Chapter 1

ER Model and Relational Model

LEARNING OBJECTIVES

- *Data model*
- *Schemas*
- *Three-schema architecture*
- *ER model*
- *Types of attributes*
- *Mapping cardinality*
- *Complex attributes*
- *Entity types, entity sets and value sets*
- *Weak entity set*
- *Relational database*
- *NULL in tuples*
- *Inherent constraint*
- *Referential and entity integrity constraint*

Introduction

A database is a collection of related data. By data, we mean facts that can be recorded and that have implicit meaning.

Example: Consider the names, telephone numbers and addresses of the people. We can record this data in an indexed address book and store it as Excel file on a hard drive using a personal computer. This is a collection of related data with an implicit meaning and hence is a database.

A database management system (DBMS) is a collection of programs that enables users to create and maintain a database. The DBMS is a general-purpose software system that facilitates the processes of defining, constructing, manipulating and sharing databases among various users and applications.

1. Defining the database involves specifying the data types, structures, and constraints for the data to be stored in the database.
2. Constructing the database is the process of storing the data itself on some storage medium that is controlled by the DBMS
3. Manipulating a database includes such functions as querying the database to retrieve specific data, updating the database to reflect changes.
4. Sharing a database allows multiple users and programs to access the database concurrently.
5. Fundamental characteristics of the database approach is that it provides some level of data abstraction by hiding details of data storage that are not needed by users.

Data Model

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 abstraction.

Schemas

In any data model, it is important to distinguish between the description of the database and the database itself. The description of a database is called the *database schema*, which is specified during database design and is not expected to change frequently.

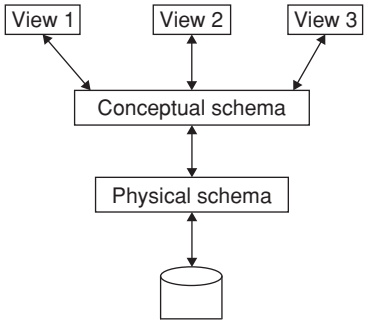
The actual data in a database may change frequently, for example the student database changes every time we add a student or enter a new grade for a student. The data in the database at a particular moment in time is called a *database state* or *snapshot*. It is also called the *current set of occurrences* or *instances in the database*.

The distinction between database schema and database state is very important. When we define a new database, we specify its database schema only to the DBMS. At this point, the corresponding database state is the empty state with no data. We get the initial state of the database when the database is first loaded with the initial data. The DBMS stores the description of the schema constructs and constraints, also called the *metadata* in the DBMS catalog so that DBMS software can refer to the schema whenever it needs. The schema is sometimes called the *intension*, and the database state an *extension* of the schema.

Three-schema Architecture

The goal of the three-schema architecture is to separate the user applications and the physical database.

Levels of Abstraction



1. 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.
2. 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.
3. The internal level has an internal schema, which describes the physical storage structures of the database. It describes the complete details of data storage and access paths for the database.

ER Model

Entity relationship model is a popular high-level conceptual data model. This model and its variations are used for the conceptual design of database applications, and many database design tools employ its concepts. ER model describes data as entities, relationships and attributes. The basic object that the ER model represents is an entity.

Entity

It is an object that exists and is distinguishable from other objects
(or)
Entity is a "thing" in the real world with an independent existence.
(or)
An entity is something that has a distinct, separate existence, although it need not be a material existence. In particular, abstractions and legal functions are usually regarded as entities. In general, there is no presumption that an entity is animate.

1. An object with a physical existence
   Example: A particular person, car, house, employee.
2. An object with a conceptual existence
   Example: A company, a job, a university course. Each entity has attributes, the particular properties that describe it.
   Example: An employee entity can be described by employee's name, age, address, salary and job.

Entity Set

Set of entities of same type that shares the same properties.

Example: All persons, all companies etc.

Example: Entity sets of customer and loan

Table 1 *Customer Entity Set*

Customer-id	Cust-name	Cust-street	City
C-143	John	MG Road	Sec.bad
C-174	Mary	SP Road	Hyd.bad
C-183	Tony	KD Road	Sec.bad
C-192	Satya	SG Road	Eluru

Table 2 *Loan Entity Set*

Loan-no	Amount
L-30	$3000
L-31	$4000
L-32	$3500
L-33	$4500
L-34	$5000

An entity is represented by a set of attributes and by a descriptive properties possessed by all members of an entity set.

Types of Attributes

1. Simple versus composite
2. Single valued versus multivalued
3. Stored versus derived.

Composite attribute: Composite attributes can be divided into smaller subparts, which represent more basic attribute that has their own meaning (Figure 1).

Example: A common example is Address, it can be broken down into a number of subparts such as street address, city, postal code; street address is further broken down into Number, street Name and Apartment number.

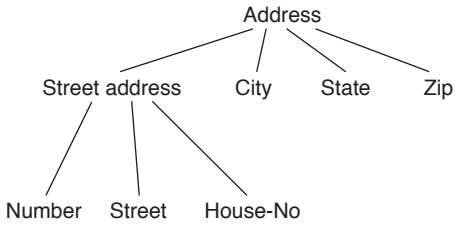


Figure 1 A hierarchy of composite attributes.

Street address is a composite attribute. Attributes that are not divisible are called simple (or) atomic attributes.

Single-valued versus multivalued attributes: Most attributes have a single value for a particular entity, such attributes are called *single-valued attribute*.

Example: Age is a single-valued attribute

Multivalued attributes: An attribute can have a set of values for the same entity.

Example: College degrees attribute for a person
Example: Name is also a multivalued attribute (Figure 2).

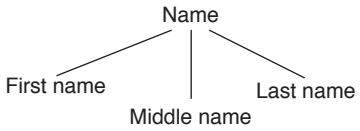


Figure 2 Multivalued attribute.

Stored versus derived attributes: Two (or) more attribute values are related.

Example: Age can be derived from a person's date of birth. The age attribute is called derived attribute and is said to be derivable from the DOB attribute, which is called a stored attribute.

Domain: The set of permitted values for each attribute.

Example: A person's age must be in the domain {0-130}

Relationship Sets

A relationship is an association among several entities. Relationship sets that involve two entity sets are binary. Generally, most relationships in databases are binary. Relationship sets may involve more than two entity sets.

Example: Employee of a bank may have responsibilities at multiple braches, with different jobs at different branches, then there is a ternary relation between employee, job and branch.

Mapping Cardinality

For a binary relationship set, mapping cardinality must be:

1. One-to-one
2. One-to-many
3. Many-to-one
4. Many-to-many

One-to-one: An entity in A is associated with at most one entity in B and an entity in B is associated with at most one entity in A (Figure 3).

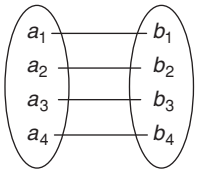


Figure 3 One-to-one relationship set.

One-to-many: An entity in A is associated with any number of entities in B. But an entity in B is associated with at most one entity in A (Figure 4).

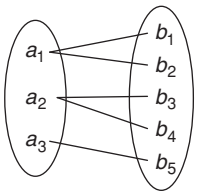


Figure 4 One-to-many relationship set.

Many-to-one: An entity in A is associated with at most one entity in B. But an entity in B can be associated with any number of entities in A (Figure 5).

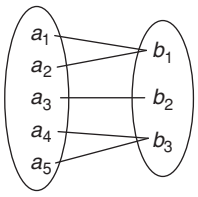


Figure 5 Many-to-one relationship set.

Many-to-many: An entity in A is associated with any number of entities in B. But an entity in B can be associated with any number of entities in A (Figure 6).

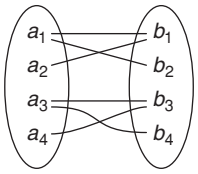
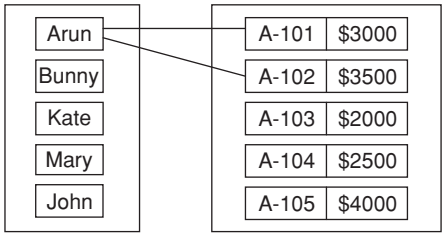


Figure 6 Many-to-many relationship set.

