2069

1.) Explain the following terms :
• Data mining
Analysing large amount of data searching for the occurrences of specific patterns or relationships.
• ECA model
Model used for specifying active database rules. It has three components:

1. Event that triggers the rule
2. Condition that determines whether the rules or action should be executed
3. Action to be taken

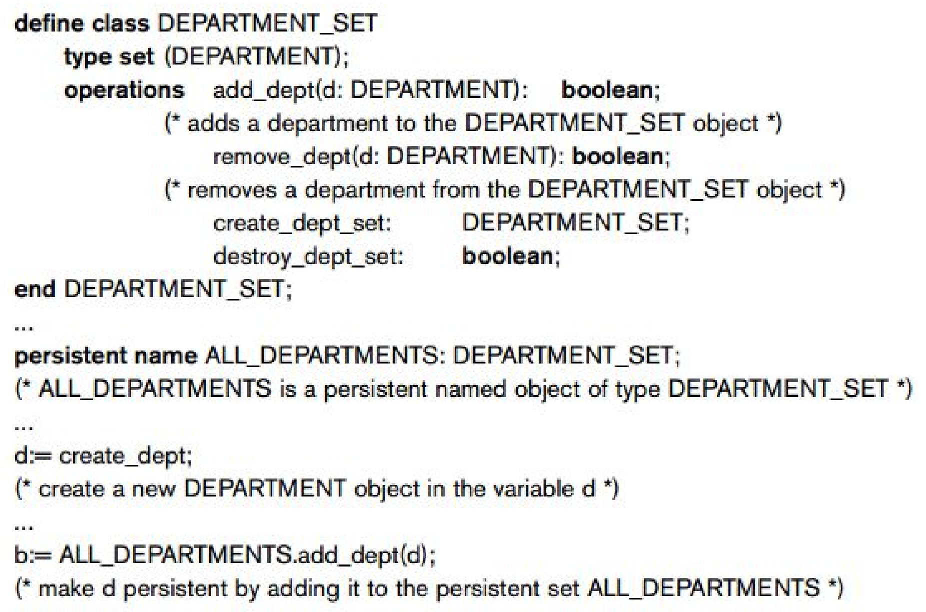
• Spatial database
Databases that keep track of objects in a multidimensional space. Eg. cartographic databases that store maps include two dimensional spatial descriptions of their objects like countries, states, cities, roads.
• Specialization and generalization in an ERR model
Speci
• XML and HTML
• **GIS**
Geographic information systems (GIS) are used to collect, model, store, and analyze information describing physical properties of the geographical world. The scope of GIS broadly encompasses two types of data: (1) spatial data, originating from maps, digital images, administrative and political boundaries, roads, transportation networks; physical data such as rivers, soil characteristics, climatic regions, land elevations, and (2) nonspatial data, such as socio-economic data (like census counts), economic data, and sales or marketing information. GIS is a rapidly developing domain that offers highly innovative approaches to meet some challenging technical demands

2.) How can you convert an ERR design to relational design? Discuss with suitable example. [Unit 2]
http://tinman.cs.gsu.edu/~raj/4710/sp03/ch9-part1.pdf

3.) What is OID? How persistent objects are maintained in OODatabase? [Unit 3]
An ODMS provides a unique identity to each independent object stored in the database. This unique identity is typically implemented via a unique, system-generated object identifier (OID). The value of an OID is not visible to the external user, but is used internally by the system to identify each object uniquely and to create and manage inter-object references. The OID can be assigned to program variables of the appropriate type when needed. The main property required of an OID is that it be immutable; that is, the OID value of a particular object should not change. This preserves the identity of the real-world object being represented. Hence, an ODMS must have some mechanism for generating OIDs and preserving the immutability property. It is also desirable that each OID be used only once; that is, even if an object is removed from the database, its OID should not be assigned to another object. These two

properties imply that the OID should not depend on any attribute values of the object, since the value of an attribute may be changed or corrected. We can compare this with the relational model, where each relation must have a primary key attribute whose value identifies each tuple uniquely. In the relational model, if the value of the primary key is changed, the tuple will have a new identity, even though it may still represent the same real-world object. Alternatively, a real-world object may have different names for key attributes in different relations, making it difficult to ascertain that the keys represent the same real-world object (for example, the object identifier may be represented as Emp_id in one relation and as Ssn in another).

Persistent objects are stored in the database and persist after program termination. The typical mechanisms for making an object persistent are naming and reachability. The naming mechanism involves giving an object a unique persistent name within a particular database. This persistent object name can be given via a specific statement or operation in the program, as shown in Figure 11.3.



```
define class DEPARTMENT_SET
    type set (DEPARTMENT);
    operations   add_dept(d: DEPARTMENT):    boolean;
            (* adds a department to the DEPARTMENT_SET object *)
                remove_dept(d: DEPARTMENT): boolean;
            (* removes a department from the DEPARTMENT_SET object *)
                create_dept_set:        DEPARTMENT_SET;
                destroy_dept_set:       boolean;
end DEPARTMENT_SET;
...
persistent name ALL_DEPARTMENTS: DEPARTMENT_SET;
(* ALL_DEPARTMENTS is a persistent named object of type DEPARTMENT_SET *)
...
d:= create_dept;
(* create a new DEPARTMENT object in the variable d *)
...
b:= ALL_DEPARTMENTS.add_dept(d);
(* make d persistent by adding it to the persistent set ALL_DEPARTMENTS *)
```

Figure 11.3
Creating persistent objects by naming and reachability.

The named persistent objects are used as entry points to the database through which users and applications can start their database access. Obviously, it is not practical to give names to all objects in a large database that includes thousands of objects, so most objects are made persistent by using the second mechanism, called reachability. The reachability mechanism works by making the object reachable from some other persistent object. An object B is said to be reachable from an object A if a sequence of references in the database lead from object A to object B. If we first create a named persistent object N, whose state is a set (or possibly a bag) of objects of some class C, we can make objects of C persistent by adding them to the set, thus making them reachable from N. Hence, N is a named object that defines a persistent collection of objects of class C. In the object model standard, N is called the extent of C

4.) Discuss the relative advantages of centralized and distributed database.[Unit 1]
A distributed database allows a user convenient and transparent access to data which is not stored at the site, while allowing each site control over its own local data. A distributed database can be made more reliable than a centralized system because if one site fails, the database can continue functioning, but if the centralized system fails, the database can no longer continue with its normal operation. Also, a distributed database allows parallel execution of queries and possibly splitting one query into many parts to increase throughput.

• A centralized system is easier to design and implement. A centralized system is cheaper to operate because messages do not have to be sent.

5.) Describe different implementation issues with object relational database system.[Unit 3]

There are various implementation issues regarding the support of an extended type system with associated functions (operations). We briefly summarize them here:

• The ORDBMS must dynamically link a user-defined function in its address space only when it is required. As we saw in the case of the two ORDBMSs, numerous functions are required to operate on two- or three-dimensional spatial data, images, text, and so on. With a static linking of all function libraries, the DBMS address space may increase by an order of magnitude. Dynamic linking is available in the two ORDBMSs that we studied.

