Nivathe
Evan Bioness

16/07/19 ADBMS

Query performance tools

Theory
- → Indexes
- → Types of Indexes
- → Views
- → cluster
- → sequences

lab
- Implementation of Index, tclister
- " View and sequences

Types of Indexes

1. Implicit :- automatically generated by sql engine
2. Explicit

① → single column index — Create index <index Name> ON <TNT> (<column no>);
② → composit column indexes — [Create index <index name> ON <TNT> (col, col2);
③ → unique index [Create Unique index <index> ON <TNT> (col);
④ → cluster index [Create clustured index <index-name> ON <TNT> ();
⑤ → Bitmap index [Create bitmap index <index-name> ON <TNT> (col);
⑥ → Function base Index
⑦ → B-Three

Index select * from FTE where salary > 10000 and salary < 20000;

Create Index <index no>
ON <FTE> (salary Asc)

display
select Row id, salary from FTE;

id	Name	Gender	designation District	Salary
1	Aman	M	AD	2500000
2	Anish	M	AD	2000000
3	Ritu	F	MD	350000
4	Ram	M	CEO	15000
5	Rani	F	DSP	4500

→ ~~query performances tools~~ —

Eg

Salary	Row address
150000	BBBBBBBB.RRRR.FFFF
2000000	
2500000	
350000	
45000	

BBBBBBBB → called datablock where data record are store on DB

RRRR → second NO

FFFF

Data file & Name	Data file NO	size
—	1	10MB
—	2	"
—	3	"
—	5	"

→ a single column index is created based on only

→ • Implicit - automatically generated by oracle
• E → ——— user
index always implement on col'm,
work like pointer

• In composite index - implement more then 1 col
unique — - table having duplicate value in col then using unique value of attr

- cluster index create only once in particular table

- Bitmap Index - oracle not support Bitmap index but in the case of ~~mno~~ provide most uniqueness of
- Func^n based Index - It will use when col^n use any arithmetic expression.

→ create table of full time employee where attr is emp id, emp name, gender, designation, salary implement explicit index (single, composit unique, cluster)

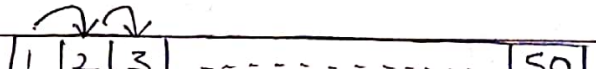
17/July/2019

B-Tree Index

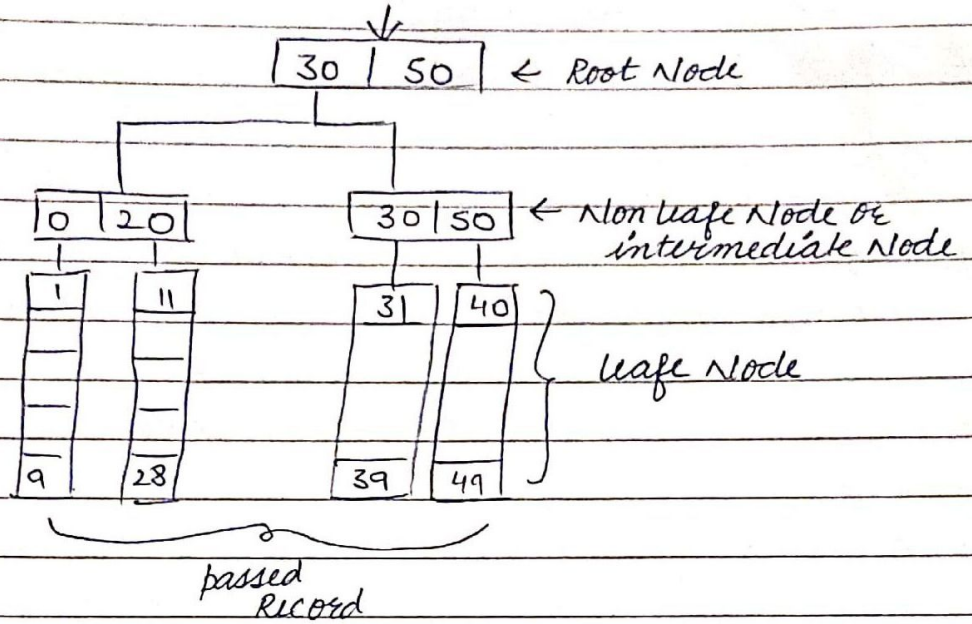
FTE

Query performance tools	Id	Name	gender	desig.	Sal
1 Index					
2 View					
3 Sequence	50				
4 cluster					

Before apply Index



- left to Right
- ~~sequential order~~



→ View is saved sql query. It is virtual table of base table. It is used for security purpose. It can implement on row and column level. If we have update the view table, main table also affected

→ operation of View

- Insert (insert into <view-name> values ( )
- update : update <view-name> SET <attribute-name> = ' ' WHERE <attr name> = ' ';
- delete : Delete from <view-name> WHERE <attr. name>
- drop : Drop view <view name> = ' ' ;

→ create table my dept
→ Insert data
→ Create view

Create View <view-name> AS SELECT <attribute>
FROM <-tablename> ;

Syntax

CREATE VIEW <View-Name> As
SELECT <colname1>, <col2> FROM
<tablename> WHERE <col-name> = <Expression list>
groupby <grouping criteria> having <predicate> ;

display→ SELECT * FROM VIEW-NAME ;

⇒ Create table of My department, insert 5 values
↓ then implement the ~~table~~ dml operations
(Create the views)

Tablename - mydept
Tablename - my employee

Create a view <view-foremployee>
As

```
Select* from myemployee As A
inner join mydept As B
  ON B.id = A.deptid ;
```

select * from <View Name> ;

→ Create 2 table where attr are employee-id, emp-nam
gender, salary city ← Employee

mydept → dept-id, dept-name

then implement inner, outer join

⇒ Row level View

```
select A.*, B.dept_name from myemployee As A
    inner join mydept As B
        on B.id = A.deptid
            WHERE dept_name = 'HR';
```

→ select * from view_employee;

⇒ Column level View

```
create View <view.name>
    As
Select A.emp.id, A.empname, A.gender, A.city
    B.deptName
from my employee As A
inner join My dept As B
    ON  B.id = A.deptid;
```

⇒ Alter View

```
Alter View < View Name>
    As
Select A.*, B.dept-name, from
    myemployee As A
  innerjoin mydept As B
    ON  B.id = A.dept.id
Where deptname='HR' OR dept_Name='Account';
```

→ Drop View < View Name>

⇒ Query Performance Tools

INDEX
CLUSTER
VIEW
SEQUENCE → Syntax
→ Example
→ Insert data in table using sequence
→ display
→ alter
→ drops

⇒ Sequence

CREATE SEQUENCE <sequence_name>
[start with <values>
Increment by <values>
MaxValue <values>
MinValue <values>
cycle / No cycle
cache <integer value>
];

specific starting no
Interval b/w sequence Nio
Repeated Data

pro.id		pro_price

→ created sequence productseq
start with 1
increment by 1
minvalues 1
maxvalue 1000
cycle, cache 5;

* sequence is to generate value of primary key

→ Insert into product values (prod-seq, Nextval 'name', 'cost');
enter the value for name 'mobile'
" " " price : 20000
Sql > select * from product

increment (2 or 3 .....)

