(V-1)
compatible

PARALLEL ARCHITECTURE

SPEED UP → adding more resources result in proportionally less running time for a fixed amount of data

eg 10sec to scan a database of 10,000 records using 1 CPU

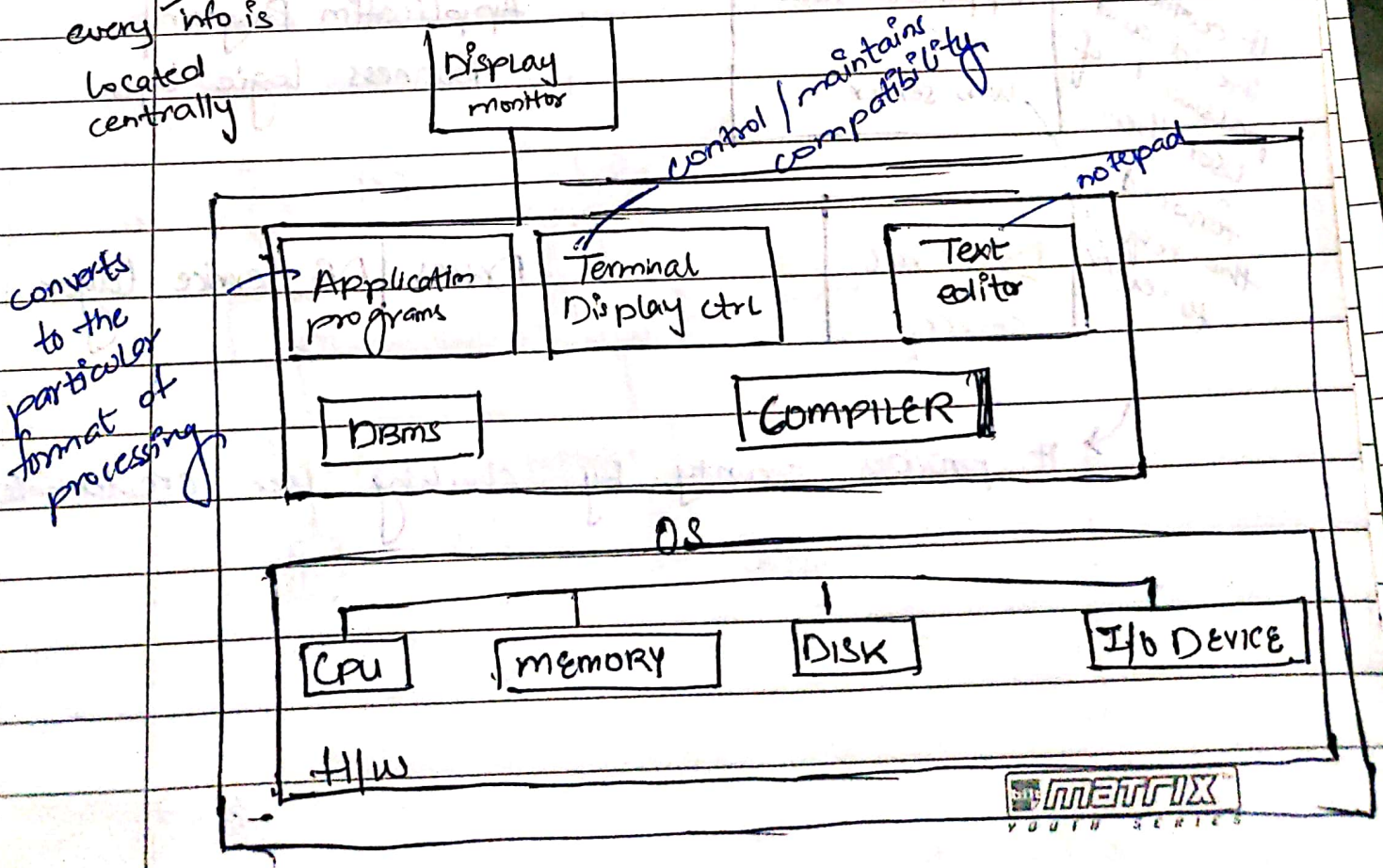
IMP →

SCALE UP → if resources are increased in proportion to increase in data or problem size the overall time should remain the same

eg 1 sec to scan a database of 1000 records using 1 CPU

1sec to scan a database of 10,000 records using 10 CPU

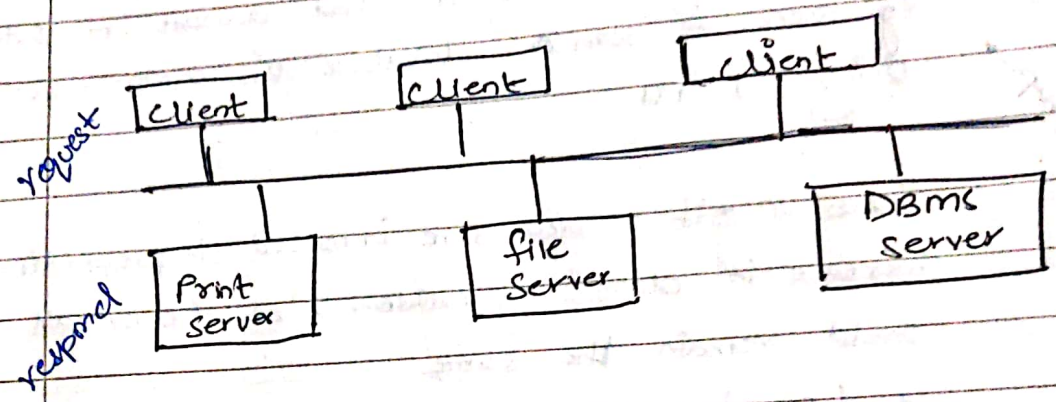
CENTRALIZED DATABASE ARCHITECTURE



Scanned by CamScanner

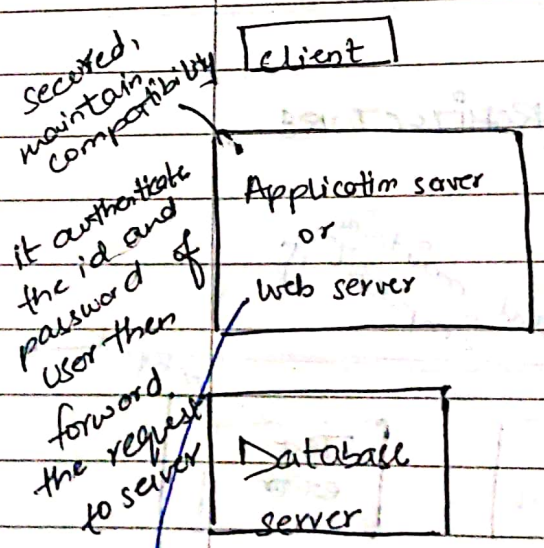
communicating without an intermediate layer

