LAB/ PRACTICAL

LAB/ PRACTICAL

UNION (U)

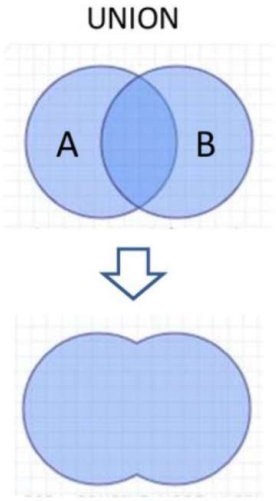
R1 U R2

UNION is symbolized by ∪ symbol. It includes all tuples that are in tables A OR in B. It also eliminates duplicate tuples. So, set A UNION set B would be expressed as:

The result <- A ∪ B

For a union operation to be valid, the following conditions must hold -

R and S must be the same number of attributes.
Attribute domains need to be compatible.
Duplicate tuples should be automatically removed.



In this example, if the R and S table have A and B tuples respectively, the union operation will obtain and concatenating them into one relation with a maximum of tuples.

Query:

List all Branches where there is either have a staff worked or no.

RA

π BranchNo (BRANCH) ∪ π BranchNo (STAFF)

Table: BRANCH **Table: STAFF**

BranchNo
B006
B005
B007
B003
B004
B002
B008
B009

∪

BranchNo
B005
B003
B003
B007
B003
B005

SQL

```
SELECT BranchNo
FROM BRANCH
UNION
SELECT BranchNo
FROM STAFF
```

RESULT

BRANCH.BranchNo U STAFF.BranchNo
B002
B003
B004
B005
B006
B007
B008
B009

Query:

List all cities where there is either a branch office or property for rent

RA

π City (BRANCH) ∪ π City (PROPERTYFORRENT)

Table: BRANCH

CITY
Glasgow
London
Aberdeen
Glasgow
Bristol
London
Bradford
Glasgow

∪

Table: PROPERTYFORRENT

CITY
Aberdeen
London
Glasgow
Glasgow
Glasgow
Glasgow

SQL

```
SELECT City
FROM BRANCH
UNION
SELECT City
FROM PROPERTYFORRENT
```

RESULT

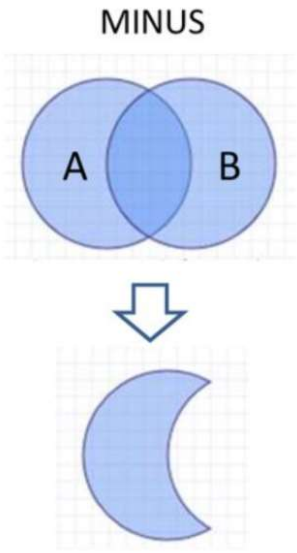
CITY
Aberdeen
Bradford
Bristol
Glasgow
London

SET DIFFERENCE (-) --> minus

R1 - R2

This operation creates a relation containing tuples in the first relation that are not in the second relation. Just like the previous one, both relations must be union-compatible

The result <- A - B



The result of A - B means a relation consisting of the tuples that are in Relation A but not in B. A and B must union-compatible.

Query:

List all cities where there is a branch office but no properties for rent

RA

π City (BRANCH) - π City (PROPERTYFORRENT)

Table: BRANCH

CITY
Glasgow
London
Aberdeen
Glasgow
Bristol
London
Bradford
Glasgow

MINUS

Table: PROPERTYFORRENT

CITY
Aberdeen
London
Glasgow
Glasgow
Glasgow
Glasgow

SQL

```
SELECT City
FROM BRANCH
MINUS
SELECT City
FROM PROPERTYFORRENT
```

RESULT

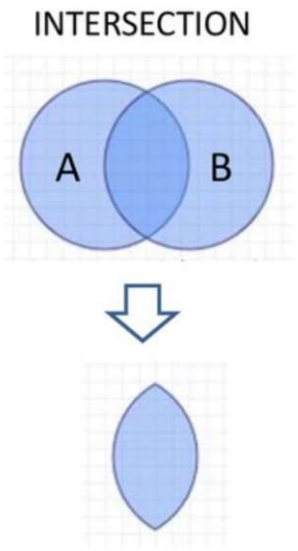
CITY
Bradford
Bristol

INTERSECTION (∩)

R ∩ S

This operation creates a relation containing tuples in both first and second relations. Both relations must be union-compatible.

R ∩ S = R - (R - S)



Defines a relation consisting of a set of all tuple that is in both A and B. However, A and B must be union-compatible.

Query:

List all cities where there is both a branch office and at least one property for rent

RA

π City (BRANCH) ∩ π City (PROPERTYFORRENT)

Table: BRANCH

CITY
Glasgow
London
Aberdeen
Glasgow
Bristol
London
Bradford
Glasgow

Table: PROPERTYFORRENT

∩

CITY
Aberdeen
London
Glasgow
Glasgow
Glasgow
Glasgow

SQL

```
SELECT City
FROM BRANCH
INTERSECT
SELECT City
FROM PROPERTYFORRENT
```

RESULT

CITY
Aberdeen
Glasgow
London

DIVISION (/)

R / S --> R ÷ S

This operation creates a relation containing selected attributes in the first relation, matching with every tuple in the second relation.

It defines a relation over the attributes C that consists of a set of tuples from R that match a combination of every tuple in S.

Query:

Identify all clients who have viewed all properties with three rooms

RA

(π ClientNo, PropertyNo (CLIENT Natural Join Symbol JPG Image client.clientno = viewing.cliento (VIEWING))) ÷

(π PropertyNo (VIEWING Natural Join Symbol JPG Image viewing.propertyno = propertyforrent.propertyno ^ viewing.rooms = 3 (PROPERTYFORRENT)))

RESULT 1<--π ClientNo, PropertyNo (CLIENT ⋈ client.clientno = viewing.cliento (VIEWING))

CLIENTNO	PROPERTYNO
CR56	PA14
CR76	PG4
CR56	PG4
CR62	PA14
CR56	PG36

RESULT 2 <--π PropertyNo (VIEWING ⋈ viewing.propertyno = propertyforrent.propertyno ^ viewing.rooms = 3 (PROPERTYFORRENT))

PROPERTYNO
PG4
PG36

RESULT 1 ÷ RESULT 2 --> RESULT1 / RESULT 2

SQL : RESULT 1

```
SELECT Client.ClientNo, Viewing.PropertyNo
FROM VIEWING
JOIN CLIENT
ON (Client.CLientNo = Viewing.ClientNo)
```

SQL : RESULT 2

```
SELECT DISTINCT Viewing.PropertyNo
FROM VIEWING
JOIN PROPERTYFORRENT
ON (Viewing.PropertyNo = PropertyForRent.PropertyNo)
WHERE PropertyForRent.Rooms = 3
```

RESULT

CLIENTNO CR56 has viewed all the properties that have 3 rooms.

CLIENTNO
CR56