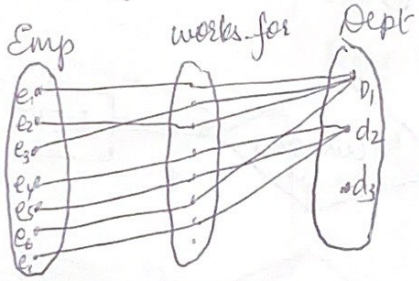
Relationships and constraints:

Structural constraints
- ↳ Cardinality ratio → Max (1:1, 1:N, M:N).
- ↳ Participation const/ Existential constraint → Min

Relationships.
↳ association between entities:

Eg 1: (1:M).

Every employee works for a department, & a dep can have many employees. A new department can have need not have any employees.



Deg=2.

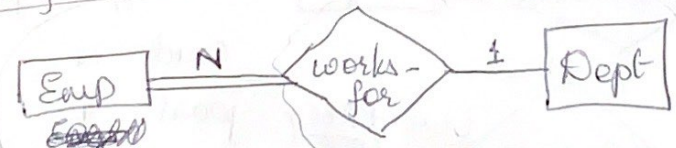
Cardinality Ratio: what is the maximum # of relationships that an instance of an entity can participate in.

(Max): Emp 1 — Dept N

Participation/Existance constraint: what is the min. # of relationships that an instance of an entity can partipate.

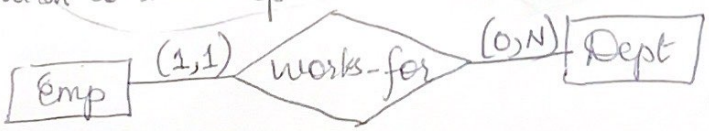
(Min): 1 — 0

cardinality constraint rep:



cardinality ratio/Single line; double line

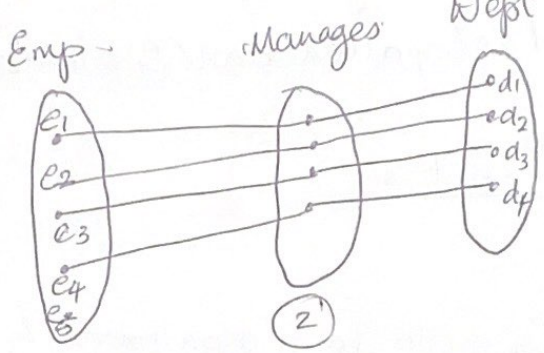
participation constraint rep.



(Min, max) notation

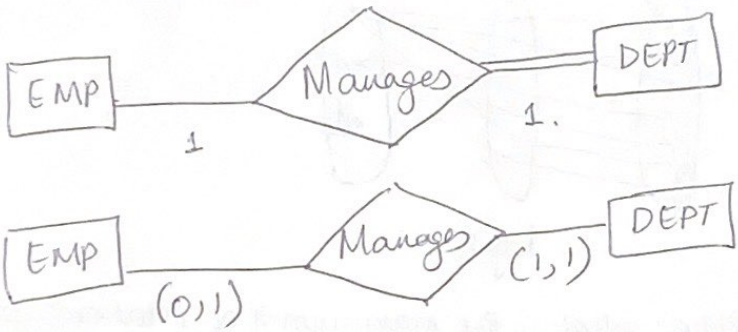
Relationships 1:1

RA: Every Department should have a manager and only one employee manages a dept. & [an emp can manage only one dept]