Example: One customer can have multiple accounts

Customer(c-Name) (Acc. no, Amount)

Table 3 *Example of One-to-many Relationship Set*



In Table 3, many-to-one relationship is not possible.

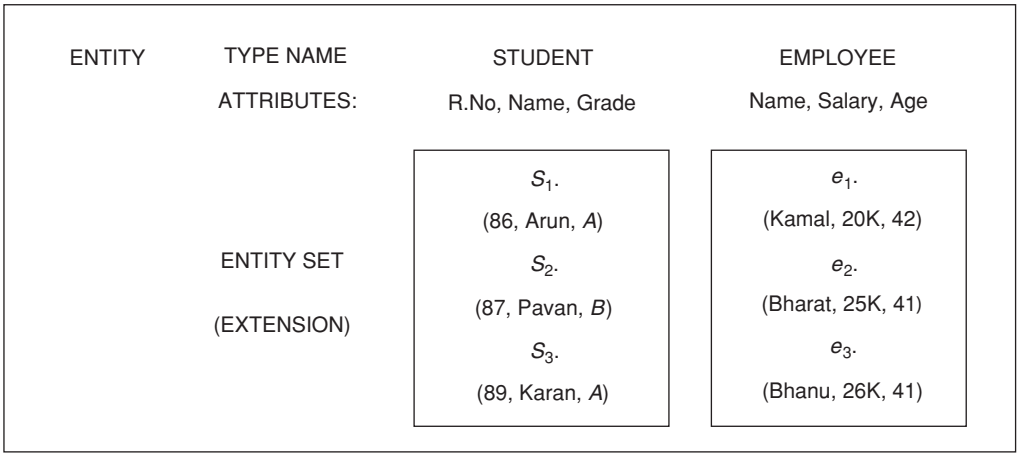
Complex Attributes

Composite and multivalued attributes can be nested in an arbitrary way. We can represent arbitrary nesting by grouping components of a composite attribute between parentheses () and separating the components with commas, and by displaying multivalued attributes between braces { }. Such attributes are called complex attributes.

Example: A person can have more than one residence and each residence can have multiple phones, an attribute AddressPhone for a person can be specified as shown below. {AddressPhone ({Phone (Areacode, phoneNumber)}, Address (StreetAddress (StreetNumber, streetName, ApartmentNumber), city, state, zip))}

Entity types, entity sets and value sets

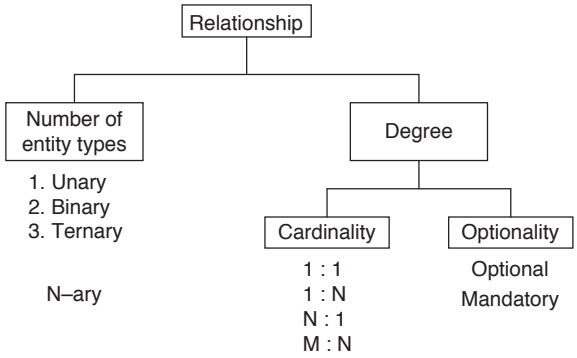
An entity type defines a collection of entities that have the same attributes. Each entity type in the database is described by its name and attribute. The following figure shows two entity types, named STUDENT and EMPLOYEE and a list of attributes for each



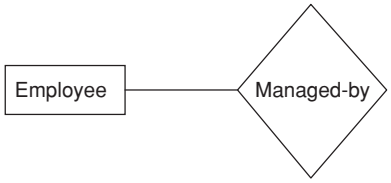
The collection of all entities of a particular entity type in the database at any point in time is called an *entity set*.

Types of relations

1. Unary relation.
2. Binary relation.
3. Ternary relation.
4. Quadnary relation.
5. N-ary relation



Unary relation If a relationship type is between entities in a single entity type then it is called a unary relationship type.

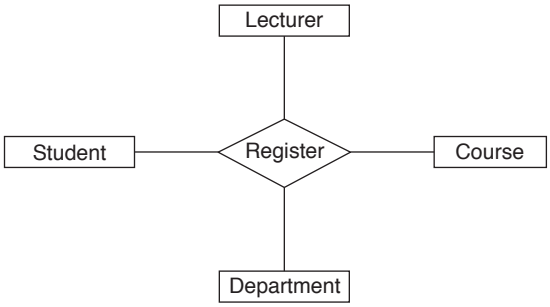


In employee entity, we will have all the employees including 'manager', this relation indicates, employees are managed by manager.

Binary relation If a relationship type is between entities in one type and entities in another type then it is called a *binary relation*, because two entity types are involved in the relation.

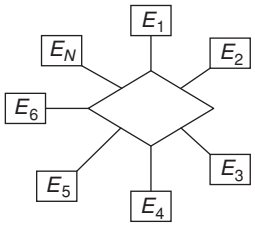
The above relation indicates that customers purchased product (or) products are purchased by customers.

Quadnary relation If a relationship type is among entities of four different types, then it is called *quadnary relation*.



In the above ER-diagram, any two entities can have a relation.

N-ary relation '*N*' number of entities will participate in a relation, and each entity can have a relation with all the other entities.

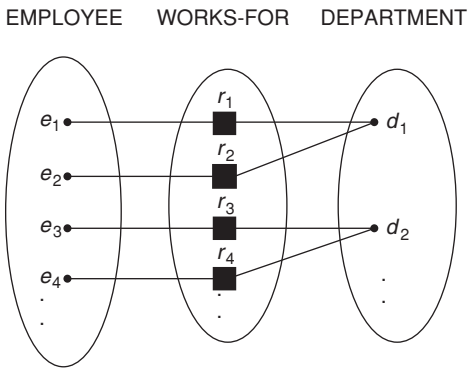


Cardinality Ratio and Participation Constraints

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

1. The participation constraint specifies whether the existence of an entity depends on its being related to another entity via the relationship type. This constraint specifies the minimum number of relationship instances that each entity can participate in, and is some times called the *minimum cardinality constraint*.
2. There are two types of participation constraints:
   a. Total participation
   b. Partial participation

Example: If a company policy states that every employee must work for a department, then an employee entity can exist only if it participates (or) works for at least one department.



Every entity in the EMPLOYEE set must be related to a DEPARTMENT entity via WORK-FOR. Total participation is also called *existence dependency*. If we do not expect every employee to manage a department, so the participation of EMPLOYEE in the relationship type is partial, means not necessarily all employees' entities are related to some department entity.

Cardinality ratio and participation constraints are taken together as the structural constraints of a relationship type. In ER diagrams, total participation is displayed as a double line connecting the participating entity type to the relationship, whereas partial participation is represented by a single line.

ER Diagrams (Figure 7 and 8)

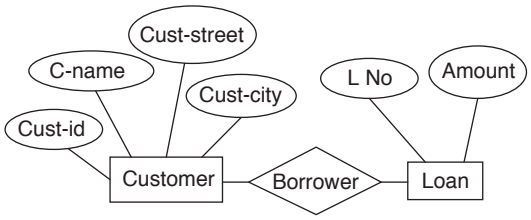


Figure 7 ER diagram.

Notations:

→ Represents entity

→ Represents relationship sets

→ Represents links between attributes to entity sets and entity sets to relationship sets

→ Represents attributes

→ Represents multivalued attributes

→ Represents primary key attribute

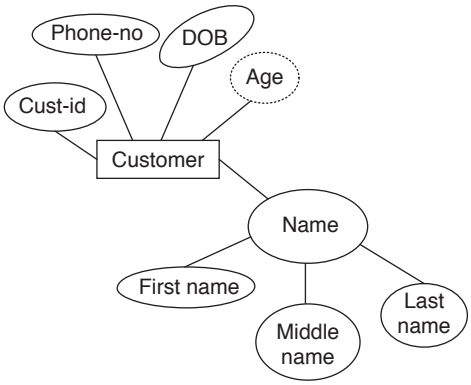
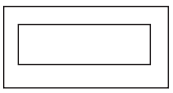


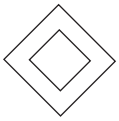
Figure 8 ER diagram.

