UCS310: DATABASE MANAGEMENT SYSTEM

SUBMITTED TO:

Mr. Sanjeev Rao

SUBMITTED BY:

Riddhi Garg (102103282)

GROUP-CO10

THE MUSIC LIBRARY

INDEX

ACKNOWLEDGEMENT

We have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and teachers. We would like to extend our sincere thanks to all of them.

We are highly indebted to **Dr. Sanjeev Rao** for his guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

We would like to express our gratitude towards our parents for their kind co-operation and encouragement which helped us in completion of this project.

Our thanks and appreciations also go to our colleague in developing the project and people who have willingly helped us out with their abilities.

INTRODUCTION:

The music database is designed to store details of a music collection, including the **albums** in the collection, the **artists** who made them, the **tracks** on the albums.

The Music Database

The music database stores details of a personal music library, and could be used to manage your MP3, CD, or vinyl collection. Because this database is for a personal collection, it's relatively simple and stores only the relationships between artists, albums, and tracks. It ignores the requirements of many music genres, making it most useful for storing popular music and less useful for storing jazz or classical music.

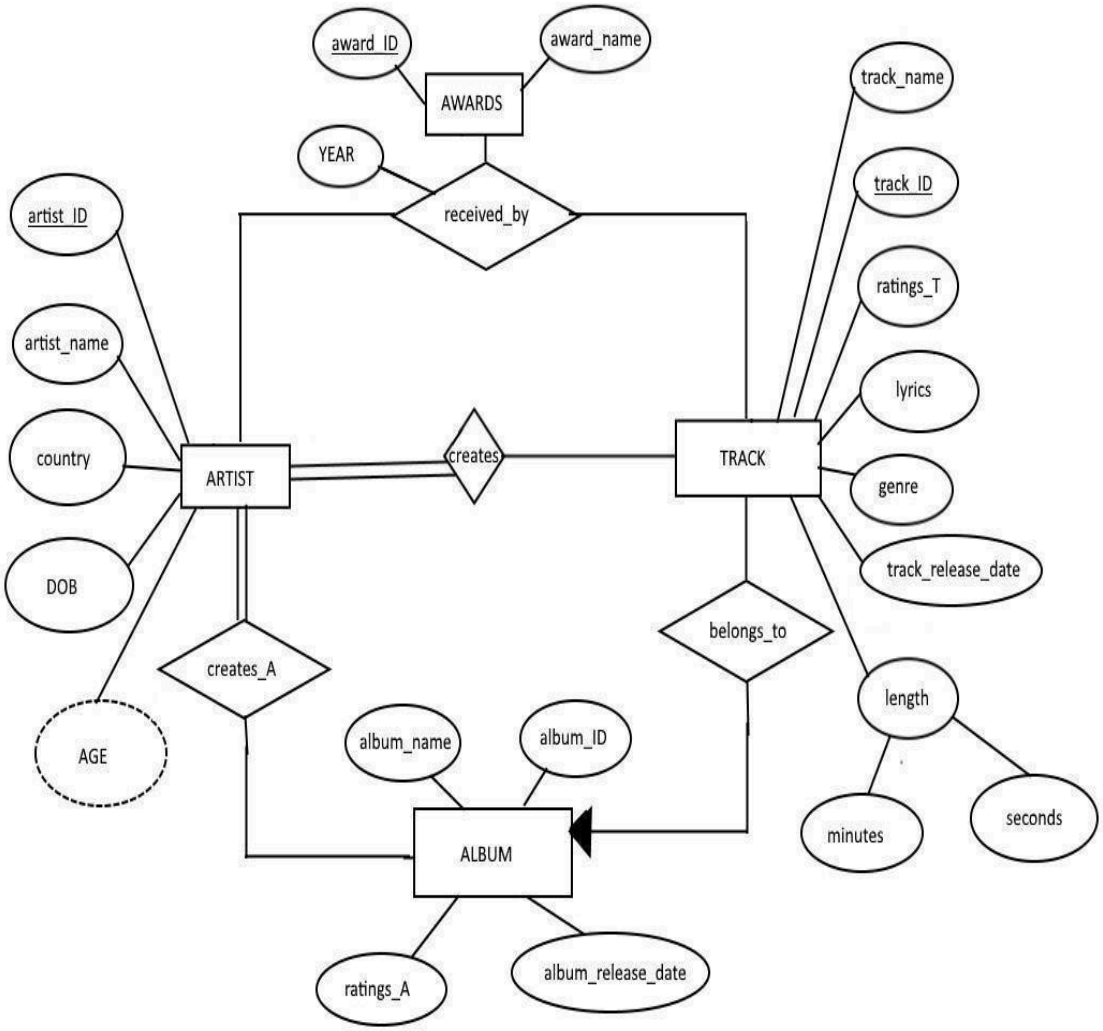
We first draw up a clear list of requirements for our database:

- The collection consists of albums.
- An album is made by exactly one artist.
- An artist makes one or more albums. An album contains one or more tracks
- Artists, albums, and tracks each have a name.
- Each track is on exactly one album.
- Each track has a time length, measured in minutes and seconds.

There's no requirement to capture composers, group members or sidemen, recording date or location, the source media, or any other details of artists, albums, or tracks.

The ER diagram derived from our requirements is shown in below figure. The attributes are straightforward: artists, albums, and tracks have names, as well as identifiers to uniquely identify each entity.

ER-Diagram


award_ID
award_name
AWARDS
YEAR
received_by
artist_ID
artist_name
country
DOB
AGE
ARTIST
creates
creates_A
TRACK
track_name
track_ID
ratings_T
lyrics
genre
track_release_date
belongs_to
length
minutes
seconds
album_name
album_ID
ALBUM
ratings_A
album_release_date

What it doesn’t do….?

