



e-ISSN: 2319-8753 | p-ISSN: 2347-6710

IJRSET

International Journal of Innovative Research in
SCIENCE | ENGINEERING | TECHNOLOGY

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

Volume 11, Issue 4, April 2022

ISSN INTERNATIONAL
STANDARD
SERIAL
NUMBER
INDIA

Impact Factor: 8.118

9940 572 462

6381 907 438

ijirset@gmail.com

www.ijirset.com



Relational Data Model

Navaneeth Singh

UG Student, C.G.M Engineering College, Bihar, India

ABSTRACT: 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. A row represents a fact that typically corresponds to a real-world entity or relationship. The table name and column names are used to help to interpret the meaning of the values in each row.

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

Domains, Attributes, Tuples, and Relations

A domain D is a set of atomic values. By atomic we mean that each value in the domain is indivisible as far as the formal relational model is concerned. A common method of specifying a domain is to specify a data type from which the data values forming the domain are drawn.

An **ENTITY** is a “thing” or “object” in the real world that is distinguishable from all other objects. E.g. Each student in a college or class. An entity has a set of properties. E.g. Name, address, register number etc of a student. The values for some set of properties may uniquely identify an entity. E.g. Register number of a student. An entity is represented by a set of attributes. Attributes are descriptive properties possessed by each member of an entity set. E.g. Name, address, register number etc of a student. Each entity may have its own value for each attribute. For each attribute there is a set of permitted values called the **domain** or **value set** of that attribute. Formally, an attribute of an entity set is a function that maps from the entity set into a domain. An entity may have several attributes. Each entity can be described by a set of (attribute, data value) pairs, one pair for each attribute of the entity set. E.g. { (name, hayes), (Register-No, 1001), (Score, 97)}. An entity set (Table) is a set of entities of the same type that share the same properties, or attributes. E.g. Set of all persons who are customers at a given bank.

I. RELATIONAL MODEL

RELATIONAL MODEL CONCEPTS: The relational model represents both data and the relationships among those data using relation. Informally, each relation resembles a table of values. Each row in the table represents a collection of related data values. In the relational model, each row in the table represents a fact that typically corresponds to a real-world entity or relationship. The table name and column names are used to help in interpreting the meaning of the values in each row. All values in a column are of the same data type. In the formal relational model terminology, a row is called a tuple, a column header is called an attribute, and the table is called a relation. The data type describing the types of values that can appear in each column is represented by a domain of possible values. It is possible for several attributes to have the same domain.

A relation is used to represent information about any entity (such as book, publisher, author, etc.) and its relationship with other entities in the form of attributes (or columns) and tuples (or rows). The relational model includes three components:

1. **STRUCTURAL COMPONENT:** It consists of two components, namely, entity types and their relationships
2. **MANIPULATIVE COMPONENT:** It consists of a set of operations that are performed on a table or group of tables
3. **INTEGRITY CONSTRAINTS:** These are a set of rules that governs the entity types and their relationships.

A relation comprises:

1. **RELATION SCHEMA:** Relation schema (or relation intension) depicts the attributes of the table. A relation schema consists of a relation name R and a set of attributes A1, A2,....., An. It is represented by R(A1,A2,.....,An) and is used to describe a relation R.

E.g. BOOK(ISBN, Book_title, Category, Price, Copyright_Date, Year, page_Count, P_ID)
AUTHOR(Autor_ID, Aname, City, State, Zip, Phone)



PUBLISHER(P_ID, Pname, Address, State, Phone, Email_ID)

2. RELATION INSTANCE: Relation instance (or relation extension) is a two-dimensional table with a time-varying set of tuples. A relation instance r of the relation schema $R(A_1, A_2, \dots, A_n)$ is a set of n -tuples, denoted by $r(R)$. A relation instance is an ordered set of attribute values v that belongs to the domain D and it can be denoted as $r = \{t_1, t_2, \dots, t_m\}$, where $t = \{v_1, \dots, v_n\}$.

ISBN	BookTitle	Category	Price	CopyrightDate	Year	PageCount	P_ID
001-354-1	C++	Textbook	40	2001	2002	700	P003
003-456-5	UNIX	Textbook	26	2003	2005	500	P001
005-786-9	Call Away	Novel	30	2007	2009	600	p003

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 or relationship. The table name and column names are used to help in interpreting the meaning of the values in each row.

E.g.

STUDENT	Name	Number	Class	Major
	Smith	17	1	CS
	Brown	8	2	CS

In the above table (STUDENT), each row represents about a particular student entity. The column names (Name, Number, Class, etc.)- specify how to interpret the data values in each row. All values in a column are of the same data type. A row is called a **tuple**, a column header is called an **attribute**, and the table is called a **relation**.

