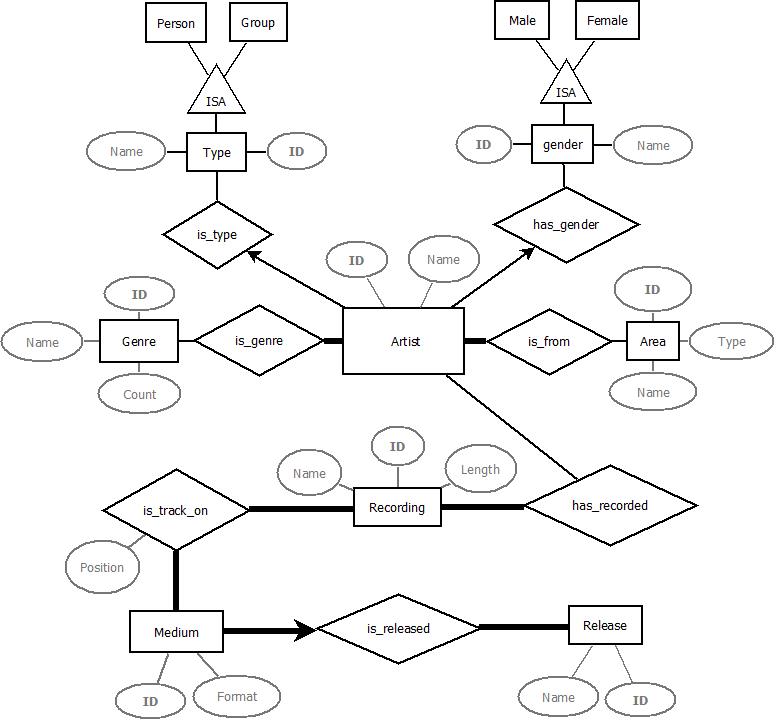
**1 Deliverable 1**

**1.1 ER Diagram**



**1.2 Design choices**

Because we cannot ask customers for their expectations about the database, we did several assumptions listed below. The goal was to maximize the number of caught constraints in our design while avoiding duplication. These are subject to change over time with our design evolution.

1. An artist can have several origins (AREAS) and several genres. However he can only have at most one GENDER and at most one TYPE when those are given.
2. Ideally every artist of type « Group » should have its gender attribute set to « Other ». This is a constraint that we cannot capture in our design.
3. We keep a name field in tables TYPE and GENDER to assure consistency in the beginning of this project, but it would be possible to get rid of them and use only the \_id field to establish an artist’s type or gender by the use of some arbitrary convention.
4. Every recordings in the database should be related to at least one artist, but an artist can have zero recording.
5. All the recordings must appear on at least one medium. A medium must have at least one recording.
6. With the add of a position attribute to the relation is\_track\_on, we capture the definition of a track.
7. A tuple of medium is related to exactly one release.
8. There is no cascade in deletion.

**1.3 SQL DDL code for table creation**

CREATE TABLE Type (

Type\_id INTEGER,

Name CHAR (10),

PRIMARY KEY (Type\_id)) ;

CREATE TABLE Gender (

Gender\_id INTEGER,

Name CHAR (10),

PRIMARY KEY (Gender\_id)) ;

CREATE TABLE Genre (

Genre\_id INTEGER,

Name CHAR (50),

Count INTEGER,

PRIMARY KEY (Genre\_id)) ;

CREATE TABLE Area (

Area\_id INTEGER,

Name CHAR (100),

Type\_of\_area CHAR (100),

PRIMARY KEY (Area\_id)) ;

CREATE TABLE Artist (

Artist\_id INTEGER,

Name CHAR (30),

Type\_id INTEGER,

Gender\_id INTEGER,

Genre\_id INTEGER NOT NULL,

Area\_id INTEGER NOT NULL,

PRIMARY KEY (Artist\_id),

FOREIGN KEY (Type\_id) REFERENCES Type,

FOREIGN KEY (Gender\_id) REFERENCES Gender,

FOREIGN KEY (Genre\_id) REFERENCES Genre,

FOREIGN KEY (Area\_id) REFERENCES Area) ;

CREATE TABLE Recording (

Recording\_id INTEGER,

Name CHAR (100),

Length INTEGER,

PRIMARY KEY (Recording\_id)) ;

CREATE TABLE has\_recorded (

Artist\_id INTEGER, // implicit NOT NULL (used in primary key)

Recording\_id INTEGER, // implicit NOT NULL (idem)

