DBMS

Data – raw facts and figures
Information – processed data

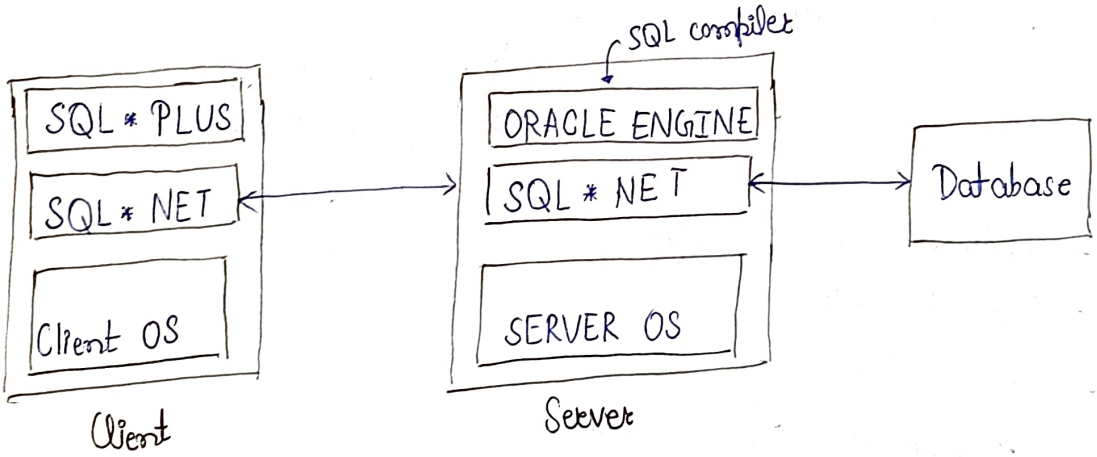
Database – collection of related data from which user can retrieve the desired data easily

- Data stored in form of tables.

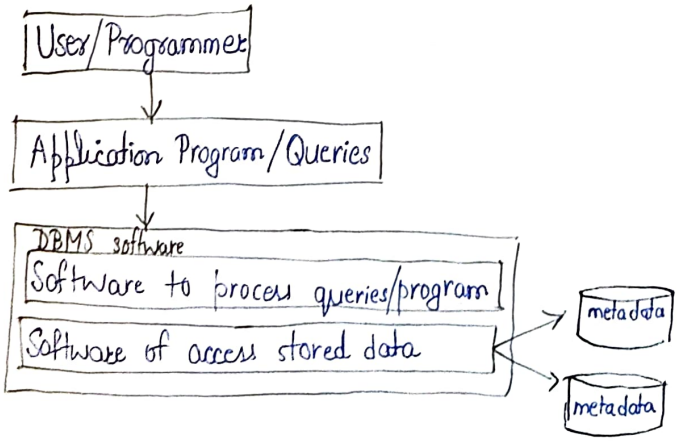
* Tuple – row of table
* Attribute – column of table

DBMS – software through which
- database is created
- data is modified in the database
- data is retrieved from the database

e.g. – ORACLE



Architecture of DBMS



Users of DBMS

(I) End Users
- Naive Users — interacts using some application software
- Sophisticated Users — interacts by writing their own queries

(II) System Analysts

(III) Application Programmer

(IV) DBA (Database Administrator)

→ Schema definition and modification
- structure of database

Schema – overall structure of database
Instance – at a particular time, the state of data stored in the database

→ Security enforcement and administration
- controls who can access what portion of the database
- decides constraints on the database
- decides which attribute is "PRIMARY KEY"
  - no duplicates
  - no "NULL" values
    - data doesn't exist
    - data not known

→ Preliminary Database Design

→ Data Analysis

→ Routine Monitoring
- regular checking of database

→ New Software Installation

DBMS v/s Traditional File system

(i) Controlling Redundancy
- duplication of data
  - → wastage of storage space
  - → inconsistency among different instances of the same data

* Normalization – technique used in DBMS to avoid inconsistency by dividing larger schemas into smaller ones

(ii) Restriction of unauthorized access

(iii) Providing search technique for efficient query process (Query Optimization)

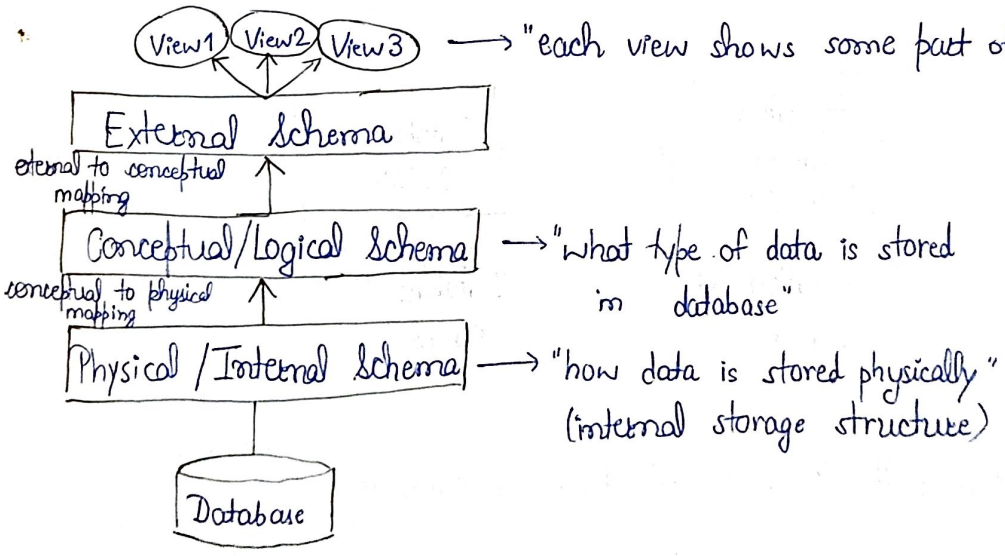
(iv) Backup and Recovery

(v) Enforcing Integrity Constraint
- → Entity Integrity Constraint (primary key constraint)
- → Referential Integrity Constraint (foreign key constraint)
- → Check Constraint (checks for a specific condition)

(vi) Permitted Interference and Access Rule

(vii) Providing multiple user interface

Three Schema Architecture

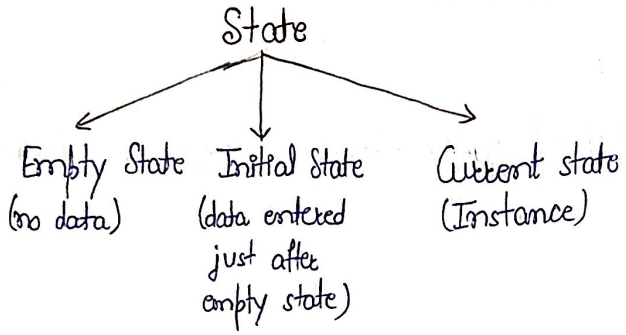


Data Types in SQL:

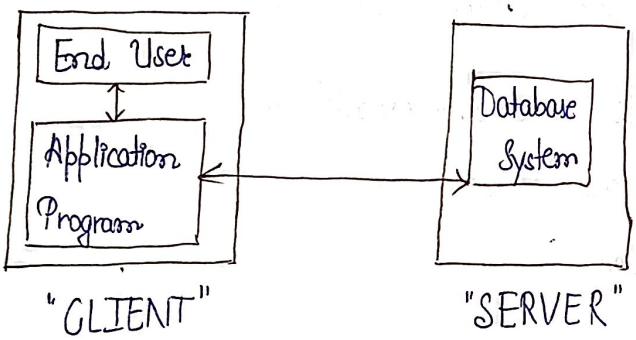
char(n), varchar(n), varchar2(n), number(n)

