SQL Join Operations

Database System Concepts, 7th Ed.

Outline

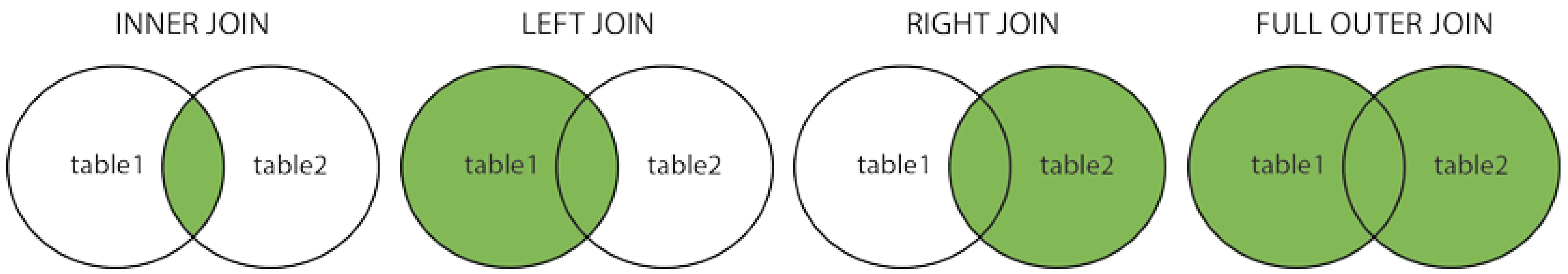
- SQL Join Types
  - Inner Join
  - Natural Join
  - Outer Join

Joined Relations

- **Join operations** take two relations and return as a result another relation.
- A join operation is a Cartesian product which requires that tuples in the two relations match (under some condition). It also specifies the attributes that are present in the result of the join
- The join operations are typically used as subquery expressions in the **from** clause
- There are different types of joins in SQL:
  - Inner join
  - Natural join
  - Outer join

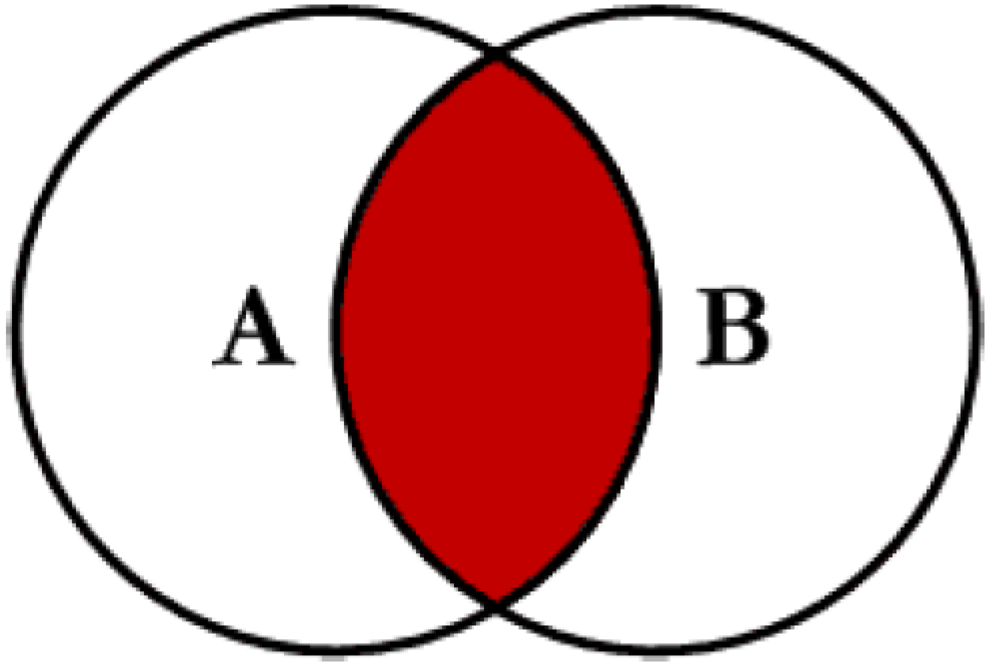
SQL Join Types

- **(INNER) JOIN:** Returns records that have matching values in both tables
- **LEFT (OUTER) JOIN:** Returns all records from the left table, and the matched records from the right table
- **RIGHT (OUTER) JOIN:** Returns all records from the right table, and the matched records from the left table
- **FULL (OUTER) JOIN:** Returns all records when there is a match in either left or right table



Inner Join/Natural Join

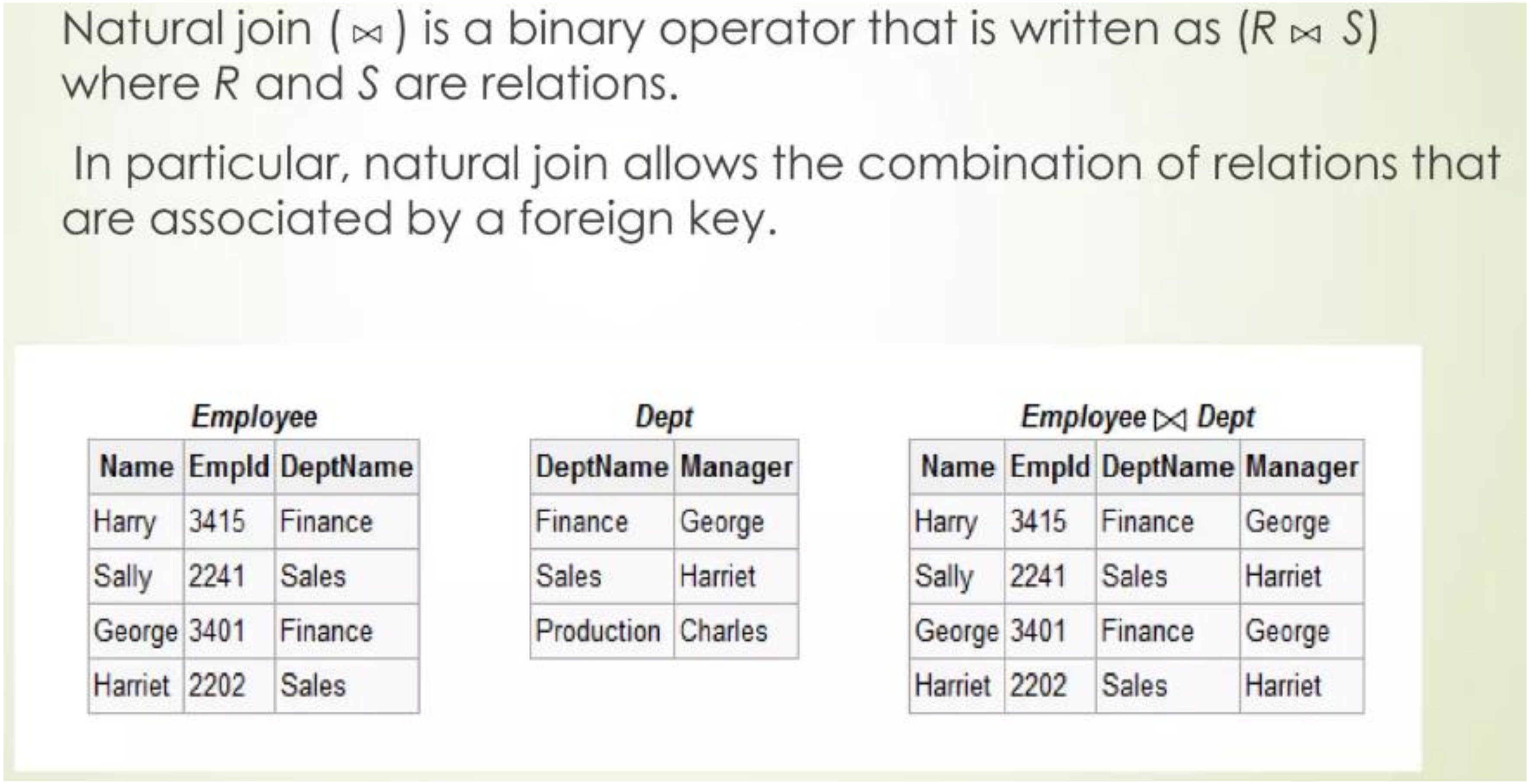
- **Inner Join/Natural Join** produces only the set of records that match in both Table A and Table B
- Most commonly used, best understood join



Creating Natural Joins

- The `NATURAL JOIN` clause is based on all columns in the two tables that have the same name.
- It selects rows from the two tables that have equal values in all matched columns.
- If the columns having the same names have different data types, an error is returned.

Natural Join



Natural join (⋈) is a binary operator that is written as (R ⋈ S) where R and S are relations.

In particular, natural join allows the combination of relations that are associated by a foreign key.

Employee

Name	EmpId	DeptName
Harry	3415	Finance
Sally	2241	Sales
George	3401	Finance
Harriet	2202	Sales

