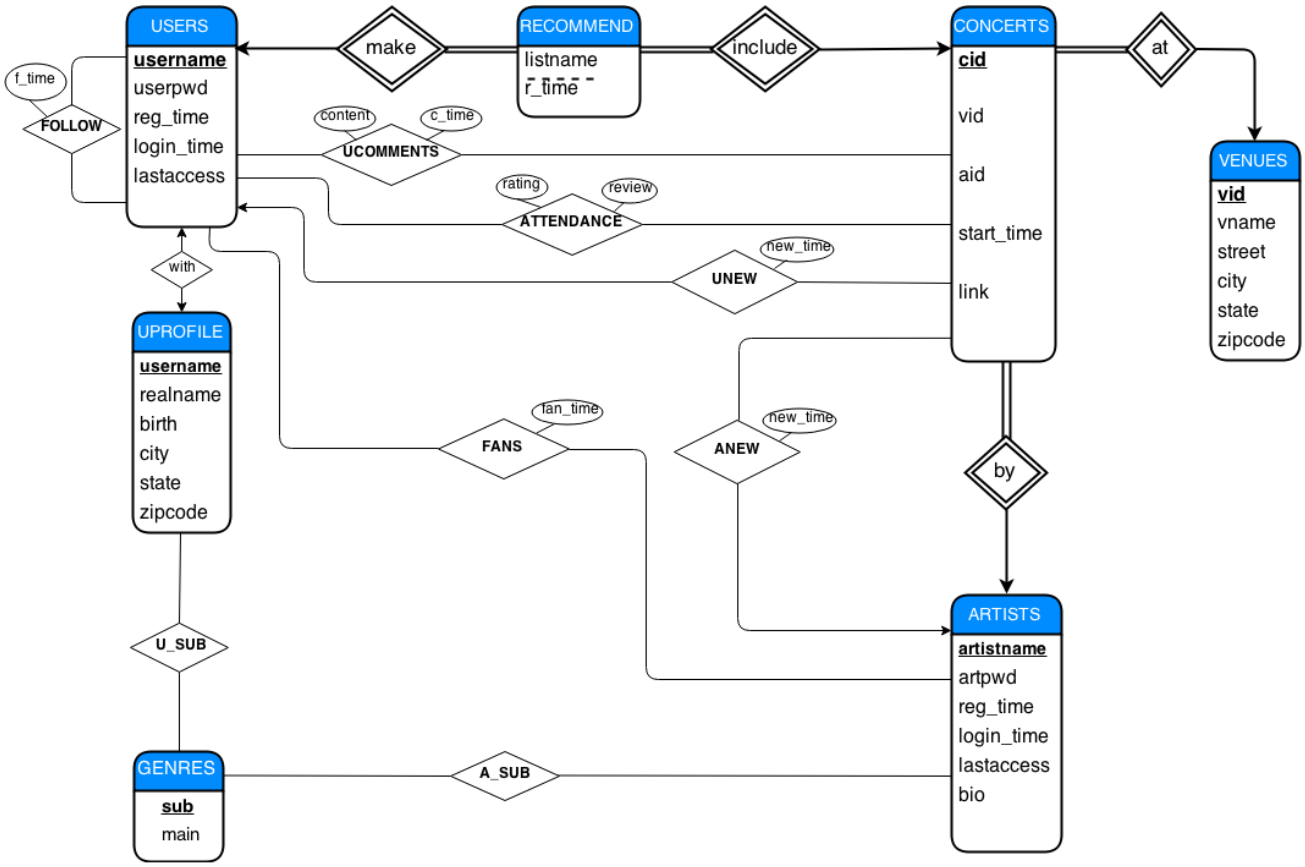
# CS 6083 Database Systems

## Course Project – Part 1

Xun Gong (xg411), Wei Yu(wy492)

2014.12.06

1. Entity-relationship model diagram




2. Schema description

`USERS(username, userpwd, reg_time, login_time, lastaccess)`

A user in table `USERS` have an `username` as the primary key to identify itself. `userpwd` is created by user when a user signs up, and `reg_time` is created automatically. `login_time` indicates user's login timestamp of this time and `lastaccess` indicates user's timestamp of the last time.

`UPROFILE(username, realname, birth, city, state, zipcode)`
foreign key `username` referencing `USERS`

Table `UPROFILE` contains a user's real name (`realname`), date of birth (`birth`), and the `city`, `state`, `zipcode` of residence.

`U_SUB(username, sub)`
foreign key `username` referencing `USERS`
foreign key sub referencing `GENRES`

Table U_SUB represents various tastes of a user.

FOLLOW(from_usr, to_usr, f_time)
foreign key from_usr referencing USERS.username
foreign key to_usr referencing USERS.username

Table FOLLOW represents relations between users. Anyone of users can follow another one with a timestamp (f_time).

ATTENDANCE(username, cid, rating, review)
foreign key username referencing USERS
foreign key cid referencing CONCERTS

A user can choose whether to go to a concert with empty values of rating and review. If a user have been to a concert, he/she can give a rating or a review of this concert.

CONCERTS(cid, vid, artistname, start_time, link)
foreign key vid referencing VENUES
foreign key artistname referencing ARTISTS

A concert is performed by an artist (artistname) in a venue (vid) with a time of start (start_time).

UCOMMENTS(username, cid, c_time, content)
foreign key username referencing USERS
foreign key cid referencing CONCERTS

UNEW(username, cid, new_time)
foreign key username referencing USERS
foreign key cid referencing CONCERTS

ANEW(artistname, cid, new_time)
foreign key artistname referencing ARTISTS
foreign key cid referencing CONCERTS

FANS(username, artistname, fan_time)
foreign key username referencing USERS
foreign key artistname referencing ARTISTS

ARTISTS(artistname, artistname, artpwd, reg_time, login_time, lastaccess, bio)

A_SUB(artistname, sub)
foreign key artistname referencing ARTISTS
foreign key sub referencing GENRES

VENUES(vid, vname, street, city, state, zipcode)

GENRES(sub, main)

RECOMMEND(username, cid, listname, r_time)
foreign key username referencing USERS
foreign key cid referencing CONCERTS

