

# SQL - Joins

Lecture By  
Binu Jasim  
01-Aug-2016

## Student

rollNo	name	dept	CGPA
123	Alice	CSE	8.2
201	Bob	EEE	5.6
399	Cherry	CSE	8.2

## Course

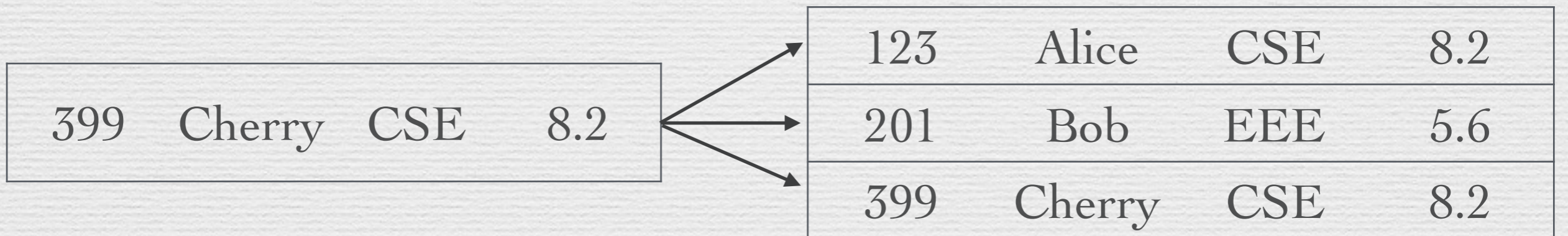
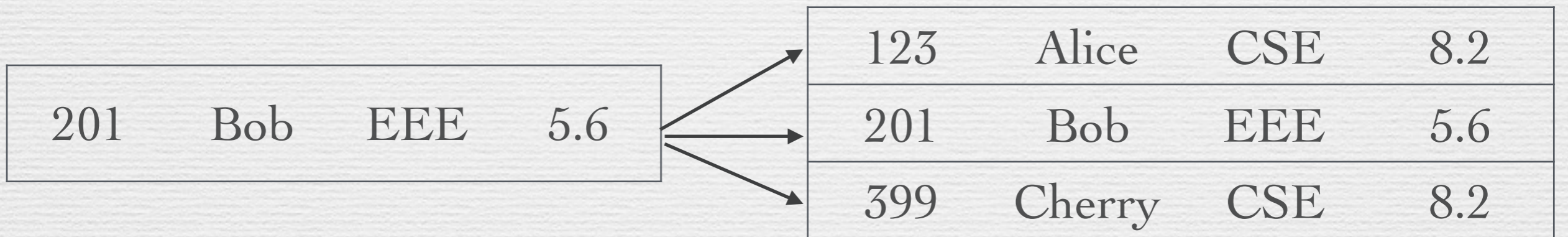
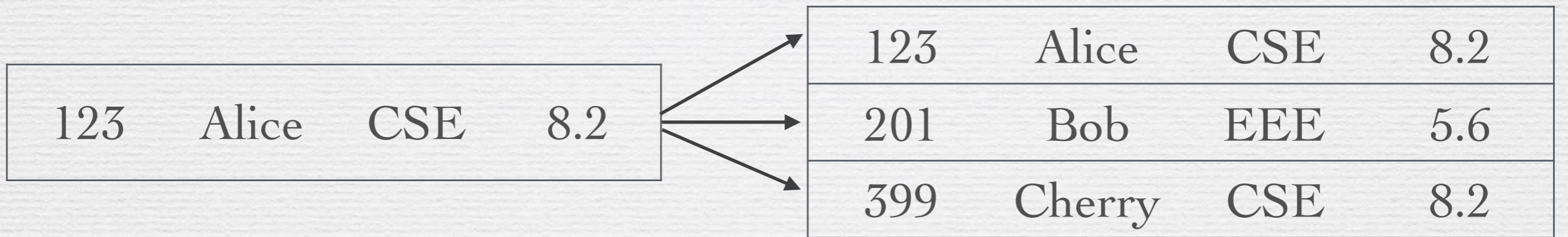
rollNo	cName	dept	marks
123	DBMS	CSE	48
123	OS	CSE	36
399	DBMS	CSE	25
201	DBMS	CSE	40
123	Statistics	Maths	39
201	Control	EEE	35.5
399	Control	ECE	34

# Self Joins

Q5. Find all pairs of students who are from the same department?

Q5. Find all pairs of students who are from the same department?

```
select S1.name, S2.name, S1.dept
from Student S1, Student S2
where
S1.dept = S2.dept
S1.rollNo <> S2.rollNo;;
```



Q5. Find all pairs of students who are from the same department? (*modified*)

```
select S1.name, S2.name, S1.dept
from Student S1, Student S2
where
S1.dept = S2.dept
and S1.rollNo < S2.rollNo;
```

Q5. Find all pairs of students who are from the same department? (*another way*)

```
select S1.name, S2.name, S1.dept
from Student S1 JOIN Student S2
ON S1.dept = S2.dept
and S1.rollNo < S2.rollNo;
```

Note: This is the *explicit join*. (The earlier one is called implicit join). SQL standard 1992 deprecated implicit joins in favour of the explicit join.

