

RDBMS and SQL

Madhavan Mukund

<https://www.cmi.ac.in/~madhavan>

Lecture 7, 7 October 2026

Data definition in SQL

- Add constraints on values
 - Force non-`null` name
 - Minimum amount for `salary`

```
create table instructor
(ID varchar(5),
 name varchar(20) not null,
 dept_name varchar(20),
 salary numeric(8,2)
    check (salary > 29000)
)
```

<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>
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

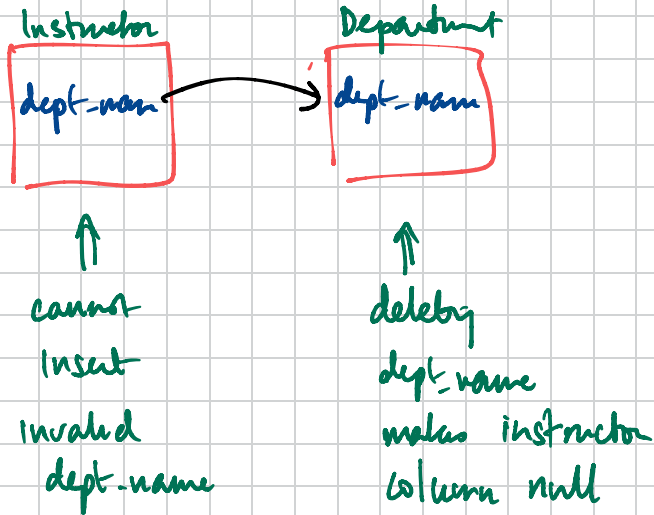
Data definition in SQL

- Add information about keys
 - `ID` is a primary key
 - `dept_name` is a foreign key

```
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 (dept_name))
```

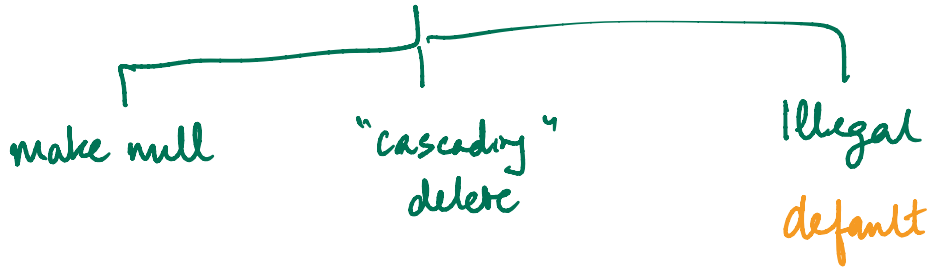
<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Foreign keys



foreign key constraint

└ what to do on deletion of foreign key



Querying in SQL — Set comparisons

Recall these examples from relational algebra.

- Find all faculty members from Physics who earn more than **at least one** faculty member from Comp.Sci.
- Find all faculty members from Physics who earn more than **every** faculty member from Comp.Sci.

"Quantifiers" $\exists x$
 $\forall x$

<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>
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

instructor

Set comparisons — some and all

- Find all faculty members from Physics who earn more than **at least one** faculty member from Comp.Sci.

<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>
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

instructor

Set comparisons — **some** and **all**

- Find all faculty members from Physics who earn more than **at least one** faculty member from Comp.Sci.

```
select name
  from instructor
   where dept_name = 'Physics'
        and salary > some (select salary
```

like
∃

```
  from instructor
   where dept_name = 'Comp. Sci.');
```

table of salaries
from CS

Set comparisons — **some** and **all**

- Find all faculty members from Physics who earn more than **at least one** faculty member from Comp.Sci.
- Find all faculty members from Physics who earn more than **every** faculty member from Comp.Sci.

