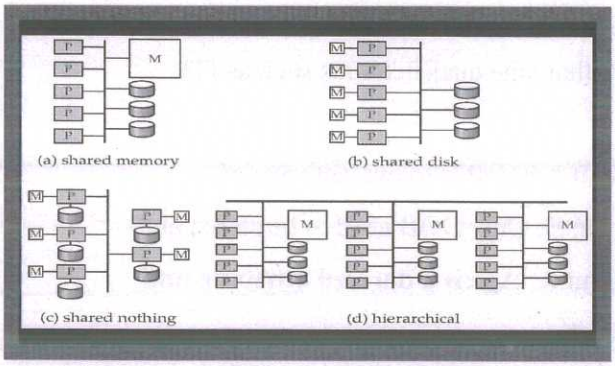
SCHEME OF VALUATION

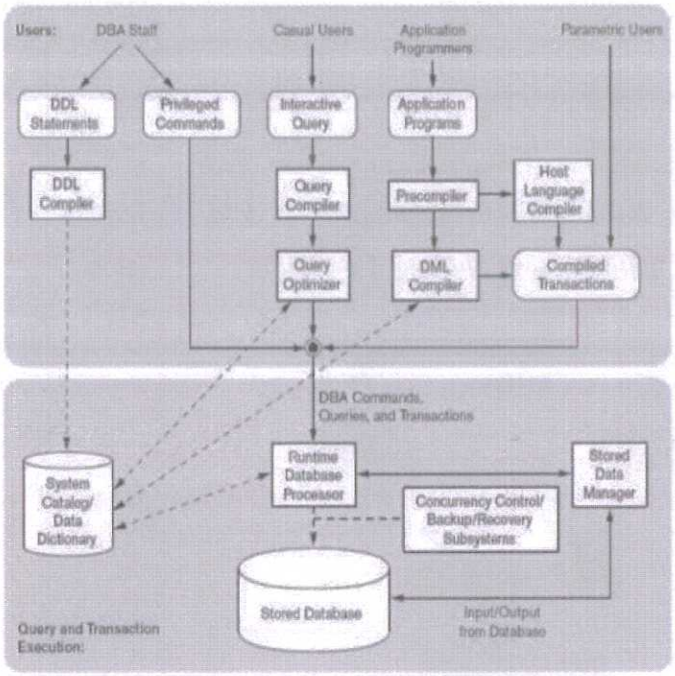
(Scoring Indicators)

Revision: 2015Course Code:3132 **Course Title :Database Management System**				
Qst. No.	**Scoring Indicator**	**Split up score**	**Sub Total**	**Total**
	PART A			
1	• DBMS is a collection of programs that allows to **define , construct, manipulate & share** databases among various users and applications (Give 1 mark for example, if written)			2
2	• Schema refers to the structure (definition) of the database which may not change frequently • Instance is the data in the database at a particular moment of time. (Marks may be given for correct examples also)	1+1		2
3.	• Degree of a relation is the number of columns or attributes in the relation (Write 1 example)	1+1		2
4.	• Triggers are procedural codes that executes automatically during database events such as INSERT, DELETE and UPDATE. • If schema constraints such as PRIMARY KEY, FOREIGN KEY, UNIQUE, NULL, NOT NULL, DEFAULT, CHECK etc. are not capable to implement some business rules, then triggers can be defined to implement them. • It is mainly used to maintain the integrity of the data in a database. (Any one point is enough)	2		2
5.	• A functional dependency, denoted by **X -> Y** , between two sets of attributes X and Y that are subsets of the relation schema $R(A_1, A_2, \ldots, A_n)$ specifies a constraint on the tuples of relation state **r** of **R**. The constraint is that, for any two tuples t_1 and t_2 in **r**, **if $t_1[X] = t_2[X]$, then $t_1[Y]=t_2[Y]$.** (Writing correct example is enough)	2		2

