5)

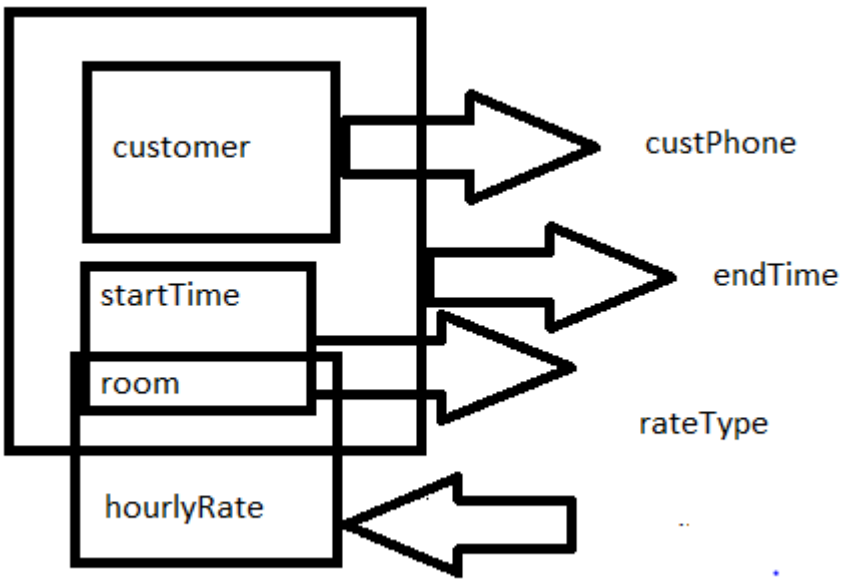
• customer -> custPhone

• (customer, room, startTime) -> endTime

• rateType -> room, hourlyRate

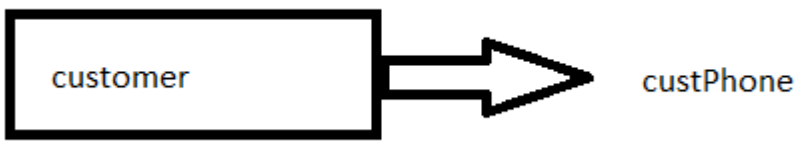
• (room, startTime) -> rateType

We established that all intersection of each tuple and attribute contains one and only one value (atomic) hence the above relations are in in 1NF. Now, we must convert them into second normal form (2NF) if it is in first normal form (1NF) and every non-primary-key attribute is fully functionally dependent on the primary key.



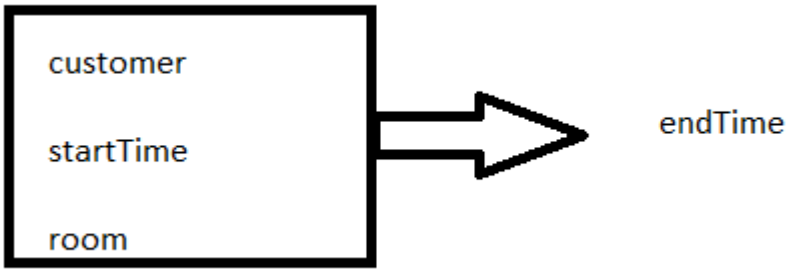
The primary key (composite keys) from the above dependency shows (customer, startTime, room). We must make every non-primary key to be fully functionally dependent on the primary key. Remove all partial dependencies exist which are customer → custPhone and (startTime, room) → rateType by separating the dependencies individually.

CUSTOMER



Primary key: customer

CUSTOMERROOMBOOKING

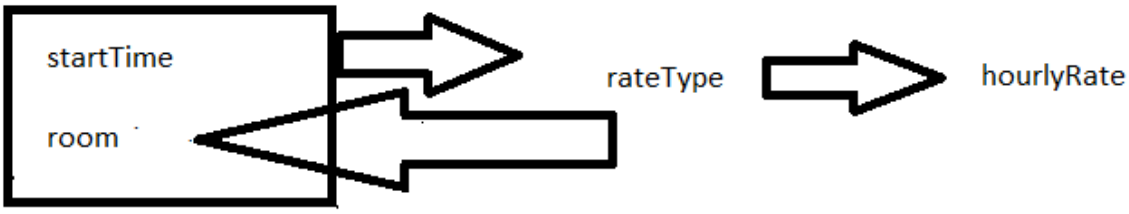


Primary key: (customer, startTime, room)

