Course Title : Database System
Course Code : CCE - 223

Assignment : 04

Submitted To

Dr. Md. Samsuzzaman
Professor
Department of Computer and Communication Engineering.
Faculty Of Computer Science & Engineering.

Submitted By

Name	:	Md Mohidul Alam
Id	:	1902016
Reg. No	:	08722
Session	:	2019 - 2020

Patuakhali Science & Technology University
Dumki, Patuakhali - 8602

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

Figure 2.17 Employee database.

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

Primary key of a relation:

- ❖ A primary key is a special column of a relation used for uniquely identifying a relation.
- ❖ The primary key can be one column or combination of columns. The primary key will uniquely identify all the records of a table.

The features of a primary are:

- ❖ The primary key contains a unique value for each row of data.
- ❖ The primary key column cannot contain null values.

The database has three relations that are, employee, works, and company.

- ➢ The primary key for the first relation can be “person_name'' that can identify the employees in the relation. The other available columns are “street” and “city” and that can come common to many employees. So the columns “street” and “city” cannot identify the employees in the table.

- ➢ The primary key for the second relation can be “person_name” that can identify the employees in the relation. The other available columns are ''company_name” and “salary” and that can come common to many works. That is may companies provide same kind of jobs with same amount of salary. So the columns ''company_name” and “salary” cannot identify the rows in the table.

- ➢ The primary key for the third relation can be “company_name” that can identify the company in the relation. The other available column is “city” and that can come common to many companes. So the column “city” cannot identify the companies in the table.

(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.

Foreign key:

- A foreign key is an attribute/column which is used for establishing relationship between two tables/relations.
- A foreign key must be primary key of the table to which it is referring.

Violation of foreign key constraint:

The foreign key of one table violates when the primary key that taken as foreign key is updated or deleted from the respected table.

Violation of foreign key occurs when an insert or update on a foreign key table is performed without a matching value in the primary key table.
For example, the table given are department and instructor. The primary key dept_name of department relation is taken as the foreign key of instructor relation.
So when inserting a row to the instructor without matching the primary key of department can cause violation of foreign key. For example consider the query given below-------**INSERT INTO instructor VALUES ('ins1', ‘sam', ‘economics', '200000');**
If there is no such a dept_name like “economics”, it will cause the violation in foreign key and it will show some error messages.

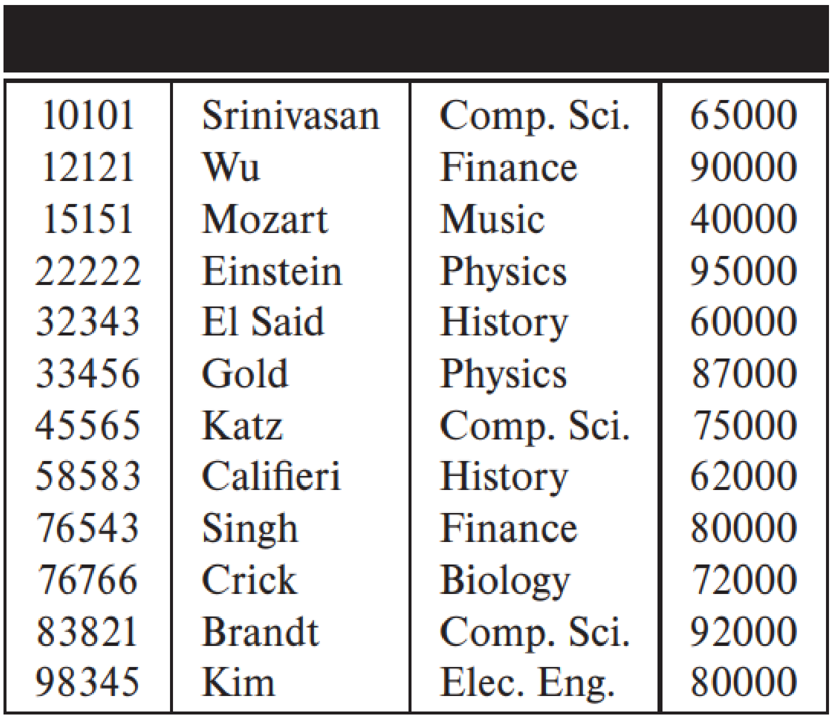
When deleting a primary key of a relation which is taken as the foreign key to another table, it will cause violation of foreign key. For example consider the below query,
DELETE FROM department WHERE dept_name='Physics';
The above deletes the row of details of “Physics” department. The dept_name is foreign key in another relation. So deleting the details affect the second relation. Here, the foreign key violation occurs.

