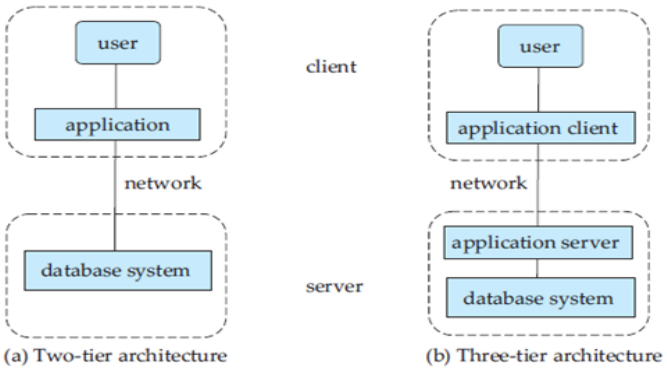
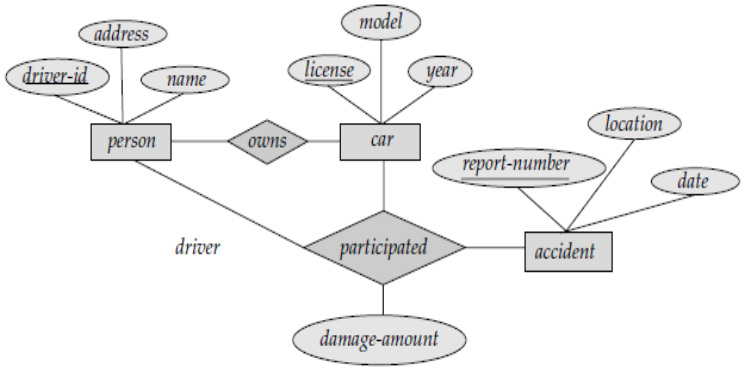
Answer Key

Name of Examination	**Final Assessment Test, Winter 2022-23Semester, (APRIL 2023)**				
Slot: A2 +TA2	Course Mode: CBL		Class Number (s): VL2022230504178		
Course Code:	**CSE2031**	Course Code:	**Principles of Database Management Systems**		
Emp. No.:	18844	Faculty Name:	Dr.V.Ilayaraja	School: SCOPE	
Contact No.:	9843189648	Email:	ilayaraja.v@vit.ac.in		

General Instructions (if any): Answer All Questions

Q. No.	Sub-division	Question Text	Marks	Unit / Module No.	Difficulty Level E/A/T	BL	CO
Answer any TEN Total Marks: 10 X 10 = 100 Marks							
1.	a)	List four significant differences between file-processing system and DBMS. 1.1 List four significant differences between a file-processing system and a DBMS. **Answer:** Some main differences between a database management system and a file-processing system are: • Both systems contain a collection of data and a set of programs which access that data. A database management system coordinates both the physical and the logical access to the data, whereas a file-processing system coordinates only the physical access. • A database management system reduces the amount of data duplication by ensuring that a physical piece of data is available to all programs authorized to have access to it, whereas data written by one program in a file-processing system may not be readable by another program. • A database management system is designed to allow flexible access to data (i.e., queries), whereas a file-processing system is designed to allow predetermined access to data (i.e., compiled programs). • A database management system is designed to coordinate multiple users accessing the same data at the same time. A file-processing system is usually designed to allow one or more programs to access different data files at the same time. In a file-processing system, a file can be accessed by two programs concurrently only if both programs have read-only access to the file.	3	1	E	BL2	CO1
	b)	Describe the two and three tier client-server architecture with neat diagrams in detail. **Client/Server Architectures**  (a) Two-tier architecture (b) Three-tier architecture	7	1	E	BL1	CO1

