Extended Entity Relation Diagram (EER)

If we have superclass - subclass type of relation b/w entity types.

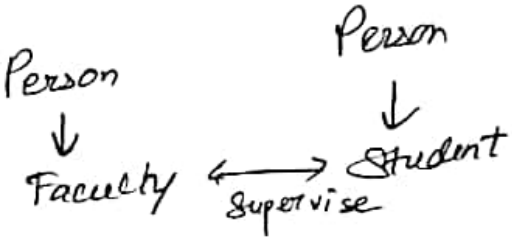
→ not in ERD.

A subset of an entity-set of a particular entity type may have certain additional attributes

OR

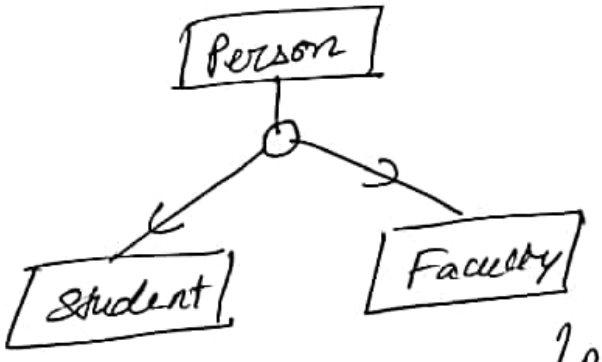
A subset may take part in certain relation.

Example:



To incorporate above two things, we need the concept of inheritance

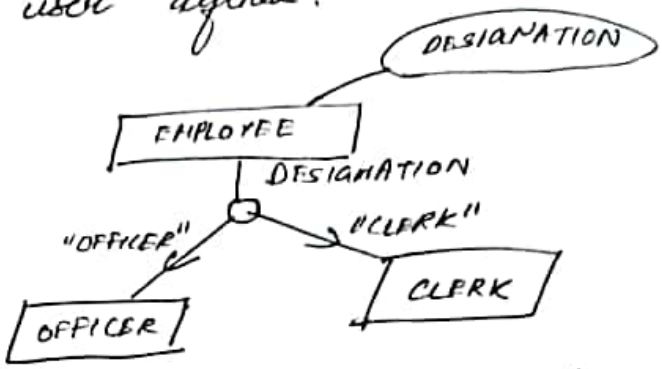
How is it shown?



Allows → generalisation (Bottom-Up)
specialisation. (Top-Down)

Constraints on generalisation - specialisation

→ Relation may predicate defined or user defined.

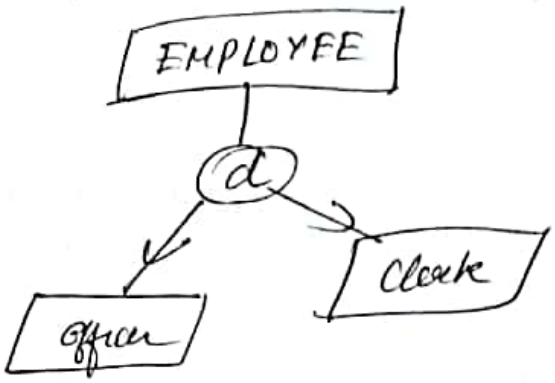


A superclass entity, will belong to a subclass based on the predicate (well-defined)

When Attribute defining the predicate. → called attribute defined generalisation - specialisation

User-defined: group of subclass defined by the user during input, Eg: If in above example there was no designation attribute & "officer" / "clerk" was defined during input.

→ disjointness constraint.