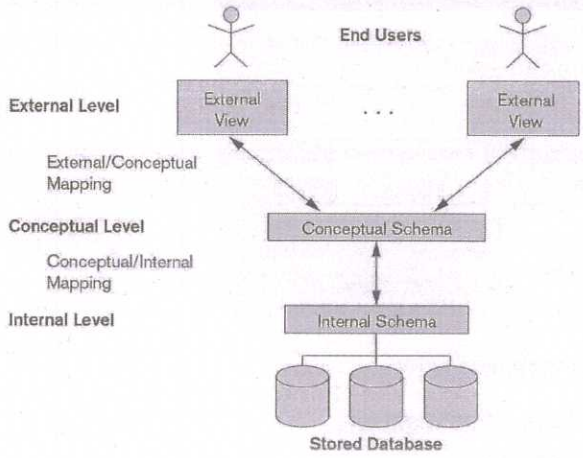
II	PART -B			
1.	1. Controlling redundancy 2. Restricting unauthorized access 3. Providing persistent storage for program objects 4. Enforcing integrity constraints 5. Providing Storage structures and search techniques for efficient query processing 6. Caching or Buffering 7. Providing Backup & Recovery 8. Providing Multiple user interfaces 9. Permitting inference and actions using rules 10. Additional implications of using the database approach such as Enforce standards, Reduced application development time, Flexibility to implement, , Increase availability, Save Cost (Explain any four)	4 x 1.5		6
2.	**SELECT (σ)** is a unary relational algebra operation used to choose a subset of tuples from a relation that satisfies the specified selection criteria. **Syntax** σ < selection-criteria> (R) Example : σ dept=4 (EMPLOYEE Degree of the relation resulting from a SELECT operation is same as the degree of R **PROJECT (π)** is another unary operation which selects the specified columns or attributes from a relation **Syntax** π < attribute-list> (R) Example : $\pi_{name, gender}$(EMPNAME) The result set of PROJECT operation is a set of distinct tuples eliminating duplicates. Degree of the result set is equal to the number of attributes in <attribute-list>	3+3		6
3.	Consider the following attributes related to a student entity in a college : 1. **Student_name** - It can be considered as an example for **composite attribute** because it can further be divided to sub	4 * 1.5		6

	components such as first_name , mid_initials, last_name 2. **Qualification** – It may be a **multi-valued** attribute for students having more than one qualifications such as ITI, BSc, Diploma etc. 3. **Age & DOB** - Here, it is possible to evaluate age of a student from his/her DOB. Only DOB need to be stored and age can be derived from it. **Age is a derived attribute and DOB is stored attribute**			
4.	NOT NULL DEFAULT CHECK PRIMARY KEY UNIQUE FOREIGN KEY (explain any 4 briefly with 1 example for each)	4 * 1.5		6
5.	***1) INNER JOIN (JOIN) :*** This is the default type of JOIN. The keyword INNER is optional. In inner join operation, a tuple is included in the result only if a matching or related tuple exists in the other relation **OUTER JOIN :** Outer joins are used when the user wants to keep all the tuples in **table1**, or all those in **table2**, or all tuples in both relations in the result of JOIN, regardless of whether or not they have matchingtuples in the other relation. There are three types of OUTER JOIN : **2) LEFT OUTER JOIN :**Here every tuple from the left table must appear in the result. If it does not have amatching tuple, it is padded with NULL values for the attributes in the right table **3) RIGHT OUTER JOIN :**Every tuple from the right table must appear in the result. **4) FULL OUTER JOIN :** All tuples from both tables will appear and NULL values will be filled forattributes of other table which does not have matching tuple (Write example also)	Explanation : 3 Example : 3		6
6.	1. **Data distribution and replication issues** 2. **New transaction model is required** 3. **Query processing issues** 4. **Recovery and fault tolerance issues** 5. **Design issues** 6. **Division of labour issues** 7. **Location-based service** 8. **Security issues (Explain any four)**	4 * 1.5		6

