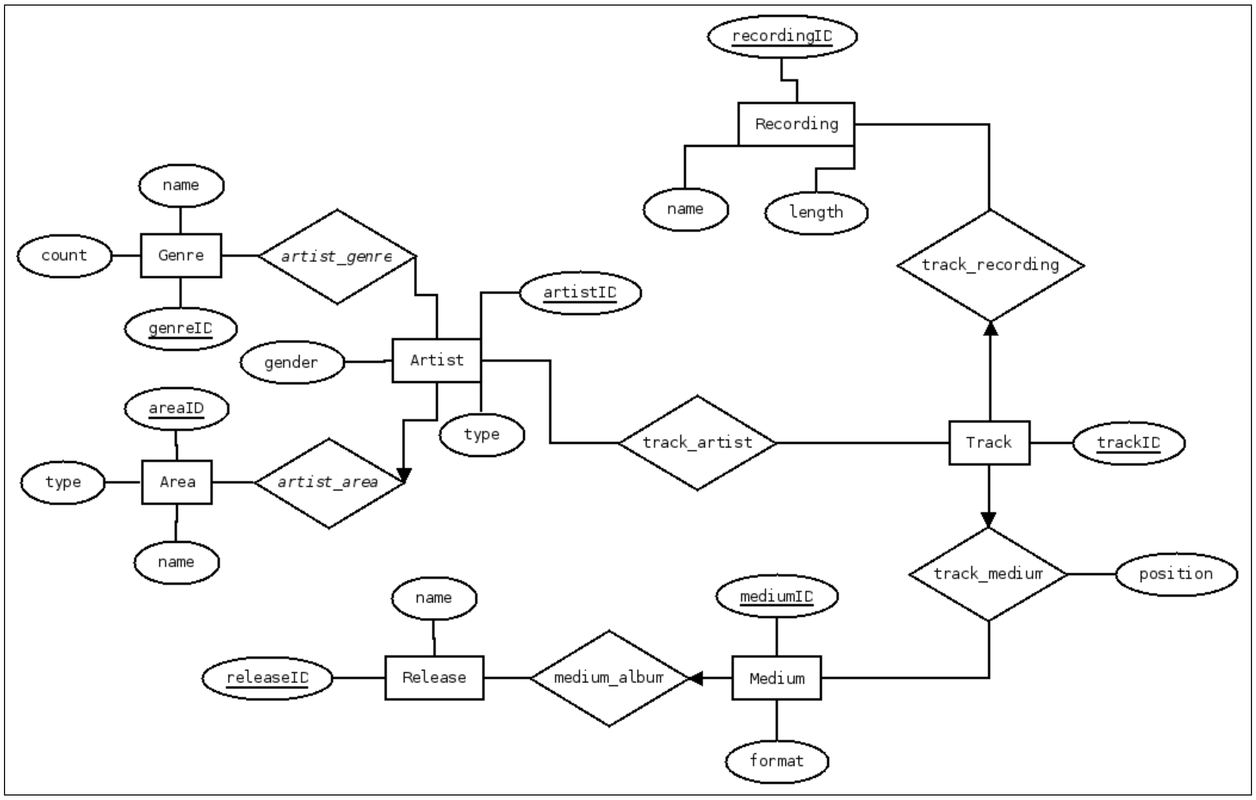
CS-322 Introduction to Database Systems
Project Deliverable #3

Due on Tuesday, June 3rd, 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(4000) NOT NULL,
  type VARCHAR(255),
  PRIMARY KEY (areaID) ) ;

CREATE TABLE Genres (
  genreID INTEGER,
  name VARCHAR(4000) NOT NULL,
  count INTEGER DEFAULT 0,
  PRIMARY KEY (genreID) ) ;

CREATE TABLE Artists (
  artistID INTEGER,
  name VARCHAR(4000) NOT NULL,
  type VARCHAR(255),
  gender VARCHAR(255),
  areaID INTEGER,
  PRIMARY KEY (artistID),
  FOREIGN KEY (areaID) REFERENCES Areas ) ;


CREATE TABLE Recordings (
  recordingID INTEGER,
  name VARCHAR(4000) ,
  length INTEGER,
  PRIMARY KEY (recordingID) ) ;