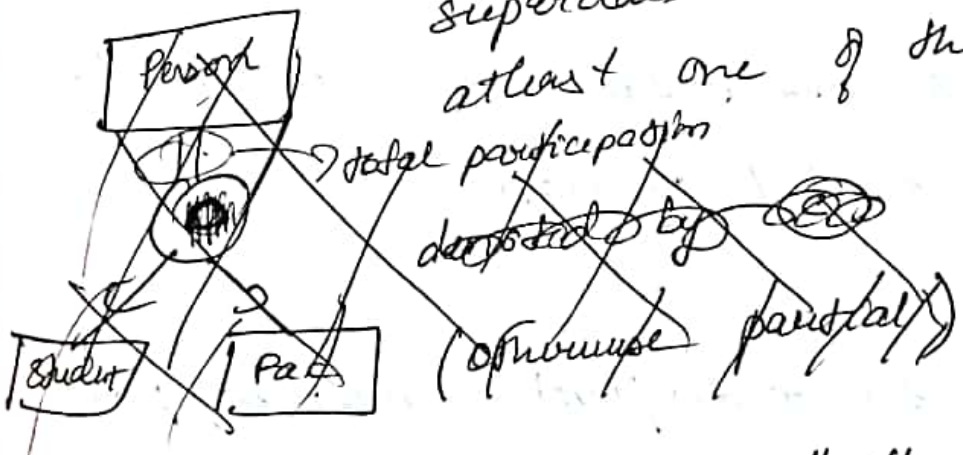
Superclass element can belong to atmost one subclass.

→ completeness constraint.

c

In above, an instance of superclass instance can be part of no subclass

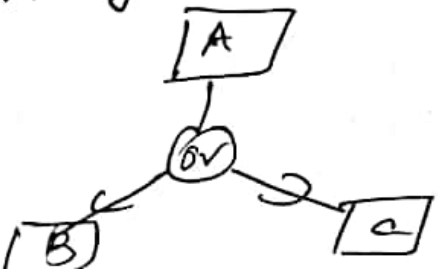
This requires → an instance of superclass to be part of atleast one of the subclasses.



= (total nes)

(overlapping denoted by "ov"

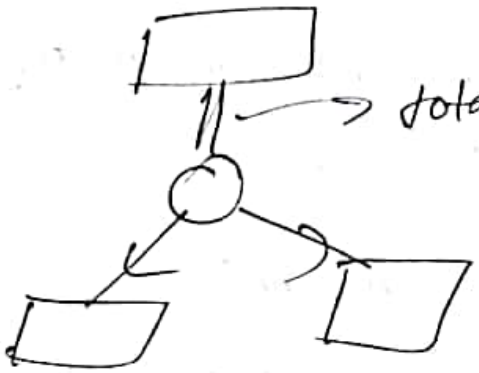
entity may belong to both b & c)



Possibilities :

{disjoint, overlap} × {total, partial}

completeness == total / partial



| = partial.

↳ i.e superclass entity may not belong to any subclass.

total == completeness (i.e superclass entity must belong to atleast one subclass)

along with total → it can be disjoint / overlap.

↳ total + disjoint → must be part of only one subclass.

total + overlap → must be part of atleast one subclass.

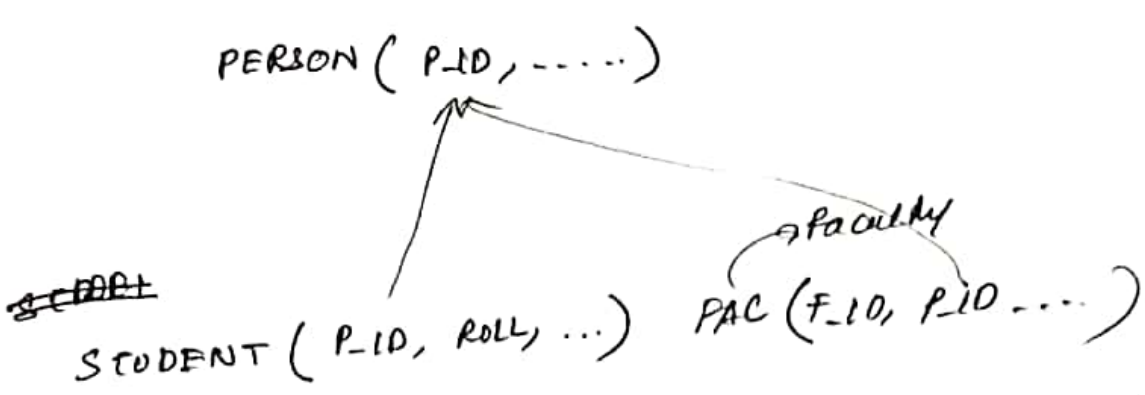
// Superclass - subclass relation ⇒ only one superclass → superclass hierarchy.
(Specialisation hierarchy)

// Specialisation lattice
A subclass is having more than one superclass.

