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

Figure 2.17 Employee database.

1. *Consider the employee database of Figure 2.17. What are the appropriate primary keys?*
   i. *Person name and Company name are the best columns candidate to be primary key with the most probable unique rows ((records))*

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.*

   What is foreign-key constraint Violation ??
   A foreign key constraint violation occurs when you try to insert or update a record in a table that references a primary key value that does not exist in the referenced table.

   For example, let's say you have two tables:

   1. *orders table with columns order_id (primary key) and customer_id (foreign key referencing the customers table).*
   2. *customers table with columns customer_id (primary key) and customer_name.*

   If you try to insert a new order with a customer_id that does not exist in the customers table, you will get a foreign key constraint violation error. This is because the customer_id in the orders table is a foreign key that references the customer_id primary key in the customers table.

   The foreign key constraint ensures referential integrity between the two tables, meaning that every value in the foreign key column of the orders table must match a value in the primary key column of the customers table.

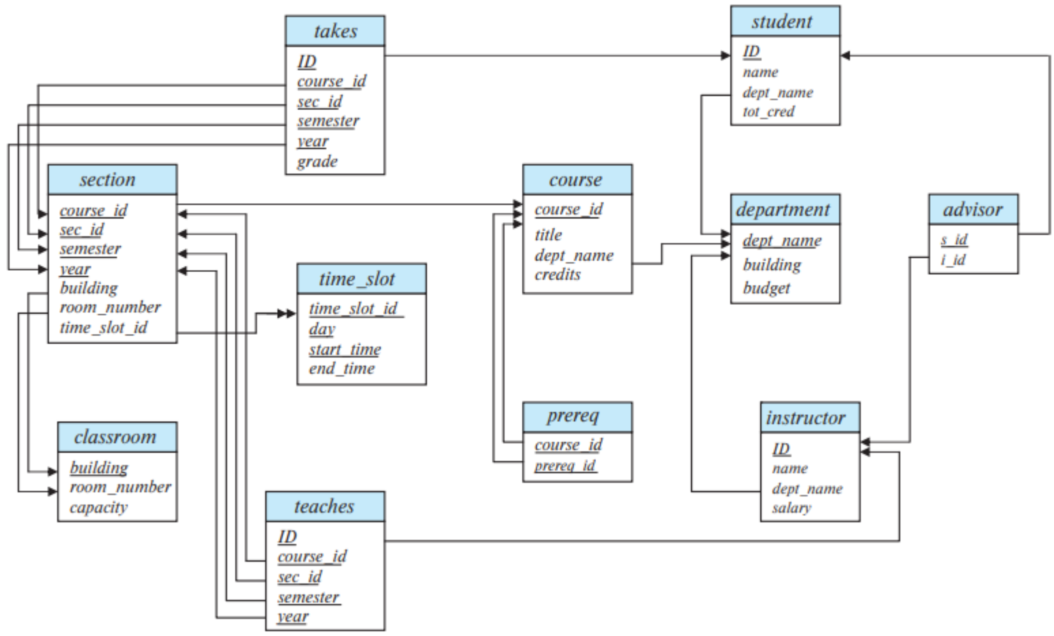


Figure 2.9 Schema diagram for the university database.

a. if we try adding a new record to instructor with a new department name that is not in department relation we will have an foreign key violation will happen since foreign key must keep integrity.
b. If we try removing a department name from the department relation we will have an error since this unique name could still be present as Instructor.dept_name, and we will get an error.

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.*

a) Why day and start_time are part of the primary key:

i) **A time slot can occur multiple times in a week:** Since a particular time slot (identified by time_slot_id) can be scheduled on multiple days, the day attribute is necessary to uniquely identify each occurrence.

ii) **Different time slots can have the same start_time:** A university may have multiple classes starting at the same time but on different days, so start_time alone is not enough to uniquely identify a record.

iii) **Uniqueness constraint:** The combination of time_slot_id, day, and start_time ensures that each instance of a time slot is uniquely identified.

b) Why end_time is not part of the primary key:

i) **End time is functionally dependent on start time and time slot ID:** Once a time slot is defined with a specific start_time, the end_time is automatically determined and does not add to the uniqueness of the tuple.

ii) **Redundancy:** Including end_time in the primary key would be unnecessary because start_time already determines when a session begins, and end time is just a dependent attribute.

ID	*name*	*dept_name*	*salary*
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.

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) No we can not conclude this fact. Just because names are unique in some records ((snapshot)) of the table, it does not mean that it is a valid primary key , sicnce we could have two instructors with the same name in the rest of the dataset

My question : What is the difference between super key and primary key ??

