Michael Patson
CS 340 Summer 2018

Website: http://flip3.engr.oregonstate.edu:7708/
Week 5 FINAL

Learning an instrument can be overwhelming. One of the hardest things to understand is how it all fits together. Therefore, I will make a database which helps novice players improve their skills. The database will ask for a student's name, and type of guitar they own (acoustic or electric). The database will track the chords the student has learned. It will also keep a database of all chords. The database will also track what songs a student has learned

Database Outline, in Words
The entities in the database will be:

- Student- The student is main entity and will have a relationship with every other entity in the database
    - s_id: Each student will be assigned an ID which is auto-incrementing and will serve as the primary key
    - fname: each student will have a string of up to 255 characters for their first name
    - lname: each student will have a string of up to 255 characters for their last name (this will default to null and can be left blank for privacy reasons)
    - vocals- can you sing? every band needs a singer, boolean- yes or no defaults to no. Will be implemented as a "tinyint"

- Guitar-info about the instrument the student has will be stored in the guitar entity
    - g_id: Each guitar will be assigned an ID which is auto-incrementing and will serve as the primary key
    - name: alias for the student to recognize their guitar, string of up to 255 characters
    - type: electric or acoustic, a string with a maximum of 8 characters (this will have to be one or the other)
    - student is a foreign key to student_ID

- Chords-
    - C_ID: int(11) primary key, the chord ID
    - name- a chord will have a name, this is a standard chord practice
    - major sound – chords have a major sound, one of the 12 notes and is a string
    - fingering- chords are composed of notes, standard convention
    - fret- this is an int, what fret is the root note in
- Songs-

- Song_Id primary key, song ID
- name: variable string, name of the song
- artist: original recording artist of song

The relations ships in my database will be

- **Students have guitars.** A student MAY have a guitar. A guitar must belong to one student.
- **Student Learned chords-**one student can learn chords
- **Students Learned songs-** student can learn songs, songs can be input with no student. A student then can claim the songs
- **Songs have chords-** many songs are composed of many chords. The same chords make up many songs.

Guitar Lesson Database

FEEDBACK FOR Draft 2)

I like your topic, I play guitar, too! I think your project is good, I have a few comments as following: ERD 1. yes 2. yes 3. yes 4. I am not sure but maybe you don't need to add chord#1~7 as attributes in your "song" table. The chords#1~7 are more like the sample data you need to entry. Schema 1. yes 2. yes 3. yes 4. no DDQ: 1. yes 2. yes 3. yes 4. yes

Chin-Chou Ko , Jul 11 at 1:39am

General: I would create a table called "songChords" or something similar which is just a song foreign key, chord foreign key, and chord number (if order of chords for a song is important). Given that songs can have up to 7 chords, you may end up with lots of empty cells ERD: - Everything looks accurate as best I can tell for the attributes, but I wish it was a little cleaner. For example, all the song chords lines could be cleaned up so they don't overlap the is composed relationship. - You have guitars in the ERD as full participation, but the document does not say every guitar needs a student - Should Chords Learned By Student relationship be M? the way it is now, 1 student can learn many chords but chords can be learned by up to 1 student - Songs are the same way as chords. As the diagram shows, students can learn 0 to N songs, but each song can be learned by 0 to 1 students - For the Songs to Chords relationship, I would recommend representing that with another table of relationships (song ID to Chord ID) then just make sure each song ID only has up to 7 relationships at a time. I would think querying would be easier, instead of checking 7 chord attributes 1 by 1, you could just query all the chord relationships attached to a single song. Schema: - I think your schema makes sense, I like how the arrows cascade (I don't however know if this is allowed) DDQ File: - Your databases imported without an issue. The only thing I would've wanted would be more sample data, at least something in all the tables. - You also allow guitars to not be referenced to a student, but your ERD shows that as mandatory. Overall: Aside from my comments, I think you did a great job. Some relationships I didn't understand made sense once I read them in the DDQ file, I would've preferred to see them in the Database Outline (ie: Guitars do not need to be related to a student since your on delete of a student foreign key allows null) so I knew what I was looking for in the ERD. Good luck! I tried to learn guitar back in college and gave up because it's literally impossible. Anything that would make it easier is awesome!

Brooks Przybylek , Jul 12 at 8:49p

My Response to feedback

To Chin-Chou:

Good feedback, not sure what the number yes and no response refer too.   Additionally, the chords are kept in a separate data table, chords can be changed/or modified for simplicity. For example, some students may play an F chord as a diminished chord instead of a true F chord based on level of experience.
To Brooks:
I have updated the documentation to include a guitar must be owned by a student… if you know any guitars that are not owned, let me know!

Good call on the student to chord relationship to student and the song relationship to student. What I wanted to show was that a student only needs to learn a song or chord once, but as we continue to learn in this class, the topics become more apparent. I was thinking it was more like a certification in the BSG world. But many songs can be learned by many students. Good comments and very helpful.

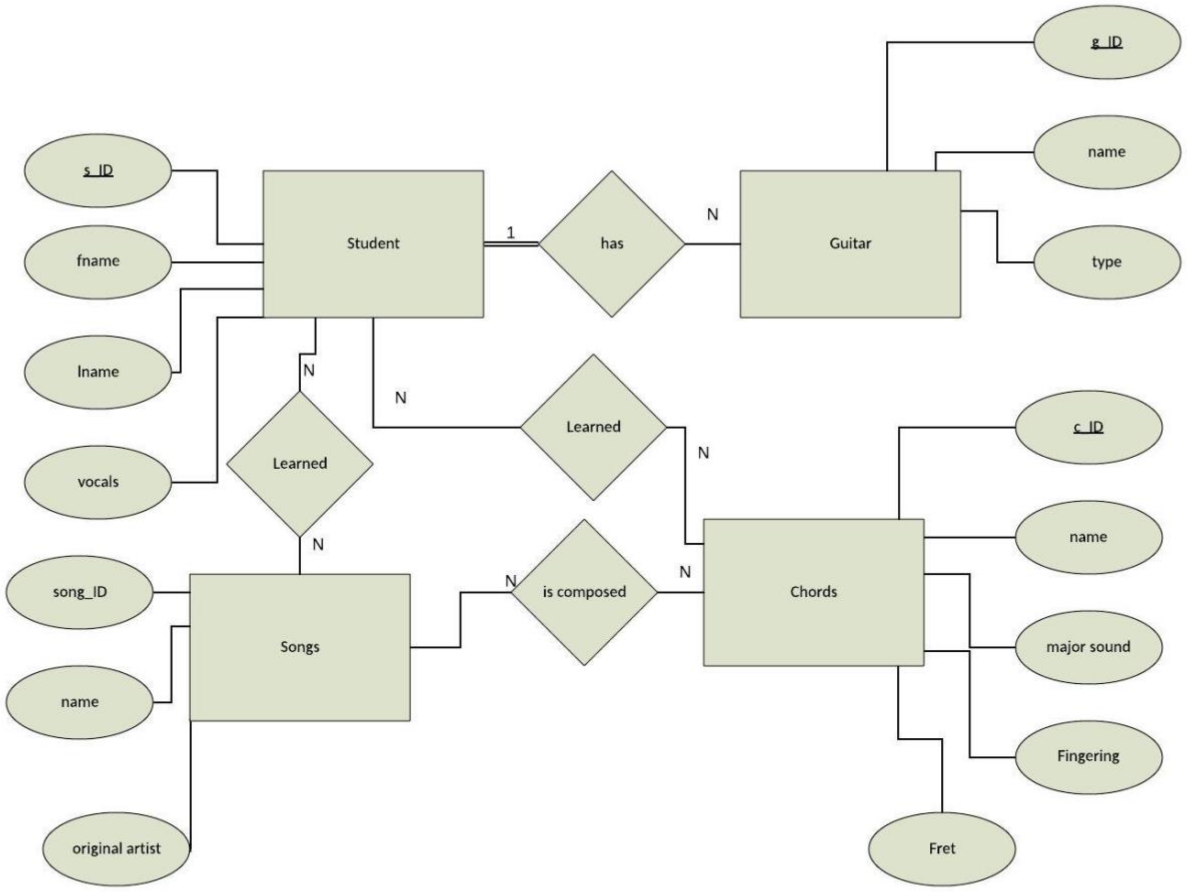
(FEEDBACK FOR DRAFT 1)UPDATED Project Outline

