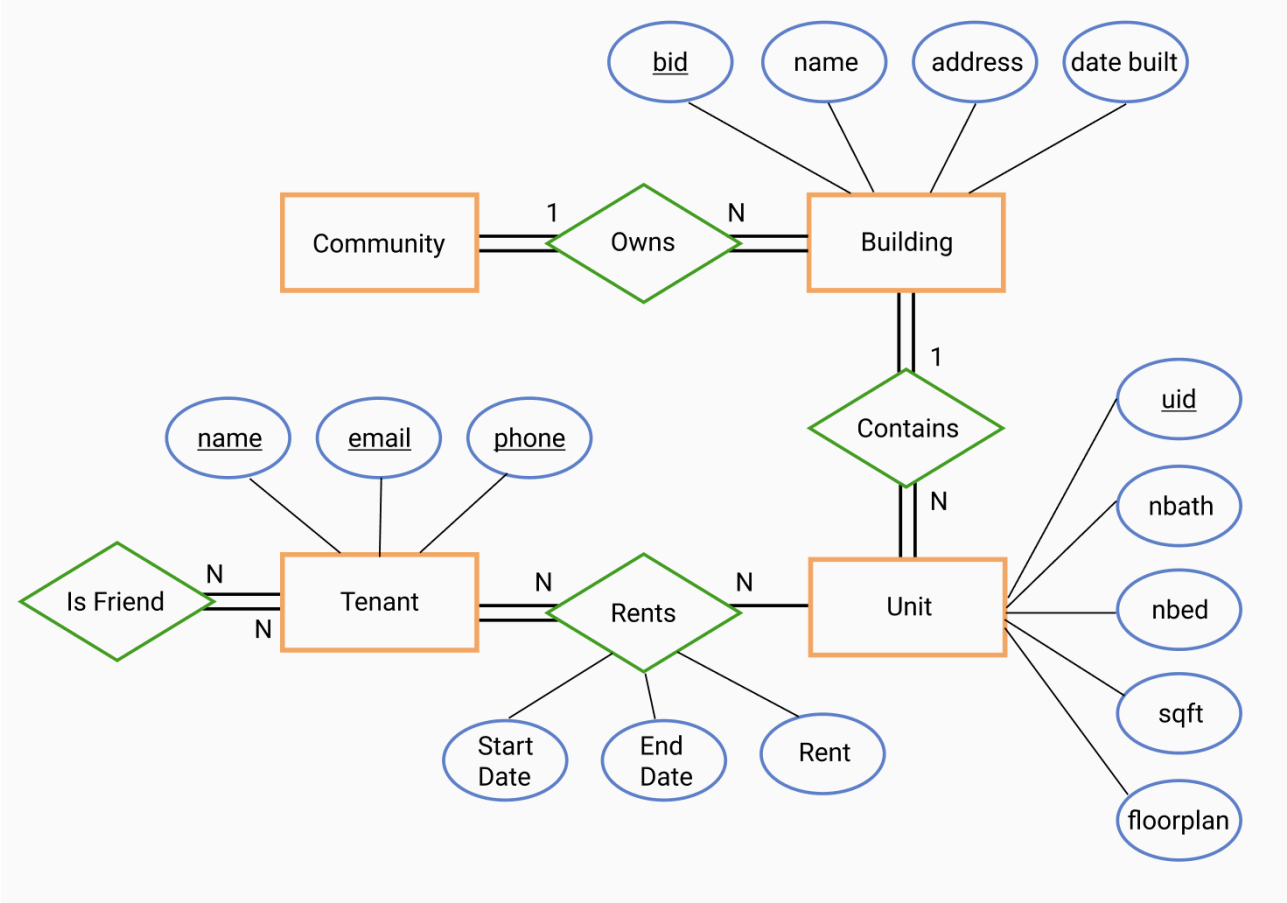
1.



Assumptions:

For ‘Owns’ relationship:

1. Each community owns N building.
2. Each building is owned by 1 community.
3. Each building must belong to one community.
4. Each community must have 1 or more buildings.

