CMR INSTITUTE OF TECHNOLOGY

Affiliated to VTU, Approved by AICTE, Accredited by NBA and NAAC with "A++" Grade

ITPL MAIN ROAD, BROOKFIELD, BENGALURU-560037, KARNATAKA, INDIA

Department of Computer Science Engineering

Answer Scheme & Model Solution- IAT2

Sub: Database Management System		Sub Code: 18CS53	Sem/Branch: V / CSE		Sections: A,B,C
			MARKS	**CO**	**RBT**
Question	1	Explain how the different update operations deal with constraint violations.	10	CO2	L2
Scheme			1M+3M +3M+ 3M		
Solution		There are three basic operations that can change the states of relations in the database: 1. Insert - used to insert one or more new tuples in a relation 2. Delete- used to delete tuples 3. Update (or Modify)- used to change the values of some attributes in existing tuples Whenever these operations are applied, the integrity constraints specified on the relational database schema should not be violated. There are four types of constraints: 1. Domain constraints : if an attribute value is given that does not appear in the corresponding domain or is not of the appropriate data type 2. Key constraints : if a key value in the new tuple t already exists in another tuple in the relation r(R) 3. Entity integrity: if any part of the primary key of the new tuple t is NULL 4. Referential integrity : if the value of any foreign key in t refers to a tuple that does not exist in the referenced relation. **The Insert Operation** The Insert operation provides a list of attribute values for a new tuple t that is to be inserted into a elation R. Insert can violate any of the mentioned four types of constraints Examples:			

1. Operation:
Insert <‘Cecilia’, ‘F’, ‘Kolonsky’, NULL, ‘1960-04-05’, ‘6357 Windy Lane, Katy, TX’, F, 28000, NULL, 4>
Result: This insertion violates the entity integrity constraint (NULL for the primary key Ssn), so it is rejected
2. Operation:
Insert <‘Alicia’, ‘J’, ‘Zelaya’, ‘999887777’, ‘1960-04-05’, ‘6357 Windy Lane, Katy, TX’, F, 28000, ‘987654321’, 4>
Result: This insertion violates the key constraint because another tuple with the same Ssn value already exists in the EMPLOYEE relation, and so it is rejected.
3. Operation:
Insert <‘Cecilia’, ‘F’, ‘Kolonsky’, ‘677678989’, ‘1960-04-05’, ‘6357 Windswept, Katy,
TX’, F, 28000, ‘987654321’, 7>

Result: This insertion violates the referential integrity constraint specified on Dno in EMPLOYEE because no corresponding referenced tuple exists in DEPARTMENT with Dnumber = 7.
4. Operation:
Insert <‘Cecilia’, ‘F’, ‘Kolonsky’, ‘677678989’, ‘1960-04-05’, ‘6357 Windy Lane,Katy,
TX’, F, 28000, NULL, 4>
Result: This insertion satisfies all constraints, so it is acceptable.

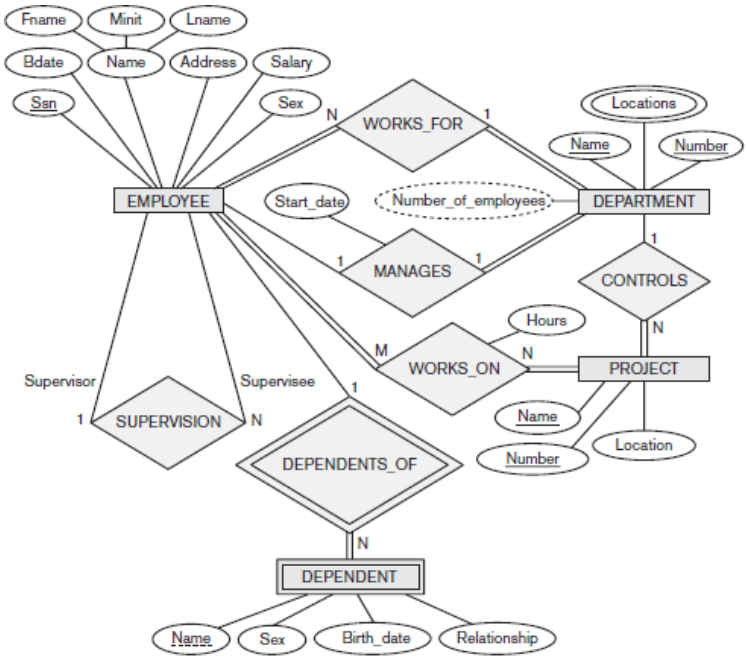
If an insertion violates one or more constraints, the default option is to reject the insertion.It would be useful if the DBMS could provide a reason to the user as to why the insertion was rejected. Another option is to an attempt to correct the reason for rejecting the insertion