<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>
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

instructor

Set comparisons — **some** and **all**

- Find all faculty members from Physics who earn more than **at least one** faculty member from Comp.Sci.
- Find all faculty members from Physics who earn more than **every** faculty member from Comp.Sci.

```
select name
  from instructor
   where dept_name = 'Physics'
        and salary > all (select salary
                           from instructor
                           where dept_name = 'Comp. Sci.');
```

Single column
is substitutable
for a value

CS salary column

✓

Test if a relation is empty — *exists*

- Find all faculty members from Physics who earn more than **at least one** faculty member from Comp.Sci.

<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>
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

instructor

Test if a relation is empty — *exists*

- Find all faculty members from Physics who earn more than **at least one** faculty member from Comp.Sci.
 - For each faculty member from Physics, check if the set of faculty members from Comp. Sci. who earn less is non-empty

<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>
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

instructor

Test if a relation is empty — `exists`

- Find all faculty members from Physics who earn more than **at least one** faculty member from Comp.Sci.
- For each faculty member from Physics, check if the set of faculty members from Comp. Sci. who earn less is non-empty

```
select name
  from instructor as I
    where dept_name = 'Physics'
      and exists (select salary
                  from instructor as J
                  where dept_name = 'Comp. Sci.'
                  and I.salary > J.salary);
```

Implementy $\exists, \forall$

all CS < Physics

Test if a relation is empty — *exists*

- Find all faculty members from Physics who earn more than *every* faculty member from Comp.Sci.

<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>
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

instructor

Test if a relation is empty — *exists*

- Find all faculty members from Physics who earn more than *every* faculty member from Comp.Sci.
 - For each faculty member from Physics, check if the set of faculty members from Comp. Sci. who more is empty

<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>
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

instructor

Test if a relation is empty — exists

- Find all faculty members from Physics who earn more than **every** faculty member from Comp.Sci.
- For each faculty member from Physics, check if the set of faculty members from Comp. Sci. who more is empty

```
select name
from instructor as I
  where dept_name = 'Physics'
     and not exists (select salary
                    from instructor as J
                     where dept_name = 'Comp. Sci.'
                        and I.salary < J.salary);
```

$$\neg \exists x \Rightarrow \forall x$$

$$\neg \exists x P \Rightarrow \forall x \neg P$$

$$CS > Physics$$

Null values

- `null` denotes a missing or unknown value

Null values

- `null` denotes a missing or unknown value
- Any arithmetic expression involving `null` result in `null`
 - $5 + \text{null} \rightsquigarrow \text{null}$

Null values

- `null` denotes a missing or unknown value
- Any arithmetic expression involving `null` result in `null`
 - $5 + \text{null} \leadsto \text{null}$
- Use `is null` to check for `null` values

```
select name
  from instructor
 where salary is null
```

In Python

$x == \text{None}$
or $x \text{ is } \text{None}$

Null values

- `null` denotes a missing or unknown value
- Any arithmetic expression involving `null` result in `null`
 - $5 + \text{null} \leadsto \text{null}$
- Use `is null` to check for `null` values
- Converse: `is not null`

```
select name
  from instructor
  where dept_name is not null
```

not (is null)

Null values

- `null` denotes a missing or unknown value
- Any arithmetic expression involving `null` result in `null`
 - $5 + \text{null} \leadsto \text{null}$
- Use `is null` to check for `null` values
- Converse: `is not null`
- Comparisons involving `null` result in `unknown` truth value
 - $5 < \text{null} \leadsto \text{unknown}$

Null values

- `null` denotes a missing or unknown value
- Any arithmetic expression involving `null` result in `null`
 - $5 + \text{null} \leadsto \text{null}$
- Use `is null` to check for `null` values
- Converse: `is not null`
- Comparisons involving `null` result in `unknown` truth value
 - $5 < \text{null} \leadsto \text{unknown}$
- Standard **shortcut evaluation** of `and`, `or`
 - `false and unknown` $\leadsto$ `false`
`true and unknown`, `unknown and unknown` $\leadsto$ `unknown`
 - `true or unknown` $\leadsto$ `true`
`false or unknown`, `unknown or unknown` $\leadsto$ `unknown`

if $\checkmark$ `c1` or `c2` **NOT CHECKED**
 $\xrightarrow{\text{L to R}}$

unknown and false?

valid index

if $i < \text{len}(l)$ and $l[i] > 100$:

Σ

if $i < \text{len}(l)$:

if $l[i] > 100$:

Σ

Joins in SQL

- Join — cartesian product combined with selection

*select * in X, Y
where --*

Joins in SQL

- Join — cartesian product combined with selection
- Three specific types of join
 - Inner join — default
 - Natural join
 - Outer join ?