Dept

DeptName	Manager
Finance	George
Sales	Harriet
Production	Charles

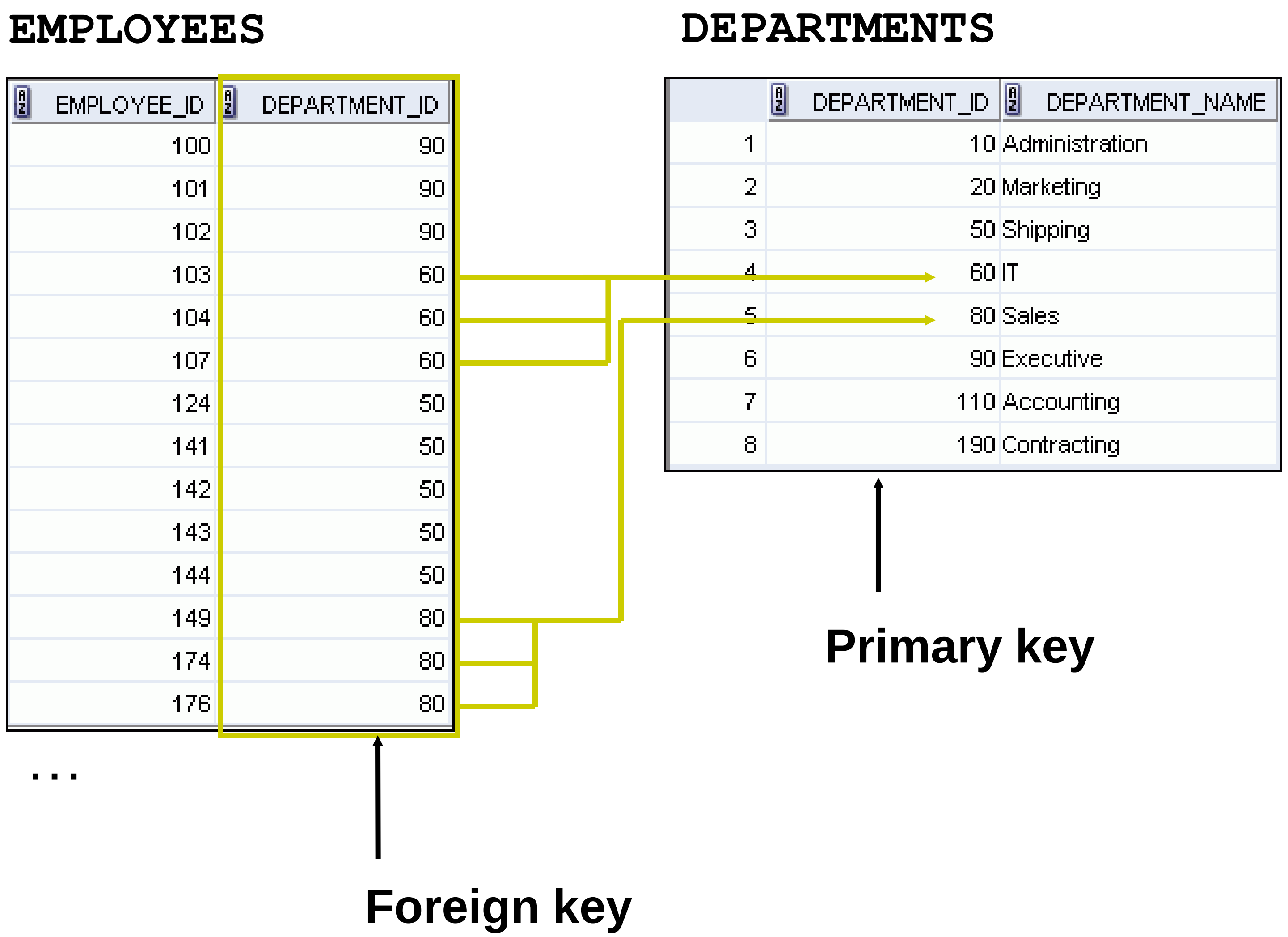
Employee ⋈ *Dept*

Name	EmpId	DeptName	Manager
Harry	3415	Finance	George
Sally	2241	Sales	Harriet
George	3401	Finance	George
Harriet	2202	Sales	Harriet

Creating Joins with the USING Clause

- If several columns have the same names but the data types do not match, use the USING clause to specify the columns for the equijoin.
- Use the USING clause to match only one column when more than one column matches.
- The NATURAL JOIN and USING clauses are mutually exclusive.

Joining Column Names



EMPLOYEES

EMPLOYEE_ID	DEPARTMENT_ID
100	90
101	90
102	90
103	60
104	60
107	60
124	50
141	50
142	50
143	50
144	50
149	80
174	80
176	80

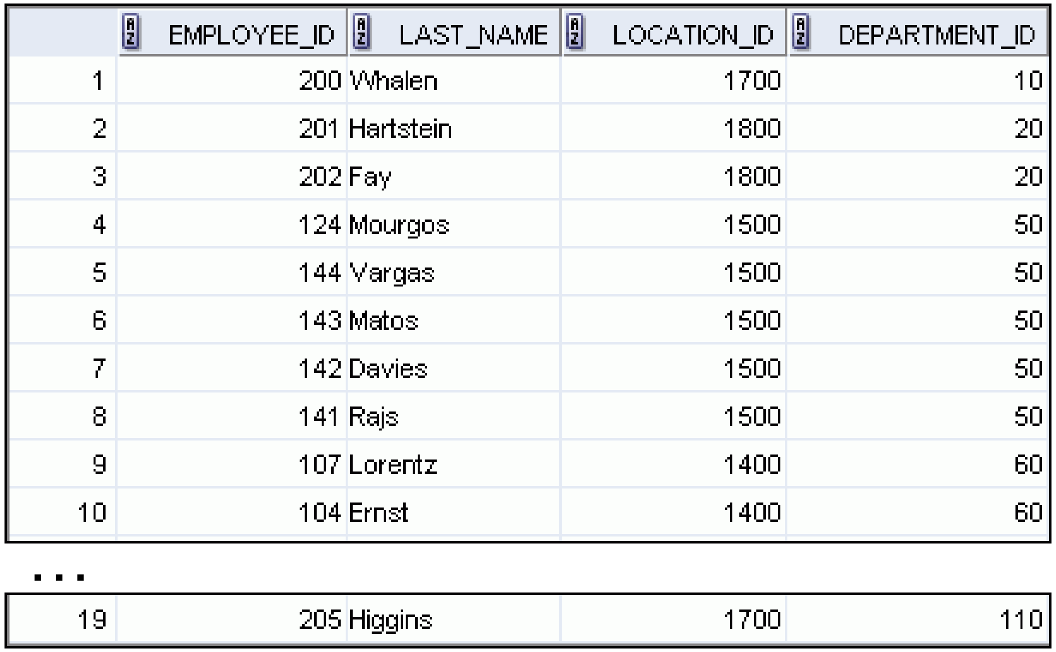
…

DEPARTMENTS

	DEPARTMENT_ID	DEPARTMENT_NAME
1	10	Administration
2	20	Marketing
3	50	Shipping
4	60	IT
5	80	Sales
6	90	Executive
7	110	Accounting
8	190	Contracting

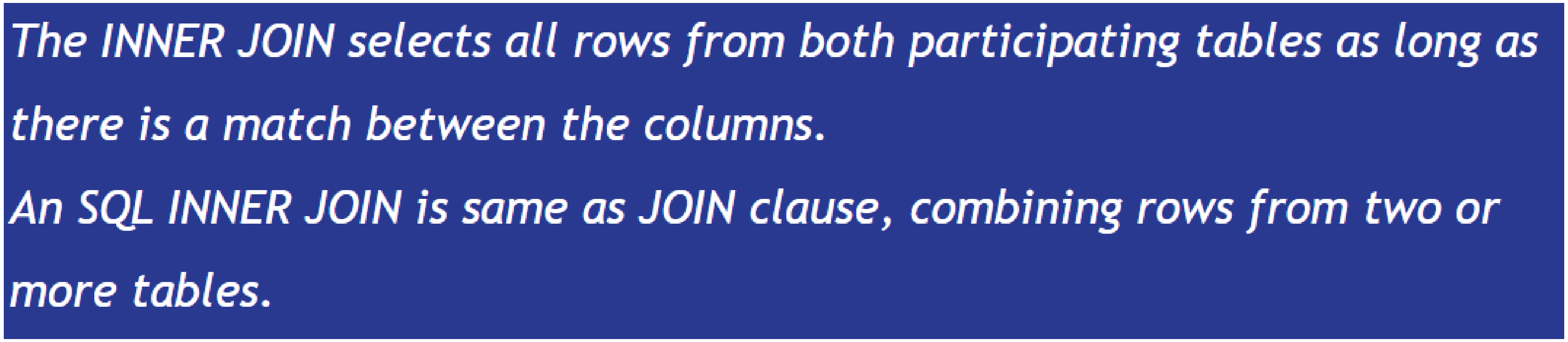
Retrieving Records with the `USING` Clause

```
SELECT employee_id, last_name,
       location_id, department_id
FROM   employees JOIN departments
USING (department_id) ;
```



	EMPLOYEE_ID	LAST_NAME	LOCATION_ID	DEPARTMENT_ID
1	200	Whalen	1700	10
2	201	Hartstein	1800	20
3	202	Fay	1800	20
4	124	Mourgos	1500	50
5	144	Vargas	1500	50
6	143	Matos	1500	50
7	142	Davies	1500	50
8	141	Rajs	1500	50
9	107	Lorentz	1400	60
10	104	Ernst	1400	60
...				
19	205	Higgins	1700	110

Inner Join



The INNER JOIN selects all rows from both participating tables as long as there is a match between the columns.

An SQL INNER JOIN is same as JOIN clause, combining rows from two or more tables.

Inner Join Example

table_A

A	M
1	m
2	n
4	o

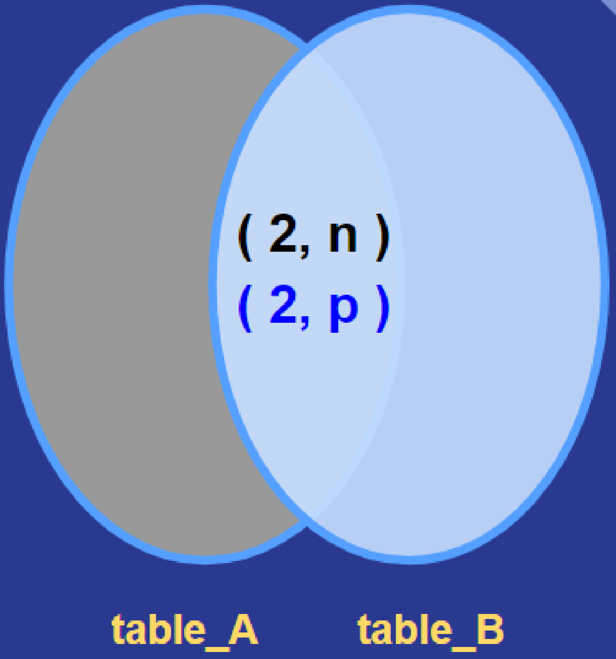
table_B

A	N
2	p
3	q
5	r

```
SELECT * FROM table_A
INNER JOIN table_B
ON table_A.A=table_B.A;
```

A	M	A	N
2	n	2	p

Output



- SELECT * FROM table_A **INNER JOIN** table_B **ON** table_A.A = table_B.A
- This is the same as doing SELECT * FROM table_A, table_B **WHERE** table_A.A = table_B.A

Inner Join/Natural Join

- **A NATURAL join** is just an inner join where the join is implicitly created using **any** matching columns between the two tables

- SELECT * FROM TableA **NATURAL JOIN** TableB

Natural Join in SQL (Cont.)