(2) CLIENT SERVER ARCHITECTURE (2 TIER)



(3) 3 TIER ARCHITECTURE

eg chalkpad



GUI | Presentation layer

Application Program | Business logic layer

DBMS | DB Service layer

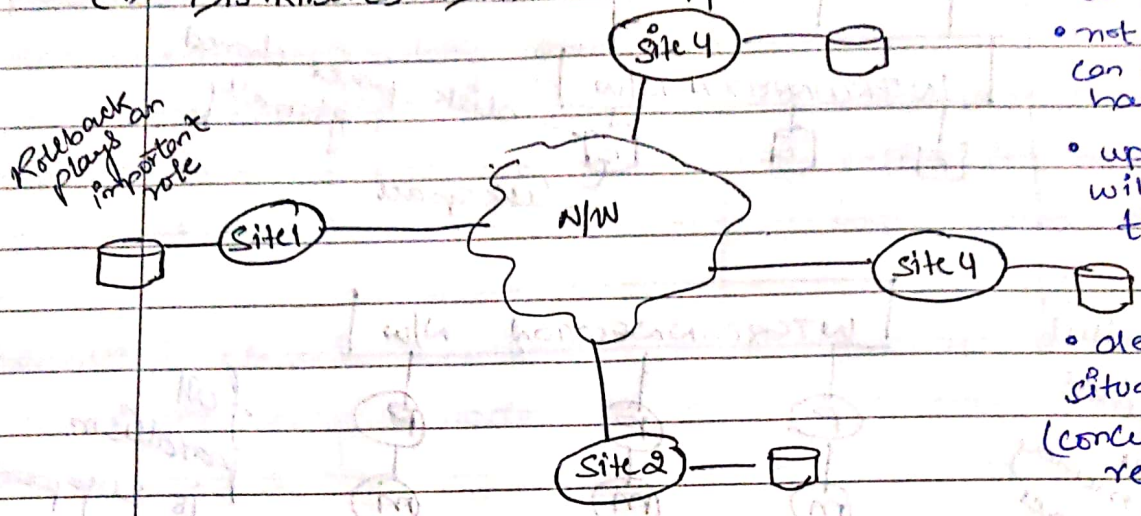
It provides security by checking user credentials

eg Airline Booking

disadvantage security

There is never any data loss

(4) DISTRIBUTED DATABASE ARCHITECTURE



- expensive
- complex
- not secured can be hacked
- updatin will take time
- deadlock situation occurs (concurrent request)

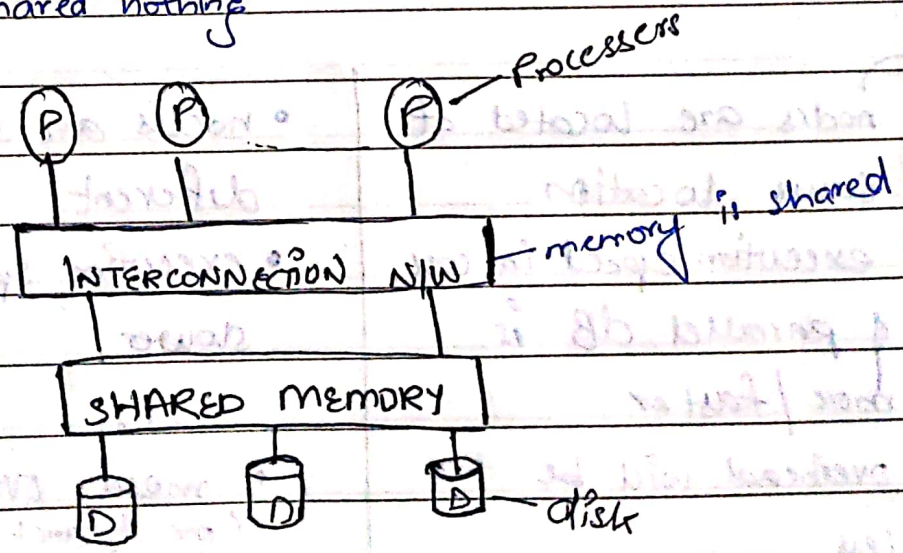
(5) PARALLEL DATABASE ARCHITECTURE

(i) shared memory

(ii) shared disk

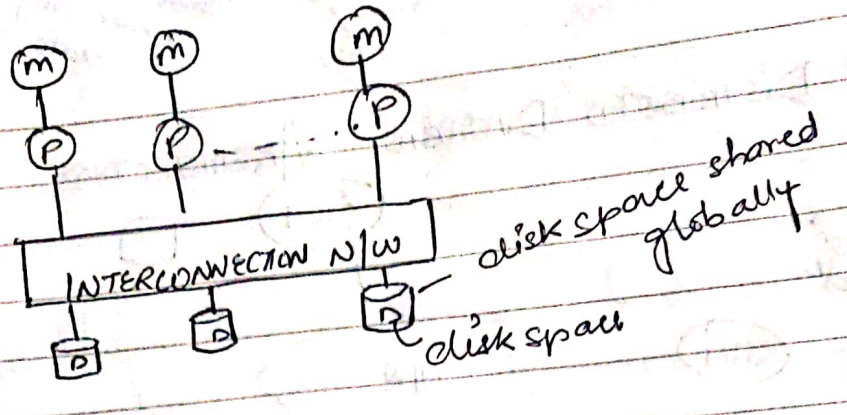
(iii) shared nothing

(i)

