|  |
| --- |
|  |
|  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| REVISION HISTORY | | | | | |
| Ver. | Description of Change | Author | Date | Approved | |
| Name | Effective Date |
| 1.0 | Initial status | [Kiryl Bucha](mailto:Kiryl_Bucha@epam.com) | 12-JAN-2012 |  |  |
| 2.0 | Updated in accordance with renewed content | [Elias Nema](mailto:Elias_Nema@epam.com) | 20-JAN-2014 |  |  |

Contents (if needed)

[1. Prerequisite Task 3](#_Toc425446255)

[2. Row Migration 3](#_Toc425446256)

[2.1. Task 3](#_Toc425446257)

[2.2. Results 3](#_Toc425446258)

[3. Heap Organized Tables 3](#_Toc425446259)

[3.1. Task 1 – Heap Understanding 3](#_Toc425446260)

[3.2. Task Results: 3](#_Toc425446261)

[3.3. Task 2 – Understanding Heap Table Segments 4](#_Toc425446262)

[3.4. Task Results: 4](#_Toc425446263)

[4. Index Organized Tables 4](#_Toc425446264)

[4.1. Task 3: Compare performance 4](#_Toc425446265)

[4.2. Task Results: 6](#_Toc425446266)

[5. Index Clustered Tables 6](#_Toc425446267)

[5.1. Task 4: Cluster Storage by Blocks 6](#_Toc425446268)

[5.2. Task Results: 7](#_Toc425446269)

[6. Hash Clustered Tables 7](#_Toc425446270)

[6.1. Task 5: Analyses Cluster Storage by Blocks 7](#_Toc425446271)

# Prerequisite Task

Connect as system to pluggable database pdb. Create a new tablespace tbs\_pdb\_test with the new datafile pdb\_test\_001.dbf:

CREATE TABLESPACE TBS\_PDB\_TEST

DATAFILE

'PDB\_TEST\_001.DBF' SIZE 104857600 AUTOEXTEND ON NEXT 20971520

NOLOGGING

SEGMENT SPACE MANAGEMENT AUTO

EXTENT MANAGEMENT LOCAL AUTOALLOCATE;

Create a new user:

create user $username$ identified by $pwd$ default tablespace tbs\_pdb\_test;

Grant Connect Role and Resource Role:

grant connect to $username$;

grant resource to $username$;

grant dba to $username$;

grant select on scott.dept to $username$;

grant select on scott.emp to $username$;

# Heap Organized Tables

## Task 1 – Heap Understanding

Step 1:

create table t

( a int,

b varchar2(4000) default rpad('\*',4000,'\*'),

c varchar2(3000) default rpad('\*',3000,'\*')

)

/

Step 2:

insert into t (a) values ( 1);

insert into t (a) values ( 2);

insert into t (a) values ( 3);

commit;

delete from t where a = 2 ;

commit;

insert into t (a) values ( 4);

commit;

Step 3:

select a from t;

## Task Results:

Add your personal description of what happened and why with screenshots where needed.

**NOTE:** Adjust columns B and C to be appropriate for your block size if you would like to reproduce this. For example, if you have a 2KB block size, you do not need column C, and column B should be a VARCHAR2(1500) with a default of 1,500 asterisks. Since data is managed in a heap in a table like this, as space becomes available, it will be reused.

To find what is you block size you can use:

show parameter block\_size;

select distinct bytes/blocks from user\_segments;

SELECT name,value

FROM v$parameter

WHERE name = 'db\_block\_size';

## Task 2 – Understanding Heap Table Segments

Step 1:

create table t ( x int primary key, y clob, z blob );

Step 2:

select segment\_name, segment\_type from user\_segments;

Step 3:

Create table t (

x int primary key,

y clob,

z blob

)

SEGMENT CREATION IMMEDIATE

/

Step 4:

# select segment\_name, segment\_type 2 from user\_segments;

Step 5:

# SELECT DBMS\_METADATA.GET\_DDL('TABLE','T') FROM dual

## Task Results:

Add your personal description of what happened and why with screenshots where needed.