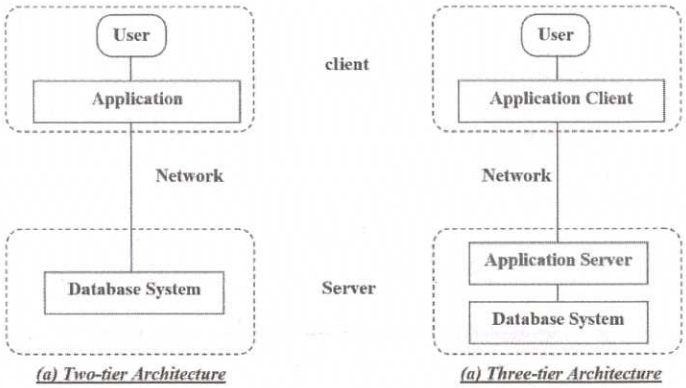
7.	 ***Shared Memory (Tightly coupled)*** All the processors share a common memory, typically via a bus or through an interconnection network. A processor can send messages to other processors much faster by using memory writes than by sending a message through a communication mechanism. This architecture is not scalable beyond 32 or 64 processors because the bus or interconnection network becomes a bottleneck **<u>Shared Disk (loosely coupled)</u>** All the processors share a common set of disks directly via an interconnection network, but the processors have private memories. Since each processor has its own memory, the memory bus is not a bottle neck It offers a cheap way to provide a degree of fault Limitation with scalability. Here the interconnection of disk sub-system is a bottleneck. **Shared Nothing (loosely coupled)** More Scalable , Easily supports a large number of processors High cost of communication and non-local disk access. **Hierarchical** This model is a hybrid of the preceding three architectures. It combines the characteristics of shared-memory, shared-disk, and shared-nothing architectures.	Diagram : 3 Explanation : 3		

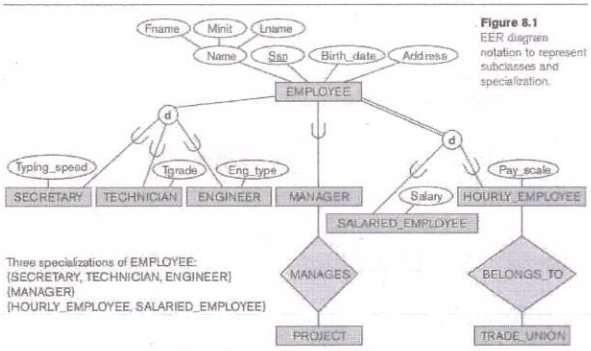
III a)	PART C - Unit I **DBMS COMPONENT MODULES**  Brief explanation of each COMPONENTS 1)USERS (DBA , Casual Users , Application Programmers , Parametric Users) 2)DDL Compiler 3)Query Compiler , Query Optimizer 4)Pre Compiler , Host language Compiler , DML Compiler to support Embedded SQL 5)Runtime Database Processor 6)Storage Manager Components (Authorization and Integrity manager , Transaction Manager , File Manager , Buffer Manager) 7)Database Catalog / Data Dictionary (Meta data) 8)Stored Database	Diagram : 3 Explanation : 6		9

III b)	PART C - Unit I **Sales, Accounting, Human resources, Manufacturing, Online retailers , Banking and Finance, Banking, Credit card** transactions, **Finance, Universities, Airlines, Telecommunication** , 6) **Scientific applications, Storage and retrieval of images,Storage and retrieval of videos,Data Mining , Spatial, Time series, ERP systems, CRM systems, Information retrieval (IR)** which deals with books,manuscripts, and various forms of library-based articles. (Any six)	6 x 1		6

IV a)	PART C - Unit I Three schema architecture is proposed to achieve and visualise the characteristics of database approach as :*Self describing nature , Insulation between programs & data, data abstraction, multiple views, sharing & multiuser transactions.*  Explain Internal level, Conceptual level, External level and mapping between each schema level	Diagram : 3 Explanation : 5		8

