FACULTY OF TECHNOLOGY AND ENGINEERING,
THE MAHARAJA SAYAJIRAO UNIVERSITY OF BARODA

Music Library Management System

DBMS PROJECT

BE-II

(2022-23)

Submitted by	Seat No	PRN No
Priyansh Chaudhari	453006	8021052688
Aaditya Patel	453041	8021075016
Aarsh Patel	453042	8021075066
Devarshi Patel	453043	8021075029
Yatharth Patel	453054	8021055374

INDEX

Project Description

The Music Library Management System is a database management system designed to manage music collections of users. It is designed to allow users to create playlists, add songs, and track their listening history.

The system provides various features like adding, deleting, and modifying songs, albums, and playlists. It also provides the facility to search for songs, albums, and artists. The user can also view their listening history and create personalized playlists.

The system is designed to ensure data security and integrity by implementing proper user authentication and access control mechanisms. It also includes backup and recovery mechanisms to ensure that data is not lost in case of system failures.

Overall, the Music Library Management System provides an efficient way for music enthusiasts to organize and manage their music collections. It simplifies the process of managing music and provides a great user experience.

Cardinalities

User - UserHistory	1:1
User - Playlist	1:M
Playlist - Song	M:N
Song - Artist	M:1
Song - Album	M:1
Artist - Album	1:M

Tables

User1(UserName, Email, Password, DateJoined, LastLogin)

Artist(ArtistName, Bio)

Album(AlbumName, ArtistName, ReleaseYear, Genre)

Song(SongName, ArtistName, AlbumName, durationInSeconds, Genre)

Playlist(PlaylistName, UserName, PlaylistDate)

PlaylistSong(PlaylistName, SongName, PlaylistDate)

UserHistory(UserName, SongName, DateTimePlayed)

user_login(srno, username, password)

trash_bin(PlaylistName, SongName, trash_date)

button_click(button_id)

Note:

Primary Key

Foreign Key

Restrictions & Assumptions

- Here, the table `trash_bin` is created without mentioning it in the ER diagram because it is only used to store the data of deleted playlist. And the data will get automatically deleted from `trash_bin` after 30 days.
- Tables `user_login` and `button_click` are not mentioned in ER diagram as both the tables do not have any relation with other entities. Table `user_login` store details when any user tries to login. And table `button_click` stores button id.
- An assumption is made that no two songs have same name as table song has song name as primary key.
- An assumption is made that an album can be released by only one artist. In album or song details only one artist's name can be shown.
- Here, we are taking song duration in seconds only. Later it can be converted into minutes. Reason behind taking song duration in number format is that if date datatype is chosen then it will always show date along with minute and seconds. i.e.
- if date datatype is chosen then it will show like: '`10-04-2023 03:20`'
  but if number datatype is chosen then it will show like: '`200`'.

Functions

TO FIND ARTIST OF GIVEN SONG

```
create or replace function find_artist (sname in
song.songname%type) return varchar2
is
  artname1 artist.artistname%type;
begin
 select artistname into artname1 from song where
songname=sname;
  return artname1;
exception
when no_data_found then return 'null';
end find_artist;
```

TO FIND ALBUM OF GIVEN SONG

```
create or replace function find_album (sname in
song.songname%type) return varchar2
is
  albname1 album.albumname%type;
begin
 select albumname into albname1 from song where songname=sname;
  return albname1;
exception
when no_data_found then return 'null';
end find_album;

declare
  sname song.songname%type;
  artname artist.artistname%type;
  albname album.albumname%type;
  r1 artist%rowtype;
  r2 album%rowtype;
begin
    sname:='Shape of You';
    artname:=find_artist(sname);
    albname:=find_album(sname);
```

```
dbms_output.put_line('The song '|| sname || ' is sang by the
artist '|| artname);
dbms_output.put_line('The song '|| sname || ' is from the album
name '|| albname || ' of the artist '|| artname);
  select * into r1 from artist where artistname=artname;
  select * into r2 from album where albumname=albname;
dbms_output.put_line('The details of artist: '|| r1.bio);
dbms_output.put_line('The song was release in year: '||
r2.releaseyear);
dbms_output.put_line('The genre of these album is: '|| r2.genre
);
end;
```