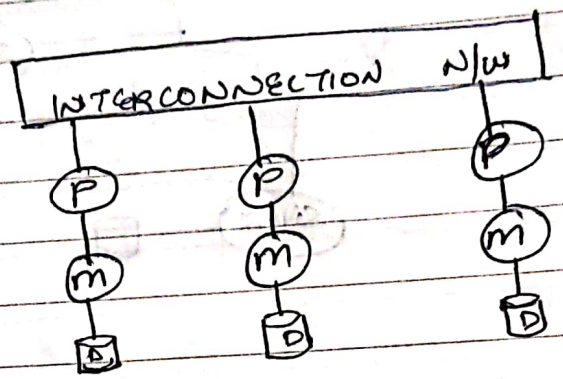


Scanned by CamScanner

(ii)



(iii)



all processes have their particular memory & disk space
no sharing is there

full parallism is supported

DIFFERENCE B/W PARALLEL & DISTRIBUTED

	PARALLEL	DISTRIBUTED
processes/tasks	• nodes are located at same location	• nodes are scattered/at different places
EXECUTION SPEED	• execution speed in case of parallel dB is more/faster	• execution speed is slower
OVERHEAD	• overhead will be less	• more overheard (one branch needs to perform all the tasks and updation on other branches are done)
HOMOGENEOUS/HETROGENEOUS	• It is homogeneous (system is same)	• It can be homo/hetrogeneous (depends on branches)

Scanned by CamScanner

		DB can never be lost (reliability & availability is always there)
PERFORMANCE	• less reliability and availability	• more availability & reliability therefore high performance
BACKUP	backup is available at only 1 node	• backup is available at every site/node

matrix YOUTH SERIES

Scanned by CamScanner

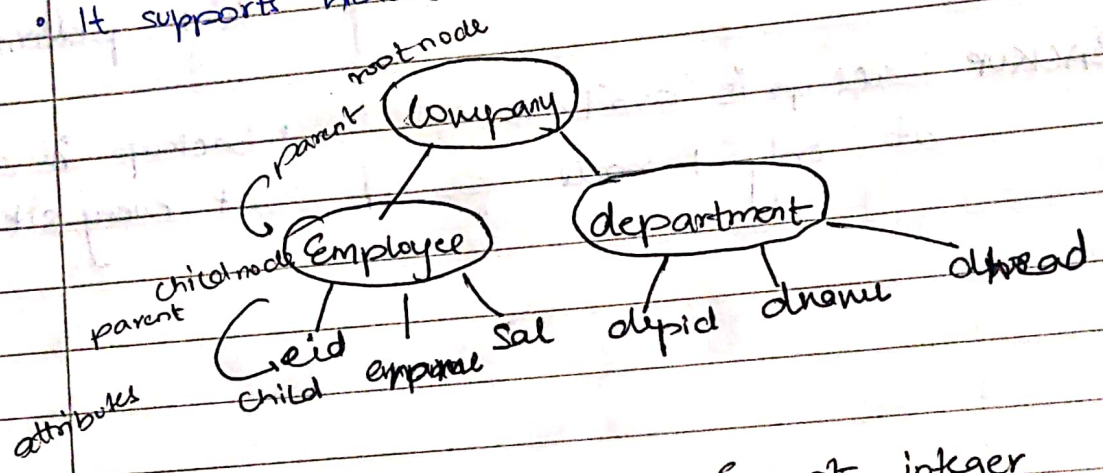
Extended part of HTML

root node

XML (EXTENSIBLE MARKUP LANGUAGE)

case sensitive

- To make page more dynamic
- no predefined tags are used. user create there own tags and use them.
- maintains consistency and provides framework
- It supports hierarchical structure (tree structure)



Except integer other data must be in ' ' or " "

```
<root node>
<company>
<employee>
<eid> 101 </eid>
<ename> 'abc' </ename>
<sal> 5000 </sal>
</employee>
<department>
<deptid> 10 </deptid>
<dname> CSE </dname>
<dhead> 'xyz' </dhead>
</department>
</company>
```

child element

.xml extension

To give this XML document meaning we do-

(1) DTD - document type defination

(2) schema

DTD (DOCUMENT TYPE DEFINATION)

Grammer:

eid, ename etc — structure → DTD [.dtd] — XML schema defination

content → schema [.xsd]

101, 'ABC' — [type of content int, string]

→ DTD file

root node — child nodes of root node

<!ELEMENT company (employee, department)>

<!ELEMENT employee (eid, ename, sal)>

~~<!ELEMENT department (#PCDATA)~~

<!ELEMENT eid (#PCDATA)> — parse character data: It display the stored data without knowing its type.

<!ELEMENT ename (#PCDATA)>

<!ELEMENT sal (#PCDATA)> — they are the final attributes

<!ATTLIST eid (#REQUIRED)> // for making eid mandatory

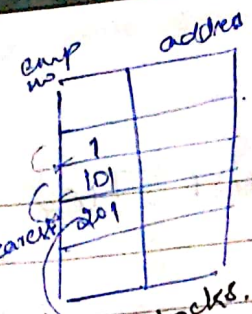
matrix YOUTH SERIES

Scanned by CamScanner

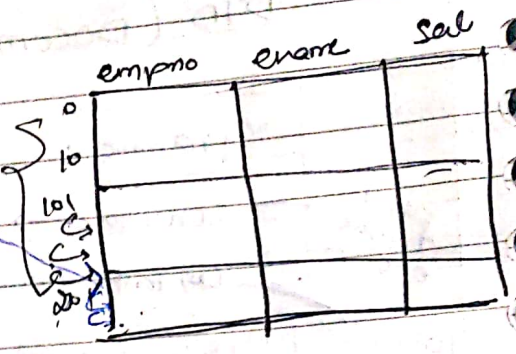
search is performed using limited data

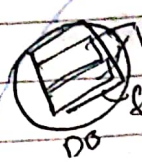
INDEXES

- sparse
- dense — used for unordered data



search 210





Address

000000 FFF BBBBBB RRR

object address	file	Blocks	Row
DB 6	3	6	3

(subdivisions)

Row - ID FORMAT

→ to traverse each and every row to search data. (like normal searching)

(1) DATA OBJECT No. - assigned to each data object such as table, dB, then it is created and it is unique within the dB. represented with 6 0's

(2) RELATIVE FILE NO :- It is unique to each file within a dB or tablespace represented by 3 f's Data files are the files used by the oracle engine to store user's data

(3) Block No:- represent the position of the block containing the rows. within the file

(4) Row No - identifies the position of the row, pos in the block header. Its row no. is 3 times R.