CREATE TABLE Releases (
  releaseID INTEGER,
  name VARCHAR(4000) NOT NULL,
  PRIMARY KEY (releaseID) ) ;

CREATE TABLE Mediums (
  mediumID INTEGER,
  releaseID INTEGER,
  format VARCHAR(255),
  PRIMARY KEY (mediumID),
  FOREIGN KEY (releaseID) REFERENCES Releases ) ;


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

SQL script for entities table creation

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


CREATE TABLE Track_artist (
  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 (**Recording, Release**) and their physical incarnation (**Track, Medium**). Since data is often incomplete, most of the entities 'can be related' but do not have to. We put a NOT NULL constraint on most of the name attributes of the entities, with the exception of **Recording** 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 **Track** is related to:

  Recording: A track can be a physical incarnation of a known recording.

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

  Medium: A track can be recorded on some 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, 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.

- A **Release** is the logical aggregation of songs, labeled by a title, and can be recorded on multiple mediums. Conversely, a medium identifies a singular recording of an album, enforced by a foreign key constraint.

The integrity of the count attribute in **Genre**, are not guaranteed by the table creation. It will later be enforced later on by the import and delete data commands.

Design changes from deliverable 1

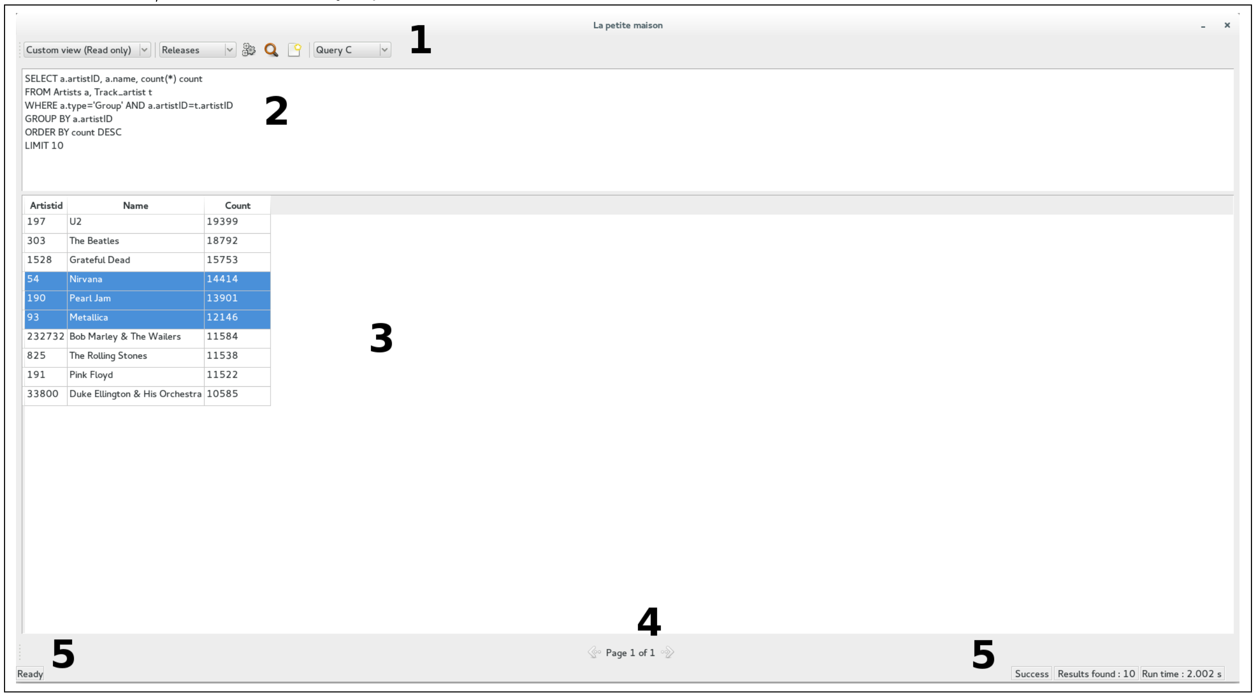
We removed all of our 'at least one' constraints to facilitate the import of new data. For the same reason, we also changed our model to be closer to the given data, so that one can easily add any information into the database and not just track-related information like we initially had in mind. That is, in our first model, everything had to be related to a track to be relevant but now every entities are independent.

