CHAPTER 2

Introduction to the Relational Model

Practice Exercises

2.1 Consider the employee database of Figure 2.17. What are the appropriate primary keys?

Answer:
The appropriate primary keys are shown below:

employee (*person_name*, *street*, *city*)
works (*person_name*, *company_name*, *salary*)
company (*company_name*, *city*)

2.2 Consider the foreign-key constraint from the *dept_name* attribute of *instructor* to the *department* relation. Give examples of inserts and deletes to these relations that can cause a violation of the foreign-key constraint.

Answer:

- Inserting a tuple:

  (10111, Ostrom, Economics, 110000)

---

employee (*ID*, *person_name*, *street*, *city*)
works (*ID*, *company_name*, *salary*)
company (*company_name*, *city*)

Figure 2.17 Employee database.

into the *instructor* table, where the *department* table does not have the department Economics, would violate the foreign-key constraint.

- Deleting the tuple:

  (Biology, Watson, 90000)

  from the *department* table, where at least one student or instructor tuple has *dept_name* as Biology, would violate the foreign-key constraint.

2.3 Consider the *time_slot* relation. Given that a particular time slot can meet more than once in a week, explain why *day* and *start_time* are part of the primary key of this relation, while *end_time* is not.

Answer:
The attributes *day* and *start_time* are part of the primary key since a particular class will most likely meet on several different days and may even meet more than once in a day. However, *end_time* is not part of the primary key since a particular class that starts at a particular time on a particular day cannot end at more than one time.

2.4 In the instance of *instructor* shown in Figure 2.1, no two instructors have the same name. From this, can we conclude that *name* can be used as a superkey (or primary key) of *instructor*?

Answer:
No. For this possible instance of the instructor table the names are unique, but in general this may not always be the case (unless the university has a rule that two instructors cannot have the same name, which is a rather unlikey scenario).

2.5 What is the result of first performing the Cartesian product of *student* and *advisor*, and then performing a selection operation on the result with the predicate *s_id* = ID? (Using the symbolic notation of relational algebra, this query can be written as $\sigma_{s\_id=ID}(student \times advisor)$.)

Answer:
The result attributes include all attribute values of *student* followed by all attributes of *advisor*. The tuples in the result are as follows: For each student who has an advisor, the result has a row containing that student's attributes, followed by an *s_id* attribute identical to the student's ID attribute, followed by the *i_id* attribute containing the ID of the students advisor.

Students who do not have an advisor will not appear in the result. A student who has more than one advisor will appear a corresponding number of times in the result.

2.6 Consider the employee database of Figure 2.17. Give an expression in the relational algebra to express each of the following queries:

a. Find the name of each employee who lives in city "Miami".

branch(*branch_name*, *branch_city*, *assets*)
customer (*ID*, *customer_name*, *customer_street*, *customer_city*)
loan (*loan_number*, *branch_name*, *amount*)
borrower (*ID*, *loan_number*)
account (*account_number*, *branch_name*, *balance*)
depositor (*ID*, *account_number*)

Figure 2.18 Bank database.

b. Find the name of each employee whose salary is greater than $100000.

c. Find the name of each employee who lives in "Miami" and whose salary is greater than $100000.

Answer:

a. $\Pi_{person\_name}(\sigma_{city = \text{"Miami"}}(employee))$

b. $\Pi_{person\_name}(\sigma_{salary > 100000}(employee \bowtie works))$

c. $\Pi_{person\_name}(\sigma_{city = \text{"Miami"} \wedge salary > 100000}(employee \bowtie works))$

2.7 Consider the bank database of Figure 2.18. Give an expression in the relational algebra for each of the following queries:

a. Find the name of each branch located in "Chicago".

b. Find the ID of each borrower who has a loan in branch "Downtown".

Answer:

a. $\Pi_{branch\_name}(\sigma_{branch\_city = \text{"Chicago"}}(branch))$

b. $\Pi_{ID}(\sigma_{branch\_name = \text{"Downtown"}}(borrower \bowtie_{borrower.loan\_number = loan.loan\_number} loan))$.

