JOINS AND SUB QUERIES

Engr. Laraib Siddiqui

Join

- Join is a special form of cross product of two tables.
- In Cartesian product we join a tuple of one table with the tuples of the second table. But in join there is a special requirement of relationship between tuples.
- For example if there is a relation STUDENT and a relation BOOK then it may be required to know that how many books have been issued to any particular student. Now in this case the primary key of STUDENT that is stId is a foreign key in BOOK table through which the join can be made.

Natural Join

- Natural join (⋈) is a binary operator that is written as ($R \bowtie S$) where R and S are relations.
- It considers only those pairs of tuples with the same value on those attributes that appear in the schemas of both 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

Natural Join (Example)

```
SELECT *
FROM Employee NATURAL JOIN Department;
```

DepartmentID	Employee.LastName	Department.DepartmentName
34	Smith	Clerical
33	Jones	Engineering
34	Robinson	Clerical
33	Steinberg	Engineering
31	Rafferty	Sales

Outer Join

- An **outer join** does not require each record in the two joined tables to have a matching record.
- The joined table retains each record—even if no other matching record exists.
- Outer joins subdivide further into left outer joins, right outer joins, and full outer joins, depending on which table(s) one retains the rows from (left, right, or both).

Left Outer Join

- The result of a **left outer join** (or simply **left join**) for table A and B always contains all records of the "left" table (A), even if the join-condition does not find any matching record in the "right" table (B).
- This means that if the ON clause matches o (zero) records in B, the join will still return a row in the result—but with NULL in each column from B.
- This means that a **left outer join** returns all the values from the left table, plus matched values from the right table (or NULL in case of no matching join predicate).
- If the right table returns one row and the left table returns more than one matching row for it, the values in the right table will be repeated for each distinct row on the left table.

Customer : Table

	CustID	Name
+	1	Brian
+	2	Issa
+	4	Rick
+	5	Kelley
+	6	Mike

Job : Table

	JobID	CustID	Employer
+	13	1	Squirrel Mart
+	14	2	MaliZone
+	15	9	Dilbert Inc
+	16	5	Stonehenge Ltd
+	17	4	OOPs Consulting

Tables to be joined

Left [outer] Join

```
SELECT *
FROM Customer C LEFT JOIN Job J
ON C.CustID = J.CustID
```

SQL command being executed

Values that do not match join condition (will be excluded)

Matching values exist in both tables

Left Join : Select Query

	C.CustID	Name	JobID	J.CustID	Employer
	1	Brian	13	1	Squirrel Mart
	2	Issa	14	2	MaliZone
	4	Rick	17	4	OOPs Consulting
	5	Kelley	16	5	Stonehenge Ltd
	6	Mike			

No matching value in other table, match to NULL instead

NULL value; No matching value was found

Right Outer Join

- A **right outer join** (or **right join**) closely resembles a left outer join, except with the treatment of the tables reversed.
- Every row from the "right" table (B) will appear in the joined table at least once. If no matching row from the "left" table (A) exists, NULL will appear in columns from A for those records that have no match in B.

Customer : Table

	CustID	Name
+	1	Brian
+	2	Issa
+	4	Rick
+	5	Kelley
+	6	Mike

Job : Table

	JobID	CustID	Employer
+	13	1	Squirrel Mart
+	14	2	MaliZone
+	15	9	Dilbert Inc
+	16	5	Stonehenge Ltd
+	17	4	OOPs Consulting

Tables to be joined

Right [outer] Join

```
SELECT *
FROM Customer C RIGHT JOIN Job J
ON C.CustID = J.CustID
```

SQL command being executed

Customer : Table

	CustID	Name
+	1	Brian
+	2	Issa
+	4	Rick
+	5	Kelley
+	6	Mike

Job : Table

	JobID	CustID	Employer
+	13	1	Squirrel Mart
+	14	2	MaliZone
+	15	9	Dilbert Inc
+	16	5	Stonehenge Ltd
+	17	4	OOPs Consulting

Values that do not match join condition (will be excluded)

Matching values exist in both tables

Right Join : Select Query

C.CustID	Name	JobID	J.CustID	Employer
1	Brian	13	1	Squirrel Mart
2	Issa	14	2	MaliZone
4	Rick	17	4	OOPs Consulting
5	Kelley	16	5	Stonehenge Ltd
		15	9	Dilbert Inc

No matching value in other table, match to NULL instead

NULL value; No matching value was found

Full Outer Join

- Conceptually, a **full outer join** combines the effect of applying both left and right outer joins.
- Where records in the FULL OUTER JOINed tables do not match, the result set will have NULL values for every column of the table that lacks a matching row.
- For those records that do match, a single row will be produced in the result set (containing fields populated from both tables).

