Assignments #5– Sample Solutions

8.16 - Specify the following queries on the COMPANY relational database schema shown in Figure 3.5, using the relational operators discussed in this chapter. Also show the result of each query as it would apply to the database state of Figure 3.6.

(a) Retrieve the names of employees in department 5 who work more than 10 hours per week on the 'ProductX' project.

(b) List the names of employees who have a dependent with the same first name as themselves.

(c) Find the names of employees that are directly supervised by 'Franklin Wong'.

(d) For each project, list the project name and the total hours per week (by all employees) spent on that project.

(e) Retrieve the names of employees who work on every project.

(f) Retrieve the names of employees who do not work on any project.

(g) For each department, retrieve the department name, and the average salary of employees working in that department.

(h) Retrieve the average salary of all female employees.

(i) Find the names and addresses of employees who work on at least one project located in Houston but whose department has no location in Houston.

(j) List the last names of department managers who have no dependents.

Answers:

In the relational algebra, as in other languages, it is possible to specify the same query in multiple ways. We give one possible solution for each query. We use the symbol s for SELECT, P for PROJECT, J for EQUIJOIN, * for NATURAL JOIN, and f for FUNCTION.

(a) EMP_W_X <-- ( s PNAME='ProductX' (PROJECT)) J (PNUMBER),(PNO) (WORKS_ON) EMP_WORK_10 <-- (EMPLOYEE) J (SSN),(ESSN) ( s HOURS>10 (EMP_W_X))
RESULT <- P LNAME,FNAME ( s DNO=5 (EMP_WORK_10))

Result:
LNAME FNAME
Smith John
English Joyce

(b) E <- (EMPLOYEE) J (SSN,FNAME),(ESSN,DEPENDENT_NAME) (DEPENDENT)
R <- P LNAME,FNAME (E)

Result (empty):
LNAME FNAME

(c) WONG_SSN <- P SSN ( s FNAME='Franklin' AND
LNAME='Wong' (EMPLOYEE))
WONG_EMPS <- (EMPLOYEE) J (SUPERSSN),(SSN) (WONG_SSN)
RESULT <- P LNAME,FNAME (WONG_EMPS)

Result:
LNAME FNAME
Smith John
Narayan Ramesh
English Joyce

(d) PROJ_HOURS(PNO,TOT_HRS) <- PNO f SUM HOURS (WORKS_ON)
RESULT <- P PNAME,TOT_HRS ( (PROJ_HOURS) J (PNO),(PNUMBER)
(PROJECT) )

Result:
PNAME TOT_HRS
ProductX 52.5
ProductY 37.5
ProductZ 50.0
Computerization 55.0
Reorganization 25.0
Newbenefits 55.0

(e) PROJ_EMPS(PNO,SSN) <- P PNO,ESSN (WORKS_ON)
ALL_PROJS(PNO) <- P PNUMBER (PROJECT)
EMPS_ALL_PROJS <- PROJ_EMPS -: ALLPROJS (* DIVISION operation *)
RESULT <- P LNAME,FNAME (EMPLOYEE * EMP_ALL_PROJS)

Result (empty):
LNAME FNAME

(f) ALL_EMPS <- P SSN (EMPLOYEE)
WORKING_EMPS(SSN) <- P ESSN (WORKS_ON)
NON_WORKING_EMPS <- ALL_EMPS - WORKING_EMPS (* DIFFERENCE *)
RESULT <- P LNAME,FNAME (EMPLOYEE * NON_WORKING_EMPS)
Result (empty):
(g) \text{DEPT\_AVG\_SALS(DNUMBER,AVG\_SAL)} \leftarrow \text{DNO } f \text{ AVG SALARY (EMPLOYEE)}
\text{RESULT} \leftarrow \text{P DNUMBER,AVG\_SAL} ( \text{DEPT\_AVG\_SALS } * \text{DEPARTMENT })
\text{Result:}
\text{DNUMBER } \text{AVG\_SAL}
\begin{align*}
\text{Research} & : 33250 \\
\text{Administration} & : 31000 \\
\text{Headquarters} & : 55000
\end{align*}