3. Create a database system called **LiveVibe** as below:

```
CREATE TABLE users (
   username varchar(20),
   userpwd  varchar(20),
   reg_time  datetime,
   login_time  datetime,
   lastaccess  datetime,
   primary key (username));

CREATE TABLE genres (
   sub varchar(20),
   main varchar(20),
   primary key (sub));

CREATE TABLE venues (
   vid char(10),
   vname varchar(40),
   street varchar(40),
   city  varchar(20),
   state  varchar(20),
   zipcode char(5),
   primary key (vid));

CREATE TABLE artists (
   artistname varchar(30),
   artpwd  varchar(20),
   bio varchar(300),
   reg_time  datetime,
   login_time  datetime,
   lastaccess  datetime,
   primary key (artistname));

CREATE TABLE uprofile (
   username varchar(20),
   realname  varchar(30),
   birth   datetime,
   city  varchar(20),
   state  varchar(20),
   zipcode char(5),
   primary key (username),
```

```
   foreign key (username) references users(username));

CREATE TABLE u_sub (
   username varchar(20),
   sub varchar(20),
   primary key (username, sub),
   foreign key (username) references users(username),
   foreign key (sub) references genres(sub));

CREATE TABLE follow (
  from_usr varchar(20),
  to_usr varchar(20),
  f_time datetime,
  primary key (from_usr, to_usr),
  foreign key (from_usr) references users(username),
  foreign key (to_usr) references users(username));

CREATE TABLE concerts (
  cid char(10),
  vid char(10),
  artistname varchar(30),
  start_time datetime,
  link varchar(40),
  primary key (cid),
  foreign key (vid) references venues(vid),
  foreign key (artistname) references artists(artistname));



CREATE TABLE attendance (
  username varchar(20),
  cid char(10),
  rating int(2),
  review  varchar(300),
  rv_time datetime,
  primary key (username, cid),
  foreign key (username) references users(username),
  foreign key (cid) references concerts(cid));

CREATE TABLE ucomments (
   username varchar(20),
   cid char(10),
   c_time datetime,
   primary key (username, cid, c_time),
   foreign key (username) references users(username),
   foreign key (cid) references concerts(cid));

CREATE TABLE unew (
```

```
    username varchar(20),
    cid char(10),
    new_time datetime,
    primary key (username, cid),
    foreign key (username) references users(username),
    foreign key (cid) references concerts(cid));

CREATE TABLE fans (
    username varchar(20),
    artistname varchar(30),
    fan_time datetime,
    primary key (username, artistname),
    foreign key (username) references users(username),
    foreign key (artistname) references artists(artistname));

CREATE TABLE anew (
    artistname varchar(30),
    cid char(10),
    new_time datetime,
    primary key (artistname, cid),
    foreign key (artistname) references artists(artistname),
    foreign key (cid) references concerts(cid));

CREATE TABLE recommend (
    username varchar(20),
    cid char(10),
    listname varchar(30),
    rm_time datetime,
    primary key (username, cid, listname),
    foreign key (username) references users(username),
    foreign key (cid) references concerts(cid));

CREATE TABLE a_sub (
    artistname varchar(30),
    sub varchar(20),
    primary key (artistname, sub),
    foreign key (artistname) references artists(artistname),
    foreign key (sub) references genres(sub));
```

4. Write SQL queries for the following tasks

We wrote **Stored Procedures AND views** to simplify later PHP implementation, I would like to demonstrate them all following User Operation Flow.

1) **User :**
   a) Sign up:
   After fill in the register form, and user submit. It will call **uprofile_insert()** based on their register type.

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `uprofile_insert`
(IN `uname` VARCHAR(20), IN `pwd` VARCHAR(20), IN `realn` VARCHAR(30),
 IN `bir` DATETIME, IN `ct` VARCHAR(20), IN `st` VARCHAR(20),
 IN `zip` CHAR(5), IN `rt` DATETIME)

BEGIN
    INSERT INTO users
    SET username=uname, userpwd=pwd, reg_time=rt, login_time=rt, lastaccess=rt;
    INSERT INTO uprofile
    SET username=uname, realname = realn, birth=bir, city=ct, state=st, zipcode=zip;
END$$
```

   b) Login
   We need to check which the login type of user, it could be normal user/fans, and it could be artists. We build this process within **check_id(), check_id_sub()** and **check_pwd()**, we guarantee same login user interface but different conditional control underhood.

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `check_id`
(IN `submitted_name` VARCHAR(20))
BEGIN
  CALL check_id_sub(submitted_name, @type);
    SELECT @type;
END$$
```

   **check_id()** will call a sub stored procedure called **check_id_sub()** to do the dirty work, deciding a login person is user or artist.

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `check_id_sub`
(IN `submitted_name` VARCHAR(20), OUT `usertype` VARCHAR(10))
BEGIN
  DECLARE urow INT;
    DECLARE arow INT;

  SELECT COUNT(distinct username) INTO urow
    FROM users
    WHERE username = submitted_name;

    IF urow != 0 THEN
    SET usertype = "user";
    ELSE
        SELECT COUNT(distinct artistname) INTO arow
        FROM artists
    WHERE artistname = submitted_name;

        IF arow != 0 THEN
      SET usertype = "artist";
      ELSE
      SET usertype = NULL;
      END IF;
    END IF;
END$$
```

After we get the persons login type, we can match the username and password, check if successfully login LiveVibe.

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `check_pwd`
(IN `submitted_name` VARCHAR(20), IN `submitted_pwd` VARCHAR(20),
 IN `usertype` VARCHAR(10))
BEGIN
    IF usertype = "user" THEN

          SELECT username
            FROM users
            WHERE username = submitted_name AND userpwd = submitted_pwd;

        ELSEIF usertype = "artist" THEN

          SELECT artistname
            FROM artists
            WHERE artistname = submitted_name AND artpwd = submitted_pwd;

        END IF;
END$$
```

After we logged in , we need to update login time and the last access time, we use **update_login_time()** and **update_LAT()** to complete these tasks.

