UNIT - II

DATABASE DESING USING ER DIAGRAMS

a Relational Algebra
b Relational Calculus

Def:

Entity Relationship Model (E-R)

ER model was developed to facilitate Data base design by allowing specification of an enterprise schema that represents the overall structure of data base

Entity:

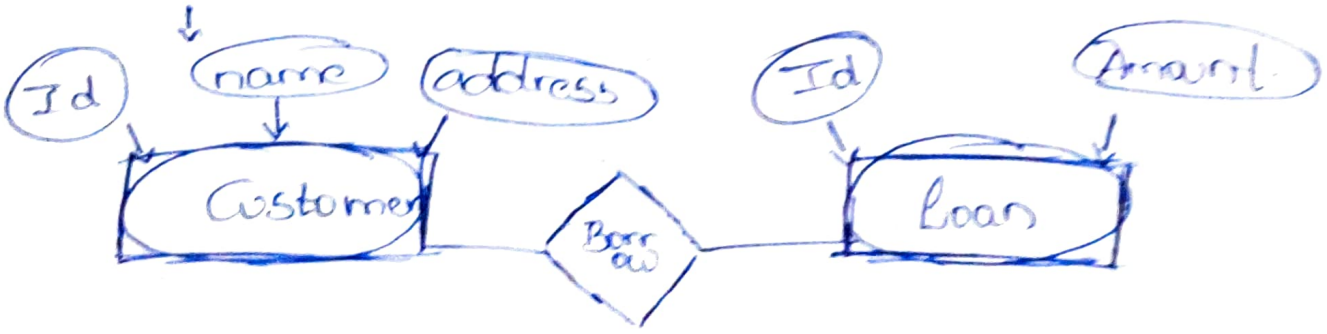
Entity is an object in the real world distinguishable from all other objects

An Entity set is a set of entities of same type that share same properties

Ex: A set of customer in a bank

A Relationship is an association among several entities

Customer is an entity & loan is another entity



A relationship set having an own attribute that attribute is called Descriptive attribute

- Binary relationship set that involves two entity sets,
- Ternary relationship set involves three entity sets.

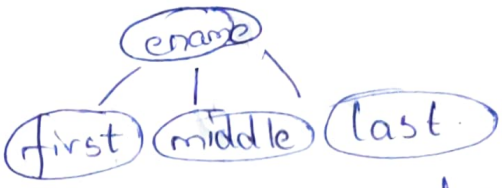
The Degree of relationship set is number of entities participated

Types of Attributes:

i) Simple (vs) Composite
ii) Single (vs) Multi
iii) Stored (vs) Derived.

Simple Attribute: which can not be divided - sal

Composite : Opp to simple attribute



Single : which has only one value Id.

Multi : which may have many values

Stored :

Derived: Deriving from other attribute.

Ex: grades, hiredate.

Constraints:

An ER schema may define certain constraints

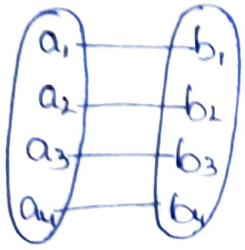
a Mapping Cardinalities

b Key Constraints

c participation Constraints.

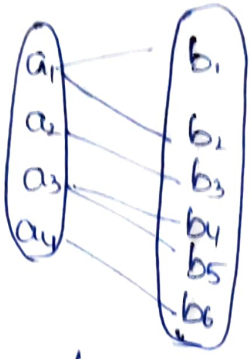
Mapping Cardinalities:

i



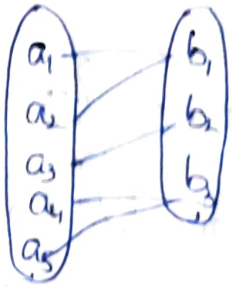
one to one.

ii



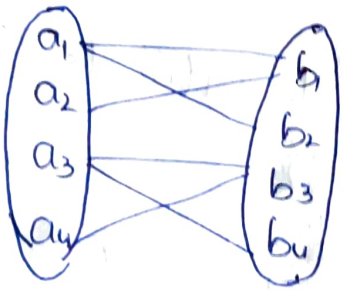
one to many.

iii



many to one.

iv



Many to Many.

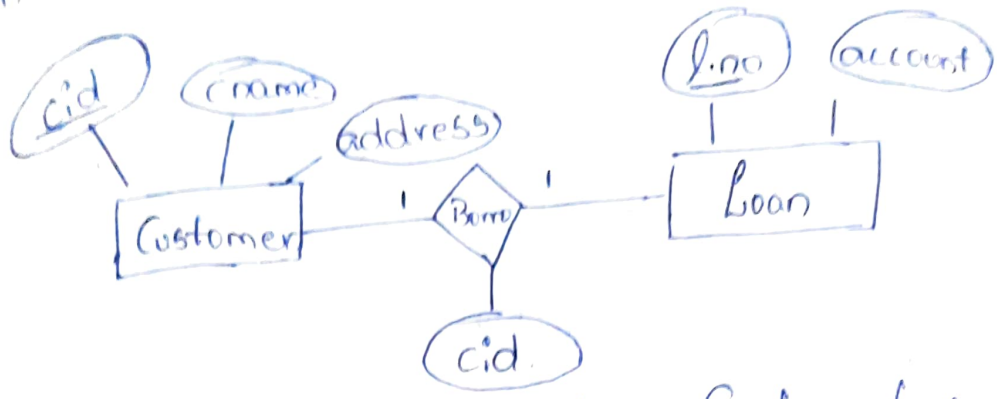
Mapping cardinalities are also called as cardinality ratios it expresses the number of entities to which another entity can be associated through a relationship set.

It is useful in describing relationsets

Defs & Exs:

(i) One to One: An Entity in A is associated with almost one entity in B & an Entity in B is associated with almost one entity in A.

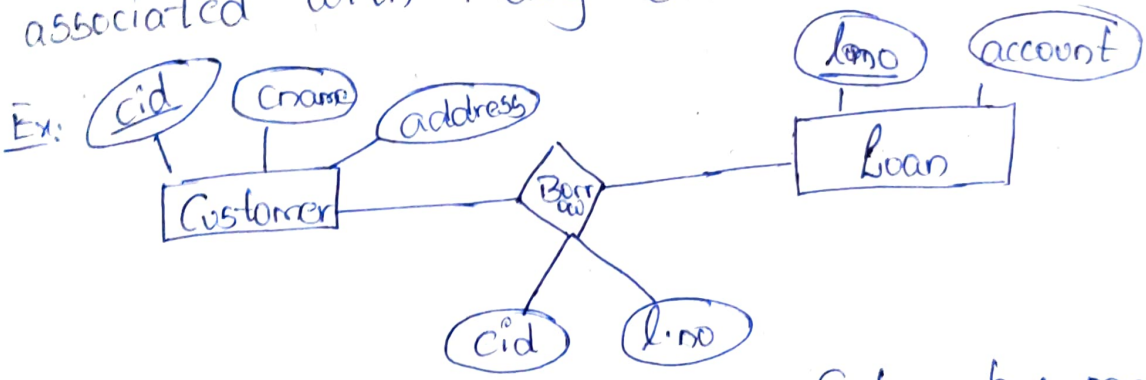
Ex:



A customer has one loan & loan has only one customer. So either of the primary keys of loan (or) Customer can be taken as attribute to borrower relationshipset.