The Delete Operation
The Delete operation can violate only referential integrity. This occurs if the tuple being deleted is referenced by foreign keys from other tuples in the database. To specify deletion, a condition on the attributes of the relation selects the tuple (or tuples) to be deleted.
Examples:
1. Operation:
Delete the WORKS_ON tuple with Essn = ‘999887777’ and Pno =10. Result: This deletion is acceptable and deletes exactly one tuple.
2. Operation:
Delete the EMPLOYEE tuple with Ssn = ‘999887777’.
Result: This deletion is not acceptable, because there are tuples in WORKS_ON that refer to this tuple. Hence, if the tuple in EMPLOYEE is deleted, referential integrity violations will result.
3. Operation:
Delete the EMPLOYEE tuple with Ssn = ‘333445555’

		Result: This deletion will result in even worse referential integrity violations, because the tuple involved is referenced by tuples from the EMPLOYEE, DEPARTMENT, WORKS_ON, and DEPENDENT relations. Several options are available if a deletion operation causes a violation 1. restrict - is to reject the deletion 2. cascade, is to attempt to cascade (or propagate) the deletion by deleting tuples that reference the tuple that is being deleted 3. Set null or set default - is to modify the referencing attribute values that cause the violation; each such value is either set to NULL or changed to reference another default valid tuple. **The Update (or Modify) operation:** It is used to change the values of one or more attributes in a tuple (or tuples) of some relation R. It is necessary to specify a condition on the attributes of the relation to select the tuple (or tuples) to be modified. Examples: 1. Operation: Update the salary of the EMPLOYEE tuple with Ssn = '999887777' to 28000. Result: Acceptable. 2. Operation: Update the Dno of the EMPLOYEE tuple with Ssn = '999887777' to 7. Result: Unacceptable, because it violates referential integrity. 3. Operation: Update the Ssn of the EMPLOYEE tuple with Ssn = '999887777' to '987654321'. Result: Unacceptable, because it violates primary key constraint by repeating a value that already exists as a primary key in another tuple; it violates referential integrity constraints because there are other relations that refer to the existing value of Ssn Updating an attribute that is neither part of a primary key nor of a foreign key usually causes no problems; the DBMS need only check to confirm that the new value is of the correct data type and domain.			
Question	2	Discuss the correspondence between the ER model construct and the relational model constructs. Show how each ER model can be mapped to the relational model.	10	CO2	L2
Scheme			3M+7M		
Solution		Correspondence between the ER model construct and the relational model constructs:			

ER Model	Relational Model
Entity type	“Entity” relation
1:1 or 1:N relationship type	Foregin key
M:N relationship type	“Relationship” relation and two foreign keys
n ary relationship type	“Relationship” relation and n foreign keys
Simple attributes	Attributes
Composite attributes	Set of simple component attributes
Multivalued attributes	Relation and foreign key
Value set	Domain
Key attribute	Primary key or secondary key

Relational Database Design using ER-to-Relational mapping



Step 1: Mapping of Regular Entity Types

- For each regular entity type, 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.