- The **from** clause can have multiple relations combined using natural join:

  select $A_1, A_2, \ldots A_n$
  from r_1 **natural join** r_2 **natural join .. natural join** r_n
  where P ;

Sample Tables

TableA

PK	Value
1	FOX
2	COP
3	TAXI
6	WASHINGTON
7	DELL
5	ARIZONA
4	LINCOLN
10	LUCENT

TableB

PK	Value
1	TROT
2	CAR
3	CAB
6	MONUMENT
7	PC
8	MICROSOFT
9	APPLE
11	SCOTCH

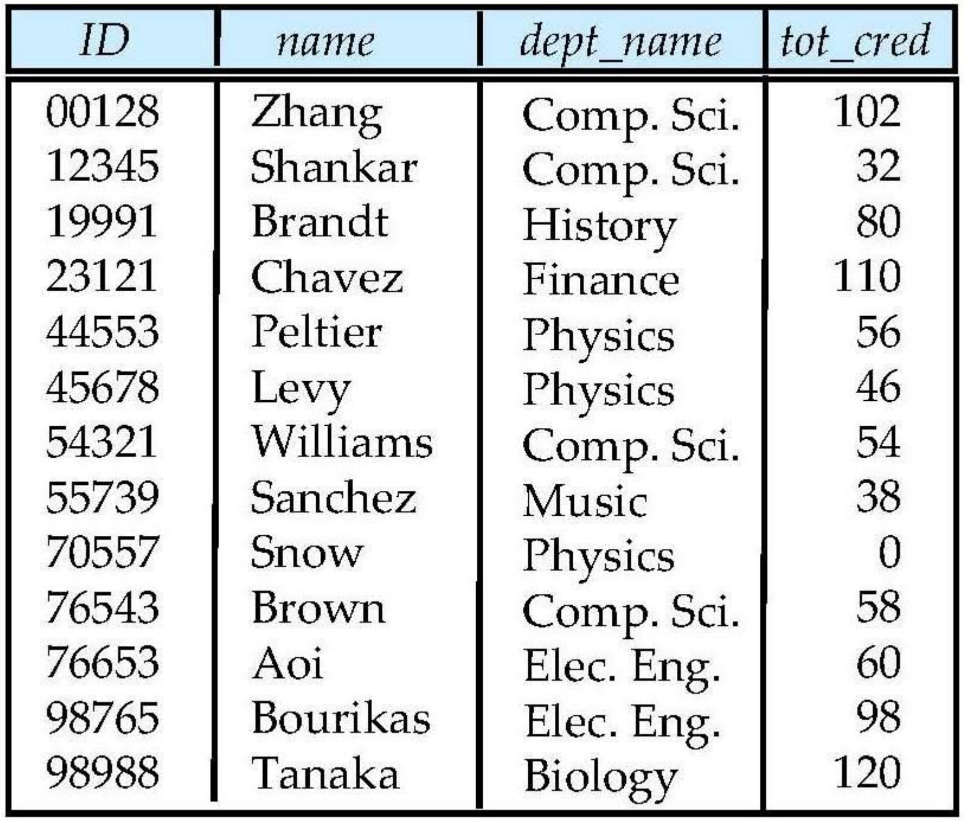
Inner Join

- SELECT * FROM TableA **INNER JOIN** TableB **ON** TableA.PK = TableB.PK

- This is the same as doing SELECT * FROM TableA, TableB **WHERE** TableA.PK = TableB.PK

TableA		
Value	PK	Value
FOX	1	TROT
COP	2	CAR
TAXI	3	CAB
WASHINGTON	6	MONUMENT
DELL	7	PC

Student Relation

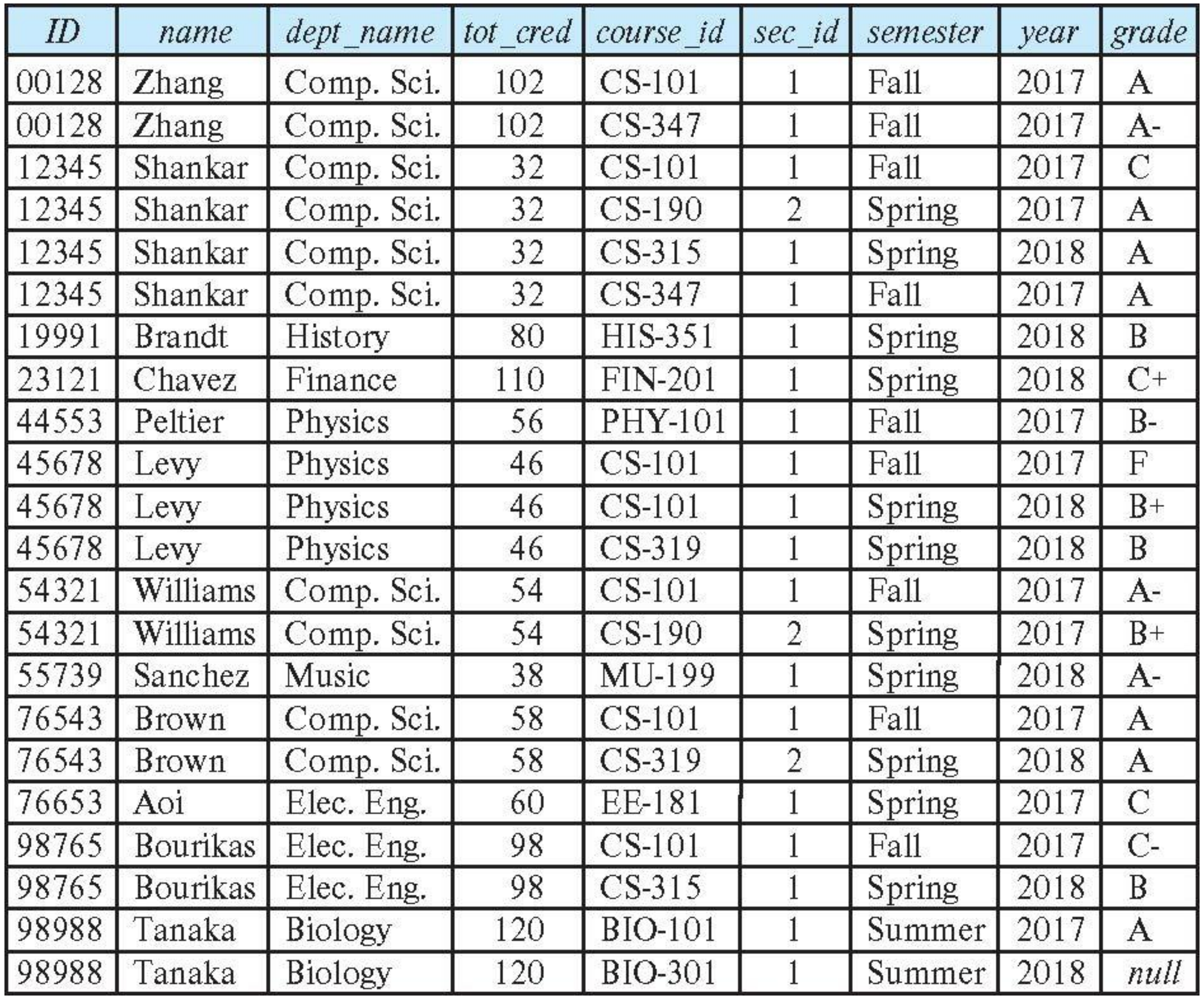


ID	*name*	*dept_name*	*tot_cred*
00128	Zhang	Comp. Sci.	102
12345	Shankar	Comp. Sci.	32
19991	Brandt	History	80
23121	Chavez	Finance	110
44553	Peltier	Physics	56
45678	Levy	Physics	46
54321	Williams	Comp. Sci.	54
55739	Sanchez	Music	38
70557	Snow	Physics	0
76543	Brown	Comp. Sci.	58
76653	Aoi	Elec. Eng.	60
98765	Bourikas	Elec. Eng.	98
98988	Tanaka	Biology	120

Takes Relation

ID	*course_id*	*sec_id*	*semester*	*year*	*grade*
00128	CS-101	1	Fall	2017	A
00128	CS-347	1	Fall	2017	A-
12345	CS-101	1	Fall	2017	C
12345	CS-190	2	Spring	2017	A
12345	CS-315	1	Spring	2018	A
12345	CS-347	1	Fall	2017	A
19991	HIS-351	1	Spring	2018	B
23121	FIN-201	1	Spring	2018	C+
44553	PHY-101	1	Fall	2017	B-
45678	CS-101	1	Fall	2017	F
45678	CS-101	1	Spring	2018	B+
45678	CS-319	1	Spring	2018	B
54321	CS-101	1	Fall	2017	A-
54321	CS-190	2	Spring	2017	B+
55739	MU-199	1	Spring	2018	A-
76543	CS-101	1	Fall	2017	A
76543	CS-319	2	Spring	2018	A
76653	EE-181	1	Spring	2017	C
98765	CS-101	1	Fall	2017	C-
98765	CS-315	1	Spring	2018	B
98988	BIO-101	1	Summer	2017	A
98988	BIO-301	1	Summer	2018	*null*

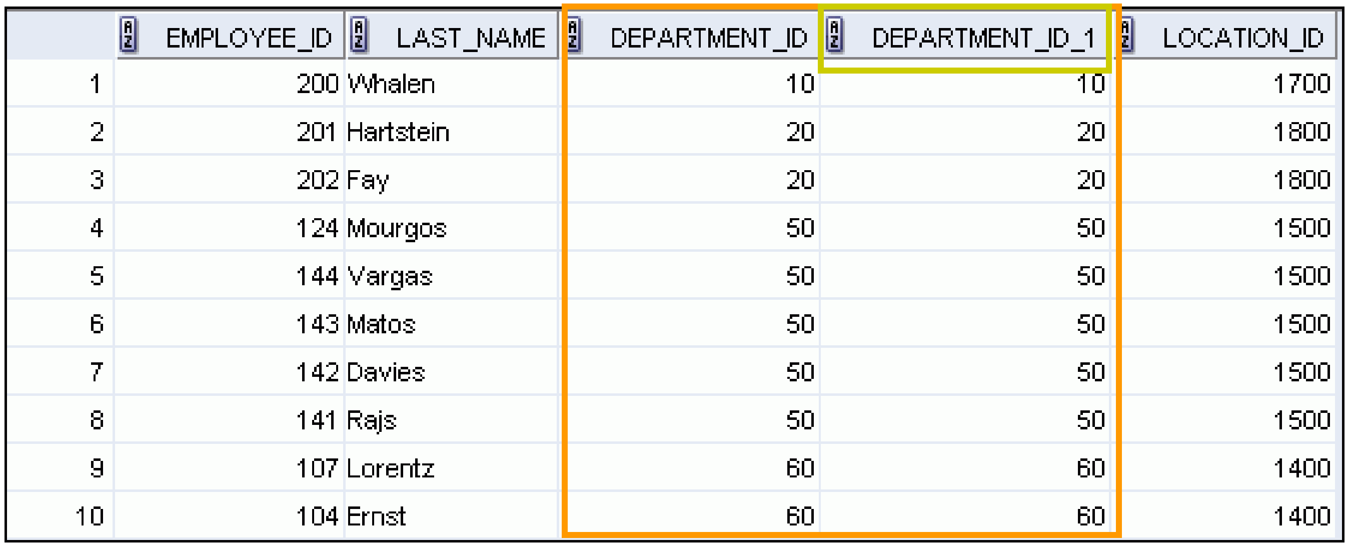
student **natural join** *takes*



ID	*name*	*dept_name*	*tot_cred*	*course_id*	*sec_id*	*semester*	*year*	*grade*
00128	Zhang	Comp. Sci.	102	CS-101	1	Fall	2017	A
00128	Zhang	Comp. Sci.	102	CS-347	1	Fall	2017	A-
12345	Shankar	Comp. Sci.	32	CS-101	1	Fall	2017	C
12345	Shankar	Comp. Sci.	32	CS-190	2	Spring	2017	A
12345	Shankar	Comp. Sci.	32	CS-315	1	Spring	2018	A
12345	Shankar	Comp. Sci.	32	CS-347	1	Fall	2017	A
19991	Brandt	History	80	HIS-351	1	Spring	2018	B
23121	Chavez	Finance	110	FIN-201	1	Spring	2018	C+
44553	Peltier	Physics	56	PHY-101	1	Fall	2017	B-
45678	Levy	Physics	46	CS-101	1	Fall	2017	F
45678	Levy	Physics	46	CS-101	1	Spring	2018	B+
45678	Levy	Physics	46	CS-319	1	Spring	2018	B
54321	Williams	Comp. Sci.	54	CS-101	1	Fall	2017	A-
54321	Williams	Comp. Sci.	54	CS-190	2	Spring	2017	B+
55739	Sanchez	Music	38	MU-199	1	Spring	2018	A-
76543	Brown	Comp. Sci.	58	CS-101	1	Fall	2017	A
76543	Brown	Comp. Sci.	58	CS-319	2	Spring	2018	A
76653	Aoi	Elec. Eng.	60	EE-181	1	Spring	2017	C
98765	Bourikas	Elec. Eng.	98	CS-101	1	Fall	2017	C-
98765	Bourikas	Elec. Eng.	98	CS-315	1	Spring	2018	B
98988	Tanaka	Biology	120	BIO-101	1	Summer	2017	A
98988	Tanaka	Biology	120	BIO-301	1	Summer	2018	*null*