**update_login_time()**

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `update_login_time`
(IN `submit_name` VARCHAR(20), IN `loginT` DATETIME, IN `usertype` VARCHAR(10))
BEGIN
  IF usertype = "user" THEN
    UPDATE users SET login_time = loginT
      WHERE username = submit_name;
  ELSEIF usertype = "artist" THEN
    UPDATE artists SET login_time = loginT
      WHERE artistname = submit_name;
    END IF;
END$$
DELIMITER ;
```

**update_LAT()**

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `update_LAT`
(IN `submit_name` VARCHAR(20), IN `LAT` DATETIME, IN `usertype` VARCHAR(10))
BEGIN
  IF usertype = "user" THEN
    UPDATE users SET lastaccess = LAT
      WHERE username = submit_name;
  ELSEIF usertype = "artist" THEN
    UPDATE artists SET lastaccess = LAT
      WHERE artistname = submit_name;
    END IF;
END$$
DELIMITER ;
```

c) Follow & Fan

Follow another user can be done by click on a button. Then we will update the **follow** table. For become an fan of a artist, the mechanism is the same, and **fans** table will be updated.

**follow_action()**

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `follow_action`
(IN `from_uname` VARCHAR(20), IN `to_uname` VARCHAR(20))
BEGIN
  INSERT INTO follow SET
    from_usr = from_uname,
    to_usr = to_uname,
    f_time = NOW();
DELIMITER ;
```

**fan – SQL:**

```
INSERT INTO fans SET
username = $username,
artistname = $artistname,
fan_time = NOW();
```

d) Plan to a concert
User can show their interests to a specific concert by clicking Plan to go button, this action will create a records in **attendance** table without rating and review value.

```
INSERT INTO attendance SET
username = $username, cid = $cid,
rating = NULL, review = NULL, rv_time = NULL;
```

e) Comment on a Concert
Any user can comment under the concert page for many times:

```
INSERT INTO ucomments SET
username = $username,cid = $cid,
content = $content, c_time = NOW();
```

f) Post a review and rating
We are assuming a logged in user can rating/review a concert only after showed his/her plan to go this concert. And of course, we assume no user can do this to a concert which hasn't started yet.
**rating:**

```
UPDATE attendance SET rating = $rating
WHERE username = $username AND cid = $cid;
```

**review:**

```
UPDATE attendance SET review = $review, rv_time = NOW()
WHERE username = $username AND cid = $cid;
```

g) Post a additional concert that not in LiveVibe
**user_Post() is for adding concert info concerts table and update unew table:**

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `user_Post`
(IN `cid_IN` CHAR(10), IN `vid_IN` CHAR(10), IN `aname` VARCHAR(30),
IN `date_time` DATETIME, IN `con_link` VARCHAR(40), IN `poster` VARCHAR(20))
BEGIN
    INSERT INTO concerts SET
        cid = cid_IN, vid = vid_IN, artistname = aname,
        start_time = date_time, link = con_link;

    INSERT INTO unew SET
        username = poster, cid = cid_IN, new_time = NOW();
END$$
DELIMITER ;
```

h) Make recommendation list of concerts

```
INSERT INTO recommend SET
username = $username, cid = $cid,
listname = $listname, rm_time = NOW();
```

i) Reputation System

For Star User with high reputation, they are allowed to post concert of bands or artist in LiveVibe, they can also make recommendation list. We define three views to monitor our three factors for calculating reputation score of each user.

**View 1: usr_follower**

```
CREATE VIEW `usr_follower` AS
select `users`.`username` AS `to_usr`,count(distinct `follow`.`from_usr`) AS `flwer_num`
from (`users` left join `follow` on((`users`.`username` = `follow`.`to_usr`)))
group by `users`.`username`;
```

**View 2: usr_following**

```
CREATE VIEW `usr_following` AS
select `users`.`username` AS `from_usr`,count(distinct `follow`.`to_usr`) AS `flw_num`
from (`users` left join `follow` on((`users`.`username` = `follow`.`from_usr`)))
group by `users`.`username`;
```

**View 3: usr_review**

```
CREATE VIEW `usr_review` AS
select `users`.`username` AS `username`,count(distinct `attendance`.`review`) AS `review_num`
from (`users` left join `attendance` on((`users`.`username` = `attendance`.`username`)))
group by `users`.`username`;
```

**Reputation Calculate Algorithm:**

$repu = 0.4 * $follower + 0.5 * $reviews + 0.1 * $following;

If reputation result is larger than **2**, we say this user is a star user and he/she can post concert information to LiveVibe.

**2) Band and Concert:**

a) Register and Login

**artist_insert() to create a artist profile:**

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `artist_insert`
(IN `aname` VARCHAR(30), IN `apwd` VARCHAR(20),
 IN `b` VARCHAR(300), IN `rt` DATETIME)
BEGIN
    INSERT INTO artists
    SET artistname=aname, artpwd=apwd, bio = b, reg_time=rt,
        login_time=rt, lastaccess=rt;
END$$
DELIMITER ;
```

**login_check** is the same as user because we integrate a conditional check into the procedure.

b) Post New Concert

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `art_Post`
(IN `cid_IN` CHAR(10), IN `vid_IN` CHAR(10),
 IN `aname` VARCHAR(30), IN `date_time` DATETIME,
 IN `con_link` VARCHAR(40))
BEGIN
    INSERT INTO concerts SET
        cid = cid_IN, vid = vid_IN, artistname = aname,
        start_time = date_time, link = con_link;

    INSERT INTO anew SET
        artistname = aname, cid = cid_IN, new_time = NOW();
END$$
DELIMITER ;
```

**3) Browse/Search Queries:**

- All concerts with a specific genre:

```
SELECT artistname FROM a_sub WHERE sub = ? ORDER BY artistname ASC;

SELECT C.cid, C.artistname, C.start_time,
       V.vname, V.street, V.city, V.state, V.zipcode
FROM a_sub AS G JOIN artists AS A JOIN concerts AS C JOIN venues AS V
ON G.artistname = A.artistname AND C.artistname = A.artistname AND C.vid = V.vid
WHERE G.sub = ?
ORDER BY C.start_time ASC
```

- All concerts recommended by my star user I followed
  &
  All newly posted concerts since the last time I logged in:

  We integrate these two task into one stored procedure **usr_new_feed():**
  After call this procedure, we will get all the concerts from the star users I followed, the result will be ordered by add_time of concerts.
  If star users you followed just add a concert to one of the recommendation list he made, your feed will updated and compare add_time with your lastaccesstime, and show a spin sign to indicate it's new. We add a **usr_newMsg_num()** procedure to assist new feed checking.