→ Derived attribute

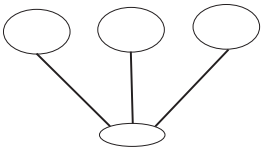
→ Multivalued attribute



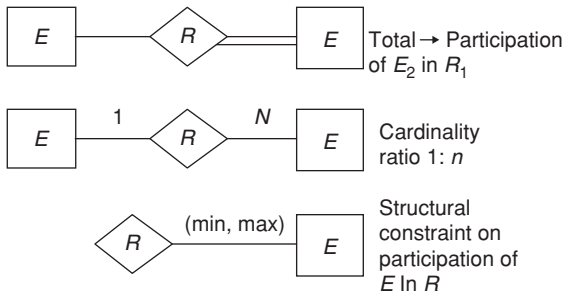
→ Weak entity



→ Weak entity set relation



→ Composite attribute



Cardinality Constraints

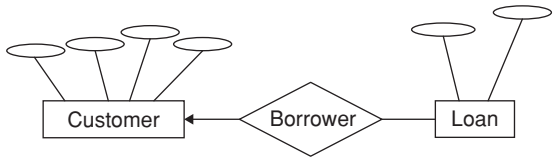
One-to-one

Each entity of one entity set is related to at most one entity of the other set. Only one matching record exists between two tables.

Example: Assume each owner is allowed to have only one dog and each dog must belong to one owner. The own relationship between dog and owner is one-to-one. One-to-one relationships can often combine the data into one table.

Examples:

1. One birdfeeder is located in one place in the yard.
2. One state has one governor.
3. One yard has one address.
4. One patient has one phone number.
2. One student has one ID.



One customer is associated to one loan via borrower

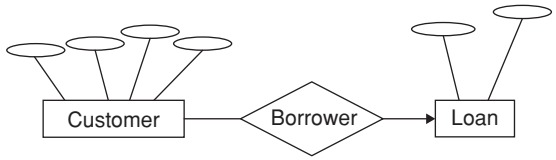
One-to-many

Examples:

1. One birdfeeder is visited by many birds.
2. One student can have many degrees.
3. One Book can be written by many authors.
4. One yard contains many bird feeders.
5. One patient has many prescriptions.

In the one-to-many relationship, a loan is associated with one customer via borrower.

Many-to-one

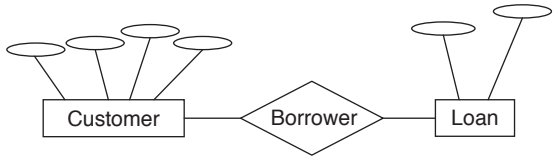


A customer is associated with at most one loan via borrower.

Many-to-many

Examples:

1. Many students are taught by many teachers.
2. Many patients are treated by many doctors.
3. Many medications are taken by many patients.
4. Many customers buy many products.
5. Many books are written by many authors.



Customer is associated with several loans and loan is associated with several customers.

ER Diagram with a Ternary Relationship

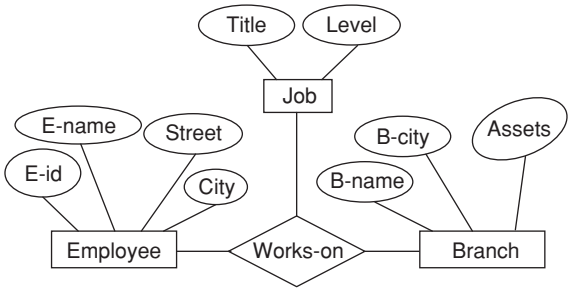


Figure 9 ER diagram with a ternary relationship.

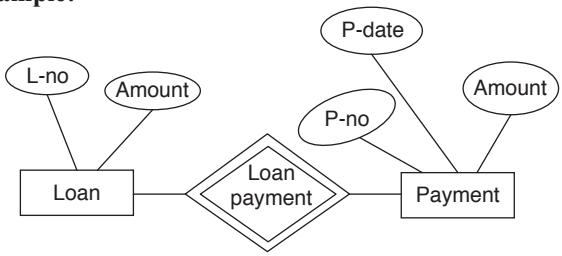
Weak Entity Set

An entity set that does not have a primary key is called *weak entity set*.

Weak entity set is represented by → 

Underline the primary key of a weak entity with a dashed line.

Example:



Relational Database

Relational Model

The relational model represents the database as a collection of relations. When a relation is thought of as a table of values, each row in the table represents a collection of related data values. Each row in the table represents a fact that typically corresponds to a real-world entity. The table name and

column names are used to help in interpreting the meaning of the values in each row.

In formal relational model terminology, a row is called a *tuple*, a column header is called an *attribute*, and the table is called a *relation*.

Domain

A domain is a set of atomic values. A common method of specifying a domain is to specify a data type from which the data values forming the domain are drawn. It is also useful to specify a name for the domain, to help in interpreting its values

Example:

1. *Set of telephone numbers*: The set of valid numbers in a particular country.
2. *Employee id numbers*: The set of valid employee numbers in a company.
3. *Names*: The set of character strings that represent names of persons.
4. *Grade-point average*: Possible values of computed grade point averages, each must be real (floating point) number between 0 and 4
5. *Research department names*: The set of research department names in a specialization, such as computer science, chemistry and applied mathematics.
6. *Research department codes*: The set of Research department codes, such as CS, CHE, AM.

The preceding is called *logical definitions of domains*. The data type for research department names is set of character strings that represent valid department names. A domain is thus given a name, data type, and format. Additional information for interpreting the values of a domain can also be given, for example a numeric domain such as person weights should have the units of measurements, such as kilograms or pounds.

1. The relational model is often described as having the following three aspects:
   - *Structural aspect*: The data in the database is perceived by the user as tables.
   - *Integrity aspect*: Those tables has to satisfy certain integrity constraints.
   - *Manipulative aspect*: The operators available to the user for manipulating those tables, for purposes of data retrieval, these operators derive tables form tables, the most important operators are 'SELECT', 'PROJECT' and JOIN.

Relation Schema

A relation schema 'R' denoted by $R(A_1, A_2, \ldots A_n)$ is made up of a relation name R and a list of attributes $A_1, A_2, \ldots A_n$. Each Attribute A_i is the name of role played by some domain D in the relation schema R. D is called the domain of A_i and is denoted by dom(A_i).

A relation schema is used to describe a relation and R is called the name of this relation. The degree of a relation is the number of attributes 'n' of its relation schema.

Example:

A relation schema of degree '7', which describes an employee is given below:

EMPLOYEE (Name, EId, HomePhone, Address, Office phone, Age, Salary)

Using the data type of each attribute, the definition is written as:

EMPLOYEE (Name: String, EId: INT, Homephone: INT, Address: String, OfficePhone: String, Age: Real, Salary: INT)

For this relation schema, EMPLOYEE is the name of the relation, which has '7' attributes.

2. A relation 'r' of the relation schema $R(A_1, A_2 \ldots A_n)$, also denoted by $r(R)$, is a set of n-tuples. $r = \{t_1, t_2 \ldots t_m\}$. Each n-tuple 't' is an ordered list of n values $t = <V_1, V_2 \ldots N_n>$, where each value V_i, $1 \le i \ldots n$, is an element of dom(A_i) or is a special null value.
3. The ith value in tuple t, which corresponds to the attribute A_i, is referred to as $t[A_i]$.
4. The following figure shows EMPLOYEE relation. Each type in a relation represents a particular employee entity. We display the relation as a table where each tuple is shown as a row and each attribute corresponds to a column header, indicating a role or interpretation of the values in that column. Null values represent attributes whose values are unknown or do not exist for some individual EMPLOYEE tuple.