DOMAINS: The data type describing the types of values that can appear in each column is represented by a **domain** of possible values. Method of specifying a domain is to specify a data type from which the data values forming the domain are drawn. Data type or format is also specified for each domain. A domain is thus given a name, data type, and format.

E.g.: Names: The set of character strings that represent names of persons. Employee_age: Possible ages of employees of a company; each must be a value between 15 and 80 years old.

II. CHARACTERISTICS OF RELATION

1. Tuples in a relation do not have any particular order.
2. The attributes and the values within tuples are ordered. The ordering of attributes is not important.
3. Each value in a tuple is an atomic value; that is, it is not divisible into components. Hence, composite and multi valued attributes are not allowed. Multi valued attributes must be represented by separate relations, and composite attributes are represented only by their simple component attributes. A special value called nulls, which are used to represent the values of attributes that may be unknown or may not apply to a tuple.

ENTITY RELATIONSHIP (E-R) MODEL: The E-R model views the real world as a set of basic objects (known as entities), their characteristics (known as attributes) and associations among these objects (known as relationships). The information gathered from the user forms the basis for designing the E-R model. The nouns in the requirements specification document are represented as the entities. Additional nouns that describe the nouns corresponding to entities form the attributes. Verbs become the relationships among various entities. E-R model represents the overall logical structure of a database. It is a semantic model. It attempts to represent the meaning of the data.

ENTITY AND ATTRIBUTE: An entity is a distinguishable object that has an independent existence in the real world. The attributes are the properties of an entity that characterize and describe it. For example, BOOK entity can have Price as an attribute. **ENTITY TYPE:** A set or a collection of entities that share the same attributes but different values. E.g. EMPLOYEE, BOOK. **ENTITY SET:** The collection of all instances of a particular entity type in the database at any point of time. It is usually referred using the same name as the entity type.



KEY ATTRIBUTE: The attribute (or combination of attributes) whose values are distinct for each individual instance of an entity type. An entity type may have more than one key attribute. **SUPER KEY:** A set of one or more attributes, when taken together, helps in uniquely identifying each entity is called a super key. **CANDIDATE KEY:** A minimal super key that does not contain any extra attributes in it. **PRIMARY KEY:** The candidate key, which is chosen by the database designer to uniquely identify entities. **COMPOSITE KEY:** If a primary key is formed by the combination of two or more attributes.

WEAK ENTITY AND STRONG ENTITY: An entity type that does not have any key attribute of its own is called a weak entity. An entity type that has a key attribute is called a strong entity. The entities belonging to a weak entity type are identified by the attributes of a strong entity type in combination of one of their attribute, thus the strong entity types are known as **identifying or parent entity type**. The weak entity type, on the other hand, is known as **child or subordinate entity type**.

ATTRIBUTE TYPES: The basic object that the ER model represents is an entity, which is a "thing" in the real world with an independent existence. An entity maybe an object with a physical existence (E.g. a particular person, car, house) or it may be an object with a conceptual existence (E.g. a company, a job, or a university course). Each entity has attributes-the particular properties that describe it. For example, an employee entity may be described by the employee's name, age, address, salary, and job. A particular entity will have a value for each of its attributes. The attribute values that describe each entity become a major part of the data stored in the database.

- 1. SIMPLE (ATOMIC) AND COMPOSITE:** Simple or atomic attribute cannot be divided into sub parts. E.g. Designation, Age, Gender etc. Composite attributes can be divided into sub parts. Composite attributes help to group together related attributes. E.g. Name attribute can be divided into First name, Middle name and Last name.
- 2. SINGLE VALUED AND MULTI VALUED:** Single valued attributes have a single value for a particular entity. E.g. Age. Multi valued attributes have a set of values for a particular entity. College_Degree.
- 3. STORED AND DERIVED:** Attributes that are stored in to the database is called stored attributes. Attributes that are derived from the values of other related attributes or entities are called derived attributes. E.g. The attribute Age can be derived from the stored attribute Date_Of_Birth and the current date.
- 4. NULL VALUE:** A null value is used when an entity does not have a value for an attribute.
- 5. COMPLEX ATTRIBUTES:** Composite and multi-valued attributes can be nested in an arbitrary way to form complex attribute.

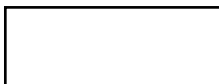
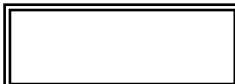
RELATIONSHIPS: A relationship is an association between two or more entities. E.g. The relationship PUBLISHED_BY associates a book with its publisher. **RELATIONSHIP SET:** Set of relationships of the same type. A relationship type R amongn entity types E1, E2 . . . En defines a set of associations among entities from these entity types.

