Oracle® Tutor™

Section 3 数据管理（Data Management）

1.Manage Materialized Views to improve rewrite and refresh performance

物化视图可以分成：复杂计算类和高级复制类。

create materialized view mv1 refresh fast start with sysdate next sysdate+1/24/60 as select id,name from t group by id;

如果一但包含分组、聚集，物化视图日志必须有 ROWID。

如果有最大值、最小值，无论如何都不会快速刷新：

drop materialized view mv1;

CREATE MATERIALIZED VIEW mv1

BUILD IMMEDIATE

REFRESH fast on commit

ENABLE QUERY REWRITE

AS

SELECT a.cc,max(a.id2),count(*),count(a.id2),count(a.id1) from t1 a,t2 b where a.id1=b.id2 group by a.cc;

select dbms_metadata.get_ddl('MATERIALIZED_VIEW','MYCLOB1','U1') from dual;

2.Configure and manage distributed materialized views

networklink

3.Create and Manage encrypted tablespaces

tde 可能不考

4.Manage Transport of tablespaces across platforms

RMAN> convert datafile '/home/oracle/bigendian01.dbf' from platform 'Solaris[tm] OE (64-bit)' format '/u01/app/oracle/oradata/orcl/bigendian01.dbf'

SQL> conn / as sysdba
Connected.
SQL> select tablespace_name from dba_tablespaces;

TABLESPACE_NAME

SYSTEM
SYSAUX
UNDOTBS1
TEMP
USERS
EXAMPLE
BIGENDIAN

附上：

有限改字符集的方法
shutdown immediate
startup mount
ALTER SYSTEM ENABLE RESTRICTED SESSION;
alter database open;
ALTER DATABASE CHARACTER SET WE8ISO8859P1;
（此命令查 ALTER DATABAE 命令，搜索 chara，查第 4 个 chara 既是。
ORA-12712: new character set must be a superset of old character set

5.Configure a schema to support a star transformation query

Note: Oracle does not recommend setting CURSOR_SHARING to FORCE in a DSS environment or if you are using complex queries. Also, star transformation is not supported with CURSOR_SHARING set to either SIMILAR or FORCE. For more information, see the "Enabling Query Optimizer Features" on page 11-5.

alter system set cursor_sharing=exact;
alter system set star_transformation_enabled=TRUE;
TRUE：Oracle 优化器自动识别语句中的事实表和约束维度表并进行星型转换。这一切优化尝试都在 CBO 的范畴内，优化器需要确定转换后的执行计划成本要低于不转换的执行计划；同时优化器还会尝试利用物化的临时表，如果那样真的好的话。

物化的临时表不启用的 TRUE:
TEMP_DISABLE: 当一个维度表超过 100 个块时，如果简单地设置 **star_transformation_enabled** 为 TRUE 来启用星型变换，那么会话会创建一个内存中的全局临时表(global temporary table)来保存已过滤的维度数据，这在过去会造成很多问题；这里说的 100 个块其实是隐式参数 **_temp_tran_block_threshold**(number of blocks for a dimension before we temp transform)的默认值，此外隐式参数 **_temp_tran_cache**(determines if temp table is created with cache option，默认为 TRUE)决定了这类临时表是否被缓存住；为了避免创建全局临时表可能带来的问题，就可以用到 TEMP_DISABLE 这个禁用临时 表的选项，让优化器不再考虑使用物化的临时表。
select * from x$ksppi ;

FALSE：优化器不会考虑星型转换。

示例：我们以 ORACLE 默认 SAMPLE SH 为例

```
explain plan for
SELECT c.cust_city, t.calendar_quarter_desc, SUM(s.amount_sold) sales_amount
 FROM sh.sales s, sh.times t, sh.customers c, sh.channels ch
 WHERE s.time_id = t.time_id AND s.cust_id = c.cust_id AND s.channel_id = ch.channel_id
  AND c.cust_state_province = 'FL'
  AND ch.channel_desc = 'Direct Sales'
  AND t.calendar_quarter_desc IN ('2000-01', '2000-02','1999-12') GROUP BY c.cust_city,
t.calendar_quarter_desc;
```

未开启 star_transformation_enabled 的执行计划

```
| Id  | Operation                      | Name        | Rows  | Bytes | Cost (%CPU)| Time     | Pstart| Pstop |
|   0 | SELECT STATEMENT               |             |   607 | 46132 |   968   (3)| 00:00:12 |       |       |
|   1 |  HASH GROUP BY                 |             |   607 | 46132 |   968   (3)| 00:00:12 |       |       |
|*  2 |   HASH JOIN                    |             |  2337 |  173K|   967   (3)| 00:00:12 |       |       |
|   3 |    PART JOIN FILTER CREATE     | :BF0000     |   274 |  4384 |    18   (0)| 00:00:01 |       |       |
|*  4 |     TABLE ACCESS FULL          | TIMES       |   274 |  4384 |    18   (0)| 00:00:01 |       |       |
|*  5 |    HASH JOIN                   |             | 12456 |  729K|   948   (3)| 00:00:12 |       |       |
|   6 |     MERGE JOIN CARTESIAN       |             |   383 | 14937 |   409   (1)| 00:00:05 |       |       |
|*  7 |      TABLE ACCESS FULL         | CHANNELS    |     1 |    13 |     3   (0)| 00:00:01 |       |       |
|   8 |      BUFFER SORT               |             |   383 |  9958 |   406   (1)| 00:00:05 |       |       |
|*  9 |       TABLE ACCESS FULL        | CUSTOMERS   |   383 |  9958 |   406   (1)| 00:00:05 |       |       |
|  10 |     PARTITION RANGE JOIN-FILTER|             |  918K|    18M|   533   (3)| 00:00:07 |:BF0000|:BF0000|
|  11 |      TABLE ACCESS FULL         | SALES       |  918K|    18M|   533   (3)| 00:00:07 |:BF0000|:BF0000|

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("S"."TIME_ID"="T"."TIME_ID")
   4 - filter("T"."CALENDAR_QUARTER_DESC"='1999-12' OR "T"."CALENDAR_QUARTER_DESC"='2000-01' OR
              "T"."CALENDAR_QUARTER_DESC"='2000-02')
   5 - access("S"."CUST_ID"="C"."CUST_ID" AND "S"."CHANNEL_ID"="CH"."CHANNEL_ID")
   7 - filter("CH"."CHANNEL_DESC"='Direct Sales')
   9 - filter("C"."CUST_STATE_PROVINCE"='FL')
```

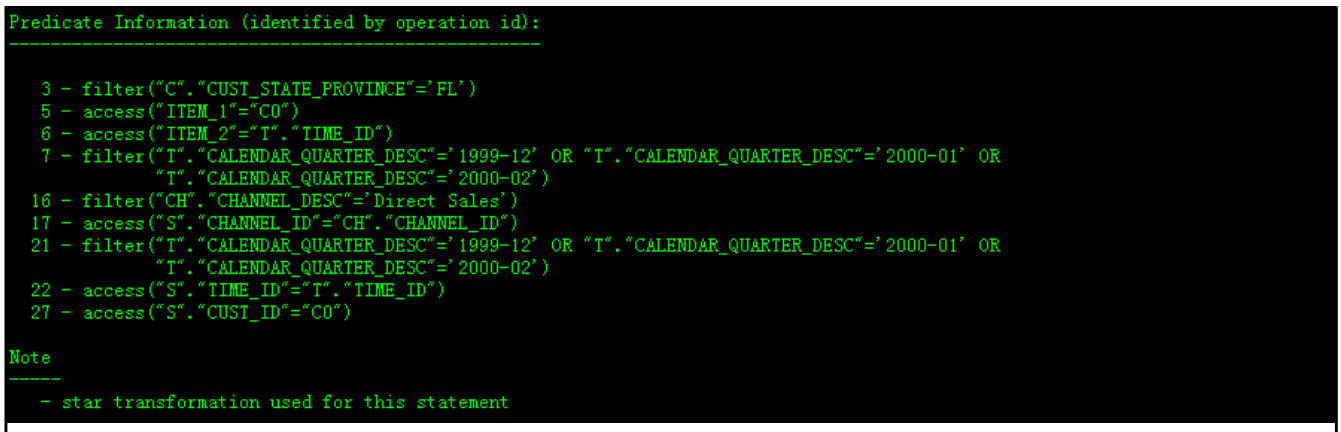
以上执行计划顺序是：4 3 7 9 8 6 11 10 5 2 1 0

开启 star_transformation_enabled 后的执行计划

```
| Id  | Operation                      | Name                     | Rows  | Bytes | Cost (%CPU)| Time     | Pstart| Pstop |
|   0 | SELECT STATEMENT               |                          |   238 | 13566 |   551   (1)| 00:00:07 |       |       |
|   1 |  TEMP TABLE TRANSFORMATION     |                          |       |       |            |          |       |       |
|   2 |   LOAD AS SELECT               | SYS_TEMP_0FD9D6617_D3180 |       |       |            |          |       |       |
|*  3 |    TABLE ACCESS FULL           | CUSTOMERS                |   383 |  9958 |   406   (1)| 00:00:05 |       |       |
|   4 |   HASH GROUP BY                |                          |   238 | 13566 |   145   (3)| 00:00:02 |       |       |
|*  5 |    HASH JOIN                   |                          |   238 | 13566 |   144   (2)| 00:00:02 |       |       |
|*  6 |     HASH JOIN                  |                          |   238 |  9996 |   142   (2)| 00:00:02 |       |       |
|*  7 |      TABLE ACCESS FULL         | TIMES                    |   274 |  4384 |    18   (0)| 00:00:01 |       |       |
|   8 |      VIEW                      | VW_ST_A3F94988           |   238 |  6188 |   123   (1)| 00:00:02 |       |       |
|   9 |       NESTED LOOPS             |                          |   238 | 13566 |   100   (1)| 00:00:02 |       |       |
|  10 |        PARTITION RANGE SUBQUERY|                          |   237 |  6660 |    56   (2)| 00:00:01 |KEY(SQ)|KEY(SQ)|
|  11 |         BITMAP CONVERSION TO ROWIDS|                      |   237 |  6660 |    56   (2)| 00:00:01 |       |       |
|  12 |          BITMAP AND            |                          |       |       |            |          |       |       |
|  13 |           BITMAP MERGE         |                          |       |       |            |          |       |       |
|  14 |            BITMAP KEY ITERATION|                          |       |       |            |          |       |       |
|  15 |             BUFFER SORT        |                          |       |       |            |          |       |       |
|* 16 |              TABLE ACCESS FULL | CHANNELS                 |     1 |    13 |     3   (0)| 00:00:01 |       |       |
|* 17 |             BITMAP INDEX RANGE SCAN| SALES_CHANNEL_BIX        |       |       |            |          |KEY(SQ)|KEY(SQ)|
|  18 |           BITMAP MERGE         |                          |       |       |            |          |       |       |
|  19 |            BITMAP KEY ITERATION|                          |       |       |            |          |       |       |
|  20 |             BUFFER SORT        |                          |       |       |            |          |       |       |
|* 21 |              TABLE ACCESS FULL | TIMES                    |   274 |  4384 |    18   (0)| 00:00:01 |       |       |
|* 22 |             BITMAP INDEX RANGE SCAN| SALES_TIME_BIX           |       |       |            |          |KEY(SQ)|KEY(SQ)|
|  23 |           BITMAP MERGE         |                          |       |       |            |          |       |       |
|  24 |            BITMAP KEY ITERATION|                          |       |       |            |          |       |       |
|  25 |             BUFFER SORT        |                          |       |       |            |          |       |       |
|  26 |              TABLE ACCESS FULL | SYS_TEMP_0FD9D6617_D3180 |   383 |  1915 |     2   (0)| 00:00:01 |       |       |
|* 27 |             BITMAP INDEX RANGE SCAN| SALES_CUST_BIX           |       |       |            |          |KEY(SQ)|KEY(SQ)|
|  28 |        TABLE ACCESS BY USER ROWID | SALES                    |     1 |    29 |    67   (0)| 00:00:01 | ROWID | ROWID |
|  29 |     TABLE ACCESS FULL          | SYS_TEMP_0FD9D6617_D3180 |   383 |  5745 |     2   (0)| 00:00:01 |       |       |
```

下面我们可以看到执行计划尾部的 Note 中已经明确提示了使用

3 2 7 16 15 17 14 13 21 20 22 19 18 26 25 27 24 23 12 11 10 28 9 8 6 29 5 4 1 0



star transformation used for this statment

There are some preconditions should be met before star transforming join can be used by the optimizer:

There must be one fact table and more than 2 dimension tables.
There should be bitmap indexes created on the join columns in the fact table.
There should be statistics gathered on the fact table.
The parameter “star_transformation_enabled” should be set to TRUE or TEMP_DISABLE. Or the
hint (STAR_TRANSFORMATION) should be used in the SQL statement.
Please note that if the SQL statement uses the bind variable, the star transforming join will not be used
by the optimizer as the optimizer needs to know the statistics of the fact table. The bind variable will
make the optimizer have no idea of the statistics.

