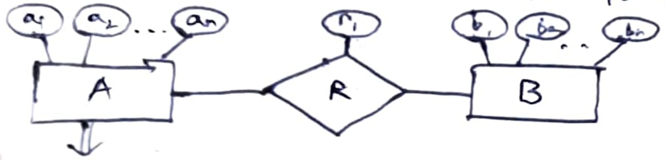
Mapping From ERD to Relations of Relational Model

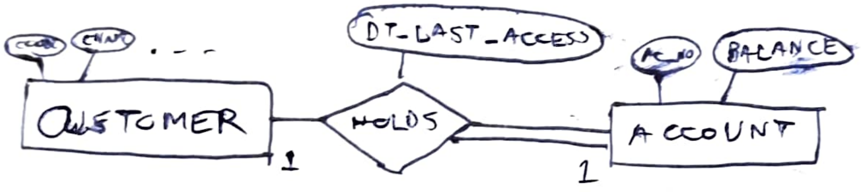
21/1/26

- Mapping of Relations in ER diagram

structural constraint < Mapping Cardinality / Participation Constrt.



A(a1, a2 ... an)



CUSTOMER(CCODE, CNAME)
ACCOUNT(AC_NO,

either keep the PK of ACCOUNT as FK in CUSTOMER
or
keep PK of CUSTOMER as FK in ACCOUNT
or
both

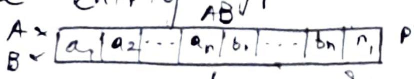
- ONE-TO-ONE

→ FK based approach

Take action in the totally participating side | Copy the ~~pt~~ PK of related entity type as FK in one of the 2 entity types. Also put the attr. of reln in the entity type with FK

If the entity type is not totally participating
⇒ NULL in FK

If partially participating ⇒ null value to be allowed
→ Merged relation : Combine both the entity types in single reln.
Partial participation ⇒ NULL



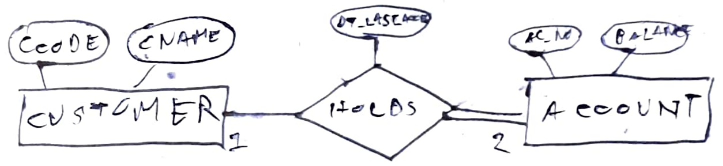
PK : PK of A or B

For customer with no account account no & corr. attr. in the ~~tuple~~ tuple will be null

~~PK~~ # Merged reln useful when both relns are totally participating, otherwise would have to keep null values in the tuples for partially participating reln. When both relns totally part. → merged reln. preferred over equijoin

→ Cross-referencing reln or relationship reln.
For the reln. of ERD, create a new reln.
⇒ Attr ⇒ PK of both the participating entity types & its own attr.
R(a1, b1, r1)
PK of entity types will be FK here
PK of R ⇒ PK of either of the PK of participating entity types
Cross-ref preferred if one of the relns is partly. participating → the entities which are not related will remain in corr. relns & not be in ER ~~the~~ reln.
But if both relns totally part. → merged reln has advantages

• ONE TO MANY



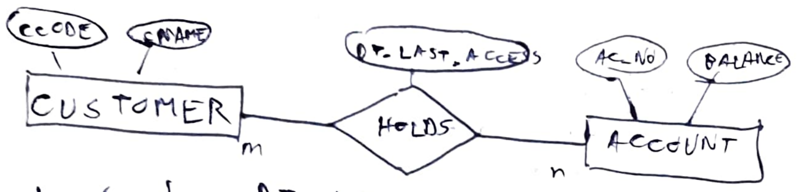
Merged reln. not possible

→ FK-Based approach
In the many side keep the PK of the other type as FK & also put the attr. of reln.