II E-R DIAGRAM

E-R DIAGRAM: An E-R diagram is a specialized graphical tool that demonstrates the inter-relationships among various entities of a database. It is used to represent the overall logical structure of the database. In ER diagrams the emphasis is on representing the schemas rather than the instances. This is more useful in database design because a database schema changes rarely, whereas the contents of the entity set change frequently.

E-R DIAGRAM NAMING CONVENTIONS: The meaning of the context is conveyed as much as possible. The usage of names should be consistent throughout the E-R diagram. The usage of hyphens, spaces, digits and special characters should be avoided. Singular names should be used for entity types. If a primary key of one entity type is used as a foreign key in another entity type, it should be used unchanged or with some extensions.

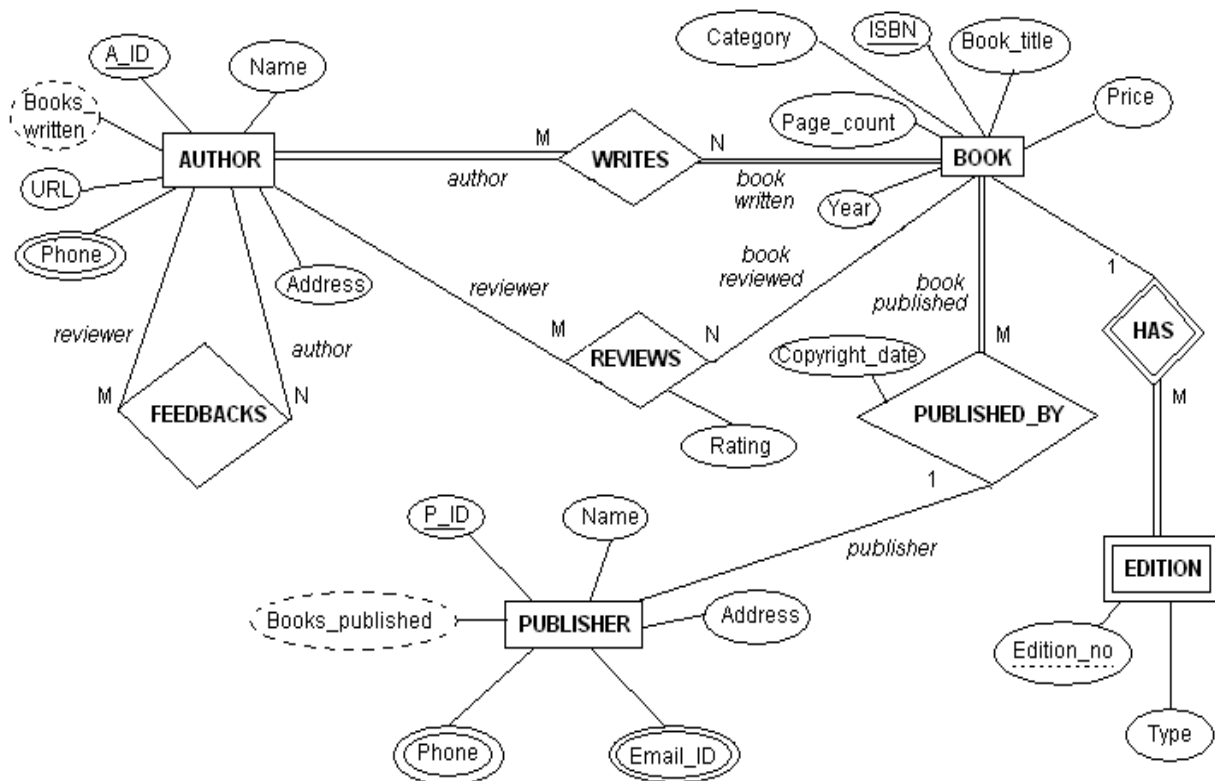
E-R DIAGRAM NOTATIONS:

Symbol	Meaning	Symbol	Meaning
	Entity type		Weak entity type



	Relationship		Identifying relationship
	Attribute		Key attribute
	Partial key of weak entity attribute		Multi-valued attribute
	Derived attribute		Connects attribute to entity type and entity types to relationship types. Total Participation
	1:1 Relationship		1:M Relationship
	M:1 Relationship		M:N Relationship