Data Independence

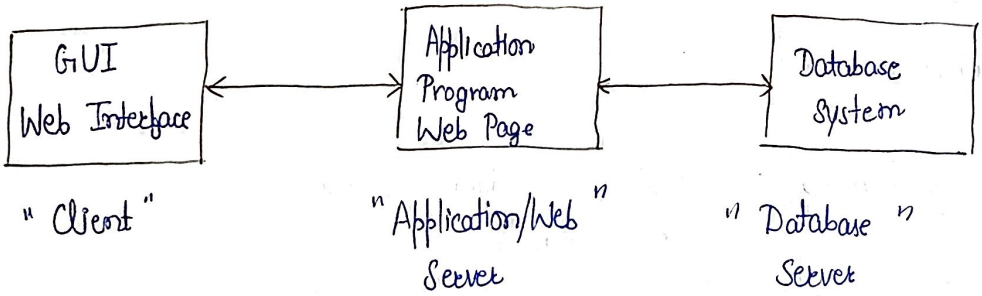
- Physical Data Independence — changing physical structure doesn't affect the logical layer
- Logical Data Independence — conceptual structure changes doesn't affect the view level



Two tier Architecture



Three Tier Architecture



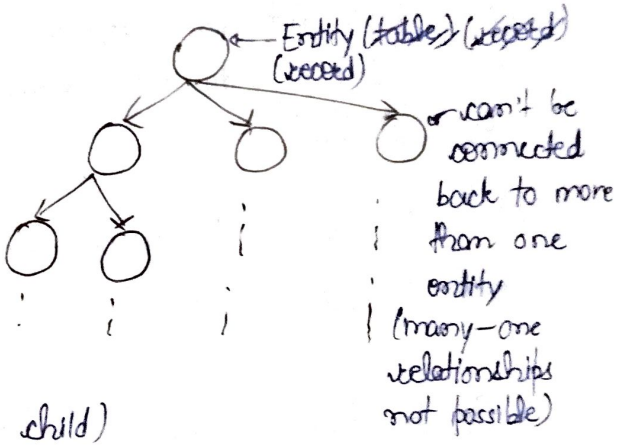
* Data Model — tool that helps to represent data in database

Data Model

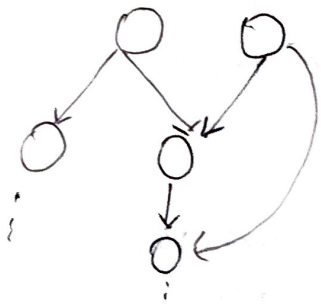
- Object-based Model (Object Oriented Model / Entity Relationship (ER) Model)
  - used at external and conceptual levels
  - uses entities (objects) and relationships b/w them
    - in ER model
    - in object-oriented model
- Physical Model → used at physical level
- Record-based Model
  - each tuple treated as a "record"

Record-based Model

- Hierarchical Model
  - 1:n relationship
  - oldest data model
  - easy to implement
  - lack of structural independence
  - operational anamoly (entities need to traversed from parent to child)



- Network Model
  - one child can be connected to more than one parent
  - implementation is complex
  - more structural independence



- Relational Model
  - data is stored in "tables"
  - rows – tuples
  - columns – attributes
  - * Degree of Relation: no. of attributes/columns
  - * Cardinality of Relation: no. of tuples/rows
  - structural independent
  - conceptually simpler.
  - easy to design and maintain

Database Language

(i) DDL – Data Definition Language
- to define the structure of data in database

e.g.- CREATE

(ii) DML – Data Manipulation Language
- to modify data in database

e.g.- UPDATE, DELETE, INSERT, ALTER

(iii) DRL – Data Retrieval Language
- to retrieve data from database

e.g.- SELECT

(iv) TPC – Transaction Processing Control

e.g. – ROLLBACK, COMMIT

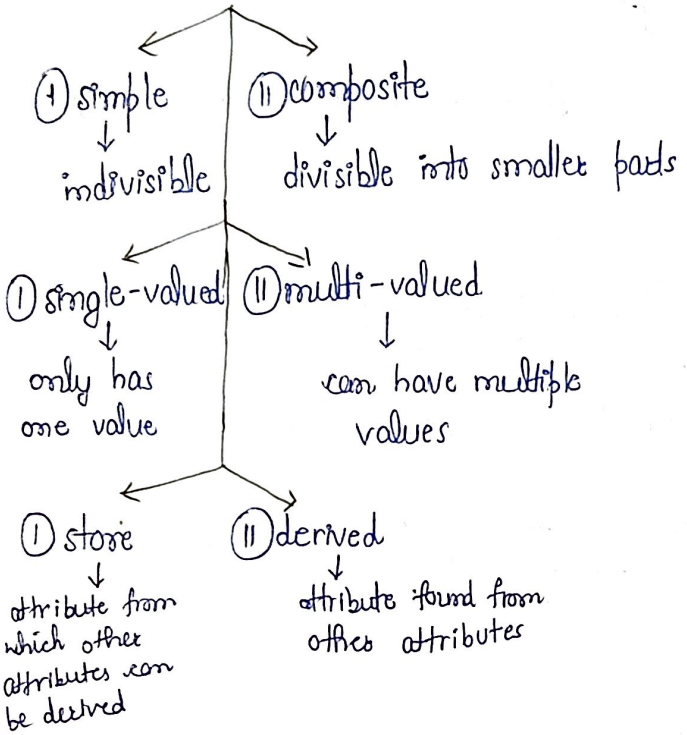
```
Create table student
 ( name    varchar2(10),
   roll    number(10),
   address varchar2(100)
 );
```

```
Insert into student values
 ('&name', '&roll', '&address');
```

Entity - Relationship (ER) Model (Object-based Model)

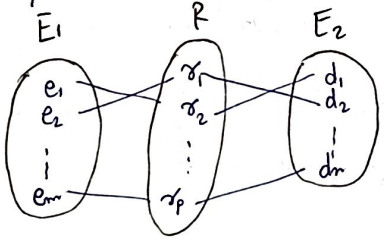
* Entity — any real life object
* Entity Set — set of similar entities

Attribute → property through which entity can be identified

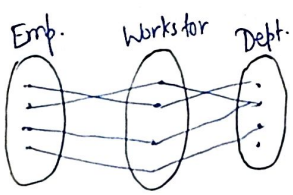


* NULL attribute
  - ↳ doesn't exist
  - ↳ not known to DB designer
* ~~Composite~~ Complex attribute
  - ↳ combination of multiple composite attributes

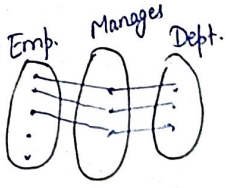
* Relation: A relationship 'R' between two entity sets 'E_1' and 'E_2' associates one entity of 'E_1' with a entity of 'E_2'



- Full Participation — each entity of E_1 is connected to some entity of E_2 through the relation 'R'.
- Partial Participation — not all entities of E_1 are connected to some entity of E_2 through the relation 'R'



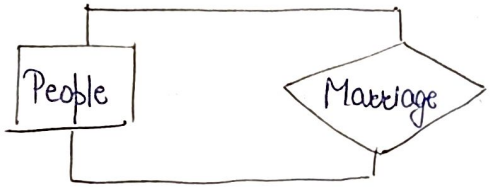
Full Participation



Partial Participation

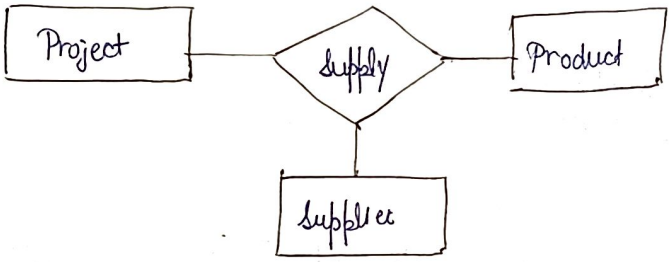
* Degree of Relation: no. of entity sets connected through that relation.

