Solutions to Exercise Sheet 5

Questions 10 and 12

a. Car rental business:

There are two tricky issues that arise in this problem.

- How should the different types of vehicles be handled? The first idea that occurs to mind is to treat the different kinds of vehicles as subclasses of a general vehicle entity. But, on a closer look, this does not seem appropriate. First of all, there is nothing different in the way the rental company deals with the different kinds of vehicles. Renting the vehicles out and processing returns are common to all the kinds, and there is not much else that needs to be done differently for the vehicle classes. On the other hand, the vehicle classes have attributes such as the rental rate and substitutability. Hence, we need make vehicle classes themselves as a new form of entity. Note the similarity with how to treat hotel rooms (with roomtype as an entity).

- We have invented a new entity called “rental contract” even though it wasn’t mentioned in the textual requirements. It is necessitated by thinking about car rentals over a period of time, and the process requirements that the customers first ask for a vehicle class (perhaps in advance) and get assigned a specific vehicle. The for_v and for_c relationships talk about a specific vehicle being assigned to a rental contract or only a vehicle class.

We choose to make employee a weak entity because his/her identity can be made to depend on that of the commercial customer.

Relational table schemas: Rental contracts have many relationships where the maximum multiplicity is 1. All those relationships can be absorbed into a single table.

```plaintext
vehicle(vid, license plate)
vehicle_class(classid, name, rental_rate)
substitutable_by(class1, class2)
staff(staffid, firstname, lastname)
customer(cid, commercial)
commercial_customer(cid, name, address, discount°)
private_customer(cid, firstname, lastname, address)
employee(cid, firstname, lastname)
rental_contract(contractid, cid, classid, vid°, staffid°)
```
b. United Nations:

Relational table schemas:

geographic_unit(unitid, name, continent, population, area, capital, independent)
borders(unitid, unitid)
ethnic_group(unitid, group, population, percentage)
alliance(allianceid, name, type)
alliance_member(unitid, allianceid)
territory(unitid, controlling_country)
occupied_country(unitid, occupying_country)

Note that we have added boolean attribute independent to geographic units, which captures whether a unit is an independent country. We do not need a separate table for independent countries. However, territories and occupied countries need separate tables because of the additional information required for them.