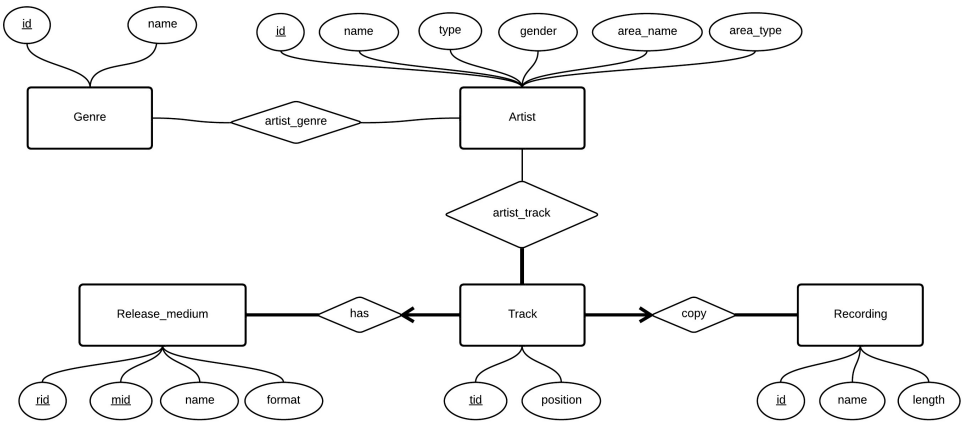
Project Report

Group 18: Chengzhen Wu, Huiqi Mao, Ruofan Zhou

May 26, 2014

1 ER model

After reading the feedback of diverable1, we modified our ER model as below:



In the given project data, we firstly recognize 'area', 'artist' and 'genre' each as three individual entities. Each artist is from at most one area, so it's a many-to-one relation. Several artists can belong to different genres and one genre can contain several artists. So the relation between 'artist' and 'genre' is many-to-many.

Secondly, we think about the relationship among 'release', 'recording', track' and 'medium'. We imagine a scene to describe these relations. The csv file of 'release' contains the names of releases. They could be stored in the mediums, such as CD, 12" Vinyl and so on. What's more, one release could have several CDs to contain many tracks, or in different medium (I'm not sure about this, but possible). So the relation between 'release' and 'medium' is one-to-many. Next, each track in different mediums must correspond to one recording. So the relation between 'track' and 'recording' is many-to-one. Each 'track' must be in one of 'medium's. So the relation is many-to-one.

Finally, we merge one-to-may relations. 'release' and 'medium' are merged into 'Release_medium'. 'area' can be merged into 'artist' as attributes. The 'has' relation between 'Release_medium' and 'track' is merged into 'track' using 'mid' as foreign key. So is the 'copy' relation between 'recording' and 'track' using 'id' of recording as foreign key.

Additionally, we ignore the 'count' in 'genre', which could be created as view in the database.

2 SQL based on ER model

```
--ENTITY Genre
--we don't need gcount in genre
CREATE TABLE  Genre (
  GID INTEGER,
  Gname VARCHAR(100),
  PRIMARY KEY (GID)
  );

--ENTITY Artist
CREATE TABLE  Artist (
  AID INTEGER,
  Aname VARCHAR(100),
  Atype CHAR(6),
  gender CHAR(6),
  area_name VARCHAR(60),
  area_type CHAR(15)
  PRIMARY KEY (AID)
  );

--ENTITY Recording
CREATE TABLE  recording (
  RID INTEGER,
  Rname VARCHAR(100),
  Rlength INTEGER,
  PRIMARY KEY (RID)
  );

--ENTITY RelaseMedium
CREATE TABLE  ReleaseMedium (
  MID INTEGER,
  RID INTEGER,
  name VARCHAR(400),
  format CHAR(45),
  PRIMARY KEY (RID,MID)
  );

--ENTITY Track
CREATE TABLE  Track (
  TID INTEGER,
  position INTEGER,
  MID INTEGER NOT NULL,
  RID INTEGER NOT NULL,
  REID INTEGER NOT NULL,
  PRIMARY KEY (TID),
  FOREIGN KEY (RID,MID) REFERENCES ReleaseMediuma ON DELETE NO ACTION,
  FOREIGN KEY (REID) REFERENCES Recording ON DELETE NO ACTION
  );

--RELATIONSHIP artist_genre
CREATE TABLE artist_genre (
  AID INTEGER NOT NULL,
  GID INTEGER NOT NULL,
  PRIMARY KEY (AID, GID),
  FOREIGN KEY (AID) REFERENCES Artist ON DELETE NO ACTION ,
  FOREIGN KEY (GID) REFERENCES Genre ON DELETE NO ACTION
  );

--RELATIONSHIP artist_track
CREATE TABLE artist_track(
  AID INTEGER NOT NULL,
  TID INTEGER NOT NULL,
```

```
  PRIMARY KEY (AID, TID),
  FOREIGN KEY (AID) REFERENCES Artist ON DELETE NO ACTION ,
  FOREIGN KEY (TID) REFERENCES Track ON DELETE NO ACTION
  );
```

3 Queries

We finished the queries based on our model:

A

```
select artist.NAME
  from ARTIST artist
  where artist.area_name='Switzerland';
```

B

```
(select area_male.gender as type,
    area_male.area_name,area_male.sum as Number_of_Artist
  from (
    select artist.gender,artist.AREA_NAME,count(*)as sum
      from ARTIST artist
      where artist.GENDER='Male' and artist.AREA_NAME <> 'null'
      GROUP BY artist.AREA_NAME, artist.gender
      order by count(*) desc) area_male
  where rownum = 1)
union
(select area_female.gender as type,
    area_female.area_name,area_female.sum as Number_of_Artist
  from (
    select artist.gender,artist.AREA_NAME,count(*)as sum
      from ARTIST artist
      where artist.GENDER='Female' and artist.AREA_NAME <> 'null'
      GROUP BY artist.AREA_NAME, artist.gender
      order by count(*) desc)area_female
  where rownum = 1)
union
(select area_group.type as type,
    area_group.area_name,area_group.sum as Number_of_Artist
  from (
    select artist.type,artist.AREA_NAME,count(*)as sum
      from ARTIST artist
      where artist.TYPE= 'Group' and artist.AREA_NAME <> 'null'
      GROUP BY artist.AREA_NAME, artist.gender, artist.type
      order by count(*) desc)area_group
```

```
  where rownum = 1);
```

C

```
select artist.name
  from (select artist.NAME, count(*)
    from artist artist, artist_track A_T
    where artist.type='Group' and A_T.aid=artist.id
    group by artist.name order by count(*) desc) artist
  where Rownum <= 10;
```

D

```
select artist2.name
  from (
    select artist.id, artist.name
    from ARTIST artist
      join ARTIST_TRACK art_track on artist.ID = art_track.AID
      join TRACK track on art_track.TID = track.TID
      join RELEASEMEDIUM release_medium
          on track.MID = release_medium.MID
      where artist.type = 'Group'
      group by artist.id, artist.name
      order by count(*) desc) info, artist artist2
  WHERE rownum <= 10 and artist2.id = info.id;
```

E

```
select artist.name
--project the artist name according to artist id
  from (
    select artist.id
      from artist artist, artist_genre art_genre, genre genre
      where artist.id = art_genre.aid
          and art_genre.GID = genre.ID
          and artist.gender = 'Female'
      group by artist.id
      having count(*) = (
        select max(genre_count.count)
--find the maximum number of genre of a female artist
          from (
          select count(*)as count
            from artist artist2, artist_genre art_genre2,
                genre genre2
            where artist2.id = art_genre2.aid
                and art_genre2.GID = genre2.ID
```

```
                and artist2.gender = 'Female'
            group by artist2.id) genre_count))max_count,
                artist artist
  where artist.id = max_count.ID;
```

F

```
select female_count.area_name
  from (
    select artist.area_name, count(*)as count
      from artist artist
      where artist.AREA_TYPE='City' and gender ='Male'
      group by artist.area_name, artist.gender) male_count,
      (select artist.area_name, count(*)as count
        from artist artist
        where artist.AREA_TYPE='City' and gender ='Female'
        group by artist.area_name, artist.gender) female_count
  where male_count.area_name = female_count.area_name and
      female_count.count > male_count.count;
```

G

```
create view med_track as
  select release_medium.MID,count(*) as tracks
    from ReleaseMedium release_medium,TRACK track
    where track.MID = release_medium.MID
    group by release_medium.MID order by count(*) desc;

select med_track.mid
  from  med_track
  where med_track.tracks =
      (select MAX(med_track.tracks)
                        from  med_track);
```

H

```
select lst.area_name, lst.name
  from (
  select artist.area_name,artist.name, rank() over
      (partition by artist.area_name
      order by count(*) desc) as rank
    from artist_track at1, (
      select artist.area_name,artist.id ,artist.name
        from ARTIST artist where artist.gender = 'Male') artist
        where artist.id =at1.AID and artist.area_name in (
          select a1.area_name
```

```
            from artist a1
            where a1.area_name is not null
            group by a1.area_name
            having count(*)>30)
        group by artist.area_name, artist.id,artist.name) lst
    where lst.rank = 1
union
select lst.area_name, lst.name
  from (
    select artist.area_name,artist.name, rank() over
        (partition by artist.area_name
        order by count(*) desc) as rank
    from artist_track at1, (
    select artist.area_name,artist.id ,artist.name
      from ARTIST artist where artist.gender = 'Female') artist
      where artist.id =at1.AID and artist.area_name in (
        select a1.area_name
          from artist a1
          where a1.area_name is not null
          group by a1.area_name
          having count(*)>30)
     group by artist.area_name, artist.id,artist.name) lst
  where lst.rank = 1
union
select lst.area_name, lst.name
  from (
    select artist.area_name,artist.name, rank() over
        (partition by artist.area_name
        order by count(*) desc) as rank
      from artist_track at1, (
      select artist.area_name,artist.id ,artist.name
        from ARTIST artist where artist.type = 'Group') artist
        where artist.id =at1.AID and artist.area_name in (
        select a1.area_name
          from artist a1
          where a1.area_name is not null
          group by a1.area_name
          having count(*)>30)
      group by artist.area_name, artist.id,artist.name) lst
  where lst.rank = 1;
```

I

```
select recording.name
```

```
  from RECORDING ,(
  select track.rcid, rank()
      over (order by count(distinct track.mid) desc) as rank
    from TRACK track
    where exists (
      select artrack.TID
        from ARTIST artist, ARTIST_TRACK artrack
        where artist.name = 'Metallica'
            and  artrack.aid=artist.id
            and track.tid = artrack.tid)
    group by track.rcid) toptrak
  where toptrak.rank<=25 and recording.id = toptrak.rcid;
```

J

```
select genre.NAME, artistrank.NAME
  from genre ,(
  select artistfilter.GID, artistfilter.NAME, rank()
    over (PARTITION BY artistfilter.GID ORDER by count(*) desc)
        as rank
    from artist_track at1, (
    select artist.ID, topgenreartist.GID, artist.NAME
      from artist , (
      select ag2.AID, ag2.GID
       from artist_genre ag2 ,(
       select ag1.gid as gid, rank() over (order by count(*) desc)
           as rank
         from Artist_GENRE ag1
         group by ag1.gid) genrelst
       where ag2.gid = genrelst.gid
           and genrelst.rank<=10) topgenreartist
      where artist.gender = 'Female'
          and artist.id = topgenreartist.aid)artistfilter
    where artistfilter.id = at1.aid
    group by artistfilter.gid,
        at1.aid, artistfilter.NAME)artistrank
  where artistrank.rank = 1 and genre.ID=artistrank.GID;
```

K

```
(select distinct genre1.name
  from genre genre1)
--get the list of all genres
minus
(select distinct genre.NAME
```

```
  from genre genre, artist_genre artist_genre, artist artist
  where genre.ID = artist_genre.GID
        and artist.ID = artist_genre.AID
        and artist.gender = 'Female');
--change 'Female' to 'Male'\ artist.type='Group'
```

L

```
select *
  from (
  select MaleArtist.area_name ,
      MaleArtist.id, count(*), rank()
      over (Partition by MaleArtist.area_name
      order by count(artist_track.tid))as rank
    from artist_track artist_track,(
      select artist.id, artist.name, artist.area_name
        from artist, (
        select artist.area_name
          from artist
          where artist.type = 'Group'
              and artist.area_name is not null
          group by artist.area_name
          having count(*)>10) arealist
        where artist.area_name  = arealist.area_name
            and artist.gender = 'Male') MaleArtist
      where artist_track.aid = MaleArtist.id
      group by MaleArtist.area_name,MaleArtist.id)
  where rank<=5;
```

M

```
create view compilation as
  select track.mid
    from track ,artist_track at1
    where track.tid = at1.tid
    group by track.mid
    having count(distinct at1.aid)>1;

select *
  from (
  select  artist.name
    from artist  join (
    select ct.tid, artist_track.aid
      from (
      select t.tid
```

```
        from track t join compilation c on t.mid = c.mid) ct
        join artist_track on ct.tid = artist_track.tid) cta
      on artist.id = cta.aid
      where artist.type = 'Group'
      group by artist.id, artist.name
    order by count(*) desc)
  where ROWNUM<=10;
```

N

```
create view ReleaseTrack as
  select ReleaseMedium.rid,track.tid
    from ReleaseMedium join track on
        track.mid = releasemedium.mid;

create view album as
  select R.rid, R.tid
    from ReleaseTrack R
    where exists (
    select artist.id
      from artist
        where not exists (
        select A_T.aid
          from Artist_Track A_T
          where A_T.aid <> Artist.id and A_T.tid = R.tid));

select colla.rid
  from (
  select album.rid
    from album, artist_track at1
    where album.tid = at1.tid
    group by album.rid
    order by count(distinct at1.aid) desc) colla
  where rownum<=10;
```

