SQL - Join

CS 4750
Database Systems

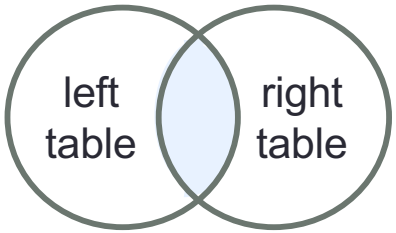
[A. Silberschatz, H. F. Korth, S. Sudarshan, Database System Concepts, Ch.4.1]
[https://www.dofactory.com/sql/join]

JOIN

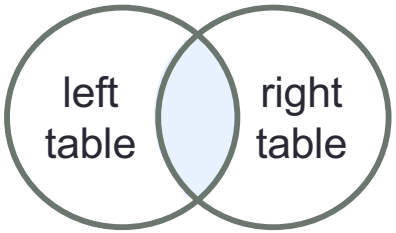
- Combine related tables or sets of data on key values
- All tables (or relations) must have a unique identifier (primary key)
- Any related tables must contain a copy of the unique identifier (foreign key) from the related table

NATURAL JOIN

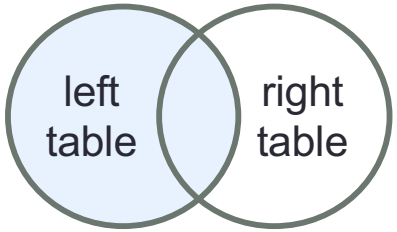
(most common)



(INNER) JOIN

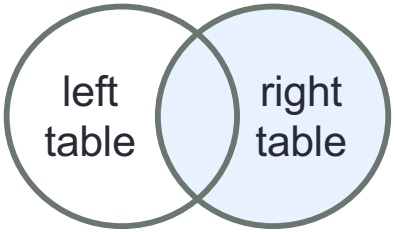


LEFT JOIN



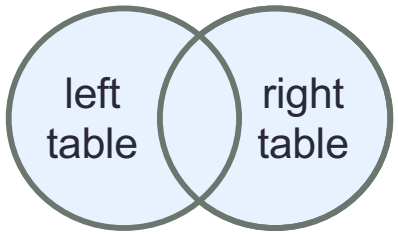
(preserve all rows in left table)

RIGHT JOIN



(preserve all rows in right table)

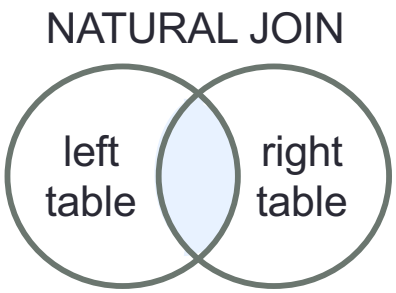
FULL JOIN



(preserve all rows, uncommon, thus skip)

NATURAL JOIN

Not include the repeated column



one(p1, colA, colB)

p1	colA	colB
1	A	aa
2	B	bb
3	C	cc
4	D	dd
5	E	ee
7	G	gg
8	H	hh

two(p2, p1, colX, colY)

p2	p1	colX	colY
9001	1	55	ABC
9002	3	77	PQR
9003	3	95	DEF
9004	7	62	GHI
9005	5	50	KLM
9006	2	42	STU
9007	7	83	XYZ

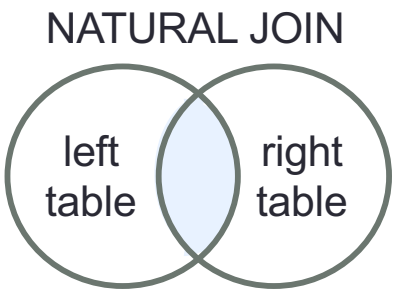
```
SELECT *
FROM one NATURAL JOIN two;
```

Join on common named attribute(s).
If no match found, move on.

```
for each row1 in one:
   for each row2 in two:
      if (row1.p1 = row2.p1):
         output (row1.p1, row1.colA, row1.colB,
                 row2.p2, row2.colX, row2.colY)
```

