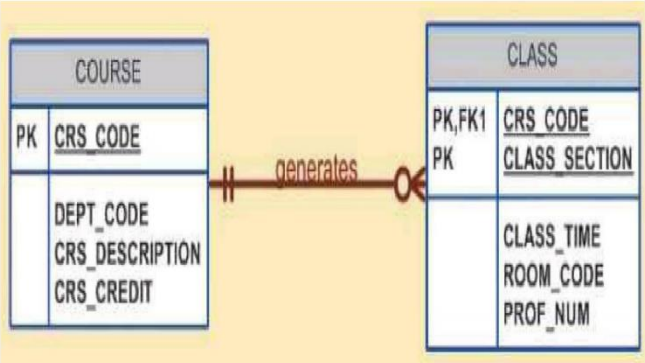
Cenan 2021 RO exam

Subject 2

Entity Types

A weak entity depends on another entity (strong entity).

In the weak entity, a part of the primary key is a foreign key to the strong entity.



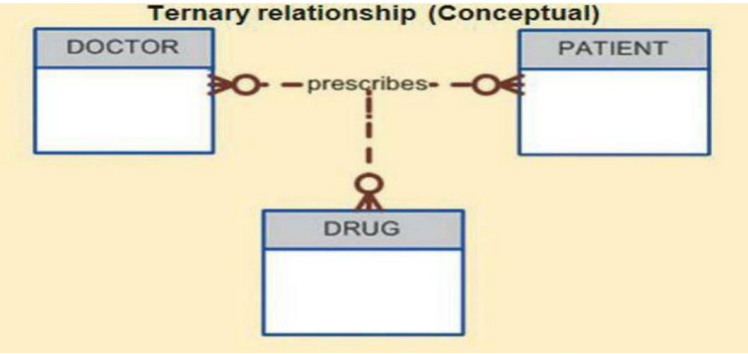
Here, course is a strong entity, and class is a weak entity, because Class's PK is made up of the PK of Course (and other items). Similar to UML aggregation ?

Relationships

Unary relationship can be a recursive one (employee-manager) a.k.a one-entity-relationship.

Binary relationships are the most common (like the one above).

Ternary relationships involve 3 entities (and are usually split up into binary rels).



Participation Constraints

If the existence of an entity is mandatory => mandatory participation constraint

Not mandatory => optional participation constraint

Nullable constraint

Example: an employee does not need to have a manager (he is the boss) => optional

Specialization-Generalization

Like OOP inheritance for DBs.

Ex: Item class(table) ← Book table, Tea table, etc… (top-down)

Generalization = the same, but the other way around (bottom-up)

Subject 3

(a.1.) $A \rightarrow B, C \rightarrow D, BD \rightarrow E, E \rightarrow C$

a.1. We take all fields (A, B, C, D, E), and we look if we find one of them on the right => we eliminate them.

B, C, D, E are on the right => A **must** be part of the PK (along with others)

We take the identified attributes (A) and compute their transitive closure:

From A, we can determine B, from B we can determine **nothing**.

From AB, nothing.

- ⇨ Transitive closure of A = AB (how much we can determine from A) => A alone cannot be PK (we cannot determine all fields from it)

Next, we add one letter to the previous combination (A), to see if the new key can be a candidate key.

Take AB => already done

Take AC => compute $(AC)^+$ = AC(BD) = ACBD(E) => AC candidate key

Take AD => ABCDE (candidate key 2=AD), AE => ABCDE (key 3=AE)

Take 3 field combinations => cannot => we are done

Minimal Keys = AC, AD, AE (Prime attributes = A, C, D, E)

1NF: any tabular data is in 1NF (trivial)

2NF:

Now, we look at our initial formulas and try to find those that:

- ⇨ On the left have **only** part of a minimal key (not complete keys)
- ⇨ On the right have a non-prime attribute (B in this case)

We found A -> B => data is not 2NF

(a.2.) AB → C, C → D, C → E, D → A, E → B

a.2

C, D, E, A, B on the right => none of them have to be included in the PK

A => A, B => B, C=>CDAEB, D=>DA, E=>EB, (C) key

At every step, take keys that do not contain the previously found minimal key (C in this case)

AB=>ABCDE, AD=>AD, AE=>AEBCD, (AB, AE) keys

BD=>BDACE, BE=> BE, DE=> DEABC, (BD, DE) key

Keys: **C, AB, AE, BD, DE**, prime attributes: **A, B, C, D, E**

Is 1NF, is 2NF

3NF:

on the right -> only prime attributes

on the left -> a key (those previously determined)

if **both** fail => not 3NF

This one is 3NF

BCNF:

on the left -> a key (those previously determined)

Not BCNF (due to D->A and E->B)

Transactions, ACID

Ex: stock buying/selling transactions

Transaction = a unit of work in the DB (atomicity)

Operations: Initiate transaction, release funds, receive stocks

Atomic: either all of them succeed or, if any fails, the entire operation fails (rollback)

Consistent: when the state cannot be reached (not enough funds, etc…)

Isolation: transactions' effects should not influence other transactions (like they are sequential)

Durability: after transaction committed, effects remain even after power loss (for ex)

Subject 4

- person(pId, name, address)
- title(titleId, name, yearPublished)
- item(itemNumber, title (refer title.titleId), dateAcquired, pricePaid)
- borrowing(book_item (→ item.itemNumber), dateBorrowed, borrower (→ person.pID), dateReturned, penalty)

a. Lista titlurile dobândite în cursul anului 2019 în ordine alfabetică
b. Lista persoanelor care în prezent au cărți împrumutate
c. .Lista persoanelor care nu au împrumutat vreo carte în anul 2020
d. Cât de multe cărți de "baze de date" sunt disponibile
e. Câți bani au fost utilizai pentru achiziționarea cărților în 2020
f. Care a fost venitul anual obținut din penalizări
g. Pentru care cărți venitul din penalizări a depășit prețul de achiziție

a.

```
Select title.name
from title join item on item.title=title.titleId
where year(dateAcquired) = 2019
order by title.name
```

b.

```
select person.name
from person join borrowing on pId=borrower
where dateReturned is NULL
```

c.

```
select person.name

from person p1

where pId not in (select pId from person p2 join borrowing on p2.pId=borrower where
year(dateBorrowed) = 2020)
```

d.

```
select count(name)

from title join item on titleId=title join borrowing on book_item=itemNumber

where name like '%Databases%' and dateReturned is not NULL
```

e.

```
select sum(pricePaid)

from item

where year(dateAcquired)=2020
```

f.

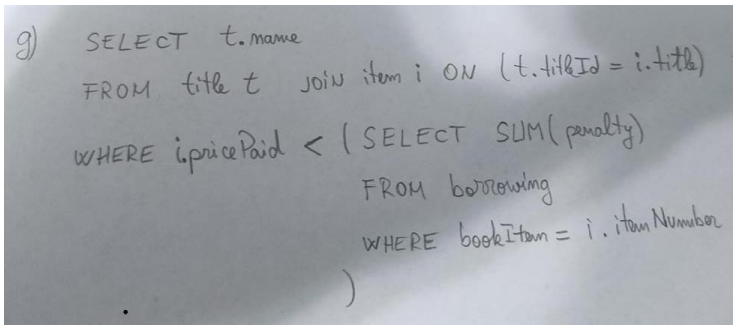
```
select year(dateReturned), sum(penalty)

from borrowing

where dateReturned is not NULL

group by year(dateReturned)
```

g.




```
select t.name

from title t join item I on
t.titleId=i.title join borrowing b on
b.book_item=i.itemNumber

group by i.id

having sum(penalty) > i.pricePaid
```