

6.19 Specify the following queries on the database schema shown in Figure 6.5 using the relational operators discussed in this chapter. Also show the result of each query if applied to the database of Figure 6.6.

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 σ for SELECT, Π for PROJECT, ϑ for EQUIJOIN, $*$ for NATURAL JOIN, and f for FUNCTION.

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

```
EMP_W_X <-- ( $\sigma$  PNAME='ProductX'(PROJECT))  $\vartheta$  (PNUMBER),(PNO)(WORKS_ON)
EMP_WORK_10 <-- (EMPLOYEE)  $\vartheta$  (SSN),(ESSN) ( $\sigma$  HOURS>10(EMP_W_X))
RESULT <--  $\Pi$  LNAME,FNAME ( $\sigma$  DNO=5(EMP_WORK_10))
```

Result:

LNAME	FNAME
Smith	John
English	Joyce

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

```
E <-- (EMPLOYEE)  $\vartheta$  (SSN,FNAME),(ESSN,DEPENDENT_NAME) (DEPENDENT)
R <--  $\Pi$  LNAME,FNAME (E)
```

Result (empty):

LNAME	FNAME
-----	-----

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

```
WONG_SSN <--  $\Pi$  SSN ( $\sigma$  FNAME='Franklin' AND
LNAME='Wong'(EMPLOYEE))
WONG_EMPS <-- (EMPLOYEE)  $\vartheta$  (SUPERSSN),(SSN) (WONG_SSN)
RESULT <--  $\Pi$  LNAME,FNAME (WONG_EMPS)
```

Result:

LNAME	FNAME
Smith	John
Narayan	Ramesh
English	Joyce

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

$PROJ_HOURS(PNO, TOT_HRS) \leftarrow PNO \uparrow \sum HOURS(WORKS_ON)$
 $RESULT \leftarrow \Pi_{PNAME, TOT_HRS} ((PROJ_HOURS) \vartheta_{(PNO), (PNUMBER)} (PROJECT))$

Result:

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

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

$PROJ_EMPS(PNO, SSN) \leftarrow \Pi_{PNO, ESSN} (WORKS_ON)$
 $ALL_PROJS(PNO) \leftarrow \Pi_{PNUMBER} (PROJECT)$
 $EMPS_ALL_PROJS \leftarrow PROJ_EMPS \text{ :- } ALLPROJS \quad (* \text{ DIVISION operation } *)$
 $RESULT \leftarrow \Pi_{LNAME, FNAME} (EMPLOYEE * EMP_ALL_PROJS)$

Result (empty):

LNAME	FNAME
-----	-----

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

$ALL_EMPS \leftarrow \Pi_{SSN} (EMPLOYEE)$
 $WORKING_EMPS(SSN) \leftarrow \Pi_{ESSN} (WORKS_ON)$
 $NON_WORKING_EMPS \leftarrow ALL_EMPS - WORKING_EMPS \quad (* \text{ DIFFERENCE } *)$
 $RESULT \leftarrow \Pi_{LNAME, FNAME} (EMPLOYEE * NON_WORKING_EMPS)$

Result (empty):

LNAME	FNAME
-----	-----

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

$DEPT_AVG_SALS(DNUMBER, AVG_SAL) \leftarrow DNO \uparrow \text{AVG SALARY}(EMPLOYEE)$
 $RESULT \leftarrow \Pi_{DNUMBER, AVG_SAL} (DEPT_AVG_SALS * DEPARTMENT)$

Result:

DNUMBER	AVG_SAL
Research	33250
Administration	31000
Headquarters	55000

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

$RESULT(AVG_F_SAL) \leftarrow f_{AVG\ SALARY}(\sigma_{SEX='F'}(EMPLOYEE))$

Result:

AVG_F_SAL
31000

(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.

$E_P_HOU(SSN) \leftarrow$
 $\Pi_{ESSN} (WORKS_ON \vartheta_{(PNO),(PNUMBER)}(\sigma_{PLOCATION='Houston'}(PROJECT)))$
 $D_NO_HOU \leftarrow$
 $\Pi_{DNUMBER}(DEPARTMENT) - \Pi_{DNUMBER}(\sigma_{DLOCATION='Houston'}(DEPARTMENT))$
 $E_D_NO_HOU \leftarrow \Pi_{SSN} (EMPLOYEE \vartheta_{(PNO),(DNUMBER)}(D_NO_HOU))$
 $RESULT_EMPS \leftarrow E_P_HOU - E_D_NO_HOU$ (* this is set DIFFERENCE *)
 $RESULT \leftarrow \Pi_{LNAME,FNAME,ADDRESS} (EMPLOYEE * RESULT_EMPS)$

Result:

LNAME	FNAME	ADDRESS
Wallace	Jennifer	291 Berry, Bellaire, TX

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

$DEPT_MANAGERS(SSN) \leftarrow \Pi_{MGRSSN} (DEPARTMENT)$
 $EMPS_WITH_DEPENDENTS(SSN) \leftarrow \Pi_{ESSN} (DEPENDENT)$
 $RESULT_EMPS \leftarrow DEPT_MANAGERS - EMPS_WITH_DEPENDENTS$
 $RESULT \leftarrow \Pi_{LNAME,FNAME} (EMPLOYEE * RESULT_EMPS)$

Result:

LNAME	FNAME
Borg	James

6.20 Suppose each of the following update operations is applied directly to the database of Figure 6.6. Discuss all integrity constraints violated by each operation, if any, and the different ways of enforcing these constraints.