O

```
select R.name
  from ReleaseMedium R
  group by R.rid , R.name
  having count(*) = (
    select max(RM.count)
      from(
      select ReleaseMedium.rid, count(*) as count
        from ReleaseMedium
```

```
        group by ReleaseMedium.rid
    ) RM
  );
```

P

```
select genre_name.name
  from (
  select artist_genre2.gid, count(*)as count
    from (
    select artist.id
      from genre genre, artist artist, artist_genre artist_genre
      where artist.type ='Group' and genre.id = artist_genre.gid
          and artist_genre.aid = artist.id
      group by artist.id
      having count(*)>=3)groupid, artist_genre artist_genre2
    where artist_genre2.aid = groupid.id
    group by artist_genre2.gid) lst,genre genre_name
  where genre_name.id = lst.gid
      and lst.count = (
    SELECT max(gid_count.count)
--find the count of the most popular genre first
      FROM (
      select artist_genre2.gid, count(*)as count
        from (
        select artist.id
          from genre genre, artist artist,
              artist_genre artist_genre
          where artist.type ='Group'
              and genre.id = artist_genre.gid
              and artist_genre.aid = artist.id
          group by artist.id
          having count(*)>=3)groupid, artist_genre artist_genre2
        where artist_genre2.aid = groupid.id
        group by artist_genre2.gid) gid_count
     );
```

Q

```
select *
  from(
  select recording.name, count(*)
    from recording recording
    where recording.id<1000000
        and recording.name not like '[%]'
```

```
--get rid of [untitled] or[unknown] etc.
    group by recording.name
    order by count(*) desc)
  where rownum<=5;
```

R

```
--artist_count_track is a table with artist id
--    and the number of tracks he has.
--artist_count_release is a table with artist id
--    and the number of releases his track has contributed to.
--join2 simply join the above two tables together
--    in order to caculate the ratio in the next step

create view artist_count_track as
  select artist_track.aid, count(*) as track_count
    from artist_track artist_track
    group by artist_track.aid;

create view artist_count_release as
  select join1.aid, count(distinct releasemedium.rid)
      as release_count
    from (
    select artist_track.aid, track.mid
      from artist_track artist_track, track track
      where artist_track.tid = track.tid) join1,
          releasemedium releasemedium
    where join1.mid = releasemedium.mid
    group by join1.aid;

select artist.name
  from (
  select count_t.aid
    from artist_count_track count_t,
        artist_count_release count_r
    where count_t.aid = count_r.aid
    order by (count_t.track_count/count_r.release_count)
        desc)join2, artist artist
  where rownum <=10 and artist.id = join2.aid;
--select the top 10 and get the artist name from artist id
```

S

```
create view hittrack as
  select t.rcid
```

```
  from track t
  group by t.rcid
  having count(distinct t.mid)>100;

create view htid as
  select t.tid, t.rcid
  from track t, hittrack ht
  where t.rcid = ht.rcid;

create view hitartist as
  select at1.aid
  from artist_track at1,htid t
  where t.tid = at1.tid
  group by at1.aid
  having count(distinct t.rcid)>10;

create view hitability as
  select lst.aid , sum(lst.count) as score
  from (
  select hiat.aid, t.rcid ,count(distinct t.mid)
      as count, rank() over (partition by hiat.aid
      order by count(distinct t.mid) desc) as rank
    from track t,(
    select hta.aid,at1.tid
      from hitartist hta, artist_track at1
      where hta.aid = at1.aid) hiat
      where t.tid = hiat.tid
      group by hiat.aid,t.rcid
      having count(distinct t.mid)>100) lst
    where lst.rank<=10
    group by lst.aid;

select artist.name, ab.score
  from hitartist art , hitability ab,artist
  where art.aid = ab.aid and artist.id = art.aid;
```

4 Analyze the Queries

We've already created 8 indexes:

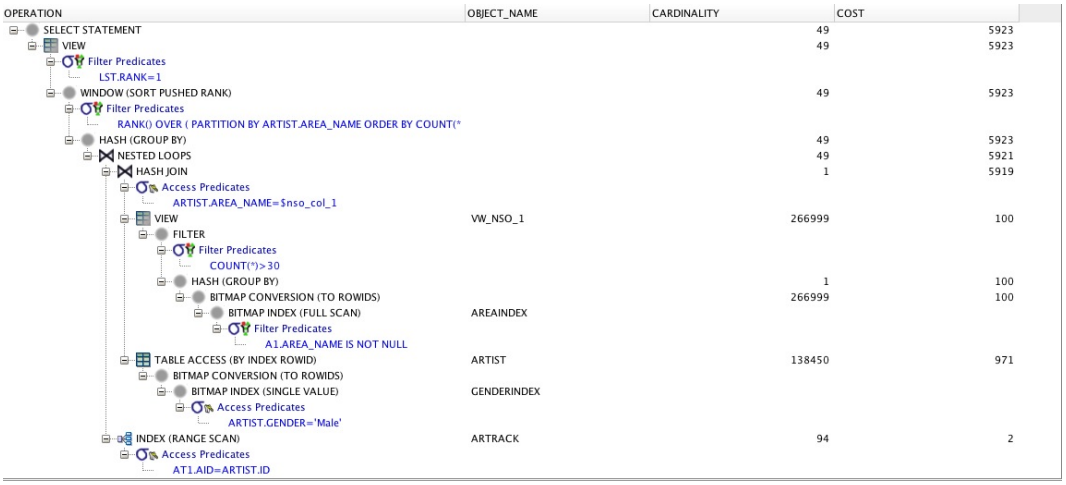
```
create bitmap index genderindex
on  artist(gender);
```

```
create bitmap index typeindex
on artist(type);

create bitmap index areaindex
on artist(area_name);

create index nameindex
on artist(name);

create index artrack
on  artist_track(aid);

create index artracktid
on  artist_track(tid);

create index genreindex
on artist_genre(gid);

create index tracktid
on track(mid);
```

We choosed Query H, Query I and Query J.

4.1 Query H

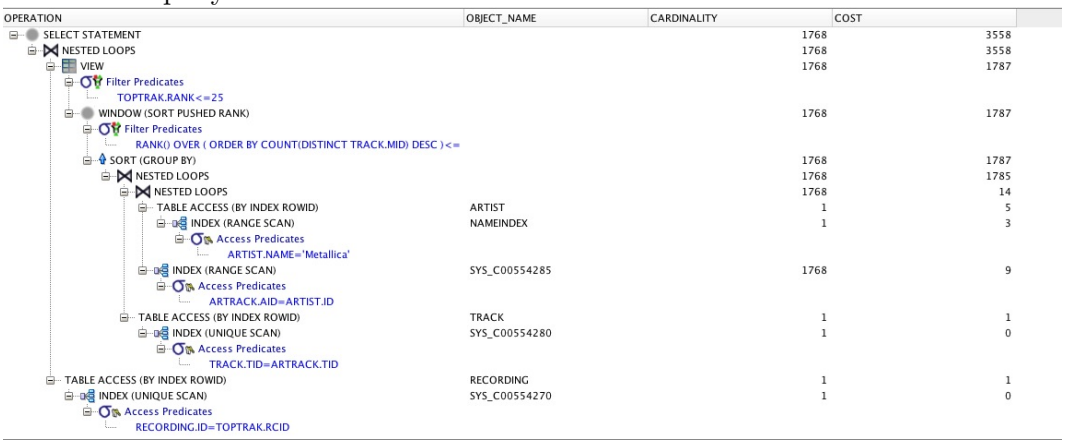
An AID-index is built on TABLE artist_track because it can speed up the range scan(In the SQL, we used only AID for this TABLE, so it's obvious that we should use this index).
There is a bitmap index built on area_name for TABLE artist to quickly group the artists by attribute "area_name". It will take much more time to grasp the number of artists in every area if we don't have this index (there are 264690 areas and 815387 artists, thus may cost about O(10^{10}) for grouping without area-indexing).
The system also used the bitmap index on gender for TABLE artist, since bitmap indexes are compact and they work best for columns with a small set of values(the attribute "gender" contains only 3 kinds of value: "Male", "Female" and "Other"). Then we can quickly partition artists from areas by gender.
Finally, we succeeded in running the query in **about 30s**. The join(artist and artist_track) takes most of time(60%+ of IO cost) since both contain a large amount of data, and relatively little time with group(by gender and area_name, about 20% IO cost) when we use the indexes.



OPERATION	OBJECT_NAME	CARDINALITY	COST
SELECT STATEMENT		49	5923
VIEW		49	5923
Filter Predicates			
LST.RANK=1			
WINDOW (SORT PUSHED RANK)		49	5923
Filter Predicates			
RANK() OVER (PARTITION BY ARTIST.AREA_NAME ORDER BY COUNT(*			
HASH (GROUP BY)		49	5923
NESTED LOOPS		49	5921
HASH JOIN		1	5919
Access Predicates			
ARTIST.AREA_NAME=$nso_col_1			
VIEW	VW_NSO_1	266999	100
FILTER			
Filter Predicates			
COUNT(*)>30			
HASH (GROUP BY)		1	100
BITMAP CONVERSION (TO ROWIDS)		266999	100
BITMAP INDEX (FULL SCAN)	AREAINDEX		
Filter Predicates			
A1.AREA_NAME IS NOT NULL			
TABLE ACCESS (BY INDEX ROWID)	ARTIST	138450	971
BITMAP CONVERSION (TO ROWIDS)			
BITMAP INDEX (SINGLE VALUE)	GENDERINDEX		
Access Predicates			
ARTIST.GENDER='Male'			
INDEX (RANGE SCAN)	ARTRACK	94	2
Access Predicates			
AT1.AID=ARTIST.ID			

