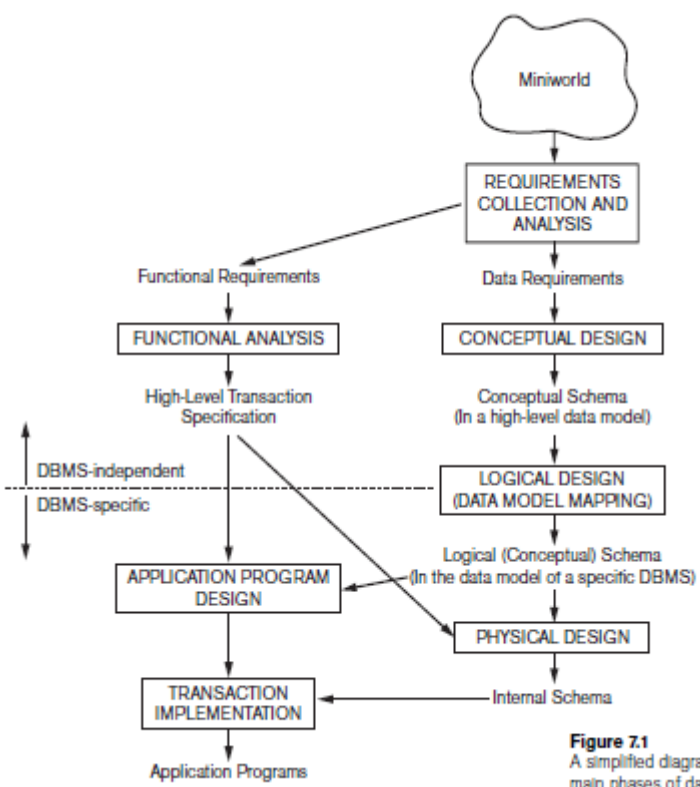


Improvement Test – November 2016

Sub:	Data Base Management Systems						Code:	10CS54	
Date:	16/11/2016	Duration:	90 mins	Max Marks:	50	Sem:	V	Branch:	ISE

Note: Answer any five full questions		Marks	OBE	
			CO	RBT
1	<p>a) What is a database? What are the implicit properties of a database? A database is a collection of related data.1 By data, we mean known facts that can be recorded and that have implicit meaning. For example, consider the names, telephone numbers, and addresses of the people you know. You may have recorded this data in an indexed address book or you may have stored it on a hard drive, using a personal computer and software such as Microsoft Access or Excel. This collection of related data with an implicit meaning is a database. A database has the following implicit properties:</p> <ul style="list-style-type: none"> ■ A database represents some aspect of the real world, sometimes called the miniworld or the universe of discourse (UoD). Changes to the miniworld are reflected in the database. ■ A database is a logically coherent collection of data with some inherent meaning. A random assortment of data cannot correctly be referred to as a database. ■ A database is designed, built, and populated with data for a specific purpose.It has an intended group of users and some preconceived applications in which these users are interested. 	4M	CO1	L1
	<p>b) Explain ,with the help of a neat sketch,different phases of database design</p> <div style="text-align: center;">  <p>Figure 7.1 A simplified diagram to illustrate the main phases of database design.</p> </div>	6M	CO1	L4
2	<p>a) Briefly explain about Actors on the Scene,in view of Database users. Database Administrators In any organization where many people use the same resources, there is a need for a chief administrator to oversee and manage these resources. In a database environment,the primary resource is the database itself, and the secondary resource</p>	10M	CO1	L4

is the DBMS and related software. Administering these resources is the responsibility of the database administrator (DBA). The DBA is responsible for authorizing access to the database, coordinating and monitoring its use, and acquiring software and hardware resources as needed. The DBA is accountable for problems such as security breaches and poor system response time. In large organizations, the DBA is assisted by a staff that carries out these functions.

Database Designers

Database designers are responsible for identifying the data to be stored in the database and for choosing appropriate structures to represent and store this data. These tasks are mostly undertaken before the database is actually implemented and populated with data. It is the responsibility of database designers to communicate with all prospective database users in order to understand their requirements and to create a design that meets these requirements. In many cases, the designers are on the staff of the DBA and may be assigned other staff responsibilities after the database design is completed. Database designers typically interact with each potential group of users and develop views of the database that meet the data and processing requirements of these groups. Each view is then analyzed and integrated with the views of other user groups. The final database design must be capable of supporting the requirements of all user groups.

