2.5quintillion data is been gernerated everyfday which helps us in decision making in business operation so to manage this data becomes important ther fore there is

a need to have a database to mamange this data

rdbms

data is stored in tables ,

tables which are linked to each other using relationship

therfore called as relational data base

each table stores data about specific type of object

example products , customers & orders

all this tables(products , customers & orders) are linked to each other some relationship here it is related to online shopping

sql is the language we use to work with his RDBMS

RDMS in market->mysql,sqlserver,oracle

functions are procedures stored inside the database for quering data

eg you can have store procedure to give about customer in particular city

so we call that procedure and say give me all customer in san francisco

every row is a record and every row represent a record

suppose customer id is one of a coloum and id 7 has johns details

this customer details is used for orders

now imagine john changes his phones number or address anything etc will have to change those everywhere necessary

now if we only use customer id in order table so whatever changes are made in johns detailes only in customer table chnages will be made

rest since customer id is unique of johns the changes aref urther carried out without actually chnaging the order table since the changes to particular customer id is made in customer table

thus order table is related to customer table via customer id therfore relational database

select \*

from (name of the table)

\*-> all

select \*

from x

WHERE custoner\_id=1;

WHERE USED TO FITER THE DATA

ed WHERE points>3000 etc

comparision operators :>,>=,<,<=,=,!=(not equal),<>(not equal)

select \*

from x

order by (on what basiss u have to sort)

ORDER BY used to sort the data

select \*

from x

firstname AS name

so now table will say name and now firstname is modified to name AS is used for that

if u have two letter

firstname as my name use "my name" 'my name'

DISTINCT keyword for so that we dont have repeated data

suppose we want states where the customers are from

eg mumbai

but now two customers are from mumbia we dont want mumbai mumbai twice

select \*

from x

where (condition 1) and (condition 2)

where (condition 1) OR (condition 2)

where (condition 1) or (condition 2) and (condition 3) 2 & 3 will get evaluated first since and has more importance than or

where NOT(condition) negate the condition

NOTE once u start the sql server u have select paticular database for query by use--> USE (database name)

before writing query to get data from a database u have to select that database

in opertaor - when u wnat people staying in 3 cities u want them all

query is

select \*

from customer

where city = 'a' or city = 'b' or city = 'c'

u cant write where city = 'a', 'b, 'c '

beacuse city = 'a' this is a boolean statement right to the OR is 'b' which is a string

or combines conditions it does not combine boolean expression and a string

we can use in operator where city IN ('a','b','c')

opposite where city NOT IN ('a','b','c')

use in operator whne u want to compare attribute with a list of values

between operator select \*

from customers

where points BETWEEN 1000 and 3000

(between operator for range of value comparison

-- where points>=1000 and points<=3000

like operator use to see patterns

select \*

from customers

where last\_name LIKE 'b%' (customers wtih surname staring with b and % indicates how much characters after b therfore b%

version %b(end letter o=should be b) %b%(we just want to know if the surname contains any b before after middle doesnt matter so % b %)

\_ ->underscore operator \_y meaning 2 letter word end has to be y

\_\_\_\_\_y 5 letter word end has to be y

oppo NOT LIKE

% to represent n no of charaters

- to represent one character

LIKE only workds with ''

so even if its like a phone number query should be

where phone like '%9' phone number ending with 9

-- where last\_name like '%field%'

where last\_name REGEXP 'field' (with regexp we dont have to type % helps us find more complex operators )

versions ->where last\_name REGEXP '^field' last name should start with field

where last\_name REGEXP 'field$' last name should end with field

^ indicate the starting of the string

$ ending of the string

where last\_name regexp 'field|mac' last name having field or mac in it

where last\_name regexp '[gim]e' last name would have ge or im or me [] to match any single character listedin the bracket optional characters e is mandatory

where last\_name regexp 'e[xyz]' ex or ey or ez

where last\_name regexp '[a-h]e' a to h followed by e - to represetn the

summary REGEXP

^ start of the string whatever

$ end

| logical or

[abcd]

[a-f]

query

select \*

from customers

-- where first\_name regexp '^elka|^ambur'

-- where last\_name regexp 'ey$|on$'

where last\_name regexp '^my|se'