We’ve kept the music database simple because adding extra features doesn’t help you learn anything new, it just makes the explanations longer. If you wanted to use the music database in practice, then you might consider adding the following features:

1. Support for compilations or various-artists albums, where each track may be by a different artist and may then have its own associated album-like details such as a recording date and time. Under this model, the album would be a strong entity, with many-to-many relationships between artists and albums.
2. Playlists, a user-controlled collection of tracks. For example, you might create a playlist of your favorite tracks from an artist.
3. Source details, such as when you bought an album, what media it came on, how much you paid, and so on.
4. Album details, such as when and where it was recorded, the producer and label, the band members or sidemen who played on the album, and even its artwork.
5. Smarter track management, such as modeling that allows the same track to appear on many albums.

TABLES USED:

ARTIST:

COLUMN_NAME	DATA_TYPE	CONSTRAINTS
ARTIST_ID	NUMBER	PRIMARY KEY
ARTIST_NAME	VARCHAR2(23)	
COUNTRY	VARCHAR2(23)	
DOB	DATE	

TRACK:

COLUMN_NAME	DATA_TYPE	CONSTRAINTS
TRACK_NAME	VARCHAR2(10)	
TRACK_ID	NUMBER(3)	PRIMARY KEY
RATINGS_T	NUMBER(1)	CHECK
LYRICS	CLOB	
GENRE	VARCHAR(30)	
TRACK_RELEASE_DATE	DATE	
ALBUM_ID	NUMBER(3)	FOREIGN KEY
MINUTES	NUMBER(2)	CHECK
SECONDS	NUMBER(2)	CHECK

ALBUM:

COLUMN_NAME	DATA_TYPE	CONSTRAINTS
ALBUM_NAME	VARCHAR2(30)	
ALBUM_ID	NUMBER	PRIMARY KEY
RATINGS_A	FLOAT	CHECK
ALBUM_RELEASE_DATE	DATE	

AWARD:

COLUMN_NAME	DATA_TYPE	CONSTRAINTS
AWARD_ID	NUMBER(3)	PRIMARY KEY
AWARD_NAME	VARCHAR2(30)	

RECEIVED_BY:

COLUMN_NAME	DATA_TYPE	CONSTRAINTS
ARTIST_ID	NUMBER(3)	FOREIGN KEY
AWARD_ID	NUMBER(3)	FOREIGN KEY
TRACK_ID	NUMBER(3)	FOREIGN KEY
YEAR	NUMBER(4)	

CREATES_A:

COLUMN_NAME	DATA_TYPE	CONSTRAINTS
ARTIST_ID	NUMBER(3)	FOREIGN KEY
ALBUM_ID	NUMBER(3)	FOREIGN KEY

CREATES_T:

COLUMN_NAME	DATA_TYPE	CONSTRAINTS
ARTIST_ID	NUMBER(3)	FOREIGN KEY
TRACK_ID	NUMBER(3)	FOREIGN KEY

Functional dependency:

A functional dependency is a constraint that specifies the relationship between two sets of attributes where one set can accurately determine the value of other sets. It is denoted as $X \rightarrow Y$, where X is a set of attributes that is capable of determining the value of Y. The attribute set on the left side of the arrow, X is called **Determinant**, while on the right side, Y is called the **Dependent.**

NORMALIZATION:

Normalization is a process of organizing the data in database to avoid data redundancy, insertion anomaly, update anomaly & deletion anomaly.

First normal form (1NF)

As per the rule of first normal form, an attribute (column) of a table cannot hold multiple values. It should hold only atomic values.

Second normal form (2NF)

A table is said to be in 2NF if both the following conditions hold:

- Table is in 1NF (First normal form)
- No non-prime attribute is dependent on the proper subset of any candidate key of table.

An attribute that is not part of any candidate key is known as non-prime attribute.

Third Normal form (3NF)

A table design is said to be in 3NF if both the following conditions hold:

- Table must be in 2NF
- Transitive functional dependency of non-prime attribute on any super key should be removed.

An attribute that is not part of any candidate key is known as non-prime attribute.

In other words, 3NF can be explained like this: A table is in 3NF if it is in 2NF and for each functional dependency X-> Y at least one of the following conditions hold:

- X is a super key of table
- Y is a prime attribute of table

An attribute that is a part of one of the candidate keys is known as prime attribute.

ARTIST:

ARTIST_ID	ARTIST_NAME	COUNTRY	DOB
1	eminem	us	17-OCT-72
2	taylor swift	us	13-DEC-89
3	rihana	barbados	20-DEC-88
4	bruno mars	us	08-OCT-85
5	michael jackson	us	29-AUG-58

Functional Dependency:

ARTIST_ID ->ARTIST_NAME, COUNTRY, DOB

Candidate Keys: ARTIST_ID
Non-prime attributes: ARTIST_NAME, COUNTRY, DOB

1NF: This table is in first normal form because all the attributes in the relation have atomic domains.

2NF: This table is in second normal form because there is no partial dependency present.

3NF: This table is in third normal form because there is no transitive dependency.

TRACK:

TRACK_NAME	TRACK_ID	RATINGS_T	LYRICS	GENRE	TRACK_RELEASE_DATE	ALBUM_ID	MINUTES	SECONDS
walk of water	1	5	I walk on waterBut I aint no JesusI walk on water But only when it freezes	hip hop	15-DEC-17	1	4	12
not afraid	2	4	I amm not afraid (I am not afraid) Yeah To take a stand (to take a stand) It has been a ride Everybody (everybody) I guess I had to go to that place	hip hop	14-APR-11	2	5	16
so bad	3	4	Yeah, haha You feel that, baby? Yeah, I feel it too Damn (I am so bad, I am so good that I am so bad)	hip hop	14-APR-11	2	3	45
mean	15	4	You With your words like knives and swords and weapons That you use againt me You	country music	25-OCT-10	9	6	30
beautiful	4	5	Lately I have been hard to reach I have been too long on my own Everybody has a private world where they can be alone Are you calling me?	hip hop	05-MAR-09	3	4	20
crack a bottle	5	4	Oh, ladies and gentlemen The moment you have all been waiting for In this corner, weighing 175 pounds	hip hop	05-MAR-09	3	4	29
beat it	6	4	They told him, "Don t you ever come around here" "Don t wanna see your face, you better disappear" The fire is in their eyes and their words are really clear	pop	25-MAR-83	4	4	49
the girl is mine	7	4	Every night she walks right in my dreams Since I met her from the start I m so proud I am the only one Who is special in her heart	pop	25-MAR-83	4	4	5
heal the world	8	5	There is a place in your heart And I know that it is love And this place it was brighter than tomorrow And if you really try	pop	26-JUN-93	5	4	15
full attack	9	3	ecstasy... in the air I dont care cannot tell me nothing i am impaired the	contemporary	19-AUG-12	6	3	10
shredders	10	4	work, work, work, work, work, work be said me have to work, work, work, work, work	contemporary	19-AUG-12	6	3	50
the lazy song	11	5	Today I don t feel like doing anything I just wanna lay in my bed Don t feel like picking up my phone So leave a message at the tone	funk	15-JAN-11	7	4	32
it will rain	12	4	f you ever leave me, baby Leave some morphine at my door Cause it would take a whole lot of medication	funk	15-JAN-11	7	4	10
treasure	13	4	Give me your, give me your, give me your attention, baby I got to tell you a little something about yourself	funk	22-MAR-12	8	4	16
mine	14	4	You were in college, working part-time, waiting tables Left a small town, never looked back	country music	25-OCT-10	9	4	30
fifteen	16	4	You take a deep breath And you walk through the doors	country music	11-NOV-08	10	5	20

Functional Dependency:

TRACK_ID ->TRACK_NAME, RATING_T, TRACK_RELEASE_DATE, ALBUM_ID,
LYRICS, GENRE, MINUTES, SECONDS

Candidate Keys: TRACK_ID
Non-prime attributes: TRACK_NAME, RATING_T, TRACK_RELEASE_DATE, ALBUM_ID, LYRICS, GENRE, MINUTES, SECONDS

1NF: This table is in first normal form because all the attributes in the relation have atomic domains.

2NF: This table is in second normal form because there is no partial dependency present.

3NF: This table is in third normal form because there is no transitive dependency.

ALBUM:

ALBUM_ID	ALBUM_NAME	RATINGS_A	ALBUM_RELEASE_DATE
1	revival	4.6	15-DEC-17
2	recovery	4.8	14-APR-11
3	relapse	4.3	05-MAR-09
4	thriller	4.5	25-MAR-83
5	dangerous	4.1	27-JUN-92
6	battleship	3.9	19-AUG-12
7	doo-wops	2.8	15-JAN-11
8	unorthodox	3.4	22-MAR-12
9	speak now	4.2	25-OCT-10
10	fearless	4.4	11-NOV-08

Functional Dependency:

ALBUM_ID ->ALBUM_NAME, RATING_A, ALBUM_RELEASE_DATE

Candidate Keys: ALBUM_ID
Non-prime attributes: ALBUM_NAME, RATING_A, ALBUM_RELEASE_DATE

1NF: This table is in first normal form because all the attributes in the relation have atomic domains.

2NF: This table is in second normal form because there is no partial dependency present.

3NF: This table is in third normal form because there is no transitive dependency.

AWARD:

AWARD_ID	AWARD_NAME
1	mtv
2	grammy
3	billboard music
4	american music
5	favourite pop/rock
6	artist of the year
7	best art direction

Functional Dependency:

AWARD_ID ->AWARD_NAME

Candidate Keys: AWARD_ID

Non-prime attributes: AWARD_NAME

1NF: This table is in first normal form because all the attributes in the relation have atomic domains.

2NF: This table is in second normal form because there is no partial dependency present.

3NF: This table is in third normal form because there is no transitive dependency.

RECEIVED BY:

ARTIST_ID	AWARD_ID	TRACK_ID	YEAR
1	1	2	2001
1	2	1	2008
1	3	4	2004
2	1	15	2015
2	2	14	2009
2	3	16	2017
3	1	9	2005
3	2	10	2010
3	7	10	2005
4	5	11	2011
4	6	13	2006
5	2	7	2013
5	5	6	2015
5	6	7	2009

Functional Dependency:

AWARD_ID, ARTIST_ID, TRACK_ID ->YEAR

Candidate Keys: AWARD_ID, ARTIST_ID, TRACK_ID

Non-prime attributes: YEAR

1NF: This table is in first normal form because all the attributes in the relation have atomic domains.

2NF: This table is in second normal form because there is no partial dependency present.

3NF: This table is in third normal form because there is no transitive dependency.

CREATES_A:

ARTIST_ID	ALBUM_ID
1	1
1	2
1	3
2	9
2	10
3	6
4	7
4	8
5	4
5	5

Functional Dependency:

ARTIST_ID, ALBUM_ID ->NONE

Candidate Keys: ARTIST_ID, ALBUM_ID

Non-prime attributes: NONE

1NF: This table is in first normal form because all the attributes in the relation have atomic domains.

2NF: This table is in second normal form because there is no partial dependency present.

3NF: This table is in third normal form because there is no transitive dependency.

CREATES_T:

ARTIST_ID	TRACK_ID
1	1
1	2
1	3
1	4
1	5
2	14
2	15
2	16
3	9
3	10
4	11
4	12
4	13
5	6
5	7
5	8

Functional Dependency:

ARTIST_ID, TRACK_ID ->NONE

Candidate Keys: ARTIST_ID, TRACK_ID

Non-prime attributes: NONE

1NF: This table is in first normal form because all the attributes in the relation have atomic domains.

2NF: This table is in second normal form because there is no partial dependency present.

3NF: This table is in third normal form because there is no transitive dependency.

