Solution

Q1.

(1) False

(2) False

(3) True

(4) True

(5) False

(6) True

(7) False

(8) False

(9) True

(10) True

Q2.

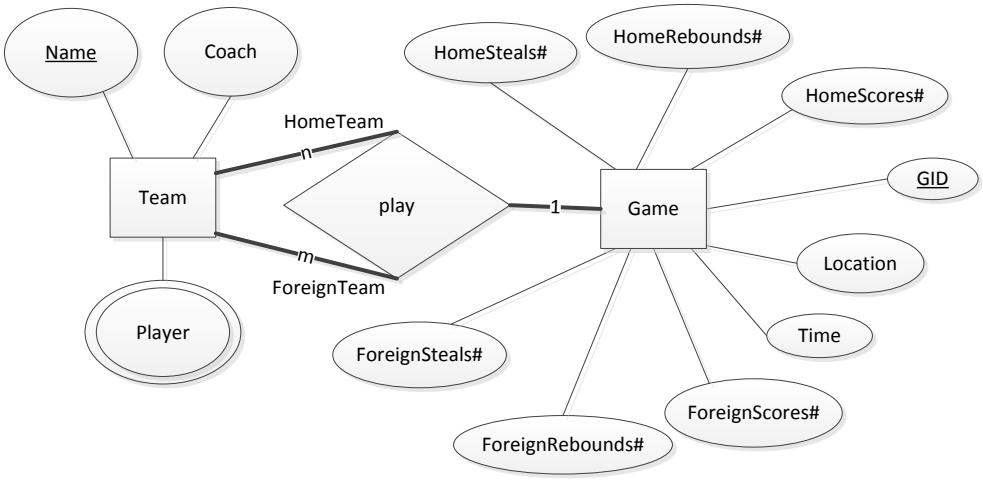
(a)

```
interface Person {
        attribute string name;
        attribute string birth_date;
        relationship Person motherOf
                inverse Person::childrenOfFemale
        relationship Person fatherOf
                inverse Person::childrenOfMale
        relationship Set<Person> children
                inverse Person::parentsOf
        relationship Set<Person> childrenOfFemale
                inverse Person::motherOf
        relationship Set<Person> childrenOfMale
                inverse Person::fatherOf
        relationship Set<Person> parentsOf
                inverse Person::children
```

}

(b)

i)



ii)

Q3.

(a)

Key constraint: candidate key values must be unique for every relation instance.

Entity integrity: an attribute that is part of a primary key cannot be NULL.

Referential integrity: The third kind has to do with “foreign keys”.

i) Check all three constraints.

ii) Referential constraints.

iii) None

(b) No. Because $(AC)^+ = \{A, B, C, D, H\}$ and $G \notin (AC)^+$

(c) 1st NF. Because AB is the only key and C is partially dependent on A.

(d) $F_m = \{Y \rightarrow X, Z \rightarrow Y, Z \rightarrow W\}$.

Step 1: $F_m = F$

Step 2: Replace $Z \rightarrow XYW$ with $Z \rightarrow X, Z \rightarrow Y, Z \rightarrow W$

Step 3: Remove $Z \rightarrow X$ since it could be obtained by $Z \rightarrow Y$ and $Y \rightarrow X$.

Step 4: No more FD could be removed, done

Q4.

(a)

i) A primary index is an index on a set of fields that includes the primary key. It is guaranteed not to contain duplicates. A secondary index is an index that is not a primary index. It can have duplicates.

ii) A dense index has at least one data entry search key value that appears in a record in the indexed file. A sparse index contains an entry for each page of records in a data file. It must be clustered. Thus, we can have at most one sparse index on a data file. A sparse index is typically much smaller than a dense index.

iii) A clustered index is one in which the ordering of data entries is the same as the ordering of data records. We can have at most one clustered index on a data file. An unclustered index is an index that is not clustered. We can have several unclustered indexes on a data file.

(b)

No. it is not a good idea, because it does not provide useful indexing information for retrieval and is useless.

(c)

No, in this case, the index is unclustered, each qualifying data entries could contain an rid that points to a distinct data page, leading to as many data page I/O as the number of data entries that match the range query. In this case, if the number of data entries is larger than the number of data blocks then using index is worse than file scan.

Q5.

(a) See the lecture notes for such an example.

(b)

i) A straightforward way to do this is to construct a serial schedule using the two-phase locking protocol. I just show the key part of schedule.

T1	T2
write_lock(A)	
⋮	
write_lock(B)	
⋮	
unlock(A)	
unlock(B)	
	write_lock(B)
	⋮
	write_lock(A)
	⋮
	unlock(B)
	unlock(A)

ii)

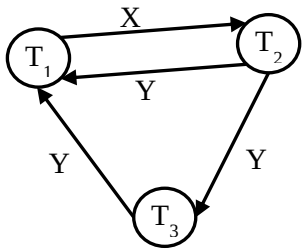
No, it is impossible to have a non-serial but serializable schedule. Consider that the last operation in T1 is WRITE(B), and the last operation in T2 is WRITE(A). Meanwhile, T1 starts with READ(A), and T2 starts with READ(B). Therefore, in any serializable schedule of T1 and T2, either READ(A) in T1 should be after WRITE(B) in T2, or READ(B) in T2 should be after WRITE(B) in T1.

iii)

Yes, It is very straightforward, I just show key part of the schedule

T1	T2
⋮	⋮
write_lock(A)	
WRITE(A)	
	write_lock(B)
	WRITE(B)
read_lock(B)	
waiting for B*	read_lock(A)
waiting for B*	**waiting for A**
⋮	⋮

(c)



The schedule is not serializable, because there is a cycle in its schedule graph.