-- where last\_name regexp 'b[ru]'

null operator

select \*

from customer

where phone is null

joins

inner join

left join

right join

full join

natural join is use when you want to reduce number of columns, used when u want no of columns returned are less

natural join used to give output based on columns in both the tables on which the joint operation is implemented

gives you output based on the columns on which it is applied

we dont use on clause in natural join

NOTE :JOINS are used to map one to many relationship

therefore for many to many relationship

we use two join statements

suppose there are employeer every employeer works on oe project

every proect consistes of m different technologies

now u have three tables

project

technologies

project\_technologies this contains the n project mapping to m technologies

now u have to join project to project\_technology and technology to project\_technology

hash function are also join use to join large tables

eg where user wants most of the joined table rows

when u want two large tables u use hash join

two parts

build phase- you create in memory hash index on the left side of the input

probe phase- u go to the right side of the input each row at a time and find the match using the index created in the build phase

build phase probe phase

left side | right side

create index in memory hash go each row at a time find a match using indexes build in build phase

self join join of a table with itself,each row in a

table joins to itself

cross join is when join clause is applied to each row of a table to every row of another table

wwwhen used with where clause it is similar to inner join

without where clause i is like a cartesian product

join operstion on 3 tables

2 join operation

order by clause used to sort data based on numerical order 1 2 3 or alpha method a b c

ascending order to get descending order

select \*

from customer

order by x DESC(use this keyword)

we can take many columns into consideration

order by x , y, z

order by x desc , y desc , z many versions

limit clause

if u write a query

select \*

from customer

u will get all the elements what if u want to limit the customers to 3 so use limit clause

select \*

from customer

limit 3

versions

limit 6 , 3 --> this query tells mysql to skip first 6 records and then pick records

limit clause always comes at the end so suppose

select \*

from customers

where ....

order by ...

LIMIT <-- always at the last

note : we have not stored customer information(phone no , email ,address)

in order table bcoz this information will change in the future

and if a guven customer has placed multiple orders then

will have to come back and chnage multiple records

we dont wanna do that thats whyy we have separate tables for customers and orders respectively

but what if we want to display the result with customers details and their orders details

thats where we use join keyword

eg select \*

from orders

JOIN customers ON order.customer\_id , customers.customer\_id

(u need to have common column based on whicb u will be able to join the tables )

now incase the query is

select order\_id , first\_name , last\_name , shipped\_date , customer\_id

from orders

join customers

on orders.customer\_id= customers.customer\_id

this query will get u a error that the column customer\_id in field list is ambiguous

bcoz sql doesnt know which customer\_id u referring to since both contains this coloumn

when written like this error is solved

select order\_id , first\_name , last\_name , shipped\_date , oreders.customer\_id

or customers.customer\_id

from orders

join customers

on orders.customer\_id= customers.customer\_id

to make the code less complex as we repeating customers orders too much we can do this

select first\_name , last\_name , shipped\_date , o.customer\_id

from orders o

join customers c

on o.customer\_id= c.customer\_id

self join

select e.employee\_id , e.first\_name , m.first\_name as manager

from employees e

join employees m

on e.reports\_to = m.employee\_id

joining multiple tables

select o.order\_id, o.order\_date,c.first\_name, c.last\_name , os.name

from orders o

join customers c

on o.customer\_id=c.customer\_id

join order\_statuses os

on o.status=os.order\_status\_id

select p.payment\_id ,c.name,p.invoice\_id, p.date , p.amount , pm.name

from payments p

join payment\_methods pm

on p.payment\_method = pm.payment\_method\_id

join clients c

on c.client\_id = p.client\_id

compound join is when there are two primary keys

select \*

from order\_items oi

join order\_item\_notes oin

on oi.order\_id=oin.order\_id

and oi.product\_id = oin.product\_id

implicit join syntax

select \*

from orders o , customers c

where o.order\_id = c.customer\_id

but its suggested not to use impicite join bcoz

when u forget to write the where clause it becomes cross join

were every object of order is joined to every object of customer \

so if order hs 10 items and so is customers

then result will contain 100 items

so we always use explicit syntax which is

select \*

from x

join y

ON ...

in sql we have two join

inner join

outer join

so far we only seen inner join by default if u write join it considers it as inner join