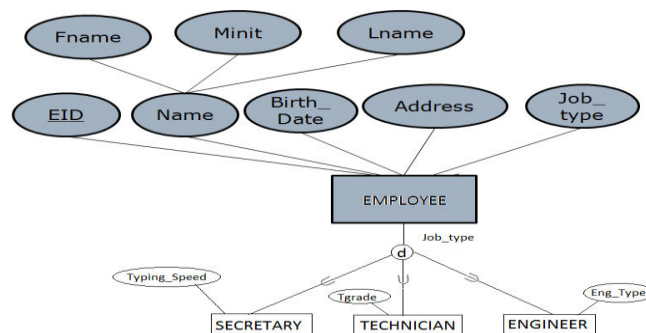
E.g. E-R DIAGRAM FOR ONLINE BOOK DATABASE



III. ENHANCED E-R DIAGRAM

ENHANCED E-R (EER) MODEL: The EER (Enhanced ER) model includes all the modelling concepts of the ER model. In addition, it includes the concepts of subclass and super class and the related concepts of specialization and generalization. Another concept included in the EER model is that of a category or union type, which is used to represent a collection of objects that is the union of objects of different entity types. Associated with these concepts is the important mechanism of attribute and relationship inheritance.

SUBCLASS AND SUPER CLASS AND INHERITANCE: In many cases an entity type has numerous sub groupings of its entities that are meaningful and need to be represented explicitly because of their significance to the database application. For example, the entities those are members of the EMPLOYEE entity type may be grouped further into SECRETARY, ENGINEER, MANAGER, TECHNICIAN, SALARIED_EMPLOYEE, HOURLY_EMPLOYEE, and so on. Each of these sub-grouping of an entity set is called **subclass**. Entity type from which the subclasses are derived is called **super class**. Specialization is the process of defining a set of subclasses of an entity type. The relationship between a super class and anyone of its subclasses is a super class/subclass or simply class/subclass relationship. A member entity of the subclass represents the same real-world entity as some member of the super class. An entity that is a member of a subclass **inherits** all the attributes of the entity as a member of the super class. The entity also inherits all the relationships in which the super class participates.





SPECIALIZATION: An entity type may include sub-groupings of its entities in such a way that entities of one subgroup are distinct in some way from the entities of other subgroups. Specialization is the process of defining a set of subclasses of an entity type; this entity type is called the superclass of the specialization. The set of subclasses that form a specialization is defined on the basis of some distinguishing characteristic of the entities in the superclass. The process of refining the higher-level entity types (super class) into lower-level entity types (subclass) by adding some additional features to each of them is a top-down design approach and is known as **specialization**. Specialization process allows doing the following

1. Define a set of subclasses of an entity type
2. Establish additional specific attributes with each subclass
3. Establish additional specific relationship types between each subclass and other entity types or other subclasses.

GENERALIZATION: The generalization process can be viewed as being functionally the inverse of the specialization process. Suppress the differences among several entity types, identify their common features, and generalize them into a single superclass of which the original entity types are special subclasses. The term generalization refers to the process of defining a generalized entity type from the given entity types. This design process is a bottom up approach.

RELATIONAL ALGEBRA

RELATIONAL ALGEBRA: A data model must include a set of operations to manipulate the database, in addition to the data model's concepts for defining database structure and constraints. Two formal languages for the relational model: relational algebra and relational calculus. Those are used to specify the basic retrieval requests. Relational algebra and relational calculus are logically equivalent, however, they differ on the basis of the approach they follow to retrieve data from a relation.

1. **RELATIONAL CALCULUS:** Provides higher level declarative notations for specifying relational queries. A relational calculus expression creates a new relation, which is specified in terms of variables that range over rows of the stored database relations (in tuple calculus) or over columns of the stored relations (in domain calculus). In a calculus expression, there is no order of operations to specify how to retrieve the query result. A calculus expression specifies only what information the result should contain. The relational calculus is important because it has a firm basis in mathematical logic
2. **RELATIONAL ALGEBRA:** The basic set of operations for the relational model is the relational algebra. These operations enable a user to specify basic retrieval requests. The result of retrieval is a new relation, which may have been formed from one or more relations. The algebra operations thus produce new relations, which can be further manipulated using operations of the same algebra. A sequence of relational algebra operations forms a relational algebra expression, whose result will also be a relation that represents the result of a database query. Relational algebra is a procedural query language that consists of a set of operations that take one or two relations as input and result into a new relation as an output. Operations are divided into two groups:
 - a) Consists of operations developed specifically for relational databases such as select, project, rename, join, and division
 - b) Includes the set-oriented operations such as union, intersection, difference and Cartesian product.