End Users

End users are the people whose jobs require access to the database for querying, updating, and generating reports; the database primarily exists for their use. There are several categories of end users:

- Casual end users occasionally access the database, but they may need different information each time. They use a sophisticated database query language to specify their requests and are typically middle- or high-level managers or other occasional browsers.

- Naive or parametric end users make up a sizable portion of database end users. Their main job function revolves around constantly querying and updating the database, using standard types of queries and updates—called canned transactions—that have been carefully programmed and tested. The tasks that such users perform are varied:

 - Bank tellers check account balances and post withdrawals and deposits.

 - Reservation agents for airlines, hotels, and car rental companies check availability for a given request and make reservations. Employees at receiving stations for shipping companies enter package identifications via bar codes and descriptive information through buttons to update a central database of received and in-transit packages.

- Sophisticated end users include engineers, scientists, business analysts, and others who thoroughly familiarize themselves with the facilities of the DBMS in order to implement their own applications to meet their complex requirements.

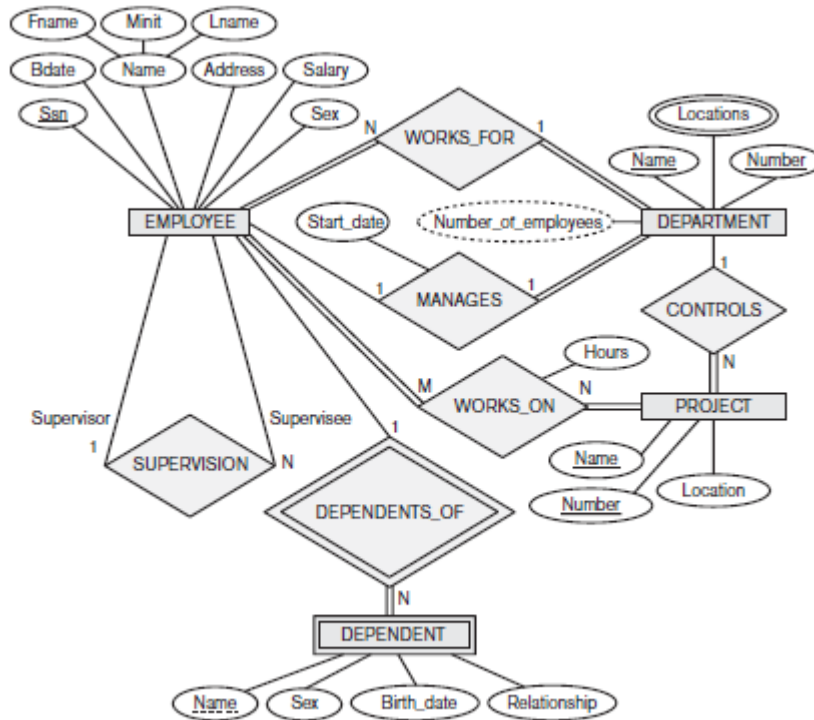
- Standalone users maintain personal databases by using ready-made program packages that provide easy-to-use menu-based or graphics-based interfaces. An example is the user of a tax package that stores a variety of personal financial data for tax purposes.

A typical DBMS provides multiple facilities to access a database. Naive end users need to learn very little about the facilities provided by the DBMS; they simply have to understand the user interfaces of the standard transactions designed and implemented for their use. Casual users learn only a few facilities that they may use repeatedly. Sophisticated users try to learn most of the DBMS facilities in order to achieve their complex requirements. Standalone users typically become very proficient in using a specific software package.