—--

所谓星形查询(Star Query)是指一个事实表(Fact Table)与多个
维度表(Dimension Table)的关联查询,并且维度表仅与事实表之间关联
,维度表之间不存在关联关系。星形查询分为两个阶段:

1, 第一阶段是由事实表利用位图索引的位图信息(或者由 B*树索引的
ROWID 转换得来的位图信息)进行位图操作,进而获得相应数据集;
2,第二阶段则将第一阶段获得的数据集与维度表进行关联,获取最终查询
结果。
星形转换是将事实表与多个维度表的普通关联查询转换为星形查询的一项
优化技术。

1、connect as the oracle user and execute the:
./setup_star_schema_lab.sh

$ sqlplus sh/sh
SQL> @first_run
2、this query seems to use a large number of bytes to access the SALES

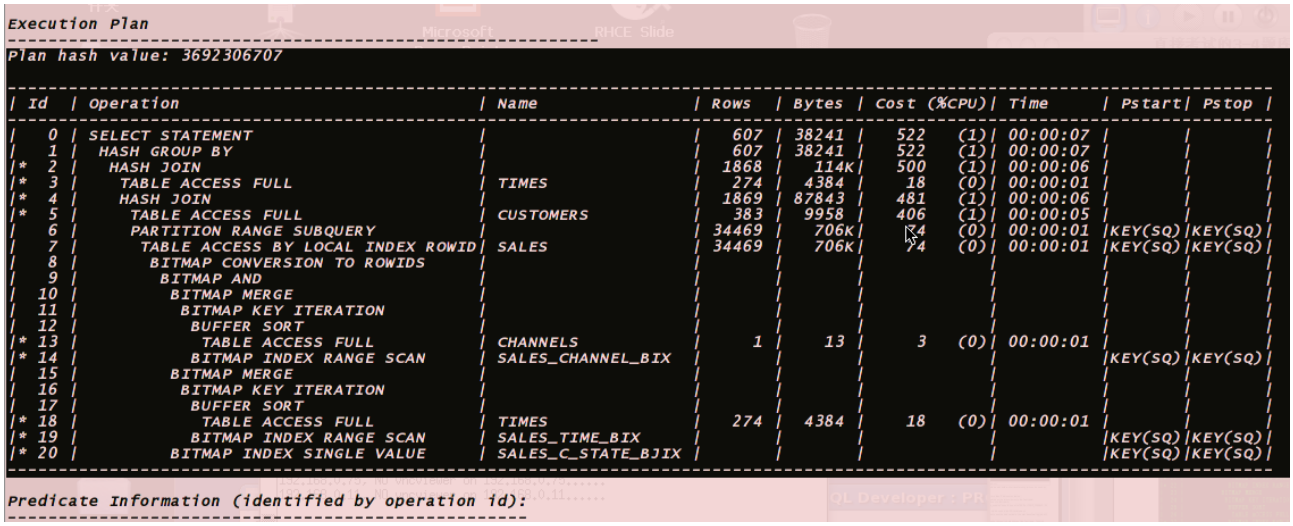
table. Basically, the optimizer performs a full scan of this table. This might not be the best way to handle it.
3、Without modifying the SH schema, how can you improve the execution plan for the query mentioned in step 2? Verify your solution and explain why it is probably a better solution.
SQL> @second_run

4、How would you enhance the previous optimization without changing the SH schema? You can use third_run.sql.
a) Let the optimizer decide if it is better to use a temporary table. You can try to set the
STAR_TRANSFORMATION_ENABLED parameter to TRUE.
@third_run.sql

5、How do you eliminate one access on the CUSTOMERS table from the previous execution plan for the same SELECT statement seen in step 3? (消除一次对 customers 表的访问)
a) Create a bitmap join index between the SALES and CUSTOMERS tables.

6, Because the CUSTOMERS_PK primary key constraint is not enforced, it is not possible to create a bitmap join index between the SALES and CUSTOMERS tables.
SQL> alter table customers enable constraint customers_pk;

SQL> CREATE BITMAP INDEX sales_c_state_bjix ON
sales(customers.cust_state_province)
FROM sales, customers
WHERE sales.cust_id=customers.cust_id
LOCAL NOLOGGING COMPUTE STATISTICS;
7、Verify that you solved the problem from step 5. You can use fourth_run.sql.
SQL> @fourth_run.sql



```
Execution Plan
----------------------------------------------------------
Plan hash value: 3692306707

-------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                                | Name                | Rows  | Bytes | Cost (%CPU)| Time     | Pstart| Pstop |
-------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                         |                     |   607 | 38241 |   522   (1)| 00:00:07 |       |       |
|   1 |  HASH GROUP BY                           |                     |   607 | 38241 |   522   (1)| 00:00:07 |       |       |
|*  2 |   HASH JOIN                              |                     |  1868 |   114K|   500   (1)| 00:00:06 |       |       |
|*  3 |    TABLE ACCESS FULL                     | TIMES               |   274 |  4384 |    18   (0)| 00:00:01 |       |       |
|*  4 |    HASH JOIN                             |                     |  1869 | 87843 |   481   (1)| 00:00:06 |       |       |
|*  5 |     TABLE ACCESS FULL                    | CUSTOMERS           |   383 |  9958 |   406   (1)| 00:00:05 |       |       |
|   6 |     PARTITION RANGE SUBQUERY             |                     | 34469 |   706K|    74   (0)| 00:00:01 |KEY(SQ)|KEY(SQ)|
|   7 |      TABLE ACCESS BY LOCAL INDEX ROWID   | SALES               | 34469 |   706K|    74   (0)| 00:00:01 |KEY(SQ)|KEY(SQ)|
|   8 |       BITMAP CONVERSION TO ROWIDS        |                     |       |       |             |          |       |       |
|   9 |        BITMAP AND                        |                     |       |       |             |          |       |       |
|  10 |         BITMAP MERGE                     |                     |       |       |             |          |       |       |
|  11 |          BITMAP KEY ITERATION            |                     |       |       |             |          |       |       |
|  12 |           BUFFER SORT                    |                     |       |       |             |          |       |       |
|* 13 |            TABLE ACCESS FULL             | CHANNELS            |     1 |    13 |     3   (0)| 00:00:01 |       |       |
|* 14 |           BITMAP INDEX RANGE SCAN        | SALES_CHANNEL_BIX   |       |       |             |          |KEY(SQ)|KEY(SQ)|
|  15 |         BITMAP MERGE                     |                     |       |       |             |          |       |       |
|  16 |          BITMAP KEY ITERATION            |                     |       |       |             |          |       |       |
|  17 |           BUFFER SORT                    |                     |       |       |             |          |       |       |
|* 18 |            TABLE ACCESS FULL             | TIMES               |   274 |  4384 |    18   (0)| 00:00:01 |       |       |
|* 19 |           BITMAP INDEX RANGE SCAN        | SALES_TIME_BIX      |       |       |             |          |KEY(SQ)|KEY(SQ)|
|* 20 |         BITMAP INDEX SINGLE VALUE        | SALES_C_STATE_BJIX  |       |       |             |          |KEY(SQ)|KEY(SQ)|
-------------------------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------
```

6.Administer external tables

11gloader 新特性

```
 CREATE TABLE sales_transactions_ext
(PROD_ID NUMBER, CUST_ID NUMBER,
 TIME_ID DATE, CHANNEL_ID NUMBER,
 PROMO_ID NUMBER, QUANTITY_SOLD NUMBER,
 AMOUNT_SOLD NUMBER(10,2), UNIT_COST NUMBER(10,2),
 UNIT_PRICE NUMBER(10,2))
ORGANIZATION external (TYPE oracle_loader
  DEFAULT DIRECTORY data_file_dir ACCESS PARAMETERS
  (RECORDS DELIMITED BY NEWLINE CHARACTERSET US7ASCII
    PREPROCESSOR exec_file_dir:'gunzip' OPTIONS '-c'
    BADFILE log_file_dir:'sh_sales.bad_xt'
    LOGFILE log_file_dir:'sh_sales.log_xt'
    FIELDS TERMINATED BY "|" LDRTRIM
    ( PROD_ID, CUST_ID,
      TIME_ID         DATE(10) "YYYY-MM-DD",
      CHANNEL_ID, PROMO_ID, QUANTITY_SOLD, AMOUNT_SOLD,
      UNIT_COST, UNIT_PRICE))
  location ('sh_sales.dat.gz')
REJECT LIMIT UNLIMITED;
```

可执行程序如果发现“KUP-01008: the bad identifier was: OPTIONS ”

```
#/bin/sh
/usr/bin/gunzip -c <$1
```

如果文件是 zip，那么可执行程序用 zcat

7.Implement Data Pump export and import jobs for data transfer

8.Implement Data Pump to and from remote databases

In the following example, the source_database_link would be replaced with the name of a valid database link that must already exist.
> impdp hr/hr TABLES=employees DIRECTORY=dpump_dir1
NETWORK_LINK=source_database_link EXCLUDE=CONSTRAINT

9.Configure and use parallel execution for queries

show parameter parallel

优先顺序 Hint > session > object -- hint select /*+ parallel(e 4) */ * from hr.employees e; -- objects alter table employees parallel 4; alter index emp_id parallel 4; -- session alter session enable parallel query; alter session enable parallel dml; alter session enable parallel ddl; ALTER SESSION FORCE PARALLEL DDL PARALLEL 5; SELECT pdml_status,PQ_STATUS FROM v$session where sid=170;
并行查询会占用更多的内存排序区，并行成本=成本*并行数*2,例如开 1G 的排序区，20 个并行进程，那么在极限情况下内存占用应为 1G*20*2=40G 内存小于此量会占用 pagefile 性能急剧下降
alter system set parallel_max_servers=50 -- 建议至少 2 倍于表指定并行的数量
redo 并行 x$ksppi 中查询隐含参数

10.Use SQL*Loader

11.Administer, manage and tune parallel execution

一定会考

```
alter session set parallel_degree_policy=auto;
alter session set parallel_min_time_threshold=20;
v$io_calibration_status
select  * from v$io_calibration_status;
set serveroutput on;
DECLARE
   lat number;
   iops number;
  mbps number;
BEGIN
  DBMS_RESOURCE_MANAGER.CALIBRATE_IO (2, 10, iops, mbps, lat);
  DBMS_OUTPUT.PUT_LINE ('max_iops = ' || iops);
  DBMS_OUTPUT.PUT_LINE ('latency = ' || lat);
  dbms_output.put_line('max_mbps = ' || mbps);
```

```
end;
/
```

Command-Line Solutions for Practice 4-11: Enabling Parallel DML

1. Connected as user SH, create the partitioned table LITTLE_SALES by executing the following SQL statements:

```
CREATE TABLE little_sales
PARTITION BY HASH (time_id)
(PARTITION LS1,PARTITION LS2)
PARALLEL
AS
SELECT * FROM sales WHERE 1=2;
```

Note that the LITTLE_SALES table has only two partitions, and the dictionary DOP is set to DEFAULT. Execute the following SQL statements to insert some rows into the LITTLE_SALES table:

```
INSERT INTO little_sales
SELECT *
FROM sales
WHERE ROWNUM<5000;
```

Is the INSERT statement executed in parallel? You can confirm it by the V$PQ_SESSTAT view.

Answer: A DML statement can be parallelized only if you specifically issue an ALTER SESSION statement to enable parallel DML:

```
SQL> connect SH/SH
SQL> CREATE TABLE little_sales
  2  PARTITION BY HASH (time_id)
  3  (PARTITION LS1,PARTITION LS2)
  4  PARALLEL
  5  AS
  6  SELECT * FROM sales WHERE 1=2;

Table created.

SQL> INSERT INTO little_sales
  2  SELECT *
  3  FROM sales
  4  WHERE ROWNUM<5000;

4999 rows created.

SQL> select * from v$pq_sesstat where STATISTIC = 'Allocation
Height';

STATISTIC                      LAST_QUERY SESSION_TOTAL
------------------------------ ---------- -------------
Allocation Height                       0             0
```

```
SQL> commit;

Commit complete.

SQL> ALTER SESSION ENABLE PARALLEL DML;

Session altered.

SQL> INSERT INTO little_sales
  2  SELECT *
  3  FROM sales
  4  WHERE ROWNUM<5000;

4999 rows created.

SQL> select * from v$pq_sesstat where STATISTIC = 'Allocation
Height';

STATISTIC                      LAST_QUERY SESSION_TOTAL
------------------------------ ---------- -------------
Allocation Height                       2             0

SQL> commit;

Commit complete.
```

2. Enable parallel DML in your session. When done, delete all rows from the `LITTLE_SALES` table where `PROD_ID` is an even number. Do not commit your changes. Try to retrieve rows from the `LITTLE_SALES` table. What happens and why?

   Answer: As explained during this lesson, after your transaction has modified an object in parallel, it is impossible to query the content of that object until the parallel changes are committed or rolled back. If you try anyway, you get an `ORA-12838` error message.

```
SQL> ALTER SESSION FORCE PARALLEL DML PARALLEL 5;
Session altered.

SQL> DELETE little_sales WHERE MOD(prod_id,2)=0;
2469 rows deleted.

SQL> select * from little_sales;
select * from little_sales
              *
ERROR at line 1:
ORA-12838: cannot read/modify an object after modifying it in
parallel
```

3. From a second session connected as SYS, count the number of rows in the SH.LITTLE_SALES table. Then, delete rows from SH.LITTLE_SALES with an odd PROD_ID value. What happens, and why?