Employee

Name	EId	Home Phone	Address	Office Phone	Age	Salary
Mahesh	30-01	870-223366	Warangal	NULL	35	40 k
Ramesh	30-02	040-226633	Hyderabad	NULL	36	40 k
Suresh	30-03	040-663322	Kolkata	040-331123	35	42 k
Dinesh	30-04	040-772299	Bangalore	040-321643	36	40 k

Fig: The attributes and tuples of a relation EMPLOYEE. The earlier definition of a relation can be restated as follows:
A relation $r(R)$ is a mathematical relation of degree 'n' on the domains dom(A_1), …, dom(A_n), which is a subset of the Cartesian product of the domains that define R:

$$r(R) \subset (\text{dom}(A_1) \times \text{dom}(A_2) \ldots \times \text{dom}(A_n))$$

Characteristics of Relations

There are certain characteristics that make a relation different from a file or a table.

Ordering of tuples in a relation: A relation is defined as a set of tuples. Tuples in a relation do not have any particular order. In a file, records are stored on disk in order. 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.

Tuple ordering is not part of a relation definition, because a relation attempts to represent facts at a logical or abstract level. Many logical orders can be specified on a relation. Tuples in the EMPLOYEE relation could be logically ordered by values of Name or by EID or by Age or by some other attribute. When a relation is implemented as a file or displayed as a table, a particular ordering may be specified on the records of the file or the rows of the table.

NULL in Tuples

Each value in a tuple is an atomic value; that is, it is not divisible into components. Hence, composite and multivalued attributes are not allowed. This model is sometimes called the *flat relational model*. Multivalued attributes must be represented by separate relations, and composite attributes are represented only by their simple component attributes in the basic relational model.

NULLS are used to represent the values of attributes that may be unknown or may not apply to tuple. For example, some student tuples have null in their office phones because they do not have an office. In this case, the meaning behind NULL is not applicable. If a student has a NULL for home phone, it means either he/she does not have a home phone or he/she has one but we do not know it, in this case the meaning of NULL is 'Unknown'.

Relational Model Constraints

In a relational database, there will be many relations, and the tuples in those relations are related in various ways. There are many restrictions or constraints on the actual values in a database state.

Constraints on database can generally be divided into three main categories as follows:

1. Constraints that are inherent in the data model, we call them *inherent model-based constraints*.
2. Constraints that can be directly expressed in the schemes of the data model, by specifying them in the DDL (Data Definition Language). We call these *schema-based constraints*.
3. Constraints that cannot be directly expressed in the schemas of the data model, and they must be expressed and enforced by the application programs are *application-based constraints*.

Inherent Constraint

The constraint that a relation cannot have duplicate tuples is an *inherent constraint*. Another important category of constraints is data dependencies, which include 'functional dependencies' and 'multivalued dependencies'. They are used mainly for testing the 'goodness' of the design of a relational database and are utilized in a process called normalization.

Schema-based Constraints

1. Domain constraints
2. Key constraints
3. Constraints on nulls (Not null constraint)
4. Entity integrity constraints
5. Referential integrity constraints
6. Unique constraint
7. Check constraint

Domain constraints

Domain constraints specify that within each tuple, the value of each attribute 'X' must be an atomic value from the domain dom(X).

The data types associated with domains include standard numeric data types for integers

1. Short integer
2. Integer
3. Long integer
4. Real numbers
   - Float
   - Double-precision float
5. Characters
6. Booleans
7. Fixed-length strings
8. Variable-length strings
9. Date, time, time stamp
10. Money data types

Key constraints

A relation is a set of tuples. All elements of a set are distinct; hence, all tuples in a relation must also be distinct. This means no two tuples can have the same combination of values for all their attributes.

1. 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 of these attributes.

   Suppose that we denote one subset of attributes by 'SK', then for two distinct tuples t_1 and t_2 in a relation state 'r' of R, we have the following constraint:

   $$t_1[\text{SK}] = t_2[\text{SK}]$$

2. Any such set of attributes SK is called a *super key* of the relation schema *R*.

   SK specifies a uniqueness constraint that no two distinct tuples in any state *r* or *R* can have the same value for SK.
3. Every relation has at least one default super key, the set of all its attributes. *A* key, '*K*' of a relation schema *R* is a super key of *R* with the additional property that removing any attributes '*X*' from *K* leaves a set of attributes *K*' that is not a super key of *R* any more.

A key satisfies the following two constraints:

1. Two distinct tuples in any state of the relation cannot have identical values for all the attributes in the key.
2. A super key from which we cannot remove any attributes and still have the uniqueness constraints mentioned in above condition is known as a *minimal super key*.

   The first condition applies to both keys and super keys. The second condition is required only for keys.

Example: Consider the employee relation in Page no. 9. The attribute set {EId} is a key of employee because no two employee tuples can have the same value for EId.

Any set of attributes that include EId will form a super key.

1. {EId, Homephone, Name}
2. {EId, Age, Salary}
3. {Name, EId, Address}

However, the super key {EId, Name, Age} is not a key of EMPLOYEE, because removing Name or age or both from the set leaves us with a super key. Any super key formed from a single attributes is also a key. A key with multiple attributes must require all its attributes to have the uniqueness property.

A relation schema may have more than one key. In that case, each of the keys is called a *candidate key*.

Example: Employee relation has three candidate keys.
{Name, EId, Homephone}

One of the candidate keys is chosen as primary key of the relation.

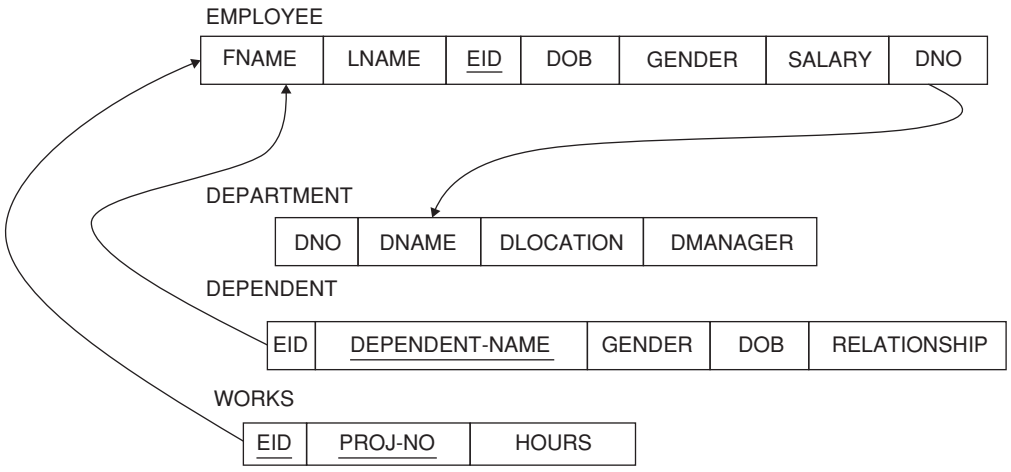
Another constraint on attributes specifies whether null values are permitted in tuples or not. If we want some tuples to have a valid (or) non-null value, we need to use NOT NULL constraint on that attribute.

Referential and entity integrity constraint

The entity Integrity constraint states that no primary key value can be null. If we have NULL values in the primary key column, we cannot identify some tuples in a relation.

1. Key constraints and entity Integrity constraints are specified on individual relations
2. Referential integrity constraint is specified between relations and used to maintain the consistency among tuples in the two relations.
3. Referential Integrity constraints states that a tuple in one relation that refers to another relation must refer to an existing tuple in that relation.
4. To understand the concept of Referential Integrity, first we have to understand the concept of FOREIGN KEY.
5. Suppose we have two relations R_1 and R_2. A set of attributes FK in relation schema R_1 is a foreign key of R_1 that references relation R_2 if it satisfies the following two rules:
   - The attributes in FK have the same domains as the primary key attributes PK of R_2, FK will have to refer to PK.
   - A value of FK in a tuple t_1 of the current state $r_1(R_1)$ either occurs as a value of PK for some tuple t_2 in the current state $r_2(R_2)$ or is null. We have t_1[FK] = t_2[PK], and we say that the tuple t_1 references to the tuple t_2. In this definition, R_1 is called the *referencing relation* and R_2 is the *referenced relation*.
6. A *foreign key* can refer to its own relation. We can diagrammatically display referential integrity constraints by drawing a directed Arc from each foreign key to the relation it references. The arrow head may point to the primary key of the referenced relation.

