# UCS310
# Database Management System

## Introduction to Relational Data Model

Lecture-05 Date: 23 Jan 2023

Dr. Sumit Sharma
Assistant Professor
Computer Science and Engineering Department
Thapar Institute of Engineering and Technology, Patiala

# Recap

- Introduction to the mathematical structure of relational model
  - Relational schema and instances
- Basic Terminologies
  - Attributes, degree, cardinality
- Keys of Relation
- Relational Model Constraints: Schema-Based
  - Domain
  - Key
  - Entity integrity
  - Referential integrity

# Keys of Relation

- Keys in DBMS is an attribute or set of attributes that help to identify a row (tuple) in a relation (table)
- Allow to find the relation between two tables
- Help to enforce identity and integrity in the relationship

- ❑ Primary Key
- ❑ Candidate Key
- ❑ Alternate Key
- ❑ Composite Key
- ❑ Foreign Key
- ❑ Super Key

# Keys

- Let, $\boldsymbol{K} \subseteq \boldsymbol{R}$
- $\boldsymbol{K}$ is a **superkey** of $\boldsymbol{R}$ if values for $\boldsymbol{K}$ are *sufficient to identify a unique tuple* of each possible relation $\boldsymbol{r}(\boldsymbol{R})$
  - Example: {ID} and {ID,name} are both superkeys of instructor.
- Superkey $\boldsymbol{K}$ is a **candidate key** if $\boldsymbol{K}$ is minimal
  - Example: {ID} is a candidate key for the Instructor
- One of the candidate keys is selected to be the **primary key**.
  - Which one?
- A **surrogate key** (or synthetic key) in a database is a unique identifier for either an entity in the modeled world or an object in the database
  - The surrogate key is not derived from the application data, unlike a natural (or business) key which is derived from the application data

# Keys

- **Superkey**: Roll#, (Roll#,DOB)
- **Candidate key**: Roll#, (First Name, Last Name), Passport#, Aadhar#
    - Passport # cannot be a key. Why?
- **Primary key:** : Roll#
- **Secondary/ Alternate Key**: (First Name, Last Name), Aadhar#
- **Simple key**: consists of a single attribute
- **Composite Key**: (First Name, Last Name)
    - Consists of more than one attribute to uniquely identify an entity occurrence
    - One or more attributes, which make up the key, are not simple keys in their own right

| Roll# | First Name | Last Name | DOB | Passport# | Aadhar# | Department |
|---|---|---|---|---|---|---|
| CS2022001 | Prateek | Keserwani | 07-Nov-1994 | L4034654 | 1728-6174-9239 | Computer |
| EC2022111 | Apeksha | Aggarwal | 04-Dec-1998 | NULL | 4388-2185-1248 | Electronic |
| MA2022213 | Prateek | Sharma | 19-May-1997 | NULL | 1236-4583-0235 | Maths |
| EC2022420 | Amit | Aggarwal | 19-Sept-1998 | M8816623 | 2365-5698-1470 | Electronic |

# Keys

- **Foreign key** constraint: Value in one relation must appear in another
  - **Referencing** relation
    - Enrolment: Foreign Key- Roll#, Course #
  - **Referenced** relation
    - Student, Course

***Students***

| Roll# | First Name | Last Name | DOB | Passport# | Aadhar# | Department |
|---|---|---|---|---|---|---|

