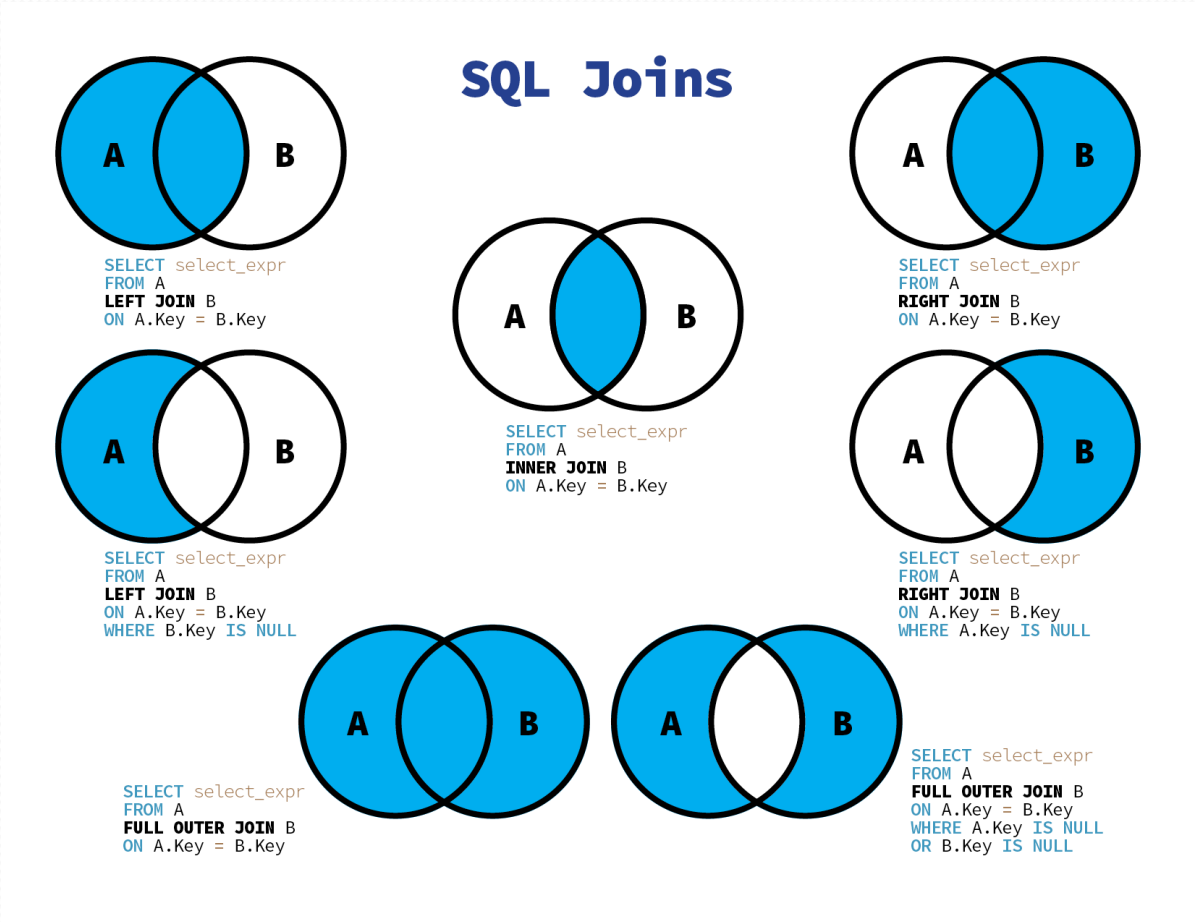
CSc 484
Database Management Systems

Ken Gamradt

Spring 2024

Join

Join


SQL Joins
A
B
SELECT select_expr
FROM A
LEFT JOIN B
ON A.Key = B.Key
SELECT select_expr
FROM A
INNER JOIN B
ON A.Key = B.Key
SELECT select_expr
FROM A
RIGHT JOIN B
ON A.Key = B.Key
SELECT select_expr
FROM A
LEFT JOIN B
ON A.Key = B.Key
WHERE B.Key IS NULL
SELECT select_expr
FROM A
RIGHT JOIN B
ON A.Key = B.Key
WHERE A.Key IS NULL
SELECT select_expr
FROM A
FULL OUTER JOIN B
ON A.Key = B.Key
SELECT select_expr
FROM A
FULL OUTER JOIN B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL

SQLite

teaches

ID	course_id	sec_id	semester	year
10101	CS-101	1	Fall	2017
10101	CS-315	1	Spring	2018
10101	CS-347	1	Fall	2017
12121	FIN-201	1	Spring	2018
15151	MU-199	1	Spring	2018
22222	PHY-101	1	Fall	2017
32343	HIS-351	1	Spring	2018
45565	CS-101	1	Spring	2018
45565	CS-319	1	Spring	2018
76766	BIO-101	1	Summer	2017
76766	BIO-301	1	Summer	2018
83821	CS-190	1	Spring	2017
83821	CS-190	2	Spring	2017
83821	CS-319	2	Spring	2018
98345	EE-181	1	Spring	2017

section

course_id	sec_id	semester	year	building	room_number	time_slot_id
BIO-101	1	Summer	2017	Painter	514	B
BIO-301	1	Summer	2018	Painter	514	A
CS-101	1	Fall	2017	Packard	101	H
CS-101	1	Spring	2018	Packard	101	F
CS-190	1	Spring	2017	Taylor	3128	E
CS-190	2	Spring	2017	Taylor	3128	A
CS-315	1	Spring	2018	Watson	120	D
CS-319	1	Spring	2018	Watson	100	B
CS-319	2	Spring	2018	Taylor	3128	C
CS-347	1	Fall	2017	Taylor	3128	A
EE-181	1	Spring	2017	Taylor	3128	C
FIN-201	1	Spring	2018	Packard	101	B
HIS-351	1	Spring	2018	Painter	514	C
MU-199	1	Spring	2018	Packard	101	D
PHY-101	1	Fall	2017	Watson	100	A

Natural Join

- SQL supports an operation called the NATURAL JOIN
  - Two tuples coming from each relation are combined as a new tuple for the resulting relation only when **attributes with the same name have the same value**

-- SQLite

```
select ID, course_id, building
    from teaches NATURAL JOIN section
```

ID	course_id	building
10101	CS-101	Packard
10101	CS-315	Watson
10101	CS-347	Taylor
12121	FIN-201	Packard
15151	MU-199	Packard
22222	PHY-101	Watson
32343	HIS-351	Painter
45565	CS-101	Packard
45565	CS-319	Watson
76766	BIO-101	Painter
76766	BIO-301	Painter
83821	CS-190	Taylor
83821	CS-190	Taylor
83821	CS-319	Taylor
98345	EE-181	Taylor

- SQL SERVER doesn't support **NATURAL JOIN**
- MySQL and PostgreSQL support **NATURAL JOIN**

Natural Join

- The process of a **NATURAL JOIN** is different from a join
  - Natural join considers only those pairs of tuples with the same value on those attributes that appear in the schemas of both relations

```
select *
    from teaches NATURAL JOIN section
```

ID	course_id	sec_id	semester	year
10101	CS-101	1	Fall	2017
10101	CS-315	1	Spring	2018
10101	CS-347	1	Fall	2017

course_id	sect_id	semester	year	building	room_nu	time_slo
CS-101	1	Fall	2017	Packard	101	H
CS-101	1	Spring	2018	Packard	101	F
CS-190	1	Spring	2017	Taylor	3128	E
CS-190	2	Spring	2017	Taylor	3128	A
CS-315	1	Spring	2018	Watson	120	D

- Matching columns have the same name in both relations

Natural Join

- The process of **NATURAL JOIN**
  - Tuples are combined only when the attributes appearing in both relations have the same values

ID	course_id	sec_id	semester	year
10101	CS-101	1	Fall	2017
10101	CS-315	1	Spring	2018
10101	CS-347	1	Fall	2017

course_id	sect_id	semester	year	building	room_nu	time_slo
CS-101	1	Fall	2017	Packard	101	H
CS-101	1	Spring	2018	Packard	101	F
CS-190	1	Spring	2017	Taylor	3128	E
CS-190	2	Spring	2017	Taylor	3128	A
CS-315	1	Spring	2018	Watson	120	D

course_id	sec_id	semester	year	ID	building	room_num	time_slot_i
CS-101	1	Fall	2017	10101	Packard	101	H
CS-315	1	Spring	2018	10101	Watson	120	D