- If multiple keys were identified for E during the conceptual design, the information describing the attributes that form each additional key is kept in order to specify secondary (unique) keys of relation R
- In our example-COMPANY database, we create the relations EMPLOYEE, DEPARTMENT, and PROJECT
- we choose Ssn, Dnumber, and Pnumber as primary keys for the relations EMPLOYEE, DEPARTMENT, and PROJECT, respectively
- The relations that are created from the mapping of entity types are called entity relations

because each tuple represents an entity instance.

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary

DEPARTMENT

Dname	Dnumber

PROJECT

Pname	Pnumber	Plocation

Step 2: Mapping of Weak Entity Types

- For each weak entity type, create a relation R and include all simple attributes of the entity type as attributes of R
- Include primary key attribute of owner as foreign key attributes of R
- In our example, we create the relation DEPENDENT in this step to correspond to the weak entity type DEPENDENT
- We include the primary key Ssn of the EMPLOYEE relation—which corresponds to the owner entity type—as a foreign key attribute of DEPENDENT; we rename it as Essn
- The primary key of the DEPENDENT relation is the combination {Essn,Dependent_name}, because Dependent_name is the partial key of DEPENDENT
- It is common to choose the propagate (CASCADE) option for the referential triggered action on the foreign key in the relation corresponding to the weak entity type, since a weak entity has an existence dependency on its owner entity.
- This can be used for both ON UPDATE and ON DELETE.

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship

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:

- foreign key approach

- merged relationship approach
- crossreference or relationship relation approach

The foreign key approach

- Choose one of the relations—S, say—and include as a foreign key in S the primary key of T.
- It is better to choose an entity type with total participation in R in the role of S
- Include all the simple attributes (or simple components of composite attributes) of the 1:1 relationship type R as attributes of S.
- In our example, we map the 1:1 relationship type by choosing the participating entity type DEPARTMENT to serve in the role of S because its participation in the MANAGES relationship type is total
- We include the primary key of the EMPLOYEE relation as foreign key in the DEPARTMENT relation and rename it Mgr_ssn.
- We also include the simple attribute Start_date of the MANAGES relationship type in the DEPARTMENT relation and rename it Mgr_start_date

Merged relation approach:

- merge the two entity types and the relationship into a single relation
- This is possible when both participations are total, as this would indicate that the two tables will have the exact same number of tuples at all times.

3. Cross-reference or relationship relation approach:

- set up a third relation R for the purpose of cross-referencing the primary keys of the two relations S and T representing the entity types.

required for binary M:N relationships

- The relation R is called a relationship relation (or sometimes a lookup table), because each tuple in R represents a relationship instance that relates one tuple from S with one tuple from T
- The relation R will include the primary key attributes of S and T as foreign keys to S and T.
- The primary key of R will be one of the two foreign keys, and the other foreign key will be a unique key of R.

The drawback is having an extra relation, and requiring an extra join operation when combining related tuples from the tables.

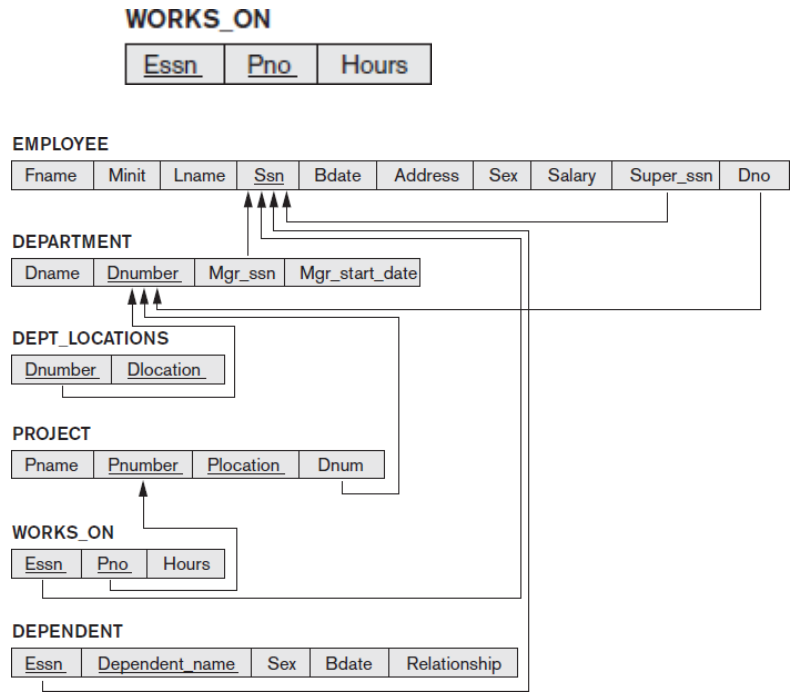
Step 4: Mapping of Binary 1:N Relationship Types