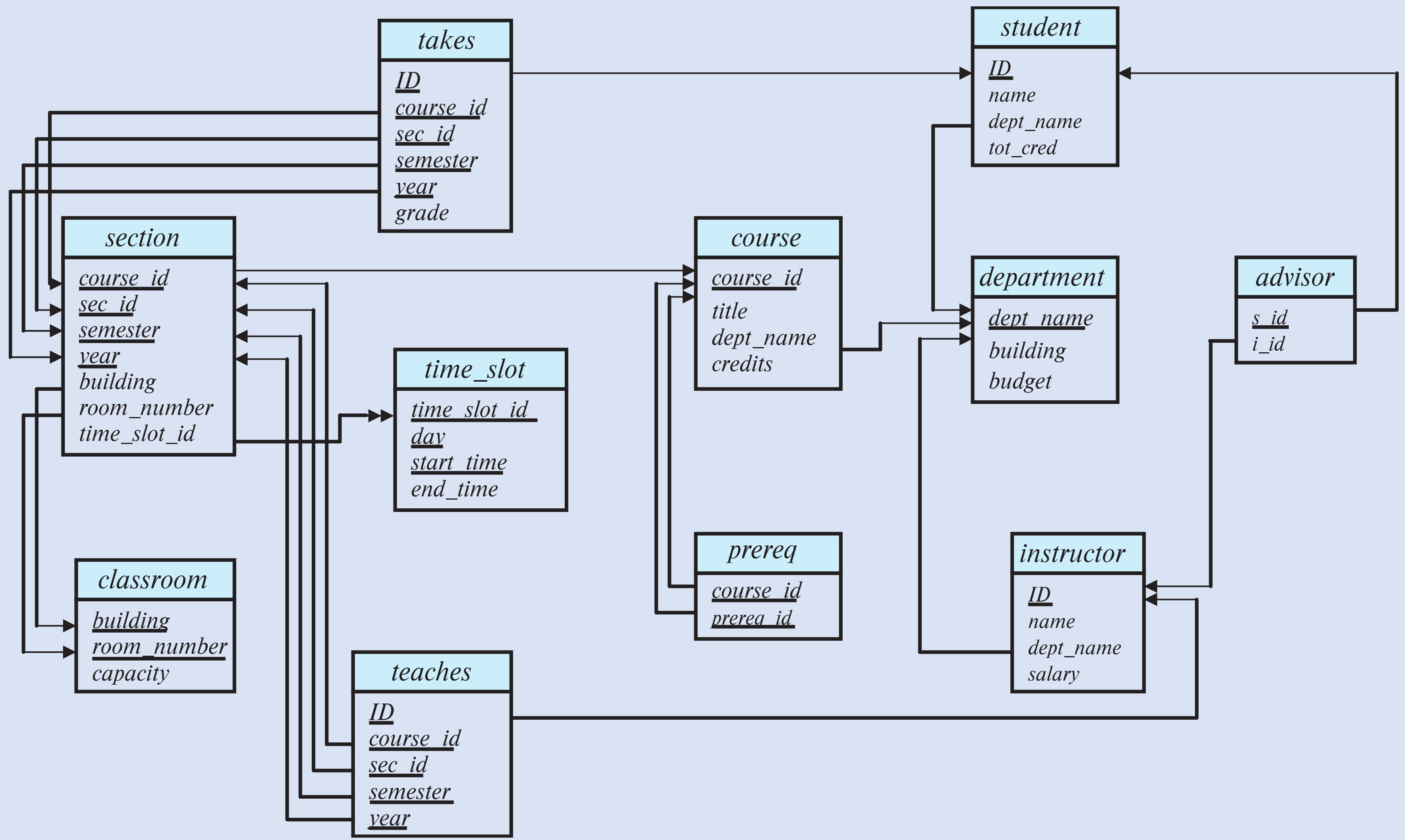
***Courses***

| Course# | Course Name | Credit | L-T-P | Department |
|---|---|---|---|---|

***Enrolment***

| Course# | Roll# | Instructor ID |
|---|---|---|

# Schema Diagram for University Database



# Relational Query Languages

- Procedural versus non-procedural, or declarative
- "Pure" languages:
  - Relational algebra - Procedural
  - Tuple relational calculus – Non- Procedural
  - Domain relational calculus – Non- Procedural
- The above 3 pure languages are equivalent in computing power
- We will concentrate in this chapter on relational algebra
  - Not Turing-machine equivalent
  - Consists of 6 basic operations

# Relational Algebra

- A procedural language consisting of a set of operations that take one or two relations as input and produce a new relation as their result
- Six basic operators
  - select: $\sigma$
  - project: $\prod$
  - union: $\cup$
  - set difference: –
  - Cartesian product: $\times$
  - rename: $\rho$

# Select Operation – selection of rows (tuples)

- Relation r

| A | B | C | D |
|---|---|---|---|
| α | α | 1 | 7 |
| α | β | 5 | 7 |
| β | β | 12 | 3 |
| β | β | 23 | 10 |

- $\sigma_{A=B \land D>5}(r)$

| A | B | C | D |
|---|---|---|---|
| α | α | 1 | 7 |
| β | β | 23 | 10 |

# Select Operation

- The **select** operation selects tuples that satisfy a given predicate.
- Notation: $\sigma_p(r)$
- $p$ is called the **selection predicate**
- Example: select those tuples of the *instructor* relation where the instructor is in the "Physics" department
  - Query

$$\sigma_{dept\_name = \text{“Physics”}}(instructor)$$

  - Result

| *ID* | *name* | *dept_name* | *salary* |
|---|---|---|---|
| 22222 | Einstein | Physics | 95000 |
| 33456 | Gold | Physics | 87000 |