TO FIND PLAYLISTS OF A USER

```
create or replace function find_playlist (uname
user1.username%type) return sys_refcursor
as
  all_playlistname sys_refcursor;
begin
 open all_playlistname for
    select Playlistname,playlistdate
    from Playlist where username=uname
    order by playlistdate;
  return all_playlistname;
end find_playlist;

declare
   uname user1.username%type;
   pname playlist.playlistname%type;
   pdate playlist.playlistdate%type;
  all_playlistname sys_refcursor;
begin
     uname:='JohnDoe';
     all_playlistname := find_playlist(uname);
   loop
     fetch all_playlistname into pname,pdate;
     exit when all_playlistname%notfound;
     dbms_output.put_line('Username '|| uname || ' has Playlist
' || pname || ' created on date '|| pdate) ;
   end loop;
   close all_playlistname;
```

```
end;
```

FUNCTION TO FETCH TOP 10 SONGS

```
create or replace function top_10_songs
return sys_refcursor
as
  top_songs sys_refcursor;
begin
  open top_songs for
    select s.SongName, s.ArtistName, s.AlbumName,
s.durationInSeconds, s.Genre, count(*) as views_count
    from UserHistory uh
    inner join Song s on uh.SongName = s.SongName
    group by s.SongName, s.ArtistName, s.AlbumName,
s.durationInSeconds, s.Genre
    order by views_count desc
    fetch first 10 rows only;
  return top_songs;
end;
```

FUNCTION TO SHOW RECOMMENDED SONGS

```
create or replace function recommend_songs(user_name
user1.username%type)
  return sys_refcursor
as
  recommended_songs sys_refcursor;
begin
  open recommended_songs for
    select distinct s.SongName, s.ArtistName, s.AlbumName,
s.durationInSeconds, s.Genre
    from UserHistory u1
    inner join UserHistory u2 on u1.SongName = u2.SongName
    inner join Song s on u2.SongName = s.SongName
    where u1.UserName = user_name and u2.UserName <> user_name
    and s.SongName not in (select SongName from UserHistory where
UserName = user_name)
    group by s.SongName, s.ArtistName, s.AlbumName,
s.durationInSeconds, s.Genre
    order by count(*) desc
```

```
    fetch first 10 rows only;
  return recommended_songs;
end;
```

Procedures

TO INSERT SONG

```
create or replace procedure insert_song(is_SongName
song.SongName%type,is_ArtistName
song.ArtistName%type,is_AlbumName
song.AlbumName%type,is_durationInSeconds
song.durationInSeconds%type,is_Genre song.genre%type)
    as
    cursor c1 is select ArtistName from Artist ;
    cursor c2 is select AlbumName  from album  ;

art_exist boolean :=false;
alb_exist boolean :=false;
is_bio varchar2(255);
is_releaseYear number(4);
begin
for r1 in c1
    loop
     if is_ArtistName=r1.ArtistName then
          art_exist:=true;
          end if;
    end loop;
for r2 in c2
    loop
     if is_AlbumName=r2.AlbumName then
          alb_exist:=true;
          end if;
    end loop;
if art_exist= false then
dbms_output.put_line('Enter the detail about Artist: ');
is_bio := 'abc';
insert into artist values(is_ArtistName, is_bio);
end if ;
if alb_exist= false then
is_releaseYear := 2023 ;
insert into Album values(is_AlbumName, is_ArtistName,
is_releaseYear , is_Genre);
end if;
```

```
 insert into song values(is_SongName, is_ArtistName,
is_AlbumName, is_durationInSeconds, is_Genre);
exception
when no_data_found then dbms_output.put_line('error');
end insert_song;
```

TO INSERT SONG IN A PLAYLIST

```
create or replace procedure insert_song_into_playlist(ip_songname
in PlaylistSong.SongName%type , ip_PlaylistName in
PlaylistSong.PlaylistName%type )
    As
    cursor c1 is select * from Song ;
    cursor c2 is select * from playlist ;
song_exist boolean := false ;
playlist_exist boolean := false ;
    Begin
for r1 in c1
        loop
            if ip_songname = r1.songname then
        song_exist := true ;
  end if ;
end loop ;

for r2 in c2
    loop
       if ip_PlaylistName = r2.playlistName then
       playlist_exist := true ;
  end if ;
end loop ;
if (song_exist = false ) then dbms_output.put_line(' ERROR ! Song
does not exist  ') ;
end if ;
if (playlist_exist = false ) then dbms_output.put_line(' ERROR !
playlist does not exist  ') ;
end if;
if (song_exist = true and playlist_exist = true ) then
  insert into PlaylistSong values(ip_PlaylistName , ip_songname
,sysdate );
end if;
end insert_song_into_playlist;
```