→ Relationship Reln.
PK will be PK of many side (ri PK of one side will repeat multiple lines)

• MANY-TO-MANY

one customer can have many acc & one acc can have joint holders



if desired DT_LAST_ACCESS for CUST irr. of ACC_NO → PK of CUST

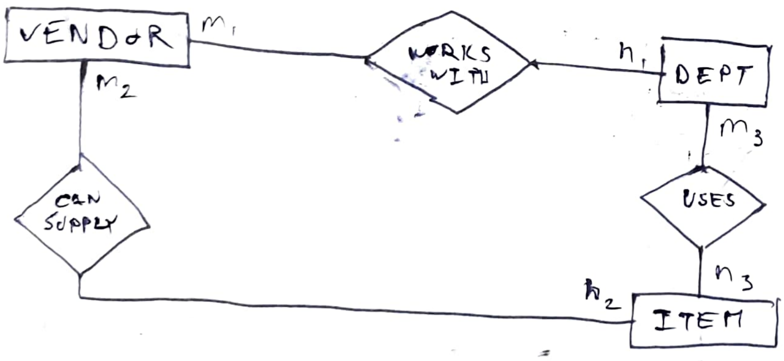
" " ACC_NO → PK of ACC

" " for specific CUST & specific ACC_NO → PK of reln. (comb. of PK of CUST & ACC)

→ Relationship Reln.

HOLDS(CCODE, ACC_NO, LAST_DT_ACCESS)

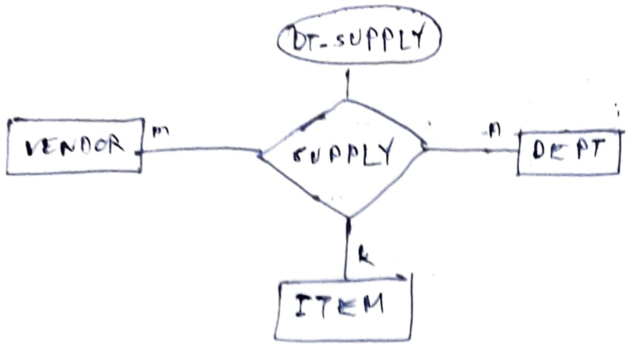
CCODE → FK, ACC_NO → FK, (CCODE, ACC_NO) → PK



if we want those vendors who have supplied a specific item to a specific dept → multiple vendors may exist

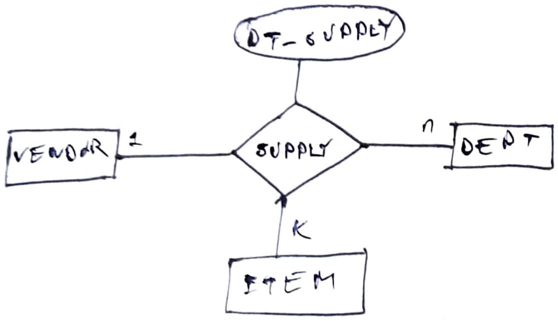
$i \Rightarrow S_1\{V\}$

$d \Rightarrow S_2\{V\}$ → $n(\{S_1 \cap S_2\}) > 1$

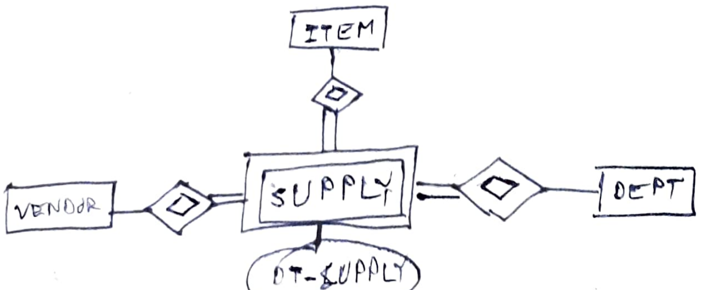


⇒ Ternary reln.
~~SQ~~ SUPPLY (VENDOR ID, DEPT_ID, ITEM_ID, DT-SUPPLY)
P.K. ⇒ UPK of participating entity types

whether to go for higher-deg reln or set of bin. reln. depends on req.
→ if req. is ~~event~~ supply-event-based → go for higher-deg
→ " " " who can sell what to where → set of bin. reln.