Example: TA is both student & faculty.
(issue happens when this entity participates in a relation)

Solution: (Design 1)
- for superclass, consider a relation:
for each subclass, a separate schema*
copy the pk of the superclass.
& it will be the foreign key in subclass. as well as the primary key. in the subclass.

* with additional attributes is made.



Thus may be {total, partial} × {disjoint, overlap}

No issue for any case of above.

Whether total/partial & disjoint/overlap we can follow design mentioned in last page. → No issue.

But, Equijoin may have to be carried out, to get complete info. of a subclass *. (∵ Additional attributes only in subclass).

Superclass schema → gives common info all subclasses.
(No outer union operation required)

* (subclass equijoin superclass needed)

Design 2

(multiple relations formed)

For each subclass, separate relation schema, one for superclass.

attribute of superclass ∪ its own attributes.

If overlapped → Common info is repeated twice.

(instance part of two subclasses) → redundancy

No issue if disjoint.

Partial → can't store the instance since, the superclass entity inst. does not belong to any subclass.

→ This design fails.

∴ Good soln for total & disjoint.

No expensive equijoin needed to know all details of subclass.

But union, outer union needed to get all common superclass level info.

Design 3

① Single Schema (four designs used multiple schema)

↓

Superclass attributes

∪

Attr of all subclasses

& one type attribute

{

to denote which subclass.

no multiple relations needed.

(Since relational model does not allow multivalued attr, type can have only one value → must be disjoint)

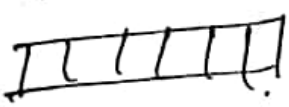
→ Lots of null values.

→ No equijoin / outer union needed.

→ To allow non-disjoint / overlapping generalization / specialisation relation,

For R subclasses,

Keep a R bit bit mask.



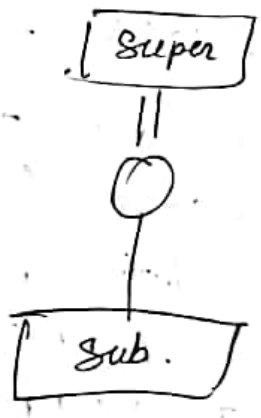
K bits.

If ith bit set, instance belongs to that ith subclass.

If ith & jth bit set, it belongs to both ith & jth subclass.

// We can also keep 2 single bit fields if the database allows.

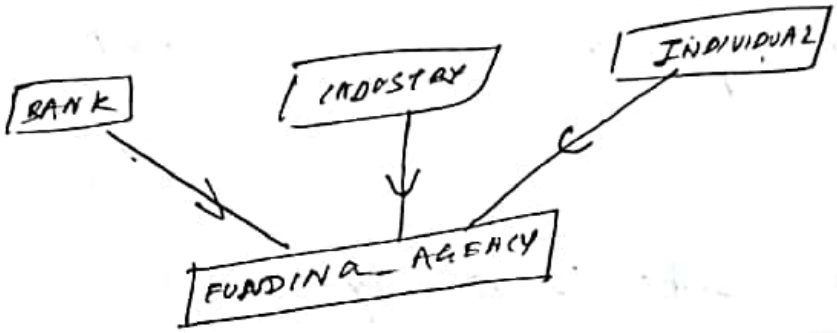
Earlier we did.



All superclasses will take part in some subclass.

