



Program : **B.Tech**

Subject Name: **Database Management System**

Subject Code: **IT-405**

Semester: **4th**



LIKE & FOLLOW US ON FACEBOOK
facebook.com/rgpvnotes.in

UNIT-II

Relational algebra is a procedural query language, which takes instances of relations as input and yields instances of relations as output. It uses operators to perform queries. An operator can be either unary or binary. They accept relations as their input and yield relations as their output. Relational algebra is performed recursively on a relation, and intermediate results are also considered relations.

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**. A domain of possible values represents the data type describing the types of values that can appear in each column. We now define these terms—domain, tuple, attribute, and relation formally.

Domains, Attributes, Tuples, and Relations

Deposit Relation

bname	Account	Ename	Balance
Bhanwarkuwan	SBI1200	Ram	5000
Tilak Nagar	SBI1238	Amit	1000

Customer Relation

Ename	Street	City
Ramesh	MG road	Indore
Jhon	RNT Marg	Indore

- It has four attributes.
- For each attribute, there is a permitted set of values, called the **domain** of that attribute.
- E.g. the domain of *name* is the set of all branch names.

Let/denote the domain of *name*, and D_2 , D_3 and D_4 the remaining attributes' domains respectively.

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 are forming the domain are drawn. It is also useful to specify a name for the domain, to help in interpreting its values. Some examples of domains follow:

- *Use_phone_numbers*. The set of ten-digit phone numbers valid in the United States.
- *Local_phone_numbers*. The set of seven-digit phone numbers valid within an area code in the United States. The use of local phone numbers is quickly becoming obsolete, being replaced by standard ten-digit numbers.

A relation schema R , denoted by $R(A_1, A_2, \dots, A_n)$, is made up of a relation name R and a list of attributes, A_1, A_2, \dots, A_n . Each attribute A_i is the name of a role played by some domain D in the relation schema R . D is called the domain of A_i and is denoted by $\text{dom}(A_i)$. A relation schema is used to describe a relation R is called the name of this relation. The degree (or arity) of a relation is the number of attributes n of its relation schema.

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

STUDENT (Name: string, Ssn: string, Homophone: string, Address: string, Office phone: string, Age: integer, Gpa: real)

Characteristics of Relations

The Ordering of Tuples in a Relation. A relation is defined as a set of tuples. Mathematically, elements of a set have no order among them hence, tuples in a relation do not have any order.

The ordering of Values within a Tuple and an Alternative Definition of a Relation. According to the other definition of a relation, an n -tuple is an ordered list of n values, so the ordering of values in a tuple and hence of attributes in a relation schema is essential.

Values and NULLs in the Tuples. Each value in a tuple is an atomic value that is; it is not divisible into components within the framework of the underlying relational model. Hence, composite and multivalued attributes are not allowed. An important concept is that of NULL values, which are used to represent the values of attributes that may be unknown or may not apply to a tuple.

Relational Model Notation

A relation schema R of degree n is denoted by $R (A_1, A_2, A_n)$.

- The uppercase letters Q, R, S denote relation names.
- The lowercase letters q, r, s denotes relation states.
- The letters t, u, v denotes tuples.
- In general, the name of a relation schema such as **STUDENT** also indicates the current set of tuples in that relation—the current relation state—whereas **STUDENT (Name, Ssn,)** refers only to the relational schema.
- An attribute A can be qualified with the relation name R to which it belongs by using the dot notation $R.A$ —for example, **STUDENT. Name** or **STUDENT. Age**. This is because the same name may be used for two attributes in different relations.

Relational Model Constraints and Relational Database Schemas

Constraints on databases can generally be divided into three main categories:

1. Constraints that are inherent in the data model. We call these inherent model-based constraints or implicit constraints. Example: In the relational model, no two tuples in a relation can be duplicates. Why? Because a relation is a set of tuples, as opposed to a bag/multiset or a sequence.
2. Constraints that can be directly expressed in schemas of the data model, typically by specifying them in the DDL. We call these schema-based constraints or specific constraints.
3. Constraints that cannot be directly expressed in the schemas of the data model, and hence must be expressed and enforced by the application programs. We call these application-based or semantic constraints or business rules.

Types of Schema-based Constraints

Domain Constraints

Domain constraints specify that within each tuple, the value of each attribute A must be an atomic value from the domain $\text{dom}(A)$.

Key Constraints and Constraints on NULL Values

In the formal relational model, a relation is defined as a set of tuples. All elements of a set are distinct; hence, all tuples in a relation must also be distinct.

A superkey SK specifies a uniqueness constraint that no two distinct tuples in any state r of R can have the same value for SK . Every relation has at least one default super key—the set of all its attributes. A superkey can have redundant attributes, however, so a more useful concept is that of a key, which has no redundancy.

A relation schema may have more than one key. In this case, each of the keys is called a **candidate key**. For example, the **CAR** relation has (**Licence_no** **Eng_sr_no** **Model** **Make_year** **model**) two candidate keys: **License_number** and **Engine_serial_number**. It is common to designate one of the candidate keys as the **primary key** of the relation. This is the candidate key whose values are used to identify tuples in the relation.

Relational Databases and Relational Database Schemas

The definitions and constraints we have discussed so far apply to single relations and their attributes. A relational database usually contains many relations, with tuples in relations that are related in many ways. In this section, we define a relational database and a relational database schema.

A relational database schema S is a set of relation schemas $S = \{R_1, R_2, \dots, R_m\}$ and a set of integrity constraints IC . A relational database state DB of S is a set of relation states $DB = \{r_1, r_2, \dots, r_m\}$ such that each r_i is a state of R_i and such that the r_i relation states satisfy the integrity constraints specified in IC . Below Figure shows a relational database schema that we call $COMPANY = \{EMPLOYEE, DEPARTMENT, DEPT_LOCATIONS, PROJECT, WORKS_ON, DEPENDENT\}$. The underlined attributes represent primary keys.