	<p>System Analysts and Application Programmers (Software Engineers)</p> <p>System analysts determine the requirements of end users, especially naive and parametric end users, and develop specifications for standard canned transactions that meet these requirements. Application programmers implement these specifications as programs; then they test, debug, document, and maintain these canned transactions. Such analysts and programmers—commonly referred to as software developers or software engineers—should be familiar with the full range of Capabilities provided by the DBMS to accomplish their tasks.</p>			
3	<p>a) Write the algorithm for ER-to-Relational Mapping .</p> <p>Step 1: Mapping of Regular Entity Types. For each regular (strong) entity type E in the ER schema, create a relation R that includes all the simple attributes of E. Include only the simple component attributes of a composite attribute. Choose one of the key attributes of E as the primary key for R. If the chosen key of E is a composite, then the set of simple attributes that form it will together form the primary key of R.</p> <p>Step 2: Mapping of Weak Entity Types. For each weak entity type W in the ER schema with owner entity type E, create a relation R and include all simple attributes (or simple components of composite attributes) of W as attributes of R. In addition, include as foreign key attributes of R, the primary key attribute(s) of the relation(s) that correspond to the owner entity type(s); this takes care of mapping the identifying relationship type of W. The primary key of R is the combination of the primary key(s) of the owner(s) and the partial key of the weak entity type W, if any.</p> <p>Step 3: Mapping of Binary 1:1 Relationship Types. For each binary 1:1 relationship type R in the ER schema, identify the relations S and T that correspond to the entity types participating in R. There are three possible approaches: (1) the foreign key approach, (2) the merged relationship approach, and (3) the cross-reference or relationship relation approach.</p> <p>Step 4: Mapping of Binary 1:N Relationship Types. For each regular binary 1:N relationship type R, identify the relation S that represents the participating entity type at the N-side of the relationship type. Include as foreign key in S the primary key of the relation T that represents the other entity type participating in R; we do this because each entity instance on the N-side is related to at most one entity instance on the 1-side of the relationship type. Include any simple attributes (or simple components of composite attributes) of the 1:N relationship type as attributes of S.</p> <p>Step 5: Mapping of Binary M:N Relationship Types. For each binary M:N relationship type R, create a new relation S to represent R. Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types; their combination will form the primary key of S. Also include any simple attributes of the M:N relationship type (or simple components of composite attributes) as attributes of S. Notice that we cannot represent an M:N relationship type by a single foreign key attribute in one of the participating relations (as we did for 1:1 or 1:N relationship types) because of the M:N cardinality ratio; we must create a separate relationship relation S.</p> <p>Step 6: Mapping of Multivalued Attributes. For each multivalued attribute A, create a new relation R. This relation R will include an attribute corresponding to A, plus the primary key attribute K—as a foreign key in R—of the relation that represents the entity type or relationship type that has A as a multivalued attribute. The primary key of R is the combination of A and K. If the multivalued attribute is composite, we include its simple components</p> <p>Step 7: Mapping of N-ary Relationship Types. For each n-ary relationship type R, where $n > 2$, create a new relation S to represent R. Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types. Also include any simple attributes of the n-ary relationship type (or simple components of composite attributes)</p>	5M	CO2	L1

as attributes of S. The primary key of S is usually a combination of all the foreign keys that reference the relations representing the participating entity types. However, if the cardinality constraints on any of the entity types E participating in R is 1, then the primary key of S should not include the foreign key attribute that references the relation E corresponding to E

b) Convert the below ER model to Relational Model. Give reasons for each step.



Relational Model

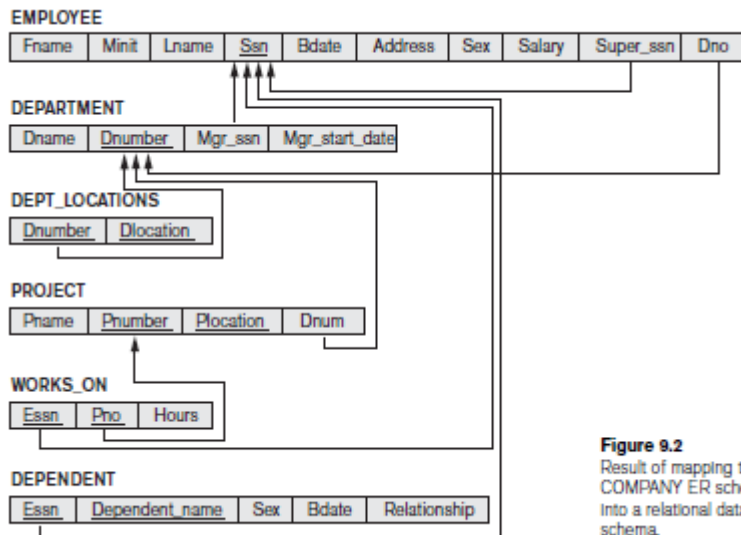


Figure 9.2 Result of mapping the COMPANY ER schema into a relational database schema.