NO
X

Joins in SQL

- Join — cartesian product combined with selection
- Three specific types of join
 - Inner join — default
 - Natural join
 - Outer join

- Join ~~on~~ (column names) *Using*

```
select *  
from course join prereq  
on (course_id)
```

implicitly Inner join
Same behaviour as natural join, only one copy of `course_id`

Joins in SQL

- Join — cartesian product combined with selection
- Three specific types of join
 - Inner join — default
 - Natural join
 - Outer join

- Join ~~en~~^{using} (column names)

```
select *  
  from course join prereq  
    on (course_id)
```

Same behaviour as natural join, only one copy of `course_id`

- Join ~~using~~^{on} condition

```
select *  
  from course join prereq  
on using course.course_id =  
    prereq.course_id
```

Same as ,

Like $\bowtie_{\theta}$, all columns are copied to output

Natural join in SQL

- Match tuples with the same values for *all* common attributes
- Retain only *one copy* of each common column.

Natural join in SQL

- Match tuples with the same values for *all* common attributes
- Retain only *one copy* of each common column.
- List the names of ~~instructors~~ ^{student} along with the course ID of the courses that they ~~taught~~ ^{took}

```
select name, course_id  
from student natural join takes;
```

Natural join in SQL

- Match tuples with the same values for *all* common attributes
- Retain only *one copy* of each common column.
- List the names of ~~instructors~~ along with the course ID of the courses that they ~~taught~~

```
select name, course_id  
from student natural join takes;
```

Natural join in SQL

- Match tuples with the same values for *all* common attributes
- Retain only *one copy* of each common column.
- List the names of ~~instructors~~ along with the course ID of the courses that they ~~taught~~

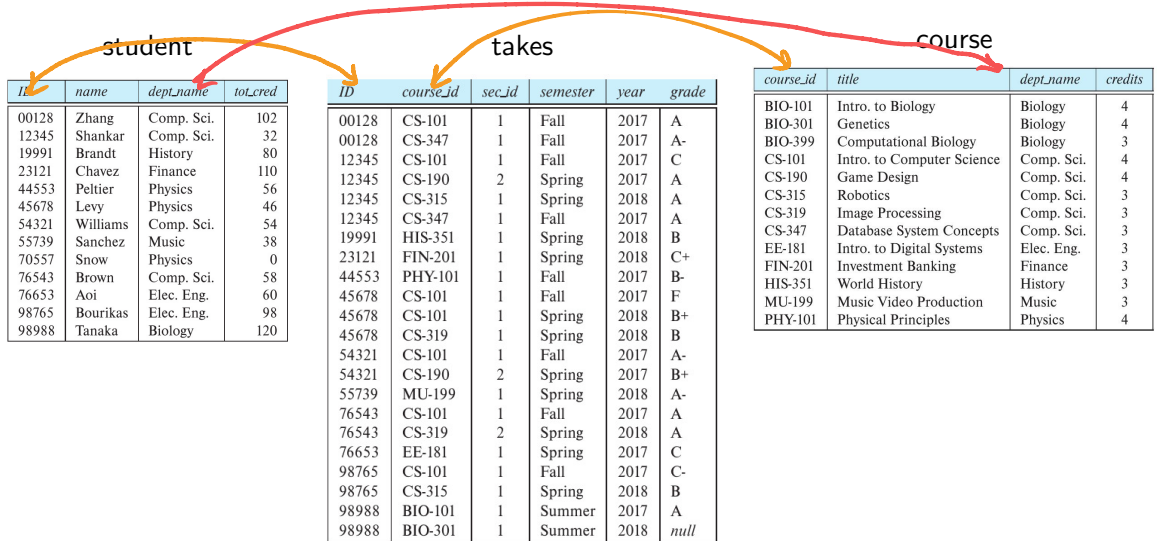
```
select name, course_id
  from student natural join takes;
```

- Can join multiple relations at a time

```
select A1, A2, ..., Am
  from r1 natural join r2
      natural join ...
      natural join rn
  where P ;
```

But be careful about what this means!

