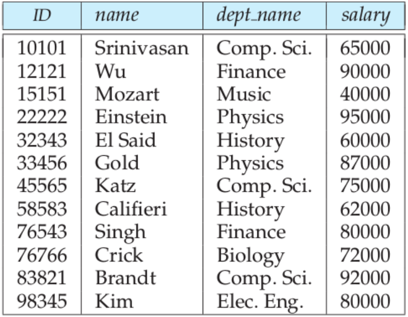
Database Management Systems (CSN-351)
Relational Model and SQL

BTech 3rd Year (CS) + Minor + Audit

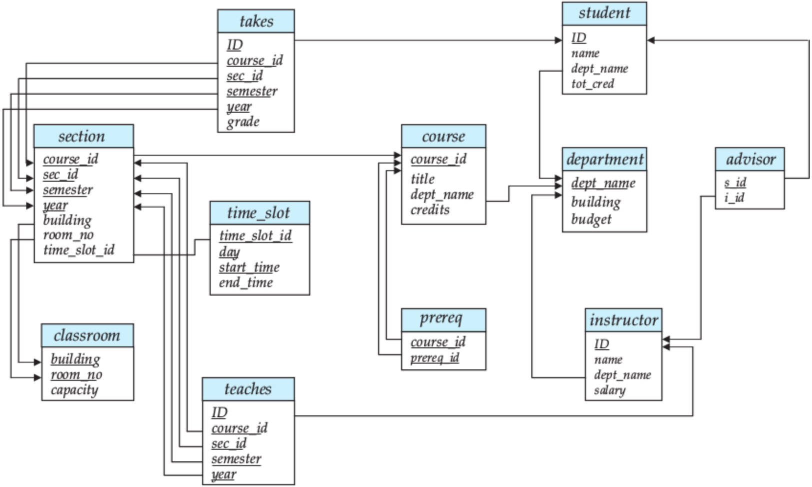
Instructor: **R**anita **B**iswas
Department of Computer Science and Engineering
Indian Institute of Technology Roorkee
Roorkee, Uttarakhand - 247 667, India

Example of a Relation



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

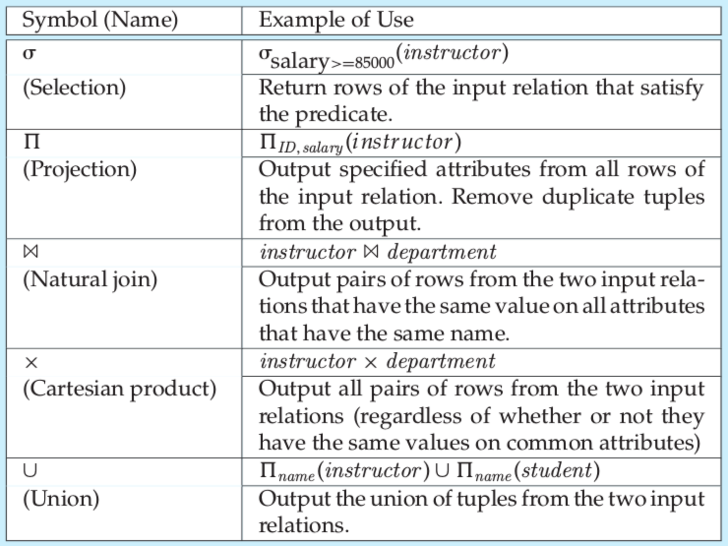
Schema Diagram for University Database



Relational Operations

- *Selection* of tuples
- Selection of attributes (*Projection*)
- Joining of two relations (*Cartesian Product*)
- *Set union* of two relations
- *Set difference* of two relations
- *Set intersection* of two relations
- *Natural join* of two relations

Symbols in Relational Algebra



Symbol (Name)	Example of Use
σ	$\sigma_{\text{salary}>=85000}(instructor)$
(Selection)	Return rows of the input relation that satisfy the predicate.
Π	$\Pi_{ID,\,salary}(instructor)$
(Projection)	Output specified attributes from all rows of the input relation. Remove duplicate tuples from the output.
$\bowtie$	$instructor \bowtie department$
(Natural join)	Output pairs of rows from the two input relations that have the same value on all attributes that have the same name.
$\times$	$instructor \times department$
(Cartesian product)	Output all pairs of rows from the two input relations (regardless of whether or not they have the same values on common attributes)
$\cup$	$\Pi_{name}(instructor) \cup \Pi_{name}(student)$
(Union)	Output the union of tuples from the two input relations.