EMPLOYEE

Fname	Minit	Lname	<u>Ssn</u>	Bdata	Address	Sex	Salary	Super_ssn	Dno
-------	-------	-------	------------	-------	---------	-----	--------	-----------	-----

DEPARTMENT

DNAME	Dnumber	Mgr_ssn	Mgr_str_Date
-------	---------	---------	--------------

DEPT_ LOCATIONS

<u>Dnumber</u>	<u>DLocation</u>
----------------	------------------

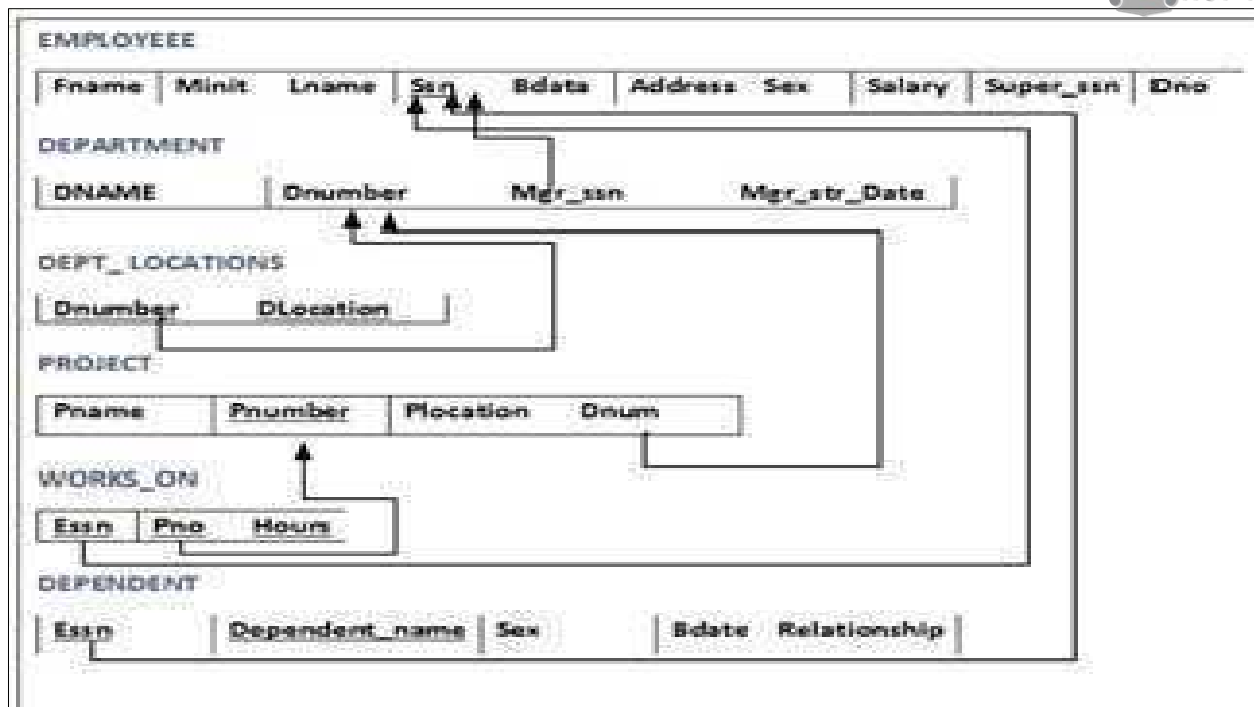
PROJECT

Pname	<u>Pnumber</u>	Plocation	Dnum
-------	----------------	-----------	------

Integrity, Referential Integrity, and Foreign Keys

The entity integrity constraint states that no primary key value can be NULL. This is because the primary key value is used to identify individual tuples in a relation. Having NULL values for the primary key implies that we cannot identify some tuples. For example, if two or more tuples had NULL for their primary keys, we may not be able to distinguish them if we try to reference them from other relations.

The referential integrity constraint is specified between two relations and is used to maintain the consistency among tuples in the two relations. Informally, the referential integrity constraint states that a tuple in one relation that refers to another relation must refer to an existing tuple in that relation.



The relational algebra is essential for several reasons. First, it provides a formal foundation for relational model operations. Second, and perhaps more important, it is used as a basis for implementing and optimising queries in the query processing and optimisation modules that are integral parts of relational database management systems (RDBMSs).

Difference Between Calculus & algebra

BASIS FOR COMPARISON	RELATIONAL ALGEBRA	RELATIONAL CALCULUS
Basic	Relational Algebra is a Procedural language.	Relational Calculus is Declarative language.
States	Relational Algebra states how to obtain the result.	Relational Calculus states what result we must obtain.
Order	Relational Algebra describes the order in which operations must be performed.	Relational Calculus does not specify the order of operations.
Domain	Relational Algebra is not domain dependent.	Relation Calculus can be domain dependent.
Related	It is close to a programming language.	It is close to the natural language.

The fundamental operations of relational algebra are as follows –

- Select
- Project
- Union
- Set different
- Cartesian product
- Rename

We will discuss all these operations in the following sections.

Select Operation (σ)

It selects tuples that satisfy the given predicate from a relation.

Notation – $\sigma(r)$

Where σ stands for selection predicate and r stands for relation. p is the propositional logic formula which may use connectors like and, or, and not. These terms may use relational operators like $=$, \neq , \geq , $<$, $>$, \leq .