Natural join in SQL



Natural join in SQL

- Consider the following query

```
select name, title  
  from student natural join takes natural join course;
```

ID

course-id

Natural join in SQL

- Consider the following query

```
select name, title  
  from student natural join takes natural join course;
```

- dept_name occurs in student and course

Natural join in SQL

- Consider the following query

```
select name, title  
from student natural join takes natural join course;
```

- dept_name occurs in student and course
- Will only report courses that a student takes from their own department!

Natural join in SQL

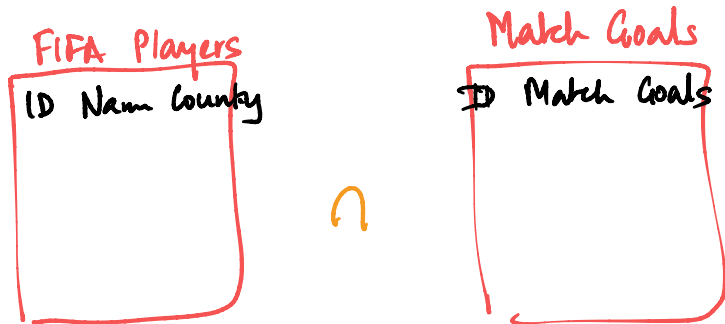
- Consider the following query

```
select name, title
  from student natural join takes natural join course;
```

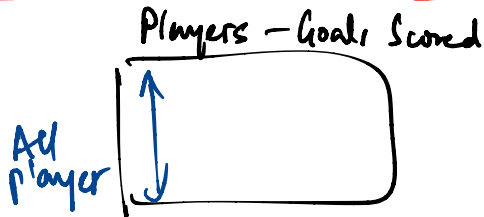
- dept_name occurs in student and course
 - Will only report courses that a student takes from their own department!
- The correct version of the query is

```
select name, title
  from student natural join takes, course
    where takes.course_id = course.course_id;
```

Outer join



Some players
have not
played / scored





Outer Join

- An extension of the join operation that avoids loss of information.
- Computes the join and then adds tuples from one relation that does not match tuples in the other relation to the result of the join.
- Uses *null* values.
- Three forms of outer join:
 - left outer join
 - right outer join
 - full outer join



Outer Join Examples

- Relation *course*

<i>course_id</i>	<i>title</i>	<i>dept_name</i>	<i>credits</i>
BIO-301	Genetics	Biology	4
CS-190	Game Design	Comp. Sci.	4
CS-315	Robotics	Comp. Sci.	3

- Relation *prereq*

<i>course_id</i>	<i>prereq_id</i>
BIO-301	BIO-101
CS-190	CS-101
CS-347	CS-101

- Observe that

course information is missing for CS-437 347

prereq information is missing for CS-315



Left Outer Join

- *course* **natural left outer join** *prereq*

<i>course_id</i>	<i>title</i>	<i>dept_name</i>	<i>credits</i>	<i>prereq_id</i>
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-101
CS-315	Robotics	Comp. Sci.	3	<i>null</i>

- In relational algebra: *course* ⋈ *prereq*





Right Outer Join

- *course* **natural right outer join** *prereq*

<i>course_id</i>	<i>title</i>	<i>dept_name</i>	<i>credits</i>	<i>prereq_id</i>
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-101
CS-347	<i>null</i>	<i>null</i>	<i>null</i>	CS-101

