Q.1 a. Define data independence. Explain two types of data independence.

Answer:
Data independence can be defined as the capacity to change the schema at one level of a database system without having to change the schema at the next higher level. Two types of data independence are:

1. **Logical level independence**: Logical level independence is the capacity to change the conceptual schema without having to change external schemas or application programs. We may change the conceptual schema to expand the database (by adding a record type or data item), to change constraints, or to reduce the database (by removing the record type or data item).

2. **Physical level independence**: Physical level independence is the capacity to change the internal schema without having to change the conceptual schema. Changes to internal schema may be needed because some physical files had to be reorganized.

b. What impact "constraints" can have on relationship types? Describe the different types of Relationship Constraints.

Answer:
Relationship types usually have certain constraints that limit the possible combinations of entities that may participate in the corresponding relationship set. The different types of relationship constraints are:

1. **Cardinality ratios for binary relationships:** The cardinality ratio for binary relationship specifies the maximum number of relationship instances that an entity can participate in the possible cardinality ratios for binary relationship types are 1:1, 1: N, N:1, and M:N.

2. **Participation constraints and existence dependencies:** The participation constraint specifies whether the existence of an entity depends on its being related to another entity via the relationship type. This constraint specifies the minimum number of relationship instances that each entity can participate in, and is sometimes called the minimum cardinality constraint. There are two types of participation constraints- total and partial. Total participation is also called existence dependency.

c. How does domain relational calculus differ from tuple calculus?

Answer:
Domain calculus differs from the tuple calculus in the type of variables used in formulas. Rather than having variables range over tuple, the variables range over single values from

domain of attributes. To form a relation of degree *n* for a quick result, we must have *n* of these domain variables-one for each attribute.

d. What is the use of the schema evolution commands available in SQL? Describe the various schema evolution commands available in SQL.

Answer:
The schema evolution commands available in SQL can be used to alter a schema by adding or dropping tables, attributes, constraints, and other schema elements. The various schema evolution commands available in SQL are:

1. **DROP command:** The DROP command can be used to drop named schema elements, such as tables, domains, or constraints. One can also drop a schema. For example, if a whole schema is not needed anymore, the DROP SCHEMA command can be used.

2. **The ALTER command:** The definition of base table or of other named schema elements can be changed by using the ALTER command. For base tables, the possible alter table actions include adding or dropping a column (attribute), changing a column definition, and adding or dropping table constraints.

e. Explain the two levels at which one can discuss the "goodness" of relation schemas.

Answer:
The two levels at which we can discuss the "goodness" of relation schemas are:

1. **The logical (or Conceptual) level-** how users interpret the relation schemas and the meaning of their attributes. Having good relation schemas at this level enables users to understand clearly the meaning of the data in the relations, and hence to formulate their queries correctly.

2. **The Implementation (or Storage) level-** how the tuples in a base relation are stored and updated. This level applies only to schemas of base relations-which will be stored as files-whereas at the logical level we are interested in schemas of both base relations and views.

f. Describe the three phases of concurrency control protocol.

Answer:
The three phases of concurrency control protocols are:

1. **Read phase:** A transaction can read values of committed data items from the database. However, updates are applied only to local copies (versions) of the data items kept in the transaction workspace.

2. **Validation phase:** Checking is performed to ensure that serializability will not be violated if the transaction updates are applied to the database.

3. **Write phase:** If the validation phase is successful, the transaction updates are applied to the database; otherwise, the updates are discarded and the transaction is restarted.

g. Describe the two main techniques for recovery from non catastrophic failures.

Answer:
The two main techniques for recovery from non catastrophic failures are:

1. **Deferred update:** The deferred update techniques do not physically update the data on disk until after a transaction reaches its commit point; then the updates are recorded in the database. Before reaching commit, all transaction updates are recorded in the transaction workspace (buffers). During commit, the updates are first recorded persistently in the log and then written to the database. If a transaction fails before reaching its commit point, it will not have changed the database in any way.