(I) Unary Relation — associates one entity of an entity set to another entity of the same.

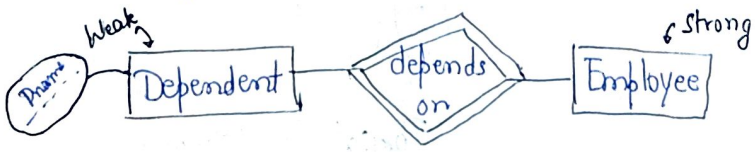


(II) Binary Relation — associates one entity of an entity-set 'E_1' to an entity of another entity set 'E_2'

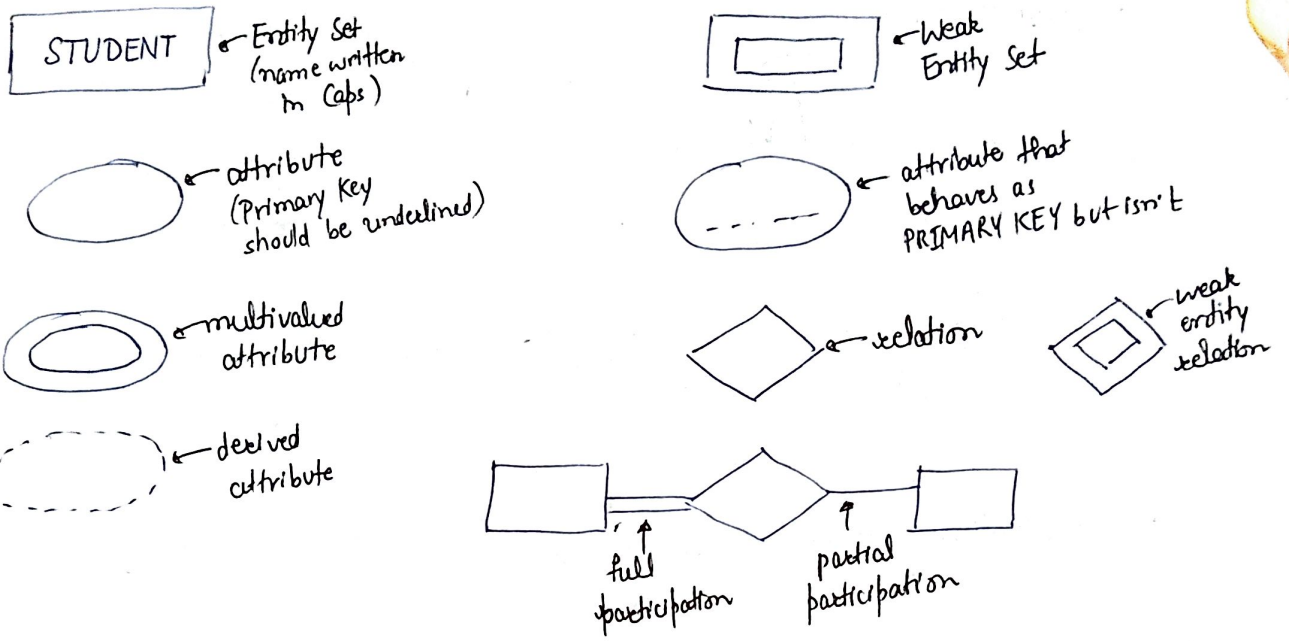
(III) Ternary Relation — associates entities between three entity sets.



* Domain of an attribute: range of attributes acceptable for that particular attribute
* Role of a relation: name of the relationship (what it is doing)
* Cardinality Ratio: no. of entities connected from entity set 'E_1' to no. of entities connected to that of entity set 'E_2' through relation 'R'
  - (I) 1:1 (one-one)
  - (II) M:N (many-many)
  - (III) M:1 (many-one)
  - (IV) 1:M (one-many)
* Weak Entity Set: can't be uniquely identified by its own attributes alone, and depends on a related strong entity set for identification (no PRIMARY KEY)



Entity Relationship (ER) Diagram



(I) List of entities and attributes

AUTHOR {authorid, name, address}

BOOKS {__}

PUBLISHER {__}

REVIEWER {__}

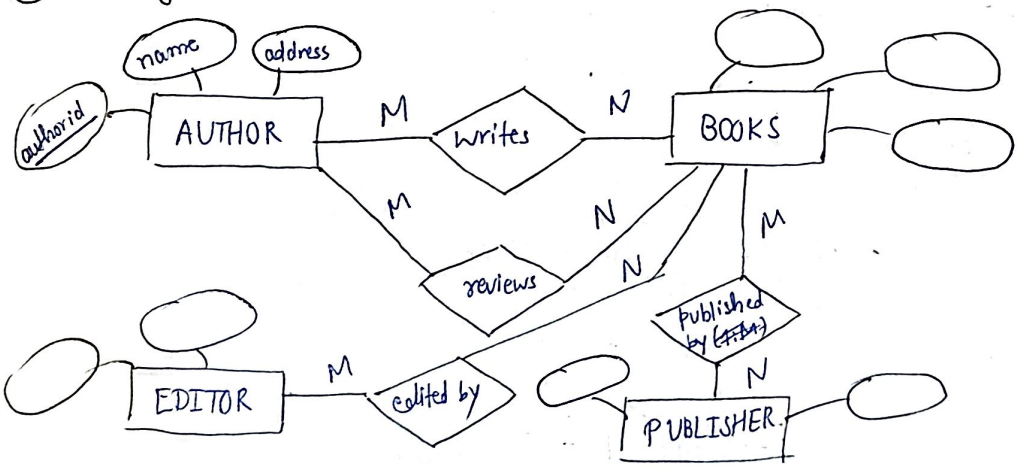
(II) Relationship

AUTHOR → writes → Book (M:N)

AUTHOR → reviews → Book (M:N)

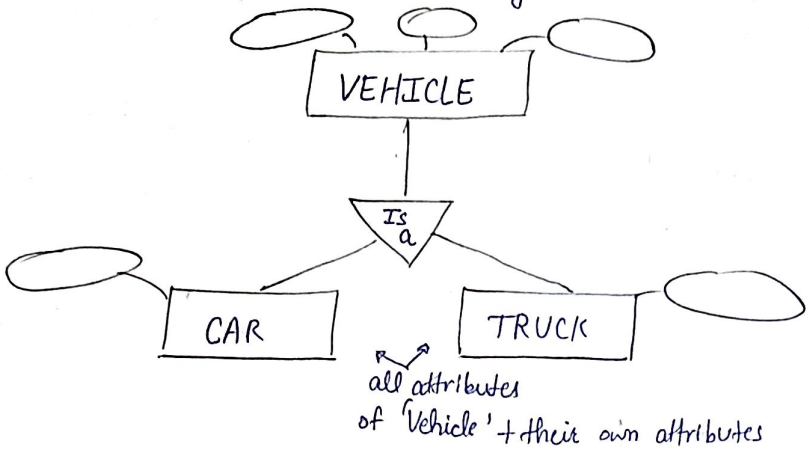
:

(III) ER diagram



* Extended ER diagram (EER): also represents hierarchies

  • Specialization – process of defining subclasses from a superclass (sub-entities from an entity)

