Two alternative E-R diagrams are shown in Figure 7.6.
The relational schema, including primary-key and foreign-key constraints, corresponding to the second alternative is shown below.

*customer(*customer_id*,*
name,
address)
*packet(*packet_id*,*
weight)
*place(*place_id*,*
city,
country,
address)
*sends(*sender_id*,*
receiver_id,
packet_id,
time_received,
time_sent
foreign key *sender_id* **references** *customer,*
foreign key *receiver_id* **references** *customer,*
foreign key *packet_id* **references** *packet*
)
has_gone_through(
packet_id,
place_id
foreign key *packet_id* **references** *packet,*
foreign key *place_id* **references** *place*
)

7.23 Design a database for an airline. The database must keep track of customers and their reservations, flights and their status, seat assignments on individual flights, and the schedule and routing of future flights.

Your design should include an E-R diagram, a set of relational schemas, and a list of constraints, including primary-key and foreign-key constraints.

Answer:

The E-R diagram is shown in Figure 7.7. We assume that the schedule of a flight is fixed across time, although we allow specification of on which days a flight is scheduled. For a particular instance of a flight however we record actual times of departure and arrival. In reality, schedules change with time, so the schedule and routing should be for a particular flight for specified dates, or for a specified range of dates; we ignore this complexity.

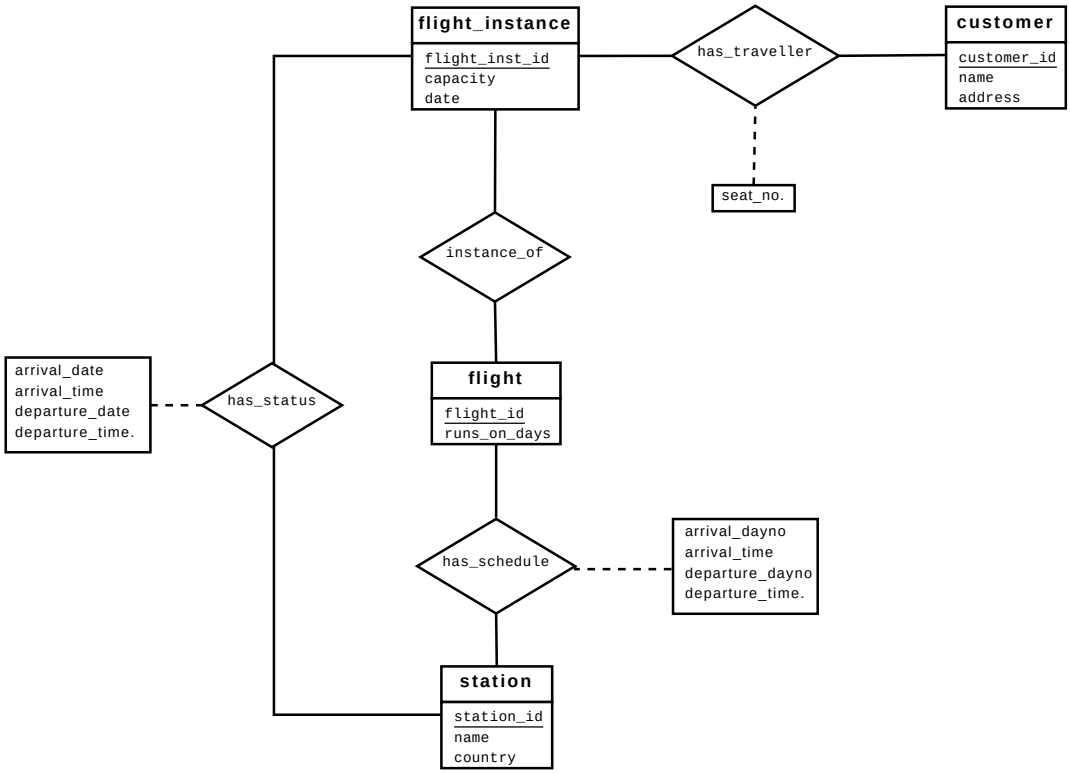

Figure 7.7 ER Diagram for Exercise 7.23

flight_instance(flight_inst_id, capacity, date)
customer(customer_id, name, address)
flight(flight_id, runs_on_days)
station(station_id, name, country)
has_traveller(
    flight_inst_id,
    customer_id,
    seat_number,
    foreign key *flight_inst_id* **references** *flight_instance,*
    foreign key *customer_id* **references** *customer*
)
instance_of(
    flight_inst_id,
    flight_id,
    foreign key *flight_inst_id* **references** *flight_instance,*
    foreign key *flight_id* **references** *flight*
)

has_schedule(
flight_id,
station_id,
order,
arrival_dayno,
arrival_time,
departure_dayno,
departure_time,
foreign key *flight_id* **references** *flight*,
foreign key *station_id* **references** *station*
)
has_status(
flight_inst_id,
station_id,
arrival_date,
arrival_time,
departure_date
departure_time
foreign key *flight_inst_id* **references** *flight_instance*,
foreign key *station_id* **references** *station*
)

7.24 In Section 7.7.3, we represented a ternary relationship (repeated in Figure 7.27a) using binary relationships, as shown in Figure 7.27b. Consider the alternative shown in Figure 7.27c. Discuss the relative merits of these two alternative representations of a ternary relationship by binary relationships.

