Josh Huff, huffj@oregonstate.edu
Professor Wolford
CS 340 – Winter 2017
March 19, 2017

Josh Huff's Metal Database (jhmdB)
Project Outline

What is Josh Huff's Metal Database (jhmDB)?

The jhmDB is a database for cataloging the artists, songs, albums, and subgenres of heavy metal music. Heavy metal is a lot like spicy hot sauce and dark humor – it isn't for everyone, but those who appreciate it tend to have a deep and abiding love for it.

This database could help fans of the genre (typically called "metalheads") find new artists to follow (by searching for the year a band was formed and limiting it to the very recent), conceptualize a history of the genre (by searching for specific years that albums came out), try new subgenres (by searching for songs and albums of that type), and settle arguments about where a band comes from and how many albums they have written and recorded.

Josh Huff, huffj@oregonstate.edu
Professor Wolford
CS 340 – Winter 2017
March 19, 2017

Josh Huff's Metal Database (jhmdB)
Database Outline

Entities and Relationships

band

Bands in the collection have a unique name. Bands have attributes such as country of origin, number of members, and year of formation. A band can have one or many albums. An album belongs to only one band. A band's primary key is an auto-incrementing integer.

album

Albums have attributes name and year of release. A band can have one or many albums. An album can belong to only one band. A track may appear on many albums, and albums contain many tracks. Every album has one subgenre, but a subgenre often contains many albums. An album's primary key is an auto-incrementing integer.

track

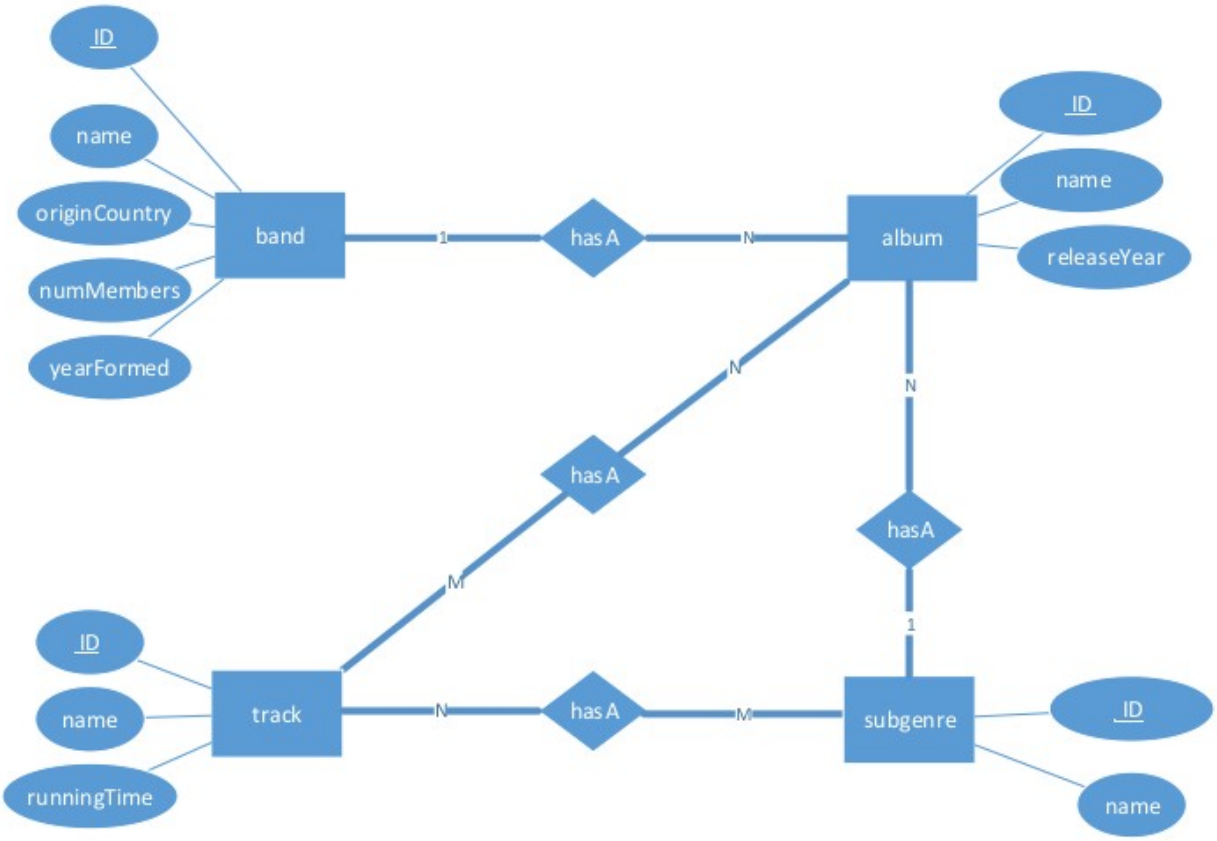
Tracks (usually, but not always, songs) have attributes name and running time. A track may appear on many albums, and albums contain many tracks. A track may be considered of one or more subgenres and subgenres contain many tracks. A track's primary key is an auto-incrementing integer.

subgenre

Subgenres are labels that describe the sound of an album or track. The main categories are: alternative, black, classic, death, doom, industrial, power, thrash. There are others, but the endless subdivisions get uselessly complex and nit-picky; some combination of the above is sufficient. Every album has one subgenre, but a subgenre often contains many albums. A track may be considered of one or more subgenres and subgenres contain many tracks. A subgenre's primary key is an auto-incrementing integer.