For ‘Contains’ relationship:

1. Each building contains N units.
2. Each unit is contained in 1 building.
3. Each building must have unit.

4. Each unit must belong to one building.

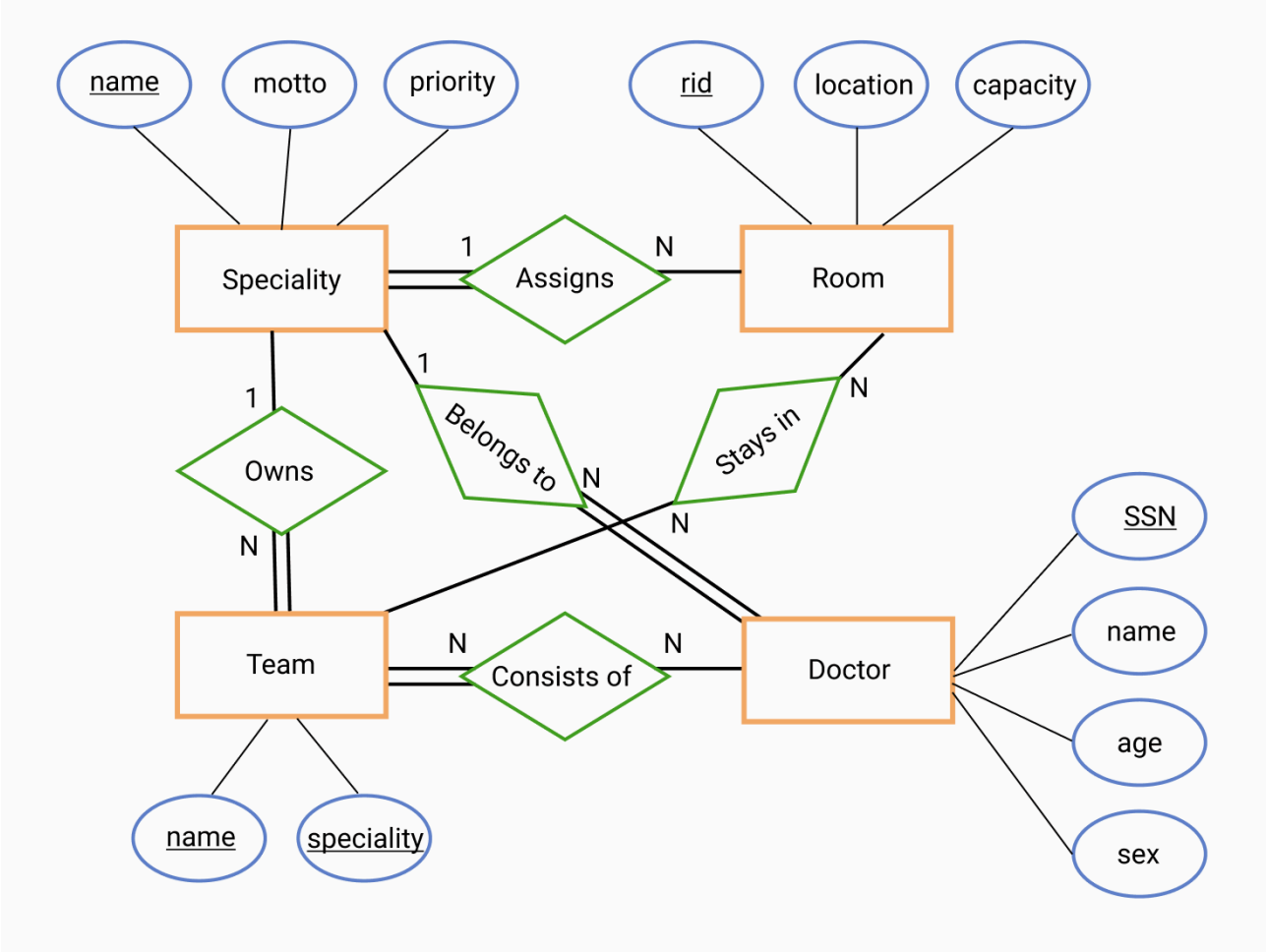
For ‘Rents’ relationship:

1. Each tenant can rent N units.
2. Each unit can be rented by N tenants.
3. Each tenant must rent 1 or more units.
4. Each unit need not have any tenants.

For “Is Friend” relationship:

1. Each tenant can have N friends.
2. Each friend need not have any friends.

2.



Assumptions:

For ‘Assigns’ relationship:

1. Each speciality assigns N rooms.
2. Each room belongs to 1 speciality.
3. Each speciality must have rooms assigned to it.
4. Each room need not belong to any speciality.

For ‘Consists of’ relationship:

1. Each team consists of N doctors.
2. Each doctor can be consisted in N teams.
3. Each team must have doctors.
4. Each doctor need not belong to any team.

For ‘Owns’ relationship:

1. Each speciality owns N teams.
2. Each team is owned by 1 speciality.
3. Each speciality needs not own any teams.
4. Each team must be owned by one speciality.