For example –

$\sigma_{\text{subject} = \text{"database"}}(\text{Books})$

Output – Selects tuples from books where the subject is 'database'.

$\sigma_{\text{subject} = \text{"database"} \text{ and } \text{price} = \text{"450"}}(\text{Books})$

Output – Selects tuples from books where the subject is 'database', and 'price' is 450.

$\sigma_{\text{subject} = \text{"database"} \text{ and } \text{price} = \text{"450"} \text{ or } \text{year} > \text{"2010"}}(\text{Books})$

Output – Selects tuples from books where the subject is 'database', and 'price' is 450 or those books published after 2010.

Project Operation (Π)

It projects column(s) that satisfy a given predicate.

Notation – $\Pi_{A1, A2, A_n}(r)$

Where $A1, A2, A_n$ are attribute names of relation r .

Duplicate rows are automatically eliminated, as the relation is a set.

For example –

$\Pi_{\text{the subject, author}}(\text{Books})$

Selects and projects columns named as subject and author from the relation Books.

Union Operation (\cup)

It performs a binary union between two given relations and is defined as –

$r \cup s = \{t \mid t \in r \text{ or } t \in s\}$

Notation – $r \cup s$

Where r and s are either database relations or relation result set (temporary relation).

For a union operation to be valid, the following conditions must hold –

r and s must have the same number of attributes.

Attribute domains must be compatible.

Duplicate tuples are automatically eliminated.

$\Pi_{\text{author}}(\text{Books}) \cup \Pi_{\text{author}}(\text{Articles})$

Output – Projects the names of the authors who have either written a book or an article or both.

Set Difference ($-$)

The result of set difference operation is tuples, which are present in one relation but are not in the second relation.

Notation – $r - s$

Finds all the tuples that are present in r but not in s .

$\Pi_{\text{author}}(\text{Books}) - \Pi_{\text{author}}(\text{Articles})$

Output – Provides the name of authors who have written books but not articles.

Cartesian Product (\times)

Combines information of two different relations into one.

Notation – $r \times s$

Where r and s are relations and their output will be defined as –

$$r \times s = \{ q \ t \mid q \in r \text{ and } t \in s \}$$

$\sigma_{\text{author} = 'tt'}(\text{Books} \times \text{Articles})$

Output – Yields a relation, which shows all the books and articles written by tt.

Rename Operation (ρ)

The results of relational algebra are also relations but without any name. The rename operation allows us to rename the output relation. 'rename' operation is denoted with small Greek letter rho ρ .

Notation – $\rho_x(E)$

Where the result of expression E is saved with the name of x .

Additional operations are –

- Set intersection
- Assignment
- Natural join

We can define the three operations UNION, INTERSECTION, and SET DIFFERENCE on two union-compatible relations R and S as follows:

■ **UNION:** The result of this operation, denoted by $R \cup S$, is a relation that includes all tuples that are either in R or in S or in both R and S . Duplicate tuples are eliminated.

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

■ **SET DIFFERENCE (or MINUS):** The result of this operation, denoted by $R - S$, is a relation that includes all tuples that are in R but not in S .

The set operations UNION, INTERSECTION, and MINUS. (a) Two union-compatible relations.

(b) $\text{STUDENT} \cup \text{INSTRUCTOR}$. (c) $\text{STUDENT} \cap \text{INSTRUCTOR}$. (d) $\text{STUDENT} - \text{INSTRUCTOR}$.

(e) $\text{INSTRUCTOR} - \text{STUDENT}$.

CARTESIAN PRODUCT operation—also known as CROSS PRODUCT or CROSS JOIN—which is denoted by \times .

This is also a binary set operation, but the relations on which it is applied do not have to be union-compatible. The JOIN operation, denoted by \bowtie , is used to combine related tuples from two relations into single “longer” tuples. This operation is very important for any relational database with more than a single relation because it allows us to process relationships among relations.

$\text{DEPT_MGR} \leftarrow \text{DEPARTMENT} \bowtie_{\text{Mgr_ssn}=\text{Snn}} \text{EMPLOYEE}$

