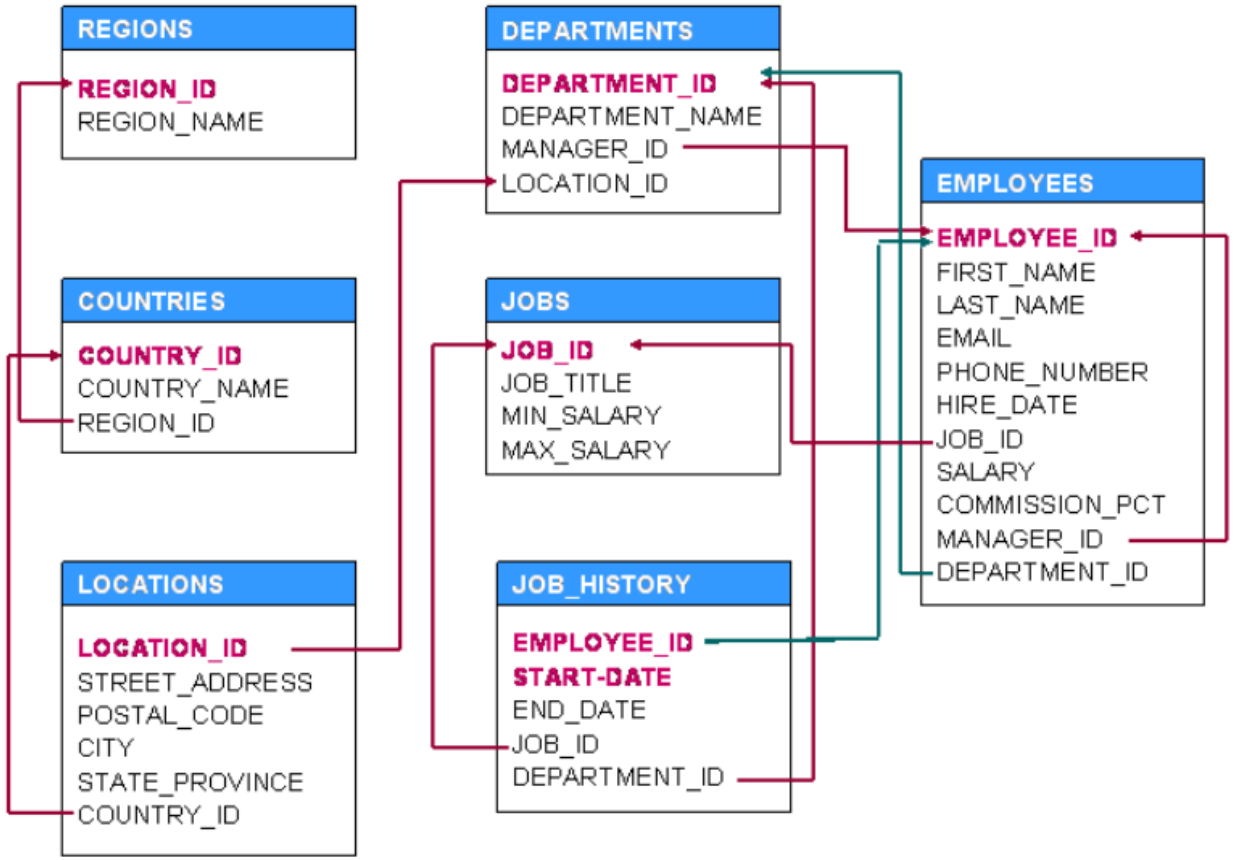
SQL Joins

CS 341 Database Systems Lab

HR Schema


REGIONS
REGION_ID
REGION_NAME
DEPARTMENTS
DEPARTMENT_ID
DEPARTMENT_NAME
MANAGER_ID
LOCATION_ID
EMPLOYEES
EMPLOYEE_ID
FIRST_NAME
LAST_NAME
EMAIL
PHONE_NUMBER
HIRE_DATE
JOB_ID
SALARY
COMMISSION_PCT
MANAGER_ID
DEPARTMENT_ID
COUNTRIES
COUNTRY_ID
COUNTRY_NAME
REGION_ID
JOBS
JOB_ID
JOB_TITLE
MIN_SALARY
MAX_SALARY
LOCATIONS
LOCATION_ID
STREET_ADDRESS
POSTAL_CODE
CITY
STATE_PROVINCE
COUNTRY_ID
JOB_HISTORY
EMPLOYEE_ID
START-DATE
END_DATE
JOB_ID
DEPARTMENT_ID

Cartesian Product

```
SELECT Customers.CustomerName,
Orders.OrderID
FROM Customers, Orders;
```

JOINS

```
SELECT column_name(s)
FROM table1, table2
WHERE table1.column_name = table2.column_name;
```

```
SELECT column_name(s)
FROM table1
INNER JOIN table2
ON table1.column_name = table2.column_name;
```