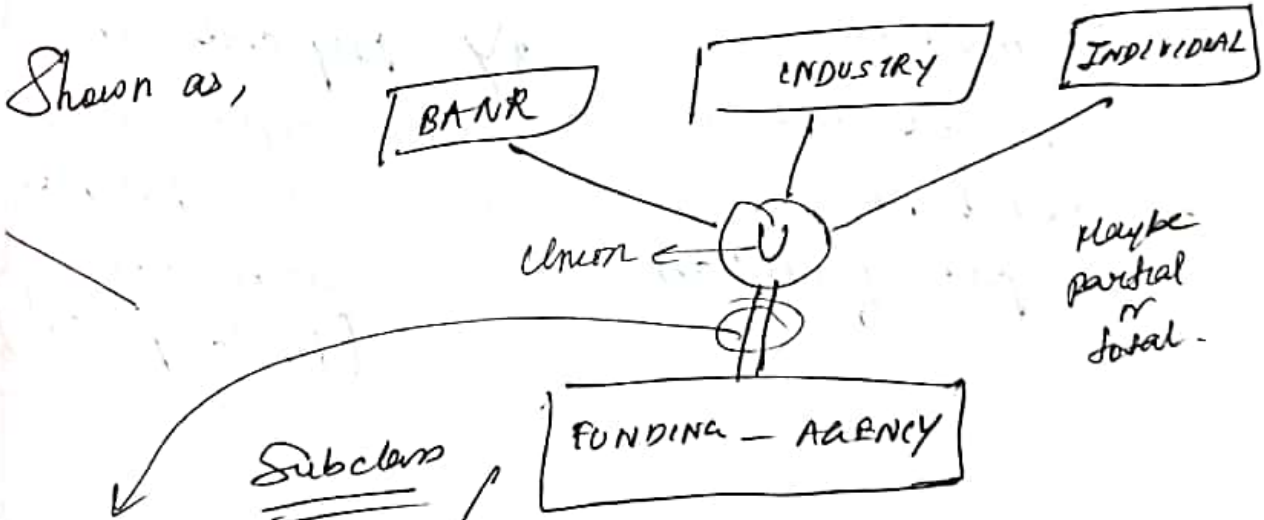
// An subclass that may have multiple superclass of different types and subclass entity set is a subset of union of those superclass entity sets.

Eg: FUNDING_AGENCY



(Generally, when subclass has multiple superclass, subclass entity set = intersection of superclass entity sets)

Shown as,



Subclass

→ Union / Category of no. of superclasses of diff. types.

To show total participation

check next page.

~~Difference than gender~~

The Lower II means funding agency is one of bank / industry / individual

Difference

Generalization — Specialisation

1) only 1 superclass

2) Lattice subclass & entity set is intersection of superclass entity sets.

3) In multiple inheritance, all attributes are part of subclass

Union.

1) No. of superclasses of different type

2) is union of subsets of superclass entity sets.