UNARY RELATIONAL OPERATIONS: Operating on a single relation is known as unary operations. Unary operations are:

1. **SELECT OPERATION:** The select operation retrieves all those tuples (rows) from a relation R that satisfy a specific condition. The select operation is specified as $\sigma_{\langle \text{selection_condition} \rangle}(\mathbf{R})$. The comparison operators =, \neq , <, \leq , >, \geq can be used to specify the conditions for selecting required tuples from a relation. Conditions can be concatenated to one another to give more specific conditions using logical operators AND(\wedge), OR(\vee) and NOT(\neg).

E.g. $\sigma_{\text{Category}=\text{"Novel"}}(\text{BOOK})$
 $\sigma_{\text{Category}=\text{"Novel"} \wedge \text{Page_Count} \geq 400}(\text{BOOK})$
 $\sigma_{\text{Dno}=4}(\text{EMPLOYEE})$

The SELECT operation can also be visualized as a horizontal partition of the relation into two sets of tuples—those tuples that satisfy the condition and are selected, and those tuples that do not satisfy the condition and are discarded. SELECT operation is applied to a single relation. SELECT operation is applied to each tuple individually. The number of tuples in the resulting relation is always less than or equal to the number of tuples in R. The result of SELECT operation has the same attributes as R. Select operation is commutative

$$\sigma_{\langle \text{cond1} \rangle}(\sigma_{\langle \text{cond2} \rangle}(\mathbf{R})) = \sigma_{\langle \text{cond2} \rangle}(\sigma_{\langle \text{cond1} \rangle}(\mathbf{R})).$$



2. **PROJECT OPERATION:** The project operation is used to select some required attributes (Column) from a relation R, while discarding the other attributes. The result of the PROJECT operation can hence be visualized as a vertical partition of the relation into two relations: one has the needed columns(attributes) and contains the result of the operation, and the other contains the discarded columns. The project operation is specified as $\pi_{\langle \text{attribute_list} \rangle}(\mathbf{R})$.

E.g. $\pi_{\text{ISBN, Book_title, Price}}(\text{BOOK})$
 $\pi_{\text{Gender, Salary}}(\text{EMPLOYEE})$

The result of project operation has only the attributes specified, and in the same order as specified. Project operation removes duplicate tuples. It is not commutative.

SEQUENCE OF OPERATIONS: Several relational operations can be written as a single relational algebra expression by nesting the operations. We can also apply one operation at a time and create intermediate results. Relations that hold intermediate results can give names using rename operation.

$\rho_{\text{Fname, Lname, Salary}}(\sigma_{\text{DeptNo}=5}(\text{EMPLOYEE}))$

3. **RENAME OPERATION:** It is often simpler to break down a complex sequence of operations by specifying intermediate result relations than to write a single relational algebra expression. We can also use this technique to rename the attributes in the intermediate and result relations. The rename operation is used to rename either the relation name or the attribute name or both. It is a unary operation. The rename operations are specified as:

$\rho_S(\mathbf{R})$ - Renames the relation only

$\rho_S(B_1, B_2, \dots, B_n)(\mathbf{R})$ - Renames both relation and its attributes

$\rho(B_1, B_2, \dots, B_n)(\mathbf{R})$ - Renames the attributes only

E.g. $\rho_{\mathbf{R}}(\text{FIRSTNAME, LASTNAME, SALARY}) \leftarrow \pi_{\text{FNAME, LNAME, SALARY}}(\text{EMPLOYEE})$

RELATIONAL ALGEBRA OPERATIONS FROM SET THEORY

1. **UNION** – The result of this operation, denoted by $\mathbf{R} \cup \mathbf{S}$, 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. Binary operation.

R	Name
Smith	
Brown	
Jack	

S	Name
John	
Smith	

RUS	Name
Smith	
Brown	
Jack	
John	

2. **INTERSECTION** – The result of this operation, denoted by $\mathbf{R} \cap \mathbf{S}$, is a relation that includes all tuples that are in R and S. Binary operation.

R	Name
Smith	
Brown	
Jack	

S	Name
John	
Smith	
Brown	

R∩S	Name
Smith	
Brown	

3. **SET DIFFERENCE (MINUS)** – The result of this operation, denoted by $\mathbf{R} - \mathbf{S}$, is a relation that includes all tuples that are in R but not in S. Binary operation.

R	Name
Smith	
Brown	
Jack	
Bob	

S	Name
Brown	
Mary	
Andrew	

R-S	Name
Smith	
Jack	
Bob	

UNION and INTERSECTION are commutative operations. $\mathbf{R} \cup \mathbf{S} = \mathbf{S} \cup \mathbf{R}$. $\mathbf{R} \cap \mathbf{S} = \mathbf{S} \cap \mathbf{R}$.