(ii) Many-to Many: An Entity in A is associated with many entities in B & an Entity in B is associated with many entities in A.

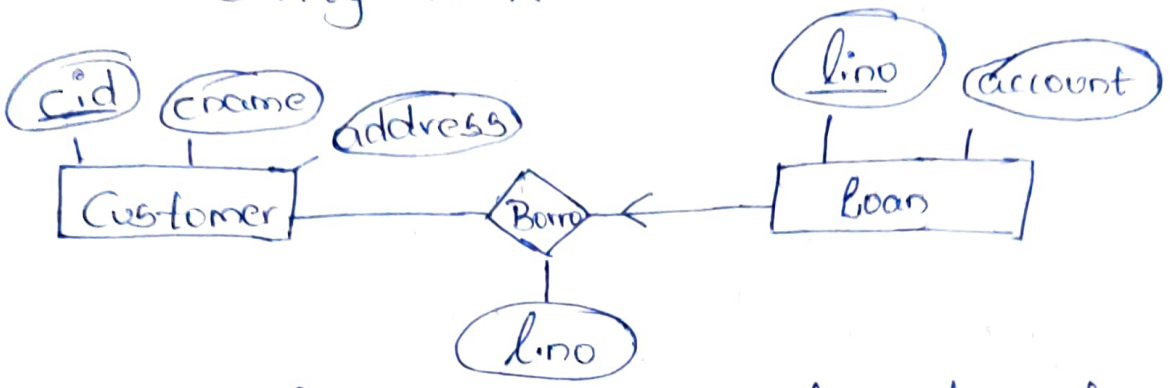
Ex:



A customer has many loans & loan has many customers, then the combination of both the prim keys can be taken attribute to borrower relationshipset (cid, l.no).

iii) One to Many:

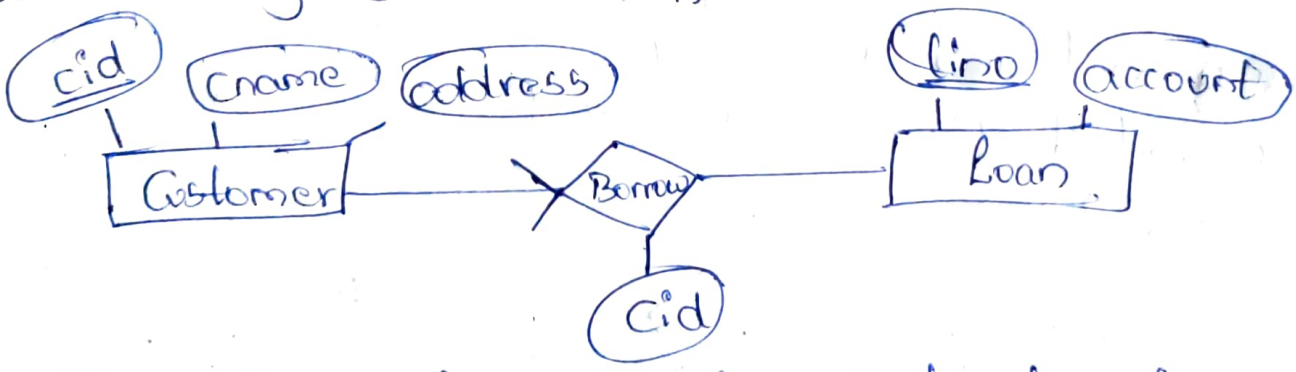
An Entity in A is associated with many entities in B & An Entity in B is associated with one entity in A.



A customer has many loans but a loan has only one customer, so lino is attribute to Relationship set.

iv) Many to One:

An Entity in A is associated with one entity in B & An Entity in B is associated with many entities in A.



A customer has one loan but a loan has many customers, so cid is attribute to relationship set.

Key Constraints:

a Super key:

It is a set of one or more attributes allows you to identify each entity or ~~every~~ unique entity.

No proper subset is a super key. Such minimal super keys are called candidate key

$C_1 \rightarrow$ St. Id. → Prime attribute.

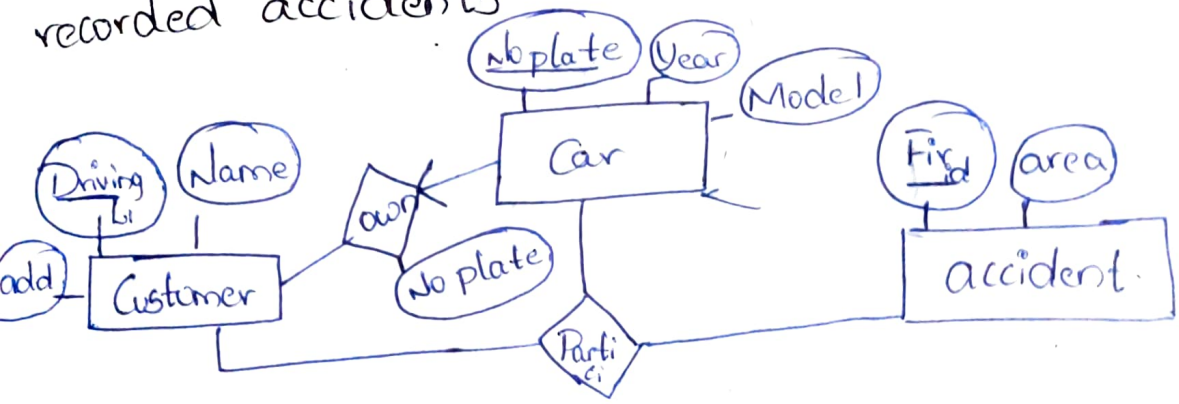
$C_2 \rightarrow$ Adhr no – Alternate key (or) 2ndary key

$C_3 \rightarrow$ (Name, Dob)

Here we choose S.ID as primarkey

- A relationship set may have atmost one primary key.