INSERTION:

```
CREATE TABLE ARTIST(ARTIST_ID NUMBER PRIMARY KEY,
ARTIST_NAME VARCHAR2(23), COUNTRY VARCHAR2(23), DOB DATE);

DECLARE

PROCEDURE INSERT_ARTIST (A_ID NUMBER, A_NAME VARCHAR2 ,
A_COUNTRY VARCHAR2 , A_DOB DATE) AS

BEGIN

INSERT INTO ARTIST VALUES(A_ID, A_NAME,A_COUNTRY,A_DOB);

EXCEPTION

WHEN DUP_VAL_ON_INDEX THEN

   DBMS_OUTPUT.PUT_LINE ('YOU ARE INSERTING DUPLICATE DETAILS
FOR ARTIST');

WHEN OTHERS THEN

   DBMS_OUTPUT.PUT_LINE('SQLERRM');

END;

BEGIN

INSERT_ARTIST (1,'eminem','us',to_date('17-10-1972','dd-mm-yyyy'));

INSERT_ARTIST (2,'taylor swift','us',to_date('13-12-1989','dd-mm-yyyy'));

INSERT_ARTIST (3,'rihana','barbados',to_date('20-12-1988','dd-mm-yyyy'));

INSERT_ARTIST (4,'bruno mars','us',to_date('8-10-1985','dd-mm-yyyy'));

INSERT_ARTIST (5,'michael jackson','us',to_date('29-8-1958','dd-mm-yyyy'));

END;

SELECT * FROM ARTIST;
```

```
 CREATE TABLE ALBUM(ALBUM_ID NUMBER PRIMARY KEY,
ALBUM_NAME VARCHAR2(23), RATINGS_A FLOAT
CHECK(RATINGS_A<=5), ALBUM_RELEASE_DATE DATE);

DECLARE
```

```
PROCEDURE INSERT_ALBUM (A_ID NUMBER, A_NAME VARCHAR2 ,
A_RATING NUMBER , A_RELEASE DATE) AS

BEGIN

INSERT INTO ALBUM VALUES(A_ID, A_NAME,A_RATING,A_RELEASE);

EXCEPTION

WHEN DUP_VAL_ON_INDEX THEN

   DBMS_OUTPUT.PUT_LINE ('YOU ARE INSERTING DUPLICATE DETAILS
FOR ALBUM');

WHEN OTHERS THEN

   DBMS_OUTPUT.PUT_LINE ('AN ERROR HAS OCCURRED WHILE
INSERTING!!');

END;

BEGIN

INSERT_ALBUM (1,'revival',4.6,to_date('15-12-2017','dd-mm-yyyy'));

INSERT_ALBUM (2,'recovery',4.8,to_date('14-04-2011','dd-mm-yyyy'));

INSERT_ALBUM (3,'relapse',4.3,to_date('05-03-2009','dd-mm-yyyy'));

INSERT_ALBUM (4,'thriller',4.5,to_date('25-03-1983','dd-mm-yyyy'));

INSERT_ALBUM (5,'dangerous',4.1,to_date('27-06-1992','dd-mm-yyyy'));

INSERT_ALBUM (6,'battleship',3.9,to_date('19-08-2012','dd-mm-yyyy'));

INSERT_ALBUM (7,'doo-wops',2.8,to_date('15-01-2011','dd-mm-yyyy'));

INSERT_ALBUM (8,'unorthodox',3.4,to_date('22-03-2012','dd-mm-yyyy'));

INSERT_ALBUM (9,'speak now',4.2,to_date('25-10-2010','dd-mm-yyyy'));

INSERT_ALBUM (10,'fearless',4.4,to_date('11-11-2008','dd-mm-yyyy'));

END;

SELECT * FROM ALBUM;
```

```
CREATE TABLE TRACK(TRACK_NAME VARCHAR2(30), TRACK_ID
NUMBER(3) PRIMARY KEY, RATINGS_T NUMBER(1)
CHECK(RATINGS_T<=5),

   LYRICS CLOB, GENRE VARCHAR(30),TRACK_RELEASE_DATE
DATE,ALBUM_ID NUMBER(3), MINUTES NUMBER(2) CHECK(MINUTES<60),

   SECONDS NUMBER(2) CHECK(SECONDS<60),FOREIGN KEY(ALBUM_ID)
REFERENCES ALBUM ON DELETE CASCADE);

DECLARE

PROCEDURE INSERT_TRACK (T_NAME VARCHAR2 , T_ID NUMBER ,
T_RATING NUMBER , T_LYRICS CLOB

, T_GENRE VARCHAR2 , T_RELEASE DATE , A_ID NUMBER , T_MIN
NUMBER , T_SEC NUMBER) AS

BEGIN

INSERT INTO TRACK
VALUES(T_NAME,T_ID,T_RATING,T_LYRICS,T_GENRE,T_RELEASE,A_ID,T_
MIN,T_SEC);

EXCEPTION

WHEN DUP_VAL_ON_INDEX THEN

   DBMS_OUTPUT.PUT_LINE('YOU ARE INSERTING DUPLICATE DETAILS
FOR TRACK');

WHEN OTHERS THEN

   DBMS_OUTPUT.PUT_LINE(SQLERRM);

END;

BEGIN

INSERT_TRACK('walk of water', 1, 5,'I walk on waterBut I aint no JesusI walk on
water But only when it freezes',

   'hip hop',to_date('15-12-2017','dd-mm-yyyy'),1,4,12);

INSERT_TRACK('not afraid',2 , 4,'I amm not afraid (I am not afraid) Yeah To take a
stand (to take a stand) It has been a ride Everybody (everybody) I guess I had to go to
that place',

   'hip hop',to_date('14-04-2011','dd-mm-yyyy'),2,5,16);

INSERT_TRACK('so bad', 3, 4,'Yeah, haha You feel that, baby? Yeah, I feel it too
Damn (I am so bad, I am so good that I am so bad)',
```

'hip hop',to_date('14-04-2011','dd-mm-yyyy'),2,3,45);

INSERT_TRACK('mean', 15, 4,'You With your words like knives and swords and weapons That you use againt me You',

'country music',to_date('25-10-2010','dd-mm-yyyy'),9,6,30);