2.	a)	Construct an E-R diagram for a car-insurance company whose customers own one or more cars each. Each car has associated with it zero to any number of recorded accidents. Also, indicate the key and mapping constraints.  Figure 2.1 E-R diagram for a Car-insurance company.	3	2	A	BL3	CO2
	b)	A university registrar's office maintains data about the following entities: (a) courses, including number, title, credits, syllabus, and prerequisites; (b) course offerings, including course number, year, semester, section number, instructor(s), timings, and classroom; (c) students, including student-id, name, and program; and (d) instructors, including identification number, name, department, and title. Further, the enrollment of students in courses and grades awarded to students in each course they are enrolled for must be appropriately modeled. Construct an E-R diagram for the registrar's office. Document all assumptions that you make about the mapping constraints. Also indicate the key for each entity.	7	2	A	BL3	CO2

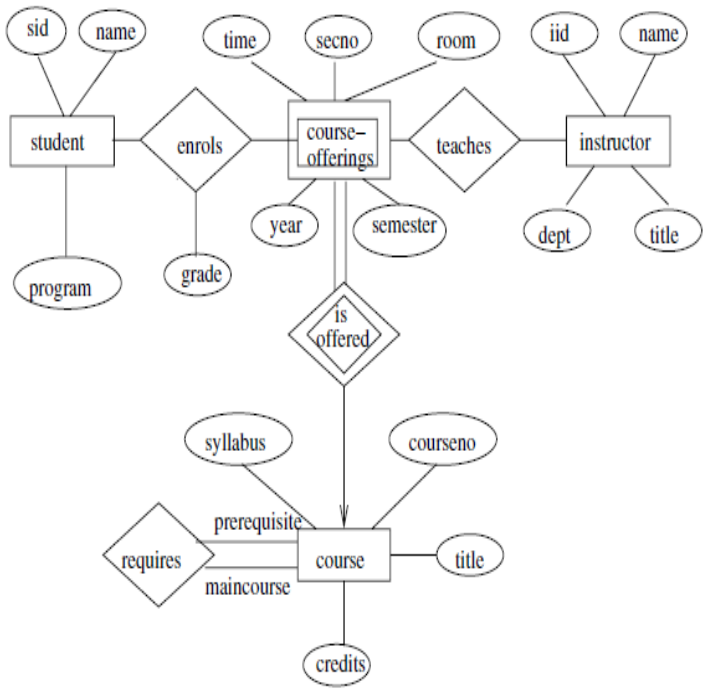
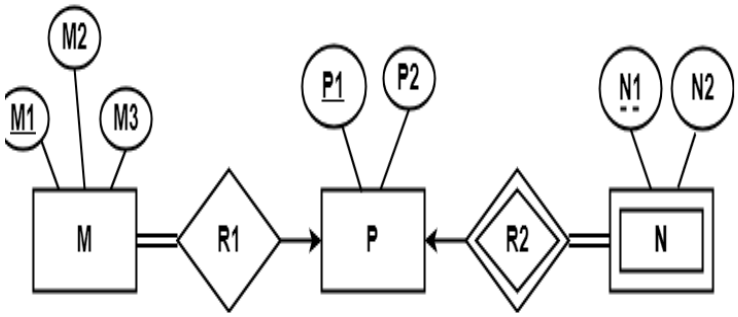


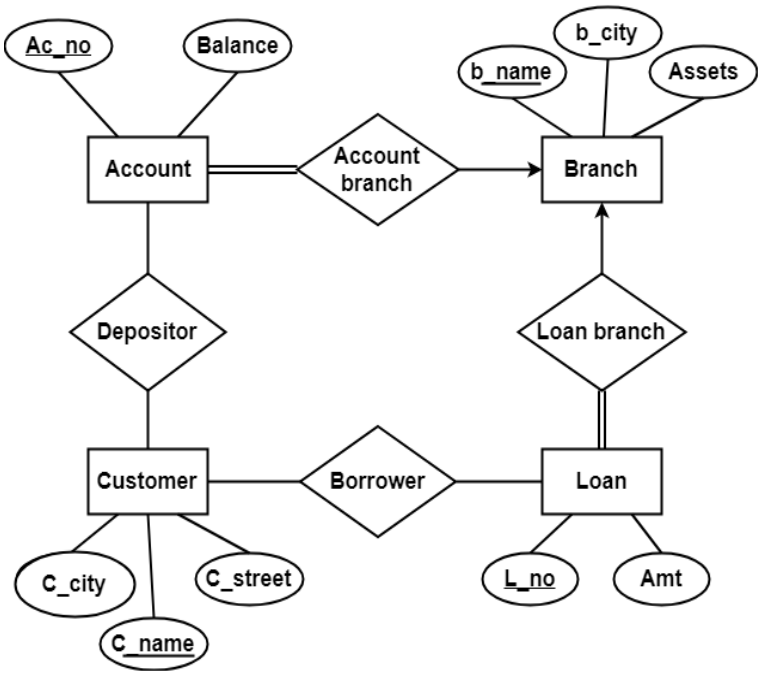
Figure 2.3 E-R diagram for a university.