PRIMARY KEY (Artist\_id, Recording\_id),

FOREIGN KEY (Artist\_id) REFERENCES Artist,

FOREIGN KEY (Recording\_id) REFERENCES Recording) ;

CREATE TABLE Medium (

Medium\_id INTEGER,

Format CHAR (30),

Release\_id INTEGER NOT NULL,

PRIMARY KEY (Medium\_id),

FOREIGN KEY (Release\_id) REFERENCES Release) ;

CREATE TABLE Release (

Release\_id INTEGER,

Name CHAR (200),

PRIMARY KEY (Release\_id)) ;

CREATE TABLE is\_track\_on (

Recording\_id INTEGER,

Medium\_id INTEGER,

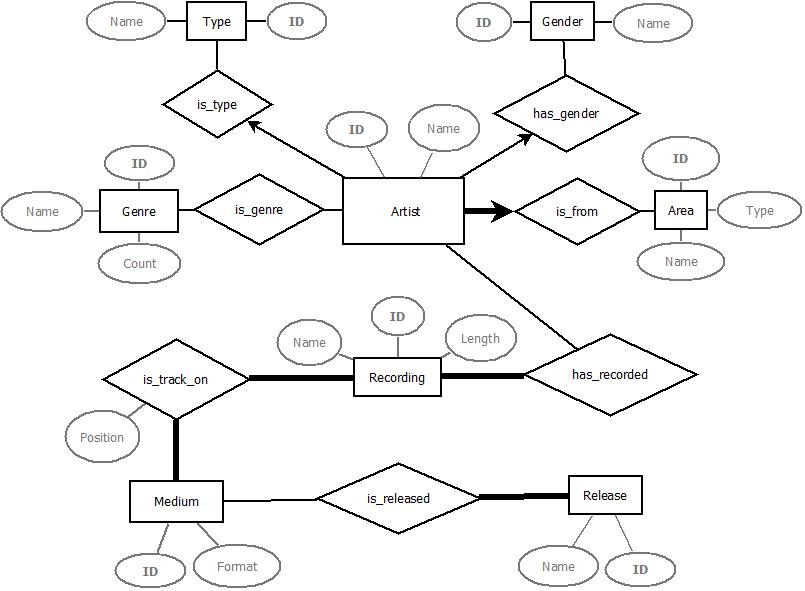
Position INTEGER,

PRIMARY KEY (Recording\_id, Medium\_id, Position),

FOREIGN KEY (Recording\_id) REFERENCES Recording,

FOREIGN KEY (Medium\_id) REFERENCES Medium) ;

**2 Deliverable 2**

**2.1 ER Diagram**

**2.2 Design Choices**

From the feedbacks of the deliverable 1, we made some improvements to our diagram.

First we got rid of the ISAs relationships because they were carrying more complexity than help. Each Type and Gender tables will contain few entries, respectively {‘Person’, ‘Group’} and {‘Male’, ‘Female’, ‘Other’}. This could allow us to add an eventual new Type if needed (not expected in this project though).

We limited the ability for an artist to be from more than one Area as proposed in the feedback. Now an artist must be from exactly one Area.

We allowed an artist to have no genre. Even though the number of genre should cover most needs we thought it was a useless limitation.

The new relation between Medium and Release allow the first to be part of several or none Release (many-to-many relation). We kept the fact that a Release must have at least one Medium.

We can use arbitrary convention for Type and Gender directly in queries, such as 0 for Person and 1 for Group; 0 for Male, 1 for Female and 2 for Other.

**2.3 SQL DDL Code for table creation**

CREATE TABLE Type (

Type\_id INTEGER,

Name CHAR (10),

PRIMARY KEY (Type\_id)) ;

CREATE TABLE Gender (

Gender\_id INTEGER,

Name CHAR (10),

PRIMARY KEY (Gender\_id)) ;

CREATE TABLE Genre (

Genre\_id INTEGER,

Name CHAR (50),

Count INTEGER,

PRIMARY KEY (Genre\_id)) ;

CREATE TABLE Area (

Area\_id INTEGER,

Area\_name CHAR (100),

Type\_of\_area CHAR (100),

PRIMARY KEY (Area\_id)) ;

CREATE TABLE Artist (

Artist\_id INTEGER,

Artist\_name CHAR (30),

Type\_id INTEGER,

Gender\_id INTEGER,

Area\_id INTEGER NOT NULL,

PRIMARY KEY (Artist\_id),

FOREIGN KEY (Type\_id) REFERENCES Type,

FOREIGN KEY (Gender\_id) REFERENCES Gender,

FOREIGN KEY (Area\_id) REFERENCES Area) ;

CREATE TABLE is\_genre (

Artist\_id INTEGER,

Genre\_id INTEGER,

PRIMARY KEY (Artist\_id, Genre\_id),

FOREGIN KEY (Artist\_id) REFERENCES Artist,

FOREIGN KEY (Genre\_id) REFERENCES Genre) ;

CREATE TABLE Recording (

Recording\_id INTEGER,

Name CHAR (100),

Length INTEGER,

PRIMARY KEY (Recording\_id)) ;

CREATE TABLE has\_recorded (

Artist\_id INTEGER, // implicit NOT NULL (used in primary key)

Recording\_id INTEGER, // implicit NOT NULL (idem)

PRIMARY KEY (Artist\_id, Recording\_id),

FOREIGN KEY (Artist\_id) REFERENCES Artist,

FOREIGN KEY (Recording\_id) REFERENCES Recording) ;

CREATE TABLE Medium (

Medium\_id INTEGER,

Format CHAR (30),

PRIMARY KEY (Medium\_id)) ;

CREATE TABLE Release (

Release\_id INTEGER,

Name CHAR (200),

PRIMARY KEY (Release\_id)) ;

CREATE TABLE is\_track\_on (

Recording\_id INTEGER,

Medium\_id INTEGER,

Position INTEGER,

PRIMARY KEY (Recording\_id, Medium\_id, Position),

FOREIGN KEY (Recording\_id) REFERENCES Recording,

FOREIGN KEY (Medium\_id) REFERENCES Medium) ;

CREATE TABLE is\_released (

Medium\_id INTEGER,

Release\_id INTEGER,

PRIMARY KEY (Medium\_id, Release\_id),

FOREIGN KEY (Medium\_id) REFERENCES Medium,

FOREIGN KEY (Release\_id) REFERENCES Release) ;

**2.4 Import of the Datas**

We created a table for each CSV in which we imported them. We used queries to extract the wanted datas in our own table. For instance, some datas such as ‘\N’ in recording.csv were changed to ‘-1’ or simply ignored when not convenient to our table definition.

**2.4 Queries**

A. Print the names of artists from Switzerland, i.e., artists whose area is Switzerland. You should not include the names of the artists associated with individual cantons and towns in Switzerland.

SELECT Artist\_name

FROM ( SELECT \*

FROM Area

WHERE Area\_name='Switzerland') NATURAL JOIN Artist

B. Print the names of areas with the highest number male artists, female artists and groups. For each of these 3 areas, print the number of artists of each of the three types in the area.

C. List the names of 10 groups with the most recorded tracks.

SELECT Artist.name

FROM ( SELECT Artist\_id, COUNT(Recording\_id) AS c1

FROM has\_recorded, Artist

WHERE Type\_id = '0'

GROUP BY Artist\_id ) NATURAL JOIN Artist

ORDER BY c1 DESC

LIMIT 10

D. List the names of 10 groups with the most releases.

SELECT Artist.name

FROM ( TODO )

ORDER BY c1 DESC

LIMIT 10

E. Print the name of a female artist associated with the most genres.

SELECT Artist\_name

FROM ( SELECT Artist\_id, COUNT(Genre\_id) AS G

FROM ( SELECT \*

FROM ( SELECT \*

FROM Artist

WHERE Gender\_id=1 ) NATURAL JOIN is\_genre)

GROUP BY Artist\_id )