- 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

	Include any simple attributes (or simple components of composite attributes) of the 1:N relationship type as attributes of S In our example, we now map the 1:N relationship types WORKS_FOR, CONTROLS, and SUPERVISION For WORKS_FOR we include the primary key Dnumber of the DEPARTMENT relation as foreign key in the EMPLOYEE relation and call it Dno. For SUPERVISION we include the primary key of the EMPLOYEE relation as foreign key in the EMPLOYEE relation itself—because the relationship is recursive—and call it Super_ssn. The CONTROLS relationship is mapped to the foreign key attribute Dnum of PROJECT, which references the primary key Dnumber of the DEPARTMENT relation. Step 5: Mapping of Binary M:N Relationship Types For each binary M:N relationship type • Create a new relation S • Include primary key of participating entity types as foreign key attributes in S • Include any simple attributes of M:N relationship type In our example, we map the M:N relationship type WORKS_ON by creating the relation WORKS_ON.We include the primary keys of the PROJECT and EMPLOYEE relations as foreign keys in WORKS_ON and rename them Pno and Essn, respectively. We also include an attribute Hours in WORKS_ON to represent the Hours attribute of the relationship type. The primary key of the WORKS_ON relation is the combination of the foreign key attributes {Essn, Pno}. The propagate (CASCADE) option for the referential triggered action should be specified on the foreign keys in the relation corresponding to the relationship R, since each relationship instance has an existence dependency on each of the entities it relates. This can be used for both ON UPDATE and ON DELETE. Step 6: Mapping of Multivalued Attributes For each multivalued attribute • Create a new relation • Primary key of R is the combination of that Atrribute and PK of the attached entity • If the multivalued attribute is composite, include its simple components In our example, we create a relation DEPT_LOCATIONS The attribute Dlocation represents the multivalued attribute LOCATIONS of DEPARTMENT, while			

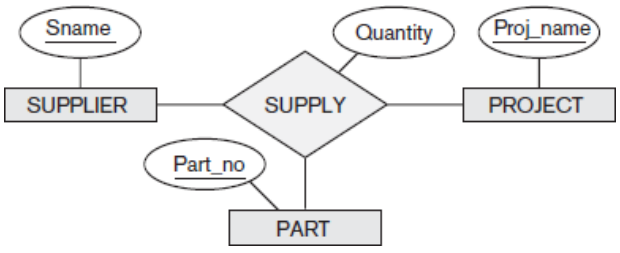
Dnumber—as foreign key—represents the primary key of the DEPARTMENT relation.