2.8 Consider the employee database of Figure 2.17. Give an expression in the relational algebra to express each of the following queries:

a. Find the ID and name of each employee who does not work for "BigBank".

b. Find the ID and name of each employee who earns at least as much as every employee in the database.

Answer:

a. To find employees who do not work for BigBank, we first find all those who *do* work for BigBank. Those are exactly the employees *not* part of the

desired result. We then use set difference to find the set of all employees minus those employees that should not be in the result.

$$\Pi_{ID,person\_name}(employee) - \Pi_{ID,person\_name}(employee \bowtie_{employee.ID=works.ID} (\sigma_{company\_name=\}\text{BigBank}''}(works)))$$

b. We use the same approach as in part *a* by first finding those employess who do not earn the highest salary, or, said differently, for whom some other employee earns more. Since this involves comparing two employee salary values, we need to reference the *employee* relation twice and therefore use renaming.

$$\Pi_{ID,person\_name}(employee) - \Pi_{A.ID,A.person\_name}(\rho_A(employee) \bowtie_{A.salary<B.salary} \rho_B(employee))$$

2.9 The **division operator** of relational algebra, "÷", is defined as follows. Let $r(R)$ and $s(S)$ be relations, and let $S \subseteq R$; that is, every attribute of schema S is also in schema R. Given a tuple t, let $t[S]$ denote the projection of tuple t on the attributes in S. Then $r \div s$ is a relation on schema $R - S$ (that is, on the schema containing all attributes of schema R that are not in schema S). A tuple t is in $r \div s$ if and only if both of two conditions hold:

- t is in $\Pi_{R-S}(r)$
- For every tuple t_s in s, there is a tuple t_r in r satisfying both of the following:
  a. $t_r[S] = t_s[S]$
  b. $t_r[R - S] = t$

Given the above definition:

a. Write a relational algebra expression using the division operator to find the IDs of all students who have taken all Comp. Sci. courses. (Hint: project *takes* to just ID and *course_id*, and generate the set of all Comp. Sci. *course_id*s using a select expression, before doing the division.)

b. Show how to write the above query in relational algebra, without using division. (By doing so, you would have shown how to define the division operation using the other relational algebra operations.)

Answer:

a. $\Pi_{ID}(\Pi_{ID,course\_id}(takes) \div \Pi_{course\_id}(\sigma_{dept\_name='\text{Comp. Sci}'}(course))$

b. The required expression is as follows:

$$r \leftarrow \Pi_{ID,course\_id}(takes)$$

$$s \leftarrow \Pi_{course\_id}(\sigma_{dept\_name=\text{'Comp. Sci'}}(course))$$

$$\Pi_{ID}(takes) - \Pi_{ID}((\Pi_{ID}(takes) \times s) - r)$$

In general, let $r(R)$ and $s(S)$ be given, with $S \subseteq R$. Then we can express the division operation using basic relational algebra operations as follows:

$$r \div s = \Pi_{R-S}(r) - \Pi_{R-S}((\Pi_{R-S}(r) \times s) - \Pi_{R-S,S}(r))$$

To see that this expression is true, we observe that $\Pi_{R-S}(r)$ gives us all tuples t that satisfy the first condition of the definition of division. The expression on the right side of the set difference operator

$$\Pi_{R-S}((\Pi_{R-S}(r) \times s) - \Pi_{R-S,S}(r))$$

serves to eliminate those tuples that fail to satisfy the second condition of the definition of division. Let us see how it does so. Consider $\Pi_{R-S}(r) \times s$. This relation is on schema R, and pairs every tuple in $\Pi_{R-S}(r)$ with every tuple in s. The expression $\Pi_{R-S,S}(r)$ merely reorders the attributes of r.

Thus, $(\Pi_{R-S}(r) \times s) - \Pi_{R-S,S}(r)$ gives us those pairs of tuples from $\Pi_{R-S}(r)$ and s that do not appear in r. If a tuple t_j is in

$$\Pi_{R-S}((\Pi_{R-S}(r) \times s) - \Pi_{R-S,S}(r))$$

then there is some tuple t_s in s that does not combine with tuple t_j to form a tuple in r. Thus, t_j holds a value for attributes $R - S$ that does not appear in $r \div s$. It is these values that we eliminate from $\Pi_{R-S}(r)$.