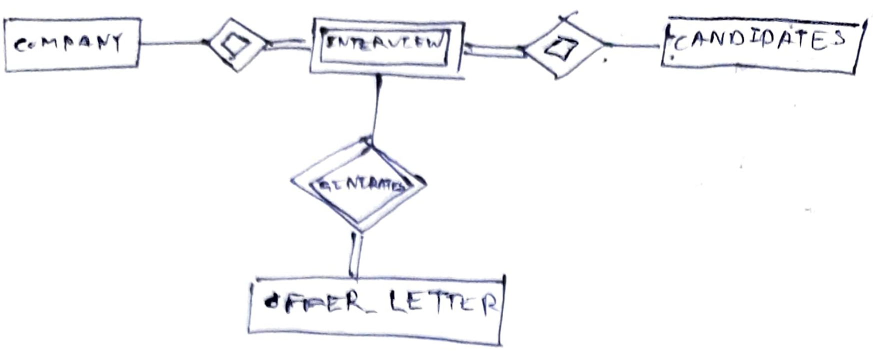


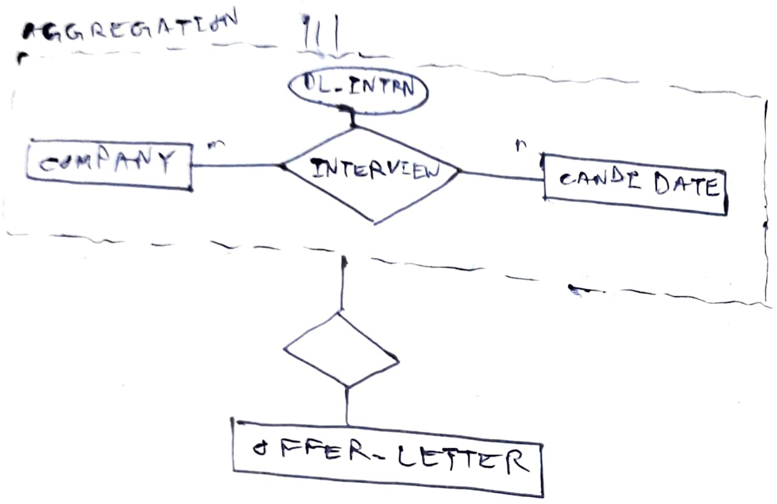
If constraint present that only bin reln. allowed
→ Replace ternary reln. by another entity type
→ make the participating entity types as its owner



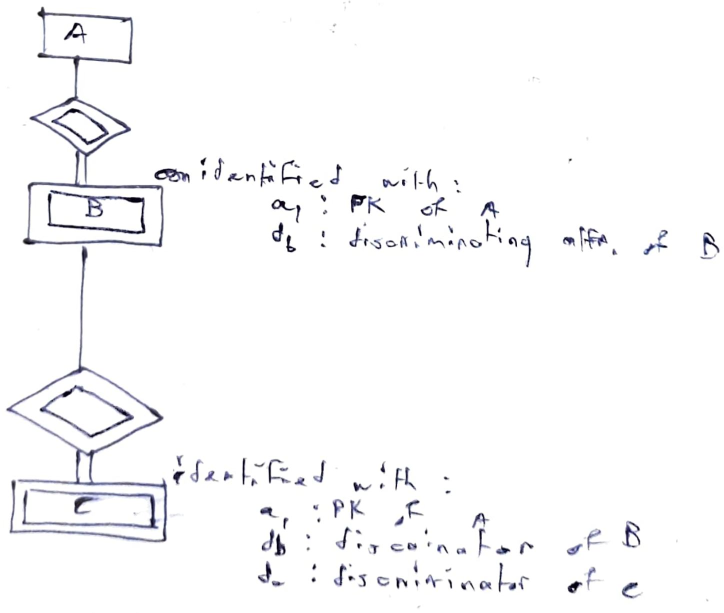
$(\overset{FK}{\text{V-ID}}, \overset{FK}{\text{ICODE}}, \overset{FK}{\text{DCODE}}, \text{DT-SUPPLY})$