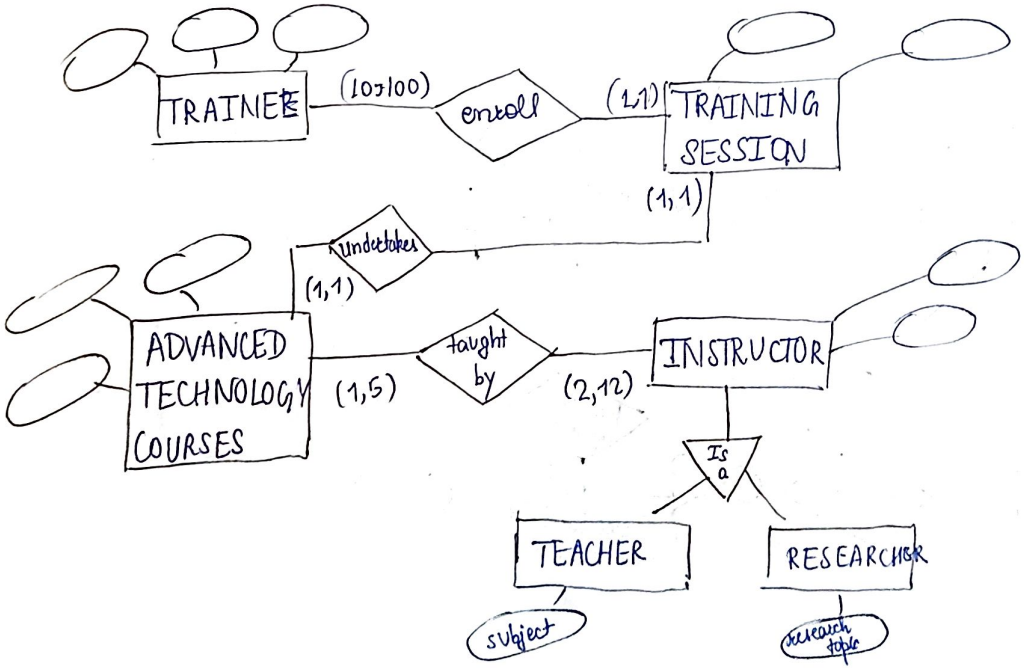


  • Generalization – process of combining entities into a generalized superclass

Case Study:

Design ER diagram for IT training group DB that'll need the i/f for its training program. Clearly indicate the entities, relationships and key constraints. The descr of the environment is as follows:

the company has 12 instructors and can handle upto 100 trainees for each training session, the company offers 5 advanced technology courses – each of which is taught by a team of 2 or more instructors, each instructor is assigned to a max^{m} of 2 teaching teams or maybe assigned to do research, each trainee undertakes 1 advanced tech course per training session



Case Study:

Design ER diagram for a company DB as per the requirements given below. Make appropriate assumptions to complete the specifications.

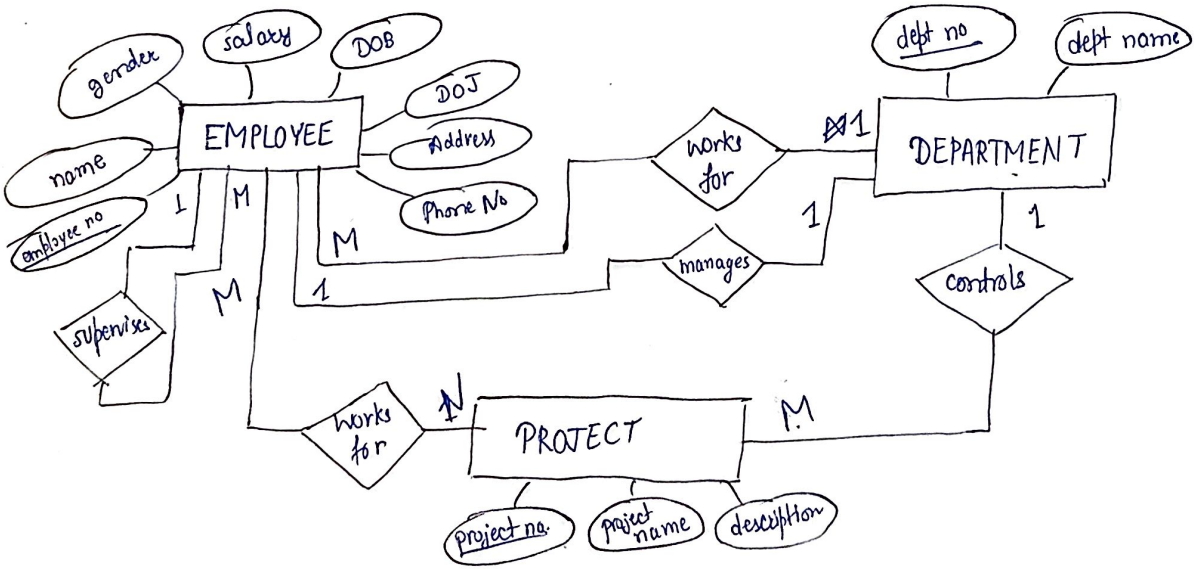
(I) The company stores the i/f about the current working employees. The i/f includes employee no., name, gender, salary and DOB, Date of joining, address and phone no. Each employee works for a dept. for a particular project for a particular no. of hours.

(II) The i/f about dept includes dept. no, dept. name. Each dept. controls some project currently running in the company. Also each dept. is managed by a particular employee who becomes the managers for the dept. These employees also supervise all other employees in that dept.

(III) The project i/f includes project no., project name, description

(IV) An ~~company~~ employee can only work for only 1 dept, however a dept. can have any no. of employees. A dept. is managed by only 1 manager and a manager can manage only 1 dept. A dept. can control only ~~1~~ many projects, however 1 project can be handled by only 1 dept. Any no. of employees can work on any no. of projects.

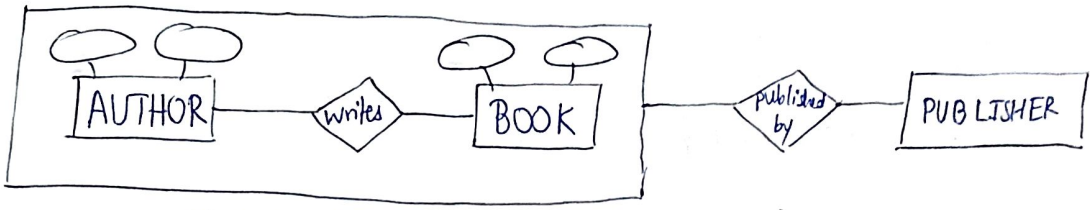
Specify the key attributes of each entity type, name and mapping cardinality. ~~ratio~~.



Case Study:

Consider a bank gateway's DB having customer, loan, account, employee and branch as entity types. Each branch allows customers to open accounts and borrow loans. A customer can open more than one account and one account may also belong to one or more customers. Similarly, a customer can takeout more than one loan and a loan may be held by more than one customer. The bank employs a no. of employees working in different branches of the bank. Add appropriate attributes to each entity type. Represent the key attribute, weak entity type and cardinality ratios. Make appropriate assumptions to complete the specification. Draw ER diagram for bank DB.

* Aggregation: considering two or more entities and their relationships as a higher-level entity
  - the process to ~~represent~~ consider high-level entity constituting of two or more entities and their relationships.



High-level entity

Relational Model

- Relation: representing attributes and values in tabular form
- Relational DB: collection of similar relations.

* Degree of Relation: no. of attributes (columns)
* Cardinality of Relation: no. of tuples (rows)

* Relational Database Instance: at a particular time (instant), the state of the DB.