Data import

After having quite a lot of trouble trying to import the data on the oracle server, we decided to use a local database instead. We used SQLite to handle this database.

Interface

We chose to use Python to design the software with the SQLAlchemy API to interface with the database. Therefore we used PyQt5 to design the GUI itself providing clean and practical tools. SQLAlchemy offers a very useful model as it reflects accurately the actual database schema and makes it easier to map the data to the Model/View offered by Qt.



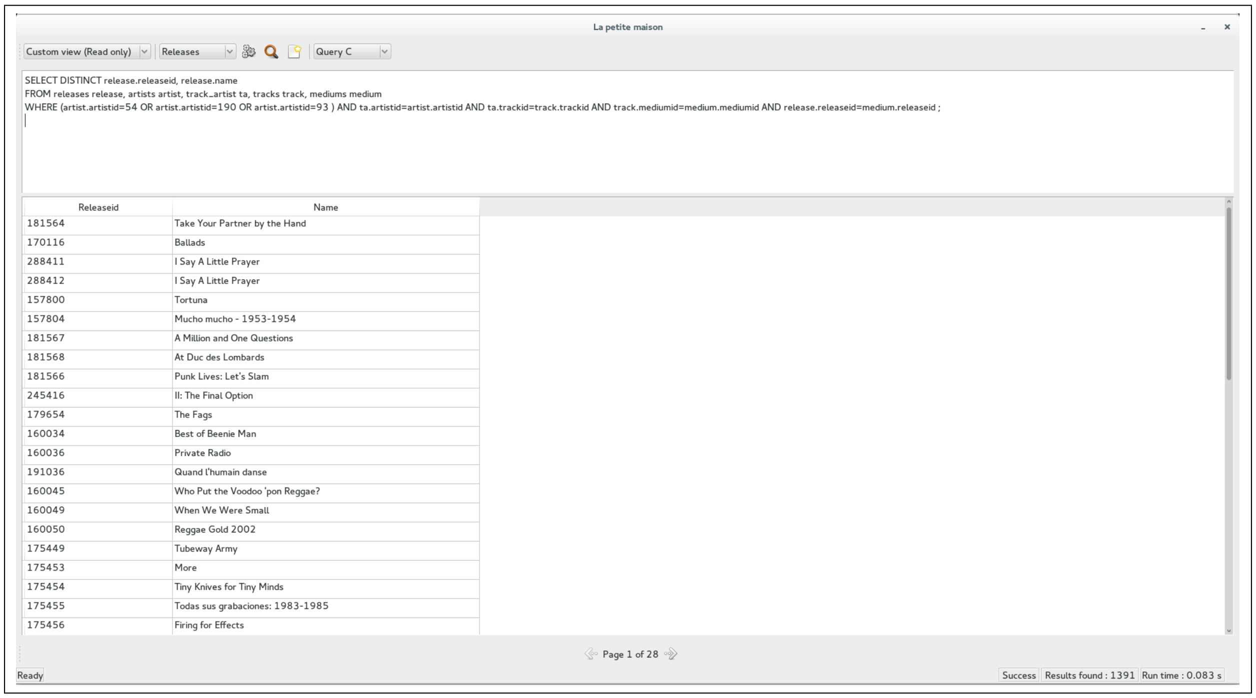
```
SELECT a.artistID, a.name, count(*) count
FROM Artists a, Track_artist t
WHERE a.type='Group' AND a.artistID=t.artistID
GROUP BY a.artistID
ORDER BY count DESC
LIMIT 10
```

Artistid	Name	Count
197	U2	19399
303	The Beatles	18792
1528	Grateful Dead	15753
54	Nirvana	14414
190	Pearl Jam	13901
93	Metallica	12146
232732	Bob Marley & The Wailers	11584
825	The Rolling Stones	11538
191	Pink Floyd	11522
33800	Duke Ellington & His Orchestra	10585

Our interface allows the user to write their own queries, while providing them with friendly ways to build simple queries to navigate through the database, to edit or delete records and to add new ones. The above picture shows our interface, after loading and running query C. The permissions and access to the different elements of the functionnality is handled through modes. Though we did not have the time to do it, this could easily be associated with user permissions. We have four different modes, which are a cross product between table view (tables' columns as in database) or custom view, and read/write or write only.