IV b)	PART C - Unit I ***Two-tier Architecture*** Here, **Server handles q**uery and transaction functionality related to SQL processing **Client handles u**ser interface programs and application programs When a client program or command requires a database access, the program establishes a connection to DBMS using APIs (Application Programming Interface) like ODBC, JDBC etc. This architecture is called two tier architecture because the software components are distributed over two machines, **client & server**. ***Three-Tier Architectures for Web Applications*** In 3-tier architecture, there is an intermediate layer between the client and the database server as shown in fig. The intermediate layer or middle tier is called the **application server** or the **web server .Application server** handles intermediate rules andconstraints before data is	Diagram : 3 Explanation : 4		7

	passed up to the user or down to the DBMS. Database server performsdata management services  *(a) Two-tier Architecture* *(a) Three-tier Architecture*			

	PART C - Unit II			
V a)	Specialization is the process of defining a set of subclasses of an entity type. The entity type is called the **superclass** of specialization. Specialization is based on some distinguishing characteristics of the entities in the superclass.  **Figure 8.1** EER diagram notation to represent subclasses and specialization. Generalization is the reverse process of abstraction in which we identify the common features of several entity types and generalize them into a single super class of which the original entity types are special subclasses.	3 + 3 + 2		8

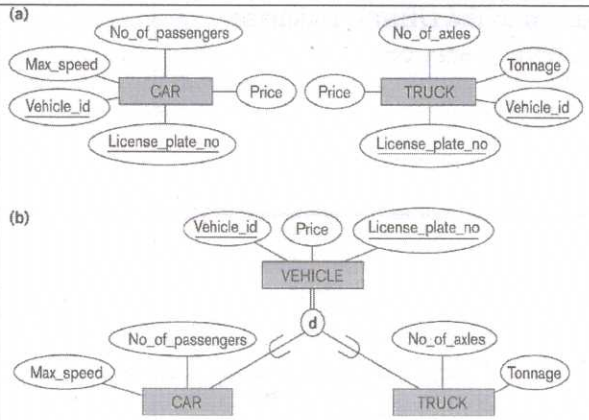


Figure 4.3
Generalization. (a) Two entity types, CAR and TRUCK.
(b) Generalizing CAR and TRUCK into the superclass VEHICLE.

Also, explain the concepts : Disjoint Vs. Overlapping & Partial Vs. Total

PART C - Unit II

V b)

BOOK (BookId – Primary key)			
BookId	Author	Title	Price

MEMBER (Member_Id – Primary Key)			
Member_Id	M_Name	M_Type	M_Date

ISSUE (bookid & memberid – composite key) Bookid & member id are foreign keys)				
BookId	Member_Id	Issue_date	Due_date	Ret_Date

MEMBER_CONTACT Memberid & contact together forms primary key Memberid is the foreign key	
Member_Id	Contact

It is also possible to design a single table for BOOK & ISSUE (1:N relationship)

Entity Mapping – 3

Relationship Mapping – 2

Multivalued attribute - 2

3+2+2

7

VI a)	PART C - Unit II **Candidate Key:** • The minimal set of attribute which can uniquely identify a tuple (row or record) is known as candidate key. • It is the minimal Super Key. **Super Key:** • The set of attributes which can uniquely identify a tuple (row or record) is known as Super Key. • Adding zero or more attributes to candidate key generates super key.A candidate key is a super key but vice versa is not true. **Primary Key:** • There can be more than one candidate key in a relation out of which one can be chosen as primary key. • The entity integrity constraint says that the value of primary key can never be NULL. **Alternate Key:** • The candidate key other than primary key is called as alternate key **Composite Key** • A ***composite key*** is a candidate key with a combination of two or more attributes. • The candidate key can be simple (having only one attribute) or composite as well. **Foreign Keys** • Foreign Keys are constraints used to specify the referential integrity between two relations • Foreign keys help to maintain the consistency of data among the tuples in both relations. • Foreign keys are a set of one or more attributes in a relation schema R1 which references the primary key of relation schema R2 while inserting or updating its values. (Explain with an example)	6 * 1.5		9