   Answer: Locking for PDML statements is different from locking for serial DML statements. Although it is still possible to query all rows of the LITTLE_SALES table from another session, it is impossible to modify rows that are not modified by the first session. This is because the execution coordinator of the first session is holding an exclusive lock on each partition.

```
SQL> connect / as sysdba
Connected.
SQL> select count(*) from sh.little_sales;

  COUNT(*)
----------
      9998

SQL> DELETE sh.little_sales WHERE  MOD(prod_id,2) != 0;

HANG!
```

4. How can you verify this from the first session?

   Answer: You can query the V$LOCK view for exclusive locks held on the impacted partitions. In the following solution, ID1 represents the data object number for the corresponding partitions.

```
SQL> SELECT sid, type, lmode, id1, id2
  2  FROM   v$lock
  3  WHERE  sid IN (SELECT sid
  4    FROM v$session
  5    WHERE username='SH') AND type='TM' AND lmode=6;

       SID TY      LMODE        ID1        ID2
---------- -- ---------- ---------- ----------
       159 TM          6      55313          0
```

5. Roll back your changes from both sessions, and exit from the second session. Then, from the first session, drop the LITTLE_SALES table and truncate the PLAN_TABLE table.

```
SQL1> -- From session 1

SQL1> rollback;
Rollback complete.

SQL2> -- From session 2
…
```

```
2542 rows deleted.

SQL2> rollback;
Rollback complete.

SQL2> exit

SQL1> DROP  TABLE little_sales;
Table dropped.

SQL1> TRUNCATE TABLE plan_table;
Table truncated.
```

Command-Line Solutions for Practice 4-12: Managing Processes for Parallel SQL Execution

1. Connect as SYSDBA and query the current values of the initialization parameters dealing with parallel execution that are of interest to you. Execute the following SQL statements:

```
SELECT name, value FROM v$parameter
WHERE UPPER(name) IN
('LARGE_POOL_SIZE','PROCESSES','SESSIONS',
'PARALLEL_EXECUTION_MESSAGE_SIZE','PARALLEL_MIN_SERVERS',
'PARALLEL_MAX_SERVERS','PARALLEL_ADAPTIVE_MULTI_USER',
'PARALLEL_MIN_PERCENT');
```

```
SQL> connect / as sysdba
Connected.
SQL> COL name FORMAT A35
SQL> COL value FORMAT A15
SQL> SELECT name, value FROM v$parameter
  2  WHERE UPPER(name) IN ('LARGE_POOL_SIZE','PROCESSES','SESSIONS',
  3  'PARALLEL_EXECUTION_MESSAGE_SIZE','PARALLEL_MIN_SERVERS',
  4  'PARALLEL_MAX_SERVERS','PARALLEL_ADAPTIVE_MULTI_USER',
  5  'PARALLEL_MIN_PERCENT');

NAME                                VALUE
----------------------------------- ---------------
processes                           40
sessions                            49
large_pool_size                     0
parallel_min_percent                0
parallel_min_servers                0
parallel_max_servers                20
parallel_execution_message_size     2148
parallel_adaptive_multi_user        TRUE

8 rows selected.
```

2. Connect as user SH and execute the following statements to create the TEMP_CHANNELS table.

```
CREATE TABLE temp_channels
  PARALLEL 5 AS
  SELECT *
  FROM sh.channels
  WHERE channel_id IN ('2','3','4');
```

```
SQL> connect sh/sh
SQL> CREATE TABLE temp_channels
  2  PARALLEL 5 AS
  3  SELECT *
  4  FROM sh.channels
  5  WHERE channel_id IN ('2','3','4');
```

```
Table created.
```

3. Connect as SYSDBA and change the PARALLEL_MAX_SERVERS parameter to 5 and the PARALLEL_ADAPTIVE_MULTI_USER to FALSE. Verify that the change is in effect.

```
SQL> connect / as sysdba
SQL> alter system set parallel_max_servers = 5;

System altered.

SQL> alter system set parallel_adaptive_multi_user = false;

System altered.

SQL> COL name FORMAT A35
SQL> COL value FORMAT A15
SQL> SELECT name, value FROM v$parameter
  2  WHERE UPPER(name) IN ('LARGE_POOL_SIZE','PROCESSES','SESSIONS',
  3  'PARALLEL_EXECUTION_MESSAGE_SIZE','PARALLEL_MIN_SERVERS',
  4  'PARALLEL_MAX_SERVERS','PARALLEL_ADAPTIVE_MULTI_USER',
  5  'PARALLEL_MIN_PERCENT');

NAME                                VALUE
----------------------------------- ---------------
processes                           40
sessions                            49
large_pool_size                     0
parallel_min_percent                0
parallel_min_servers                0
parallel_max_servers                5
parallel_execution_message_size     2148
parallel_adaptive_multi_user        FALSE

8 rows selected.
```

4. In a single statement, create a TEMP_SALES table as user SH with minimal logging and with a DOP set to 5. This table should be a subset of the SALES table; only containing rows with CHANNEL_ID values '2' or '3'. Execute the following SQL statements to create the table. Examine V$PQ_SESSTAT to determine the DOP for the operation.

```
CREATE TABLE temp_sales NOLOGGING PARALLEL 5 AS
   SELECT *
   FROM sh.sales
   WHERE channel_id IN ('2','3');
```

```
SQL> connect sh/sh

SQL> CREATE TABLE temp_sales NOLOGGING PARALLEL 5 AS
  2     SELECT *
```

```
  3     FROM sh.sales
  4     WHERE channel_id IN ('2','3');

Table created.

SQL> select * from v$pq_sesstat;

STATISTIC                      LAST_QUERY SESSION_TOTAL
------------------------------ ---------- -------------
Queries Parallelized                    0             0
DML Parallelized                        0             0
DDL Parallelized                        1             2
DFO Trees                               1             2
Server Threads                          5             0
Allocation Height                       5             0
Allocation Width                        1             0
Local Msgs Sent                       149           186
Distr Msgs Sent                         0             0
Local Msgs Recv'd                     149           186
Distr Msgs Recv'd                       0             0
```

5. Execute the following SQL statements. This script joins the two tables TEMP_SALES and TEMP_CHANNELS using a DOP of 5. Examine V$PQ_SESSTAT to determine the DOP for the operation.

```
SELECT  count(*)
FROM temp_sales s, temp_channels c
WHERE (s.channel_id) = (c.channel_id);
```

```
SQL> connect sh/sh
Connected.
SQL> SELECT  count(*)
  2  FROM temp_sales s, temp_channels c
  3  WHERE (s.channel_id) = (c.channel_id);

  COUNT(*)
----------
    917048

SQL> select * from v$pq_sesstat;

STATISTIC                      LAST_QUERY SESSION_TOTAL
------------------------------ ---------- -------------
Queries Parallelized                    1             1
DML Parallelized                        0             0
DDL Parallelized                        0             0
DFO Trees                               1             1
Server Threads                          4             0
Allocation Height                       2             0
Allocation Width                        1             0
Local Msgs Sent                        92            92
Distr Msgs Sent                         0             0
```

```
Local Msgs Recv'd                       96            96
Distr Msgs Recv'd                        0             0

11 rows selected.
```

Answer: You can see from the query of V$PQ_SESSTAT that the number of server threads is 4 and the allocation height is 2. This means that the statement executed with a DOP of 2 with 2 sets of slaves even though you specified a DOP of 5, and that the database should have been able to accommodate this number.

6. Connect as SYSDBA and restore the value of PARALLEL_MAX_SERVERS to its original value. Reconnect as user SH and rerun the query in step 5 and inspect V$PQ_SESSTAT when finished. Compare the results with those in step 5. What are your conclusions?

```
SQL> connect / as sysdba
Connected.
SQL> alter system set PARALLEL_MAX_SERVERS = 20;

System altered.

SQL> connect SH/SH
Connected.
SQL> SELECT  count(*)
  2  FROM temp_sales s, temp_channels c
  3  WHERE (s.channel_id) = (c.channel_id);

  COUNT(*)
----------
    917048

SQL> select * from v$pq_sesstat;

STATISTIC                      LAST_QUERY SESSION_TOTAL
------------------------------ ---------- -------------
Queries Parallelized                    1             1
DML Parallelized                        0             0
DDL Parallelized                        0             0
DFO Trees                               1             1
Server Threads                         10             0
Allocation Height                       5             0
Allocation Width                        1             0
Local Msgs Sent                       224           224
Distr Msgs Sent                         0             0
Local Msgs Recv'd                     234           234
Distr Msgs Recv'd                       0             0

11 rows selected.
```

Answer: You can see from the query of V$PQ_SESSTAT that the number of server threads is 10 and the allocation height is 5. This means that the statement executed

with a DOP of 5 with 2 sets of slaves. When the same statement is run in an environment where the value of PARALLEL_MAX_SERVERS is equal to that of the requested DOP as was the case in step 5, the DOP is reduced by the adaptive multiuser algorithm to ensure there are enough parallel resources for other user requests.

7. When you are finished, please drop the TEMP_CHANNELS and TEMP_SALES tables.

```
SQL> drop table temp_channels;
Table dropped.

SQL> drop table temp_sales;
Table dropped.
```

Oracle® Tutor™

Section 4 数据仓库管理（Data Warehouse Management ）

1.Administer partitioned tables and indexes using appropriate methods and keys

SQL Language Reference=>16 SQL Statements: CREATE SYNONYM to CREATE TRIGGER

#interval partitioning 按照指定规则自动创建分区,初始分区为创建时指定的分区,当新数据插入时自动创建新分区

```
CREATE TABLE interval_demo ( customer_id number, name varchar2(20))
PARTITION BY RANGE (customer_id) INTERVAL (10) (PARTITION p1 VALUES
LESS THAN (10) tablespace users);
```

检查命令

```
col high_value format a20
col segment_name format a20
col partition_name format a20
SELECT partition_name, high_value FROM user_tab_partitions WHERE
table_name like '%DEMO%';
select SEGMENT_NAME,PARTITION_NAME from user_segments where
```

segment_name like '%DEMO%';

刚创建好的状态

```
PARTITION_NAME       HIGH_VALUE
-------------------- --------------------
P1                   10

SEGMENT_NAME         PARTITION_NAME
-------------------- --------------------
INTERVAL_DEMO        P1
```

```
--插入 50 条数据
begin
for a in 1..50 loop
insert into
interval_demo
values(a,'NAME');
end loop;
commit;
end;
/
```

```
PARTITION_NAME       HIGH_VALUE
-------------------- --------------------
P1                   10
SYS_P21              20
SYS_P22              30
SYS_P23              40
SYS_P24              50
SYS_P25              60

SEGMENT_NAME         PARTITION_NAME
-------------------- --------------------
INTERVAL_DEMO        P1
INTERVAL_DEMO        SYS_P21
INTERVAL_DEMO        SYS_P22
INTERVAL_DEMO        SYS_P23
INTERVAL_DEMO        SYS_P24
INTERVAL_DEMO        SYS_P25
```

#Reference Partitioning 外键自动关联主键分区

--主分区表必须以主键为分区关键字必须 not null 并且不支持 interval 形式分区表

```
CREATE TABLE ref_m_demo(customer_id number not null,name
varchar2(20),
CONSTRAINT ref_m_demo_pk PRIMARY KEY(customer_id))

PARTITION BY RANGE (customer_id) (PARTITION p1 VALUES LESS THAN
(10),PARTITION p2 VALUES LESS THAN (20));
```

--外键同步分区表,同样，外键关联字段必须 not null

```
CREATE TABLE ref_s_demo (emp_id number not null, city varchar2(20),
CONSTRAINT ref_s_demo_fk FOREIGN KEY(emp_id) REFERENCES
ref_m_demo(customer_id))
PARTITION BY REFERENCE (ref_s_demo_fk);
```

```
col owner format a5
col segment_name format a20
col partition_name format a20
select OWNER,
SEGMENT_NAME,PARTITION_NAME
from dba_segments where
SEGMENT_NAME like 'REF%';
```

```
OWNER SEGMENT_NAME         PARTITION_NAME
----- -------------------- --------------
SH    REF_M_DEMO           P1
SH    REF_M_DEMO           P2
SH    REF_M_DEMO_PK
SH    REF_S_DEMO           P1
SH    REF_S_DEMO           P2
```

分区类型

Range 数值范围
Hash 关键词值进行分区，分的更均匀，适合等值查询
List 等于指定值，精确匹配分区
range-hash 组合分区，大 range 小 hash
Range-list 组合分区，大 range 小 list 子分区定义模板，如果在木分区之内特意

指定了子分区信息则不使用模板

```
select  i.index_name ,  decode(  ip.status,null,i.status,ip.status), i.partitioned
 from dba_ind_partitions ip , dba_indexes i
 where ip.index_name(+)=i.index_name
 and i.table_name='T2' and i.table_owner='SH';
```

```
alter table p1 rename partition part_1 to partx_1;--分区改名字
alter table p1 truncate partition partx_1update index;--清除分区数据自动更新索引
alter table p1 drop partition part_3; --删除分区时自动清理掉本地索引删除分区
alter table p1 add partition part_4 values less than ('40') tablespace users;--增加 range 分区
ALTER TABLE "HR"."P1" MOVE PARTITION "PART_MAX" TABLESPACE "P3" ;--移动
alter table p1 split partition part_4 at ('35');---range partition 切割分区
alter table p3 split partition part_1 values('good');---list partition 切割分区
alter table p1 merge partitions part_3,part_4;---range ,list 合并分区
alter table p2 coalesce partition;--收缩 hash 分区，每次减少一个合并分区
alter table p3 modify partition part_4 drop values('zzz');--修改 list 分区分区关键字
alter table p3 modify partition part_4 add values('zzz');--修改 list 分区分区关键字
分区表的分区与普通表进行交换
alter table p1 exchange partition part_1 with table test [without validation];不校验数据是否符合分区规范
alter table p1 exchange partition part_1 with table test [with validation];--default 校验数据
alter table p1 exchange partition part_1 with table test [including indexes];
alter table p1 exchange partition part_1 with table test [excluding indexes];--default
导出一个分区
exp hr/hr tables=p1:part_1 file=p1_part_1.dmp
移动分区，包括 LOB 的
alter table image_table move tablespace users lob (image_DATA) store as (tablespace users);
```

alter table tp1 exchange partition p3 with table tpsg including indexes;

如果包含 including indexes，非分区表必需有索引。

可以分区表交换为双层分区表的子分区：
创建单层分区表：

```
CREATE TABLE t1 (i NUMBER, j NUMBER)
    PARTITION BY HASH(i)
     (PARTITION p1, PARTITION p2);
```

创建复合分区表（双层）：

```
CREATE TABLE t2 (i NUMBER, j NUMBER)
    PARTITION BY RANGE(j)
    SUBPARTITION BY HASH(i)
      (PARTITION p1 VALUES LESS THAN (10)
         SUBPARTITION t2_pls1
         SUBPARTITION t2_pls2,
       PARTITION p2 VALUES LESS THAN (20)
         SUBPARTITION t2_p2s1
         SUBPARTITION t2_p2s2));
```

交换：

```
ALTER TABLE t2 EXCHANGE PARTITION p1 WITH TABLE t1
    WITH VALIDATION;
```

注意：T1 和 T2 中的分区关键字必须相同。

```
alter table tp1 split partition p2 at (30) into (partition p2_1,partition p2_2);
```

显示结果：

```
SQL> select * from tp1 partition(p2_1);

       ID1        ID2        ID3 CC
---------- ---------- ---------- --------------------
        29         29         29 abc

SQL> select * from tp1 partition(p2_2);

       ID1        ID2        ID3 CC
---------- ---------- ---------- --------------------
        30         30         30 abc
        39         39         39 abc
```