2. **Immediate update:** In the immediate update techniques, the database may be updated by some operations of a transaction before the transaction reaches its commit point. However, these operations are typically recorded in the log on the disk by force writing before they are applied to the database, making recovery still possible. If a transaction fails after recording some changes in the database but before reaching its commit point, the effects of its operations must be undone, that is, the transaction must be rolled back.

Q.2 a. What are the characteristics that distinguish the database approach from the traditional approach of programming with the files?

Answer:
In traditional file processing, each user defines and implements the files needed for a specific software application as part of programming the application. In the database approach, a single repository of data is maintained that is defined once and then is accessed by various users. The main characteristics that distinguish the database approach from the traditional approach of programming with the files are:

1. **Self-describing nature of a database system:** The fundamental characteristic of the database approach is that the database system contains not only the database itself but also a complete definition or description of the database structure and constraints. This definition is stored in the DBMS catalog which contains information such as the structure of each file, the type and storage format of each data item, and the various constraints on the data. In traditional file processing, data definition is typically part of the application program themselves. Hence, these programs are constrained to work with only one specific database, whose structure is declared in the application programs.

2. **Insulation between program and data, and database abstraction:** In traditional file system, the structure of data files is embedded in the application programs, so any changes to the structure of a file may require changing all programs that access this file. By contrast, DBMS access programs do not require such changes in most cases. The structure of data files is stored in the DBMS catalog separately from the access programs.

3. **Support of the multiple views of the data:** A database typically has many users, each of whom may require a different perspective or view of the database. A view may be a subset of the database or it may contain virtual data that is derived from the database files but is not explicitly stored. Some users may not need to be aware of whether the data they refer to is stored or derived. A multiuser DBMS whose users have a variety of distinct applications must provide facilities for defining multiple views.

4. **Sharing of data and multiuser transaction processing:** A multiuser DBMS must allow multi users to access the database at the same time. This is essential if data for multiple applications is to be integrated and maintained in a single database. The DBMS must include concurrency control software to ensure that several users trying to update the same data do so in a controlled manner so that the result of the updates is correct. Moreover, the DBMS must enforce several transaction properties. The isolation property ensures that each transaction appears to execute in isolation of other transactions, even though hundreds of transactions may be executing concurrently. The atomicity property ensures that either all the database operations in a transaction are executed or none are.

b. What is a data model? What are the different categories of data models?

Answer:

A data model is a collection of concepts that can be used to describe the structure of a database. It provides the necessary means to achieve this abstraction. Most data models also include a set of basic operations for specifying retrievals and updates on the database. Data model may also include concepts to specify the dynamic aspect or behavior of a database application. This allows the database designer to specify a set of valid user-defined operations that are allowed on the database objects. The generic

operations to insert, delete, modify, or retrieve any kind of object are often included in the basic data model operations. The different types of data models are:

1. **High-level or conceptual data models:** These data models provide concepts that are close to the way many users perceive data. They use concepts such as entities, attributes, and relationships.

2. **Low-level or physical data models:** These data models provide concepts that describe the details of how data is stored in the computer. They describe how data is stored as files in the computer by representing information such as record formats, record orderings, and access paths.

3. **Representational or implementation data models:** These data models provide concepts that may be understood by the end users but that are not too far removed from the way data is organized within the computer. These data models hide some details of data storage but can be implemented on a computer system in a direct way. These models are frequently used in traditional commercial DBMSs.

c. Explain the different types of interfaces provided by DBMS.

Answer:

The different types of interfaces provided by a DBMS include the following:

1. **Menu-based interfaces for web clients or browsing:** These interfaces present the users with the lists of options called menus, that lead the user through the formulation of a request menus do away with the need to memorize the specific commands and syntax of a query language. Browsing interfaces allow a user to look through the contents of a database in an exploratory and unstructured manner.

2. **Forms- based interfaces:** A forms-based interface displays a form to each user. Users can fill out all of the form entries to insert new data, or they fill out only certain entries, in which case the DBMS will retrieve matching data for the remaining entries.

