Selected Exercises from Fundamentals of Database Systems

Andrew Kowalczyk

I selected around 40 questions from the 6^{th} edition of *Fundamentals of Database Systems* by Ramez Elmasri and Shamkant Navathe. This was part of my requirement for my Introduction to Database Systems course taken in the Fall of 2013.

Chapter 1

1.8 Identify some informal queries and update operations that you would expect to apply to the database shown in Figure 1.2.

Queries

1. Find the names of all students majoring in Mathematics.
2. What are the prerequisites of the Database course?
3. Find the the transcript of the student named Brown. We would find <Course_name, Section_identifier, Semester, Year, Grade> for each course section that Brown has taken.

Updates

1. Insert a new student in the database whose Name=Kowalczyk, Student_number=25, Class=4, and Major=CS.
2. Change the grade that Brown received in Discrete Mathematics to a D.

1.9 What is the difference between controlled and uncontrolled redundancy? Illustrate with examples.

Redundancy is the term given when the same data is stored multiple times in several places in a database. If you look at Figure 1.5(a) in the text, you can see that the name of the student with Student_number=8 is Brown is stored multiple times. Redundancy is controlled when the database management system (DBMS) ensures that multiple copies of the same data are consistent. To illustrate this, let's say we are adding a new record

with Student_number=8 to be stored in the database of Figure 1.5(a). If we were to have uncontrolled redundancy, the DBMS would have no control over this. If we were to have controlled redundancy, the DBMS would ensure that Student_name=Brown in that record.

1.10 Specify all the relationships among the records of the database shown in Figure 1.2.

1. Every GRADE_REPORT record is related to one STUDENT record and one SECTION record.
2. Every SECTION record is related to a COURSE record.
3. Every PREREQUISITE record relates two COURSE records. One being a course and the other being a prerequisite to that course.

1.11 Give some additional views that may be needed by other user groups for the database shown in Figure 1.2.

1. A view of each class section that groups all the students who took that section and their respective grade.
2. A view that gives the number of courses taken and the grade point average for each student.

1.12 Cite some examples of integrity constraints that you think can apply to the database shown in Figure 1.2.

Key constraints

1. Student_number must be unique for each STUDENT record.
2. Course_number must be unique for each COURSE record.

Referential integrity constraints

1. The Course_number in a SECTION record must also exist in some COURSE record.
2. The Student_number in a GRADE_REPORT record must also exist in some STUDENT record.

Domain constraints

1. Grades in a a given GRADE_REPORT record must be one of these values: A, B, C, D, F, I, W .

Chapter 2

2.12 Think of different users for the database shown in Figure 1.2. What types of applications would each user need? To which user category would each belong, and what type of interface would each need?

1. Students add and drop classes. Actions that they can do are as listed:
   (a) Register themselves in a section of a course
   (b) Drop themselves from a section of a course
2. Registrar. They enter data of registration of students in sections of courses, and later enter the grades of the students. Actions that they can do are as listed:
   (a) Check whether a student who is registered in a course has the appropriate prerequisite courses
   (b) Add a student to a section of a course
   (c) Enter the student grades for a section
3. Admissions. Their main application would be to enter newly accepted students into the database. Actions that they can do are as listed:
   (a) Add students to the school's records

Chapter 3

3.11 Suppose that each of the following Update operations is applied directly to the database state shown in Figure 3.6. Discuss all integrity constraints violated by each operation, if any, and the different ways of enforcing these constraints.

In each of examples, the different ways of enforcing the constraints is listed in preferential order (*i.e.* 1 is most preferred, 2 is less preferred, etc.).

(a) Insert <"Robert", "F", "Scott", "943775543", "1972-06-21", "2365 Newcastle Rd, Bellaire, TX", M, 58000, "888665555", 1> into EMPLOYEE.

No constraints violated.

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

Since there is no tuple in the DEPARTMENT relation with DNUM=2, this insertion would violate referential integrity for that very reason. We can enforce the constraint with these options:

1. Rejecting the insertion
2. Changing the value of DNUM in the new PROJECT tuple to an existing DNUMBER value in the DEPARTMENT relation
3. Inserting a new DEPARTMENT tuple with DNUMBER=2

(c) Insert <"Production", 4, "943775543", "2007-10-01"> into DEPARTMENT.

Oh no! Since there already exists a DEPARTMENT tuple with DNUMBER=4, this would violate the key constraint. Enforcement of this constraint would happen by:

1. Rejecting the insertion
2. Changing the value of DNUMBER in the new DEPARTMENT tuple to a value that does not violate the key constraint

Also, since there is no tuple in the EMPLOYEE relation with SSN="943775543", this insertion also happens to violate referential integrity. Let us enforce the constraint by either:

1. Rejecting the insertion
2. Changing the value of MGRSSN to an existing SSN value in EMPLOYEE
3. Inserting a new EMPLOYEE tuple with SSN="943775543"

(d) Insert <"677678989", NULL, "40.0"> into WORKS_ON.