(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.

The time slot relation is given below:
Time_slot (time_slot_id, day, start_time, end_time);

• The class section has an associated time_slot_id.The time_slot gives the information on which days of the week and what times a particular “time_slot_id” meets.For example, the “time_slot_id” is “A” and it starts at time 8 A.M and end at 8.50 A.M in the Monday,Wednesday and Friday.
So it is possible for a time_slot_id to have multiple sessions with in single day, at multiple times. So the "time_slot_id” and “day” cannot identify a time_slot for each course.So it is better to take the “start_time” along with “time_slot_id” to identify a section of a course.

• The column "end_time" is not much influence in identifying a time slot. So it is not taken as primary key.



10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Figure 2.1 The *instructor* relation.

(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?

❖ A column can be set as primary key attribute when it is having a unique value for each row of data.

❖ In the given relation instructor, if an attribute named “name” is having unique values for each row that is no two instructor have the same name, the attribute can be taken as primary key for the relation.

❖ The attribute cannot have the null values.

(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 σs id=ID(student × advisor).)

Effect of Cartesian product on two relations:

❖ The Cartesian product operation (×) is used to combine the information from two relations.
❖ The Cartesian product of two relations r1 and r2 can be written like,r1 × r2.
❖ For the given relations we can write, (Student × advisor)
❖ In the above operation, all the tuples from both the relation is taken and a new relation is created with all the tuples.
❖ If there is similar column names like “id” in both the relation, then the column name along with the relation name. For example, student.id and advisor.id

The join operation:

- The join operation is used to select the common tuples from two relations. The Cartesian product gives all the relations from both relation, but the join operation selects the required tuples.For example, if the user has two relations “student” and “advisor”. If the user wants the details of the students whose advisor is same. Then the join operation can be applied here. Like, σ s_id=ID(student × advisor)
- Here the user gets only those tuples of student × advisor that the details of students whose advisor is same.

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

Figure 2.17 Employee database.

(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”.
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.

- **SQL statement:** Πperson_name(σcity =“Miami” (employee))

The above relational algebra expression is used to get name of the employees who lives in the “Miami” city from the “employee” table.

- **SQL statement:** Πperson_name(σsalary >“$100000” (works))

The above relational algebra expression is used to get name of the employees whose salary is above “$100000” from the table “works”.

- **SQL statement:** Πperson_name (σcity =“Miami” ∧ salary >“$100000”=(employee × works))

The above relational algebra expression is used to get name of the employees who lives in the “Miami” city and their salary is above “$100000”. The data is in two relations. So the user should join the two relation named “employee” and “works”.

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.

(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”.

- **Relational algebra expression:** Πbranch_name(σbranch_city =“Chicago” (branch))

The above relational expression is used to get the branch_name which is located at “Chicago” from the branch relation.

- **Relational algebra expression:** ΠID (σbranch_name=”Downtown”(borrower × loan))

The above relational expression is used to get the ID of borrower who has taken loan from the branch “Downtown”. To get the ID, two tables’ borrower and loan is joined using join operation. Then the branch_name is selected as “Downtown”. The ID of the borrower is taken from the join table.

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

Figure 2.17 Employee database.

(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

- **Relational algebra statement:** ΠID,person_name(σdept_name ¬ =“BigBank” (employee×works))

The above relational algebra expression is used to get the employee ID and name who does not work for the “BigBank”.