(h) \text{RESULT(AVG\_F\_SAL)} \leftarrow f \text{ AVG SALARY ( s } \text{SEX='F' (EMPLOYEE) )}
\text{Result:}
\text{AVG\_F\_SAL}
31000

(i) \text{E\_P\_HOU(SSN)} \leftarrow
\text{P ESSN (WORKS\_ON J(PNO),(PNUMBER) ( s E\_LOCATION='Houston' (PROJECT)))}
\text{D\_NO\_HOU} \leftarrow
\text{P DNUMBER (DEPARTMENT)} - \text{P DNUMBER ( s D\_LOCATION='Houston' (DEPARTMENT))}
\text{E\_D\_NO\_HOU} \leftarrow \text{P SSN (EMPLOYEE J(PNO),(DNUMBER) (D\_NO\_HOU))}
\text{RESULT\_EMPS} \leftarrow \text{E\_P\_HOU - E\_D\_NO\_HOU} (* \text{this is set DIFFERENCE} *)
\text{RESULT} \leftarrow \text{P LNAME,FNAME,ADDRESS (EMPLOYEE } * \text{RESULT\_EMPS)}
\text{Result:}
\text{LNAME } \text{FNAME } \text{ADDRESS}
Wallace Jennifer 291 Berry, Bellaire, TX

(j) \text{DEPT\_MANAGERS(SSN)} \leftarrow \text{P MGRSSN (DEPARTMENT)}
\text{EMPS\_WITH\_DEPENDENTS(SSN)} \leftarrow \text{P ESSN (DEPENDENT)}
\text{RESULT\_EMPS} \leftarrow \text{DEPT\_MANAGERS - EMPS\_WITH\_DEPENDENTS}
\text{RESULT} \leftarrow \text{P LNAME,FNAME (EMPLOYEE } * \text{RESULT\_EMPS)}
\text{Result:}
\text{LNAME } \text{FNAME}
Borg James

8.24 - Specify queries (a), (b), (c), (e), (f), (i), and (j) of Exercise 6.16 in both tuple and domain relational calculus.

\textbf{Answers:}

(a) Retrieve the names of employees in department 5 who work more than 10 hours per week on the 'ProductX' project.
\text{Tuple relational Calculus:}
\{ \text{e.LNAME, e.FNAME | EMPLOYEE(e) AND e.DNO=5 AND (EXISTS p) (EXISTS w)} \}
\{ \text{WORKS\_ON(w) AND PROJECT(p) AND e.SSN=w.ESSN AND} \}
w.PNO=p.PNUMBER AND p.PNAME='ProductX' AND w.HOURS>10 } }

Domain relational Calculus:
{ qs | EMPLOYEE(qrstuvwxyz) AND z=5 AND (EXISTS a) (EXISTS b) (EXISTS e) (EXISTS f) (EXISTS g) ( WORKS_ON(efg) AND PROJECT(abcd) AND t=e AND f=b AND a='ProductX' AND g>10 ) } }

(b) List the names of employees who have a dependent with the same first name as themselves.

Tuple relational Calculus:
{ e.LNAME, e.FNAME | EMPLOYEE(e) AND (EXISTS d) ( DEPENDENT(d) AND e.SSN=d.ESSN AND e.FNAME=d.DEPENDENT_NAME ) } }

Domain relational Calculus:
{ qs | (EXISTS t) (EXISTS a) (EXISTS b) ( EMPLOYEE(qrstuvwxyz) AND DEPENDENT(abcd) AND a=t AND b=q ) } }

(c) Find the names of employees that are directly supervised by 'Franklin Wong'.

