**Samarth -- Documenting DB Schema**

This DataBase provides a large stockpile of royalty-free music. It allows users to play music, create playlists, and view information about the music and albums and their primary creators and other collaborators on it, etc.

The purple-headed tables in the DB schema are entities (in terms of Entity-Relationship diagrams). The blue-headed tables are relationships (also called associative entities).

1. **The Entities in the DB**

The various Entities in this DB schema are shown by a purple-headed table (also known as a “relations”). The entities in this DB are as follows:

ARTIST (**art\_id:INT**, art\_name:VARCHAR, art\_bio:VARCHAR, art\_flwrs:INT, art\_mbrs:INT)

The entity gives information on various artists.

Each artist in this system will have exactly one tuple (row) in this ralation (table).

The various attributes (column names) in this relation are as follows.

art\_id – Each artist in the system is identified by a distict id which is his/her art\_id.

art\_name – Name of the artist identified by art\_id.

art\_bio – Biography of this artist (i.e. it is a short summary that explains the life of this artist, including his/her education, accomplishments, endeavors, careers, etc).

art\_flwrs – The number of users who have chosen to follow this artist. A “user” is anyone who wishes to use this system and has registered with the system.

art\_mbrs – In this system an artist may give permission to one or more users to be his/her “members”. A member of an artist has permission to contribute to the works of the artist such as uploading music to that artist or contributing to him/her otherwise. The “membership” relationship between an artist and a user is shown by the MEMBER table.

The attribute art\_mbrs, is a derived attribute indicating the number of members of this artist, i.e., the number of rows in MEMBER table that correspond to this artist.

The primary key (PK) of ARTIST table is art\_id. So no two rows of ARTIST will have the same value in the art\_id column. In other words, each artist in the system will have exactly one row in ARTIST.

ALBUM (**alb\_id:INT**, alb\_type:ENUM, alb\_name:VARCHAR, alb\_cvr:VARCHAR, alb\_trks:INT, alb\_likes:INT, FK: art\_id:INT)

This entity provides information on various albums available in the system.

alb\_id – Each album in the system is identified by a distinct alb\_id.

alb\_type – Type of this music album, which can be "Single", "EP", or "LP".

alb\_trks – Number of tracks in this album.

alb\_name – Name of this album.

alb\_cvr – Cover of this album (i.e. The front packaging art of a commercially released studio album); AKA the album art.

alb\_likes – Number of users who like this album. The relationship of a user liking an album is shown in the table LIKE\_ALBUM.

art\_id – This identifies the main artist of this album. If there are other artists on this album (called “collaborators”), they will be shown in the COLLABORATE\_ALBUM table.

The Primary Key (PK) of ALBUM is alb\_id since it identifies a specific album.

ALBUM has a foreign Key (FK) art\_id via the reference art\_id in ARTIST. So at any time, 3all the art\_id values present in ALBUM must also be present in ARTIST table.

GENRE (**gen\_id:INT**, gen\_name:VARCHAR)

This entity provides information about various genres of music that exist in the system.

gen\_id – Each genre in the system is identified by a distinct id named gen\_id. Note that a given genre may or may not have a corresponding album at present.

gen\_name – Name of this genre (e.g. Jazz, Classical, Folk, Country, Blues, Dance, Pop, Hip Hop, etc.)

Primary Key (PK) of GENRE is gen\_id.

USER (**usr\_id:INT**, usr\_name:VARHCAR, usr\_pass:VARCHAR, usr\_email:VARCHAR, usr\_phone:VARCHAR, usr\_friends:INT)

This entity provides information on various users of this system. A user is someone who has registered into the system.

usr\_id – Each user has a distinct id, named the usr\_id.

usr\_name – Name of this user.

usr\_pass – Unique password of this user.

usr\_email – Email address of this user. No two users can have the same email address.

usr\_phone – Phone number of this user.

usr\_friends – Number of friends this user has. The pairs of friends in the system are shown in the table FRIEND.

Primary Key (PK) of USER is usr\_id.

TRACK (**trk\_id :INT**, trk\_name:VARCHAR, trk\_file:VARCHAR, trk\_pos:INT, trk\_lyrics:VARCHAR, trk\_len:TIME, trk\_strms:INT, trk\_likes:INT, FK: alb\_id:INT)

trk\_id – Each music track in the system has a distinct id, named trk\_id. Note that each track belongs to a specific album. So if a music track appears in two different albums, then you need to view this as two different tracks, with two different trk\_id values, in the TRACK table.

trk\_name – Name of this track.

trk\_file -- A digital file that stores audio data for a track, such as MP3 files, etc.

trk\_pos – Track position, i.e., the numeric (positive integer) position of this track within its album. For example, the first track in an album has trk\_pos 1.

try\_lyrics – The sequence of words of this track.

trk\_len – Length of this track.

trk\_strms -- Refers to how many times this music track has been streamed (i.e., listened). Note that information about which user has listened to this track how many times is shown in the table TRACK\_LISTEN in the column trk\_lst\_listens.

trk\_likes – Number of users who like this track. Information about which user likes which track is listed in the table LIKE\_TRACK.

alb\_id – The specific album to which this track belongs. Note that each track must belong to a specific album.