OUTER JOIN

select c.customer\_id,c.first\_name , o.order\_id

from customers c

join orders o

on c.customer\_id=o.customer\_id

order by c.customer\_id

THIS QUEERY IS AN INNER JOIN WHERE THE CUSTOMER TABLE

CUSTOMER WHO HAS ORDERS ARE DISPLAYED

BUT THINK IF THERE ARE CUSTOMERS WHO DONT HAVE ORDERS WE WANT EVEN THEM TO BE DISPLAYED

THAS WHEN WE USE OUTER JOIN

IN MYSQL WE HAVE TWO OUTER JOINS

LEFT JOIN

RIGHT JOIN

select c.customer\_id,c.first\_name , o.order\_id

from customers c

LEFT join orders o

on c.customer\_id=o.customer\_id

order by c.customer\_id

all customers from the left table that is customers table are displayed and orders table is joined based on

which customer has placed the order person whose not placed the order the order id is null

r

outer join beween multiple tables

select c.customer\_id,c.first\_name , o.order\_id,sh.name

from customers c

left join orders o -- left outer join we get all the customers irrespective of they have an order or not

on c.customer\_id=o.customer\_id

left join shippers sh-- we get all the orders irrespective of if they have a shipper or not

on o.shipper\_id = sh.shipper\_id; -- inner join

order by c.order\_id;

we get all the customers irrespective of they have an order or not

values are null at customers whose not placed nay order

we get all the orders irrespective of they have a shipper or not

values are null at places where orders dont have a shipper

query

select o.order\_date , o.order\_id ,c.first\_name,sh.name ,os.name

from orders o

left join order\_statuses os

on os.order\_status\_id=o.status

left join customers c

on o.customer\_id = c.customer\_id

left join shippers sh

on sh.shipper\_id=o.shipper\_id

order by sh.name

NOTE

on e.reports\_to=m.employee\_id --

on e.employee\_id = m.reports\_to

this both will give you diff results

self outer join

select e.employee\_id , e.first\_name, m.first\_name as manager

from employees e

left join employees m

on e.reports\_to=m.employee\_id

USING CLAUSE

as queries get complex the join statments come in the way they make it hard to read

thats where we use USING clause

whenever the join condition(column name) that is the on condition is same

eg m.employee\_id= e. employee\_id

select o.order\_id , c.first\_name ,sh.name as shipper

from orders o

join customers c

using (customer\_id)

left join shippers sh

using(shipper\_id)

USING can only be used when column names are same

for composite keys join

select \*

from order\_items oi

left join order\_item\_notes oin

using (product\_id, order\_id)

select p.date , c.name , p.amount ,pm.name

from payments p

join payment\_methods pm

on p.payment\_method= pm.payment\_method\_id

join clients c

using (client\_id)

NATURAL JOIN

easier way to join the tabels

but it sometimes gives unexpected result

select o.order\_id , c.first\_name

from orders o

natural join customers c

NOTE : HERE WE DONT SPECIFY THE ON CONDITION THE SQL

JOINS THE TABLE BASED ON THE COMMON COLUMNS

THUS WE DONT HAVE CONTROL OVER IT WE DONT USE MUCH

CROSS JOIN

we use cross join to join every record from first table toevery record of second table

select \*

from

customers c

cross join product p (explicit syntax)

EVERY RECORD ON THE CUSTOMERS TABLE WILL GET JOINED

TO EVERY RECORD ON THE PRODUCTS TABLE

SO WE DONT HAVE A ON CONDITION

select \*

from customers c , orders o (implicit syntax)

suppose u got diff colours of tshirt and u have there

sizes s m l

thats where u use cross join

size s -> red yellow blue etc

UNIONS

select order\_id, order\_date ,'Active' as status

from orders

where order\_date >="2019-01-01"

UNION

select order\_id, order\_date ,'Archive' as status

from orders

where order\_date <"2019-01-01"

WE CAN DO THIS FOR DIFFERENT TABLES AS WELL

select customer\_id , first\_name , points ,'bronze' as type

from customers

where points<2000

union

select customer\_id , first\_name , points ,'silver' as type

from customers

where points between 2000 and 3000

union

select customer\_id , first\_name , points ,'gold' as type

from customers

where points>3000

order by first\_name

now here if u use alias and write sorting things that is

order by ... it will throw an error

COLUMN ATTRIBUTES

INSERT INTO CUSTOMERS(first\_name , last\_name ,birth\_date , adress, city , state)

values(,'aparna','mane','1998-06-12','beverly hill','city','california');