Design an E-R Diagram for a car insurance company whose customers own one or more cars each, each car is associated with it 0 to any no. of recorded accidents

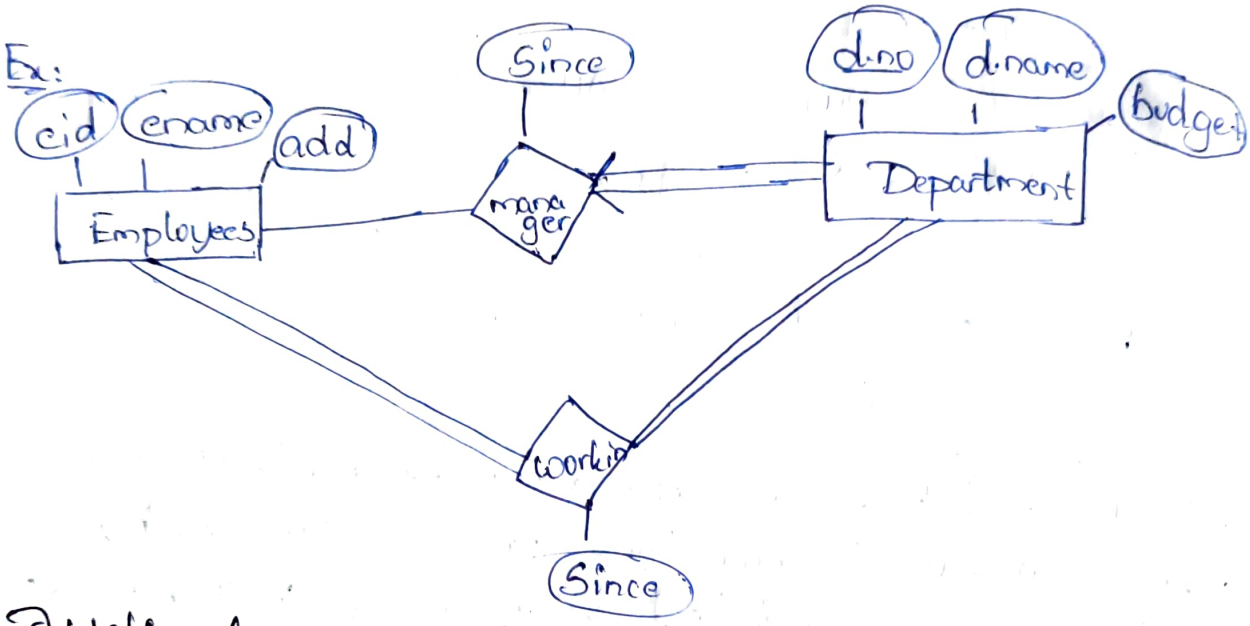


Participation Constrains:

The participation of Entity set E in a relationship set R is said to total if every entity in E participates atleast one relationship in R.

if only some entities in E participates in relationship set R, then it is said to be (employee) partial.

Ex:



Additional Features of ER model:

1. Weak Entity set
2. Specialization & generelaization.
3. Aggregation.

1. Weak Entity set:

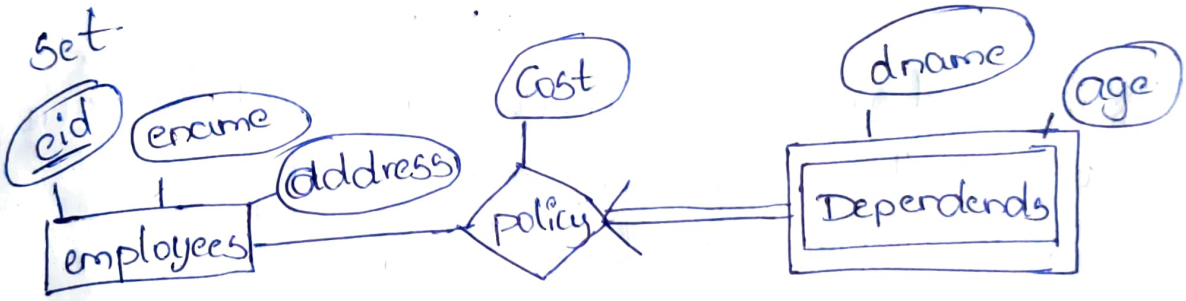
An Entity set may not have sufficient attributes to form a primary key, such entity set is called as "Weak Entity set".

Ex: Employees purchasing to cover their dependends we might choose to identify a dependend through employee id.

The following restrictions must hold by weak entity set

- The owner entity & weak entity set must participate in a one-many relation
- The weak entity must have total participation in relationship set.

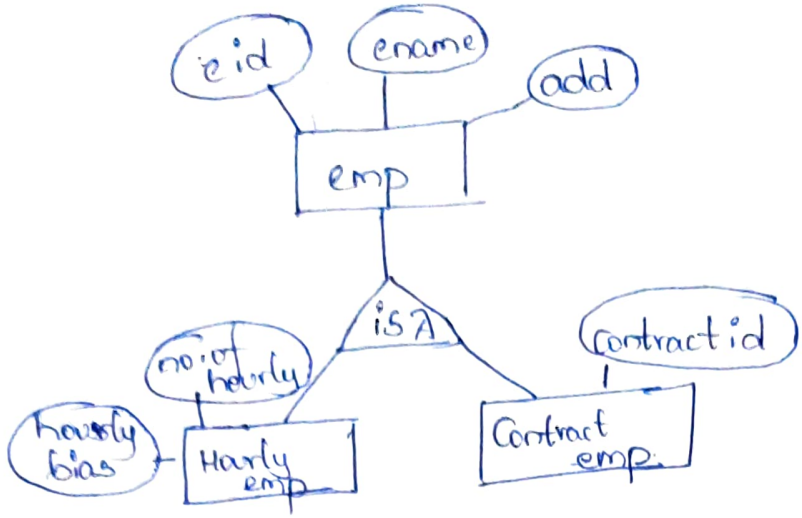
The set of attributes of a weak entity set uniqely identify a weak entity for a given owner entity is called as partial key of weak entity set.



Specialization:

Specialization is the process of identifying subsets of an entity set that shares some distinguish characteristic, so super class is defined first & the sub classes are defined next

Hourley-employes & Contract_employees are constructed from employee entity set



Generelization:

Generelization is identifying common charaterstics and form a collection of entity set and creating a new Entity set. i.e., Subclasses are defined first then super class is defined next.

In these we have two constraints

:: Overlap Constraints: Determines whether two or more subclases allow to contain same entity

