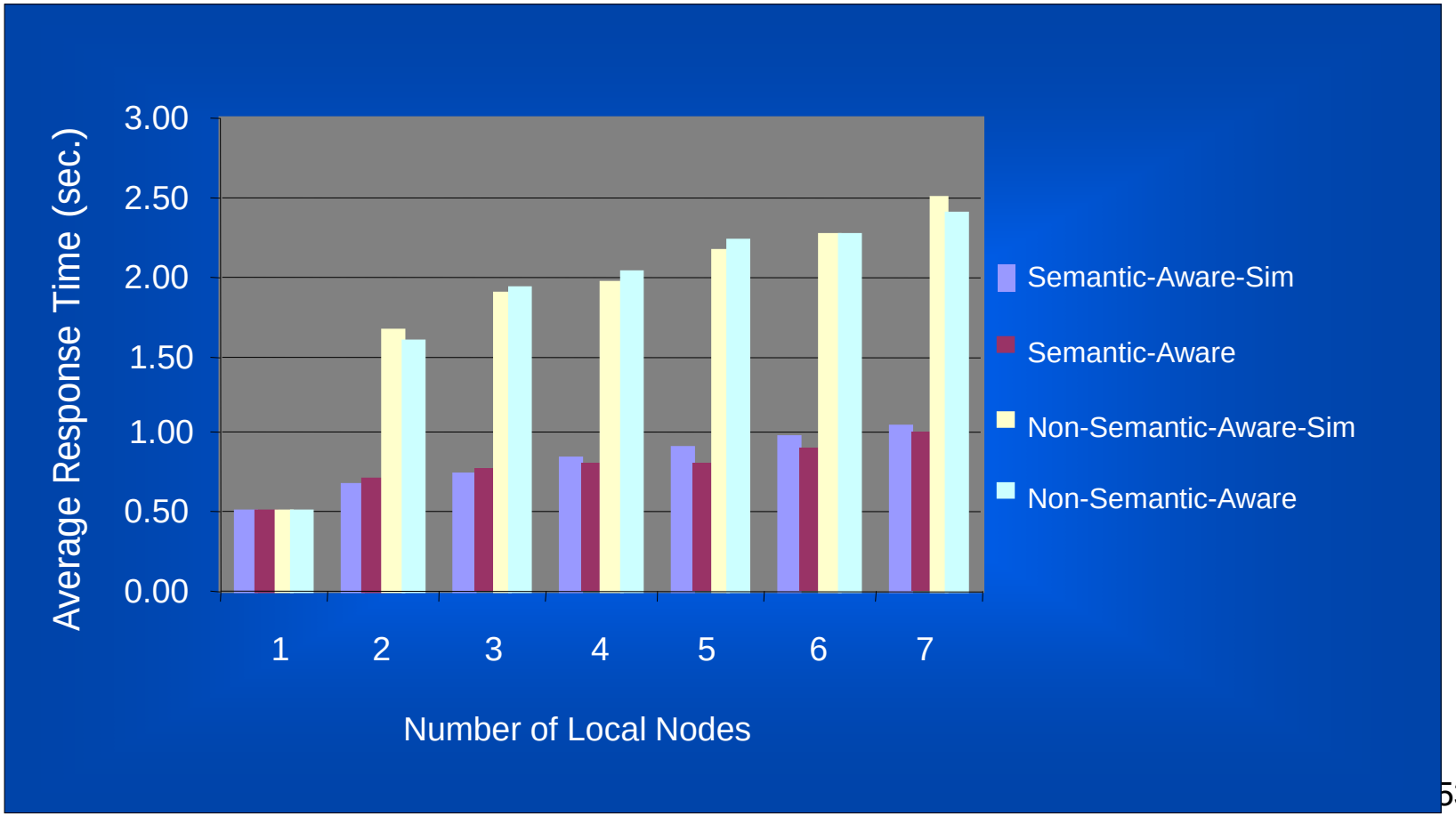
- A simulator was developed to simulate MAMDAS and quantitatively study its behavior.
- The simulator was extended to compare and contrast client/server and agent-based paradigms within the scope of the Summary Schemas Model.
- The simulator mimics our computational environment based on a set of rich statistical input parameters.