Keys:

(I) Super Key — set of attributes through which one tuple can be uniquely identified

(II) Candidate Key — minimal subset of super key through which a tuple is uniquely identified

(III) Primary Key — the candidate key chosen to uniquely identify each tuple (NOT NULL + UNIQUE)

(IV) Unique Key — takes unique value but can also take NULL values

Characteristics of Relational Model:

- Order of Tuples
- Ordering values of tuples
- Unknown values for any attribute — NULL
- No two tuples are same

(V) Partial Key — in weak entity set, ~~one~~ attribute that behaves like Primary Key

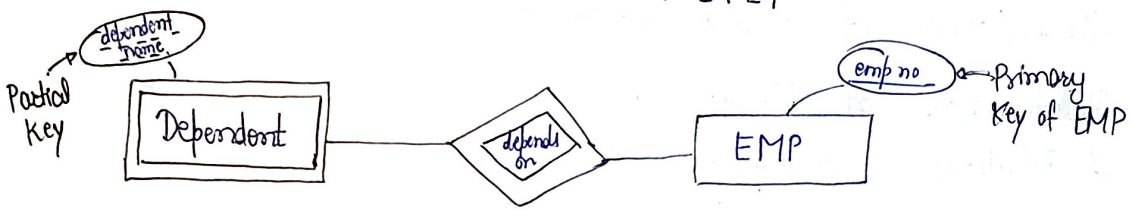
SUP (Sno, Sname, city, status, address, date)

{Sno, city}
{Sno, Sname, address}
{Sno, city, status}
{Sno, Sname, city}
{Sname, address, status}
} Super Keys

{Sno}, {Sname, address} } Candidate keys (minimal subset of super keys)

{Sno} ← PRIMARY KEY

{Sname, address} ← ALTERNATE KEY



(VI) FOREIGN KEY — the attribute of one relation can be accessed in another relation by enforcing a link b/w the attributes of two relations.

- A relation that references another relation is known as "Referencing relation"
- A relation that is being referenced is known as "Referenced relation"

EMP

Eno	Ename	Address	Dno
10	aaa	Patna	1A
11	bbb	Delhi	1A
12	ccc	Mumbai	1B
13	ddd	Kolkata	1D

1D → inconsistency

DEPT

Dno	Dname
1A	B.Tech
1B	BCA
1C	MCA

Dno → PRIMARY KEY of
FOREIGN KEY of

* Self-referencing foreign key:

EMP

Eno	Ename	Addres	Mno
10	--	--	12
12	--	--	NULL
13	--	--	14
14	--	--	12

Eno ← Primary key

Mno ← self-referencing foreign key

Integrity Constraints ← set of rules imposed on the model

(i) Entity Integrity Constraints — all Primary Keys are UNIQUE and NOT NULL

(ii) Domain (Integrity) Constraint — each attribute should take a range of values called the "domain" of that attribute

(iii) Referential Integrity Constraint — if the domain of an attribute 'S_1' of a relation 'R_1' should be same as the domain of an attribute 'S_2' of a relation 'R_2', if:

$$R_1 \rightarrow S_1 \text{ (P.K)}$$
$$R_2 \rightarrow S_2 \text{ (F.K)}$$

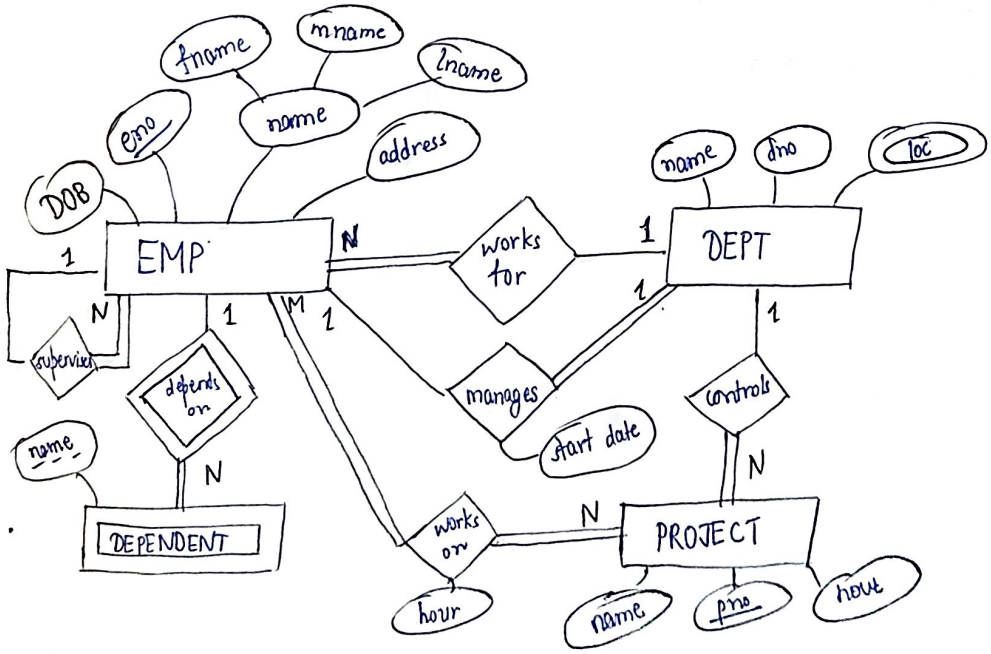
- each tuple value for the F.K should be from the domain of 'S_1', or NULL.

(i) domain for a given set of attributes say 'S_1' in a relation 'R_1' should be the values appear in a certain set of attributes (say) 'S_2' in the 'R_2' relation the attribute in 'S_1' is same as the attribute in 'S_2' in 'R_2' relation, so we can say it refers to 'R_2'

(ii) A value of foreign key in tuple 'T_1' of the current state, either occurs as a value of primary key for some tuple 'T_2' or it is NULL, so:

$$T_1[\text{F.K.}] = T_2[\text{P.K.}] \;||\; \text{NULL}$$

Conversion of E-R model to Relational model



EMP (DOB, eno, fname, mname, lname, address, dno (F.K.), sno (self F.K))

DEPT (name, dno, eno, start date)

DEPT-location (dno, loc)

DEPENDENT (eno (F.K), name)

PROJECT (pno, name, hour, F.K → dno)

WorksOn (eno, pno, hour)

(i) Composite attributes are put as simplest attributes in relational model

(ii) For multi-valued attributes, we have a separate table (PK → attribute + PK of original table)

(iii) For weak-entity sets, combined attribute of its partial key and primary key of the strong-entity set it is dependent on, acts as PK of that table.

(iv) For 1:1 relations, the part having full participation includes the PK of the other part and any attribute of the relation is also added.

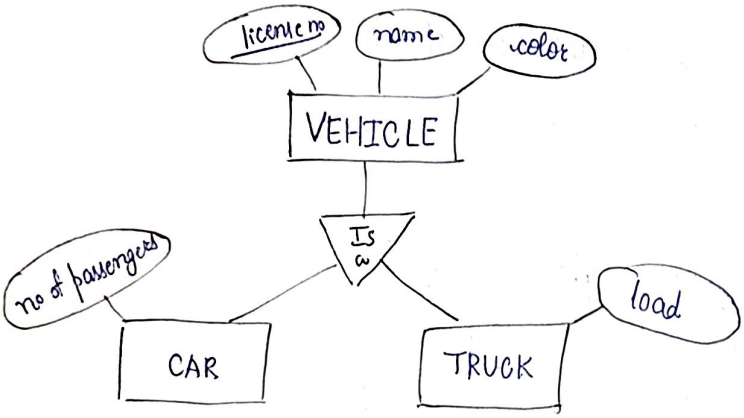
(v) For 1:1 relations, if both sides are full participation, both tables are merged with PK = combined PKs of both tables

(vi) For 1:N relation, the side with 'N' includes the P.K of the side with '1' as F.K.

(vii) For M:N relation, a table is created with P.K = combined P.Ks of both table.