$\text{RESULT} \leftarrow \Pi_{\text{Dname, Lname}}(\text{DEPT_MGR})$

(a) STUDENT <table> <tr><th>Fn</th><th>Ln</th></tr> <tr><td>Susan</td><td>Yao</td></tr> <tr><td>Ramesh</td><td>Shah</td></tr> <tr><td>Johnny</td><td>Kohler</td></tr> <tr><td>Barbara</td><td>Jones</td></tr> <tr><td>Amy</td><td>Ford</td></tr> <tr><td>Jimmy</td><td>Wang</td></tr> <tr><td>Ernest</td><td>Gilbert</td></tr> </table>	Fn	Ln	Susan	Yao	Ramesh	Shah	Johnny	Kohler	Barbara	Jones	Amy	Ford	Jimmy	Wang	Ernest	Gilbert	INSTRUCTOR <table> <tr><th>Fname</th><th>Lname</th></tr> <tr><td>John</td><td>Smith</td></tr> <tr><td>Ricardo</td><td>Browne</td></tr> <tr><td>Susan</td><td>Yao</td></tr> <tr><td>Francis</td><td>Johnson</td></tr> <tr><td>Ramesh</td><td>Shah</td></tr> </table>	Fname	Lname	John	Smith	Ricardo	Browne	Susan	Yao	Francis	Johnson	Ramesh	Shah	(b) <table> <tr><th>Fn</th><th>Ln</th></tr> <tr><td>Susan</td><td>Yao</td></tr> <tr><td>Ramesh</td><td>Shah</td></tr> <tr><td>Johnny</td><td>Kohler</td></tr> <tr><td>Barbara</td><td>Jones</td></tr> <tr><td>Amy</td><td>Ford</td></tr> <tr><td>Jimmy</td><td>Wang</td></tr> <tr><td>Ernest</td><td>Gilbert</td></tr> <tr><td>John</td><td>Smith</td></tr> <tr><td>Ricardo</td><td>Browne</td></tr> <tr><td>Francis</td><td>Johnson</td></tr> </table>	Fn	Ln	Susan	Yao	Ramesh	Shah	Johnny	Kohler	Barbara	Jones	Amy	Ford	Jimmy	Wang	Ernest	Gilbert	John	Smith	Ricardo	Browne	Francis	Johnson
Fn	Ln																																																			
Susan	Yao																																																			
Ramesh	Shah																																																			
Johnny	Kohler																																																			
Barbara	Jones																																																			
Amy	Ford																																																			
Jimmy	Wang																																																			
Ernest	Gilbert																																																			
Fname	Lname																																																			
John	Smith																																																			
Ricardo	Browne																																																			
Susan	Yao																																																			
Francis	Johnson																																																			
Ramesh	Shah																																																			
Fn	Ln																																																			
Susan	Yao																																																			
Ramesh	Shah																																																			
Johnny	Kohler																																																			
Barbara	Jones																																																			
Amy	Ford																																																			
Jimmy	Wang																																																			
Ernest	Gilbert																																																			
John	Smith																																																			
Ricardo	Browne																																																			
Francis	Johnson																																																			
(c) <table> <tr><th>Fn</th><th>Ln</th></tr> <tr><td>Susan</td><td>Yao</td></tr> <tr><td>Ramesh</td><td>Shah</td></tr> </table>	Fn	Ln	Susan	Yao	Ramesh	Shah	(d) <table> <tr><th>Fn</th><th>Ln</th></tr> <tr><td>Johnny</td><td>Kohler</td></tr> <tr><td>Barbara</td><td>Jones</td></tr> <tr><td>Amy</td><td>Ford</td></tr> <tr><td>Jimmy</td><td>Wang</td></tr> <tr><td>Ernest</td><td>Gilbert</td></tr> </table>	Fn	Ln	Johnny	Kohler	Barbara	Jones	Amy	Ford	Jimmy	Wang	Ernest	Gilbert	(e) <table> <tr><th>Fname</th><th>Lname</th></tr> <tr><td>John</td><td>Smith</td></tr> <tr><td>Ricardo</td><td>Browne</td></tr> <tr><td>Francis</td><td>Johnson</td></tr> </table>	Fname	Lname	John	Smith	Ricardo	Browne	Francis	Johnson																								
Fn	Ln																																																			
Susan	Yao																																																			
Ramesh	Shah																																																			
Fn	Ln																																																			
Johnny	Kohler																																																			
Barbara	Jones																																																			
Amy	Ford																																																			
Jimmy	Wang																																																			
Ernest	Gilbert																																																			
Fname	Lname																																																			
John	Smith																																																			
Ricardo	Browne																																																			
Francis	Johnson																																																			

Variations of JOIN: The EQUIJOIN and NATURAL JOIN

The most common use of JOIN involves join conditions with equality comparisons only. Such a JOIN, where the only comparison operator used is =, is called an EQUIJOIN.

The standard definition of NATURAL JOIN requires that the two join attributes (or each pair of join attributes) have the same name in both relations. If this is not the case, a renaming operation is applied first.

PROJ_DEP DEPT_LOCS \leftarrow DEPARTMENT * DEPT_LOCATIONS.

DEPT_LOCS				
Dname	Dnumber	Mgr_ssn	Mgr_start_date	Location
Headquarters	1	888665555	1981-06-19	Houston
Administration	4	987654321	1995-01-01	Stafford
Research	5	333445555	1988-05-22	Bellaire
Research	5	333445555	1988-05-22	Sugarland
Research	5	333445555	1988-05-22	Houston

The DIVISION Operation

The DIVISION operation, denoted by \div , is useful for a special kind of query that sometimes occurs in database applications. An example is Retrieve the names of employees who work on all the projects that 'John Smith' works on.

$SMITH \leftarrow \sigma_{Fname='John' \text{ AND } Lname='SMITH'}(EMPLOYEE)$
 $SMITH_PNOS \leftarrow \pi_{pno}(WORKS_ON \bowtie_{Essn=Ssn} SMITH)$

	PURPOSE	NOTATION
SELECT	Selects all tuples that satisfy the selection condition from a relation R.	$\sigma_{\langle \text{selection condition} \rangle}(R)$
PROJECT	Produces a new relation with only some of the attributes of R and removes duplicate tuples.	$\pi_{\langle \text{attribute} \rangle}$

THETA JOIN	Produces all combinations of tuples from R and R ₂ that satisfy the join condition.	$R_1 \bowtie R_2$
EQUIJOIN	Produces all the combinations of tuples from R ₁ and R ₂ that satisfy a join condition with only equality comparisons.	$R_1 \bowtie R_2$
NATURAL JOIN	Same as EQUIJOIN except that the join attributes of R ₂ are not included in the resulting relation; if the join attributes have the same names, they do not have to be specified at all.	$R_1 \bowtie R_2$ OR $R_1 \bowtie R_2$
UNION	Produces a relation that includes all the tuples in R ₁ or R ₂ or both R ₁ and R ₂ ; R ₁ and R ₂ must be union-compatible.	$R_1 \cup R_2$
INTERSECTION	Produces a relation that includes all the tuples in both R ₁ and R ₂ ; R ₁ and R ₂ must be union-compatible.	$R_1 \cap R_2$
DIFFERENCE	Produces a relation that includes all the tuples in R ₁ that are not in R ₂ ; R ₁ and R ₂ must be union-compatible.	$R_1 - R_2$
CARTESIAN PRODUCT	Produces a relation that has the attributes of R ₁ and R ₂ and includes as tuples all possible combinations of tuples from R ₁ and R ₂ .	$R_1 \times R_2$
DIVISION	Produces a relation R(X) that includes all tuples t[X] in R ₁ (Z) that appear in R ₁ in combination with every tuple from R ₂ (Y), where $Z = X \cup Y$.	$R_1(Z) \div R_2(Y)$