UNION and INTERSECTION are associative operations. $\mathbf{R} \cup (\mathbf{S} \cup \mathbf{T}) = (\mathbf{R} \cup \mathbf{S}) \cup \mathbf{T}$.

$\mathbf{R} \cap (\mathbf{S} \cap \mathbf{T}) = (\mathbf{R} \cap \mathbf{S}) \cap \mathbf{T}$.



4. **CARTESIAN PRODUCT (Cross Product or Cross Join):** Binary set operation, denoted by \times . This operation produces a new element by combining every member (tuple) from one relation with every member (tuple) from the other relation. The result of $R(A_1, A_2, \dots, A_n) \times S(B_1, B_2, \dots, B_m)$ is a relation $Q(A_1, A_2, \dots, A_n, B_1, B_2, \dots, B_m)$. Degree of R is n and degree of S is m , then the degree $R \times S$ is $n+m$ and have $m \times n$ tuples.

EMPLOYEE			DEPENDENT			
Name	Salary	Ssn	Essn	DepName	Gender	BrDate
Alice	25000	9912	123	Alice	F	5/4/1997
Jennifer	31000	7612	234	Joyce	M	1/5/1970
Joyce	25000	7531				

EMPLOYEE X DEPENDENT						
Name	Salary	Ssn	Essn	DepName	Gender	BrDate
Alice	25000	9912	123	Alice	F	5/4/1997
Alice	25000	9912	234	Joyce	M	1/5/1970
Jennifer	31000	7612	123	Alice	F	5/4/1997
Jennifer	31000	7612	234	Joyce	M	1/5/1970
Joyce	25000	7531	123	Alice	F	5/4/1997
Joyce	25000	7531	234	Joyce	M	1/5/1970

BINARY RELATIONAL OPERATIONS

1. **JOIN :** Denoted by \Join . Used to combine related tuples from two relations into single tuples. Allow to process relationships among relations. JOIN can be stated in terms of a CARTESIAN PRODUCT followed by a SELECT operation. The result of $R(A_1, A_2, \dots, A_n) \Join S(B_1, B_2, \dots, B_m)$ is a relation $Q(A_1, A_2, \dots, A_n, B_1, B_2, \dots, B_m)$. Degree of R is n and degree of S is m , then the degree $R \Join S$ is $n+m$. This operation produces a new element by combining every member (tuple) from one relation with every member (tuple) from the other relation – whenever the combination satisfies the join condition. The general form of join operation on two relations R and S is $R \Join \langle \text{Join condition} \rangle S$

EMPLOYEE			DEPENDENT			
Name	Salary	Ssn	Essn	DepName	Gender	BrDate
Alice	25000	9912	123	Alice	F	5/4/1997
Jennifer	31000	7612	7612	Joy	M	1/5/1970
Joyce	25000	7531				

EMPLOYEE $\Join_{Ssn=Essn}$ DEPENDENT						
Name	Salary	Ssn	Essn	DepName	Gender	BrDate
Jennifer	31000	7612	7612	Joy	M	1/5/1970

- a) **EQUIJOIN:** JOIN where the only comparison operator = is called an EQUIJOIN. In the result of an EQUIJOIN always have one or more pairs of attributes that have identical values in every tuple. E.g. the example shown above.
- b) **NATURAL JOIN:** Binary operation, denoted by $*$. It allows to combine certain selections and Cartesian product into one operation. Forms a Cartesian product of its two arguments, performs a selection forcing equality on those attributes that appear in both relation schemas, and finally removes duplicate **attributes**.

EMPLOYEE			DEPENDENT			
Name	Salary	Ssn	Ssn	DepName	Gender	BrDate
Alice	25000	9912	123	Alice	F	5/4/1997
Jennifer	31000	7612	7612	Joy	M	1/5/1970
Joyce	25000	7531				

EMPLOYEE * DEPENDENT						
Name	Salary	Ssn	Ssn	DepName	Gender	BrDate
Jennifer	31000	7612	7612	Joy	M	1/5/1970



2. **DIVISION:** Binary operation, denoted by \div . It is suited to queries that include the phrase “For All” . $T(Y) = R(Z) \div S(X)$. For a tuple t to appear in the result T of the DIVISION, the values in t must appear in R in combination with every tuples in S .