Using Renaming

```
SELECT e.employee_id, e.last_name, e.department_id,
       d.department_id, d.location_id
FROM   employees e INNER JOIN departments d
ON     (e.department_id = d.department_id);
```



	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_ID_1	LOCATION_ID
1	200	Whalen	10	10	1700
2	201	Hartstein	20	20	1800
3	202	Fay	20	20	1800
4	124	Mourgos	50	50	1500
5	144	Vargas	50	50	1500
6	143	Matos	50	50	1500
7	142	Davies	50	50	1500
8	141	Rajs	50	50	1500
9	107	Lorentz	60	60	1400
10	104	Ernst	60	60	1400

…

Applying Additional Conditions to a Join

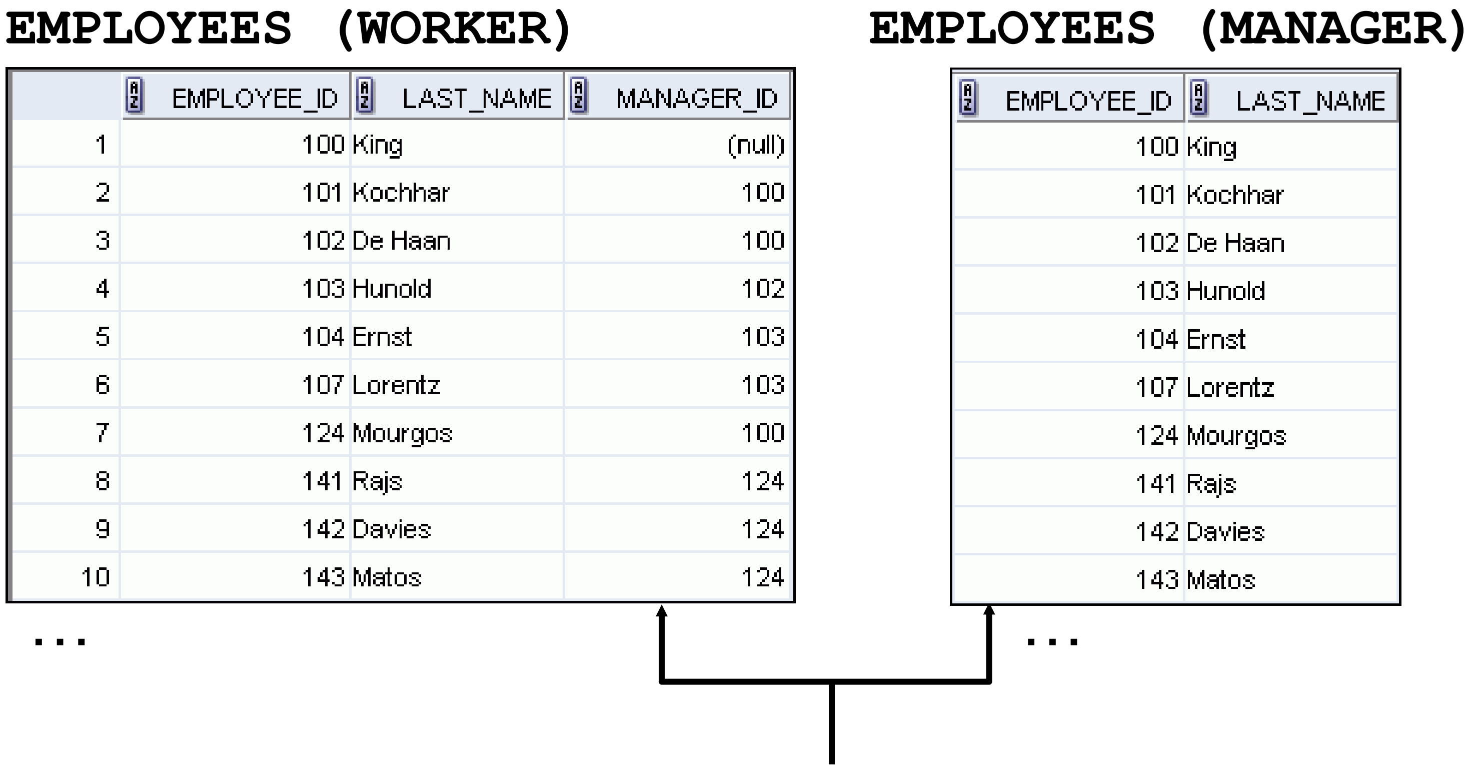
- Use the `AND` clause or the `WHERE` clause to apply additional conditions:

```
SELECT e.employee_id, e.last_name, e.department_id,
       d.department_id, d.location_id
FROM   employees e JOIN departments d
ON      (e.department_id = d.department_id)
AND    e.manager_id = 149 ;
```

Or

```
SELECT e.employee_id, e.last_name, e.department_id,
       d.department_id, d.location_id
FROM   employees e JOIN departments d
ON      (e.department_id = d.department_id)
WHERE   e.manager_id = 149 ;
```

Self Join: Joining a Table to Itself



EMPLOYEES (WORKER)

	EMPLOYEE_ID	LAST_NAME	MANAGER_ID
1	100	King	(null)
2	101	Kochhar	100
3	102	De Haan	100
4	103	Hunold	102
5	104	Ernst	103
6	107	Lorentz	103
7	124	Mourgos	100
8	141	Rajs	124
9	142	Davies	124
10	143	Matos	124

…

EMPLOYEES (MANAGER)

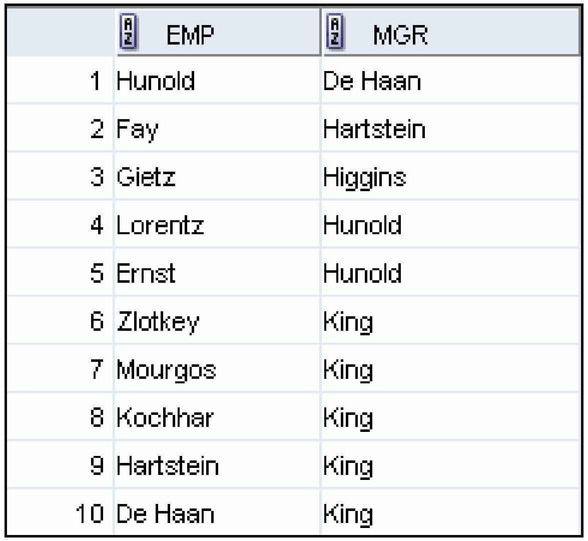
EMPLOYEE_ID	LAST_NAME
100	King
101	Kochhar
102	De Haan
103	Hunold
104	Ernst
107	Lorentz
124	Mourgos
141	Rajs
142	Davies
143	Matos

…

`MANAGER_ID` in the `WORKER` table is equal to `EMPLOYEE_ID` in the `MANAGER` table.

Self-Joins Using the ON Clause

```
SELECT worker.last_name emp, manager.last_name mgr
FROM   employees worker JOIN employees manager
ON    (worker.manager_id = manager.employee_id);
```



	EMP	MGR
1	Hunold	De Haan
2	Fay	Hartstein
3	Gietz	Higgins
4	Lorentz	Hunold
5	Ernst	Hunold
6	Zlotkey	King
7	Mourgos	King
8	Kochhar	King
9	Hartstein	King
10	De Haan	King

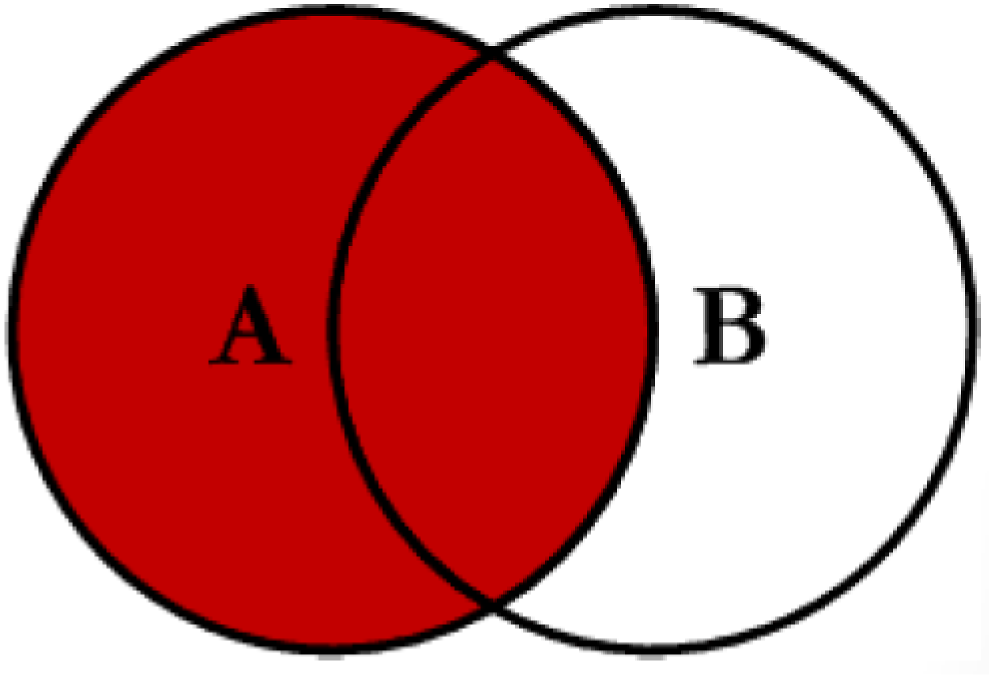
…

Outer Join

- An extension of the join operation that **avoids loss of information**.
- Computes the join and then adds tuples form 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 join or left outer join
  - Right join or right outer join
  - full outer join

Left Outer Join

- Left outer join produces a complete set of records from Table A, with the matching records (where available) in Table B. If there is no match, the right side will contain null.