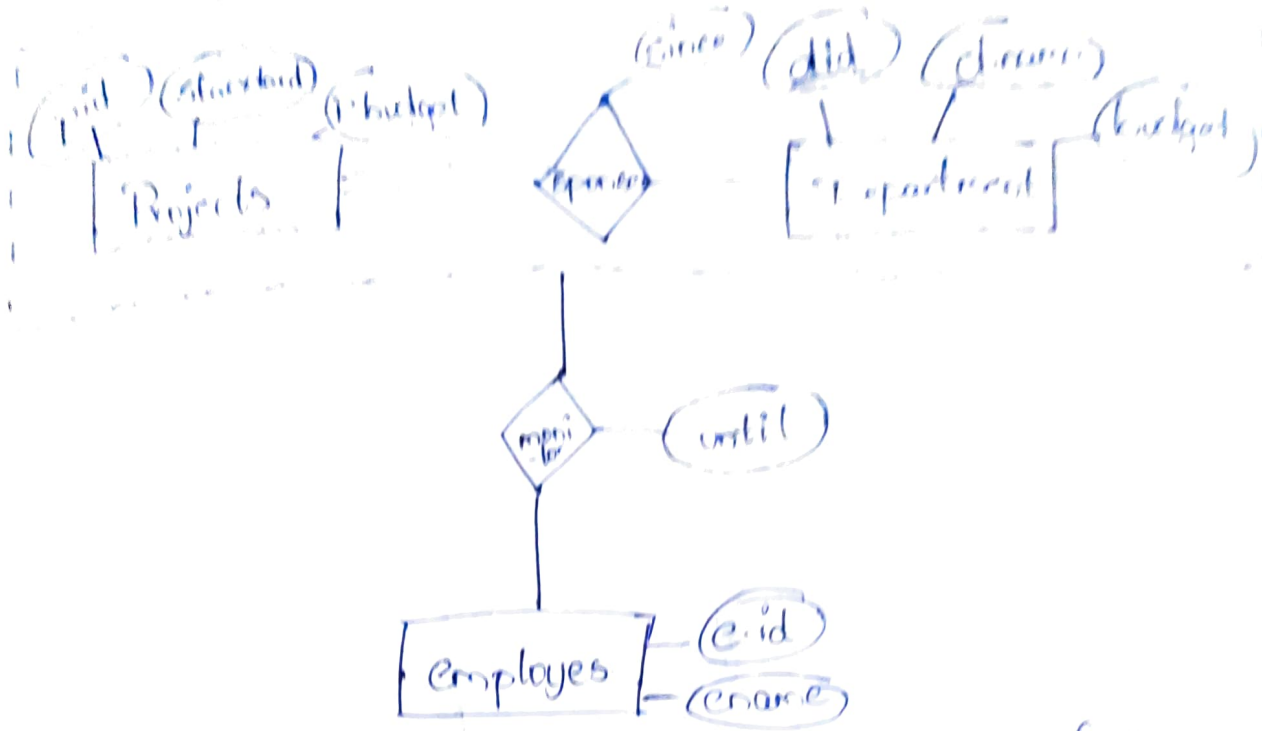
Ex: Hourly employee overlaps senior employees.

:: Covering: It determines whether the entities in the sub class collectively include all the entities in the super class.

The porperty of generalization is that every instance of superclass is an instance of a subclass.

Ex: Cars & bikes cover motor Vehicles.

Aggregation:



To Model a relationship b/w Collection of Entities & Relationship i.e., Aggregation allows us to indicate that a relationship set participates in another relationship set

It is Represented with a dashed box

Ex: Each project is sponsored by 1 (or) more sponsers, the sponsers relationship captures that information.

A department that sponser a project might assign employee to monitor the sponership.

Conceptual Design With the ER Model.

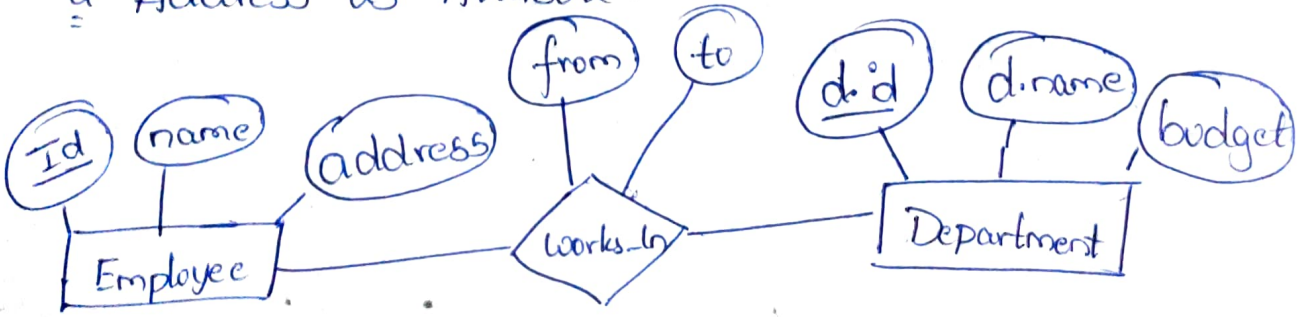
! Entity vs Attribute

It is sometimes not clear whether a property should be modeled as an attribute or as an entity set. Consider adding address information to Employee Entity. One option is to use attribute address. This option is appropriate if we need to record only one address per employee, and it suffices to think of an address as a string.

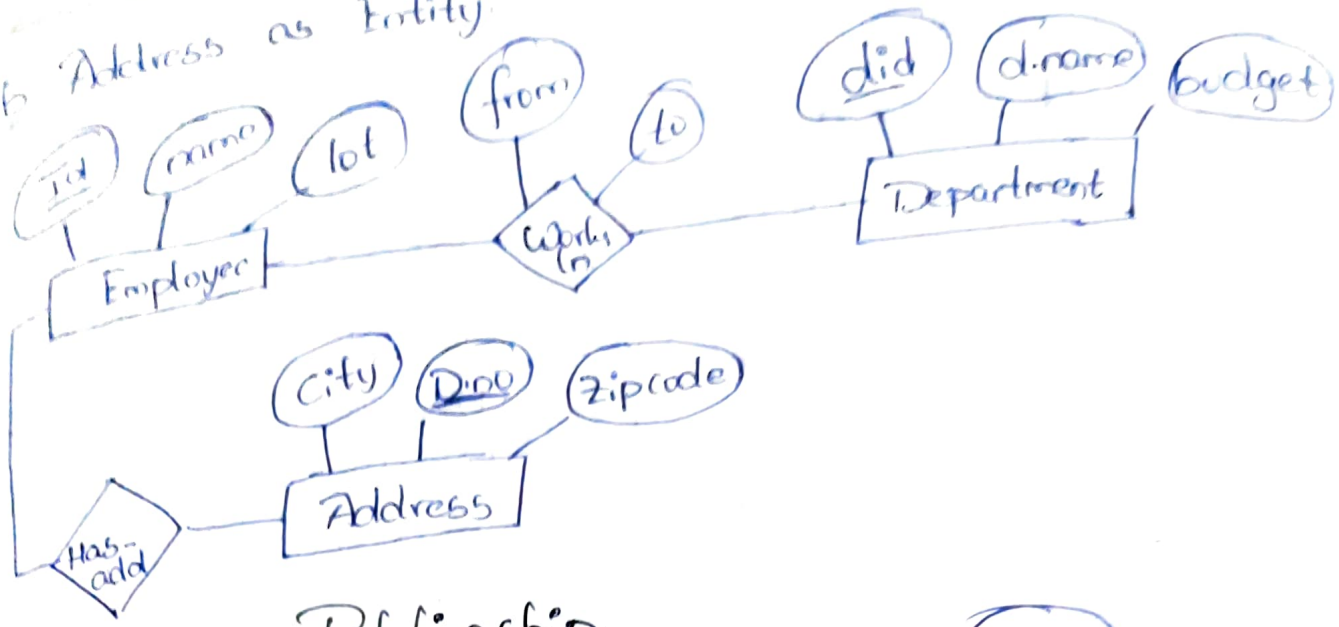
An alternative is to create an entity set called Address and to record associations between employees and address using relationship set (Has_address). This complex alternative is necessary in two situations.