Q6. Find all students who have taken a course offered by both CSE and Maths?



Q6. Find all students who have taken a course offered by both CSE and Maths?

```
select rollNo  
from Course where dept="CSE"
```

**intersect**

```
select rollNo  
from Course where dept="Maths";
```

Q6. Find all students who have taken a course from both CSE and Maths using self Join?  
(*another approach*)

```
select distinct C1.rollNo  
from Course C1, Course C2  
where  
C1.rollNo = C2.rollNo  
and  
C1.dept="CSE" and C2.dept="Maths";
```

**Exercise:** Find name and rollNo of all the students who have taken a course from both CSE and Maths using Join?

Q7. Find all pairs of courses offered in the same department?

Q7. Find all pairs of courses offered in the same department?

```
select distinct C1.cName, C2.cName,  
C1.dept  
from Course C1 JOIN Course C2  
ON C1.dept = C2.dept  
and C1.cName <> C2.cName;
```

cName	cName	dept
DBMS	OS	CSE
OS	DBMS	CSE

# Key

Student

rollNo	name	dept	CGPA
123	Alice	CSE	8.2
201	Bob	EEE	5.6


Course

rollNo	cName	dept	marks
123	DBMS	CSE	48
123	OS	CSE	36
399	DBMS	CSE	25
201	DBMS	CSE	40

# Key

Student

rollNo	name	dept	CGPA
123	Alice	CSE	8.2
201	Bob	EEE	5.6



*rollNo* is unique  
across rows

# Key

None of *rollNo*, *cName* or *dept* is unique!

Is *(rollNo, cName)* unique?

Course

rollNo	cName	dept	marks
123	DBMS	CSE	48
123	OS	CSE	36
399	DBMS	CSE	25
201	DBMS	CSE	40



# Key

- The way we define a field or group of fields as key is as follows:

```
CREATE TABLE faculty(name, dept,  
    PRIMARY KEY(name, dept));
```

- If we try to insert the same row again, the system will not allow it.
- This is one kind of integrity constraints enforced by a DBMS

# Tables without Keys

- It is possible to define tables without keys according to SQL specification (*due to practical considerations*)
- This is one place where SQL deviates from its underlying Relational model principles

name	dept
Alice	CSE
Bob	EEE
Alice	CSE

# More On Joins

1. Inner Join
2. Left Outer Join
3. Right Outer Join
4. Full Outer Join
5. Natural Join

# INNER JOIN

- The default Join
- It is represented by
- We can join on more than 1 fields
- keyword is `INNER JOIN` *or simply* `JOIN`

# LEFT OUTER JOIN

Faculty

name	dept
ABC	CSE
SKB	EEE
PPP	CSE

Teaching

fName	cName
ABC	Algorithms
SKB	Power System
ABC	Probability
ZZZ	EVS

# LEFT OUTER JOIN

```
select * from  
Faculty LEFT OUTER JOIN Teaching  
ON name = fName;
```

name	dept	fName	cName
ABC	CSE	ABC	Algorithms
ABC	CSE	ABC	Probability
PPP	CSE	NULL	NULL
SKB	EEE	SKB	Power System

# LEFT OUTER JOIN

- Include all rows in the left table. If no match found in the right table add null to the missing attributes.
- The RIGHT OUTER JOIN is the same as the LEFT OUTER JOIN with the order of tables reversed.

# FULL OUTER JOIN

- All rows in the left table and the right table should appear at least once. If no match on the joining field, add null to the missing attributes.

- Faculty X Teaching

name	dept	fName	cName
ABC	CSE	ABC	Algorithms
ABC	CSE	ABC	Probability
PPP	CSE	NULL	NULL
SKB	EEE	SKB	Power System
NULL	NULL	ZZZ	EVS



# FULL OUTER JOIN

- Note that FULL OUTER JOIN is not the same as cross product (cartesian product)
- Most DBMS don't support full outer join
- eg. SQLite supports inner and left outer join only
- Mostly because it is rarely required and if required can be implemented with nested queries. (*Exercise for later*)

# Natural JOIN



- Join based on equality of columns with *same name*.
- Dangerous! (Two columns may have the same name)
- Theoretical importance only!
- Not supported by most DBMS

# Exercise

- What is Self Join? Is it Inner join or Outer join? Why? (*Ans: It is none of them - Try to do self join using inner join as well as left outer join and show that results can be different*)
- What happens to OUTER JOIN if no joining condition is given?