NATURAL JOIN

Not include the repeated column



one(p1, colA, colB)

p1	colA	colB
1	A	aa
2	B	bb
3	C	cc
4	D	dd
5	E	ee
7	G	gg
8	H	hh

two(p2, p1, colX, colY)

p2	p1	colX	colY
9001	1	55	ABC
9002	3	77	PQR
9003	3	95	DEF
9004	7	62	GHI
9005	5	50	KLM
9006	2	42	STU
9007	7	83	XYZ

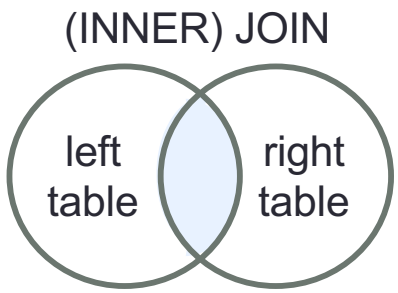
p1	colA	colB	p2	colX	colY
1	A	aa	9001	55	ABC
2	B	bb	9006	42	STU
3	C	cc	9002	77	PQR
3	C	cc	9003	95	DEF
5	E	ee	9005	50	KLM
7	G	gg	9004	62	GHI
7	G	gg	9007	83	XYZ

```
SELECT *
FROM one
NATURAL JOIN two
```

Note the order of the columns; order of rows does not matter

(INNER) JOIN

Include the repeated column



one(p1, colA, colB)

p1	colA	colB
1	A	aa
2	B	bb
3	C	cc
4	D	dd
5	E	ee
7	G	gg
8	H	hh

two(p2, p1, colX, colY)

p2	p1	colX	colY
9001	1	55	ABC
9002	3	77	PQR
9003	3	95	DEF
9004	7	62	GHI
9005	5	50	KLM
9006	2	42	STU
9007	7	83	XYZ

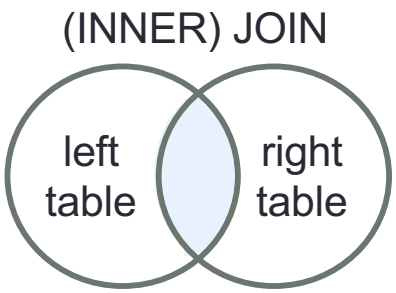
Join on specified attribute(s). If no match found, move on.

```
SELECT *
FROM one JOIN two ON one.p1 = two.p1;
```

```
for each row1 in one:
   for each row2 in two:
      if (row1.p1 = row2.p1):
         output (row1.p1, row1.colA, row1.colB,
                 row2.p2, row2.p1, row2.colX, row2.colY)
```

(INNER) JOIN

Include the repeated column



one(p1, colA, colB)

p1	colA	colB
1	A	aa
2	B	bb
3	C	cc
4	D	dd
5	E	ee
7	G	gg
8	H	hh

two(p2, p1, colX, colY)

p2	p1	colX	colY
9001	1	55	ABC
9002	3	77	PQR
9003	3	95	DEF
9004	7	62	GHI
9005	5	50	KLM
9006	2	42	STU
9007	7	83	XYZ

p1	colA	colB	p2	p1	colX	colY
1	A	aa	9001	1	55	ABC
2	B	bb	9006	2	42	STU
3	C	cc	9002	3	77	PQR
3	C	cc	9003	3	95	DEF
5	E	ee	9005	5	50	KLM
7	G	gg	9004	7	62	GHI
7	G	gg	9007	7	83	XYZ

```
SELECT *
FROM one JOIN two
ON one.p1 = two.p1
```

⇕

```
SELECT *
FROM one INNER JOIN two
ON one.p1 = two.p1
```

Note the order of the columns; order of rows does not matter

Cartesian (Cross) Product

one(p1, colA, colB)

p1	colA	colB
1	A	aa
2	B	bb
3	C	cc
4	D	dd
5	E	ee
7	G	gg
8	H	hh

two(p2, p1, colX, colY)

p2	p1	colX	colY
9001	1	55	ABC
9002	3	77	PQR
9003	3	95	DEF
9004	7	62	GHI
9005	5	50	KLM
9006	2	42	STU
9007	7	83	XYZ

```
SELECT *
FROM one JOIN two;
```

⟺

```
SELECT *
FROM one, two;
```

```
for each row1 in one:
   for each row2 in two:
      output (row1.p1, row1.colA, row1.colB,
              row2.p2, row2.p1, row2.colX, row2.colY)
```