ORDER BY G DESC

LIMIT 1

**2.5 Screenshots**

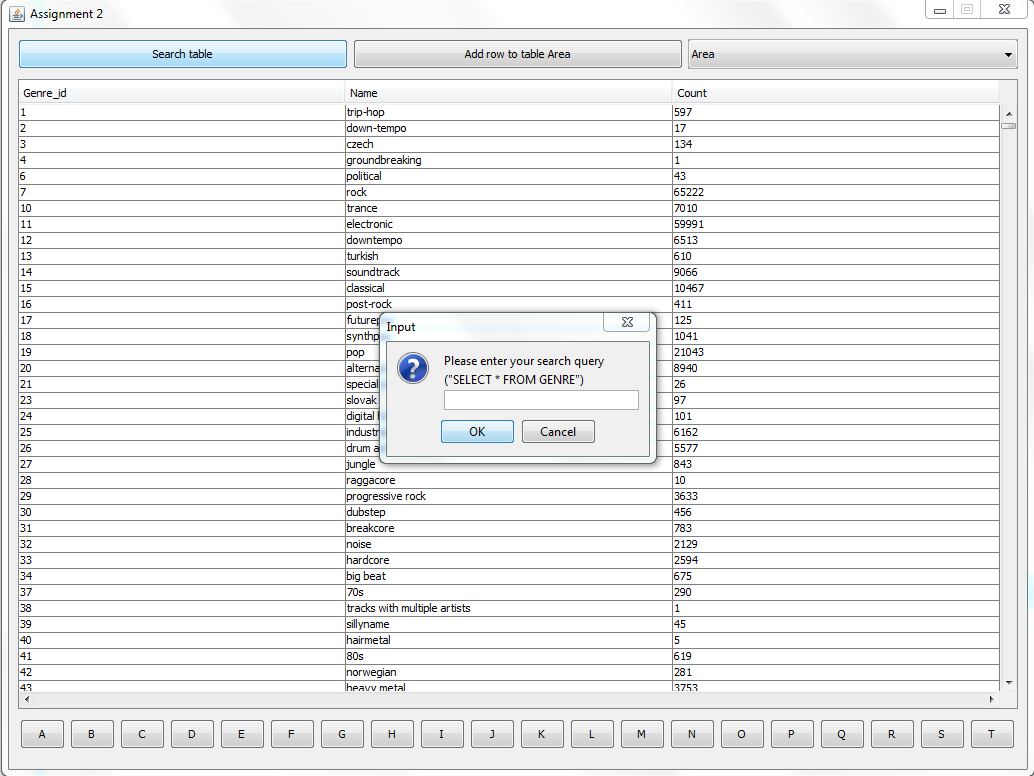
Here are a few screenshots of our interface. We chose to make a Java solution.