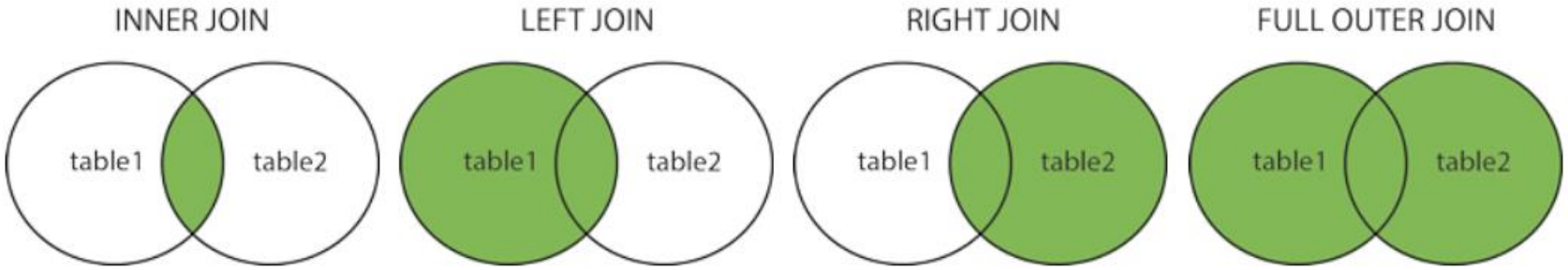
```
SELECT column_name(s)
FROM table1
INNER JOIN table2
USING column_name;
```

```
SELECT column_name(s)
FROM table1 T1
INNER JOIN table2 T2
ON T1.column_name = T2.column_name;
```

Using table aliases. In Oracle, you can simply write the alias after the table name without using AS. For column aliases, we can do the same or use AS.

ON vs USING

- Generally, `ON` is used but `USING` is handy in some situations.
- `USING` – Columns having the exact same name – NATURAL JOIN
  - Note: We also have `NATURAL JOIN` which automatically equates all matching columns with same names.
- `ON` –Specifies the condition for join
  - Any condition – THETA JOINs
  - If condition checks for equivalence – EQUI JOIN

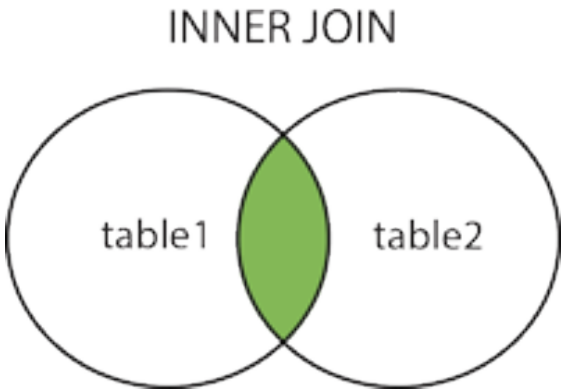


- (INNER) JOIN: Returns records that have matching values in both tables
- LEFT (OUTER) JOIN: Returns all records from the left table, and the matched records from the right table
- RIGHT (OUTER) JOIN: Returns all records from the right table, and the matched records from the left table
- FULL (OUTER)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

INNER JOIN

```
SELECT column_name(s)
FROM table1
INNER JOIN table2
ON table1.column_name = table2.column_name;
```

```
SELECT Orders.OrderID, Customers.CustomerName
FROM Orders
INNER JOIN Customers ON Orders.CustomerID =
Customers.CustomerID;
```



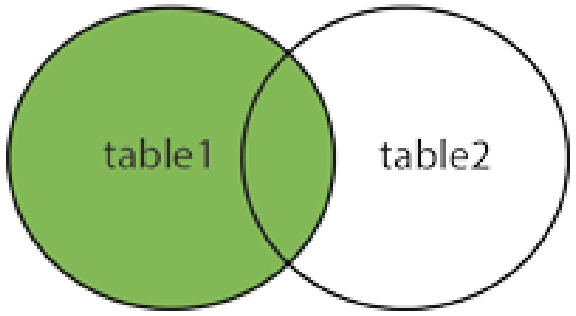
Multiple Tables

```
SELECT Orders.OrderID, Customers.CustomerName, Shippers.ShipperName
FROM ((Orders
INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID)
INNER JOIN Shippers ON Orders.ShipperID = Shippers.ShipperID);
```

Using brackets is a good practice.