Sql> alter sequence prod_seq increment by 2;

Sequence Attended

Sql> select * from product;

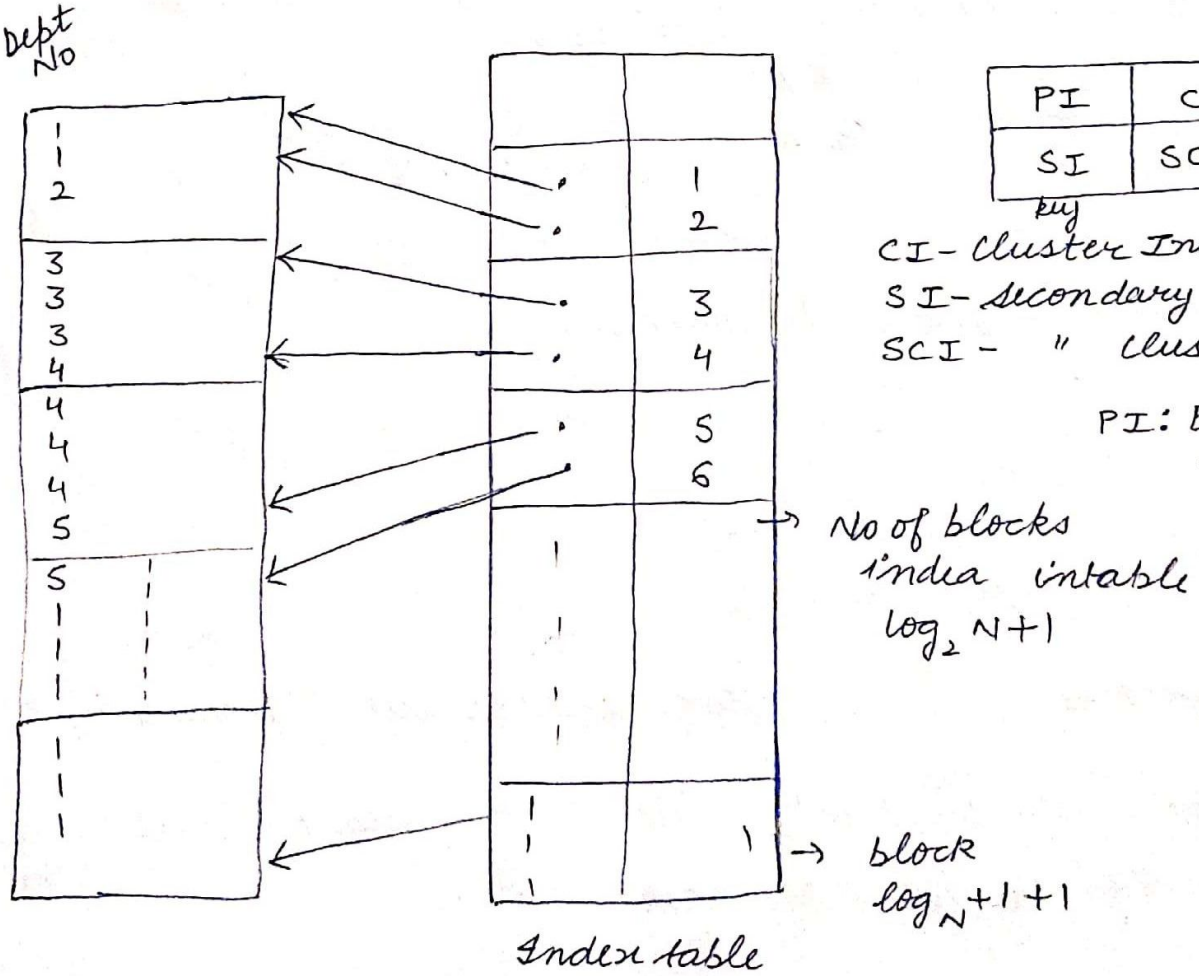
id	Pr.No	Pr.cost

Sql> drop sequence <seq Name>;

19/July/2019

⇒ Sql performance Tools

1. Index
2. View
3. Sequence
4. ~~Index~~ Cluster
   - → def^n and exp.
   - → create cluster
   - → create index on cluster
   - → Create table
   - → Insert data in the table
   - → display



PI	CI
SI	SCI

key

CI - Cluster Index

SI - Secondary "

SCI - " Cluster Index

PI: Primary Index

→ • Cluster is a database object that store data that is related 2 or more table in single disc space

- cluster is a schema object that contain data from one or more table, basically it has 2 types
  1. Index cluster
  2. hash "
- It must contain more then one cluster and all of the table in form of Cluster
- It will also include more than 1 table but it will use

$$H(x) = X \bmod M$$

⇒ Create cluster

```
CREATE CLUSTER C1 (d number(2))
                cluster created
CREATE INDEX C1 on CLUSTER C1 ;
                index created
```

→ Create table

```
CREATE TABLE Dept44 (dNo Number(2), dname varchar(20))
                    cluster C1 (dNo);
                Table Created

CREATE TABLE emp44 (empNo number(4), ename varchar(20)
                    dno Number(2)) CLUSTER C1 (dNo)
                Table Created
```

→ insert Value

```
sql> INSERT INTO dept44 Value (10,'CSE');
sql>    "       "    "    Value (20,'EEE');
Sql> INSERT INTO emp44 Value (1,'A',10);
       "       "    "    "    (2,'B',20);
```

→ display

sql> Select RowID, D.no, D.Name from dept44;

RowID	D.no	D.Name
AABBKK	—	—
BABBKK	—	—

sql> Select RowID, empNo, eName, DNo from emp44;

RowID	empNo	eName	DNo
ABBBKKK			
BABBKK			

22/July/19

→ create a table info system
1. implement the index with
2. implement view in particular column (create table of dept, ——— in both student table and dept ")
3. create the ~~index~~ cluster in student info system

→ what is the sequence . Implement where max value is 200 , min -10 increment by 5 , cache index 30

include(unique index, cluster index, Bitmap index)

Xml

→ Introduction to xml
→ Compare with Html
→ xml example

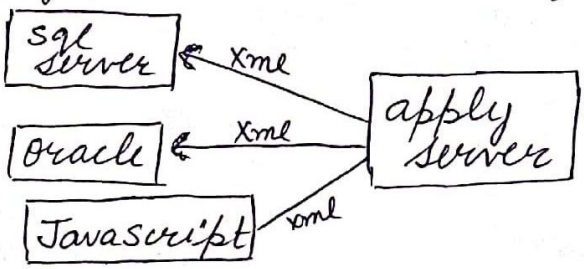
→ feature

- separate data from Html
- simplify data sharing
- simply platform choose

Data —import→ [Xml] —Export→ Java Script

→ Xml Extensable Markup Langauage

- subset of SGML
- store and transport data
- not used for display data
- platform and langauage expenend
- help to communicate b/w platform
- standard generalised Markup langauage → SGML



→ **Imp

Xml	Html
• Used for exchange data or structure of data	• It is use to create structure of web page
• User define tags, It doesn't have predefined tags	• It doesn't have predefined tags
• tags are case sensitive	• Case Insensitive
• Xml is strict langauage	• It is not strict langauage

