**MUSIC STORE DATABASE MANAGEMENT SYSTEM**

**Database Project - 2017**

**Section 1**

**Name: SIDDHARTHA SAMBIDI**

**WIU ID: 916-59-4705**

**Oracle Account: srs148**

**Name: ARUN RAJU GUDAPATI**

**WIU ID: 916-23-5979**

**Oracle Account:arg114**

**Database has been created in account "srs148"**

**Password:** ‘Mahesh07’

1. **DESIGN PHASE:**

a) **Describe the enterprise:**

**i). Introduction and brief enterprise description.**

Music Store database is a store franchise that provides services to customer to buy all kinds of albums from store. Albums are organized in such way that customers can find what they need quickly and efficiently. The Music Store is brand name with franchises all across United States of America located in more than 26 states. Our **MUSIC STORE** Database deals with relations between store and customers, store and albums, albums and songs, artists and its purchases.

**ii) What functions should the system perform?**

The function of this project is to maintain a database that holds all the information of albums and its relations with customer and store. Database involves providing information about albums and its types inside store to the visiting customer. Customer gets to check the album details including rating and price and he/she is allowed to buy albums from the visited store.

**iii). Who are the end users? Remember that the DBA is NOT an end user.**

The end users in our project are STORE and CUSTOMER. Store has access to everything in the database which includes albums and its related data like songs, purchases, customers who purchased albums etc. Store can insert, delete, update and retrieve anything related to albums and customers. CUSTOMER has only access to view albums info in the store and his purchase details.

**iv). How will data Obsolescence be handled?**

Music store database stores purchase details of the customer for 10 years. Customer’s purchase data older than 10 years is retrieved from the database by the store. This is how we are handling old/obsolescence data in our database.

**v). Where did you get the idea for this project?**

The idea of this project evolved came up because we wanted to do something interesting and that which is common and related to real life activity. There are many popular online music stores but music stores (offline) in real world are not so famous because of its organization and customer service. Hence we wanted to make a music store data base with good organized data that provides info to the customer. Lastly, my love for music helped me in coming up with this idea for project.

b) **Entity Relationship Design:**

i) **Entity Description**

**1) STORE** (Store-Id, StoreName, StoreLocation, PhoneNumber )

The entity STORE contains all information of the store with store name is same as its brand name located in different cities across United States of America.

* Store-Id - Identifies unique ID of the store
* StoreName - Gives name of the store.
* StoreLocation - Gives store location city/state
* PhoneNumber – Gives contact number of the store

**2) ALBUM** (Album-ID, AlbumName, ReleaseDate, Price, Rating)

This entity has information about the Albums like album name its price, release date, price and rating.

* Album-ID - Identifies each Album using unique ID
* AlbumName - Provides name of the album (Not Unique)
* Release Date – Provides the release date of album
* Price - Gives price of the album.
* Rating - Gives the rating for the album

**3) CUSTOMER** (Customer-ID, CustomerName, MobileNumber, Email)

This entity has all the information about customer who makes any purchase from the store.

* Customer-ID - Identifies each customer using unique ID
* CustomerName – Provides customer Name while purchasing
* MobileNumber – Provides mobile number of the customer.
* Email – Provides email of the customer during purchase

**4)** **PURCHASE** (Purchase-ID, PurchaseDate, PurchaseCost, Quantity)

This entity has the information about the purchase made by the customer

* Purchase-ID - Identifies each purchase ID
* PurchaseDate - Gives purchase date of the purchase made.
* PurchaseCost – Gives the total purchase cost of the purchases made
* Quantity – Gives the number of albums purchased at a time.

**5) ARTIST** (ArtistID#, ArtistName, Country, AlbumsSold)

This entity has the information about the artist who composed the album and released into market (STORE).

* ArtistID – Identifies artist with an unique id
* ArtistName – Gives the artist name who composed the album
* Country – Gives the country or location where he/she was born
* AlbumsSold – Gives info on number of albums sold of the given artist.

6) **SONG** (SongID#, SongName, Language, Genre)

This entity has description of the songs in the album.

* SongID# - Identifies each song with an ID
* SongName – Gives name of the song in the albums
* Language – Gives the language of the song in the album
* Genre – provides info about the type of song

ii) **Relationship Description.**

* **Store/Album (one to many) - Sells**

This one to many relationship shows that each store can have many albums to be sold

* **Store/Customer (many to many) – Visited by**

This many to many relation shows 1 customer can visit many stores and 1 store can be visited by many customers.