Difference Between Primary Key and Super Key

Feature	Primary Key	Super Key
Definition	A minimal set of attributes that uniquely identifies a row.	A set of one or more attributes that uniquely identifies a row (may contain extra attributes).
Uniqueness	Always unique for each row in a table.	Always unique but may have extra attributes that are not necessary for uniqueness.
Minimality	Must be **minimal**, meaning no attribute can be removed without losing uniqueness.	Not necessarily minimal; it can include extra attributes beyond the necessary ones.
Number per Table	Only **one** primary key per table.	A table can have **multiple** super keys.
Example	`{ID}` (if ID is unique for each student)	`{ID, name}`, `{ID, dept_name}`, `{ID, salary, name}` (all are super keys but not minimal)

Example in an Instructor Table

ID	Name	Dept_Name	Salary
101	Alice	CS	90,000
102	Bob	Math	85,000
103	Alice	CS	95,000

1. **Primary Key:** `{ID}` → It uniquely identifies each row and is minimal.
2. **Super Keys:**
   - `{ID, name}`
   - `{ID, dept_name}`
   - `{ID, salary, name}`
   - `{ID}` (which is also the **primary key**, as it is minimal)

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).)*

   a) **it would have the result of a join operation that will result in a relation that will have every information of student table along with the id of their instructor**
      i) *SELECT * FROM student JOIN advisor On s.id = ID*

6. Consider the employee database of Figure 2.17. Give an expression in the relational algebra to express each of the following queries:
   c) Find the name of each employee who lives in city "Miami".
   d) Find the name of each employee whose salary is greater than $100000.
   e) Find the name of each employee who lives in "Miami" and whose salary is greater than $100000.

   i. Π(σ(employee ; city = "Miami") ; name)
   ii. Π(σ(employee ; salary > 100000) ; name)
   iii. Π(σ(employee ; (city = "Miami") AND (salary > 100000)) ; name)

---

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.

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*
   a. Π(σ(branch ; (city = "chicago")) ; name)

b. Π(σ(borrower ; (loan_number ∈ [Π(σ(loan ; (branch_name= "Downtown")) ;loan_number)]) ; ID)

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.

c. Qa = Π(σ(works; company_name != "Big_Bank") ; person_name, id)
   M = ρ [Min[Π(σ(works); salary) ; minimum_salary]
   Qb = Π(σ(works; salary > minimum_salary) ; id, name)

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.)

The **division operator** is used when we want to find entities that are related to **all** records in another relation.

Example Scenario: Finding Students Who Have Taken All Required Courses

Suppose we have two relations:

1. **r(Student_Course)**
   - Stores which student has taken which course.
   - Schema: **r(Student_ID, Course)**

Student_ID	Course
1	Math
1	Physics
2	Math
2	Physics
2	Chemistry
3	Math
3	Chemistry

2. **s(Required_Courses)**
   - Stores the list of required courses.
   - Schema: **s(Course)**

Course
Math
Physics

We compute:

$$r \div s$$

- **Schema of the result: r(Student_ID, Course) ÷ s(Course)** gives us **(Student_ID)**.
- This finds students who have taken **ALL** the required courses.

Step-by-Step Explanation:

1. **ΠR−S(r) → Projection on Student_ID**
   - Extracts unique students:

Student_ID
1
2
3

2. **Check if every tuple in `s` (Required Courses) appears for a student in `r`**
   - Student 1 has taken **Math, Physics** → ✅ (Meets requirement)
   - Student 2 has taken **Math, Physics, Chemistry** → ✅ (Meets requirement)
   - Student 3 has taken **Math, Chemistry** → ❌ (Missing Physics)

Final Result (r ÷ s):

Student_ID
1
2

(i) We only need `ID` and `course_id`, so we project:

r = Π(Takes ; ID, course_id)

Extract all Computer Science courses from `course`

s = r = Π(σ(courses; depth_name = "comp_sci"); ID, course_id)

result = r % s : all comp_sci student who have taken all the courses.

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

a) A **relation** refers to a table or a set of tuples (records) in a database. It consists of a collection of data organized into rows and columns, where each row represents a unique record, and each column corresponds to an attribute of the record. A relation essentially holds the actual data stored in a table.

b) On the other hand, a **relation schema** refers to the blueprint or structure that defines the organization of a relation. It specifies the name of the relation and the attributes (columns) it contains, including the data types and constraints for each attribute. A relation schema defines the format in which the data is stored but does not include the actual data itself.