Natural Join

- The resulting relation does not repeat attributes appearing in both relations
- The order of attributes in result relation:
  - First, attributes common to the schema of both relations
  - Second, attributes unique to the schema of the first relation
  - Lastly, attributes unique to the schema of the second relation

course_id	sec_id	semester	year	ID	building	room_num	time_slot_i
CS-101	1	Fall	2017	10101	Packard	101	H
CS-315	1	Spring	2018	10101	Watson	120	D

VS

```
select *
    from teaches as T, section as S
    where T.course_id = S.course_id and T.sec_id = S.sec_id and
          T.semester = S.semester and T.year = S.year
```

id character var	course_id character var	sec_id character var	semester character var	year numeric (4)	course_id character var	sec_id character var	semester character var	year numeric (4)	building character var	room_numbe character var	time_slot_id character var
10101	CS-101	1	Fall	2017	CS-101	1	Fall	2017	Packard	101	H
10101	CS-315	1	Spring	2018	CS-315	1	Spring	2018	Watson	120	D

Natural Join

E.g., list the name of all instructors and all course_ids they have taught

instructor

id	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000.00
12121	Wu	Finance	90000.00
15151	Mozart	Music	40000.00
22222	Einstein	Physics	95000.00
32343	El Said	History	60000.00
33456	Gold	Physics	87000.00
45565	Katz	Comp. Sci.	75000.00
58583	Califieri	History	62000.00
76543	Singh	Finance	80000.00
76766	Crick	Biology	72000.00
83821	Brandt	Comp. Sci.	92000.00
98345	Kim	Elec. Eng.	80000.00

teaches

id	course_id	sec_id	semester	year
10101	CS-101	1	Fall	2017
10101	CS-315	1	Spring	2018
10101	CS-347	1	Fall	2017
12121	FIN-201	1	Spring	2018
15151	MU-199	1	Spring	2018
22222	PHY-101	1	Fall	2017
32343	HIS-351	1	Spring	2018
45565	CS-101	1	Spring	2018
45565	CS-319	1	Spring	2018
76766	BIO-101	1	Summer	2017
76766	BIO-301	1	Summer	2018
83821	CS-190	1	Spring	2017
83821	CS-190	2	Spring	2017
83821	CS-319	2	Spring	2018
98345	EE-181	1	Spring	2017

Natural Join

E.g., list the name of all instructors and all course_ids they have taught

Query | Query History

```
select distinct name, course_id
    from instructor as I, teaches as T
    where I.id = T.id;

select distinct name, course_id
    from instructor as I join teaches as T
    on (I.id = T.id);
```

Data Output | Messages | Notifications

	name character varying (20)	course_id character varying (8)
1	Srinivasan	CS-347
2	Katz	CS-101
3	Crick	BIO-301
4	El Said	HIS-351
5	Srinivasan	CS-315
6	Kim	EE-181
7	Einstein	PHY-101
8	Brandt	CS-190
9	Srinivasan	CS-101
10	Brandt	CS-319
11	Crick	BIO-101
12	Katz	CS-319
13	Wu	FIN-201
14	Mozart	MU-199

Natural Join

E.G., find all student IDs who have taken course sections taught by the instructor with ID 10101

teaches

id [PK] characte	course_id [PK] characte	sec_id [PK] characte	semester [PK] characte	year [PK] numeric
10101	CS-101	1	Fall	2017
10101	CS-315	1	Spring	2018
10101	CS-347	1	Fall	2017
12121	FIN-201	1	Spring	2018
15151	MU-199	1	Spring	2018
22222	PHY-101	1	Fall	2017
32343	HIS-351	1	Spring	2018
45565	CS-101	1	Spring	2018
45565	CS-319	1	Spring	2018
76766	BIO-101	1	Summer	2017
76766	BIO-301	1	Summer	2018
83821	CS-190	1	Spring	2017
83821	CS-190	2	Spring	2017
83821	CS-319	2	Spring	2018
98345	EE-181	1	Spring	2017

takes

id [PK] characte	course_id [PK] characte	sec_id [PK] characte	semester [PK] characte	year [PK] numeric	grade character var
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]

Natural Join

- Solution
  - Natural join???

teaches (**ID**, **course_id**, **sec_id**, **semester**, **year**) -- instructor ID

takes (**ID**, **course_id**, **sec_id**, **semester**, **year**, grade) -- student ID

A natural join operation will combine the tuples with all attributes having the same name