Foreign key: customer references CUSTOMER (customer)

Foreign key: (startTime, room) references ROOMRATE (startTime, room)

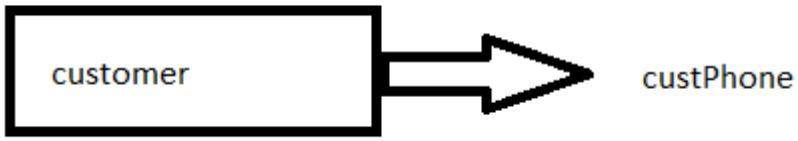
ROOMRATE



Primary key: (startTime, room)

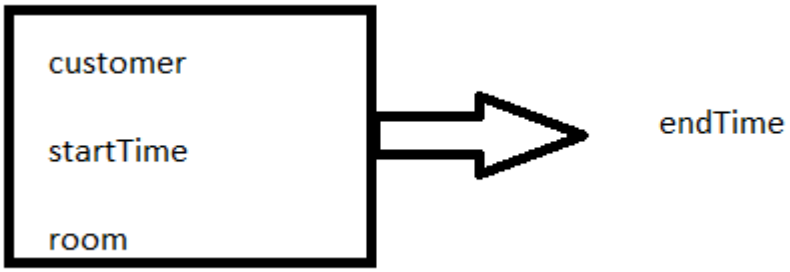
After which, we must now convert the above relations into (3NF) if it is already in (2NF) and in which no non-primary-key attribute is transitively dependent upon its primary key. The only transitive dependency from above relations that needs to be removed is rateType → hourlyRate.

CUSTOMER



Primary key: customer

CUSTOMERROOMBOOKING

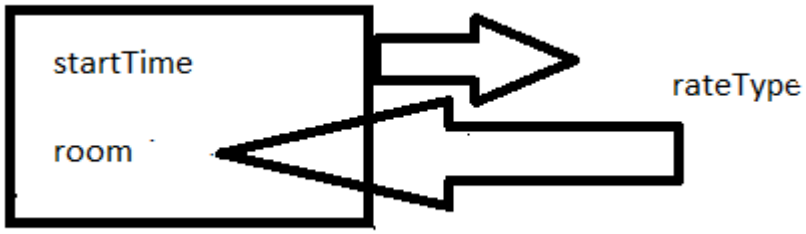


Primary key: (customer, startTime, room)

Foreign key: customer references CUSTOMER (customer)

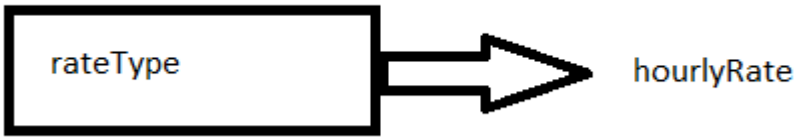
Foreign key: (startTime, room) references ROOMRATE (startTime, room)

ROOMRATE



Primary key: (startTime, room)

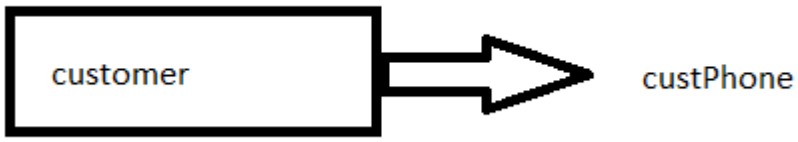
RATE



Primary key: rateType

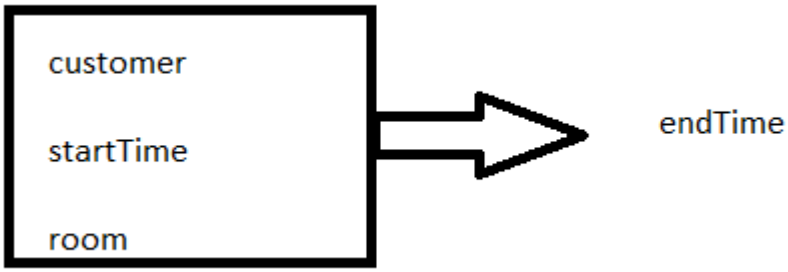
Lastly, we need to convert above relations into Boyce-Codd Normal Form (BCNF) states that a relation is in BCNF if and only if every determinant is a candidate key. The only trivial dependency from the above relations is rateType → room. To resolve this BCNF violation is to insert the room attribute into the relational table RATE.