* **Customer/Purchase (one to many) - Makes**

This one to many relation describes one customer can make many purchases.

* **Album/Purchase (one to many) - Has**

This one to many relation describes that one purchase can have many albums in its purchase

* **Album/Song (many to many) - Lists**

This many to many relation describes one song can be listed on many albums and one album can have many songs.

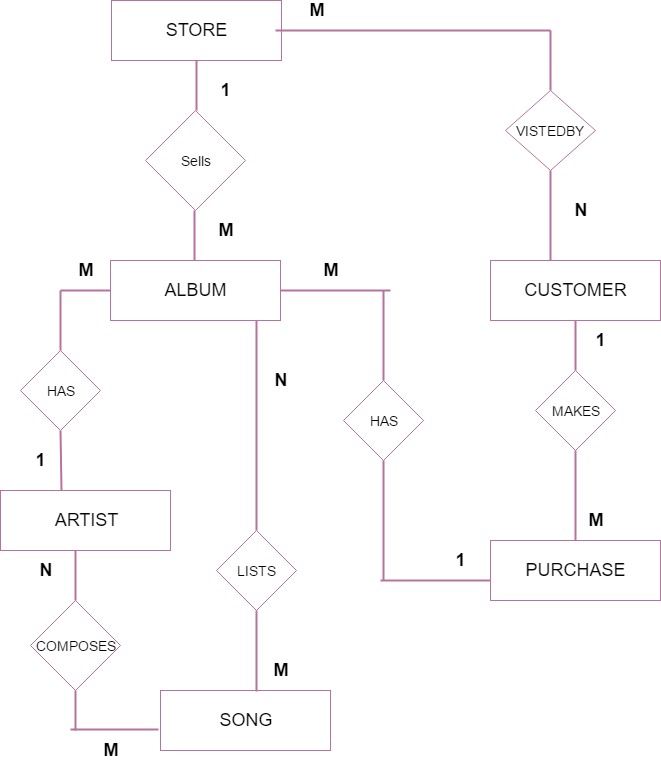
* **Album/Artist (one to many) - Has**

This one to many relation describes one artist can have many albums.

* **Artist/Song (many to many) - Composes**

This many to many relation describes one artist can compose many songs and one song can be composed by many artists.

iii) **E-R Diagram:**



**c)** **Conceptual Level:**

**TABLES:**

* STORE (**storeid,** storename, storelocation, phonenumber)

* ARTIST (**artistid**, artistname, country, awardswon, albumssold)
* CUSTOMER (**customerid**, customername, mobile, email)
* PURCHASE (**purchaseid**, purchasecost, purchasedate, quantity, customerid)
* SONG (**songid**, songname, language, genre)
* ALBUM (**albumid,** albumname, releasedate, price, rating, purchaseid, artistid, storeid)
* COMPOSES (**artistid**, **songid**)
* LISTS (**albumid**, **songid**)
* VISITEDBY(**storeid**, **customerid**)

**TABLE:** STORE (**storeid,** storename, storelocation, phonenumber)

Primary Key: StoreID

Foreign Key: N/A

3NF – Normalized form (As storeid is a super key and functionally determines all attributes)

|  |  |  |
| --- | --- | --- |
| ATTRIBUTES | COLUMN DOMAINS | INTEGRITY CHECKS |
| StoreId | Varchar2(10) | Primary Key (Not Null) |
| StoreName | Char(10) | Not Null |
| StoreLocation | Char(10) | Not Null |
| PhoneNumber | Number(10) | Unique |

**TABLE:** ARTIST (**artistid**, artistname, country, awardswon, albumssold)

Primary Key: ArtistID

Foreign Key: N/A

3NF – Normalized form (As artistid is a super key and functionally determines all attributes)

|  |  |  |
| --- | --- | --- |
| ATTRIBUTES | COLUMN DOMAINS | INTEGRITY CHECKS |
| Artistid | Varchar2(10) | Primary Key (Not Null) |
| Artistname | Char(15) | Not Null |
| Country | Char(10) | N/A |
| Awardswon | Number(6) | N/A |
| Albumsold | Number(10) | N/A |