reln. with reln. (i.e. assn. b/w 2 relns) not possible in ER model





Inheritance



identified with:
a_1 : PK of A
d_b : discriminating attr. of B

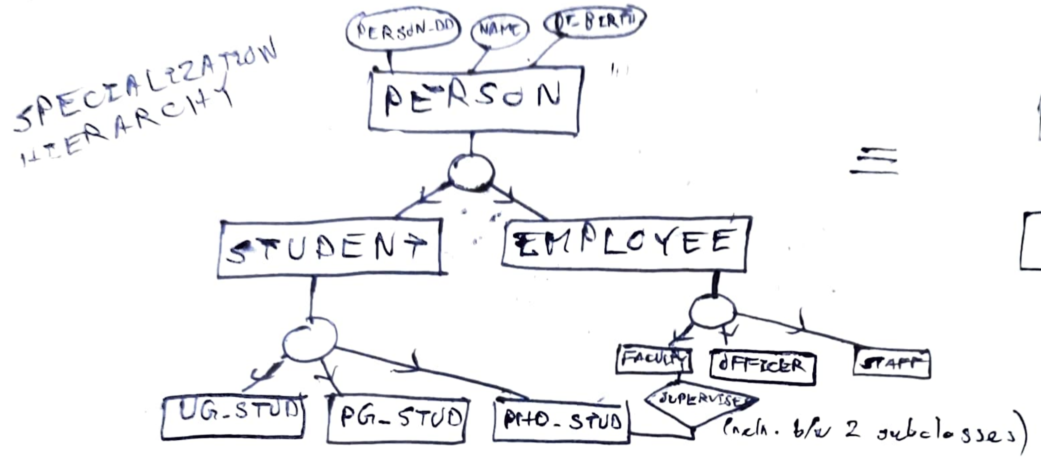
identified with :
a_1 : PK of A
d_b : discriminator of B
d_c : discriminator of C

- Extended ER model

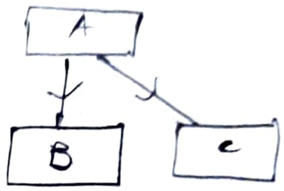
Super Class & Sub Class

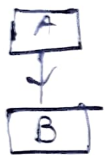
◦ Specialization

we have a super class
A subset of super class entity set have specialties
such subsets ⇒ subclasses



≡





B is subclass of A

In each subclass - superclass reln, a subclass has a single parent

- Why specialized subclass?
  → A subset of superclass may have addl. attr.
  → certain subclasses may have diff. relationship

A subclass with multiple superclasses ⇒ shared subclass (multiple inheritance)

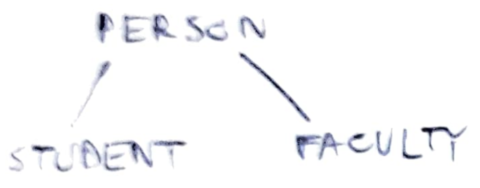
A specialization hierarchy with atleast one shared subclass, it's called lattice

Generalisation & Specialisation

22/1/26

- Constraints & Characteristics of Specialization

  - condn./predicate-defined subclasses
    → subclass can be identified based on certain criteria by looking into the attr. of the superclass



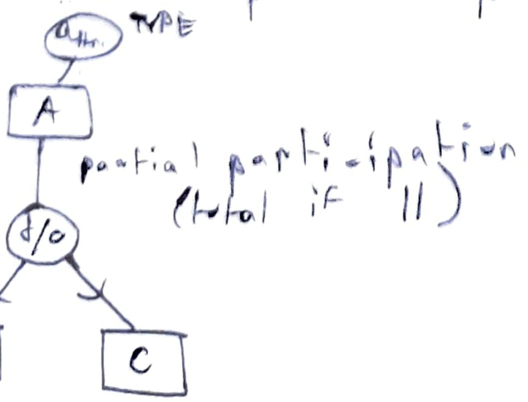
→ suppose there are files for PERSO
STUDENT & FACULTY
— each PERSON has person_id
— to find a person_id where to sea
— there must be some attr. in the PERSON file that determines whe
STUDENT or FACULTY

- attr. used for determining subclass
  → defining attr. is user-defined (?)