Cartesian (Cross) Product

one(p1, colA, colB)

p1	colA	colB
1	A	aa
2	B	bb
3	C	cc
4	D	dd
5	E	ee
7	G	gg
8	H	hh

two(p2, p1, colX, colY)

p2	p1	colX	colY
9001	1	55	ABC
9002	3	77	PQR
9003	3	95	DEF
9004	7	62	GHI
9005	5	50	KLM
9006	2	42	STU
9007	7	83	XYZ

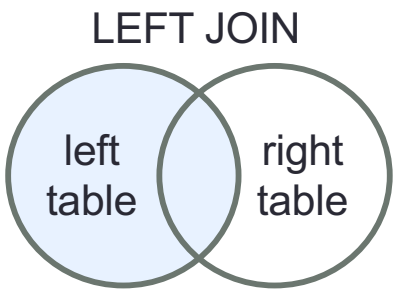
one × two

p1	colA	colB	p2	p1	colX	colY
1	A	aa	9001	1	55	ABC
1	A	aa	9005	5	50	KLM
1	A	aa	9004	7	62	GHI
1	A	aa	9003	3	95	DEF
1	A	aa	9007	7	83	XYZ
1	A	aa	9002	3	77	PQR
1	A	aa	9006	2	42	STU
2	B	bb	9002	3	77	PQR
2	B	bb	9006	2	42	STU
2	B	bb	9001	1	55	ABC
2	B	bb	9005	5	50	KLM
2	B	bb	9004	7	62	GHI
2	B	bb	9003	3	95	DEF
2	B	bb	9007	7	83	XYZ
3	C	cc	9003	3	95	DEF
3	C	cc	9007	7	83	XYZ
…	…	…	…	…	…	…

Note the order of the columns; order of rows does not matter

NATURAL LEFT OUTER JOIN

Augment a join by the dangling tuples



one(p1, colA, colB)

p1	colA	colB
1	A	aa
2	B	bb
3	C	cc
4	D	dd
5	E	ee
7	G	gg
8	H	hh

two(p2, p1, colX, colY)

p2	p1	colX	colY
9001	1	55	ABC
9002	3	77	PQR
9003	3	95	DEF
9004	7	62	GHI
9005	5	50	KLM
9006	2	42	STU
9007	7	83	XYZ

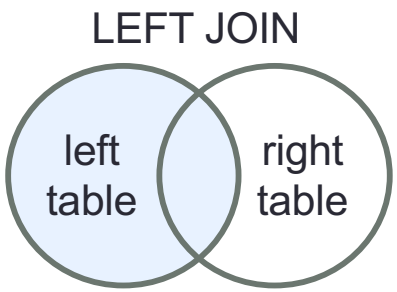
Join on common named attribute(s). If no match found, pad with NULL values

```
SELECT *
FROM one NATURAL LEFT OUTER JOIN two;
```

Preserve all rows in left table

NATURAL LEFT OUTER JOIN

Augment a join by the dangling tuples



`one(p1, colA, colB)`

p1	colA	colB
1	A	aa
2	B	bb
3	C	cc
4	D	dd
5	E	ee
7	G	gg
8	H	hh

`two(p2, p1, colX, colY)`

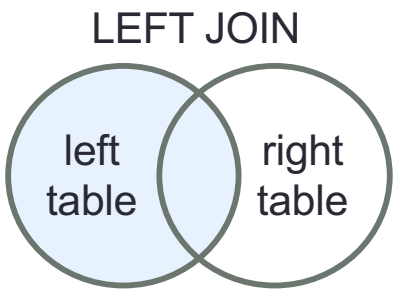
p2	p1	colX	colY
9001	1	55	ABC
9002	3	77	PQR
9003	3	95	DEF
9004	7	62	GHI
9005	5	50	KLM
9006	2	42	STU
9007	7	83	XYZ

p1	colA	colB	p2	colX	colY
1	A	aa	9001	55	ABC
2	B	bb	9006	42	STU
3	C	cc	9002	77	PQR
3	C	cc	9003	95	DEF
4	D	dd	*NULL*	*NULL*	*NULL*
5	E	ee	9005	50	KLM
7	G	gg	9004	62	GHI
7	G	gg	9007	83	XYZ
8	H	hh	*NULL*	*NULL*	*NULL*

