CSE 412

Project Phase 3

Timothy Millea, Michael Rainsford, Alex Reyes, Michael Scott, Adam Sinck

Index Explanation:

Indexes were used in the case of foreign keys. Foreign keys are going to be commonly used when updates are done, and therefore will be searched more often than other attributes. An excellent example of this is the index on the “person” for person name and person birthdate. Any other sub-entity of person, band member for example, when updated, will need to also search for the corresponding person entry, and this is done by the primary key of person. So anytime a sub-entity of person is updated, those indexes will be searched, hence justifying indexing those attributes.

Changes From Phase 2:

The understanding by the group was the phase 2 was the finalization of the database architecture. All changes to the schema were done from phase 1 to phase 2. Indexes were already used in phase 2 to facilitate foreign key usage, as mySQL does not support foreign key usage without indexing. Because the schema was already tested rigorously with queries in phase 2, and was already using indexing, the requirements for the application and phase 3 were already met, and no changes were needed.

Query Descriptions

1. Insert Query.



* 1. This query takes in the parameters of ‘band\_name’, ‘band\_start\_date’, ‘person\_name’, ‘person\_birthdate’, ‘member\_start\_date’, and ‘member\_end\_date.’ After receiving these parameters included by the user, the the insert query creates a new ‘band\_member’ that contains all of the above mentioned properties. Once this band member is successfully created, then the insert query adds this member into the database for later access. This is obviously useful because it allows the database to be constantly changing and improving with new bands and band members. Without an insert query new bands/band members would never be able to be added to the database and therefore our application would be obsolete relatively quickly.

1. Select Queries
   1. Select\_bands\_with\_given\_tour\_date



* + 1. This query takes one parameter, and that is the start\_date for when the user is going on tour. After receiving this input, the query searches through the database and returns all of the persons that are in a band, and that band is going to be on tour after the given date. This is useful because a user may want to search for a band on tour after they finish the school semester, so they would enter their graduation date and check for tour dates that line up with their availability.
  1. Select\_bands\_with\_oldest\_or\_youngest\_member



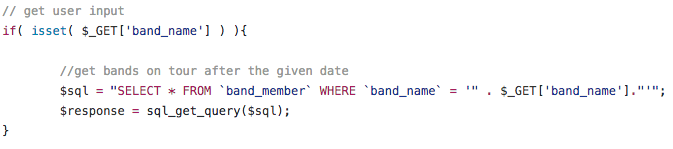
* + 1. This query takes one parameter, and that is whether or not the band member they are looking for is either the oldest or the youngest. After this, the query then computes the appropriate subquery based on the input parameter. After receiving this input, the query searches through the database and returns all of the persons that are either the youngest or the oldest member in the band. This is useful because a user may want to search for which band member is the oldest or youngest, and by doing this, and coupling it with other search queries, it allows the user flexibility in the obscurity of their search interests.
  1. Select\_people\_in\_bands\_with\_songs\_of\_given\_rating



* + 1. This query takes one parameter, and that is a billboard ranking. After receiving this input, the query searches through the database and returns all of the bands that have received a billboard ranking equal to, or lesser than the given ranking. This is useful because a user may want to sort the query results based on their considered “talent” and rank them against other bands and band members.
  1. Select\_small\_or\_large\_bands



* + 1. This query takes one parameter, and that is whether or not the band they are looking for is a large band or a small band, i.e. bands greater than or less than ten members. After this, the query then computes the appropriate subquery based on the input parameter. After receiving this input, the query searches through the database and returns all of the bands that have the appropriate number of members in them. This is useful because a user may want to search for a band to join, and knows that a smaller band is more likely to accept more members than a larger band and so they would only want to look for small bands.
  1. get\_band\_members



* + 1. This query takes one parameter, and that is a band member’s name. After receiving this input, the query searches through the database and returns the band member, and all of the related information, based on the given name. This is useful because a user may want to insert a new member into the database, and after doing so, the user may want to double check that the band member was added successfully by searching for them specifically by name.