4.2 Query I

The indexes are built on attribute name on for TABLE artist, and indexes of tables of their own. In the plan explaition, we can see that all IO cost by table access is very little thanks to the indexes.
We ran the query in **539 ms**. The "exists" cost most of the IO cost.



OPERATION	OBJECT_NAME	CARDINALITY	COST
SELECT STATEMENT		1768	3558
NESTED LOOPS		1768	3558
VIEW		1768	1787
Filter Predicates			
TOPTRAK.RANK<=25			
WINDOW (SORT PUSHED RANK)		1768	1787
Filter Predicates			
RANK() OVER (ORDER BY COUNT(DISTINCT TRACK.MID) DESC)<=			
SORT (GROUP BY)		1768	1787
NESTED LOOPS		1768	1785
NESTED LOOPS		1768	14
TABLE ACCESS (BY INDEX ROWID)	ARTIST	1	5
INDEX (RANGE SCAN)	NAMEINDEX	1	3
Access Predicates			
ARTIST.NAME='Metallica'			
INDEX (RANGE SCAN)	SYS_C00554285	1768	9
Access Predicates			
ARTRACK.AID=ARTIST.ID			
TABLE ACCESS (BY INDEX ROWID)	TRACK	1	1
INDEX (UNIQUE SCAN)	SYS_C00554280	1	0
Access Predicates			
TRACK.TID=ARTRACK.TID			
TABLE ACCESS (BY INDEX ROWID)	RECORDING	1	1
INDEX (UNIQUE SCAN)	SYS_C00554270	1	0
Access Predicates			
RECORDING.ID=TOPTRAK.RCID			

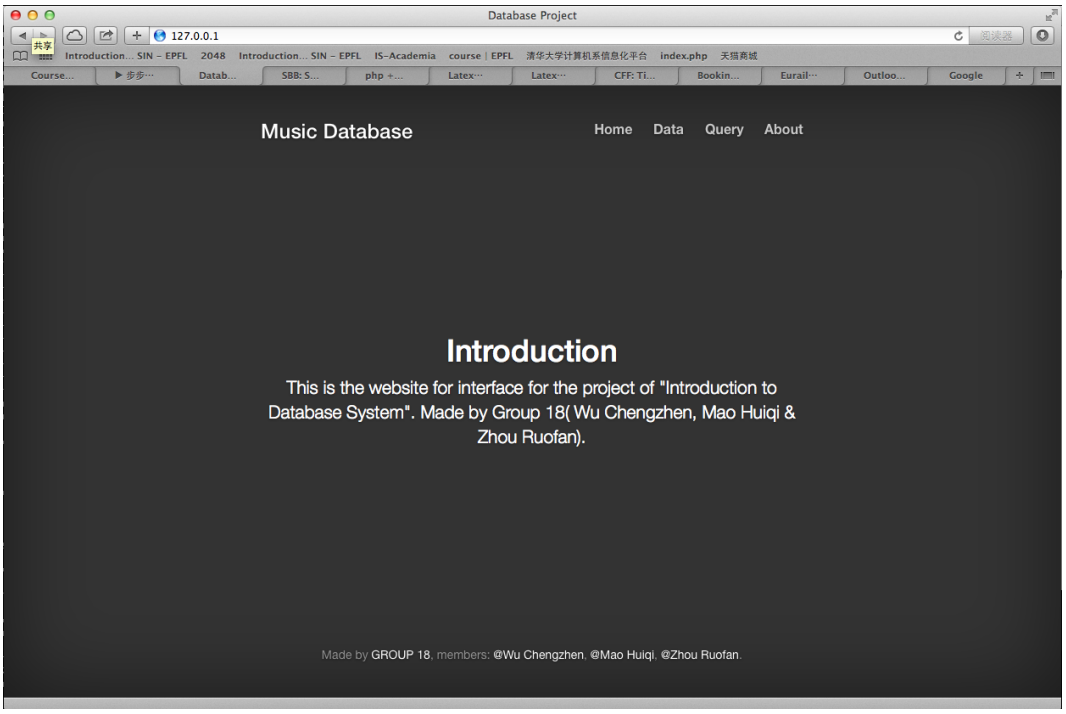
4.3 Query J

The indexes used are gender-index on TABLE artist and gid-index on TABLE genre. For gender-index, we've already discuessed in anlysis for Query H. And for gid-index, it managed to fast-full scan for the "group by gid". Thus, the indexes are important when we do such kind of queries(especially for "GROUP BY").
The query cost **7270 ms**. 50