Every dept Card: (Max): 1
part (Min): 0

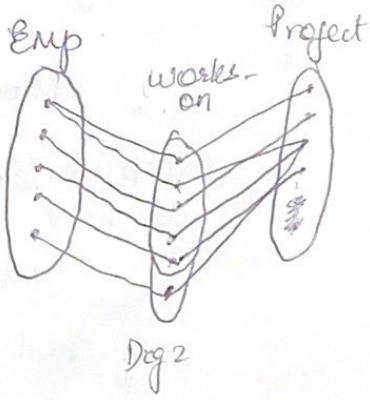
1.
1

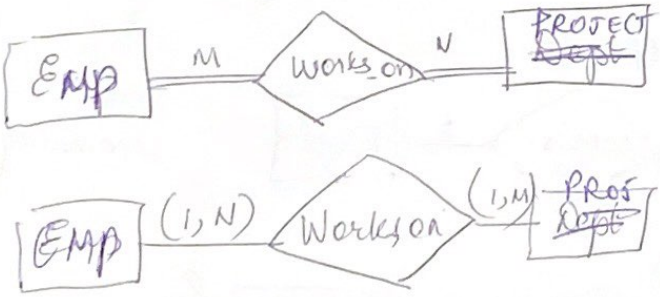


M:N relationship:

Every emp has to work on ≥ projects; and every project should have at-least on emp working on it.

at least one
(one/more)

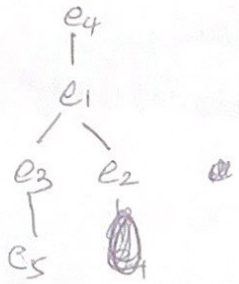


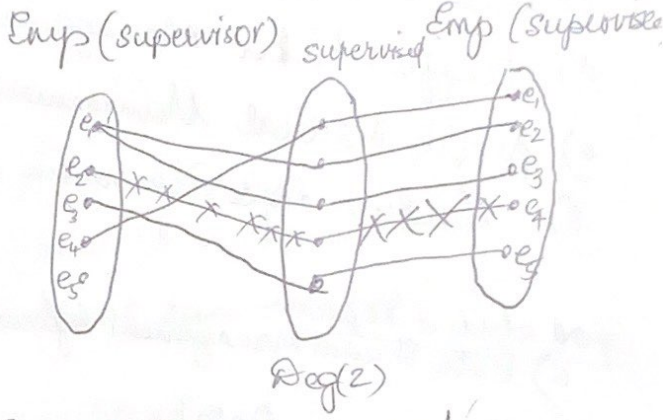


Card : N M
part : 1 1

Recursive Relationship: Every employee reports to a supervisor who is an employee.

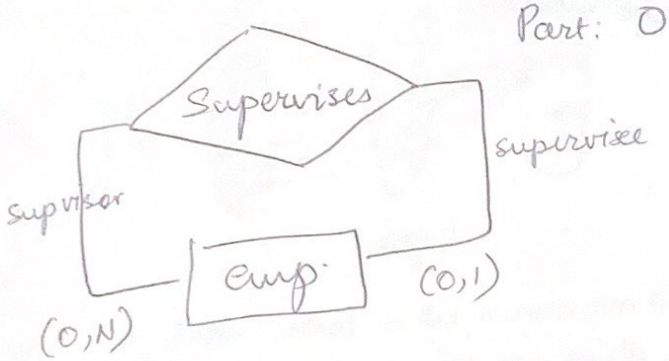
(Every course has pre-req's; that are courses themselves).





Car: N 1

Part: O O



Eg: book:

ACC_NO	YR_PUB	TITLE
734216	1982	Algorithm design
237233	1995	Database systems
631529	1992	Compiler Design

→ each row of the relation is referred to as a 'tuple' in the relation

Relational Algebra:

A formal Query Language based on a set of operations on relations:-

Fundamental operations: {base on set theory}

* select
* PROJECT
* CARTESIAN PRODUCT
* UNION (set)
* SET DIFFERENCE
* RENAME

Additional operations: {can be defined using fundamental operations}

* Natural Join (widely used)
* Intersection
* Assignment
* Division / Quotient
* θ-Join. (generalization of natural join)

Note: none of these operations modify the values in the database, they only create newer relations

The SELECT operation (σ):

* The purpose of 'σ' is to find a subset of tuples from a relation; and provide a mechanism to extract only these tuples out of the relation.

* it is a unary operation as it works only on 1 relation at a time.

