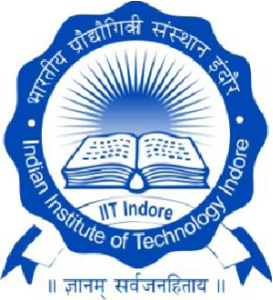
****

**Indian Institute of Technology, Indore**

**Computer Science & Engineering**

**CS 257: Lab Assignment on Case Study**

**Music Player & Music Management System**

Submitted By:

Kunal Gupta 150001015

Punit Lakshwani 150001025

Shivam Tayal 150001034

**STEP-WISE ANALYSIS OF CASE STUDY**

# 1.Identification of Entity Sets:

* Track – Songs that will played in the Music Player.
* Artist – Artists of the songs in the track entity.
* Album – Album of the songs in track entity.
* Playlist – Playlists created by user.
* Directories – Location of folder where tracks are located.

2.Modeling Attributes and Keys:

1. **Track:**

* TrackId
* Trackname
* Location
* Genre
* DateAdded
* Favourite

**Primary key:** (TrackId)

Tracks are the songs and music files in the system. Each track will be given unique TrackID.

* TrackId : Track ID of the track.
* Trackname : Track name of the track.
* Location : Location of track in the system.
* Genre : Genre of the track (like pop, dance, jazz, etc.).
* DateAdded : The date on which the track was added to the database.
* Favourite : Boolean flag which is true if song is added to favourites else false.

1. **Artist:**

* ArtistId
* Artistname

**Primary key:** (ArtistId)

Artist entity contains the different artist ids and artist names.

* ArtistId : Unique Id given to the artist.
* Artistname : Name of the Artist.

1. **Album:**

* AlbumId
* Albumname

**Primary key:** (AlbumId)

Album entity contains the different album names and their unique ids.

* AlbumId : Unique id given to a album.
* Albumname : Name of the Album.

1. **Playlist:**

* PlaylistId
* Playlistname

**Primary key:** (PlaylistId)

Playlist entity contains the all the playlists created by users.

* PlaylistId : Unique id given to each playlist.
* Playlistname : Name of the playlist given by user.

1. **Directories:**

* FolderLocation

**Primary key:**  (FolderLocation)

Directories contains the address of the the select directories.

* FolderLocation : Location of the directory.

3. Identification of Relationships:

Track, Artist and Album are related to each other. **TrackInfo** contains the artist and album of each track.

PlayList and Track are related to each other. **PlayListInfo** contains the track and playlist ids along with the order of track in playlist.

Directories contains tracks. **TrackLocation** relates track to its directory.

|  |  |  |
| --- | --- | --- |
| S.No. | Relationship | Entity Sets |
| 1 | TrackInfo | Track, Artist, Album |
| 4 | PlaylistInfo | Playlist, Track |
| 5 | TrackLocation | Directories,Track |

##### ***5. Description Of Relations:***

1. **TrackInfo** : (Track, Artist, Album)

**Type** : Ternary Relationship

Mapping cardinality

Track 🡨🡪 Artist : (m : 1)

Track 🡨🡪 Album : (m : 1)

Artist 🡨🡪 Album : (1 : m)

Track : Total participation

Artist : Total participation

Album : Total participation

1. **PlaylistInfo** : (PlayList, Track)

**Type** : Binary Relationship

Mapping cardinality : (m : n)

Playlist : Total participation

Track : Partial participation

1. **TrackLocation** : (Directories, Track)

**Type** : Binary Relationship

Mapping cardinality : (1 : m)

Directories : Partial participation

Track : Total participation

##### ***6.Transforming Entities and Relations into Tables:***

* **Entity sets into tables :**

1. Track :

* TrackId int PRIMARY KEY NOT NULL,
* Trackname varchar(50) NOT NULL,
* Location varchar(500) NOT NULL UNIQUE,
* DateAdded timestamp CURRENTTIMESTAMP,
* Genre varchar(30) NOT NULL,
* Favourite boolean DEFAULT (0)

1. Artist :

* ArtistId int PRIMARY KEY NOT NULL,
* Artistname varchar(50) NOT NULL UNIQUE

1. Album :

* AlbumId int PRIMARY KEY NOT NULL,
* Albumname varchar(50) NOT NULL UNIQUE

1. Playlist :

* PlaylistId int PRIMARY KEY NOT NULL,
* Playlistname varchar(100) NOT NULL UNIQUE

1. Directories :

* FolderLocation varchar(500) NOT NULL UNIQUE
* **Relationships into tables :**

1. TrackInfo :

* Primary Key (TrackId)
* Foriegn Key (ArtistId, AlbumId)

1. PlayListInfo:

* Primary Key (PlayListId, TrackId)
* Trackorder int NOT NULL

1. TrackLocation:

* Primary Key (Track.Location)
* Foriegn Key (FolderLocation)

##### ***5. Functions:***

1) Search for a artist

CREATE FUNCTION searchartist (name varchar(50)) RETURNS int(11)

BEGIN

declare id int;

set id = (select artistid from artist where artistname = name);

if (id > 0) then

return id;

end if;

RETURN -1;

END

2) Search for a album

CREATE FUNCTION searchalbum (name varchar(50)) RETURNS int(11)

BEGIN

declare id int;

set id = (select albumid from artist where albumname = name);

if (id > 0) then

return id;

end if;

RETURN -1;

END

##### ***6. Procedures:***

1. Adding a Directory

CREATE PROCEDURE adddirectory (in location varchar(500))

BEGIN

insert into directories values (location);