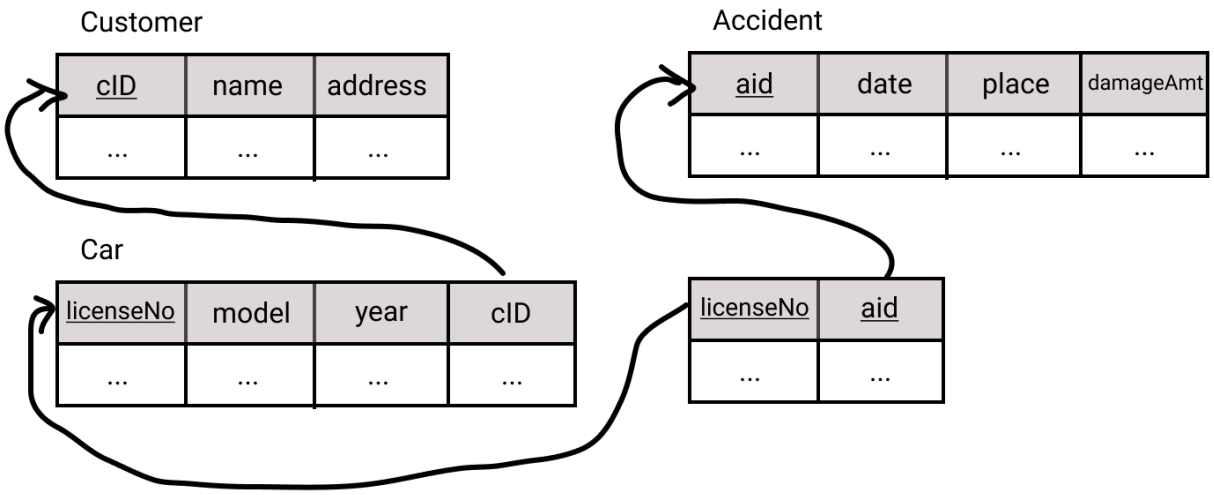
For ‘Belongs to’ relationship:

1. Each speciality has N doctors.
2. Each doctor belongs to 1 speciality.
3. Each speciality needs not have any doctors.
4. Each doctor must belong to a speciality.

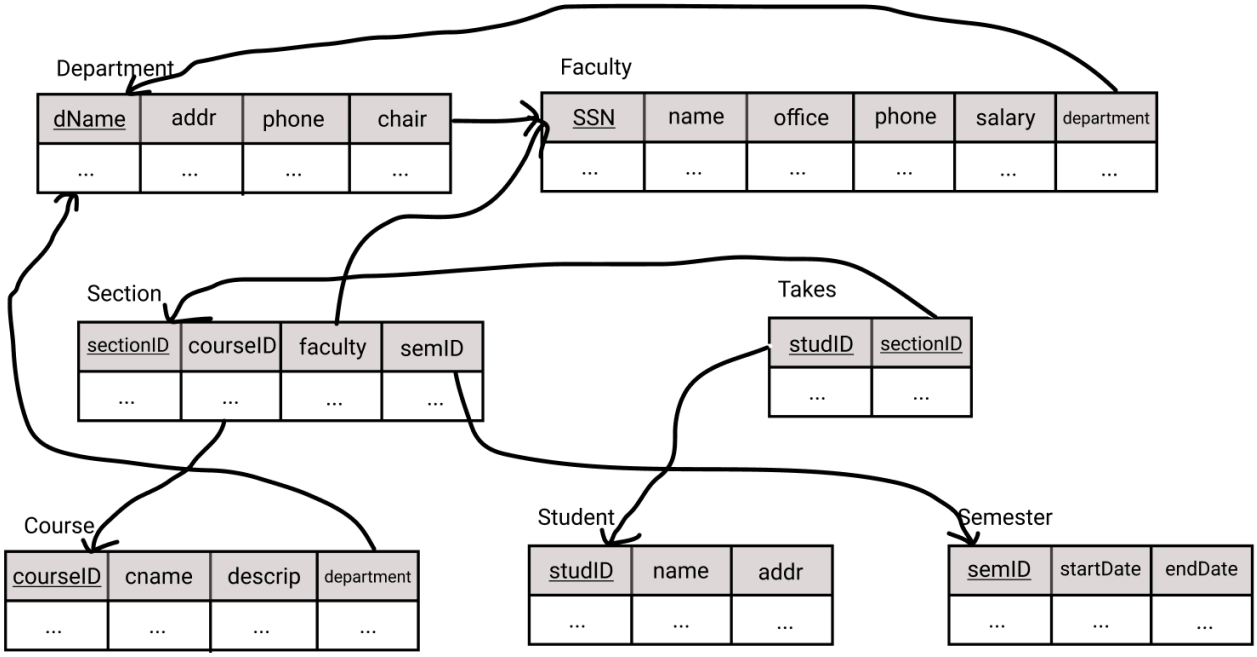
For ‘Stays in’ relationship:

1. Each team stays in N rooms.
2. Each room contains N teams.
3. Each team needs not stay in any room.
4. Each room need not have teams stay in.