Scanned by CamScanner

/

Each record in the datablock is given a unique record no.

How To CREATE AN INDEX

1) UNIQUE

2) NON UNIQUE

COMPOSITE INDEX considering more than 1 attribute in index

create index index_name on table_name(col_name1, col_name 2)

can be single as well

for unique

create unique index

BITMAP INDEX

Bitmapindex index_name

create ~~index~~ ~~Bitmap~~ on emp(Image);

only for image

CLUSTER

can store more than 1 table is same address space

In joins to move from 1 address to another ∴ it consumes more time to overcome this time consumption problem cluster is created

can never be created if already created table address are allocated it can never be changed

A db object that stores data related to 2 or more table in a single disk space | memory location.

MATRIX YOUTH SERIES

Scanned by CamScanner

1) create cluster
2) create index on cluster
3) create table

block area

create cluster cl(d number(2));
cluster name

create index ci1 on cluster cl; — dno indexing in block

create table dept1 (dno number(2), dname varchar(20))
cluster cl(dno); — primary key

create table emp1 (empno number(20), ename varchar(20),
dno number(2)) cluster cl(dno);

value must be same

Scanned by CamScanner

PL/SQL

It is not case sensitive

Declare (if no variables required then we can ignore)

compulsion to right

Begin
→ start;

Exception (not compulsory)

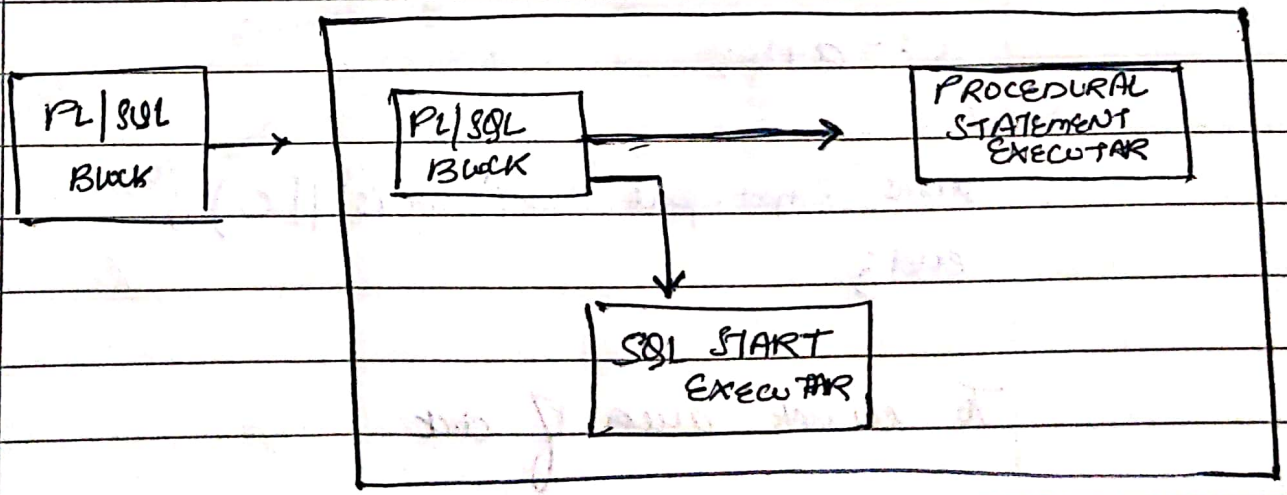
end; ; (compulsory)

→ for Comments

Single line → --

multiple lines → /*

*/



matrix YOUTH SERIES

Scanned by CamScanner

→ To terminate the statement ; (semicolon) is must

- Begin and end define the scope of the code
- To print any statement DBMS_output.put_line ('Hello');

To print a variable → a=2

DBMS_output.put_line ('a='||a);

package (inbuilt)

put_line → function.

To ASSIGN VALUE
a:=2.

PROGRAM TO ADD TWO NUMBERS

```
Declare
a number(20);        or  a number(20)
                         =2;
b number(20);
c number(20);
Begin
  a:=:a;   // to take value from the user
  b:=:b;
  c:=a+b;

  dbms_output.put_line('sum is'||c);
end;
```

2Q CMD LINE

```
set serveroutput on;

a:=&a;
```

To calculate area of circle

TYPES OF DATA

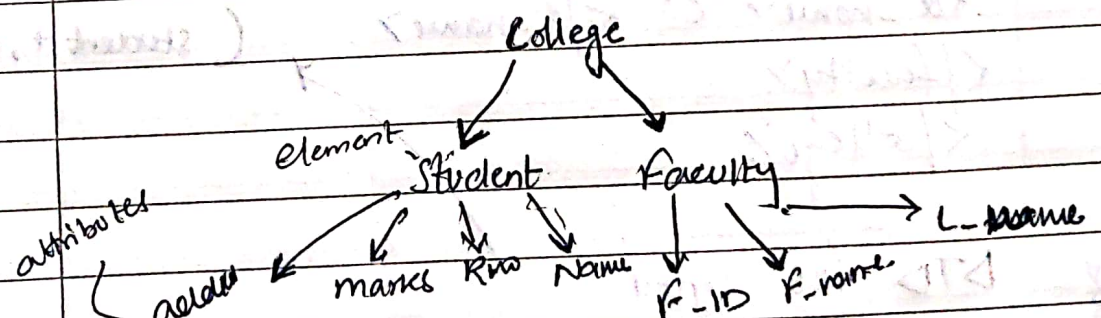
(1) STRUCTURED - data follows a specific structure and constraints eg. data in dB

(2) UNSTRUCTURED DATA - eg clobs (word document, excel sheet etc), Blobs (where images are kept. ie size can vary. images also don't have specific structure/ format jpg, png).

where document type of data is kept

(3) SEMISTRUCTURED DATA - the data that is partially structured as well as unstructured. eg XML document.