- 1 : The upper tool bar offers most of the available functions to create and run queries. From left to right we have the mode selection, the table selection, the buttons to run the current query, to search for keywords in the current results and to add a new record to the current table (only in table view) and lastly the list of the loadable existing queries. A pending query can be canceled by clicking a second time on the run button.
- 2 : The query is displayed in a text box. It can only be edited in custom mode.
- 3 : The results of the last query are shown in a table.
- 4 : The navigation tool bar allows user to switch between result pages.
- 5 : The status bar gives feedback about the queries' runs. On the left, the status (ready or query pending) is given. On the right, if the last request was successful, how many results it found and its run time.

Right click on the table provides a contextual menu where the user can choose to edit or delete the selected entries (in view mode) or to make some follow up queries depending on the selected field. So for instance if we select Nirvana, Pearl Jam and Metallica in our table and right click, we can then show all releases featuring those artists.



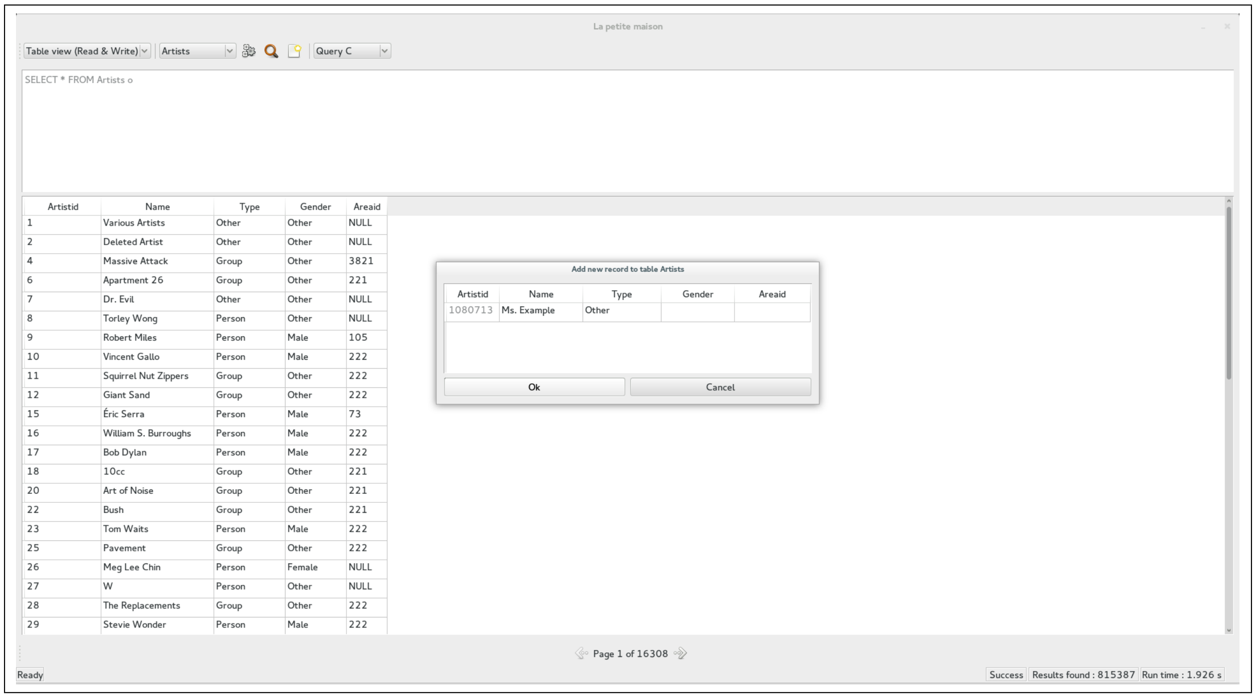
```
SELECT DISTINCT release.releaseid, release.name
FROM releases release, artists artist, track_artist ta, tracks track, mediums medium
WHERE (artist.artistid=54 OR artist.artistid=190 OR artist.artistid=93 ) AND ta.artistid=artist.artistid AND ta.trackid=track.trackid AND track.mediumid=medium.mediumid AND release.releaseid=medium.releaseid ;
```