Note the order of the columns; order of rows does not matter

LEFT OUTER JOIN

Augment a join by the dangling tuples



one(p1, colA, colB)

p1	colA	colB
1	A	aa
2	B	bb
3	C	cc
4	D	dd
5	E	ee
7	G	gg
8	H	hh

two(p2, p1, colX, colY)

p2	p1	colX	colY
9001	1	55	ABC
9002	3	77	PQR
9003	3	95	DEF
9004	7	62	GHI
9005	5	50	KLM
9006	2	42	STU
9007	7	83	XYZ

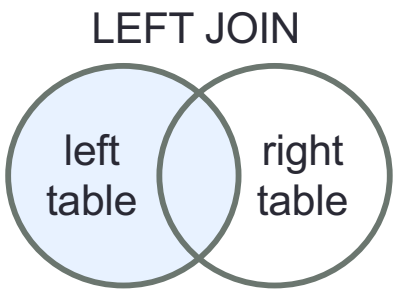
Join on specified attribute(s). If no match found, pad with NULL values

Preserve all rows in left table

```
SELECT *
FROM one LEFT OUTER JOIN two
ON one.p1 = two.p1;
```

LEFT OUTER JOIN

Augment a join by the dangling tuples



one(p1, colA, colB)

p1	colA	colB
1	A	aa
2	B	bb
3	C	cc
4	D	dd
5	E	ee
7	G	gg
8	H	hh

two(p2, p1, colX, colY)

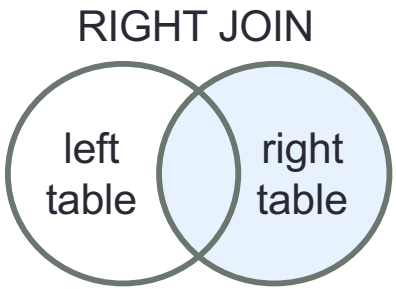
p2	p1	colX	colY
9001	1	55	ABC
9002	3	77	PQR
9003	3	95	DEF
9004	7	62	GHI
9005	5	50	KLM
9006	2	42	STU
9007	7	83	XYZ

p1	colA	colB	p2	p1	colX	colY
1	A	aa	9001	1	55	ABC
2	B	bb	9006	2	42	STU
3	C	cc	9002	3	77	PQR
3	C	cc	9003	3	95	DEF
4	D	dd	*NULL*	*NULL*	*NULL*	*NULL*
5	E	ee	9005	5	50	KLM
7	G	gg	9004	7	62	GHI
7	G	gg	9007	7	83	XYZ
8	H	hh	*NULL*	*NULL*	*NULL*	*NULL*

Note the order of the columns; order of rows does not matter

NATURAL RIGHT OUTER JOIN

Augment a join by the dangling tuples



one(p1, colA, colB)

p1	colA	colB
1	A	aa
2	B	bb
3	C	cc
4	D	dd
5	E	ee
7	G	gg
8	H	hh

two(p2, p1, colX, colY)

p2	p1	colX	colY
9001	1	55	ABC
9002	3	77	PQR
9003	3	95	DEF
9004	7	62	GHI
9005	5	50	KLM
9006	2	42	STU
9007	7	83	XYZ

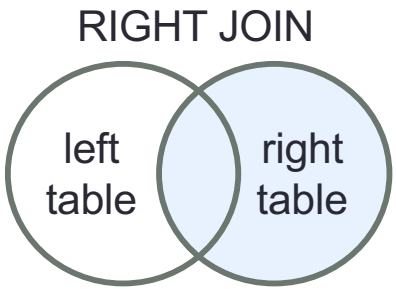
Join on common named attribute(s).
If no match found, pad with NULL values

```
SELECT *
FROM one NATURAL RIGHT OUTER JOIN two;
```

Preserve all rows in right table

NATURAL RIGHT OUTER JOIN

Augment a join by the dangling tuples



one(p1, colA, colB)

p1	colA	colB
1	A	aa
2	B	bb
3	C	cc
4	D	dd
5	E	ee
7	G	gg
8	H	hh

two(p2, p1, colX, colY)

p2	p1	colX	colY
9001	1	55	ABC
9002	3	77	PQR
9003	3	95	DEF
9004	7	62	GHI
9005	5	50	KLM
9006	2	42	STU
9007	7	83	XYZ

