Database Systems

Introduction to the Relational Model

Ishrar Nazah Chowdhury
Reg No. : 2019331104

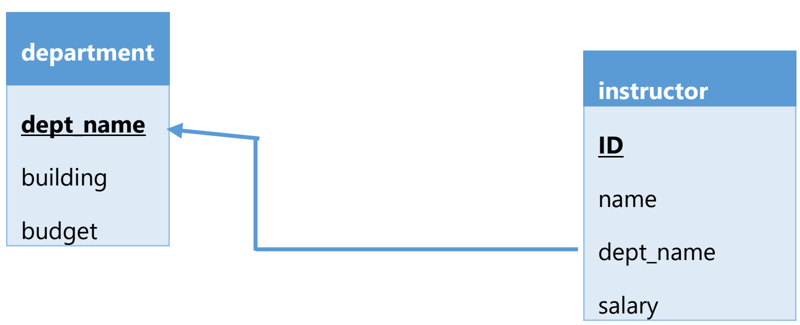
2.1

Primary key for:

- **Employee**: <u>person_name</u>
- **Works**: <u>person_name, company_name</u>
- **Company**: <u>company_name</u>

2.2

Examples of insert and deletes to the relations *instructor* and *department* that would cause violation of the foreign-key constraint are:



- Inserting a tuple into the *instructor* table with a **<u>dept_name</u>** that does not exist in the *department* table would violate the foreign-key constraint.
- Deleting a tuple from the *department* table, but there exists one or more tuples that contain that **<u>dept_name</u>** in the *instructor* table, which will now reference to a value that doesn't exist.

2.3

Each tuple in the relation represents a specific meeting time for a specific course, which must occur on a specific day at a specific start time. Therefore, the combination of **<u>day</u>** and **<u>start_time</u>** uniquely identifies each tuple in the relation since no two tuples can have the same day and start_time values for a given course.

On the other hand, the **end_time** is not part of the primary key because it can vary within a single day for a particular course, and different courses can have the same **end_time** for the same day and start_time. For example, two courses may have the same start_time and day but different **end_times** because they have different durations. Therefore, the end_time is not necessary for identifying a particular tuple in the *time_slot* relation and including it in the primary key would not ensure uniqueness.

2.4

It may be the case that two instructors have the same name, for new entries in the future. Thus, by using **<u>name</u>** as a superkey (or primary key) of *instructor,* we wouldn't be able to uniquely identify each tuple.

2.5

- Cartesian product of *student* and *advisor* creates a new table that has every possible pair of tuples from *student* and *advisor* tables.
- Performing the selection operation on the result will select only the tuples where **ID** of *student* is equal to the **s_id** of *advisor*. Here, **s_id** is the only primary key, so it is unique thus we get only one record from the *advisor* table for each record from the *student* table.

The resulting table will contain all the students along with their advisors.

2.6

employee
<u>person_name</u>
street
city

works
<u>person_name</u>
<u>company_name</u>
salary

company
<u>company_name</u>
city

a) $\pi_{person\_name}(\sigma_{city="Miami"}(employee))$

b) $\pi_{person\_name}(\sigma_{salary>100000}(works))$

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

2.7

branch
<u>branch_name</u>
branch_city
assets

borrower
<u>ID</u>
loan_number

loan
<u>loan_number</u>
branch_name
amount

a) $\pi_{branch\_name}(\sigma_{branch\_city="Chicago"}(branch))$

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

2.8

a) $\pi_{person\_name}(\sigma_{company\_name \neq "BigBank"}(works))$

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