• Client-server issues deal with the placement and activation of functions. If the server needs to perform a function, it is best to do so in the DBMS address space rather than remotely, due to the large amount of overhead. If the function demands computation that is too intensive or if the server is attending to a very large number of clients, the server may ship the function to a separate client machine. For security reasons, it is better to run functions at the client using the user ID of the client. In the future functions are likely to be written in interpreted languages like JAVA.

• It should be possible to run queries inside functions. A function must operate the same way whether it is used from an application using the application program interface (API), or whether it is invoked by the DBMS as a part of executing SQL with the function embedded in an SQL statement. Systems should support a nesting of these "callbacks."

• Because of the variety in the data types in an ORDBMS and associated operators, efficient storage and access of the data is important. For spatial data or multidimensional data, new storage structures such as R-trees, quad trees, or Grid files may be used. The ORDBMS must allow new types to be defined with new access structures. Dealing with large text strings or binary files also opens up a number of storage and search options. It should be possible to explore such new options by defining new data types within the ORDBMS.

6.) Discuss the different techniques for executing equijoin of two files located at different sites. What main factors affect the cost of data transfer? [Unit 1]

In a distributed system, several additional factors further complicate query processing. The first is the cost of transferring data over the network. This data includes intermediate files that are transferred to other sites for further processing, as well as the final result files that may have to be transferred to the site where the query result is needed.
Factors affecting cost of data transfer:
No of tuples in relation:
No of sites and relations:
Type of Network: Although these costs may not be very high if the sites are connected via a high-performance local area network, they become quite significant in other types of networks.

Technique 1:
Consider the query Q: For each employee, retrieve the employee name and the name of the department for which the employee works. This can be stated as follows in the relational algebra:

$$Q: \pi_{\text{Fname,Lname,Dname}}(\text{EMPLOYEE} \bowtie_{\text{Dno=Dnumber}} \text{DEPARTMENT})$$

The result of this query will include 10,000 records, assuming that every employee is related to a department. Suppose that each record in the query result is 40 bytes long.

The query is submitted at a distinct site 3, which is called the result site because the query result is needed there. Neither the EMPLOYEE nor the DEPARTMENT relations reside at site 3. There are three simple strategies for executing this distributed query:

1. Transfer both the EMPLOYEE and the DEPARTMENT relations to the result site, and perform the join at site 3. In this case, a total of 1,000,000 + 3,500 = 1,003,500 bytes must be transferred.

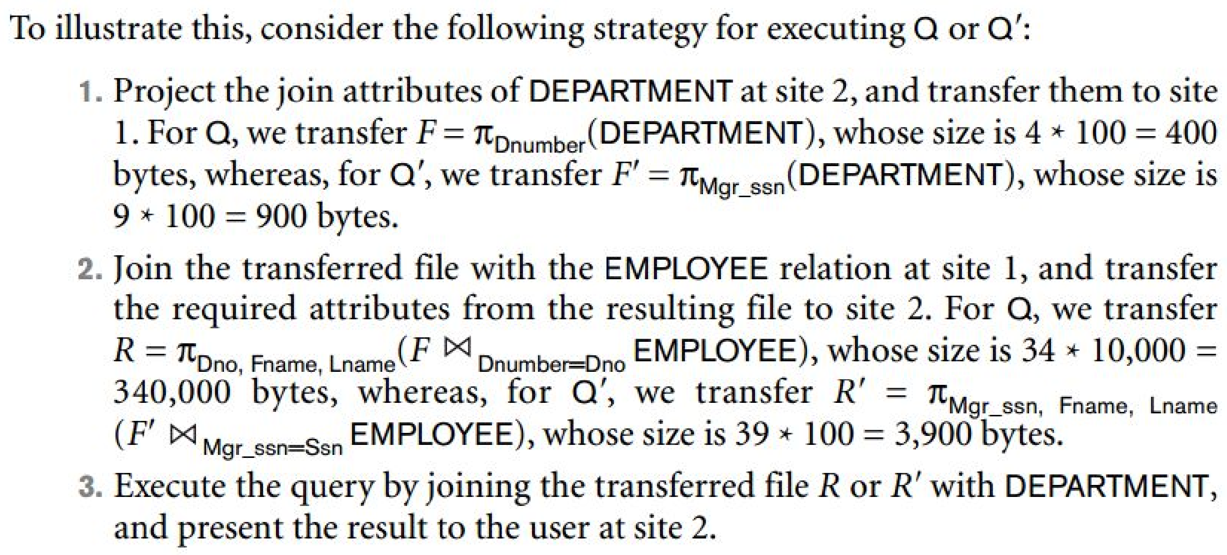
2. Transfer the EMPLOYEE relation to site 2, execute the join at site 2, and send the result to site 3. The size of the query result is 40 * 10,000 = 400,000 bytes, so 400,000 + 1,000,000 = 1,400,000 bytes must be transferred.

3. Transfer the DEPARTMENT relation to site 1, execute the join at site 1, and send the result to site 3. In this case, 400,000 + 3,500 = 403,500 bytes must be transferred

Technique 2: Using semijoin

The idea behind distributed query processing using the semijoin operation is to reduce the number of tuples in a relation before transferring it to another site. Intuitively, the idea is to send the joining column of one relation R to the site where the other relation S is located; this column is then joined with S. Following that, the join attributes, along with the attributes required in the result, are projected out and shipped back to the original site and joined with R. Hence, only the joining column of R is transferred in one direction, and a subset of S with no extraneous tuples or attributes is transferred in the other direction. If only a small fraction of the tuples in S participate in the join, this can be quite an efficient solution to minimizing data transfer

To illustrate this, consider the following strategy for executing Q or Q':



To illustrate this, consider the following strategy for executing Q or Q′:

1. Project the join attributes of DEPARTMENT at site 2, and transfer them to site 1. For Q, we transfer $F = \pi_{\text{Dnumber}}(\text{DEPARTMENT})$, whose size is $4 * 100 = 400$ bytes, whereas, for Q′, we transfer $F' = \pi_{\text{Mgr\_ssn}}(\text{DEPARTMENT})$, whose size is $9 * 100 = 900$ bytes.
2. Join the transferred file with the EMPLOYEE relation at site 1, and transfer the required attributes from the resulting file to site 2. For Q, we transfer $R = \pi_{\text{Dno, Fname, Lname}}(F \bowtie_{\text{Dnumber=Dno}} \text{EMPLOYEE})$, whose size is $34 * 10{,}000 = 340{,}000$ bytes, whereas, for Q′, we transfer $R' = \pi_{\text{Mgr\_ssn, Fname, Lname}}(F' \bowtie_{\text{Mgr\_ssn=Ssn}} \text{EMPLOYEE})$, whose size is $39 * 100 = 3{,}900$ bytes.
3. Execute the query by joining the transferred file R or R' with DEPARTMENT, and present the result to the user at site 2.