3.	a)	Design a relational schema corresponding to the given E-R diagram. Also, choose a primary key for a relational schema with respect to mapping cardinality constraints.	3	2	A	BL3	CO2
	b)	Find the minimum number of tables required to represent the given ER diagram in relational model. Also, choose a primary key for a relational schema with respect to mapping cardinality constraints.	7	2	T	BL3	CO2



Applying the rules, minimum 3 tables will be required-

- MR1 (M1 , M2 , M3 , P1)
- P (P1 , P2)
- NR2 (P1 , N1 , N2)



Applying the rules that we have learnt, minimum 6 tables will be required-

- Account (Ac_no , Balance , b_name)
- Branch (b_name , b_city , Assets)
- Loan (L_no , Amt , b_name)
- Borrower (C_name , L_no)
- Customer (C_name , C_street , C_city)
- Depositor (C_name , Ac_no)

4. The Gill Art Gallery wishes to maintain data on their customers, artists and arts. They may have several arts by each artist in the gallery at one time. The gallery may sell an art, then buy it back at a later date and sell it to another customer.

Consider the following customer history form and Demonstrate the unnormalized relation and convert into 1NF, 2NF and 3NF. Justify each normal form.

Gallery Customer History Form

Customer Name

Jackson, Elizabeth Phone (206) 284-6783
123 – 4th Avenue
Fonthill, ON
L3J 4S4

Purchases Made

Artist	Art Title	Purchase Date	Sales Price
03 - Carol Channing	Laugh with Teeth	09/17/2000	7000.00
15 - Dennis Frings	South toward Emerald Sea	05/11/2000	1800.00
03 - Carol Channing	At the Movies	02/14/2002	5550.00
15 - Dennis Frings	South toward Emerald Sea	07/15/2003	2200.00

UNF:

customer [custno, cust_name, cust_addr, cust_phone, (artist_id, artist_name, art_title, pur_date, price)]

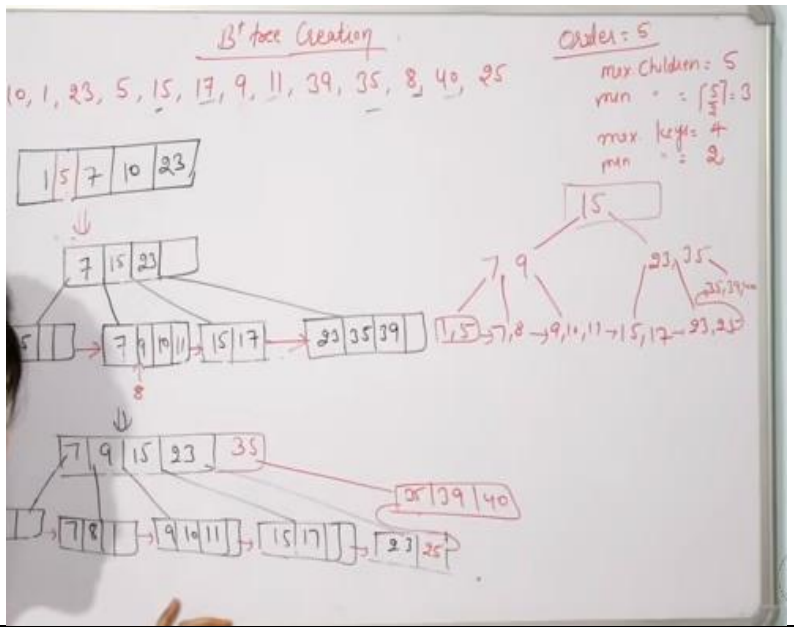
1NF:
customer [custno, cust_name, cust_addr, cust_phone]
cust_art [custno, art_code, pur_date, artist_id, artist_name, art_title, price]