Customer : Table

	CustID	Name
+	1	Brian
+	2	Issa
+	4	Rick
+	5	Kelley
+	6	Mike

Job : Table

	JobID	CustID	Employer
+	13	1	Squirrel Mart
+	14	2	MaliZone
+	15	9	Dilbert Inc
+	16	5	Stonehenge Ltd
+	17	4	OOPs Consulting

Tables to be joined

Full (outer) Join

```
SELECT *
FROM Customer C FULL JOIN Job J
ON C.CustID = J.CustID
```

SQL command being executed

Customer : Table

	CustID	Name
+	1	Brian
+	2	Issa
+	4	Rick
+	5	Kelley
+	6	Mike

Job : Table

	JobID	CustID	Employer
+	13	1	Squirrel Mart
+	14	2	MaliZone
+	15	9	Dilbert Inc
+	16	5	Stonehenge Ltd
+	17	4	OOPs Consulting

Values that do not match join condition (will be excluded)

Matching values exist in both tables

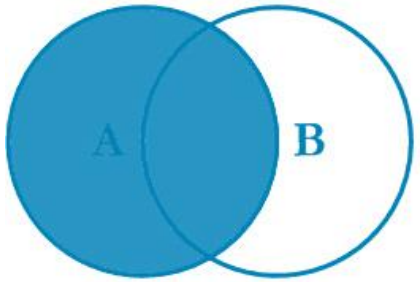
Full Join : Union Query

	C.CustID	Name	JobID	J.CustID	Employer
▶			15	9	Dilbert Inc
	1	Brian	13	1	Squirrel Mart
	2	Issa	14	2	MaliZone
	4	Rick	17	4	OOPs Consulting
	5	Kelley	16	5	Stonehenge Ltd
	6	Mike			

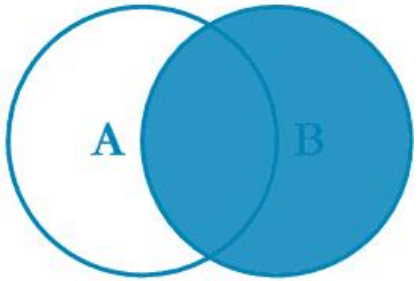
No matching value in other table, match to NULL instead

NULL value; No matching value was found

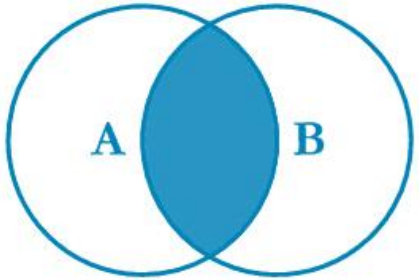
SQL JOINS



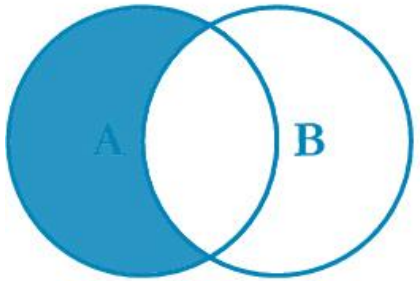

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```



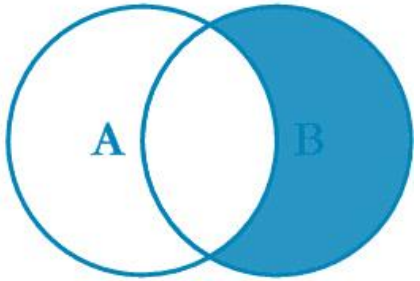

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```



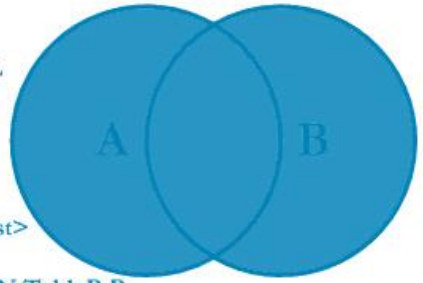

```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```



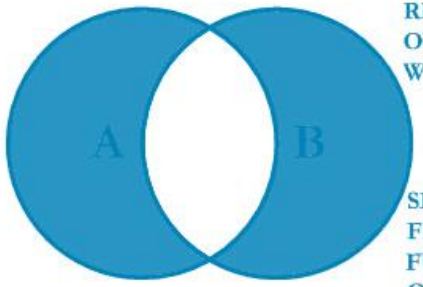

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```




```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```




```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```