- In relational algebra: *course* ⋈ *prereq*





Full Outer Join

- *course* **natural full outer join** *prereq*

<i>course_id</i>	<i>title</i>	<i>dept_name</i>	<i>credits</i>	<i>prereq_id</i>
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-101
CS-315	Robotics	Comp. Sci.	3	<i>null</i>
CS-347	<i>null</i>	<i>null</i>	<i>null</i>	CS-101

- In relational algebra: *course* $\bowtie$ *prereq*



Joined Relations – Examples

- *course* **natural** right outer join *prereq*

<i>course_id</i>	<i>title</i>	<i>dept_name</i>	<i>credits</i>	<i>prereq_id</i>
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-101
CS-347	<i>null</i>	<i>null</i>	<i>null</i>	CS-101

- *course* full outer join *prereq* **using** (*course_id*)

<i>course_id</i>	<i>title</i>	<i>dept_name</i>	<i>credits</i>	<i>prereq_id</i>
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-101
CS-315	Robotics	Comp. Sci.	3	<i>null</i>
CS-347	<i>null</i>	<i>null</i>	<i>null</i>	CS-101



Joined Relations – Examples

- **course inner join prereq on**
course.course_id = prereq.course_id

<i>course_id</i>	<i>title</i>	<i>dept_name</i>	<i>credits</i>	<i>prereq_id</i>	<i>course_id</i>
BIO-301	Genetics	Biology	4	BIO-101	BIO-301
CS-190	Game Design	Comp. Sci.	4	CS-101	CS-190

- What is the difference between the above, and a natural join?
- **course left outer join prereq on**
course.course_id = prereq.course_id

<i>course_id</i>	<i>title</i>	<i>dept_name</i>	<i>credits</i>	<i>prereq_id</i>	<i>course_id</i>
BIO-301	Genetics	Biology	4	BIO-101	BIO-301
CS-190	Game Design	Comp. Sci.	4	CS-101	CS-190
CS-315	Robotics	Comp. Sci.	3	<i>null</i>	<i>null</i>

Views in SQL

- **Views** are virtual tables

Views in SQL

- **Views** are virtual tables
- Hide sensitive information from some users — hide salary

```
create view limited_instructor as  
  select id, name, dept_name  
  from instructor
```

Views in SQL

- **Views** are virtual tables
- Hide sensitive information from some users — hide salary

```
create view limited_instructor as
  select id, name, dept_name
  from instructor
```

- Create convenient “intermediate tables”

```
create view name_title as
  select instructor.name, course.title
  from instructor natural join teaches, course
  where teaches.course_id = course.course_id
```

Views in SQL

- **Views** are virtual tables
- Hide sensitive information from some users — hide salary

```
create view limited_instructor as
  select id, name, dept_name
  from instructor
```

- Create convenient “intermediate tables”

```
create view name_title as
  select instructor.name, course.title
  from instructor natural join teaches, course
  where teaches.course_id = course.course_id
```

- Views are recomputed when used to stay current

Views in SQL

- **Views** are virtual tables
- Hide sensitive information from some users — hide salary

```
create view limited_instructor as
  select id, name, dept_name
  from instructor
```

- Create convenient “intermediate tables”

```
create view name_title as
  select instructor.name, course.title
  from instructor natural join teaches, course
  where teaches.course_id = course.course_id
```

- Views are recomputed when used to stay current
- **Materialized views** — save recomputation, but danger of being out of date

Advanced SQL

- Many other features
 - Transactions
 - Assertions and triggers
 - ...

Advanced SQL

- Many other features
 - Transactions
 - Assertions and triggers
 - ...
- Can call SQL from other programming languages
 - Almost every language has library functions to invoke SQL
 - Transfer data between online forms and databases
 - ...

Security — SQL injection attacks

- User input can be malicious commands to corrupt database
- Always validate data entered in a form before passing on to SQL

Security — SQL injection attacks

- User input can be malicious commands to corrupt database
- Always validate data entered in a form before passing on to SQL