Sample Tables

TableA

PK	Value
1	FOX
2	COP
3	TAXI
6	WASHINGTON
7	DELL
5	ARIZONA
4	LINCOLN
10	LUCENT

TableB

PK	Value
1	TROT
2	CAR
3	CAB
6	MONUMENT
7	PC
8	MICROSOFT
9	APPLE
11	SCOTCH

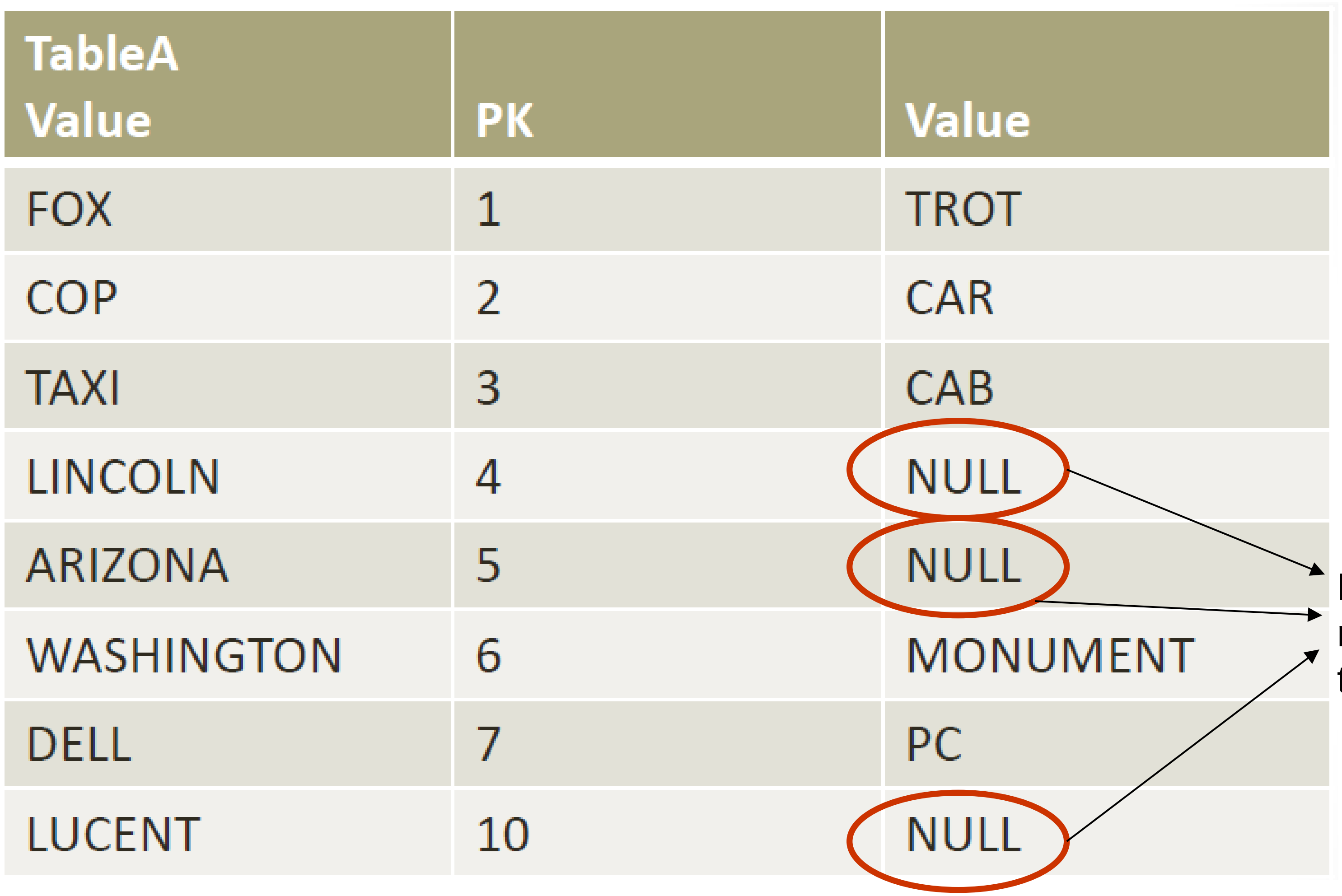
Left Outer Join

- SELECT * FROM TableA **LEFT OUTER JOIN** TableB **ON** TableA.PK = TableB.PK

TableA Value	PK	TableB PK	Value
FOX	1	1	TROT
COP	2	2	CAR
TAXI	3	3	CAB
LINCOLN	4	NULL	NULL
ARIZONA	5	NULL	NULL
WASHINGTON	6	6	MONUMENT
DELL	7	7	PC
LUCENT	10	NULL	NULL

Tables Before Left Outer Join

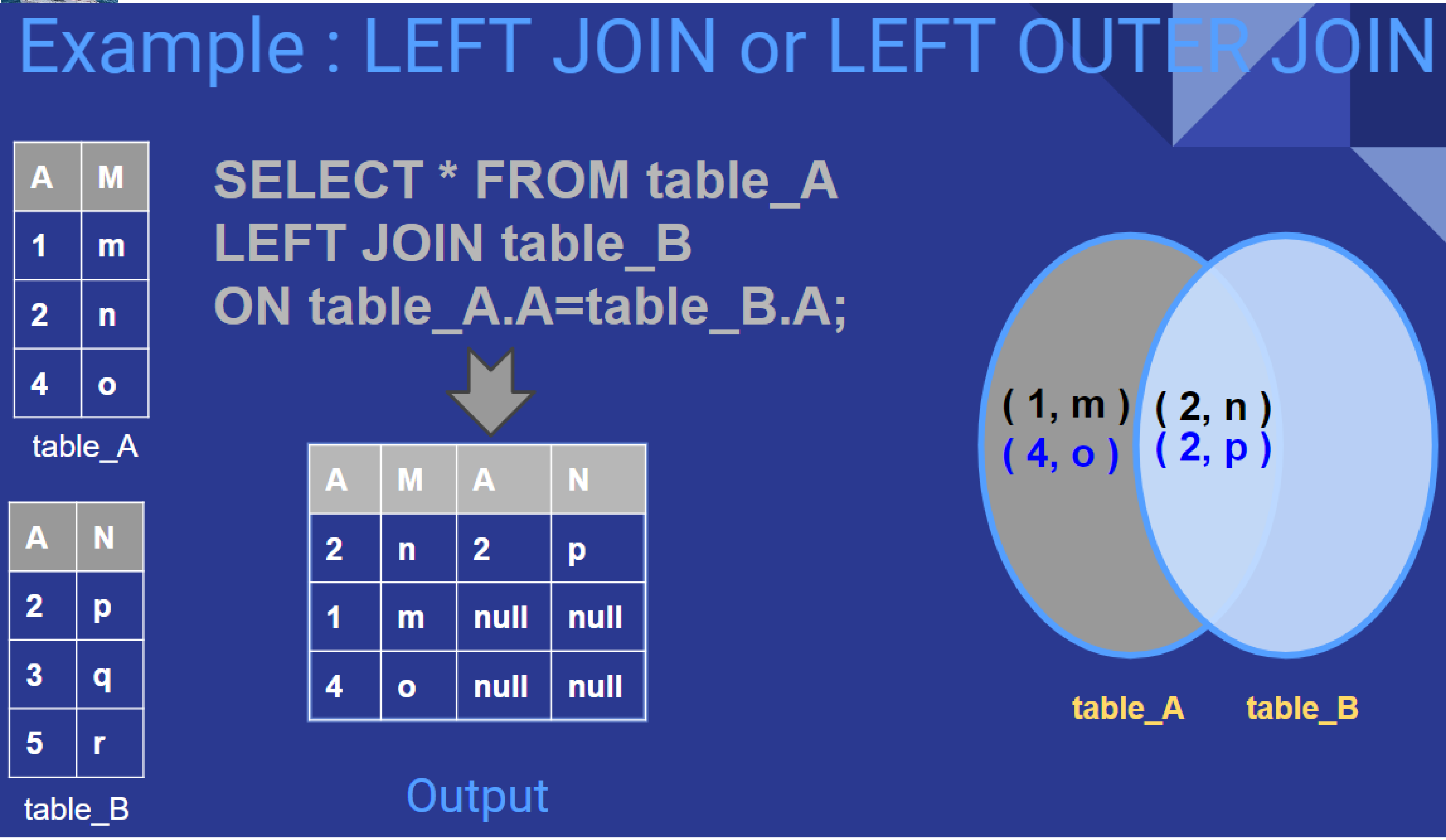
Left Outer Join



TableA Value	PK	Value
FOX	1	TROT
COP	2	CAR
TAXI	3	CAB
LINCOLN	4	NULL
ARIZONA	5	NULL
WASHINGTON	6	MONUMENT
DELL	7	PC
LUCENT	10	NULL

No matching tuples

Left Join Example



Example : LEFT JOIN or LEFT OUTER JOIN

A	M
1	m
2	n
4	o

table_A

A	N
2	p
3	q
5	r

table_B

```
SELECT * FROM table_A
LEFT JOIN table_B
ON table_A.A=table_B.A;
```

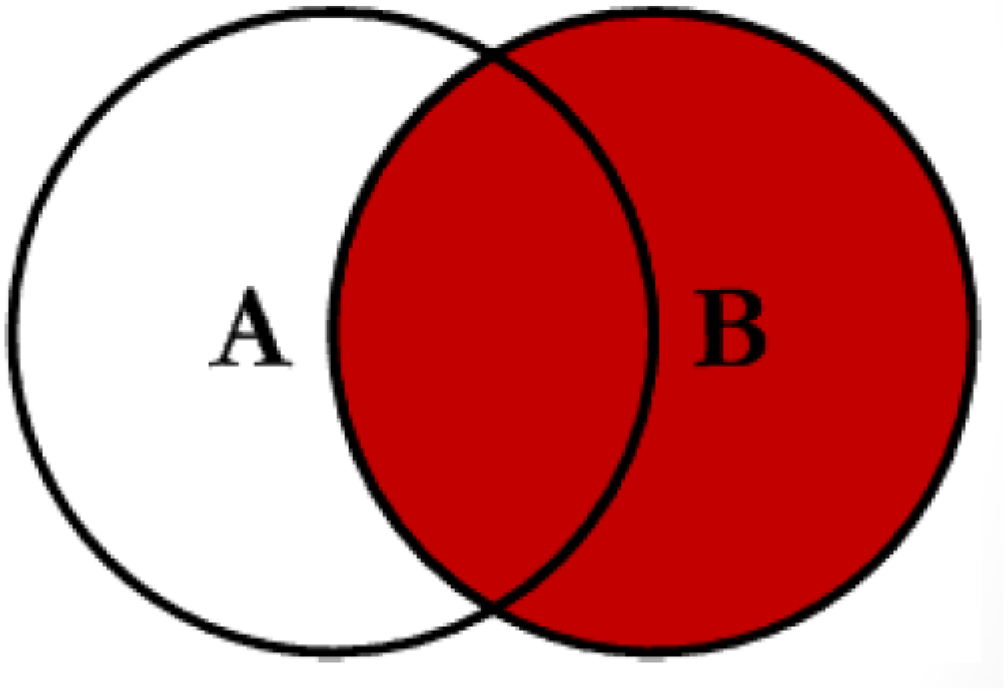
A	M	A	N
2	n	2	p
1	m	null	null
4	o	null	null

Output

(1, m)
(4, o)
(2, n)
(2, p)
table_A
table_B

Right Outer Join

- Right outer join produces a complete set of records from Table B, with the matching records (where available) in Table A. If there is no match, the left side will contain null.



