CMR
INSTITUTE
OF TECHNOLOGY

SCHEME OF EVALUATION

Internal Assessment Test I – Sept. 2016

Sub:	**DATABASE MANAGEMENT SYSTEMS**			**Code:**	10CS54
Date: 08/09/2016	Duration: 90 mins	Max Marks: 50	**Sem: 5 A**	**Branch:**	CSE

Note: ***Answer any four full questions from part A. Part B is compulsory.*** ***(5 X 10 = 50)***

Question No.		Description	Marks Split up		Total Marks
1.	a)	✓ Data model concept. ✓ Categories of data model.	1 4	5M	10M
	b)	✓ Diagram of 3 schema architecture ✓ Explanation	2 3	5M	
2.		✓ Diagram ✓ Explanation	4 6		10M
3.	a)	✓ Cardinality ratio definition ✓ Cardinality ratio example ✓ Participation definition and types ✓ Example	1 1.5 1 1.5	5M	10M
	b)	✓ Weak Entity type ✓ Identifying entity type and relationship ✓ ER Notation ✓ Role of Partial Key	1.5 1.5 1 1	5M	
4.	a)	✓ Relation Schema definition ✓ Characteristics of Relations	1 5	6M	10M
	b)	✓ Set theoretic operations ✓ Explanation of any two + Examples	1 3	4M	
5.		✓ Domain Constraint	2		10M
		✓ Key Constraint ➢ Super key and keys ➢ Candidate key and Primary key	1+1+1		
		✓ Not Null Constraint	1		
		✓ Entity Integrity Constraint	1		
		✓ Referential Integrity Constraint ➢ Foreign key	1+2		
6.	a)	i. Expression for i ii. Expression for ii iii. Expression for ii iv. Expression for iv	1.5 1.5 1.5 1.5	6M	10M
	b)	✓ ITEM X COMPANY ✓ ITEM * COMPANY	2 2	4M	
7.		✓ ER diagram either Employee or College Database			10M

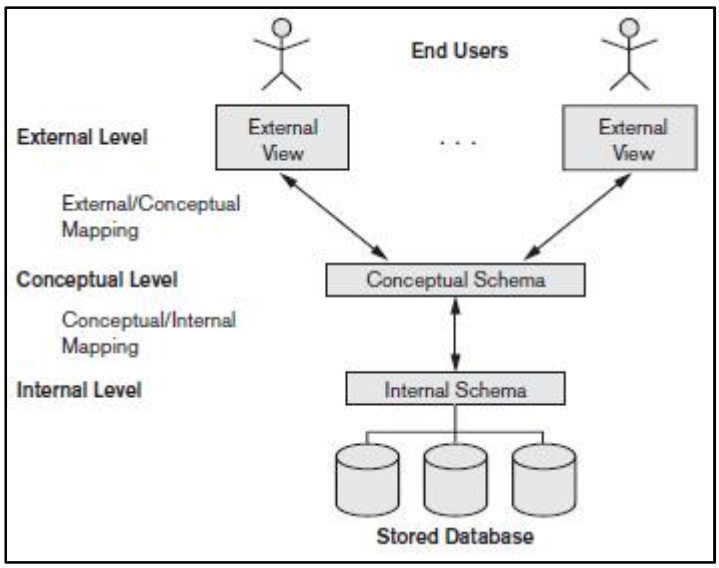
SOLUTIONS

PART –A

1a) Explain the concept of a data model? Explain different categories of data models.

A data model is a collection of concepts that can be used to describe the structure of a database.
It provides the necessary means to achieve data abstraction so that different users can perceive data at their preferred level of detail. Structure of a database corresponds to the data types, relationships, and constraints that apply to the data. Most data models also include a set of basic operations for specifying retrievals and updates on the database.
The following are the categories of the data models.
High-level or conceptual data models provide concepts that are close to the way many users perceive data. Conceptual data models use concepts such as entities, attributes, and relationships. An entity represents a real-world object or concept, such as an employee or a project from the mini-world that is described in the database. An attribute represents some property of interest that further describes an entity, such as the employee's name or salary. A relationship among two or more entities represents an association among the entities, for example, a works-on relationship between an employee and a project.
Low-level or physical data models provide concepts that describe the details of how data is stored on the computer storage media. Concepts provided by low-level data models are generally meant for computer specialists, not for end users. Physical data models describe how data is stored as files in the computer by representing information such as record formats, record orderings, and access paths. An access path is a structure that makes the search for particular database records efficient.
Between these two extremes is a class of **representational (or implementation) data models**, which provide concepts that may be easily understood by end users but that are not too far removed from the way data is organized in computer storage. Representational data models hide many details of data storage on disk but can be implemented on a computer system directly.