7.) Differentiate between attributes and elements in XML? List some of the important attributes used in specifying elements in XML schema.[Unit 5]

Two main structuring concepts are used to construct an XML document: elements and attributes. It is important to note that the term attribute in XML is not used in the same manner as is customary in database terminology, but rather as it is used in document description languages such as HTML and SGML.4 Attributes in XML provide additional information that describes elements, as we will see.

Figure below shows an example of an XML element called “Projects”. As in HTML, elements are identified in a document by their start tag and end tag. The tag names are enclosed between angled brackets < ... >, and end tags are further identified by a slash,

```
<?xml version= “1.0” standalone=“yes”?>
<Projects>
<Project>
        <Name>ProductX</Name>
        <Price>100</Price>
</Project>

<Project>
        <Name>ProductY</Name>
        <Price>200</Price>
</Project>
</Projects>
```

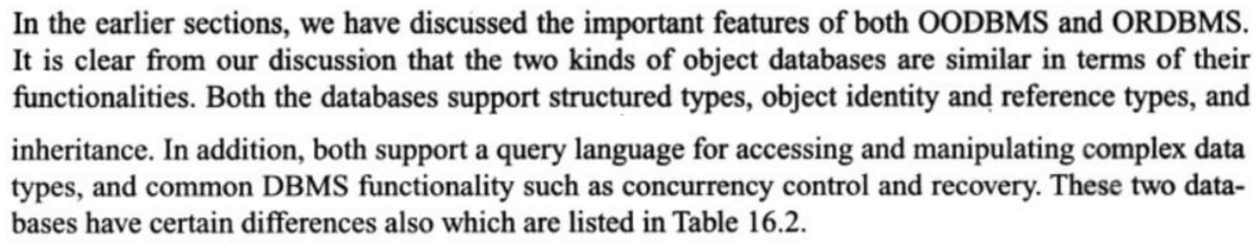
Complex elements are constructed from other elements hierarchically, whereas simple elements contain data values.

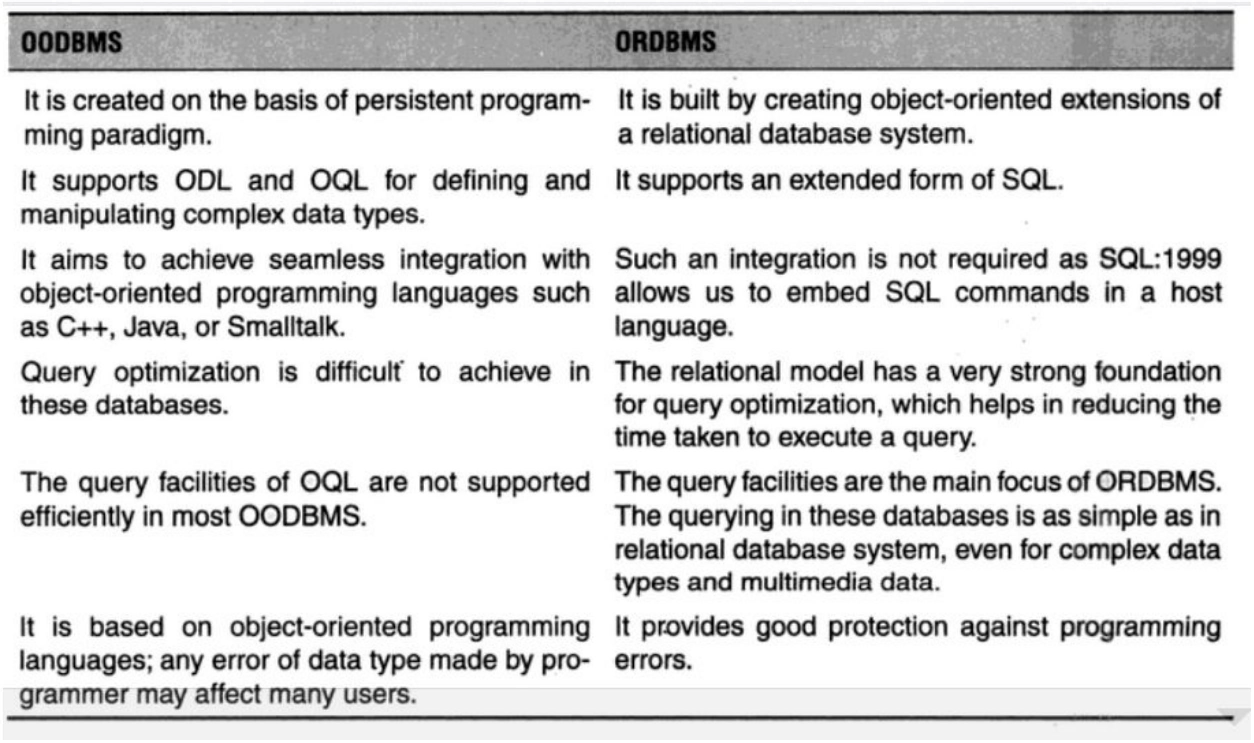
XML attributes are generally used in a manner similar to how they are used in HTML, namely, to describe properties and characteristics of the elements (tags) within which they appear. It is also possible to use XML attributes to hold the values of simple data elements; however, this is generally not recommended. An exception to this rule is in cases that need to reference another element in another part of the XML document. To do this, it is common to use attribute values in one element as the references. This resembles the concept of foreign keys in relational databases, and is a way to get around the strict hierarchical model that the XML tree model implies.

The main difference between attributes and elements is that attributes cannot be nested. Instead they are assigned string values only.Possible types of attributes are : ID, IDREF, CDATA, NMTOKEN.

In XML schema, the name attribute of the xsd:element tag specifies the element name.In XML schema, the attributes type, minOccurs, and maxOccurs in the xsd:element tag specify the type and multiplicity of each element in any document that conforms to the schema specifications. If we specify a type attribute in an xsd:element, the structure of the element must be described separately, typically using the xsd:complexType element of XML schema.

8.) Distinguish object oriented database and object relational databases. [Unit 3]


In the earlier sections, we have discussed the important features of both OODBMS and ORDBMS. It is clear from our discussion that the two kinds of object databases are similar in terms of their functionalities. Both the databases support structured types, object identity and reference types, and inheritance. In addition, both support a query language for accessing and manipulating complex data types, and common DBMS functionality such as concurrency control and recovery. These two databases have certain differences also which are listed in Table 16.2.



OODBMS	ORDBMS
It is created on the basis of persistent programming paradigm.	It is built by creating object-oriented extensions of a relational database system.
It supports ODL and OQL for defining and manipulating complex data types.	It supports an extended form of SQL.
It aims to achieve seamless integration with object-oriented programming languages such as C++, Java, or Smalltalk.	Such an integration is not required as SQL:1999 allows us to embed SQL commands in a host language.
Query optimization is difficult to achieve in these databases.	The relational model has a very strong foundation for query optimization, which helps in reducing the time taken to execute a query.
The query facilities of OQL are not supported efficiently in most OODBMS.	The query facilities are the main focus of ORDBMS. The querying in these databases is as simple as in relational database system, even for complex data types and multimedia data.
It is based on object-oriented programming languages; any error of data type made by programmer may affect many users.	It provides good protection against programming errors.

9.) What is a data warehouse? How does it differ from a database? [Unit 4]

A data warehouse is a subject-oriented, integrated, time-variant and non-volatile collection of data in support of management's decision making process:
Subject-Oriented: A data warehouse can be used to analyze a particular subject area. For example, "sales" can be a particular subject.
Integrated: A data warehouse integrates data from multiple data sources. For example, source A and source B may have different ways of identifying a product, but in a data warehouse, there will be only a single way of identifying a product.
Time-Variant: Historical data is kept in a data warehouse. For example, one can retrieve data from 3 months, 6 months, 12 months, or even older data from a data warehouse. This contrasts with a transactions system, where often only the most recent data is kept. For example, a transaction system may hold the most recent address of a customer, where a data warehouse can hold all addresses associated with a customer.
Non-volatile: Once data is in the data warehouse, it will not change. So, historical data in a data warehouse should never be altered.

We define database as a collection of related data and a database system as a database and database software together. A data warehouse is also a collection of information as well as a supporting system. However, a clear distinction exists in their structure, functioning, performance, and purpose :

- Traditional databases are transactional (relational, object-oriented, network, or hierarchical). Data warehouses have the distinguishing characteristic that they are mainly intended for decision-support applications. They are optimized for data retrieval, not routine transaction processing.

- Traditional databases support online transaction processing (OLTP), which includes insertions, updates, and deletions, while also supporting information query requirements.In contrast, Data warehouses provide access to data for complex analysis, knowledge discovery, and decision making. They support high-performance demands on an organization’s data and information.

- Traditional relational databases are optimized to process queries that may touch a small part of the database and transactions that deal with insertions or updates of a few tuples per relation to process. Thus, they cannot be optimized for OLAP (online analytical processing), DSS(decision-support systems), or data mining. By contrast, data warehouses are designed precisely to support efficient extraction, processing, and presentation for analytic and decision-making purposes.

- In comparison to traditional databases, data warehouses generally contain very large amounts of data from multiple sources that may include databases from different data models and sometimes files acquired from independent systems and platforms.

- Unlike most transactional databases, data warehouses typically support time-series and trend analysis, both of which require more historical data than is generally maintained in transactional databases.

- Compared with transactional databases, data warehouses are nonvolatile. This means that information in the data warehouse changes far less often and may be regarded as non–real-time with periodic updating.

- In transactional systems, transactions are the unit and are the agent of change to the database; by contrast, data warehouse information is much more coarse-grained and is refreshed according to a careful choice of refresh policy, usually incremental.

10.) Explain mobile computing architecture with suitable diagram. [Unit 3]

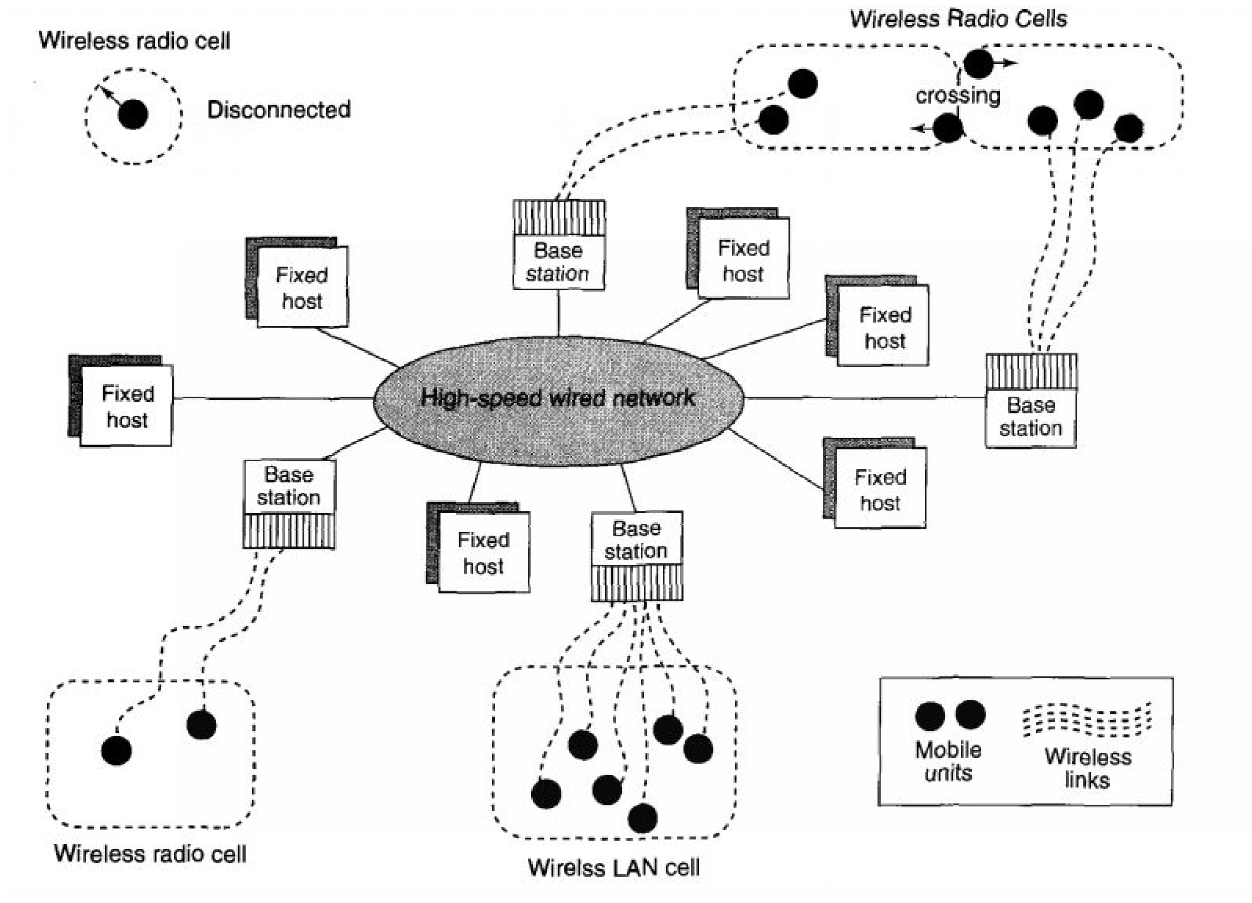


FIGURE 29.1 A general architecture of a mobile platform (Adapted from Dunham

The general architecture of a mobile platform is illustrated in Figure 29.1. It is a distributed architecture where a number of computers, generally referred to as Fixed Hosts and Base Stations, are interconnected through a high-speed wired network. Fixed hosts are general purpose computers that are not typically equipped to manage mobile units but can be configured to do so. Base stations function as gateways to the fixed network for the Mobile Units. They are equipped with wireless interfaces and offer network access services of which mobile units are clients.

Wireless Communications. The wireless medium on which mobile units and base stations communicate have bandwidths significantly lower than those of a wired network. The current generation of wireless technology has data rates that range from the tens to hundreds of kilobits per second (2G cellular telephony) to tens of megabits per second (wireless Ethernet, popularly known as WiFi). Modem (wired) Ethernet, by comparison, provides data rates on the order of hundreds of megabits per second.