Data which have information of schema in itself is called semistructured data.



Scanned by CamScanner

Complex Element - which can further divided

Simple Element - which cannot further divided

eclipse

XML DOCUMENT

<?xml version="1.0" encoding="UTF-8"?>

<!Doctype college System "first.dtd">

keyword

root element

dtd file

<college>
<student>
<marks> 80 </marks>
<rno> 1 </rno>
<name> 'A' </name>
<Address> 'chd' </Address>
</student>
<faculty>
<f_id> 101 </f_id>
<f_name> 'B' </f_name>
<l_name> 'C' </l_name>
</faculty>
</college>

If there are more than 1 child nodes like student

(student +, faculty)

first.dtd

DTD FILE (.dtd)

<!ELEMENT college (student, faculty)>
<!ELEMENT student (marks, rno, name, address)>
<!ELEMENT marks (#PCDATA)>
<!ELEMENT rollno>
<!ELEMENT name>
<!ELEMENT address>

attribute list.

<!ATTLIST RNO (#REQUIRED)>

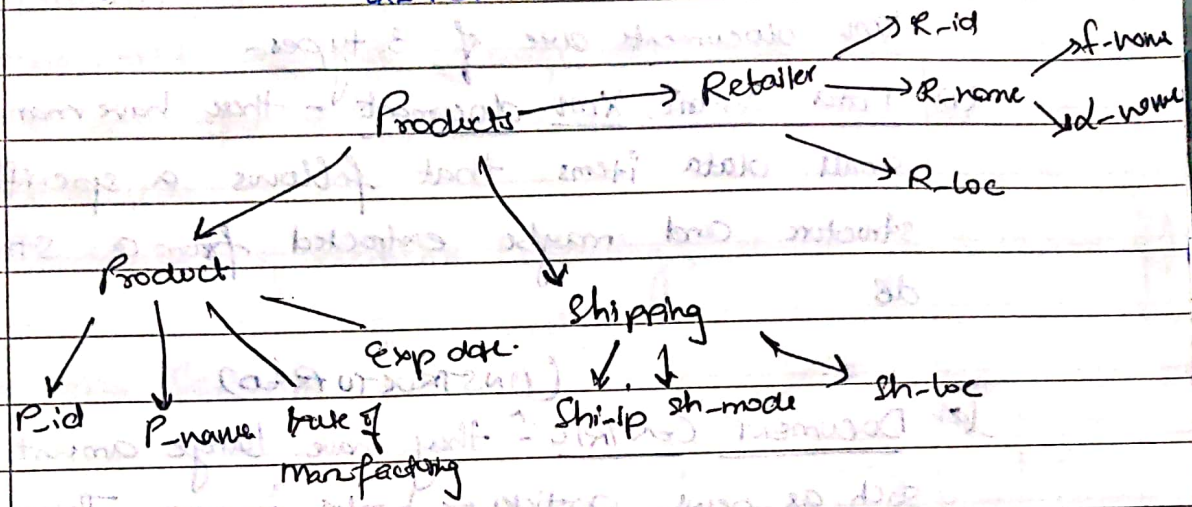
compulsory to fill

Scanned by CamScanner

<!ATTLIST student RNo (#REQUIRED)>
if declaring afterwards.

<!ELEMENT faculty (EID, name, d_name)>
<!ELEMENT eid (#PCDATA)>
name
d_name

(ii)



Scanned by CamScanner

COMPLEX ELEMENT - these are constructed from other elements hierarchally. whereas simple elements contain data values

XML documents are of 3 types -

(1) Data centric XML document - these have many small data items that follows a specific structure and maybe extracted from a structure db

(2) DOCUMENT CENTRIC (UNSTRUCTURED) - they have large amount of text such as news, articles or books, images. They ~~are~~ donot have any structure of their own

(3) HYBRID (Semistructured) - they have parts that contained structured data and other parts that are unstructured. They may or may not have predefined schema

- NOTATIONS IN XML DTD

⊙ A* where A is element name means element can be repeated 0 or more times in document

⊙ A+ + following an element means it can be repeated 1 or more times in a document

Scanned by CamScanner

- A? following an element means it can be repeated 0 or 1 time

- LIMITATIONS OF DTD
- no valid datatype (always #PCDATA)
- It has special syntax and require specialised processors
- ordering of elements (ordering similar as in hierarchy)
- XML SCHEMA — (.xsd)

(i) schema description and XML namespace — to refer to the address (where you want to create webpage)

(by default)

<?xml version="1.0" encoding="UTF-8"?>

<xsd : schema xmlns : xsd = "https://www.class.org/2010/xml-schema">

overall reference of address — tag name — attribute — Object of xmlns (for not mentioning the address again and again) — address

this is a commonly used standard for XML schema commands. such type of def. are called as XML namespace coz it defines the set of commands that can be used. (xsd)
The file name is assigned to the variable using attribute xmlns which is storing the actual address. It is used as a prefix to all schema commands.