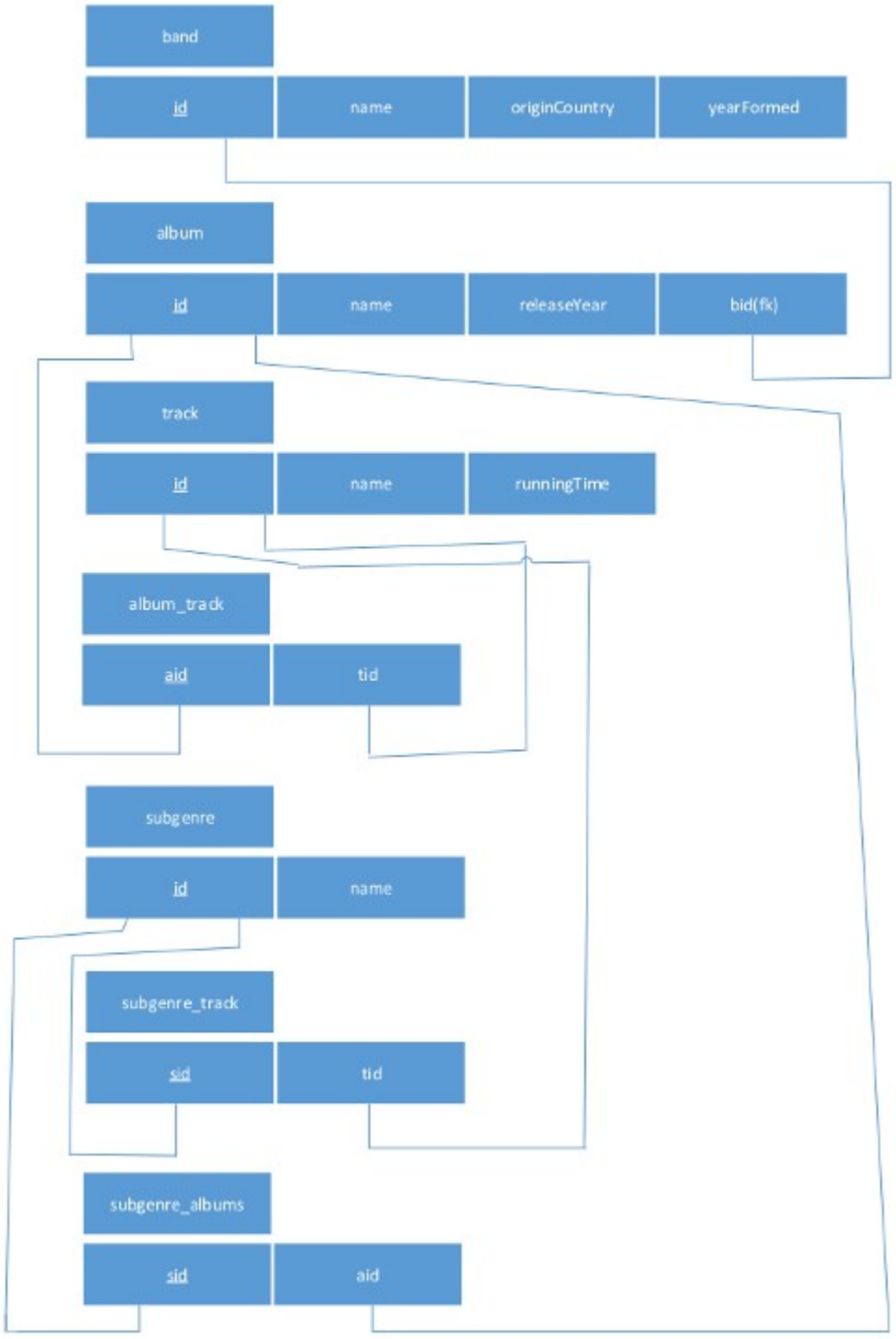
Josh Huff, huffj@oregonstate.edu
Professor Wolford
CS 340 – Winter 2017
March 19, 2017

Josh Huff's Metal Database (jhmdB)
ER Diagram



Josh Huff, huffj@oregonstate.edu
Professor Wolford
CS 340 – Winter 2017
March 19, 2017

Josh Huff's Metal Database (jhmdB)
Database Schema



Josh Huff, huffj@oregonstate.edu
Professor Wolford
CS 340 – Winter 2017
March 19, 2017

