Database ---

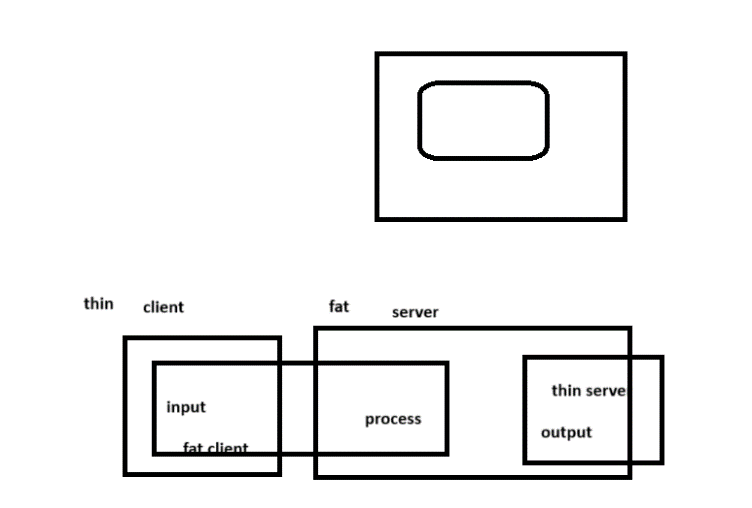
Normal – Traditional

DBMS - Foxpro, Dbase, Clipper –

EF Codd - Relational Theory --- 12 Rules

Relational Database Management Systems - is a software follows minimum two tier architecture and follows Codd rules.

SqlServer --- Microsoft



System databases –

Master,model,msdb,tempdb

Create database <dbname>

Table / relation( two dimensional array)

Create table <tablename>(<column1> <datatype>,…..)

* String --- char,varchar,text,nchar,nvarchar,ntext
* Date – date,datetime,smalldatetime,longdatetime ,timestamp
* Numeric – int,smallint,tinyint,bigint,decimal,float,real etc
* Binary - image,binary,varbinary

Alter database <databasename> add file(

Name=’name’,

Filename=’physicalfilename’,

Size=10,

Filegrowth=1,

Maxsize=100

alter database hitha modify file(name='hitha',size=10)

alter database hitha add file(

name='hitha1',

filename='d:\\emphasiz\hitha1.mdf',

size=10,

maxsize=100,

filegrowth=1)

)

Alter database hitha remove file hitha1

create database hitha

on(//datafile details)

log on(//logfile details)

items

itemcode itemname price qty

1 bingo 12 100

2 lays 12 34

Orders

Orderid customername customeraddress

1 shivani UP

2 varun RR Nagar

Orderitems

Order id itemcode qty price

1 1 1 12

1 2 2 12

1 5 4 45­­

2 2 3 10

2 5 4 12

Select \* from <tablename>

Select [ <column>/expression,<column>/expression]/\* from <table>

Select name, from stock

Select 10 + 20

JOIN - retrieve data from two or more tables as it is coming from a single table

=============

Equi

Outer

Cross join

Cross join syntax

Select table1.column1,table2.column1,…/\* from table1,table 2,.. --- cartesian product

Equi join syntax:

Select table1.column1,table2.column1,…/\* from table1,table 2,.. where/on <joincondition>

And to join n tables there should be minimum n-1 join conditions required

Data integrity - ensuring correctness of data

For these use constraints – rules imposed on column

1.check -- ensure that values falls in to a range ,group or pattern -- domain integrity

2.unique -- ensure that only unique values are inserted into column – entity integrity

3.primary key -- ensure only unique and not null values are inserted in to the column – entity integrity

4.foreign key --- ensure that value inserted is present in another table column-referencial integrity

Constraints can be applied in table level and column level

Constraints can be applied during table creation or after creation

Syntax:

[ Constraint <constraint name> ] <constraint type> [specification]

select \* from orders

alter table orders add constraint uniqueorder unique(orderid)

alter table orders alter column orderid int not null

alter table orders add primary key(orderid)

select \* from order\_items

update order\_items set orderid = 4 where orderid is null

alter table order\_items add foreign key(orderid) references Orders(orderid)

email varchar(320) check(email like '%[@][gmail][.][com,in]')

create table orders(orderno int,itemid int,qty int,created\_date date default getdate(),

constraint pkorderitem primary key(orderno,itemid),check(qty between 1 and 100)

)

create table student

(id int primary key ,

name varchar(40),

gender varchar(7) constraint chkgender check(lower(gender) in('male','female')),

age tinyint check(age between 1 and 150),

pincode char(6) check(pincode like '[5][0-9][0-9][0-9][0-9][0-9]'))

============

26-2-2025

============

Function – Single Row & Multi Row Functions(Aggregate Functions)

String Functions -upper,lower,left,ltrim,len,replace,replicate,soundex etc

Date Functions --- getdate(),dateadd(),datepart()

Arithmetic Function

Miscellaneous Functions

Conversion Functions

Aggregate Functions (Group Functions)

etc

mobileno char(10) check(mobileno like’[6-9][0-9] [0-9] [0-9] [0-9]’)

Select replace('jack','j','bl')

select replicate

--conversion functions

select convert(varchar(10) ,2024) + customername from orders

select cast(2025 as varchar) + 100

select convert(varchar,getdate(),107)

select try\_parse('Monday, 13 December 2020' as datetime2 using 'en-US') as result

select cast('12-12-2020' as datetime2)

select left(customername,1) from orders

create table customers (custid char(6))

delete from customers

insert into customers values('C00001'),('C00002')

select \* from customers

update customers set custid = left(custid,1)+ format(cast(right(custid,5) as int)+1 ,'d5')

update customers set custid = left(custid,1) + cast(right(custid,4) as int)+1

select firstname,MiddleName,lastname,

case MaritalStatus

when 'M' then 'Married'

when 'S' then 'Single'

else

' '

end as Maritalstatus

from DimEmployee

select \* from dimemployee where baserate = (select max(baserate) from DimEmployee )

select \* from DimEmployee where baserate in (select max(baserate) from

DimEmployee group by MaritalStatus)

-- increase the baserate of all employees by 1 if their baserate is less than

-- the average rate in their gender

Correlated subquery - main execute first and for each row of the outer query inner will execute

Update dimemployee set baserate = baserate +1 where baserate >(select avg(baserate) from dimemployee de where de.gender=dimemployee.gender)

And employeekey=1

select \* from DimEmployee e1 where 3 = (

select count(\*) from DimEmployee e2 where e2.BaseRate>=e1.BaseRate)

select avg(baserate) from DimEmployee where gender='M'

--subquery in the where clause

Update dimemployee set baserate = baserate +1

where baserate <(select avg(baserate)

from dimemployee de where de.gender=dimemployee.gender)

--subquery in from clause

select \* from

(select maritalstatus as stat,

avg(baserate) as average from DimEmployee group by MaritalStatus) table1

--subquery in select list

select gender,baserate,

(select avg(baserate) from DimEmployee e1 where e1.gender=DimEmployee.gender) as average

from DimEmployee