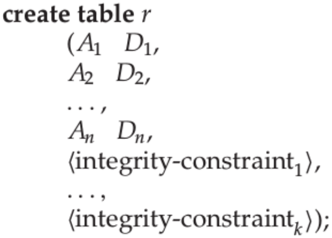
Structured Query Language (SQL)

- Data-definition language (DDL) — defining relation schemas, deleting relations, and modifying relation schemas
- Data-manipulation language (DML) — query information from the database and to insert tuples into, delete tuples from, and modify tuples in the database

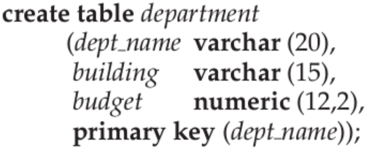
Domain Types in SQL

- **char(n)**: A fixed-length character string with user-specified length n. The full form, character, can be used instead.
- **varchar(n)**: A variable-length character string with user-specified maximum length n. The full form, character varying, is equivalent.
- **int**: An integer (a finite subset of the integers that is machine dependent). The full form, integer, is equivalent.
- **smallint**: A small integer (a machine-dependent subset of the integer type).
- **numeric(p, d)**: A fixed-point number with user-specified precision. The number consists of p digits (plus a sign), and d of the p digits are to the right of the decimal point. Thus, numeric(3,1) allows 44.5 to be stored exactly, but neither 444.5 or 0.32 can be stored exactly in a field of this type.
- **real, double precision**: Floating-point and double-precision floating-point numbers with machine-dependent precision.
- **float(n)**: A floating-point number, with precision of at least n digits.

Create Table

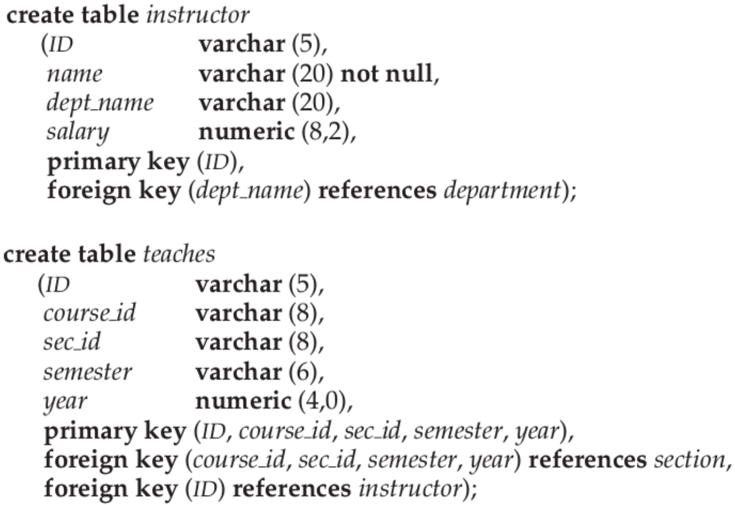


create table *r*
    (A_1 D_1,
    A_2 D_2,
    . . . ,
    A_n D_n,
    ⟨integrity-constraint$_1$⟩,
    . . . ,
    ⟨integrity-constraint$_k$⟩);



create table *department*
    (*dept_name* **varchar** (20),
    building **varchar** (15),
    budget **numeric** (12,2),
    primary key (*dept_name*));

Create Table (contd.)