Example
```
<College>
  <class>
    <Name> ..... </Name>
  </class>
</College>
```

Example
```
<Html>
  <body>
    <P> ..... </P>
  </body>
</html>
```

Example

```
<?xml version="1.0" encoding="UTF-8" ?>
<personaldata>
    <user>
        <fname> ani </fname>
        <lname> singh </lname>
        <email> anisingh@gmail.com </email>
    </user>
    </user>

    :

</personddata>
```

Header

* Browser → localhost/xml tutorial / data.xml

→ Create xml file of college where college is root Element it is having 2 classes, display name and roll no of students (Include)

⇒ Types

Internal DTD

element declared within a xml file

Syntax:

```
<!DOCTYPE root_element
  [element_declaration]>
```

External DTD

element declared outside of xml file

Syntax:

```
<!DOCTYPE root_element
  SYSTEM "file_name">
```

⇒ XML DTD (Xml document Type defⁿ)

- use to describe xml langauage
- use to defⁿ the structure of xml document
- contains list of legal statement
- use to perform the validation

Syntax <! DOCTYPE element DTD identifier
[declaration 1
declaration 2] >

⇒ Internal DTD

```
<? xml version = "1.0" encoding, "UTF-8">
<! DOCTYPE address [<! ELEMENT address (name,
                                company, phone)>
        <! ELEMENT name (#PCDATA)
                          ↑(          character
                                      data)
        <! ELEMENT company (#PCDATA)
        <! ELEMENT phone (#PCDATA)
              :
<Address>      ]>
    <Name> ------ </Name>
    <company> ----- </company
    <Phone> --- -- </phone>
....  </Address
```

all Imp 23/07/19

→ Xml schema → def^n example

→ Types → simple
→ complex

→ difference b/w Xml schema and EDTD

→ Xml / Name space and firefox Exp

Xml / Schema

→ known as XS of ("Xml schema def^n)
→ describe and Validate the structure of Xml data
→ It is like DTD but provide more control on Xml structure

→ ~~It is~~ It is a method of expression constants about Xml document.

→ Types → complex
→ simple

Xml/NS (Xml Namespace)

- provide unique name of data
- It is set of uniqueness
- Identified by URI (Uniform resource Identifier)
- must be start "xml/ns"

Syntax: <element xml/ns 'name = "URI" >
attribute name ∈ URI

Conflict - generally, conflict occur when we try to more than 1 xml document from diff Xml applications

→ difference b/w DTD and Xml schema

DTD	Xml schema (XSD)
• document type defn	• Xml schema defn
• doesn't support datatype	• support
• doesn't support namespace	• support
• not define order of child element	• order define
• not extensible	• extensible

→ <! Doctype Address>
[! Element Address (name)
! Element name (#PCDATA)
]>

→ <XS element name = "address">
<XS: complex type>
<XS : sequence>
<XS : element name = "name" type = "XS . string">
</XS : sequence>
</XS : complex type>
</XS : element>
</XS : schema>

XML simple type → use only content of text
Ex XS : int , XS : string

Syntax: <XS : element name = " phone" type" XS : int" />

Complex type

Add XSD as extension

```
<? XML Version = "1.0" encoding = " UTF-8" ? >
<XS: Schema XM/ns : XS = " schema">
< XS : element name = " address" >
< XS : complex type >
< XS : sequence >
<XS : element name = " name" type = " XS : string" />
<XS : element name = " phone" type = "XS : int" />
</XS : sequence >
</XS : complex type >
</XS : element >
</XS : schema >
```

→ Add xml

```
<? XML version = "1.0", encoding = "UTF-8" ? >
< address >
< xsd : schemalocation = " path---- add.xsd />
   < name > aman </name >
   < phone > ------ </phone >
</address >
```

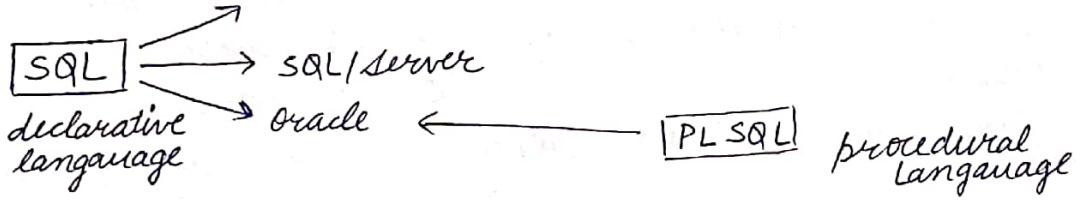
Namespace

file : 1.xml

```
<? xml version = "1.0" encoding = "UTF-8" >
< c1 : class xml xs : c1 = " class1 ----
prefix < c1 : Name > xyz </ c1 : name >
                    </ c1 : class >
```

24/7/19

PL/SQL (Lab)



SQL	PLSQL
• create table and manage data	• condition
• no condition, no loop	• loops
•	• Exception handling
• no security	• security

→ PL/SQL block

PL/SQL pgm. → PL/SQL engine → client/server

SQL statement → SQL engine → DB engine

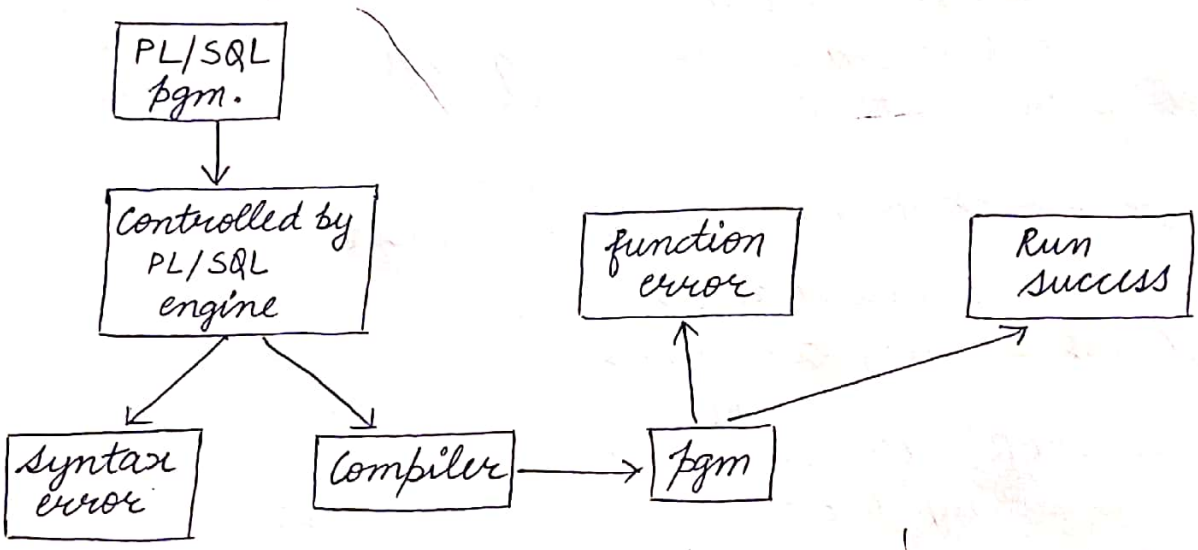
→ program structure

declarative section (optional)

pgm. logic — Begin / End (mandatory)