3. **Graphical user interfaces:** a graphical user interface (GUI) typically displays a schema to the user in diagrammatic form. The user can then specify a query by manipulating the diagram. In many cases, GUIs utilize both menus and forms. Most GUIs use a pointing device, such as a mouse, to pick certain parts of the displayed schema diagram.

4. **Natural language interfaces:** These interfaces accept requests written in English or some other language and attempt to "understand" them. These interfaces have their own schema which is similar to database conceptual schema, as well as dictionary of important words.

5. **Interfaces for parametric users:** Parametric users, such as bank tellers, often have a small set of operations that they must perform repeatedly. Systems analysts and programmers design and implement a special interface for each known class of naïve users. Usually, a small set of abbreviated commands is included, with the goal of minimizing the number of keystrokes required for each request.

6. **Interfaces for the DBA:** Most database systems contain privileged commands that can be used only by the DBA's staff. These include commands for creating accounts, setting system parameters, granting account authorization, changing a schema, and reorganizing the storage structures of a database.

Q.3 a. Describe the different types of attributes that occur in the ER model.

Answer:

The different types of attributes that occur in the ER model are described below:

1. **Composite vs. simple (Atomic) attributes:** Composite attributes can be divided into smaller subparts, which represent more basic attributes with independent meanings. Attributes that are not divisible are called simple or atomic attributes. Composite attributes can form a hierarchy. The value of a composite attribute is the concatenation of the values of the constituent simple attributes. Composite attributes are useful to model situations in which a user sometimes refers to the composite attribute as a unit but at other times refers specifically to its components.

2. **Single-valued vs. multivalued attributes:** Most attributes have a single value for a particular entity, such attributes are called single-valued. For example, age is a single-valued attribute of a person. In some cases an attribute can have a set of values for the same entity. Such attributes are called multivalued. A multivalued attribute may have lower and upper bounds to constrain the number of values allowed for each individual entity.

3. **Stored vs. derived attributes**: In some cases, two (or more) attribute values are related- for example, the age and Birth Date attributes of a person. For a particular person entity, the value of Age can be determined from the current (today's) date

and the value of that person's BirthDate. The Age attribute is hence called a derived attribute and is said to be derivable from the BirthDate attribute, which is called a stored attribute.

4. **Null values**: In some cases a particular entity may not have an applicable value for an attribute. For example, flat number attribute of an address applies on to addresses that are in flats and not to other types of residences, such as single-family homes. An address for a single-family home would have null for its Flat number attribute. Null can also be used if we do not know the value of an attribute for a particular entity.