- **Relational algebra statement:** ΠID,person_name(σsalary>=γ(min(salary))(employee×works))

The above relational algebra expression is used to get the employee ID and name of each employee who earns at least as much as every employee in the database. To get the smallest salary of an employee in the database the user can use the aggregate function “min”.

(2.9) The division operator of relational algebra, “÷”, is defined as follows. Let r(R) and s(S) be relations, and let S ⊆ 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 ΠR−S(r)
• For every tuple ts in s, there is a tuple tr in r satisfying both of the following:
a. tr[S] = ts[S]
b. tr[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.)

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.)

Relational algebra statement: ΠID,course_id(σdept_name =“Comp.sci” (student÷takes))
The above relational algebra expression is used to get the student ID of all student who have took all course from department “Comp.Sci”.

Relational algebra statement: ∏ID,course_id(student ⋈ takes ⋈ ∏course_id(σdept_name= ‘Comp.Sci.’(course)))
The above relational algebra expression is used to get the ID and name of the students who have taken courses from computer science department.

(2.10) Describe the differences in meaning between the terms relation and relation schema.

Relation	Relation schema
A relation is known as a Table	A relation schema is known as Table definition.
The relation contains the actual data in the form of rows and columns.	A relation schema is the basic information describing a table.
The relation is an instance of relation schema. The schema includes all the relations in a particular database.	The schema includes set of column names, the data types associated with the column.
A relation is a property or predicate that ranges over more than one argument.	The relational schema refers to meta-data elements which are used to describe structures and constraints of data representing particular domain.

(2.11) 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?

❖ A column can be set as primary key attribute when it is having a unique value for each row of data.

❖ In the given relation advisor, two attributes are present which are s_id and i_id. The s_id is the id of each student and i_id is the instructor id of each advisor. Currently s_id is the primary key of the advisor relation.

❖ If a student can have multiple advisors, the s_id cannot be taken as the primary key for the advisor relation. But the user can take combination of both s_id and i_id as the primary key. Because the instructor can also instruct more than one student. So it the combined the details, no duplicates will come in the relation.

❖ The attribute cannot have the null values.

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.

(2.12) 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?
b. Given your choice of primary keys, identify appropriate foreign keys.

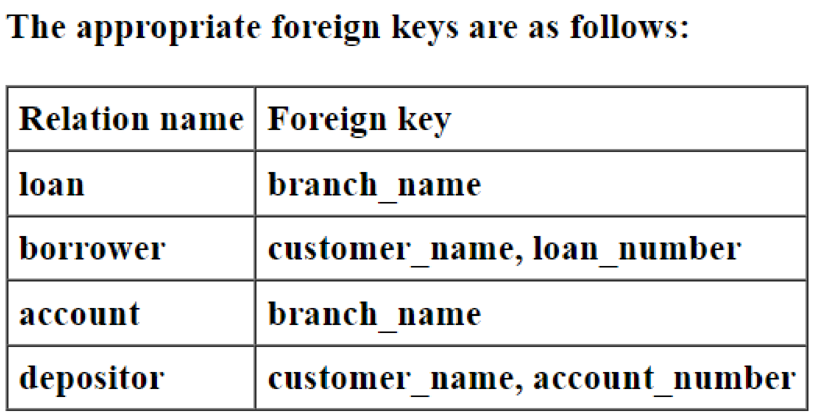
Relation name	**Primary key**
branch	branch_name
customer	customer_name
loan	loan_number
borrower	customer_name, loan_number
account	account_number
depositor	customer_name, account_number

Explanation:

- ✓ The branch_name in brach relation is unique name which identifies each tuple of branch relation.
- ✓ In the customer relation, each customer is identified by their customer_name. No two customers can have the same name.
- ✓ Each loan has a unique number which identifies each tuple of loan relation.
- ✓ The relation borrower is having composite primary key. The primary key is a combination of customer name and loan_number.
- ✓ Each account in account relation is unique which identifies each tuple of account.
- ✓ The relation depositor is having the composite key. The primary key is a combination of customer_name and account_number.