- disjointness constr. whether a superclass can belo to one or more subclasses.
  → if atmost one ⇒ disjoint specialization
  → more than one ⇒ overlapped

- completeness constraint whether a superclass instance must belong to atleast one subclass or not
  → if must ⇒ total participation
  else ⇒ partial participation

e.g.



- Mapping Generalization - Specialization to Relations of Relational Model

- single relation
  attr ⇒ attr. of superclass ∪ attr. of each of the subclasses
  can handle all cases of disjoint/overlapped & total/partial

Need a type attr.

$a_1 \ldots a_n$	$b_1 \ldots b_m$	$c_1 \ldots c_k$

if a superclass doesn't belong to any subclass ⇒ NULL

better if total, disjt & if no. of subclass attr. is v. less

If total & disjt. ⇒ some attr. null

overlapped ⇒ consider multibit type attr.

Ⓐ Multiple Relns (in general better design)

- one reln. for the superclass
  attr ⇒ superclass attr.

- one reln. for each subclass
  attr. ⇒ PK of superclass + subclass attr.
  & PK of superclass ⇒ PK/candidate key ~~1~~ also FK referring to superclass
  reln. so no (not create any discriminant key)

$A(a_1, a_2 \ldots a_n)$
$B(a_1, b_1 \ldots b_m)$
$C(a_1, c_1, c_2 \ldots c_k)$

Total, disjt
Total, overlapped
Partial, disjt.
Partial, overlapped

No addl. null value

Partial ⇒ instance into A, nothing in subclasses

Total ⇒ instance into A & in the subclasses

Disjt ⇒ in one subclass
overlapped ⇒ ... multiple subclasses

→ all instances of superclass (with general attr.)
→ all details of a subclass instance → go for equijoin (costly op.) (go for only if freq. req.)

Ⓑ Relations only for the subclasses
(no separate reln for superclass)

Attr ⇒ attr of superclass ∪ attr. of subclass

$B(a_1, a_2 \ldots a_n, b_1, b_2 \ldots b_m)$
$C(a_1, a_2 \ldots a_n, c_1, c_2 \ldots c_k)$

PK will be that of superclass

Total part. & disjt ⇒ all info goes to corresp. subclass

Total part. & overlapped ⇒ info stored in multiple reln. ⇒ redundancy

Partial

A superclass instance not belonging to any subclass
→ find all superclass instance
    ⇒ union oper. ✗
→ find all details ~~of~~ of subclass instance (equijoin is already supported)

Ⓒ Single Relation

attr → attr. of superclass ∪ attr. of each of the subclasses
    ∪ a type field
        ↘ to which subclass

$a_1, a_2 \ldots a_n$	$b_1 \ldots b_m$	$c_1 \ldots c_k$	TYPE

Ⓓ Multiple Types

Disjt. ✓
overlapped ✗
partial ⇒ lots of null values

Total/disjt.
& no. of addl. attr. is v. low ← Lots of Null values

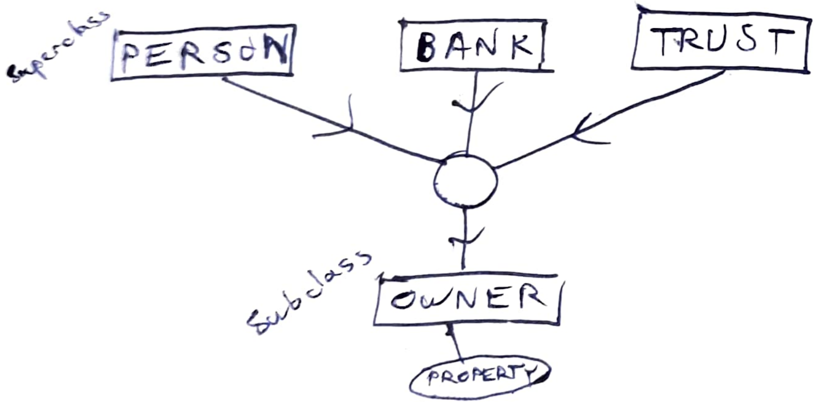
- all instances of superclass → outer join
- all details of subclass instances → join