(viii) For self-referential F.K, a self-referential F.K. is added to that table

(ix) For ternary (1:N) relation, a table is created with P.K = P.Ks of all three entities' P.K

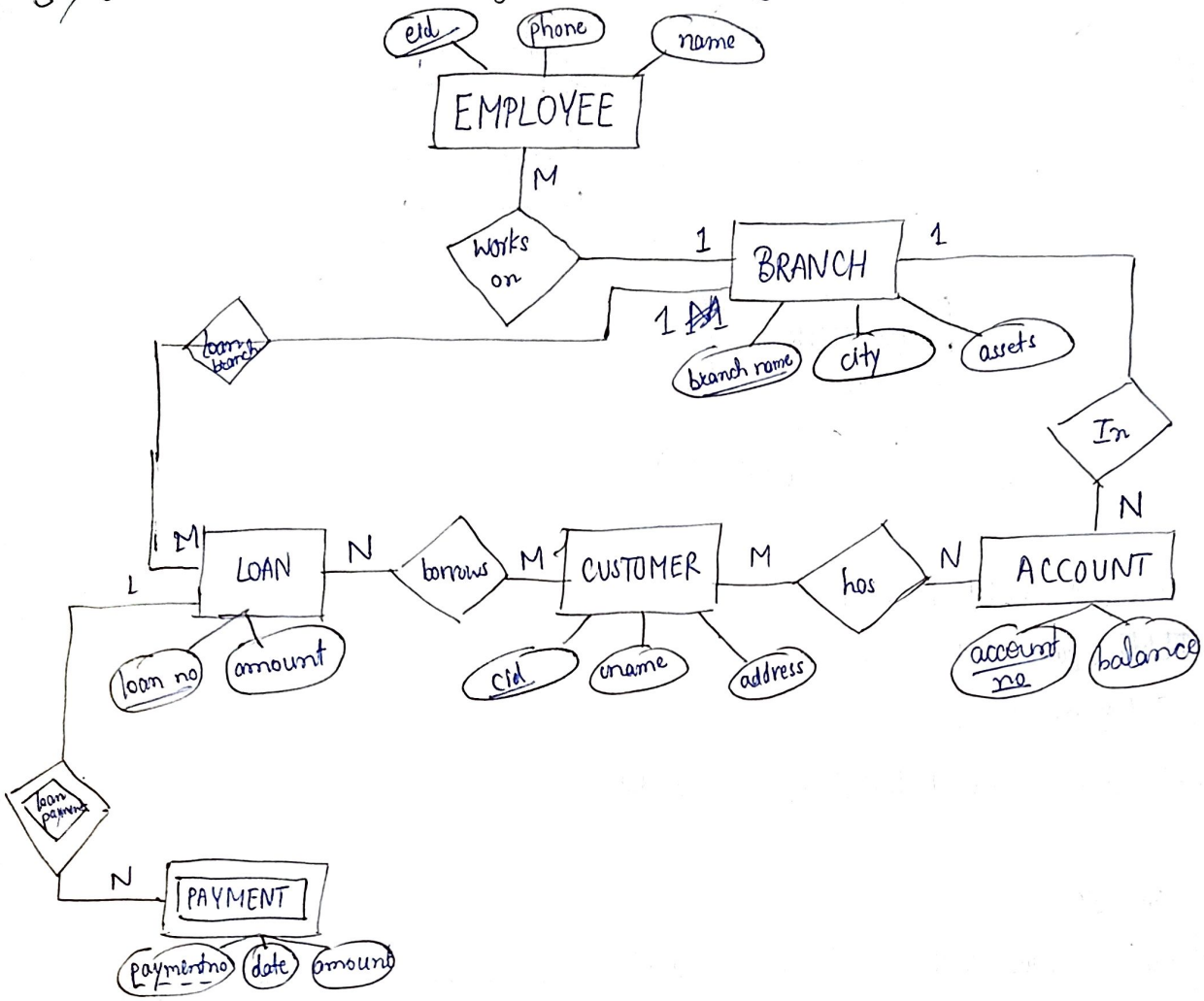


VEHICLE (license no., name, color)

CAR (license no., name, color, no. of passengers)

TRUCK (license no., name, color, load)

Q.) Draw relational model for the following E-R model



EMPLOYEE (eid, phone, name, branchname)

BRANCH (branch name, city, assets) ~~loan no~~

ACCOUNT (account no, balance, branch name)

CUSTOMER (cid, cname, address)

LOAN (loan no, amount, branch name)

PAYMENT (payment no, loan no, date, amount)

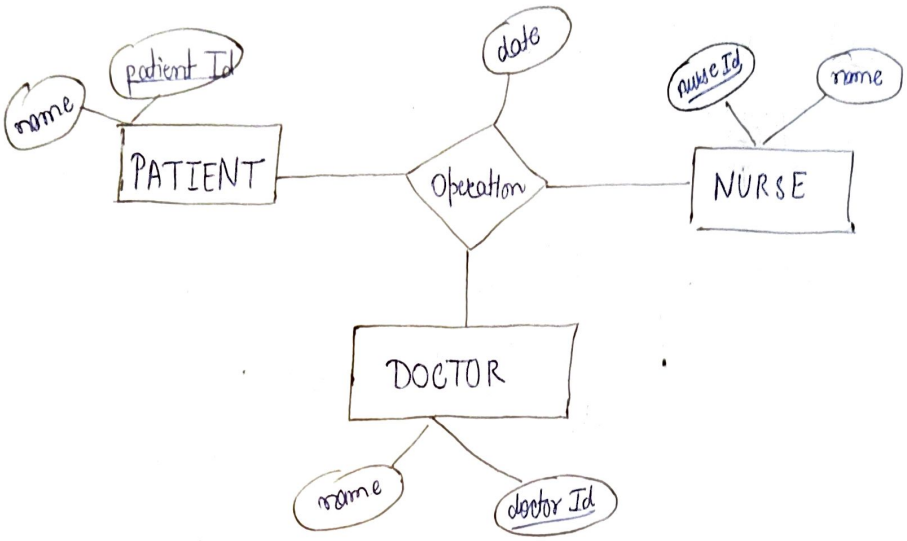
BORROW (cid, loan no)

DEPOSIT ~~HAS~~ (cid, account no)

or

CUSTOMER – ACCOUNT

Ternary relation to relational model



DOCTOR (doctorId , name)

PATIENT (patientId , name)

NURSE (nurseId , name)

OPERATION (patientId, doctorId, nurseId, date)

- Updation in SQL:

  update <tablename> set <columnname> = <new value> where <condition>

  e.g.- emp (eno, name, address, dob, designation, salary);

  update emp set address = "Delhi" where address = "Patna";

  update emp set salary = 60000 where salary > 10000 and address = "Mumb

- Deletion in SQL:

  delete from <tablename> where <condition>

  e.g.- delete from emp where salary < 10000 and address = "Patna";

- Adding new column in SQL:

  update <tablename> add (<new column name> datatype (size));

  e.g.- updata emp add (doj date);

- Select in SQL:

  Select <columnname1, columnname 2> from <table name> where <condition>;

  eg- Select (*) from emp order by salary desc;

  (*) → all columns; order by → shows in ascending; desc → shows in descending (if present)

Professor (PID, pname, dept)

Course (code, dept Id, cname, syllabus)

Prof_course (PID, code, semester)

Q) Identify PK and FK for schema and populate this relation with some tuples and give some examples of insertion operation in the Professor_post course rel" that violates the referential integrity constraints, and of another that doesn't. Identify various candidate keys for this schema. Make necessary assumptions wherever necessary.

CUSTOMER (custno, cust_name, address)

ORDER (orderno, orderdate, custno, city, amount)

PRODUCT (prodno, price, orderno)

Q) Specify FK constraints for above DB. Also insert some tuples in the relation. ~~set~~ Show some e.g. of deletion of tuples that violates referential integrity constraints. Make any assumptions whenever necessary.