Foreign key:

- ❖ A foreign key is an attribute/column which is used for establishing relationship between two tables/relations.
- ❖ A foreign key must be primary key of the table to which it is referring.



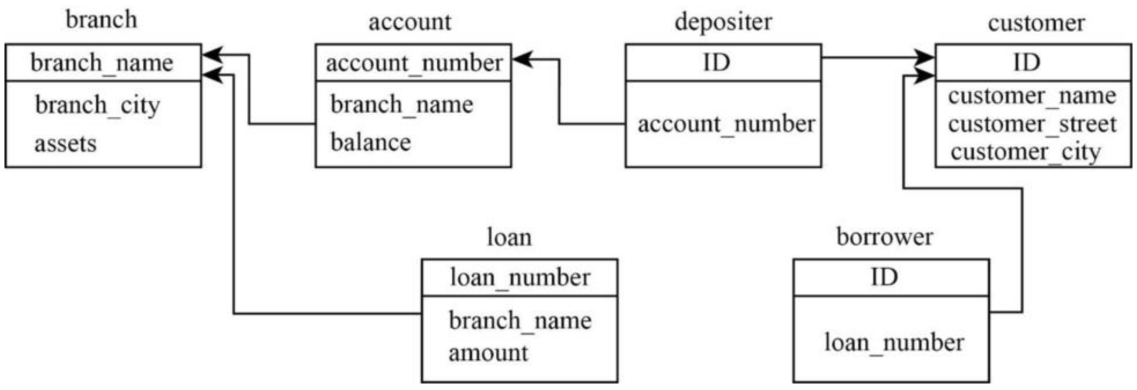
The appropriate foreign keys are as follows:

Relation name	Foreign key
loan	branch_name
borrower	customer_name, loan_number
account	branch_name
depositor	customer_name, account_number

Explanation:

- ✓ In loan relation, branch_name is a foreign key. It references branch_name in branch relation.
- ✓ In borrower relation, customer_name and loan_number are foreign keys. customer_name in borrower relation references customer_name in customer relation. loan_number in borrower relation references loan_number in loan relation.
- ✓ In account relation, branch_name is a foreign key. It references branch_name in relation.
- ✓ In depositor relation, customer_name and loan_number are foreign keys. customer_name in depositor relation references customer_name in customer relation. loan_number in depositor relation references loan_number in loan relation.

(2.13) Construct a schema diagram for the bank database of Figure 2.18.



Explanation :

❖ The above schema diagram represents the bank database. The diagram has six relations which are, branch, customer, loan, borrower, account and depositor. Each relation with their primary key given as below, 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)

❖ In the figure, the boxes carries a relation and primary keys are marked with an under line, The foreign keys are shown using the arrow line.

(2.14) 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".
b. Find the ID, name, and city of residence of each employee who works for "BigBank".
c. Find the ID, name, street address, and city of residence of each employee who works for "BigBank" and earns more than $10000.
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.

Query: ∏ID, person_name(σcompany_name=”BigBank”(works))
Explanation:The given query is used to find ID and name of each employee who works for BigBank in the employee database.
Here projection is performed on ID and person name and then selection is performed on big bank workers.
It is used to select all the tuples that satisfies the given predicate from the relation.
It also finds all the needed attributes from the relation.

2.15 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.
b. Find the ID of each depositor who has an account with a balance greater than $6000.
c. Find the ID of each depositor who has an account with a balance greater than $6000 at the "Uptown" branch.

SQL statement:

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

Explanation:
The above relational algebra expression is used to get the loan_number with loan amount greater than “$10000”.
SQL statement:

$\Pi_{ID}(\sigma_{balance>\text{“\$6000”}}(depositor \bowtie_{depositor.account\_number= account.account\_number} account))$

Explanation:
The above relational algebra expression is used to get the ID of each depositor who have balance greater than “$6000”.
SQL statement:

$\Pi_{ID}(\sigma_{balance>\text{“\$6000”}\wedge branch\_name=\text{”Uptown”}>\text{“\$100000”}}=((depositor \bowtie_{depositor.account\_number = account.account\_number} account))$

Explanation:
The above relational algebra expression is used to get the ID of each depositor who have balance greater than “$6000”. And the user is taken account from “Uptown” branch.