Example:



7. *Referential integrity rule*: The database must not contain any unmatched foreign key values.

If '*B*' References '*A*', then A must exist.

NOT NULL constraint

1. NOT NULL constraint restricts a column from having a NULL value. NOT NULL constraint can be applied to any column in a table.
2. We cannot give NULL values under that column
3. NOT NULL Constraint enforces a column to contain a proper value.
4. This constraint cannot be defined at table level.

Example: CREATE TABLE
student(RNo:INT Name:varchar(70)NOTNULL age:INT)
Suppose a row is inserted into the following table,

Insert into student values <11, NULL, 20>

In the schema, we enforced NOT NULL constraint on Name column, means Name cannot have NULL value, when the above insert command is executed, the system gives, NOT NULL constraint violation.

UNIQUE constraint

The column on which UNIQUE constraint is enforced should not have any duplicate values.

1. UNIQUE constraint can be enforced on any column except the primary key column.
2. By default primary key column will not accept any duplicate values that are handled by key constraint.
3. UNIQUE constraint can be applied at column level or table level.

Example: CREATE TABLE
student (RNo:INT Name:varchar(60) Grade:CHAR(1))
Assume that the table contains following tuples

Student

R no.	Name	Grade
11	Sita	B
12	Anu	A
13	Bala	A

Suppose the following tuple is inserted into the student table.

1. Insert into student values <14, 'Anu', '*B*'>
2. UNIQUE constraint is enforced on Name column, in the student table we have 'Anu', and again the new tuple contains name 'Anu', this Insert command violates the UNIQUE constraint.

CHECK constraint

This constraint is used to restrict a value of a column between a range.

1. When a value is inserted into particular column, before storing that, a check will be performed to see whether the values lie within the specified range.
2. If the value entered is out of range, it will not accept and violation happens.
3. It is like checking a condition before saving data into a column.

Example: Create table student (RNo:INT CHECK (Rno>0)
Name:varchar(60)
Dept:varchar(4))
Suppose the following tuple is inserted into student table.

1. Insert into student values <-4, 'Bhanu', 'CS'>
2. CHECK constraint is enforced on RNo column, the RNo should be greater than '0', but '-4' is given.
3. CHECK constraint is violated.

Creating table from a table: A view is called a *derived table* (or) *virtual table*, because the data stored in views is taken from already existing tables.

A view can also be defined as a logical subset of data from one or more tables.

Syntax:

```
CREATE    view view-name As
SELECT    column-names
FROM      table-name
WHERE     condition
```

Example: Consider the following table "sales".

Sales

Order-Id	Order Name	Previous Balance	Customer
21	Order 3	3000	Ana
22	Order 4	1000	Adam
23	Order 5	3000	Brat
24	Order 6	2000	John
25	Order 7	2000	Ana
26	Order 8	4000	Ana

Query to create a view:

```
CREATE view sales-view As
SELECT    *
FROM      Sales
WHERE     customer = 'Ana'
```

1. The data fetched from select statement will be stored in an object called 'sales-view'.
2. To display the contents stored in view, execute the following statement.

```
SELECT    *
FROM      Sales-view
```

Removal of specific rows:
Consider the following SQL query:

```
Delete *
FROM Sales
```

The above query will delete all the tuples from sales.

To remove specific rows, we have to specify the condition in WHERE clause.

Consider the table "sales" given in the above example.

Remove the rows from sales table whose previous balance is 3000.

SQL query:

```
Delete      *
FROM        Sales
WHERE       Previous-balance = 3000
```

Output:

Order Id	Order Name	Previous Balance	Customer
22	Order 4	1000	Adam
24	Order 6	2000	John
25	Order 7	2000	Ana
26	Order 8	4000	Ana

Referential actions: Referential Integrity can be violated if the value of any foreign key refers to a tuple that does not exist in the referenced relation.

When certain violations occur, we need to perform some alternate action. Those actions are as follows:

1. ON DELETE CASCADE
2. ON UPDATE CASCADE
3. ON DELETE SET NULL
4. ON DELETE SET DEFAULT

Example: Consider the given database:

SUPPLIERS:

Supplier Number	Supplier Name	Status	City
SN1	Suma	30	Hyderabad
SN2	Hari	20	Chennai
SN3	Anu	10	Hyderabad
SN4	Mahesh	20	Bombay
SN5	Kamal	30	Delhi

PARTS:

Part number	Part name	Colour	Weight	City
PN1	X	Red	13.0	Chennai
PN2	Y	Green	13.5	Bombay
PN3	X	Yellow	13.2	Hyderabad
PN4	Y	Green	14.1	Calcutta
PN5	Z	Red	14.3	Hyderabad
PN6	Z	Blue	14.2	Bombay

PROJECT:

Project Number	Project Name	City
PJ1	Display	Chennai
PJ2	OCR	Bombay
PJ3	RAID	Chennai
PJ4	SORTER	Hyderabad
PJ5	EDS	Chennai
PJ6	Tape	Bombay
PJ7	Console	Hyderabad

SHIPMENTS:

Supplier Number	Part Number	Project Number	Quantity
SN1	PN1	PJ1	300
SN1	PN1	PJ4	400
SN2	PN3	PJ1	350
SN2	PN3	PJ2	450
SN2	PN3	PJ3	640
SN2	PN3	PJ4	320
SN2	PN3	PJ5	330
SN2	PN3	PJ6	520
SN2	PN3	PJ7	480
SN2	PN5	PJ2	460
SN3	PN3	PJ1	440
SN3	PN4	PJ2	410
SN4	PN6	PJ3	310
SN4	PN6	PJ7	320
SN5	PN2	PJ2	340
SN5	PN2	PJ4	350
SN5	PN5	PJ5	360
SN5	PN5	PJ7	370
SN5	PN6	PJ2	380
SN5	PN1	PJ4	420
SN5	PN3	PJ4	440
SN5	PN4	PJ4	450
SN5	PN5	PJ4	400
SN5	PN6	PJ4	410

Consider the following statement:

```
DELETE FROM SUPPLIER
WHERE SUPPLIER – NUMBER = 'SN1'
```

It deletes the supplier tuple for supplier 'SN1'. The database has some other tables which have 'SN1' tuple (Shipments table). The application does not delete those suppliers, then it will find a violation, and an exception will be raised.

An alternate approach is possible, one that might be preferable in some cases, and that is for the system to perform an appropriate 'compensating action' that will guarantee that the overall result does still satisfy the constraint. In the example, the compensating action would be for the system to delete the shipments for supplier SN1 "automatically".

We can achieve this effect by extending the foreign key as indicated below:

```
CREATE TABLE SHIPMENT{.....}.....
FOREIGN KEY {SUPPLIER – NUMBER}
REFERENCES
SUPPLIER ON DELETE CASCADE
```

The specification ON DELETE CASCADE defines a delete rule for this particular foreign key, and the specification CASCADE is the referential action for that delete rule. The meaning of these specifications is that a DELETE operation on the suppliers relvar will 'Cascade" to delete matching tuples (if any) in the shipments relvar as well.

Same procedure is applied for all the referential actions.

Triggers

Triggers are precompiled procedures that are stored along with the database and invoked automatically whenever some specified event occurs.