Tuple relational Calculus:
{ e.LNAME, e.FNAME | EMPLOYEE(e) AND (EXISTS s) ( EMPLOYEE(s) AND s.FNAME='Franklin' AND s.LNAME='Wong' AND e.SUPERSSN=s.SSN ) } }

Domain relational Calculus:
{ qs | (EXISTS y) (EXISTS a) (EXISTS c) (EXISTS d) ( EMPLOYEE(qrstuvwxyz) AND EMPLOYEE(abcddefghij) AND a='Franklin' AND c='Wong' AND y=d ) } }

(e) Retrieve the names of employees who work on every project.

Tuple relational Calculus:
{ e.LNAME, e.FNAME | EMPLOYEE(e) AND (FORALL p) ( NOT(PROJECT(p)) OR (EXISTS w) ( WORKS_ON(w) AND p.PNUMBER=w.PNO AND w.ESSN=e.SSN ) ) } }

Domain relational Calculus:
{ qs | (EXISTS t) ( EMPLOYEE(qrstuvwxyz) AND (FORALL b) ( NOT(PROJECT(abcd)) OR (EXISTS e) (EXISTS f) (WORKS_ON(efg) AND e=t AND f=b ) ) ) } }

(f) Retrieve the names of employees who do not work on any project.

Tuple relational Calculus:
{ e.LNAME, e.FNAME | EMPLOYEE(e) AND NOT(EXISTS w) ( WORKS_ON(w) AND w.ESSN=e.SSN ) } }

Domain relational Calculus:
{ qs | (EXISTS t) ( EMPLOYEE(qrstuvwxyz) AND NOT(EXISTS a) ( WORKS_ON(abc) AND a=t ) ) } }

(i) Find the names and addresses of employees who work on at least one project located in Houston but whose department has no location in Houston.

Tuple relational Calculus:


\{ e.LNAME, e.FNAME, e.ADDRESS | EMPLOYEE(e) AND (EXISTS p) (EXISTS w) (WORKS_ON(w) AND PROJECT(p) AND e.SSN=w.ESSN AND w.PNO=p.PNUMBER AND p.PLOCATION='Houston' AND NOT(EXISTS l) ( DEPT_LOCATIONS(l) AND e.DNO=l.DNUMBER AND l.DLOCATION='Houston' ) ) \}

Domain relational Calculus:
\{ qsv | (EXISTS t) (EXISTS z) ( EMPLOYEE(qrstuvwxyz) AND (EXISTS b) (EXISTS c) (EXISTS e) (EXISTS f) ( WORKS_ON(efg) AND PROJECT(abcd) AND t=e AND f=b AND c='Houston' AND NOT(EXISTS h) NOT(EXISTS i) ( DEPT_LOCATIONS(hi) AND z=h AND i='Houston' ) ) ) \}

(j) List the last names of department managers who have no dependents.

Tuple relational Calculus:
\{ e.LNAME | EMPLOYEE(e) AND (EXISTS d) ( DEPARTMENT(d) AND e.SSN=d.MGRSSN AND NOT(EXISTS x) (DEPENDENT(x) AND e.SSN=x.ESSN) ) \}

Domain relational Calculus:
\{ s | (EXISTS t) ( EMPLOYEE(qrstuvwxyz) AND (EXISTS c) ( DEPARTMENT(abcd) AND t=c AND NOT(EXISTS e) (DEPENDENT(efghi) AND e=t) ) \}

8.26 - Specify queries c, d, and e of Exercise 6.18 in both tuple and domain relational calculus.

Answers:

(c) For each section taught by professor King, retrieve the course number, semester, year, and number of students who took the section.
This query cannot be done in basic relational calculus as it requires a COUNT function.

(d) Retrieve the name and transcript of each senior student (Class=5) majoring in COSC. Transcript includes course name, course number, credit hours, semester, year, and grade for each course completed by the student.