Join – using

- Solution
  - A form of natural join construct that allows you to specify exactly which columns should be equated

```
                                    -- inner join
select distinct takes.id            -- SQL Server does not support USING
    from teaches join takes using ( course_id, sec_id, semester, year )
    where teaches.id = '10101';
```

- Specify the list of attribute names to be equated when joining
- Both relations being joined must have attributes with the specified names

- Benefit: provide the benefit of natural join while avoiding the danger of equating attributes erroneously

Join

- E.g., list the name of the instructors/advisors and the number of students each advisor has

```
select name, count(s_id) as NumOfStudent
    from instructor, advisor
    where id = i_id
    group by name          -- advisor
```

s_id	i_id
00128	45565
12345	10101
23121	76543
44553	22222
45678	22222
76543	45565
76653	98345
98765	98345
98988	76766

-- instructor

id	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000.00
12121	Wu	Finance	90000.00
15151	Mozart	Music	40000.00
22222	Einstein	Physics	95000.00
32343	El Said	History	60000.00
33456	Gold	Physics	87000.00
45565	Katz	Comp. Sci.	75000.00
58583	Califieri	History	62000.00
76543	Singh	Finance	80000.00
76766	Crick	Biology	72000.00
83821	Brandt	Comp. Sci.	92000.00
98345	Kim	Elec. Eng.	80000.00

Join – On

- The **ON** condition allows a general predicate over the relations being joined
  - The predicate is written like a **WHERE** clause predicate except for the use of the **ON** keyword

```
select name, count(s_id)
    from instructor join advisor ON (id = i_id) -- prefix should be used when
    group by name;                              -- attributes from two relations
                                                -- have the same name
```

- **JOIN**…**ON**… statement can be replaced by moving the predicate in the **ON** clause to the **WHERE** clause

Join – On

E.g., list the student's id, name, course_id, and grade of each course they have taken

student

id [PK] characte	name character var	dept_name character var	tot_cred numeric (3)
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

id [PK] characte	course_id [PK] characte	sec_id [PK] characte	semester [PK] characte	year [PK] numeric	grade character var
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]

Join – On

- The resulting relation is the same as the statement using the **WHERE** clause to specify the join condition
  - All attributes from the two relations maintained

```
select *
    from student join takes ON (student.id = takes.id)  -- inner join
```

id character var	name character var	dept_name character var	tot_cred numeric (3)	id character var	course_id character var	sec_id character var	semester character var	year numeric (4)	grade character var
00128	Zhang	Comp. Sci.	102	00128	CS-101	1	Fall	2017	A
00128	Zhang	Comp. Sci.	102	00128	CS-347	1	Fall	2017	A-
12345	Shankar	Comp. Sci.	32	12345	CS-101	1	Fall	2017	C
12345	Shankar	Comp. Sci.	32	12345	CS-190	2	Spring	2017	A
12345	Shankar	Comp. Sci.	32	12345	CS-315	1	Spring	2018	A
12345	Shankar	Comp. Sci.	32	12345	CS-347	1	Fall	2017	A
19991	Brandt	History	80	19991	HIS-351	1	Spring	2018	B
23121	Chavez	Finance	110	23121	FIN-201	1	Spring	2018	C+
44553	Peltier	Physics	56	44553	PHY-101	1	Fall	2017	B-
45678	Levy	Physics	46	45678	CS-101	1	Fall	2017	F
45678	Levy	Physics	46	45678	CS-101	1	Spring	2018	B+
45678	Levy	Physics	46	45678	CS-319	1	Spring	2018	B
54321	Williams	Comp. Sci.	54	54321	CS-101	1	Fall	2017	A-
54321	Williams	Comp. Sci.	54	54321	CS-190	2	Spring	2017	B+

Three-table join

- E.g., list the student's name together with their advisor's name

instructor

id	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000.00
12121	Wu	Finance	90000.00
15151	Mozart	Music	40000.00
22222	Einstein	Physics	95000.00
32343	El Said	History	60000.00
33456	Gold	Physics	87000.00
45565	Katz	Comp. Sci.	75000.00
58583	Califieri	History	62000.00
76543	Singh	Finance	80000.00
76766	Crick	Biology	72000.00
83821	Brandt	Comp. Sci.	92000.00
98345	Kim	Elec. Eng.	80000.00

student

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

advisor

s_id	i_id
00128	45565
12345	10101
23121	76543
44553	22222
45678	22222
76543	45565
76653	98345
98765	98345
98988	76766