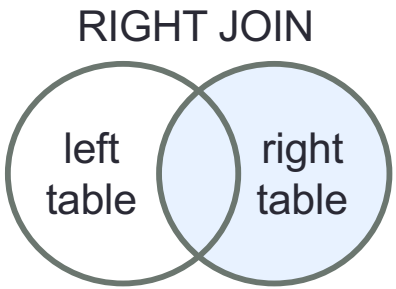
p1	colA	colB	p2	colX	colY
1	A	aa	9001	55	ABC
2	B	bb	9006	42	STU
3	C	cc	9002	77	PQR
3	C	cc	9003	95	DEF
5	E	ee	9005	50	KLM
7	G	gg	9004	62	GHI
7	G	gg	9007	83	XYZ

Since two.p1 is a foreign key referencing one.p1, matches are always founded

Note the order of the columns; order of rows does not matter

RIGHT OUTER JOIN

Augment a join by the dangling tuples



one(p1, colA, colB)

p1	colA	colB
1	A	aa
2	B	bb
3	C	cc
4	D	dd
5	E	ee
7	G	gg
8	H	hh

two(p2, p1, colX, colY)

p2	p1	colX	colY
9001	1	55	ABC
9002	3	77	PQR
9003	3	95	DEF
9004	7	62	GHI
9005	5	50	KLM
9006	2	42	STU
9007	7	83	XYZ

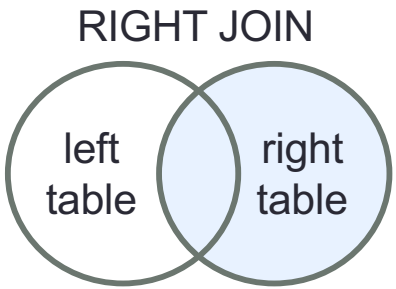
Join on specified attribute(s).
If no match found, pad with NULL values

```
SELECT *
FROM one RIGHT OUTER JOIN two
ON one.p1 = two.p1;
```

Preserve all rows in right table

RIGHT OUTER JOIN

Augment a join by the dangling tuples



one(p1, colA, colB)

p1	colA	colB
1	A	aa
2	B	bb
3	C	cc
4	D	dd
5	E	ee
7	G	gg
8	H	hh

two(p2, p1, colX, colY)

p2	p1	colX	colY
9001	1	55	ABC
9002	3	77	PQR
9003	3	95	DEF
9004	7	62	GHI
9005	5	50	KLM
9006	2	42	STU
9007	7	83	XYZ

p1	colA	colB	p2	p1	colX	colY
1	A	aa	9001	1	55	ABC
2	B	bb	9006	2	42	STU
3	C	cc	9002	3	77	PQR
3	C	cc	9003	3	95	DEF
5	E	ee	9005	5	50	KLM
7	G	gg	9004	7	62	GHI
7	G	gg	9007	7	83	XYZ

Since two.p1 is a foreign key referencing one.p1, matches are always founded

Note the order of the columns; order of rows does not matter

Self Joins

- Join with itself
- Useful when we need multiple copies of the same table or when the table has a FOREIGN KEY which references its own PRIMARY KEY
- Can be viewed as joining two copies of the same table
- To do self join, use aliases

(Challenging) Example: Self Joins

Find names of all TAs who are assigned to Database Systems and Software Analysis

`Hiring(TA_id, Name, Year, Two_week_hours)`

TA_id	Name	Year	Two_week_hours
1234	Minnie	4	20
2345	Humpty	3	24
3456	Dumpty	4	30
3333	Minnie	3	12
5678	Mickey	2	16

`TA_Assignment(TA_id, Course)`

TA_id	Course
1234	Database Systems
2345	Database Systems
3456	Cloud Computing
1234	Software Analysis
5678	Web Programming Lang.
2345	Software Analysis

How should we write SQL?

