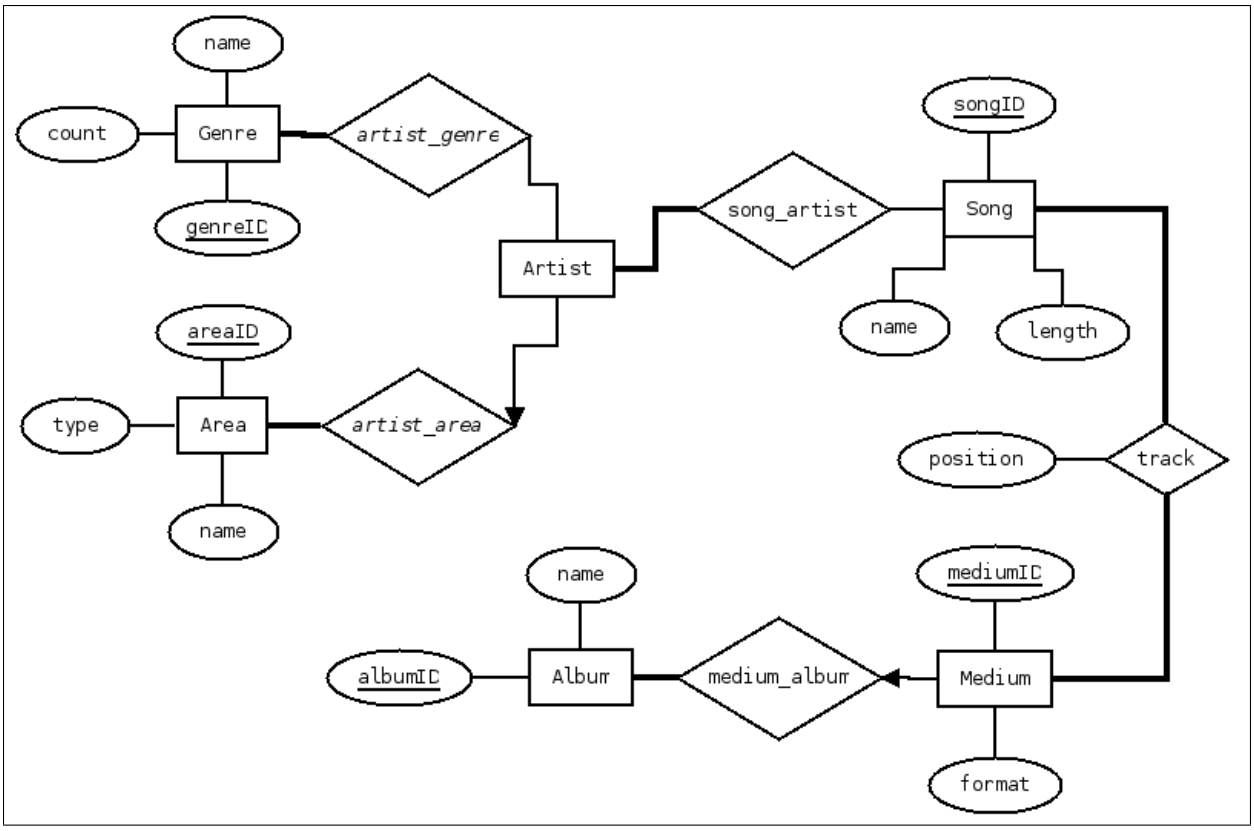
CS-322 Introduction to Database Systems
Project Deliverable #1

Due on Sunday, March 23^{rd}, 2014

Group 24
Klay, Ameho, Vinh Mau

ER model for music database



SQL DDL code for table creations

```
CREATE TABLE Areas (
  areaID CHAR(20),
  name CHAR(40) NOT NULL,
  type CHAR(20),
  PRIMARY KEY (areaID) ) ;

CREATE TABLE Genres (
  genreID CHAR(20),
  name CHAR(40) NOT NULL,
  count INTEGER DEFAULT 0,
  PRIMARY KEY (genreID) ) ;

CREATE TABLE Artists (
  artistID CHAR(20),
  name CHAR(40) NOT NULL,
  areaID CHAR(20),
  PRIMARY KEY (artistID),
  FOREIGN KEY (areaID) REFERENCES Areas ) ;


CREATE TABLE Songs (
  songID CHAR(20),
```

```
  name CHAR(40) ,
  length INTEGER,
  PRIMARY KEY (songID) ) ;


CREATE TABLE Albums (
  albumID CHAR(20),
  name CHAR(40) NOT NULL,
  PRIMARY KEY (albumID) ) ;

CREATE TABLE Mediums (
  mediumID CHAR(20),
  albumID CHAR(20),
  format CHAR(20),
  PRIMARY KEY (mediumID),
  FOREIGN KEY (albumID) REFERENCES Albums ) ;
```

SQL script for entities table creation

```
CREATE TABLE Artist_genre (
  artistID CHAR(20),
  genreID CHAR(20),
  PRIMARY KEY (artistID, genreID),
  FOREIGN KEY (artistID) REFERENCES Artists,
  FOREIGN KEY (genreID) REFERENCES Genres ) ;

CREATE TABLE Song_artist (
  songID CHAR(20),
  artistID CHAR(20),
  PRIMARY KEY (songID, artistID),
  FOREIGN KEY (songID) REFERENCES Songs ,
  FOREIGN KEY (artistID) REFERENCES Artists ) ;

CREATE TABLE Tracks (
  songID CHAR(20),
  mediumID CHAR(20),
  position INTEGER,
  PRIMARY KEY (songID, mediumID),
  FOREIGN KEY (songID) REFERENCES Songs ,
  FOREIGN KEY (mediumID) REFERENCES Mediums ) ;
```

SQL script for relations table creation

Design choices & data constraints

There are three main concepts in our music database : **Song**, **Artist** and **Album**. Both **Song** and **Album** were divided between their descriptive data and their physical incarnation. We decided to enforce **Song** as the only necessary information to describe music, emulating the approach taken by most popular music player softwares since data can be incomplete. We put a NOT NULL constraint on most of the name attributes of the entities, with the exception of **Song** for the reason just stated. Since they are not required fields to describe music, they should have a valid name when they are in fact used.

- A **Song** is related to:

  Artist: A song can exist without known artists, but can also have several artists to describe collaborations.

  Medium: Though a song is not necessarily part of an album, it has to be recorded on some medium. There is therefore a participation constraint of **Song** in **Medium**. Their relation is characterized by the track position on the medium.

- An **Artist** is defined by a:

  Genre: A genre can regroup multiple artists, but makes no sense as an empty container, thus triggering a participation constraint, whereas an artist can be difficult to define as catering to a specific genre, or crossing boundaries between genres nullifying the need for a constraint. We kept the count attribute, choosing small update costs over on-demand higher computation costs.

  Area: An artist's location can be pinpointed to a specific creation grounds, hence can be expressed by a foreign key constraint. But several artists can be compelled to share their musical feelings in the same studio, and a place which doesn't house such a creative conundrum isn't worth keeping track of in a music database.

- An **Album** is the logical aggregation of songs, labeled by a title, and can be recorded on multiple — at least one, participation constraint— media. Conversely, a medium identifies a singular recording of an album, enforced by a foreign key constraint.

Some constraints, the integrity of the count attribute in **Genre** and the several "at least one" constraints, are not guaranteed by the table creation. They will later be enforced later on by the import and delete data commands.