Since PNO (part of the primary key of WORKS_ON) is null, this violates entity integrity. We have these options to enforce this constraint:

1. Rejecting the insertion
2. Changing the value of PNO in the new WORKS_ON tuple to a value of PNUMBER that exists in the PROJECT relation

Since there is no tuple in the EMPLOYEE relation with SSN="677678989", this insertion would violate referential integrity. We may enforce the constraint by:

1. Rejecting the insertion
2. Changing the value of ESSN to an existing SSN value in EMPLOYEE
3. Inserting a new EMPLOYEE tuple with SSN="677678989"

(e) Insert <"453453453","John","M","1990-12-12","spouse"> into DEPENDENT.

Nope! Not a single constraint was violated.

(f) Delete the WORKS_ON tuples with Essn = "333445555".

No constraints violated in the making of this delete.

(g) Delete the EMPLOYEE tuple with Ssn = "987654321".

Unfortunately, the employee trying to be deleted is referenced in the WORKS_ON, DEPENDENT, DEPARTMENT, and EMPLOYEE relations. We can enforce such an abolishment by either:

1. Rejecting the deletion
2. 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".

This deletion would completely violate referential integrity because two tuples exist in the WORKS_ON relation that reference the tuple being deleted from PROJECT. Silly us, let's enforce the constraint by:

1. Rejecting the deletion
2. Deleting the tuples in the WORKS_ON relation whose value for the primary key PNUMBER for the tuple being deleted from PROJECT with PNO=1

(i) Modify the Mgr_ssn and Mgr_start_date of the DEPARTMENT tuple with Dnumber = 5 to "123456789" and "2007-10-01", respectively.

No violation of constraints.

(j) Modify the Super_ssn attribute of the EMPLOYEE tuple with Ssn = "999887777" to "943775543".

Goodness gracious, great balls of fire! Since there is no tuple in the EMPLOYEE relation with SSN="943775543", this update would violate referential integrity. In order to enforce this constraint, we can choose from:

1. Rejecting the deletion
2. 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 constraints violated.

Chapter 4

Chapter 5

Chapter 6

6.15. Show the result of each of the sample queries in Section 6.5 as it would apply to the database state in Figure 3.6.

Query 1: Find the name and address of all employees who work for the 'Research' department.

FNAME	LNAME	ADDRESS
John	Smith	731 Fondren, Houston, TX
Franklin	Wong	638 Voss, Houston, TX
Ramesh	Narayan	975 Fire Oak, Humble, TX
Joyce	English	5631 Rice, Houston, TX

Query 2: For every project located in 'Stafford', list the project number, the controlling department number, and the department manager's last name, address, and birth date.

PNUMBER	DNUM	LNAME	ADDRESS	BDATE
10	4	Wallace	291 Berry, Bellaire, TX	20-JUN-31
30	4	Wallace	291 Berry, Bellaire, TX	20-JUN-31

Query 3: Find the names of all employees who work on all the projects controlled by department number 5.

LNAME	FNAME

Query 4: Make a list of project numbers for projects that involve an employee whose last name is 'Smith' as a worker or as a manager of the department that controls the project.

PNO
1
2

Query 5: List the names of all employees with two or more dependents.

LNAME	FNAME
Smith	John
Wong	Franklin

Query 6 List the names of employees who have no dependents.

LNAME	FNAME
Zelaya	Alicia
Narayan	Ramesh
English	Joyce
Jabbar	Ahmad
Borg	James

Query 7: List the names of managers who have at least one dependent.

LNAME	FNAME
Wallace	Jennifer
Wong	Franklin

6.16 Specify the following queries on the COMPANY relational database schema shown in Figure 5.5, using the relational operators discussed in this chapter. Also show the result of each query as it would apply to the database state in Figure 3.6.

I use the symbol σ for SELECT, π for PROJECT, $\rhd\lhd$ for EQUIJOIN, * for NATURAL JOIN, and $\mathfrak{F}$ for FUNCTION.

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

Relational Operators

EMP_W_X $\Leftarrow$ (σ $_{\text{PNAME="ProductX"}}$ (PROJECT)) $\rhd\lhd$ $_{\text{(PNUMBER),(PNO)}}$ (WORKS_ON)
EMP_WORK_10 $\Leftarrow$ (EMPLOYEE) $\rhd\lhd$ $_{\text{(SSN),(ESSN)}}$ (σ $_{\text{HOURS>10}}$ (EMP_W_X))
RESULT $\Leftarrow$ π $_{\text{LNAME,FNAME}}$ (σ $_{\text{DNO=5}}$ (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.