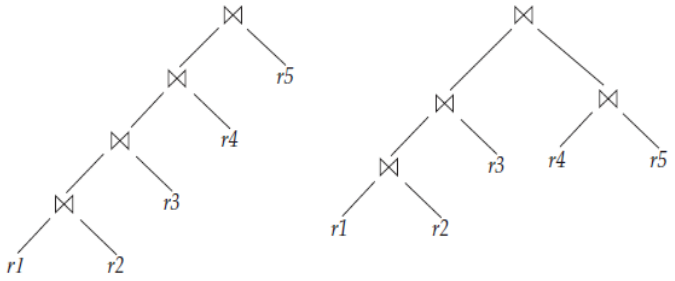
note: the key chosen for the repeating group is the piece of art itself (a code was assigned), however because a piece of art may be bought by a customer more than once, the purchase date was added as part of the key to make the rows unique.

2NF:
customer [custno, cust_name, cust_addr, cust_phone]
cust_art [custno, art_code, pur_date, price]
art [art_code, art_title, artist_id, artist_name]

3NF:
customer [custno, cust_name, cust_street, cust_city, cust_prov, cust_pstlcd, cust_phone]

cust_art [custno, art_code, pur_date, price]
art [art_code, art_title, artist_id(FK)]
artist [artist_id, artist_fname, artist_lname]

5.		Construct a B+ tree for following key values using order 5 and show the constructed output for each insertion. 7, 10, 1, 23, 5, 15, 17, 9, 11, 39, 35, 8, 40, 25.  	10	4	T	BL3	CO5
6.	a)	a) Consider the following relational database, where the primary keys are underlined. Give an expression in the relational algebra to express each of the following queries: employee (person-name, street, city) works (person-name, company-name, salary) company (company-name, city) manages (person-name, manager-name) i) Find the names of all employees who work for First Bank Corporation. ii) Find the names, street and cities of residence of all employees who work for First Bank Corporation and earn more than Rs.10,000 per annum. a. $\Pi_{person\text{-}name}(\sigma_{company\text{-}name = \text{"First Bank Corporation"}}(works))$ c. $\Pi_{person\text{-}name,\ street,\ city}$ $(\sigma_{(company\text{-}name = \text{"First Bank Corporation"} \wedge salary > 10000)}$ $works \bowtie employee)$	3	5	A	BL3	CO3
	b)	Brief about the cost based and heuristic query optimization with necessary diagrams. **Cost based:** The procedure for generating equivalent expressions can be modified to generate all possible evaluation plans as follows: A new class of equivalence rules, **called physical equivalence rules,** is added that allows a logical operation, such as a join, to be transformed to a physical operation, such as a hash join, or a nested-loops join. To make the approach work efficiently requires the following: **1.** A space-efficient representation of expressions that avoids making multiple copies of the same sub expressions when	7	5	E	BL2	CO3

		equivalence rules are applied.
		2. Efficient techniques for detecting duplicate derivations of the same expression.
		3. A form of dynamic programming based on **memoization**, which stores the optimal query evaluation plan for a sub expression when it is optimized for the first time; subsequent requests to optimize the same sub expression are handled by returning the already memoized plan.
		4. Techniques that avoid generating all possible equivalent plans, by keeping track of the cheapest plan generated for any sub expression up to any point of time, and pruning away any plan that is more expensive than the cheapest plan found so far for that sub expression.
		