Three-table join

- E.g., list the student's name together with their advisor's name

```
select I.name as Advisor, S.name as Student
    from instructor as I, student as S, advisor as A
    where I.ID = A.i_ID and S.ID = A.s_ID
```

advisor character var	student character var
Katz	Zhang
Srinivasan	Shankar
Singh	Chavez
Einstein	Peltier
Einstein	Levy
Katz	Brown
Kim	Aoi
Kim	Bourikas
Crick	Tanaka

Three-table join

- Apply **JOIN**...**ON**...
  - The result of the JOIN is a relation

```
select I.name as Advisor, S.name as Student
    from instructor as I join advisor as A on (I.ID = A.i_ID)
    -- perform join of instructor and advisor
    -- temp relation
        join student as S on (S.ID = A.s_ID);
        -- perform join of temp and student
```

id character var	name character var	dept_name character var	salary numeric (8,2)	s_id character var	i_id character var
45565	Katz	Comp. Sci.	75000.00	00128	45565
10101	Srinivasan	Comp. Sci.	65000.00	12345	10101
76543	Singh	Finance	80000.00	23121	76543
22222	Einstein	Physics	95000.00	44553	22222
22222	Einstein	Physics	95000.00	45678	22222
45565	Katz	Comp. Sci.	75000.00	76543	45565
98345	Kim	Elec. Eng.	80000.00	76653	98345
98345	Kim	Elec. Eng.	80000.00	98765	98345
76766	Crick	Biology	72000.00	98988	76766

advisor character var	student character var
Katz	Zhang
Srinivasan	Shankar
Singh	Chavez
Einstein	Peltier
Einstein	Levy
Katz	Brown
Kim	Aoi
Kim	Bourikas
Crick	Tanaka

Outer join

- E.g., list all the instructor's name and the number of students they have if they are advisors

```
select name, count(s_id) as NumOfStudents
    from instructor as I, advisor as A
    where id = i_id
    group by name
```

s_id	i_id
00128	45565
12345	10101
23121	76543
44553	22222
45678	22222
76543	45565
76653	98345
98765	98345
98988	76766

id	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000.00
12121	Wu	Finance	90000.00
15151	Mozart	Music	40000.00
22222	Einstein	Physics	95000.00
32343	El Said	History	60000.00
33456	Gold	Physics	87000.00
45565	Katz	Comp. Sci.	75000.00
58583	Califieri	History	62000.00
76543	Singh	Finance	80000.00
76766	Crick	Biology	72000.00
83821	Brandt	Comp. Sci.	92000.00
98345	Kim	Elec. Eng.	80000.00

name	numofstudents
Einstein	2
Katz	2
Singh	1
Kim	2
Srinivasan	1
Crick	1

Outer join

- The outer join operation works in a manner similar to the join operation, but it preserve those tuples that would be lost in a join by creating tuples in the result containing null values
- Three forms of outer join
  - **Left outer join**: preserve tuples only in the relation to the **left of** the left outer join operation
    - **Left join**
  - **Right outer join**: preserve tuples only in the relation to the **right of** the right outer join operation
    - **Right join**
  - **Full outer join**: preserve tuples in **both** relations
    - **Full join**
- The join operation we discussed previously are also called inner-join
  - Distinguish from the outer-join

Left Outer join

```
select *                -- left join
    from instructor left outer join advisor on (id = i_id);
```

- How left outer join operates

Instructor

	ID	name	dept_name	salary
1	10101	Srinivasan	Comp. Sci.	65000.00
2	12121	Wu	Finance	90000.00
3	15151	Mozart	Music	40000.00
4	22222	Einstein	Physics	95000.00

Advisor

	s_ID	i_ID
1	00128	45565
2	12345	10101
3	23121	76543
4	44553	22222
5	45678	22222
6	76543	45565

- Step 1: compute the result of the inner join

ID	name	dept_nam	salary	s_ID	i_ID
10101	Srinivasan	Comp. Sci.	65000	12345	10101
22222	Einstein	Physics	95000	44553	22222
22222	Einstein	Physics	95000	45678	22222

Left Outer join

```
select *                -- left join
    from instructor left outer join advisor on (id = i_id);
```

- Step 2: for every tuple t in the left-side relation that doesn't match any tuple in the right-side relation, add a tuple r to the result constructed as follows:
  - The attributes of tuple r that are derived from the left-side relation are filled in with the values from tuple t
  - The remaining attributes of r are filled with null values