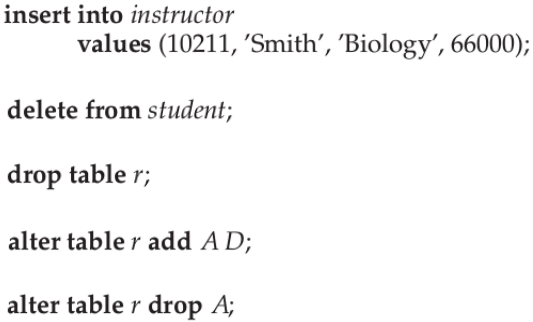


```
create table instructor
    (ID             varchar (5),
     name           varchar (20) not null,
     dept_name      varchar (20),
     salary         numeric (8,2),
     primary key (ID),
     foreign key (dept_name) references department);

create table teaches
    (ID             varchar (5),
     course_id      varchar (8),
     sec_id         varchar (8),
     semester       varchar (6),
     year           numeric (4,0),
     primary key (ID, course_id, sec_id, semester, year),
     foreign key (course_id, sec_id, semester, year) references section,
     foreign key (ID) references instructor);
```

Data-Manipulation Language

- *Query* information from the database
- *Insert* tuples into the database
- *Delete* tuples from the database
- *Modify* tuples in the database

Insert, Delete, Drop, Alter

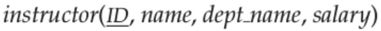


```
insert into instructor
        values (10211, 'Smith', 'Biology', 66000);

delete from student;

drop table r;

alter table r add A D;

alter table r drop A;
```

Queries on a Single Relation

instructor($\underline{\textit{ID}}$, *name*, *dept_name*, *salary*)

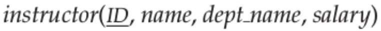
Queries on a Single Relation



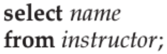
$instructor(\underline{ID}, name, dept\_name, salary)$

- Find the names of all instructors.

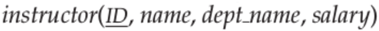
Queries on a Single Relation


instructor(*ID*, *name*, *dept_name*, *salary*)

- Find the names of all instructors.

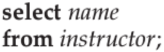

```
select name
from instructor;
```

Queries on a Single Relation



instructor(*ID*, *name*, *dept_name*, *salary*)

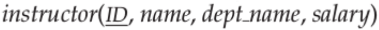
- Find the names of all instructors.



```
select name
from instructor;
```

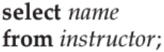
- Find the department names of all instructors.

Queries on a Single Relation



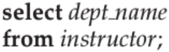
instructor(<u>*ID*</u>, *name*, *dept_name*, *salary*)

- Find the names of all instructors.



```
select name
from instructor;
```

- Find the department names of all instructors.



```
select dept_name
from instructor;
```

Queries on a Single Relation

instructor(<u>*ID*</u>, *name*, *dept_name*, *salary*)

- Find the names of all instructors.

```
select name
from instructor;
```

- Find the department names of all instructors.

```
select dept_name
from instructor;
```

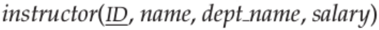
```
select distinct dept_name
from instructor;
```

Queries on a Single Relation (contd.)

$$instructor(\underline{ID}, name, dept\_name, salary)$$

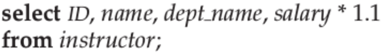
- Find the information about all instructors and show their salaries with a hike of 10%.

Queries on a Single Relation (contd.)



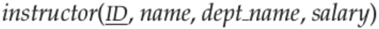
instructor(ID, *name*, *dept_name*, *salary*)

- Find the information about all instructors and show their salaries with a hike of 10%.