Releaseid	Name
181564	Take Your Partner by the Hand
170116	Ballads
288411	I Say A Little Prayer
288412	I Say A Little Prayer
157800	Tortuna
157804	Mucho mucho - 1953-1954
181567	A Million and One Questions
181568	At Duc des Lombards
181566	Punk Lives: Let's Slam
245416	II: The Final Option
179654	The Fags
160034	Best of Beenie Man
160036	Private Radio
191036	Quand l'humain danse
160045	Who Put the Voodoo 'pon Reggae?
160049	When We Were Small
160050	Reggae Gold 2002
175449	Tubeway Army
175453	More
175454	Tiny Knives for Tiny Minds
175455	Todas sus grabaciones: 1983-1985
175456	Firing for Effects

Page 1 of 28

Ready Success Results found : 1391 Run time : 0.083 s

The update, search and insert queries are handled by pop up dialogs, providing tables to be filled which are then turned into queries. For instance, we can add a new artist.



SELECT * FROM Artists o

Artistid	Name	Type	Gender	Areaid
1	Various Artists	Other	Other	NULL
2	Deleted Artist	Other	Other	NULL
4	Massive Attack	Group	Other	3821
6	Apartment 26	Group	Other	221
7	Dr. Evil	Other	Other	NULL
8	Torley Wong	Person	Other	NULL
9	Robert Miles	Person	Male	105
10	Vincent Gallo	Person	Male	222
11	Squirrel Nut Zippers	Group	Other	222
12	Giant Sand	Group	Other	222
15	Éric Serra	Person	Male	73
16	William S. Burroughs	Person	Male	222
17	Bob Dylan	Person	Male	222
18	10cc	Group	Other	221
20	Art of Noise	Group	Other	221
22	Bush	Group	Other	221
23	Tom Waits	Person	Male	222
25	Pavement	Group	Other	222
26	Meg Lee Chin	Person	Female	NULL
27	W	Person	Other	NULL
28	The Replacements	Group	Other	222
29	Stevie Wonder	Person	Male	222

Add new record to table Artists

Artistid	Name	Type	Gender	Areaid
1080713	Ms. Example	Other		

Page 1 of 16308

Ready Success Results found : 815387 Run time : 1.926 s

Queries & performance analysis

SQL Queries

```sql
-- 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    arti.name
FROM  Artists arti, Areas area
WHERE arti.areaID=area.areaID AND area.name LIKE 'Switzerland'
```

SQL script for query A

```sql
-- 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.

-- counts the number of male, female artists and groups in all areas
WITH counts AS (
SELECT a.areaid,
 count (CASE WHEN a.gender = 'Female' AND a.type <> 'Group' THEN 1 END) AS femnum,
 count (CASE WHEN a.gender = 'Male' AND a.type <> 'Group' THEN 1 END) AS malnum,
 count (CASE WHEN a.type = 'Group' THEN 1 END) AS grpnum
FROM Artists a
GROUP BY a.areaid )

-- Area with the most female artists
SELECT 'Female' AS areaType, l.*, max ( counts.femnum ) , counts.malnum, counts.grpnum
FROM counts, Areas l
WHERE counts.areaid = l.areaid

UNION

-- Area with the most male artists
SELECT 'Male' AS areaType, l.*, counts.femnum , max ( counts.malnum ), counts.grpnum
FROM counts, Areas l
WHERE counts.areaid = l.areaid

UNION

-- Area with the most groups
SELECT 'Group'AS areaType, l.*, counts.femnum , counts.malnum , max ( counts.grpnum )
FROM counts, Areas l
WHERE counts.areaid = l.areaid
```

SQL script for query B

```sql
-- List the names of 10 groups with the most recorded tracks.

SELECT a.name
```

```
FROM Artists a, Track_artist t
WHERE a.type='Group' AND a.artistID=t.artistID
GROUP BY a.artistID
ORDER BY count(*) DESC
LIMIT 10
```

SQL script for query C

```
-- List the names of 10 groups with the most releases.

SELECT a.name
FROM Artists a, Track_artist ta, Tracks t, Mediums m
WHERE a.type="Group" AND a.artistID=ta.artistID
AND t.trackID = ta.trackID AND m.mediumID=t.mediumID
GROUP BY a.artistID
ORDER BY count(DISTINCT m.releaseID) DESC
LIMIT 10
```

SQL script for query D

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

SELECT name
FROM (
SELECT name, MAX(c)
FROM (
SELECT name, a.artistID, COUNT(g.genreID) c
FROM Artists a, Artist_genre g
WHERE a.gender='Female' AND a.type <> 'Group' AND a.artistid=g.artistID
GROUP BY a.artistID )
)
```

SQL script for query E

```
-- List all cities which have more female than male artists.