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) If a student (represented by `s_id`) can have more than one advisor in the advisor relation, then `s_id` alone cannot serve as the primary key of the relation. This is because a primary key must uniquely identify each record in a relation, and if a student can have multiple advisors, multiple records with the same `s_id` would exist, violating the uniqueness requirement for a primary key.

b) In this case, the primary key of the **advisor** relation should be a combination of `s_id` (the student's ID) and `advisor_id` (the advisor's ID). This composite key ensures that each unique pairing of a student and an advisor is uniquely identifiable, allowing a student to have multiple advisors while maintaining the uniqueness of each record in the relation.

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

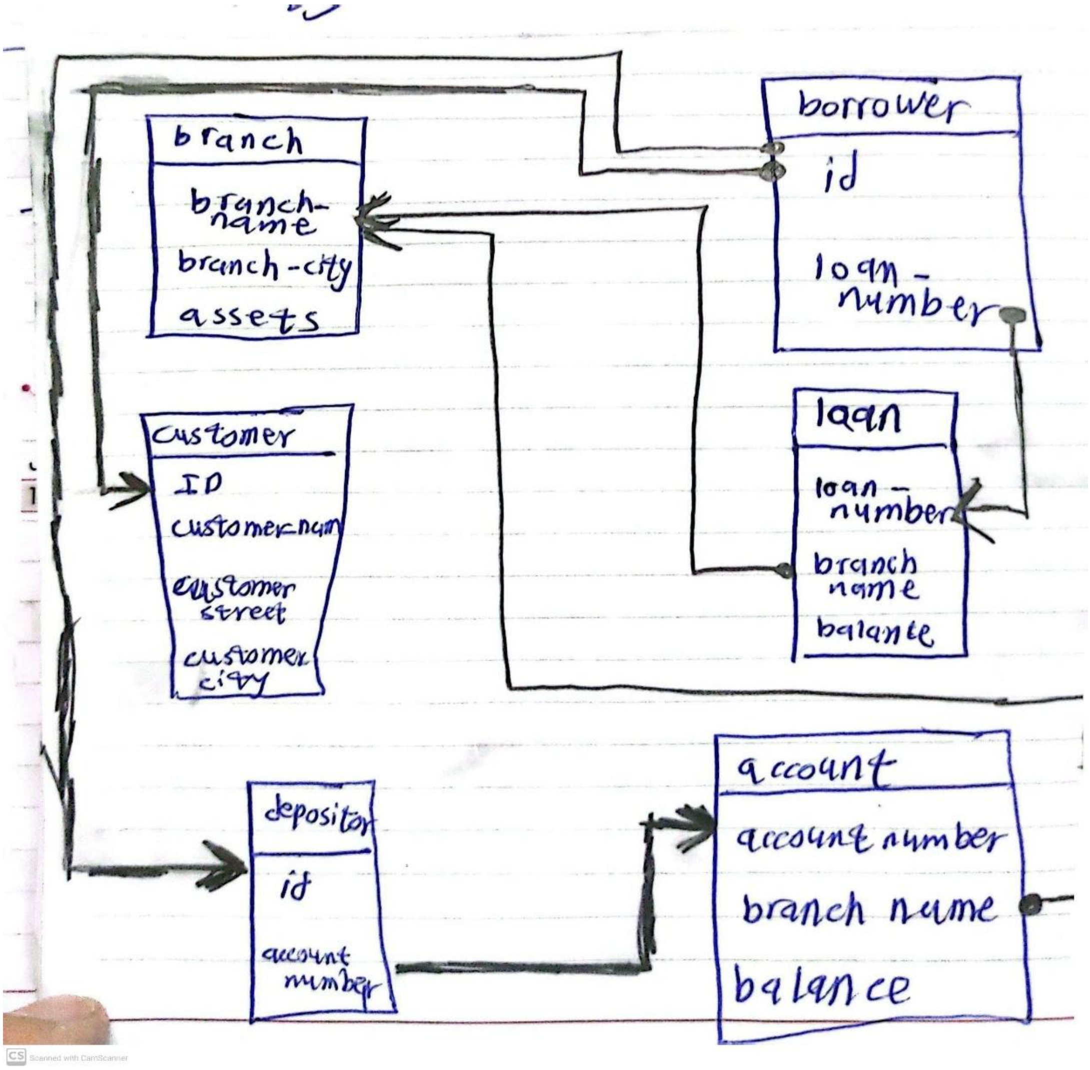
the only primary key that changes is for relation borrower and will become (id, loan number)

- **Primary Keys**:
  - branch: branch name
  - customer: ID
  - loan: loan number
  - borrower: (ID, loan number)
  - account: account number

- depositor: (ID, account number)

- **Foreign Keys**:
  - loan: branch name references branch(branch name)
  - borrower: ID references customer(ID), loan number references loan(loan number)
  - account: branch name references branch(branch name)
  - depositor: ID references customer(ID), account number references account(account number)

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



15.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 thesame city as the company for which she or he works.