Exception handling — Exceptional logic (optional)

→ processing



lexical analysis
syntax analysis
semantics analysis
} H/W oriented

↓ code

Code optimization
Target code
} S/W oriented

```
declare
{
BEGIN
{
  <program>
  Function
}
  }
END
```

PL/SQL is used in oracle and PL/SQL engine processed the PL/SQL statement. It can be stored client-system. It will support the conditional statement (control structure)

→ DECLARE

```
message varchar 2 (20): = Hello World ;
BEGIN
  dbms_output . put_line (message)
END ;
```

practice

→

- Create cluster
- Create index of cluster
- insert data in cluster using index
- drop the clusters

→ Create cluster <clusterName> (d number (2));
Create index <indexName>
ON cluster <clusterName>

PLSQL

→ DROP CLUSTER <cluster name>
including table . cascade constraints;

→ DECLARE

```
V-num1 Number := 5;
V-num2 NUMBER :=3;
V-temp NUMBER ;
```

* more secure then SQL

BEGIN

```
IF  V-num1 > V-num2  THEN
    V- temp := V-num1
    V-num1 := V-num2
    V-num2 := V-temp ;
```

```
DBMS_OUTPUT. PUT-LINE ('V-num1 = ' || V-num1);
DBMS-OUTPUT . PUT-LINE ('V-num2 = ' || V-num2);
END;
```

O/P : V-num1 = 3
V-num2 = 5

25/Jul/19 Test

Ques: Create table of University Exam. control System having 2 table

1. Student sitting arrangement (student rollno, sitting plan (Room no), paper code)
2. Faculty data (Faculty Name, Id, Exam-duty (Room no))

ques: what is a View, Create a view in column label

ques: Using row id, delete the duplicate row in faculty data table

ques: View the student rollno and faculty id

ques: 1 Create the view ~~in~~ more then 1 col in any table.

2 Create a sequence (alt. primary key) of product data upto 10000

26/Jul/19

View

- single column View
- composite view
- join more then one table and view in column level
- Row level View

DML

⇒ CREATE VIEW <View-Name>
As
SELECT A.*, B.designation
FROM <tableName> As A
INNER JOIN <tableName2> As B
ON A.emp_id = B.emp_id
WHERE designation = 'SM';

→ Select * From

31/July/2019

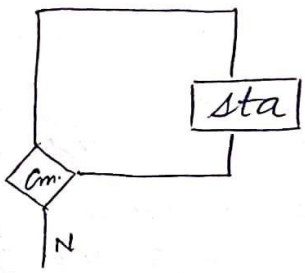
Control statements

Iteration - It indicate the repeat particular block of code
In generally loop and Endloop keyword commonly used
loop is placed before first statement and endloop
placed after last statement in sequence.

Types

→ Simple loop

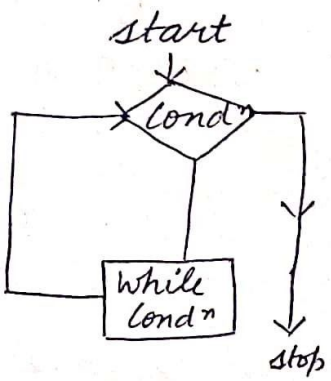
Syntax
```
LOOP
<execution of code>
<exit cond^n>
END LOOP;
```




```
DECLARE
        x number:=10;
BEGIN
        loop
        dbms ------ (x);
         if x>50 then ← [x:=x+10;]
          exit;
        END IF;
        END LOOP;
        dbm ---------- (after exit x is:||x);
         END;
```

→ While loop

```
DECLARE
x number(2):=10;
BEGIN
   while a<20  LOOP
    dbm ------ ('Value of a:||a);
     a:=a+1;
     END LOOP;
    END;
```



→ For loop

```
[lable-name] FOR datatype
        IN [Reverse] lower_value .. upper_value
    seq of statements
    END loop [Lable name]
```

→
```
BEGIN
  FOR no IN 1..5 loop
    DBMS_____('iteration :'||no);
  END LOOP;
  END;
```

→
```
DECLARE
   i number(1);
   j number(1);
BEGIN
for i IN 1..3 loop
for j IN 1..3 loop
dbms_____('i is :'||i);
dbms_____('j is :'||j);
 END loop; inner loop
 END loop; outer loop
 END;
```
don't write

→
```
DECLARE
a number(2):=10;
BEGIN
while a<20 LOOP
 dbms_____('value of a is :'||a);
   a:=a+1;
 if a=15 then
   a:=a+1;
   continue;
 ENDIF;
 END LOOP;
 END;
```

ques: How to declare variable in PL/SQL Explain with airthmetic operations.

ques: Out of 5 variables data find the greater value and sort in order.

ques: pallendrome no

ques: F = ma

ques: WAP to find no of vowel and consonents in string

ques: Factorial of a no using while loop

ques: print pattern using loop

```
*
* *
* * *
* * * *
* * * * *
```

ques: Excellent, verygood, good, fair, poor
find grade of student

1/August/2019

Xml (extensiable Markup Language)

* diff b/w Xml and Html

- header file of Xml <? xml version = "1.0" encoding = "utf-8" ?>
- DTD (document type defn)
- both xml and DTD are same, only writing pattern is diff.

6/Aug/2019 (Lab)

Exception Handling

7/Aug/2019

⇒ Cursor

a Cursor is used to process individual row returned by database system for the query (Cursor hold the row returned by SQL)

Cursor is defined as ~~Unbuffered~~ work area in RAM
It has 2 types

- Implicit (predefined) → associated with DML, no-need of declaration
- Explicit (userdefined) → user defined, SELECT
  (declaration section are used to execute the system)

Implicit	Explicit
• Command:	• Command:
SQL %. Found <T F	OPEN }
SQL %. Not Found <T F	FETCH } lycle
SQL %. Row Count → (allRow)	EXECUTE }
SQL %. open → open data	CLOSE }

- SQL%.FOUND It is a boolean that will return True and False. True will return when the most recent DML affects 1 or more row
- Not Found return when the most recent DML does not affect 1 or more Row
- Row Count return the no of row in DML

→ NOT FOUND

```
Begin
Update emp1
SET Sal = Sal + 1000;
If SQL%.NOTFOUND THEN
    dbms ___ ('employee.SET');
elsif
     sql%. Found then
          dbms --- ('selected')
end if
end
```