Figure 1 Search Functionality

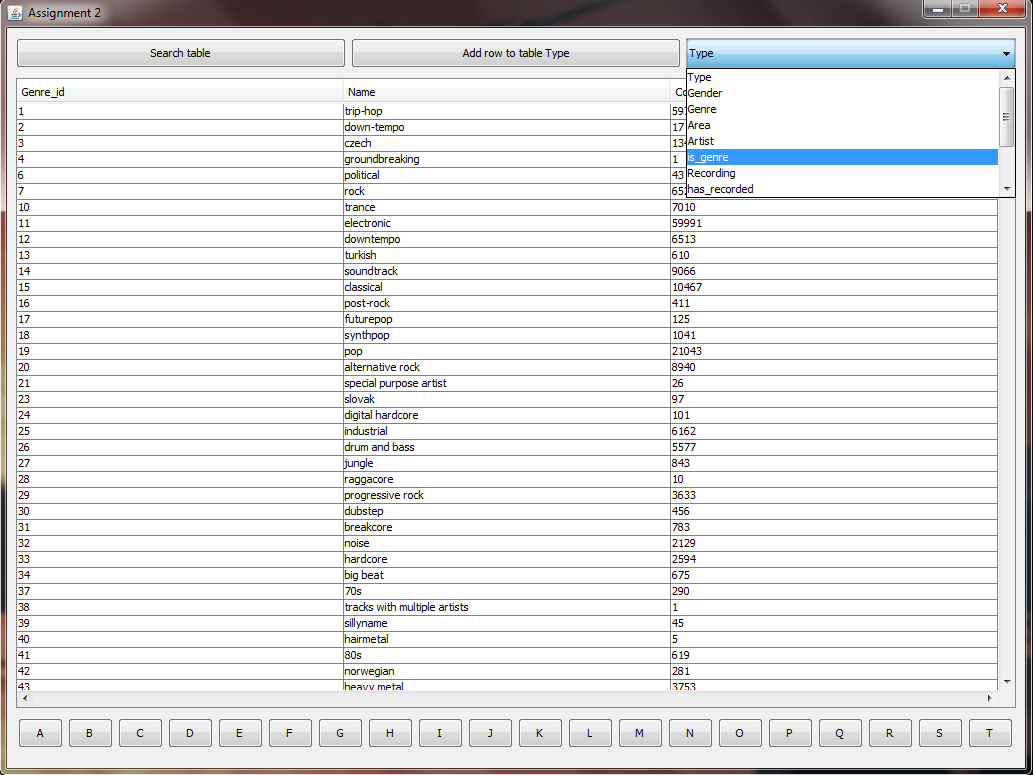


Figure 2 Tables View

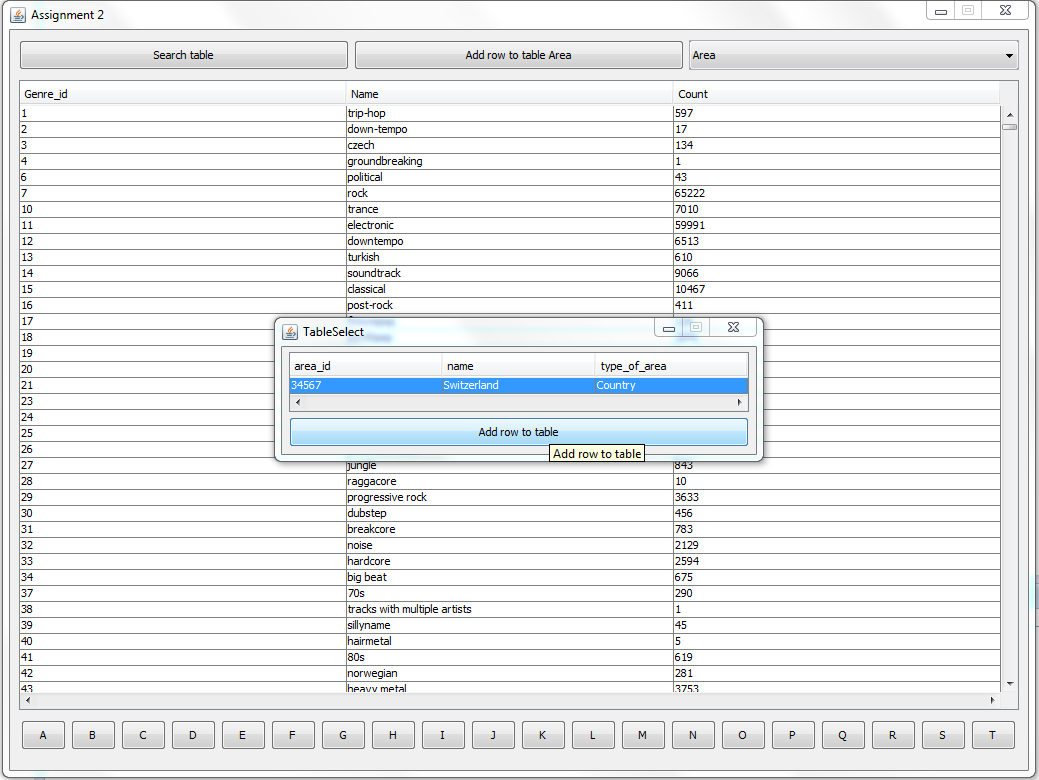
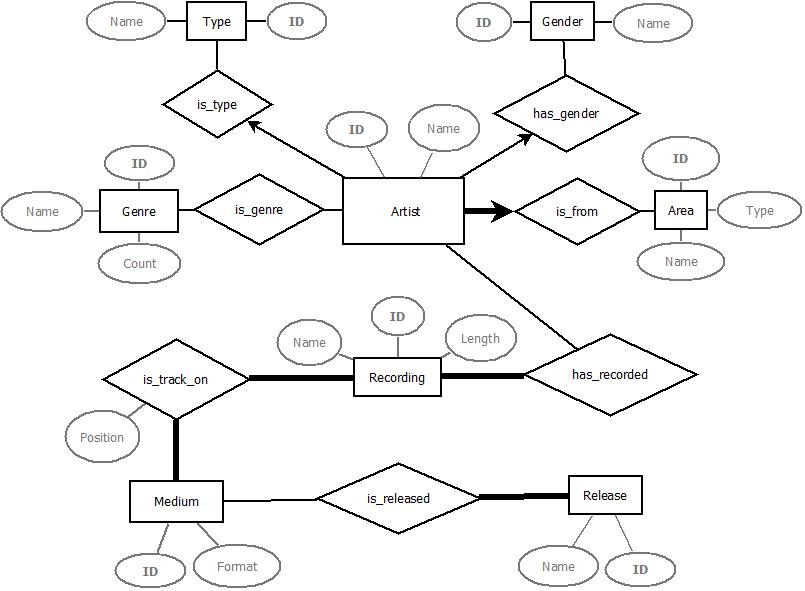


Figure 3 Data Entry Functionality

**3 Deliverable 3**

**3.1 ER Diagram**

**3.2 Design Choices**

From the feedbacks of the deliverable 2,

**3.3 SQL DDL Code for table creation**

Identical to Deliverable 2.

