Problem Statement
Project Goals:
1. To build a prototype multimedia DBMS using Oracle that manipulates textual music lyrics (.doc files), music audio files (.m4a files), and music videos (.m4v files).
2. Employ the SQL PL/SQL programming concepts from the INFO 607 course
3. Create a quality report for inclusion into my professional portfolio

Importance:

Given the results and skills-demanding nature of the technology job market, this project is very important as it will give prospective employers a glimpse of my SQL, PL/SQL skill set.

High-Level Functionality:

- Implemented the schema from the approved ERD using SQL DDL.
- Created triggers which automatically increment or decrement derived attributes whenever specified data is inserted or deleted into/from specified tables.
- Created the DML for data insertion, and three special procedures for inserting the MM files into the track, lyric, and video tables.
- Created a procedure which outputs all the videos of an artist, given the artist's name.
- Created a procedure which outputs all the videos of a genre, given the genre.
- Created a procedure which outputs all the information about the album and the artist, given the title of a video.
- Created a query which selects all the songs, albums, and name of the videos, given the genre.
- Created a query which selects the number of songs in the album (derived attribute), track title, the artist's name, the genre of music, and the video name.

In-Scope/Out-of-Scope:

The ERD, schema, DDL, DML, Stored Procedures, triggers, and queries are included in my deliverables. A special front-end interface for this system was excluded (i.e. this DBMS is accessed through Oracle’s default interface) due to time/skill constraints.
CONCEPTUAL ANALYSIS

Computing Environment: Linux Ubuntu 11.04 OS running on a Intel Core i5 655k CPU (Dual core) 64-bit architecture with 16GB RAM using Oracle 10g XE.

Development method: The method employed for analysis, design, and implementation was a custom agile approach. The process began with specifying a few different use cases, drawing an ERD, coding, and then iteratively refining my ERD, use cases, and code.

Requirement analysis: This project requires the MMDBMS to manipulate three different kinds of multimedia files: audio (.m4a, .mp3, etc), text documents (.doc), and video (.m4v, etc.). In addition to this data, the database must store information about the artist [artist ID, artist name, country, number of albums (derived), number of songs(derived)], and information about the album [album ID, album title, artist ID, genre of music, number of songs in the album (derived)].

Functional Requirements:
Data Model:

Discussion on the Model: This model is a clean, logical representation of the main tables, attributes, and relationships. As you can see, no videos may be inserted into the database without an existing related track record in the Track table. Similar logic applies to the Lyric table and Track table, in terms of referential integrity (FK). The content attribute of the Track, Lyrics, and Video tables is of BLOB data type.

Implementation
Overview of implementation: The architecture of the implementation follows the data model precisely, and includes all of the DML, procedures, triggers, and queries necessary to realize the use case model above. The tools used is the Oracle system with SQL and PL/SQL languages.

Structure of implementation: The implementation is composed of SQL DLL statements, PL/SQL trigger code, SQL DML statements, PL/SQL procedures, and SQL queries.

Inputs: the data inserted into the database is either number type, varchar2 type, or BLOB type.

The following data reference tables can be used when running the queries and procedures:

<table>
<thead>
<tr>
<th>Artist_id</th>
<th>Artist_name</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234</td>
<td>2-PAC</td>
<td>US</td>
</tr>
<tr>
<td>1235</td>
<td>Blink-182</td>
<td>US</td>
</tr>
<tr>
<td>1236</td>
<td>James Durbin</td>
<td>US</td>
</tr>
<tr>
<td>1237</td>
<td>Jay-Z</td>
<td>US</td>
</tr>
<tr>
<td>1238</td>
<td>Katy Perry</td>
<td>US</td>
</tr>
<tr>
<td>1239</td>
<td>Machine Head</td>
<td>US</td>
</tr>
<tr>
<td>1240</td>
<td>Megadeth</td>
<td>US</td>
</tr>
<tr>
<td>1241</td>
<td>Nickelback</td>
<td>US</td>
</tr>
<tr>
<td>1242</td>
<td>Nirvana</td>
<td>US</td>
</tr>
<tr>
<td>1243</td>
<td>Outkast</td>
<td>US</td>
</tr>
<tr>
<td>1244</td>
<td>Pearl Jam</td>
<td>US</td>
</tr>
<tr>
<td>1245</td>
<td>Stick to Your Guns</td>
<td>US</td>
</tr>
<tr>
<td>1246</td>
<td>Switchfoot</td>
<td>US</td>
</tr>
<tr>
<td>1247</td>
<td>The Pharcyde</td>
<td>US</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Album_id</th>
<th>album_title</th>
<th>artist_id</th>
<th>genre</th>
</tr>
</thead>
<tbody>
<tr>
<td>4321</td>
<td>Me Against the World</td>
<td>1234</td>
<td>Hip Hop</td>
</tr>
<tr>
<td>4322</td>
<td>Enema of the State</td>
<td>1235</td>
<td>Alternative Rock</td>
</tr>
<tr>
<td>4323</td>
<td>Memories of a Beautiful Disaster</td>
<td>1236</td>
<td>Alternative Rock</td>
</tr>
<tr>
<td>4224</td>
<td>The Blueprint</td>
<td>1237</td>
<td>Hip Hop</td>
</tr>
<tr>
<td>4225</td>
<td>Teenage Dream</td>
<td>1238</td>
<td>Pop</td>
</tr>
<tr>
<td>4226</td>
<td>Unto the Locust</td>
<td>1239</td>
<td>Alternative Rock</td>
</tr>
<tr>
<td>4227</td>
<td>Th1rt3en</td>
<td>1240</td>
<td>Metal</td>
</tr>
<tr>
<td>4228</td>
<td>Here and Now</td>
<td>1241</td>
<td>Pop</td>
</tr>
<tr>
<td>4229</td>
<td>Nevermind</td>
<td>1242</td>
<td>Alternative Rock</td>
</tr>
<tr>
<td>4230</td>
<td>Southernplayalisticadillacmuzik</td>
<td>1243</td>
<td>Hip Hop</td>
</tr>
<tr>
<td>4231</td>
<td>Rearview Mirrors</td>
<td>1244</td>
<td>Alternative Rock</td>
</tr>
<tr>
<td>4232</td>
<td>For What Its Worth</td>
<td>1245</td>
<td>Alternative Rock</td>
</tr>
<tr>
<td>4233</td>
<td>Vice Verses</td>
<td>1246</td>
<td>Rock</td>
</tr>
<tr>
<td>4234</td>
<td>Bizarre Ride 2</td>
<td>1247</td>
<td>Hip Hop</td>
</tr>
</tbody>
</table>

Outputs: the output is a subset of the input data.

Running the Database: To run the database, you must first compile all the code, in the order which it appears below. Before you insert the multimedia files using the insert
procedures, you log in as database administrator and issue the following commands at the sqlplus prompt:

```
create or replace directory temp as '/THEPATHONYOURMACHINETOTHEMMFILES'
grant read on directory temp to USERNAME
```

Oracle will not be able to find the multimedia files on your host OS unless you issue the commands above.

No other special steps are necessary to run the database. To execute the procedures, type `exec procedure_name(parameters,...);` Queries are run as normal on the Oracle system.

**CODE:**

**DDL**

```sql
CREATE TABLE Artist(
    artist_id number(9),
    artist_name varchar(30),
    country char(2),
    num_albums number(3),
    num_songs number(5)
);

ALTER TABLE Artist
    add CONSTRAINT artist_pk PRIMARY KEY (artist_id);

CREATE TABLE Album(
    album_id number(9),
    album_title varchar(45),
    artist_id number(9),
    genre varchar(20),
    num_songs_in_album number(3)
);

ALTER TABLE Album
    add CONSTRAINT album_pk PRIMARY KEY (album_id);

ALTER TABLE Album
    add CONSTRAINT artist_fk FOREIGN KEY (artist_id) REFERENCES Artist(artist_id);

CREATE TABLE Track(
    track_id number(9),
    track_title varchar2(30),
    artist_id number(9),
    album_id number(9),
    content BLOB
);
```
ALTER TABLE Track
add CONSTRAINT track_pk PRIMARY KEY (track_id);

ALTER TABLE Track
add CONSTRAINT album_fk FOREIGN KEY (album_id) REFERENCES Album(album_id);

ALTER TABLE Track
add CONSTRAINT artist2_fk FOREIGN KEY (artist_id) REFERENCES Artist(artist_id);

CREATE TABLE Lyrics (  
lyric_id number(9),
track_id number(9),
version varchar(30),
content BLOB  
);

ALTER TABLE Lyrics
add CONSTRAINT lyrics_pk PRIMARY KEY ( lyric_id);

ALTER TABLE Lyrics
add CONSTRAINT track_fk FOREIGN KEY (track_id) REFERENCES Track (track_id);

