

Chapter 3: Introduction to SQL

Edited by Radhika Sukapuram. Original slides by

Database System Concepts, 6th Ed.

©Silberschatz, Korth and Sudarshan

See www.db-book.com for conditions on re-use



Nested Subqueries

- ❑ SQL provides a mechanism for the nesting of subqueries. A **subquery** is a **select-from-where** expression that is nested within another query.
- ❑ Nesting can be done in the following SQL query

```
select  $A_1, A_2, \dots, A_n$   
from  $r_1, r_2, \dots, r_m$   
where  $P$ 
```

as follows:

- ❑ A_i can be replaced by a subquery that generates a single value.
- ❑ r_i can be replaced by any valid subquery
- ❑ P can be replaced with an expression of the form:

$B <\text{operation}> (\text{subquery})$

Where B is an attribute and $<\text{operation}>$ to be defined later.



Subqueries in the Where Clause



Subqueries in the Where Clause

- A common use of subqueries is to perform tests:
 - For set membership
 - For set comparisons
 - For set cardinality.



section

<i>course_id</i>	<i>sec_id</i>	<i>semester</i>	<i>year</i>	<i>building</i>	<i>room_number</i>	<i>time_slot_id</i>
BIO-101	1	Summer	2009	Painter	514	B
BIO-301	1	Summer	2010	Painter	514	A
CS-101	1	Fall	2009	Packard	101	H
CS-101	1	Spring	2010	Packard	101	F
CS-190	1	Spring	2009	Taylor	3128	E
CS-190	2	Spring	2009	Taylor	3128	A
CS-315	1	Spring	2010	Watson	120	D
CS-319	1	Spring	2010	Watson	100	B
CS-319	2	Spring	2010	Taylor	3128	C
CS-347	1	Fall	2009	Taylor	3128	A
EE-181	1	Spring	2009	Taylor	3128	C
FIN-201	1	Spring	2010	Packard	101	B
HIS-351	1	Spring	2010	Painter	514	C
MU-199	1	Spring	2010	Packard	101	D
PHY-101	1	Fall	2009	Watson	100	A



Set Membership

- Find courses offered in Fall 2009 and in Spring 2010

```
select distinct course_id
from section
where semester = 'Fall' and year= 2009 and
       course_id in (select course_id
                      from section
                      where semester = 'Spring' and year= 2010);
```

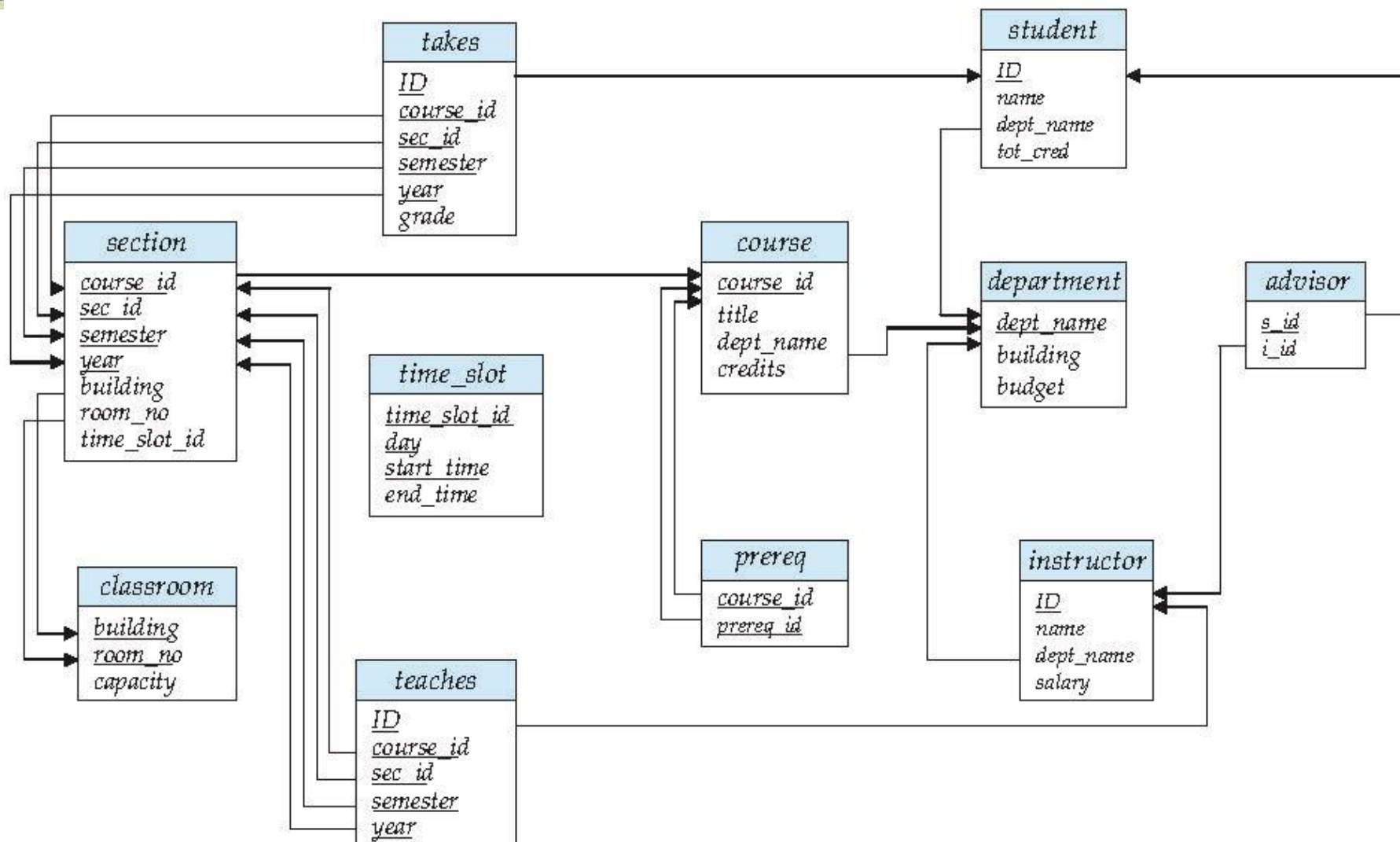
- Find courses offered in Fall 2009 but not in Spring 2010

```
select distinct course_id
from section
where semester = 'Fall' and year= 2009 and
       course_id not in (select course_id
                          from section
                          where semester = 'Spring' and year= 2010);
```