# Index Organized Tables

## Task 3: Compare performance

Step 1:

CREATE TABLE emp AS

SELECT

object\_id empno

, object\_name ename

, created hiredate

, owner job

FROM

all\_objects

/

Create Index:

alter table emp add constraint emp\_pk primary key(empno)

Calculate Statistic:

begin

dbms\_stats.gather\_table\_stats( user, 'EMP', cascade=>true );

end;

Step 2:

CREATE TABLE heap\_addresses

(

empno REFERENCES emp(empno) ON DELETE CASCADE

, addr\_type VARCHAR2(10)

, street VARCHAR2(20)

, city VARCHAR2(20)

, state VARCHAR2(2)

, zip NUMBER

, PRIMARY KEY (empno,addr\_type)

)

/

Step 3:

CREATE TABLE iot\_addresses

(

empno REFERENCES emp(empno) ON DELETE CASCADE

, addr\_type VARCHAR2(10)

, street VARCHAR2(20)

, city VARCHAR2(20)

, state VARCHAR2(2)

, zip NUMBER

, PRIMARY KEY (empno,addr\_type)

)

ORGANIZATION INDEX

/

Step 4: Initial inserts:

INSERT INTO heap\_addresses

SELECT empno, 'WORK' , '123 main street' , 'Washington' , 'DC' , 20123 FROM emp;

INSERT INTO iot\_addresses

SELECT empno , 'WORK' , '123 main street' , 'Washington' , 'DC' , 20123 FROM emp;

--

INSERT INTO heap\_addresses

SELECT empno, 'HOME' , '123 main street' , 'Washington' , 'DC' , 20123 FROM emp;

INSERT INTO iot\_addresses

SELECT empno, 'HOME' , '123 main street' , 'Washington' , 'DC' , 20123 FROM emp;

--

INSERT INTO heap\_addresses

SELECT empno, 'PREV' , '123 main street' , 'Washington' , 'DC' , 20123 FROM emp;

INSERT INTO iot\_addresses

SELECT empno, 'PREV' , '123 main street' , 'Washington' , 'DC' , 20123 FROM emp;

--

INSERT INTO heap\_addresses

SELECT empno, 'SCHOOL' , '123 main street' , 'Washington' , 'DC' , 20123 FROM emp;

INSERT INTO iot\_addresses

SELECT empno, 'SCHOOL' , '123 main street' , 'Washington' , 'DC' , 20123 FROM emp;

Commit;

Step 5: Calculate statistic:

exec dbms\_stats.gather\_table\_stats( $username$, 'HEAP\_ADDRESSES' );

exec dbms\_stats.gather\_table\_stats( $username$, 'IOT\_ADDRESSES' );

Step 6: Compare Trace and Performance:

Explain 1:

SELECT \*

FROM emp ,

heap\_addresses

WHERE emp.empno = heap\_addresses.empno

AND emp.empno = 42;

Explain 2:

SELECT \*

FROM emp ,

iot\_addresses

WHERE emp.empno = iot\_addresses.empno

AND emp.empno = 42;

Compare results and explain Cost value calculation and difference on execution plan.

## Task Results:

Add your personal description of what happened and why with screenshots where needed.

# Index Clustered Tables

## Task 4: Cluster Storage by Blocks

Step 1:

CREATE cluster emp\_dept\_cluster( deptno NUMBER( 2 ) )  
 SIZE 1024   
STORAGE( INITIAL 100K NEXT 50K );

Step 2:

CREATE INDEX idxcl\_emp\_dept on cluster emp\_dept\_cluster;

Step 3:

CREATE TABLE dept

(

deptno NUMBER( 2 ) PRIMARY KEY

, dname VARCHAR2( 14 )

, loc VARCHAR2( 13 )

)

cluster emp\_dept\_cluster ( deptno ) ;

CREATE TABLE emp

(

empno NUMBER PRIMARY KEY

, ename VARCHAR2( 10 )

, job VARCHAR2( 9 )

, mgr NUMBER

, hiredate DATE

, sal NUMBER

, comm NUMBER

, deptno NUMBER( 2 ) REFERENCES dept( deptno )

)

cluster emp\_dept\_cluster ( deptno ) ;

Step 4:

INSERT INTO dept( deptno , dname , loc)

SELECT deptno , dname , loc

FROM scott.dept;

commit;

INSERT INTO emp ( empno, ename, job, mgr, hiredate, sal, comm, deptno )

SELECT rownum, ename, job, mgr, hiredate, sal, comm, deptno

FROM scott.emp

commit;

Step 5:

SELECT \*

FROM

(

SELECT dept\_blk, emp\_blk, CASE WHEN dept\_blk <> emp\_blk THEN '\*' END flag, deptno

FROM

(

SELECT dbms\_rowid.rowid\_block\_number( dept.rowid ) dept\_blk, dbms\_rowid.rowid\_block\_number( emp.rowid ) emp\_blk, dept.deptno

FROM emp , dept

WHERE emp.deptno = dept.deptno

)

)

ORDER BY deptno

Step 6: Drop all tables;

## Task Results:

Add your personal description of what happened and why with screenshots where needed.

# Hash Clustered Tables

## Task 5: Analyses Cluster Storage by Blocks

Repeat all the same steps for the Hash Cluster.

# Row Migration\*

## Task

Create two examples: one with row migration, another with row chaining.

## Results

Add your personal description of what happened and why with screenshots where needed.