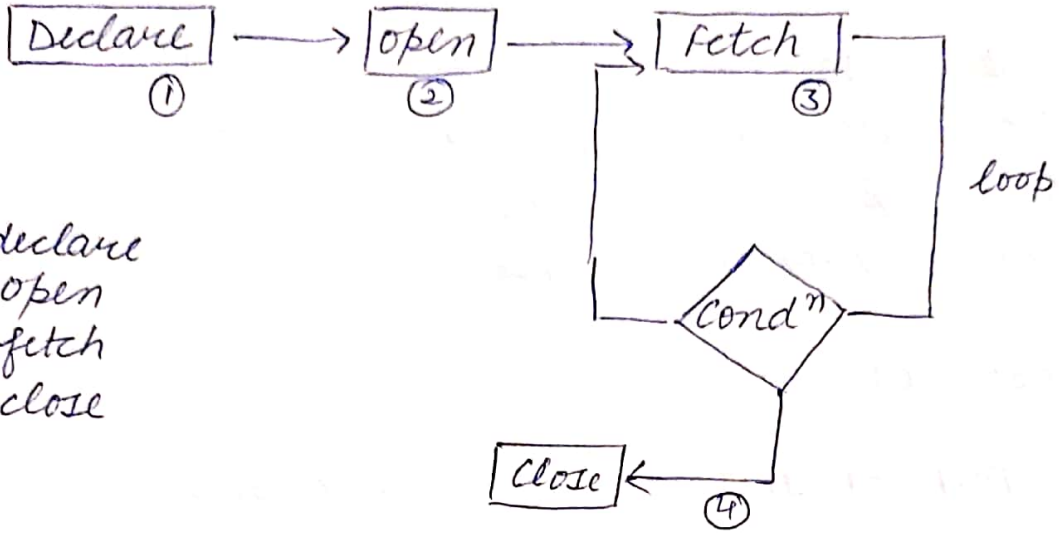
→ UPDATE

```
Begin
      UPDATE  S
      SET status = 40
      WHERE status = 30;

DBMS ___ ('number of updates')
    SQL%. ROWCOUNT ;
END;
```

→ DELETE

```
VARIABLE rows_deleted  VARCHAR2 (30)
DECLARE
    empno employees employee_id %. TYPE := 176 ;
BEGIN
      DELETE FROM employees
         WHERE employee_id = empno ;
        : rows_deleted ;= (SQL%. ROWCOUNT || row deleted.');

END;
/
 PRINT rows_deleted
```

EXPLICIT



- declare
- open
- fetch
- close

① **Declare**

Cursor < Cursor name > ← any suitable name

IS
SELECT
st1
st2
!
!

② **open** open < Cursor Name >

③ **Fetch** Fetch < Cursor Name >

④ **close**

close < Cursor name >

⇒ Fetch employee_name , emp_no from employee table using Cursor

```
DECLARE
   Cursor C1 IS SELECT
     empno, empname From emp ;
   V-empno   emp.empno %Type ;
    V-ename  emp.ename %Type ;
   Begin
        OPEN  C1;
      loop
        FETCH  c1  INTO  V-empno , V- ename
   Exit  When C1 %NOT FOUND ;
     dbms ——— (V-empno || '|| V-ename ) ;
   End loop ;
      Close C1 ;
   END ;
```

⇒ Declare

```
  Vname varchar (20)
Cursor C1 is select
  ename from emp
     where Di'd = 20 ;
  Begin
      Open c1 ;
  loop
      fetch C1 into Vname ;
        ———————
          - ——— Statement ;
      dbms ------ (Vname ) ;
  end loop ;
  end ;
```

c1'd	ename	D-i'd
11	AA	20
22	BB	10

9/August/2019

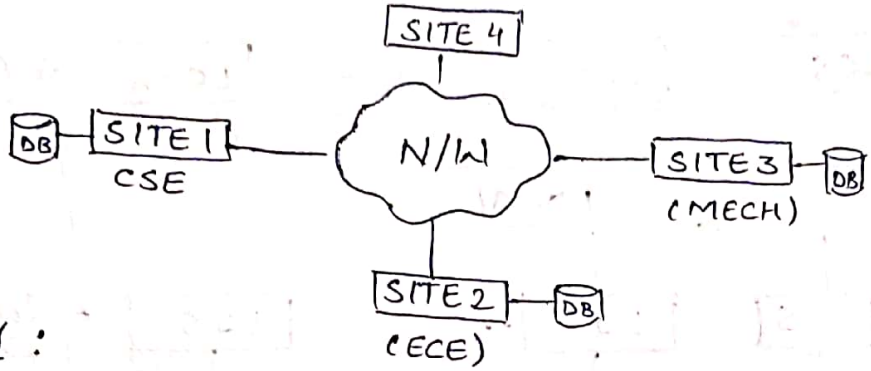
(31)

(27)

⇒ DDBMS (distributed DBMS)

- collection of multiple logical interconnected DB

DBMS / ADBMS / RDBMS / DDBMS



Features :

- ~~DDBMS~~
- stores data in number of sites
- sites are interconnected by the system
- Improve data durability and reliability

disad :

- Reduce operating cost
- data complexity and management, controling data
- deadlock
- security of database