EMPLOYEE	
Name	ProjNo
Alice	9
Alice	7
Joyce	9
Wilfred	7
Wilfred	9
Smith	8

SMITH_PROJ
ProjNo
9
7

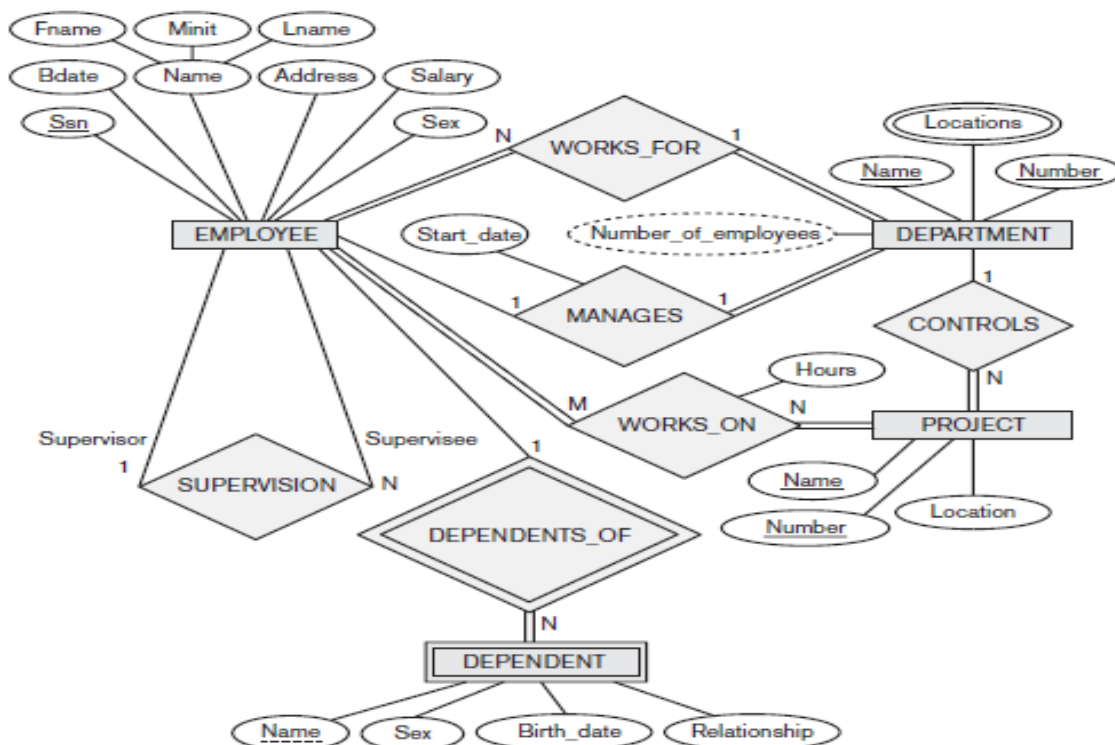
EMPLOYEE \div SMITH_PROJ
NAME
Alice
Wilfred

Relational Database Design Using ER-to-Relational Mapping

The steps of an algorithm for ER-to-relational mapping. We use the COMPANY database example to illustrate the mapping procedure.

Step 1: Mapping of Regular Entity Types. For each regular (strong) entity type E in the ER schema, create a relation R that includes all the simple attributes of E . Include only the simple component attributes of a composite attribute. Choose one of the key attributes of E as the primary key for R . If the chosen key of E is a composite, then the set of simple attributes that form it will together form the primary key of R .

Example, we create the relations EMPLOYEE, DEPARTMENT, and to correspond to the regular entity types EMPLOYEE, DEPARTMENT, and PROJECT The foreign key and relationship attributes, if any, are not included yet; they will be added during subsequent steps. These include the attributes Super_ssn and Dno of EMPLOYEE, Mgr_ssn and Mgr_start_date of DEPARTMENT, and Dnum of PROJECT. In our example, we choose Ssn, Dnumber, and Pnumber as primary keys for the relations EMPLOYEE, DEPARTMENT, and PROJECT, respectively. Knowledge that Dname of DEPARTMENT and Pname of PROJECT are secondary keys is kept for possible use later in the design.





Step 2: Mapping of Weak Entity Types. For each weak entity type W in the ER schema with owner entity type E, create a relation R and include all simple attributes of W as attributes of R. In addition, include as foreign key attributes of R, the primary key attribute(s) of the relation(s) that correspond to the owner entity type(s); this takes care of mapping the identifying relationship type of W. The primary key of R is the combination of the primary key(s) of the owner(s) and the partial key of the weak entity type W, if any. If there is a weak entity type E2 whose owner is also a weak entity type E1, then E1 should be mapped before E2 to determine its primary key first.

In the example, we create the relation DEPENDENT in this step to correspond to the weak entity type DEPENDENT (see Figure 9.3(b)). 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 Essn, although this is not necessary.