Tuple relational Calculus:
\{ s.Name, c.CourseName, c.CourseNumber, c.CreditHours, t.Semester, t.Year, eeeeeeeeGrade | STUDENT(s) AND COURSE(c) AND SECTION(t) AND GRADE_REPORT(g) AND s.Class=5 AND s.Major='COSC' AND s.StudentNumber=g.StudentNumber AND g.SectionIdentifier=t.SectionIdentifier AND t.CourseNumber=c.CourseNumber \}

Domain relational Calculus:
\{ aefklp | (EXISTS b) (EXISTS c) (EXISTS d) (EXISTS n) (EXISTS o) (EXISTS j) (EXISTS i) (STUDENT(abcd) AND COURSE(efgh) AND SECTION(ijklm) AND GRADE_REPORT(nop) AND c=5 AND d='COSC' AND b=n AND i=o AND j=f) \}
(e) Retrieve the names and major departments of all straight A students (students who have a grade of A in all their courses).

Tuple relational Calculus:
\{ s.Name, s.Major | STUDENT(s) AND (FORALL g) ( NOT(GRADE_REPORT(g)) OR NOT(s.StudentNumber=g.StudentNumber) OR g.Grade='A' ) \}

Domain relational Calculus:
\{ ad | (EXISTS b) ( STUDENT(abcd) AND (FORALL e) (FORALL g) ( NOT(GRADE_REPORT(efg)) OR NOT(b=e) OR g='A' ) ) \}

6.5 - Consider the database shown in Figure 1.2, whose schema is shown in Figure 2.1. What are the referential integrity constraints that should hold on the schema? Write appropriate SQL DDL statements to define the database.

Answer:

The following referential integrity constraints should hold (we use the notation: R.(A1, ..., An) --> S.(B1, ..., Bn)
to represent a foreign key from the attributes A1, ..., An of R (the referencing relation) to S (the referenced relation)):
PREREQUISITE.(CourseNumber) --> COURSE.(CourseNumber)
PREREQUISITE.(PrerequisiteNumber) --> COURSE.(CourseNumber)
SECTION.(CourseNumber) --> COURSE.(CourseNumber)
GRADE_REPORT.(StudentNumber) --> STUDENT.(StudentNumber)
GRADE_REPORT.(SectionIdentifier) --> SECTION.(SectionIdentifier)

One possible set of CREATE TABLE statements to define the database is given below.

CREATE TABLE STUDENT ( Name VARCHAR(30) NOT NULL,
                        StudentNumber INTEGER NOT NULL,
                        Class CHAR NOT NULL,
                        Major CHAR(4),
                        PRIMARY KEY (StudentNumber) );
CREATE TABLE COURSE ( CourseName VARCHAR(30) NOT NULL,
                      CourseNumber CHAR(8) NOT NULL,
                      CreditHours INTEGER,
                      Department CHAR(4),
                      PRIMARY KEY (CourseNumber),
                      UNIQUE (CourseName) );
CREATE TABLE PREREQUISITE ( CourseNumber CHAR(8) NOT NULL,
                            PrerequisiteNumber CHAR(8) NOT NULL,
                            PRIMARY KEY (CourseNumber, PrerequisiteNumber),
                            FOREIGN KEY (CourseNumber) REFERENCES COURSE (CourseNumber),
                            FOREIGN KEY (PrerequisiteNumber) REFERENCES COURSE (CourseNumber) );
CREATE TABLE SECTION ( SectionIdentifier INTEGER NOT NULL,
                       CourseNumber CHAR(8) NOT NULL,
                       Semester VARCHAR(6) NOT NULL,
                       Year CHAR(4) NOT NULL,
                       Instructor VARCHAR(15),
                       PRIMARY KEY (SectionIdentifier),
                       FOREIGN KEY (CourseNumber) REFERENCES COURSE (CourseNumber) ) ;
CREATE TABLE GRADE_REPORT ( StudentNumber INTEGER NOT NULL,
SectionIdentifier INTEGER NOT NULL,
Grade CHAR,
PRIMARY KEY (StudentNumber, SectionIdentifier),
FOREIGN KEY (StudentNumber) REFERENCES
STUDENT (StudentNumber),
FOREIGN KEY (SectionIdentifier) REFERENCES
SECTION (SectionIdentifier) );