Scanned by CamScanner

for title & heading

(2) ANNOTATION, DOCUMENTATION AND LANGUAGE :-

always used with documentation as comment given inside it is displayed as heading

```
<xsd: annotation>
<xsd: documentation xml: lang = "en">  company schema
</xsd: documentation>
</xsd: annotation>
```

type of language to be used in our xml page

These are used for providing comments / heading / title and other description in other document

(3) ELEMENTS & TYPES (root element have only 1 element)

```
<xsd: element name = "company">
<xsd: complex type>
<xsd: sequence>
<xsd: element name = "department" type = "Dept"
min_occurs = "0"  max_occurs = "unbounded" />
```

tag

we are having more child node not attribute.

sequence of mentioning is same as representing

Company → project → department employee

(child element have more than 1 attribute)

column can be kept empty (how many values needs to go into the column)

XML DOCUMENT

can be redefined (root element)

Scanned by CamScanner

CONTROL STRUCTURES

(i) CONDITIONAL CONTROL STATEMENTS
- if-then-End if
- if-then-Else
- if-then-elsif

(ii) Iterative control statement
- Basic loop
- while loop
- for loop

(iii) Sequential control statement
- Goto
- Continue
- Exit

- if(<condition>) then
  sequence of statements
  End if;

- if <condition> then
  sequences of statements
  Else
  sequences of statements
  End if

- if <condition> then
  Seq. of statements;
  Elsif <condition> then
  Seq. of statements;
  Else
  Seq. of statements
  End if;

Elsif

- case
  when Exp1 then result 1
  when Exp2 then result 2
  when Exp 3 then result 3
  Else
  End;

- a:=:a
  case a
  when 'A' then 'Good'
  ⋮
  dbms_output.put_line(' ')

- WAP to check whether a no. is a Armstrong no or not
- Print fibonnaci series
- multiplication table of a given no.
- Palindrome no. (reverse same)

ITERATIVE

Basic loop:

```
loop
    sequence of statements;
End loop.
```

do-while.

```
loop
sequence of statements
Exit when <condition>;
End loop;
```

while

```
while <condition>
loop
sequence of statements
End loop;
```

for (by default increment by 1)

```
for counter IN [REVERSE] Initial
                    value .. final value
loop
    statements
End loop;
```

eg for i in reverse 1..5

eg for a in 1..5

- WAP to accept no. from user & print in reverse order
- reverse of string — length, substr.
- factorial of a number

→ continue, exit

```
Goto label_name;
_____
_____
_____
<<label_name>>
```

jumps.

```
declare
a number;
begin
select sellp into a from product where pro='pr';
 if (a<2000)
 goto old_price;
Else
dbms_output.put_line('current price is' || a); End if;
<<old price>>
update product set sellp=5000 where pro='pr';
End;
```

Scanned by CamScanner

~~ELEMENTS~~ XML SCHEMA

(1) XML namespace :-

<p xml version="1.0" encoding="UTF-8"?>
<xsd : schema xmlns:xsd="https://www——">

(2) ANNOTATION & DOCUMENTATION

<xsd annotation>
<xsd documentation xml:lang="en"> company </xsd
d: documentation> </xsd: annotation>

(3) ELEMENTS & TYPES

<xsd element name="Company">
<xsd: complex type>
<xsd: sequence>
<xsd element name="department" type="dept"
min_occurs="0" max_occurs="unbounded"> (</>) that means we closed it
<xsd element name="employee" type="emp"
min_occurs="1" max_occurs="unbounded">

company
Dept emp
dno dname eno ename

<xsd: unique name="employee ename Unique">
node name — attribute to be made unique — keyword
<xsd: selector xpath="employee"/>
attribute. — node on which unique tag is to be applied
is path for konsa field wo is that is made unique
<xsd: field xpath="employee ename"/>
</xsd: unique>

always used together

Scanned by CamScanner

for making Primary Key

<xsd: key name = "~~department~~ employee empno key"> (tag)

<xsd: selector xpath = "employee"> — node in which primary key is applied

<xsd field xpath = "employee empno" /> — attribute that is to be made Primary Key

node name

</xsd: key>

foreign Key

<xsd: keyref name = "employee Department dnokeyref" (foreign key table; primary key node)

refer = "Department dnokey"> (Which is primary key node)

<xsd: selector xpath = "employee" />

<xsd: field xpath = "employee Department dnokey" /> (foreign key; primary key)

</xsd: keyref>

Company

Department

Name dno dloc

<xsd: complex type name = "Department"> — as Dept node has multiple attributes

<xsd: sequence> — first name then dno then dloc

<xsd element name = "department Name"

type = "xsd: string" /></

```
<xsd: element name = "department dno"
        type = "xsd: Integer" /> 
```

element name

```
<xsd: element name = "department dloc"
    type = "xsd: String" />
```

datatype

```
</xsd: sequence>
</xsd: complex type>
</xsd: schema>
```

XML LANGUAGES

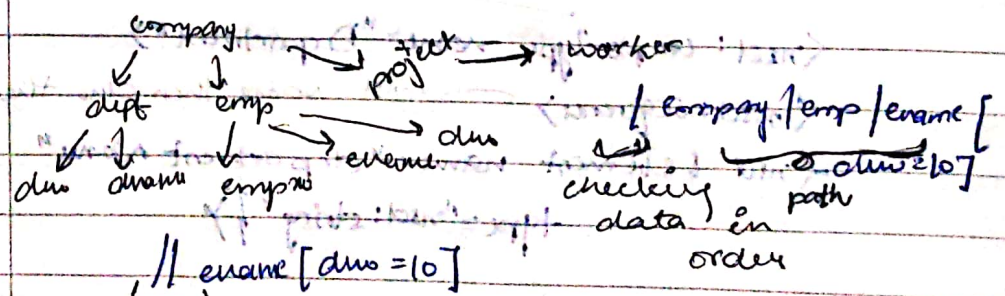
Xpath

- It provides language constructs for specifying path expression to identify certain nodes or attributes with an XML document that match specific pattern.

Xpath →

Xquery

- An Xpath generally returns a sequence of items that specify a certain pattern as specified by expression these items are either values or elements or attributes



query

```
//ename[dno=10]
```

it will go to root node and sequentially checks all nodes till it get ename

- The names in Xpath expression are node names in XML document. There are two main seperators which are used while specifying a path ie (/), (//)

- A single slash before a tag specify that the tag must appear as a direct child of previous tag

- A double slash specify that the tag can appear as a descendant of previous tag at any level

eg (1) /company // display all child nodes & attributes

(2) /company/department // attributes of department

(3) //employee [sal gt 70000] /ename
— as we don't know exact location
node
attribute

(4) /company/project/worker [hrs gt 70] /hrs
display column

→ column name written after single slash will be displayed

Location /company
(
address (complete)
or store in a variable $x = "
$x/company

Marathe

- It is compulsary to include file name in xpath query.

