INFDEV026B Tentamen OP2 2018
Solution

Francesco Di Giacomo, Ahmad Omar

Question I: Database normalization (5 pts.)

- The table is in 1NF. All values are atomic. It is not in the 2NF because of the partial dependency on part of the key. For example company_name is only dependent on company_code
- Normalization:
  - The normalized tables in 2NF
    Table **Company**: company_code | company_name
    Table **Flight**: flight_number | departure | destination | distance | price
    Table **Flight-Company**: flight_number | company_code
  - The normalized tables in BCNF
    Table **Company**: company_code | company_name
    Table **Flight**: flight_number | departure | destination | distance
    Table **Flight-Company**: flight_number | company_code
    Table **Cost**: departure | destination | price

Question II - Transactions (5 pts.)

- Table 2 conflicts and 2PL equivalent execution
  - Possible conflicts are: read-write conflict on B between T2 and T3; write-read conflict on C between T2 and T1
  - 2PL equivalent execution

T1	T2	T3
S(A)		
R(A)		
	S(B)	
	R(B)	
		S(B)
		R(B)
		Waitlock(B)
	X(C)	
	W(C)	
Waitlock(C)		
	Commit	
	Unlock(C)	
		X(B)
		W(B)
		S(A)
		R(A)
		Commit
		Unlock(B)
S(C)		
R(C)		
Commit		
Unlock(A)		
unlock(C)		

- Table 3 wait graph and deadlocks

T1	T2	T3	T4
S(A)			
R(A)			
	S(B)		
	R(B)		
		S(B)	
		R(B)	
		Waitlock(B)	
			Waitlock(B)
	Waitlock(A)		
Waitlock(B)			

Wait graph:
T3 -> T2
T4 -> T2
T2 -> T1
T1 -> T2
Aborting T2 could break the deadlock

Question III: Map-Reduce (5 pts.)

```
//TODO1: Complete the implementation of Map
for (int i = 0; i < collection.Count(); i++){
        result[i] = transformation(collection.ElementAt(i));
  }

//TODO2: complete with the initialization of the accumulator
T2 result = init;

//TODO3: complete the body of the for-loop of Reduce
result = operation(result, collection.ElementAt(i));

//TODO4: Replace with the correct lambda
CourseTable.Map(c => new { Name = c.Name, Duration = c.Duration }).ToList()

//TODO5: complete the body of the lambda to implement the query
 if (a.Month >= 9)
       l.Add(a);

//TODO6: Complete with the correct call to Join
CourseAssignmentTable, t => t.Item1.Code == t.Item2.CourseCode

//TODO7: Complete with the correct call to Join
AssignmentTable, t => t.Item1.Item2.AssignmentCode == t.Item2.Code
```

Question IV: Graph databases (5 pts.)

```
create(c1:Course{code:'dev1', name:'development 1',duration:2}),
                  (a1:Assignment{day: 10, month: 2, code:'a1', type: 'practicum'}),
                  (e1:Exam{maxscore:10, duration:150, chance:2,code:'e1'}),
                  (a1)-[g1:GivenAt]->(c1),
                  (e1)-[t1:Test]->(c1)
return g1,t1;
```

The queries are the following:

```
//Query1
match (e:Exam{chance:2})
return e;

OR

match (e:Exam) where e.chance == 2
return e

//Query2
match (a:Assignment)-[g:GivenAt]->(c:Course{code:"dev1"})
where a.month < 10
return a;

OR

match (a:Assignment)-[g:GivenAt]->(c:Course{})
where a.month < 10  AND c.code == "dev1"
return a;
```