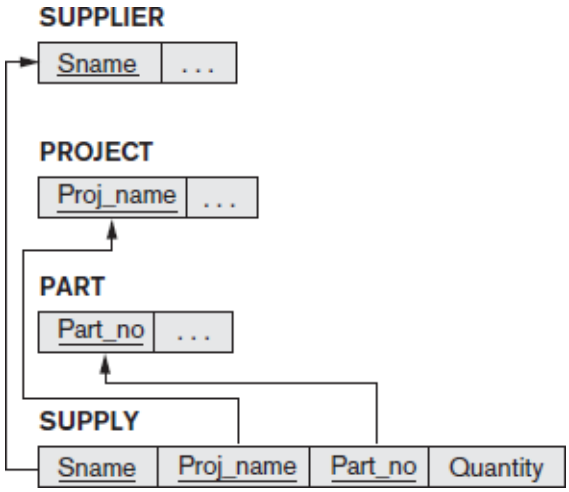
- The primary key of DEPT_LOCATIONS is the combination of {Dnumber, Dlocation}
- A separate tuple will exist in DEPT_LOCATIONS for each location that a department has
- The propagate (CASCADE) option for the referential triggered action should be specified on the foreign key in the relation R corresponding to the multivalued attribute for both ON UPDATE and ON DELETE.



Step 7: Mapping of N-ary Relationship Types

- For each n-ary relationship type R
- Create a new relation S to represent R
- Include primary keys of participating entity types as foreign keys
- Include any simple attributes as attributes
- The primary key of S is usually a combination of all the foreign keys that reference the relations representing the participating entity types.
- For example, consider the relationship type SUPPLY.This can be mapped to the relation SUPPLY whose primary key is the combination of the three foreign keys {Sname, Part_no, Proj_name}.





Question	3	Explain in detail all the operations from set theory in relational algebra with examples for each.	10	CO2	L2
Scheme			2.5M*4		
Solution		UNION: The result of this operation, denoted by R U S, is a relation that includes all tuples that are either in R or in S or in both R and S. Duplicate tuples are eliminated. INTERSECTION: The result of this operation, denoted by R ∩ S, is a relation that includes all tuples that are in both R and S. SET DIFFERENCE (or MINUS): The result of this operation, denoted by R – S, is a relation that includes all tuples that are in R but not in S. Example: Consider the following two relations: STUDENT & INSTRUCTOR			

STUDENT

Fn	Ln
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

INSTRUCTOR

Fname	Lname
John	Smith
Ricardo	Browne
Susan	Yao
Francis	Johnson
Ramesh	Shah

STUDENT ∪ INSTRUCTOR

Fn	Ln
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert
John	Smith
Ricardo	Browne
Francis	Johnson

STUDENT ∩ INSTRUCTOR

Fn	Ln
Susan	Yao
Ramesh	Shah

STUDENT – INSTRUCTOR

Fn	Ln
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

INSTRUCTOR − STUDENT

Fname	Lname
John	Smith
Ricardo	Browne
Francis	Johnson

The CARTESIAN PRODUCT (CROSS PRODUCT) Operation

The CARTESIAN PRODUCT operation—also known as CROSS PRODUCT or CROSS JOIN denoted by × is a binary set operation, but the relations on which it is applied do *not*

		have to be union compatible. This set operation produces a new element by combining every member (tuple) from one relation (set) with every member (tuple) from the other relation (set) Example: Consider two relations STUDENT(SNO, FNAME, LNAME) and DETAIL(ROLLNO, AGE) below: SNO FNAME LNAME 1 Albert Singh 2 Nora steve ROLLNO AGE 5 18 9 21 On applying CROSS PRODUCT on STUDENT and DETAIL: STUDENT × DETAILS SNO FNAME LNAME ROLLNO AGE 1 Albert Singh 5 18 1 Albert Singh 9 21 2 Nora steve 5 18 2 Nora steve 9 21			
Question	4	Consider the following database : Suppliers(sID, sName, address) Parts(pID, pName, colour) Catalog(sID, pID, price) Write relational algebra queries for the following: i. Find all suppliers who supply both "ABC" and "XYZ" parts. ii. List the names and colors of all the parts supplied by the supplier "PQR". iii. For every supplier find the number of parts that they supply.	10	CO2	L3
Scheme			3M+ 3.5M+ 3.5M		
Solution		(i) $\pi_{SID,SNAME}\left(\pi_{SID}\ (\pi_{PID}(\sigma_{PNAME="ABC"\ AND\ PNAME="XYZ"}\ )PARTS)\right) \bowtie CATALOG) \bowtie SUPPLIERS$ (ii) $PartsbyPQR \leftarrow \pi_{SID,PID}(CATALOG * \left(\sigma_{SNAME="PQR"}\ (SUPPLIERS)\right))$ $RESULT \leftarrow \pi_{PNAME,COLOR}\ (PARTS * PartsbyPQR)$ (iii) $\pi_{PID}\ (CATALOG * PARTS)$			
Question	5	Consider the following relation schema Works (Pname, Cname, Salary) Lives (Pname, Street, City) Located-in	10	CO3	L3