(Challenging) Example: Self Joins

Find names of all TAs who are assigned to Database Systems and Software Analysis

Hiring(TA_id, Name, Year, Two_week_hours)

TA_id	Name	Year	Two_week_hours
1234	Minnie	4	20
2345	Humpty	3	24
3456	Dumpty	4	30
3333	Minnie	3	12
5678	Mickey	2	16

TA_Assignment(TA_id, Course)

TA_id	Course
1234	Database Systems
2345	Database Systems
3456	Cloud Computing
1234	Software Analysis
5678	Web Programming Lang.
2345	Software Analysis

```
SELECT H.Name
From  Hiring AS H, TA_Assignment AS T
WHERE H.TA_id = T.TA_id AND
      T.Course = "Database Systems" AND
      T.Course = "Software Analysis";
```

Will this work?

No.
Returns empty set

(Challenging) Example: Self Joins

Find names of all TAs who are assigned to Database Systems and Software Analysis

Hiring(TA_id, Name, Year, Two_week_hours)

TA_id	Name	Year	Two_week_hours
1234	Minnie	4	20
2345	Humpty	3	24
3456	Dumpty	4	30
3333	Minnie	3	12
5678	Mickey	2	16

TA_Assignment(TA_id, Course)

TA_id	Course
1234	Database Systems
2345	Database Systems
3456	Cloud Computing
1234	Software Analysis
5678	Web Programming Lang.
2345	Software Analysis

```
SELECT H.Name
From  Hiring AS H, TA_Assignment AS T1,
      TA_Assignment AS T2
WHERE H.TA_id = T1.TA_id AND
      H.TA_id = T2.TA_id AND
      T1.Course = "Database Systems" AND
      T2.Course = "Software Analysis";
```

Will this work?

Name
Minnie
Humpty

(Challenging) Example: Self Joins

Find names of all TAs who are assigned to Database Systems and Software Analysis

Hiring(TA_id, Name, Year, Two_week_hours)

TA_id	Name	Year	Two_week_hours
1234	Minnie	4	20
2345	Humpty	3	24
3456	Dumpty	4	30
3333	Minnie	3	12
5678	Mickey	2	16

TA_Assignment(TA_id, Course)

TA_id	Course
1234	Database Systems
2345	Database Systems
3456	Cloud Computing
1234	Software Analysis
5678	Web Programming Lang.
2345	Software Analysis

```
SELECT H.Name
From   Hiring AS H, TA_Assignment AS T1,
       TA_Assignment AS T2
WHERE H.TA_id = T1.TA_id AND
      H.TA_id = T2.TA_id AND
      T1.Course = "Database Systems" AND
      T2.Course = "Software Analysis";
```

Let's consider each part of the query

All pairs of courses a TA is assigned

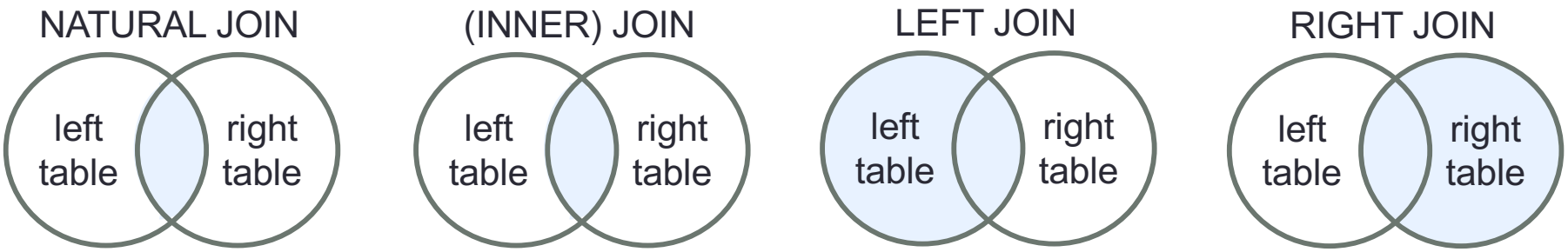
For each pair, check for courses

Wrap-Up

Join combines data across tables

- Nested-loop
- Natural join (most common)
- Inner join (filter Cartesian product)
- Outer joins (preserve non-matching tuples)
- Self join pattern

Different joining techniques can be used to achieve particular goals



What's next?

- SQL – subqueries