LEFT JOIN or RIGHT JOIN

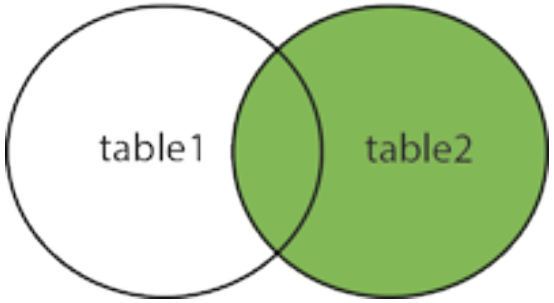
LEFT JOIN




```
SELECT column_name(s)
FROM table1
LEFT JOIN table2
ON table1.column_name = table2.column_name;
```

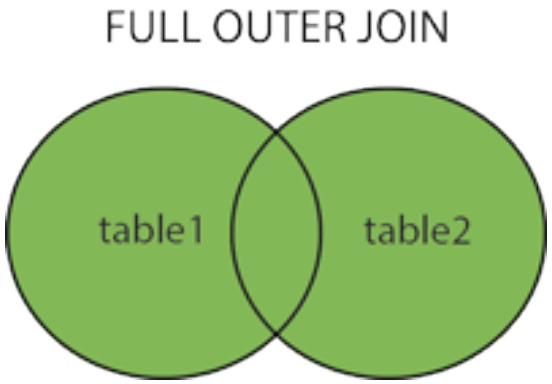
```
SELECT column_name(s)
FROM table1
RIGHT JOIN table2
ON table1.column_name = table2.column_name;
```

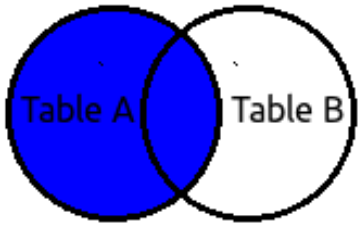
RIGHT JOIN



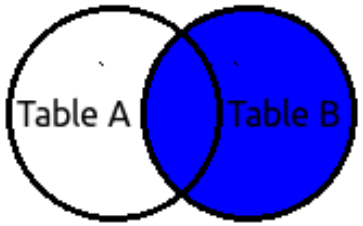
FULL OUTER JOIN

```
SELECT column_name(s)
FROM table1
FULL OUTER JOIN table2
ON table1.column_name = table2.column_name
WHERE condition;
```

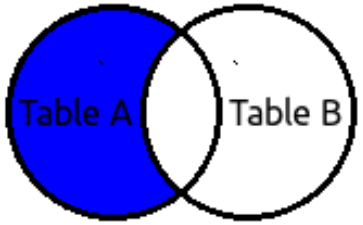




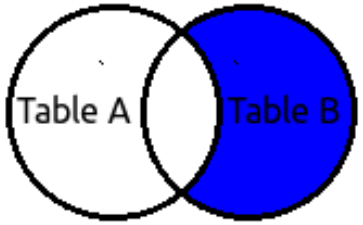

```
SELECT [list] FROM
    [Table A] A
LEFT JOIN
    [Table B] B
ON A.Value = B.Value
```



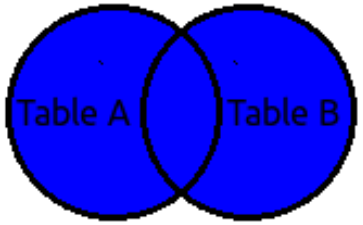

```
SELECT [list] FROM
    [Table A] A
RIGHT JOIN
    [Table B] B
ON A.Value = B.Value
```



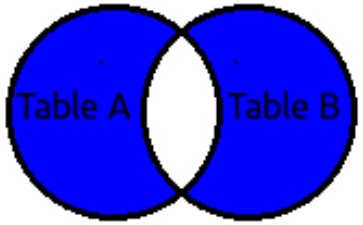

```
SELECT [list] FROM
    [Table A] A
LEFT JOIN
    [Table B] B
ON A.Value = B.Value
WHERE B.Value IS NULL
```



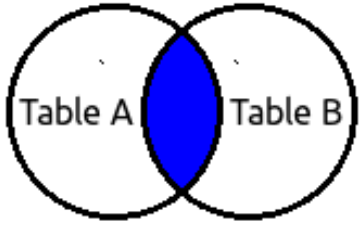

```
SELECT [list] FROM
    [Table A] A
RIGHT JOIN
    [Table B] B
ON A.Value = B.Value
WHERE A.Value IS NULL
```




```
SELECT [list] FROM
    [Table A] A
FULL OUTER JOIN
    [Table B] B
ON A.Value = B.Value
```




```
SELECT [list] FROM
    [Table A] A
FULL OUTER JOIN
    [Table B] B
ON A.Value = B.Value
WHERE A.Value IS NULL
OR B.Value IS NULL
```




```
SELECT [list] FROM
    [Table A] A
INNER JOIN
    [Table B] B
ON A.Value = B.Value
```

CROSS JOIN = Cartesian Product

- Note: The CROSS JOIN keyword returns all records from both tables whether the other table matches or not.

```
SELECT Customers.CustomerName,
Orders.OrderID
FROM Customers
CROSS JOIN Orders;
```

```
SELECT Customers.CustomerName,
Orders.OrderID
FROM Customers, Orders;
```