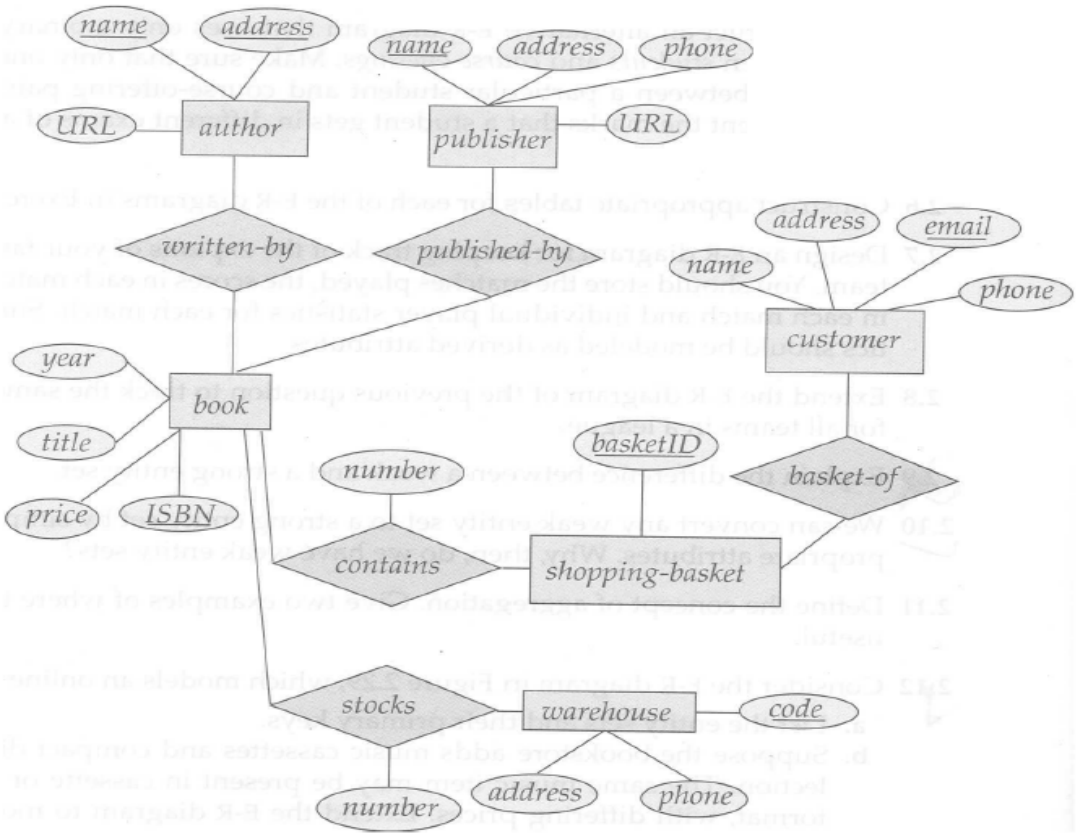
5. **Complex attributes**: Composite and multivalued attributes can be nested in arbitrary way. We can represent arbitrary nesting by grouping components of a composite attribute between parentheses ( ) and separating the components with commas, and by displaying multivalued attributes between { }. Such attributes are called complex attributes.

b. Consider the database of an online bookstore.

- **Every *book* has a title, isbn, year and price. The store also keeps the data of *author* and *publisher* for any book.**
- **For authors, the database keeps the name, address and the URL of their homepage. For publishers, the database keeps the name, address, phone number and the URL of their website.**
- **The store has several *warehouses*, each of which has a code, address and phone number. The warehouse stocks several books. A book may be stocked at multiple warehouses.**
- **The database records the number of copies of a book stocked at various warehouses.**
- **The bookstore keeps the name, address, email-id, and phone number of its *customers*.**
- **A customer owns several *shopping basket*. A shopping basket is identified by a basketID and contains several books.**
- **Some shopping baskets may contain more than one copy of the same book. The database records the number of copies of each book in any shopping basket**

Design an ER diagram for such a bookstore.

Answer: ER diagram for such a bookstore is given below:



Q.4 a. What are the different types of constraints that can be specified on a relational database?

Answer:
Constraints on databases can generally be divided into the following four main categories:

1. **Inherent model-based constraints:** These constraints are inherent in the data model.
2. **Schema-based constraints:** These constraints can be directly expressed in the schemas of the data model, typically by specifying them in the DDL.
3. **Application-based constraints:** Constraints that cannot be directly expressed in the schemas of the data model, and hence must be expressed and enforced by the application programs.
4. **Data dependencies:** These include functional dependencies and multivalued dependencies. They are mainly used for testing the "goodness" of the design of a relational database and are utilized in the process called normalization.

b. Define entity integrity constraint, referential integrity constraint and foreign keys.

Answer:

The entity integrity constraint states that no primary key value can be null. This is because the primary key value is used to identify individual tuples in a relation. Having null value for the primary key implies that we cannot identify some tuples. Key constraints and entity integrity constraints are specified on individual relations. The referential integrity constraint is specified between two relations and is used to maintain the consistency among tuples in the two relations. Informally, the referential integrity constraint states that a tuple in one relation that refers to another relation must refer to an existing tuple in that relation. A foreign key specifies a referential integrity constraint between the two relation schemas R_1 and R_2.

c. Describe the three basic update operations on a relational database. How they can violate constraints?