2、切割列表分区：

```
ALTER TABLE sales_by_region
  SPLIT PARTITION region_east VALUES ('CT', 'MA', 'MD')
```

```
  INTO
  ( PARTITION region_east_1
       TABLESPACE tbs2,
    PARTITION region_east_2
      STORAGE (NEXT 2M PCTINCREASE 25))
  PARALLEL 5;
```

region_east_1 中包含'CT', 'MA', 'MD'，原分区中其他的值被切割到 region_east_2 分区中。

3、切割 Range-Hash 分区

```
ALTER TABLE all_seasons SPLIT PARTITION quarter_1
    AT (TO_DATE('16-dec-1997','dd-mon-yyyy'))
    INTO (PARTITION q1_1997_1 SUBPARTITIONS 4 STORE IN (ts1,ts3),
         PARTITION q1_1997_2);
```

q1_1997_1 有四个子分区，依次存放在两个表空间：ts1、ts3 中。
q1_1997_2 没有设定值，它继承原来分区的特性。

4、切割 Range-List 分区

```
ALTER TABLE quarterly_regional_sales SPLIT PARTITION q1_1999
  AT (to_date('15-Feb-1999','dd-mon-yyyy'))
  INTO ( PARTITION q1_1999_jan_feb
           TABLESPACE ts1,
         PARTITION q1_1999_feb_mar
           STORAGE (NEXT 2M PCTINCREASE 25) TABLESPACE ts2)
  PARALLEL 5;
```

补充：注意：包含索引的分区交换操作要想成功，两个表的索引类型必须完全一致，也就是说，原表的索引是分区的，目标表索引也是分区，分区类型要一致，其他可以不一致，只要分区类型一致就行。

五、分区的 Coalescing

```
ALTER TABLE ouu1 COALESCE PARTITION;
```

拼合表的所有分区

```
ALTER TABLE diving MODIFY PARTITION us_locations COALESCE SUBPARTITION;
```

拼合子分区下的所有分区

ALTER INDEX hgidx COALESCE PARTITION;
拼合全局索引的所有分区

六、合并分区
alter table tp1 merge partitions p2_1,p2_2 into partition p2;

alter table tp1 merge partitions p2_1,p2_2 into partition p2 update indexes;

七、为列表索引增加、删除值：
ALTER TABLE sales_by_region MODIFY PARTITION region_south ADD VALUES ('OK', 'KS');

ALTER TABLE sales_by_region MODIFY PARTITION region_south DROP VALUES ('OK', 'KS');
（如果被删除的值对应有行，删除将报错）

八、分区的改名：
alter table tp1 rename partition p2 to tp2;

九、截断分区
对于索引组织表，截断分区不能使用 Update indexes，应该使用 UPDATE GLOBAL INDEXES 代替。
alter table tp1 truncate partition tp2;
截断后对应区的本地索引继续保持可用，全局索引变为不可用。

截断子分区：
ALTER TABLE diving TRUNCATE SUBPARTITION us_locations DROP STORAGE;

十、交换分区
create table tpsg as select * from tp1 where 0=1;
create index tpsg_id1 on tpsg(id1);
insert into tpsg values(48,48,48,'tpsg');
commit;

alter table tp1 exchange partition p3 with table tpsg including indexes;

如果包含 including indexes，非分区表必需有索引。

可以分区表交换为双层分区表的子分区：

创建单层分区表：

```
CREATE TABLE t1 (i NUMBER, j NUMBER)
    PARTITION BY HASH(i)
     (PARTITION p1, PARTITION p2);
```

创建复合分区表（双层）：

```
CREATE TABLE t2 (i NUMBER, j NUMBER)
    PARTITION BY RANGE(j)
    SUBPARTITION BY HASH(i)
      (PARTITION p1 VALUES LESS THAN (10)
         SUBPARTITION t2_pls1
         SUBPARTITION t2_pls2,
       PARTITION p2 VALUES LESS THAN (20)
         SUBPARTITION t2_p2s1
         SUBPARTITION t2_p2s2));
```

交换：

```
ALTER TABLE t2 EXCHANGE PARTITION p1 WITH TABLE t1
    WITH VALIDATION;
```

注意：T1 和 T2 中的分区关键字必须相同。

子分区和普通表的交换：

```
ALTER TABLE sales EXCHANGE SUBPARTITION q3_1999_s1
     WITH TABLE q3_1999 INCLUDING INDEXES;
```

十二、HASH 分区

```
create table tp2 (id1 int,id2 int,cc varchar2(10)) partition by hash(id1)
(partition p1,partition p2);
```

十三、全局索引的分区：

```
create index tp2_id1 on tp2(id1) global partition by range(id1)
(partition p1 values less than (30),
 partition p2 values less than (60),
 partition p3 values less than (MAXVALUE)
);

create index tp2_id1 on tp2(id1) global partition by hash(id1)
(partition p1 , partition p2 , partition p3 );
```

全局索引分区只能是范围、HASH 的。

1):
Range Partitioning Example

```
CREATE TABLE sales_range
(salesman_id NUMBER(5),
salesman_name VARCHAR2(30),
sales_amount NUMBER(10),
sales_date DATE)
PARTITION BY RANGE(sales_date)
(
PARTITION sales_jan2000 VALUES LESS
THAN(TO_DATE('02/01/2000','MM/DD/YYYY')),
PARTITION sales_feb2000 VALUES LESS
THAN(TO_DATE('03/01/2000','MM/DD/YYYY')),
PARTITION sales_mar2000 VALUES LESS
THAN(TO_DATE('04/01/2000','MM/DD/YYYY')),
PARTITION sales_apr2000 VALUES LESS
THAN(TO_DATE('05/01/2000','MM/DD/YYYY'))
);
```

2):
List Partitioning Example

```
CREATE TABLE sales_list
(salesman_id NUMBER(5),
salesman_name VARCHAR2(30),
sales_state VARCHAR2(20),
sales_amount NUMBER(10),
sales_date DATE)
PARTITION BY LIST(sales_state)
(
PARTITION sales_west VALUES('California', 'Hawaii'),
PARTITION sales_east VALUES ('New York', 'Virginia', 'Florida'),
PARTITION sales_central VALUES('Texas', 'Illinois'),
PARTITION sales_other VALUES(DEFAULT)
);
```

3):
Hash Partitioning Example

```
CREATE TABLE sales_hash
(salesman_id NUMBER(5),
salesman_name VARCHAR2(30),
sales_amount NUMBER(10),
week_no NUMBER(2))
PARTITION BY HASH(salesman_id)
PARTITIONS 4
STORE IN (ts1, ts2, ts3, ts4);
```

4):
Composite Partitioning Range-Hash Example

```
CREATE TABLE sales_composite
(salesman_id NUMBER(5),
salesman_name VARCHAR2(30),
sales_amount NUMBER(10),
sales_date DATE)
PARTITION BY RANGE(sales_date)
SUBPARTITION BY HASH(salesman_id)
SUBPARTITION TEMPLATE(
SUBPARTITION sp1 TABLESPACE ts1,
SUBPARTITION sp2 TABLESPACE ts2,
SUBPARTITION sp3 TABLESPACE ts3,
SUBPARTITION sp4 TABLESPACE ts4)
(PARTITION sales_jan2000 VALUES LESS
THAN(TO_DATE('02/01/2000','MM/DD/YYYY'))
PARTITION sales_feb2000 VALUES LESS
THAN(TO_DATE('03/01/2000','MM/DD/YYYY'))
PARTITION sales_mar2000 VALUES LESS
THAN(TO_DATE('04/01/2000','MM/DD/YYYY'))
PARTITION sales_apr2000 VALUES LESS
THAN(TO_DATE('05/01/2000','MM/DD/YYYY'))
PARTITION sales_may2000 VALUES LESS
THAN(TO_DATE('06/01/2000','MM/DD/YYYY')));
```

```
CREATE TABLE emp (deptno NUMBER, empname VARCHAR(32), grade
NUMBER)
PARTITION BY RANGE(deptno) SUBPARTITION BY HASH(empname)
SUBPARTITIONS 8 STORE IN (ts1, ts3, ts5, ts7)
(PARTITION p1 VALUES LESS THAN (1000),
```

```
PARTITION p2 VALUES LESS THAN (2000)
STORE IN (ts2, ts4, ts6, ts8),
PARTITION p3 VALUES LESS THAN (MAXVALUE)
(SUBPARTITION p3_s1 TABLESPACE ts4,
SUBPARTITION p3_s2 TABLESPACE ts5));

-- range-list
CREATE TABLE sample_regional_sales
(deptno number, item_no varchar2(20),
txn_date date, txn_amount number, state varchar2(2))
PARTITION BY RANGE (txn_date)
SUBPARTITION BY LIST (state)
(PARTITION q1_1999 VALUES LESS THAN (TO_DATE('1-APR-1999','DD-MON-
YYYY'))
TABLESPACE tbs_1
(SUBPARTITION q1_1999_northwest VALUES ('OR', 'WA'),
SUBPARTITION q1_1999_southwest VALUES ('AZ', 'UT', 'NM'),
SUBPARTITION q1_1999_northeast VALUES ('NY', 'VM', 'NJ'),
SUBPARTITION q1_1999_southeast VALUES ('FL', 'GA'),
SUBPARTITION q1_others VALUES (DEFAULT) TABLESPACE tbs_4
),
PARTITION q2_1999 VALUES LESS THAN ( TO_DATE('1-JUL-1999','DD-MON-
YYYY'))
TABLESPACE tbs_2
(SUBPARTITION q2_1999_northwest VALUES ('OR', 'WA'),
SUBPARTITION q2_1999_southwest VALUES ('AZ', 'UT', 'NM'),
SUBPARTITION q2_1999_northeast VALUES ('NY', 'VM', 'NJ'),
SUBPARTITION q2_1999_southeast VALUES ('FL', 'GA'),
SUBPARTITION q2_1999_northcentral VALUES ('SD', 'WI'),
SUBPARTITION q2_1999_southcentral VALUES ('OK', 'TX')
),
PARTITION q3_1999 VALUES LESS THAN (TO_DATE('1-OCT-1999','DD-MON-
YYYY'))
TABLESPACE tbs_3
(SUBPARTITION q3_1999_northwest VALUES ('OR', 'WA'),
SUBPARTITION q3_1999_southwest VALUES ('AZ', 'UT', 'NM'),
SUBPARTITION q3_others VALUES (DEFAULT) TABLESPACE tbs_4
),
PARTITION q4_1999 VALUES LESS THAN ( TO_DATE('1-JAN-2000','DD-MON-
```

```
YYYY'))
TABLESPACE tbs_4
);

-- template
CREATE TABLE emp_sub_template (deptno NUMBER, empname VARCHAR(32),
grade NUMBER)
PARTITION BY RANGE(deptno) SUBPARTITION BY HASH(empname)
SUBPARTITION TEMPLATE
(SUBPARTITION a TABLESPACE ts1,
SUBPARTITION b TABLESPACE ts2,
SUBPARTITION c TABLESPACE ts3,
SUBPARTITION d TABLESPACE ts4
)
(PARTITION p1 VALUES LESS THAN (1000),
PARTITION p2 VALUES LESS THAN (2000),
PARTITION p3 VALUES LESS THAN (MAXVALUE)
);

1):
Creating a Range-Partitioned Global Index
CREATE INDEX month_ix ON sales(sales_month)
GLOBAL PARTITION BY RANGE(sales_month)
(PARTITION pm1_ix VALUES LESS THAN (2)
PARTITION pm2_ix VALUES LESS THAN (3)
PARTITION pm3_ix VALUES LESS THAN (4)
PARTITION pm4_ix VALUES LESS THAN (5)
PARTITION pm5_ix VALUES LESS THAN (6)
PARTITION pm6_ix VALUES LESS THAN (7)
PARTITION pm7_ix VALUES LESS THAN (8)
PARTITION pm8_ix VALUES LESS THAN (9)
PARTITION pm9_ix VALUES LESS THAN (10)
PARTITION pm10_ix VALUES LESS THAN (11)
PARTITION pm11_ix VALUES LESS THAN (12)
PARTITION pm12_ix VALUES LESS THAN (MAXVALUE));

CREATE INDEX hgidx ON tab (c1,c2,c3) GLOBAL
PARTITION BY HASH (c1,c2)
(PARTITION p1 TABLESPACE tbs_1,
```

```
PARTITION p2 TABLESPACE tbs_2,
PARTITION p3 TABLESPACE tbs_3,
PARTITION p4 TABLESPACE tbs_4);
```