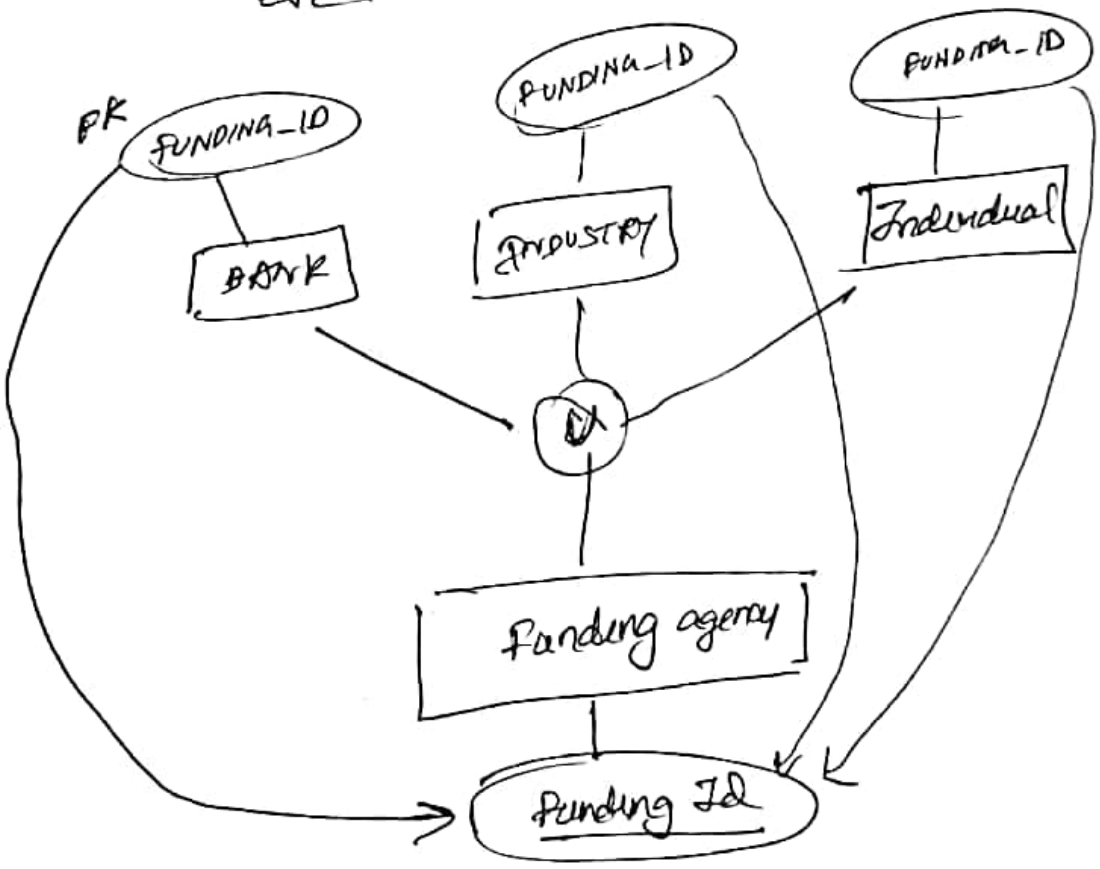
3) Depends on the superclasses giving birth to Subclass (Philosophical)

// In above example,

bank, industry, individual are all different types.

Here, we may have no common key attributes.

we introduce surrogate key funding id & add it as foreign key to bank, industry etc



No. of customers are there.