} automatically supported

• <u>Category / Union</u>

A subclass has multiple superclasses of diff. entity type (may not have common key)

Subclass is called union/category



Multiple inherit

- each ~~of~~ gen-spec. has only one superclass
- subclass set will contain the intersection of superclass sets
- subclass will have all superclass attr.

→ each superclass : one reln.
→ $\frac{\text{union}}{\text{categ.}}$: only its own attributes

Union / Category

- categ. has multiple superclasses (maybe of diff. entity type)
- subclass set ⊂ union of superclass sets
- a subclass instance will have the attr. of corresp. superclass (not of all)
- union/categ. : only its own attr.

If there is no common key among superclasses ⇒ design a key
For category - surrogate key
copy this surrogate key in the superclasses as FK

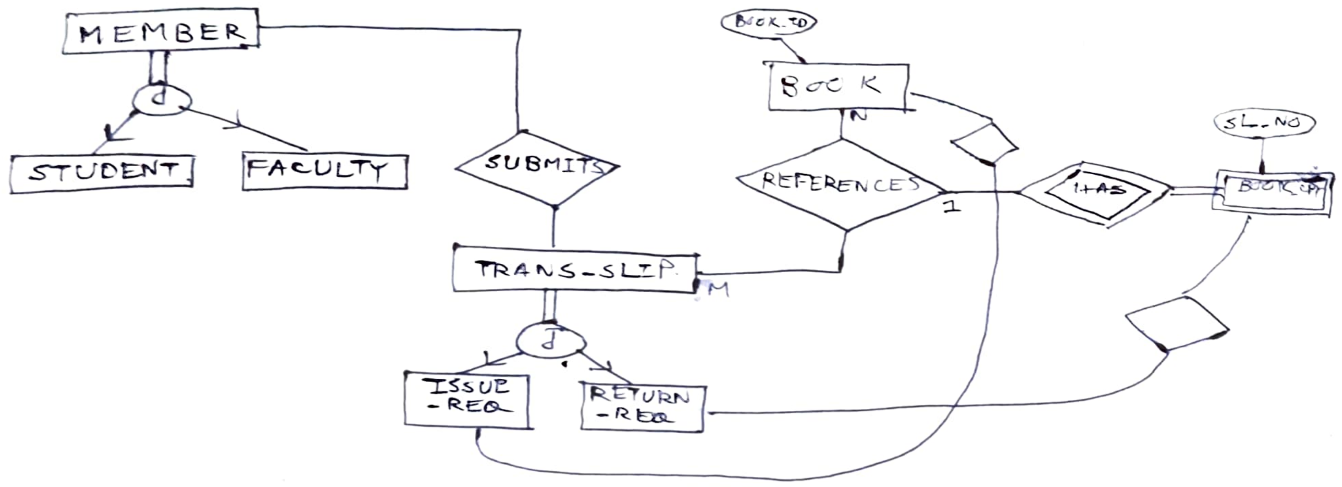
Total Participation

If subclass set ≡ union of superclass set
→ Maybe thought of as gen-spec.

$ cE1 + GT9	$
$cE1	$
$	$

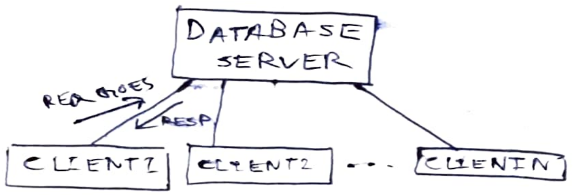
we are ordo
in reverse
are taking a
& reducing → such
are called handle &
what we are doing
called handle pruning
# the prefixes which
appear in stack in co
logic is called viable
prefix
e.g.