2): local index: index columns is the same as partition key columns

```
CREATE INDEX loc_dept_ix ON dept(deptno) LOCAL;

CREATE INDEX emp_ix ON emp(deptno)
LOCAL STORE IN (ts7, ts8, ts9);

---
ALTER TABLE q1_sales_by_region
ADD PARTITION q1_nonmainland VALUES ('HI', 'PR')
STORAGE (INITIAL 20K NEXT 20K) TABLESPACE tbs_3
NOLOGGING;

ALTER TABLE sales ADD PARTITION q1_2000
VALUES LESS THAN (2000, 04, 01) COMPRESS
SUBPARTITIONS 8 STORE IN tbs5;

ALTER TABLE diving MODIFY PARTITION locations_us
ADD SUBPARTITION us_locs5 TABLESPACE us1;

ALTER TABLE quarterly_regional_sales
ADD PARTITION q1_2000 VALUES LESS THAN (TO_DATE('1-APR-2000','DD-
MON-YYYY'))
STORAGE (INITIAL 20K NEXT 20K) TABLESPACE ts3 NOLOGGING
(
SUBPARTITION q1_2000_northwest VALUES ('OR', 'WA'),
SUBPARTITION q1_2000_southwest VALUES ('AZ', 'UT', 'NM'),
SUBPARTITION q1_2000_northeast VALUES ('NY', 'VM', 'NJ'),
SUBPARTITION q1_2000_southeast VALUES ('FL', 'GA'),
SUBPARTITION q1_2000_northcentral VALUES ('SD', 'WI'),
SUBPARTITION q1_2000_southcentral VALUES ('OK', 'TX')
);
-- ----------------------------------------------------------------
ALTER INDEX q1_sales_by_region_locix
MODIFY DEFAULT ATTRIBUTES TABLESPACE tbs_4;
```

```
ALTER INDEX q1_sales_by_region_locix
REBUILD PARTITION q1_nonmainland TABLESPACE tbs_4;

ALTER INDEX hgidx ADD PARTITION p5;

ALTER TABLE ouu1
COALESCE PARTITION;

ALTER TABLE diving MODIFY PARTITION us_locations
COALESCE SUBPARTITION;

ALTER INDEX hgidx COALESCE PARTITION;

1):
ALTER TABLE sales DROP PARTITION dec98;
ALTER INDEX sales_area_ix REBUILD;

ALTER INDEX sales_area_ix REBUILD PARTITION jan99_ix;
ALTER INDEX sales_area_ix REBUILD PARTITION feb99_ix;
ALTER INDEX sales_area_ix REBUILD PARTITION mar99_ix;

2):
DELETE FROM sales WHERE TRANSID < 10000;
ALTER TABLE sales DROP PARTITION dec98;

3):
ALTER TABLE sales DROP PARTITION dec98
UPDATE INDEXES;

-- ---------------------------------
1):
ALTER TABLE sales
DISABLE CONSTRAINT dname_sales1;
ALTER TABLE sales DROP PARTITTION dec98;
ALTER TABLE sales
ENABLE CONSTRAINT dname_sales1;

2):
```

```
DELETE FROM sales WHERE TRANSID < 10000;
ALTER TABLE sales DROP PARTITION dec94;

-- ----------------------------------

ALTER INDEX npr DROP PARTITION P1;
ALTER INDEX npr REBUILD PARTITION P2;

-- ----------------------------------
ALTER TABLE table_name
DISABLE CONSTRAINT constraint_name KEEP INDEX

ALTER TABLE stocks
EXCHANGE PARTITION p3 WITH TABLE stock_table_3;

ALTER TABLE t2 EXCHANGE PARTITION p1 WITH TABLE t1
WITH VALIDATION;

-- ---------------------------------------

ALTER TABLE sales EXCHANGE SUBPARTITION q3_1999_s1
WITH TABLE q3_1999 INCLUDING INDEXES;

-- ----------------------------------------
ALTER TABLE four_seasons
MERGE PARTITIONS quarter_one, quarter_two INTO PARTITION quarter_two
UPDATE INDEXES;

ALTER TABLE four_seasons MODIFY PARTITION
quarter_two REBUILD UNUSABLE LOCAL INDEXES;


ALTER TABLE q1_sales_by_region
MERGE PARTITIONS q1_northcentral, q1_southcentral
INTO PARTITION q1_central
STORAGE(MAXEXTENTS 20);

ALTER TABLE all_seasons
MERGE PARTITIONS quarter_1, quarter_2 INTO PARTITION quarter_2
```

```
SUBPARTITIONS 8;

ALTER TABLE quarterly_regional_sales
MERGE SUBPARTITIONS q1_1999_northwest, q1_1999_southwest
INTO SUBPARTITION q1_1999_west
TABLESPACE ts4;
-- ----------------------------------------
ALTER TABLE emp
MODIFY DEFAULT ATTRIBUTES FOR PARTITION p1 TABLESPACE ts1;

ALTER TABLE scubagear MODIFY PARTITION ts1 UNUSABLE LOCAL INDEXES;

ALTER TABLE dept MODIFY PARTITION p1
REBUILD UNUSABLE LOCAL INDEXES;

ALTER TABLE sales_by_region
MODIFY PARTITION region_south
ADD VALUES ('OK', 'KS');

ALTER TABLE quarterly_regional_sales
MODIFY SUBPARTITION q1_1999_southeast
ADD VALUES ('KS');

ALTER TABLE sales_by_region
MODIFY PARTITION region_south
DROP VALUES ('OK', 'KS');

ALTER TABLE quarterly_regional_sales
MODIFY SUBPARTITION q1_1999_southeast
DROP VALUES ('KS');

-- ---------------------------------------
template:
ALTER TABLE emp_sub_template
SET SUBPARTITION TEMPLATE
(SUBPARTITION e TABLESPACE ts1,
SUBPARTITION f TABLESPACE ts2,
SUBPARTITION g TABLESPACE ts3,
SUBPARTITION h TABLESPACE ts4
```

);

You can drop a subpartition template by specifying an empty list:

```
ALTER TABLE emp_sub_template
SET SUBPARTITION TEMPLATE ( );

-- ---------------------------------------

ALTER TABLE parts MOVE PARTITION depot2
TABLESPACE ts094 NOLOGGING COMPRESS;

-- ----------------------------------------

ALTER TABLE scuba_gear MOVE SUBPARTITION bcd_types
TABLESPACE tbs23 PARALLEL (DEGREE 2);

-- ----------------------------------------

ALTER TABLE scubagear RENAME PARTITION sys_p636 TO tanks;

-- -----------------------------------------

ALTER TABLE sales_by_region
SPLIT PARTITION region_east VALUES ('CT', 'MA', 'MD')
INTO
( PARTITION region_east_1
TABLESPACE tbs2,
PARTITION region_east_2
STORAGE (NEXT 2M PCTINCREASE 25))
PARALLEL 5;

ALTER TABLE sales_by_region
SPLIT PARTITION region_unknown VALUES ('MT', 'WY', 'ID')
INTO
( PARTITION region_wildwest,
PARTITION region_unknown);

ALTER TABLE all_seasons SPLIT PARTITION quarter_1
```

```
AT (TO_DATE('16-dec-1997','dd-mon-yyyy'))
INTO (PARTITION q1_1997_1 SUBPARTITIONS 4 STORE IN (ts1,ts3),
PARTITION q1_1997_2);

ALTER TABLE quarterly_regional_sales SPLIT PARTITION q1_1999
AT (to_date('15-Feb-1999','dd-mon-yyyy'))
INTO ( PARTITION q1_1999_jan_feb
TABLESPACE ts1,
PARTITION q1_1999_feb_mar
STORAGE (NEXT 2M PCTINCREASE 25) TABLESPACE ts2)
PARALLEL 5;

-- ----------------------------------------

ALTER TABLE sales TRUNCATE PARTITION dec98;
ALTER INDEX sales_area_ix REBUILD;
```

2.Perform partition maintenance operations

删除和移动

3.Maintain indexes on a partitioned table

这个一定会考

4.Implement securefile LOB

加载 securefile，LOB 考创建，存储和加载代码构造能力，阅读代码能力。

SecureFiles and Large Objects Developer's Guide=>4 Using Oracle SecureFiles LOBs
Net Services Reference=> 5 Parameters for the sqlnet.ora File
Administrator's Guide=> 13 Managing Tablespaces => Creating Tablespaces=> Encrypted Tablespaces
SQL Language Reference=> 12 SQL Statements: ALTER TABLE to ALTER TABLESPACE
SQL Language Reference=> 16 SQL Statements: CREATE SYNONYM to CREATE TRIGGER

启用

```
mkdir -p /ora/db11g/admin/PROD/wallet
```

vi sqlnet.ora # 最后一行添加

```
WALLET_LOCATION=(SOURCE=(METHOD=FILE)
(METHOD_DATA=(DIRECTORY=/ora/db11g/admin/PROD/wallet/)))
```

```
SQL>ALTER SYSTEM SET ENCRYPTION KEY IDENTIFIED BY "welcome1";
#报错 ORA-28368: cannot auto-create wallet 的话 是目录或者 sqlnet.ora 的问题
```

#SQL> ALTER SYSTEM SET db_securefile = 'ALWAYS'; -- 修改模式

ALWAYS：尝试将所有 LOB 创建为 SecureFile LOB，但是仅可将自动段空间管理(ASSM)表空间外的任何 LOB 创建为 BasicFile LOB	FORCE：强制将所有 LOB 创建为 SecureFile LOB	PERMITTED：允许创建 SecureFiles（默认值）	NEVER：禁止创建 SecureFiles	IGNORE：禁止创建 SecureFiles，并忽略使用 SecureFiles 选项强制创建 BasicFiles 而导致的任何错误

-- 创建 securefile
CREATE TABLE t1 (a CLOB ENCRYPT **IDENTIFIED BY foo**) LOB(a) STORE AS SECUREFILE (CACHE); --指定密码的 securefile
CREATE TABLE lob_1(id number, doc CLOB **ENCRYPT USING 'AES128'**) LOB(doc) STORE AS **SECUREFILE (DEDUPLICATE LOB)**;
CREATE TABLE lob_2 (id number, doc CLOB) LOB(doc) STORE AS**SECUREFILE(COMPRESS HIGH KEEP_DUPLICATES)** ;
-- 由于创建好的 securefile 未创建相应的 LOB 段，无法看到已经启用 securefile 我们插入几条数据后才能看到加密段
insert into lob_1 values(1,'asdfgjkwqetui2rtyu');
insert into lob_2 values(1,'asdfgjkwqetui2rtyu');
commit;

--此时可以看到数据已经有 securefile 加密的 LOB 段了
SELECT segment_name, segment_type, segment_subtype FROM user_segments where segment_type like 'LOB%';

```
SEGMENT_NAME                           SEGMENT_TYPE        SEGMENT_SU
-------------------------------------- ------------------- ----------
SYS_IL0000107267C00002$$               LOBINDEX            ASSM
SYS_IL0000107270C00002$$               LOBINDEX            ASSM
SYS_LOB0000107267C00002$$              LOBSEGMENT          SECUREFILE
SYS_LOB0000107270C00002$$              LOBSEGMENT          SECUREFILE
```

注意：索引不被加密

delete lob_1; delete lob_2; commit;-- 删除数据后段仍然存在	truncate table lob_1; truncate table lob_2; --清空表后加密段还健在	drop table lob_1 purge; drop table lob_2 purge; -- 这样就删没了

ALTER TABLE t1 MODIFY LOB(a) (**KEEP_DUPLICATES**); #禁用取消重复
ALTER TABLE t1 MODIFY **PARTITION p1 LOB(a) (DEDUPLICATE LOB)**; #启用分区取消重复

ALTER TABLE t1 MODIFY LOB(a) (**NOCOMPRESS**); #禁用压缩
ALTER TABLE t1 MODIFY **PARTITION p1 LOB(a) (COMPRESS HIGH)**; #对单一分区中的 SecureFiles 启用压缩

```
ALTER TABLE t1 MODIFY (a CLOB ENCRYPT USING '3DES168'); #使用 3DES168 启用加密
ALTER TABLE t1 MODIFY PARTITION p1( LOB(a) ( ENCRYPT ); #对分区启用加密
ALTER TABLE t1 MODIFY (a CLOB ENCRYPT IDENTIFIED BY ghYtp); #使用口令启用加密并构建加密密钥
```

SecureFiles 迁移：示例

```
CREATE TABLE cust(c_id NUMBER PRIMARY KEY,c_zip NUMBER,c_name
VARCHAR(30) DEFAULT NULL,c_lob CLOB);
INSERT INTO cust VALUES(1, 94065, 'hhh', 'ttt');
commit;
```

```
CREATE TABLE cust_int(c_id NUMBER NOT NULL,c_zip NUMBER,c_name
VARCHAR(30) DEFAULT NULL,c_lob CLOB) LOB(c_lob) STORE AS SECUREFILE
(NOCACHE FILESYSTEM_LIKE_LOGGING);
```

```
-- 注意修改文档中 owner 的部分
DECLARE
col_mapping VARCHAR2(1000);
BEGIN
-- map all the columns in the interim table to the original table
col_mapping :='c_id c_id , '||'c_zip c_zip , '||'c_name c_name, '||'c_lob c_lob';
DBMS_REDEFINITION.START_REDEF_TABLE('SH', 'cust', 'cust_int',
col_mapping);
END;
/
DECLARE
error_count pls_integer := 0;
BEGIN
DBMS_REDEFINITION.COPY_TABLE_DEPENDENTS('SH', 'cust', 'cust_int', 1,
TRUE,TRUE,TRUE,FALSE, error_count);
DBMS_OUTPUT.PUT_LINE('errors := ' || TO_CHAR(error_count));
END;
/
EXEC DBMS_REDEFINITION.FINISH_REDEF_TABLE('sh', 'cust', 'cust_int');
```

```
select table_name , column_name ,segment_name,SECUREFILE  from
user_lobs;
```

alter table aa move lob(xxx) store as securefile ;

如果是分区表：

alter table tp3 move partition p1 tablespace users lob(cc) store as(tablespace users);

只改变某一个区的大对象的表空间：

alter table tp3 move partition p1 lob(cc) store as(tablespace users);

5.Create and manage LOB segments

```
declare
 l_clob clob;
```

```
 l_bfile bfile;
begin
  insert into aa(clob1) values (empty_clob()) return clob1 into l_clob;
  l_bfile := bfilename('DIR1', 'c1.mp3');
  dbms_lob.fileopen(l_bfile);
  dbms_lob.loadfromfile(l_clob, l_bfile, dbms_lob.getlength(l_bfile));
  dbms_lob.fileclose(l_bfile);
end;
```

用 SQL*Loader 加载 LOB 型数据
控制文件如下：

```
load data
infile *
into table aa
TRAILING NULLCOLS
(aaa filler char(200),clob1 lobfile(aaa) terminated by eof)
begindata
c1.mp3
```

注意，clob1 lobfile(aaa)，aaa 是对前一列的引用，但此处并不需要加:号。这是 lobfile（）函数的特殊要求，其他地方不可以这样使用。

这种方式 CLOB 和 BLOB 一样适用，不需作任何更改。

6.Implement fine-grained access control

context 可能没考

7.Create and manage contexts

Command-Line Solutions for Practice 4-4: Create an Application Context

In this practice, you create an application context, set the context using a secure package, and test the context.

1) Connect as `SYSTEM/ORACLE@orcl`. Using the `SYS_CONTEXT` procedure, display the following session-related attributes:

   `CURRENT_USER`

   `SESSION_USER`

   `PROXY_USER`

   `IP_ADDRESS`

   `NETWORK_PROTOCOL`

   `AUTHENTICATION_TYPE`

   `AUTHENTICATION_DATA`

   `CLIENT_IDENTIFIER`

   You can use one of the following techniques to call `SYS_CONTEXT`:

```
SELECT sys_context('userenv','…')FROM dual;

EXEC dbms_output.put_line(syscontext('userenv','…'));
```

   (Make sure to issue `SET SERVEROUTPUT ON` before executing `DBMS_OUTPUT`.)