Josh Huff's Metal Database (jhmdB)
Table Creation Queries

```
CREATE TABLE `band` (
    `id` int(11) PRIMARY KEY AUTO_INCREMENT,
    `name` VARCHAR(255) NOT NULL,
    `originCountry` VARCHAR(255),
    `numMembers` TINYINT,
    `yearFormed` int(11),
    UNIQUE KEY(name)
) ENGINE=InnoDB;

CREATE TABLE `album` (
    `id` int(11) PRIMARY KEY AUTO_INCREMENT,
    `name` VARCHAR(255) NOT NULL,
    `releaseYear` INT,
    `bid` int(11),
    FOREIGN KEY (`bid`) REFERENCES `band`(`id`)
) ENGINE=InnoDB;

CREATE TABLE `track` (
    `id` int(11) PRIMARY KEY AUTO_INCREMENT,
    `name` VARCHAR(255) NOT NULL,
    `runningTime` int(11)
) ENGINE=InnoDB;

CREATE TABLE `album_track` (
    `aid` int(11) NOT NULL,
    `tid` int(11) NOT NULL,
    PRIMARY KEY (`aid`, `tid`),
    FOREIGN KEY (`aid`) REFERENCES `album`(`id`),
    FOREIGN KEY (`tid`) REFERENCES `track`(`id`)
) ENGINE=InnoDB;

CREATE TABLE `subgenre` (
    `id` int(11) PRIMARY KEY AUTO_INCREMENT,
    `name` VARCHAR(255) NOT NULL
) ENGINE=InnoDB;

CREATE TABLE `subgenre_track` (
    `sid` int(11) NOT NULL,
    `tid` int(11) NOT NULL,
    FOREIGN KEY (`sid`) REFERENCES `subgenre`(`id`),
    FOREIGN KEY (`tid`) REFERENCES `track`(`id`),
    PRIMARY KEY (`sid`, `tid`)
) ENGINE = 'innoDB';

CREATE TABLE `subgenre_albums` (
    `sid` int(11) NOT NULL,
    `aid` int(11) NOT NULL,
    PRIMARY KEY (`sid`, `aid`),
    FOREIGN KEY (`sid`) REFERENCES `subgenre`(`id`),
    FOREIGN KEY (`aid`) REFERENCES `album`(`id`)
) ENGINE = 'innoDB';
```

Josh Huff, huffj@oregonstate.edu
Professor Wolford
CS 340 – Winter 2017
March 19, 2017

Josh Huff's Metal Database (jhmdB)
General Use Queries

INSERTION QUERIES

```
/**** Inserting into Band Table
*/

INSERT INTO band (name, originCountry, numMembers, yearFormed) VALUES
   ([bandName], [country], [numberOfMembers], [yearOfFormation]);

/**** Inserting into Album Table
*/

INSERT INTO album (name, releaseYear, bid) VALUES
   ([albumName], [yearOfRelease], [bandId]);

/**** Inserting into Track table
*/

INSERT INTO track (name, runningTime) VALUES
   ([trackName], [duration]);

/**** Inserting into Album/Track table
*/

INSERT INTO album_track (aid, tid) VALUES
   ([albumId], [trackId]);

/**** Inserting into Subgenre table
*/

INSERT INTO subgenre (name) VALUES
   ([subgenreName]);

/**** Inserting into Subgenre/Track table
*/

INSERT INTO subgenre_track (sid, tid) VALUES
   ([subgenreId], [trackId]);

/**** Inserting into Subgenre/Albums table
*/

INSERT INTO subgenre_album (sid, aid) VALUES
   ([subgenreId], [albumId]);
```

Josh Huff, huffj@oregonstate.edu
Professor Wolford
CS 340 – Winter 2017
March 19, 2017

Josh Huff's Metal Database (jhmdB)
General Use Queries

SELECTION QUERIES

```
/**** Selecting everything to do with a particular band.
*/

SELECT * band WHERE band.name = "[bandName]";

/**** Selecting only band names from a particular country
*/

SELECT band.name FROM band WHERE originCountry = "[countryName]";

/**** Selecting only band names and the names of their albums released before a particular year
*/

SELECT band.name, album.name FROM band
   INNER JOIN album
   ON band.id = album.bid
   WHERE releaseYear < [year];

/**** Selecting only track names of an artist that are considered a specific subgenre
*/
SELECT band.name, track.name FROM band
   INNER JOIN album
   ON album.bid = band.id
   INNER JOIN album_track
   ON album_track.aid = album.id
   INNER JOIN track
   ON track.id = album_track.tid
   INNER JOIN subgenre_track
   ON subgenre_track.tid = track.id
   WHERE subgenre.name = "[specificSubgenre]";

/**** Selecting only albums considered a specific subgenre
*/
SELECT album.name FROM album
   INNER JOIN subgenre_albums
   ON subgenre_albums.aid = album.id
   INNER JOIN subgenre
   ON subgenre.id = subgenre_albums.sid
   WHERE subgenre.name = "[specificSubgenre]";

/**** Updating the number of members in a band
*/

UPDATE band SET numMembers = [newNumber] WHERE band.id = [bandID];
```