**usr_new_feed():**

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `usr_new_feed`
(IN `uname` VARCHAR(20))
BEGIN
    DECLARE lat DATETIME;
    DECLARE newM INT;

    SET newM = 0;

    SELECT lastaccess INTO lat
    FROM users
    WHERE username = uname;

    SELECT COUNT(*) INTO newM
    FROM users AS Me JOIN follow AS Star JOIN recommend AS R
    ON Me.username = uname AND Me.username = Star.from_usr
       AND Star.to_usr = R.username AND R.rm_time >= lat
    GROUP BY Me.username;

    IF newM > 0 THEN
        (SELECT Star.to_usr AS star, R.listname, R.rm_time, R.cid,
                C.artistname, C.start_time, V.vname, V.street,
                V.city, V.state, V.zipcode
        FROM users AS Me JOIN follow AS Star JOIN recommend AS R
                         JOIN concerts AS C JOIN venues AS V
        ON Me.username = uname AND Me.username = Star.from_usr AND
           Star.to_usr = R.username AND R.rm_time >= lat AND
           R.cid = C.cid AND C.vid = V.vid)
        UNION
        (SELECT Star.to_usr AS star, R.listname, R.rm_time, R.cid,
                C.artistname, C.start_time, V.vname, V.street,
                V.city, V.state, V.zipcode
        FROM users AS Me JOIN follow AS Star JOIN recommend AS R
             JOIN concerts AS C JOIN venues AS V
        ON Me.username = uname AND Me.username = Star.from_usr AND
           Star.to_usr = R.username AND R.rm_time < lat AND
           R.cid = C.cid AND C.vid = V.vid)
        ORDER BY rm_time DESC LIMIT 3;
    ELSE
        SELECT Star.to_usr AS star, R.listname, R.rm_time, R.cid,
               C.artistname, C.start_time, V.vname, V.street,
               V.city, V.state, V.zipcode
        FROM users AS Me JOIN follow AS Star JOIN recommend AS R
             JOIN concerts AS C JOIN venues AS V
        ON Me.username = uname AND Me.username = Star.from_usr AND
           Star.to_usr = R.username AND R.cid = C.cid AND C.vid = V.vid
        ORDER BY rm_time DESC LIMIT 3;
    END IF;
END$$
DELIMITER ;
```

**usr_newMsg_num():**

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `usr_newMsg_num`
(IN `uname` VARCHAR(20))
BEGIN
    DECLARE lat DATETIME;
    DECLARE newM INT;

    SET newM = 0;

    SELECT lastaccess INTO lat
    FROM users
    WHERE username = uname;

    SELECT COUNT(*) INTO newM
    FROM users AS Me JOIN follow AS Star JOIN recommend AS R
    ON Me.username = uname AND Me.username = Star.from_usr AND
       Star.to_usr = R.username AND R.rm_time >= lat
    GROUP BY Me.username;

    SELECT newM as newMessage;
END$$
DELIMITER ;
```

- All concerts in a time range:

```
SELECT C.cid, C.artistname, C.start_time,
       V.vname, V.street, V.city, V.state, V.zipcode
FROM concerts AS C JOIN venues AS V
ON C.vid = V.vid AND C.start_time >= ? AND C.start_time <= ?
ORDER BY C.start_time ASC
```

- Search user

  **SELECT username FROM users WHERE username = ?;**

- Search artist

  **SELECT artistname FROM artists WHERE artistname = ?;**

**4) System Recommendations:**

We implemented a simple recommendation called Vibe Sense. It will list concerts from the genres you like.

**usr_vibe_sense():**

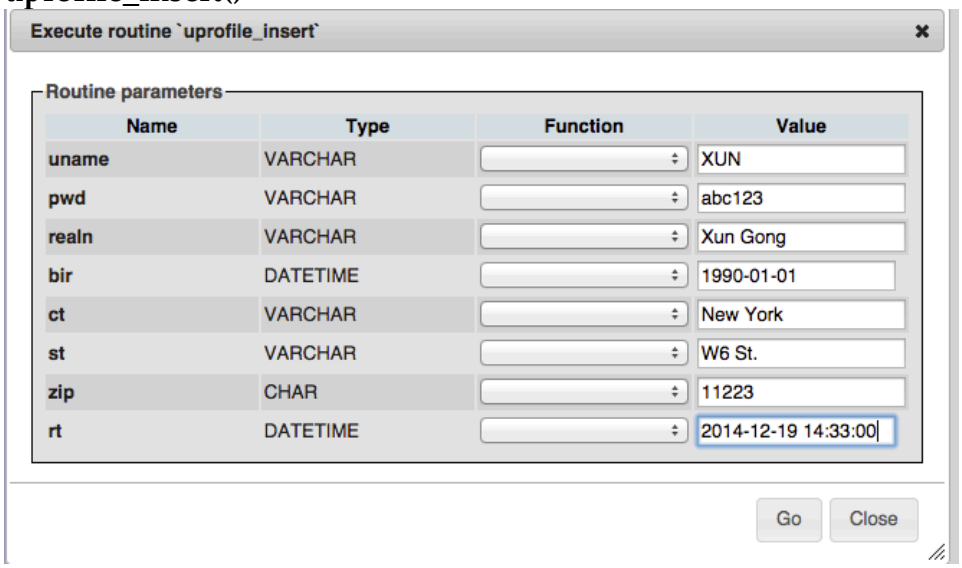
```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `usr_vibe_sense`
(IN `uname` VARCHAR(20))
BEGIN
    SELECT C.cid, C.artistname, C.start_time, V.vname, V.street,
           V.city, V.state, V.zipcode
    FROM u_sub AS U JOIN a_sub AS A JOIN concerts AS C JOIN venues AS V
    ON U.sub = A.sub AND A.artistname = C.artistname AND
       C.vid = V.vid AND C.start_time > NOW()
    WHERE U.username = uname
    ORDER BY C.start_time ASC;
END$$
DELIMITER ;
```