Suppose we have a view called HYDERABAD - SUPPLIER defined as follows:

```
CREATE VIEW HYDERABAD-SUPPLIER
AS SELECT SUPPLIER - NUMBER, SUPPLIER-
NAME, STATUS
FROM SUPPLIER
WHERE CITY = 'HYDERABAD',
```

Normally, if the user tries to insert a row into this view, SQL will actually insert a row into the underlying base table SUPPLIERS with CITY value whatever the default is for the CITY column. Assuming that default is not Hyderabad, the net effect is that the new row will not appear in the view; therefore, let us create a triggered procedure as follows:

```
CREATE TRIGGER HYDERABAD -
SUPPLIER - INSERT
INSTEAD OF INSERT ON HYDERABAD
- SUPPLIER
REFERENCING NEW ROW AS R
FOR EACH ROW
INSERT INTO SUPPLIERS (SUPPLIER -
NUMBER, SUPPLIER - NAME, STATUS, CITY)
VALUES (R. SUPPLIER - NUMBER, R.
SUPPLIER - NAME, R. STATUS, 'HYDERABAD');
```

Inserting a row into the view will now cause a row to be inserted into the underlying base table with CITY value equal to Hyderabad inserted of the default value.

In general, CREATE TRIGGER specifies, among other things, an event, a condition, and an action.

The event is an operation on the database ("INSERT ON HYDERABAD - SUPPLIER" in the example)

1. The "condition" is a Boolean expression that has to evaluate to TRUE in order for the action to be executed.
2. The 'action' is the triggered procedure ("INSERT INTO SUPPLIERS …)
3. The event and condition together are sometimes called the *triggering event*. The combination of all three (event, condition, and action) is usually called a trigger.
4. Possible events include INSERT, DELETE, UPDATE, reaching end-of-transaction (COMMIT) reaching a specified time of day, exceeding a specified elapsed time, violating a specified constraint, etc.
5. A database that has associated triggers is sometimes called an active database.

Base Table Constraints

SQL-base table constraints are specified on either CERATE TABLE or ALTER TABLE. Each such constraint is a candidate key constraint, a foreign key constraint, or a CHECK constraint.

Candidate keys: An SQL candidate key definition takes one of the following two forms:

```
PRIMARY KEY (< column name comma list>)
UNIQUE (< column name comma list>)
The following example illustrates base table constraints of all three kinds:
CREATE TABLE SHIPMENTS
(SUPPLIER-NUMBER. SUPPLIER-NUMBER
NOT NULL, PART - NUMBER PART-NUMBER
NOT NULL, QUANTITY NOT NULL
PRIMARY KEY (SUPPLIER - NUMBER,
PART NUMBER)
FOREIGN KEY (SUPPLIER-NUMBER)
REFERENCES SUPPLIERS
ON DELETE CASCADE
ON UPDATE CASCADE,
FOREIGN KEY (PART-NUMBER)
REFERENCES PARTS
ON DELETE CASCADE
ON UPDATE CASCADE
CHECK(QUANTITY ≤ QUANTITY (0) AND
QUANTITY ≤ QUANTITY (1000));
```

A check constraint of the form CHECK (< column name > IS NOT NULL) can be replaced by a simple NOT NULL specification in the definition of the column.

EXERCISES

Practice Problems I

Directions for questions 1 to 20: Select the correct alternative from the given choices.

1. Consider the following two tables T_1 and T_2. Show the output for the following operations:

Table T_1

P	Q	R
11	a	6
16	b	9
26	a	7

Table T_2

A	B	C
11	b	7
26	c	4
11	b	6

What is the number of tuples present in the result of given algebraic expressions?

(i) $T_1 \bowtie_{T1.P = T2.A} T_2$

(A) 2 (B) 3
(C) 4 (D) 5

(ii) $T_1 \bowtie_{T1.Q = T2.B} T2$

(A) 2 (B) 3
(C) 4 (D) .5

(iii) $T_1 \bowtie_{(T1.p = T2.A \text{ AND } T1.R = T2.C)} T_2$

(A) 1 (B) 2
(C) 3 (D) .4

2. Suppose $R_1(A, B)$ and $R_2(C, D)$ are two relation schemas. Let R_1 and R_2 be the corresponding relation instances. B is a foreign key that refers to C in R_2. If data in R_1 and R_2 satisfy referential integrity constraints, which of the following is true?

(A) $\pi_B(R_1) - \pi_c(R_2) = \phi$

(B) $\pi_C(R_2) - \pi_B(R_1) = \phi$

(C) $\pi_B(R_1) - \pi_C(R_2) \neq \phi$

(D) Both A and B

3. Consider the following relations:

A, B and C

A

Id	Name	Age
12	Arun	60
15	Shreya	24
99	Rohit	11

B

Id	Name	Age
15	Shreya	24
25	Hari	40
98	Rohit	20
99	Rohit	11

C

Id	Phone	Area
10	2200	02
99	2100	01

How many tuples does the result of the following relational algebra expression contain? Assume that the scheme of ($A \cup B$) is the same as that of A.

$(A \cup B) \bowtie_{A.Id>40 \ v \ c.Id<15} C$

(A) 6 (B) 7
(C) 8 (D) 9

4. Consider the relations A, B and C given in Question 3. How many tuples does the result of the following SQL query contain?

```
SELECT    A.Id
FROM      A
WHERE     A.Age>
ALL       (SELECT B.Age
FROM      B
WHERE     B.Name = 'Arun')?
```

(A) 0 (B) 1
(C) 2 (D) 3

5. Consider a database table T containing two columns X and Y each of type integer. After the creation of the table, one record ($X = 1$, $Y = 1$) is inserted in the table. Let MX and MY denote the respective maximum value of X and Y among all records in the table at any point in time. Using MX and MY, new records are inserted in the table 128 times with X and Y values being MX + 1, 2 * MY + 1, respectively. It may be noted that each time after the insertion, values of MX and MY change. What will be the output of the following SQL query after the steps mentioned above are carried out?

SELECT Y FROM T WHERE $X = 7$;?

(A) 15 (B) 31
(C) 63 (D) 127

6. Database table by name loan records is given below:

Borrower	Bank Manager	Loan Amount
Ramesh	Sunderajan	10000
Suresh	Ramgopal	5000
Mahesh	Sunderajan	7000

What is the output of the following SQL Query SELECT count (*)

FROM((SELECT Borrower, Bank-manager

FROM Loan-Records)AS S

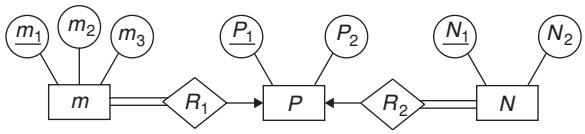
NATURAL JOIN

(SELECT Bank-manager, Loan-Amount

FROM Loan-Records) AS T);

(A) 3 (B) 4
(C) 5 (D) 6

7. Consider the following ER diagram:



What is the minimum number of tables needed to represent M, N, P, R_1, R_2?

(A) 2 (B) 3
(C) 4 (D) 5

8. Let E_1 and E_2 be two entities in an ER diagram with simple single-valued attributes. R_1 and R_2 are two relationships between E_1 and E_2, where R_1 is one-to-many and R_2 is many-to-many. R_1 and R_2 do not have any attributes of their own. What is the minimum number of tables required to represent this situation in the relational model?

(A) 2 (B) 3
(C) 4 (D) 5

9. The following table has two attributes A and C where A is the primary key and C is the foreign key referencing A with ON DELETE CASCADE.

A	C
2	4
3	4
4	3
5	2
7	2
9	5
6	4

What is the set of all tuples that must be additionally deleted to preserve referential integrity when the tuple (2, 4) is deleted?

(A) (5, 2),(7, 2) (B) (5, 2), (7, 2),(9, 5)
(C) (5, 2),(9, 5) (D) (2, 4), (7, 2)

10. Consider the following SQL query

SELECT DISTINCT $a_1, a_2, a_3 \ldots a_n$

FROM $R_1, R_2 \ldots R_m$

WHERE P

For any arbitrary predicate P, this query is equivalent to which relational algebra expression?