SELECT name FROM (
SELECT l.name,
COUNT(CASE WHEN a.gender = 'Female' THEN 1
     WHEN a.gender = 'Male' THEN -1 END) maj
FROM Areas l, Artists a
WHERE l.areaID=a.areaID AND a.type <> 'Group' AND l.type='City'
GROUP BY l.areaID
) WHERE maj>0 -- majority of female artists
```

SQL script for query F

```
-- List the mediums with the highest number of tracks.

-- number of tracks per medium
WITH numtrack AS (
SELECT mediumid, COUNT(*) AS c
FROM Tracks
GROUP BY mediumid ),

-- maximum number of tracks in a medium
max AS ( SELECT MAX(c) m FROM numtrack )

-- select all medium with the max number of tracks
SELECT m.*
FROM Mediums m, numtrack, max
WHERE m.mediumid = numtrack.mediumid AND c = max.m
```

SQL script for query G

```
-- For each area that hAS more than 30 artists, list the male artist,
-- the female artist and the group with the most tracks recorded.

-- areas with at least 30 artists
WITH loc AS (
SELECT areaid
FROM artists
GROUP BY areaid
having COUNT(*)>30 )

-- female artist in area with most tracks
SELECT *, MAX ( numtrack ) FROM (
SELECT a.*, COUNT(distinct trackid) AS numtrack
FROM artists a, track_artist ta, loc
WHERE a.type<>'Group' AND a.gender='Female' AND a.areaid = loc.areaid
AND a.artistid=ta.artistid
GROUP BY a.artistid )
GROUP BY areaid

UNION

-- male artist in area with most tracks
SELECT *, MAX ( numtrack ) FROM (
SELECT a.*, COUNT(distinct trackid) AS numtrack
FROM artists a, track_artist ta, loc
WHERE a.type<>'Group' AND a.gender='Male' AND a.areaid = loc.areaid
AND a.artistid=ta.artistid
GROUP BY a.artistid )
GROUP BY areaid

UNION

-- group in area with most tracks
SELECT *, MAX ( numtrack ) FROM (
SELECT a.*, COUNT(distinct trackid) AS numtrack
FROM artists a, track_artist ta, loc
WHERE a.type='Group' AND a.areaid AND a.areaid = loc.areaid
AND a.artistid=ta.artistid
GROUP BY a.artistid )
GROUP BY areaid
```

SQL script for query H

```sql
-- American metal group Metallica is asking its fans to choose the
-- setlist for its upcoming concert in Switzerland.
-- Assuming that the Metallica fans will choose the songs
-- that have appeared on the highest
-- number of mediums, list the top 25 songs.

SELECT R.*
FROM Track_artist ta,Tracks t, Artists a, Recordings r
WHERE a.name="Metallica" AND a.artistID=ta.artistID
AND t.trackID=ta.trackID AND t.recordingId=r.recordingId
GROUP BY t.recordingID
ORDER BY COUNT(DISTINCT t.mediumid) DESC
LIMIT 25
```

SQL script for query I

```sql
-- For each of the 10 genres with the most artists, list the most popular female artist.
-- Most popular <=> with the most tracks recorded

Select ArtistId, "NAME", AreaId Gender ,"TYPE", GenreId
  from (
  -- Seqnum is used to keep only ONE artist per genre
  Select r.* , row_number() over (partition by genreId order by artistId) as seqnum
  from (
  -- Max counter is used to keep the artists with the most tracks for every genreId
    Select t.* , max(counter) over (partition by genreId) as maxcounter
    from (
    -- Return , for each selected genre, all artists with their number of recorded tracks ("counter"
         Select artilist.ArtistId, artilist."NAME", artilist.AreaId, artilist.gender,
               artilist."TYPE" , count(*) counter , genreId
         from Track_Artist
         INNER JOIN (
          Select Artists.*  , genreid
          from Artists
          INNER JOIN (
               Select ArtistId, genreids.genreid
               from Artist_Genre
               INNER JOIN (
                   Select GenreId
                   from (
                        Select *
                        From (
                          Select GenreId, count(*) counter
                          from Artist_GENRE
                          GROUP BY GenreId
                          ORDER BY counter DESC  )
                        LIMIT 10)
               ) genreids
               ON genreids.genreid = Artist_Genre.genreid ) artigenre
          ON artigenre.artistId = Artists.ArtistId
          where Artists.gender = 'Female' ) artilist
```