## 5. Test

### 1) User :

#### a) Sign up:

**uprofile_insert()**




+ Options

| | username | realname | birth | city | state | zipcode |
|---|---|---|---|---|---|---|
| Edit Copy Delete | johndoe | John Doe | 1985-05-12 13:44:34 | New York | NY | 10012 |
| Edit Copy Delete | magicmike | Mike Fassbender | 1977-04-02 00:00:00 | Los Angeles | CA | 90001 |
| Edit Copy Delete | mchotdog | Barack Obama | 1961-08-04 23:44:34 | Washington | DC | 20500 |
| Edit Copy Delete | test_user | Test User | 1995-05-20 15:15:00 | New York | NY | 10007 |
| Edit Copy Delete | XUN | Xun Gong | 1990-01-01 00:00:00 | New York | W6 St. | 11223 |

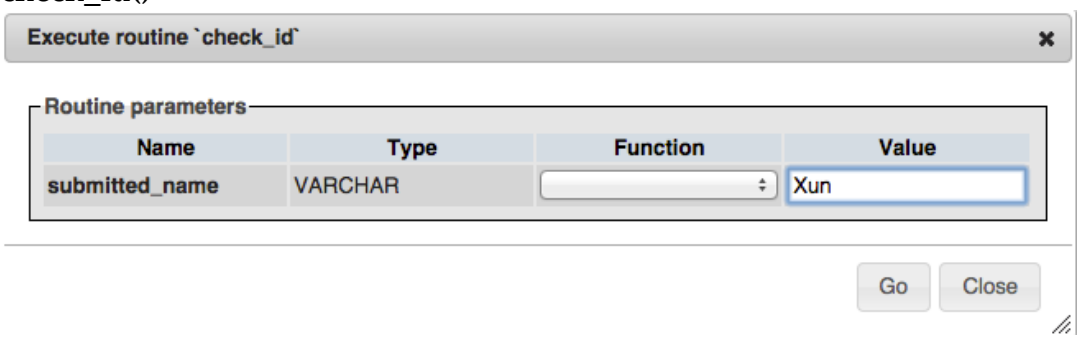
+ Options

| | username | userpwd | reg_time | login_time | lastaccess |
|---|---|---|---|---|---|
| Edit Copy Delete | johndoe | abc123 | 2011-05-12 13:44:34 | 2014-12-11 23:10:01 | 2014-12-11 23:13:30 |
| Edit Copy Delete | magicmike | abc123 | 2014-01-04 12:34:34 | 2014-11-23 13:22:48 | 2014-11-25 16:42:53 |
| Edit Copy Delete | mchotdog | abc123 | 2008-09-23 23:44:34 | 2014-12-11 20:42:09 | 2014-12-11 20:42:09 |
| Edit Copy Delete | test_user | abc123 | 2014-12-08 09:45:37 | 2014-12-11 22:36:24 | 2014-12-11 22:37:03 |
| Edit Copy Delete | XUN | abc123 | 2014-12-19 14:33:00 | 2014-12-19 14:33:00 | 2014-12-19 14:33:00 |

Check All With selected: Change Delete Export

#### b) Login

**check_id()**



Execution results of routine `check_id`

| @type |
| --- |
| user |

## check_pwd():

Execute routine `check_pwd`

Routine parameters

| Name | Type | Function | Value |
| --- | --- | --- | --- |
| submitted_name | VARCHAR | | Xun |
| submitted_pwd | VARCHAR | | abc123 |
| usertype | VARCHAR | | user |

Go Close

Execution results of routine `check_pwd`

| username |
| --- |
| XUN |

## update_login_time()

Execute routine `update_login_time`

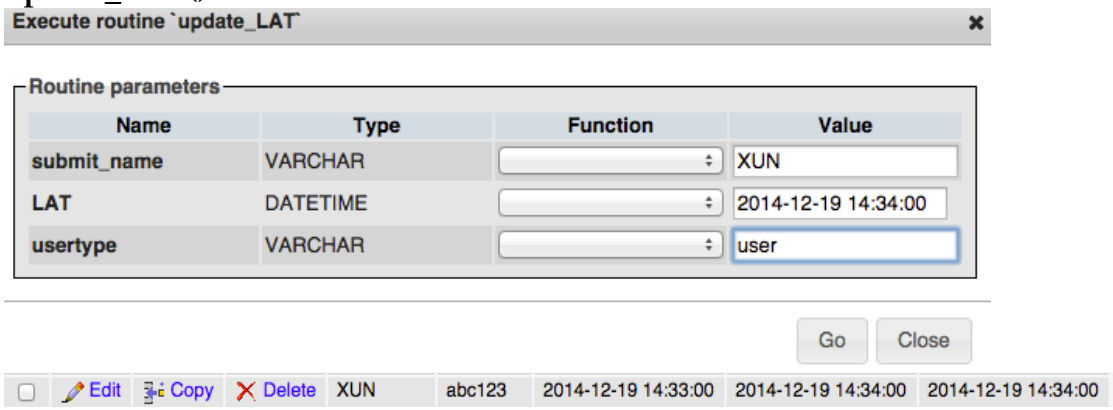
Routine parameters

| Name | Type | Function | Value |
| --- | --- | --- | --- |
| submit_name | VARCHAR | | XUN |
| loginT | DATETIME | | 2014-12-19 14:34:00 |
| usertype | VARCHAR | | user |

Go Close

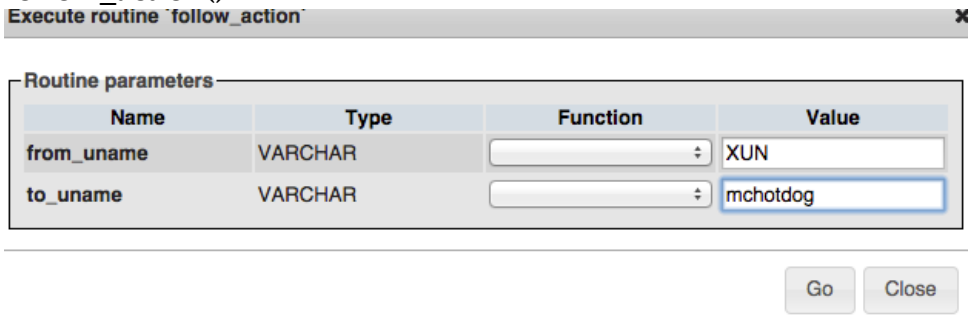
| | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| | Edit | Copy | Delete | XUN | abc123 | 2014-12-19 14:33:00 | 2014-12-19 14:34:00 | 2014-12-19 14:33:00 |

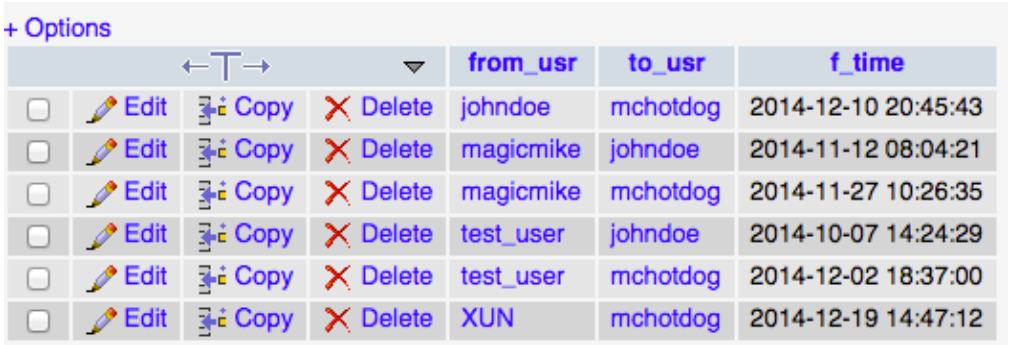
**update_LAT()**




c) Follow & Fan

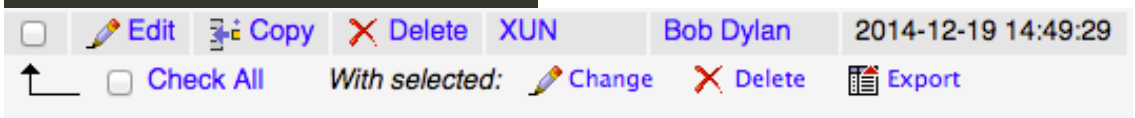
**follow_action()**




**fan – SQL:**

```
INSERT INTO fans SET
username = $username,
artistname = $artistname,
fan_time = NOW();
```



d) Plan to a concert

```
INSERT INTO attendance SET
username = $username, cid = $cid,
rating = NULL, review = NULL, rv_time = NULL;
```

+ Options

| | username 1 | cid | rating | review | rv_time |
|---|---|---|---|---|---|
| Edit Copy Delete | XUN | 5500109291 | NULL | NULL | NULL |
| Edit Copy Delete | mchotdog | 5500000945 | 7 | Review: A little bit disappointed. | 2014-11-30 11:37:48 |
| Edit Copy Delete | mchotdog | 5500000791 | NULL | NULL | NULL |

e) Comment on a Concert

```
INSERT INTO ucomments SET
username = $username,cid = $cid,
content = $content, c_time = NOW();
```

+ Options

| | username | cid | content | c_time 1 |
|---|---|---|---|---|
| Edit Copy Delete | XUN | 5500109291 | Just Leave a comment here! | 2014-12-19 14:54:15 |
| Edit Copy Delete | johndoe | 5500109291 | Finally Done. Leave A message. | 2014-12-11 23:13:27 |
| Edit Copy Delete | mchotdog | 5500000231 | Can't wait to this concert! | 2014-12-11 18:11:31 |

f) Post a review and rating

**rating:**

```
UPDATE attendance SET rating = $rating
WHERE username = $username AND cid = $cid;
```

**review:**

```
UPDATE attendance SET review = $review, rv_time = NOW()
WHERE username = $username AND cid = $cid;
```

| | username | cid | rating | review 1 | rv_time |
|---|---|---|---|---|---|
| Edit Copy Delete | mchotdog | 5500000432 | 8 | Review: My girlfriend got crazy last night. | 2014-11-25 13:34:14 |
| Edit Copy Delete | mchotdog | 5500000945 | 7 | Review: A little bit disappointed. | 2014-11-30 11:37:48 |
| Edit Copy Delete | johndoe | 5500000432 | 6 | Not bad. | 2014-12-02 13:00:00 |
| Edit Copy Delete | johndoe | 5500000953 | 8 | I just updated my review. Check this out. | 2014-12-11 13:03:12 |

g) Post a additional concert that not in LiveVibe

**user_Post()**

Execute routine `user_Post`

Routine parameters

| Name | Type | Function | Value |
|---|---|---|---|
| cid_IN | CHAR | | |
| vid_IN | CHAR | | |
| aname | VARCHAR | | |
| date_time | DATETIME | | |
| con_link | VARCHAR | | |
| poster | VARCHAR | | |

Go Close

+ Options

| | username | cid | new_time |
|---|---|---|---|
| Edit Copy Delete | mchotdog | 5500000231 | 2014-10-13 08:24:25 |
| Edit Copy Delete | mchotdog | 5500109291 | 2014-12-11 19:42:12 |

+ Options

| | cid | vid | artistname | start_time | link |
|---|---|---|---|---|---|
| Edit Copy Delete | 5500000189 | 8800000843 | OneRepublic | 2015-04-14 20:00:00 | http://www.bandsintown.com |
| Edit Copy Delete | 5500000231 | 8800000678 | Bob Dylan | 2015-07-01 20:00:00 | http://www.bandsintown.com |
| Edit Copy Delete | 5500000343 | 8800000999 | Linkin Park | 2014-12-12 19:00:00 | http://www.bandsintown.com |
| Edit Copy Delete | 5500000432 | 8800000111 | Damien Rice | 2014-11-24 19:00:00 | http://www.bandsintown.com |
| Edit Copy Delete | 5500000513 | 8800000333 | Belle & Sebastian | 2015-06-10 19:00:00 | http://www.bandsintown.com |
| Edit Copy Delete | 5500000634 | 8800000843 | Snapline | 2015-03-05 19:00:00 | http://www.bandsintown.com |
| Edit Copy Delete | 5500000791 | 8800000111 | Billy Joel | 2014-12-14 20:00:00 | http://www.bandsintown.com |
| Edit Copy Delete | 5500000945 | 8800000843 | Interpol | 2014-11-28 20:00:00 | http://www.bandsintown.com |
| Edit Copy Delete | 5500000953 | 8800000111 | Justin Timberlake | 2014-01-25 19:00:00 | http://www.bandsintown.com |
| Edit Copy Delete | 5500009876 | 8800000765 | Bob Dylan | 2014-11-10 19:00:00 | http://www.rollingstone.com/ |
| Edit Copy Delete | 5500109291 | 8800000999 | Maroon 5 | 2015-02-03 19:30:00 | http://www.maroon5.com/v |

Check All With selected: Change Delete Export

h) Make recommendation list of concerts

```
INSERT INTO recommend SET
username = $username, cid = $cid,
listname = $listname, rm_time = NOW();
```

+ Options

| | username | cid | listname | rm_time |
|---|---|---|---|---|
| Edit Copy Delete | mchotdog | 5500000189 | Where to go this winter | 2014-11-25 15:12:13 |
| Edit Copy Delete | mchotdog | 5500000231 | 2015 Must | 2014-12-10 07:26:38 |
| Edit Copy Delete | mchotdog | 5500000343 | Where to go this winter | 2014-12-11 23:10:00 |
| Edit Copy Delete | mchotdog | 5500000432 | Where to go this winter | 2014-11-25 15:12:13 |
| Edit Copy Delete | mchotdog | 5500000513 | Test Listname | 2014-12-11 18:47:35 |
| Edit Copy Delete | mchotdog | 5500000513 | Where to go this winter | 2014-11-25 15:12:13 |
| Edit Copy Delete | mchotdog | 5500000634 | 2015 Must | 2014-12-01 06:33:25 |

i) Reputation System

**View 1: usr_follower**

| to_usr | flwer_num |
|---|---|
| johndoe | 2 |
| magicmike | 0 |
| mchotdog | 4 |
| test_user | 0 |
| XUN | 0 |

**View 2: usr_following**

| from_usr | flw_num |
|---|---|
| johndoe | 1 |
| magicmike | 2 |
| mchotdog | 0 |
| test_user | 2 |
| XUN | 1 |

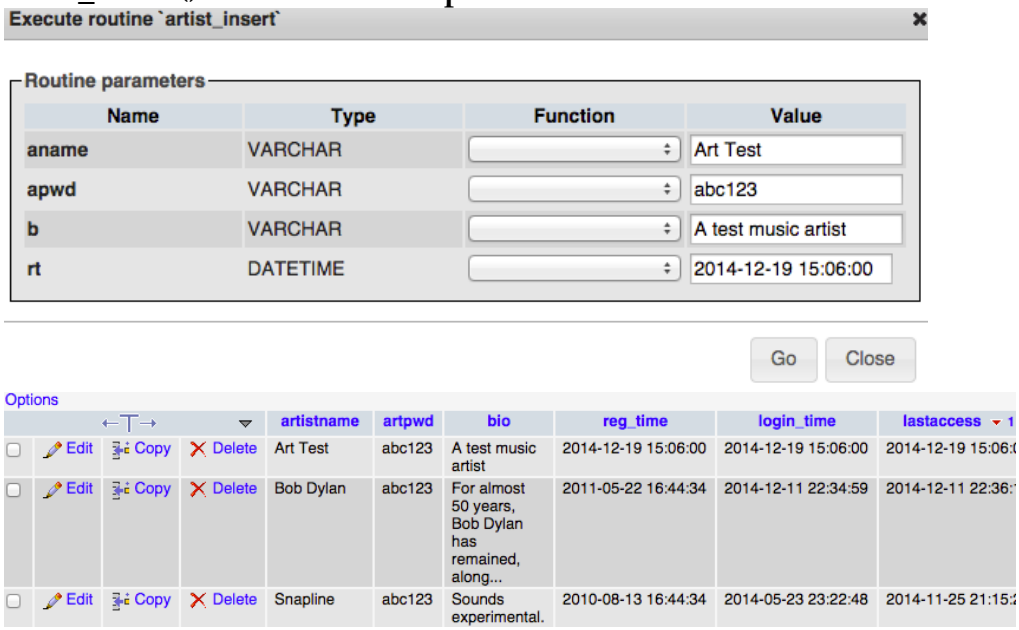
**View 3: usr_review**

| username | review_num |
|---|---|
| johndoe | 2 |
| magicmike | 0 |
| mchotdog | 2 |
| test_user | 0 |
| XUN | 0 |

**2) Band and Concert:**

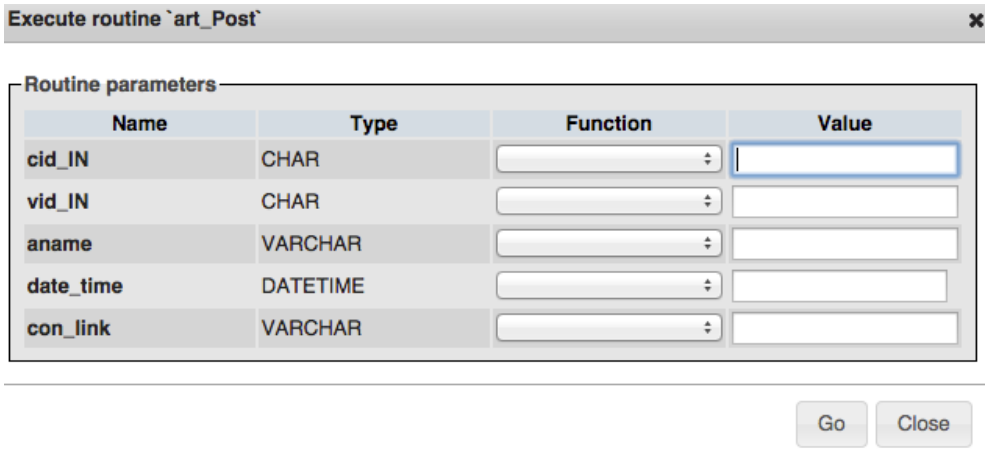
c) Register and Login

**artist_insert() to create a artist profile:**



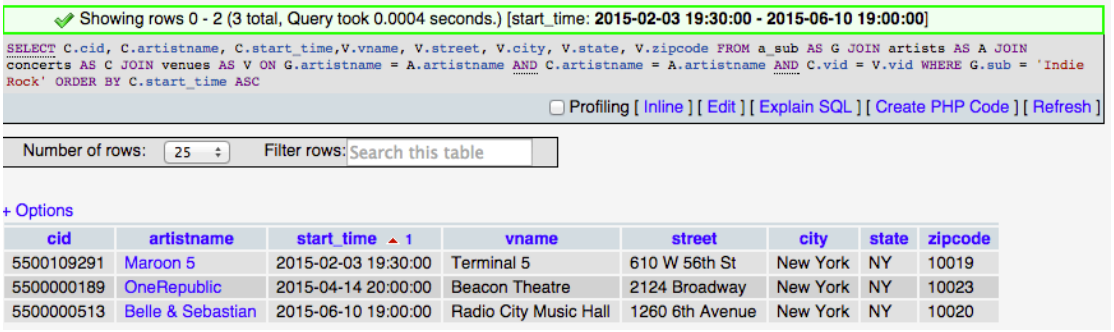

**login_check** is the same as user because we integrate a conditional check into the procedure.

d) Post New Concert



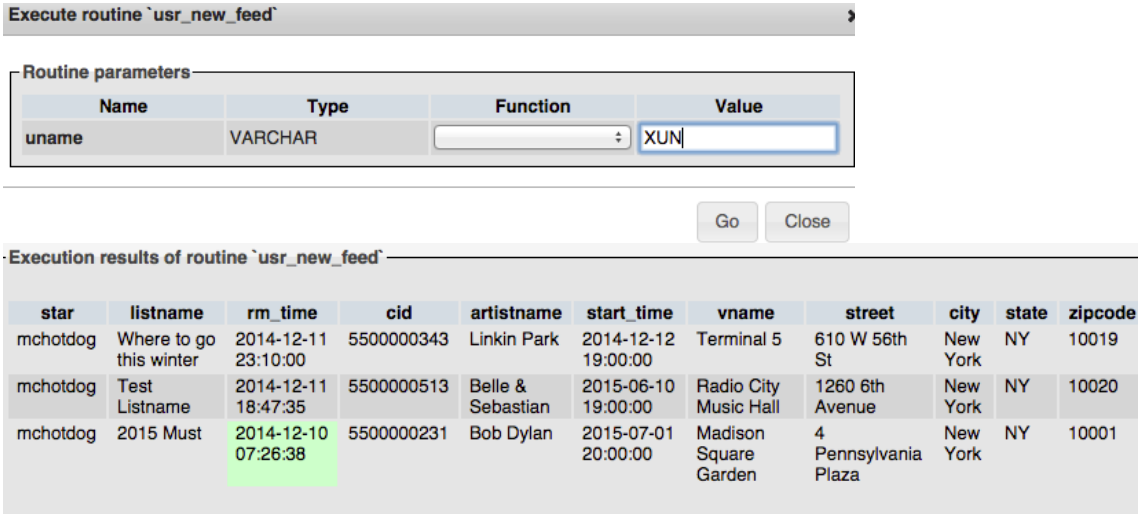

### 3) Browse/Search Queries:

- All concerts with a specific genre:

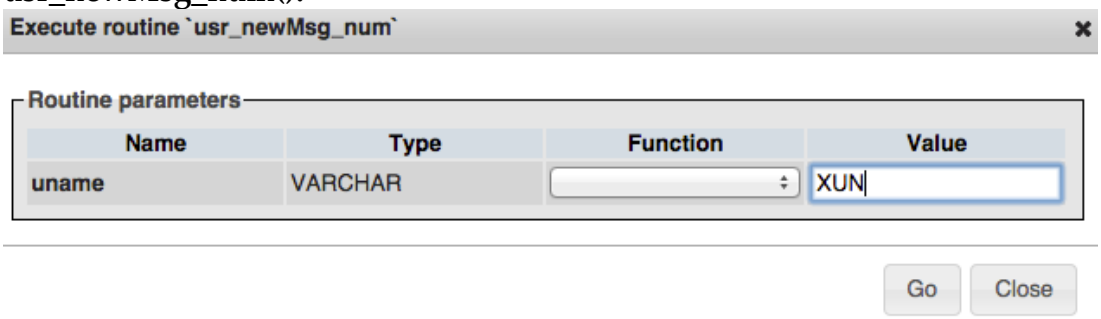



- All concerts recommended by my star user I followed
  &
  All newly posted concerts since the last time I logged in:

**usr_new_feed():**




**usr_newMsg_num():**



Execution results of routine `usr_newMsg_num`

| newMessage |
| --- |
| 0 |

- All concerts in a time range:

```
SELECT C.cid, C.artistname, C.start_time, V.vname, V.street, V.city, V.state, V.zipcode FROM concerts AS C JOIN venues AS V ON C.vid
= V.vid AND C.start_time >= '2013-01-01' AND C.start_time <= '2015-01-01' ORDER BY C.start_time ASC
```

Profiling [ Inline ] [ Edit ] [ Explain SQL ] [ Create PHP Code ] [ Refresh ]

Number of rows: 25 Filter rows: Search this table

+ Options

| cid | artistname | start_time ▲ 1 | vname | street | city | state | zipcode |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 5500000953 | Justin Timberlake | 2014-01-25 19:00:00 | Barclays Center | 620 Atlantic Ave | Brooklyn | NY | 11217 |
| 5500009876 | Bob Dylan | 2014-11-10 19:00:00 | Music Hall of Williamsburg | 66 North 6th St. | Brooklyn | NY | 11211 |
| 5500000432 | Damien Rice | 2014-11-24 19:00:00 | Barclays Center | 620 Atlantic Ave | Brooklyn | NY | 11217 |
| 5500000945 | Interpol | 2014-11-28 20:00:00 | Beacon Theatre | 2124 Broadway | New York | NY | 10023 |
| 5500000343 | Linkin Park | 2014-12-12 19:00:00 | Terminal 5 | 610 W 56th St | New York | NY | 10019 |
| 5500000791 | Billy Joel | 2014-12-14 20:00:00 | Barclays Center | 620 Atlantic Ave | Brooklyn | NY | 11217 |

- Search user

```
SELECT username FROM users WHERE username = 'XUN'
```

Number of rows: 25 Filter rows: Search

+ Options

| username |
| --- |
| Edit Copy Delete XUN |

- Search artist

Showing rows 0 - 0 (1 total, Query took 0.0002 seconds.)

```
SELECT artistname FROM artists WHERE artistname = 'Bob Dylan'
```

Prof

Number of rows: 25 Filter rows: Search this table

+ Options

| artistname |
| --- |
| Edit Copy Delete Bob Dylan |

Check All With selected: Change Delete Export

## 4) System Recommendations:

### usr_vibe_sense():