VI b)	PART C - Unit II • A relationship type **R** among **n** entity types E1, E2, ..., En defines a set of associations among entities from these entity types. • A relationship set **R** is a collection of relationship instances that exists between the entities in the participation entity types. Explain any three from following with examples : 1) Binary relationships (may be 1:1 , 1:N , M:N) 2) Ternary relationships (Degree 3)	3 x 2		6

	3) Recursive relationships (Participants from same entity) 4) Identifying relationships (between weak entity & its strong owner entity)			

VII a)	PART C - Unit III Aggregate functions are used to summarise information from multiple tuples in a table by grouping them basedon values of one or more attributes. The following aggregate functions can be used with SQL :**COUNT , SUM, MAX, MIN** and **AVG**. 1. **COUNT(*/fieldname)** : - returns the number of tuples as specified in a query 2. **SUM** (numeric field) : returns the sum of the numeric field as specified in the query 3. **AVG(numeric field)** : evaluates and returns the average value of specified attribute for the set of tuplesspecified in the query 4. **MAX(numeric field)** : returns the maximum or highest value of the specified attribute 5. **MIN(numeric field)** : returns the minimum or lowest value of the specified attribute Aggregate functions are specified in SELECT statement with optional GROUP BY & HAVING clause as shown in following examples. (Write examples for each with GROUP BY and HAVING clause) Listing – 2 Explanation – 3 Example - 3	2+3+3		8

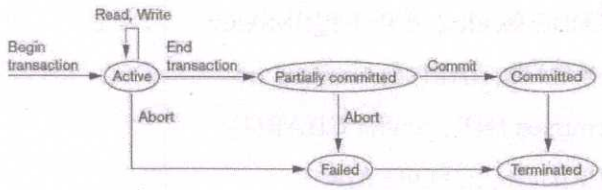
VII b)	PART C - Unit III Cursors are temporary place holder for storing the data set (read only, not updateable) retrieved by SELECT query	Explanation : 4 Example : 3		7

	Steps to use the cursors in stored procedure: 1. Declare Cursor (Specify cursor name & select query) 2. Open cursor 3. Fetch data from cursor (one record at a time , advance to next record after fetching) 4. Close cursor (Write a sample procedure which uses cursor)			

	PART C - Unit III			

VIII a)	a) CREATE TABLE faculty(id INT PRIMARY KEY , f_name VARCHAR(15), contact_no CHAR(10), coursed INT , gender CHAR(1), salary NUMERIC(10,2), FOREIGN KEY(courseid) REFERENCES courses(id));
	CREATE TABLE courses(id INT PRIMARY KEY, coursename VARCHAR(20), duration NUMERIC(2));
	b) INSERT INTO courses VALUES (10, 'C-Programming',3),(11,'Java Programming',4);
	INSERT INTO faculty VALUES(2001, 'Merin','1234567890',10,'F',10000);
	c) SELECT * FROM faculty WHERE gender='F' AND salary>20000;
	d) SELECT id, name, salary FROM faculty WHERE salary=(SELECT MAX(salary) FROM faculty);
	e) UPDATE faculty SET salary=salary+salary*5/100 WHERE courseid=10;
		5 x 2		10

	PART C - Unit III			
VIII b)	A transaction is a logical unit of database processing	Diagram :		5

	that includes one or more access operations (read - retrieval, write - insert or update, delete). A **transaction** is an atomic unit of work that should either be completed in its entirety or not done at all **Transaction states** : (Explain briefly) • Active state • Partially committed state • Committed state • Failed state • Terminated State  	2 Explain : 3		