		Heuristics in Optimization:
		• A drawback of cost-based optimization is the cost of optimization itself.
		• Although the cost of query optimization can be reduced by clever algorithms, the number of different evaluation plans for a query can be very large, and finding the optimal plan from this set requires a lot of computational effort.
		• Hence, optimizers use **heuristics** to reduce the cost of optimization.
		• An example of a heuristic rule is the following rule for transforming relational algebra queries:
		  • Perform selection operations as early as possible.
		  • Perform projections early.
		 
		
		(a) Left-deep join tree (b) Non-left-deep join tree
		optimization cost budget
		plan caching.					

7.	a)	During its execution, a transaction passes through several states, until it finally commits or aborts. List all possible sequences of states through which a transaction may pass. Justify why each state transition may occur.	3	5	A	BL4	CO3

		Answer: The possible sequences of states are: a. *active → partially committed → committed*. This is the normal sequence a successful transaction will follow. After executing all its statements it enters the *partially committed* state. After enough recovery information has been written to disk, the transaction finally enters the *committed* state. b. *active → partially committed → aborted*. After executing the last statement of the transaction, it enters the *partially committed* state. But before enough recovery information is written to disk, a hardware failure may occur destroying the memory contents. In this case the changes which it made to the database are undone, and the transaction enters the *aborted* state. c. *active → failed → aborted*. After the transaction starts, if it is discovered at some point that normal execution cannot continue (either due to internal program errors or external errors), it enters the failed state. It is then rolled back, after which it enters the *aborted* state.					
	b)	What is a cascadeless schedule? Why is cascadeless of schedules desirable? Are there any circumstances under which it would be desirable to allow noncascadeless schedules? Give an example schedule for noncascadeless. Justify your answer. • A cascadeless schedule is one where, for each pair of transactions *Ti* and *Tj* such that *Tj* reads data items previously written by *Ti*, the commit operation of *Ti* appears before the read operation of *Tj*. • Cascadeless schedules are desirable because the failure of a transaction does not lead to the aborting of any other transaction. • Of course this comes at the cost of less concurrency. • If failures occur rarely, so that we can pay the price of cascading aborts for the increased concurrency, noncascadeless schedules might be desirable. • **noncascadeless shedule** •	7	5	A	BL4	CO3

8.	a)	What type of problem is there in a following schedule? Justify the drawback of this problem.
		T_3: lock-X(B);
		read(B);
		$B := B - 50$;
		write(B);
		lock-X(A);
		read(A);
		$A := A + 50$;
		write(A);
		unlock(B);
		unlock(A).
		unlocking is delayed to the end of the transaction.
			3	6	A	BL4	CO4

	b)	Write the rules of timestamp based protocol when transaction issues a read and write operation. Explain in detail.	7	6	A	BL2	CO4

T_8	T_9	T_{10}
read(A)		
read(B)		
write(A)		
	read(A)	
	write(A)	
		read(A)
abort		

