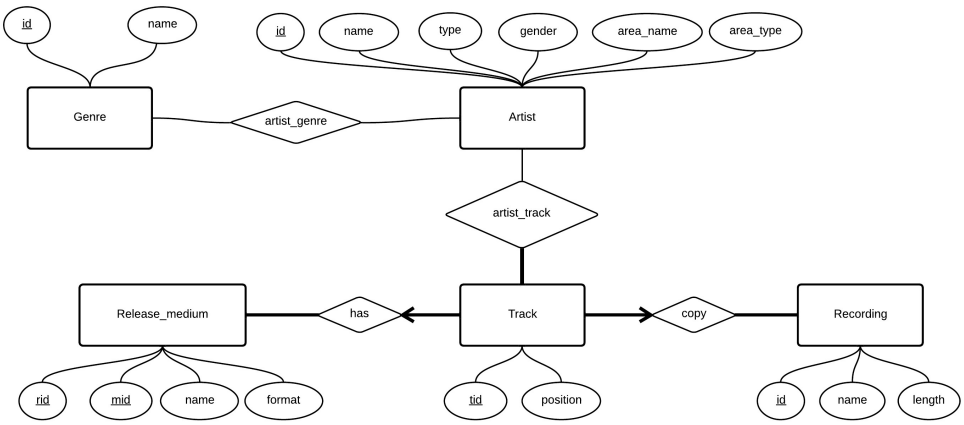
Project Report

Group 18: Chengzhen Wu, Huiqi Mao, Ruofan Zhou

May 7, 2014

1 ER model

After reading the feedback of diverable1, we modified our ER model as below:



In the given project data, we firstly recognize 'area', 'artist' and 'genre' each as three individual entities. Each artist is from at most one area, so it's a many-to-one relation. Several artists can belong to different genres and one genre can contain several artists. So the relation between 'artist' and 'genre' is many-to-many.

Secondly, we think about the relationship among 'release', 'recording', track' and 'medium'. We imagine a scene to describe these relations. The csv file of 'release' contains the names of releases. They could be stored in the mediums, such as CD, 12" Vinyl and so on. What's more, one release could have several CDs to contain many tracks, or in different medium (I'm not sure about this, but possible). So the relation between 'release' and 'medium' is one-to-many. Next, each track in different mediums must correspond to one recording. So the relation between 'track' and 'recording' is many-to-one. Each 'track' must be in one of 'medium's. So the relation is many-to-one.

Finally, we merge one-to-may relations. 'release' and 'medium' are merged into 'Release_medium'. 'area' can be merged into 'artist' as attributes. The 'has' relation between 'Release_medium' and 'track' is merged into 'track' using 'mid' as foreign key. So is the 'copy' relation between 'recording' and 'track' using 'id' of recording as foreign key.

Additionally, we ignore the 'count' in 'genre', which could be created as view in the database.

2 SQL based on ER model

```
--ENTITY Genre
--we don't need gcount in genre
CREATE TABLE  Genre (
  GID INTEGER,
  Gname VARCHAR(100),
  PRIMARY KEY (GID)
  );

--ENTITY Artist
CREATE TABLE  Artist (
  AID INTEGER,
  Aname VARCHAR(100),
  Atype CHAR(6),
  gender CHAR(6),
  area_name VARCHAR(60),
  area_type CHAR(15)
  PRIMARY KEY (AID)
  );

--ENTITY Recording
CREATE TABLE  recording (
  RID INTEGER,
  Rname VARCHAR(100),
  Rlength INTEGER,
  PRIMARY KEY (RID)
  );

--ENTITY RelaseMedium
CREATE TABLE  ReleaseMedium (
  MID INTEGER,
  RID INTEGER,
  name VARCHAR(400),
  format CHAR(45),
  PRIMARY KEY (RID,MID)
  );

--ENTITY Track
CREATE TABLE  Track (
  TID INTEGER,
  position INTEGER,
  MID INTEGER NOT NULL,
  RID INTEGER NOT NULL,
  REID INTEGER NOT NULL,
  PRIMARY KEY (TID),
  FOREIGN KEY (RID,MID) REFERENCES ReleaseMediuma ON DELETE NO ACTION,
  FOREIGN KEY (REID) REFERENCES Recording ON DELETE NO ACTION
  );

--RELATIONSHIP artist_genre
CREATE TABLE artist_genre (
  AID INTEGER NOT NULL,
  GID INTEGER NOT NULL,
  PRIMARY KEY (AID, GID),
  FOREIGN KEY (AID) REFERENCES Artist ON DELETE NO ACTION ,
  FOREIGN KEY (GID) REFERENCES Genre ON DELETE NO ACTION
  );

--RELATIONSHIP artist_track
CREATE TABLE artist_track(
  AID INTEGER NOT NULL,
  TID INTEGER NOT NULL,
```

```
  PRIMARY KEY (AID, TID),
  FOREIGN KEY (AID) REFERENCES Artist ON DELETE NO ACTION ,
  FOREIGN KEY (TID) REFERENCES Track ON DELETE NO ACTION
  );
```