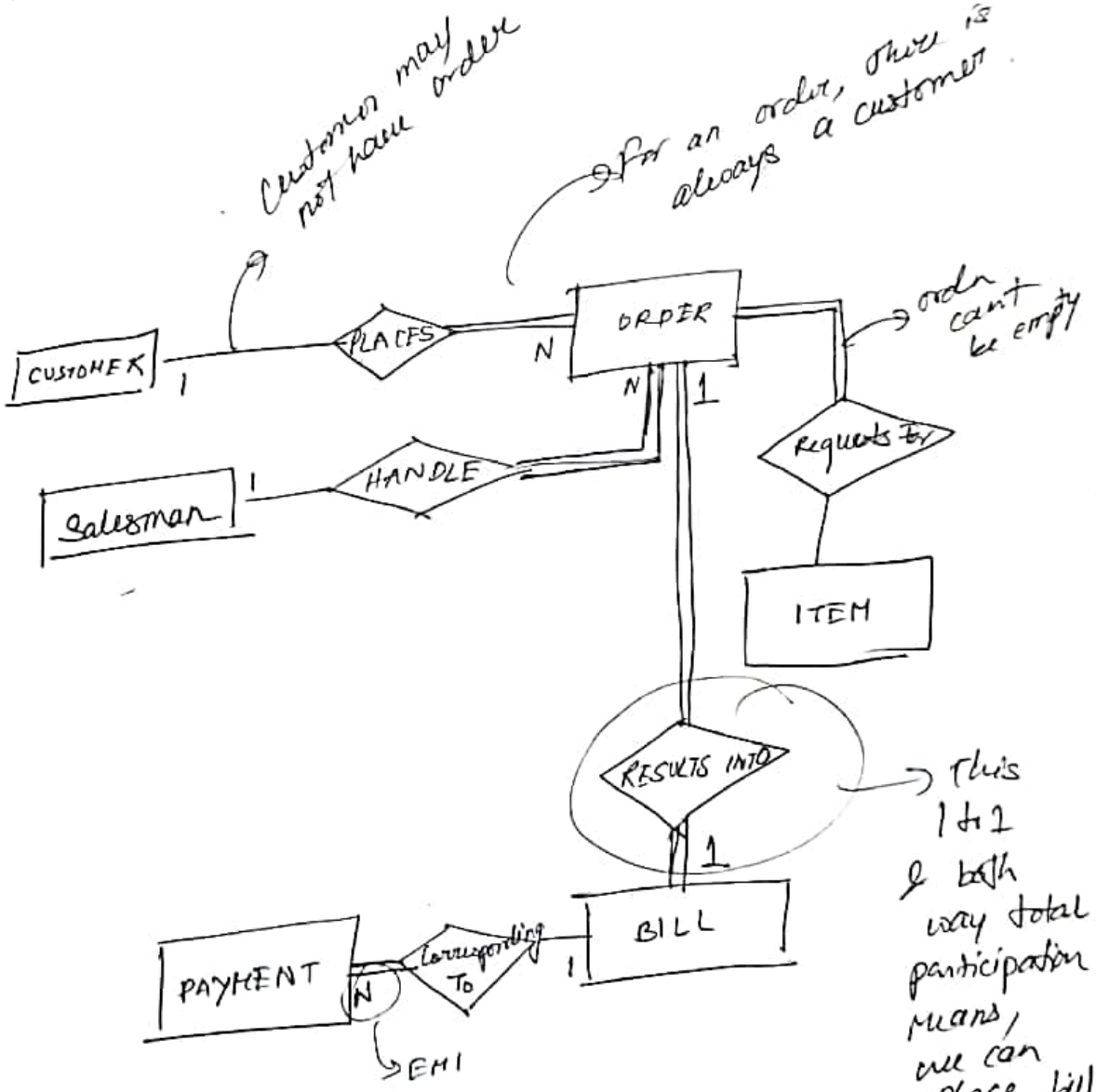
organisation deals with various items.

Customer places the order.

Orders may have request for various items.

Bills & Payment info for the orders to be maintained.

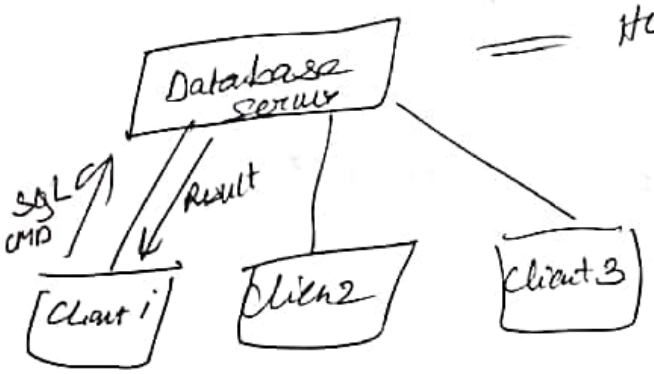
One Salesman handles one or more orders.



SQL

- DDL (Specify the schema)
- DML

SQL PLUS → Environment where one can write / execute / edit the sql commands.



= HOSTNAME (where server is)

= ~~SQLPLUS~~.

User ID/ Pwd for authenticating client SQL PLUS on client side.

= We can write command on multiple lines, but we can only edit current line & not previous line.

```
create table dept
(c1 ,- c2 . . . . )
colname type size column level constraint
 "    "   "
```

Types: Char (10)

Varchar (5) → initially len 5, extendable.

Number (5,2) → 

↳ 5 digits, of which 2 are after decimal

Last command stays in buffer

SQL> / Enter, to execute last cmd from buffer.

// : End of command & execute

/ press enter At new line, to signify, end of Command.

// Oracle

System tablespace is created, that holds data dictionary → ≡Table

/ We can name constraints. Why?