The primary key of the DEPENDENT relation is the combination {Essn, Dependent_name}, because Dependent_name is the partial key of DEPENDENT.

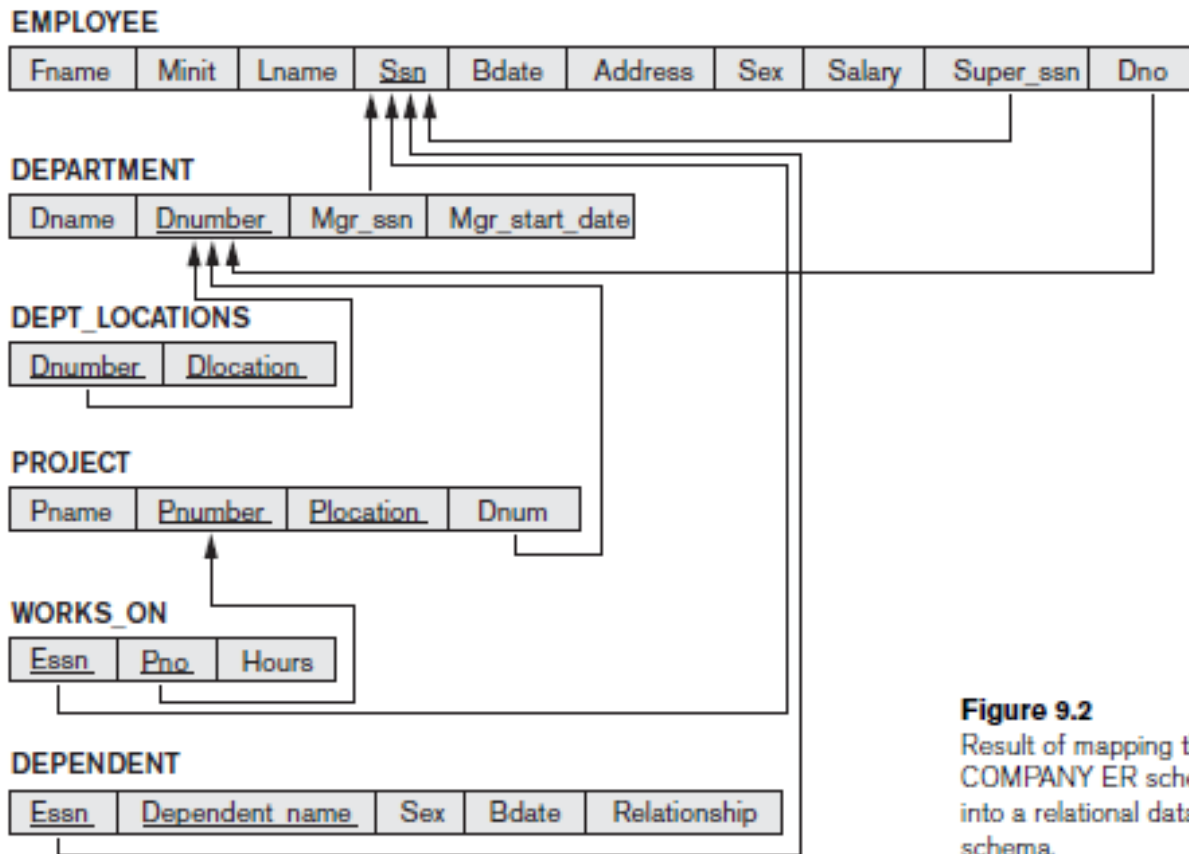


Figure 9.2
Result of mapping the COMPANY ER schema into a relational database schema.

Step 3: Mapping of Binary 1:1 Relationship Types. For each binary 1:1 relationship type R in the ER schema, identify the relations S and T that correspond to the entity types participating in R. There are three possible approaches: (1) the foreign key approach, (2) the merged relationship approach, and (3) the cross reference or relationship relation approach. The first approach is the most useful and should be followed unless special conditions exist.

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 of the 1:1 relationship type R as attributes of S.

In the example, we map the 1:1 relationship type MANAGES from by choosing the participating entity type DEPARTMENT to serve in the role of S because its participation in the MANAGES relationship type is total (every department has a manager). 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



INNO SPACE
SJIF Scientific Journal Impact Factor
Impact Factor: 8.118



ISSN INTERNATIONAL
STANDARD
SERIAL
NUMBER
INDIA



INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

 **9940 572 462**  **6381 907 438**  **ijirset@gmail.com**



www.ijirset.com

Scan to save the contact details