- Evaluate inner query, then outer query



Set Operations





Set Membership (Cont.)

- Find the total number of (distinct) students who have taken course sections taught by the instructor with *ID* 10101

```
select count (distinct ID)  
from takes  
where (course_id, sec_id, semester, year) in  
      (select course_id, sec_id, semester, year  
       from teaches  
       where teaches.ID= 10101);
```

- Note: Above query can be written in a much simpler manner.
The formulation above is simply to illustrate SQL features.



Set Comparison – “some” Clause

- Find names of instructors with salary greater than that of some (at least one) instructor in the Biology department.

```
select distinct T.name  
from instructor as T, instructor as S  
where T.salary > S.salary and S.dept name = 'Biology';
```

- Same query using > **some** clause

```
select name  
from instructor  
where salary > some (select salary  
                        from instructor  
                        where dept name = 'Biology');
```



Definition of “some” Clause

- $F \text{ <comp> some } r \Leftrightarrow \exists t \in r \text{ such that } (F \text{ <comp> } t)$
Where <comp> can be: $<, \leq, >, =, \neq$

$$(5 < \text{some } \begin{array}{|c|} \hline 0 \\ \hline 5 \\ \hline 6 \\ \hline \end{array}) = \text{true}$$

(read: 5 < some tuple in the relation)

$$(5 < \text{some } \begin{array}{|c|} \hline 0 \\ \hline 5 \\ \hline \end{array}) = \text{false}$$

$$(5 = \text{some } \begin{array}{|c|} \hline 0 \\ \hline 5 \\ \hline \end{array}) = \text{true}$$

$$(5 \neq \text{some } \begin{array}{|c|} \hline 0 \\ \hline 5 \\ \hline \end{array}) = \text{true (since } 0 \neq 5)$$

$(= \text{some}) \equiv \text{in}$

However, $(\neq \text{some}) \not\equiv \text{not in}$



instructor

<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

teaches

<i>ID</i>	<i>course_id</i>	<i>sec_id</i>	<i>semester</i>	<i>year</i>
10101	CS-101	1	Fall	2009
10101	CS-315	1	Spring	2010
10101	CS-347	1	Fall	2009
12121	FIN-201	1	Spring	2010
15151	MU-199	1	Spring	2010
22222	PHY-101	1	Fall	2009



Set Comparison – “all” Clause

- Find the names of all instructors whose salary is greater than the salary of all instructors in the Biology department.

```
select name
from instructor
where salary > all (select salary
                      from instructor
                      where dept name = 'Biology');
```



Definition of “all” Clause

□ $F \text{ <comp> all } r \Leftrightarrow \forall t \in r (F \text{ <comp> } t)$

$(5 < \text{all } \begin{array}{|c|} \hline 0 \\ \hline 5 \\ \hline 6 \\ \hline \end{array}) = \text{false}$

$(5 < \text{all } \begin{array}{|c|} \hline 6 \\ \hline 10 \\ \hline \end{array}) = \text{true}$

$(5 = \text{all } \begin{array}{|c|} \hline 4 \\ \hline 5 \\ \hline \end{array}) = \text{false}$

$(5 \neq \text{all } \begin{array}{|c|} \hline 4 \\ \hline 6 \\ \hline \end{array}) = \text{true (since } 5 \neq 4 \text{ and } 5 \neq 6)$

$(\neq \text{all}) \equiv \text{not in}$

However, $(= \text{all}) \not\equiv \text{in}$



Test for Empty Relations

- The **exists** construct returns the value **true** if the argument subquery is nonempty.
- **exists** $r \Leftrightarrow r \neq \emptyset$
- **not exists** $r \Leftrightarrow r = \emptyset$



Use of “exists” Clause

- Yet another way of specifying the query “Find all courses taught in both the Fall 2009 semester and in the Spring 2010 semester”

```
select course_id
from section as S
where semester = 'Fall' and year = 2009 and
      exists (select *
              from section as T
              where semester = 'Spring' and year = 2010
                  and S.course_id = T.course_id);
```

- Evaluate **from** clause of outer query, then predicate of outer **where** clause (partially), then select *course_id* of that tuple
- Evaluate inner query using the *course_id* (that is, *S.course_id*)
- Do the same for all tuples of the outer query

- **Correlation name** – variable *S* in the outer query
- **Correlated subquery** – the inner query
- Scoping rules