Besides data rates, other characteristics also distinguish wireless connectivity options. Some of these characteristics include range, interference, locality of access, and support for packet switching. Some wireless access options allow seamless roaming thoughout a geographical region (e.g., cellular networks), whereas WiFi networks are localized around a base station. Some wireless networks, such as WiFi and Bluetooth, use unlicensed areas of the frequency spectrum, which may cause interference with other appliances, such as cordless telephones. Finally, modem wireless networks can transfer data in units called packets, that are commonly used in wired networks in order to conserve bandwidth. Wireless applications must consider these characteristics when choosing a communication option. For example, physical objects block infrared frequencies. While inconvenient for some applications, such blockage allows for secure wireless communications within a closed room.

Client/Network Relationships. Mobile units can move freely in a geographic mobility domain, an area that is circumscribed by wireless network coverage. To manage the mobility of units, the entire geographic mobility domain is divided into one or more smaller domains, called cells, each of which is supported by at least one base station. The mobile discipline requires that the movement of mobile units be unrestricted throughout the cells of a geographic mobility domain, while maintaining information access contiguity-Le., movement, especially intercell movement, does not negatively affect the data retrieval process.

2070

1.) Explain the following terms :
• Extent
• Temporal database
• Degree of homogeneity of DBMS
• X Path
• Classification and clustering
• OLAP
2.) Draw an ER Diagram for a hospital with a set of patients and set of doctors associated with each patient a log of various tests and examinations conducted. [Unit 2]
3.) What is the difference between an object and a..... in the object oriented data model (OOBM) ? [Unit 3]
See q7 2071

4.) What are the main difference between designing a relational database and an object database? [Unit 3]

One of the main differences between ODB and RDB design is how relationships are handled. In ODB, relationships are typically handled by having relationship properties or reference attributes that include OID(s) of the related objects. These can be considered as OID references to the related objects. Both single references and collections of references are allowed. References for a binary relationship can be declared in a single direction, or in both directions, depending on the types of access expected. If declared in both directions, they may be specified as inverses of one another, thus enforcing the ODB equivalent of the relational referential integrity constraint.

In RDB, relationships among tuples (records) are specified by attributes with matching values. These can be considered as value references and are specified via foreign keys, which are values of primary key attributes repeated in tuples of the referencing relation. These are limited to being single-valued in each record because multivalued attributes are not permitted in the basic relational model. Thus, M:N relationships must be represented not directly, but as a separate relation (table).

Mapping binary relationships that contain attributes is not straightforward in ODBs, since the designer must choose in which direction the attributes should be included. If the attributes are included in both directions, then redundancy in storage will exist and may lead to inconsistent data. Hence, it is sometimes preferable to use the relational approach of creating a separate table by creating a separate class to represent the relationship. This approach can also be used for n-ary relationships, with degree n > 2.

Another major area of difference between ODB and RDB design is how inheritance is handled. In ODB, these structures are built into the model, so the mapping is achieved by using the inheritance constructs, such as derived (:) and extends. In relational design,, there are several options to choose from since no built-in construct exists for inheritance in the basic relational model. It is important to note, though, that object-relational and extended-relational systems are adding features to model these constructs directly as well as to include operation specifications in abstract data types.

The third major difference is that in ODB design, it is necessary to specify the operations early on in the design since they are part of the class specifications. Although it is important to specify operations during the design phase for all types of databases, it may be delayed in RDB design as it is not strictly required until the implementation phase.

There is a philosophical difference between the relational model and the object model of data in terms of behavioral specification. The relational model does not mandate the database designers to predefine a set of valid behaviors or operations, whereas this is a tacit requirement in the object model. One of the claimed advantages of the relational model is the support of ad hoc queries and transactions, whereas these are against the principle of encapsulation.

In practice, it is becoming commonplace to have database design teams apply object-based methodologies at early stages of conceptual design so that both the structure and the use or operations of the data are considered, and a complete specification is developed during conceptual design. These specifications are then mapped into relational schemas, constraints, and behavioral artifacts such as triggers or stored procedures.

5.) Discuss some applications of active database. How do spatial databases differ from regular database? [Unit 3]

Potential applications of active rules:

<u>Allow notification of certain conditions that occur:</u> For example, an active database may be used to monitor, say, the temperature of an industrial furnace. The application can periodically insert in the database the temperature reading records directly from temperature sensors, and active rules can be written that are triggered whenever a temperature record is inserted, with a condition that checks if the temperature exceeds the danger level, and results in the action to raise an alarm.

<u>To enforce integrity constraints:</u> This can be done by specifying the types of events that may cause the constraints to be violated and then evaluating appropriate conditions that check whether the constraints are actually violated by the event or not. Hence, complex application constraints, often known as business rules, may be enforced that way. For example, in the UNIVERSITY database application, one rule may monitor the GPA of students whenever a new grade is entered, and it may alert the advisor if the GPA of a student falls below a certain threshold; another rule may check that course prerequisites are satisfied before allowing a student to enroll in a course; and so on.

<u>Automatic maintenance of derived data</u>, such as the examples of rules R1 through R4 that maintain the derived attribute Total_sal whenever individual employee tuples are changed. A similar application is to use active rules to maintain the consistency of materialized views whenever the base relations are modified. Alternately, an update operation specified on a view can be a triggering event, which can be converted to updates on the base relations by using an instead of trigger. These applications are also relevant to the new data warehousing technologies. A related application maintains that replicated tables are consistent by specifying rules that modify the replicas whenever the master table is modified.

Spatial databases incorporate functionality that provides support for databases that keep track of objects in a multidimensional space. For example, cartographic databases that store maps include two-dimensional spatial descriptions of their objects—from countries and states to rivers, cities, roads, seas, and so on.

A spatial database is optimized to store and query data related to objects in space, including points, lines and polygons. Satellite images are a prominent example of spatial data. Queries posed on these spatial data, where predicates for selection deal with spatial parameters, are called spatial queries. For example, “What are the names of all bookstores within five miles of the College of Computing building at Georgia Tech?” is a spatial query. Whereas typical databases process numeric and character data, additional functionality needs to be added for databases to process spatial data types. A query such as “List all the customers located within twenty miles of company headquarters” will require the processing of spatial data types typically outside the scope of standard relational algebra and may involve consulting an external geographic database that maps the company headquarters and each customer to a 2-D map based on their address. Effectively, each customer will be associated to a position. A traditional B+-tree index based on customers' zip codes or other nonspatial attributes cannot be used to process this query since traditional indexes are not capable of ordering multidimensional coordinate data. Therefore, there is a special need for databases tailored for handling spatial data and spatial queries

6.) Write a schema that provides tags for a person's first name, last name, weight, and shoe size. Weight and shoe size tags should have attributes to designate measuring systems. [Unit 5]

7.) Distinguish between structured and unstructured complex objects. [Unit 3]

An unstructured complex object facility provided by a DBMS permits the storage and retrieval of large objects that are needed by the database application.