   Answer:

   Both ways of viewing `userenv` context attributes are shown in the following solution.

```
$ sqlplus /nolog

SQL*Plus: Release 10.2.0.2.0 - Production on Tue Jan 20 11:36:28
2009

Copyright (c) 1982, 2005, Oracle. All Rights Reserved.
SQL> SET ECHO ON
SQL> CONNECT system/oracle@orcl
Connected.

SQL> SET SERVEROUTPUT ON

SQL> EXEC dbms_output.put_line(sys_context('USERENV',
'CURRENT_USER'));
SYSTEM

PL/SQL procedure successfully completed.

SQL> EXEC dbms_output.put_line(sys_context('USERENV',
'SESSION_USER'));
SYSTEM
```

```
PL/SQL procedure successfully completed.

SQL> EXEC dbms_output.put_line(sys_context('USERENV',
'IP_ADDRESS'));
10.190.112.211

PL/SQL procedure successfully completed.

SQL> EXEC dbms_output.put_line(sys_context('USERENV',
'NETWORK_PROTOCOL'));
tcp

PL/SQL procedure successfully completed.

SQL> EXEC dbms_output.put_line(sys_context('USERENV',
'AUTHENTICATION_TYPE'));
DATABASE

PL/SQL procedure successfully completed.

SQL> SELECT sys_context('USERENV', 'CURRENT_USER') FROM DUAL;

SYS_CONTEXT('USERENV','CURRENT_USER')
----------------------------------------------------------
SYSTEM

SQL> SELECT sys_context('USERENV', 'SESSION_USER') FROM DUAL;

SYS_CONTEXT('USERENV','SESSION_USER')
----------------------------------------------------------
SYSTEM

SQL> SELECT sys_context('USERENV', 'IP_ADDRESS') FROM DUAL;

SYS_CONTEXT('USERENV','IP_ADDRESS')
----------------------------------------------------------
10.190.112.211

SQL> SELECT sys_context('USERENV', 'NETWORK_PROTOCOL') FROM DUAL;

SYS_CONTEXT('USERENV','NETWORK_PROTOCOL')
----------------------------------------------------------
tcp

SQL> SELECT sys_context('USERENV', 'AUTHENTICATION_TYPE') FROM DUAL;

SYS_CONTEXT('USERENV','AUTHENTICATION_TYPE')
----------------------------------------------------------
DATABASE
```

2) Implement a local application context with the following properties:

   Name: EMP_USER

   Owned by: SEC

   Contains the following attributes, which are listed with the column from the HR.EMPLOYEES table that is used to obtain the attribute value:

   Attribute Column from HR.EMPLOYEES

   ID EMPLOYEE_ID

   NAME FIRST_NAME || ' ' || LAST_NAME

   EMAIL EMAIL

Answer:

```
SQL> CONNECT sec/sec
Connected.
SQL> CREATE CONTEXT emp_user USING current_emp;

Context created.
```

3) The row in the EMPLOYEES table that is used to populate the attributes is selected by comparing the EMAIL column to the SESSION_USER attribute from SYS_CONTEXT.

   The procedure that sets the application context has the following properties:

   Owned by: SEC user

   Part of: CURRENT_EMP package

   Name: SET_EMP_INFO

   Called from a logon trigger named EMP_LOGON that is also owned by SEC. This trigger applies to all users.

You re-create a modified version of this package and context in a later practice, so save all your work. If you are not familiar with creating packages in PL/SQL, execute $HOME/labs/lab_04_04_03_a.sql to create the package and package body. If you are not familiar with creating logon triggers in PL/SQL, execute $HOME/labs/lab_04_04_03_b.sql to create the logon trigger.

Answer:

```
SQL> @lab_04_04_03_a.sql
SQL> CREATE OR REPLACE PACKAGE current_emp IS
  2    PROCEDURE set_emp_info;
  3  END;
  4  /

Package created.

SQL> CREATE OR REPLACE PACKAGE BODY current_emp IS
```

```
  2    PROCEDURE set_emp_info
  3    IS
  4      v_employee_id       hr.employees.employee_id%TYPE;
  5      v_first_name        hr.employees.first_name%TYPE;
  6      v_last_name         hr.employees.last_name%TYPE;
  7    BEGIN
  8      SELECT employee_id,
  9             first_name,
 10             last_name
 11        INTO v_employee_id,
 12             v_first_name,
 13             v_last_name
 14        FROM hr.employees
 15        WHERE email = SYS_CONTEXT('USERENV', 'SESSION_USER');
 16      DBMS_SESSION.SET_CONTEXT('emp_user', 'id',
 17        v_employee_id);
 18      DBMS_SESSION.SET_CONTEXT('emp_user', 'name',
 19        v_first_name || ' ' || v_last_name);
 20      DBMS_SESSION.SET_CONTEXT('emp_user', 'email',
 21        SYS_CONTEXT('USERENV', 'SESSION_USER'));
 22    EXCEPTION
 23      WHEN no_data_found THEN NULL;
 24    END;
 25  END;
 26  /

Package body created.

SQL> @lab_04_04_03_b.sql
SQL> CREATE or REPLACE TRIGGER emp_logon
  2  AFTER LOGON ON DATABASE
  3  BEGIN
  4    current_emp.set_emp_info;
  5  END;
  6  /

Trigger created.
```

4) Test the context that you created by performing these steps:

   a) Create a user named `SKING` with the `CREATE SESSION` privilege.

   b) Log in as `SKING`.

   c) Use `SYS_CONTEXT` to verify that the `EMP_USER` context attributes are set. If you use `DBMS_OUTPUT`, remember to issue the `SET SERVEROUTPUT ON` command.

Answer:

```
SQL> CONNECT sec/sec
Connected.
SQL>
```

```
SQL> GRANT create session TO sking IDENTIFIED BY sking;

Grant succeeded.

SQL>
SQL> CONNECT sking/sking
Connected.
SQL>
SQL> SET SERVEROUTPUT ON
SQL>
SQL> EXEC dbms_output.put_line(sys_context('emp_user', 'id'));
100

PL/SQL procedure successfully completed.

SQL> EXEC dbms_output.put_line(sys_context('emp_user', 'name'));
Steven King

PL/SQL procedure successfully completed.

SQL> EXEC dbms_output.put_line(sys_context('emp_user', 'email'));
SKING

PL/SQL procedure successfully completed.
```

5) Review `lab_04_04_05.sql`, connect as `SKING`, and execute the script, which lists all the application context attributes set in the current session. Because Label Security is installed, the `LBAC$LABELS LBAC$LASTSEQ` attributes are part of the context but not populated because Label Security is not yet configured.

Answer:

```
SQL> connect sking/sking
Connected.
SQL> @lab_04_04_05.sql
SQL> SET ECHO OFF
SQL> SET SERVEROUTPUT ON
SQL>
SQL> DECLARE
  2    list dbms_session.AppCtxTabTyp;
  3    cnt number;
  4  BEGIN
  5    dbms_session.list_context (list, cnt);
  6    IF cnt = 0
  7      THEN dbms_output.put_line('No contexts active.');
  8      ELSE
  9        FOR i IN 1..cnt LOOP
 10          dbms_output.put_line(list(i).namespace
 11            ||' ' || list(i).attribute
 12            || ' = ' || list(i).value);
 13        END LOOP;
 14    END IF;
```

```
 15  END;
 16  /
EMP_USER NAME = Steven King
EMP_USER EMAIL = SKING
EMP_USER ID = 100
LBAC$LABELS LBAC$LASTSEQ =

PL/SQL procedure successfully completed.

SQL>
```

6) Log in as `SEC` and select information about the application context that you created from the data dictionary.

Answer:

```
SQL> CONNECT sec/sec
Connected.
SQL>
SQL> SELECT *
  2    FROM dba_context
  3    WHERE namespace = 'EMP_USER';

NAMESPACE                      SCHEMA
------------------------------ ----------------------------
PACKAGE                        TYPE
------------------------------ ----------------------
EMP_USER                       SEC
CURRENT_EMP                    ACCESSED LOCALLY
```

7) What happens when you call `DBMS_SESSION.SET_CONTEXT` to set an attribute in the `EMP_USER` context? Assume that `SKING` wants to change the context setting.

Answer:

Because the application context is set with a package, `SKING` does not have sufficient privileges to execute the `DBMS_SESSION.SET_CONTEXT` procedure.

```
SQL> connect sking/sking
Connected.
SQL> @lab_04_04_05.sql
EMP_USER NAME = Steven King
EMP_USER EMAIL = SKING
EMP_USER ID = 100
LBAC$LABELS LBAC$LASTSEQ =

PL/SQL procedure successfully completed.

SQL> EXEC DBMS_SESSION.SET_CONTEXT('emp_user', 'id', 1);
BEGIN DBMS_SESSION.SET_CONTEXT('emp_user', 'id', 1); END;
```

```
*
ERROR at line 1:
ORA-01031: insufficient privileges
ORA-06512: at "SYS.DBMS_SESSION", line 82
ORA-06512: at line 1


SQL> @lab_04_04_05.sql
EMP_USER NAME = Steven King
EMP_USER EMAIL = SKING
EMP_USER ID = 100
LBAC$LABELS LBAC$LASTSEQ =

PL/SQL procedure successfully completed.
```

Command-Line Solutions for Practice 4-5: Implement a Fine-Grained Access Control Policy

In this practice you create, enable, and test a Fine-Grained Access Control policy.

1) How does FGAC determine which rows belong in the VPD for the current user?

   Answer:

   FGAC adds a predicate (condition) to the `WHERE` clause on a `SELECT` or DML statement with an `AND` operator.

2) How does FGAC know which tables are defined in the VPD?

   Answer:

   You include a table name or view name when the FGAC policy is created.

3) The `SEC` user also needs the privilege to create policies. As `SYSTEM`, grant `SEC` the ability to execute the package that creates policies.

   Answer:

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL>
SQL> GRANT execute ON dbms_rls TO sec;

Grant succeeded
```

4) What privilege exempts the user from access policies? Why does the `SEC` user need this privilege? Grant it to `SEC`.

   Answer:

   The `EXEMPT ACCESS POLICY` privilege is very powerful. Statements that are issued by a user with this privilege do not have any FGAC policies applied.

```
SQL> CONNECT system/oracle
Connected.
SQL>
SQL> GRANT exempt access policy TO sec;

Grant succeeded
```

5) The `lab_04_05_05.sql` script creates the package used by the security policy to return a predicate.

   a) Review and execute the script.

   Answer:

```
SQL> @lab_04_05_05.sql
SQL> set echo off
SQL> CONNECT sec/sec
Connected.
SQL>
SQL> CREATE OR REPLACE PACKAGE hr_policy_pkg IS
```

```
  2    FUNCTION limit_emp_emp (
  3      object_schema       IN      VARCHAR2,
  4      object_name         VARCHAR2 )
  5      RETURN VARCHAR2;
  6  END;
  7  /

Package created.

SQL>
SQL> CREATE OR REPLACE PACKAGE BODY hr_policy_pkg IS
  2    FUNCTION limit_emp_emp (
  3      object_schema       IN      VARCHAR2,
  4      object_name         VARCHAR2 )
  5      RETURN VARCHAR2
  6    IS
  7      v_emp_id NUMBER;
  8    BEGIN
  9      RETURN 'employee_id = SYS_CONTEXT(''emp_user'', ''id'')';
 10    END;
 11  END;
 12  /

Package body created.
```

b) What predicate does the policy use to limit the rows returned from the `EMPLOYEE` table?

Answer:

```
employee_id = SYS_CONTEXT('emp_user', 'id')
```

c) How does this predicate limit the rows?

Answer:

The user making the query must have an `EMAIL_ID` that matches the database username, and the `emp_user` attribute in the `syscontext` is set equal to the `employee_id` of the user (see Practice 4-4, step 4). The predicate only allows the user to access the record describing the user.

6) Test the policy function.

Answer:

```
SQL> CONNECT sec/sec
Connected.
SQL>
SQL> SELECT hr_policy_pkg.limit_emp_emp('a', 'b') FROM DUAL;

HR_POLICY_PKG.LIMIT_EMP_EMP('A','B')
------------------------------------------------------------
employee_id = SYS_CONTEXT('emp_user', 'id')
```

7) Implement a policy with the following characteristics:

The policy limits the rows that are selected from the `HR.EMPLOYEES` table.

The policy is named `HR_EMP_POL`.

The function that is used to return a predicate is `SEC.HR_POLICY_PKG.LIMIT_EMP_EMP`.

Answer:

```
SQL> CONNECT sec/sec
Connected.
SQL>
SQL> EXEC dbms_rls.drop_policy('HR', 'EMPLOYEES', 'HR_EMP_POL');
BEGIN dbms_rls.drop_policy('HR', 'EMPLOYEES', 'HR_EMP_POL'); END;

*
ERROR at line 1:
ORA-28102: policy does not exist
ORA-06512: at "SYS.DBMS_RLS", line 59
ORA-06512: at line 1

SQL> EXEC dbms_rls.add_policy ('HR', 'EMPLOYEES', 'HR_EMP_POL', -
>   'SEC', 'HR_POLICY_PKG.LIMIT_EMP_EMP', 'SELECT')

PL/SQL procedure successfully completed.
```

8) Set up the `SKING` user so that he can access the `HR.EMPLOYEES` table.

Answer:

```
SQL> connect hr/hr
Connected.
SQL> GRANT select ON employees TO sking;

Grant succeeded.