- We have to record more than one address for an employee.
- We want to capture the structure of address in our ER diagram.
  Ex: we can breakdown an address into City, State, Country and Zip code. By representing an address as an entity with these attributes. For detailed Query.

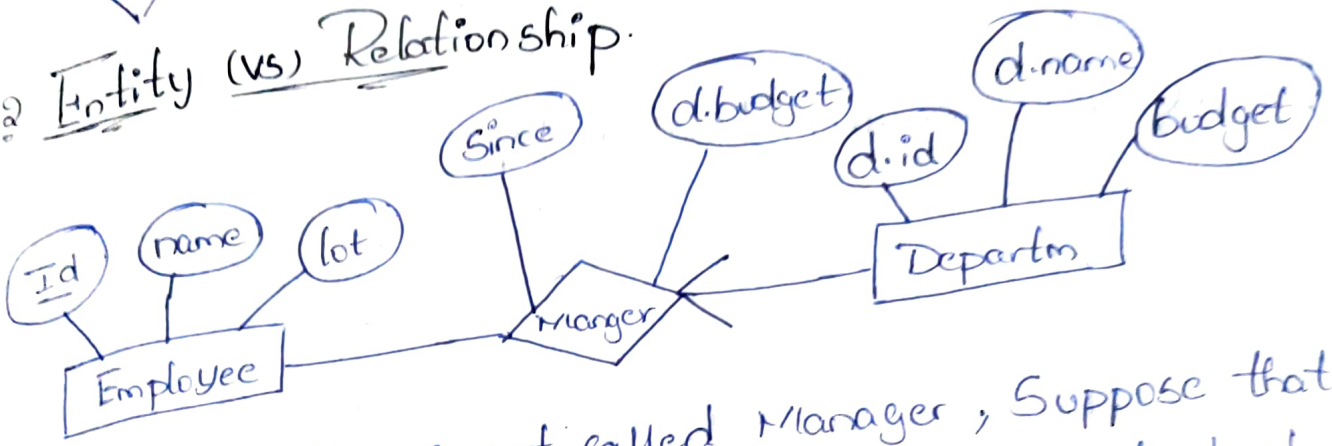
a Address as Attribute.



b. Address as Entity



2. Entity (vs) Relationship.



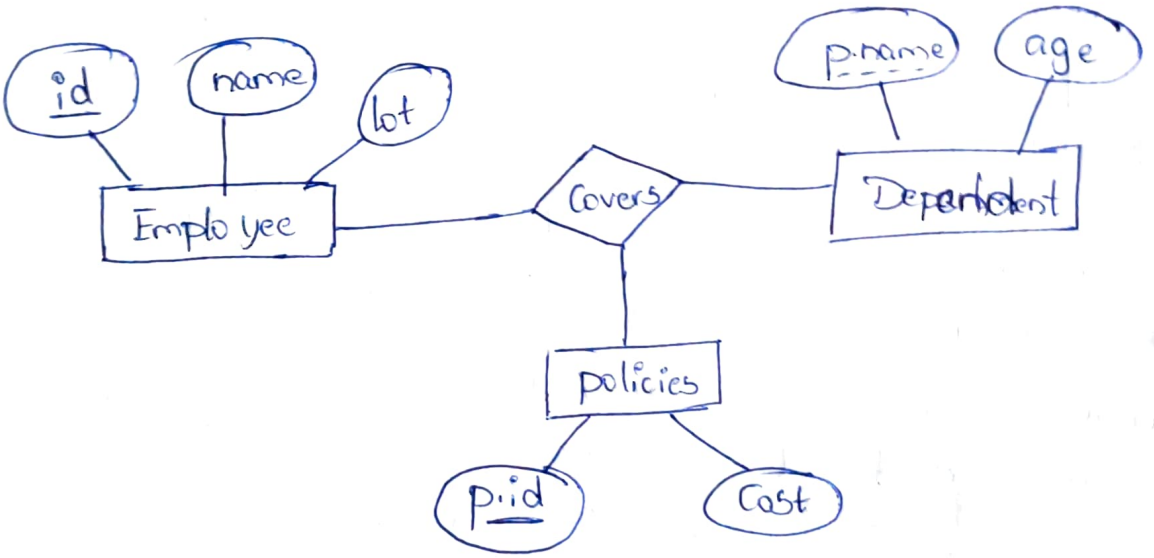
The relationship set called Manager, Suppose that each department is given a discretionary budget Given in department we know the manager, as well as the manager's starting date & budget for that department. This approach is natural.

But what if the discretionary budget is a sum that covers all departments managed by employee. In this case, each Manager relationship that involves a given employee will have the same value in budget field

We can address these problems by introducing a new entity set called Managers. The attributes since and dbudget now describe a manager entity.

As a variation, while every manager has a budget, each manager may have a different starting date for each dept. In this case dbudget is an attribute of Managers, but since is an attribute of relationship set b/w managers and depts.

3 Binary (vs) Ternary Relationships



It models a situation in which an employee can own several policies, each policy can be owned by several employees, & each department can be covered by several policies.

Suppose that we have following requirements

- A policy cannot be owned jointly by two or more employees
- Every policy must be owned by some employee.

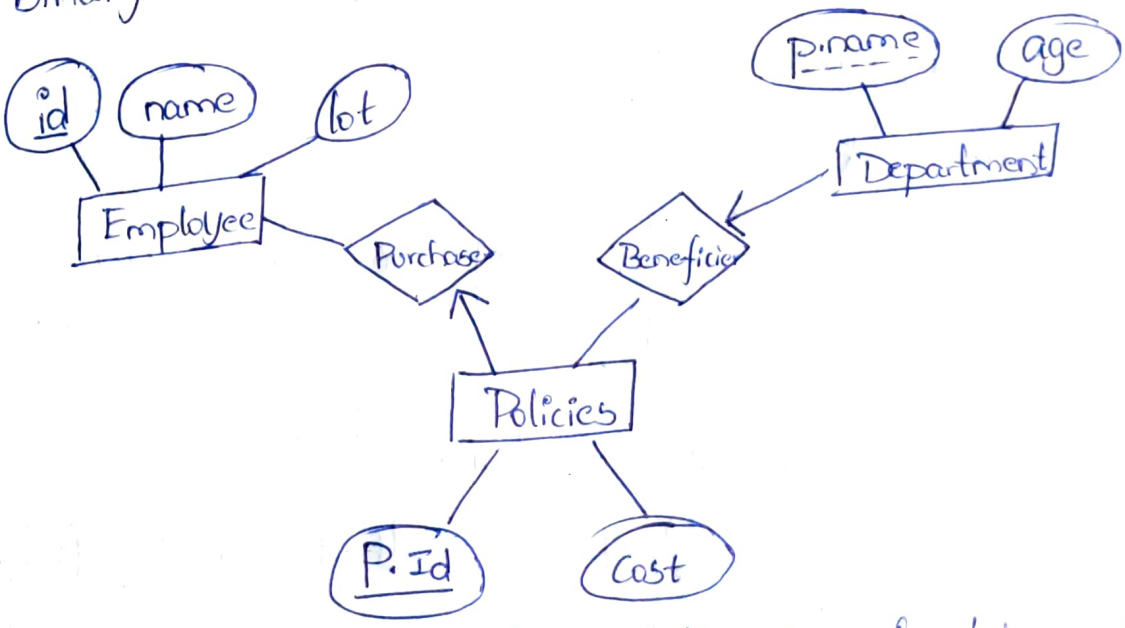
- Dependents is a weak entity set & each dependent entity is uniquely identified by taking pname in conjunction with policyid.