Example on Relational Algebra

EMPLOYEE									
Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT				DEPT_LOCATIONS	
Dname	Dnumber	Mgr_ssn	Mgr_start_date	Dnumber	Dlocation
Research	5	333445555	1988-05-22	1	Houston
Administration	4	987654321	1995-01-01	4	Stafford
Headquarters	1	888665555	1981-06-19	5	Bellaire
				5	Sugarland
				5	Houston

WORKS_ON			PROJECT			
Essn	Pno	Hours	Pname	Pnumber	Plocation	Dnum
123456789	1	32.5	ProductX	1	Bellaire	5
123456789	2	7.5	ProductY	2	Sugarland	5
666884444	3	40.0	ProductZ	3	Houston	5
453453453	1	20.0	Computerization	10	Stafford	4
453453453	2	20.0	Reorganization	20	Houston	1
333445555	2	10.0	Newbenefits	30	Stafford	4
333445555	3	10.0				
333445555	10	10.0				
333445555	20	10.0				
999887777	30	30.0				
999887777	10	10.0				
987987987	10	35.0				
987987987	30	5.0				
987654321	30	20.0				
987654321	20	15.0				
888665555	20	NULL				

DEPENDENT				
Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

Query 1. Retrieve the name and address of all employees who work for the 'Research' department.

```
RESEARCH_DEPT ← σDname = 'Research'(DEPARTMENT)
RESEARCH_EMPS ← (RESEARCH_DEPT ⋈Dnumber = Dno EMPLOYEE)
RESULT ← πFname, Lname, Address(RESEARCH_EMPS)
```

Query 2. For every project located in 'Stafford', list the project number, the controlling department number, and the department manager's last name, address, and birth date.

```
STAFFORD_PROJS ← σPlocation = 'stafford'(PROJECT)
CONTR_DEPTS ← (STAFFORD_PROJS ⋈Dnum=Dnumber DEPARTMENT)
PROJ_DEPT_MGRS ← (CONTR_DEPTS ⋈Mgr_ssn = Ssn EMPLOYEE)
RESULT ← πPnumber, Dnum, Lname, Address, Bdate(PROJ_DEPT_MGRS)
```

Query 3. Find the names of employees who work on all the projects controlled by department number 5.

```
DEPT5_PROJS ← ρ(Pno)(πPnumber(σDnum=5(PROJECT)))
EMP_PROJ ← ρ(Ssn, Pno)(πEssn, Pno(WORKS_ON))
RESULT_EMP_SSNS ← EMP_PROJ ÷ DEPT5_PROJS
RESULT ← πLname, Fname (RESULT_EMP_SSNS * EMPLOYEE)
```

Query 4. List the names of all employees with two or more dependents.

Strictly speaking, this query cannot be done in the primary (original) relational algebra. We must use the AGGREGATE FUNCTION operation with the COUNT aggregate function. We assume that dependents of the same employee have distinct Dependent_name values.

```
T1(Ssn, No_of_Dependents) ← Essn 3 COUNTDependent_name(DEPENDENT)
T2 ← σNo_of_Dependent ≥ 2(T1)
RESULT ← πLname, Fname(T2 * EMPLOYEE)
```

ID	Name	Dept_name	Salary
10101	Shrinivasan	Comp.sci	65000

12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstin	Physics	95000

Instructor Relation

instructor relation where the instructor is in the “Physics” department, we write:
dept name = “Physics” (instructor)

Relational Calculus

In contrast to Relational Algebra, Relational Calculus is a non-procedural query language, that is, it tells what to do but never explains how to do it.

Relational calculus exists in two forms –

Tuple Relational Calculus (TRC) Filtering variable ranges over tuples

Notation – {T | Condition}

Returns all tuples T that satisfies a condition.

For example –

{ T.name | Author(T) AND T.article = 'database' }

Output – Returns tuples with 'name' from Author who has written article on 'database'.

TRC can be quantified. We can use Existential (\exists) and Universal Quantifiers (\forall).

For example –

{ R | $\exists T \in \text{Authors}(T.\text{article} = \text{'database'} \text{ AND } R.\text{name} = T.\text{name})$ }

Output – The above query will yield the same result as the previous one.

Domain Relational Calculus (DRC)

In DRC, the filtering variable uses the domain of attributes instead of entire tuple values (as done in TRC, mentioned above).

Notation –

{ a1, a2, a3, ..., an | P (a1, a2, a3, ... ,an) }

Where a1, a2 are attributes and P stands for formulae built by inner attributes.

For example –

{ < article, page, subject > | $\in TP \wedge \text{subject} = \text{'database'}$ }

Output – Yields Article, Page, and Subject from the relation TP where subject is database.

Just like TRC, DRC can also be written using existential and universal quantifiers. DRC also involves relational operators.

The expression power of Tuple Relation Calculus and Domain Relational Calculus is equivalent to Relational

Algebra.

