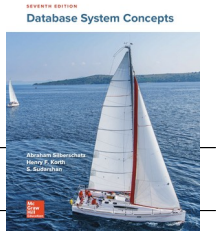
Database System Concepts



Chapter 2

REVISION 1

Terms (GNU AGPLv3)

This is **just a sample**!
I have tried to put things like **summary**, and obviously this is not the solution that I will submit or recommended! I am no way responsible for any illegal use of this file.

If you need direct solutions, please ask **DeepSeek, ChatGPT**, **Google Gemini**, **Anthropic Claude**, **Hugging Chat**, **Le Chat Mistral** or any other predictive model.

Chapter 2 | Practice Exercises

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

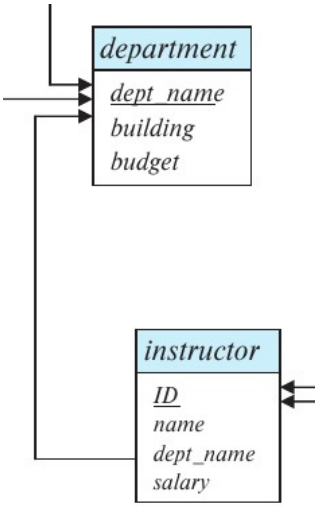
employee (*person_name*, *street*, *city*)
works (*person_name*, *company_name*, *salary*)
company (*company_name*, *city*)

Figure 2.17 Employee database.

The appropriate primary keys are shown below:

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

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.



If we try to insert the following data into the instructor relation we will break the integration,

- (1024, “Sharafat”, “CSEES”, 0)

because the “CSEES” department doesn’t exist in the department relation.

And if we try to remove the following entry from the department relation, then it can break the integrity if CIT department has any instructor,

- (“CIT”, “Library”, 10000)

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.

The attributes day and start time are part of the primary key sin e a particular class will most likely meet on several different days and may even meet more than on e in a day. However, end time is not part of the primary key sin e a particular class that starts at a particular time on a particular day cannot end at more than one time.

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?

No, because even if in this database we don't have multiple teachers who have same name, unless in the university explicitly defines that multiple instructor can't have same name, we can't conclude that "name" can be a super key.

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 $\sigma_{s\_id=ID}(student \times advisor)$.)

In this case first attributes of student and advisors will be combined. Then based on "s_id = ID". So only attributes with identical s_id and ID will be shown.

Students who have no advisor, will not be shown. Then if any student has multiple advisor then multiple times it'll be shown.

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

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

Figure 2.17 Employee database.

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

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

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

$\Pi_{person\_name}(\sigma_{salary > 10000}(employee \bowtie_{person\_name} works))$

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

$\Pi_{person\_name}(\sigma_{city = \text{"Miami"} \wedge salary > 10000}(employee \bowtie_{person\_name} works))$

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

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.

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

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

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

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

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".**

$\Pi_{\text{ID, person\_name}} (\text{employee})$

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

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

$\Pi_{\text{ID, person\_name}} (\text{employee})$

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

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 ÷ 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 ÷ s if and only if both of two conditions hold:

- t is in Π_{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_ids using a select expression, before doing the division.)**

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

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.)**

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

Chapter 2 | Exercises

10. Describe the differences in meaning between the terms relation and relation schema.

Relation refers to the tables in a database. And relation schema refers to the structure of relations and the logical connection in between them.

11. Consider the advisor relation shown in the schema diagram in Figure 2.9, with s_id as the primary key of advisor. Suppose a student can have more than one advisor. Then, would s_id still be a primary key of the advisor relation? If not, what should the primary key of advisor be?

If a student can have more than one advisor, then both the s_id and i_id will be primary key. Because in this case we will have multiple students with same student id. So in order to uniquely identify we will need both s_id and i_id to be primary key.

12. Consider the bank database of Figure 2.18. Assume that branch names and customer names uniquely identify branches and customers, but loans and accounts can be associated with more than one customer.

a. What are the appropriate primary keys?

The appropriate primary keys will be,

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)

primary keys are underlined

b. Given your choice of primary keys, identify appropriate foreign keys.

The appropriate foreign keys will be,

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)

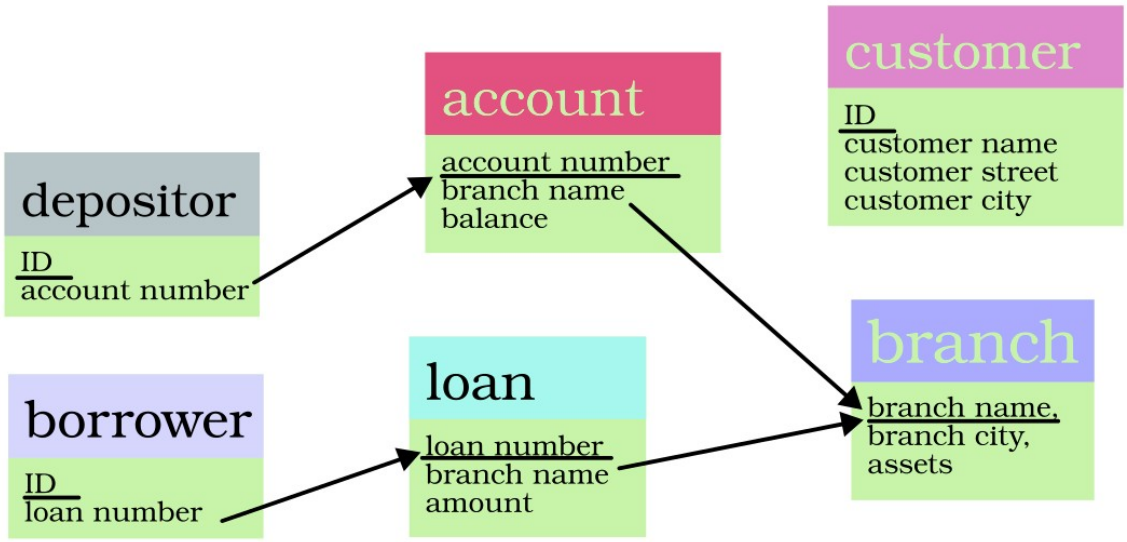
primary keys are marked as italic and secondary keys are underlined