CUSTOMER



Primary key: customer

CUSTOMERROOMBOOKING



Primary key: (customer, startTime, room)

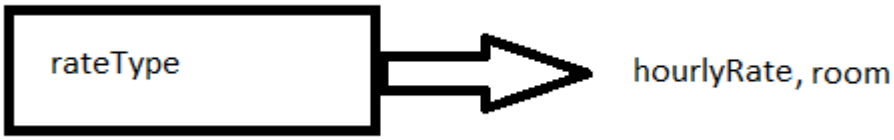
Foreign key: customer references CUSTOMER (customer)

Foreign key: (startTime, room) references ROOMRATE (startTime, room)

ROOMRATE

Primary key: (startTime, room)

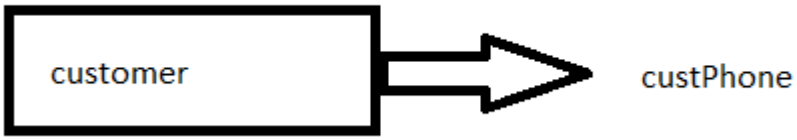
RATE



Primary key: rateType

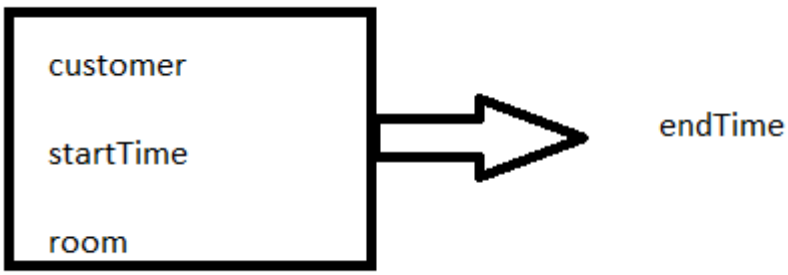
From this statement, each room has a different rate type and each room may have different charge depending on the time (period) when the room is booked. We need to alter the relational table ROOMRATE.

CUSTOMER



Primary key: customer

CUSTOMERROOMBOOKING

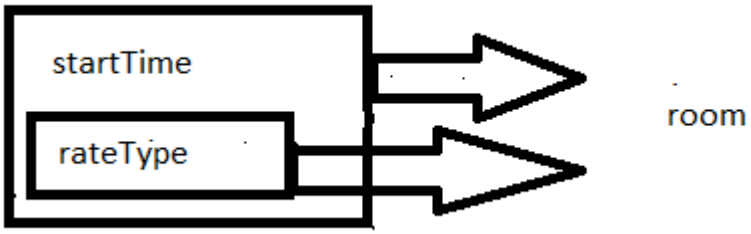


Primary key: (customer, startTime, room)

Foreign key: customer references CUSTOMER (customer)

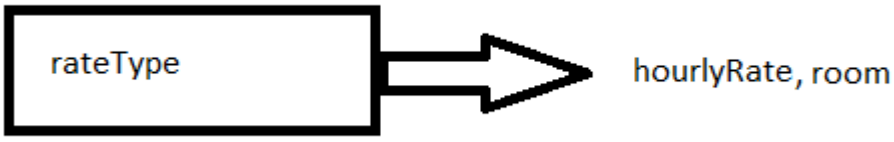
Foreign key: (startTime, room) references ROOMRATE (startTime, room)

ROOMRATE



Primary key: (startTime, room)

RATE



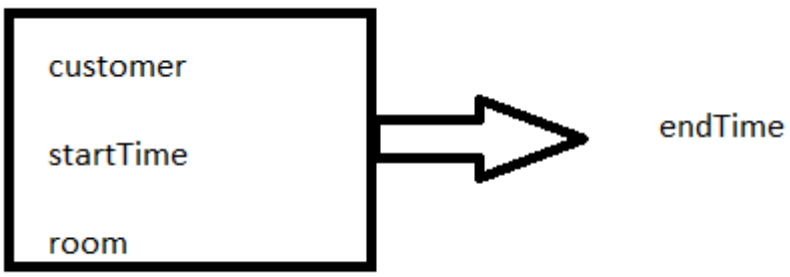
Primary key: rateType

From the new ROOMRATE, there exists a partial dependency hence we must remove it.