END

1. Adding a Track

CREATE PROCEDURE addtrack(in trackname varchar(50),in artistname varchar(50),in albumname varchar(100),in location varchar(500),in genre varchar(30))

Begin

declare artistid int;

declare albumid int;

declare trackid int;

set trackid = (select newid from id);

set artistid = searchartist(artistname);

set albumid = searchalbum(albumname);

insert into track(trackid,trackname,location,genre) values(trackid, trackname, location, genre);

if (artistid = -1) then

set artistid = (select newid from id);

insert into artist values (artistid, artistname);

end if;

if (albumid = -1) then

set albumid = (select newid from id);

insert into album values (albumid, albumname);

end if;

insert into trackinfo values (trackid, artistid, albumid);

end

3) Adding a Track to a Playlist

CREATE PROCEDURE addtracktoplaylist (in pid int,in tid int)

BEGIN

declare torder int;

set torder = (select count(\*) from playlistinfo where playlistid=pid);

set torder = torder + 1;

insert into playlistinfo values(pid,tid,torder);

END

4) Creating a PlayList

CREATE PROCEDURE createplaylist (in PlaylistName varchar(100))

BEGIN

declare pid int;

set pid = (select playlistid from id);

insert into playlist values(pid,PlaylistName);

END

5) Deleting a Directory

CREATE PROCEDURE deletedirectory (in location varchar(500))

BEGIN

declare tid int;

declare trackids cursor for select trackid from track where track.location like concat(location, '%'); delete from directories where folderlocation = location;

open trackids;

get\_trackid : LOOP

fetch trackids into tid;

call deletetrack(tid);

end loop get\_trackid;

close trackids;

END

6) Deleting a track from a playlist

CREATE PROCEDURE deletefromplaylist (in pid int,in tid int)

BEGIN

declare torder int;

set torder=(select trackorder from playlistinfo where playlistid=pid and trackid=tid);

delete from playlistinfo where playlistid=pid and trackid=tid;

update playlistinfo set trackorder=trackorder-1 where playlistid=pid and trackorder>torder;

END

7) Deleting a playlist

CREATE PROCEDURE deleteplaylist (in pid int)

BEGIN

delete from playlist where playlistid=pid;

delete from playlistinfo where playlistid=pid;

END

8) Deleting a track

CREATE PROCEDURE deletetrack (in TrackId int)

BEGIN

delete from track where track.trackid=TrackId;

delete from trackinfo where trackinfo.trackid=TrackId;

delete from playlistinfo where playlistinfo.trackid=TrackId;

END

9) Adding a track to Favourites

CREATE PROCEDURE favourite (in TrackId int)

BEGIN

update track set favourite=True where track.trackid=TrackId;

END

10) Moving a track in playlist (up or down)

CREATE PROCEDURE movetrack (in pid int, in tid int, in direction int)

BEGIN

declare swaptid int;

declare torder int;

set torder = (select trackorder from playlistinfo where playlistid = pid and trackid = tid);

if(direction = 0) then

set swaptid = (select trackid from playlistinfo where playlistid = pid and trackorder = torder - 1);

update playlistinfo set trackorder = torder where playlistid = pid and trackid = swaptid;

update playlistinfo set trackorder = torder - 1 where playlistid = pid and trackid = tid;

else

set swaptid = (select trackid from playlistinfo where playlistid = pid and trackorder = torder + 1);

update playlistinfo set trackorder = torder where playlistid = pid and trackid = swaptid;

update playlistinfo set trackorder = torder + 1 where playlistid = pid and trackid = tid;

end if;

END

11) Removing a track from Favourites

CREATE PROCEDURE unfavourite (in TrackId int)

BEGIN

update track set favourite=false where track.trackid=TrackId;

END

12) Update a playlist name

CREATE PROCEDURE updateplaylist (in pid int, in newname varchar(50))

BEGIN

update playlist set playlistname = newname where playlistid = pid;

END

13) Updating a track information

CREATE PROCEDURE updatetrack (in TrackId int,in newtrack varchar(50),in newartist varchar(50),in newalbum varchar(100),in newgenre varchar(30))

BEGIN

declare ArtistId int;

declare AlbumId int;

update track set trackname = newtrack, genre = newgenre where track.trackid = TrackId;

set ArtistId=searchartist(newartist);

set AlbumId=searchalbum(newalbum);

if (ArtistId = -1) then

set ArtistId=(select newid from id);

insert into artist values (ArtistId,newartist);

end if;

if (AlbumId = -1) then

set AlbumId = (select newid from id);

insert into album values (AlbumId,newalbum);

end if;

update trackinfo set artistid=ArtistId, albumid=AlbumId where trackinfo.trackid = TrackId;

END

##### ***7. Triggers:***

1) Refresh artist and album after delete in trackinfo

CREATE TRIGGER deleterefresh AFTER DELETE ON trackinfo

FOR EACH ROW

BEGIN

DELETE FROM album WHERE albumid NOT IN (SELECT DISTINCT albumid FROM trackinfo);

DELETE FROM artist WHERE artistid NOT IN (SELECT DISTINCT artistid FROM trackinfo);

END

2) Refresh artist and album after update in trackinfo

CREATE TRIGGER deleterefresh AFTER UPDATE ON trackinfo

FOR EACH ROW

BEGIN

DELETE FROM album WHERE albumid NOT IN (SELECT DISTINCT albumid FROM trackinfo);

DELETE FROM artist WHERE artistid NOT IN (SELECT DISTINCT artistid FROM trackinfo);

END

##### ***8. ER-DIAGRAM OF MUSIC MANAGEMENT SYSTEM***