CREATE TABLE Video(  
Video_id number(9),
Video_name varchar(30),
track_id number(9),
producer varchar(25),
director varchar(25),
special_guest varchar(25),
content BLOB,
type char(15)  
);

ALTER TABLE Video
add CONSTRAINT video_id PRIMARY KEY (video_id);

ALTER TABLE Video
add CONSTRAINT track_fk_2 FOREIGN KEY (track_id) REFERENCES Track (track_id);

TRIGGERS

CREATE OR REPLACE TRIGGER update_num_albums  
after INSERT OR DELETE ON album  
FOR EACH ROW  
BEGIN
BEGIN
IF inserting THEN
  UPDATE artist
  SET num_albums = num_albums + 1
  WHERE :old.artist_id = artist.artist_id;
END IF;
IF deleting THEN
  UPDATE Artist
  SET num_albums = num_albums - 1
  WHERE :old.artist_id = artist.artist_id;
END IF;
END;

CREATE OR REPLACE TRIGGER update_num_songs
  after INSERT OR DELETE ON Track
  FOR each ROW
BEGIN
  IF inserting THEN
    UPDATE artist
    SET num_songs = num_songs + 1
    WHERE :new.artist_id = artist.artist_id;
  END IF;
  IF deleting THEN
    UPDATE Artist
    SET num_songs = num_songs - 1
    WHERE :old.artist_id = artist.artist_id;
  END IF;
END;

CREATE OR replace TRIGGER update_num_songs_in_album
  after INSERT OR DELETE ON track
  FOR each ROW
BEGIN
  IF inserting THEN
    UPDATE album
    SET num_songs_in_album = num_songs_in_album + 1
    WHERE :new.album_id = album.album_id;
  END IF;
  IF deleting THEN
    UPDATE album
    SET num_songs_in_album = num_songs_in_album - 1
    WHERE :old.album_id = album.album_id;
  END IF;
END;

DML
insert into artist (artist_id, artist_name, country, num_albums, num_songs) values (1234,'2-PAC','US', 0, 0);
insert into artist (artist_id, artist_name, country, num_albums, num_songs) values (1235, 'Blink 182', 'US', 0, 0);
insert into artist (artist_id, artist_name, country, num_albums, num_songs) values (1236, 'James Durbin', 'US', 0, 0);
insert into artist (artist_id, artist_name, country, num_albums, num_songs) values (1237, 'Jay-Z', 'US', 0, 0);
insert into artist (artist_id, artist_name, country, num_albums, num_songs) values (1238, 'Katy Perry', 'US', 0, 0);
insert into artist (artist_id, artist_name, country, num_albums, num_songs) values (1239, 'Machine Head', 'US', 0, 0);
insert into artist (artist_id, artist_name, country, num_albums, num_songs) values (1240, 'Megadeth', 'US', 0, 0);
insert into artist (artist_id, artist_name, country, num_albums, num_songs) values(1241, 'Nickelback', 'US', 0, 0);
insert into artist (artist_id, artist_name, country, num_albums, num_songs) values (1242, 'Nirvana', 'US', 0, 0);
insert into artist (artist_id, artist_name, country, num_albums, num_songs) values (1243, 'Outkast', 'US', 0, 0);
insert into artist (artist_id, artist_name, country, num_albums, num_songs) values (1244, 'Pearl Jam', 'US', 0, 0);
insert into artist (artist_id, artist_name, country, num_albums, num_songs) values (1245, 'Stick to Your Guns', 'US', 0, 0);
insert into artist (artist_id, artist_name, country, num_albums, num_songs) values (1246, 'Switchfoot', 'US', 0, 0);
insert into artist (artist_id, artist_name, country, num_albums, num_songs) values(1247, 'The Pharcyde', 'US', 0, 0);