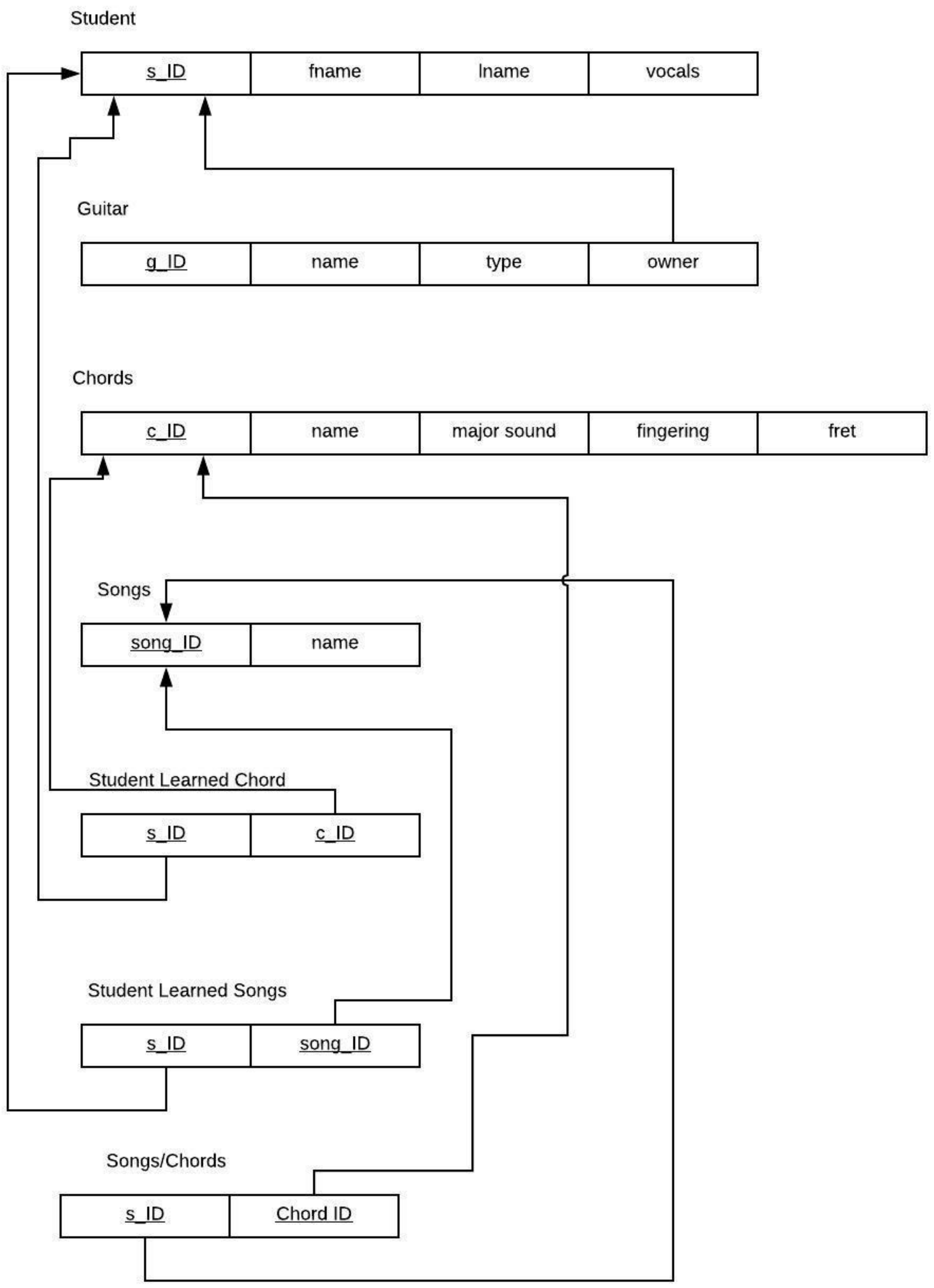
Feedback by the peer reviewer:
I like the choices for this database. It's a good simple people database. It's easy to understand that students will have an experience level with playing the guitar as well as the fact that there are different types of guitars. There are a few things tho that seem important. Although you covered pretty much all of the attributes for your entities, you missed out on specifying some constraints. You only specified the type of data. Yuo could add some details like, Student's can't have a guitar that doesn't exist, maybe students could have multiple of one type of guitar. Maybe the entry for first name should not be NULL. I was also wondering how you would decide to classify the certifications. Were they going to be a set of strings in a table with a unique id? or were they going numbers as an attribute to represent whether a person has both certifications or one? Because of this, Vocals and Certifications seem more like attributes for a student to me. Maybe you could add another entity table where it contains instructors to teach students guitars. That way you can have many students to one teacher or have teachers share students. -
-Austin Kwong

Actions based on the feedback:

Austin brought up how I left out some of the restraints on the database. These will be added. For example, a student cannot be certified twice for electric and acoustic guitar. I cannot specify all the types of guitars that exists, as Austin suggested, but I can certainly limit it to acoustic or electric.

Additionally the connection between certifications must have been unclear. I think it may be a big too complicated. After looking up some other examples of projects from students, I have decided to make a chord dictionary


g_ID
name
type
s_ID
fname
lname
vocals
Student
1
has
N
Guitar
N
N
Learned
Learned
N
N
c_ID
name
major sound
Fingering
Fret
Chords
N
is composed
N
song_ID
name
original artist
Songs


Student
s_ID
fname
lname
vocals
Guitar
g_ID
name
type
owner
Chords
c_ID
name
major sound
fingering
fret
Songs
song_ID
name
Student Learned Chord
s_ID
c_ID
Student Learned Songs
s_ID
song_ID
Songs/Chords
s_ID
Chord ID