5M CO2 L2

a) Consider the following database of student enrollment in courses & books adopted for each course.
 STUDENT (regno: string, name: string, major: string, bdate:date)
 COURSE (course #:int, cname:string, dept:string)
 ENROLL (regno:string, course#:int, sem:int, marks:int)
 BOOK_ADOPTION (course# :int, sem:int, book-ISBN:int)
 TEXT (book-ISBN:int, book-title:string, publisher:string,author:string)

4

10M CO4 L3

	<p>Solve the following questions using relational algebra.</p> <ol style="list-style-type: none"> Find the name of the author of the book “Let Us C”. Find the average marks for each course for every semester. List the student's name and marks in DBMS in semester 4. Find the number of students enrolled for the course DBMS. Find the number of books adopted for the 5th semester course OS. 			
5	<p>a) Consider the following relations: Student (<i>snum</i>: integer, <i>sname</i>: string, <i>major</i>: string, <i>level</i>: string, <i>age</i>: integer) Class (<i>name</i>: string, <i>meets at</i>: string, <i>room</i>: string, <i>d</i>: integer) Enrolled (<i>snum</i>: integer, <i>cname</i>: string) Faculty (<i>fid</i>: integer, <i>fname</i>: string, <i>deptid</i>: integer)</p> <p>Solve the following questions using SQL queries.</p> <ol style="list-style-type: none"> Find the names of all Juniors (level = JR) who are enrolled in a class taught by Prof. Harshith SELECT DISTINCT S.SNUM, S.SNAME FROM E1_STUDENT S, E1_CLASS C, E1_ENROLLED E, E1_FACULTY F WHERE S.SNUM = E.SNUM AND E.CNAME = C.NAME AND C.FID = F.FID AND F.FNAME = 'ANAND R' AND S.SLEVEL = 'JR' ORDER BY S.SNAME; Find the names of all classes that either meet in room R128 or have five or more Students enrolled. SELECT C.NAME, C.ROOM FROM E1_CLASS C WHERE C.ROOM = '204' OR C.NAME IN (SELECT E.CNAME FROM E1_ENROLLED E GROUP BY E.CNAME HAVING COUNT (*) >=5) ORDER BY C.NAME; Find the names of all students who are enrolled in two classes that meet at the same time. SELECT DISTINCT S.SNAME, S.SNUM FROM E1_STUDENT S WHERE S.SNUM IN (SELECT E1.SNUM FROM E1_ENROLLED E1, E1_ENROLLED E2, E1_CLASS C1, E1_CLASS C2 	10M	CO5	L3

	<p>WHERE E1.SNUM = E2.SNUM AND E1.CNAME != E2.CNAME AND E1.CNAME = C1.NAME AND E2.CNAME = C2.NAME AND C1.MEETS_AT = C2.MEETS_AT) ORDER BY S.SNAME;</p> <p>iv. Find the names of faculty members who teach in every room in which some class is taught. SELECT DISTINCT FNAME, FID FROM E1_FACULTY WHERE FID IN (SELECT FID FROM E1_CLASS GROUP BY FID HAVING COUNT (DISTINCT ROOM) >= (SELECT COUNT (DISTINCT ROOM) FROM E1_CLASS)) ORDER BY FNAME;</p> <p>v. Find the names of faculty members for whom the combined enrollment of the courses that they teach is less than five. SELECT DISTINCT FNAME, FID FROM E1_FACULTY WHERE FID IN (SELECT F.FID FROM E1_FACULTY F LEFT OUTER JOIN E1_CLASS C ON (F.FID = C.FID) GROUP BY F.FID HAVING COUNT (C.NAME) < 5) ORDER BY FNAME;</p>			
6	<p>a) What is nonadditive join property of a Decomposition. Write an algorithm for Testing Nonadditive Join Property Testing for Nonadditive Join Property Input: A universal relation R, a decomposition $D = \{R_1, R_2, \dots, R_m\}$ of R, and a set F of functional dependencies. Note: Explanatory comments are given at the end of some of the steps. They follow the format: (* comment *). 1. Create an initial matrix S with one row i for each relation R_i in D, and one column j for each attribute A_j in R. 2. Set $S(i, j) := b_{ij}$ for all matrix entries. (* each b_{ij} is a distinct symbol associated with indices (i, j) *). 3. For each row i representing relation schema R_i {for each column j representing attribute A_j {if (relation R_i includes attribute A_j) then set $S(i, j) := a_j$;}; (* each a_j is a distinct symbol associated with index (j) *).</p>	5M	CO6	L1