PL/SQL like: **XQuery**

Specifying queries in XML:-

- Xpath allows us to write expression that select item from a tree structured XML document. XQuery permits the specification of more general query on one or more documents.

- The structure of XQuery is represented as:

  FLWR — result/Return or LFWR

  for let where

variables

address

let $d = doc(www.company.com)

company emp emp project ssn ssn ssn hrs

$x = doc(www.xyz.com)

eg

let $d = doc(www.mywebpage/info.xml)

for $x IN $d/company/project[p_no=5]/proj_worker,

representing the project

$y IN $d/company/employee, where $x/-

employee node

hours gt 20 AND $y.ssn = $x.ssn

Return <res> $y/emp_name/f_name: $y/emp_name/l_name; $x/hrs

like select statement

</res>

SCOPE OF VARIABLES

```
Declare
  A number;
  B number;
Begin
  A:=A;
  B:=:B;
Declare
  C number;
Begin
  C:=A+B;
dbms_output.put_line('sum is ' || c)
end; end;
```

global

if C:=A+B written outside first end then it will show error as its scope is local.

→ %Type → it will make the datatype of variable same as column datatype. Each columns have diff variable

x emp.empno%type

variable table name column name

→ %RowType → it will make the datatype of variable same as all columns datatype. In this we need to define single variable for all columns

matrix YOUTH SERIES

Scanned by CamScanner

AUTONOMY — every node is independent of how much data they need to share.

Homogeneous
Heterogeneous

DISTRIBUTED DATABASES

These are those computing system in which we partition a big unmanageable problem into smaller pieces and solve it efficiently in a coordinated manner. It is a collection of multiple logically interrelated databases distributed over a computer network and a distributed database management system as a software that manages a distributed database while making the distribution transparent to user.

→ TRANSPARENCY

hiding implementation details from end users in distributed dB the data and s/w are distributed over multiple sides connected by a computer n/w so additional transparencies are introduced

Imp.
- (1) DATA ORGANISATION TRANSPARENCY
- (2) REPLICATION TRANSPARENCY
- (3) FRAGMENTATION TRANSPARENCY (division of dB)
- (4) LOCATION TRANSPARENCY

Scanned by CamScanner

ADVANTAGES

- Improved ease & flexibility of application development
- Increased reliability and availability (correctness)
- improved performance
- easier expansion

FRAGMENTATION IN DISTRIBUTED DATABASE

breaking the db into logical units called fragments which maybe assigned for storage at various sites

- HORIZONTAL FRAGMENTATION (σ)

It is a horizontal subset of relation which contain those tuppies which satisfy selection criteria represented by σ (select operation)

dealing with columns

- VERTICAL FRAGMENTATION (π)

primary key is always included in vertical fragmentation

matrix YOUTH SERIES

Scanned by CamScanner

TO CREATE CUSTOM EXCEPTION

```
Declare ___ Exception name
Exception1 Exception;
eno emp.empno %type;
name emp ename %type;
Begin
  eno :=: eno;
  name :=: name;
if eno is Null then
raise Exception1;
End if;
Insert into emp(empno, ename)
   values (eno, name);
dbms_output.put_line('Record inserted');
exception
 when exception1 then
   dbms_output.put_line('column field is empty');
 end;
```

RAISING AN EXCEPTION

```
Declare
Salary_too_high exception;
error_sal number;
max_sal number := 10000;
cur_sal number := 20000;
```

Scanned by CamScanner

```
Begin
Begin
  if  curr_sal > max_sal then
    raise salary_to_high;
  end if;
Exceptim
  when salary_too_high then
  dbms(—); dbms('max salary is'||max_sal);
raise; // so that program doesnot end here
                it skips next line
end;
exceptim
when sal_too_high then
  error_sal = current_sal;
  current_sal = max_sal;
  dbms('revised sal is '||currentsal);
end;
```

matrix YOUTH SERIES

Scanned by CamScanner

5 marks

Imp eg

VERTICAL FRAGMENTATION

project

Pno	Pname	Budget	Location
P1	CAD/CAM	125000 ✓	New york
P2	Instrument	250000	Japan
P3	Php	135000 ✓	China
P4	Java	310000	India
P5	CAD/CAM	500000	Japan

(1) Horizontal fragmentation

proj 1 → Budget < 200000

Pno	Pname	Budget	doc
P1			
P4			

- create a fragment where project name is CAD/CAM

Pno	Pname	Budget	Location
P1			
P5			

- create a fragment consisting of Budget (vertical fragmentation)

by default primary key column included always

Pno	Budget
P1	
P2	
P3	
P4	
P5	

- create a fragment consisting of Pname & Budget whose budget is more than 200000

MIXED FRAGMENTATION

- combination of both projection & selection

TYPES OF HORIZONTAL FRAGMENTATION

- PRIMARY HORIZONTAL FRAGMENTATION
  fragmentation done where primary key is mentioned.
  → fragmentation of a relation on the basis of primary key

- DERIVED
  fragmentation of secondary relation (where primary key made foreign key) that are dependent on primary relation that is related with foreign key

- COMPLETE HORIZONTAL FRAGMENTATION
  → Horizontal fragments have each and every tupple of the original relation will belong to atleast one partition

eg fragment 1 : σ(empage ≤ 21) emp
fragment 2 : σ(18 $<$ empage ≤ 65) emp
fragment 3 : σ(empage > 65) emp emp → fragment 1 ∪ 2 ∪ 3

reconstruction of a original table.