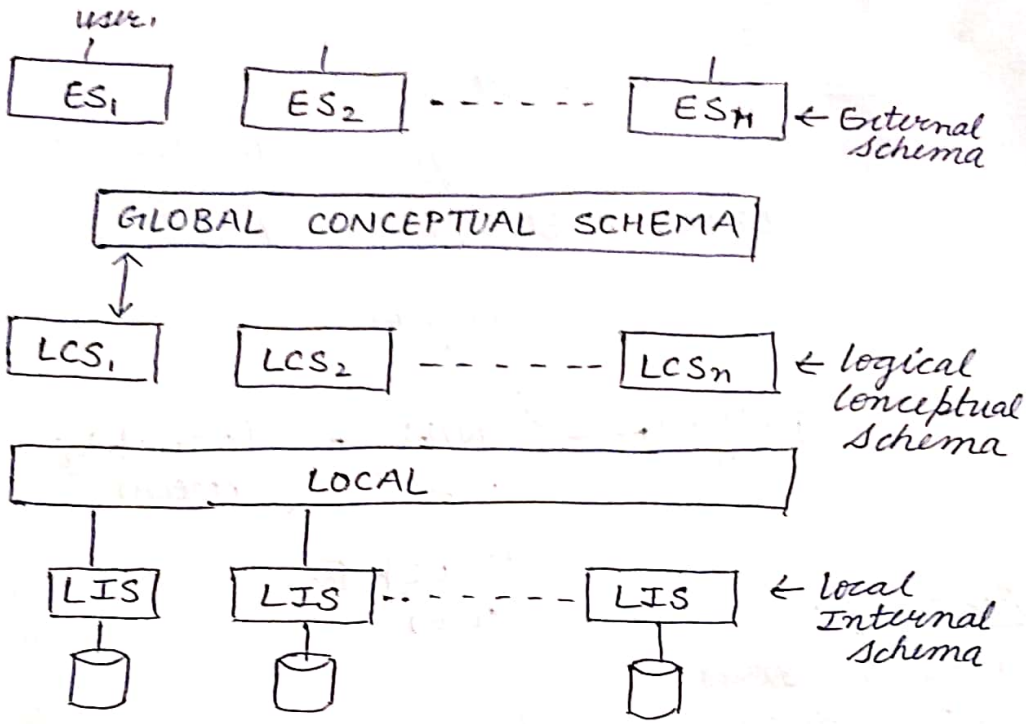
SM

⇒ DDBMS Architecture

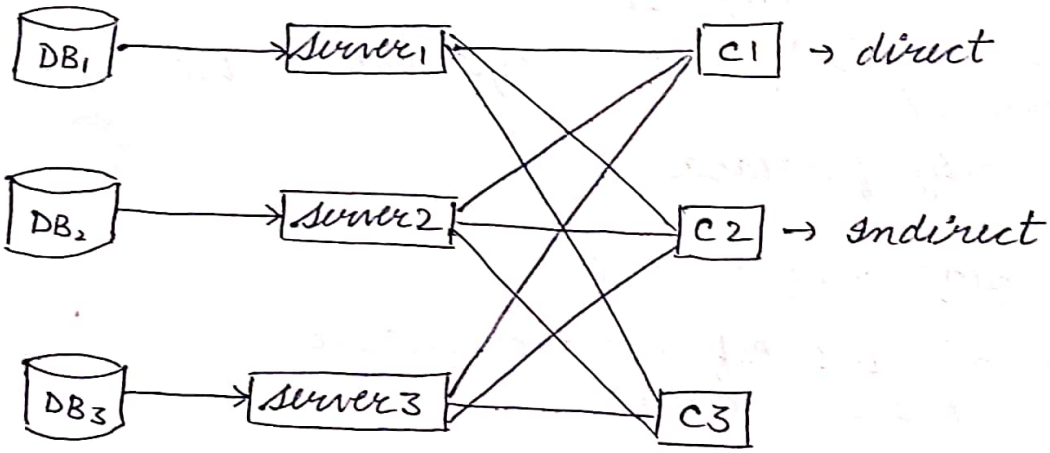
- distributed Refrence architecture
- client server arch.
- peer to peer arch.

ES1 ES2 - - - - ES4

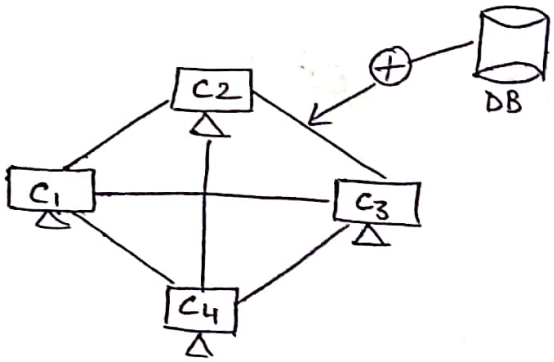
* Explain It



⇒ Client-Server Architecture

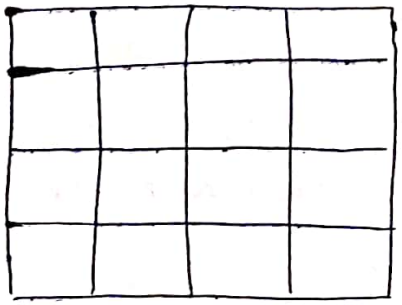


⇒ Peer to Peer



⇒ ⇒ Data Fragmentation

Horizontal Vertical



Horizontal fragmentation

$R = R_1 \bowtie R_2$

Vertical fragmentation

$R = R_1 \cup R_2$

⇒ Data Replication

fully replicated data (all sites)

partial Replicated (some sites)

⇒ Distributed Query processing -

13/August/2019

DBMS Lab

Stored function : Example

- Create function

```
CREATE OR REPLACE FUNCTION get-sal
  (id employees.employee_id %TYPE) RETURN NUMBER IS
    sal employees.salary %TYPE := 0;
BEGIN
    SELECT SALARY
    INTO sal
     FROM employees
     WHERE employee_id = id;
     RETURN sal;
END get_sal;
/
```

⇒ Find Maximum Number

```
Declare
   a number;
   b number;
   c number;
FUNCTION Findmax (x IN number, y in number)
 RETURN number
  IS
  z number;
 Begin
 If x>y then
 z:=x;
 else
 z:=y;
 endIf;
 Return z;
 END;
```

calling part

```
Begin
   a:=23;
   b:=45;
   c:= findmax(a,b)
  dbms_('max of (23,45)'||c);
  END;
```

⇒ **IS**

- It is a function which returns a single value that stored in a database.
- It will store func^n value which is created by the Create function

⇒ factorial with Recursion

```
Declare
   num number;
   factorial number;
    FUNCTION fact (x number)
        RETURN number
          IS
        f number;

Begin
     If x=0 then
        f:=1;
     else
       f:=x*fact(x-1)
endIf;
      RETURN f;
      END;

Begin
     num:=6
     factorial:=fact(num);
dbms.output('factorial'||
 num||'is'||factorial);
END;
```

Total Cost

* Create Table Before

⇒

```
Create [or Replace] function
     total-cost
  Return number
      IS
  total number(2):=0;
BEGIN
  Select Count(*) INTO total
    FROM Customer
    Return Total;
  END;
DECLARE
  c number(2);
BEGIN
   c:=total-cust()
  dbms-('total Cust'||c);
END;
```

⇒ CREATE [OR REPLACE] . function Salary-find (eno emp.empno %. type)
Return number ;
IS
V-sal emp.sal %. type
Begin
select sal into V-sal from emp
where empno = eno ;
Return V-sal ;
END ;
Declare
V-sal emp.sal %. type ;
Begin V-sal = salary-find (10) ;
dbms — (V-sal)

* Table Before (create)

e-name	Name	Sal
—	—	—

14/August/2019

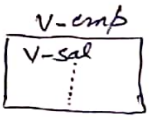
DBMS Lab

Parametrised Cursor

```
Declare
 Cursor C1 IS
 SELECT ename, sal
 FROM   emp
 WHERE  depno = 10;
Begin
  for rec  in C1 loop
dbms — ('emp:' || rec.name ||
        rec.sal) ;
 END LOOP ;
```

- Exception
  - %. open
  - %. Found
  - %. not Found
  - %. Row comp.
- Cursor
  - Implicit
  - Explicit
- Function
  - UDF
  - declare
  - open
  - fetch
  - close

⇒ CURSOR <cursor-name> (parameter datatype := value)
IS
SELECT &
ST-1
ST-2
ST-3
⋮
ST-n
Begin
open cursor-name
Fetch ———
close ———
END ;



⇒ DECLARE
V-emp number ; ← cursor name
CURSOR C-emp (V-salary number := 5000)
IS
SELECT salary FROM employee
WHERE salary < V-Salary
ORDERBY Salary DESC ;
BEGIN
open C-emp ;
loop
fetch C-emp INTO V-emp ;
dbms — (V-emp) ;
exit When C-emp %NOT FOUND ;
END LOOP ;
CLOSE C-emp ;
END ;

⇒ Declare

```
Cursor C1 (V-deptno NUMBER)
        IS
Select ename, sal
 FROM emp
 WHERE deptNO = V-deptNO;
Begin
   For rec in C1(10) loop
   DBMS — ('emp:' || rec.name || rec.sal);
END loop;
END;
```

⇒ New

```
Declare
    Cursor C-dept
     IS
    SELECT distinct dept no
      from dept;

CURSOR
      C1 (V-deptno Number) IS
      SELECT ename, sal
      FROM emp
      WHERE deptNO = V-deptNO;

BEGIN
     for dept-rec IN L-dept, LOOP
      dbms ('dept NO:.....' || dept-rec, dept_no);
```