Sample Tables

TableA

PK	Value
1	FOX
2	COP
3	TAXI
6	WASHINGTON
7	DELL
5	ARIZONA
4	LINCOLN
10	LUCENT

TableB

PK	Value
1	TROT
2	CAR
3	CAB
6	MONUMENT
7	PC
8	MICROSOFT
9	APPLE
11	SCOTCH

Right Outer Join

- SELECT * FROM TableA **RIGHT OUTER JOIN** TableB **ON** TableA.PK = TableB.PK

TableA		TableB	
Value	PK	PK	Value
FOX	1	1	TROT
COP	2	2	CAR
TAXI	3	3	CAB
WASHINGTON	6	6	MONUMENT
DELL	7	7	PC
NULL	NULL	8	MICROSOFT
NULL	NULL	9	APPLE
NULL	NULL	11	SCOTCH

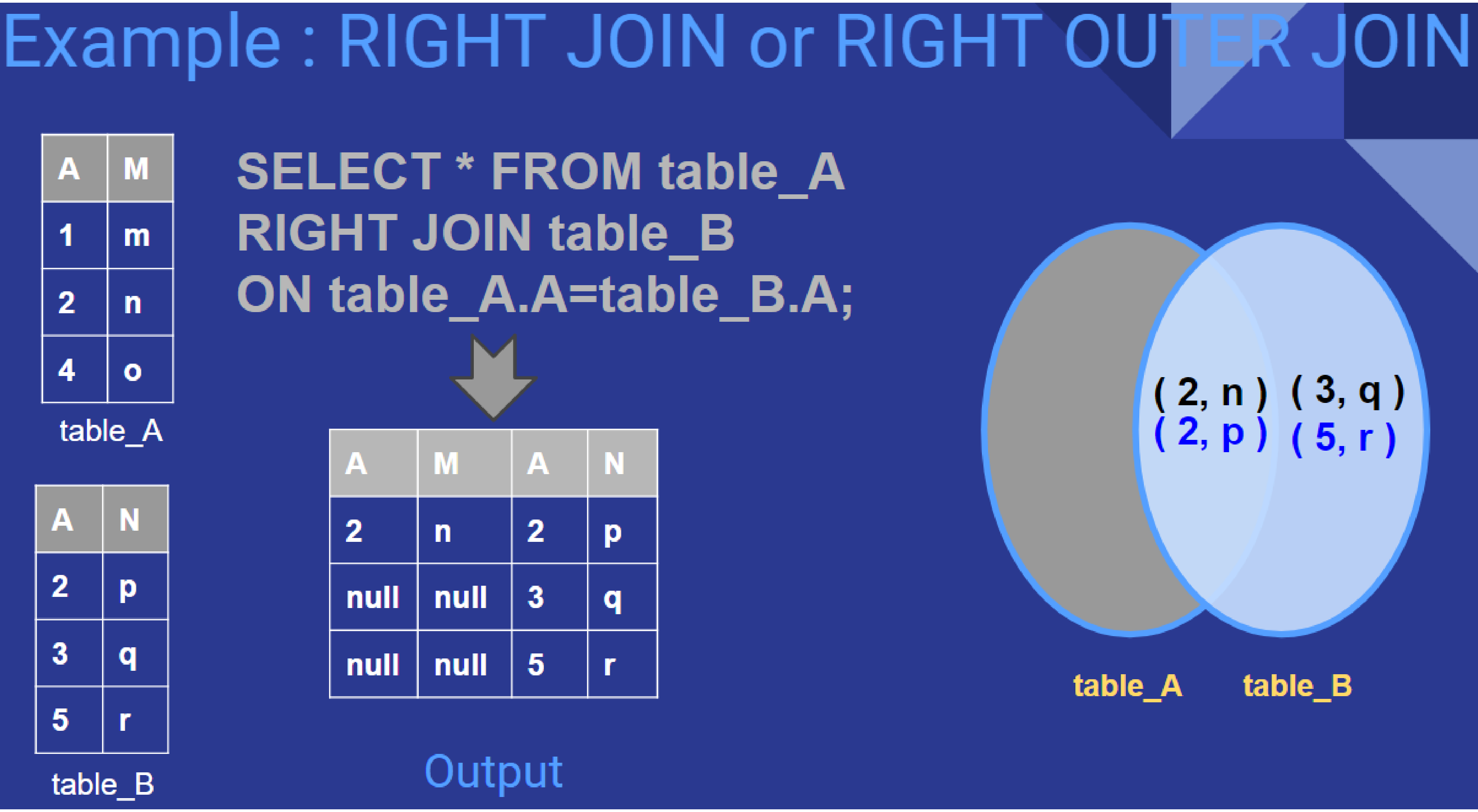
Tables Before Right Outer Join

Right Outer Join

TableA Value	PK	Value
FOX	1	TROT
COP	2	CAR
TAXI	3	CAB
WASHINGTON	6	MONUMENT
DELL	7	PC
NULL	NULL	MICROSOFT
NULL	NULL	APPLE
NULL	NULL	SCOTCH

No matching tuples

Right Join Example



Example : RIGHT JOIN or RIGHT OUTER JOIN

A	M
1	m
2	n
4	o

table_A

A	N
2	p
3	q
5	r

table_B

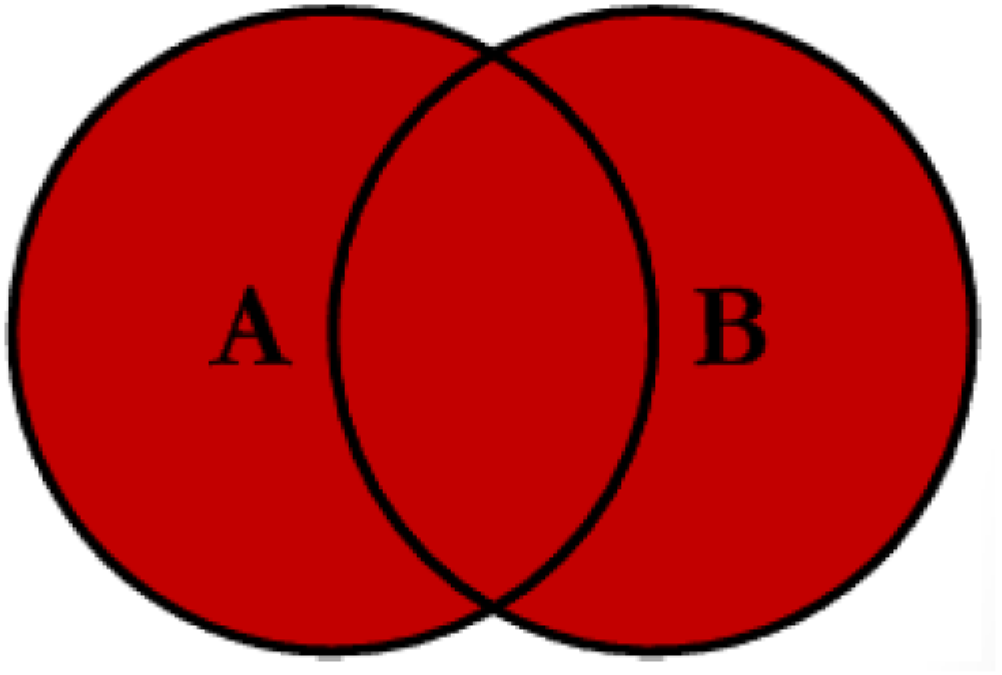
```
SELECT * FROM table_A
RIGHT JOIN table_B
ON table_A.A=table_B.A;
```

A	M	A	N
2	n	2	p
null	null	3	q
null	null	5	r

Output

Full Outer Join

- Full outer join produces the set of all records in Table A and Table B, with matching records from both sides where available. If there is no match, the missing side will contain null.



