Database Management Systems (CSN-351)
RA, TRC, Intermediate SQL

BTech 3rd Year (CS) + Minor

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

Revisiting Relational Algebra

Select	σ	Project	Π
Union	$\cup$	Intersection	$\cap$
Set-difference	$-$	Division	$\div$
Cartesian Product	$\times$	Natural Join	$\bowtie$

Revisiting Relational Algebra

Select	σ	Project	Π
Union	$\cup$	Intersection	$\cap$
Set-difference	$-$	Division	$\div$
Cartesian Product	$\times$	Natural Join	$\bowtie$

Left Semi-join	$\ltimes$	Right Semi-join	$\rtimes$
Left Anti-join	$\triangleright$	Right Anti-join	$\triangleleft$
Left Outer-join	$⟕$	Right Outer-join	$⟖$
Full Outer-join	$⟗$	Aggregation	$\mathcal{G}$

Tuple Relational Calculus

$$\{t \mid t \in instructor \wedge t[salary] > 80000\}$$

Tuple Relational Calculus

$$\{t \mid t \in instructor \wedge t[salary] > 80000\}$$

$$\{t \mid \exists s \in instructor\ (t[ID] = s[ID] \wedge s[salary] > 80000)\}$$

Tuple Relational Calculus

$$\{t \mid t \in instructor \wedge t[salary] > 80000\}$$

$$\{t \mid \exists s \in instructor\ (t[ID] = s[ID] \wedge s[salary] > 80000)\}$$

$$\{t \mid \exists s \in instructor\ (t[name] = s[name] \wedge \exists u \in department\ (u[dept\_name] = s[dept\_name] \wedge u[building] = \text{“Watson”}))\}$$

Tuple Relational Calculus

$$\{t \mid \exists s \in section\ (t[course\_id] = s[course\_id]) \\ \wedge s[semester] = \text{“Fall”} \wedge s[year] = 2009)\} \\ \vee \exists u \in section\ (u[course\_id] = t[course\_id])\} \\ \wedge u[semester] = \text{“Spring”} \wedge u[year] = 2010)\}$$

Tuple Relational Calculus

$$\{t \mid \exists s \in section\ (t[course\_id] = s[course\_id]) \\ \wedge s[semester] = \text{“Fall”} \wedge s[year] = 2009)\} \\ \vee \exists u \in section\ (u[course\_id] = t[course\_id])\} \\ \wedge u[semester] = \text{“Spring”} \wedge u[year] = 2010)\}$$

$$\{t \mid \exists s \in section\ (t[course\_id] = s[course\_id]) \\ \wedge s[semester] = \text{“Fall”} \wedge s[year] = 2009)\} \\ \wedge \exists u \in section\ (u[course\_id] = t[course\_id])\} \\ \wedge u[semester] = \text{“Spring”} \wedge u[year] = 2010)\}$$

Tuple Relational Calculus

$$\{t \mid \exists s \in section\ (t[course\_id] = s[course\_id])$$
$$\wedge\ s[semester] = \text{“Fall”} \wedge\ s[year] = 2009)\}$$
$$\wedge \neg \exists u \in section\ (u[course\_id] = t[course\_id])\}$$
$$\wedge\ u[semester] = \text{“Spring”} \wedge\ u[year] = 2010)\}$$

Tuple Relational Calculus

$$\{t \mid \exists s \in section\ (t[course\_id] = s[course\_id]) \\ \wedge s[semester] = \text{“Fall”} \wedge s[year] = 2009)\} \\ \wedge \neg \exists u \in section\ (u[course\_id] = t[course\_id])\} \\ \wedge u[semester] = \text{“Spring”} \wedge u[year] = 2010)\}$$

$$\{t \mid \exists r \in student\ (r[ID] = t[ID]) \wedge \\ (\forall u \in course\ (u[dept\_name] = \text{“Biology”} \Rightarrow \\ \exists s \in takes\ (t[ID] = s[ID] \\ \wedge s[course\_id] = u[course\_id]))\}$$

Joins in SQL

select *
from *student* **natural left outer join** *takes;*

Joins in SQL

```
select *
from student natural left outer join takes;
```

```
select *
from takes natural right outer join student;
```

Joins in SQL

```
select *
from student natural left outer join takes;
```

```
select *
from takes natural right outer join student;
```

```
select *
from (select *
      from student
      where dept_name= 'Comp. Sci')
     natural full outer join
     (select *
      from takes
      where semester = 'Spring' and year = 2009);
```

Joins in SQL

```
select *
from student left outer join takes on student.ID= takes.ID;
```

Joins in SQL

```
select *
from student left outer join takes on student.ID= takes.ID;
```

```
select *
from student left outer join takes on true
where student.ID= takes.ID;
```

Joins in SQL

```
select *
from student left outer join takes on student.ID= takes.ID;
```

```
select *
from student left outer join takes on true
where student.ID= takes.ID;
```