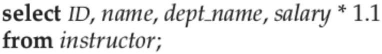
```
select ID, name, dept_name, salary * 1.1
from instructor;
```

Queries on a Single Relation (contd.)



instructor(<u>*ID*</u>, *name*, *dept_name*, *salary*)

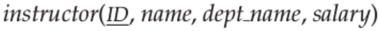
- Find the information about all instructors and show their salaries with a hike of 10%.



```
select ID, name, dept_name, salary * 1.1
from instructor;
```

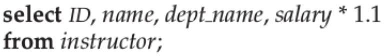
- Find the names of all instructors in the Computer Science department who have salary greater than 70,000.

Queries on a Single Relation (contd.)



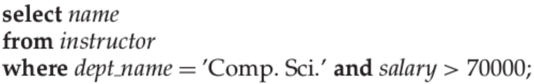
instructor(*ID*, *name*, *dept_name*, *salary*)

- Find the information about all instructors and show their salaries with a hike of 10%.



```
select ID, name, dept_name, salary * 1.1
from instructor;
```

- Find the names of all instructors in the Computer Science department who have salary greater than 70,000.



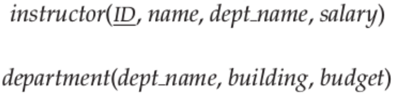
```
select name
from instructor
where dept_name = 'Comp. Sci.' and salary > 70000;
```

Queries on Multiple Relations

instructor(ID, name, dept_name, salary)

department(dept_name, building, budget)

Queries on Multiple Relations



instructor(ID, name, dept_name, salary)

department(dept_name, building, budget)

- Retrieve the names of all instructors, along with their department names and department building name.

Queries on Multiple Relations

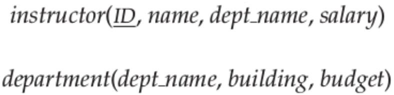
instructor(*ID*, *name*, *dept_name*, *salary*)

department(*dept_name*, *building*, *budget*)

- Retrieve the names of all instructors, along with their department names and department building name.

```
select name, instructor.dept_name, building
from instructor, department
```

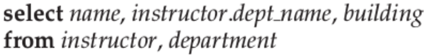
Queries on Multiple Relations



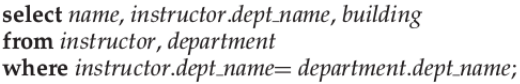
instructor(*ID*, *name*, *dept_name*, *salary*)

department(*dept_name*, *building*, *budget*)

- Retrieve the names of all instructors, along with their department names and department building name.



```
select name, instructor.dept_name, building
from instructor, department
```

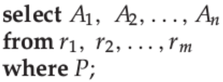


```
select name, instructor.dept_name, building
from instructor, department
where instructor.dept_name= department.dept_name;
```

The Clauses

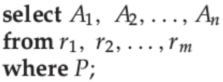
- The **select** clause is used to list the attributes desired in the result of a query.
- The **from** clause is a list of the relations to be accessed in the evaluation of the query.
- The **where** clause is a predicate involving attributes of the relation in the from clause.

The General Form

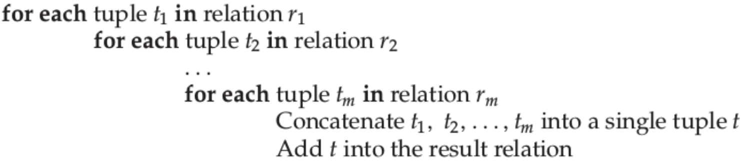


select $A_1, A_2, \ldots, A_n$
from $r_1, r_2, \ldots, r_m$
where P;

The General Form

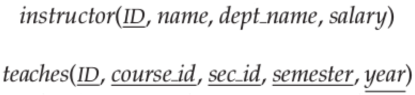


```
select $A_1$, $A_2$, ..., $A_n$
from $r_1$, $r_2$, ..., $r_m$
where $P$;
```



```
for each tuple $t_1$ in relation $r_1$
    for each tuple $t_2$ in relation $r_2$
        ...
        for each tuple $t_m$ in relation $r_m$
            Concatenate $t_1$, $t_2$, ..., $t_m$ into a single tuple $t$
            Add $t$ into the result relation
```

