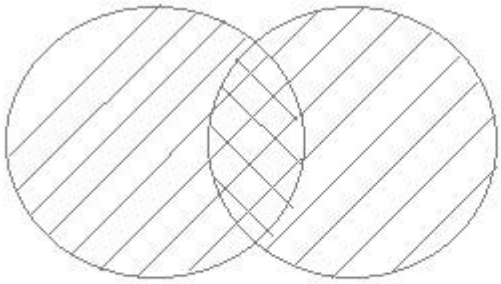
Structure Query Language (SQL)

Union All

This operation is similar to Union. But it also shows the duplicate rows.



Example of Union All

The **First** table,

ID	NAME
1	Abhi
2	Adam

The **Second** table,

ID	NAME
2	adam
3	Chester

Union All query will be like,

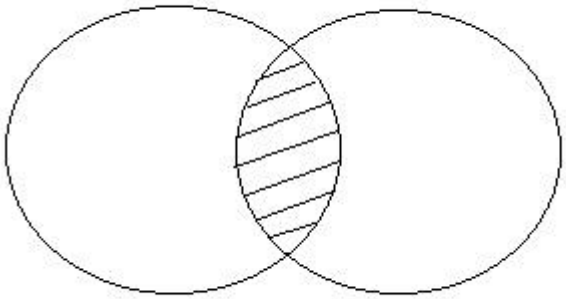
```
select * from First
UNION ALL
select * from second
```

The result table will look like,

ID	NAME
1	abhi
2	adam
2	adam
3	Chester

6.17. Intersect

Intersect operation is used to combine two SELECT statements, but it only returns the records which are common from both SELECT statements. In case of **Intersect** the number of columns and data type must be same. MySQL does not support INTERSECT operator.



Example of Intersect

The **First** table,

ID	NAME
1	Abhi
2	adam

The **Second** table,

ID	NAME
2	adam
3	Chester

Intersect query will be,

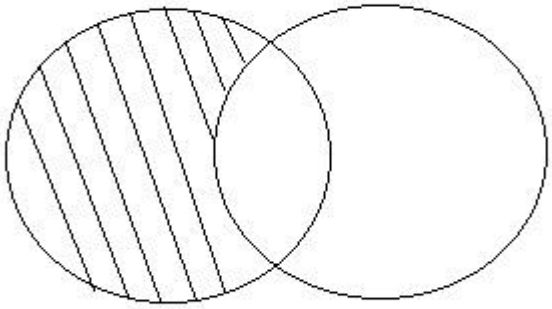
```
select * from First
INTERSECT
select * from second
```

The result table will look like

ID	NAME
2	adam

6.17.4 Minus

Minus operation combines result of two Select statements and return only those result which belongs to first set of result. MySQL does not support INTERSECT operator.



Example of Minus

The **First** table,

ID	NAME
1	Abhi
2	Adam

The **Second** table,

ID	NAME
2	adam
3	Chester

Minus query will be,

```
select * from First
MINUS
select * from second
```

The result table will look like,

ID	NAME
1	Abhi

6.18 SQL Sequence

Sequence is a feature supported by some database systems to produce unique values on demand. Some DBMS like **MySQL** supports AUTO_INCREMENT in place of Sequence. AUTO_INCREMENT is applied on columns, it automatically increments the column value by 1 each time a new record is entered into the table. Sequence is also somewhat similar to AUTO_INCREMENT but its has some extra features.

Creating Sequence

Syntax to create sequences is,

```
CREATE Sequence sequence-name
start with initial-value
increment by increment-value
maxvalue maximum-value
cycle|nocycle
```

initial-value specifies the starting value of the Sequence, **increment-value** is the value by which sequence will be incremented and **maxvalue** specifies the maximum value until which sequence will increment itself.**cycle** specifies that if the maximum value exceeds the set limit, sequence will restart its cycle from the begining. **No cycle** specifies that if sequence exceeds **maxvalue** an error will be thrown.

Example to create Sequence

The sequence query is following

```
CREATE Sequence seq_1
start with 1
increment by 1
maxvalue 999
cycle ;
```

Example to use Sequence

The **class** table,

ID	NAME
1	abhi
2	adam
4	alex

The sql query will be,

```
INSERT into class value(seq_1.nextval,'anu');
```

Result table will look like,

ID	NAME
1	abhi
2	adam
4	Alex
1	Anu

Once you use `nextval` the sequence will increment even if you don't Insert any record into the table.

6.19 SQL View

A view in SQL is a logical subset of data from one or more tables. View is used to restrict data access.

Syntax for creating a View,

```
CREATE or REPLACE view view_name AS
SELECT column_name(s)
FROM table_name
WHERE condition
```

Example of Creating a View

Consider following **Sale** table,

Oid	order_name	previous_balance	Customer
11	ord1	2000	Alex
12	ord2	1000	Adam
13	ord3	2000	Abhi
14	ord4	1000	Adam
15	ord5	2000	Alex

SQL Query to Create View

```
CREATE or REPLACE view sale_view as select * from Sale where custome
r = 'Alex';
```

The data fetched from select statement will be stored in another object called **sale_view**. We can use create seperately and replace too but using both together works better.

Example of Displaying a View

Syntax of displaying a view is similar to fetching data from table using Select statement.

```
SELECT * from sale_view;
```

6.19.1 Force View Creation

force keyword is used while creating a view. This keyword force to create View even if the table does not exist. After creating a force View if we create the base table and enter values in it, the view will be automatically updated.

Syntax for forced View is,

```
CREATE or REPLACE force view view_name AS
SELECT column_name(s)
FROM table_name
WHERE condition
```

6.19.2 Update a View

Update command for view is same as for tables.

Syntax to Update a View is,

```
UPDATE view-name
set value
WHERE condition;
```

If we update a view it also updates base table data automatically.

6.19.3 Read-Only View

We can create a view with read-only option to restrict access to the view.

Syntax to create a view with Read-Only Access

```
CREATE or REPLACE force view view_name AS
SELECT column_name(s)
FROM table_name
WHERE condition with read-only
```

The above syntax will create view for read-only purpose, we cannot Update or Insert data into read-only view. It will throw an error.

Types of View

There are two types of view,

- Simple View
- Complex View

Simple View	Complex View
Created from one table	Created from one or more table
Does not contain functions	Contain functions
Does not contain groups of data	Contains groups of data