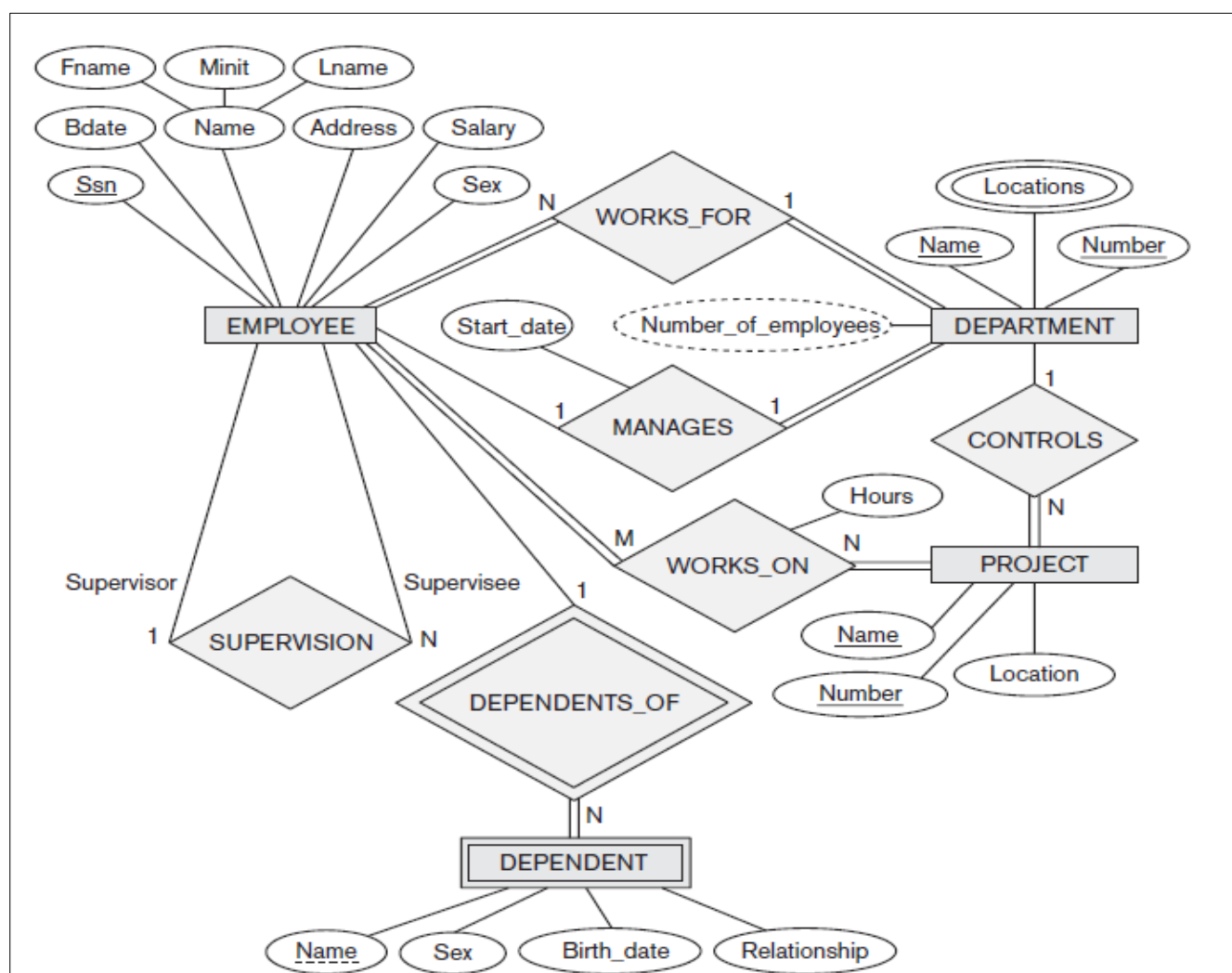
E-R Diagram

An ER schema diagram for the COMPANY database

Entities and Attributes

Entities and Their Attributes. 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 may be an object with a physical existence (for example, a person, car, house, or employee) or it may be an object with a conceptual existence (for instance, a company, a job, or a university course).

Composite versus Simple (Atomic) Attributes. **Composite attributes** can be divided into smaller subparts, which represent more basic attributes with independent meanings — for example, the Address attribute of the EMPLOYEE entity.



Single-Valued versus Multivalued Attributes. Most attributes have a single value for a particular entity; such attributes are called **single-valued**. For example, Age.

A **multivalued** attribute may have lower and upper bounds to constrain the *number of values* allowed for each entity. For example, the Colors attribute of a car.

Stored versus Derived Attributes. In some cases, two (or more) attribute values are related—for example, the Age and Birth_date attributes of a person.

Initial Conceptual Design of the COMPANY Database covers

Relationship & type

Cardinality

Week Entity

Participation Constraints: - The participation constraint specifies whether the existence of an entity depends on its being related to another entity via the relationship type.