CUSTOMER

Primary key: customer

CUSTOMERROOMBOOKING

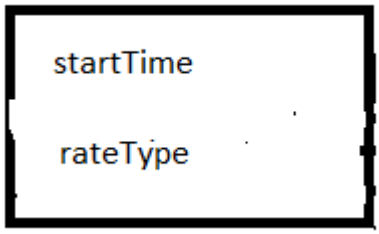


Primary key: (customer, startTime, room)

Foreign key: customer references CUSTOMER (customer)

Foreign key: (startTime, room) references ROOMRATE (startTime, room)

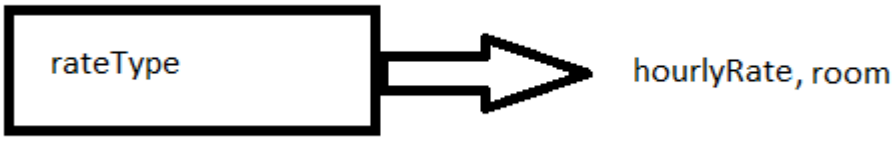
ROOMRATE



Primary key: (startTime, rateType)

Foreign key: rateType references RATE (rateType)

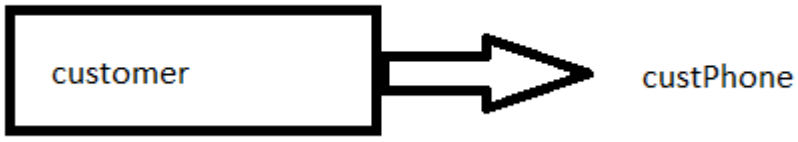
RATE



Primary key: rateType

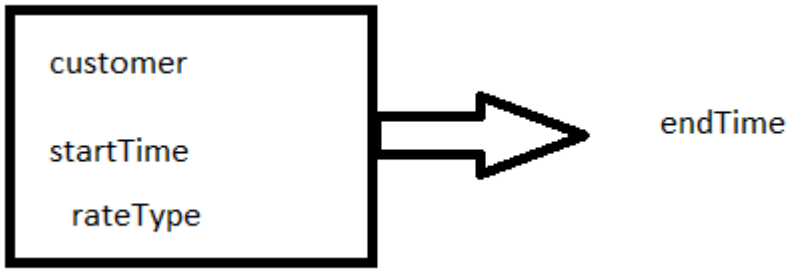
Now we can finalize the above relations by changing the composite keys in relational table CUSTOMERBOOKING.

CUSTOMER



Primary key: customer

CUSTOMERROOMBOOKING

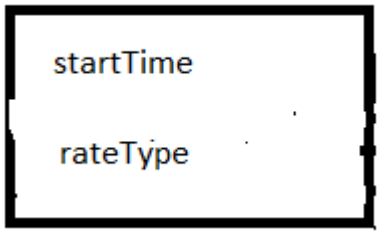


Primary key: (customer, startTime, rateType)

Foreign key: customer references CUSTOMER (customer)

Foreign key: (startTime, rateType) references ROOMRATE (startTime, rateType)

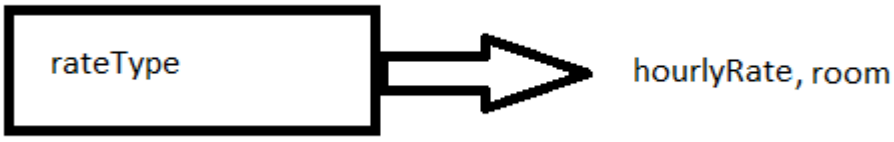
ROOMRATE



Primary key: (startTime, rateType)

Foreign key: rateType references RATE (rateType)

RATE



Primary key: rateType