INSERT_TRACK('beautiful', 4, 5,'Lately I have been hard to reach I have been too long on my own Everybody has a private world where they can be alone Are you calling me?',

'hip hop',to_date('5-3-2009','dd-mm-yyyy'),3,4,20);

INSERT_TRACK('crack a bottle', 5, 4,'Oh, ladies and gentlemen

The moment you have all been waiting for

In this corner, weighing 175 pounds',

'hip hop',to_date('5-3-2009','dd-mm-yyyy'),3,4,29);

INSERT_TRACK('beat it', 6, 4,'They told him, "Don t you ever come around here"

"Don t wanna see your face, you better disappear"

The fire is in their eyes and their words are really clear',

'pop',to_date('25-3-1983','dd-mm-yyyy'),4,4,49);

INSERT_TRACK('the girl is mine', 7, 4,'Every night she walks right in my dreams

Since I met her from the start I m so proud I am the only one

Who is special in her heart',

'pop',to_date('25-3-1983','dd-mm-yyyy'),4,4,5);

INSERT_TRACK('heal the world', 8, 5,'There is a place in your heart

And I know that it is love

And this place it was brighter than tomorrow

And if you really try',

'pop',to_date('26-6-1993','dd-mm-yyyy'),5,4,15);

INSERT_TRACK('full attack', 9, 3,'ecstasy... in the air I dont care cannot tell me nothing i am impaired the',

'contemporary',to_date('19-8-2012','dd-mm-yyyy'),6,3,10);

INSERT_TRACK('shredders', 10, 4,'work, work, work, work, work, work be said me have to work, work, work, work, work',

'contemporary',to_date('19-8-2012','dd-mm-yyyy'),6,3,50);

```
INSERT_TRACK('the lazy song', 11, 5,'Today I don t feel like doing anything I just
wanna lay in my bed

Don t feel like picking up my phone

So leave a message at the tone',

   'funk',to_date('15-1-2011','dd-mm-yyyy'),7,4,32);

INSERT_TRACK('it will rain', 12, 4,'f you ever leave me, baby

Leave some morphine at my door

Cause it would take a whole lot of medication',

   'funk',to_date('15-1-2011','dd-mm-yyyy'),7,4,10);

INSERT_TRACK('treasure',13, 4,'Give me your, give me your, give me your attention,
baby

I got to tell you a little something about yourself',

   'funk',to_date('22-3-2012','dd-mm-yyyy'),8,4,16);

INSERT_TRACK('mine', 14, 4,'You were in college, working part-time, waiting tables

Left a small town, never looked back',

   'country music',to_date('25-10-2010','dd-mm-yyyy'),9,4,30);

INSERT_TRACK('fifteen', 16, 4,'You take a deep breath

And you walk through the doors',

   'country music',to_date('11-11-2008','dd-mm-yyyy'),10,5,20);

END;

   select * from track;



 CREATE TABLE AWARD(AWARD_ID NUMBER(3) PRIMARY KEY,
AWARD_NAME VARCHAR2(30));

DECLARE

PROCEDURE INSERT_AWARD ( A_ID NUMBER , A_NAME VARCHAR2 ) AS

BEGIN

INSERT INTO AWARD VALUES (A_ID, A_NAME);

EXCEPTION
```

```
WHEN DUP_VAL_ON_INDEX THEN
   DBMS_OUTPUT.PUT_LINE ('YOU ARE INSERTING DUPLICATE DETAILS FOR AWARD');
WHEN OTHERS THEN
   DBMS_OUTPUT.PUT_LINE ('AN ERROR HAS OCCURED WHILE INSERTING!!');
END;
BEGIN
INSERT_AWARD (1,'mtv');
INSERT_AWARD (2,'grammy');
INSERT_AWARD (3,'billboard music');
INSERT_AWARD (4,'american music');
INSERT_AWARD (5,'favourite pop/rock');
INSERT_AWARD (6,'artist of the year');
INSERT_AWARD (7,'best art direction');
END;
SELECT * FROM AWARD;

CREATE TABLE RECEIVED_BY(ARTIST_ID NUMBER(3), AWARD_ID NUMBER(3), TRACK_ID NUMBER(3), YEAR NUMBER(4),
   FOREIGN KEY(ARTIST_ID) REFERENCES ARTIST ON DELETE CASCADE, FOREIGN KEY(AWARD_ID) REFERENCES AWARD
   ON DELETE CASCADE,
   FOREIGN KEY(TRACK_ID) REFERENCES TRACK ON DELETE CASCADE);
DECLARE
PROCEDURE INSERT_RECEIVED_BY ( A_ID NUMBER , AW_ID NUMBER , T_ID NUMBER , Y NUMBER) AS
BEGIN
INSERT INTO RECEIVED_BY VALUES (A_ID, AW_ID,T_ID,Y);
EXCEPTION
WHEN DUP_VAL_ON_INDEX THEN
```

```
   DBMS_OUTPUT.PUT_LINE ('YOU ARE INSERTING DUPLICATE DETAILS.');
WHEN OTHERS THEN
   DBMS_OUTPUT.PUT_LINE('SQLERRM');
END;
BEGIN
INSERT_RECEIVED_BY(1,1,2,2001);
INSERT_RECEIVED_BY(1,2,1,2008);
INSERT_RECEIVED_BY(1,3,4,2004);
INSERT_RECEIVED_BY(2,1,15,2015);
INSERT_RECEIVED_BY(2,2,14,2009);
INSERT_RECEIVED_BY(2,3,16,2017);
INSERT_RECEIVED_BY(3,1,9,2005);
INSERT_RECEIVED_BY(3,2,10,2010);
INSERT_RECEIVED_BY(3,7,10,2005);
INSERT_RECEIVED_BY(4,5,11,2011);
INSERT_RECEIVED_BY(4,6,13,2006);
INSERT_RECEIVED_BY(5,2,7,2013);
INSERT_RECEIVED_BY(5,5,6,2015);
INSERT_RECEIVED_BY(5,6,7,2009);
END;
select * from received_by;



CREATE TABLE CREATES_A(ARTIST_ID NUMBER(3), ALBUM_ID
NUMBER(3),
   FOREIGN KEY(ARTIST_ID) REFERENCES ARTIST ON DELETE CASCADE,
   FOREIGN KEY(ALBUM_ID) REFERENCES ALBUM ON DELETE CASCADE);
DECLARE
```

```
PROCEDURE INSERT_CREATES_A (A_ID NUMBER , AL_ID NUMBER) AS
BEGIN
INSERT INTO CREATES_A VALUES (A_ID, AL_ID);
EXCEPTION
WHEN DUP_VAL_ON_INDEX THEN
   DBMS_OUTPUT.PUT_LINE ('YOU ARE INSERTING DUPLICATE DETAILS.');
WHEN OTHERS THEN
   DBMS_OUTPUT.PUT_LINE('SQLERRM');
END;
BEGIN
INSERT_CREATES_A (1,1);
INSERT_CREATES_A (1,2);
INSERT_CREATES_A (1,3);
INSERT_CREATES_A (2,9);
INSERT_CREATES_A (2,10);
INSERT_CREATES_A (3,6);
INSERT_CREATES_A (4,7);
INSERT_CREATES_A (4,8);
INSERT_CREATES_A (5,4);
INSERT_CREATES_A (5,5);
END;
SELECT * FROM CREATES_A;


 CREATE TABLE CREATES_T(ARTIST_ID NUMBER(3), TRACK_ID
NUMBER(3), FOREIGN KEY(ARTIST_ID) REFERENCES ARTIST ON DELETE
CASCADE,
   FOREIGN KEY(TRACK_ID) REFERENCES TRACK ON DELETE CASCADE);
DECLARE
PROCEDURE INSERT_CREATES_T (A_ID NUMBER , T_ID NUMBER ) AS
BEGIN
```

```
INSERT INTO CREATES_T VALUES(A_ID,T_ID);
EXCEPTION
WHEN DUP_VAL_ON_INDEX THEN
   DBMS_OUTPUT.PUT_LINE ('YOU ARE INSERTING DUPLICATE DETAILS.');
WHEN OTHERS THEN
   DBMS_OUTPUT.PUT_LINE(SQLERRM);
END;
BEGIN
INSERT_CREATES_T(1,1);
INSERT_CREATES_T(1,2);
INSERT_CREATES_T(1,3);
INSERT_CREATES_T(1,4);
INSERT_CREATES_T(1,5);
INSERT_CREATES_T(2,14);
INSERT_CREATES_T(2,15);
INSERT_CREATES_T(2,16);
INSERT_CREATES_T(3,9);
INSERT_CREATES_T(3,10);
INSERT_CREATES_T(4,11);
INSERT_CREATES_T(4,12);
INSERT_CREATES_T(4,13);
INSERT_CREATES_T(5,6);
INSERT_CREATES_T(5,7);
INSERT_CREATES_T(5,8);
END;
SELECT * FROM CREATES_T;
```