1b) with a neat diagram, explain the Three – schema architecture.



The three-schema architecture is a convenient tool with which the user can visualize the schema levels in a database system. The goal of the three-schema architecture, is to separate the user applications from the physical database. In this architecture, schemas can be defined at the following three levels:

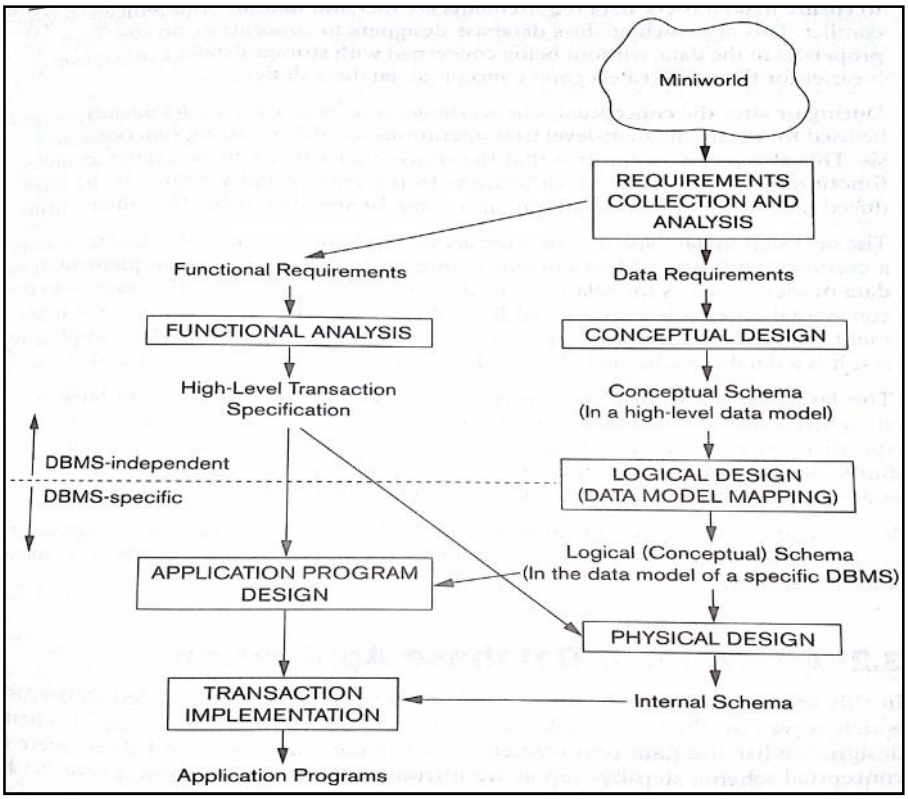
a) **Internal Level:** The internal level has an internal schema, which describes the physical storage structure of the database. The internal schema uses a physical data model and describes the complete details of data storage and access paths for the database.

b) **Conceptual level:** The conceptual level has a conceptual schema, which describes the structure of the whole database for a community of users. The conceptual schema hides the details of physical storage structures and concentrates on describing entities, data types, relationships, user operations, and constraints. Usually, a representational data model is used to describe the conceptual schema when a database system is implemented. This implementation conceptual schema is often based on a conceptual schema design in a high-level data model.

c) **External or view level:** The external or view level includes a number of external schemas or user views. Each external schema describes the part of the database that a particular user group is interested in and hides the rest of the database from that user group. As in the previous level, each external schema is typically implemented using a representational data model, possibly based on an external schema design in a high-level data model.

Mapping: The three schemas are only descriptions of data; the stored data that actually exists is at the physical level only. In a DBMS based on the three-schema architecture, each user group refers to its own external schema. Hence, the DBMS must transform a request specified on an external schema into a request against the conceptual schema, and then into a request on the internal schema for processing over the stored database. If the request is database retrieval, the data extracted from the stored database must be reformatted to match the user's external view. The processes of transforming requests and results between levels are called mappings.