Relational Algebra

- Formal language for representing relational model

(I) Unary operator — σ, Π

(II) Binary operator — $\cup, \cap, X, -, \div, \bowtie$

σ (Select Operator)

Syntax: $\sigma_{\text{<condition>}}(\text{<tablename>})$

e.g.- emp (ename, id, sal, dno, address)

$\sigma_{\text{address = "Patna"}}(\text{emp})$ ↑ equal to

$\sigma_{\text{address = "Patna"} \wedge \text{sal} \geq 10000}(\text{emp})$ ↑ and

$\sigma_{\text{address = "Patna"} \vee \text{"Delhi"}}(\text{emp})$ ↑ or

$\sigma_{\text{sal} <> 10000}(\text{emp})$ ↑ not equal to

* for retrieval of tuples

Π (Projection)

Syntax:

$$\Pi_{\text{<column name1, column name 2, -- >}}(\text{<table name>})$$

* for retrieval of columns

e.g.- $\Pi_{\text{ename, sal}}(\text{emp})$ → to select name and sal from emp

Q:) Select name and sal of all employees that live in Patna having salary greater than 10000

$$R_1 \leftarrow \sigma_{\text{address} = \text{"PATNA"} \wedge \text{sal} > 10000}(\text{emp})$$

R_1:

ename	id	sal	dno	address
abc	--	12000	--	PATNA
def	--	20000	--	PATNA

$$\text{Result} \leftarrow \Pi_{\text{ename, sal}}(R_1)$$

Result:

ename	sal
abc	12000
def	20000

OR

$$\text{Result} \leftarrow \Pi_{\text{ename, sal}}(\sigma_{\text{address} = \text{"PATNA"} \wedge \text{sal} > 10000}(\text{emp}))$$

$\cup$ (Union operator)

- Degree of both relations (no. of attributes) is same

R_1:

id	name
01	aaa
02	bbb
03	ccc

R_2:

id	name
04	ddd
05	eee
06	fff
01	aaa

$R_1 \cup R_2$:

id	name
01	aaa
02	bbb
03	ccc
04	ddd
05	eee
06	fff

∩ (Intersection Operator)

$R_1 \cap R_2$:

id	name
01	aaa

− (Difference Operator)

$R_1 - R_2$:

id	name
02	bbb
03	ccc

$R_2 - R_1$:

id	name
04	ddd
05	eee
06	fff

X (Cartesian Product)

emp

eno	ename	sal	dno
2A	aaa	1000	1A
2B	bbb	2000	1A
2C	ccc	3000	1B
2D	ddd	4000	1C

dept

dno	dname
1A	BTech
1B	BCA
1C	BBA

emp X dept:

eno	ename	sal	dno	~~dname~~ dno	dname
2A	aaa	1000	1A	1A	BTech
2A	aaa	1000	1A	1B	BCA
2A	aaa	1000	1A	1C	BBA
2B	bbb	2000	1A	1A	BTech
2B	bbb	2000	1A	1B	BCA
2B	bbb	2000	1A	1C	BBA
2C	ccc	3000	1B	1A	BTech
2C	ccc	3000	1B	1B	BCA
2C	ccc	3000	1B	1C	BBA
2D	ddd	4000	1C	1A	BTech
2D	ddd	4000	1C	1B	BCA
2D	ddd	4000	1C	1C	BBA

⋈ (Join Operator)

emp ⋈ dept:

(i) "Natural Join" → only has the common attribute once.

eno	ename	sal	dno	dname
2A	aaa	1000	1A	BTech
2B	bbb	2000	1A	BTech
2C	ccc	3000	1B	BCA
2D	ddd	4000	1C	BBA

(ii) "Equi Join"

↳ joined without any condition, only based on common attribute

emp(dno) = dept(dno)

eno	ename	sal	dno	dno	dname
2A	aaa	1000	1A	1A	BTech
2B	bbb	2000	1A	1A	BTech
2C	ccc	3000	1B	1B	BCA
2D	ddd	4000	1C	1C	BBA

Natural Join

- ↳ Inner Join (⋈) → all matching tuples only
- ↳ Outer Join
  - → Left (⟕) → all matching tuples + left unmatched ones
  - → Right (⟖) → " " " + right " "
  - → Full (⟗) → " " " + both " "

÷ (Division Operator)

. "for all" objects having specific properties

$R_1 \div R_2$: set of all attributes 'A' of R_1 such that

R_1:

A	B
a_1	b_1
a_1	b_2
a_1	b_3
a_1	b_4
a_2	b_1
a_2	b_2
a_3	b_4
a_4	b_2
a_4	b_4

R_3:

B
b_2
b_4

$R_1 \div R_3$:

A
a_1
a_4

$\mathcal{G}$ (aggregation operator)

$\mathcal{G}$ (emp)

dname $\mathcal{G}$ max(sal), avg(sal), min(sal), sum(sal)

"Grouping function" or "Aggregation function"

Q.) CUSTOMER (custno, cname, city)

ORDER (orderno, orderdate, custno, amount)

ORDER_ITEM (orderno, itemno, quantity)

ITEM (itemno, unitprice)

(I) Retrieve the order no and orderdate placed by the customer residing in "PATNA" city.

$R_1 \leftarrow$ (CUSTOMER $\bowtie$ ORDER)

natural join

$\bowtie_{\text{CUSTOMER.custno = ORDER.custno}}$

for common attributes (tablename.attributename)

$R_2 \leftarrow \sigma^{(R_1)}_{\text{city = "PATNA"}}$

Result $\leftarrow \Pi^{(R_2)}_{\text{orderno, orderdate}}$

(OR)

Result $\leftarrow \Pi_{\text{orderno, orderdata}} \left(\sigma_{\text{city = "PATNA"}} (\text{CUSTOMER} \bowtie_{\text{CUSTOMER.custno = ORDER.custno}} \text{ORDER}) \right)$

(II) Retrieve the order no and unit price for item for which an order of Qty > 50.

$\Pi_{\text{orderno, unitprice}} \left(\sigma_{\text{quantity > 50}} (\text{ORDER\_ITEM} \bowtie_{\text{ORDER\_ITEM.itemno = ITEM.itemno}} \text{ITEM}) \right)$

(III) Retrieve the order no, date and item no for order of item having the price > 20

$\Pi_{\text{orderno, orderdate, itemno}} \left(\sigma_{\text{unitprice > 20}} ((\text{ORDER} \bowtie_{\text{ORDER.orderno = ORDERITEM.orderno}} \text{ORDER\_ITEM}) \bowtie_{\text{ITEM.itemno = ORDER\_ITEM.itemno}} \text{ITEM}) \right)$

(iv) Retrieve the no. and unitprice of item for which an order is placed by customer C001

$$\Pi_{itemno,\ unitprice}\left(\sigma_{CUSTNO = C001}\left((ORDER \bowtie_{ORDER.orderno = ORDER\_ITEM.orderno} ORDER\_ITEM) \bowtie_{ORDER\_ITEM.item\ no = ITEM.itemno} ITEM\right)\right)$$

Q.) Employee (empname, street, city)

Works (empname, compname, sal)

Company (compname, city)

Manages (empname, managername)

(i) Find the names of all employees who work for SBI.

$$\Pi_{empname}\left(\sigma_{compname = \text{'SBI'}}(Works)\right)$$

~~Employee~~ ⋈ ~~Company~~ Employee

(ii) Find name, street address and cities of residence of all employees who work for SBI and earn more than 20K p.a.

$$\Pi_{empname,\ street,\ city}\left(\sigma_{compname = \text{'SBI'} \wedge\ sal\ >20000}\left(Employee \bowtie_{Employee.empname = Works.empname} Works\right)\right)$$

(iii) Find names of all employees in the DB who live in the same city as the company for which they work

$$\Pi_{empname}\left(\sigma_{Company.city = Employee.city}\left(\left(Works \bowtie_{Works.compname = Company.compname} Company\right) \bowtie_{Works.empname = Employee.empname} Employee\right)\right)$$

(IV) Find the no. of employees working in each company

$$\gamma_{\text{compname}}\, \mathcal{C}_{\text{count(empname)}}\,(\text{Works})$$

(V) Find the avg, max^m and min^m salaries for each company

$$\gamma_{\text{compname}}\, \mathcal{C}_{\text{avg(sal), max(sal), min(sal)}}\,(\text{Works})$$