The first requirement suggests that we impose a key constraint on policies with respect to Covers, but this constraint has the unintended side effect that a policy can cover only one dependent

The second requirement suggests that we impose a total participation constraint on policies. This soln is acceptable if each policy covers at least one dependent.

The third requirement forces us to introduce an identifying relationship that is binary.

The best way to module this situation is to use two binary relations.



This ER really has two relationships involving policies, and our attempt to use a single ternary relationship is inappropriate.

Relational Algebra:

Branch_schema = (branch_name, br_city, assets)

Customer_schema = (customer_name, cu_street, cust_city)

Depositer_schema = (cust_name, acc_no)

Loan_schema = (loan_no, branch_name, amount)

borrower_schema = (cust_name, loan_no)

Account_schema = (acc_no, cust_name)

Fundamental operations of Relational Algebra:

1. Select (σ)
2. Project (π)
3. Combination of select & Project
4. Union ($\cup$)
5. Set difference ($-$)
6. Cartesian product ($\times$)
7. Rename (ρ)

Find tuples pertining loans of more than 1200 made by vjw branch.

$\sigma_{branch='vjw' \wedge amount > 1200}(loan)$

Selecti operation select tuples that satifies the given predicate.

Project:

Project operation is an unary operation that returns its argument relation with certain attributes left out.

Syntax: $\Pi_{attributes}(R)$; Relation.

Union:

Customers having an acc or loan in acc.

$\Pi_{custname}(depositer) \cup \Pi_{custname}(borrower)$.

Cartesian Product:

If we have relations $r_1(R_1)$ & $r_2(R_2)$ then $r_1 \times r_2$ is a relation whose schema is concatenation of R_1 & R_2

Realtion 'R' contains all tuples 't' for which there is a tuple t_1 in r_1 & a tuple t_2 in r_2 for which $t[R_1] = t_1[R_1]$ & $t[R_2] = t_2[R_2]$

1 Sailors Database.

sid	sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

2 Reserves Database

sid	bid	day
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98.

3 Boats Database.

bid	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red.

Sailors (sid: integer , sname: string, rating: integer, age: real)

Boats (bid: integer , bname: string, color: string)

Reserves (sid: integer, bid: integer, day: date)

a) Branch_schema:

Branch_schema = (branch_name, branch_city, assets)

branch_name	branch_city	assets
Brighton	Brooklyn	7100000
Downtown	Brooklyn	9000000
Mianus	Horseneck	400000
North Town	Rye	3700000
Perryridge	Horseneck	1700000
Pownal	Bennington	300000
Redwood	Palo Alto	2100000
Round Hill	Horse neck	8000000

b) Customer_schema:

Depositor_schema = (customer_name, account_number)

Customer_name	account_number
Hayes	A-102
Johnson	A-101
Johnson	A-201
Jones	A-217
Lindsay	A-222
Smith	A-215
Turner	A-305

ii) Customer_schema = (customer_name, customer_street, customer_city)

Customer_name	Customer_street	Customer_city
Adams	Spring	Pittsfield
Brooks	Senator	Brooklyn
Curry	North	Rye
Glenn	Sand Hill	Woodside
Hayes	Main	Harrison
Johnson	Alma	Palo Alto
Jones	Main	Harrison
Lindsay	Park	Pittsfield
Smith	North	Rye
Turner	Putnam	Stamford
Williams	Nassau	Princeton

iii) Loan_schema = (loan_number, branch_name, amount).

loan_number	branch_name	amount
L-11	Round Hill	900
L-14	Downtown	1500
L-15	Perry ridge	1500
L-16	Perry ridge	1300
L-17	Downtown	1000
L-23	Redwood	2000
L-93	Mianus	500

iv Borrower_schema = (Customer_name, loan_number)

Customer_name	Loan_number
Adams	L-16
Curry	L-93
Hayes	L-15
Jackson	L-14
Jones	L-19
Smith	L-11
Smith	L-23
Williams	L-17

v Account_schema = (account_number, branch_name, balance)

account_number	branch_name	balance
A-101	Downton	500
A-102	Perryridge	400
A-201	Brighton	900
A-215	Mianus	700
A-217	Brighton	750
A-222	Redwood	700
A-305	Round Hill	350

Find the names of all customer who have loan at Perriridge branch

$$\Pi_{cname}\left(\sigma_{l.lno = b.lno \wedge b.name = "perryridge"}(loan \times borrower)\right)$$

$$\Pi_{cname}\left(\sigma_{bname = "parrybridge"}\left(\sigma_{l.lno = b.lno}(loan \times borrower)\right)\right)$$

$$\Pi_{cname}\left(\sigma_{b.name = "parrybridge"}(loan \bowtie borrower)\right)$$

Additional Algebra Operations:

i. Intersection (iv) Division.

ii. Joint

iii. Assignment

Set Intersection:

Find the names of Customers who have both loan & account

$$\Pi_{customer\text{-}name}(Depositor) \wedge \Pi_{Customer\text{-}name}(Borrower)$$

>> (Select cname from Depositor)
Intersect
(Select cname from borrower)

Natural Join: ($\bowtie$)

Consider two relations $r(R)$ & $s(S)$ natural join of r, s denoted by

(r → instance, R → schema)

"$r \bowtie s$" is a relation on schema $(R \cup S)$

$$r \bowtie s = \Pi_{R \cup S}\left(\sigma_{r.A_1 = s.A_1 \wedge r.A_2 = s.A_2 \ldots \ldots \wedge r.A_n = s.A_n}(r \times s)\right)$$

Q. Names of Customer who have both loan & acc.

$R.A = \Pi_{customer\_name}(borrower \times loan)$.