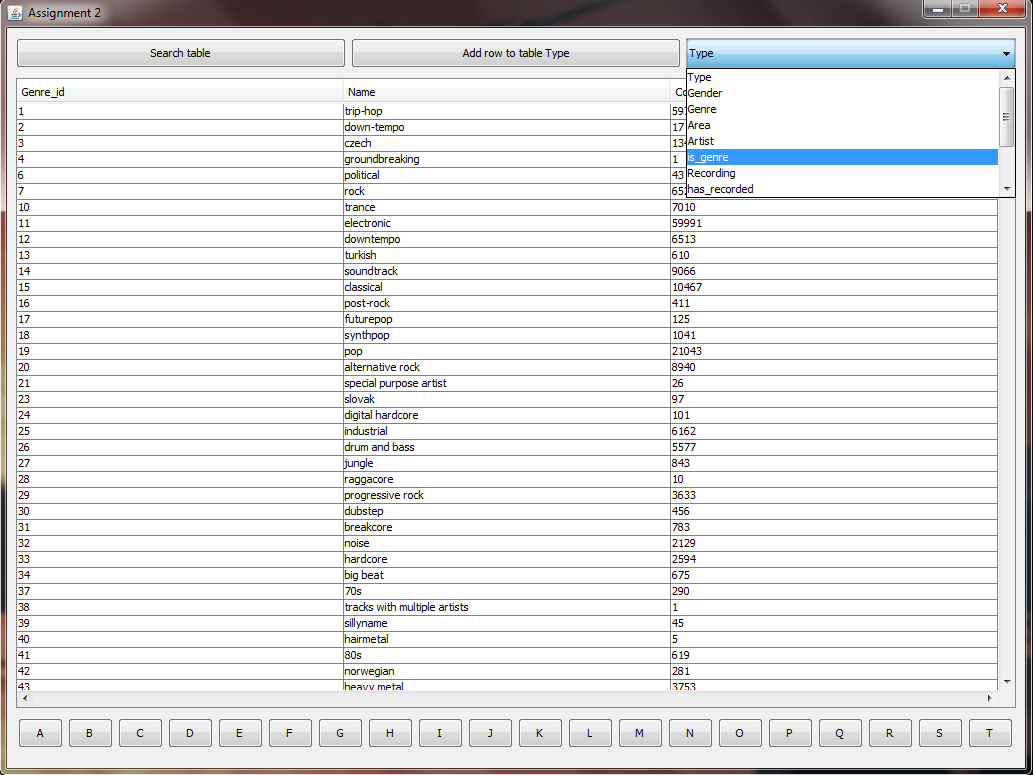
**3.4 Import of the Datas**

We created tables in which we imported the CSV files (area.csv into area\_csv table). We then performed queries on these tables to import datas into our personal tables.

TODO: add some more specific choices we had to make in order to perform correct import.

**3.4 Queries**

**3.5 Screenshots**

Here are a few screenshots of our interface. We chose to make a Java solution.