Join types
inner join
left outer join
right outer join
full outer join

Join conditions
natural
on <predicate>
using $(A_1, A_2, \ldots, A_n)$

Views in SQL

create view *faculty* **as**
select *ID*, *name*, *dept_name*
from *instructor*;

Views in SQL

```
create view faculty as
select ID, name, dept_name
from instructor;
```

```
create view physics_fall_2009 as
    select course.course_id, sec_id, building, room_number
    from course, section
    where course.course_id = section.course_id
                and course.dept_name = 'Physics'
                and section.semester = 'Fall'
                and section.year = '2009';
```

Views in SQL

```
create view faculty as
select ID, name, dept_name
from instructor;
```

```
create view physics_fall_2009 as
    select course.course_id, sec_id, building, room_number
    from course, section
    where course.course_id = section.course_id
                and course.dept_name = 'Physics'
                and section.semester = 'Fall'
                and section.year = '2009';
```

```
select course_id
from physics_fall_2009
where building= 'Watson';
```

Views in SQL

create view *departments_total_salary*(*dept_name*, *total_salary*) **as**
select *dept_name*, **sum** (*salary*)
from *instructor*
group by *dept_name*;

Views in SQL

```
create view departments_total_salary(dept_name, total_salary) as
    select dept_name, sum (salary)
    from instructor
    group by dept_name;
```

```
create view physics_fall_2009_watson as
                select course_id, room_number
                from physics_fall_2009
                where building= 'Watson';
```

Views in SQL

```
create view departments_total_salary(dept_name, total_salary) as
    select dept_name, sum (salary)
    from instructor
    group by dept_name;
```

```
create view physics_fall_2009_watson as
                select course_id, room_number
                from physics_fall_2009
                where building= 'Watson';
```

```
create view physics_fall_2009_watson as
            (select course_id, room_number
            from (select course.course_id, building, room_number
                from course, section
                where course.course_id = section.course_id
                        and course.dept_name = 'Physics'
                        and section.semester = 'Fall'
                        and section.year = '2009')
            where building= 'Watson';
```

Integrity Constraints in SQL

name **varchar**(20) **not null**
budget **numeric**(12,2) **not null**

Integrity Constraints in SQL

name **varchar**(20) **not null**
budget **numeric**(12,2) **not null**

unique $(A_{j_1}, A_{j_2}, \ldots, A_{j_m})$

Integrity Constraints in SQL

name **varchar**(20) **not null**
budget **numeric**(12,2) **not null**

unique $(A_{j_1}, A_{j_2}, \ldots, A_{j_m})$

```
create table section
    (course_id      varchar (8),
     sec_id         varchar (8),
     semester       varchar (6),
     year           numeric (4,0),
     building       varchar (15),
     room_number    varchar (7),
     time_slot_id   varchar (4),
     primary key (course_id, sec_id, semester, year),
     check (semester in ('Fall', 'Winter', 'Spring', 'Summer')));
```

Integrity Constraints in SQL

dept_name **varchar**(20) **references** *department*

Integrity Constraints in SQL

dept_name **varchar**(20) **references** *department*

create table *course*
(. . .
foreign key (*dept_name*) **references** *department*
on delete cascade
on update cascade,
. . .);

Integrity Constraints in SQL

dept_name **varchar**(20) **references** *department*

```
create table course
    ( . . .
    foreign key (dept_name) references department
                on delete cascade
                on update cascade,
    . . . );
```

```
create table student
      (ID           varchar (5),
       name         varchar (20) not null,
       dept_name    varchar (20),
       tot_cred     numeric (3,0) default 0,
       primary key (ID));
```

Integrity Constraints in SQL

```
create table classroom
    (building       varchar (15),
     room_number    varchar (7),
     capacity       numeric (4,0),
     primary key (building, room_number))

create table department
    (dept_name      varchar (20),
     building       varchar (15),
     budget         numeric (12,2) check (budget > 0),
     primary key (dept_name))

create table course
    (course_id      varchar (8),
     title          varchar (50),
     dept_name      varchar (20),
     credits        numeric (2,0) check (credits > 0),
     primary key (course_id),
     foreign key (dept_name) references department)
```

Integrity Constraints in SQL

create table *instructor*
(*ID* **varchar** (5),
name **varchar** (20), **not null**
dept_name **varchar** (20),
salary **numeric** (8,2), **check** (*salary* > 29000),
primary key (*ID*),
foreign key (*dept_name*) **references** *department*)

create table *section*
(*course_id* **varchar** (8),
sec_id **varchar** (8),
semester **varchar** (6), **check** (*semester* **in** ('Fall', 'Winter', 'Spring', 'Summer')),
year **numeric** (4,0), **check** (*year* > 1759 and *year* < 2100)
building **varchar** (15),
room_number **varchar** (7),
time_slot_id **varchar** (4),
primary key (*course_id*, *sec_id*, *semester*, *year*),
foreign key (*course_id*) **references** *course*,
foreign key (*building*, *room_number*) **references** *classroom*)