Q. Find the names of all branches with customers who have an account in the bank & who lives in Harrison.

Soln: customer (cname, c.city (= Harrison), street)

depositer (cname, acc.no)

account (acc.no, branch, bal)

↓ o/p.

$$\Pi_{branch}\left(\left(\left(\sigma_{c.city = Harrison}(Customer)\right) \bowtie depositer\right) \bowtie account\right)$$

Assignment Operation: ($\leftarrow$)

$$temp \leftarrow \sigma_{c.city = Harrison}(Customer)$$

$$Res \leftarrow \Pi_{branch}((temp \bowtie depositor) \bowtie account)$$

Extended Relational Algebra Operations:

i. Generelaised Project
ii. Aggregate functions
iii. Outer joins

Generelaised Projection:

$$\Pi_{F_1, F_2, \ldots F_n}(R)$$

Aggregate functions:

The arregate functions count(), min(), max(), sum(), avg().

Syntax:

$$\mathcal{G}_{E}(R)$$

Q. Sum of all the salaries in emp. table

$$\mathcal{G}_{sum(sal)}(emp)$$

Q. Sum of all the salaries in emp table acc to deptartment.

$$\mathcal{G}_{d.Id.\, \mathcal{G}_{sum(sal)}}(emp)$$ (Groups in SQL)

Queries

Q Find the details of Employees

SQL > Select * from employee

R.A > σ(employee)

Q Find the salary of Employees of employees

SQL > select ename, sal from employee

R.A > $\Pi_{ename, sal}$((employee))

Q Find the details of Employees whose salary is greater than 2000.

SQL > select * from employee where sal > 2000;

R.A > $\sigma_{sal>200}$ (employee) ;

Q Display the name & hiredate of Employee who are salesman

R.A > $\Pi_{ename, hiredate}$ ($\sigma_{job='Salesman'}$ (employee))

SQL > Select ename, hiredate from employee where job = 'salesman';

Q Display ename & deptnames

$\Pi_{ename, deptname}$ (σ (emp x dept))

R.A > $\Pi_{ename, deptname}(\sigma_{(emp \bowtie dept)})$

(or)

$\Pi_{ename, deptname}(\sigma_{e.dno = d.dno (emp \times dept)})$

SQL> Select ename, deptname
from ename e, dept d
where e.dno = d.dno;

Nested Query:

(Find the names) Display dept names & who are managers.

R.A > $\Pi_{deptname}(\sigma_{job='Manager' (emp \bowtie dept)})$

SQL > Select dname
from dept
where dno in (Select dno
from emp
where job = 'Manager')

R.A > $r \leftarrow \sigma_{job='Manager' (emp)}$

$\Pi_{dname}(\sigma_{dno\ in\ r}(dept))$

$\Pi_{deptname}((\sigma_{job='Manager' (emp)}) \bowtie dept)$

Q Find the name of an Employee whose salary is Maximum.

SQL> select ename from emp
where sal = (select max(sal) from emp);

Order to write Query:

i. from
ii. joins
iii. where
iv. Group by
v. having
vi. ~~Select~~ Distinct
vii. Select
viii. Order by.

Q To find CustomerName & branch Name

SQL> ,Select l.br_name, b.cust_name
from loan l, borrower b
wher l.no = b.no;

SQL> select l.br_name, b.customer
from loan l, borrower b
wher b.lno in (select l.no from b.no)

Loan Relation

loan_no	br_name	amt
L-170	Downtown	3000
L-230	Redwood	4000
L-260	Perrirdge	1700

Borrower Relation

Cust_name	loan_no
Jones	L-170
Smith	L-230
Hayer	L-155

Types of Outer joins

i. Left Outer join ($\bowtie$)

ii. Right Outer join ($\bowtie$)

iii. Full Outer join ($\bowtie$).

Ex: i. loan $\bowtie$ borrower

loan_no	br_name	amt	cust_name	loan_no
L-170	Downtown	3000	Jones	
L-230	Redwood	4000	Smith	

ii. loan $\bowtie$ borrow (using left Outer join)

loan_no	br_name	amt	cust_name
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith
L-260	Perriridge	1700	NULL

iii. loan $\bowtie$ borrow (using right Outer join)

loan_no	br_name	amt	cust_name
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith
L-155	NULL	NULL	Hayer.

iv. loan $\bowtie$ borrow (using full outer join)

loan_no	br_name	amt	cust_name
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith
L-260	Perriridge	1700	NULL
L-155	NULL	NULL	Hayer

Note: Outer joins are used to note missing values.

Left Outer Join (⟕):

It takes all tuples in the left relation that did not match the tuple in the right relation and pads the tuple with NULL values for all other attributes from the right relations and adds tem to the result of Natural join.

Right Outer Join (⟖):

It takes all tuples in the right relation that did not match the tuple in the left relation & pads with NULL values for all the attributes form the left relations and adds them to result of Natural join.

Outer Full Join (⟗):

It takes all tuples in the right & left relations and pads those tuples with NULL values for all other attributes missing and adds them to the result of Natural join.

Modification of Database:

Insertion : $r \leftarrow r \cup E$

Insert that the fact in the account table.

Ex: account ← account ∪ {'A-101', 'Dowton', 500}

Deletion: $r \leftarrow r - E$

Ex: account ← account − {'A-101', 'Downton', 500}

Updation: $r \leftarrow \Pi_{F_1, F_2, \ldots F_n}(r)$

Ex: account ← $\Pi_{accno, branchname, bl * 0.3}$ (Account)

Find the names & rating of sailors with a rating more than 8.

```
SQL> select name, rating
     from sailors
     where rating>8;
```

R.A > $\Pi_{name, rating}(\sigma_{rating>8}(Sailors))$

TRC > $\{t \mid \exists S \in sailors \wedge S.rating > 8 \wedge t.Sname = S.Sname \wedge t.rating = S.rating\}$;

Find the names of sailors who have reserved boat 103.

```
SQL> select Sname from sailors S
     where S.sid in ( select R.sid
                      from Reserves R
                      where R.Bid=103)
```

RA > $\Pi_{sname}\left(\left(\sigma_{bid=103}^{Reserves}\right) \bowtie sailors\right)$;

TRC > $\{t \mid \exists s \in sailors, \exists r \in Reserves$
$(r.bid = 103 \wedge r.sid = s.sid \wedge t.sname = s.sname)\}$

DRC > $\{\langle N \rangle \mid \exists \langle I, T, A \rangle (\langle I, N, T, A \rangle \in sailors \wedge$
$\exists \langle I_r, B_r, D \rangle \in Reserve \wedge I = I_r \wedge B_r = 103)\}$

Find names of sailors who reserved red boat-

SQL > select s.sname from sailors s
where R.bid in (select b.bid
from boat b
where b.colour = 'red')

(or)

select s.sname from sailors s, boat b, Reserves R
where s.sid = R.sid and R.bid = b.bid and
b.colour = red;

RA > $\Pi_{sname}\left(\left(\sigma_{b.color='red'}^{boat}\right) \bowtie Reserves \bowtie sailors\right)$

TRC > $\{t \mid \exists s \in sailors, \exists r \in Reserves, \exists b \in boat$
$(s.sid = sid \wedge bid = b.bid \wedge b.Colour = 'red' \wedge t.sname = s.sname)\}$

DRC > $\{\langle N \rangle \mid \exists \langle I, T, A \rangle (\langle I, N, T, A \rangle \in sailors \wedge$
$\exists \langle I_r, B_r, D \rangle \in Reserves \wedge \exists \langle Clr, B_B, B \rangle$
$\in Boats \wedge clr = 'red' \wedge I = I_r \wedge B_r = B_B)\}$

Find the colours of boats reserved by luber.