```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

Practice 1

Table: Buildings

Building_name	Capacity
1e	24
1w	32
2e	16
2w	20

Table: Employees

Role	Name	Building	Years_employed
Engineer	Becky A.	1e	4
Engineer	Dan B.	1e	2
Engineer	Sharon F.	1e	6
Engineer	Dan M.	1e	4
Engineer	Malcom S.	1e	1
Artist	Tylar S.	2w	2
Artist	Sherman D.	2w	8
Artist	Jakob J.	2w	6
Artist	Lillia A.	2w	7
Artist	Brandon J.	2w	7
Manager	Scott K.	1e	9
Manager	Shirlee M.	1e	3
Manager	Daria O.	2w	6

1. List all buildings and the distinct employee roles in each building (including empty buildings)
2. Find the names of the buildings that hold no employees

Source: https://sqlbolt.com/lesson/select_queries_with_outer_joins

Practice 2

Table: Movies

Id	Title	Director	Year	Length_minutes
1	Toy Story	John Lasseter	1995	81
2	A Bug's Life	John Lasseter	1998	95
3	Toy Story 2	John Lasseter	1999	93
4	Monsters, Inc.	Pete Docter	2001	92
5	Finding Nemo	Andrew Stanton	2003	107
6	The Incredibles	Brad Bird	2004	116
7	Cars	John Lasseter	2006	117
8	Ratatouille	Brad Bird	2007	115
9	WALL-E	Andrew Stanton	2008	104
10	Up	Pete Docter	2009	101
11	Toy Story 3	Lee Unkrich	2010	103
12	Cars 2	John Lasseter	2011	120
13	Brave	Brenda Chapman	2012	102
14	Monsters University	Dan Scanlon	2013	110

Table: Boxoffice

Movie_id	Rating	Domestic_sales	International_sales
5	8.2	380843261	555900000
14	7.4	268492764	475066843
8	8	206445654	417277164
12	6.4	191452396	368400000
3	7.9	245852179	239163000
6	8	261441092	370001000
9	8.5	223808164	297503696
11	8.4	415004880	648167031
1	8.3	191796233	170162503
7	7.2	244082982	217900167
10	8.3	293004164	438338580
4	8.1	289916256	272900000
2	7.2	162798565	200600000
13	7.2	237283207	301700000

Practice 2

1. Find the domestic and international sales for each movie.
2. Show the sales numbers for each movie that did better internationally rather than domestically
3. List all the movies by their ratings in descending order
4. List all movies and their combined sales in millions of dollars
5. List all movies and their ratings in percent
6. Find the total domestic and international sales that can be attributed to each director

Subqueries

A subquery is a SELECT FROM WHERE expression that is nested within another query

Find all the suppliers who

are no customers

customerID	name	street	postcode	city
1	Max Frisch	Bahnhofstrasse 7	8001	Zurich
2	Eddy Merckx	Pleinlaan 25	1050	Brussels
5	Claude Debussy	12 Rue Louise	75008	Paris
53	Albert Einstein	Bergstrasse 18	8037	Zurich
8	Max Frisch	ETH Zentrum	8092	Zurich

```
SELECT DISTINCT name
FROM Supplier
WHERE name NOT IN (SELECT name
                   FROM Customer);
```

supplierID	name	postcode	city
5	Max Frisch	8037	Zurich
2	Mario Botta	6901	Lugano

name
Mario Botta

Nested Subqueries

Find all CDs with a price smaller than average

```
SELECT *
FROM CD
WHERE price < (SELECT AVG(price)
               FROM CD);
```

cdID	name	duration	price	year
2	Carcassonne	3156	15.50	1993
3	Chromatic	3012	16.50	1993

cdID	name	duration	price	year
1	Falling into Place	2007	17.90	2007
2	Carcassonne	3156	15.50	1993
3	Chromatic	3012	16.50	1993

Set Operations

The UNION, INTERSECT and EXCEPT operations correspond relational algebra operations

the relations have to be compatible (same attributes)

these operations remove duplicates by default

```
(SELECT name
FROM Customer)
INTERSECT (SELECT name
                  FROM Supplier);
```

name
Max Frisch

customerID	name	street	postcode	city
1	Max Frisch	Bahnhofstrasse 7	8001	Zurich
2	Eddy Merckx	Pleinlaan 25	1050	Brussels
5	Claude Debussy	12 Rue Louise	75008	Paris
53	Albert Einstein	Bergstrasse 18	8037	Zurich
8	Max Frisch	ETH Zentrum	8092	Zurich

supplierID	name	postcode	city
5	Max Frisch	8037	Zurich
2	Mario Botta	6901	Lugano

Check Existence

The EXISTS operator can be used to check if a tuple exists in a subquery.

```
SELECT name
FROM Customer
WHERE EXISTS (SELECT *
              FROM Supplier
              WHERE Supplier.name = Customer.name);
```

name
Max Frisch