INSERT INTO CUSTOMERS()

values(default,'aparna','mane','1998-06-12',null,'beverly hill','city','california'default);

if u want to insert data in multiple rows

insert into shippers(name)

values('shipper1'),

('shipper2'),

('shipper3')

insert into products(name,quantity\_in\_stock,unit\_price)

values('x',45, 9.24),

('y',48, 3.27),

('z',90, 8.21)

INSERTING HIERARCHICAL ROWS

ORDER TABLE doesnt have actual items

ORDER ITEM TABLE

order\_id so what order this item is for

product\_id so we know what product has been ordered

quantity @ what quantity

unit\_price @ wat price

so actual order can have one or more order items

this is what we called parent child relationship

order table - parent

order\_items table- child

so one row in order table canhave one

or more childrens in order\_items table

insert into order\_items

insert into orders(customer\_id, order\_date, status)

values (1,'2019-01-01',1);

insert into order\_items

values(last\_insert\_id(),1,1,2.95),

(last\_insert\_id(),2,1,3.95)

this is how to insert the hierarchical data

where first the order was created

we got that new order ka order\_id to insert into that order

using last\_insert\_id () function and then insert

eg we place an order with order id 7 say

this has 7 as order id

my id i.e. customer id

the date of order placement

status shipped or processed etc

in that order i have placed

2shoes

1chappals

4shirts

2shors

5skirts

so in order\_items i have

order\_id 7

product id

quantity suppose shoes 2 chappal 1 ]

unit price

CREATING A COPY OF TABLE

SO M CREAITNG A COPY OF TABLE ORDER I.E.ORDER ARCHIVE

IN ORDER TABLE I HAVE DOZEN RECORDS

NOW I DONT WANT TO WRITE DOZEN STATEMENTS TO COPY ALL THIS DATA

INTO THAT TABLE

SO SHORTCUT to quickly copy data

create table orders\_archived as

select \* from orders

but this new table wont have a primary key sql

will ignore this attributes

refer to this as subquery

select \* from orders

subquery is a select statement that is a

part of another sql statement

suppose

select \*

from orders

where order\_date <'2019-01-01'

i use this in insert statement then it is used a subquery

using select statement as a subquery in ann insert statement

create a invoices\_archived table

but instead of client\_id we want client name

copy invoices that do have a payment

create table invoices\_archived as

select i.invoice\_id , i.number ,c.name as client , i.invoice\_date ,payment\_date , due\_date

from invoices i

join clients c

using(client\_id)

where payment\_date is not NULL

HOW TO UPDATE DATA

UPDATE INVOICES(TABLE\_NAME)

SET (WHATEVER NEW PARAMETER U WANT TO)

WHERE (ID MAINLY WHICH PRIMARY KEY OF THAT TABLE IS TO BE UPDATED)

UPDATE INVOICES

SET PAYMENT\_TOTAL =invoice\_total \*0.5, PAYMENT\_DATE =due\_date

WHERE INVOICE\_ID =3

update mutiple rows

update customers

set points= points+50

where birth\_date < '1990-01-01'

UPDATE INVOICES

SET PAYMENT\_TOTAL =invoice\_total \*0.5, PAYMENT\_DATE =due\_date

WHERE client\_ID =3

use sql\_store

UPDATE SUBQUERIES IN UPDATES

UPDATE INVOICES

SET PAYMENT\_TOTAL =invoice\_total \*0.5, PAYMENT\_DATE =due\_date

WHERE client\_ID =3

use sql\_store

in this query suppose we dont have client\_id

we have name of the client

so first we need to get the if then update

update invoices

SET PAYMENT\_TOTAL =invoice\_total \*0.5, PAYMENT\_DATE =due\_date

WHERE client\_ID=

(select client\_id

from clients

where name ='myworks')

for multiple name

SET PAYMENT\_TOTAL =invoice\_total \*0.5, PAYMENT\_DATE =due\_date

WHERE client\_ID IN

(select client\_id

from clients

where state in('ca','ny')

update orders

set comments='gold customer'

where customer\_id in (select customer\_id

from customers

where points>3000)

deleting rows

delete from invoices

where invoice\_id=1

delete from invoices

where client\_id=(select \*

from clients

where name='myworks')

restoring data

go to files

open sql script

run it

refresh the panel