(A) $\Pi_{a_1, a_2, \ldots a_n}(\sigma_p R_1 \times R_2 \times \ldots \times R_m)$

(B) $\sigma_{a_1, a_2, \ldots a_n}(\sigma_p R_1 \times R_2 \times \ldots \times R_m)$

(C) $\sigma_{a_1, a_2, \ldots a_n}(\prod_p R_1 \times R_2 \times \ldots \times R_m)$

(D) $\Pi_{R_1, R_2, \ldots Rm}(\sigma_p a_1 \times a_2 \times \ldots \times (a_n)$

11. Consider the following relation schema pertaining to a student's database:

Student (Rollno, name, address)

Enroll (Rollno, courseno, coursename)

Where the primary keys are shown underlined. The number of tuples in the student and Enroll tables are 120 and 6, respectively. What are the maximum and minimum number of tuples that can be present in

(student * Enroll)

Where '*' denotes natural join?

(A) 6, 6 (B) 6, 120
(C) 120, 6 (D) 120, 120

12. A relational schema for a train reservation database is given below

Table 4 *Passenger*

Pid	P Name	Age
0	'Sachin'	65
1	'Rahul'	66
	'Sourav'	67
3	'Anil'	69

Table 5 *Reservation*

Pid	Class	Tid
0	AC	8200
1	AC	8201
2	SC	8201
5	AC	8203
1	SC	8204
3	AC	8202

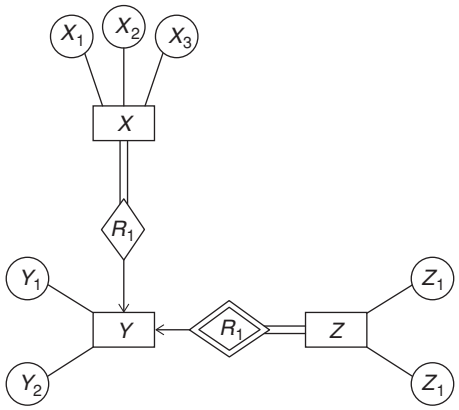
What pid's are returned by the following SQL query for the above instance of the tables?

```
SELECT    pid
FROM      Reservation
WHERE     class = 'AC' AND
          EXISTS (SELECT *
                  FROM passenger
          WHERE age > 65 AND
          Passenger.pid = Reservation.pid)
```

(A) 0, 1 (B) 1, 3
(C) 1, 5 (D) 0, 3

13. Given {customer} is a candidate key, [customer name, customer street} is another candidate key then
(A) {customer id, customer name} is also a candidate key.
(B) {customer id, customer street} is also a candidate key.
(C) {customer id, customer name, customer street} is also a candidate key.
(D) None

Common data for questions 14 and 15: Consider the following diagram,



14. The minimum number of tables needed to represent X, Y, Z, R_1, R_2 is
(A) 2 (B) 3
(C) 4 (D) 5

15. Which of the following is a correct attribute set for one of the tables for the correct answer to the above questions?
(A) $\{X_1, X_2, X_3, Y_1\}$ (B) $\{X_1, Y_1, Z_1, Z_2\}$
(C) $\{X_1, Y_1, Z_1\}$ (D) $\{M_1, Y_1\}$

16. UPDATE account SET
DA = basic * .2,
GROSS = basic * 1.3,
Where basic > 2000;
(A) The above query displays DA and gross for all those employees whose basic is ≥ 2000
(B) The above query displays DA and gross for all employees whose basic is less than 2000
(C) The above query displays DA as well as gross for all those employees whose basic is >2000
(D) Above all

17. Which of the following query transformations is correct?
R_1 and R_2 are relations C_1, C_2 are selection conditions and A_1 and A_2 are attributes of R_1
(A) $\sigma_{C1}(\sigma_{C1}(R_1)) \rightarrow \sigma_{C2}(\sigma_{C2}(R_1))$
(B) $\sigma_{C1}(\sigma_{A1}(R_1)) \rightarrow \sigma_{A1}(\sigma_{C1}(R_1))$
(C) $\pi_{A2}(\pi_{A}(R_1)) \rightarrow \pi_{A}(\pi_{A2}(R_1))$
(D) All the above

18. Consider the following query select distinct $a_1, a_2, \ldots a_n$ from $r_1, r_2 \ldots r_m$ where P for an arbitrary predicate P, this query is equivalent to which of the following relational algebra expressions:
(A) $\pi_{a_1 \ldots a_n} \sigma_p(r_1 \times r_2 \times \ldots \times r_m)$
(B) $\pi_{a_1 \ldots a_n} \sigma_p(r_1 \times r_2 \times r_3 \times \ldots . \times r_m)$
(C) $\sigma_{a_1 \ldots a_n} \pi_p(r_1 \times r_2 \times \ldots \times r_m)$
(D) $\sigma_{a_1 \ldots a_n} \pi_p(r_1 \times r_2 \times \ldots \times r_m)$

19. The relational algebra expression equivalent to the following tuple calculus expression
$\{a\}\ a \in r \wedge (a[A] = 10 \wedge a[B] = 20)$ is
(A) $\sigma_{(A = 10\ r\ B = 20)}$r
(B) $\sigma_{(A = 10)}(r) \cup \sigma_{(B = 20)}(r)$
(C) $\sigma_{(A = 10)}(r) \cap \sigma_{(B = 20)}(r)$
(D) $\sigma_{(A = 10)}(r) - \sigma_{(B = 20)}(r)$

20. Which of the following is/are wrong?
(A) An SQL query automatically eliminates duplicates.
(B) An SQL query will not work if there are no indexes on the relations.
(C) SQL permits attribute names to be repeated in the same relation
(D) All the above

Practice Problems 2

Directions for questions 1 to 20: Select the correct alternative from the given choices.

1. If *ABCDE* are the attributes of a table and *ABCD* is a super key and *ABC* is also super key then
(A) *A B C* must be candidate key
(B) *A B C* cannot be super key
(C) *A B C* cannot be candidate key
(D) *A B C* may be candidate key

2. The example of derived attribute is
(A) Name if age is given as other attribute
(B) Age if date_of_birth is given as other attribute
(C) Both (A) and (B)
(D) None

3. The weak entity set is represented by
(A) box
(B) ellipse
(C) diamond
(D) double outlined box

4. In entity relationship diagram double lines indicate
(A) Cardinality
(B) Relationship
(C) Partial participation
(D) Total participation

5. An edge between an entity set and a binary relationship set can have an associated minimum and maximum cardinality, shown in the form 1… *h* where 1 is the minimum and *h* is the maximum cardinality A minimum value 1 indicates:
(A) total participation (B) partial participation
(C) double participation (D) no participation

6. Let *R* be a relation schema. If we say that a subset *k* of *R* is a super key for *R*, we are restricting *R*, we are restricting consideration to relations *r*(*R*)in which no two district tuples have the same value on all attributes in *K*. That is if t_1 and t_2 are in *r* and $t_1 \uparrow t_2$
(A) $t_1[k] = 2t_2[K]$ (B) $t_2[K] = 2t_1[k]$
(C) $t_1[k] = t_2[k]$ (D) $t_1[k] \uparrow t_2[k]$

7. Which one is correct?
(A) Primary key ⊂ Super key ⊂ Candidate key
(B) Candidate key ⊂ Super key ⊂ Primary key
(C) Primary key ⊂ Candidate key ⊂ Super key
(D) Super key ⊂ Primary key ⊂ Candidate key

8. If we have relations *r*1(*R*1) and *r*2(*R*2), then *r*1 (*r*2 is a relation whose schema is the
(A) concatenation (B) union
(C) intersection (D) None

9. Match the following:

I	Empid	1	Multivalued
II	Name	2	Derived
III	Age	3	Composite
IV	Contact No.	4	Simple

(A) I – 4, II – 3, III – 2, Iv – 1
(B) I – 3, II – 2, III – 4, Iv – 1
(C) I – 2, II – 1, III – 4, Iv – 3
(D) I – 1, II – 3, III – 2, IV – 4

10. Match the following:

I	Double-lined ellipse	1	Multivalued attribute
II	Double line	2	Total participation
III	Double-lined box	3	Weak entity set
IV	Dashed ellipse	4	Derived attribute

