Solutions to Practice Exercises

4.1 Query:

```sql
create table loan
(loan_number char(10),
 branch_name char(15),
 amount integer,
 primary key (loan_number),
 foreign key (branch_name) references branch)
```

```sql
create table borrower
(customer_name char(20),
 loan_number char(10),
 primary key (customer_name, loan_number),
 foreign key (customer_name) references customer,
 foreign key (loan_number) references loan)
```

Declaring the pair `customer_name, loan_number` of relation `borrower` as primary key ensures that the relation does not contain duplicates.

4.2 Query:
create table employee
    (person_name    char(20),
     street         char(30),
     city           char(30),
     primary key (person_name)
    )

create table works
    (person_name    char(20),
     company_name  char(15),
     salary        integer,
     primary key (person_name),
     foreign key (person_name) references employee,
     foreign key (company_name) references company
    )

create table company
    (company_name  char(15),
     city          char(30),
     primary key (company_name)
    )

create table manages
    (person_name     char(20),
     manager_name   char(20),
     primary key (person_name),
     foreign key (person_name) references employee,
     foreign key (manager_name) references employee
    )

Note that alternative datatypes are possible. Other choices for not null attributes may be acceptable.

a. check condition for the works table:

   check((employee_name, company_name) in
   (select e.employee_name, c.company_name
   from employee e, company c
   where e.city = c.city
   ))

b. check condition for the works table:
check(
  salary < all
  (select manager_salary
   from (select manager_name, manages.employee_name as emp_name, salary as manager_salary
   from works, manages
   where works.employee_name = manages.manager_name)
   where employee_name = emp_name
  )
)

The solution is slightly complicated because of the fact that inside the select expression’s scope, the outer works relation into which the insertion is being performed is inaccessible. Hence the renaming of the employee_name attribute to emp_name. Under these circumstances, it is more natural to use assertions.

4.3 The tuples of all employees of the manager, at all levels, get deleted as well! This happens in a series of steps. The initial deletion will trigger deletion of all the tuples corresponding to direct employees of the manager. These deletions will in turn cause deletions of second level employee tuples, and so on, till all direct and indirect employee tuples are deleted.

4.4 The assertion’s name is arbitrary. We have chosen the name perry. Note that since the assertion applies only to the Perryridge branch we must restrict attention to only the Perryridge tuple of the branch relation rather than writing a constraint on the entire relation.

create assertion perry check
  (not exists (select *
    from branch
    where branch_name = 'Perryridge' and
    assets ≠ (select sum(amount)
      from loan
      where branch_name = 'Perryridge') )))

4.5 Writing queries in SQL is typically much easier than coding the same queries in a general-purpose programming language. However not all kinds of queries can be written in SQL. Also nondeclarative actions such as printing a report, interacting with a user, or sending the results of a query to a graphical user interface cannot be done from within SQL. Under circumstances in which we want the best of both worlds, we can choose embedded SQL or dynamic SQL, rather than using SQL alone or using only a general-purpose programming language.

Embedded SQL has the advantage of programs being less complicated since it avoids the clutter of the ODBC or JDBC function calls, but requires a specialized preprocessor.