4. Repeat the following loop until a complete loop execution results in no changes to S
 {for each functional dependency $X \rightarrow Y$ in F
 {for all rows in S that have the same symbols in the columns corresponding to attributes in X
 {make the symbols in each column that correspond to an attribute in Y be the same in all these rows as follows: If any of the rows has an a symbol for the column, set the other rows to that same a symbol in the column. If no a symbol exists for the attribute in any of the rows, choose one of the b symbols that appears in one of the rows for the attribute and set the other rows to that same b symbol in the column ; } ; } ; }
 5. If a row is made up entirely of a symbols, then the decomposition has the nonadditive join property; otherwise, it does not.

b) Test whether the below decomposition is lossy or lossless.
 $R = \{Ssn, Ename, Pnumber, Pname, Plocation, Hours\}$
 $R_1 = EMP = \{Ssn, Ename\}$
 $R_2 = PROJ = \{Pnumber, Pname, Plocation\}$
 $R_3 = WORKS_ON = \{Ssn, Pnumber, Hours\}$
 $F = \{Ssn \rightarrow Ename; Pnumber \rightarrow \{Pname, Plocation\}; \{Ssn, Pnumber\} \rightarrow Hours\}$

$R = (Ssn, Ename, Pnumber, Pname, Plocation, Hours)$ $D = (R_1, R_2, R_3)$
 $R_1 = EMP = (Ssn, Ename)$
 $R_2 = PROJ = (Pnumber, Pname, Plocation)$
 $R_3 = WORKS_ON = (Ssn, Pnumber, Hours)$

$F = (Ssn \rightarrow Ename; Pnumber \rightarrow (Pname, Plocation); \{Ssn, Pnumber\} \rightarrow Hours)$

	Ssn	Ename	Pnumber	Pname	Plocation	Hours
R_1	a_1	a_2	b_{13}	b_{14}	b_{15}	b_{16}
R_2	b_{21}	b_{22}	a_3	a_4	a_5	b_{26}
R_3	a_1	b_{32}	a_3	b_{34}	b_{35}	a_6

(Original matrix S at start of algorithm)

	Ssn	Ename	Pnumber	Pname	Plocation	Hours
R_1	a_1	a_2	b_{13}	b_{14}	b_{15}	b_{16}
R_2	b_{21}	b_{22}	a_3	a_4	a_5	b_{26}
R_3	a_1	a_2	a_3	a_4	a_5	a_6

(Matrix S after applying the first two functional dependencies; last row is all "a" symbols so we stop)

So the decomposition is lossless.

a) Define 4NF and 5NF .Give examples.
 Definition. A **multivalued dependency** $X \twoheadrightarrow Y$ specified on relation schema R, where X and Y are both subsets of R, specifies the following constraint on any relation state r of R: If two tuples t1 and t2 exist in r such that $t1[X] = t2[X]$, then two tuples t3 and t4 should also exist in r with the following properties, where we use Z to denote $(R - (X \cup Y))$:
 ■ $t3[X] = t4[X] = t1[X] = t2[X]$.
 ■ $t3[Y] = t1[Y]$ and $t4[Y] = t2[Y]$.
 ■ $t3[Z] = t2[Z]$ and $t4[Z] = t1[Z]$.
 Definition. A **relation schema R is in 4NF** with respect to a set of dependencies F (that includes functional dependencies and multivalued dependencies) if, for every nontrivial multivalued dependency $X \twoheadrightarrow Y$ in F+17 X is a superkey for R.