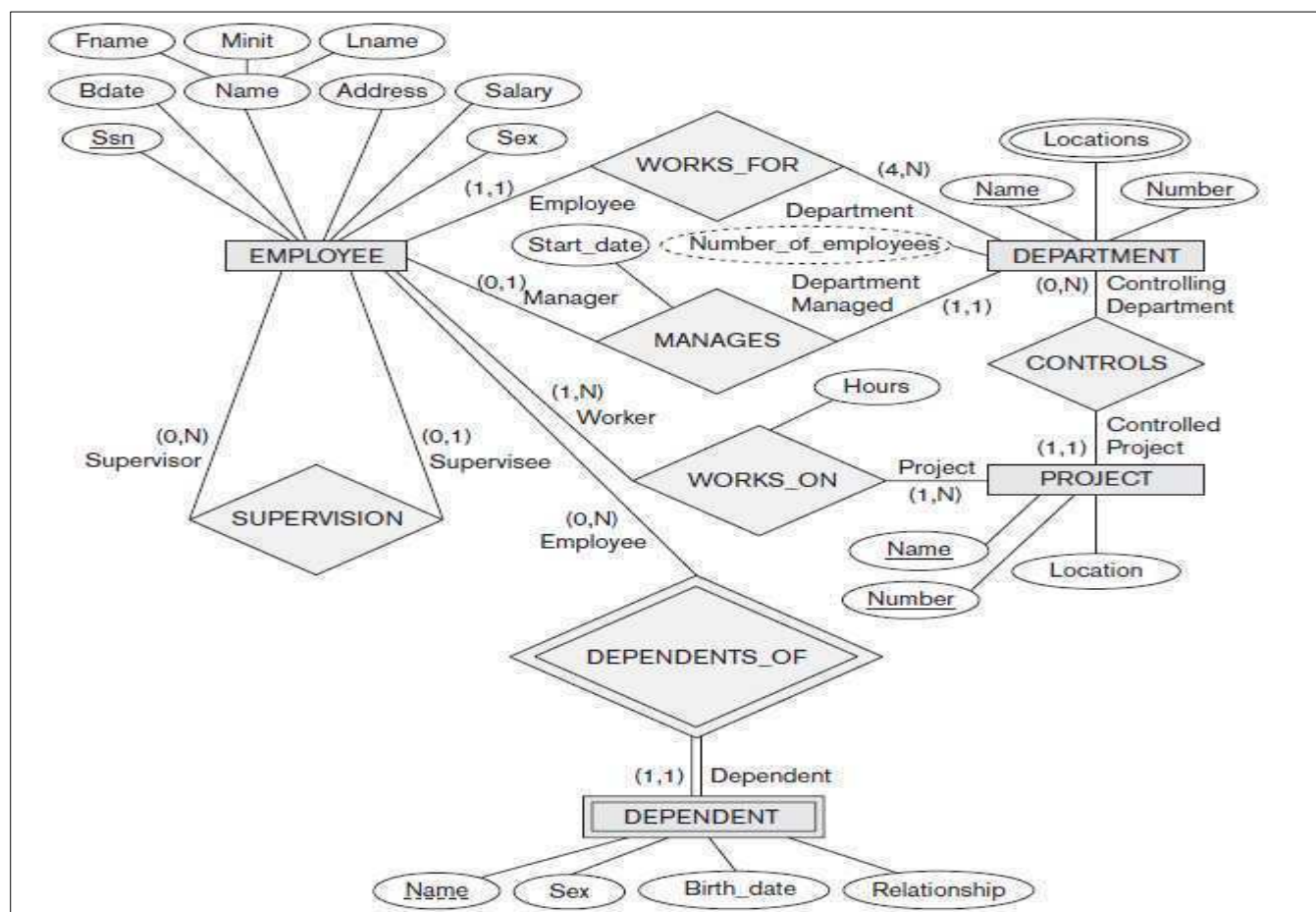
Company schema, with structural constraints specified using (min, max) notation and role names.

Enhanced Entity-Relationship (EER) Model

Semantic data modelling concepts that were incorporated into conceptual data models such as the ER Model. ER model can be enhanced to include these concepts, leading to the **Enhanced ER (EER)** model.

Subclasses: - An entity type is used to represent both a type of entity and the entity set or collection of entities of that type that exist in the database. For example, the entity type EMPLOYEE describes the type (that is, the attributes and relationships) of each employee entity, and refers to the current set of EMPLOYEE entities in the COMPANY database.

Superclasses: - We call each of these subgroupings a subclass or subtype of the EMPLOYEE entity type, and the EMPLOYEE entity type is called the superclass or supertype for each of these subclasses.



Extended E-R Model

Specialisation

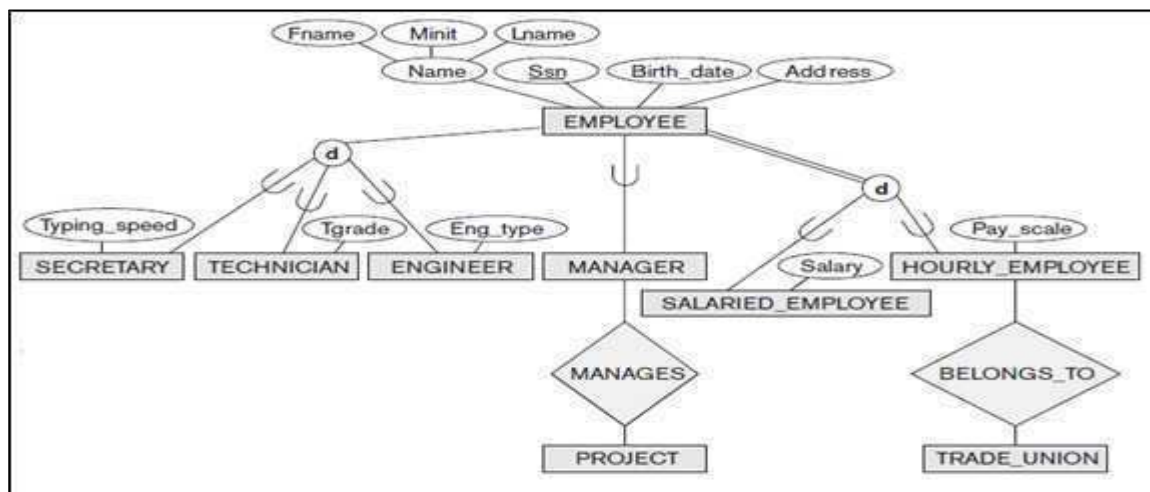
Specialisation is the process of defining a *set of subclasses* of an entity type. This entity type is called the **superclass** of the specialisation. The set of subclasses that forms a specialisation is defined by some distinguishing characteristic of the entities in the superclass. For example, the set of subclasses {SECRETARY, ENGINEER, TECHNICIAN} is a specialisation of the superclass EMPLOYEE that distinguishes among employee entities based on the *job type* of each employee entity.