Example

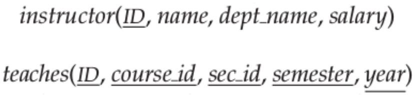


$\mathit{instructor}(\underline{\mathit{ID}}, \mathit{name}, \mathit{dept\_name}, \mathit{salary})$

$\mathit{teaches}(\underline{\mathit{ID}}, \underline{\mathit{course\_id}}, \underline{\mathit{sec\_id}}, \underline{\mathit{semester}}, \underline{\mathit{year}})$

- For all instructors in the Computer Science department who have taught some course, find their names and the course ID of all courses they taught.

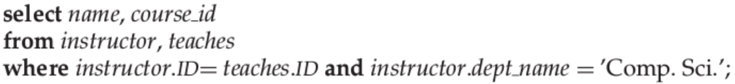
Example



instructor(*ID*, *name*, *dept_name*, *salary*)

teaches(*ID*, *course_id*, *sec_id*, *semester*, *year*)

- For all instructors in the Computer Science department who have taught some course, find their names and the course ID of all courses they taught.



```
select name, course_id
from instructor, teaches
where instructor.ID= teaches.ID and instructor.dept_name = 'Comp. Sci.';
```

Steps of Execution

- Generate a Cartesian product of the relations listed in the **from** clause
- Apply the predicates specified in the **where** clause on the result of Step 1.
- For each tuple in the result of Step 2, output the attributes (or results of expressions) specified in the **select** clause.

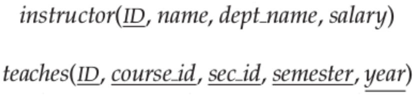
Natural Join

$$\textit{instructor}(\underline{\textit{ID}}, \textit{name}, \textit{dept\_name}, \textit{salary})$$

$$\textit{teaches}(\underline{\textit{ID}}, \underline{\textit{course\_id}}, \underline{\textit{sec\_id}}, \underline{\textit{semester}}, \underline{\textit{year}})$$

- For all instructors in the university who have taught some course, find their names and the course ID of all courses they taught.

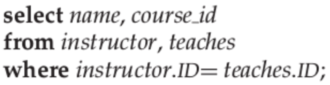
Natural Join



instructor(*ID*, *name*, *dept_name*, *salary*)

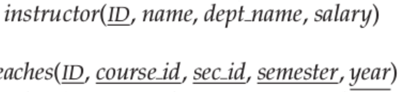
teaches(*ID*, *course_id*, *sec_id*, *semester*, *year*)

- For all instructors in the university who have taught some course, find their names and the course ID of all courses they taught.



```
select name, course_id
from instructor, teaches
where instructor.ID= teaches.ID;
```

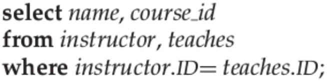
Natural Join



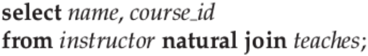
$\textit{instructor}(\underline{\textit{ID}}, \textit{name}, \textit{dept\_name}, \textit{salary})$

$\textit{teaches}(\underline{\textit{ID}}, \underline{\textit{course\_id}}, \underline{\textit{sec\_id}}, \underline{\textit{semester}}, \underline{\textit{year}})$

- For all instructors in the university who have taught some course, find their names and the course ID of all courses they taught.

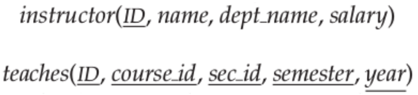


```
select name, course_id
from instructor, teaches
where instructor.ID= teaches.ID;
```



```
select name, course_id
from instructor natural join teaches;
```

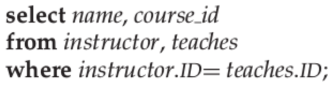
Natural Join



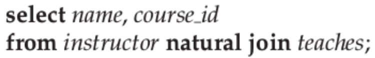
instructor(*ID*, *name*, *dept_name*, *salary*)

teaches(*ID*, *course_id*, *sec_id*, *semester*, *year*)