2) Explain with a neat diagram the different phases of database design.



The above diagram depicts the different phases of database design process. The first step shown is requirements collection and analysis. During this step, the database designers interview

prospective database users to understand and document their data requirements. The result of this step is a concisely written set of users' requirements. These requirements should be specified in as detailed and complete a form as possible. In parallel with specifying the data requirements, it is useful to specify the known functional requirements of the application. These consist of the user-defined operations (or transactions) that will be applied to the database, and they include both retrievals and updates.

Once all the requirements have been collected and analyzed, the next step is to create a conceptual schema for the database, using a high-level conceptual data model. This step is called conceptual design. The conceptual schema is a concise description of the data requirements of the users and includes detailed descriptions of the entity types, relationships, and constraints; these are expressed using the concepts provided by the high-level data model. Because these concepts do not include implementation details, they are usually easier to understand and can be used to communicate with nontechnical users. The high-level conceptual schema can also be used as a reference to ensure that all users' data requirements are met and that the requirements do not include conflicts. This approach enables the database designers to concentrate on specifying the properties of the data, without being concerned with storage details.

During or after the conceptual schema design, the basic data model operations can be used to specify the high-level user operations identified during functional analysis. This also serves to confirm that the conceptual schema meets all the identified functional requirements.

The next step in database design is the actual implementation of the database, using a commercial DBMS. Most current commercial DBMSs use an implementation data model—such as the relational or the object database model—so the conceptual schema is transformed from the high-level data model into the implementation data model. This step is called logical design or data model mapping, and its result is a database schema in the implementation data model of the DBMS.

Finally, the last step is the physical design phase, during which the internal storage structures, access paths, and file organizations for the database files are specified. In parallel with these activities, application programs are designed and implemented as database transactions corresponding to the high-level transaction specifications.

3a) What are structural constraints of a relationship type. Explain each with examples.

Cardinality ratio and participation constraints, taken together are the structural constraints of a relationship type.

The **cardinality ratio** for a binary relationship specifies the maximum number of relationship instances that an entity can participate in.

For example, consider a binary relationship type WORKS_FOR between Department and Employee entity types, DEPARTMENT:EMPLOYEE is of cardinality ratio 1:N, meaning that each department can be related to numerous employees, but an employee can be related to (work for) only one department.

The possible cardinality ratios for binary relationship types are 1:1, 1:N, N:1, and M:N.

The binary relationship MANAGES which relates a department entity to the employee who manages that department; the cardinality ratio is 1:1. This represents the constraint that an employee can manage only one department and that a department has only one manager.

The relationship type WORKS_ON between Employee entity and the Project entity that he works for, is of cardinality ratio M:N, representing that an employee can work on several projects and a project can have several employees.

The **participation constraint** specifies whether the existence of an entity depends on its being related to another entity via the relationship type. There are two types of participation constraints—total and partial.

If a company policy states that every employee must work for a department, then an employee entity can exist only if it participates in a WORKS_FOR relationship instance. Thus, the participation of EMPLOYEE in WORKS_FOR is called total participation, meaning that every entity in "the total set" of employee entities must be related to a department entity via WORKS_FOR. Total participation is also called existence dependency.

On the other hand, we do not expect every employee to manage a department, so the participation of EMPLOYEE in the MANAGES relationship type is partial, meaning that some or "part of the set of" employee entities are related to a department entity via MANAGES, but not necessarily all.

3b) What is a weak entity type? Explain the role of partial key in design of weak entity type.

Entity types that do not have key attributes of their own are called weak entity types. Entities belonging to a weak entity type are identified by being related to specific entities from another entity type in combination with some of their attribute values. We call this other entity type the **identifying or owner entity type** and we call the relationship type that relates a weak entity type to its owner the **identifying relationship** of the weak entity type. A weak entity type always has a total participation constraint (existence dependency) with respect to its identifying relationship, because a weak entity cannot be identified without an owner entity.

Consider the entity type DEPENDENT, related to EMPLOYEE, which is used to keep track of the dependents of each employee. The attributes of DEPENDENT are Name (the first name of the dependent), BirthDate, Sex, and Relationship (to the employee). Two dependents of two distinct employees may, by chance, have the same values for Name, BirthDate, Sex, and Relationship, but they are still distinct entities. They are identified as distinct entities only after determining the particular employee entity to which each dependent is related. Each employee entity is said to own the dependent entities that are related to it.