insert into album (album_id, album_title, artist_id, genre, num_songs_in_album) values (4321, 'Me Against The World', 1234, 'Hip Hop', 0);
insert into album (album_id, album_title, artist_id, genre, num_songs_in_album) values (4322, 'Enema of the State', 1235, 'Alternative Rock', 0);
insert into album (album_id, album_title, artist_id, genre, num_songs_in_album) values(4323, 'Memories of a Beautiful Disaster', 1236, 'Alternative Rock', 0);
insert into album (album_id, album_title, artist_id, genre, num_songs_in_album) values (4324, 'The Blueprint', 1237, 'Hip Hop', 0);
insert into album (album_id, album_title, artist_id, genre, num_songs_in_album) values (4325, 'Teenage Dream', 1238, 'Pop', 0);
insert into album (album_id, album_title, artist_id, genre, num_songs_in_album) values (4326, 'Unto the Locust', 1239, 'Alternative Rock', 0);
insert into album (album_id, album_title, artist_id, genre, num_songs_in_album) values (4327, 'Th1rt3en', 1240, 'Metal', 0);
insert into album (album_id, album_title, artist_id, genre, num_songs_in_album) values (4328, 'Here and Now', 1241, 'Pop', 0);
insert into album (album_id, album_title, artist_id, genre, num_songs_in_album) values (4329, 'Nevermind', 1242, 'Alternative Rock', 0);
CREATE OR REPLACE PROCEDURE insert_track (track_idno number, artist_idno number, album_idno number, pfname varchar2) IS

src_file BFILE;
dst_file BLOB;
lgh_file BINARY_INTEGER;
BEGIN
src_file := bfilename('TEMP', pfname);

-- insert a NULL record to lock
INSERT INTO track (track_id, artist_id, album_id, content)
VALUES (track_idno, artist_idno, album_idno, EMPTY_BLOB())
RETURNING content INTO dst_file;

-- lock record
SELECT content INTO dst_file
FROM track
WHERE artist_id = artist_idno
AND album_id = album_idno
AND track_id = track_idno
FOR UPDATE;

-- open the file
dbms_lob.fileopen(src_file, dbms_lob.file_readonly);

-- determine length
lgh_file := dbms_lob.getlength(src_file);

-- read the file
dbms_lob.loadfromfile(dst_file, src_file, lgh_file);
UPDATE track
SET content = dst_file
WHERE track_id = track_idno
AND artist_id = artist_idno
AND album_id = album_idno;

-- close file
dbms_lob.fileclose(src_file); update table with column data
END insert_track;

exec insert_track(1, 1234, 4321, 'Dear-Mama.m4a')
exec insert_track(2, 1235, 4322, 'All-the-Small-Things.m4a');
exec insert_track(3, 1236, 4323, 'Love-Me-Bad.m4a');
exec insert_track(4, 1237, 4324, 'Izzo-HOVA.m4a')
exec insert_track(5, 1238, 4325, 'The-One-That-Got-Away.m4a');
exec insert_track(6, 1239, 4326, 'Locust.m4a');
exec insert_track(7, 1240, 4327, 'Public-Enemy-No-1.m4a');
exec insert_track(8, 1241, 4328, 'When-We-Stand-Together.m4a')
exec insert_track(9, 1242, 4329, 'Smells-Like-Teen-Spirit.m4a');
exec insert_track(10, 1243, 4330, 'Players-Ball.m4a');
exec insert_track(11, 1244, 4331, 'Even-Flow.m4a');
exec insert_track(12, 1245, 4332, 'This-Is-More.m4a')
exec insert_track(13, 1246, 4333, 'Restless.m4a');
exec insert_track(14, 1247, 4334, 'Passin-Me-By.m4a');

CREATE OR REPLACE PROCEDURE insert_lyrics (lyric_idno number, track_idno number, version varchar2, pfname varchar2) IS

src_file BFILE;
dst_file BLOB;
lgh_file BINARY_INTEGER;
BEGIN
src_file := bfilename('TEMP', pfname);

-- insert a NULL record to lock
INSERT INTO lyrics (lyric_id, track_id, version , content)
VALUES (lyric_idno, track_idno, version, EMPTY_BLOB())
RETURNING content INTO dst_file;

-- lock record
SELECT content
INTO dst_file
FROM lyrics
END insert_lyrics;
WHERE  track_id = track_idno
AND lyric_id = lyric_idno
FOR UPDATE;

-- open the file
dbms_lob.fileopen(src_file, dbms_lob.file_readonly);

-- determine length
lgh_file := dbms_lob.getlength(src_file);

-- read the file
dbms_lob.loadfromfile(dst_file, src_file, lgh_file);

-- update the blob field
UPDATE lyrics
SET content = dst_file
WHERE track_id = track_idno
AND lyric_id = lyric_idno;