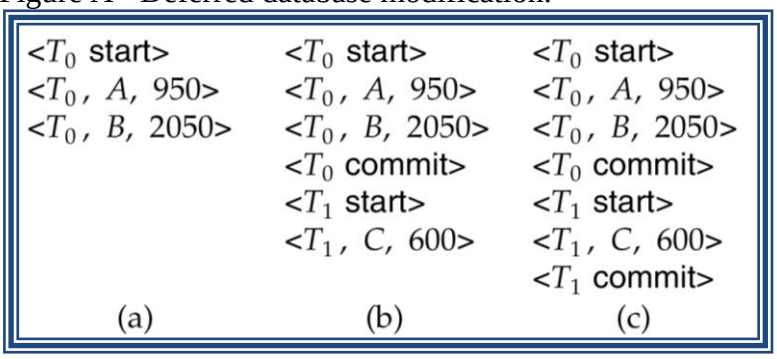
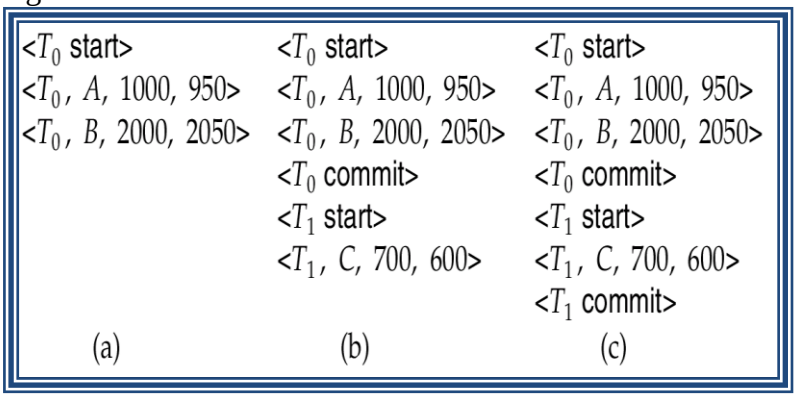
		• Suppose a transaction T_i issues a **read**(Q) 1. If TS(T_i) < **W**-timestamp(Q), then T_i needs to read a value of Q that was already overwritten. Hence, the **read** operation is rejected, and T_i is rolled back. 2. If TS(T_i)≥ **W**-timestamp(Q), then the **read** operation is executed, and R-timestamp(Q) is set to the maximum of R-timestamp(Q) and TS(T_i). • Suppose that transaction T_i issues **write**(Q). 1. If TS(T_i) < R-timestamp(Q), then the value of Q that T_i is producing was needed previously, and the system assumed that that value would never be produced. Hence, the **write** operation is rejected, and T_i is rolled back. 2. If TS(T_i) < W-timestamp(Q), then T_i is attempting to write an obsolete value of Q. Hence, this **write** operation is rejected, and T_i is rolled back. 3. Otherwise, the **write** operation is executed, and W-timestamp(Q) is set to TS(T_i).					
9.		Consider the following log instances a, b and c in Figure A and B. If log on stable storage at time of crash at every instance, State the recovery mechanism for each log instance in both deferred (consider Figure A) and immediate (consider Figure B) database modification. Figure A - Deferred database modification.  (a) No redo actions need to be taken (b) redo(T0) must be performed since <T0 commit> is present (c) redo(T0) must be performed followed by redo(T1) since <T0 commit> and <Ti commit> are present	10	6	A	BL4	CO4

Figure B - Immediate database modification



(a) undo (T_0): B is restored to 2000 and A to 1000.
(b) undo (T_1) and redo (T_0): C is restored to 700, and then *A* and *B* are set to 950 and 2050 respectively.
(c) redo (T_0) and redo (T_1): A and B are set to 950 and 2050 respectively. Then *C* is set to 600

10		Describe about the features, advantages and disadvantages of various NoSQL data models. Also give an example for each and brief about an example. **Key Value Stores** • Most Based on Dynamo: Amazon Highly Available Key-Value Store **• Data Model:** – Global key-value mapping – Big scalable HashMap – Highly fault tolerant (typically) **• Examples:** – Redis, Riak, Voldemort **Pros:** – Simple data model – Scalable **• Cons** – Create your own “foreign keys” – Poor for complex data **Column Family** • Most Based on BigTable: Google’s Distributed Storage System for Structured Data **• Data Model:** – A big table, with column families – Map Reduce for querying/processing **• Examples:** – HBase, HyperTable, Cassandra **Pros:** – Supports Simi-Structured Data – Naturally Indexed (columns) – Scalable **• Cons** – Poor for interconnected data **Document Databases** **Data Model:** – A collection of documents – A document is a key value collection – Index-centric, lots of map-reduce	10	7	E	BL2	CO6

		• **Examples:** – CouchDB, MongoDB **Pros:** – Simple, powerful data model – Scalable • **Cons** – Poor for interconnected data – Query model limited to keys and indexes – Map reduce for larger queries **Graph Databases** **Data Model:** – Nodes and Relationships • **Examples:** – Neo4j, OrientDB, Infinite Graph, Allegro Graph **Pros:** – Powerful data model, as general as RDBMS – Connected data locally indexed – Easy to query • **Cons** – Sharding (lots of people working on this) • Scales UP reasonably well – Requires rewiring your brain					