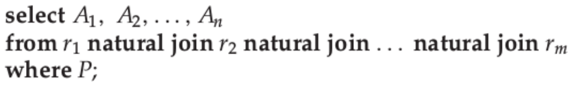
- For all instructors in the university who have taught some course, find their names and the course ID of all courses they taught.



select *name*, *course_id*
from *instructor*, *teaches*
where *instructor.ID*= *teaches.ID*;

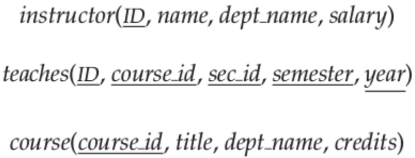


select *name*, *course_id*
from *instructor* **natural join** *teaches*;



select A_1, A_2, ..., A_n
from r_1 **natural join** r_2 **natural join** ... **natural join** r_m
where P;

Natural Join (contd.)



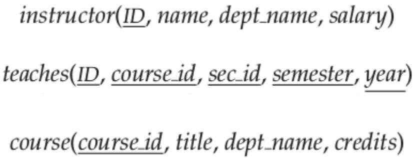
instructor(ID, name, dept_name, salary)

teaches(ID, course_id, sec_id, semester, year)

course(course_id, title, dept_name, credits)

- List the names of instructors along with the the titles of courses that they teach.

Natural Join (contd.)

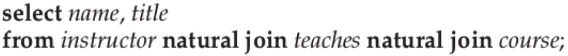


instructor(*ID*, *name*, *dept_name*, *salary*)

teaches(*ID*, *course_id*, *sec_id*, *semester*, *year*)

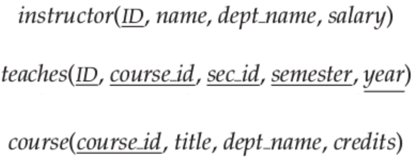
course(*course_id*, *title*, *dept_name*, *credits*)

- List the names of instructors along with the the titles of courses that they teach.



select *name*, *title*
from *instructor* **natural join** *teaches* **natural join** *course*;

Natural Join (contd.)

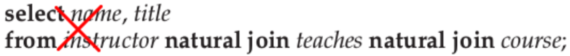


instructor(*ID*, *name*, *dept_name*, *salary*)

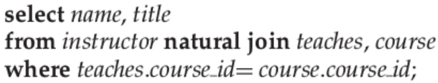
teaches(*ID*, *course_id*, *sec_id*, *semester*, *year*)

course(*course_id*, *title*, *dept_name*, *credits*)

- List the names of instructors along with the titles of courses that they teach.

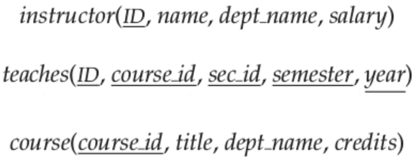


```
select name, title
from instructor natural join teaches natural join course;
```



```
select name, title
from instructor natural join teaches, course
where teaches.course_id= course.course_id;
```

Natural Join (contd.)

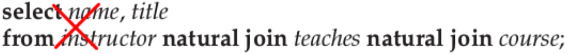


instructor(ID, *name*, *dept_name*, *salary*)

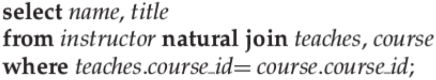
teaches(ID, *course_id*, *sec_id*, *semester*, *year*)

course(*course_id*, *title*, *dept_name*, *credits*)

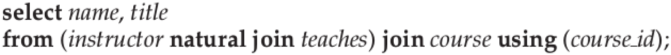
- List the names of instructors along with the titles of courses that they teach.



```
select name, title
from instructor natural join teaches natural join course;
```



```
select name, title
from instructor natural join teaches, course
where teaches.course_id= course.course_id;
```



```
select name, title
from (instructor natural join teaches) join course using (course_id);
```