Typical examples of such objects are bitmap images and long text strings (such as documents); they are also known as binary large objects, or BLOBs for short. Character strings are also known as character large objects, or CLOBs for short.

These objects are unstructured in the sense that the DBMS does not know what their structure is---only the application that uses them can interpret their meaning. For example, the application may have functions to display an image or to search for certain keywords in a long text string.

The objects are considered complex because they require a large area of storage and are not part of the standard data types provided by traditional DBMSs. Because the object size is quite large, a DBMS may retrieve a portion of the object and provide it to the application program before the whole object is retrieved. The DBMS may also use buffering and caching techniques to prefetch portions of the object before the application program needs to access them.

The DBMS software does not have the capability to directly process selection conditions and other operations based on values of these objects, unless the application provides the code to do the comparison operations needed for the selection. In an OODBMS, this can be accomplished by defining a new abstract data type for the uninterpreted objects and by providing the methods for selecting, comparing, and displaying such objects. For example, consider objects that are two-dimensional bitmap images. Suppose that the application needs to select from a collection of such objects only those that include a certain pattern. In this case, the user must provide the pattern recognition program as a method on objects of the bitmap type. The OODBMS then retrieves an object from the database and runs the method for pattern recognition on it to determine whether the object includes the required pattern.

A structured complex object differs from an unstructured complex object in that the object's structure is defined by repeated application of the type constructors provided by the OODBMS.

Hence, the object structure is defined and known to the OODBMS. As an example, consider the DEPARTMENT object shown. At the first level, the object has a tuple structure with six attributes: DNAME, DNUMBER, MGR, LOCATIONS, EMPLOYEES, and PROJECTS. However, only two of these attributes-namely, DNAME and DNUMBER-have basic values; the other four have complex structure and hence build the second level of the complex object structure. One of these four (MGR) has a tuple structure, and the other three (LOCATIONS, EMPLOYEES, PROJECTS) have set structures. At the third level, for a MGR tuple value, we have one basic attribute (MANAGERSTARTDATE) and one attribute (MANAGER) that refers to an employee object, which has a tuple structure. For a LOCATIONS set, we have a set of basic values, but for both the EMPLOYEES and the PROJECTS sets, we have sets of tuple-structured objects.

Two types of reference semantics exist between a complex object and its components at each level. The first type, which we can call ownership semantics, applies when the subobjects of a complex object are encapsulated within the complex object and are hence considered part of the complex object. The second type, which we can call reference semantics, applies when the components of the complex object are themselves independent objects but may be referenced from the complex object. For example, we may consider the DNAME, DNUMBER, MGR, and LOCATIONS attributes to be owned by a DEPARTMENT, whereas EMPLOYEES and PROJECTS are references because they reference independent objects. The first typeis also referred to as the is-part-of or is-component-ofrelationship; and the second type is called the is-associated-with relationship, since it describes an equal association between two independent objects. The is-part-of relationship (ownership semantics) for constructing complex objects has the property that the component objects are encapsulated within the complex object and are considered part of the internal object state. They need not have object identifiers and can only be accessed by methods of that object. They are deleted if the object itself is deleted. On the other hand, referenced components are considered as independent objects that can have their own identity and methods. When a complex object needs to access its referenced components, it must do so by invoking the appropriate methods of the components, since they are not encapsulated within the complex object. Hence, reference semantics represents relationships among independent objects. In addition, a referenced component object may be referenced by more than one complex object and hence is not automatically deleted when the complex object is deleted.

8.) What is data warehouse? List the characteristics of data warehouse. [Unit 4]

See q9 2069 for definition
Data warehouses have the following distinctive characteristics:
■ Multidimensional conceptual view
■ Generic dimensionality
■ Unlimited dimensions and aggregation levels
■ Unrestricted cross-dimensional operations
■ Dynamic sparse matrix handling
■ Client-server architecture

■ Multiuser support
■ Accessibility
■ Transparency
■ Intuitive data manipulation
■ Consistent reporting performance
■ Flexible reporting

9.) What are the advantages and disadvantages of extending the relational data model by means of ORDBMS? [Unit 3]

The main advantages of extending the relational data model come from **reuse and sharing**. Reuse comes from the ability to extend the DBMS server to perform standard functionality centrally, rather than have it coded in each application. For example, applications may require spatial data types that represent points, lines, and polygons, with associated functions that calculate the distance between two points, the distance between a point and a line, whether a point is contained within a polygon, and whether two polygonal regions overlap, among others. If we can embed this functionality in the server, it saves having to define it in each application that needs it, and consequently allows the functionality to be shared by all applications. These advantages also give rise to i**ncreased productivity** both for the developer and for the end-user.

Another obvious advantage is that the extended relational approach **preserves the significant body of knowledge and experience that has gone into developing relational applications**. This is a significant advantage, as many organizations would find it prohibitively expensive to change. If the new functionality is designed appropriately, this approach should allow organizations to take advantage of the new extensions in an evolutionary way without losing the benefits of current database features and functions.

The ORDBMS approach has the obvious disadvantages of

- Complexity and associated increased costs.

- There are the proponents of the relational approach who believe the essential simplicity and purity of the relational model are lost with these types of extension.

- ORDBMS vendors are attempting to portray object models as extensions to the relational model with some additional complexities. Object applications are not as data-centric as relational-based ones. Object-oriented models and programs deeply combine relationships and encapsulated objects to more closely mirror the ‘real world’. Such broad sets of relationships cannot be expressed well in SQL, and involves functional programs interspersed in the object definitions.

10.) Enumerate the limitations of conventional database compared to multimedia database. [Unit 3]

2071

1.) Explain the following terms :
a.) Data Warehouse
b.) Distribution Transparency
c.) X Query
d.) Distribution transaction
e.) Knowledge base
f.) Classification and clustering

2.) Distinguish multiple inheritance and selective inheritance in OO concepts. [Unit 3]

<u>Single</u> <u>Inteheritance</u> occurs when a certain type T is a subtype of only one supertype S and hence inherits its functions. This leads to the creation of a type hierarchy.

<u>Multiple</u> <u>inheritance</u> occurs when a certain subtype T is a subtype of two (or more) types and hence inherits the functions (attributes and methods) of both supertypes. For example, we may create a subtype ENGINEERING_MANAGER that is a subtype of both MANAGER and ENGINEER. This leads to the creation of a type lattice rather than a type hierarchy.

<u>Selective</u> <u>inheritance</u> occurs when a subtype inherits only some of the functions of a supertype. Other functions are not inherited. In this case, an EXCEPT clause may be used to list the functions in a supertype that are not to be inherited by the subtype.

One problem that can occur with multiple inheritance is that the supertypes from which the subtype inherits may have distinct functions of the same name, creating an ambiguity. For example, both MANAGER and ENGINEER may have a function called Salary. If the Salary function is implemented by different methods in the MANAGER and ENGINEER supertypes, an ambiguity exists as to which of the two is inherited by the subtype ENGINEERING_MANAGER. It is possible, however, that both ENGINEER and MANAGER inherit Salary from the same supertype (such as EMPLOYEE) higher up in the lattice. The general rule is that if a function is inherited from some common supertype, then it is inherited only once. In such a case, there is no ambiguity; the problem only arises if the functions are distinct in the two supertypes. There are several techniques for dealing with ambiguity in multiple inheritance.