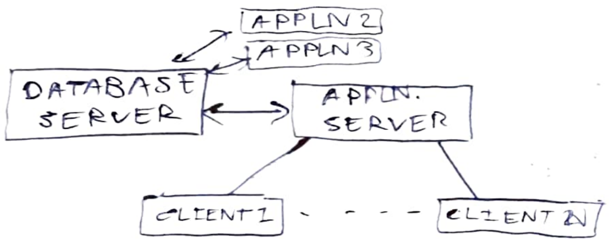
29/1/26



Relations
SQL (Structured Query Language)
→ statements of DDL Type
→ Statements of DML Type

2-TIER





ORACLE ⇒ Appln. called SQLPLUS → provides interface

DB server & client may not be on same device
To connect to server → needs hostname → encapsulates server addr, port addr, protocol etc.

SQL > 1 ___
2 ___
3 ___
4 ___

① To end a statement put ';' & press enter
— End of statement & execute
* after execution, it is in buffer

① Press enter in a blank line
⇓
End of the statement

Last command is stored in buffer

SQL > RUN → command in buffer is executed

# SQLPLUS is line-editor → doesn't allow change in prev. commands
# closing window → abrupt termination → may lead to data loss
⇒ go for SQL > EXIT;

DEPT (DCODE, DNAME)
SUBJECT (SCODE, SNAME, . . .)
STUDENT (ROLL, NAME, DT_BTH . . . , DCODE)
ATTENDANCE (ROLL, SCODE)
RESULT (ROLL, SCODE, SCORE)

RELATION IN RELATIONAL MODEL } → SQL TABLE

schema needs to be specified using DDL-like statements

Column name	type & size	constraints
	CHAR(5)	PRIMARY KEY
	VARCHAR (initial-size)	UNIQUE
	NUMBER (Total no. of digits, no. of digits after decimal)	NOT NULL, UNIQUE
		CHECK(CONDITION)

SQL > SELECT *
FROM CAT; } → lists all my tables

SQL > DESC STUDENT ↵

```
SQL> CREATE TABLE SUBJECT
     (SCODE CHAR(5) PRIMARYKEY CONSTRAINT PK_SUBJECT,
     SNAME   CHAR(10)   NOTNULL,
     CATEGORY  CHAR(2),
     TYPE  CHAR(1)  CHECK(TYPE = 'T' OR TYPE = 'S'));
```

```
SQL> CREATE TABLE STUDENT
     (ROLL  NUMBER(3,0)  PRIMARY KEY,
      NAME   CHAR(20),
      DT_BTH   DATE,
FK, ~~[illegible]~~ DCODE   CHAR(5)  REFERENCES  DEPT(DCODE));
```

→ constraint mentioned at describing the colm. so column-level constraint

FK, have to have some domain as the ~~PK~~ referenced attn. i.e. has to have same type & size

```
SQL> CREATE  TABLE  ATTRIBUTE
     (ROLL  NUMER(3,0)  REFERENCES  STUDENT(ROLL),
      SCODE  CHAR(5)  REFERENCES  SUBJECT(SCODE),
      PRIMARY KEY(ROLL, SCODE) CONSTRAINT  NAME);
```

For any col. exc composite attr, constr can be kept both at col. & table-level ~~[illegible]~~ For comp. → only table

RESULT will contain tuple only ~~when~~ for comb. of ROLL & SCODE present in ATTENDANCE

```
SQL> CREATE  TABLE  RESULT
     (ROLL  NUMBER(3,0),
      SCODE  CHAR(5),
      SCORE  NUMBEP(3,0), DEFAULT 0,
      PRIMARY KEY (ROLL, SCODE),
      FOREIGN KEY(ROLL, SCODE) REFERENCES ATTENDANCE(ROLL, SCODE
```

composite foreign key so written this way

```
SQL> DROP TABLE tablename;
```

schema & its contents both are deleted