SQL>
```

9) As `SKING`, execute the `lab_04_04_05.sql` script, which displays the current context attributes.

Answer:

```
SQL> connect sking/sking
Connected.
SQL> @lab_04_04_05
EMP_USER NAME = Steven King
EMP_USER EMAIL = SKING
EMP_USER ID = 100
LBAC$LABELS LBAC$LASTSEQ =

PL/SQL procedure successfully completed.
```

10) Which rows are returned when SKING queries the HR.EMPLOYEE table without a WHERE clause? Try it.

Answer:

```
SQL> select employee_id, first_name, last_name, email
  2  from HR.EMPLOYEES;

EMPLOYEE_ID FIRST_NAME           LAST_NAME
----------- -------------------- -------------------------
EMAIL
-------------------------
        100 Steven               King
SKING
```

11) As the SEC user, delete the FGAC policy just created.

Answer:

```
SQL> CONNECT sec/sec
Connected.
SQL>
SQL> EXEC dbms_rls.drop_policy('HR', 'EMPLOYEES', 'HR_EMP_POL');

PL/SQL procedure successfully completed.
```

```
begin
dbms_rls.add_policy
(
object_schema => 'HOSPITAL',
object_name => 'PATIENTS',
policy_name => 'PATIENT_VIEW_POLICY',
policy_function => 'GET_DOCTOR_ID',
function_schema => 'HOSPITAL',
statement_types => 'SELECT, INSERT, UPDATE, DELETE',
update_check => true,
enable => true
);
end;

-- ---------------------------------

create or replace function get_doctor_id
(
p_schema_name in varchar2,
p_table_name in varchar2
)
return varchar2
is
l_doctor_id number;
begin
select doctor_id
into l_doctor_id
from doctors
where doctor_name = USER;
return 'doctor_id = '||l_doctor_id;
end;

-- -------------------------------------

-- ------------------------
```

The user’s original query

```
SELECT * FROM PATIENTS
```

was rewritten to

```
SELECT * FROM
(SELECT * FROM PATIENTS)
```

```
WHERE DOCTOR_ID = 1
/

DROP CONTEXT vpd_context;
CREATE CONTEXT vpd_context USING ap.app_security_context;

-- CREATE OR REPLACE PACKAGE ap.app_security_context
PROCEDURE setuserinfo;

DBMS_SESSION.SET_CONTEXT('vpd_context', 'cc_limit', cclimit);
DBMS_SESSION.SET_CONTEXT('vpd_context', 'crd_memo', crdmemo);

-- ------------------------------------------------

CREATE OR REPLACE TRIGGER ap.on_logon
AFTER LOGON
ON DATABASE
```

※※※※※※※※※※※※※※※※※※※※※※※※※※※※※※※※※※※

VPD

1、VPLD 命令：文档：Packages and Types Reference 搜索关键字：dbms_rls

2、VPLD：文档：Security Guide 搜索关键字：VPD

※※※※※※※※※※※※※※※※※※※※※※※※※※※※※※※※※※※

一、VPD

```
CREATE OR REPLACE function vpd1( object_schema IN VARCHAR2, object_name
VARCHAR2)
return varchar2
as
begin
	return 'fg2>=95';
end;
/

BEGIN
dbms_rls.add_policy(object_schema => 'u01',
object_name => 't2',
policy_name => 'hr_policy',
function_schema =>'u01',
policy_function => 'vpd1',
statement_types =>'select,insert,update,delete',
sec_relevant_cols=>'fg1,id');
END;
/
```

上面是一个例子，注意，相关列可以和条件无关。
SQL> update t2 set fg2=79 where id=11;
0 rows updated.
更新 0 行，这是因为 VPD 的条件被附加到了 id=11 之后。
删除也一样。
SQL> delete t2 where id=11;
0 rows deleted.

但插入 fg2 列不小于等于 95 的行是可以成功的。

如果把函数中的条件改成和相关列有关的：
```
CREATE OR REPLACE function vpd1( object_schema IN VARCHAR2, object_name
VARCHAR2)
return varchar2
as
begin
        return 'id>=15';
end;
/
```

SQL> insert into t2 values(11,'xyz',39,89);

1 row created.

SQL> select * from t2;

no rows selected
插入照样可以成功。但显示就没有行了，因为所有的行都被 VPD 过滤了。

5 种类型：
– DBMS_RLS.STATIC
– DBMS_RLS.SHARED_STATIC
– DBMS_RLS.CONTEXT_SENSITIVE
– DBMS_RLS.SHARED_CONTEXT_SENSITIVE
– DBMS_RLS.DYNAMIC

8.Administer flashback data archive and schema evolution

一定会考
以 HR 用户身份，选择在创建闪回数据归档之后到执行错误 DML 之前这段时间内的一个时间。要查看 Fox 先生在该时间的雇员记录，请执行以下查询（用所选历史日期替换 '10' MINUTE，示例格式如下：‘50’ SECOND, ‘10’ DAY,

‘5’ MONTH）：
注：如果指定了启动闪回数据归档之前的时间，将出现 ORA-1466 错误。缩短时间间隔，然后重试。如果仍然看到薪金为 12600，请增加时间间隔。