Q:) Author (AID , aname, state, city, zip, phone, URL)
Book (ISBN , booktitle, category, price , copyrightdate, year, page-count, pid)
Publisher (Pid , pname, address, state, compname, phone, emailid)
Author-Book (AID, ISBN)
Review (Rid, ISBN, Rating)

(I) Retrieve the city, phone, and URL of Author whose name is Ram

$$\Pi_{\text{city, phone, URL}}\left(\sigma_{\text{aname='Ram'}}(\text{Author})\right) \;\Big|\; \{t.\text{city}, t.\text{phone}, t.\text{URL} \mid \text{Author}(t) \wedge t.\text{name} = \text{'Ram'}\}$$

(II) Retrieve name, address and phone of all publishers, located in Bihar state

$$\Pi_{\text{pname, address, phone}}\left(\sigma_{\text{state='Bihar'}}(\text{Publisher})\right) \;\Big|\; \{t.\text{name}, t.\text{address}, t.\text{phone} \mid \text{Publisher}(t) \wedge t.\text{state} = \text{'Bihar'}\}$$

(III) Retrieve title and price of all textbooks with a pagecount > 600.

$$\Pi_{\text{booktitle, price}}\left(\sigma_{\text{page\_count}>600 \wedge \text{category='textbook'}}(\text{Book})\right) \;\Big|\; \{t.\text{booktitle}, t.\text{price} \mid \text{Book}(t) \wedge t.\text{page\_count} > 600 \wedge t.\text{category} = \text{'textbook'}\}$$

(IV) Retrieve the ISBN, title and price of the books belonging to either novel or language category.

$$\Pi_{\text{ISBN, booktitle, price}}\left(\sigma_{\text{category='Novel'} \vee \text{category='Language'}}(\text{Book})\right) \;\Big|\; \{t.\text{ISBN}, t.\text{booktitle}, t.\text{price} \mid \text{Book}(t) \wedge (t.\text{category} = \text{'Novel'} \vee t.\text{category} = \text{'Language'})\}$$

(v) Retrieve the id, name, address and phone of the publishers publishing novels

$$\Pi_{Publish.pid,\ pname,\ address,\ phone}\left(\sigma_{category='novel'}\left(Book \bowtie_{Publisher.pid=Book.pid} Publisher\right)\right)$$

$$\{t.pid, t.pname, t.address, t.phone \mid Publisher(t) \wedge Book(s) \wedge s.pid = t.pid \wedge s.category = 'Novel' \}$$

(or) $\wedge (\exists s)\, Book(s)$

(vi) Retrieve title and price of all the books published by "Hills Publications".

$$\Pi_{booktitle,\ price}\left(\sigma_{pname='Hills\ Publication'}\left(Book \bowtie_{Publisher.pid=Book.pid} Publisher\right)\right)$$

(vii) Retrieve title, reviewer's id and rating of all textbooks

$$\Pi_{booktitle,\ Rid,\ Rating}\left(\sigma_{category='textbook'}\left(Book \bowtie_{Book.ISBN=Review.ISBN} Review\right)\right)$$

$$\{t.booktitle, S.Rid, s.Rating \mid Book(t) \wedge Rating(s) \wedge t.ISBN = s.ISBN \wedge t.category = 'textbook'\}$$

(viii) Retrieve id, name, URL of the authors and category of book C++

$$\Pi_{AID,\ aname,\ URL}\left(\sigma_{category='C++'}\left(\left(Book \bowtie_{Book.ISBN=Author\text{-}Book.ISBN} Author\text{-}Book\right) \bowtie_{Author.AID=Author\text{-}Book.AID} Author\right)\right)$$

(ix) Retrieve title, category and price of all the books written by 'Sham'.

$$\Pi_{booktitle,\ category,\ price}\left(\sigma_{aname='Sham'}\left(\left(Book \bowtie_{Book.ISBN=Author\text{-}Book.ISBN} Author\text{-}Book\right) \bowtie_{Author.AID=Author\text{-}Book.AID} Author\right)\right)$$

$$\{t.booktitle, t.category, t.price \mid Book(t) \wedge (\exists s)(\exists p)(Author(s) \wedge Author\_Book(p) \wedge t.ISBN = p.ISBN \wedge s.AID = p.AID \wedge s.aname = \text{'Sham'})\}$$

(x) Retrieve name and address of publishers who have not published any book.

$$R \leftarrow \Pi_{pid}^{(Publishers)} - \Pi_{pid}^{(Books)}$$

$$result \leftarrow \Pi_{pname,\ address}\left(R \bowtie_{R.pid=Publishers.pid} Publishers\right)$$

(x) Retrieve the name of all publishers, ISBN, booktitle published by them.

$$\Pi_{pname, ISBN, booktitle} \left(Publishers \bowtie_{Publishers.pid = Book.pid} Book \right)$$

(or) we can also do natural join

(xi) Retrieve id and no. of books written by each author

$$_{AID}\gamma_{count(ISBN)} (Author\text{-}Book)$$

if we want aname, let the above be R,

$$\Pi_{aname} \left(R \bowtie_{R.AID = Author.AID} Author \right)$$

(xii) Retrieve ISBN and average rating given to each book

$$_{ISBN}\gamma_{avg(Rating)} (Review)$$

(xiii) Retrieve the name of the publishers who have published all categories of books.

~~Result~~ $\leftarrow \Pi_{pid, pname}^{(Publishers)} \div \Pi_{pid}^{(Books)}$

~~result~~ $\leftarrow \Pi_{pname} \left(R \bowtie_{R.pid = Publishers.pid} Publishers \right)$

$$\Pi_{pname, category} \left(Books \bowtie_{Books.pid = Publishers.pid} Publishers \right) \div \Pi_{category}^{(Books)}$$

Relational Calculus

(i) Tuple Relational Calculus
(ii) Domain Relational Calculus

Tuple

$\{T \mid P(T)\}$ ← all tuples in table 'P'

- To get name of employee whose sal > 10000

$\{t.name \mid emp(t) \wedge t.sal > 10000\}$

- If F_1 and F_2 are two formulas, we can have:

$F_1 \wedge F_2$, $F_1 \vee F_2$, $F_1 \Rightarrow F_2$, $\neg F_1$, $\neg F_2$

Quantifier

↳ Universal ($\forall$) — "for all"
→ Existential ($\exists$) — "there exists"

$F_1 \wedge F_2$ → true if both are true
$F_1 \vee F_2$ → false if both are false
$F_1 \Rightarrow F_2$ → false if F_1 is true but F_2 is false

$(\exists T)(F)$ → at least one tuple assigned to free occurences of T in F_1 then true, otherwise false
$(\forall T)(F)$ → if F is true for every tuple assigned to free occurences of T in F_1 then true, else false.

Domain

Author (a_0 (aid), a_1 (aname), a_2 (state), a_3 (city), a_4 (zip), a_5 (phone), a_6 (url))

- city, phone, url where aname = 'Ram'

$\{a_3, a_5, a_6 \mid (\exists a_1)\ Author(a_0, a_1, a_2, a_3, a_4, a_5, a_6) \wedge a_1 = \text{'Ram'}\}$

- name, address, phone of publishers in state='Bihar'

$\{P_1, P_2, P_4 \mid (\exists P_3)\ Publisher(P_0, P_1, P_2, P_3, P_4, P_5) \wedge P_3 = \text{'Bihar'}\}$

- id, name, address, phone of publishers publishing novels

$\{P_0, P_1, P_2, P_4 \mid (\exists b_2)(\exists b_3)\ Book(b_0, b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8) \wedge Publisher(P_0, P_1, P_2, P_3, P_4, P_5) \wedge b_7 = P_0 \wedge b_2 = \text{'Novel'}\}$