Answer:

The three basic update operations on a relational database are:

1. **Insert:** Insert is used to insert a new tuple or tuples in a relation. Insert operation can violate domain constraints if an attribute value is given that does not appear in the corresponding domain. Key constraints can be violated if a key value in the new tuple *t* already exists in another tuple in the relation *r*(R). Entity integrity can be violated if the primary key of the new tuple *t* is null. Referential integrity can be violated if the value of any foreign key in *t* refers to a tuple that does not exist in the referenced relation.
2. **Delete:** Delete is used to delete tuple or tuples in a relation. The delete operation can violate only referential integrity, if the tuple being deleted is referenced by the foreign keys from other tuples in the database.
3. **Update (or modify):** Update (or modify) is used to change the values of some attributes in existing tuples. If the foreign key is modified, the DBMS must ensure that the new value refers to an existing tuple in the referenced relation (or is null). Similar options exist to deal with referential integrity violations caused by Update as those options discussed for delete operation. In fact, when a referential integrity constraint is specified in the DDL, the DBMS will allow the user to choose separate options to deal with a violation caused by Delete and a violation caused by Update.

Whenever these operations are applied, the integrity constraints specified on the relational database schemas should not be violated.

Q.5 a. Define relational algebra and relational calculus. Why relational algebra is important for the relational model?

Answer:

The basic set of operations for the relational model is the relational algebra. These operations enable a user to specify basic retrieval requests. The result of a retrieval is a new relation, which may have been formed from one or more relations. The algebra operations thus produce new relations which can be further manipulated using operations of the same algebra. The algebra defines a set of operations for the relational model.

The relational calculus provides a higher- level declarative notation for specifying relational queries. A relational calculus expression creates a new relation, which is specified in terms of variables that range over rows of the stored database relations (in tuple calculus) or over columns of the stored relations (in domain calculus). In a calculus expression, there is no order of operations to specify how to retrieve the query result-a calculus expression specifies only what information the result should contain. The relational calculus is important because it has a firm basis in mathematical logic and because the SQL for RDBMSs has some of its foundations in the tuple relational calculus.

The relational algebra is important due to the following reasons:

1. It provides a formal foundation for relational model operations.
2. It is used as a basis for implementing and optimizing queries in RDBMSs.
3. Some of its concepts are incorporated into the SQL for RDBMSs.

b. Suppose you are given a relation R = (A,B,C,D,E) with the following functional dependencies: {CE →D,D→ B,C → A}.

- **Find all candidate keys.**
- **Identify the best normal form that R satisfies (1NF, 2NF, 3NF, or BCNF).**

If the relation is not in BCNF, decompose it until it becomes BCNF. At each step, identify a new relation, decompose and re-compute the keys and the normal forms they satisfy.

Answer:
a. The only key is {C,E}
b. The relation is in 1NF
c. Decompose into R1=(A,C) and R2=(B,C,D,E). R1 is in BCNF, R2 is in 2NF. Decompose R2 into, R21=(C,D,E) and R22=(B,D). Both relations are in BCNF.

c. Describe the basic constraints that can be specified in SQL as part of table creation.

Answer:
The basic constraints that can be specified in SQL as part of table creation are:

1. **Specifying attribute constraints and attribute defaults:** As SQL allows NULLs as attribute values, a constraint NOT NULL may be specified if NULL is not permitted for a particular attribute. This is always implicitly specified for the

attributes that are part of the primary key of each relation, but it can be specified for any other attributes whose values are required to be NULL. Another type of constraint can restrict attribute or domain values using the CHECK clause following an attribute or domain definition. The CHECK clause can also be used in conjunction with the CREATE DOMAIN statement.