DELETION:

```
DECLARE

ERROR_ON_DELETE EXCEPTION;

   PROCEDURE DELETE_DATA (
   A VARCHAR2
   )
   AS
   BEGIN
   DELETE FROM ARTIST WHERE ARTIST_NAME=A;
   DBMS_OUTPUT.PUT_LINE('NO OF ENTRIES DELETED:
'||SQL%ROWCOUNT);
IF
SQL%ROWCOUNT=0 THEN
RAISE ERROR_ON_DELETE;
END IF;
EXCEPTION
WHEN ERROR_ON_DELETE THEN
   DBMS_OUTPUT.PUT_LINE('THIS RECORD IS NOT AVAILABLE IN
DATABASE');
WHEN OTHERS THEN
   DBMS_OUTPUT.PUT_LINE('THATS AN ERROR');
   DBMS_OUTPUT.PUT_LINE(SQLERRM);
END;
BEGIN
   DELETE_DATA('rihana');
END;

DECLARE
```

```
ERROR_ON_DELETE EXCEPTION;


   PROCEDURE DELETE_DATA (

   A NUMBER

   )

   AS

   BEGIN
   DELETE FROM ALBUM WHERE ALBUM_ID=A;
   DBMS_OUTPUT.PUT_LINE('NO OF ENTRIES DELETED:
'||SQL%ROWCOUNT);

IF

SQL%ROWCOUNT=0 THEN

RAISE ERROR_ON_DELETE;

END IF;

EXCEPTION

WHEN ERROR_ON_DELETE THEN

   DBMS_OUTPUT.PUT_LINE('THIS RECORD IS NOT AVAILABLE IN
DATABASE');

WHEN OTHERS THEN

   DBMS_OUTPUT.PUT_LINE('THATS AN ERROR');

   DBMS_OUTPUT.PUT_LINE(SQLERRM);

END;

BEGIN

   DELETE_DATA(3);

END;


DECLARE

ERROR_ON_DELETE EXCEPTION;

   PROCEDURE DELETE_DATA(A NUMBER, B NUMBER, C NUMBER) AS
```

```
   BEGIN
   DELETE FROM RECEIVED_BY WHERE ARTIST_ID=A AND AWARD_ID=B
AND TRACK_ID=C;
   DBMS_OUTPUT.PUT_LINE('NO OF ENTRIES DELETED:
'||SQL%ROWCOUNT);
IF
SQL%ROWCOUNT=0 THEN
RAISE ERROR_ON_DELETE;
END IF;
EXCEPTION
WHEN ERROR_ON_DELETE THEN
   DBMS_OUTPUT.PUT_LINE('THIS RECORD IS NOT AVAILABLE IN
DATABASE');
WHEN OTHERS THEN
   DBMS_OUTPUT.PUT_LINE('THATS AN ERROR');
   DBMS_OUTPUT.PUT_LINE(SQLERRM);
   END;
BEGIN
   DELETE_DATA(2,1,15);
END;



DECLARE
ERROR_ON_DELETE EXCEPTION;

   PROCEDURE DELETE_DATA(
     A NUMBER
   , B NUMBER
)
   AS
```

```
   BEGIN
   DELETE FROM CREATES_A WHERE ARTIST_ID=A AND ALBUM_ID=B;
   DBMS_OUTPUT.PUT_LINE('NO OF ENTRIES
DELETED:'||SQL%ROWCOUNT);
IF
SQL%ROWCOUNT=0 THEN
RAISE ERROR_ON_DELETE;
END IF;
EXCEPTION
WHEN ERROR_ON_DELETE THEN
   DBMS_OUTPUT.PUT_LINE('THIS RECORD IS NOT AVAILABLE IN
DATABASE');
WHEN OTHERS THEN
   DBMS_OUTPUT.PUT_LINE('THATS AN ERROR');
   DBMS_OUTPUT.PUT_LINE(SQLERRM);

   END;
BEGIN
   DELETE_DATA(4,7);
END;



DECLARE
ERROR_ON_DELETE EXCEPTION;

   PROCEDURE DELETE_DATA(
     A NUMBER
   , B NUMBER
   )
```

```
AS
   BEGIN
   DELETE FROM CREATES_T WHERE ARTIST_ID=A AND TRACK_ID=B;
   DBMS_OUTPUT.PUT_LINE('NO OF ENTRIES DELETED:
'||SQL%ROWCOUNT);
IF
SQL%ROWCOUNT=0 THEN
RAISE ERROR_ON_DELETE;
END IF;
EXCEPTION
WHEN ERROR_ON_DELETE THEN
   DBMS_OUTPUT.PUT_LINE('THIS RECORD IS NOT AVAILABLE IN
DATABASE');
WHEN OTHERS THEN
   DBMS_OUTPUT.PUT_LINE('THATS AN ERROR');
   DBMS_OUTPUT.PUT_LINE(SQLERRM);

   END;
BEGIN
   DELETE_DATA(5,8);
END;
```