Answer: In the model of Figure 7.27b, there can be instances where E, A, B, C, R_A, R_B and R_C cannot correspond to any instance of A, B, C and R.

The model of Figure 7.27c will not be able to represent all ternary relationships. Consider the ABC relationship set below.

A	B	C
1	2	3
4	2	7
4	8	3

If ABC is broken into three relationships sets AB, BC and AC, the three will imply that the relation (4, 2, 3) is a part of ABC.

7.25 Consider the relation schemas shown in Section 7.6, which were generated from the E-R diagram in Figure 7.15. For each schema, specify what foreign-key constraints, if any, should be created.

Answer: The foreign-key constraints are as specified below.

teaches(
    foreign key *ID* **references** *instructor,*
    foreign key *(course_id, sec_id, semester, year)* **references** *sec_course*
)

takes(
    foreign key *ID* **references** *student,*
    foreign key *(course_id, sec_id, semester, year)* **references** *sec_course*
)

prereq(
    foreign key *course_id* **references** *course,*
    foreign key *prereq_id* **references** *course*
)

advisor(
    foreign key *s_ID* **references** *student,*
    foreign key *i_id* **references** *instructor*
)

sec_course(
    foreign key *course_id* **references** *course,*
    foreign key *(sec_id, semester, year)* **references** *section*
)

sec_time_slot(
    foreign key *(course_id, sec_id, semester, year)* **references** *sec_course*
    foreign key *time_slot_id* **references** *time_slot*
)

sec_class(
    foreign key *(course_id, sec_id, semester, year)* **references** *sec_course*
    foreign key *(building, room_number)* **references** *classroom*
)

inst_dept(
    foreign key *ID* **references** *instructor*
    foreign key *dept_name* **references** *department*
)

stud_dept(
    foreign key *ID* **references** *student*
    foreign key *dept_name* **references** *department*
)

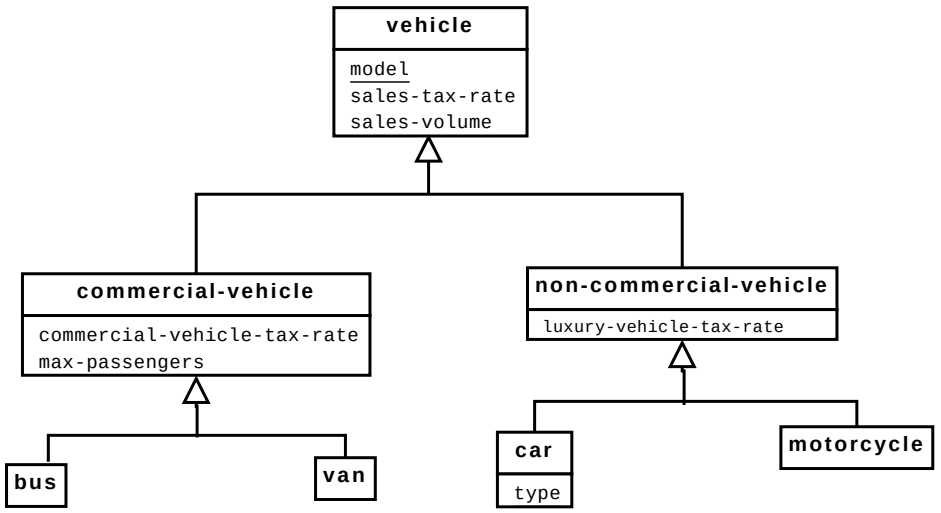

Figure 7.8 E-R diagram of motor-vehicle sales company.

course_dept(
foreign key *course_id* **references** *course*
foreign key *dept_name* **references** *department*
)

7.26 Design a generalization–specialization hierarchy for a motor vehicle sales company. The company sells motorcycles, passenger cars, vans, and buses. Justify your placement of attributes at each level of the hierarchy. Explain why they should not be placed at a higher or lower level.

Answer: Figure 7.8 gives one possible hierarchy; note that there could be many alternative solutions. The generalization–specialization hierarchy for the motor-vehicle company is given in the figure. *model*, *sales-tax-rate* and *sales-volume* are attributes necessary for all types of vehicles. Commercial vehicles attract commercial vehicle tax, and each kind of commercial vehicle has a passenger carrying capacity specified for it. Some kinds of non-commercial vehicles attract luxury vehicle tax. Cars alone can be of several types, such as sports-car, sedan, wagon etc., hence the attribute *type*.

7.27 Explain the distinction between condition-defined and user-defined constraints. Which of these constraints can the system check automatically? Explain your answer.

Answer: In a generalization–specialization hierarchy, it must be possible to decide which entities are members of which lower level entity sets. In a condition-defined design constraint, membership in the lower level entity-sets is evaluated on the basis of whether or not an entity satisfies an explicit condition or predicate.User-defined lower-level entity sets are not constrained by a membership condition; rather, entities are assigned to a given entity set by the database user.