Eg1: $\sigma_{YR\_PUB = 1992}(book)$

(expression/condition)

OP:

ACC_NO	YR_PUB	TITLE
631529	1992	compilers design

Eg2: $\sigma_{ACC\_NO \geq 56782}(book)$

Eg3: $\sigma_{(ACC\_NO \geq 56782) \wedge (YR\_PUB = 1992)}(book)$

The PROJECT operation (π):

* It is a unary operation

* However; the purpose of 'π' is to select a subset of attributes from a relation

Eg1: $\pi_{ACC\_NO, TITLE}(BOOK)$

NOTE: if there is any duplicate entries created by 'π' they will be removed.

Eg 2: $\pi_{TITLE}(\sigma_{YR\text{-}PUB > 1992}(Book))$

TITLE

All attributes of Book

nested query.

Query: list the titles of those books whose YR_of Pub > 1992.

Eg 3: $\sigma_{YR\text{-}PUB = 1991}(\pi_{YR\_PUB, TITLE}(Book))$

YR_PUB

Query:

The cartesian Product operation (×)

Eg:

A
1
2
3

B
a
b

A × B

A	B
1	a
1	b
2	a
2	b
3	a
3	b

Eg 2:

r

A	B
a	1
b	2
a	2

s

B	C
3	1 a
2	2 b

$r \times s \neq dom(A) \times dom(B) \times dom(C)$

r × s

r.A	r.B	s.B	s.C
a	1	3	1 a
a	1	2	2 b
b	2	3	1 a
b	2	2	2 b
a	2	3	1 a
a	2	2	2 b

* Cartesian product is used to combine 2 relations.

Eg: BOOK (ACC_NO , YR_PUB , TITLE)

USER (CARD_NO , B_NAME, B_ADDRESS)

BORROW (ACC_NO. , CARD_NO ; DOI)

Query: Find the acc_No of all books issued to "TIM".

USER x BORROW

attributes:

(USER . CARD_NO, USER. B_NAME, USER - B_ADDRESS
BORROW. ACC_NO, BORROW. CARD_NO, BORROW. DOI)

$$\Pi_{\text{BORROW.ACC\_NO}} \left(\sigma_{\substack{(\text{USER.CARD\_NO} = \text{BORROW.CARD\_NO}) \\ \wedge \\ (\text{USER. B\_NAME} = \text{"TIM"})}} (\text{USER} \times \text{BORROW}) \right)$$

$$\Pi_{\text{ACC\_NO}} \left(\sigma_{\substack{(\text{USER.CARD\_NO} = \text{BORROW.CARD\_NO}) \\ \wedge \\ (\text{B\_NAME} = \text{"TIM"})}} (\text{USER} \times \text{BORROW}) \right)$$

3 The RENAME operation (ρ)

Query: Find the names of all users who have the same address as "TIM"

USER x USER $\rightsquigarrow$ (USER x ρ_{USER_1} (USER))

problem → how to differentiate bet attributes

$$\Pi_{USER1.B\_NAME}\left(\sigma_{(USER.B\_NAME='TIM') \wedge (USER1.B\_ADDR = USER.B\_ADDR)}\left(USER \times \rho_{USER1}(USER)\right)\right)$$

ε SET difference operation (−): → result: those tuples in r that are not there in S
* Binary operator r − s.

* Find the ACC_NO of all books which are available in the library.

$$\Pi_{ACC\_NO}(BOOK) - \Pi_{ACC\_NO}(BORROW)$$

* Find the titles of all books which have not yet been issued.

ε Set union operation (∪): → result: all tuples in both r and s that are unique and should have common attribute
* Binary Operator r ∪ s

* Find out all books which are either issued or have been supplied by a supplier.

$$\Pi_{ACC\_NO}(SUPP) \cup \Pi_{ACC\_NO}(BORROW)$$

→ Degree of relations have to be same.
→ Domains of ith attributes of R_1 & R_2 have to be same
$r_1 \cup r_2 = t \mid t \in r_1$ or $t \in r_2$