CURSOR

```
DECLARE
NOT_FOUND EXCEPTION;
FOUND NUMBER:=0;
PROCEDURE retrieve(id number) as
cursor C1 is select * from TRACK where TRACK_ID = id;
REC C1%rowtype;
begin
open C1;
loop fetch C1 into REC;
exit when
C1%notfound;
FOUND:=1;
dbms_output.put_line('TRACK NAME: ' || REC.TRACK_NAME ||' RATING:
'||REC.RATINGS_T ||' DURATION: '||REC.MINUTES||':'||REC.SECONDS);
dbms_output.put_line('RELEASE DATE: ' || REC.TRACK_RELEASE_DATE||'
GENRE: '||REC.GENRE); dbms_output.put_line('LYRICS: ');
dbms_output.put_line(REC.LYRICS);
end loop;
close C1;
IF FOUND <>1 THEN
RAISE NOT_FOUND;
END IF;
EXCEPTION
WHEN NOT_FOUND THEN dbms_output.put_line('NO RECORDS FOUND FOR
THIS TRACK ');
WHEN OTHERS THEN
dbms_output.put_line(SQLERRM);
end;
```

```
begin
retrieve(14);
end;

DECLARE
NOT_FOUND EXCEPTION;
FOUND NUMBER:=0;
PROCEDURE retrieve(id number) as
cursor C1 is select * from CREATES_A where ARTIST_ID = id;
REC C1%rowtype;
ALB ALBUM%ROWTYPE;
A_ID ARTIST.ARTIST_ID%TYPE;
begin
SELECT DISTINCT ARTIST_ID INTO A_ID FROM RECEIVED_BY WHERE
ARTIST_ID=id;
dbms_output.put_line('ARTIST WHOSE ID IS '|| A_ID ||' HAS CREATED
FOLLOWING ALBUMS' );
open C1;
loop
fetch C1 into REC;
exit when
C1%notfound;
FOUND:=1;
SELECT * INTO ALB FROM ALBUM WHERE ALBUM_ID=REC.ALBUM_ID;
dbms_output.put_line('NAME: '||ALB.ALBUM_NAME||' RATING:
'||ALB.RATINGS_A||' RELEASE DATE: '||ALB.ALBUM_RELEASE_DATE);
end loop;
close C1;
IF    FOUND    <>    1
THEN            RAISE
```

```
NOT_FOUND;     END
IF;
```

```
EXCEPTION
WHEN NOT_FOUND THEN dbms_output.put_line('NO RECORDS FOUND FOR
THIS ARTIST ');
WHEN OTHERS THEN
dbms_output.put_line(SQLERRM);
end;
begin
   retrieve(5);
END;


DECLARE
NOT_FOUND EXCEPTION;
FOUND NUMBER:=0;
PROCEDURE retrieve(id number) as
cursor C1 is select * from CREATES_T where ARTIST_ID = id;
REC C1%rowtype;
TRK TRACK%ROWTYPE;
A_ID ARTIST.ARTIST_ID%TYPE;
begin
SELECT DISTINCT ARTIST_ID INTO A_ID FROM RECEIVED_BY WHERE
ARTIST_ID=id;
dbms_output.put_line('ARTIST WHOSE ID IS '|| A_ID ||' HAS CREATED
FOLLOWING TRACKS' );
open C1;
loop
fetch C1 into REC;
exit when C1%notfound;
FOUND:=1;
SELECT * INTO TRK FROM TRACK WHERE TRACK_ID=REC.TRACK_ID;
dbms_output.put_line ('NAME: '||TRK.TRACK_NAME||' RATING:
'||TRK.RATINGS_T||' TRACK DATE: '||TRK.TRACK_ID);
end loop;     close C1;
IF FOUND <> 1 THEN
RAISE NOT_FOUND;
END IF;
EXCEPTION
WHEN NOT_FOUND THEN dbms_output.put_line('NO RECORDS FOUND FOR
THIS ARTIST ');
```

```
WHEN OTHERS THEN
dbms_output.put_line(SQLERRM);
end;
begin
   retrieve(5);
END;

DECLARE
   NOT_FOUND EXCEPTION;
   FOUND NUMBER:=0;
   PROCEDURE retrieve(id number) as
   cursor C1 is select * from RECEIVED_BY where ARTIST_ID = id;
   REC C1%rowtype;
   AWD AWARD%ROWTYPE;
   A_ID ARTIST.ARTIST_ID%TYPE; TRK TRACK.TRACK_NAME%TYPE;
   begin
   SELECT DISTINCT ARTIST_ID INTO A_ID FROM RECEIVED_BY WHERE
ARTIST_ID=id;
   dbms_output.put_line('ARTIST WHOSE ID IS '|| A_ID ||' HAS WON
FOLLOWING
AWARDS' );
open C1;
loop
fetch C1 into REC;
exit when C1%notfound;
FOUND :=1;
SELECT * INTO AWD FROM AWARD
WHERE AWARD_ID=REC.AWARD_ID;

SELECT TRACK_NAME INTO TRK FROM TRACK WHERE
TRACK_ID=REC.TRACK_ID;
dbms_output.put_line('NAME: '||AWD.AWARD_NAME||' FOR TRACK:
'||TRK||' YEAR: '||REC.YEAR);
end loop;
close C1;
IF FOUND <> 1 THEN
RAISE  NOT_FOUND;
END IF;
EXCEPTION
WHEN NOT_FOUND THEN
dbms_output.put_line('NO RECORDS FOUND FOR THIS ARTIST');
WHEN OTHERS THEN
dbms_output.put_line(SQLERRM);
```

```
end;
begin
   retrieve(2);
END;
```