# Select Operation

- We allow comparisons using

  $=, \neq, >, \geq. <. \leq$

  in the selection predicate.
- We can combine several predicates into a larger predicate by using the connectives:

  $\wedge$ **(and)**, $\vee$ **(or)**, $\neg$ **(not)**
- Example: Find the instructors in Physics with a salary greater $90,000, we write:

  $\sigma_{dept\_name = \text{“Physics”} \wedge salary > 90{,}000}\ (instructor)$
- The select predicate may include comparisons between two attributes
  - Example, find all departments whose name is the same as their building name:
  - $\sigma_{dept\_name = building}\ (department)$

# Project Operation

- A unary operation that returns its argument relation, with certain attributes left out
- Notation:

$$\Pi_{A_1, A_2, A_3 \ldots . A_k}(r)$$

where $A_1, A_2, \ldots, A_k$ are attribute names and $r$ is a relation name

- The result is defined as the relation of $k$ columns obtained by erasing the columns that are not listed
- Duplicate rows removed from result since relations are sets

# Project Operation Example

- Example: eliminate the *dept_name* attribute of *instructor*
- Query:

$$\Pi_{ID,\ name,\ salary}\ (instructor)$$

- Result:

| *ID* | *name* | *salary* |
|---|---|---|
| 10101 | Srinivasan | 65000 |
| 12121 | Wu | 90000 |
| 15151 | Mozart | 40000 |
| 22222 | Einstein | 95000 |
| 32343 | El Said | 60000 |
| 33456 | Gold | 87000 |
| 45565 | Katz | 75000 |
| 58583 | Califieri | 62000 |
| 76543 | Singh | 80000 |
| 76766 | Crick | 72000 |
| 83821 | Brandt | 92000 |
| 98345 | Kim | 80000 |

# Composition of Relational Operations

- The result of a relational-algebra operation is relation; therefore, relational-algebra operations can be composed together into a **relational-algebra expression**

- Consider the query -- Find the names of all instructors in the Physics department.

$$\Pi_{name}(\sigma_{dept\_name = \text{“Physics”}}(instructor))$$

- Instead of giving the name of a relation as the argument of the projection operation, we give an expression that evaluates a relation

# Cartesian-Product Operation

- The Cartesian-product operation (denoted by X) allows us to combine information from any two relations.
- Example: the Cartesian product of the relations *instructor* and t*eaches* is written as:

  *instructor* X *teaches*

- We construct a tuple of the result out of each possible pair of tuples: one from the *instructor* relation and one from the *teaches* relation (see next slide)
- Since the instructor *ID* appears in both relations we distinguish between these attributes by attaching to the attribute the name of the relation from which the attribute originally came.
  - *instructor.ID*
  - *teaches.ID*

# The *instructor* × *teaches* table

| *instructor.ID* | *name* | *dept_name* | *salary* | *teaches.ID* | *course_id* | *sec_id* | *semester* | *year* |
|---|---|---|---|---|---|---|---|---|
| 10101 | Srinivasan | Comp. Sci. | 65000 | 10101 | CS-101 | 1 | Fall | 2017 |
| 10101 | Srinivasan | Comp. Sci. | 65000 | 10101 | CS-315 | 1 | Spring | 2018 |
| 10101 | Srinivasan | Comp. Sci. | 65000 | 10101 | CS-347 | 1 | Fall | 2017 |
| 10101 | Srinivasan | Comp. Sci. | 65000 | 12121 | FIN-201 | 1 | Spring | 2018 |
| 10101 | Srinivasan | Comp. Sci. | 65000 | 15151 | MU-199 | 1 | Spring | 2018 |
| 10101 | Srinivasan | Comp. Sci. | 65000 | 22222 | PHY-101 | 1 | Fall | 2017 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 12121 | Wu | Finance | 90000 | 10101 | CS-101 | 1 | Fall | 2017 |
| 12121 | Wu | Finance | 90000 | 10101 | CS-315 | 1 | Spring | 2018 |
| 12121 | Wu | Finance | 90000 | 10101 | CS-347 | 1 | Fall | 2017 |
| 12121 | Wu | Finance | 90000 | 12121 | FIN-201 | 1 | Spring | 2018 |
| 12121 | Wu | Finance | 90000 | 15151 | MU-199 | 1 | Spring | 2018 |
| 12121 | Wu | Finance | 90000 | 22222 | PHY-101 | 1 | Fall | 2017 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 15151 | Mozart | Music | 40000 | 10101 | CS-101 | 1 | Fall | 2017 |
| 15151 | Mozart | Music | 40000 | 10101 | CS-315 | 1 | Spring | 2018 |
| 15151 | Mozart | Music | 40000 | 10101 | CS-347 | 1 | Fall | 2017 |
| 15151 | Mozart | Music | 40000 | 12121 | FIN-201 | 1 | Spring | 2018 |
| 15151 | Mozart | Music | 40000 | 15151 | MU-199 | 1 | Spring | 2018 |
| 15151 | Mozart | Music | 40000 | 22222 | PHY-101 | 1 | Fall | 2017 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 22222 | Einstein | Physics | 95000 | 10101 | CS-101 | 1 | Fall | 2017 |
| 22222 | Einstein | Physics | 95000 | 10101 | CS-315 | 1 | Spring | 2018 |
| 22222 | Einstein | Physics | 95000 | 10101 | CS-347 | 1 | Fall | 2017 |
| 22222 | Einstein | Physics | 95000 | 12121 | FIN-201 | 1 | Spring | 2018 |
| 22222 | Einstein | Physics | 95000 | 15151 | MU-199 | 1 | Spring | 2018 |
| 22222 | Einstein | Physics | 95000 | 22222 | PHY-101 | 1 | Fall | 2017 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |

# Join Operation

- The Cartesian-Product

  $$instructor \times teaches$$

  associates every tuple of instructor with every tuple of teaches.
  - Most of the resulting rows have information about instructors who did NOT teach a particular course.
- To get only those tuples of "$instructor \times teaches$" that pertain to instructors and the courses that they taught, we write:

  $$\sigma_{instructor.id = teaches.id} (instructor \times teaches))$$

  - We get only those tuples of "$instructor \times teaches$" that pertain to instructors and the courses that they taught.
- The result of this expression, shown in the next slide

# Join Operation

- The table corresponding to:

$$\sigma_{instructor.id = teaches.id}(instructor \times teaches))$$

| *instructor.ID* | *name* | *dept_name* | *salary* | *teaches.ID* | *course_id* | *sec_id* | *semester* | *year* |
|---|---|---|---|---|---|---|---|---|
| 10101 | Srinivasan | Comp. Sci. | 65000 | 10101 | CS-101 | 1 | Fall | 2017 |
| 10101 | Srinivasan | Comp. Sci. | 65000 | 10101 | CS-315 | 1 | Spring | 2018 |
| 10101 | Srinivasan | Comp. Sci. | 65000 | 10101 | CS-347 | 1 | Fall | 2017 |
| 12121 | Wu | Finance | 90000 | 12121 | FIN-201 | 1 | Spring | 2018 |
| 15151 | Mozart | Music | 40000 | 15151 | MU-199 | 1 | Spring | 2018 |
| 22222 | Einstein | Physics | 95000 | 22222 | PHY-101 | 1 | Fall | 2017 |
| 32343 | El Said | History | 60000 | 32343 | HIS-351 | 1 | Spring | 2018 |
| 45565 | Katz | Comp. Sci. | 75000 | 45565 | CS-101 | 1 | Spring | 2018 |
| 45565 | Katz | Comp. Sci. | 75000 | 45565 | CS-319 | 1 | Spring | 2018 |
| 76766 | Crick | Biology | 72000 | 76766 | BIO-101 | 1 | Summer | 2017 |
| 76766 | Crick | Biology | 72000 | 76766 | BIO-301 | 1 | Summer | 2018 |
| 83821 | Brandt | Comp. Sci. | 92000 | 83821 | CS-190 | 1 | Spring | 2017 |
| 83821 | Brandt | Comp. Sci. | 92000 | 83821 | CS-190 | 2 | Spring | 2017 |
| 83821 | Brandt | Comp. Sci. | 92000 | 83821 | CS-319 | 2 | Spring | 2018 |
| 98345 | Kim | Elec. Eng. | 80000 | 98345 | EE-181 | 1 | Spring | 2017 |