Relational Operators

$E \Leftarrow (EMPLOYEE) \rhd\lhd_{(SSN,FNAME),(ESSN,DEPENDENT\_NAME)} (DEPENDENT)$
$R \Leftarrow \pi_{LNAME,FNAME} (E)$

Result

LNAME	FNAME

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

Relational Operators

$WONG\_SSN \Leftarrow \pi_{SSN} (\sigma_{FNAME="Franklin" \text{ AND } LNAME="Wong"} (EMPLOYEE))$
$WONG\_EMPS \Leftarrow (EMPLOYEE) \rhd\lhd_{(SUPERSSN),(SSN)} (WONG\_SSN)$
$RESULT \Leftarrow \pi_{LNAME,FNAME} (WONG\_EMPS)$

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.

Relational Operators

$PROJ\_HOURS(PNO,TOT\_HRS) \Leftarrow_{PNO} \mathfrak{F}_{SUM\ HOURS} (WORKS\_ON)$
$RESULT \Leftarrow \pi_{PNAME,TOT\_HRS} ((PROJ\_HOURS) \rhd\lhd_{(PNO),(PNUMBER)} (PROJECT))$

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.

Relational Operators

PROJ_EMPS(PNO,SSN) $\Leftarrow \pi_{\text{PNO,ESSN}}$ (WORKS_ON)
ALL_PROJS(PNO) $\Leftarrow \pi_{\text{PNUMBER}}$ (PROJECT)
EMPS_ALL_PROJS $\Leftarrow$ PROJ_EMPS $\div$ ALLPROJS
RESULT $\Leftarrow \pi_{\text{LNAME,FNAME}}$ (EMPLOYEE * EMP_ALL_PROJS)

Result

LNAME	FNAME

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

Relational Operators

ALL_EMPS $\Leftarrow \pi_{\text{SSN}}$ (EMPLOYEE)
WORKING_EMPS(SSN) $\Leftarrow \pi_{\text{ESSN}}$ (WORKS_ON)
NON_WORKING_EMPS $\Leftarrow$ ALL_EMPS - WORKING_EMPS
RESULT $\Leftarrow \pi_{\text{LNAME,FNAME}}$ (EMPLOYEE * NON_WORKING_EMPS)

Result

LNAME	FNAME

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

Relational Operators

DEPT_AVG_SALS(DNUMBER,AVG_SAL) $\Leftarrow {}_{\text{DNO}}\mathfrak{F}_{\text{AVG SALARY}}$ (EMPLOYEE)
RESULT $\Leftarrow \pi_{\text{DNUMBER,AVG\_SAL}}$ (DEPT_AVG_SALS * DEPARTMENT)

Result

DNUMBER	AVG_SAL
Research	33250
Administration	31000
Headquarters	55000

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

Relational Operators

RESULT(AVG_F_SAL) $\Leftarrow \mathfrak{F}_{\text{AVG SALARY}}$ ($\sigma_{\text{SEX="F"}}$ (EMPLOYEE))

Result

AVG_F_SAL
31,000

(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.

Relational Operators

E_P_HOU(SSN) $\Leftarrow \pi_{\text{ESSN}}$ (WORKS_ON $\rhd\lhd_{\text{(PNO),(PNUMBER)}}$ ($\sigma_{\text{PLOCATION="Houston"}}$ (PROJECT)))
D_NO_HOU $\Leftarrow \pi_{\text{DNUMBER}}$ (DEPARTMENT) - π_{DNUMBER} ($\sigma_{\text{DLOCATION="Houston"}}$ (DEPARTMENT))
E_D_NO_HOU $\Leftarrow \pi_{\text{SSN}}$ (EMPLOYEE $\rhd\lhd_{\text{(PNO),(DNUMBER)}}$ (D_NO_HOU))
RESULT_EMPS $\Leftarrow$ E_P_HOU - E_D_NO_HOU
RESULT $\Leftarrow \pi_{\text{LNAME,FNAME,ADDRESS}}$ (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.

Relational Operators

DEPT_MANAGERS(SSN)$<-$ π_{MGRSSN} (DEPARTMENT)
EMPS_WITH_DEPENDENTS(SSN) $\Leftarrow \pi_{\text{ESSN}}$ (DEPENDENT)
RESULT_EMPS $\Leftarrow$ DEPT_MANAGERS - EMPS_WITH_DEPENDENTS
RESULT $\Leftarrow \pi_{\text{LNAME,FNAME}}$ (EMPLOYEE * RESULT_EMPS)

Relational Operators

LNAME	FNAME
Borg	James

Chapter 7

Chapter 8

Chapter 9

Chapter 10