3 Alternative based on Real Data

When importing the real data, we find several problems. So we decide to change the schema in order to import as much data as possible.

In our ER model, we merged the 'area' and 'artist'. However, since the data is not incomplete, a few AreaIDs in 'artist' don't appear in 'area', such as 'AreaID=241'. So we use an 'area' entity instead. And the foreign key in 'artist' is disabled. Other foreign key constraints, except in 'medium', are also disabled to make it easy to import data. Additionally, in the original file of 'artist.csv', we change '
N' to '0', which is null in the 'area' table.

Another problem with the data is deplicate, since the data in artist_track has duplicate, we set the foreign-key disabled in the table artist_track.

We separate the release_medium table into 'release' and 'medium', considering the limited space. Because the name of release could be very long. Duplication could cause a waste of memory. But the foreign key in 'medium' is still retained, because it works.

In the 'genre' entity, 'count' is remained for further possible queries(We find that 'count' from real data is not really the count we caculate).

4 Alternative SQL

```
CREATE TABLE  Init_Genre (
  GID VARCHAR(20),
  Gname VARCHAR(300),
  count Integer,
  PRIMARY KEY (GID)
  );

create table Init_Area(
  AreaID VARCHAR(20),
  AreaName VARCHAR(150),
  Area_type VARCHAR(15),
  PRIMARY KEY (AreaID)
  );

CREATE TABLE  Init_Artist (
  AID VARCHAR(20),
  Aname VARCHAR(200),
  Atype CHAR(10),
  gender CHAR(6),
  --area_name VARCHAR(100),
  --area_type CHAR(15),
  AreaID VARCHAR(20),
  PRIMARY KEY (AID),
  foreign key (AreaID) references Init_area(AreaID)
  );
```

```
CREATE TABLE Init_Recording (
   RID VARCHAR(20),
   Rname VARCHAR(2000),
   Rlength VARCHAR(20),
   PRIMARY KEY (RID)
   );


CREATE TABLE Init_Release (
   RID VARCHAR(20),
   name VARCHAR(1000),
   PRIMARY KEY (RID)
   );

CREATE TABLE Init_Medium (
   MID VARCHAR(20),
   RID VARCHAR(20),
   format CHAR(45),
   PRIMARY KEY (MID),
   FOREIGN KEY (RID) REFERENCES Init_Release(RID)
   );


CREATE TABLE  Init_Track (
   TID VARCHAR(20),
   REID VARCHAR(20) NOT NULL,
   MID VARCHAR(20) NOT NULL,
   position INTEGER,
   PRIMARY KEY (TID),
   FOREIGN KEY (MID) REFERENCES Init_Medium(MID),
   FOREIGN KEY (REID) REFERENCES Init_Recording(RID)
   );

CREATE TABLE Init_artist_genre (
   AID VARCHAR(20) NOT NULL,
   GID VARCHAR(20) NOT NULL,
   PRIMARY KEY (AID, GID),
   FOREIGN KEY (AID) REFERENCES Init_Artist(AID) ON DELETE CASCADE,
   FOREIGN KEY (GID) REFERENCES Init_Genre(GID) ON DELETE CASCADE
   );

CREATE TABLE Init_artist_track(
   AID VARCHAR(20) NOT NULL,
   TID VARCHAR(20) NOT NULL,
   PRIMARY KEY (AID, TID),
   FOREIGN KEY (AID) REFERENCES Init_Artist(AID), --these foreign keys are disabled for data duplicate
   FOREIGN KEY (TID) REFERENCES Init_Track(TID)
   );
```

5 Queries

We finished the queries based on our model:

A.

B.

C.

D.

E.

F.

G.

6 Interface