One solution is to have the system check for ambiguity when the subtype is created, and to let the user explicitly choose which function is to be inherited at this time. A second solution is to use some system default. A third solution is to disallow multiple inheritance altogether if name ambiguity occurs, instead forcing the user to change the name of one of the functions in one of the supertypes. Indeed, some OO systems do not permit multiple inheritance at all.

3.) Define state of an object. Distinguish between persistent and transient objects. [Unit 3]

The state of an object is it’s value. The state (current value) of a complex object may be constructed from other objects (or other values) by using certain type constructors.

The atom constructor is used to represent all basic atomic values, such as integers, real numbers, character strings, Booleans, and any other basic data types that the system supports directly.

The tuple constructor can create structured values and objects of the form $<a_1:i_1, a_2:i_2, \ldots, a_n:i_n>$, where each a_j is an attribute name and each i_j is a value or an OID.

The other commonly used constructors are collectively referred to as collection types but have individual differences among them. The set constructor will create objects or literals that are a set of distinct elements $\{i_1, i_2, \ldots, i_n\}$, all of the same type. The bag constructor (also called a multiset) is similar to a set except that the elements in a bag need not be distinct. The list constructor will create an ordered list $[i_1, i_2, \ldots, i_n]$ of OIDs or values of the same type. A list is similar to a bag except that the elements in a list are ordered, and hence we can refer to the first, second, or jth element. The array constructor creates a single-dimensional array of elements of the same type. The main difference between array and list is that a list can have an arbitrary number of elements whereas an array typically has a maximum size. Finally, the dictionary constructor creates a collection of key-value pairs (K, V), where the value of a key K can be used to retrieve the corresponding value V. The main characteristic of a collection type is that its objects or values will be a collection of objects or values of the same type that may be unordered (such as a set or a bag) or ordered (such as a list or an array).

Transient objects exist in the executing program and disappear once the program terminates. Persistent objects are stored in the database and persist after program termination

The typical mechanisms for making an object persistent are naming and reachability.The naming mechanism involves giving an object a unique persistent name within a particular database. The named persistent objects are used as entry points to the database through which users and applications can start their database access. The reachability mechanism works by making the object reachable from some other persistent object.

4.) Discuss how time is represented in temporal databases and compare the different time dimensions. [Unit 3]

For temporal databases, time is considered to be an ordered sequence of points in some granularity that is determined by the application. For example, suppose that some temporal application never requires time units that are less than one second. Then, each time point represents one second using this granularity.

Because there is no known beginning or ending of time, one needs a reference point from which to measure specific time points. A calendar organizes time into different time units for convenience. Most calendars group 60 seconds into a minute, 60 minutes into an hour, 24 hours into a day (based on the physical time of earth's rotation around its axis), 7 days into a week and so on.

A temporal database will store information concerning when certain events occur, or when certain facts are considered to be true. There are several different types of temporal information.

- Point events or facts are typically associated in the database with a single time point in some granularity. For example, a bank deposit event may be associated with the timestamp when the deposit was made.

- Duration events or facts, on the other hand, are associated with a specific time period in the database.For example, an employee may have worked in a company from August 15, 2003 until November 20, 2008.

A time period is represented by its start and end time points [START-TIME, ENDTIME]. For example, the above period is represented as [2003-08-15, 2008-11-20]. Such a time period is often interpreted to mean the set of all time points from starttime to end-time, inclusive, in the specified granularity. Hence, assuming day granularity, the period [2003-08-15, 2008-11-20] represents the set of all days from August 15, 2003, until November 20, 2008, inclusive.

Valid Time and Transaction Time Dimensions
Given a particular event or fact that is associated with a particular time point or time period in the database, the association may be interpreted to mean different things. The most natural interpretation is that the associated time is the time that the event occurred, or the period during which the fact was considered to be true in the real world. If this interpretation is used, the associated time is often referred to as the valid time. A temporal database using this interpretation is called a valid time database.

However, a different interpretation can be used, where the associated time refers to the time when the information was actually stored in the database; that is, it is the value of the system time clock when the information is valid in the system. In this case, the associated time is called the transaction time. A temporal database using this interpretation is called a transaction time database.

Other interpretations can also be intended, but these are considered to be the most common ones, and they are referred to as time dimensions. In some applications, only one of the dimensions is needed and in other cases both time dimensions are required, in which case the temporal database is called a bitemporal database.

5.) What is the difference between structured and unstructured complex object? Differentiate identical versus equal objects with examples. [Unit 3]
6.) What are the advantages and disadvantages of OODBMS? [Unit 3]

7.) What are the differences and similarities between objects and literals in the ODMG object Model? [Unit 3]

Objects and literals are the basic building blocks of the object model. The main difference between the two is that an object has both an object identifier and a state (or current value), whereas a literal has a value (state) but no object identifier. In either case, the value can have a complex structure.

The object state can change over time by modifying the object value.
A literal is basically a constant value, possibly having a complex structure, but it does not change.

Every object must have an immutable OID, whereas a literal value has no OID and its value just stands for itself. Thus, a literal value is typically stored within an object and cannot be referenced from other objects. In many systems, complex structured literal values can also be created without having a corresponding OID if needed.

An object has five aspects: identifier, name, lifetime, structure, and creation.
1. The object identifier is a unique system-wide identifier (or Object_id).Every object must have an object identifier.
2. Some objects may optionally be given a unique name within a particular ODMS—this name can be used to locate the object, and the system should return the object given that name. Obviously, not all individual objects will have unique names. Typically, a few objects, mainly those that hold collections of objects of a particular object class/type—such as extents—will have a name. These names are used as entry points to the database; that is, by locating these objects by their unique name, the user can then locate other objects that are referenced from these objects. Other important objects in the application may also have unique names, and it is possible to give more than one name to an object. All names within a particular ODB must be unique.
3. The lifetime of an object specifies whether it is a persistent object (that is, a database object) or transient object (that is, an object in an executing program that disappears after the program

terminates). Lifetimes are independent of classes/types—that is, some objects of a particular class may be transient whereas others may be persistent.
4. The structure of an object specifies how the object is constructed by using the type constructors:Atomic, Structured and Collection. All values of the basic built-in data types are considered to be literals.
5. Object creation refers to the manner in which an object can be created. This is typically accomplished via an operation new for a special Object_Factory interface. We shall describe this in more detail later in this section.

In the object model, a literal is a value that does not have an object identifier. However, the value may have a simple or complex structure. There are three types of literals: atomic, structured, and collection.

8.) Describe the main reasons for the potential advantage for distributed database. What additional functions does it have over centralized DBMS? [Unit 1]
Advantages • Potential for parallel execution • More resources available to process queries faster • Potential for replicated data • Processing closer to users' locations • Reduced communications time • Duplication in case of failures

9.) Describe the characteristics of mobile computing environment in detail. [Unit 3]
The characteristics of mobile computing include

High communication latency: Latency is caused by the processes unique to the wireless medium, such as coding data for wireless transfer, and tracking and filtering wireless signals at the receiver.