		(Cname, City) Where Pname = Person name, Cname = Company name. Write the SQL queries for the following: (i) Make a list of all company names and their cities if an employee whose last name is 'Scott' is working for them. (ii) Retrieve people working for both "ABC" and "XYZ" companies. (iii) Find average salary, max salary and min salary paid in "ABC" company			
Scheme			3.5M+ 3.5M+ 3M		
Solution		(i) Select w.Cname, l.City From Works w, Located-in l Where w.Cname=l.Cname And Pname= "% Scott"; (or) Select Cname, city From located-in Where Cname IN (select Cname From works Where Cname= "% Scott") ; (ii) select Pname From works Where Cname= "ABC" and Cname="XYZ"; (iii) Select Cname, MAX(SALARY),MIN(SALARY),AVG(SALARY) From Works Where Cname= "ABC";			
Question	6a	Explain with syntax the six clauses of an SQL SELECT statement? Which of the six clauses are required and which are optional?	5	CO3	L2
Scheme			4M+1M		
Solution		**SELECT** <attribute and function list> **FROM** <table list> [**WHERE** <condition>] [**GROUP BY** <grouping attribute(s)>] [**HAVING** <group condition>] [**ORDER BY** <attribute list>]; • The SELECT clause lists the attributes or functions to be retrieved.			

		• The FROM clause specifies all relations (tables) needed in the query, including joined relations, but not those in nested queries. • The WHERE clause specifies the conditions for selecting the tuples from these relations, including join conditions if needed. • GROUP BY specifies grouping attributes, whereas HAVING specifies a condition on the groups being selected rather than on the individual tuples. • Finally, ORDER BY specifies an order for displaying the result of a query. Mandatory clauses: Select, From, Where Optional: Group By, Having, Order By			
Question	6b	Write a note on types of view implementation.	5	CO3	L2
Scheme			2.5M+ 2.5M		
Solution		The problem of efficiently implementing a view for querying is complex. Two main approaches have been suggested. □ One strategy, called query modification, involves modifying or transforming the view query (submitted by the user) into a query on the underlying base tables. For example, the query SELECT Fname, Lname FROM WORKS_ON1 WHERE Pname='ProductX'; would be automatically modified to the following query by the DBMS: SELECT Fname, Lname FROM EMPLOYEE, PROJECT, WORKS_ON WHERE Ssn=Essn AND Pno=Pnumber AND Pname='ProductX'; The disadvantage of this approach is that it is inefficient for views defined via complex queries that are time-consuming to execute, especially if multiple queries are going to be applied to the same view within a short period of time. □ The second strategy, called view materialization, involves physically creating a temporary view table when the view is first queried and keeping that table on the assumption that other queries on the view will follow. In this case, an efficient strategy for automatically updating the view table when the base tables are updated must be developed in order to keep the view up-to-date. Techniques using the concept of incremental update have been developed for this purpose, where the DBMS can determine what new tuples must be inserted, deleted, or modified in a materialized view table when a database update is applied to one of the defining base tables.			

	The view is generally kept as a materialized (physically stored) table as long as it is being queried. If the view is not queried for a certain period of time, the system may then automatically remove the physical table and recompute it from scratch when future queries reference the view.			