SQL> Select b.colours from boats b
where b.bid in (select R.bid
from reserves R
where R.sid = (select s.sid
from sailors s
where s.sname
= 'luber'))

(or)

Select b.colour from sailors s, reserves r, boats b
where s.sname = 'luber' and s.sid = R.sid and
r.Bid = b.bid.

R.A > $\pi_{colour}((\sigma_{sname='luber'}(sailors) \bowtie reserves \bowtie boats))$

TRC> $\{t / \exists s \in sailors, \exists r \in reserves, \exists b \in boats, (s.sid = r.sid \wedge r.bid = b.bid \wedge s.sname = 'luber' \wedge t.colour = b.colour)\}$.

DRC> $\{C / \exists \langle B_B, B \rangle (\langle C, B_B, B \rangle \in Boats \wedge \exists \langle I_r, B_r, D \rangle \in Reserves \wedge \exists \langle I, T, A, N \rangle \in Sailors \wedge I = I_r \wedge B_B = B_r \wedge N = 'luber')\}$

Find names of sailors who have reserved atleast one boat.

SQL>
```
select distinct(sname)
from sailors s, reserves r
where r.sid = s.sid;
```
(or)
```
select distinct(sname)
from sailors s
where s.sid in (select r.sid
                from reserves r);
```

R.A > Π_{sname} (sailors ⋈ reserves);

TRC > $\{ t \mid \exists s \in sailors, \exists r \in Reserves;$
$(r.sid = s.sid \wedge t.sname = s.sname) \}$;

DRC > $\{ N \mid \exists \langle I, T, A \rangle (\langle N, I, T, A \rangle \in Sailors \wedge$
$\exists \langle Ir, Br, D \rangle \in Reserves \wedge I = Ir) \}$

for all p=>q

Find the names of sailors who have reserved all boats

R.A > $\Pi_{sname}\left(\left(\Pi^{(Reserves)}_{sid,bid} \right) / \left(\Pi_{bid}(Boats) \right) \right)$

TRC > $\{ t \mid \exists s \in sailors, \forall B \in Boats, \exists R \in Reserves$
$(R.bid = B.bid \wedge R.sid = s.sid \wedge t.sname = s.sname) \}$

D.R.C: $\{\langle N\rangle / \exists \langle I,T,A\rangle (\langle I,N,T,A\rangle \in Sailors \wedge \forall \langle B_B, B_N, C\rangle \in Boats \Rightarrow \exists \langle I_r, B_r, D\rangle \in Reserves (I = I_r \wedge B_r = B_B))\}$

Find the names of sailors who have reserved red or green boat.

SQL> select s.sname
from boats b, reserves r, sailors s
where b.bid = r.bid and r.sid = s.sid and
b.color in ('Red', 'Green').

R.A>

$$P(temp, (\sigma^{Boats}_{clour='Red'}) \cup (\sigma^{Boats}_{colour='Green'}))$$

$$\Pi_{Sname}(temp \bowtie reserves \bowtie Sailors).$$

TRC> $\{t / \exists s \in sailors, \exists r \in reserves, \exists b \in boats$
$(b.bid = r.bid \wedge r.sid = s.sid \wedge (b.color = 'Red' \vee b.color = 'Green') \wedge t.sname = s.sname)$.

Find the names of sailors who have reserved both red and green boats

SQL:

```
select s.sname
from boats b, Reserves r, Sailors s
where b.bid = r.bid and r.sid = s.sid and
      b.color = 'Red'

      Intersect

slect s.sname
from boats b, reserves r, Sailors s
where b.bid = r.bid and r.sid = s.sid and
      b.color = 'Green'.
```

RA: $\Pi_{sname}\left(\left(\left(\Pi_{sid}\left(\sigma_{color='red'}(Boats)\right) \bowtie reserves\right) \cap \left(\Pi_{sid}\left(\sigma_{color='Green'}(Boats)\right) \bowtie reserves\right)\right) \bowtie Sailors\right)$

TRC:

Find the sids of sailors with age over 20 who have not reserved a red boat.

RA: $\left(\Pi_{sid}\left(\sigma_{age>20}(Sailors)\right) - \left(\Pi_{sid}\left(\sigma_{color='Red'}(Boats)\right) \bowtie reserves\right)\right)$

Find the names of sailors who have reserved all red boats.

R.A :

$$\rho(temp1, (\Pi_{sid,bid}(Reserves)) / (\Pi_{sid}(\sigma_{color='red'}(Boats))))$$

$$\Pi_{sname}(temp1 \bowtie sailors)$$

T.R.C :

$$\{t \mid \exists s \in sailors, \forall B \in boats, (B.color='red' \Rightarrow (\exists R \in Reserves \wedge B.bid = R.bid \wedge R.sid = S.sid \wedge t.sname = S.sname))\}$$

DRC

Find the names of sailors with rating > 7

D.R.C :

$$\{\langle N \rangle \mid \exists \langle I, T, A \rangle (\langle I, T, N, A \rangle \in sailors \wedge T > 7)\}$$

Definitions:

TRC: A TRC expression is of the form

$$\{t \mid p(t)\}$$

where p is a formula & t is a tuple variable a TRC formula is build up of atoms

An atom is of the form

① $t \in r$ where t is tuple variable, r is relation

② $t[x] \sim s[y]$ t & s are tuple variables. x, y are attributes. $\sim$ operator

③ $t[x] \textcircled{\sim} c$ C is constant

DRC : $\{ \langle x_1, x_2, x_3 \cdots x_n \rangle \mid p(\langle x_1, x_2 \cdots x_n \rangle) \}$

p is a formula & $x_1, x_2, x_3 \cdots x_n$ are domain variables

1. $\langle x_1, x_2, x_3 \cdots x_n \rangle \in r$ r is relation
2. $x \textcircled{\sim} y$
3. $x \textcircled{\sim} c$ C is constant

Safety Of Expressions:

A TRC expression may generate infinite number of tuples that are not in the relation

$\{t \mid \sim (t \in \text{sailors})\}$

we donot allow this type of expression using free variables i.e., it is not a safe expression

To avoid this use Quantifies $\in$ (or) $\forall$ to find the variables

Expressive Power of Languages:

A TRC, DRC restricted to safe express is equivalent in expressive power to the basic relational algebra.

TRC, DRC does not have any equivalent of aggregate operations