In ER diagrams, both a weak entity type and its identifying relationship are distinguished by surrounding their boxes and diamonds with double lines.

A partial key, which is the set of attributes, is used to uniquely identify weak entities that are related to the same owner entity. The partial key attribute is underlined with a dashed or dotted line.

4a) What is a relation schema? Explain characteristics of Relations.

A relation schema R, denoted by R(A1, A2, . . ., An), is made up of a relation name R and a list of attributes A1, A2, . . ., An. Each attribute Ai is the name of a role played by some domain D in the relation schema R. A relation schema is used to describe a relation; R is called the name of this relation. D is called the domain of Ai and is denoted by dom(Ai).

Characteristics of Relations

Ordering of Tuples in a Relation: A relation is defined as a set of tuples. Mathematically, elements of a set have no order among them; hence tuples in a relation do not have any particular order. However, in a file, records are physically stored on disk so there always is an order among the records. This ordering indicates first, second, i^{th}, and last records in the file. Similarly, when we display a relation as a table, the rows are displayed in a certain order.

Ordering of Values within a Tuple, and an Alternative Definition of a Relation: According to the preceding definition of a relation, an n-tuple is an ordered list of n values, so the ordering of values in

a tuple and hence of attributes in a relation schema definition is important. However, *at a logical level, the order of attributes and their values are not really important as long as the correspondence between attributes and values is maintained.*

An alternative definition of a relation can be given, making the ordering of values in a tuple unnecessary. In this definition, a relation schema R = {A1, A2, . . ., An} is a set of attributes, and a relation r(R) is a finite set of mappings r = {t1, t2, . . ., tm}, where each tuple t_i is a mapping from R to D, and D is the union of the attribute domains; that is, D = dom(A1) U dom(A2) U. . .U dom(An). In this definition, t[Ai] must be in dom(Ai) for 1<=**i**<=n for each mapping t in r. Each mapping t_i is called a tuple.

***Values in the Tuples*:** Each value in a tuple is an atomic value; that is, it is not divisible into components within the framework of the basic relational model. Hence, composite and multi valued attributes are not allowed. Much of the theory behind the relational model was developed with this assumption in mind, which is called the first normal form assumption.

The values of some attributes within a particular tuple may be unknown or may not apply to that tuple. A special value, called null, is used for these cases. In general, we can have several types of null values, such as "value unknown," "value exists but not available," or "attribute does not apply to this tuple.

Interpretation of a Relation: The relation schema can be interpreted as a declaration or a type of assertion. For example, the relation schema of the STUDENT(Name, SSN, HomePhone, Address, Age, GPA) can be asserted as, a student entity has a Name, SSN, HomePhone, Address, Age, and GPA. Each tuple in the relation can then be interpreted as a fact or a particular instance of the assertion.

An alternative interpretation of a relation schema is as a predicate; in this case, the values in each tuple are interpreted as values that satisfy the predicate.

4b) List the set theoretic operations in relational data model. Explain any two with examples.

The following are the set theoretic operations are used to merge the elements of two sets in various ways in relational algebra,

- UNION
- INTERSECTION
- SET DIFFERENCE

When these operations are adapted to relational databases, the two relations on which any of the above three operations are applied must have the same type of tuples; this condition is called union compatibility.

UNION: The result of this operation, denoted by R ∪ 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.

Example:

R:

Name
Ram
Raju
Rakesh

S:

Name
Ram
Rajesh
Ramu

R ∪ S:

Name
Ram
Raju
Rakesh
Rajesh
Ramu

INTERSECTION: The result of this operation, denoted by R ∩ S, is a relation that includes all tuples that are in both R and S.

Example:

R:

Name
Ram
Raju
Rakesh

S:

Name
Ram
Rajesh
Ramu

R ∩ S:

Name
Ram

SET DIFFERENCE: 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:

R:

Name
Ram
Raju
Rakesh

S:

Name
Ram
Rajesh
Ramu

R − S:

Name
Raju
Rakesh

5) Explain schema-based constraints in relational model.

