CUSTOMER(id-document-number, first-name, last-name, date-of-birth, nationality)
Primary key = (id-document-number)
Candidate key = (first-name, last-name, date-of-birth)

CITY(cname, population)
Primary key = (cname)

HOTEL(cname, hotelName, capacity, stars)
Primary key = (cname, hotelName)
Foreign key = (cname) references CITY (cname)

TRAVEL-AGENCY(name, phone, fax, cname)
Primary key = (name)
Candidate key = (phone)
Foreign key = (cname) references CITY(cname)

TRAVEL-AGENT(employee-number, first-name, last-name, mobile-phone, aname)
Primary key = (employee-number)
Candidate key = (mobile-phone)
Foreign key = (aname) references TRAVEL-AGENCY(name)

BOOKING (id-document-number, cname, hotelName, booking-date, employee-number)
Primary key = (id-document-number, cname, hotelName, booking-date)
Foreign key 1 = (id-document-number) references CUSTOMER (id-document-number)
Foreign key 2 = (cname, hotelName) references HOTEL (cname, hotelName)
Foreign key 3 = (employee-number) references TRAVEL-AGENT(employee-number)

There are two ways how a multivalued attribute "facilities[1..*]" can be transformed into the relational schemas. Initially, for both ways we create a new class FACILITY(facilityItem ID). Then, we create "many-to-many" association between HOTEL class and FACILITY class. Now, we consider the following two cases.

(1)
A facility may exist without being owned by any hotel. For example, a facility like Zoo is not owned by any hotel in a moment, correct me if I am wrong, but maybe in the future … It means that "many-to-many" association between HOTEL class and FACILITY class is "many" and it is "optional" (0..*) on HOTEL side. Then, the logical design provides the following relational schemas.

FACILITY(facilityItem)
Primary key = (facilityItem)

HAS-FACILITY (cname, hotelName, facilityItem)
Primary key = (cname, hotelName, facilityItem)
Foreign key 1 = (cname, hotelName) references HOTEL (cname, hotelName)
Foreign key 2 = (facilityItem) references FACILITY(facilityItem)

(2)
A facility cannot exist without being owned by any hotel. It means that "many-to-many" association between association between HOTEL class and FACILITY class is "many" and it is "compulsory" 1..* on HOTEL side. It also means that that in the design above (1) a column facilityItem in a relational table FACILITY always has the same values as a column facilityItem in a relational table HAS-FACILITY. Therefore, a relational table FACILITY is redundant and we can drop it together with Foreign key 2 in HAS-FACILITY table.

HAS-FACILITY (cname, hotelName, facilityItem)
Primary key = (cname, hotelName, facilityItem)
Foreign key 1 = (cname, hotelName) references HOTEL (cname, hotelName)

It is also possible to consider facility as a physical facility and claim that association between HOTEL and FACILITY is "one-to-many" with compulsory 1 on HOTEL side, i.e. each physical facility belongs to only one hotel and obviously a physical facility cannot exist without a hotel. Then, the logical design provides a solution (2).