Sample Tables

TableA

PK	Value
1	FOX
2	COP
3	TAXI
6	WASHINGTON
7	DELL
5	ARIZONA
4	LINCOLN
10	LUCENT

TableB

PK	Value
1	TROT
2	CAR
3	CAB
6	MONUMENT
7	PC
8	MICROSOFT
9	APPLE
11	SCOTCH

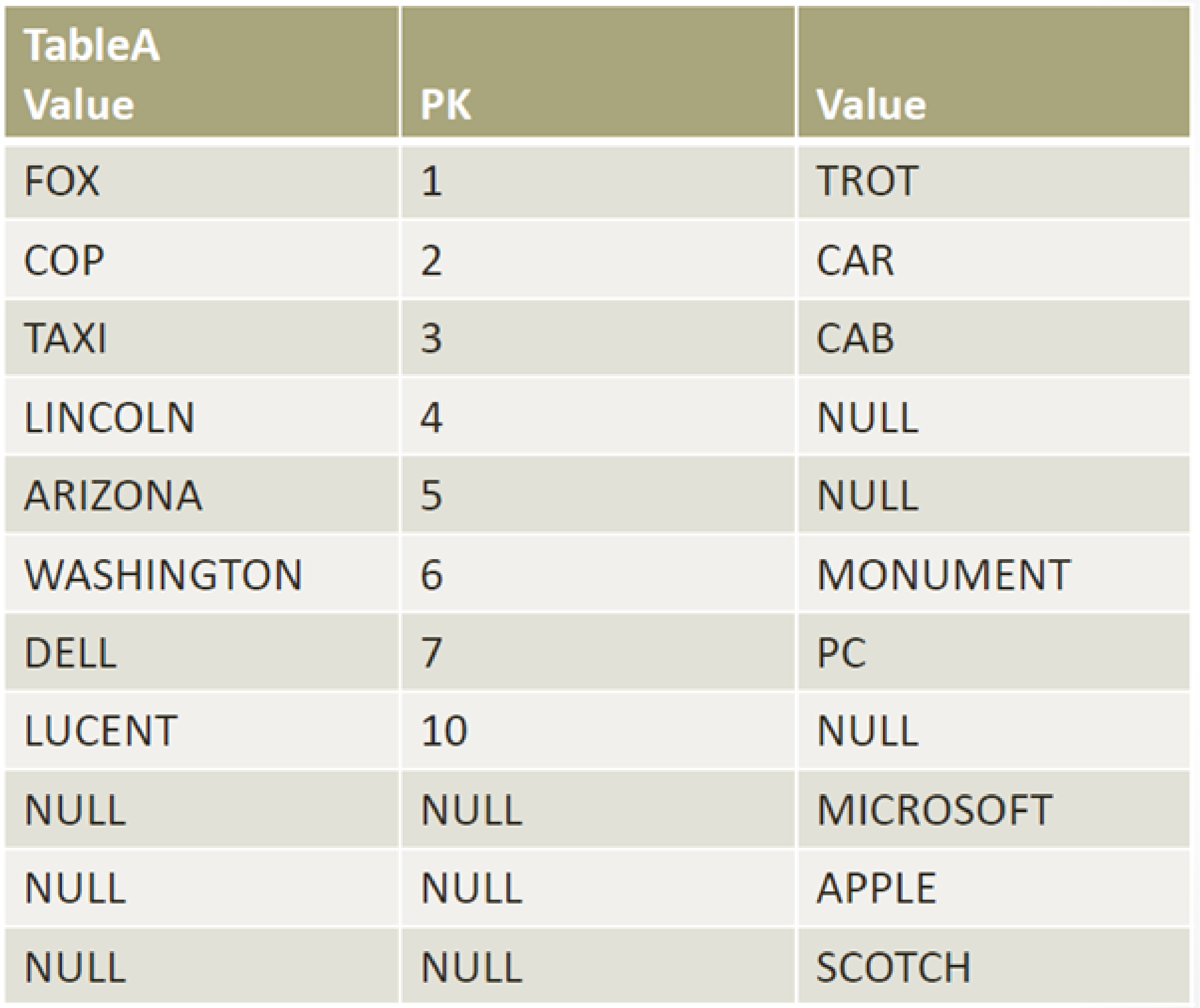
Full Outer Join

- SELECT * FROM TableA **FULL OUTER JOIN** TableB **ON** TableA.PK = TableB.PK

TableA Value	PK	TableB PK	Value
FOX	1	1	TROT
COP	2	2	CAR
TAXI	3	3	CAB
LINCOLN	4	NULL	NULL
ARIZONA	5	NULL	NULL
WASHINGTON	6	6	MONUMENT
DELL	7	7	PC
LUCENT	10	NULL	NULL
NULL	NULL	8	MICROSOFT
NULL	NULL	9	APPLE
NULL	NULL	11	SCOTCH

Tables Before Full Outer Join $\Rightarrow$

Full Outer Join



TableA		
Value	PK	Value
FOX	1	TROT
COP	2	CAR
TAXI	3	CAB
LINCOLN	4	NULL
ARIZONA	5	NULL
WASHINGTON	6	MONUMENT
DELL	7	PC
LUCENT	10	NULL
NULL	NULL	MICROSOFT
NULL	NULL	APPLE
NULL	NULL	SCOTCH

Full Outer Join in MySQL

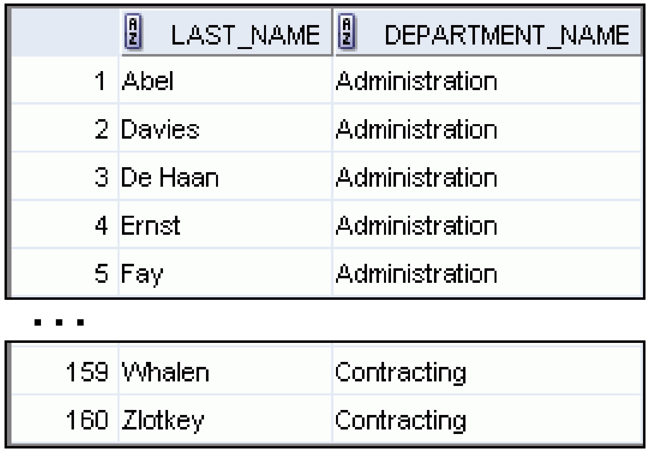
- MySQL **does not have** FULL OUTER JOIN support
- Can be simulated using UNION, LEFT and RIGHT JOINs

```
SELECT * FROM TableA LEFT JOIN TableB ON TableA.PK = TableB.PK
UNION
SELECT * FROM TableA RIGHT JOIN TableB ON TableA.PK = TableB.PK
```

Creating Cross Joins

- The CROSS JOIN clause produces the cross-product of two tables.
- This is also called a Cartesian product between the two tables.

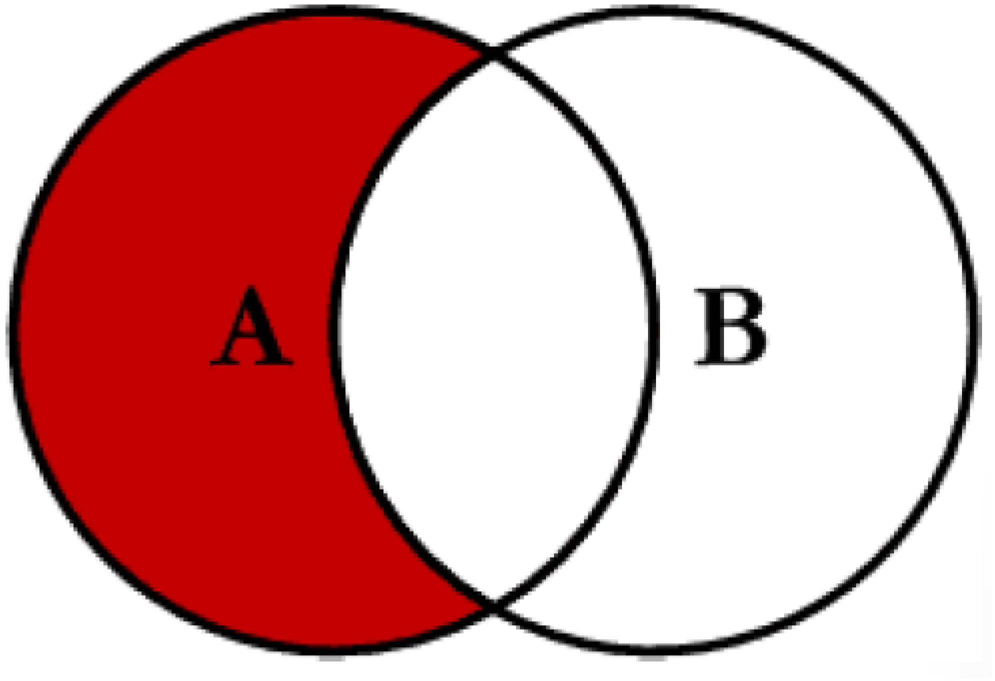
```
SELECT last_name, department_name
FROM   employees
CROSS JOIN departments ;
```



	LAST_NAME	DEPARTMENT_NAME
1	Abel	Administration
2	Davies	Administration
3	De Haan	Administration
4	Ernst	Administration
5	Fay	Administration
...		
159	Whalen	Contracting
160	Zlotkey	Contracting

Left Join Excluding Inner Join

- This query will return all of the records in the left table (table A) that do not match any records in the right table (table B).



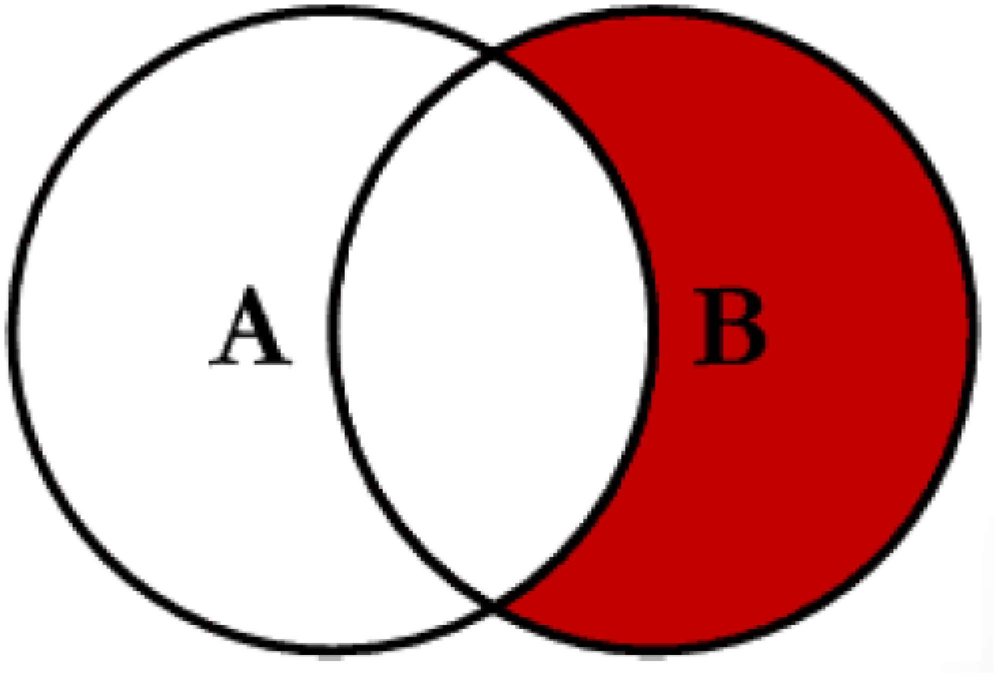
Left Join Excluding Inner Join

- SELECT * FROM TableA LEFT JOIN TableB ON TableA.PK = TableB.PK WHERE TableB.PK IS NULL
- Perform left outer join, then exclude the records we do not want from the right side via a where clause.

TableA		TableB	
Value	PK	PK	Value
LINCOLN	4	NULL	NULL
ARIZONA	5	NULL	NULL
LUCENT	10	NULL	NULL

Right Join Excluding Inner Join

- This query will return all of the records in the right table (table B) that do not match any records in the left table (table A).