2.14 چهارشنبه / دی 2025 8 Jan. ۱۴۴۶ ۷ رجب

a) $\Pi_{id, person\text{-}name}\left(\sigma_{company\text{-}name = "Big\text{-}Bank"}\left[employee \bowtie_{person\text{-}name} works\right]\right)$

b) $\Pi_{id, person\text{-}name, city}\left(\sigma_{company\text{-}name = "Big\text{-}Bank"}\left[employee \bowtie_{person\text{-}name} works\right]\right)$

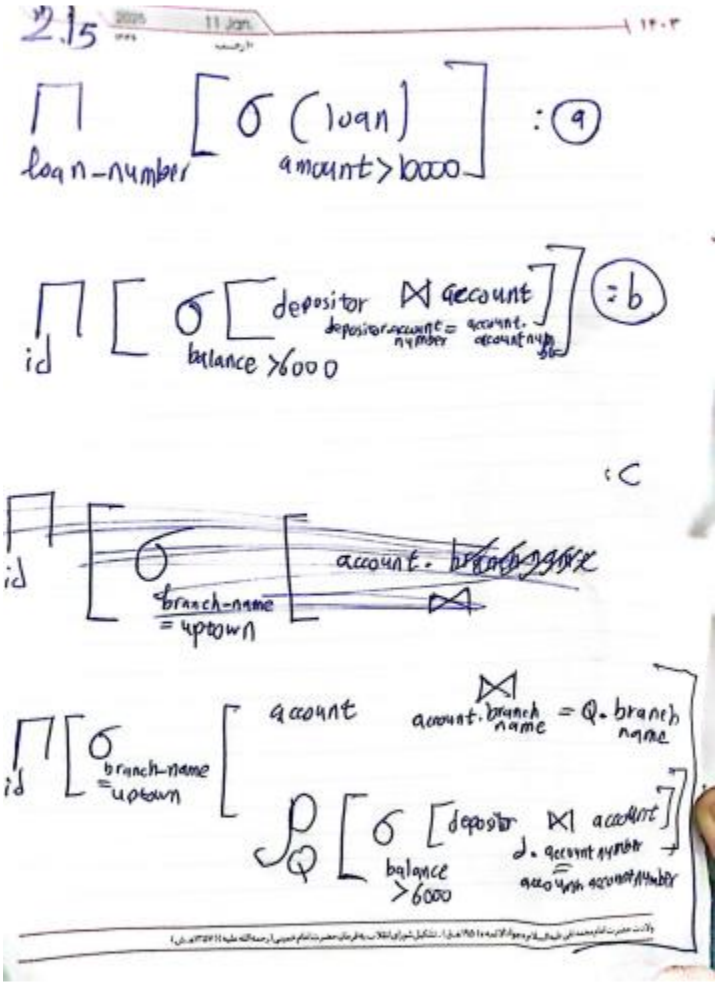
c) $\Pi_{id, name, street, city}\left(\sigma_{salary > 100000}\left[employee \bowtie_{person\text{-}name} works\right]\right)$

d $\Pi_{id, name}\left(\sigma_{company.city = employee.city}\left[employee \times company\right]\right)$

16. 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.



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

- **Missing or Unknown Data**:

  Null values are used to represent the absence of information or when data is unknown. For instance, if a customer does not provide their phone number or birthdate, the corresponding field in the database might be set to null to indicate that the information is missing.

- **Not Applicable Data**:

  Null values can be used when a particular attribute is not relevant for certain records. For example, if a table stores information about employees and their supervisor's ID, an employee who has no supervisor (e.g., a CEO or a solo contributor) might have a null value in the supervisor ID field because it is not applicable to them.

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

- **Imperative**: Focuses on **how** to achieve a result, offering fine control over state and performance. Best for system-level and performance-critical applications.
- **Functional**: Focuses on **what** to compute using pure functions and immutability, making it great for data transformations, concurrency, and reducing bugs related to mutable state.
- **Declarative**: Focuses on **what** needs to be done, abstracting the implementation details. Ideal for tasks like querying databases (SQL), web development (HTML), or configuration (YAML).

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.

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.

2.18:

a: $\Pi_{id,name}\left(\Pi_{\substack{depart\text{-}name \\ = physics}}\left(instructor \bowtie_{dept\,name} department\right)\right)$

b: $\Pi_{id,name}\left(\Pi_{\substack{building = \\ watson}}\left(instructor \bowtie_{department} department\right)\right)$

c: $Studen \bowtie_{id} \sigma_{\substack{course.department \\ name \\ = comp\text{-}sci}}\left(takes \bowtie_{id} course\right)$