PostgreSQL

ID	name	dept_nam	salary	s_ID	i_ID
10101	Srinivasan	Comp. Sci.	65000	12345	10101
22222	Einstein	Physics	95000	44553	22222
22222	Einstein	Physics	95000	45678	22222
12121	Wu	Finance	90000	null	null
15151	Mozart	Music	40000	null	null

Unordered

SQL Server

ID	name	dept_name	salary	s_ID	i_ID
10101	Srinivasan	Comp. Sci.	65000.00	12345	10101
12121	Wu	Finance	90000.00	NULL	NULL
15151	Mozart	Music	40000.00	NULL	NULL
22222	Einstein	Physics	95000.00	44553	22222
22222	Einstein	Physics	95000.00	45678	22222

Right Outer join

```
select *                -- right join
    from instructor right outer join advisor on (id = i_id);
```

- How right outer join operates

Instructor

	ID	name	dept_name	salary
1	10101	Srinivasan	Comp. Sci.	65000.00
2	12121	Wu	Finance	90000.00
3	15151	Mozart	Music	40000.00
4	22222	Einstein	Physics	95000.00

Advisor

	s_ID	i_ID
1	00128	45565
2	12345	10101
3	23121	76543
4	44553	22222
5	45678	22222
6	76543	45565

- Step 1: computer the result of the inner join

ID	name	dept_nam	salary	s_ID	i_ID
10101	Srinivasan	Comp. Sci.	65000	12345	10101
22222	Einstein	Physics	95000	44553	22222
22222	Einstein	Physics	95000	45678	22222

Right Outer join

```
select *                -- right join
    from instructor right outer join advisor on (id = i_id);
```

- Step 2: tuples from the right-side relation that do not match any tuple in the left-side relation are padded with nulls and are added to the result

id	name	dept_name	salary	s_ID	i_ID
NULL	NULL	NULL	NULL	00128	45565
10101	Srinivasan	Comp. Sci.	65000.00	12345	10101
NULL	NULL	NULL	NULL	23121	76543
22222	Einstein	Physics	95000.00	44553	22222
22222	Einstein	Physics	95000.00	45678	22222
NULL	NULL	NULL	NULL	76543	45565

Full Outer join

```
select *                -- full join
    from instructor full outer join advisor on (id = i_id);
```

- A combination of the left and right outer-join types
- Step 1: computer the result of the inner join

ID	name	dept_nam	salary	s_ID	i_ID
10101	Srinivasan	Comp. Sci.	65000	12345	10101
22222	Einstein	Physics	95000	44553	22222
22222	Einstein	Physics	95000	45678	22222

Full Outer join

```
select *                -- full join
    from instructor full outer join advisor on (id = i_id);
```

- Step 2: extend with nulls those tuples from the left-hand-side relation and right-hand-side relation that did not match with any from another relation

Instructor

	ID	name	dept_name	salary
1	10101	Srinivasan	Comp. Sci.	65000.00
2	12121	Wu	Finance	90000.00
3	15151	Mozart	Music	40000.00
4	22222	Einstein	Physics	95000.00

Advisor

	s_ID	i_ID
1	00128	45565
2	12345	10101
3	23121	76543
4	44553	22222
5	45678	22222
6	76543	45565

Full Outer join

```
select *                -- full join
    from instructor full outer join advisor on (id = i_id);
```

- Step 2: extend with nulls those tuples from the left-side relation and right-side relation that did not match with any from another relation

id	name	dept_name	salary	s_ID	i_ID
10101	Srinivasan	Comp. Sci.	65000.00	12345	10101
12121	Wu	Fubabce	90000.00	NULL	NULL
15151	Mozart	Music	40000.00	NULL	NULL
22222	Einstein	Physics	95000.00	44553	22222
22222	Einstein	Physics	95000.00	45678	22222
NULL	NULL	NULL	NULL	00128	45565
NULL	NULL	NULL	NULL	23121	76543
NULL	NULL	NULL	NULL	76543	45565

Acknowledgements

- Database System Concepts – SQLite
  - https://www.db-book.com/university-lab-dir/sqljs.html
- Quora.com
  - https://www.quora.com/Does-the-join-order-matter-in-SQL
- WIKIPEDIA
  - https://en.wikipedia.org/wiki/SQL