Database ---

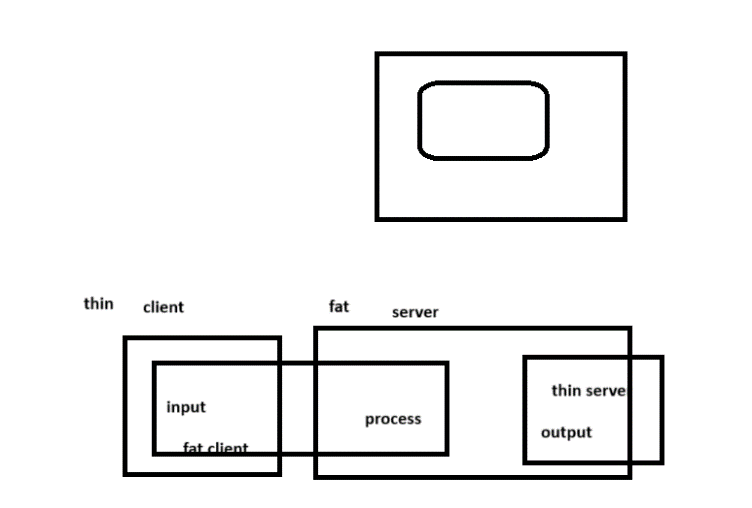
Normal – Traditional

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

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

27-2-2025

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

.Net Framework --: is software needs to be installed in development machines, tools and base class library

.net Framework 1.0 - 4.8 - works only in windows platform

.Net Languages - VB.Net, C#.Net,C++.Net, Java.Net,F#.net etc

.Net Runtime -

Abc.cpp ->compile->obj ->make ->executable

C#->compile ->Assembly.dll/exe

MSIL/IL/PE

C#

Int a; ----- 1011001

Vb.net

Dim a as integer -- 1011001

CLS – common language specification

CTS – common type system

C# language —

Class calculator{

Int number1;

Int number2;

Int Result;

Void add(){

}

Void multiplity(){

…

}

}

Void eat(){}

Void main(){

Eat();

\*ptr = new Calculuator();

Ptr->Add();

delete ptr;

}

C# is a pure object oriented language

A project – collection files(code files, configuration files, images etc)

Build -> Assembly

Dotnet cli – IDE, download template from cloud

Top Level statements

Solution – is a collection of projects

Syntax:

[<access specificer>] [<modifier>] <datatype> <variablename>;

Eg: int a;

protected static float price;

Access Specifier – public , private, protected, internal, file, protected internal , private protected

Modifer: static, abstract, sealed, volatile, transcient ,virtual ,unsafe etc

Datatypes: byte,sbyte,short,ushort,int ,unit,long,ulong,decimal,float,double,char,bool

Variablenames can begin with char,underscore,followed by digits

C# -

Int -- Int32

Real estate

Data

Process

Calculator

Num1,num1

Add

Mul

Div

Sub

Encapsulation- class

Class, struct,record

Area,price,owner,location,contactperson,type,

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

28-2-2025

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

Dotnet is an name of the initiative started by Microsoft to solve the difficulties of N-Tier Application Development.

Dotnet Framework

OOAD-- object oriented analyses and design(UML)

Visio - class diagram, sequence diagram, statemachine diagram

Kind of, has a, is a,uses

Inheritance , composition , utilization

Abstract class – is a class used to hold common properties of related classes

[<acceesspecifier>] [<modifier>] class <classname>{

Body

}

Default access specifier for class is internal

Default access specifier for functions and properties is private

Class Engine{

Void poweron(){}

Void poweroff(){}

Void combustion(){}

Int fuel,air,spark;

}

abstract class Car{

abstract Engine afcd;

abstract Void acceleariont();

abstract oid stopcar();

}

Class Ford:Car{

Engine ford;

Void acclearation(){//ford accleration}

Void stopcar(){//ford stopiing}

}

Class Honda:Car{

Engine honda;

Void acclearation(){//honda accleration}

Void stopcar(){//honda stopiing}

}

Class Driver{

Void Driving(Car c){

c.accelration();

c.stopcar();

}

}

Driver anil = new Driver();

anil.Driving(new Honda());

anil.Driving(new Ford());

encapsulation,inheritance

abstraction:- provide necessary features to end user from users perspective. Users can be same class, child class in the same name namespace, child classes in other namespace, other classes in the same namespaces, other classes in another namespace.

One way to implement abstraction is using access specifiers

Array- Group of variables of same/different type

Syntax:

<Datatype> []<arrayname> = new <datatype>[size];