IX a)	PART C - Unit IV Data mining refers to the mining or discovery of new information from vast amounts of data. Datamining is a part of knowledge discovery process. It is the process of extraction or discovery of newknowledge or information from existing huge databases **Goals of data mining technology**. **1) Prediction**: - Data mining can predict how certain attributes within the data will behave in thefuture. volume of stores during certain time periods etc. **2) Identification:** - Data patterns can be used to identify the existence of an item, an event, or anactivity. **3) Classification: -** Data mining can partition the data so that different classes or categories can beidentified based on combinations of parameters. **4) Optimization**: - Another goal of data mining is to optimize the use of limited resources such astime, space, money or materials to maximize output variables such as sales or profits under agiven set of			

	constraints.			

	PART C - Unit IV			
IX b)	Conditions for 3NF: 1) Relation must be in 2NF 2) There should not be any transitive dependency from primary key to any non-prime attributes **Regarding given table :** Already in 1 NF – all attributes are atomic Already 2 NF – No partial dependency (Primary key is atomic) not in 3NF because, there exists a transitive dependency { CUST_ID ->POSTAL_CODE} {POSTAL_CODE -> CITY} So decompose as : 1) CUSTOMER(CUST_ID,CUST_NAME,POSTAL_CODE) 2) CITY(POSTAL_CODE, CITY)	Conditions : 3 Testing & Decomposition : 5		8

	PART C - Unit IV			
X a)	**Transparency in distributed databases** Transparency generally refers to the idea of hiding implementation details from end users. In DDB, the following types of transparencies are possible : **Location transparency** : Refers to the fact that the command used to perform a task is independent of the location of the data and location of the node where the command is issued **Naming transparency** : Once a database object is named, it can be accessed from any site without additional specification	5 + 2		7

	Replication transparency: Replication transparency makes the user unaware of the existence of these copies. **Fragmentation transparency:**Fragmentation transparency makes the user unaware of the existence of fragments. Two types : **Horizontal fragmentation&Vertical fragmentation** **Design transparency & Execution transparency** : It refers to freedom from knowing how the DDB is designed and where a transaction executes ■ **Autonomy** – each site is able to retain a degree of control over data stored locally. In a distributed system, there is a global database administrator responsible for the entire system. A part of these responsibilities will be given to the local database administrator for each site. Thus the possibility of local autonomy is a major advantage.			

X b)	PART C - Unit IV • Decomposition is the procedure for normalization. • If the relation schema R(A1, A2, A3, …., An) does not satisfy a particular normal form, wedecompose R into smaller schemas. • Decomposition of R may be D = (R1, R2, ….Rk) , where R1, R2, …, Rk are smaller schemas whereeach Ri is a subset of R Example :- Relation R: StudentGrades(RollNo, StudName, Course, Grade) can be decomposed as : R1 : StudInfo(RollNo, StudName) R2 : GradeInfo(RollNo, Course, Grade) Desirable properties of Decomposition 1. **Attribute Preservation** :Each attribute in R must appear in at least one relational schema Ri in the decomposition so that no attributes are lost. The above	Concept : 2 Properties : 3 * 2		8

	example preserves the attributes ,no attributes are lost after decomposition 2. **Dependency preservation** :Each functional dependency exist in R must be distributed among the decomposed relations (R1,R2, ….Rk) so that no FDs are lost. In our example, Functional Dependency in R is : ROLLNO -> { STUDNAME , COURSE, GRADE } After decomposition, FD in R1 is : ROLLNO -> STUDNAME After decomposition, FD in R2 is : ROLLNO ->{ COURSE , GRADE } , So no dependencies are lost. 3. **Non-additive or lossless join** :It ensures that spurious tuples are not generated when a natural join operation is applied on therelations in the decomposition. It should also make sure that the information in an instance r of Rmust be preserved in the instances r1, r2, ….rk In our example, if we natural join R1 and R2 based on common field RollNo, spurious tupleswill not be generated. So it obeys non-additive or lossless join property.			