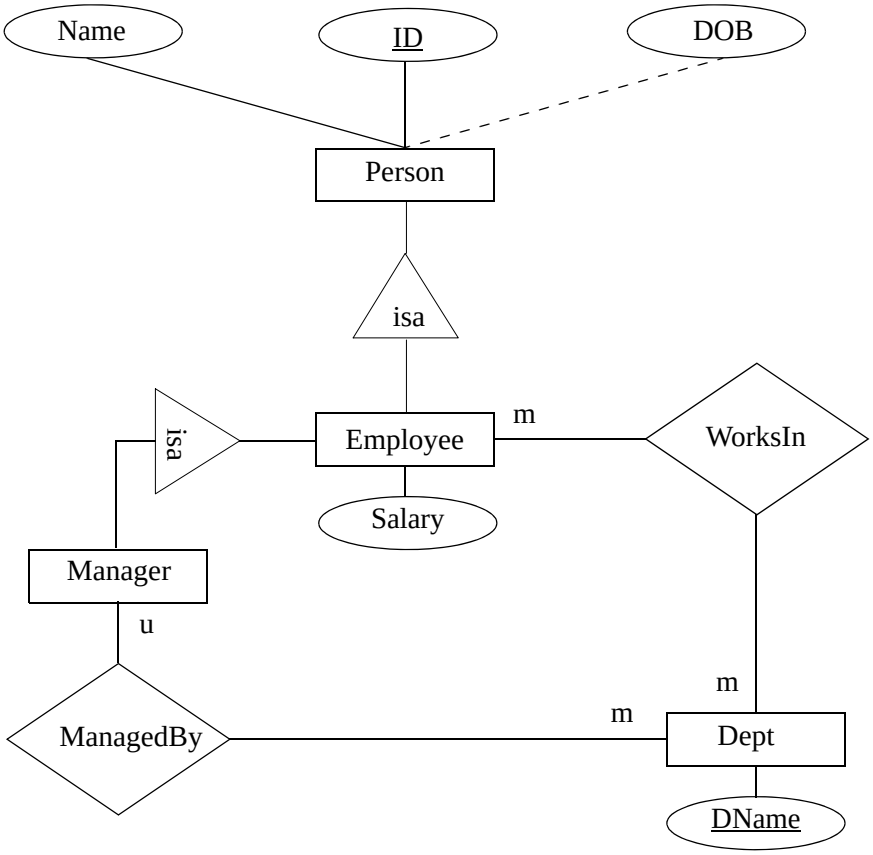
Sample Test 1 and solutions

Problems are not necessarily in increasing order of difficulty. Although exact compliance with SQL, JDBC, Java, OQL, and XML standards is not required, your answers should be thoughtful. For example, to register a JDBC-driver, a statement such as "Register(MyDriver);" can be used.

Scores / problem #	1	2	3	4	5	6	Total
Total	15	20	15	20	15	15	100
Your Score							

Solutions are in red. *Some comments are shown in blue italics.*

1. Consider the following ER diagram and the proposed relational scheme.



The proposed relational scheme:

- Person (Name, ID, DOB)
- Emp (ID, Dname, Salary)
- Dept (Dname, ManagerID)

The proposed relational scheme has the problem that it does not capture the whole semantics of the ER diagram. For the relational scheme, do the following.

A. Designate keys.

B. Designate foreign keys.

C. Parts A and B may fix some problems with the relational scheme when compared to the ER diagram. But some problems may still remain in the relational scheme. List all such problems. You can express your answers informally.

A. Designate Keys as follows:

- Person (Name, ID, DOB)
- Emp (ID, Dname, Salary)
- Dept (Dname, ManagerID)

B. Foreign keys:

- Emp.ID references Person.ID

C. Leftover problems:

- We would like Dept.ManagerID to reference Emp.ID, but latter is not a key in Emp. Instead, incorporate this by a constraint, a query, that Dept.ManagerID occurs in Emp.ID column.
- We would like Emp.Dname to reference Dept.Dname, but latter is not a key in Dept. Instead, incorporate this by a constraint, a query, that Emp.Dname occurs as Dept.Dname.
- Person.DOB is nullable.

2. These problems are on the University database of Project 1. The relation schemes are listed below for your reference.

- Person (Name, ID, Address, DOB)
- Instructor (InstructorID, Rank, Salary)
- Student (StudentID, Classification, GPA, MentorID, CreditHours)
- Course (CourseCode, CourseName, PreReq)
- Offering (CourseCode, SectionNo, InstructorID)
- Enrollment (CourseCode, SectionNo, StudentID, Grade)

Express the following as SQL queries.