The Primary Key (PK) of TRACK is trk\_id.

The Foreign Key (FK) of TRACK is alb\_id, referencing alb\_id in ALBUM table. This is the unique album to which this track belongs.

PLAYLIST (**ply\_id:INT**, ply\_name:VARCHAR, ply\_desc:VARCHAR, ply\_cvr:VARCHAR, ply\_flwrs:INT, FK: usr\_id:INT)

This entity represents a playlist (a sequential list of tracks) created by a user.

ply\_id – Each playlist in the system has a distinct id, named ply\_id.

ply\_name – Name of this playlist.

ply\_desc – Description of this playlist.

ply\_cvr – Cover of this playlist (i.e. an image that represents this playlist’s theme or mood—very helpful to all listeners to identify a playlist and discover new music).

ply\_flwrs – Number of users who have chosen to follow this playlist. The table FOLLOW\_PLAYLIST lists which users follow which playlists.

usr\_id – This identifies the user who created this playlist. Here usr\_id is a value in the usr\_id column in the USER table. Besides the user who created a given playlist, it may have collaborators contributing to it. The table COLLABORATE\_PLAYLIST lists which users are collaborators on which playlist.

The Primary Key (PK) of PLAYLIST is ply\_id.

The Foreign Key (FK) of PLAYLIST is usr\_id, with reference to the usr\_id attribute of the USER table.

CHAIN (**chn\_id:INT**, FK: ply\_id:INT)

The entity CHAIN lists all the chains in the system.

chn\_id – Each chain has a distinct id, named chn\_id.

ply\_id – Corresponding to each chain, there is a unique playlist, identified by ply\_id.

Primary Key (PK) of CHAIN is chn\_id

Foreign Key (FK) of CHAIN is ply\_id.

1. **The Relationships (AssociativeEntities) in the DB**

MEMBER (FK: usr\_id:INT, FK: art\_id:INT)

This relationship lists which user is a member of which artist.

The usr\_id value identifies a user (as in USER table) and art\_id identifies an artist (as in ARTIST table), such that this user is a member of this artist.

Note that a given usr\_id value in USER may have zero or more entries in MEMBER table. Similarly a given art\_id value in ARTIST table may have zero or more entries in the MEMBER table.

The Primary Key (PK) of MEMBER is (usr\_id, art\_id)

Foreign Keys (FK) of MEMBER are usr\_id (referencing usr\_id in USER) and art\_id (referencing art\_id in ARTIST). Since they are foreign keys, any usr\_iod value in MEMBER must also be present as an usr\_id value in USER. Similarly any art\_id value in MEMBER must also be present in ARTIST.

FOLLOW\_ARTIST (FK: usr\_id:INT, FK: art\_id:INT)

This relationship lists which user follows which artist.

The usr\_id value identifies a user (as in USER table) and art\_id identifies an artist (as in ARTIST table), such that this user follows this artist.

Note that a given usr\_id value in USER may have zero or more entries in FOLLOW\_ARTIST table. Similarly a given art\_id value in ARTIST table may have zero or more entries in the FOLLOW\_ARTIST table.

The Primary Key (PK) of FOLLOW\_ARTIST is (usr\_id, art\_id).

Foreign Keys (FK) of FOLLOW\_ARTIST are usr\_id (referencing usr\_id in USER) and art\_id (referencing art\_id in ARTIST). Since they are foreign keys, any usr\_id value in FOLLOW\_ARTIST must also be present as an usr\_id value in USER. Similarly any art\_id value in FOLLOW\_ARTIST must also be present in ARTIST.

COLLABORATE\_ALBUM (FK: art\_id:INT, FK: alb\_id:INT)

Each album has one primary creator artist and zero or more collaborators artists. The creator artist is not a collaborator on that album.

This relationship lists which artist collaborates on which album.

The art\_id identifies an artist (as in ARTIST table) and alb\_id value identifies an album (as in ALBUM table), such that this artist collaborates on this album.

Note that a given alb\_id value in ALBUM may have zero or more entries in COLLABORATE\_ALBUM table. Similarly a given art\_id value in ARTIST table may have zero or more entries in the COLLABORATE\_ALBUM table.

The Primary Key (PK) of COLLABORATE\_ALBUM is (art\_id, alb\_id).

Foreign Keys (FK) of COLLABORATE\_ALBUM are art\_id (referencing art\_id in ARTIST) and alb\_id (referencing alb\_id in ALBUM). Since they are foreign keys, any alb\_id value in COLLABORATE\_ALBUM must also be present as an alb\_id value in ALBUM. Similarly any art\_id value in COLLABORATE\_ALBUM must also be present in ARTIST.

LIKE\_ALBUM (FK: usr\_id:INT, FK: alb\_id:INT)

This relationship lists which user likes which album.

The description is similar to FOLLOW\_ARTIST above.

Primary Key (PK) of LIKE\_ALBUM is (usr\_id, alb\_id).

Foreign Keys (FK) of LIKE\_ALBUM are usr\_id and alb\_id.

ALBUM\_GENRE (FK: alb\_id:INT, FK: gen\_id:INT)

This relationship lists which artist collaborates on which album.

Primary Key (PK) of ALBUM\_GENRE is (alb\_id, gen\_id).

Foreign Keys (FK) of ALBUM\_GENRE are alb\_id and gen\_id.

TRACK\_LISTEN (FK: usr\_id:INT, FK: trk\_id:INT, FK: trk\_lst\_date:DATE, trk\_lst\_listens:INT)

This relationship shows a summary of listening history of a user listening to a particular track.

The value of usr\_id identifies a user and trk\_id identifies a particular music track.

The value of trk\_lst\_date tells the last date when this user listened to this track.

The value of trk\_lst-listens tells how many times in the past this user has listened to this track.

Primary Key (PK) of TRACK\_LISTEN is (usr\_id, trk\_id)

Foreign Keys (FK) of TRACK\_LISTEN are usr\_id and trk\_id.

LIKE\_TRACK (FK: usr\_id:INT, FK: trk\_id:INT)

This relationship tells which user likes which track.

Primary Key (PK) of LIKE\_TRACK is (usr\_id, trk\_id).

Foreign Keys (FK) of LIKE\_TRACK are usr\_id (with reference to usr\_id in USER table) and trk\_id (referencing trk\_id in TRACK table).

QUEUE\_TRACK (FK: usr\_id:INT, FK: trk\_id:INT, que\_trk\_pos:INT)

With each user, conceptually we have a queue where his pending requests for various tracks are waiting.

This relationship shows details of the queues associated with various users.

The value of usr\_id identifies a user.

The values of trk\_id identifies a track that this user has requested, and therefore is in the queue associated with this user.

The value of que\_trk\_pos is the position of this request in this queue. The positions are 1, 2, ...

Primary Key (PK) of QUEUE\_TRACK is (usr\_id, que\_trk\_pos).

Foreign Keys (FK) of QUEUE\_TRACK are usr\_id and trk\_id.

FOLLOW\_PLAYLIST (FK: usr\_id:INT, FK: ply\_id:INT)

This relation tells which user likes which playlists.

Primary Key (PK) of FOLLOW\_PLAYLIST is (usr\_id, ply\_id).

Foreign Keys (FK) of FOLLOW\_PLAYLIST are usr\_id and ply\_id.

COLLABORATE\_PLAYLIST (FK: usr\_id:INT, FK: ply\_id:INT)

Each playlist (identified by ply\_id) has a unique creator, identified by usr\_id in PLAYLIST entity.

In addition, a playlist may have zero or more collaborators on it.

This table shows which user collaborates on which playlist.

Primary Key (PK) of COLLABORATE\_PLAYLIST is (usr\_id, ply\_id).

Foreign Keys (FK) of COLLABORATE\_PLAYLIST are usr\_id and ply\_id.

FRIEND (FK: usr\_id1:INT, FK: usr\_id2:INT)

This table tells which user is friend of which user.

Primary Key (PK) of FRIEND is (usr\_id1, usr\_id2).

Foreign Keys (FK) of FRIEND are usr\_id1 (referencing usr\_id in USER) and usr\_id2 (referencing usr\_id in USER).

PLAYLIST\_TRACK (FK: ply\_id:INT, FK: trk\_id:INT, ply\_trk\_pos:INT)

This table shows, for each playlist, the tracks in it and their serial position in it (i.e., 1, 2, ...).

A playlist is identified by the ply\_id attribute.

A track in a playlist is identified by trk\_id, and its position in the playlist is identified by ply\_trk\_pos.

Primary Key (PK) of PLAYLIST\_TRACK is (ply\_id, ply\_trk\_pos).

Foreign Keys (FK) of PLAYLIST\_TRACK are ply\_id and trk\_id.

CHAIN\_TRACK (FK: chn\_id:INT, FK: trk\_id:INT, chn\_trk\_pos:INT)

This table shows the various tracks in a chain, and their positions within the chain.

The chain and the track are identified by chn\_id and trk\_id respecxtively.

The posistion of a given track in a given chain is identified by chn\_trk\_pos.

Primary Key (PK) of CHAIN\_TRACK (chn\_id, chn\_trk\_pos).

Foreign Keys (FK) of CHAIN\_TRACK are chn\_id and trk\_id.

TRACK\_GENRE (FK: trk\_id:INT, FK: gen\_id:INT)

A track may correspond to zero or more genres and a genre may correspond to zero or more tracks.

This table shows which tracks correspond to which genres.

Primary Key (PK) of TRACK\_GENRE is (trk\_id, gen\_id).

Foreign Keys (FK) of TRACK\_GENRE are trk\_id (with reference to TRACK) and gen\_id (with reference to GENRE).