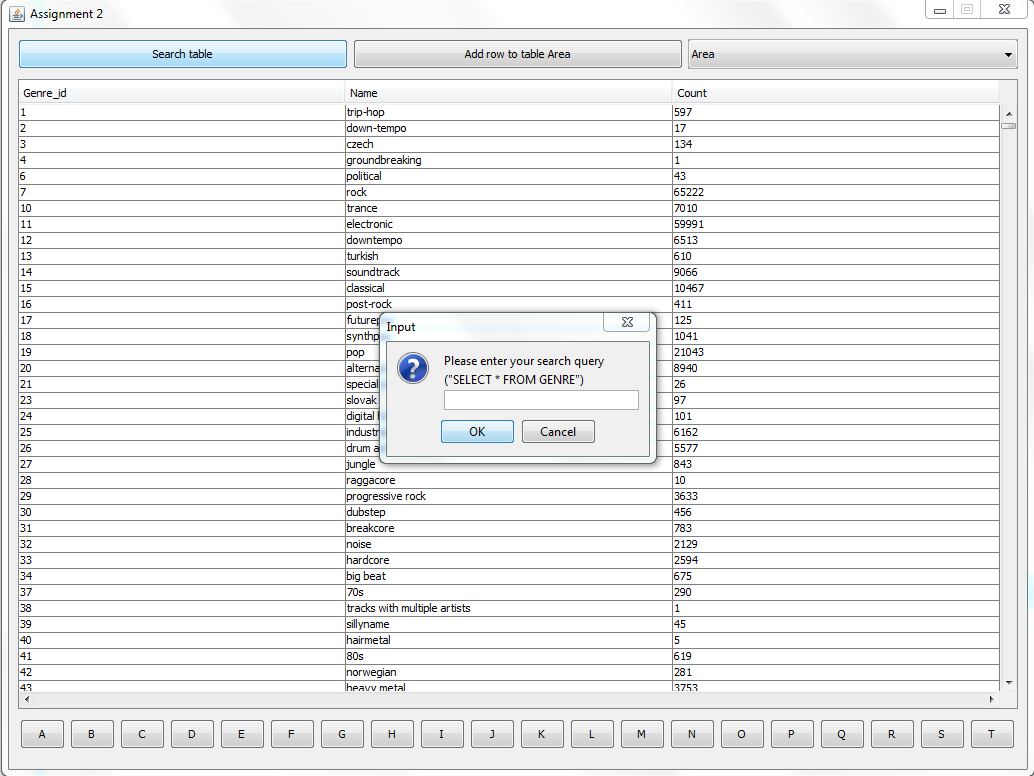


Figure 1 Search Functionality

Figure 2 Tables View

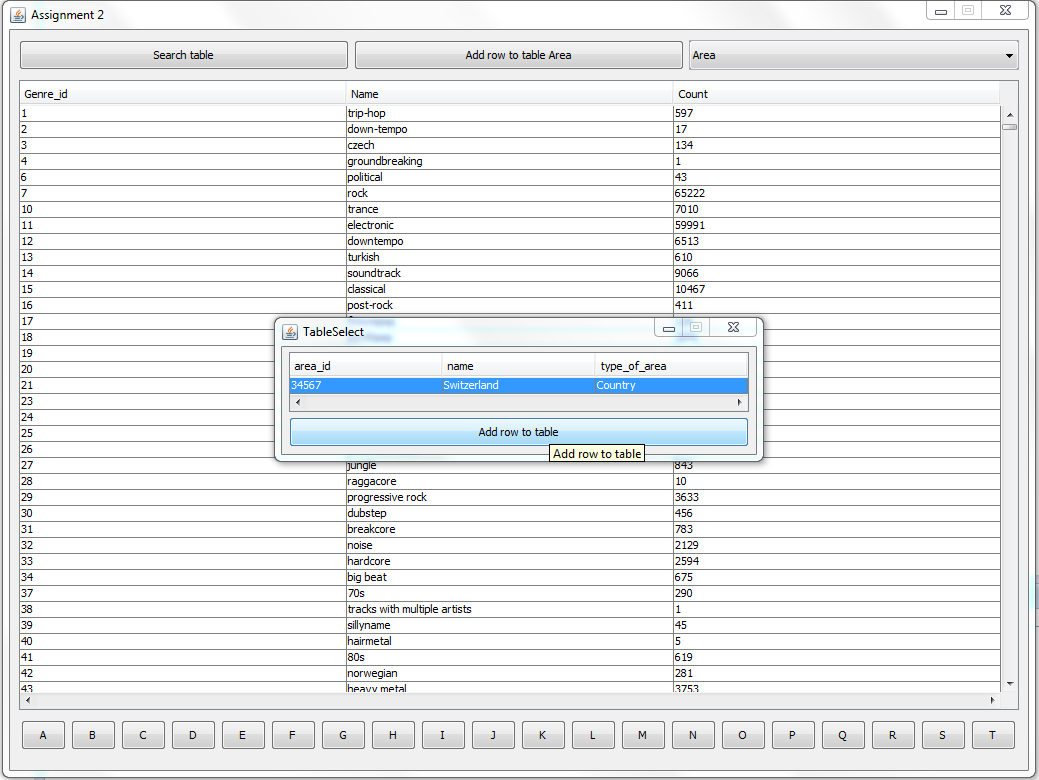


Figure 3 Data Entry Functionality