Intermittent wireless connectivity:Intermittent connectivity can be intentional or unintentional. Unintentional disconnections happen in areas wireless signals cannot reach, e.g., elevator shafts or subway tunnels. Intentional disconnections occur by user intent, e.g., during an airplane takeoff, or when the mobile device is powered down.

Limited battery life: Battery life is directly related to battery size, and indirectly related to the mobile device's capabilities.

Changing client location: Finally, clients are expected to move, which alters the network topology and may cause their data requirements to change. All of these characteristics impact data management, and robust mobile applications must consider them in their design

10.) Differentiate between XML schema and XML DTD with suitable example . [Unit 5]
http://www.differencebetween.net/technology/difference-between-xml-schema-and-dtd/

2072
1. Explain the following terms:

• Database performance tuning
• UML
• Subclass vs superclass
• Xquery
• Calendars
Because there is no known beginning or ending of time, one needs a reference point from which to measure specific time points. A calendar organizes time into different time units for convenience. Most calendars group 60 seconds into a minute, 60 minutes into an hour, 24 hours into a day (based on the physical time of earth's rotation around its axis), 7 days into a week and so on.
• Active database
2. What are query optimization techniques? Explain.
The DBMS must then devise an execution strategy or query plan for retrieving the results of the query from the database files. A query typically has many possible execution strategies, and the process of choosing a suitable one for processing a query is known as query optimization.

There are two main techniques that are employed during query optimization. The first technique is based on heuristic rules for ordering the operations in a query execution strategy. A heuristic is a rule that works well in most cases but is not guaranteed to work well in every case. The rules typically reorder the operations in a query tree. The second technique involves systematically estimating the cost of different execution strategies and choosing the execution plan with the lowest cost estimate. These techniques are usually combined in a query optimizer.

A different approach to query optimization, called semantic query optimization, has been suggested. This technique, which may be used in combination with the techniques discussed previously, uses constraints specified on the database schema—such as unique attributes and other more complex constraints—in order to modify one query into another query that is more efficient to execute. We will not discuss this approach in detail but we will illustrate it with a simple

3. Differentiate between specialization and generalization with example.
https://books.google.com.np/books?id=y7P9sa2MeGIC&lpg=PA60&dq=difference%20between%20specialization%20and%20generalization%20with%20example&pg=PA45#v=onepage&q=difference%20between%20specialization%20and%20generalization%20with%20example&f=false
4. How do single inheritance, multiple inheritance and selective inheritance differ?
See q2 2071
5. What are the differences between structured and unstructured complex objects? Explain.
See q7 2070
6. What are the object relational features that have been included in SQL-99?
The following are some of the features that have been included in SQL-99:
A.

Some type constructors have been added to specify complex objects. These include the row type, which corresponds to the tuple (or struct) constructor. An array type for specifying collections is also provided. Other collection type constructors, such as set, list, and bag constructors, are not yet part of the SQL-99 specifications, although some systems include them and they are expected to be in future versions of the standard.

B.
A mechanism for specifying object identity through the use of reference type is included. A user defined type can be used either as type for an attribute, as illustrated by the addr attribute of Emp_type, or it can be used to specify the row types of tables. For example,

```
CREATE TABLE Employee OF Emp_type REF IS emp_id SYSTEM GENERATED;
```

The above examples also illustrate how the user can specify that system-generated object identifiers for the individual rows in a table should be created. By using the syntax:

```
REF IS <oid_attribute> <value_generation_method> ;
```

the user declares that the attribute named<oid_attribute> will be used to identify individual tuples in the table. The options for <value_generation_method> are SYSTEM GENERATED or DERIVED. In the former case, the system will automatically generate a unique identifier for each tuple. In the latter case, the traditional method of using the user-provided primary key value to identify tuples is applied.

C.
Encapsulation of operations is provided through the mechanism of user-defined types that may include operations as part of their declaration.
In SQL a construct similar to class definition is provided whereby the user can create a named user-defined type with its own behavioral specification by specifying methods (or operations) in addition to the attributes.We can specify a method for Addr_type as follows:

```
CREATE TYPE Addr_type AS (
street VARCHAR (45),
city VARCHAR (25),
zip CHAR (5)
)
METHOD apt_no() RETURNS CHAR (8);
```

D.
Inheritance mechanisms are provided.

- ❖ All attributes are inherited.
- ❖ The order of supertypes in the UNDER clause determines the inheritance hierarchy.
- ❖ An instance of a subtype can be used in every context in which a supertype instance is used.
- ❖ A subtype can redefine any function that is defined in its supertype, with the restriction that the signature be the same.

- ❖ When a function is called, the best match is selected based on the types of all arguments.
- ❖ For dynamic linking, the runtime types of parameters is considered
- ❖ Consider the following example to illustrate type inheritance. Suppose that we want to create a subtype Manager_type that inherits all the attributes (and methods) of Emp_ type but has an additional attribute dept_managed. Then we can write:

```
CREATE TYPE Manager_type UNDER Emp_type AS (
dept_managed CHAR (20)
) ;
```

7. Discuss how time is represented in temporal database and compare different time dimensions.

See q4 2071

8.What are the difference and similarities between objects and literals in the ODMG Object Model?

See q7 2071

9.Describe multimedia database and what are the different types of multimedia data that are available in current systems?

Multimedia databases provide features that allow users to store and query different types of multimedia information, which includes images (such as photos or drawings), video clips (such as movies, newsreels, or home videos), audio clips (such as songs, phone messages, or speeches), and documents (such as books or articles). The main types of database queries that are needed involve locating multimedia sources that contain certain objects of interest. For example, one may want to locate all video clips in a video database that include a certain person, say Michael Jackson. Such a query is referred to as content-based retrieval. There are two approaches to this: automatic analysis and manual identification.

Today the following types of multimedia data are available in current systems:

• Text: May be formatted or unformatted. For ease of parsing structured documents, standards like SGML and variations such as HTML are being used.

• Graphics: Examples include drawings and illustrations that are encoded using some descriptive standards (e.g., COM, PICT, postscript}.

• Images: Includes drawings, photographs, and so forth, encoded in standard formats such as bitmap, JPEG, and MPEG. Compression is built into JPEG and MPEG. These images are not subdivided into components. Hence querying them by content (e.g., find all images containing circles) is nontrivial.

• Animations: Temporal sequences of image or graphic data.

• Video: A set of temporally sequenced photographic data for presentation at specified rates-for example, 30 frames per second.

• Structured audio: A sequence of audio components comprising note, tone, duration, and so forth.

• Audio: Sample data generated from aural recordings in a string of bits in digitized form. Analog recordings are typically converted into digital form before storage.

• Composite or mixed multimedia data: A combination of multimedia data types such as audio and video which may be physically mixed to yield a new storage format or logically mixed while retaining original types and formats. Composite data also contains additional control information describing how the information should be rendered.

10. Explain XML schema and XML DTD.