201300180 Data & Information – Test 2, 20.05.2016 – Solutions

Question 1

For a minimal number of tables, the covering generalization is represented by tables only for the subclasses; the two classes with a 1—1 association are represented by a single table. Depending on what you choose as key for the combined table, there are two slightly different versions.

Version 1

```
Person(p_id, surname, first_name, telephone_no,
      PK(p_id));

Company(name, address, contact_person NOT NULL, email,
      PK(name),
      FK contact_person REF Person(p_id),
      UNIQUE(contact_person));

Company_customer(p_id, address, company_name,
      PK(p_id),
      FK(p_id) REF Person,
      FK(company_name) REF Company(name)

Private_customer(p_id, address, credit_card_no,
      PK(p_id),
      FK(p_id) REF Person);
```

Version 2

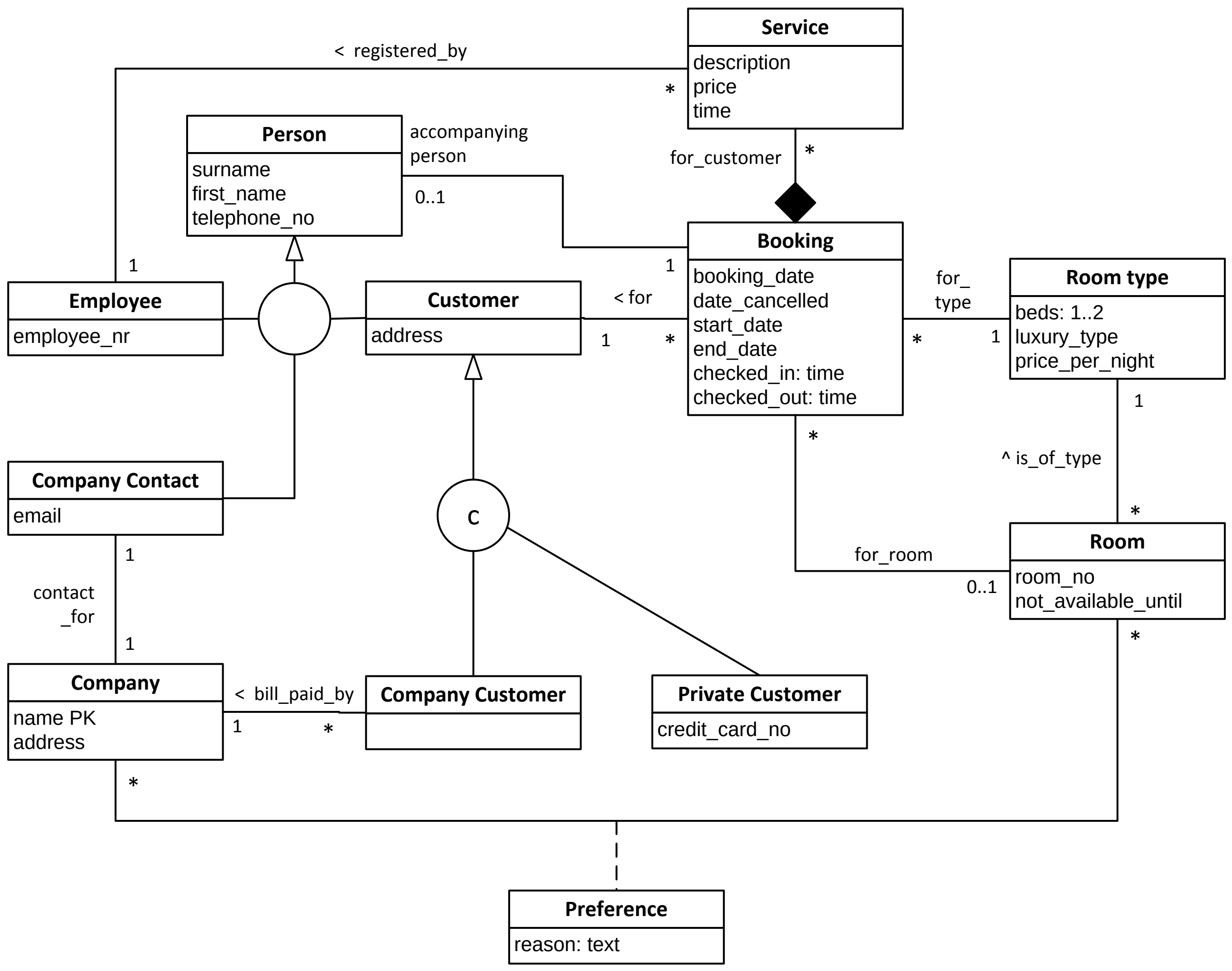
```
Person(p_id, surname, first_name, telephone_no,
      PK(p_id));

Company_contact(p_id, email, company_name NOT NULL, address,
      PK(p_id),
      FK(p_id) REF Person,
      UNIQUE(company_name));

Company_customer(p_id, address, contact_id
      PK(p_id),
      FK(p_id) REF Person,
      FK(contact_id) REF Company_contact(p_id));

Private_customer(p_id, address, credit_card_no,
      PK(p_id),
      FK(p_id) REF Person);
```