100 rows in original table then the total no. of tupples / records in the given fragments will be either 100 or more than 100

matrix YOUTH SERIES

Scanned by CamScanner

- VERTICAL FRAGMENTATION

- each site may not need all the attributes of a relation then we use vertical fragmentation which divides a relation vertically by columns

- It is a subset of a relation which is created by a subset of columns

- because there is no condition for creating a vertical fragment each fragment must include the primary key attribute of the parent relation in this way all the vertical fragments of a relation are connected

→ MIXED FRAGMENTATION

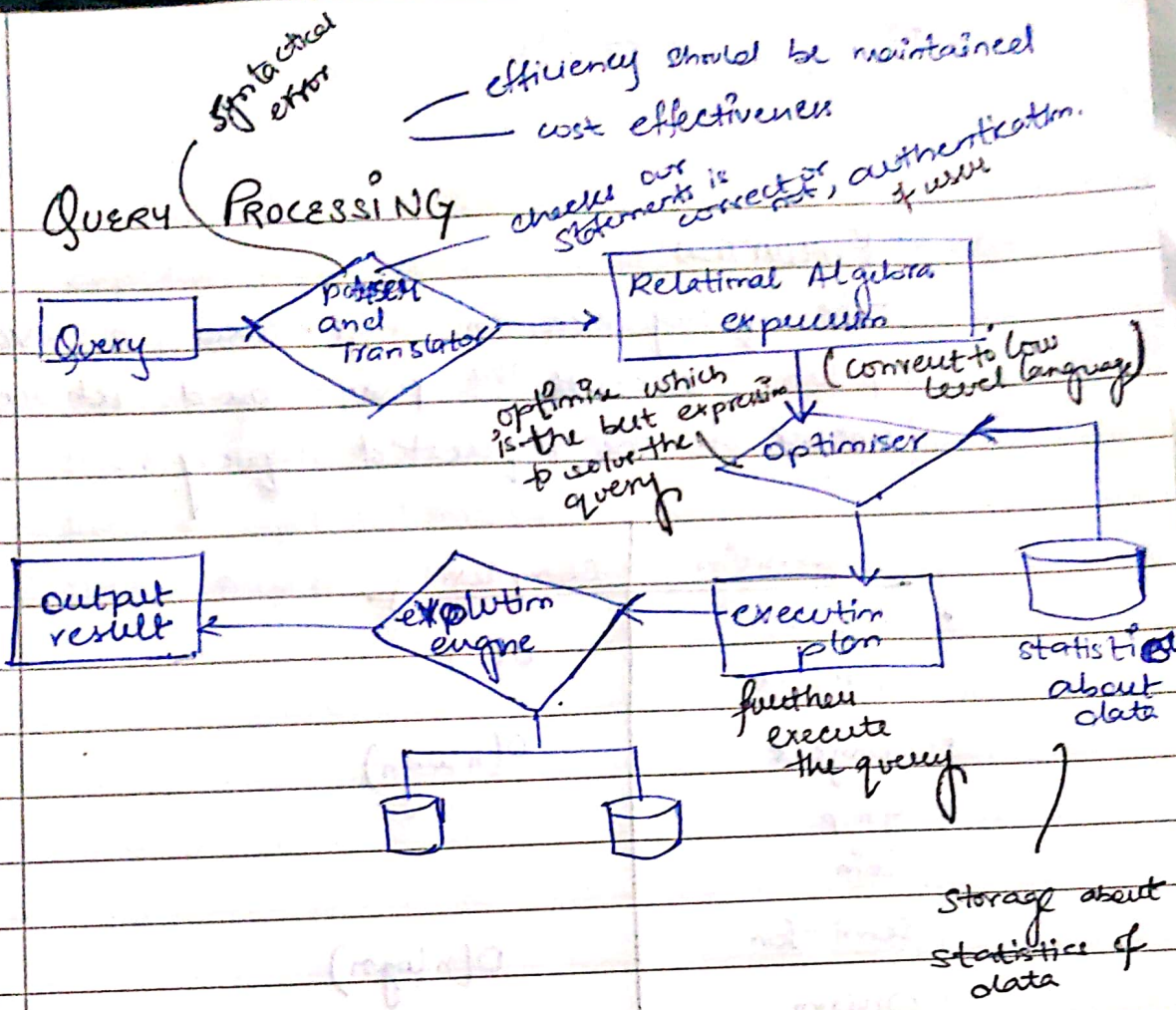
combination of vertical & horizontal fragmentation is known as mixed frag. This is achieved by select & project operation which is represented by $\pi_{li}(\sigma_{ui}(R))$

condition condition

Scanned by CamScanner

V.Imp

QUERY PROCESSING



- It is a 3 step process that transforms a high-level query into an equivalent and more efficient lower level query which is relational algebric expression

PARSER & TRANSLATOR

- checks the syntax and verifies relation.
- translate the query into an equivalent relational algebraic expression

OPTIMISATION

generates an optimal evaluation plan with lowest cost for the query plan.

Scanned by CamScanner

EVALUATION

the query execution engine takes a evaluation plan execute the plan and returns the answer of the requested query.

operation	complexity
• Select, project	$O(n)$
• project	$O(n \log n)$
• Group	
Join	
Semi-Join	$O(n \log n)$
Division	
Set operators	
• cartesian product	$O(n^2)$

emp(empno, ename, title) ; asg(eno, pno, Resp, dur)

Select ename from emp, asg where emp.eno = asg.eno and dur > 37

Que find the name of employee who are managing a project and duration are more than 37 hours.

$\Pi_{ename}(\sigma_{Dur>37} \wedge emp.eno = asg.eno (emp \times asg))$

or $\Pi_{ename}(emp \bowtie_{eno} (\sigma_{Dur>37}(asg)))$;

Scanned by CamScanner

Que emp(eno, ename, title) ; asg(eno, pno, Resp, dur)

site1 : Asg 1 = $\sigma_{eno \leq e3}$ (asg);

site 2 : Asg 2 = $\sigma_{eno > e3}$ (asg);

site 3 : emp1 = $\sigma_{eno \leq e3}$ (emp);

site 4 : emp2 = $\sigma_{eno > e3}$ (emp);

Scanned by CamScanner