2. **Specifying key and referential integrity constraints:** As keys and referential integrity constraints are very important, there are special clauses within the CREATE TABLE statement to specify them. The PRIMARY KEY clause specifies one or more attributes that make up the primary key of a relation. If a primary key has a single attribute, the clause can follow the attribute directly. The UNIQUE clause specifies alternate (secondary) keys. Referential integrity is specified via the FOREIGN KEY clause. A referential integrity constraint can be violated when tuples are inserted or deleted, or when a foreign key or primary key attribute value is modified. The default action that SQL takes for an integrity violation is to reject the update operation that will cause a violation. However, the schema designer can specify an alternative action to be taken if a referential integrity constraint is violated, by attaching a referential triggered clause to any foreign key constraint.

3. **Giving names to constraints:** A constraint may be given a constraint name, following the keyword CONSTRAINT. The names of all constraints within a particular schema must be unique. A constraint name is used to identify a particular constraint in case the constraint must be dropped later and replaced with another constraint. Giving names to constraints is optional.

4. **Specifying constraints on tuples using CHECK:** In addition to key and referential integrity constraints, which are specified by special keywords, other table constraints can be specified through additional CHECK clauses at the end of a CREATE TABLE statement. These can be called tuple-based constraints because they apply to each tuple individually and are checked whenever a tuple is inserted or modified.

Q.6 a. What are the problems that are associated with null values in tuples?

Answer:
The problems that are associated with null values in tuples are:

1. If many of the attributes do not apply to all tuples in a relation, we end up with many nulls in these tuples. This can waste space at the storage level and may also

lead to problems with understanding the meaning of the attributes and with specifying JOIN operations at the logical level.

2. Another problem with nulls is how to account for them when aggregate operations such as COUNT or SUM are applied.

3. Moreover, nulls can have multiple interpretations such as the following:
   - The attribute does not apply to this tuple.
   - The attribute value for this tuple is unknown.
   - The value is known but absent, that is, it has not been recorded yet.

Having the same representation for all nulls compromises the different meanings they may have.

b. Define normal form of a relation. What are the properties that the relational schemas should possess, the existences of which are confirmed by the process of normalization through decomposition?

Answer:

The normal form of a relation refers to the highest normal form condition that it meets, and hence indicates the degree to which it has been normalized. Normal forms, when considered in isolation from other factors, do not guarantee a good database design. The process of normalization through decomposition must also confirm the existence of the following properties that the relation schemas, taken together, should possess:

1. **The lossless join or nonadditive join property:** This property guarantees that the spurious tuple generation problem does not occur with respect to the relation schemas created after decomposition.
2. **The dependency preservation property:** This property ensures that each functional dependency is represented in some individual relation resulting after decomposition.

c. Define first normal form (1NF).

Answer:

First normal form (1NF) is now considered to be part of the formal definition of a relation in the basic (flat) relational model; historically, it was defined to disallow multivalued attributes, composite attributes, and their combinations. It states that the domain of an attribute must include only atomic (simple, indivisible) values and that the value of any attribute in a tuple must be a single value from the domain of that attribute. Hence 1NF disallows having a set of values, a tuple of values, or a combination of both as an attribute value for a single tuple. In other words, 1NF disallows "relations within relations" or" relations as attribute values within tuples." The only attribute values permitted by 1NF are single atomic (or indivisible) values.

Q.7 a. Describe the protocols of concurrency control techniques that guarantee serializability.

Answer:
The protocols of concurrency control techniques that guarantee serializability are:

1. **Locking:** One important set of protocols employs the technique of locking data items to prevent multiple transactions from accessing the items concurrently. A lock is a variable associated with the data item that describes the status of the item with respect to possible operations that can be applied to it. Generally, there is one lock for each data item in the database. Locks are used as a means of synchronizing the access by concurrent transactions to the database items.

2. **Timestamps:** A timestamp is a unique identifier for each transaction generated by the system. Typically, timestamp values are assigned in order in which the transactions are submitted to the system, so a timestamp can be thought of as the transaction start time. Concurrency control techniques based on timestamp ordering do not use locks.