Definition. A **join dependency (JD)**, denoted by $JD(R_1, R_2, \dots, R_n)$, specified on relation schema R, specifies a constraint on the states r of R. The constraint states that every legal state r of R should have a nonadditive join decomposition into R_1, R_2, \dots, R_n . Hence, for every such r we have $* (\pi R_1(r), \pi R_2(r), \dots, \pi R_n(r)) = r$

7

5M CO6 L5

10M CO6 L1

	<p>Definition. A relation schema R is in fifth normal form (5NF) (or project-join normal form (PJNF)) with respect to a set F of functional, multivalued, and join dependencies if, for every nontrivial join dependency JD(R1, R2, ..., Rn) in F+ (that is, implied by F), every Ri is a superkey of R.</p>			
8	<p>a) Write a program to demonstrate retrieval of multiple tuples from database with embedded SQL using cursors</p> <p>Program segment E2, a C program segment that uses cursors with embedded SQL for update purposes.</p> <pre> //Program Segment E2: 0) prompt("Enter the Department Name: ", dname) ; 1) EXEC SQL 2) select Dnumber into :dnumber 3) from DEPARTMENT where Dname = :dname ; 4) EXEC SQL DECLARE EMP CURSOR FOR 5) select Ssn, Fname, Minit, Lname, Salary 6) from EMPLOYEE where Dno = :dnumber 7) FOR UPDATE OF Salary ; 8) EXEC SQL OPEN EMP ; 9) EXEC SQL FETCH from EMP into :ssn, :fname, :minit, :lname, :salary ; 10) while (SQLCODE == 0) { 11) printf("Employee name is:", Fname, Minit, Lname) ; 12) prompt("Enter the raise amount: ", raise) ; 13) EXEC SQL 14) update EMPLOYEE 15) set Salary = Salary + :raise 16) where CURRENT OF EMP ; 17) EXEC SQL FETCH from EMP into :ssn, :fname, :minit, :lname, :salary ; 18) } 19) EXEC SQL CLOSE EMP ; </pre>	10M	CO5	L1

Course Outcomes		PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1:	Describe features, classifications and characteristics of database systems.	1	2	1	1	1	1	1	-		-	-	1
CO2:	Design an information model for given requirements expressed in the form of an entity relation diagram.	1	3	3	1	1	2	-	-	1	-	-	1
CO3:	Design a relational data model for a given information model.	1	3	3	1	1	2	-	-	1	-	-	1
CO4:	Write relational algebra query for a given problem.	1	3	3	1	1	2	-	-	1	-	-	1
CO5:	Write SQL for CRUD to fulfill given requirement.	1	3	3	1	1	2	-	-	1	-	-	1
CO6:	Apply normalization techniques for a given relational model.	1	3	3	1	1	2	-	-	1	-	-	1

Cognitive level	KEYWORDS
L1	List, define, tell, describe, identify, show, label, collect, examine, tabulate, quote, name, who, when, where, etc.
L2	summarize, describe, interpret, contrast, predict, associate, distinguish, estimate, differentiate, discuss, extend
L3	Apply, demonstrate, calculate, complete, illustrate, show, solve, examine, modify, relate, change, classify, experiment, discover.
L4	Analyze, separate, order, explain, connect, classify, arrange, divide, compare, select, explain, infer.
L5	Assess, decide, rank, grade, test, measure, recommend, convince, select, judge, explain, discriminate, support, conclude, compare, summarize.

PO1 - *Engineering knowledge*; PO2 - *Problem analysis*; PO3 - *Design/development of solutions*; PO4 - *Conduct investigations of complex problems*; PO5 - *Modern tool usage*; PO6 - *The Engineer and society*; PO7- *Environment and sustainability*; PO8 – *Ethics*; PO9 - *Individual and team work*; PO10 - *Communication*; PO11 - *Project management and finance*; PO12 - *Life-long learning*