(A) I – 1, II – 2, III – 3, IV – 4
(B) I – 2, II – 3, III – 4, IV – 1
(C) I – 3, II – 4, III – 2, IV – 2
(D) I – 4, II – 3, III – 2, IV –1

11. The natural join is a
(A) binary operation that allows us to combine certain selections and a Cartesian product into one operation
(B) unary operations that allows only Cartesian product
(C) query which involves a Cartesian product and a projection
(D) None

12. The number of entities participating in the relationship is known as
(A) maximum cardinality (B) composite identifiers
(C) degree (D) None

13 A minimum cardinality of 0 specifies
(A) non-participation
(B) partial participation
(C) total participation
(D) zero participation

14. What is not true about weak entity?
(A) They do not have key attributes.
(B) They are the examples of existence dependency.
(C) Every existence dependency results in a weak entity
(D) Weak entity will have always discriminator attributes

15. Which one is the fundamental operation in the relational algebra?
(A) Natural join (B) Division
(C) Set intersection (D) Cartesian product

16. For the given tables

A

X	Y
a_1	b_1
a_2	b_1
a_1	b_2
a_2	b_2

B

Y
b_1
b_2

$A \div B$ will return
(A) a_1, a_2 (B) a_1
(C) a_2 (D) None

17. The number of tuples selected in the above answer is
(A) 2 (B) 1
(C) 0 (D) 4

Common data for questions 18 and 19: Consider the following schema of a relational database.

Emp (empno, name, add)

Project (Pno, Prame)

Work on (empno, Pno)

Part (partno, Pname, qty, size)

Use (empno, pno, partno, no)

18. ((name(emp) ((name(emp ⋈ workon) displays
(i) The names of the employees who are not working in any project
(ii) The names of the employees who were working in every project.
(A) Only (i) (B) Only (ii)
(C) Both (A) and (B) (D) None

19. List the partno and names of the parts used in both the projects DBMS & MIS:

(A) $\sigma_{partno,\ pname,}{}^{(part} \bowtie {}^{(}\pi_{partno}(\sigma_{pname} = \text{“DBMS”}{}^{(project)} \bowtie {}^{use)}\cap \pi_{partno}(\sigma_{pname} = \text{“MIS”}{}^{(project)} \bowtie {}^{use)})$

(B) (partno, pname(part ⋈ ((partno ((pname = “DBMS”(project ⋈ use) ((partno(σ_{pname} = “MIS”$^{(project)} \bowtie {}^{use)}$)))

(C) $\pi_{partno,\ pname}$(part ⋈ (σ_{partno} (σ_{pname} = “DBMS”$^{(project} \bowtie {}^{use)}$ ((partno ((pname = “MIS”(project) ⋈ use)))

(D) None

20. The following query shows. SELECT job-status, sum (basic – salary) AVG (basic – salary) from employees group by job – status.

(i) It shows job status, sum, AVG of all data

(ii) It shows job status, Sum, AVG, with group by clause in use.

(A) only (i)

(B) only (ii)

(C) both (A) and (B)

(D) None

Previous Years' Questions

1. Let E_1 and E_2 be two entities in an E/R diagram, with simple single-valued attributes. R_1 and R_2 are two relationships between E_1 and E_2, where R_1 is one-to-many and R_2 is many-to-many. R_1 and R_2 do not have any attributes of their own. What is the minimum number of tables required to represent this situation in the relational model? **[2005]**

(A) 2 (B) 3

(C) 4 (D) 5

2. The following table has two attributes A and C where A is the primary key and C is the foreign key referencing A with on-delete cascade.

A	C
2	4
3	4
4	3
5	2
7	2
9	5
6	4

The set of all tuples that must be additionally deleted to preserve referential integrity when the tuple (2, 4) is deleted is: **[2005]**

(A) (3, 4) and (6, 4)

(B) (5, 2) and (7, 2)

(C) (5, 2), (7, 2) and (9, 5)

(D) (3, 4), (4, 3) and (6, 4)

3. Which of the following tuple relational calculus expression(s) is/are equivalent to $\forall t \in r\ (P(t))$?

I. $\neg \exists t \in r\ (P(t))$

II. $\exists t \notin r\ (P(t))$

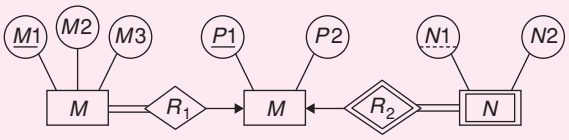
III. $\neg \exists t \in r\ (\neg P(t))$

IV. $\exists t \in r\ (\neg P(t))$ **[2008]**

(A) I only (B) II only

(C) III only (D) III and IV only

Common data for questions 4 and 5: Consider the following ER diagram:



4. The minimum number of tables needed to represent M, N, P, R_1, R_2 is? **[2008]**

(A) 2 (B) 3

(C) 4 (D) 5

5. Which of the following is a correct attribute set for one of the tables for the correct answer to the above question? **[2008]**

(A) {$M1$, $M2$, $M3$, $P1$} (B) {$M1$, $P1$, $N1$, $N2$}

(C) {$M1$, $P1$, $N1$} (D) {$M1$, $P1$}

6. Consider a relational table with a single record for each registered student with the following attributes.

1. *Registration_Num*: Unique registration number of each registered student
2. *UID*: Unique identity number, unique at the national level for each citizen
3. *BankAccount_Num*: Unique account number at the bank. A student can have multiple accounts or joint accounts. This attribute stores the primary account number
4. *Name*: Name of the student
5. *Hostel_Room*: Room number of the hostel

Which of the following options is incorrect? **[2011]**

(A) *BankAccount_Num* is a candidate key

(B) *Registration_Num* can be a primary key

(C) *UID* is a candidate key if all students are from the same country

(D) If S is a super key such that $S \cap UID$ is NULL then $S \cup UID$ is also super key.

7. Given the basic ER and relational models, which of the following is incorrect? **[2012]**

(A) An attribute of an entity can have more than one value

(B) An attribute of an entity can be composite

(C) In a row of a relational table, an attribute can have more than one value

(D) In a row of a relational table, an attribute can have exactly one value or a NULL value

8. An ER model of a database consists of entity types A and B. These are connected by a relationship R which does not have its own attribute, Under which one of the following conditions, can the relational table for R be merged with that of A? **[2017]**

(A) Relationship R is one-to-many and the participation of A in R is total.
(B) Relationship R is one-to-many and the participation of A in R is partial.
(C) Relationship R is many-to-one and the participation of A in R is total.
(D) Relationship R is many-to-one and the participation of A in R is partial.

9. In an Entity-Relationship (ER) model, suppose R is a many-to-one relationship from entity set $E1$ to entity set $E2$. Assume that $E1$ and $E2$ participate totally in R and that the cardinality of $E1$ is greater than the cardinality of $E2$.

Which one of the following is true about R? **[2018]**

(A) Every entity in $E1$ is associated with exactly one entity in $E2$.
(B) Some entity in $E1$ is associated with more than one entity in $E2$.
(C) Every entity in $E2$ is associated with exactly one entity in $E1$.
(D) Every entity in $E2$ is associated with at most one entity in $E1$.

Answer Keys

Exercises

Practice Problems 1

1. (i) B (ii) A (iii) A **2.** A **3.** B **4.** D **5.** D **6.** D **7.** B **8.** B
9. B **10.** A **11.** A **12.** B **13.** D **14.** A **15.** A **16.** C **17.** B **18.** B
19. C **20.** D

Practice Problems 2

1. D **2.** B **3.** C **4.** A **5.** A **6.** D **7.** C **8.** A **9.** A **10.** A
11. A **12.** C **13.** C **14.** A **15.** D **16.** A **17.** A **18.** A **19.** C **20.** B

Previous Years' Questions

1. B **2.** C **3.** D **4.** B **5.** A **6.** A **7.** C **8.** C **9.** A