Domain Constraints: Domain constraints specify that the value of each attribute A must be an atomic value from the domain dom(A). The data types associated with domains typically include standard numeric data types for integers (such as short-integer, integer, long-integer) and real numbers (float and double-precision float). Characters, fixed-length strings, and variable-length strings are also available, as are date, time, timestamp, and money data types.

Key Constraints and Constraints on Null: A relation is defined as a set of tuples. By definition, all elements of a set are distinct; hence, all tuples in a relation must also be distinct. This means that no two tuples can have the same combination of values for all their attributes. Usually, there are other subsets of attributes of a relation schema R with the property that no two tuples in any relation state r of R should have the same combination of values for these attributes. Suppose that we denote one such subset of attributes by SK; then for any two distinct tuples t1 and t2 in a relation state r of R, we have the constraint that

$$t_1[SK] \neq t_2[SK]$$

Any such set of attributes SK is called a super key of the relation schema R. A super key SK specifies a uniqueness constraint that no two distinct tuples in a state r of R can have the same value for SK. A super key can have redundant attributes, however, so a more useful concept is that of a key, which has no redundancy.

A key K of a relation schema R is a super key of R with the additional property that removing any attribute A from K leaves a set of attributes K' that is not a super key of R. Hence, a key is a minimal super key—that is, a super key from which we cannot remove any attributes and still have the uniqueness constraint hold.

In general, a relation schema may have more than one key. In this case, each of the keys is called a candidate key. One among of the candidate keys is designated as the primary key of the relation, which is used to uniquely identify a tuple in the relation.

Another constraint on attributes specifies whether null values are or are not permitted. For example, if every STUDENT tuple must have a valid, non-null value for the Name attribute, then Name of STUDENT is constrained to be NOT NULL.

Entity Integrity constraint: The entity integrity constraint states that no primary key value can be null. This is because the primary key value is used to identify individual tuples in a relation; having null values for the primary key implies that we cannot identify some tuples. If two or more tuples had null for their primary keys, we might not be able to distinguish them.

Referential Integrity and Foreign Keys: The referential integrity constraint is specified between two relations and is used to maintain the consistency among tuples of the two relations. The referential integrity constraint is specified between two relations and is used to maintain the consistency among tuples of the two relations.

To define referential integrity more formally, we first define the concept of a foreign key. The conditions for a foreign key, given below, specify a referential integrity constraint between the two relation schemas R1 and R2. A set of attributes FK in relation schema R1 is a foreign key of R1 that references relation R2 if it satisfies the following two rules:

1. The attributes in FK have the same domain(s) as the primary key attributes PK of R2; the attributes FK are said to reference or refer to the relation R2.
2. A value of FK in a tuple t1 of the current state r1(R1) either occurs as a value of PK for some tuple t2 in the current state r2(R2) or is null. In the former case, we have $t_1[FK] = t_2[PK]$, and we say that the tuple t1 references or refers to the tuple t2. R1 is called the referencing relation and R2 is the referenced relation.

