A Blueprint for Database Design Development and Administration

HANDS ON PRACTICE WORKBOOK 3

Ekokobe, Imgard
KUDRANT ACADEMY

Table of Contents

Read Me First

Implementing the fundamental principles of Database Design, Development and Administration.

The environment for databases is often transparent to the end-users. DBA's, however, must understand the hardware, database software, network configuration, directory structures amongst other components to effectively administer their databases.

- You have learned the fundamentals of relational database systems.
- You have learned how to design, create ERDs, normalize the design, create objects, insert, load and retrieve data.
- You have learned how to properly layout an oracle database on disk for maximum performance and reliability.
- You have learned how to install the oracle software, create databases, configure networking components, create and manage users and their objects.
- You have learned the different modes of starting and stopping an oracle database and when to use them.
- You have learned how to administer a database by loading bulk data, performing backup and recovery operations, monitor user activity and processes and monitor space.
- You have learned how to identify performance issues and resolve them.
- You have learned how to upgrade and patch an oracle database.
- You have learned how to use OEM cloud control to monitor and administer databases.

Now what?

The exercises in this workbook are designed to simulate the knowledge and skills that are used in industry to design, develop and administer databases. You will PULL FROM THIS KNOWLEDGE and implement what you have learned to the assignments laid out in this workbook.

Following directions is **KEY** in this project.

Objectives

- Install the Oracle software.
- Create a database.
- Create DBA accounts.
- Design and create objects
- Use scripts to create all objects.
- Create tablespaces on disks.
- Create tables in the tablespaces.
- Populate tables with test data.
- Backup the database using data pump export and RMAN.
- Create user accounts for end users.
- Collect statistics on how your instance is running
- Manually add and delete columns from tables.
- Monitor space utilization.
- Monitor the alert log.
- Monitor user activity.
- Test the recovery capability.
- Startup and shutdown the instance after changes are made.
- Set up cron jobs that can run continuously.
- Upgrade and patch a database.

Tools Used

Databases are carefully planned and designed after gathering and analyzing user requirements from stakeholders.

After the requirements, have been gathered and defined, a model will be created for the database.

This model will be used to use to create the Entity Relationship Diagrams (ERD) and these ERDs will then be translated into Data Definition Languages (DDL).

The DDL script will be used to create your database tables and the DML script will be used to load some data into your tables. You can also use tools such as SQL loader, data pump import to load bulk data into the database.

After the database has been populated with data, you will perform certain tasks to ensure that the database is available to the end user and can be recoverability in event of data loss.

These are some tools to help you design, develop and administer databases.

- An editor (notepad in your case).
- SQL*Plus, and capabilities to print your source code and capture the output.
- Putty
- Winscp
- ER Assistant tool.
- DBCA
- DBUA
- NETCA
- LSNRCTL
- Data pump export/import
- SQL Loader
- AWR
- ADDM
- SQL Developer
- Oracle Enterprise Manager Cloud Control 13C1

Task 1 – Create ERD

The Entity-Relationship Diagram (ERD) is used to create a model for your database for review.

Lab Assignment for Task 1

1. Develop an ERD from the DDL of the ten tables provided with this assignment.

Deliverable 1 - ERD

★ Submit Entity Relationship diagram for your normalized DDL in a word document.

Document Name: TeamName_ERD

Due Date: 07/14/17

Task 2 – Create DDL

Data Definition Languages (DDL) are used to translate your ERD into physical objects that will be created on disks. This is known as physical *design*.

Lab Assignment For Task 2

1. Create DDL.
   a. Create the DDL for the ten tables.
   b. Be sure the tables can be joined.

      Note that this step can be skipped because a DDL script has been provided for you and is located on the server under **/home/oracle/lab-scripts**.

Deliverable 2

★ DDL script.

Script name: 10tables.sql (This has been provided for you) – **DO NOT RUN THIS YET**

DDL Provided

```
SPOOL TeamName-10tables.log
/*--------------------------------------------------*/
/*     DROP OBJECTS                                 */
/*--------------------------------------------------*/
drop table MANUFACTURER;
drop table TRANSACTION;
drop table EMPLOYEE;
drop table RETURNS;
drop table PAYMENT;
drop table ORDER_ITEM;
drop table ORDERS;
drop table PRODUCT;
drop table CUSTOMER;
drop table ZIP;

/*--------------------------------------------------*/
/*     CREATE TABLES                                */
/*--------------------------------------------------*/

create table ZIP
      (ZIPCODE                char(10)          NOT NULL,
       CITY                   char(25),
       STATE                  char(2),
constraint pk_zip  primary key (ZIPCODE)
      using index tablespace USER_INDX1
      STORAGE (INITIAL 1000K NEXT 120K)
      )
PARTITION BY RANGE (STATE)
(
PARTITION ZIP_PART1 VALUES LESS THAN ('N')
      pctfree 5
      pctused 90
TABLESPACE USER_DATA1
      storage
            (
                  initial     1240k
                  next        200k
                  maxextents  unlimited
                  pctincrease 0
            ),
PARTITION ZIP_PART2 VALUES LESS THAN (MAXVALUE)
      pctfree 5
      pctused 90
TABLESPACE USER_DATA2
storage
            (
                  initial     1240k
                  next        200k
                  maxextents  unlimited
                  pctincrease 0
            )
```

```
);

create table CUSTOMER
(      CUSTOMERID  NUMBER(10) NOT NULL,
       FIRSTNAME   VARCHAR2(15),
       LASTNAME    VARCHAR2(15),
       PHONENO     CHAR(15),
       ADDRESS     VARCHAR2(35),
       ZIPCODE     CHAR(10),
constraint pk_customer_id primary  key (CUSTOMERID)
      using index tablespace USER_INDX1
      STORAGE (INITIAL 50K NEXT 10K),
constraint fk_zip foreign  key (ZIPCODE) references ZIP
        )
      pctfree 20
      pctused 40
TABLESPACE USER_DATA1
      storage
            (
                  initial 120k
                  next 10k
                  maxextents unlimited
                  pctincrease 0
            );

create table PRODUCT
      (PRODUCTID              NUMBER(10),
       PROD_UNIT_PRICE        NUMBER(10),
       DESCRIPTION      VARCHAR2(150),
constraint pk_productid  primary key (PRODUCTID)
      using index tablespace USER_INDX1
      STORAGE (INITIAL 50K NEXT 10K)
      )
      pctfree 5
      pctused 90
TABLESPACE USER_DATA2
      storage
            (
                  initial  150k
                  next   20k
                  maxextents unlimited
                  pctincrease 0
            );

create table ORDERS
      (ORDERID    NUMBER(10) NOT NULL,
       CUSTOMERID NUMBER(10) NOT NULL,
       SALES_EMP_SSN     NUMBER(9),
       ORDER_METHOD      CHAR(10),
constraint pk_orderid  primary key (ORDERID)
      using index tablespace USER_INDX1
      STORAGE (INITIAL 50K NEXT 10K),
constraint fk_customerid foreign  key (CUSTOMERID) references CUSTOMER
      )
      pctfree 5
      pctused 60
TABLESPACE USER_DATA1
```

```
      storage
            (
                  initial 100k
                  next 10k
                  maxextents unlimited
                  pctincrease 0
            );



create table ORDER_ITEM
      (ORDERID          NUMBER(10) NOT NULL,
       PRODUCTID        NUMBER(10),
       QUANTITY         NUMBER(10),
Constraint pk_order_item primary key(ORDERID,PRODUCTID)
            using index tablespace USER_INDX1
            STORAGE (INITIAL 50K NEXT 10K),
Constraint fk_order_id1 foreign key (ORDERID) references ORDERS,
Constraint fk_product_id foreign key (PRODUCTID) references PRODUCT
      )
      pctfree 5
      pctused 90
TABLESPACE USER_DATA2
      storage
            (
                  initial 50k
                  next 20k
                  maxextents unlimited
                  pctincrease 0
            );


create table PAYMENT
      (PAYMENTID        NUMBER(10),
       ORDERID          NUMBER(10),
       CARD_EXP_DATE    DATE,
       CARD_NUMBER NUMBER(30),
       PAYMENT_AMOUNT   NUMBER(20),
       PAYMENT_METHOD   VARCHAR2(5),
constraint pk_paymentid  primary key (PAYMENTID)
      using index tablespace USER_INDX1
      STORAGE (INITIAL 50K NEXT 10K),
constraint fk_order_id2 foreign key (ORDERID) references ORDERS )
      pctfree 20
      pctused 40
TABLESPACE USER_DATA2
      storage
            (
                  initial     100k
                  next        10k
                  maxextents  unlimited
                  pctincrease 0
            );

create table RETURNS
      (RETURNID         number(10)  NOT NULL,
       ORDERID          number(10)  NOT NULL,
```

```
       PAYMENTAMT       number(20),
constraint pk_returns  primary key (RETURNID)
            using index tablespace USER_INDX1
            STORAGE (INITIAL 50K NEXT 10K),
constraint fk_order_id3 foreign key (ORDERID) references orders
      )
      pctfree 20
      pctused 40
TABLESPACE USER_DATA2
      storage
            (
             initial          80K
             next             20K
             maxextents       unlimited
             pctincrease 0);

create table EMPLOYEE
      (EMPID      char(10)    NOT NULL,
       EMPFNAME          char(20),
       EMPLNAME          char(25),
       PHONENUM          number(10),
         ADDRESS               char(25),
         ZIPCODE                 char(10)   NOT NULL,
constraint pk_employee  primary key (EMPID)
            using index tablespace USER_INDX1
            STORAGE (INITIAL 50K NEXT 10K),
constraint fk_zip_id2 foreign  key (ZIPCODE) references ZIP
      )
      pctfree 20
      pctused 40
TABLESPACE USER_DATA1
      storage
            (
             initial          160K
             next             20K
             maxextents       unlimited
             pctincrease 0);

create table TRANSACTION
      (TRANSID          number(10)  NOT NULL,
       CUSTOMERID         number(10)      NOT NULL,
       TRANSDATE           date,
       TRANSPAYTYPE           Char(15),
         AMOUNT                number(10),
         EMPID                 char(10) NOT NULL,
constraint pk_transaction  primary key (TRANSID)
            using index tablespace USER_INDX1
            STORAGE (INITIAL 50K NEXT 10K),
constraint fk_customer_id2 foreign key (CUSTOMERID) references CUSTOMER,
constraint fk_empid foreign key (EMPID) references EMPLOYEE
      )
      pctfree 5
      pctused 60
TABLESPACE USER_DATA2
      storage
            (
             initial          80K
```

```
             next             10K
             maxextents       unlimited
             pctincrease 0);
create table MANUFACTURER
             (MANUFACTURERID      number(10)    NOT NULL,
              PRODUCTID           number(10)    NOT NULL,
              MANUFACTURERNAME    char(30)      NOT NULL,
constraint pk_manufacturer primary key (MANUFACTURERID)
               using index tablespace USER_INDX1
               STORAGE (INITIAL 50K NEXT 10K),
constraint fk_product_id1 foreign key (PRODUCTID) references PRODUCT
       )
       pctfree 20
       pctused 40
TABLESPACE USER_DATA1
           storage
                  (
                    initial  80K
                    next     20K
                    maxextents    unlimited
                    pctincrease     0);

/*-------------------------------------------*/
/*                     INDEXES       */
/*-------------------------------------------*/
create index  fk_zip
  on CUSTOMER (ZIPCODE)
tablespace USER_INDX1
      STORAGE (INITIAL 50K NEXT 10K)
      ;

create index  fk_customerid
  on ORDERS (CUSTOMERID)
tablespace USER_INDX1
      STORAGE (INITIAL 50K NEXT 10K)
      ;

create index  fk_order_id1
  on ORDER_ITEM (ORDERID)
tablespace USER_INDX1
      STORAGE (INITIAL 50K NEXT 10K)
      ;

create index  fk_product_id
  on ORDER_ITEM (PRODUCTID)
tablespace USER_INDX1
      STORAGE (INITIAL 50K NEXT 10K)
      ;

create index  fk_order_id2
  on PAYMENT (ORDERID)
tablespace USER_INDX1
      STORAGE (INITIAL 50K NEXT 10K)
      ;
create index fk_order_id3
on RETURNS (ORDERID)
tablespace USER_INDX1
```

```
        STORAGE (INITIAL 50K NEXT 10K)
        ;
create index fk_product_id1
on MANUFACTURER (PRODUCTID)
tablespace USER_INDX1
         STORAGE (INITIAL 50K NEXT 10K)
        ;
/*-------------------------------------------*/
/*     CHECK CONTENT OF CATALOG               */
/*-------------------------------------------*/
select substr(object_name, 1, 20), object_type, status
     from user_objects
      ;

SPOOL OFF
```

A partitioned table is a special type of table. By dividing the table into physical pieces, the table may be easier to manage. Also, by spreading the work across disks, performance may improve. Partitioned tables are particularly useful with large tables such as those used in data warehouses.

The partitioned table (ZIP) is split into two parts as shown in the DDL. The Zip table is a partitioned table

Task 3 – Create Tablespaces

Tablespaces are logical structures used to house application data. Create DDL for the tablespaces.

Lab Assignment for Task 3

Create the DDL for the tablespaces.

1. Size the tablespaces.
2. Create DDL for the tablespaces.
3. Modify the table DDL so each table is assigned to one of the tablespaces. **(This step has been completed – See DDL provided.)**

Deliverable 3

★ The DDL for tablespaces

Script name: TeamName_tbsp.sql

Due Date: 07/14/17

Sample Solution!

Prepare your DDL to create the tablespaces. Here is a sample:

```
CREATE TABLESPACE "USER_DATA1" LOGGING
DATAFILE '/u03/oradata/ORACLE_SID/user_data01.dbf' SIZE
500M REUSE AUTOEXTEND ON NEXT 100M MAXSIZE 1000M EXTENT
MANAGEMENT LOCAL;

CREATE TABLESPACE "USER_DATA2" LOGGING
DATAFILE '/u04/oradata/ORACLE_SID/user_data02.dbf' SIZE
500M REUSE AUTOEXTEND ON NEXT 100M MAXSIZE 1000M EXTENT
MANAGEMENT LOCAL;

CREATE TABLESPACE "USER_INDX1" LOGGING
DATAFILE '/u03/oradata/ORACLE_SID/user_indx01.dbf' SIZE
200M REUSE AUTOEXTEND ON NEXT 50M MAXSIZE 300M EXTENT
MANAGEMENT LOCAL;

CREATE TEMPORARY TABLESPACE "USER_TEMP"
TEMPFILE '/u04/oradata/ORACLE_SID/temp_user01.dbf' SIZE
500M REUSE AUTOEXTEND ON NEXT 100M MAXSIZE 1000M EXTENT
MANAGEMENT LOCAL;
```

Note that application data and indexes will be stored on separate tablespaces to alleviate performance issues.

Task 4 – Create User Accounts

Before you create your database objects, you have to create an application account to house these objects and the data.

Lab Assignment for Task 4

1. Set up DBA accounts. You will connect from your DBA accounts to create the application schema account.
2. Create application account.

Deliverable

★ The script used to create DBA account.
Script Name: dbaccount.sql

★ The script used to create the application account.
Script Name: appaccount.sql

Due Date: 07/14/17

Sample Solution for DBA Account!

Login from the sys account and **create your DBA accounts**. Be sure you run from a script.

```
create user KUD01DBA identified by "password"
Default tablespace users
temporary tablespace temp;

grant connect, resource, DBA to KUD01DBA;
```

Sample Solution for Application account!

Login from your DBA account and **create the application user account.** Be sure you run from a script.

```
Create user APP_USER identified by password
Default tablespace tablespace_name
Temporary tablespace tablespace_name;

grant connect, resource to APP_USER;
```

Task 5 – Create Database Objects

The DDL for the tablespaces was created in Task 3 and all ten tables have been assigned to their respective tablespaces in the script. Now create the objects.

Lab Assignment for Task 5

1. Execute the DDL for the tablespaces from your DBA account.
2. Connect as the APP_USER
3. Execute the DDL to create the objects.
4. Verify that all the objects were created by querying the data dictionary.

Deliverable 5

★ A log file displaying the output of the tablespace script.

Script name: TeamName_tab.log

★ A log file displaying the output of the ddl script for the tables.

Script name: TeamName_10tables.log

★ A report from a query against the catalog showing that the objects are valid

Due Date: 07/16/17

Checklist

- ❖ Run the tablespace DDL first because the table DDL is dependent on it.
- ❖ Check the log file and verify that everything executed successfully.
- ❖ Check the data dictionary and be sure everything looks right. Run a query against user_objects and be sure all of the objects are valid.

Task 6 – Back up the Database

After any structural changes are made to a database, it is important to take a backup before the change is made and after the change has been made.

Lab Assignment for Task 6

1. Take a full database export of your database.
2. Set up a cron job so an export is done every night at a specific time for a week ONLY. Check to be sure that the file is actually written!

Deliverable 6

- ★ The export command to backup the database.
- ★ The export backup log file.
- ★ The job scheduled to run nightly for a week.

Due Date: 07/16/17

Checklist

- ❖ Check that the directory and file permissions are set correctly.
- ❖ Check that the backup file was created and where it is supposed to be created.
- ❖ Be sure that your team members can execute the backup scripts.

Task 7 – Identify and Load Data

You have created your tables in Task 6, a database is no good without data right? The next step is to populate the tables with data. You will use SQLLOADER and other tools to load data into your database.

Lab Assignment For Task 7

1. Load 50,000 records in the zip table (use `sqlload`).
2. Load at least 1,000 records on the other tables.
3. You may also use the Insert into as select statement to load data into two of your tables.

Deliverable 7

- ★ The control files used to load all 10 tables
- ★ Insert into select statement if you chose to use this to load data into two of the tables of your choice.

Due Date: 07/18/17

Task 8 – Analyze the Tables

Update the statistics in the data dictionary for each table using the analyze command or the dbms_stats package. You can use the DBMS_STATS package or the ANALYZE statement to gather statistics about the physical storage characteristics of a table, index, or cluster. These statistics are stored in the data dictionary and can be used by the optimizer to choose the most efficient execution plan for SQL statements accessing analyzed objects.

Oracle recommends using the DBMS_STATS package for gathering optimizer statistics, but you may also use the ANALYZE statement to collect statistics unrelated to the optimizer, such as empty blocks, average space.

Lab Assignment for Task 8

Gather statistics on all ten tables.

Deliverable 8

- ★ The ANALYZE commands for the ten tables OR
- ★ The DBMS_STATS package to gather statistics of the entire schema objects
- ★ The report showing the correct number of rows in the catalog

Due Date: 07/18/17

Sample Solution!

Use the ANALYZE TABLE command. After the data is loaded, run a query against the data dictionary and notice how many rows are shown for the table you just loaded.

```
SELECT TABLE_NAME, NUM_ROWS FROM user_tables;
```

The number of rows is 0! Fifty thousand records were just loaded, but the data dictionary has zero rows for this table. Why is that?

Now run the Analyze command to gather the statistics on the table.

```
ANALYZE TABLE tablename
COMPUTE STATISTICS;
SELECT TABLE_NAME, NUM_ROWS FROM user_tables;
```

You can also use the dbms_stats package to gather statistics on the entire schema.

```
exec dbms_stats.gather_schema_stats(ownname => 'KUDRANT',options
=> 'GATHER AUTO');
```

Task 9 – Restructure Objects

You have created your application schema and populated the tables with data. You have regularly performed a backup to ensure that the database is recoverable in the event of data loss.

This data has to be maintained across the different database environments.

After the database tables have been created and fully loaded with data, user requirements may change. These changes in requirements may force structural changes to the database. For example, columns may need to be expanded in size or new columns may need to be added to tables.

DBA's are responsible for deploying this changes across the different database environments while also ensuring that these changes do not cause any performance degradation to the database.

The exercises in this task will teach you how to handle structural changes in a database.

Lab Assignment for Task 9

1. Restructure database objects by using the `ALTER TABLE` command. You will create the DDL statements to modify the tables in questions A-E in scripts and execute these scripts.
   A. Modify one of your tables and add a column. Use a query and see if it successfully added the column.
   B. Modify a table by making one column longer.
   C. Modify a table by making one column shorter.
   D. Modify a table by changing a column from a NUMBER to a CHAR.
   E. Modify a table by changing a column from a CHAR to NUMBER.
2. Restructure a table by actually creating a new object.
   A. Delete a column on a table. Do not use the `DROP COLUMN` command.

      - Create a table like the one that in which the column should be deleted. The new table should not have the old column.
      - Use a mass insert to copy the data to the new table.
      - Check the data in the new table and be sure it is right.
      - Add a primary key to the new table the using the `ALTER TABLE` command.
      - Add any foreign keys.
      - Drop the original table.
      - Rename the new table.

Deliverable 9

- ★ Your SQL scripts

Due Date: 07/20/17

Task 10 – Design and Implement a Performance Test

Part of daily database maintenance requires that the database should be available for use. DBA's usually get calls from users complaining of slow database response times when running their jobs or even while running simple SQL statements. It is the DBA's job to drill down into the database using scripts and other tools to diagnose and resolve any performance issues.

Lab Assignment for Task 10

Set up scripts to capture important performance information for the instance. Run the tests below. After each test, capture the performance statistics.

TEST 1:

Embed SQL in a script that can be executed a variable number of times. The embedded SQL should do the following:

1. Select all the rows from one table.
2. Join two tables.
3. Join ten tables primary to foreign keys.
4. While the script is executing, gather information about the session, locks, SQL, CPU and memory usage, I/O, any wait events, DB wait time, load on server.

TEST 2:

1. Update 1,000 rows and roll back the changes.
2. Insert 1,000 rows into a table and then rollback.
3. Delete 1,000 rows from a table and then rollback.
4. While the script is executing, gather information about the session, locks, SQL, CPU and memory usage, I/O, any wait events, DB wait time, load on server.

TEST 3:

1. Select all rows from one table.
2. Join two tables.
3. Join ten tables primary to foreign keys.
4. Update 1,000 rows and rollback.
5. Insert 1,000 rows and rollback.
6. Delete 1,000 rows and rollback.
7. While the script is executing, gather information about the session, locks, SQL, CPU and memory usage, I/O, any wait events, DB wait time, load on server.

Deliverable 10

- ★ Script to capture performance statistics between scenarios
- ★ Script to capture dynamic performance information during the workload execution OR
- ★ Create AWR, ADDM, ASH reports that capture this information.

Due Date: 07/21/17

Checklist!

- ❖ The scripts may need to be run multiple times during the test.

Task 11 – Test Recovery

The ability to recover the database in case of events like disk failures, user error, and natural disasters such as floods is a vital task of a DBA. You MUST not ONLY have a backup and recovery plan in place but it has to be tested to ensure that it works as expected.

For this task, you will use data pump export/import and RMAN to perform your backup and recovery tasks.

Lab Assignment for Task 15

Set up a script for the recovery tests:

1. Design a script that creates a small table and Insert 3 rows on the table. The table should not have a primary key.
2. Be sure to add a drop table statement at the beginning of the script since you will be executing this script several times.

   Script name: rec-tests.sql

TEST 1:

1. Run the script rec-tests.sql to create a small test table with three rows in it.
2. Take an data pump export backup of the table.
3. Delete two rows. Now there is one row on the table.
4. Recover the data using data pump import. Did you get the 1 row back?
5. Explain what happened.

TEST 2:

1. Run the script rec-tests.sql to create a small test table with three rows in it.
2. Take an RMAN backup
3. Delete two rows. Now there is one row on the table.
4. Recover the data using the RMAN restore and recover commands.
5. Did you get the 1 row back?
6. Explain what happened.

TEST 3:

1. Run the script rec-tests.sql to create a small test table with three rows in it.
2. Take a data pump export of the table
3. Delete two rows. Now there is one row on the table.

4. Insert a new row.
5. Recover the data using import. Did you get 2 rows back?
6. Explain what happened.

TEST 4

1. Run the script to rec-tests.sql create a small test able with three rows in it.
2. Take an RMAN backup
3. Delete two rows. Now there is one row on the table.
4. Insert a new row.
5. Recover the data using the RMAN restore and recover commands.
6. Did you get 2 rows back?
7. Explain what happened.

TEST 5:

1. Run the script rec-tests.sql to create a small test table with three rows in it.
2. Take a data pump export of the table
3. Drop the table.
4. Use data pump import again. Were you able to recover the table with the correct data?
5. Be able to explain what happened.

TEST 6:

1. Run the script rec-tests.sql to create a small test table with three rows in it.
2. Take an RMAN backup.
3. Drop the table.
4. Use RMAN recover and restore. Were you able to recover the table with the correct data?
5. Explain what happened

TEST 7:

Repeat the export/import and RMAN commands, but this time, create a primary key on the table in the script.

TEST 8:

Repeat the sequence of changes to the test table, but use flashback query in the recovery operations.

Deliverable 11

- ★ Create table DDL, insert statements, export commands, RMAN commands
- ★ A table which shows the number of rows returned by each recovery operation
- ★ An analysis of what happened when the two tools were used and why.

Enter your results in the chart below

Tests	Rows Expected	Actual Import Rows	Actual RMAN Rows	Comments
1				
2				
3				
4				
5				
6				
7				

Due Date: 07/23/17

Task 12 – Monitor the Database

You will create queries to monitor your database on a daily basis to ensure that it is operating as expected and also identify any potential issues before they occur. You will also run these scripts against your database and examine the results.

Lab Assignment For Task 12

1. Check status of database. script name: TeamName_db-status.sql
2. Monitor space. script name: TeamName_monitor-space.sql
3. Check statistics. script name: TeamName_db-stats.sql
4. Check for invalid objects. script name: TeamName_check-invobjects.sql

Deliverable 12

- ★ All SQL scripts and log files

Due Date: 07/21/17

Task 13 – Use OEM Cloud Control 13C To Perform Daily Tasks

OEM cloud control 13C is a centralized tool used to monitor and administer all your databases.

Lab Assignment For Task 13

1. Check status of database.
2. Monitor space.
3. Check statistics.
4. View objects in the database.
5. Use OEM to identify the top 10 intensive SQL statements in the database
6. Create a data pump job
7. Create database start/stop alerts (rules). Space monitoring
8. Create a tablespace
9. Check the status of your database
10. Clone a user account
11. Create AWR report for your database
12. Create ADDM report for your database

Deliverable 13

★ 12 screen shots with the OEM tool that show the 12 tasks in this project.

Due Date: 07/21/15

Task 14 – Create End Users

You database is up and running, data has been loaded into it and now you have to create end-user accounts to allow end-users to access this data and perform their daily operations.

You will create accounts for end users who have a need to perform specific tasks and have been approved by their management. You will verify this approval before creating their accounts and will give them ONLY the level of access they need to do their work.

For this task, you will create accounts for end users (another team) and grant them access to the database. You will also be an end user to another Team's database. Be sure to check that you are in the right database before you make any changes.

Lab Assignment for Task 14

1. Execute scripts to create end users. Team kud01 will be end users for kud02, kud02 for kud03, kud03 for kud04, kud04 for kud05 and so on. The last team will be end users for kud01. Each user should be entered as individuals, rather than as part of the team account.
2. Create synonyms for each of your ten tables so the prefix does not have to be used by your end users.
3. You will connect to the database where you are an end user and try to select some data from one of the tables. What happens? Why?

Deliverable 14

- ★ DDL for the synonyms
- ★ DDL to create end users
- ★ DCL to grant privileges

Checklist

- ❖ End users should change their password as soon as they enter the database.

Sample Solution!

```
create public synonym table_name for app_user.table_name;
create user kud01a identified by db01a;
grant connect to kud01a;
grant select on table_name to db01a;
```

Due Date: 07/22/17

Task 15 – Detect End Users

One of your tasks will be to identifying who is running what (jobs/processes) against your database. You will call calls from users complaining that their jobs are running slower than expected and it is your job to drill down into the database to identify the processes running and which one is clogging down the system.

Monitoring is an important function in securing the database, Oracle provides dynamic performance view (V$ views) to assist DBA's in troubleshooting database issues. You will use two of these views V$SESSION and V$SQL to monitor work performed by end users.

Lab Assignment for Task 15

1. While end-users are performing work on your database, run scripts that will show who is doing work and what code they are running.

Deliverable 15

★ SQL script to detect end-users who are currently executing code.

Due Date: 07/22/15

Sample Solution!

```
select a.username, b.sql_text from V$SESSION a,V$SQL b where
a.sql_address=b.address;
```

Task 16 – Plan the disk Layout For The Installation of Oracle 12C

Before you install oracle on a server, you have to do capacity planning. You have to plan for the storage and provision for growth in 1, 3, 5 years. You also have to plan the best layout for the files to conform to the OFA as much as possible.

Deliverable 16

★ A list with seven disks (/u01-/u07) and what will reside on them

Sample Solution!

Here is the suggested disk allocation:

- /u01 – Oracle software
- /u02 – SYSTEM, control file 1
- /u03 – UNDOTBS, USERS, control file 2
- /u04 – SYSAUX, control file 3, TEMP
- /u05 – online redo logs 1, 2, 3
- /u06 – online redo logs 1, 2, 3
- /u07 – archived redo log

Due Date: 07/23/17

Task 17 –Install the Oracle 12c1 Software

OUI will be used to install the 12c1 database. **Review the steps install oracle 12c software in your notes.**

Lab Assignment for Task 17

1. Use OUI to install the Oracle software.
2. Create a database using DBCA.
3. Look at the parameter file used by the instance.

Deliverable 17

- ★ Database installation log files
- ★ The parameter file

Due Date: 07/23/17

Task 18 – Create An Oracle 12c1 Database

You have installed the oracle 12c1 software, now it's time to create your database (collection of memory structures, background processes and files on disks). **Review the steps create an oracle 12c1 database in your notes.**

Lab Assignment for Task 18

1. Use DBCA to create an oracle database.
2. Look at the parameter file used by the instance.

Deliverable 18

- ★ Working oracle 12c1 database and log files of installation.
- ★ The parameter file

Due Date: 07/23/17

Task 19 – Check that the Installation Ran Correctly

You have installed the oracle 12c1 software and create your database, now what?

You have to verify that the installations successfully completed, the oracle instance, and database files were created on disks.

Lab Assignment for Task 19

1. Check that the oracle installation was successfully completed.
2. Check that the oracle database was successfully created and you can connect to it.

Deliverable 19

- ★ The commands used to check the oracle processes running for newly created database.
- ★ The locations of the database files created on disks.
- ★ The last few lines in the log files created when the installation completes.
- ★ The entries in the alert log related a successful installation.

Checklist

1. Check that the oracle processes running.
2. Check the log files created when the installation completes for errors. If the files are too big to look at, you can use the `tail` command.
3. Check the init.ora file. Add it to the appendix.
4. Check the alert log. Add the location of the alert log to the appendix.

Due Date: 07/23/17

Task 20 –Upgrade the 11gr2 database to 12C1

This task will be performed by the entire TEAM

You will upgrade your 11gR2 database to the next major release (12C1) to take advantage of the new features. **Review the steps to upgrade an oracle 11gr2 database in your notes.**

Lab Assignment For Task 20

1. Upgrade the database **dbtrain** to **12c**.
2. Re-register the new 12c database with the listener.
3. Check for invalid objects after the upgrade.
4. Gather statistics on the upgraded database.

Deliverable 20

- ★ Screen shots of upgrade session
- ★ Log files from upgrade
- ★ Upgraded database in 12c1 home.

Due Date: 07/25/17

Task 21 –Patch the database

This task will be performed by the entire TEAM

You have installed the Oracle 12C1 software and created a database. Now, you have to install the latest patch to the database home and database. **Review the steps patch an oracle 11gr2 and e 12c1 database in your notes.**

Lab Assignment For Task 21

1. Patch the oracle 11gr2 home and database.
2. Patch the oracle 12c1 home and database.

Deliverable 21

1. Log file from patching.
2. List of inventory of patches installed on the database server in the 11gr2 and 12c1 homes.

Checklist

- ❖ Be sure to back up the database before you begin patching.
- ❖ Be sure to log your patching session and review the log file for any ORA- errors.

Due Date: 03/31/18

Final Deliverable

- ★ Compile all your files from task 1-21 in a word document and submit to the class page.

- ★ All log files and scripts used should be added as an appendix at the end of your document.

Deliverables

Deliverable	Expected Due Date
TASK 1	01/30/18
TASK 2	N/A
TASK 3	02/07/18
TASK 4	02/07/18
TASK 5	02/07/18
TASK 6	02/13/18
TASK 7	02/15/18
TASK 8	02/15/18
TASK 9	02/21/18
TASK 10	02/21/18
TASK 11	07/23/17
TASK 12	02/27/18
TASK 13	02/27/18
TASK 14	02/28/18
TASK 15	03/07/18
TASK 16	N/A
TASK 17	N/A
TASK 18	N/A
TASKS 19	N/A
TASKS 20	03/21/18
TASKS 21	03/21/18

Note: Following directions are CRITICAL for these tasks and points will be deducted if all the files are not submitted. You will earn a total of 500 points for this lab.