PROCEDURE FOR SIGN UP

```
create procedure sign_up(  su_UserName in varchar2 ,
  su_Email in varchar2 ,
  su_Password in varchar2  ,
  su_DateJoined in date,
  su_LastLogin in date   )
    AS
    uname_exist boolean := false ;
    email_exist boolean := false ;
    cursor c1 is select * from User1 ;
    begin
        for r1 in c1
        loop
        if su_UserName  = r1.username then
         uname_exist := true ;

        end if ;
        end loop ;

    for r1 in c1
        loop
        if su_Email = r1.Email then
        email_exist :=true ;


        end if ;
        end loop ;
if   uname_exist = true  then
        dbms_output.put_line('Unsename already exist! please try
Again ');
end if ;
 if  email_exist =true then
      dbms_output.put_line('Email is not valid ! please try Again
');
end if ;
   if( uname_exist = false and email_exist = false )
  then   insert into User1 values(su_UserName , su_Email ,
Su_Password , su_DateJoined , su_LastLogin);
end if;
end sign_up;
```

TO CREATE A PLAYLIST

```
create or replace procedure create_playlist (
    cp_PlaylistName Playlist.PlaylistName%type ,
  cp_UserName Playlist.UserName%type
)
      As
    --cursor c1 is select p.playlistname , u.username from
playlist p , User1 u where p.username = u.usernam  ;
    cursor c1 is select * from playlist ;
    cursor c2 is select * from User1 ;
      playlist_exist boolean := false ;
      username_exist boolean := false ;
      Begin
          for r1 in c1
          loop
          if  cp_PlaylistName = r1.playlistName then

           playlist_exist := true ;
          end if;
end loop ;

for r2 in c2
    loop
          if  cp_UserName = r2.UserName then
           username_exist := true ;
          end if;
        end loop ;
if  playlist_exist = true then
dbms_output.put_line('Playlistname already exist! please try
Again ');
    end if;
if  username_exist = false then  dbms_output.put_line('Unsename
not exist! please try Again ');
     end if ;
         if ( playlist_exist = false and  username_exist = true )
then
         insert into Playlist values (cp_PlaylistName ,
cp_UserName , sysdate ) ;
end if;
end create_playlist ;
```

TO DELETE A SONG

```
create or replace procedure delete_song (ds_SongName in
song.songname%type)
  AS Begin
    Delete from Song where SongName =ds_SongName;
exception
    when no_data_found then
    dbms_output.put_line('song name does not exist please enter
correct Song Name');
end delete_song;
```

Triggers

TRIGGER TO SEND THE DELETED PLAYLIST TO THE TRASH BIN

```
create or replace trigger move_to_trash
before delete on playlistsong
for each row
begin
 insert into trash_bin values
(:old.playlistname,:old.songname,sysdate);
end move_to_trash;
```

TRIGGER TO REMOVE FROM THE TRASH AFTER 30 DAYS

```
create or replace trigger remove_from_trash
after delete on playlistsong
for each row
declare
tdate number(3);
tdays trash_bin.trash_date%type;
begin
 select trash_date into tdays from trash_bin where
playlist_name = :old.playlistname and
songname=:old.songname;
 tdate := sysdate - tdays;
 if (tdate > 30) then
 delete from trash_bin where
playlist_name = :old.playlistname and
songname=:old.songname;
 end if;
end remove_from_trash;
```

TRIGGER TO VERIFY USER LOGIN

```
create or replace trigger verify_user_login
before insert on user_login
for each row
declare
     a user1.username%type;
    b user1.password%type;
begin
    select username, password
    into a, b
    from user1
    where username = :new.username;

     if a = :new.username and b = :new.password then
        dbms_output.put_line('Logged in.');
    end if;
    if b <> :new.password or a<> :new.username then
        dbms_output.put_line('Invalid username or password.');
     end if;
     delete from user_login where username = a;
end verify_user_login;
```

TRIGGER TO RESPONSE BUTTON CLICKS

```
create or replace trigger button_response_trigger
after insert on button_click
for each row
begin
     if (:new.button_id = 1) then
    --procedure insert song
    pck1.insert_song('Song name', 'Artist name', 'Album name',
170, 'Genre');
     elsif (:new.button_id = 2) then
     --procedure insert song into playlist
     pck1.insert_song_into_playlist('song Name', 'Playlist
Name');
     elsif (:new.button_id = 3) then
     --procedure create playlist
     pck1.create_playlist('Playlist Name', 'User name');
     elsif (:new.button_id = 4) then
```

```
     --procedure delete from song
     pck1.delete_val('song name');
    elsif (:new.button_id = 5) then
    --procedure sign up
    pck1.sign_up('username', 'email', 'password',
to_date('2022-01-01', 'YYYY-MM-DD'),to_date('2022-01-01',
'YYYY-MM-DD'));
     end if;
end button_response_trigger;
```