```
        ON Track_Artist.artistid = artilist.artistid

        GROUP BY  artilist.ArtistId, artilist."NAME", artilist.AreaId, artilist.gender,
              artilist."TYPE" , genreid
        ORDER BY genreid, counter DESC ) t
    ) r
    where counter = maxcounter
  )
where seqnum = 1
```

SQL script for query J

```
-- List all genre with no female artist, all genre that
-- have no males artists and all genres that have no groups
--- <=> List all genre with no female artist OR no male artists OR no groups (?)

SELECT  g.*
FROM Genres g, Artist_Genre ag, artists a
WHERE g.genreid = ag.genreid AND ag.artistid=a.artistid
GROUP BY g.GenreId, g.name
HAVING (count(CASE WHEN a.gender = 'Female' THEN 1 END)=0)
OR (count(CASE WHEN a.gender = 'Male' THEN 1 END)=0)
OR (count(CASE WHEN a.type = 'Group' THEN 1 END)=0)
```

SQL script for query K

```
-- For each area with more than 10 groups, list the 5 male artists
-- that have recorded the highest number of tracks.

-- areas with more than 10 groups
WITH gareAS AS (
SELECT areaid
FROM artists
WHERE type='Group'
GROUP BY areaid
having count(*) > 10 )

-- select artist with max number of tracks
SELECT a.*, MAX ( numtrack ) AS numtrack
FROM (
-- counts tracks for male artists
SELECT a.artistid, count ( ta.trackid ) AS numtrack
FROM gareas, artists a, track_artist ta
WHERE gareas.areaid = a.areaid AND a.type <> 'Group'
     AND a.gender = 'Male' AND a.artistid = ta.artistid
GROUP BY a.artistid
 ) AS m, artists a
WHERE m.artistid = a.artistid
GROUP BY areaid
```

SQL script for query L

```
-- List the 10 groups with the highest number of tracks that appear
-- on compilations. A compilation is a medium that contains tracks
-- associated with more than one artist.

Select art.*
FROM Artists art
INNER JOIN  (
  Select ArtistId, count(*) counter
  From Track_Artist tracky
  INNER JOIN (
    Select tr.trackId
    From Tracks tr
    INNER JOIN (
      Select DISTINCT tr.mediumId mediId
      From Tracks tr
      INNER JOIN (
        Select TrackId
        From(
          SELECT TrackId, count(*) artistnumber
          FROM Track_Artist
          GROUP BY TrackId)
        where artistnumber > 1) trid
      ON trid.trackId = tr.trackId) medi
    ON medi.mediId = tr.mediumId) compilId
  ON tracky.trackId = compilId.trackId
  Group By ArtistId
  ORDER BY counter DESC) arti
ON art.artistId = arti.artistId
WHERE art."TYPE" = 'Group'
LIMIT 10
```

SQL script for query M

```
-- List the top 10 releases with the most collaborations, i.e.,
-- releases where one artist is performing all songs and the highest
-- number of different guest artists contribute to the album.

-- for each release with at least 2 artists, counts number of
-- contributing artists and the number of tracks
WITH releaseinfo AS (
SELECT  m.releaseid,
COUNT ( DISTINCT ( ta.artistid ) )  AS numart,
COUNT ( DISTINCT (t.trackid) ) AS numtrack
FROM track_artist ta, tracks t, mediums m
WHERE ta.trackid = t.trackid AND t.mediumid=m.mediumid
GROUP BY m.releaseid
HAVING numart>1
)

-- displays top 10 of collaborations

SELECT r.*, numart FROM (
-- lists compilations by countTing tracks per artist for each release
-- if an artist contributes to all tracks, then it is a compilation
-- they are then ordered by the number of contributors
SELECT DISTINCT m.releaseid, numart
FROM releaseinfo, mediums m, tracks t, track_artist ta
WHERE  m.releaseid = releaseinfo.releaseid
     AND t.mediumid = m.mediumid AND ta.trackid=t.trackid
GROUP BY m.releaseid, ta.artistid
HAVING COUNT ( ta.trackid ) = numtrack
ORDER BY numart DESC
LIMIT 10 ) AS top, releases r