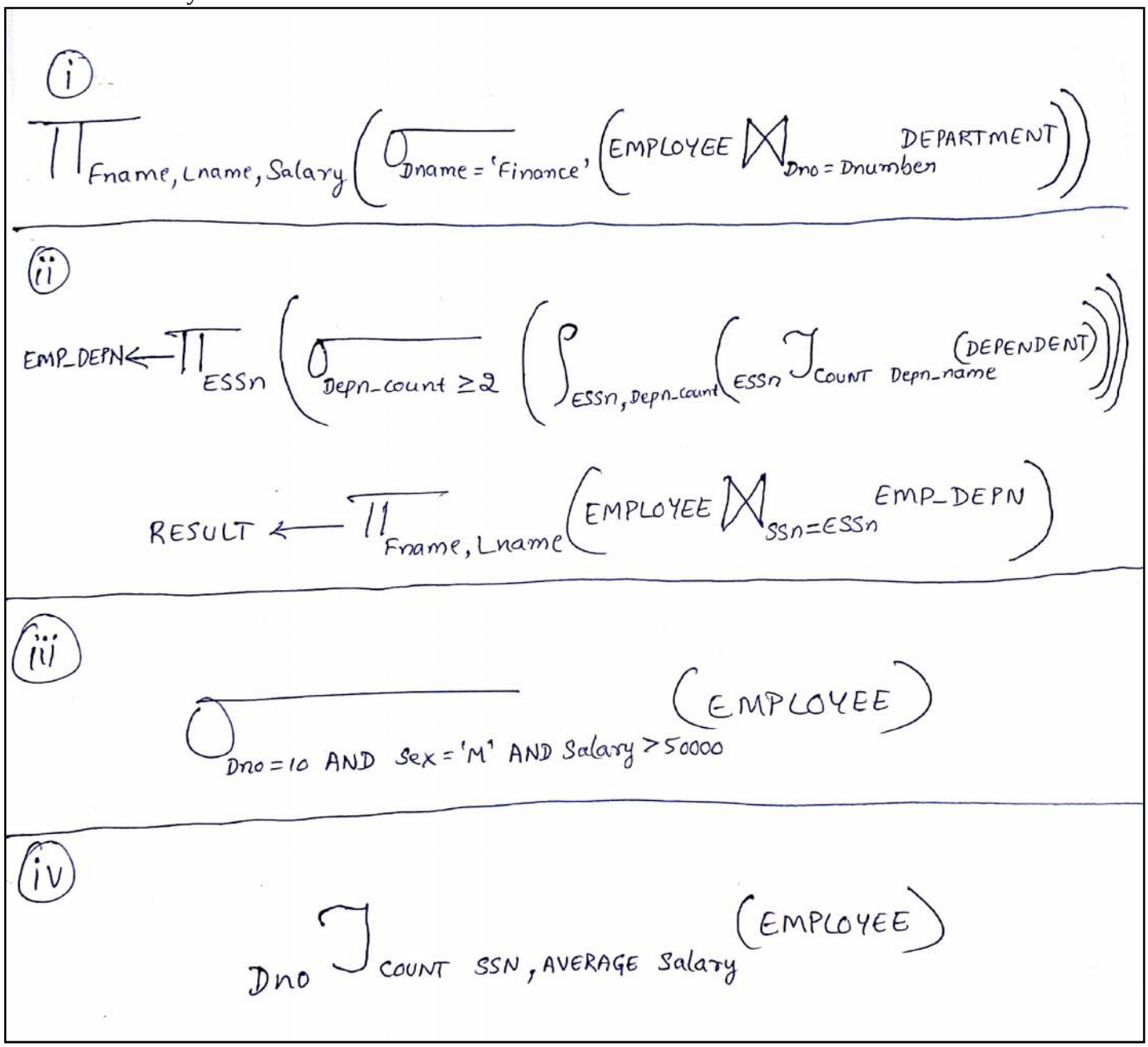
If these two conditions hold, a referential integrity constraint from R1 to R2 is said to hold.

6a) Consider the following schema for a COMPANY database:

EMPLOYEE(Fname, Lname, SSN, DOB, Sex, Salary, Super_Ssn, Dno)
DEPARTMENT(Dnumber, Dname, Mgr_ssn, mgr_str_date)
DEPT_LOCATION(Dnumber, Dlocation)
PROJECT(Pname, Pnumber, Plocation, Dno)
WORKS-ON(Essn, Pno, Hours)
DEPENDENT(Essn, Depn_name, Sex, Relationship)

Write the relation algebra expressions for the following.

i. Retrieve the names and salary of all employees who works for 'Finance' department.
ii. List the names of all employees with two or more dependents.
iii. List all male employees from Dno = 10 and earn less than 50000
iv. Retrieve department number, the number of employees in the department and their average salary.



6b) The tables ITEM and COMPANY are given below. Write the results of ITEM X COMPANY and ITEM * COMPANY

ITEM X COMPANY

ITEMID	ITEMNAME	CID	CID	CNAME	CITY
10	Chocolate	21	21	Parle	Bangalore
10	Chocolate	21	22	Britannia	Mysore
10	Chocolate	21	23	Pepsi	Bangalore
11	Cakes	22	21	Parle	Bangalore
11	Cakes	22	22	Britannia	Mysore
11	Cakes	22	23	Pepsi	Bangalore
12	Biscuit	21	21	Parle	Bangalore
12	Biscuit	21	22	Britannia	Mysore
12	Biscuit	21	23	Pepsi	Bangalore

ITEM * COMPANY

ITEMID	ITEMNAME	CID	CNAME	CITY
10	Chocolate	21	Parle	Bangalore
11	Cakes	22	Britannia	Mysore
12	Biscuit	21	Parle	Bangalore