-- close file
dbms_lob.fileclose(src_file);
END insert_lyrics;

exec insert_lyrics(1, 1, 'radio', 'All-The-Small-Things-radio.doc');
exce insert_lyrics(2, 2, 'radio', 'All-The-Small-Things-radio.doc');
exce insert_lyrics(3, 3, 'radio', 'Love-Me-Bad.doc');
exce insert_lyrics(4, 4, 'explicit', 'Izzo-HOVA-Explicit.doc');
exce insert_lyrics(5, 5, 'radio', 'The-One-That-Got-Away.doc');
exce insert_lyrics(6, 6, 'radio', 'Locust.doc');
exce insert_lyrics(7, 7, 'explicit', 'Public-Enemy-No-1.doc');
exce insert_lyrics(8, 8, 'radio', 'When-We-Stand-Together.doc');
exce insert_lyrics(9, 9, 'radio', 'Smells-Like-Teen-Spirit.doc');
exce insert_lyrics(10, 10, 'explicit', 'Players-Ball.doc');
exce insert_lyrics(11, 11, 'radio', 'Even-Flow.doc');
exce insert_lyrics(12, 12, 'radio', 'This-Is-More.doc');
exce insert_lyrics(13, 13, 'radio', 'Restless.doc');
exce insert_lyrics(14, 14, 'explicit', 'Passin-Me-By.doc');

CREATE OR REPLACE PROCEDURE insert_video ( video_idno number,
vid_name varchar,
track_idno number,
pfname varchar2) IS

src_file BFILE;
dst_file BLOB;
lgh_file BINARY_INTEGER;
BEGIN
src_file := bfilename('TEMP', pfname);
-- insert a NULL record to lock
INSERT INTO video
  (video_id, video_name, track_id, content)
VALUES
  (video_idno, vid_name, track_idno, EMPTY_BLOB())
RETURNING content INTO dst_file;

-- lock record
SELECT content
  INTO dst_file
FROM video
WHERE track_id = track_idno
  AND video_id = video_idno
FOR UPDATE;

-- open the file
dbms_lob.fileopen(src_file, dbms_lob.file_readonly);

-- determine length
lgh_file := dbms_lob.getlength(src_file);

-- read the file
dbms_lob.loadfromfile(dst_file, src_file, lgh_file);

-- update the blob field
UPDATE video
  SET content = dst_file
WHERE video_id = video_idno
  AND track_id = track_idno;

-- close file
dbms_lob.fileclose(src_file);
END insert_video;

exec insert_video(1, 'Dear-Mama', 1, 'Dear-Mama.m4v');
exec insert_video(2, 'All-the-Small-Things', 2, 'All-the-Small-Things.m4v');
exec insert_video(3, 'Love-Me-Bad', 3, 'Love-Me-Bad.m4v');
exec insert_video(4, 'Izzo-HOVA', 4, 'Izzo-HOVA.m4v');
exec insert_video(5, 'The-One-That-Got-Away', 5, 'The-One-That-Got-Away.m4v');
exec insert_video(6, 'Locust', 6, 'Locust.m4v');
exec insert_video(7, 'Public-Enemy-No-1', 7, 'Public-Enemy-No-1.m4v');
exec insert_video(8, 'When-We-Stand-Together', 8, 'When-We-Stand-Together.m4v');
exec insert_video(9, 'Smells-Like-Teen-Spirit', 9, 'Smells-Like-Teen-Spirit.m4v');
exec insert_video(10, 'Players-Ball', 10, 'Players-Ball.m4v');
exec insert_video(11, 'Even-Flow', 11, 'Even-Flow.m4v');
exec insert_video(12, 'This-is-More', 12, 'This-is-More.m4v');
exec insert_video(13, 'Restless', 13, 'Restless.m4v');
exec insert_video(14, 'Passin-Me-By', 14, 'Passin-Me-By.m4v');
PROCEDURES

For a given artist name, find all the videos of the artist.

CREATE OR REPLACE PROCEDURE find_artist_videos (a_name varchar2) IS
    iterate number;
    numrecs number;

    video_rec video.video_name%TYPE;
    CURSOR focus IS
    SELECT video.video_name
    FROM video
    JOIN track
    ON video.track_id = track.track_id
    JOIN artist
    ON artist.artist_id = track.artist_id
    WHERE artist.artist_name = a_name;

    BEGIN
        SELECT COUNT(video_id) --count number of columns in the
        INTO numrecs
        FROM video;

        OPEN focus;
        FOR iterate in 1..numrecs LOOP
            FETCH focus INTO video_rec;
            EXIT WHEN focus%NOTFOUND;
            dbms_output.put_line(video_rec);
        END LOOP;
        CLOSE focus;
        COMMIT;
    END find_artist_videos;

For a given genre, find all the videos of the artist’s genre.