Question 2



Remarks:

- Note the difference between *Room* and *Room Type*. The term *Room Type* is not mentioned in the text, but from the description it is clear that a booking is for a room type, and a specific room is allocated allocated when the visitor arrives.
- *Employee* has all attributes of *Person*, plus another one. That makes it most suited to model it as a subclass.
- *Accompanying person* is yet another type of person. It can be modelled with a subclass with no attributes and an association with booking, but a subclass is not really needed as the superclass *Person* describes everything we need to know from an accompanying person (and the generalization is not covering). The accompanying person is modelled by means of a role in the assocation.
- The assocation *for_customer*, associating *Service* with *Booking* is a proper composition. It is not intended record all services, the only reason to record is that it needs to be paid at check-out, making it dependent on *Booking*.

Question 3a

i) $AD \longrightarrow B$ No. There could be different bookings for the same dates (c states the reverse, i.e., $B \longrightarrow AD$)

ii) $S \longrightarrow AD$ Yes. From e. and c. With the contextual knowledge of the case description it is obvious that „stay“ refers to „booking“, hence $S \longrightarrow B$ *(e)*. With $B \longrightarrow AD$ *(c)*, we find $S \longrightarrow AD$.

iii) $C \longrightarrow I$ No. *C* and *I* are not in any way related.

iv) $I \longrightarrow F$ Yes, from g.

v) $I \twoheadrightarrow F$ Yes. Two arguments are possible:
(1) $I \longrightarrow F$ implies $I \twoheadrightarrow F$, therefore the MVD can be deduced from iv) or g.
(2) *F* is in not any way related to any of *ABCDRST.*

vi) $R \twoheadrightarrow IF$ Yes. *IF* is not in any way related to *ABCDST.*

vii) $E \longrightarrow C$ No. The same employee can deliver services to different customers.

viii) $CE \longrightarrow R$ No. If a customer stays more than once, the bookings can be for different rooms. Likewise, although $S \longrightarrow B$ is correct, $E \longrightarrow S$ does not hold.

ix) $CS \longrightarrow R$ Yes. From $S \longrightarrow B$ (see (ii)) and $B \longrightarrow R$ (b) we find $S \longrightarrow R$, and then also $CS \longrightarrow R$.

x) $B \longrightarrow E$ No. We do have $S \longrightarrow E$, but there can be different services for a booking.

Question 3b

1) In order to find out which FDs violate the BCNF condition, we first have to establish the candidate keys. Schema *R* has one candidate key: *ST*.
(You can find this by starting with *ABCDRST* as a trivial superkey, and discard attributes that are fuctionally dependent. *ACDR* are dependent on *B* and can be left out. From the resulting *BST*, we can eliminate *B* because of $S \longrightarrow B$, yielding *ST* as candidate key.)

All FDs in $\mathfrak{F}$ violate the BCNF condition, because all of them have a left-hand side that is not a superkey.

2) First, determine $\mathcal{F}^{+} = \{ B \longrightarrow ACDR,\ S \longrightarrow ABCDR,\ T \longrightarrow C \}$

For the remainder of 2) and 3) the solution differs depending on which – arbitrarily chosen – FD you start with.

(i) Start with (arbitrarily chosen) functional dependency $S \longrightarrow ABCDR$.