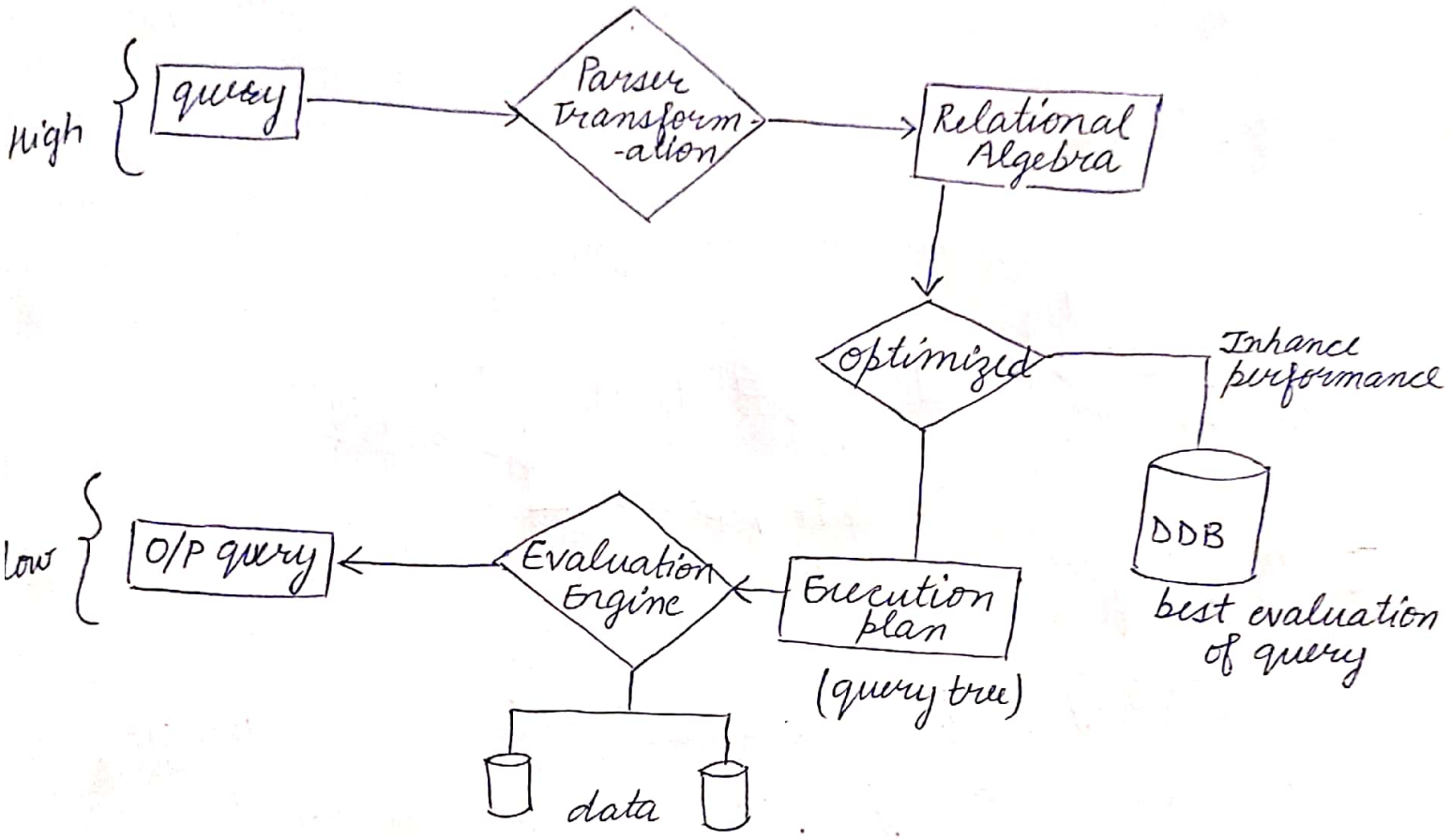
Inner Query

```
for rec IN C1 (dept-rec.dept no) loop
  dbms — ('emp:' || rec ename || 'has salary of
          || rec.sal);
```

20/August/2019

DBMS Lab

Query processing and evaluation



⇒ select book, title, price from book where price > 30

$\sigma_{price>30}(\pi_{books,title,price}(Book))$

$\sigma_{price>30}$
↑
$\pi_{book,title,price}$
↑
Book

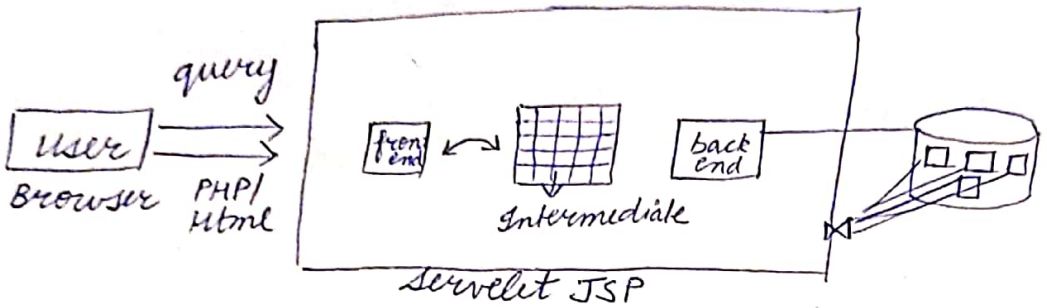
$\pi_{book,title,price}(\sigma_{price>30}(Book))$

$\pi_{book,title,price}$
↑
$\sigma_{price>30}$
↑
Book

Ques: What is parameterised cursor. Create a table of employee info information system. Insert atleast 5 data items. Find the

1. employee salary using parameterised cursor with Loop
2. display the name and dept. of each employee (employee-name)