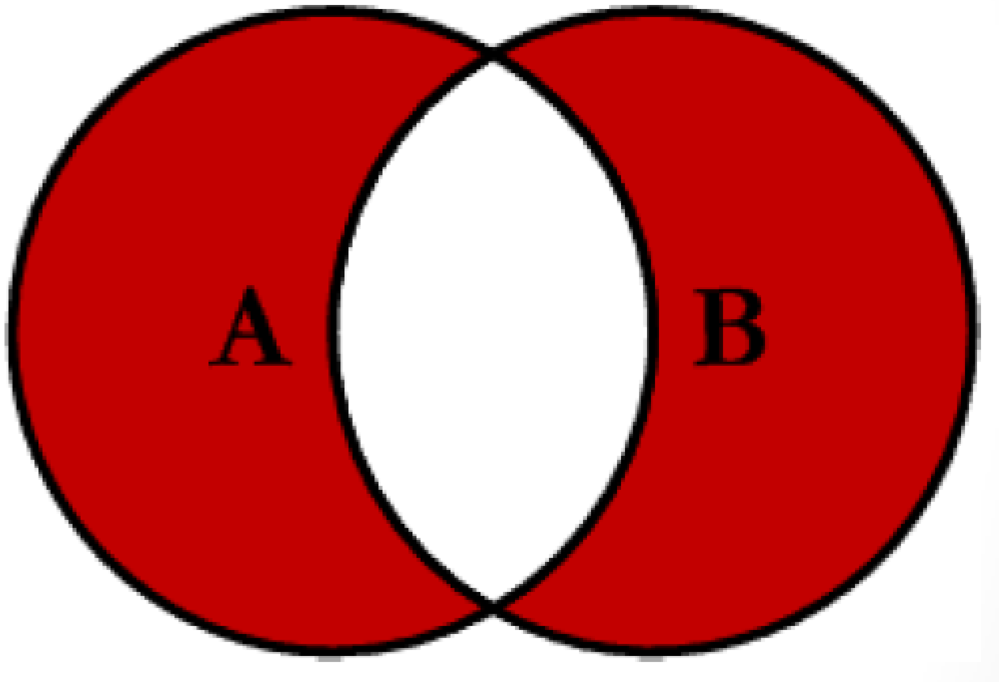
Right Join Excluding Inner Join

- SELECT * FROM TableA **RIGHT JOIN** TableB **ON** TableA.PK = TableB.PK
  WHERE TableA.PK IS NULL

TableA		TableB	
Value	PK	PK	Value
NULL	NULL	8	MICROSOFT
NULL	NULL	9	APPLE
NULL	NULL	11	SCOTCH

Full Outer Join Excluding Outer Join

- This query will return all of the records in Table A and Table B that do not have a matching record in the other table.



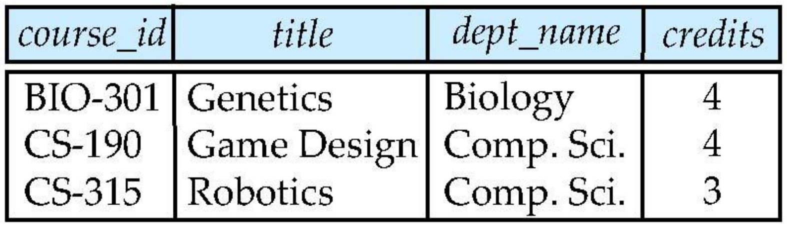
Full Outer Join Excluding Inner Join

- SELECT * FROM TableA **FULL OUTER JOIN** TableB **ON** TableA.PK = TableB.PK
  WHERE TableA.PK IS NULL OR TableB.PK IS NULL

TableA		TableB	
Value	PK	PK	Value
NULL	NULL	8	MICROSOFT
NULL	NULL	9	APPLE
NULL	NULL	11	SCOTCH
LINCOLN	4	NULL	NULL
ARIZONA	5	NULL	NULL
LUCENT	10	NULL	NULL

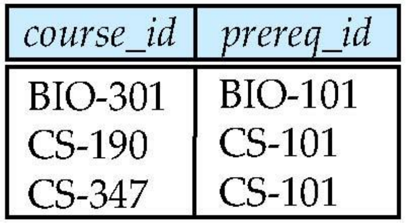
More Outer Join Examples

- Relation (Table) *course*



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

- Relation (Table) *prereq*



course_id	*prereq_id*
BIO-301	BIO-101
CS-190	CS-101
CS-347	CS-101

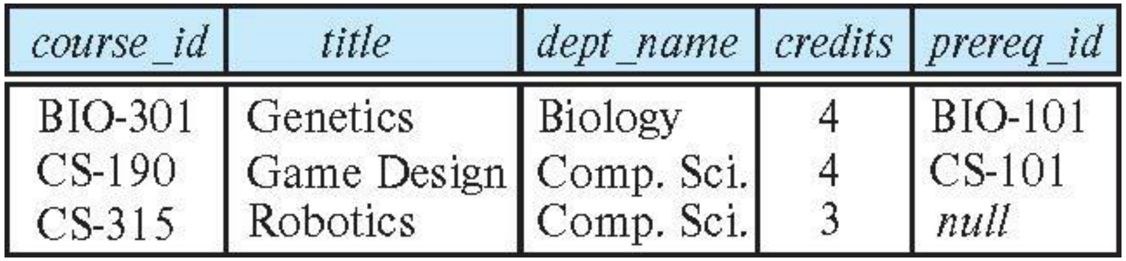
- Observe that

  course information is missing CS-347

  prereq information is missing CS-315

Left Outer Join

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

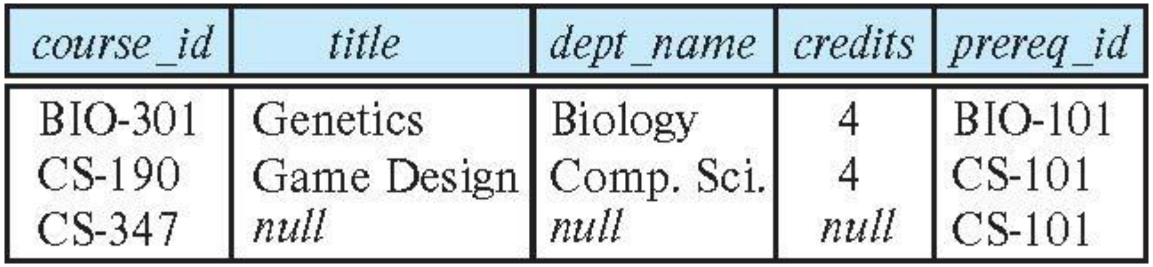


course_id	*title*	*dept_name*	*credits*	*prereq_id*
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-101
CS-315	Robotics	Comp. Sci.	3	*null*

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

Right Outer Join

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

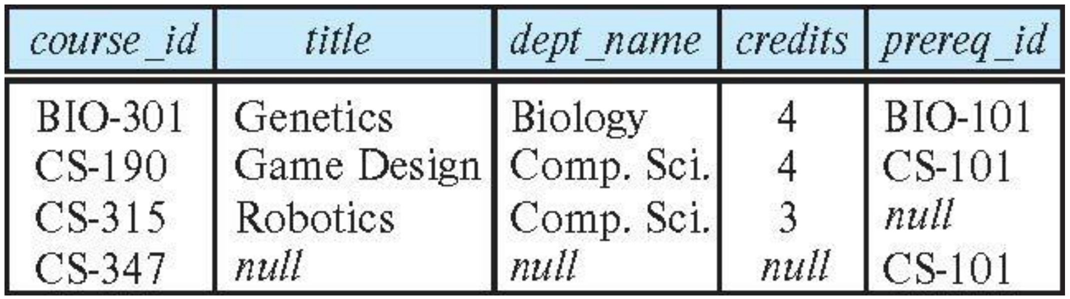


course_id	*title*	*dept_name*	*credits*	*prereq_id*
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-101
CS-347	*null*	*null*	*null*	CS-101

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

Full Outer Join

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

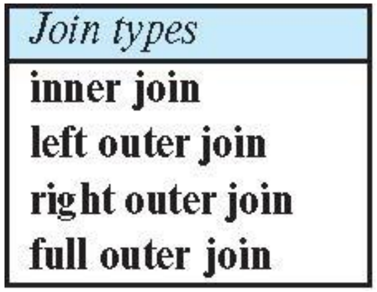


course_id	*title*	*dept_name*	*credits*	*prereq_id*
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-101
CS-315	Robotics	Comp. Sci.	3	*null*
CS-347	*null*	*null*	*null*	CS-101

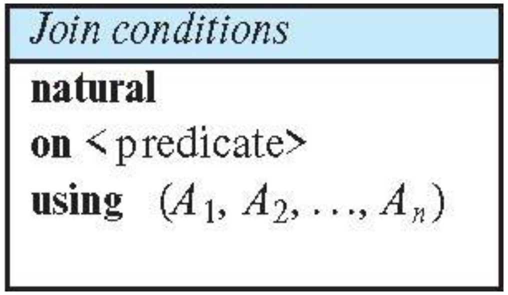
- In relational algebra: *course* ⟗ *prereq*

Joined Types and Conditions

- **Join operations** take two relations and return as a result another relation.
- These additional operations are typically used as subquery expressions in the **from** clause
- **Join condition** – defines which tuples in the two relations match.
- **Join type** – defines how tuples in each relation that do not match any tuple in the other relation (based on the join condition) are treated.



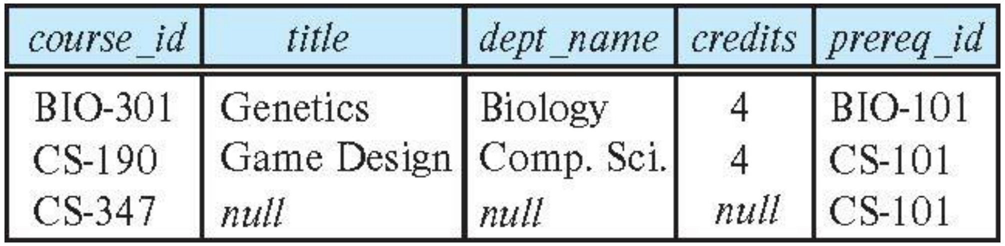
Join types
inner join
left outer join
right outer join
full outer join



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

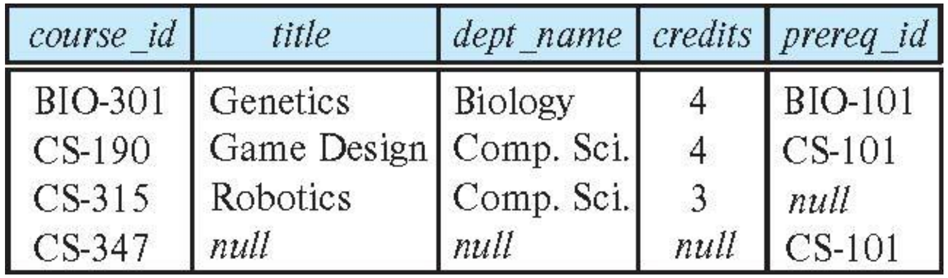
Joined Relations – Examples

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



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

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



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

Joined Relations – Examples

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

course_id	*title*	*dept_name*	*credits*	*prereq_id*	*course_id*
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

course_id	*title*	*dept_name*	*credits*	*prereq_id*	*course_id*
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	*null*	*null*

Joined Relations – Examples

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

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

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

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