WHERE r.releaseid = top.releaseid
```

SQL script for query N

```
-- List the release which is associated with the most mediums.
-- If there are more than one such release, list all such releases.
-- Problem with max in oracle, need to be combined with a GROUP BY

select r.*, count(distinct m.mediumid) as nummedium
from releases r, mediums m
where r.releaseid=m.releaseid
group by r.releaseid
having nummedium = (
select count(distinct mediumid) c
from mediums
group by releaseid
order by count(distinct mediumid) DESC
LIMIT 1 )
```

SQL script for query O

```
-- List the most popular genre among the groups wich are associated with at least 3 genres
Select Genres.genreId, Genres.name
from Genres
     Inner Join (
          Select GenreId, count(*) genrecounter
          from Artist_Genre artigenre
          Inner Join (
               Select ArtistId
               From (
                    Select arti.ArtistId, count(*) numbgenre
                    from Artist_genre
                    INNER JOIN (
                         Select artistId
                         from Artists
                         where type = 'Group') arti
                    on Artist_genre.artistId = arti.artistId
                    GROUP BY arti.ArtistId) artistgenrecount
               where numbgenre > 2 ) artithree
          ON artithree.artistId = artigenre.artistId
          Group By GenreId
          Order by genrecounter Desc) genreordered
     On genreordered.genreid = Genres.genreId
LIMIT 1
```

SQL script for query P

```
-- List the 5 titles that are associated with the most different songs
-- (recordings) along with the number of songs that share such title.

select name, count(*) numsongs
from recordings
group by name
order by numsongs DESC
limit 5
```

SQL script for query Q

```
-- List the top 10 artists according to their track-to-release ratio.
-- This ratio is computed by dividing the number of tracks an artist is associated
-- with by the number of releases this artist has contributed a track to.

select a.*, count(distinct t.trackid)/count(distinct r.releaseid) AS ratio
from artists a, track_artist ta, tracks t, mediums m, releases r
where a.artistid=ta.artistid AND ta.trackid=t.trackid
     AND t.mediumid=m.mediumid AND m.releaseid=r.releaseid
group by a.artistid
order by ratio DESC
LIMIT 10
```

SQL script for query R

```
-- The concert hit index is a measure of probability that the artist
-- can attract enough fans to fill a football
-- stadium. We define the âĂIJhit artistâĂİ as one that has more than 10 songs
-- that appear on more than 100
-- mediums and measure "hit ability" as the average number of mediums that
-- a top 10 song appears on.
-- List all âĂIJhit artistsâĂİ according to their "hit ability".

with hitsongs as (
select a.artistid, t.recordingid, count ( distinct t.mediumid ) as c
from artists a, track_artist ta, tracks t
where a.artistid=ta.artistid AND ta.trackid=t.trackid
group by a.artistid, t.recordingid
having c > 100  )
select a.*
from  artists a,
( select artistid
from hitsongs
group by artistid
having count(distinct recordingid) > 10 ) AS  hitartist ,
( select artistid, AVG(c) as ha
from hitsongs
```

```sql
group by artistid
order by c
limit 10 ) as hitability
where a.artistid = hitartist.artistid AND a.artistid = hitability.artistid
```

SQL script for query S

Run Time

We display here the run times in seconds for each query, as given to us by sqlite.

query	real	user	sys
A	0.310	0.270	0.030
B	3.136	2.635	0.490
C	1.984	1.848	0.134
D	31.450	18.799	5.163
E	0.409	0.144	0.094
F	0.110	0.097	0.146
G	13.550	12.435	0.270
H	18.063	17.701	0.337
I	0.054	0.051	0.003
J	1.355	1.039	0.097
K	1.029	0.434	0.104
L	8.023	6.756	0.551
M	61.44	54.99	6.275
N	73.316	58.726	13.975
O	10.467	9.657	0.189
P	0.220	0.209	0.004
Q	7.535	7.234	0.287
R	42.541	32.390	9.240
S	158.359	140.138	17.835

On the necessity of indexes

We chosed queries I, K and M for our analysis.