OPERATION	OBJECT_NAME	CARDINALITY	COST
MERGE JOIN		56384989	63957
VIEW		56384783	62486
Filter Predicates			
WINDOW (SORT PUSHED RANK)		56384783	62486
Filter Predicates			
HASH (GROUP BY)		56384783	62486
HASH JOIN		56384783	38403
Access Predicates			
HASH JOIN		163044	2892
Access Predicates			
TABLE ACCESS (BY INDEX ROWID)	ARTIST	37783	966
BITMAP CONVERSION (TO ROWIDS)			
BITMAP INDEX (SINGLE VALUE)	GENDERINDEX		
Access Predicates			
ARTIST.GENDER='Female'			
HASH JOIN		163045	893
Access Predicates			
AG2.GID=GENRELST.GID			
VIEW		163045	119
Filter Predicates			
GENRELST.RANK<=10			
WINDOW (SORT PUSHED RANK)		163045	119
Filter Predicates			
RANK() OVER (ORDER BY COUNT(*) DESC)«			
HASH (GROUP BY)		163045	119
INDEX (FAST FULL SCAN)	GENREINDEX	163045	70
TABLE ACCESS (FULL)	ARTIST_GENRE	163045	75
TABLE ACCESS (FULL)	ARTIST_TRACK	13066296	8364
SORT (JOIN)		39077	1472
Access Predicates			
Filter Predicates			
TABLE ACCESS (FULL)	GENRE	39077	42

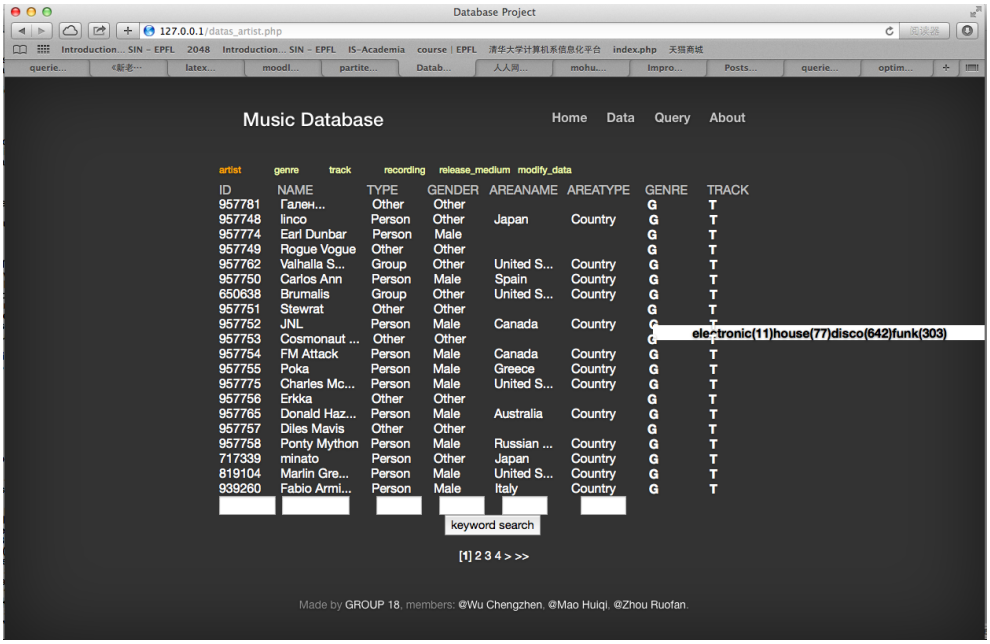
5 Interface

Since we've already upload the data to the server, it's convient for us to use PHP + Apache + Oracle to build the website as interface, like below:

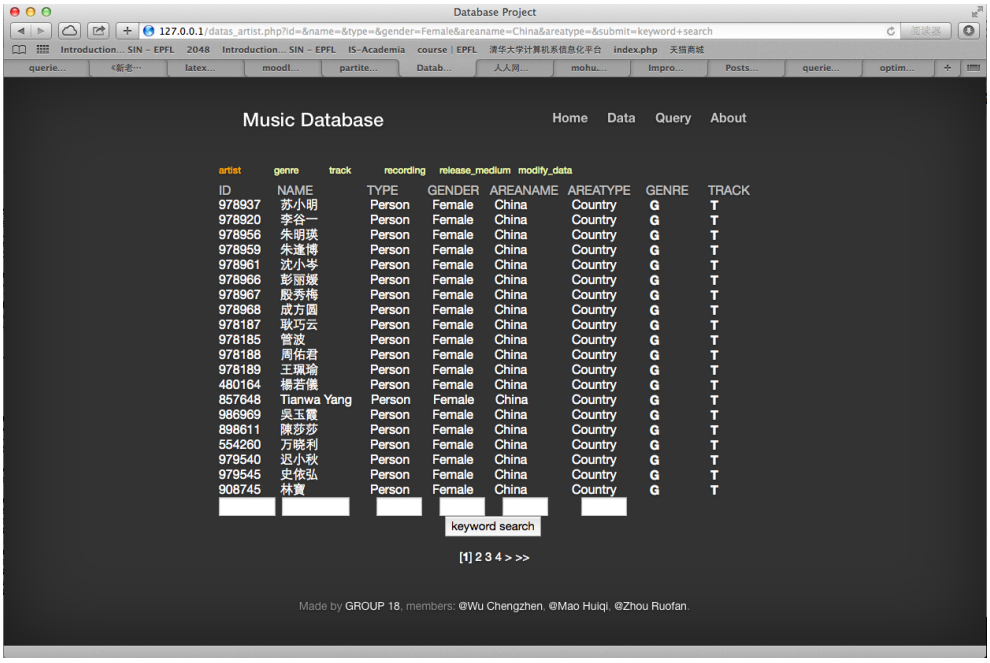


It's the index of our website. The website contains 4 parts: Home, Data, Query and About, and the functionally page is Data and Query.

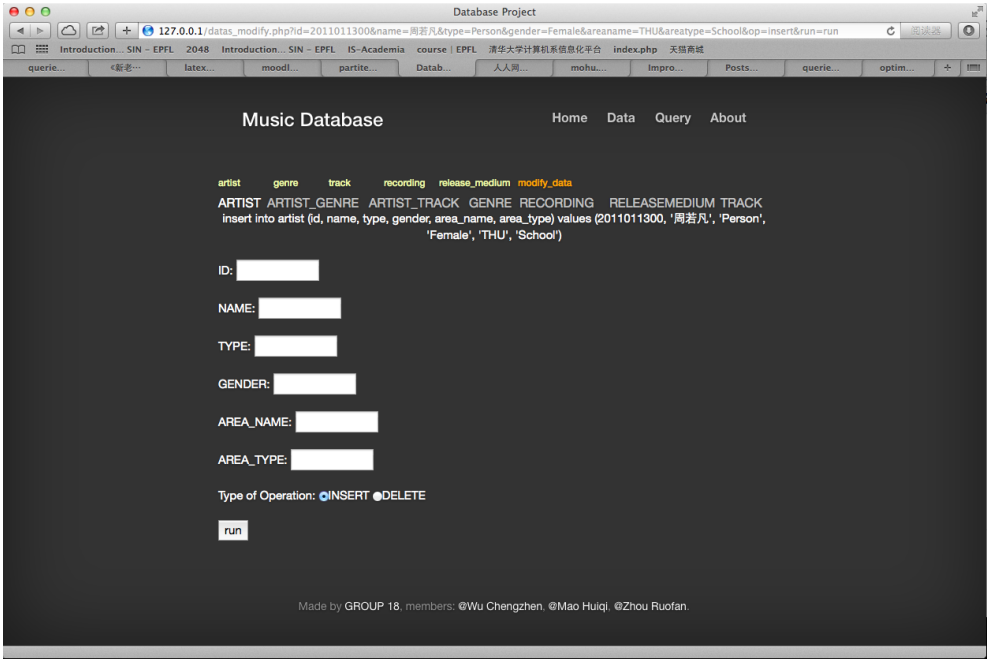
The screen shot of Data page is as below, the page shows the data of tables(you can select the table you want to see by clicking the link of table names, which are the yellow words on the uper part of the page). Each page would show 20 data and you can blowse more data by click the link of pages on the bottom part of the page. As the screen shot, it shows the 'artist' table. By moving your mouse onto those foreign keys, a prompt box showing the message of the table linked by the foreign key or relationship(as the screen shot, we move the mouse onto the 'genre' and a box showing message of the genre name and genre id of the artist).



Under each row there's a input box, and you can use it to search for keyword. Just by clicking the "Keyword Search" you can filter the data of the table. For example, next screen shot shows the result as we input 'China' and 'Female' under keyword 'AREA_NAME' and 'GENDER'.



And in the data_modify page, we implement the insert and delete of data. Once given paremeters and clicked "run", the SQL ran will be printed and the SQL will be executed. The result is as below. We've already tested and this function is available.



And we satisfied all the queries(A-S) in the Query page, and you can see the results

by clicking the query number.