A. Give name and date of birth of the youngest student who is also an instructor.

B. List StudentID of students who are not taking CS311.

A.
```
select p.Name, p.DOB
from Person p, Student s, Instructor i
where p.ID = s. StudentID
   and s.StudentID = i.InstructorID
   and p.DOB in ( select max(p1.DOB)
                  from Person p1, Student s1, Instructor i1
                  where p1.ID = s1. StudentID
                     and s1.StudentID = i1.InstructorID)
```

B.
```
select s.StudentID
from Student s
where s.StudentID not in ( select e.StudentID
                           from Enrolment e
                           where e.CourseCode = 'CS311')
```

3. Give example of two non-empty relations r(A,B) and s(B,C) and a scenario, such that the number of block accesses needed to compute the natural join of the two relations, is less than the total number of blocks occupied by the two relations.

Note that you cannot even read the two relations completely. One scenario is the following the number of tuples in r are one-10th of the number of blocks in s, and s is organized as a B+-tree of height 3, and the number of succeeding matches are small. In such a case, all of r will be scanned, only a few blocks of s will be fetched.

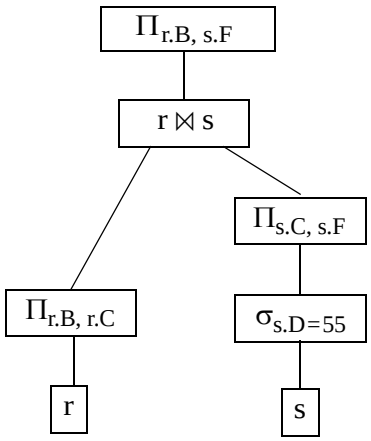
4. You are given relations r(A, B, C, E) and s(C, D, F, G). Note that the keys have been underscored. Assume that r occupies 1000 blocks and s occupies 2000 blocks on the disk. Assume all attributes occupy the same number of bytes. Also assume that every block is full and contains 20 tuples. You are given up to 12 buffers to process the query. Draw an optimal expression tree and give estimation of optimal execution of the tree. Assume that D = 55 is satisfied by 1% of the tuples of s. Assume that the output consists of 50 blocks.

```
select x.B, y.F
from r x, s y
where x.C = y.C and y.D = 55
```

See expression tree below. Apply the selection and projection on r first. *Note 1% selectivity and 50% contraction in the size of each tuple.* The size of s reduces from 2000 blocks to 2000 * 0.01 * 0.50 = 10 blocks. This costs 2000 + 10 = 2010 block accesses.

Next perform the natural join and the projection operator in tandem. Make s the outer relation, fitting all of it in 10 buffers, use 1 buffer to input r, and 1 buffer for output.

The cost here is 10 + 1000 + 50 = 1060 block accesses. Total cost is 2010 + 1060 = 3070 block accesses.



Think: What if we have only 3 buffers to process the query? Would the above strategy change?

5. A binary operator "op" is said to be *monotonic* if $r_1 \subseteq r_2$ and $s_1 \subseteq s_2$ imply $(r_1 \text{ op } s_1) \subseteq (r_2 \text{ op } s_2)$. Informally, it means that contents of the result increase (do not decrease) when the contents of operands are increased.
A. Which ones of $r \cup s$, $r \cap s$, $r - s$, $r \times s$, and $r \lozenge s$ are monotonic and which are not. (At least one is and at least one is not.)
B. Use an example to prove that one of the above is not monotonic. (No partial credit even if A is wrong.)

A. All but r - s are monotonic.

B. Suppose $r_1 = r_2 = \{1\}$, and $s_1 = \{2\}$ and $s_2 = \{1,2\}$. Then $r_1 - s_1 = \{1\}$ and $r_2 - s_2$ is empty. Clearly, $r_1 \subseteq r_2$ and $s_1 \subseteq s_2$ hold, but $(r_1 - s_1) \subseteq (r_2 - s_2)$ does not.

6. You are given the following relations. The Salary attribute represents the monthly salary. Every month an employee can also earn several bonuses that are stored in the Bonus relation.

- Emp(Name, Dept, Salary)
- Bonus(Name, TransactionID, BAmount)
- Wages(Name, WAmount)

The Wages relation is initially empty and it needs to be populated. WAmount of an employee is the sum of his/her Salary and total of his/her bonuses (BAmount), with the understanding that the grand total cannot exceed 120% of the salary. (In other words, an employee can earn a bonus that is up to 20% of his/her salary.) Assume that every employee appears in the Emp relation as well as the Bonus relation.

In plain English describe a JDBC program to populate the Wages relation at the end of a month. Assume very large relations and keep efficiency of your JDBC program in mind.

Establish a connection. Through this connection execute a query to join Emp and Bonus relations, retrieving Name, Salary, and sum of bonuses for each employee. Scan tuples of the result-set of the query and compute the wages by adding the salary and total of bonus limiting the wages to 120% of the salary-value. Through the same connection, execute a prepared insert statement on Wages relation to add names and wages of employees. Close both statements, commit the transaction, and close the connection.