```
SELECT employee_id, last_name, salary
FROM hr.employees AS OF TIMESTAMP
(SYSTIMESTAMP - INTERVAL '10' MINUTE)
WHERE last_name = 'Fox';
ALTER FLASHBACK ARCHIVE fla1 MODIFY RETENTION 2 YEAR;
```

Oracle® Tutor™

升级考试 Section3 数据和数据仓库管理（Data and Data Warehouse Management）

很难答完

1.Troubleshoot fast materialized views to fast refresh and query rewrite

2.Add a tablespace by using Transportable Tablespace Feature of Data Pump Import (cross platform transportable tablespace)

3.Configure a schema to support a star transformation query

4.Configure and use parallel execution for queries

5.Use and access SecureFile LOBS

6.Create partitioned tables (includes reference and interval partitioning)

7.Configure Flashback Data Archive

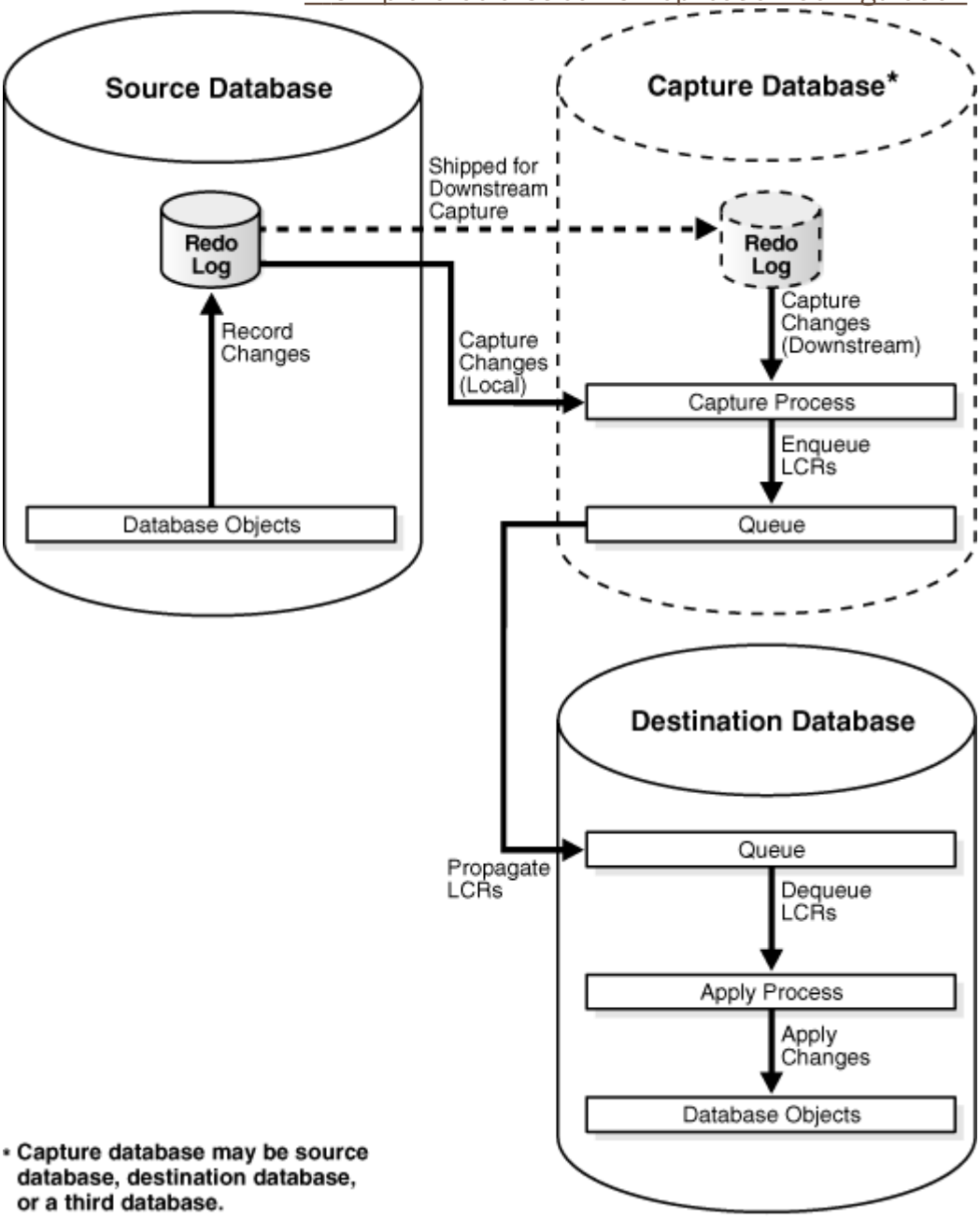
8.Use Oracle Streams to capture and propagate changes in a table

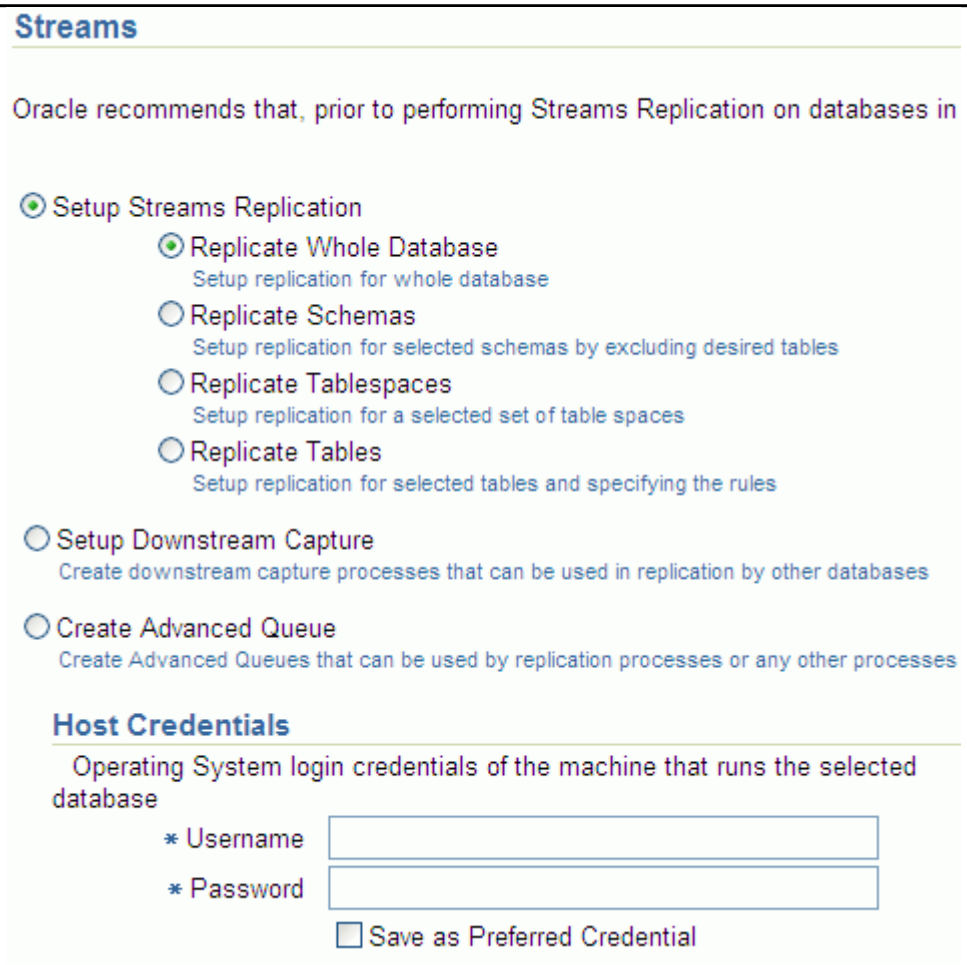
stream 有函数转换需求

Streams Advanced Queuing User's Guide

Streams Replication Administrator's Guide=> 1 Preparing for Oracle Streams Replication

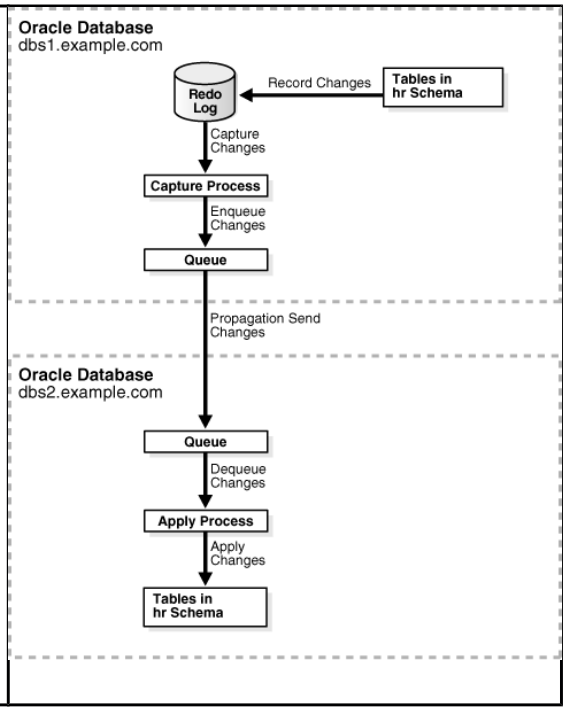
=> 2 Simple Oracle Streams Replication Configuration





Stream 捕捉 LOG 中的变化进行记录并保存在高级队列中进行发布，目标库去队列中取得相应数据并且到本地库进行重做。可以实现整个数据库以及指定用户、指定表空间、指定表的流复制。需要配置三种角色 1、数据捕捉者 2、数据分发者 3、订阅者。

考纲说了 Use Oracle Streams to capture and propagate changes in a table 只考虑表 stream 就行了
建议 GC，手工是在太麻烦了



1、Create Streams Administrator

Create Streams Administrator

Select the databases, on which you want to create Streams Administrator, and enter the DBA crede
The DBA User for creating Streams Administrator should be an existing Administrator with SYSDBA
have the same DBA user and password, and the tablespace entered for the Streams Administrator

Credentials

DBA Username	system
DBA Password	••••••
Streams Administrator Username	repadmin
Streams Administrator Password	••••••
Tablespace	users

It is not recommended that Streams Administrator use the SYSTEM tablespace.

List of Databases

Remove | Add

Select All | Select None

Select	Target Name	Target Version	Host Name	Target Type
☐	PROD	11.2.0.3.0	edu1	oracle_database
☐	ST	11.2.0.3.0	edu1	oracle_database

2、Connfig Replicate tables **#注意，这里需要使用 repadmin 登陆才可以要不然做不成功**

Streams

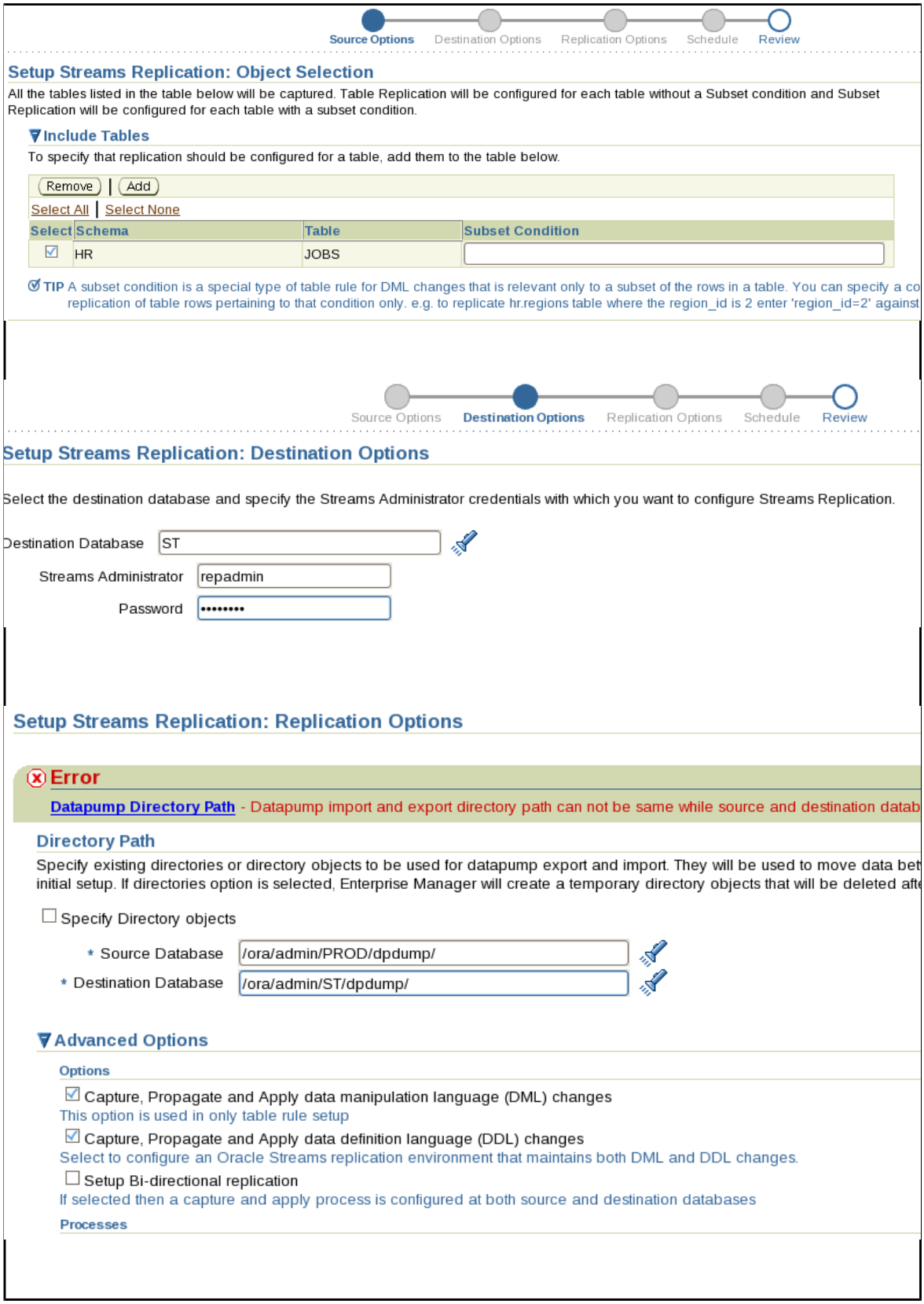
⚠ Warning

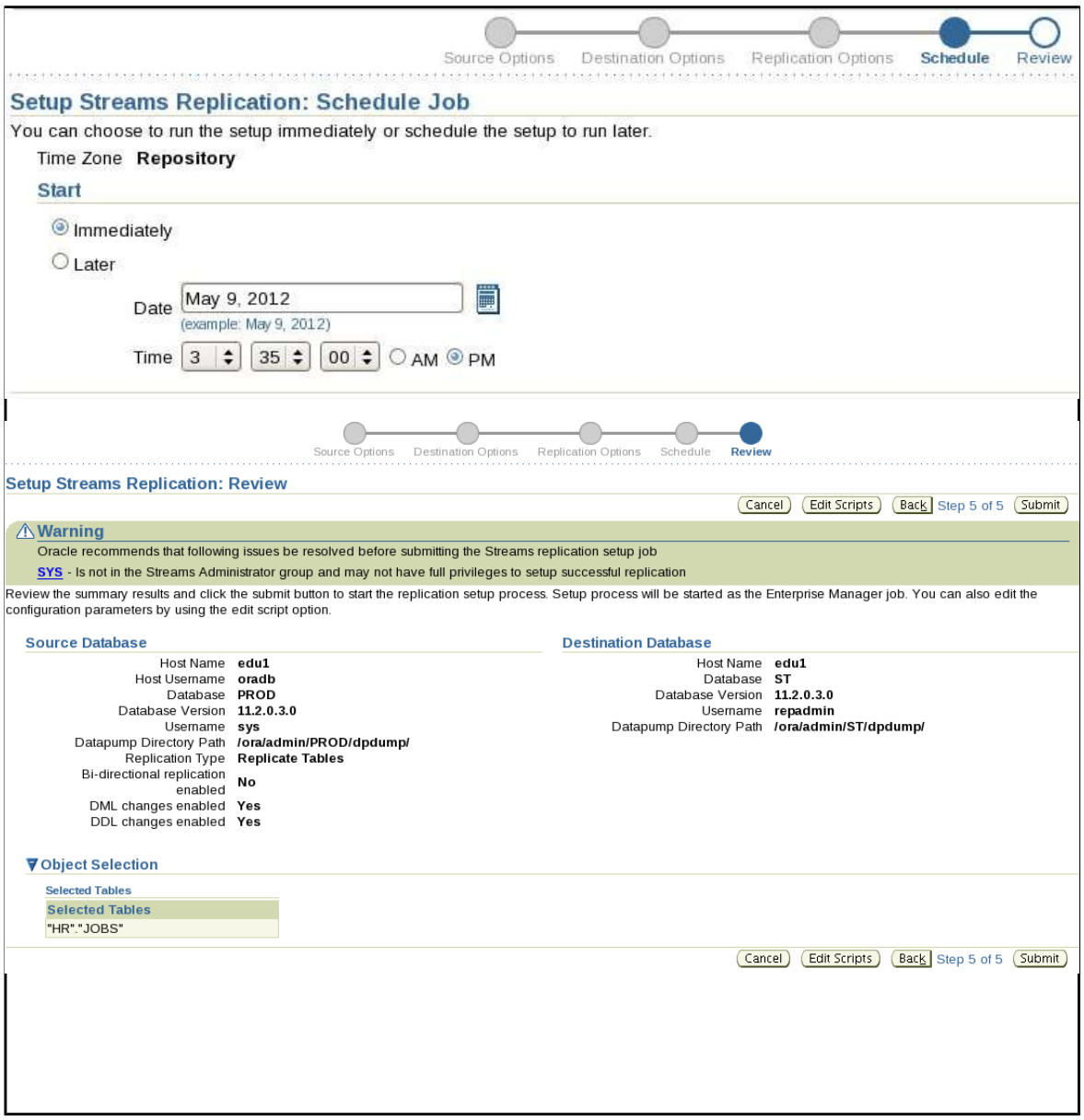
SYS - Is not in the Streams Administrator group and may not have full privileges to setup succe

Oracle recommends that, prior to performing Streams Replication on databases in your configuratio

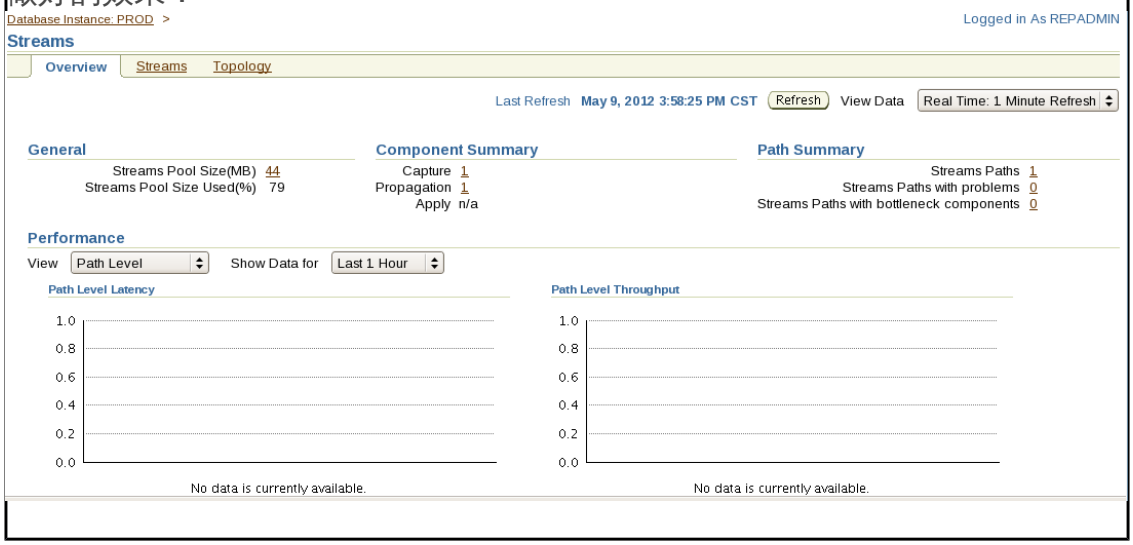
◉ Setup Streams Replication

- ○ Replicate Whole Database
  Setup replication for whole database
- ○ Replicate Schemas
  Setup replication for selected schemas by excluding desired tables
- ○ Replicate Tablespaces
  Setup replication for a selected set of table spaces
- ◉ Replicate Tables
  Setup replication for selected tables and specifying the rules


Source Options
Destination Options
Replication Options
Schedule
Review
Setup Streams Replication: Object Selection
All the tables listed in the table below will be captured. Table Replication will be configured for each table without a Subset condition and Subset Replication will be configured for each table with a subset condition.
Include Tables
To specify that replication should be configured for a table, add them to the table below.
Remove
Add
Select All
Select None
Select
Schema
Table
Subset Condition
HR
JOBS
TIP A subset condition is a special type of table rule for DML changes that is relevant only to a subset of the rows in a table. You can specify a co
replication of table rows pertaining to that condition only. e.g. to replicate hr.regions table where the region_id is 2 enter 'region_id=2' against
Source Options
Destination Options
Replication Options
Schedule
Review
Setup Streams Replication: Destination Options
Select the destination database and specify the Streams Administrator credentials with which you want to configure Streams Replication.
Destination Database
ST
Streams Administrator
repadmin
Password
Setup Streams Replication: Replication Options
Error
Datapump Directory Path - Datapump import and export directory path can not be same while source and destination datab
Directory Path
Specify existing directories or directory objects to be used for datapump export and import. They will be used to move data bet
initial setup. If directories option is selected, Enterprise Manager will create a temporary directory objects that will be deleted aft
Specify Directory objects
* Source Database
/ora/admin/PROD/dpdump/
* Destination Database
/ora/admin/ST/dpdump/
Advanced Options
Options
Capture, Propagate and Apply data manipulation language (DML) changes
This option is used in only table rule setup
Capture, Propagate and Apply data definition language (DDL) changes
Select to configure an Oracle Streams replication environment that maintains both DML and DDL changes.
Setup Bi-directional replication
If selected then a capture and apply process is configured at both source and destination databases
Processes


Source Options
Destination Options
Replication Options
Schedule
Review
Setup Streams Replication: Schedule Job
You can choose to run the setup immediately or schedule the setup to run later.
Time Zone Repository
Start
Immediately
Later
Date May 9, 2012
(example: May 9, 2012)
Time 3 35 00 AM PM
Source Options
Destination Options
Replication Options
Schedule
Review
Setup Streams Replication: Review
Cancel
Edit Scripts
Back
Step 5 of 5
Submit
Warning
Oracle recommends that following issues be resolved before submitting the Streams replication setup job
SYS - Is not in the Streams Administrator group and may not have full privileges to setup successful replication
Review the summary results and click the submit button to start the replication setup process. Setup process will be started as the Enterprise Manager job. You can also edit the configuration parameters by using the edit script option.
Source Database
Host Name edu1
Host Username oradb
Database PROD
Database Version 11.2.0.3.0
Username sys
Datapump Directory Path /ora/admin/PROD/dpdump/
Replication Type Replicate Tables
Bi-directional replication enabled No
DML changes enabled Yes
DDL changes enabled Yes
Destination Database
Host Name edu1
Database ST
Database Version 11.2.0.3.0
Username repadmin
Datapump Directory Path /ora/admin/ST/dpdump/
Object Selection
Selected Tables
Selected Tables
"HR"."JOBS"
Cancel
Edit Scripts
Back
Step 5 of 5
Submit

做好的效果：


Database Instance: PROD >
Logged in As REPADMIN
Streams
Overview
Streams
Topology
Last Refresh May 9, 2012 3:58:25 PM CST
Refresh
View Data
Real Time: 1 Minute Refresh
General
Streams Pool Size(MB) 44
Streams Pool Size Used(%) 79
Component Summary
Capture 1
Propagation 1
Apply n/a
Path Summary
Streams Paths 1
Streams Paths with problems 0
Streams Paths with bottleneck components 0
Performance
View Path Level
Show Data for Last 1 Hour
Path Level Latency
1.0
0.8
0.6
0.4
0.2
0.0
No data is currently available.
Path Level Throughput
1.0
0.8
0.6
0.4
0.2
0.0
No data is currently available.