3.



4.



5.

(a) The key of the relation is saleID. As $saleID^+$ = {saleID, saleTime, gameTitle, gamePublisher, publisherCutPercent, quantity, price, customerID, address, creditCardNo} based on closure algorithm and it is the only minimal super key.

(b) The first five fictional dependencies violate BCNF as non-key attributes functionally dependent on attributes(gameTitle, customerID) that are not super key. For example, price is functionally dependent on gameTitle, where gameTitle is not a super key.

(c) Step1: Find all keys

- Key: saleID

Step2: Is R 2NF?

- Key: saleID
- Non-key attributes: saleTime, gameTitle, gamePublisher, publisherCutPercent, quantity, price, customerID, address, creditCardNo
- R is 2NF as there are no non-key attributes functionally depend on a subset of the key.

Step3: Is R 3NF?

- No. There are non-key attributes functionally dependent on attributes that are not super key (e.g. gameTitle → price).

Step4: Extract offending functional dependence

- R1(<u>gameTitle</u>, price, gamePublisher)
- R2(<u>gamePublisher</u>, publisherCutPercent)
- R3(<u>customerID</u>, address, creditCardNo)
- R4(<u>saleID</u>, saleTime, gameTitle, quantity, price, customerID)

Step5: Check the new relations if they are BCNF

- For R1, R2, R3, R4, whenever a nontrivial functional dependency X —> A holds in R, then X is a superkey of R.

6.

(a) The keys of the relation are BEH, BEF, and CEH.

Since E doesn't appear on the RHS, E must be included in the keys.

{BE} = A, B, C, D, E, G

add H: (B, E, H) = A, B, C, D, E, F, G -> Key!

add F: (B, E, F) = A, B, C, D, E, F, G -> Key!

(B, F) add E same as above

(C, E) = A, C, E, G

add B: (B, C, E) = A, B, C, D, E, G

add H: A, B, C, D, E, F, G -> Key!

add F: A, C, E, F, G

(C, E, H), (C, H) add, same as above

(b) Choose (C, E, H) as the key. The relation R is not in 3NF because it is not in 2NF. There is embedded autonomous entity in the relation. For example, C, H→B means B is functionally dependent on a subset of key, which is a violation of 2NF. Also there are non-key attributes functionally dependent on attributes that are not super key. For example B→C, D where B is not a superkey.

(c) Choose (C, E, H) as the key.

Step1: Find all keys

- Key: (C, E, H)

Step2: Is R 2NF?

- No. There are no non-key attributes functionally depend on a subset of the key, (e.g. C, H -> B).

Step3: Is R 3NF?

- No. There are non-key attributes functionally dependent on attributes that are not super key (e.g. B -> C, D).

Step4: Extract offending functional dependence

- R1(<u>B</u>, C, D)
- R2(<u>C</u>, A, G)
- R3(<u>C</u>, <u>E</u>, <u>H</u>, F)
- R4(<u>C</u>, <u>H</u>, B)

Step5: Check the new relations if they are 3NF

- For R1, R2, R3, R4, whenever a nontrivial functional dependency X —> A holds in R, then X is a superkey of R.

(d) The relation R is not in BCNF as R in neither in 2NF (There are no non-key attributes functionally depend on a subset of the key, e.g. C, H -> B) nor 3NF (There are non-key attributes functionally dependent on attributes that are not super key e.g. B -> C, D).

(e)

Step1: Find all keys

- Key: (C, E, H)

Step2: Is R 3NF?

- No. There are non-key attributes functionally dependent on attributes that are not super key (e.g. B -> C, D).

Step3: Is R BCNF?

- No. nontrivial functional dependency X —> A in R, where X is not a superkey of R (C→ A, G & C, H → B).

Step4: Extract offending functional dependence

- R1(B, C, D)
- R2(C, A, G)
- R3(C, E, H, F)
- R4(C, H)
- R5(B, F, H)

Step5: Check the new relations if they are 3NF

- For R1, R2, R3, R4, R5, whenever a nontrivial functional dependency X —> A holds in R, then X is a superkey of R.

(f) The resulting decomposition is not functional dependency-preserving. The functional dependency C, H -> B is lost when we decompose R from from 3NF to BCNF.

Honor Code:
/* THIS CODE IS MY OWN WORK.
IT WAS WRITTEN WITHOUT CONSULTING CODE WRITTEN BY OTHER STUDENTS.

Zhou Fang*/