AGE CALCULATION

```
DECLARE
   AGE
   NUMBER;
   DOB DATE;
   FUNCTION AGE_CALC(DOB IN DATE) RETURN NUMBER IS
   begin
   AGE:=(SYSDATE-DOB)/365;
   RETURN(AGE);
   end;
begin
AGE:=AGE_CALC(to_date('17-10-1972','dd-mm-yyyy'));
DBMS_OUTPUT.PUT_LINE('AGE IS ' || ROUND(AGE,0)||' YEARS
'||ABS(ROUND(((AGE-ROUND(AGE,0))*365),0))||' DAYS');
END;
```

TRIGGERS

```
CREATE OR REPLACE TRIGGER AGE
AFTER INSERT OR DELETE OR UPDATE ON ARTIST
FOR EACH ROW
BEGIN
   IF INSERTING THEN
   DBMS_OUTPUT.PUT_LINE('INSERTING');
   ELSIF DELETING THEN
   DBMS_OUTPUT.PUT_LINE('DELETING');
   ELSIF UPDATING THEN
     DBMS_OUTPUT.PUT_LINE('UPDATING');
   END IF;
END;
BEGIN
   DELETE FROM ARTIST WHERE ARTIST_ID=1;
END;

CREATE OR REPLACE TRIGGER AWARD
AFTER INSERT OR DELETE OR UPDATE ON AWARD
FOR EACH ROW
BEGIN
   IF INSERTING THEN
   DBMS_OUTPUT.PUT_LINE('INSERTING');
   ELSIF DELETING THEN
   DBMS_OUTPUT.PUT_LINE('DELETING');
   ELSIF UPDATING THEN
     DBMS_OUTPUT.PUT_LINE('UPDATING');
```

```
   END IF;
END;
BEGIN
   DELETE FROM AWARD WHERE AWARD_ID=1;
END;



CREATE OR REPLACE TRIGGER ALBUM
AFTER INSERT OR DELETE OR UPDATE ON ALBUM
FOR EACH ROW
BEGIN
   IF INSERTING THEN
   DBMS_OUTPUT.PUT_LINE('INSERTING');
   ELSIF DELETING THEN
   DBMS_OUTPUT.PUT_LINE('DELETING');
   ELSIF UPDATING THEN
     DBMS_OUTPUT.PUT_LINE('UPDATING');
   END IF;
END;

CREATE OR REPLACE TRIGGER TRACK
AFTER INSERT OR DELETE OR UPDATE ON TRACK
FOR EACH ROW
BEGIN
   IF INSERTING THEN
   DBMS_OUTPUT.PUT_LINE('INSERTING');
   ELSIF DELETING THEN
   DBMS_OUTPUT.PUT_LINE('DELETING');
```

```
   ELSIF UPDATING THEN
     DBMS_OUTPUT.PUT_LINE('UPDATING');
   END IF;
END;
BEGIN
   UPDATE TRACK SET RATINGS_T=3 WHERE ALBUM_ID=10;
END;


CREATE OR REPLACE TRIGGER R_B
AFTER INSERT OR DELETE OR UPDATE ON RECEIVED_BY
FOR EACH ROW
BEGIN
   IF INSERTING THEN
   DBMS_OUTPUT.PUT_LINE('INSERTING');
   ELSIF DELETING THEN
   DBMS_OUTPUT.PUT_LINE('DELETING');
   ELSIF UPDATING THEN
     DBMS_OUTPUT.PUT_LINE('UPDATING');
   END IF;
END;
BEGIN
DELETE FROM RECEIVED_BY WHERE TRACK_ID=16;
END;
```

CONCLUSION

PL/SQL can be used to manage the data of a music library efficiently. This includes tasks such as creating, updating, and deleting records in the database, handling transactions, and managing metadata such as song titles, artist names, album information, and user preferences. PL/SQL's robust data manipulation capabilities can ensure accurate and reliable data management in a music library.

PL/SQL can provide efficient data management, customized music retrieval, advanced music library features, seamless integration with database systems, and enhanced performance and scalability for a music library. By leveraging the capabilities of PL/SQL, a music library can be implemented with robust data management, advanced features, and optimal performance, providing a seamless and efficient experience for users.

THE END