# Set difference

Q8. Find all students who have taken a course from CSE but not Maths?

```
select rollNo
from Course where dept="CSE"
except
select rollNo
from Course where dept="Maths"
```

## Student

rollNo	name	dept	CGPA
123	Alice	CSE	8.2
201	Bob	EEE	5.6
399	Cherry	CSE	8.2

## Course

rollNo	cName	dept	marks
123	DBMS	CSE	48
123	OS	CSE	36
399	DBMS	CSE	25
201	DBMS	CSE	40
123	Statistics	Maths	39
201	Control	EEE	35.5

# Set difference

Q8. Can you find all students who have taken a course from CSE but not Maths *without except*?

```
select C1.rollNo
from Course C1, Course C2
where
C1.rollNo = C2.rollNo
and
C1.dept="CSE" and C2.dept<>"Maths";
```

**Fails! It finds all students who have taken a course in CSE!**

Q8. Can we find all students who have taken a course from CSE but not Maths? (*what about this?*)

```
select C1.rollNo
from Course C1, Course C2
where
C1.cName = C2.cName
and
C1.dept="CSE" and C2.dept<>"Maths"
and C2.dept<>"CSE";
```

**Also fails!**  
**(misses CSE only students)**

Q9. Can you find out all the students who have taken course(s) from the CSE department only?



Q9. Can you find out all the students who have taken course(s) from the CSE department only?

We need to have a holistic view rather than looking at pairs of courses (as in joins) - so joins won't work!

**Exercise:** But it is possible to write this **using except**.

**Try It!**

# Nested Queries

Q10. What is the following query for?

```
select rollNo, name
from Student
where
rollNo in
(select rollNo
from Course where Dept="CSE");
```

Can you write the same using joins?

Q8. *(Repeat)* Find all students who have taken a course from CSE but not Maths?

```
select rollNo
from Course where dept="CSE"
and rollNo not in
(select rollNo
from Course where dept="Maths")
```

Q9. *(Repeat)* Find out all the students who have taken course(s) from the CSE department only? *(using nested queries)*

Q9. Find out all the students who have taken course(s) from the CSE department only? (*using nested queries*)

```
select rollNo from Course
where dept="CSE"
and rollNo not in
( select distinct C1.rollNo
from Course C1, Course C2
where C1.rollNo = C2.rollNo
and C1.dept <> C2.dept
and C1.dept="CSE" );
```

Q9. Find out all the students who have taken course(s) from the CSE department only? (*using nested queries and no joins*)

```
select distinct rollNo from Course
where dept="CSE" and
rollNo not in (select rollNo
from Course
where dept="CSE"
and rollNo in (select rollNo
from Course
where dept <> "CSE" ));
```

# Why JOIN <> Nested Select

Q11. What is the average CGPA of students enrolled in the CSE department?

*(just list them)*



## Student

rollNo	name	dept	CGPA
123	Alice	CSE	8.2
201	Bob	EEE	5.6
399	Cherry	CSE	8.2

## Course

rollNo	cName	dept	marks
123	DBMS	CSE	48
123	OS	CSE	36
399	DBMS	CSE	25
201	DBMS	CSE	40
123	Statistics	Maths	39
201	Control	EEE	35.5

```
select CGPA
from Student
where rollNo in
    (select rollNo from Course
     where dept="CSE");
```

CGPA

-----

8.2

5.6

8.2

```
select CGPA
from Student S, Course C
where S.rollNo = C.rollNo
and C.dept="CSE");
```

CGPA

-----

8.2

8.2

8.2

5.6

8.2

But what if we take **distinct** ?

Q12. Find all students such that there is another student from the same department?

Q12. Find all students such that there is another student from the same department?

```
select name, dept from Student S1
where exists (select * from Student S2
             where S1.dept = S2.dept
             and S1.rollNo <> S2.rollNo);
```

Q12. Find all students such that there is another student from the same department?

### Exercise:

1. Rewrite the query using nested select and **IN** operator?
2. Write the query using JOIN only?
3. Can you refer S2 which is declared in the inner select from the outside select?

Q13. Find the student with the highest mark?  
(without using the *max operator*)

Course

rollNo	cName	dept	marks
123	DBMS	CSE	48
123	OS	CSE	36
399	DBMS	CSE	25
201	DBMS	CSE	40
123	Statistics	Maths	39
201	Control	EEE	35.5

Q13. Find the student with the highest mark?  
(without using the *max operator*)

```
select rollNo, cName from Course C1
where not exists
  (select * from Course C2
   where C2.marks > C1.marks);
```



Q14. Find the student with the highest CGPA?  
(without using the *max operator*)

Student

rollNo	name	dept	CGPA
123	Alice	CSE	8.2
201	Bob	EEE	5.6
399	Cherry	CSE	8.2

Q14. Find the student with the highest CGPA?  
(without using the *max operator*)

```
select rollNo, name, CGPA
from Student S1
where not exists
  (select * from Student S2
   where S2.CGPA > S1.CGPA);
```

rollNo	name	CGPA
123	Alice	8.2
399	Cherry	8.2

Q14. Find the student with the highest CGPA? (with the *all* keyword)

```
select name, CGPA
from Student
where CGPA >= all (select CGPA from
Student);
```

Q14. Find the student with the highest CGPA? (*with the any operator*)

```
select name, CGPA  
from Student  
where not CGPA < any (select CGPA  
                        from Student);
```

*all* and *any* are not supported in sqlite.

We can always write equivalent queries with *exists*

*Exercise:* Is the following query same as the previous one?

```
select name, CGPA
from Student
where CGPA <> any (select CGPA
                    from Student);
```