13. Construct a schema diagram for the bank database of Figure 2.18.

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.

Here's a schema diagram for the above diagram,



14. 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 works for "BigBank".**

$\Pi_{\text{employee.ID, employee.person\_name}}$ (employee $\bowtie_{\text{employee.person\_name=company.person\_name}}$ $\sigma_{\text{compnay\_name="BigBank"}}$(company))

b. **Find the ID, name, and city of residence of each employee who works for "BigBank".**

$\Pi_{\text{employee.ID, employee.person\_name,city}}$ (employee $\bowtie_{\text{employee.person\_name=company.person\_name}}$ $\sigma_{\text{compnay\_name="BigBank"}}$(company))

c. **Find the ID, name, street address, and city of residence of each employee who works for "BigBank" and earns more than $10000.**

$\Pi_{\text{employee.ID, employee.person\_name,street,city}}$ ((employee $\bowtie_{\text{employee.person\_name=company.person\_name}}$ $\sigma_{\text{compnay\_name="BigBank"}}$(company)) $\bowtie_{\text{employee.person\_name=works.person\_name}}$ $\sigma_{\text{salary>10000}}$(works))

d. **Find the ID and name of each employee in this database who lives in the same city as the company for which she or he works.**

$\Pi_{\text{employee.ID, employee.person\_name}}$(employee $\bowtie_{\text{employee.city=company.city}}$ company)

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

a. **Find each loan number with a loan amount greater than $10000.**

$\Pi_{\text{loan\_number}}(\sigma_{\text{amount}>10000}(\text{loan}))$

b. **Find the ID of each depositor who has an account with a balance greater than $6000.**

$\Pi_{\text{ID}}(\sigma_{\text{balance}>10000}(\text{depositor} \bowtie_{\text{account\_number}} \text{account}))$

c. **Find the ID of each depositor who has an account with a balance greater than $6000 at the "Uptown" branch.**

$\Pi_{\text{ID}}(\sigma_{\text{balance}>6000 \wedge \text{branch\_name}=\text{"Uptown"}} (\text{depositor} \bowtie_{\text{account\_number}} \text{account}))$

16. List two reasons why null values might be introduced into a database.

To imply that a value of a database may not exist or left intentionally null value is introduced to the database.

17. Discuss the relative merits of imperative, functional, and declarative languages.

The relative merits are,

1. For imperative languages we give the direct instruction to do the task. So it has more performance.
2. For functional languages we think about what elements we may need. Then we make the function which will give the same time always. So it has less bugs.
3. In declarative languages, system does the logical part. We just have to tell the system what to do without thinking about time complexity, and the system will figure out how to pull the job.

18. Write the following queries in relational algebra, using the university schema.

a. **Find the ID and name of each instructor in the Physics department.**

$\Pi_{\text{ID, dept\_name}}(\sigma_{\text{dept\_name}=\text{"Physics"}}(\text{department}))$

b. **Find the ID and name of each instructor in a department located in the building "Watson".**

$\Pi_{\text{ID, instructor.name}}(\sigma_{\text{building}=\text{"Watson"}}((\text{department}) \bowtie_{\text{dept\_name}} \text{instructor}))$

c. **Find the ID and name of each student who has taken at least one course in the "Comp. Sci." department.**

$\Pi_{\text{ID, student.name}}(\text{student}) \cap$

$\Pi_{\text{ID, student.name}}(\sigma_{\text{dept\_name}=\text{"Comp. Sci."}}((\text{student}) \bowtie_{\text{ID}} (\text{takes})) \bowtie_{\text{course\_id}} (\text{course}))$

d. **Find the ID and name of each student who has taken at least one course section in the year 2018.**

$\Pi_{\text{ID, student.name}}(\text{student}) \cap$

$\Pi_{\text{ID, student.name}}(\sigma_{\text{section.year}=2018}(((\text{student}) \bowtie_{\text{ID}} (\text{takes})) \bowtie_{\text{course\_id}} (\text{section})))$

e. **Find the ID and name of each student who has not taken any course section in the year 2018.**

$\Pi_{\text{ID, student.name}}(\text{student}) \cap$

$\Pi_{\text{ID, student.name}}(\sigma_{\text{section.year}\neq2018}(((\text{student}) \bowtie_{\text{ID}} (\text{takes})) \bowtie_{\text{course\_id}} (\text{section})))$