(2.16) List two reasons why null values might be introduced into a database.

Null values:
The null values are used to indicate that the value is not present in the current location.
The null values can be indicated by a space or zero or any other default value.
If a field contains null value means that content of the cell is unknown at the time of looking at it.
For example, an employee whose address has changed and the new address is not yet known. So this address should be indicated using a null value.
Use of null vales in database:
The null values are used in SQL joins and used by special handling required aggregate functions and grouping operators.
Use of null values may indicate absence of information about the values of an attribute, also used in a predicate to test for a null value.
The null values are used in arithmetic comparisons.

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

Relational query languages:
The query language is a language is a language in which user requests information from the database.
The query languages are higher level when compared to the standard programming languages.
The Query languages can be categorized to three that are, imperative, functional and declarative.
Imperative query language:
In the imperative query language, the system needs to perform specific sequence of operations on the database to compute the desired result.
The imperative paradigm is one that the hardware of the computer is based on.
With enough software the user can implement Object-Oriented programming, functional Programming, and Logical programming in the imperative paradigm.
Functional language:
In functional language, the computation is expressed as the evaluation of functions that may operate on data in the database or on the result of functions; functions are side-free, and they do not update the program state.
The pure functions are easier to reason about.
The programs in functional languages are easier to test.
The programs written in functional programming languages are easy to debug.

Declarative language:
In the declarative query language, without giving a specific sequence of steps or function calls, the user describes the desired information.
The desired is typically described using some form of mathematical logic.
The desired information is obtained by the databases.
The declarative language is domain specific language. This is the universal named Structured Query Language, most commonly known as SQL.

2.18 Write the following queries in relational algebra, using the university schema.
a. Find the ID and name of each instructor in the Physics department.

Further Reading 63
b. Find the ID and name of each instructor in a department located in the
building "Watson".
c. Find the ID and name of each student who has taken at least one course
in the "Comp. Sci." department.
d. Find the ID and name of each student who has taken at least one course
section in the year 2018.
e. Find the ID and name of each student who has not taken any course
section in the year 2018.

a.
Relational algebra statement:
$\Pi_{ID}(\sigma_{dept\_name = \text{“Physics”}} (instructor))$
Explanation:
The above relational algebra expression is used to get the ID of the instructors who works for physics department. The ID is get from the table instructor.
Explanation of Solution
b.
Relational algebra statement:
$\Pi_{ID,name}(\sigma_{building=\text{“Watson”}}((instructor \bowtie_{instructor.ID=teaches.ID} teaches) \bowtie_{teaches.course\_id=section.course\_id} section))$
Explanation:
The above relational algebra expression is used to get the ID and name of the instructors in a department located at the building "Watson".

c.
Relational algebra statement:
$\prod_{ID,name}(student \bowtie takes \bowtie \prod_{course\_id}(\sigma_{dept\_name= \text{‘Comp.Sci.’}}(course)))$
Explanation:
The above relational algebra expression is used to get the ID and name of the students who have taken at least one course from computer science department.
Explanation of Solution
d.
Relational algebra statement:
$\prod_{ID,name}(student \bowtie takes \bowtie \prod_{course\_id}(\sigma_{dept\_name= \text{‘Comp.Sci.’} \wedge year=\text{”2018”}}(takes \times course)))$
Explanation:
The above relational algebra expression is used to get the ID and name of the students who have taken course from computer science department in the year 2018.
Explanation of Solution
e.
Relational algebra statement:
$\prod_{ID,name}(student \bowtie takes \bowtie \prod_{course\_id}(\sigma_{dept\_name= \text{“null.”} \wedge year=\text{”2018”}} (takes \times course)))$
Explanation:
The above relational algebra expression is used to get the ID and name of the students who have not taken any course in the year 2018.