Heterogeneous Distributed Databases



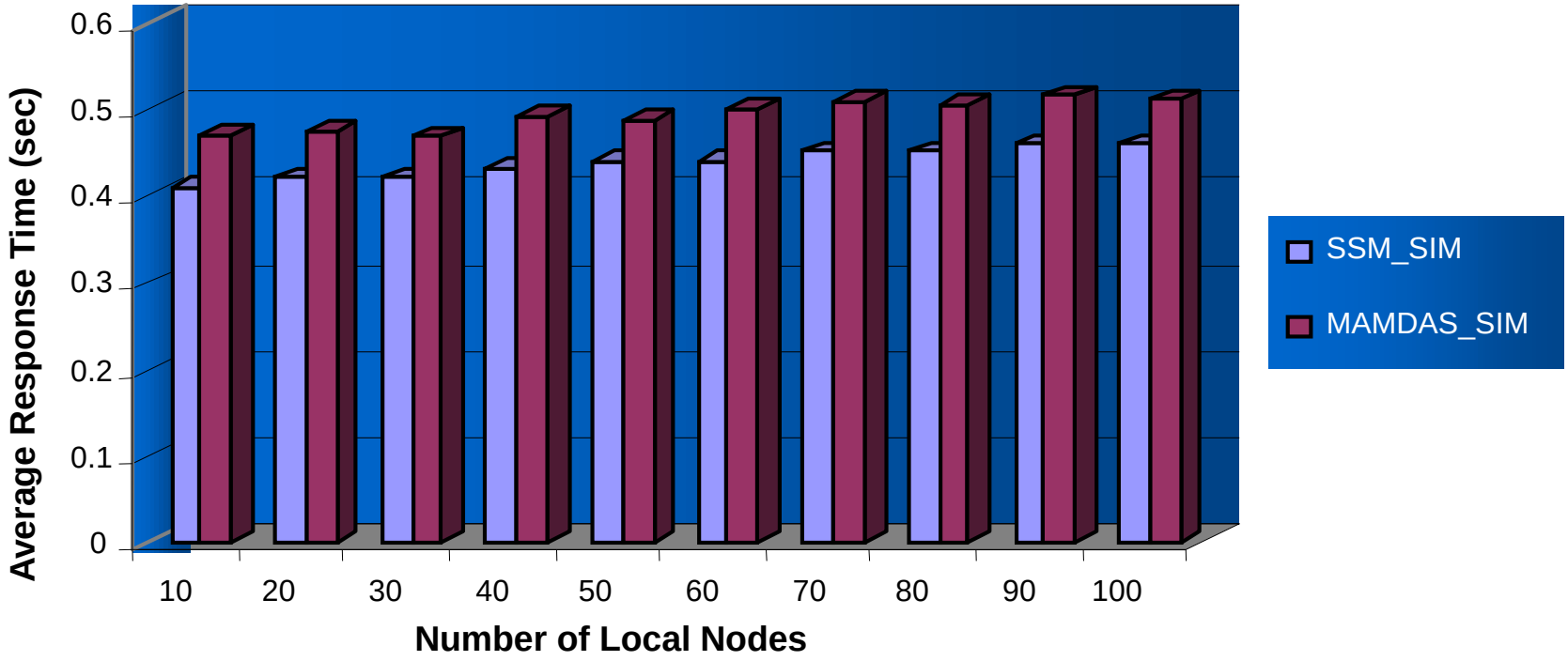
Heterogeneous Distributed Databases

- simulation results *vs.* prototype results



Heterogeneous Distributed Databases

- Assume Perfect Network Condition and Centralized Thesaurus



Heterogeneous Distributed Databases

- Number of Local Nodes = 70