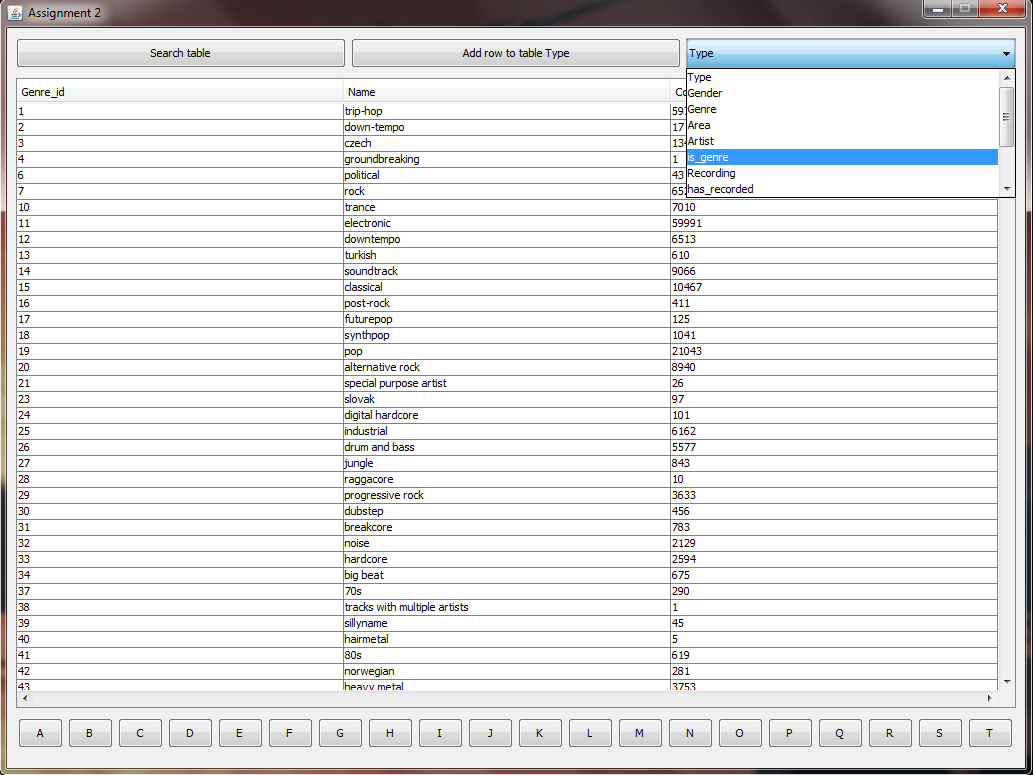


Figure 2 Tables View

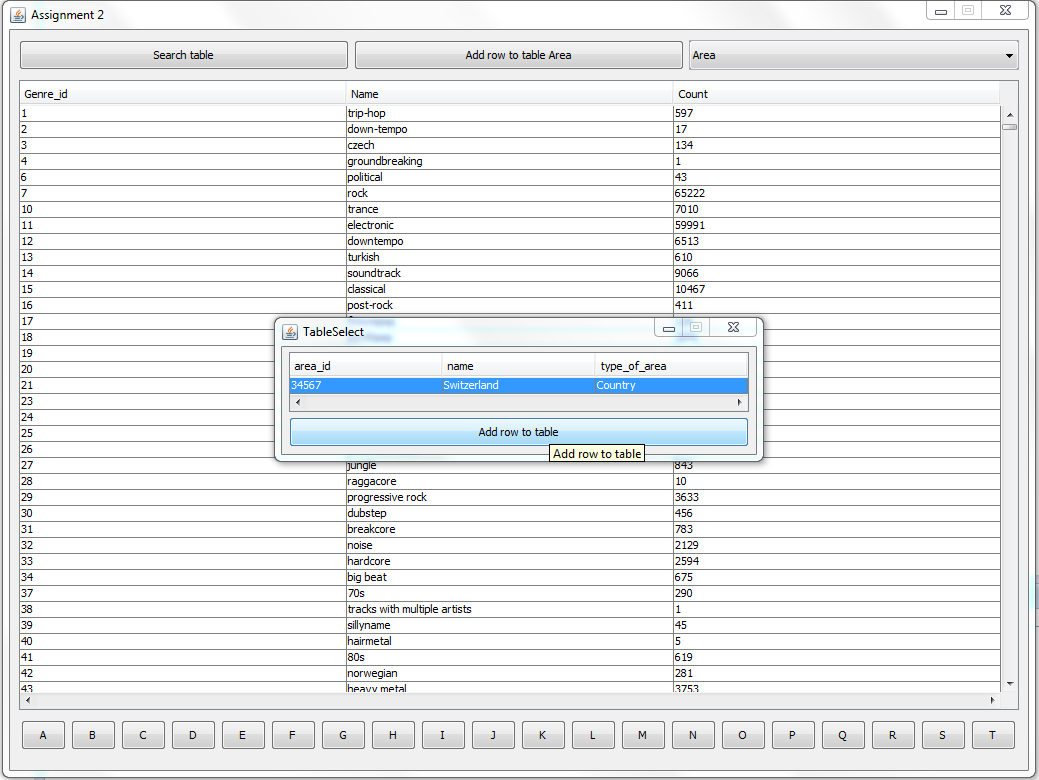


Figure 3 Data Entry Functionality