Steps for Specialization

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

EER diagram notation to represent subclasses and specialisation. Three specialisations of EMPLOYEE:

{SECRETARY, TECHNICIAN, ENGINEER}
 {MANAGER}
 {HOURLY_EMPLOYEE, SALARIED_EMPLOYEE}

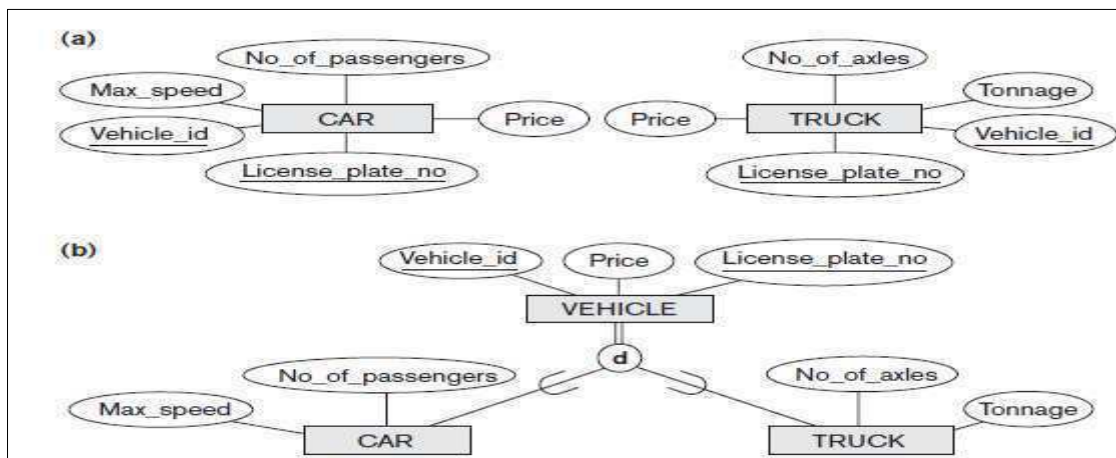


Generalisation

Generalisation

We can think of a *reverse process* of abstraction in which we suppress the differences among several entity types, identify their standard features, and **generalise** them into a single **superclass** of which the original entity types are subclasses. For example, consider the entity types CAR and TRUCK.

Generalisation. (a) Two entity types, CAR and TRUCK. (b) Generalising CAR and TRUCK into the superclass VEHICLE.

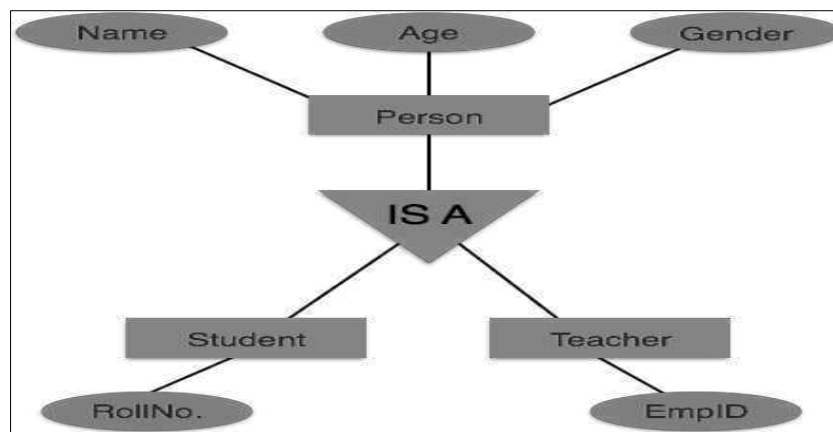


Specialization

Inheritance

We use all the above features of ER-Model to create classes of objects in object-oriented programming. The details of entities are generally hidden from the user; this process known as abstraction.

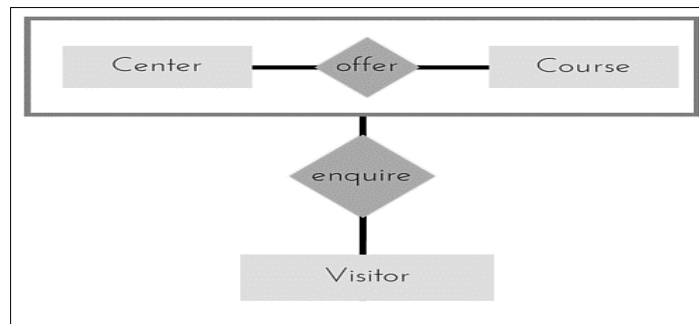
Inheritance is an essential feature of Generalization and Specialization. It allows lower-level entities to inherit the attributes of higher-level entities.

**Inheritance**

For example, the attributes of a Person class such as name, age, and gender can be inherited by lower-level entities such as Student or Teacher.

Aggregation

Aggregation is a process when the relation between two entities is treated as a single entity. Here the relation between Center and Course is acting as an Entity in relation with Visitor.

**Aggregation**



RGPVNOTES.IN

We hope you find these notes useful.

You can get previous year question papers at
<https://qp.rgpvnotes.in> .

If you have any queries or you want to submit your
study notes please write us at
rgpvnotes.in@gmail.com



LIKE & FOLLOW US ON FACEBOOK

facebook.com/rgpvnotes.in