3. **Multiversion concurrency control techniques:** Multiversion concurrency control protocols use multiple versions of a data item. They keep the old values of a data item when the item is updated. These are known as multiversion concurrency control, because several versions (values) of an item are maintained. When a transaction requires access to an item, an appropriate version is chosen to maintain the serializability of the currently executing schedule, if possible. When a transaction writes an item, it writes a new version and the old version of the item is retained.

4. **Optimistic protocols:** These protocols are based on the concept of validation or certification of a transaction after it executes its operations. No checking is done while the transaction is executing. In this scheme, updates in the transaction are not applied directly to the database items until the transaction reaches its end. During transaction execution, all updates are applied to local copies of the data items that are kept for the transaction. At the end of the transaction execution, a validation phase checks whether any of the transaction's updates violate serializability. If serializability is not violated, the transaction is committed and the database is updated from the local copies; otherwise, the transaction is aborted and then restarted later.

b. What are the advantages of distributed databases?

Answer:

The main advantages of distributed databases are:

1. **Management of distributed data with different level of transparency:** Ideally, a DBMS should be distribution transparent in the sense of hiding the details of where each file (table, relation) is physically stored in the system. The following types of transparencies are possible:
    - **Distribution or network transparency:** This refers to the freedom for the user from the operational details of the network. It includes locational transparency which refers to the fact that the command used to perform a task is independent of the location of data and the location of the system where the command was issued. Naming transparency implies that once a name is specified, the named objects can be accessed unambiguously without additional specification.
    - **Replication transparency:** Replication transparency makes the user unaware of the existence of copies of data that may be stored at multiple sites for better availability, performance, and reliability.
    - **Fragmentation transparency:** Fragmentation transparency makes the user unaware of the existence of fragments which may be horizontal or vertical.
2. **Increased reliability and availability:** When the data and DBMS software are distributed over several sites, one site may fail while the other sites continue to operate. Only the data and software that exist at the failed site cannot be accessed. This improves both reliability and availability.
3. **Improved performance:** A distributed DBMS fragments the database by keeping the data closer to where it is needed most. Data localization reduces the contention for CPU and I/O services and simultaneously reduces access delays involved in wide area networks. When a large database is distributed over multiple sites, smaller databases exist at each site. As a result, local queries and transactions accessing data at a single site have better performance because of the smaller local databases. In addition, each site has a smaller number of transactions executing than if all transactions are submitted to a single centralized database. Moreover, interquery and intraquery parallelism can be achieved by executing multiple queries at different sites, or by breaking up a query into a number of subqueries that execute in parallel. This contributes to improved performance.

4. **Easier expansion:** In a distributed environment, expansion of the system in terms of adding more data, increasing database sizes, or adding more processors is much easier.

c. Describe the issues that affect the design of federated database system (FDBS).

Answer:
The issues that affect the design of FDBS are:

1. **Differences in data models:** Databases in an organization come from a variety of data models including the so-called legacy models (network and hierarchical), the relational data model, the object data model, and even files. The modeling capabilities of the models vary. Hence, to deal with them uniformly via a single global schema or to process them is challenging. Even if two databases are both from the RDBMS environment, the same information may be represented as an attribute name, as a relation name, or as a value in different databases. This calls for an intelligent query- processing mechanism that can relate information based on metadata.
2. **Differences in constraints:** Constraint facilities for specification and implementation vary from system to system. They are comparable features that must be reconciled in the construction of global schema. For example, the relationships from ER models are represented as referential integrity constraints in a relational model. The global schema must also deal with potential conflicts among constraints.
3. **Differences in query languages:** Even with the same data model, the languages and their versions vary. For example, SQL has multiple versions like SQL-89, SQL-92, and SQL-99, and each has its own set of data types, comparison operators, string manipulation features, and so on.

Text Books

1. R. Elmasri and S. Navathe, "Fundamental of Database Systems", Addison Wesley, 5th Edition, 2007.

2. R Ramakrishnan & J Gehrke, Database Management Systems, McGraw Hill, Third Edition, 2002.