# Join Operation

- The **join** operation allows us to combine a select operation and a Cartesian-Product operation into a single operation.
- Consider relations *r* (*R*) and *s* (*S*)
- Let "theta" be a predicate on attributes in the schema R "union" S. The join operation $r \bowtie_\theta s$ is defined as follows:

$$r \bowtie_\theta s = \sigma_\theta (r \times s)$$

- Thus

$$\sigma_{instructor.id = teaches.id} (instructor \text{ x } teaches ))$$

- Can equivalently be written as

$$instructor \bowtie_{Instructor.id = teaches.id} teaches.$$

# Union Operation

- The union operation allows us to combine two relations
- Notation: $r \cup s$
- For $r \cup s$ to be valid.
  1. *r, s must have the same* ***arity*** *(same number of attributes)*
  2. *The attribute domains must be* ***compatible*** *(example: 2nd column of r deals with the same type of values as does the 2nd column of s)*
- Example: to find all courses taught in the Fall 2017 semester, or in the Spring 2018 semester, or in both

$$\Pi_{course\_id} (\sigma_{semester=\text{"Fall"}\ \Lambda\ year=2017} (section)) \cup$$

$$\Pi_{course\_id} (\sigma_{semester=\text{"Spring"}\ \Lambda\ year=2018} (section))$$

# Union Operation

- Result of:

$$\Pi_{course\_id}\,(\sigma_{\,semester=\text{“Fall”}\ \Lambda\ year=2017}(section))\ \cup$$
$$\Pi_{course\_id}\,(\sigma_{\,semester=\text{“Spring”}\ \Lambda\ year=2018}(section))$$

| *course_id* |
|---|
| CS-101 |
| CS-315 |
| CS-319 |
| CS-347 |
| FIN-201 |
| HIS-351 |
| MU-199 |
| PHY-101 |

# Set-Intersection Operation

- The set-intersection operation allows us to find tuples that are in both the input relations.
- Notation: $r \cap s$
- Assume:
  - $r$, $s$ have the *same arity*
  - attributes of $r$ and $s$ are compatible
- Example: Find the set of all courses taught in both the Fall 2017 and the Spring 2018 semesters.

$$\Pi_{course\_id} (\sigma_{semester=\text{“Fall”} \land year=2017} (section)) \cap$$
$$\Pi_{course\_id} (\sigma_{semester=\text{“Spring”} \land year=2018} (section))$$

  - Result

| *course_id* |
|---|
| CS-101 |

# Set Difference Operation

- The set-difference operation allows us to find tuples in one relation but not in another
- Notation $r - s$
- Set differences must be taken between **compatible** relations.
  - $r$ and $s$ must have the same arity
  - attribute domains of $r$ and $s$ must be compatible
- Example: to find all courses taught in the Fall 2017 semester, but not in the Spring 2018 semester

$$\Pi_{course\_id} (\sigma_{semester=\text{"Fall"} \land year=2017}(section)) -$$
$$\Pi_{course\_id} (\sigma_{semester=\text{"Spring"} \land year=2018}(section))$$

| *course_id* |
| --- |
| CS-347 |
| PHY-101 |

# The Assignment Operation

- It is convenient at times to write a relational algebra expression by assigning parts of it to temporary relation variables
- The assignment operation is denoted by ← and works like an assignment in a programming language

- Example: Find all instructor in the "Physics" and Music department

$$Physics \leftarrow \sigma_{dept\_name=\text{"Physics"}}(instructor)$$
$$Music \leftarrow \sigma_{dept\_name=\text{"Music"}}(instructor)$$
$$Physics \cup Music$$

- With the assignment operation, a query can be written as a sequential program consisting of a series of assignments followed by an expression whose value is displayed as the result of the query

# The Rename Operation

- The results of relational-algebra expressions do not have a name that we can use to refer to them. The rename operator, $\rho$, is provided for that purpose

- The expression:

$$\rho_x(E)$$

returns the result of expression $E$ under the name $x$

- Another form of the rename operation:

$$\rho_{x(A1, A2, \ldots An)}(E)$$

# Equivalent Queries

- There is more than one way to write a query in relational algebra
- Example: Find information about courses taught by instructors in the Physics department with salary greater than 90,000

- Query 1

$$\sigma_{dept\_name=\text{“Physics”} \wedge salary > 90{,}000}(instructor)$$

- Query 2

$$\sigma_{dept\_name=\text{“Physics”}}(\sigma_{salary > 90.000}(instructor))$$

- The two queries are not identical; they are, however, equivalent -- they give the same result on any database

# Equivalent Queries

- There is more than one way to write a query in relational algebra.
- Example: Find information about courses taught by instructors in the Physics department

- Query 1

$$\sigma_{dept\_name=\text{“Physics”}}(instructor \bowtie_{instructor.ID = teaches.ID} teaches)$$

- Query 2

$$(\sigma_{dept\_name=\text{“Physics”}}(instructor)) \bowtie_{instructor.ID = teaches.ID} teaches$$

- The two queries are not identical; they are, however, equivalent -- they give the same result on any database.

Thanks!