Answers:

(a) Insert < 'Robert', 'F', 'Scott', '943775543', '21-JUN-42', '2365 Newcastle Rd, Bellaire, TX', M, 58000, '888665555', 1 > into EMPLOYEE.

No constraint violations.

(b) Insert < 'ProductA', 4, 'Bellaire', 2 > into PROJECT.

Violates referential integrity because DNUM=2 and there is no tuple in the DEPARTMENT relation with DNUMBER=2. We may enforce the constraint by: (i) rejecting the

insertion of the new PROJECT tuple, (ii) changing the value of DNUM in the new PROJECT tuple to an existing DNUMBER value in the DEPARTMENT relation, or (iii) inserting a new DEPARTMENT tuple with DNUMBER=2.

(c) Insert < 'Production', 4, '943775543', '01-OCT-88' > into DEPARTMENT.

Violates both the key constraint and referential integrity. Violates the key constraint because there already exists a DEPARTMENT tuple with DNUMBER=4. We may enforce this constraint by: (i) rejecting the insertion, or (ii) changing the value of DNUMBER in the new DEPARTMENT tuple to a value that does not violate the key constraint. Violates referential integrity because MGRSSN='943775543' and there is no tuple in the EMPLOYEE relation with SSN='943775543'. We may enforce the constraint by: (i) rejecting the insertion, (ii) changing the value of MGRSSN to an existing SSN value in EMPLOYEE, or (iii) inserting a new EMPLOYEE tuple with SSN='943775543'.

(d) Insert < '677678989', null, '40.0' > into WORKS_ON.

Violates both the entity integrity and referential integrity. Violates entity integrity because PNO, which is part of the primary key of WORKS_ON, is null. We may enforce this constraint by: (i) rejecting the insertion, or (ii) changing the value of PNO in the new WORKS_ON tuple to a value of PNUMBER that exists in the PROJECT relation. Violates referential integrity because ESSN='677678989' and there is no tuple in the EMPLOYEE relation with SSN='677678989'. We may enforce the constraint by: (i) rejecting the insertion, (ii) changing the value of ESSN to an existing SSN value in EMPLOYEE, or (iii) inserting a new EMPLOYEE tuple with SSN='677678989'.

(e) Insert < '453453453', 'John', M, '12-DEC-60', 'SPOUSE' > into DEPENDENT.

No constraint violations.

(f) Delete the WORKS_ON tuples with ESSN= '333445555'.

No constraint violations.

(g) Delete the EMPLOYEE tuple with SSN= '987654321'.

Violates referential integrity because several tuples exist in the WORKS_ON, DEPENDENT, DEPARTMENT, and EMPLOYEE relations that reference the tuple being deleted from EMPLOYEE. We may enforce the constraint by: (i) rejecting the deletion, or (ii) deleting all tuples in the WORKS_ON, DEPENDENT, DEPARTMENT, and EMPLOYEE relations whose values for ESSN, ESSN, MGRSSN, and SUPERSSN, respectively, is equal

to '987654321'.

(h) Delete the PROJECT tuple with PNAME= 'ProductX'.

Violates referential integrity because two tuples exist in the WORKS_ON relations that reference the tuple being deleted from PROJECT. We may enforce the constraint by: (i) rejecting the deletion, or (ii) deleting the tuples in the WORKS_ON relation whose value for PNO=1 (the value for the primary key PNUMBER for the tuple being deleted from PROJECT).

(i) Modify the MGRSSN and MGRSTARTDATE of the DEPARTMENT tuple with DNUMBER= 5 to '123456789' and '01-OCT-88', respectively.

No constraint violations.

(j) Modify the SUPERSSN attribute of the EMPLOYEE tuple with SSN= '999887777' to '943775543'.

Violates referential integrity because the new value of SUPERSSN='943775543' and there is no tuple in the EMPLOYEE relation with SSN='943775543'. We may enforce the constraint by: (i) rejecting the deletion, or (ii) inserting a new EMPLOYEE tuple with SSN='943775543'.

(k) Modify the HOURS attribute of the WORKS_ON tuple with ESSN= '999887777' and PNO= 10 to '5.0'.

No constraint violations.

6.21 Consider the AIRLINE relational database schema shown in Figure 6.20, which describes a database for airline flight information. Each FLIGHT is identified by a flight NUMBER, and consists of one or more FLIGHT_LEGs with LEG_NUMBERS 1, 2, 3, etc. Each leg has scheduled arrival and departure times and airports, and has many LEG_INSTANCES--one for each DATE on which the flight travels. FARES are kept for each flight. For each leg instance, SEAT_RESERVATIONS are kept, as is the AIRPLANE used in the leg, and the actual arrival and departure times and airports. An AIRPLANE is identified by an AIRPLANE_ID, and is of a particular AIRPLANE_TYPE. CAN_LAND relates AIRPLANE_TYPES to the AIRPORTs in which they can land. An AIRPORT is identified by an AIRPORT_CODE. Specify the following queries in relational algebra: