LOAN SYSTEM

Supervised by: Dr. Mohamed Khafagy

By

Hossam Eldeen Essam
Ahmed Samir
Mohamed Adel
Mohammad Mustafa
Omar Mohamed
Saga Muhammed

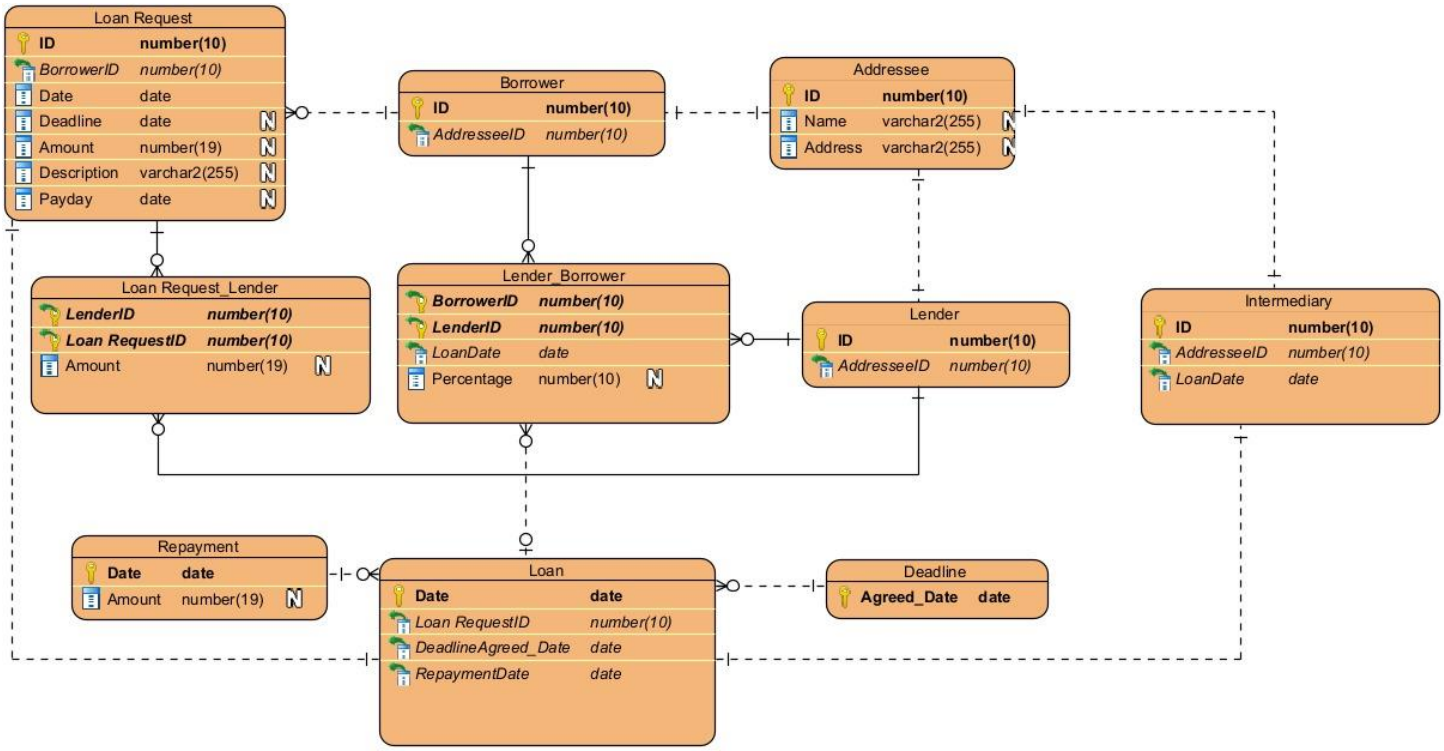
Information Technology Institute

Table of Contents

Abstract

Micro loans have gained significant popularity among borrowers in developing nations due to their provision of small-scale financing facilitated by information technology. These loans are primarily utilized to fund startups or expand existing businesses, offering borrowers a viable opportunity for repayment. One distinguishing feature of micro loans is their ability to source funds from multiple lenders, distinguishing them from traditional loan structures. To effectively manage micro loans, the development of an E-R model is crucial, encompassing the essential information.

The ERD is provided:



Introduction

The core objective of a DBMS is to establish a reliable and convenient method for storing and accessing database information. It facilitates the storage of meaningful and organized data, which consists of verifiable facts. While individuals often use software like Excel to store data in a database format, a DBMS offers a more robust and specialized solution for handling data management tasks.

We created an offline loan system database for a system that set a connection between borrowers and lenders and provide a safe environment for them to make and participate in loans ,borrowers provide data entry user with their personal info and info about the request they want to make such as exact amount of loan ,for what purpose they will use loan money and desired deadline for repaying the loan , lenders also provides data entry user with their personal info then implementers decide in which loan lenders will participate, unlike traditional loans, money comes from many lenders and finally we have the intermediary who considered as link between borrowers , lenders and the system

Assumptions

1. ERD

After studying the diagram, we decided to make the following adjustments in order to improve and speed up the performance:

1. Adding loan request id column in loan request table as a primary key instead of loan request date

Reason: Faster retrieve of all requests.

2. Adding an identifying relationship between loan request table and loan table with loan request id as foreign key column in loan table

Reason: Retrieve the approved loans from loan table based on their request id from loan request table.

3. Replacing foreign key loan request date column in loan request lender table with the new foreign key loan request id column

Reason: The first edit we made in loan request table.

2. Database

Over the next four years, the anticipated range of records in each table is expected to be between 150,000 and 600,000 entries.

2.1 Tables

150K which is a year

1. Addressee

   One row (10+255+255)*150K/1024 > (77M) = 80M per Year

   - Addresee_id (10 Bytes)
   - Name (255 Byte)
   - Address (255 Byte)

2. Borrower

   One row (10+10)*150K /1024 = 3M per Year

   - Borrower_id (10 Bytes)
   - Addresee_id (10 Bytes)

3. Lender

   One row (10+10)*150K /1024 = 3M per Year

   - Lender_id (10 Bytes)
   - Addresee_id (10 Bytes)

4. Intermediary

   One row (10+10+7)*150K/1024 = 4M per Year

   - Intermediary_id (10 Bytes)
   - Addresee_id (10 Bytes)
   - Loan_date (7 Bytes)

5. Loan_Request

   One row (10+10+7+7+19+255+7)*150K/1024>47M=50M per Year

   - Loan_request_id (10 Bytes)
   - borrower_id (10 Bytes)
   - Loan_request_date (7 Bytes)

- Deadline (7 Bytes)
- Loan_request_amount (19 Bytes)
- Description (255 Byte)
- Payday (7 Bytes)

6. Loan_request_lender

   One row (10+10+19)*150K/1024 = 6M per year
   - Lender_id (10 Bytes)
   - Loan_request_id (10 Bytes)
   - Lender_amount (19 Bytes)

7. Lender_borrower

   One row (10+10+7+10)*150K/1024 = 6M per year
   - borrower_id (10 Bytes)
   - lender_id (10 Bytes)
   - loan_date (7 Bytes)
   - percentage (10 Bytes)

8. Loan

   One row (7+10+7+7)*150K/1024 = 5M per year
   - Loan_date (7 Bytes)
   - Loan_request_id (10 Bytes)
   - Deadline_agreed_date (7 Bytes)
   - Repayment_date (7 Bytes)

9. Repayment

   One row (7+19)*150K/1024 = 4M per year
   - Repayment_date (7 Bytes)

- Repayment_amount (19 Bytes)

10. Deadline

One row (7)*150K/1024 >1M = 2M per year

- agreed_date (7 Bytes)

2.2 Views

Views for Users:-

- **Borrowers_View**
  A join Relation between Borrower_id ,Addressee_id (Borrower Table)
  and Borrower_Name, Address (Addressee Table)
  So the user associated with data entry of the borrower request couldn't access the data of non-borrowers

- **Lenders_View**
  A join Relation between Lender_id ,Addressee_id (Lender Table)
  and Lender_Name, Address (Addressee Table)
  So the user associated with data entry of the lender couldn't access the data of non-lenders

- **Intermediaries_View**
  A join Relation between Intermediary_id ,Addressee_id (Intermediary Table)
  and Intermediary_Name, Address (Addressee Table)
  So the user associated with data entry of the Intermediary couldn't access the data of non-intermediaries

Views for Reports:-

- **Active_loans_view**
  Daily used Columns (loan_date, deadline_agreed_date) from Loan Table with a specific needed condition.

- **Intermediary_Borrower_View**
  A join Relation between loan_date (lender_borrower Table) and Borrowe_name, Intermediary_Name (Addresse Table)

- **lenders_share_view**
  A Join Relation between lender_id ,loan_date (lender_borrower table), lender_name(addressee table) and lender_amount(loan_request_lender)
  To get information about lenders share for each quarter

- **loan_lenders_view**
  A join Relation between loan_date(loan table) , loan_request_amount (loan_request table) and count of distinct lender_id(loan_request_lender)
  To get monthly information about loans and their lenders

2.3 Triggers

As it is known that inserting data directly via the view is not possible, so the user of the view needs a trigger in order to enter data into the view. So we made an **"instead of trigger"** for each user-view to make inserting operation possible.

- Borrowers_View_trig for the Borrowers_view User
- Lenders_View_trig for the Lenders_view User
- Intermediaries_View_trig for the Intermediaries_view User

2.4 Materialized views

We used materialized views in the cases of yearly reports that is associated with calculations so it won't take large space and also will help in retrieving the needed result without needing to perform the calculations throughout the year

- **yearly_cash_flow_Mview**
  A materialized view that is refreshed yearly on day 31-12 of the year to calculate the cash flow for the system in that year

- yearly_percent_loan_Mview
  A materialized view that is refreshed yearly on day 01-06 of the year to calculate the percentage of loans with amount larger than 200000 in the system in that year.

2.5 Index

We decided not to make special indices and to limit ourselves to the default indices on the Primary key columns as they are the main engine in the join operations and in the overall data retrieving, we tried to create a join indices, but we could not find any reference to a specific index called “join index” in Oracle 19c.

2.6 Tablespaces

- System Tablespace
  In Oracle Database, the system tablespace is typically created automatically during the database creation process. When you install Oracle Database, it includes default tablespaces, and the system tablespace is one of them. This tablespace contains the essential data dictionary tables and other system-related structures required for the functioning of the Oracle database.

- Sysaux Tablespace
  The SYSAUX tablespace is automatically created by Oracle Database during the initial database creation process. It serves as a supplementary system tablespace introduced in Oracle 10g. It's designed to house various auxiliary system-related data that doesn't belong in the primary SYSTEM tablespace but is essential for the functioning of the database.

- User Tablespace
  In Oracle Database, the "users" tablespace is a default tablespace that is created during the database installation process. It is designed to store data for individual database users or schemas. When a new user is created in

Oracle, if a specific tablespace is not explicitly specified, the user is automatically assigned to the "users" tablespace as their default tablespace

- **Undo Tablespace**
  The Undo tablespace in Oracle Database is a specialized tablespace that stores the undo information necessary to manage and maintain the consistency and integrity of data during transactions. Undo information records the changes made to data in the database and is essential for providing read consistency, supporting transactions, and enabling database rollback operations.
  We made 1 undo tablespace with size 100 M

- **Temp Tablespace**
  The Temporary (Temp) tablespace in Oracle Database serves as a storage area for transient or temporary data that is generated during sorting, hashing, or other database operations. It is used for tasks such as sorting data for queries, creating indexes, and performing joins that require temporary space.
  We made 4 Temp Tablespaces, one for each group of users.

We divide our tables into two tablespaces (Data Entry and Implementers)
In the case of a failure occurring in The Data Entry tablespace, The Implementers tablespace remains unaffected and continue to operate normally.

- Block size is 8K
  - Data Entry Tablespace (140 M)

All the data related to the data not the actual transactions

1. addresse Table
2. borrower Table
3. lender Table
4. intermediary Table
5. loan_request Table

- Implementers Tablespace (25M)

All the data related to the loan transactions

1. loan_request_lender Table
2. lender_borrower Table
3. loan Table
4. deadline Table
5. repayment Table

2.7 User-Profile

Users

The system has 2 DBAs and 14 user with different privileges and roles divided to four groups:

1. Data-Entry Group
   2 User for borrower request data
   1 User for loan request lender data
   1User for intermediary data

2. Loan Implementer Group
   4 Users for Dates & amount

3. Audit
   1 User for borrower request
   1 User for lender share
   1 User for intermediary
   1 User for loan

4. Manager
   2 Users for accepting decisions& generating reports

5. DBAs

Loandba Owner of sys

Loandba2 Owner of Recovery Catalog

Profiles

➢ Loan Implementer Group

- Password will expire with 50 days
- Number of failed login 3
- Number of concurrent session 4

➢ Manager Group

- Password will expire with 120 days
- Number of failed login 2
- Number of concurrent session 1

➢ Audit Group

- Password will expire with 20 days
- Number of failed login 1
- Number of concurrent session 2

➢ Data entry Group

- Password will expire with 40 days
- Number of failed login 2
- Number of concurrent session 4

2.8 Auditing

An auditing system is implemented to monitor and record user database actions comprehensively. This system tracks and logs various user actions within the database, based on specific criteria like the type of SQL statement used. To prioritize auditing efforts, key tables that are important and critical are selected.

Auditing these tables allows for effective monitoring of changes, updates, and access to critical information, enhancing security and accountability. The mentioned Tables are:

- **Loan_request Table**
- **Loan_request_lender Table**
- **Repayment Table**

2.9 Database Architecture

- **Control Files**
  - Control files are mirrored and distributed across multiple devices.
  - CTL on two diff ASM disk group {BACKUP, DATA }
  - Default MB (Auto Extend)
- **Redo Log Files**
  - Log files are mirrored and distributed across multiple devices.
  - DATA & BACKUP DISK GROUP
  - No of redo log group mirrors: 3
  - No of stand by redo log group :4 Archive Files
- **Archive Files**
  - Archiving is enabled.
  - BACKUP disk backup mirroring >> 1 copy & 1 copy for stand-by database

3.Data Dump

Dump (Logical): A logical dump refers to the process of creating a logical representation or snapshot of data.

Backup (Physical): A physical backup involves creating a copy or snapshot of the actual physical storage media or files that contain the data.

- We use utility expdp to pump data from table borrower
- we need user name with privilege on schema to pump data from borrower table

4.Hardware:

- **Primary Hard: 50G (installation files)**
  5 DISK GROUPS
- **OCR Normal Redundancy**

  Hard no (1) = 5G

  Hard no (2) = 5G

- **Data Normal Redundancy**

  Hard no (1) = 20G

  Hard no (2) = 20G

- **Backup Normal Redundancy**
  First Failure Group

  Hard no (1) = 10G

  Hard no (2) = 10G

  Second Failure Group

  Hard no (3) = 10G

  Hard no (4) = 10G

- **Data Standby**

  Hard no (1) = 20G

  Hard no (2) = 20G

- **Backup Standby**

  Hard no (1) = 20G

  Hard no (2) = 20G

5. Data Guard

What is the difference between Oracle RAC and Data Guard?

• RAC has one database and several instances associates with it, but data guard has several databases (one primary and others standby databases).

• RAC is the recommended solution for instance, software and hardware level failures. Data guard is the recommended solution for the SITE failures.

• Cluster ware software is used to keep the connection and communication between all nodes of the RAC, but in data guard, cluster ware software is not used. (if the data guard is not for a RAC)

• RAC must have a shared storage, which can be accessed from all the nodes of the system, but in data guard there is no shared storage, which is common for all the sites.

• RAC can have maximum of 100 nodes. Data guard can have maximum nine standby databases.

Why Data Guard Provides the Best Data Protection Some customers still use storage-based remote mirroring (array mirroring) to protect the Oracle Database with a replica. Storage mirroring is a sophisticated technology promoted as a generic infrastructure solution that makes a simple promise – whatever is written to primary storage will also be written to a mirrored copy at the remote site. Keeping this promise, however, can have disastrous consequences for data protection and availability when the data written to primary storage is corrupt due to failures such as hardware or software defects that checksum algorithms cannot detect. Modern cloud vendors also realize this and have stepped away from

storage mirroring in favor of Oracle Data Guard to protect Oracle Relational Databases, which clearly indicates the benefits of Oracle Data Guard. Oracle Data Guard and Active Data Guard provide better data protection and availability than is possible using storage technologies alone. Most enterprises have been replacing storage mirroring for Oracle databases with Oracle Active Data Guard for their business-critical databases

Why Data Guard Provides Better Availability (HA) Remote storage mirroring cannot provide the same level of high availability as Data Guard and Active Data Guard, at least not for Oracle Databases. Fast, Automatic Failover Oracle Data Guard includes an automatic failover capability called Fast-Start Failover. For failover and switchover operations, applications can use the same client failover infrastructure as Oracle Real Application Clusters to resume their transactions at the new primary database. Oracle Data Guard Fast-Start Failover ensures that recovery point objectives (zero or a maximum allowable data loss threshold) are met. It safeguards against split-brain conditions (where two independent databases function as primary). Oracle Data Guard includes intelligent automation to reinstate the failed primary database as a synchronized standby, quickly returning the configuration to a protected state. In contrast, remote storage mirroring has no Oracle-integrated capability to automate database failover and application redirection to a standby database. Remote storage mirroring requires time-consuming reconfiguration and start-up procedures to arrive at a state comparable to that of an Oracle Data Guard standby before the failure occurs. For example, the standby

database volumes must be mounted when the primary database fails before starting the new primary database. These additional steps increase downtime and the risk of something else going wrong, which can lengthen the outage period. Database Rolling Maintenance Oracle Data Guard provides high availability while performing planned maintenance.

• Upgrades and many other types of maintenance can happen at the standby database.

• Once implemented and thoroughly tested with the new version, the standby database can transition to the primary role and serve the production workload.

• Total downtime is limited to the time required for the switchover.

• The standby can also be used to fully validate the new release before the switchover is performed without compromising data protection.

That is not possible using array mirroring. Performing maintenance in a rolling fashion and seamlessly using a standby database for preproduction testing is impossible using array mirroring.

Installation

System overview:

Os: oracle linux 7

Cluster software : oracle grid 19c

Data base software : oracle data base management system 19c

Os configurations:

Run the following command as root :

yum -y install oracle-database-preinstall-19c.x86_64

Switch to oracle user:

```
su - oracle
mv ~/.bash_profile ~/.bash_profile_bkp
vi ~/.bash_profile
# .bash_profile
# OS User: oracle
# Application: Oracle Database Software Owner
# Version: Oracle 19c
# ------------------------------------------------
# Get the aliases and functions
if [ -f ~/.bashrc ]; then
. ~/.bashrc
fi
ORACLE_BASE=/u01/app/oracle; export ORACLE_BASE
ORACLE_SID=oradb; export ORACLE_SID
ORACLE_HOME=$ORACLE_BASE/product/19.0.0/db_1; export ORACLE_HOME
NLS_DATE_FORMAT="DD-MON-YYYY HH24:MI:SS"; export NLS_DATE_FORMAT
TNS_ADMIN=$ORACLE_HOME/network/admin; export TNS_ADMIN
PATH=$PATH:$HOME/.local/bin:$HOME/bin
PATH=${PATH}:/usr/bin:/bin:/usr/local/bin
PATH=.:${PATH}:$ORACLE_HOME/bin
export PATH
LD_LIBRARY_PATH=$ORACLE_HOME/lib
LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:$ORACLE_HOME/oracm/lib
LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:/lib:/usr/lib:/usr/local/lib
```

export LD_LIBRARY_PATH

CLASSPATH=$ORACLE_HOME/JRE

CLASSPATH=${CLASSPATH}:$ORACLE_HOME/jlib

CLASSPATH=${CLASSPATH}:$ORACLE_HOME/rdbms/jlib

CLASSPATH=${CLASSPATH}:$ORACLE_HOME/network/jlib

export CLASSPATH

export TEMP=/tmp

export TMPDIR=/tmp

umask 022

Switch back to root user and run the following:

su -

groupadd asmadmin

groupadd asmdba

usermod -a -G asmdba oracle

useradd -u 54323 -g oinstall -G asmadmin,asmdba grid

passwd grid

mkdir -p /u01/app/oracle/product/19.0.0/db_1

mkdir -p /u01/app/grid

mkdir -p /u01/app/19.0.0/grid

chown -R grid:oinstall /u01

chown -R oracle:oinstall /u01/app/oracle

chmod -R 775 /u01

Switch to grid user and edit the bash profile:

```
su - grid
mv ~/.bash_profile ~/.bash_profile_bkp
vi ~/.bash_profile
# .bash_profile
# Get the aliases and functions
if [ -f ~/.bashrc ]; then
. ~/.bashrc
fi
ORACLE_SID=+ASM; export ORACLE_SID
ORACLE_BASE=/u01/app/grid; export ORACLE_BASE
ORACLE_HOME=/u01/app/19.0.0/grid; export ORACLE_HOME
ORACLE_TERM=xterm; export ORACLE_TERM
TNS_ADMIN=$ORACLE_HOME/network/admin; export TNS_ADMIN
PATH=.:${JAVA_HOME}/bin:${PATH}:$HOME/bin:$ORACLE_HOME/bin
PATH=${PATH}:/usr/bin:/bin:/usr/local/bin
export PATH
export TEMP=/tmp
export TMPDIR=/tmp
umask 022
```

Switch back to root user :

```
su -
yum install oracleasm-support
yum install kmod-oracleasm
oracleasm configure -i
```

Configuring the Oracle ASM library driver.

This will configure the on-boot properties of the Oracle ASM library driver. The following questions will determine whether the driver is loaded on boot and what permissions it will have. The current values will be shown in brackets ('[]'). Hitting <ENTER> without typing an answer will keep that current value. Ctrl-C will abort.

Default user to own the driver interface []: grid

Default group to own the driver interface []: oinstall

Start Oracle ASM library driver on boot (y/n) [n]: y

Scan for Oracle ASM disks on boot (y/n) [y]: y

Writing Oracle ASM library driver configuration: done

Load the oracleasm kernel module:

/usr/sbin/oracleasm init

Now you can start formatting the needed disks for the asm :

fdisk -l | grep "Disk /dev/sd"

fdisk /dev/sdb

press: n,p,1,enter,enter,w

oracleasm createdisk OCR /dev/sdb1

oracleasm listdisks

Changing Kernel Parameter Values :

vi /etc/sysctl.d/97-oracle-database-sysctl.conf

fs.aio-max-nr = 1048576

fs.file-max = 6815744

kernel.shmall = 2097152

kernel.shmmax = 4294967295

kernel.shmmni = 4096

kernel.sem = 250 32000 100 128

net.ipv4.ip_local_port_range = 9000 65500

net.core.rmem_default = 262144

net.core.rmem_max = 4194304

net.core.wmem_default = 262144

net.core.wmem_max = 1048576

/sbin/sysctl –system

Reboot

yum install ksh

yum install libaio-devel.x86_64

Now, OS configuration is finished , next will start the grid installation and create the first asm group:

su - grid

unzip LINUX.X64_193000_grid_home.zip -d $ORACLE_HOME

exit to return back to the root shell:

exit

cd /u01/app/19.0.0/grid/cv/rpm/

CVUQDISK_GRP=oinstall; export CVUQDISK_GRP

rpm -iv cvuqdisk-1.0.10-1.rpm

cd $ORACLE_HOME

./gridSetup.sh

Window	Action
Configuration Option	Select the following option: **"Configure Oracle Grid Infrastructure for a Standalone Server (Oracle Restart)"**
Create ASM Disk Group	1. Click on **Change Discovery Path** button 2. Enter the **Discovery Path** as follows: `/dev/oracleasm/disks/*` 3. Fill in the fields as follows: Disk Group Name: OCRDISK
	Redundancy: External Select Disks: OCRDISK1
ASM Password	Enter the password
Management Option	Make sure the Checkbox is unselected
Operating System Groups	Make sure the following are the selected values: OSASM: **asmadmin** OSDBA: **asmdba**
Installation Location	Oracle Base and Oracle Grid Home should automatically point to the values of their corresponding variables. **Note:** Observe the grid home is not under the Oracle grid base home.
Create Inventory	It should automatically point to /u01/app/oraInventory
Root Script Execution	**Mark the checkbox** "Automatically run configuration scripts" and enter the root password
Prerequisite Checks	All the Prerequisite Checks should pass except the memory. It complains the available memory is 7.5. We can ignore this warning. Select **Ignore All** checkbox then click on **Next** button. Click **Yes** on the confirmation dialog box. **Note**: If you see other warnings, you have to resolve them before you proceed.
Install Product	When the installation reaches to nearly 11%, it will display a confirmation message. Click on **Yes** button.

Now we check the asm instance status and can add further more disk groups and configure them using the following commands:

crsctl status resource -t

asmca

Switch to oracle user to start the database software installation:

su - oracle

unzip LINUX.X64_193000_db_home.zip -d $ORACLE_HOME

cd $ORACLE_HOME

./runInstaller

Window	Action
Configuration Option	Select the following option: **"Create and Configure a single instance database."**
System Class	Select the following option: **"Server Class"**
Database Edition	Select the following option: **"Enterprise Edition"**
Installation Location	Keep the default value
Configuration Type	Select the following option: **General Purpose**
Database Identifiers	Global Database Name: **oradb.localdomain** Oracle SID: **oradb** Pluggable Database Name: **pdb1**

Configuration Options	Do **not** mark the AMM checkbox **Memory**: 5120 MB **Character set**: Use Unicode (AL32UTF8) **Sample Schemas**: (optional) **Mark the checkbox** "Install sample schema in the database"
Database Storage	Make sure ASM is selected
Management Options	Make sure the checkbox is **not** marked.
Recovery Option	Mark the checkbox **Enable Recovery** Make sure ASM is selected
ASM Diskgroup	Select **DATADISK**
Schema Password	Set passwords for the accounts
Operating System Groups	Select the "**oinstall**" group for all the options, except the OSOPER keep it blank.
Root Script Execution	**Mark the checkbox** "Automatically run configuration scripts" and enter the root password
Prerequisite Checks	All the Prerequisite Checks should pass.
Summary	Click on **Install** button
Install Product	When the installation reaches to nearly 12%, if will display a confirmation message. Click on **Yes** button.
Finish	click on **Close** button

Check the database status:

srvctl status database -d orcl

sqlplus / as sysdba

Implementation

Create the schema owner:

CREATE USER loanDBA IDENTIFIED BY passwd ;

-- Creating Tablespaces

-- Data_Entry Tablespace

CREATE TABLESPACE DATA_ENTRY

DATAFILE

'+DATA/ORCL/DATAFILE/data_entry1' SIZE 300M,'+DATA/ORCL/DATAFILE/data_entry2' SIZE 300M

AUTOEXTEND OFF

LOGGING

DEFAULT

NO INMEMORY

EXTENT MANAGEMENT LOCAL AUTOALLOCATE

BLOCKSIZE 8K

SEGMENT SPACE MANAGEMENT AUTO

FLASHBACK ON;

-- Implementer Tablespace

CREATE TABLESPACE IMPLEMENTERS

DATAFILE

'+DATA/ORCL/DATAFILE/implementers1' SIZE 40M AUTOEXTEND OFF,

'+DATA/ORCL/DATAFILE/implementers2' SIZE 40M AUTOEXTEND OFF,

'+DATA/ORCL/DATAFILE/implementers3' SIZE 40M AUTOEXTEND OFF

LOGGING

DEFAULT

NO INMEMORY

EXTENT MANAGEMENT LOCAL AUTOALLOCATE

BLOCKSIZE 8K

SEGMENT SPACE MANAGEMENT AUTO

FLASHBACK ON;

-- Creating Temp Tablespaces

CREATE TEMPORARY TABLESPACE TEMP_DBA

TEMPFILE

'+DATA/ORCL/TEMPFILE/tempDBA' SIZE 20M AUTOEXTEND OFF

TABLESPACE GROUP ''

EXTENT MANAGEMENT LOCAL UNIFORM SIZE 1M

FLASHBACK ON;

ALTER DATABASE DEFAULT TEMPORARY TABLESPACE TEMP_DBA;

CREATE TEMPORARY TABLESPACE TEMP_Data_Entry

TEMPFILE

'+DATA/ORCL/TEMPFILE/tempfile1' SIZE 10M AUTOEXTEND OFF

TABLESPACE GROUP ''

EXTENT MANAGEMENT LOCAL UNIFORM SIZE 1M

FLASHBACK ON;

```
CREATE TEMPORARY TABLESPACE TEMP_Implementer
TEMPFILE
  '+DATA/ORCL/TEMPFILE/tempfile2' SIZE 10M AUTOEXTEND OFF
TABLESPACE GROUP ''
EXTENT MANAGEMENT LOCAL UNIFORM SIZE 1M
FLASHBACK ON;

CREATE TEMPORARY TABLESPACE TEMP_Audit
TEMPFILE
  '+DATA/ORCL/TEMPFILE/tempfile3' SIZE 10M AUTOEXTEND OFF
TABLESPACE GROUP ''
EXTENT MANAGEMENT LOCAL UNIFORM SIZE 1M
FLASHBACK ON;

CREATE TEMPORARY TABLESPACE TEMP_Manager
TEMPFILE
  '+DATA/ORCL/TEMPFILE/tempfile4' SIZE 10M AUTOEXTEND OFF
TABLESPACE GROUP ''
EXTENT MANAGEMENT LOCAL UNIFORM SIZE 1M
FLASHBACK ON;
--undo tablespace default one alter it
```

-- Creating PROFILES

```
-- DATA_ENTRY_PROFILE
CREATE PROFILE DATA_ENTRY_PROFILE
LIMIT
```

SESSIONS_PER_USER 2 CONNECT_TIME 100000 IDLE_TIME 100000

FAILED_LOGIN_ATTEMPTS 3

PASSWORD_LIFE_TIME 50

PASSWORD_LOCK_TIME 30

PASSWORD_GRACE_TIME 5;

-- IMPLEMENTER_PROFILE

CREATE PROFILE IMPLEMENTER_PROFILE

LIMIT SESSIONS_PER_USER 1 CONNECT_TIME 100000 IDLE_TIME 100000

FAILED_LOGIN_ATTEMPTS 2

PASSWORD_LIFE_TIME 120

PASSWORD_LOCK_TIME 100

PASSWORD_GRACE_TIME 5;

-- AUDIT_PROFILE

CREATE PROFILE AUDIT_PROFILE

LIMIT SESSIONS_PER_USER 2 CONNECT_TIME 100000 IDLE_TIME 100000

FAILED_LOGIN_ATTEMPTS 1

PASSWORD_LIFE_TIME 20

PASSWORD_LOCK_TIME 15

PASSWORD_GRACE_TIME 5;

-- MANAGER_PROFILE

CREATE PROFILE MANAGER_PROFILE

LIMIT SESSIONS_PER_USER 4 CONNECT_TIME 100000 IDLE_TIME 100000

FAILED_LOGIN_ATTEMPTS 5

PASSWORD_LIFE_TIME 150

PASSWORD_LOCK_TIME 130

PASSWORD_GRACE_TIME 5;

--DBA

ALTER USER loanDBA TEMPORARY TABLESPACE temp_dba;

CREATE USER loanDBA2 IDENTIFIED BY passwd temporary tablespace temp_dba;

GRANT CONNECT,RESOURCE,DBA,sysdba TO loanDBA;

GRANT CONNECT, RESOURCE, DBA,sysdba TO loanDBA2;

grant CREATE MATERIALIZED VIEW to loandba;

grant CREATE MATERIALIZED VIEW to loandba2;

grant CREATE ANY TABLE to loandba;

grant CREATE ANY TABLE to loandba2;

--LOAN SCHEMA

--Tables Creation Script with loanDBA Schema:

CREATE SEQUENCE loanDBA.seq_Addressee;

CREATE SEQUENCE loanDBA.seq_Borrower;

CREATE SEQUENCE loanDBA.seq_Lender;

CREATE SEQUENCE loanDBA.seq_Intermediary;

CREATE SEQUENCE loanDBA.seq_Loan_Request;

-- Table Addressee

CREATE TABLE loanDBA.Addressee (

Addressee_ID number(10) NOT NULL,

```
  Name    varchar2(255),
  Address varchar2(255),
  PRIMARY KEY (Addressee_ID))
 TABLESPACE DATA_ENTRY;
-- Table Borrower
CREATE TABLE loanDBA.Borrower (
  Borrower_ID         number(10) NOT NULL,
  Addressee_ID number(10) NOT NULL,
  PRIMARY KEY (Borrower_ID))
TABLESPACE DATA_ENTRY;
-- Table Lender
CREATE TABLE loanDBA.Lender (
  Lender_ID         number(10) NOT NULL,
  Addressee_ID number(10) NOT NULL,
  PRIMARY KEY (Lender_ID))
TABLESPACE DATA_ENTRY;
-- Table Intermediary
CREATE TABLE loanDBA.Intermediary (
  Intermediary_ID         number(10) NOT NULL,
  Addressee_ID number(10) NOT NULL,
  Loan_Date    date NOT NULL,
  PRIMARY KEY (Intermediary_ID))
TABLESPACE DATA_ENTRY;
```

```
-- Table Loan_Request
CREATE TABLE loanDBA.Loan_Request (
  Loan_Request_ID          number(10) NOT NULL,
  Borrower_ID  number(10) NOT NULL,
  Loan_Request_Date      date NOT NULL,
  Deadline    date,
  Loan_Request_Amount      number(19),
  Description varchar2(255),
  Payday      date,
  PRIMARY KEY (Loan_Request_ID))
TABLESPACE DATA_ENTRY;
-- Table Loan
CREATE TABLE loanDBA.Loan (
  Loan_Date              date NOT NULL,
  Loan_Request_ID    number(10) NOT NULL,
  Deadline_Agreed_Date date NOT NULL,
  Repayment_Date       date NOT NULL,
  PRIMARY KEY (Loan_Date))
TABLESPACE IMPLEMENTERS;
-- Table Repayment
CREATE TABLE loanDBA.Repayment (
  Repayment_Date date NOT NULL,
  Repayment_Amount number(19),
  PRIMARY KEY (Repayment_Date))
TABLESPACE IMPLEMENTERS;
```

```
-- Table Deadline
CREATE TABLE loanDBA.Deadline (
  Agreed_Date date NOT NULL,
  PRIMARY KEY (Agreed_Date))
TABLESPACE IMPLEMENTERS;

-- Table Lender_Borrower
CREATE TABLE loanDBA.Lender_Borrower (
  Borrower_ID          number(10) NOT NULL,
  Lender_ID            number(10) NOT NULL,
  Loan_Date            date NOT NULL,
  Percentage           number(10),
  PRIMARY KEY (Borrower_ID,
  Lender_ID))
TABLESPACE IMPLEMENTERS;

-- Table Loan_Request_Lender
CREATE TABLE loanDBA.Loan_Request_Lender (
  Lender_ID        number(10) NOT NULL,
  Loan_Request_ID number(10) NOT NULL,
  Lender_Amount          number(19),
  PRIMARY KEY (Lender_ID,
  Loan_Request_ID))
TABLESPACE IMPLEMENTERS;
```

--Foreign keys

ALTER TABLE loanDBA.Borrower ADD CONSTRAINT FK_Borrower FOREIGN KEY (Addressee_ID) REFERENCES loanDBA.Addressee (Addressee_ID);

ALTER TABLE loanDBA.Lender ADD CONSTRAINT FK_Lender FOREIGN KEY (Addressee_ID) REFERENCES loanDBA.Addressee (Addressee_ID);

ALTER TABLE loanDBA.Intermediary ADD CONSTRAINT FK_Intermediary FOREIGN KEY (Addressee_ID) REFERENCES loanDBA.Addressee (Addressee_ID);

ALTER TABLE loanDBA.Intermediary ADD CONSTRAINT FK_Intermediary_Loan_Date FOREIGN KEY (Loan_Date) REFERENCES loanDBA.Loan (Loan_Date);

ALTER TABLE loanDBA.Loan_Request ADD CONSTRAINT FKLoan_Req_borrower FOREIGN KEY (Borrower_ID) REFERENCES loanDBA.Borrower (Borrower_ID);

ALTER TABLE loanDBA.Lender_Borrower ADD CONSTRAINT FKLender_Borrower_lender FOREIGN KEY (Lender_ID) REFERENCES loanDBA.Lender (Lender_ID);

ALTER TABLE loanDBA.Lender_Borrower ADD CONSTRAINT FKLender_Borrower_borrower FOREIGN KEY (Borrower_ID) REFERENCES loanDBA.Borrower (Borrower_ID);

ALTER TABLE loanDBA.Lender_Borrower ADD CONSTRAINT FKLender_Borrower_loandate FOREIGN KEY (Loan_Date) REFERENCES loanDBA.Loan (Loan_Date);

ALTER TABLE loanDBA.Loan ADD CONSTRAINT FKLoan_Deadline FOREIGN KEY (Deadline_Agreed_Date) REFERENCES loanDBA.Deadline (Agreed_Date);

ALTER TABLE loanDBA.Loan ADD CONSTRAINT FKLoan_Repayment_date FOREIGN KEY (Repayment_Date) REFERENCES loanDBA.Repayment (Repayment_Date);

ALTER TABLE loanDBA.Loan ADD CONSTRAINT FKLoan_loan_Request_ID FOREIGN KEY (Loan_Request_ID) REFERENCES loanDBA.Loan_Request (Loan_Request_ID);

ALTER TABLE loanDBA.Loan_Request_Lender ADD CONSTRAINT FKLoan_Request_Lender_l_ID FOREIGN KEY (Lender_ID) REFERENCES loanDBA.Lender (Lender_ID);

ALTER TABLE loanDBA.Loan_Request_Lender ADD CONSTRAINT FKLoan_Req_Lender_l_Request FOREIGN KEY (Loan_Request_ID) REFERENCES loanDBA.Loan_Request (Loan_Request_ID);

--Creating Views

-- Borrowers_View

```
CREATE VIEW loanDBA.Borrowers_View AS

SELECT b.Borrower_ID as "Borrower_ID",ad.NAME as "Borrower_Name",b.Addressee_ID as "Borrower_Address_ID",ad.Address as "Borrower_Address"

FROM loanDBA.Addressee ad,loanDBA.borrower b

WHERE ad.Addressee_ID=b.Addressee_ID;
```

-- Lenders_View

```
CREATE VIEW loanDBA.Lenders_View AS

SELECT l. Lender_ID as " Lender_ID",ad.NAME as " Lender_Name",l.Addressee_ID as " Lender_Address_ID",ad.Address as " Lender_Address"

FROM loanDBA.Addressee ad, loanDBA.Lender l

WHERE ad.Addressee_ID=l.Addressee_ID;
```

-- Intermediaries_View

```
CREATE VIEW loanDBA.Intermediaries_View AS

SELECT i.Intermediary_ID as "Intermediary_ID",ad.NAME as " Intermediary_Name",i.Addressee_ID as " Intermediary_Address_ID",ad.Address as "Intermediary_Address"

FROM loanDBA.Addressee ad,loanDBA.Intermediary i

WHERE ad.Addressee_ID=i.Addressee_ID;
```

-- active_loans_view

```
create view loanDBA.active_loans_view as

Select loan_date, deadline_agreed_date from loandba.loan where repayment_date is null and sysdate<deadline_agreed_date ;
```

```
-- Intermediary_Borrower_View

CREATE VIEW loanDBA.Intermediary_Borrower_View AS

select distinct lb.loan_date, ad_i.name as "Intermediary Name",ad_b.name as"Borrower
Name"

from loandba.intermediary i , loandba.addressee ad_i ,loandba.addressee ad_b
,loandba.lender_borrower lb , loandba.borrower b

where ad_i.addressee_id=i.addressee_id

and ad_b.addressee_id=b.addressee_id

and i.loan_date =lb.loan_date

and b.borrower_id=lb.borrower_id;

-- lenders_share_view

CREATE VIEW loanDBA.lenders_share_view AS

Select lb.lender_id,ad.name,lb.loan_date,lrl.lender_amount

From loandba.lender_borrower lb,loandba.loan_request_lender lrl ,loandba.lender l,
loandba.addressee ad

Where lrl.lender_id =lb.lender_id

And lb.lender_id =l.lender_id

And l.addressee_id=ad.addressee_id

And lb.loan_date between sysdate-120 and sysdate  ;

-- loan_lenders_view

CREATE VIEW loanDBA.loan_lenders_view AS

Select l.loan_date,lr.loan_request_amount ,count(distinct lrl.lender_id)as num_lenders

from loandba.loan l ,loandba.loan_request lr ,loandba.loan_request_lender lrl

where lr.loan_request_id =l.loan_request_id

and lrl.loan_request_id= lr.loan_request_id

group by l.loan_date,lr.loan_request_amount
```

order by num_lenders desc;

--Creating Materialized Views

-- yearly_cash_flow_Mview

```
create materialized view loanDBA.yearly_cash_flow_Mview
refresh complete start with trunc(add_months(trunc(sysdate, 'yyyy'), 12), 'yyyy') - 1
next trunc(add_months(trunc(sysdate, 'yyyy'), 24), 'yyyy') - 1
as
select sum(repayment.repayment_amount) - sum(loan_request.loan_request_amount) as
"cash flow"
from loandba.loan_request, loandba.repayment
where  to_char(loan_request_date, 'YYYY')  = (select  to_char(SYSDATE, 'YYYY') from dual)
and to_char(repayment_date, 'YYYY')  = (select  to_char(SYSDATE, 'YYYY') from dual) ;
```

-- yearly_percent_loan_Mview

```
--loan_request must contains data to avoid dividing on zero error****
create materialized view loanDBA.yearly_percent_loan_Mview
refresh complete start with trunc(add_months(trunc(sysdate, 'YEAR'), 5), 'MM')
next trunc(add_months(trunc(sysdate, 'YEAR'), 17), 'MM')
as
select(
(select count(*) from loandba.loan_request where loan_request_amount > 200000 and
to_char(loan_request_date, 'YYYY')
  = (select  to_char(SYSDATE, 'YYYY') from dual) ) /
(select count(*) from loandba.loan_request where  to_char(loan_request_date, 'YYYY')  =
(select  to_char(SYSDATE, 'YYYY') from dual) )*100)
as percent_loan_requests from dual ;
```

--Creating Triggers

--Borrowers Trigger

```
CREATE OR REPLACE TRIGGER loandba.Borrowers_View_trig
INSTEAD OF INSERT ON loandba.Borrowers_View
FOR EACH ROW
BEGIN
  INSERT INTO loandba.Addressee (ADDRESSEE_ID, NAME, ADDRESS)
   VALUES (:new."Borrower_Address_ID", :new."Borrower_Name", :new."Borrower_Address");
   INSERT INTO loandba.borrower (BORROWER_ID,ADDRESSEE_ID) VALUES
(:new."Borrower_ID",:new."Borrower_Address_ID");
END;
```

--Lenders Trigger

```
CREATE OR REPLACE TRIGGER loandba.Lenders_View_trig
INSTEAD OF INSERT ON loandba.Lenders_View
FOR EACH ROW
BEGIN
  INSERT INTO loandba.Addressee (ADDRESSEE_ID, NAME, ADDRESS)
   VALUES (:new." Lender_Address_ID", :new." Lender_Name", :new." Lender_Address");

 INSERT INTO loandba.Lender (Lender_ID,ADDRESSEE_ID) VALUES (:new." Lender_ID",:new."
Lender_Address_ID");
END;
```

--Intermediaries Trigger

```
CREATE OR REPLACE TRIGGER loanDBA.Intermediaries_View_trig
INSTEAD OF INSERT ON loanDBA.Intermediaries_View
FOR EACH ROW
```

BEGIN

INSERT INTO loandba.Addressee (ADDRESSEE_ID, NAME, ADDRESS)

VALUES (:new." Intermediary_Address_ID", :new." Intermediary_Name", :new."Intermediary_Address");

INSERT INTO loandba.intermediary(INTERMEDIARY_ID, ADDRESSEE_ID) VALUES (:new."Intermediary_ID",:new." Intermediary_Address_ID");

END;

--Create rest of users

CREATE USER borrower_request1 IDENTIFIED BY borrower1 DEFAULT TABLESPACE DATA_ENTRY

TEMPORARY TABLESPACE TEMP_Data_Entry

QUOTA 100M ON DATA_ENTRY

PROFILE DATA_ENTRY_PROFILE;

CREATE USER borrower_request2 IDENTIFIED BY borrower2 DEFAULT TABLESPACE DATA_ENTRY

TEMPORARY TABLESPACE TEMP_Data_Entry

QUOTA 100M ON DATA_ENTRY

PROFILE DATA_ENTRY_PROFILE;

CREATE USER loan_lender IDENTIFIED BY lender DEFAULT TABLESPACE DATA_ENTRY

TEMPORARY TABLESPACE TEMP_Data_Entry

QUOTA 100M ON DATA_ENTRY

PROFILE DATA_ENTRY_PROFILE;

CREATE USER Loan_intermediary IDENTIFIED BY intermed DEFAULT TABLESPACE DATA_ENTRY

TEMPORARY TABLESPACE TEMP_Data_Entry

QUOTA 50M ON DATA_ENTRY

PROFILE DATA_ENTRY_PROFILE;

CREATE USER Implementer1 IDENTIFIED BY implementer1 DEFAULT TABLESPACE IMPLEMENTERS

TEMPORARY TABLESPACE TEMP_Implementer

QUOTA 30M ON IMPLEMENTERS

QUOTA 20M ON DATA_ENTRY

PROFILE IMPLEMENTER_PROFILE;

CREATE USER Implementer2 IDENTIFIED BY implementer2 DEFAULT TABLESPACE IMPLEMENTERS

TEMPORARY TABLESPACE TEMP_Implementer

QUOTA 30M ON IMPLEMENTERS

QUOTA 20M ON DATA_ENTRY

PROFILE IMPLEMENTER_PROFILE;

CREATE USER Implementer3 IDENTIFIED BY implementer3 DEFAULT TABLESPACE IMPLEMENTERS

TEMPORARY TABLESPACE TEMP_Implementer

QUOTA 30M ON IMPLEMENTERS

QUOTA 20M ON DATA_ENTRY

PROFILE IMPLEMENTER_PROFILE;

CREATE USER Implementer4 IDENTIFIED BY implementer4 DEFAULT TABLESPACE IMPLEMENTERS

TEMPORARY TABLESPACE TEMP_Implementer

QUOTA 30M ON IMPLEMENTERS

QUOTA 20M ON DATA_ENTRY

PROFILE IMPLEMENTER_PROFILE;

CREATE USER Audit_borrower IDENTIFIED BY aud_br DEFAULT TABLESPACE DATA_ENTRY

TEMPORARY TABLESPACE TEMP_Audit

PROFILE AUDIT_PROFILE;

CREATE USER Audit_lender IDENTIFIED BY aud_le DEFAULT TABLESPACE IMPLEMENTERS

TEMPORARY TABLESPACE TEMP_Audit

PROFILE AUDIT_PROFILE;

CREATE USER Audit_intermediary IDENTIFIED BY aud_inter DEFAULT TABLESPACE DATA_ENTRY

TEMPORARY TABLESPACE TEMP_Audit

PROFILE AUDIT_PROFILE;

CREATE USER Audit_loan IDENTIFIED BY aud_loan DEFAULT TABLESPACE IMPLEMENTERS

TEMPORARY TABLESPACE TEMP_Audit

PROFILE AUDIT_PROFILE;

CREATE USER manager_accept IDENTIFIED BY man_acc DEFAULT TABLESPACE IMPLEMENTERS

TEMPORARY TABLESPACE TEMP_Manager

PROFILE MANAGER_PROFILE;

CREATE USER manager_generate IDENTIFIED BY man_gen DEFAULT TABLESPACE IMPLEMENTERS

TEMPORARY TABLESPACE TEMP_Manager

PROFILE MANAGER_PROFILE;

--Creating Roles

CREATE ROLE borrower_request_role;

GRANT SELECT,INSERT,UPDATE ON loanDBA.Loan_request TO borrower_request_role;

GRANT SELECT,INSERT,UPDATE ON loanDBA.borrower TO borrower_request_role;

GRANT INSERT ON loanDBA.Addressee TO borrower_request_role;

GRANT SELECT ON loanDBA.borrowers_view TO borrower_request_role;

CREATE ROLE loan_implementer_role;

GRANT SELECT,INSERT,UPDATE ON loanDBA.Loan_request TO loan_implementer_role;

GRANT SELECT,INSERT,UPDATE ON loanDBA.Loan_request_lender TO loan_implementer_role;

GRANT SELECT,INSERT,UPDATE ON loanDBA.Loan TO loan_implementer_role;

GRANT SELECT,INSERT,UPDATE ON loanDBA.Lender_borrower TO loan_implementer_role;

GRANT SELECT,INSERT,UPDATE ON loanDBA.Repayment TO loan_implementer_role;

GRANT SELECT,INSERT,UPDATE ON loanDBA.deadline TO loan_implementer_role;

CREATE ROLE BORROWER_ROLE;

GRANT SELECT ON loanDBA.BORROWER TO BORROWER_ROLE;

CREATE ROLE ADDRESSEE_ROLE;

GRANT SELECT ON loanDBA.ADDRESSEE TO ADDRESSEE_ROLE;

CREATE ROLE LENDER_ROLE;

GRANT SELECT ON loanDBA.LENDER TO LENDER_ROLE;

CREATE ROLE LOAN_REQUEST_ROLE;

GRANT SELECT ON loanDBA.LOAN_REQUEST TO LOAN_REQUEST_ROLE;

CREATE ROLE LOAN_REQUEST_LENDER_ROLE;

GRANT SELECT ON loanDBA.LOAN_REQUEST_LENDER TO LOAN_REQUEST_LENDER_ROLE;

CREATE ROLE LENDER_BORROWER_ROLE;

GRANT SELECT ON loanDBA.LENDER_BORROWER TO LENDER_BORROWER_ROLE;

CREATE ROLE INTERMEDIARY_ROLE;

GRANT SELECT ON loanDBA.INTERMEDIARY TO INTERMEDIARY_ROLE;

CREATE ROLE REPAYMENT_ROLE;

GRANT SELECT ON loanDBA.REPAYMENT TO REPAYMENT_ROLE;

CREATE ROLE LOAN_ROLE;

GRANT SELECT ON loanDBA.LOAN TO LOAN_ROLE;

CREATE ROLE DEADLINE_ROLE;

GRANT SELECT ON loanDBA.DEADLINE TO DEADLINE_ROLE;

--Grant roles to users

GRANT borrower_request_role TO borrower_request1;

GRANT borrower_request_role TO borrower_request2;

GRANT loan_implementer_role TO implementer1;

GRANT loan_implementer_role TO implementer2;

GRANT loan_implementer_role TO implementer3;

GRANT loan_implementer_role TO implementer4;

GRANT BORROWER_ROLE TO Audit_borrower;

GRANT ADDRESSEE_ROLE TO Audit_borrower;

GRANT LOAN_REQUEST_ROLE TO Audit_borrower;

GRANT LENDER_ROLE TO Audit_lender;

GRANT ADDRESSEE_ROLE TO Audit_lender;

GRANT LENDER_BORROWER_ROLE TO Audit_lender;

GRANT LOAN_REQUEST_ROLE TO Audit_lender;

GRANT INTERMEDIARY_ROLE TO Audit_intermediary;

GRANT ADDRESSEE_ROLE TO Audit_intermediary;

```
GRANT BORROWER_ROLE TO Audit_intermediary;
GRANT LENDER_BORROWER_ROLE TO Audit_intermediary;

GRANT LOAN_ROLE TO Audit_loan;
GRANT LOAN_REQUEST_ROLE TO Audit_loan;
GRANT LENDER_BORROWER_ROLE TO Audit_loan;
GRANT REPAYMENT_ROLE TO Audit_loan;
GRANT DEADLINE_ROLE TO Audit_loan;

GRANT BORROWER_ROLE TO manager_accept;
GRANT ADDRESSEE_ROLE TO manager_accept;
GRANT LENDER_ROLE TO manager_accept;
GRANT INTERMEDIARY_ROLE TO manager_accept;
GRANT LOAN_REQUEST_LENDER_ROLE TO manager_accept;
GRANT LOAN_ROLE TO manager_accept;
GRANT LOAN_REQUEST_ROLE TO manager_accept;
GRANT LENDER_BORROWER_ROLE TO manager_accept;
GRANT REPAYMENT_ROLE TO manager_accept;
GRANT DEADLINE_ROLE TO manager_accept;

GRANT BORROWER_ROLE TO manager_generate;
GRANT ADDRESSEE_ROLE TO manager_generate;
GRANT LENDER_ROLE TO manager_generate;
GRANT INTERMEDIARY_ROLE TO manager_generate;
GRANT LOAN_REQUEST_LENDER_ROLE TO manager_generate;
```

GRANT LOAN_ROLE TO manager_generate;

GRANT LOAN_REQUEST_ROLE TO manager_generate;

GRANT LENDER_BORROWER_ROLE TO manager_generate;

GRANT REPAYMENT_ROLE TO manager_generate;

GRANT DEADLINE_ROLE TO manager_generate;

--Grant individual privileges to users

GRANT CONNECT TO borrower_request1;

GRANT CONNECT TO borrower_request2;

GRANT CONNECT TO loan_lender;

GRANT CONNECT TO Loan_intermediary;

GRANT CONNECT TO Implementer1;

GRANT CONNECT TO Implementer2;

GRANT CONNECT TO Implementer3;

GRANT CONNECT TO Implementer4;

GRANT CONNECT TO Audit_borrower;

GRANT CONNECT TO Audit_lender;

GRANT CONNECT TO Audit_intermediary;

GRANT CONNECT TO Audit_loan;

GRANT CONNECT TO manager_accept;

GRANT CONNECT TO manager_generate;

GRANT SELECT,INSERT,UPDATE ON loanDBA.Lender TO loan_lender;

GRANT INSERT ON loanDBA.Addressee TO loan_lender;

GRANT SELECT ON loanDBA.lenders_view TO loan_lender;

GRANT SELECT,INSERT,UPDATE ON loanDBA.Intermediary TO Loan_intermediary;

GRANT INSERT ON loanDBA.Addressee TO Loan_intermediary;

GRANT SELECT ON loanDBA.intermediaries_view TO Loan_intermediary;

GRANT SELECT ON loanDBA.active_loans_view TO Audit_loan;

GRANT SELECT ON loanDBA.Intermediary_Borrower_View TO Audit_intermediary;

GRANT SELECT ON loanDBA.loan_lenders_view TO Audit_loan;

GRANT SELECT ON loanDBA.lenders_share_view TO Audit_lender;

GRANT SELECT ON loanDBA.yearly_cash_flow_Mview TO manager_generate;

****GRANT SELECT ON loanDBA.yearly_percent_loan_Mview TO Audit_loan;**

--Generating Data

We use Toad data generation function to generate our sample of data with the DBMS_Random Package and some equations we made in order to adjust the data -especially the dates and amounts of money- to be more logic not just random.We generate 100 record for each table except for

- addressee table 300 record

(100 for each of borrower,lender and intermediary)

- Loan_request_lender table 300 record

(As we assume that 3 lenders share together in lending one loan, so the data for that loan is repeated 3 times for each lender)

The Equations we used:-

- Loan_request_date= rand(1-1-2015,sysdate);
- Loan_date= Loan_request_date+rand(1,10);

- Payday= rand(loan_date+1,loan_date+20);
- Request_deadline=rand(payday+1,2025)
- Agreed_deadline:=request_deadline - rand(0,50);
- Request_amount=rand(5000,1000000)
- Lender_amount=request_amount-rand(0,0.9)* request_amount;
- Repayment amount=request_amount+(percentage)* request_amount
- Percentage= rand(3,20)/100

--Scripts for Backup

SQLPLUS enable archivelog mode and flashback

shutdown immediate;

startup mount;

alter database archivelog;

alter database flashback on;

select flashback_on from v$database;

alter database open;

SQLPLUS - recovery catalog

Create tablespace cattbs DATAFILE '+BACKUP/orcl/datafile/cattbs' size 100M EXTENT MANAGEMENT LOCAL SEGMENT SPACE MANAGEMENT AUTO FLASHBACK ON;

Alter loandba2 QUOTA UNLIMITED ON cattbs;

GRANT recovery_catalog_owner TO loandba2;

RMAN

CONNECT CATALOG loandba2@orcl;

CREATE CATALOG;

CONNECT TARGET /

REGISTER DATABASE;

rman target / catalog loandba2/passwd@orcl;

CONFIGURE DEVICE TYPE DISK PARALLELISM 4 BACKUP TYPE TO BACKUPSET;

CONFIGURE RETENTION POLICY TO RECOVERY WINDOW OF 10 DAYS;

CONFIGURE BACKUP OPTIMIZATION ON;

CONFIGURE CHANNEL DEVICE TYPE DISK MAXPIECESIZE 300 M;

CONFIGURE EXCLUDE FOR TABLESPACE USERS;

Backup as copy database plus archivelog;

BASH

vi $ORACLE_BASE/incr0backup.sh

rman target / catalog rco/rco@orcl;

backup as compressed backupset incremental level 0 database tag 'inc' plus archivelog tag 'arch';

exit;

EOF

vi $ORACLE_BASE/incr1backup.sh

rman target / catalog rco/rco@orcl;

backup as compressed backupset incremental level 1 database tag 'inc' plus archivelog tag 'arch';

exit;

EOF

crontab -e

0 0 * * SAT $ORACLE_BASE/incr0backup.sh

0 23 * * * $ORACLE_BASE/incr1backup.sh

--DUMP BORROWER table

CREATE DIRECTORY dump AS '+BACKUP';

expdp loandba/passwd@orcl TABLES=borrower directory= dump dumpfile=dump_borrower.dmp nologfile=yes;

--Performance Tuning

--To insure maximum performance

- We gathered statistics about schema while db is not under load to improve performance

  EXEC DBMS_STATS.gather_schema_stats('loandba');

- We kept the default indexes on primary keys ids columns
- Took workload snapshots before and after each physical or logical change to database

  for AWR Report and ADDM Report

  EXEC DBMS_WORKLOAD_REPOSITORY.create_snapshot;

  select snap_id,BEGIN_INTERVAL_TIME,END_INTERVAL_TIME from dba_hist_snapshot order by BEGIN_INTERVAL_TIME desc;

  Sql> @$ORACLE_HOME/rdbms/admin/addmrpt.sql

  Sql> @$ORACLE_HOME/rdbms/admin/awrrpt.sql

- Take baseline at optimal states

  EXEC dbms_workload_repository.create_baseline(start_snap_id=>1, end_snap_id=>10, baseline_name=>'First baseline');

  EXEC dbms_workload_repository.drop_baseline(baseline_name=>'First baseline', cascade=>false);

  select BASELINE_NAME, START_SNAP_ID, END_SNAP_ID from dba_hist_baseline;

- We use performance view like

  select * from V$ACCESS where owner='loandba';

  --shows sessions that lock objects on loandba

  select * from V$SYSSTAT;

  --show statistcs

  select SQL_TEXT, CPU_TIME from V$SQLAREA ;

  --show sql statment consuming cpu

- Run top command on os to monitor usable resources
- Adjust undo retention according to policy

  ALTER SYSTEM SET UNDO_RETENTION = 10600;
- Tune size for temp tablespaces

```
-- Applying metric threshold for the full tablespace with warning 60 critical 80
 BEGIN
  -- Tablespace-specific percent full threshold.
  DBMS_SERVER_ALERT.set_threshold(
    metrics_id              => DBMS_SERVER_ALERT.tablespace_pct_full,
    warning_operator        => DBMS_SERVER_ALERT.operator_ge,
    warning_value           => '60',
    critical_operator       => DBMS_SERVER_ALERT.operator_ge,
    critical_value          => '80',
    observation_period      => 1,
    consecutive_occurrences => 1,
    instance_name           => 'orcl',
    object_type             => DBMS_SERVER_ALERT.object_type_tablespace,
    object_name             => 'DATA_ENTRY');
 END;
 /
 begin
  DBMS_SERVER_ALERT.set_threshold(
    metrics_id              => DBMS_SERVER_ALERT.tablespace_pct_full,
    warning_operator        => DBMS_SERVER_ALERT.operator_ge,
    warning_value           => '60',
```

```
    critical_operator       => DBMS_SERVER_ALERT.operator_ge,
    critical_value          => '80',
    observation_period      => 1,
    consecutive_occurrences => 1,
    instance_name           => 'orcl',
    object_type             => DBMS_SERVER_ALERT.object_type_tablespace,
    object_name             => 'IMPLEMENTERS');
END;
/
SET LINESIZE 200
COLUMN tablespace_name FORMAT A30
COLUMN metrics_name FORMAT A30
COLUMN warning_value FORMAT A30
COLUMN critical_value FORMAT A15

SELECT object_name AS tablespace_name,
       metrics_name,
       warning_operator,
       warning_value,
       critical_operator,
       critical_value
FROM   dba_thresholds
WHERE  object_type = 'TABLESPACE'
ORDER BY object_name;
```

--Scripts for Auditing

audit select ,delete , insert , update on loandba.LOAN_REQUEST by access ;

audit select ,delete , insert , update on loandba.REPAYMENT by access ;

audit select ,delete , insert , update on loandba.LOAN_REQUEST_LENDER by access ;

SELECT * FROM DBA_AUDIT_TRAIL;

--Scripts for Data Guard

Primary Server Setup

Logging

Check that the primary database is in archivelog mode.

select log_mode from v$database;

LOG_MODE

NOARCHIVELOG

If it is noarchivelog mode, switch is to archivelog mode.

shutdown immediate;

startup mount;

alter database archivelog;

alter database open;

--Create standby redo logs on the primary database (in case of switchovers). The standby redo logs should be at least as big as the largest online redo log and there should be one extra group per thread compared the online redo logs. In my case, the following standby redo logs must be created on both servers.

-- If Oracle Managed Files (OMF) is used.

alter database add standby logfile thread 1 group 10 size 50m;

alter database add standby logfile thread 1 group 11 size 50m;

alter database add standby logfile thread 1 group 12 size 50m;

alter database add standby logfile thread 1 group 13 size 50m;

--If you want to use flashback database, enable it on the primary now, so it will be enabled on the standby also. It's very useful as you will see below.

alter database flashback on;

Check the setting for the DB_NAME and DB_UNIQUE_NAME parameters. In this case they are both set to "cdb1" on the primary database.

SQL> show parameter db_name

NAME TYPE VALUE

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

db_name string orcl

SQL> show parameter db_unique_name

NAME TYPE VALUE

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

db_unique_name string orck

The DB_NAME of the standby database will be the same as that of the primary, but it must have a different DB_UNIQUE_NAME value. For this example, the standby database will have the value "orclstd".

Make sure the STANDBY_FILE_MANAGEMENT parameter is set.

alter system set standby_file_management=auto;

Service Setup

--Entries for the primary and standby databases are needed in the "$ORACLE_HOME/network/admin/tnsnames.ora" files on both servers.

--You can create these using the Network Configuration Utility (netca) or manually. The following entries were used during this setup.

--Notice the use of the SID, rather than the SERVICE_NAME in the entries.

--This is important as the broker will need to connect to the databases when they are down, so the services will not be present.

```
orcl=
 (DESCRIPTION =
   (ADDRESS_LIST =
     (ADDRESS = (PROTOCOL = TCP)(HOST = 192.168.56.106)(PORT = 1521))
   )
   (CONNECT_DATA =
     (SID = orcl)
   )
 )
orcl_std=
 (DESCRIPTION =
   (ADDRESS_LIST =
     (ADDRESS = (PROTOCOL = TCP)(HOST = 192.168.56.106)(PORT = 1525))
   )
   (CONNECT_DATA =
```

```
    (SID = orcl_std)
   )
 )
```

--The "$ORACLE_HOME/network/admin/listener.ora" file on the server contains the following configuration.

```
LISTENER_STD =
 (DESCRIPTION_LIST =
  (DESCRIPTION =
   (ADDRESS = (PROTOCOL = TCP)(HOST = production.localdomain)(PORT = 1525))
   (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC1521))
  )
 )
SID_LIST_LISTENER =
(SID_LIST =
(SID_DESC =
(GLOBAL_DBNAME = ORCL_STD)
(ORACLE_HOME = /u01/app/oracle/product/19.0.0/db_1 )
(SID_NAME = ORCL_STD)
)
)
```

Once the listener.ora changes are in place, restart the listener

```
lsnrctl stop
lsnrctl start
```

Standby Server Setup

Prepare for Duplicate

--Create a parameter file for the standby database called "/tmp/initorclstd.ora" with the following contents.

*.db_name='cdb1'

--Create a password file, with the SYS password matching that of the primary database.

$ orapwd file=/u01/app/oracle/product/19.0.0/db_1/dbs/orapworclstd password=oracle entries=10

Create Standby Using DUPLICATE

Start the auxiliary instance on the standby server by starting it using the temporary "init.ora" file.

$ export ORACLE_SID=cdb1

$ sqlplus / as sysdba

SQL> STARTUP NOMOUNT PFILE='/tmp/initcdb1_stby.ora';

Connect to RMAN, specifying a full connect string for both the TARGET and AUXILIARY instances. Do not attempt to use OS authentication.

$ rman TARGET sys/oacle@orcl AUXILIARY sys/oracle@orclstd

Now issue the following DUPLICATE command.

duplicate target database

for standby

from active database

dorecover

```
  spfile
    set db_unique_name='orclstd' COMMENT 'Is standby'
  nofilenamecheck;
```

If you need to convert file locations, or alter any initialisation parameters, you can do this during the DUPLICATE using the SET command.

```
duplicate target database
  for standby
  from active database
  dorecover
  spfile
    set db_unique_name='orclstd' COMMENT 'Is standby'
    set db_file_name_convert='+DATA','+DATA_STD'
    set log_file_name_convert='BACKUP','+BACKUP_STD'
    set job_queue_processes='0'
  nofilenamecheck;
```

Enable Broker

At this point we have a primary database and a standby database, so now we need to start using the Data Guard Broker to manage them. Connect to both databases (primary and standby) and issue the following command.

```
alter system set dg_broker_start=true;
```

On the primary server, issue the following command to register the primary server with the broker.

$ dgmgrl sys/oracle@orcl

DGMGRL for Linux: Release 19.0.0.0.0 - Production on Tue Feb 26 22:39:33 2018

Version 19.2.0.0.0

Copyright (c) 1982, 2019, Oracle and/or its affiliates. All rights reserved.

Welcome to DGMGRL, type "help" for information.

Connected as SYSDBA.

DGMGRL> create configuration my_dg_config as primary database is orcl connect identifier is orcl;

Configuration "my_dg_config" created with primary database "orcl"

DGMGRL>

Now add the standby database.

DGMGRL> add database orclstd as connect identifier is orclstd;

Database "orclstd" added

DGMGRL>

Now we enable the new configuration.

DGMGRL> enable configuration;

Enabled.

DGMGRL>

The following commands show how to check the configuration and status of the databases from the broker.

DGMGRL> show configuration;

Configuration - my_dg_config

Protection Mode: MaxPerformance

Members:

orcl - Primary database

orclstd - Physical standby database

Fast-Start Failover: DISABLED

Configuration Status:

SUCCESS (status updated 26 seconds ago)

DGMGRL> show database orcl;

Database - orcl

Role: PRIMARY

Intended State: TRANSPORT-ON

Instance(s):

orcl

Database Status:

SUCCESS

DGMGRL> show database orclstd;

Database - orclstd

Role: PHYSICAL STANDBY

Intended State: APPLY-ON

Transport Lag: 0 seconds (computed 1 second ago)

Apply Lag: 0 seconds (computed 1 second ago)

Average Apply Rate: 5.00 KByte/s

Real Time Query: OFF

Instance(s):

orclstd

Database Status:

SUCCESS

DGMGRL>

Stop/Start Managed Recovery

Managed recovery can be stopped and started on the standby database using the following commands from SQL*Plus.

-- Stop managed recovery.

alter database recover managed standby database cancel;

-- Start managed recovery.

alter database recover managed standby database disconnect;

--Database Template

- The Template is used to generate the same database with same parameters
- There was a bug in generating template using dbca so we used an alternative method

dbca -silent -createTemplateFromDB -sourceDB orcl -templateName template1 -sysDBAUserName sys -sysDBAPassword oracle

Reports

1.User loan from group audit at the end of each day will generate report to retrieve all active loan applications

```
Select loan_date, deadline_agreed_date from loandba.loan where repayment_date is null
and sysdate < deadline_agreed_date ;
```

2.User loan from group audit when needed will retrieve loans details for a specific borrower

```
Select l.loan_date, lr.loan_request_amount,lr.description,
lr.payday,l.deadline_agreed_date
From loandba.loan_request lr ,loandba.loan l
Where lr.loan_request_id=l.loan_request_id
And lr.borrower_id  =5;
```

3.User loan from group audit every month will show loans that have passed their deadline without repaying

```
Select lr.borrower_id ,lr.loan_request_amount ,l.loan_date, l.deadline_agreed_date
from loandba.loan l , loandba.loan_request lr
where l.loan_request_id=lr.loan_request_id
and l.repayment_date is null
and sysdate > l.deadline_agreed_date ;
```

4.User Loan from audit group and manger from manager group every month will list loans that were disbursed

```
Select l.loan_date, lr.loan_request_amount
from loandba.loan  l,loandba.loan_request lr
where l.loan_request_id=lr.loan_request_id
and lr.payday between sysdate - 30 and sysdate;
```

5.User manager from manager group and loan from audit group will generate report to find total amount of cash flow in the system from all loans Repayment every year

```
select sum(repayment.repayment_amount) -
sum (loan_request.loan_request_amount) as "Cash Flow"
from loandba.loan_request, loandba.repayment
where to_char(loan_request_date, 'YYYY') = (select to_char(SYSDATE, 'YYYY') from dual) and
to_char(repayment_date, 'YYYY') = (select to_char(SYSDATE, 'YYYY') from dual) ;
```

6.User loan from audit group and manger from manager group will generate report yearly to shows percentage of loans with amount > 200000

```
select(
(select count(*) from loandba.loan_request where loan_request_amount > 200000 and
to_char(loan_request_date, 'YYYY')
  = (select  to_char(SYSDATE, 'YYYY') from dual) ) /
(select count(*) from loandba.loan_request where  to_char(loan_request_date, 'YYYY')  =
(select  to_char(SYSDATE, 'YYYY') from dual) )*100)
as percent_loan_requests from dual ;
```

7.User intermediary from audit group will generate report monthly to retrieve the intermediary name and the loan date he participated in and the borrower name

```
select distinct lb.loan_date, ad_i.name as "Intermediary Name",ad_b.name as "Borrower Name"

from loandba.intermediary i , loandba.addressee ad_i ,loandba.addressee ad_b ,loandba.lender_borrower lb , loandba.borrower b

where ad_i.addressee_id=i.addressee_id

and ad_b.addressee_id=b.addressee_id

and i.loan_date =lb.loan_date

and b.borrower_id=lb.borrower_id;
```

8.User loan share from audit group will generate report monthly to get number of lenders shared in same loan and the amount of that loan order by maximum amount

```
Select l.loan_date,lr.loan_request_amount ,count(distinct lrl.lender_id)as num_lenders

from loandba.loan l ,loandba.loan_request lr ,loandba.loan_request_lender lrl

where lr.loan_request_id =l.loan_request_id

and lrl.loan_request_id= lr.loan_request_id

group by  l.loan_date,lr.loan_request_amount

order by num_lenders desc;
```

9.User lender share from audit group and manager from manager group will retrieve lenders details and loans they are involved in within 3 months each quarter of year

```
Select lb.lender_id,ad.name,lb.loan_date,lrl.lender_amount
```

From loandba.lender_borrower lb,loandba.loan_request_lender lrl ,loandba.lender l, loandba.addressee ad

Where lrl.lender_id =lb.lender_id

And lb.lender_id =l.lender_id

And l.addressee_id=ad.addressee_id

And lb.loan_date between sysdate-120 and sysdate ;

10.User borrower request from audit group will retrieve number of borrowers from specific location when needed

```
SELECT
 COUNT(b.Borrower_ID) AS Borrower_Count,
 COUNT(l.Lender_ID) AS Lender_Count
FROM loandba.Addressee ad
LEFT JOIN loandba.Borrower b ON ad.Addressee_ID = b.Addressee_ID
LEFT JOIN loandba.Lender l ON ad.Addressee_ID = l.Addressee_ID
WHERE ad.Address LIKE '%River%';
```

References

- https://www.support.dbagenesis.com/post/install-oracle-19c-with-asm
- Oracle 12c Installation on Linux with ASM (dbagenesis.com)
- https://oracle-base.com/articles/misc/tablespace-thresholds-and-alerts
- https://oracle-base.com/articles/19c/data-guard-setup-using-broker-19c#google_vignette