CREATE OR REPLACE PROCEDURE find_genre_videos (a_genre varchar2) IS
    iterate number;
    numrecs number;

    video_rec video.video_name%TYPE;
    CURSOR focus IS
    SELECT video.video_name
    FROM video
    JOIN track
    ON video.track_id = track.track_id
For a given video, find the information of the album and the artist.

CREATE OR REPLACE PROCEDURE find_album_artist (a_video varchar2) IS
iterate number;
umrecs number;

artist_rec artist%ROWTYPE;
CURSOR focus IS
SELECT artist.artist_id, artist.artist_name, artist.country, artist.num_albums, artist.num_songs
FROM artist
JOIN track
ON  track.artist_id = artist.artist_id
JOIN video
ON video.track_id = track.track_id
WHERE  video.video_name = a_video;

album_rec album%ROWTYPE;
CURSOR target IS
SELECT album.album_id, album.album_title, album.artist_id, album.genre,
album.num_songs_in_album
FROM album
JOIN artist
ON artist.artist_id = album.artist_id
JOIN track
ON  track.artist_id = artist.artist_id
JOIN video
ON video.track_id = track.track_id
WHERE video.video_name = a_video;

BEGIN
    SELECT COUNT(video_id) --count number of columns in the
    INTO numrecs
    FROM video;
    OPEN target;
    FOR iterate in 1..numrecs LOOP
        FETCH target INTO album_rec;
        EXIT WHEN target%NOTFOUND;
        dbms_output.put_line(RPAD('Album ID', 10, ' ') || RPAD('Album Title', 22, ' ') || RPAD('Artist ID', 10, ' ') || RPAD('Genre', 7, ' ') || 'num_songs_in_album');
        dbms_output.put_line(RPAD(album_rec.album_id, 9, ' ') || RPAD(album_rec.album_title, 25, ' ') || RPAD(album_rec.artist_id, 9, ' ') || RPAD(album_rec.genre, 10, ' ') || album_rec.num_songs_in_album);
    END LOOP;
    CLOSE target;
    OPEN focus;
    FOR iterate in 1..numrecs LOOP
        FETCH focus INTO artist_rec;
        EXIT WHEN focus%NOTFOUND;
        dbms_output.put_line(RPAD('Artist ID', 12, ' ') || RPAD('Artist Name',15, ' ') || RPAD('Country', 9, ' ') || RPAD('Num_Albums', 5, ' ') || 'Num Songs');
        dbms_output.put_line(RPAD(artist_rec.artist_id, 9, ' ') || RPAD(artist_rec.artist_name,15, ' ') || RPAD(artist_rec.country, 4, ' ') || RPAD(artist_rec.num_albums, 5, ' ') || artist_rec.num_songs);
    END LOOP;
    CLOSE focus;
    COMMIT;
END find_album_artist;

QUERIES

For a given genre, find all the songs, albums, existence of videos

SELECT track.track_title, album.album_title, video.video_name
FROM track
JOIN album
ON album.artist_id = track.artist_id
JOIN video
ON video.track_id = track.track_id
WHERE album.genre = 'Hip Hop';

For a given album, display, track#, track title, artist, genre, title of video.
SELECT album.num_songs_in_album, track.track_title, artist.artist_name, album.genre, video.video_name
FROM album
JOIN track
ON track.artist_id = album.artist_id
JOIN video
ON video.track_id = track.track_id
JOIN artist
ON artist.artist_id = track.artist_id
WHERE album.album_title = 'The Blueprint';

Evaluation of implementation: The code is easy to read, so comments are sparse. The implementation meets the specified requirements of the project, namely, it supports the manipulation of three different types of multimedia files.

Problems & known bugs: There is no exception handling for erroneous data in this prototype, so unexpected outcomes will result from invalid input.

Conclusion: In this project, a prototype multimedia DBMS was created which manipulates three different types of data: documents, audio files, and videos. Also, the project includes interesting procedures and queries which select different attributes related to the multimedia files and displays them to the user. Working on this project, I learned how to create triggers which perform different actions based on insertion and deletion of data, how to insert LOB files using special PL/SQL procedures, and of course how to write moderately complex procedures and queries for selecting data from the system.

Future work: With more time, I would add a GUI which allows the users to insert/delete multimedia files using a more intuitive interface than sqlplus.

Work plans: This was an individual project, so all of the work fell on my shoulders.

References:


