

CS-322 Introduction to Database Systems

Project Deliverable #1

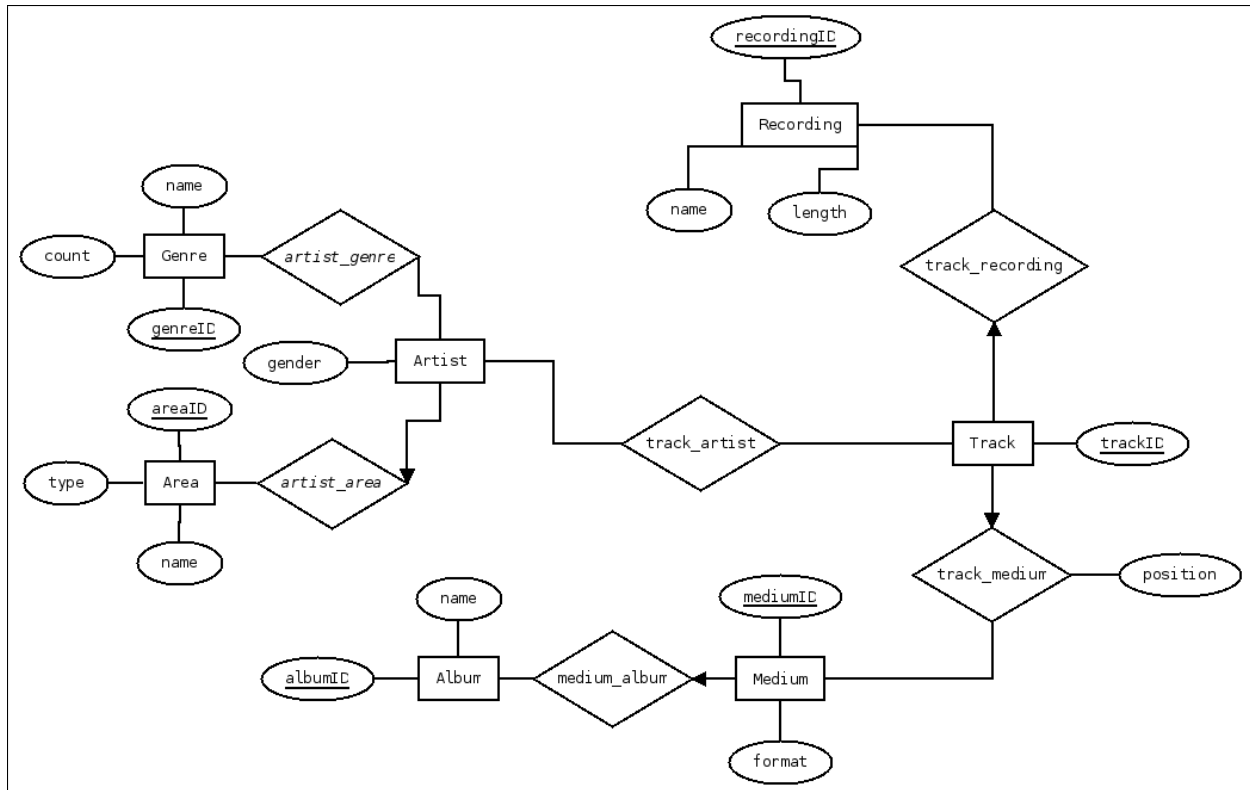
Due on Sunday, March 23rd, 2014

Group 24
Klay, Ameho, Vinh Mau



ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

ER model for music database



SQL DDL code for table creations

```
CREATE TABLE Areas (
  areaID INTEGER,
  name VARCHAR NOT NULL,
  type VARCHAR,
  PRIMARY KEY (areaID) );
```

```
CREATE TABLE Genres (
  genreID INTEGER,
  name VARCHAR NOT NULL,
  count INTEGER DEFAULT 0,
  PRIMARY KEY (genreID) );
```

```
CREATE TABLE Artists (
  artistID INTEGER,
  name VARCHAR NOT NULL,
  areaID INTEGER,
  gender CHAR(1),
  PRIMARY KEY (artistID),
  FOREIGN KEY (areaID) REFERENCES Areas );
```

```
CREATE TABLE Recordings (
  recordingID INTEGER,
```

```
25     name VARCHAR ,
    length INTEGER,
    PRIMARY KEY (recordingID) ) ;

30 CREATE TABLE Albums (
    albumID INTEGER,
    name VARCHAR NOT NULL,
    PRIMARY KEY (albumID) ) ;

35 CREATE TABLE Mediums (
    mediumID INTEGER,
    albumID INTEGER,
    format VARCHAR,
    PRIMARY KEY (mediumID),
40    FOREIGN KEY (albumID) REFERENCES Albums ) ;

CREATE TABLE Tracks (
45    trackID INTEGER,
    mediumID INTEGER,
    recordingID INTEGER,
    position INTEGER,
    PRIMARY KEY (trackID),
    FOREIGN KEY (mediumID) REFERENCES Mediums,
50    FOREIGN KEY (recordingID) REFERENCES Recordings ) ;
```

SQL script for entities table creation

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

CREATE TABLE Track_artist (
10    artistID INTEGER,
    trackID INTEGER,
    PRIMARY KEY (artistID, trackID),
    FOREIGN KEY (trackID) REFERENCES Tracks ,
    FOREIGN KEY (artistID) REFERENCES Artists ) ;
```

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.