$(S)^{+} = ABCDRS$. Splitting over S we get

- $R_1(S,A,B,C,D,R)$, with $\mathcal{F}_1 = \{ B \longrightarrow ACDR,\ S \longrightarrow ABCDR \}$
- $R_2(S,T)$, with $\mathcal{F}_2 = \{ \}$

Clearly, R_2 is in BCNF, candidate key is ST.

For R_1 we find candidate key S (all other attributes depend on S).
R_1 is not in BCNF, however, as $B \longrightarrow ACDR$ violates the condition.

So we split R_1 on $B \longrightarrow ACDR$ and determine $(B)^{+} = BACDR$.
This yields

- $R_{11}(A,B,C,D,R)$, with $\mathcal{F}_{11} = \{ B \longrightarrow ACDR \}$
- $R_{12}(B,S)$, with $\mathcal{F}_{12} = \{ S \longrightarrow B \}$

R_{11} has candidate key B and is in BCNF,
R_{12} has candidate key S and is in BCNF.

From the original functional dependencies, $T \longrightarrow C$ was lost in the decomposition in step 1.
The other FDs still exist in $\mathcal{F}_{11} \cup \mathcal{F}_{12} \cup \mathcal{F}_2$.

(ii) Start with (arbitrarily chosen) functional dependency $B \longrightarrow ACDR$.

$(B)^{+} = ABCDR$. Splitting over B we get

- $R_1(A,B,C,D,R)$, with $\mathcal{F}_1 = \{ B \longrightarrow ACDR \}$
- $R_2(B,S,T)$, with $\mathcal{F}_2 = \{ S \longrightarrow B \}$

Clearly, R_1 is in BCNF, candidate key is B.

For R_2 we find candidate key ST.
R_2 is not in BCNF, however, as $S \longrightarrow B$ violates the condition.

So we split R_2 on $S \longrightarrow B$ and determine $(S)^{+} = SB$.
This yields

- $R_{21}(S,B)$, with $\mathcal{F}_{21} = \{ S \longrightarrow B \}$
- $R_{22}(S,T)$, with $\mathcal{F}_{12} = \{ \}$

R_{21} has candidate key S and is in BCNF,
R_{22} has candidate key ST and is in BCNF.

From the original functional dependencies, $T \longrightarrow C$ was lost in the decomposition in step 1.
The other FDs still exist in $\mathcal{F}_1 \cup \mathcal{F}_{21} \cup \mathcal{F}_{22}$.

(iii) Start with (arbitrarily chosen) functional dependency $T \longrightarrow C$.

$(T)^{+}$ = TC. Splitting over T we get

- $R_1(T,C)$, with $\mathfrak{F}_1 = \{ T \longrightarrow C \}$
- $R_2(A,B,D,R,S,T)$, with $\mathfrak{F}_2 = \{ B \longrightarrow ADR, S \longrightarrow ABDR \}$

Clearly, R_1 is in BCNF, candidate key is T.

For R_2 we find candidate key ST.
R_2 is not in BCNF, both FDs violate the condition.

So we split R_2 on (arbitrarily chosen) $B \longrightarrow ADR$ and determine $(B)^{+} = ABDR$.
This yields

- $R_{21}(A,B,D,R)$, with $\mathfrak{F}_{21} = \{ A \longrightarrow BDR \}$
- $R_{22}(\mathrm{B},S,T)$, with $\mathfrak{F}_{12} = \{ S \longrightarrow B \}$

R_{21} has candidate key B and is in BCNF,
R_{22} has candidate key ST and is not in BCNF, as the FD violates the condition.

So we split R_{22} on $S \longrightarrow B$ and determine $(S)^{+} = SB$.
This yields

- $R_{221}(B,S)$, with $\mathfrak{F}_{21} = \{ S \longrightarrow B \}$
- $R_{222}(S,T)$, with $\mathfrak{F}_{12} = \{ \}$

R_{221} has candidate key S and is in BCNF,
R_{22} has candidate key ST and is in BCNF.

From the original functional dependencies, $B \longrightarrow C$ was lost in the decomposition in step 1.
The other FDs still exist in $\mathfrak{F}_1$ U $\mathfrak{F}_{21}$ U $\mathfrak{F}_{221}$ U $\mathfrak{F}_{222}$.

Note that in iii), the second and third step can be reversed, giving the same result.