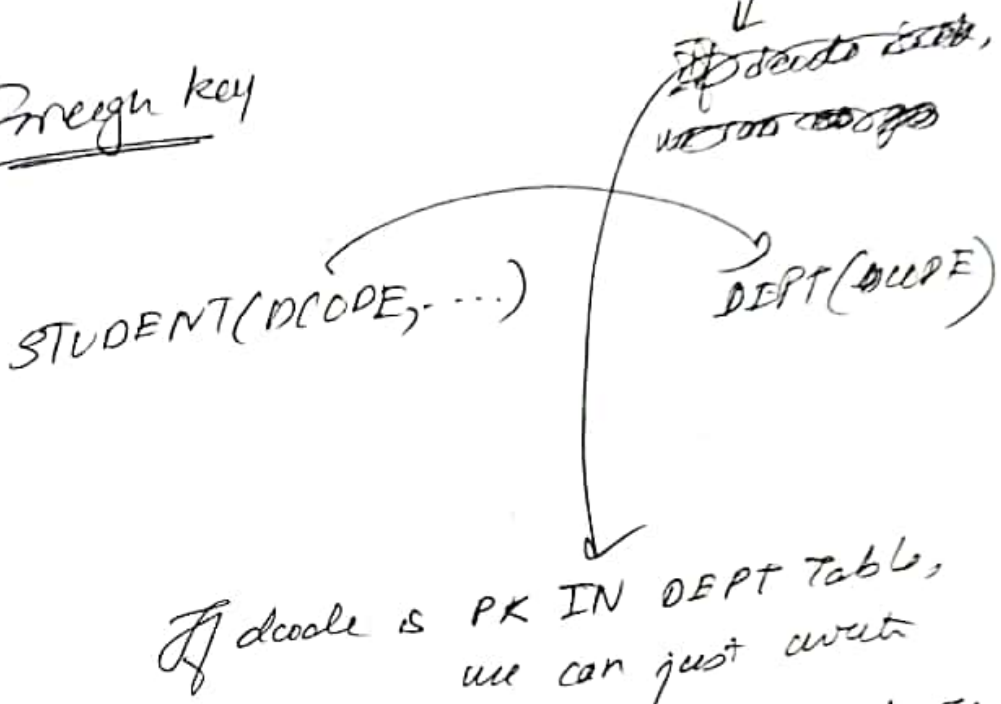
Student

// Create Table (. . .

. . . .

DCODE char(5) References DEPT(DCODE));

Foreign key



If dcode is PK IN DEPT Table, we can just create DEPT here.

Scenario: Exams Tables: Student & Subject codes.

// Combined PK. (PK not applied on single column, so not column level)

Create Table Attendance (ROLL, SCODE).

(ROLL CHAR(5) REFERENCES STUDENT (ROLL),

SCODE CHAR(5) REFERENCES SUBJECT (SCODE),

PRIMARY KEY(ROLL, SCODE));

↳ Table level constraint.

// SCORESHEET (ROLL, SCODE, SCORE)
↳ PK

Now (ROLL, SCODE) must exist in attendance table.
How to specify this?

```
(creat table scoresheet
( ROLL  CHAR(5),
  SCODE  CHAR(5),
  SCORE  NUMBER(3,0),
  PRIMARY KEY (ROLL, SCORE),
  FOREIGN KEY (ROLL, SCORE) REFERENCES
                            ATTENDANCE)
```

Two diff "references" → two column level foreign keys

→ One table level foreign key (combining two columns, since we require the pair to exist & not any of them, in the attendance referenced table)

// Describe tablename.

// Insert into student
values (v1, v2, v3 .....)

Matched with ~~columns~~ columns in order as specified in DDL table creation statement.

// To specify date:

TO_DATE (date string, format)

format: DD-MON-YYYY
- DD: 2 digit
- MON: 3 char month name
- YYYY: 4 digit year.

// Insert into student (c1, c3)
values (v1, v3)

Note: c2 in this case must allow null / default

// Transactions.

(EXIT; used to save changes).

(DML)

//

(DDL always saved independently)