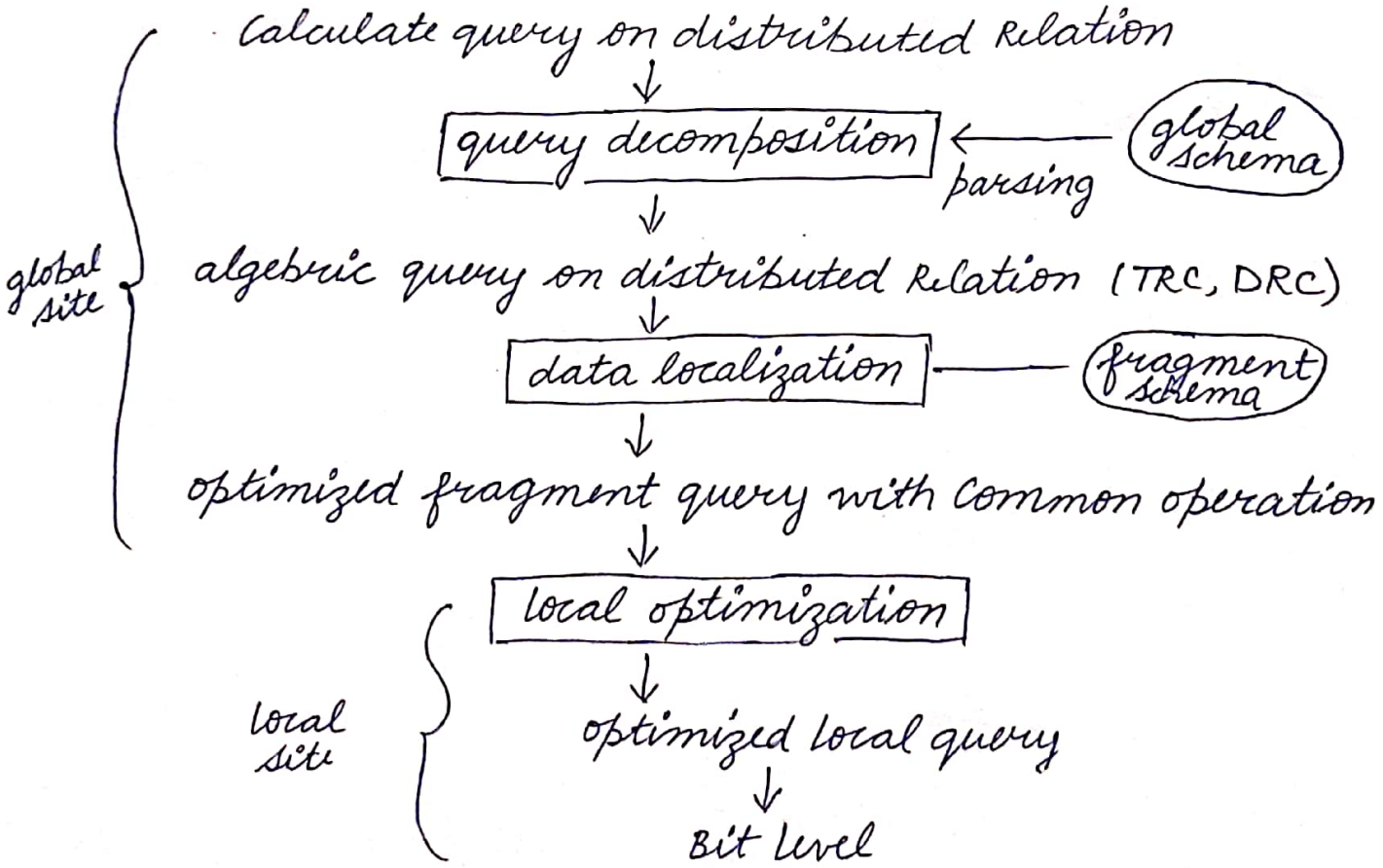
21/August/2019



→ Communication cost
→ query performance

⇒ How query performance is done ?

DISTRIBUTED DBMS



⇒ Query processing

- It include transformation of high level to low level query expression that can be used to physical level of file system. The query optimization is the actual execution of query which we get result.

⇒ Query optimization

It is a process in which multiple execution query plan for satisfying a query one or more examine and a most efficient query.

→ simple join
→ semi join
→ parallel join

] which is better ?

⇒ $\pi_{ISBN,\ price,\ pid,\ pname}(Book \bowtie publisher)$

(15) Byte (6) (4) (50) — S_1 — S_2

15+6+4+50 = 75 bytes

Book: 500 tuples ↑ 100 byte/tuple

publisher: 150 tuples ↑ 150 bytes/tuple

500 tuples in Book each tuple is of 100 byte / tuple
100 —— publisher —— 150 byte / tuple

S_1 → Book = 5000 X100 = 50000 bytes
S_2 → publisher = 100 X150 = 15000 bytes

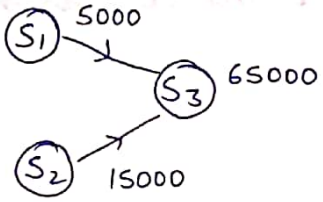
→ After join operation

500 tuples each tuple is of 75 bytes

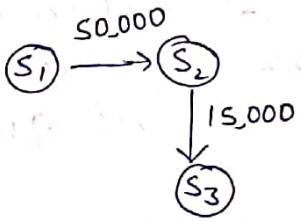
500 X 75 = 37500 bytes
(Max) bytes
in
natural join
Max 500
tuples

min → 100 X 75 = 7500

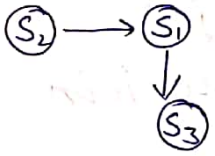
(1)



(2)

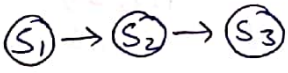


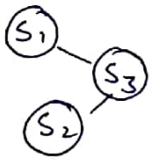
(3)



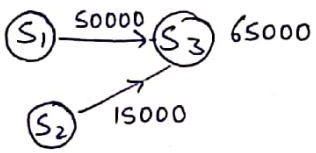
Before join $S_1 \rightarrow 50,000$
$S_2 \rightarrow 15000$

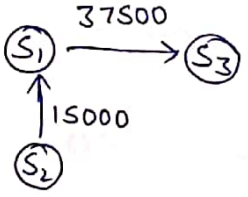
after join $S_1 \rightarrow 500 \times 75 = 37500$
$S_2 \rightarrow 100 \times 75 = 7500$



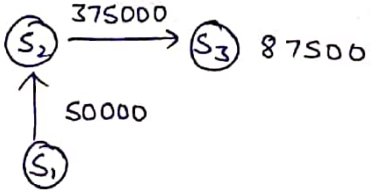


→





This is best



3 September 2019

Procedure

syntax

```
CREATE OR [REPLACE] PROCEDURE
    <procedure-name>                    [parameter
    {IS/AS}                              IN, OUT, INOUT]

    declaration section
    Begin
        Exception
            end <procedure-name>
```

IN (By detail / Pass)	OUT (send)	INOUT (Reuse)
Calling program		Procedure
• By default	• specify	• Both
• Values are pass to sub prog	• Value return	• Both
• actual parameter	• formal	• Both
• Constant	• return	• Both

→ Steps :-

1. Create procedure
2. Call procedure
3. Print
4. drop

→ CREATE OR [REPLACE] PROCEDURE

```
max [x in number
     y in number
     z out number]
```

```
IS
Begin
  if x<y then
    z:=x;
  else
    z:=y;
  end max;
```

```
Declare
    a number;
    b number;
    c number;
Begin
    a:=35;
    b:= 45;
    max(a,b,c);
  dbms_('max':||c);
  end;
```

⇒ Procedure

It is used to repeat an execution. Procedure is already stored in PL/SQL program i.e called the stored procedure. It may or not return or more value, but func" is not like that. It must return a value. It can pass one or more value at a time

Procedure	Function
• It is used to mainly execute the certain process.	• It will perform the particul-ar calculations.
• It can't call using select op.	• It can be called by select op.
• Use out parameters for retur-ning value	• Use return keyword for returning the value
• not mandatory to return the value	• mandatory to return the value

⇒

```
CREATE OR REPLACE PROCEDURE Salary
    (e in number, amt IN number , s out number)
   IS
  BEGIN
    update emp set sal = sal+amt
    where empno = 10;
                  (E)
    COMMIT;
    END ;Salary;

Declare
    K number
Begin
    Salary (7449, 2500, K);
    dbms — (K);
    end;
select * from emp ;
```