**TABLE:** CUSTOMER (**customerid**, customername, mobile, email)

Primary Key: CustomerID

Foreign Key: N/A

3NF – Normalized form (As customerid is a super key and functionally determines all attributes)

|  |  |  |
| --- | --- | --- |
| ATTRIBUTES | COLUMN DOMAINS | INTEGRITY CHECKS |
| CustomerID | Varchar2(10) | Primary Key (Not Null) |
| Customername | Char(12) | Not Null |
| Mobile | Number(13) | Not Null |
| Email | Varchar2(20) | N/A |

**TABLE:** PURCHASE (**purchaseid**, purchasecost, purchasedate, quantity, customerid)

Primary Key: purchaseid

Foreign Key: customerid

3NF – Normalized form (As purchaseid is a super key and functionally determines all attributes and customerid references customer table which is self-dependent.

|  |  |  |
| --- | --- | --- |
| ATTRIBUTES | COLUMN DOMAINS | INTEGRITY CHECKS |
| PurchaseID | Varchar2(11) | Primary Key (Not Null) |
| PurchaseCost | Number(5,2) | Not Null |
| PurchaseDate | Date | Not Null |
| Quantity | Number(10) | Not Null |
| CustomerID | Varchar2(10) | N/A |

**TABLE:** SONG (**songid**, songname, language, genre)

**Primary Key**: songid

**Foreign Key**: N/A

**3NF** – Normalized form (As songid is a super key and functionally determines all attributes)

|  |  |  |
| --- | --- | --- |
| **ATTRIBUTES** | **COLUMN DOMAINS** | **INTEGRITY CHECKS** |
| SongID | Varchar2(7) | Primary Key (Not Null) |
| SongName | Char(15) | Not Null |
| Language | Date | N/A |
| Genre | char(10) | N/A |

**TABLE:** ALBUM (**albumid,** albumname, releasedate, price, rating, purchaseid, artistid, storeid)

**Primary Key:** AlbumID

**Foreign Key**: purchaseid, artistid, storeid

|  |  |  |
| --- | --- | --- |
| **ATTRIBUTES** | **COLUMN DOMAINS** | **INTEGRITY CHECKS** |
| AlbumID | Varchar2(10) | Primary Key (Not Null) |
| AlbumName | Varchar2(15) | Not Null |
| ReleaseDate | Date | Not Null |
| Price | Number(5,2) | Not Null |
| Rating | Number(3,2) | N/A |
| PurchaseID | Varchar2(11) | N/A |
| ArtistID | Varchar2(10) | N/A |
| StoreID | Varchar2(10) | N/A |

**TABLE:** COMPOSES (**artistid**, **songid**)

Primary Key: (artistid, songid)

Foreign Key: artistid, songid

|  |  |  |
| --- | --- | --- |
| ATTRIBUTES | COLUMN DOMAINS | INTEGRITY CHECKS |
| SongID | Varchar2(7) | Not Null |
| ArtistID | VarChar2(10) | Not Null |

**TABLE:** LISTS (**albumid**, **songid**)

Primary Key: (albumid,songid)

Foreign Key: songid, albumid

|  |  |  |
| --- | --- | --- |
| ATTRIBUTES | COLUMN DOMAINS | INTEGRITY CHECKS |
| SongID | Varchar2(7) | Not Null |
| AlbumID | VarChar2(10) | Not Null |

**TABLE:** VISITEDBY (**storeid**, **customerid**)

Primary Key: (storeid, customerid)

Foreign Key: storeid, customerid

|  |  |  |
| --- | --- | --- |
| ATTRIBUTES | COLUMN DOMAINS | INTEGRITY CHECKS |
| StoreID | Varchar2(10) | Not Null |
| CustomerID | VarChar2(10) | Not Null |

d) **External View**

|  |  |  |
| --- | --- | --- |
| **TABLES** | **STORE** | **CUSTOMERS** |
| Store | Select, Update, Alter | No Access |
| Album | Select, Update, Insert, Delete, Alter | Select |
| Customer | Select, Update, Insert, Delete, Alter | No Access |
| Purchase | Select, Update, Insert, Delete | Select |
| Artist | Select, Update, Alter | No Access |
| Song | Select, Update, Alter | No Access |
| Composes | Select | No Access |
| Lists | Select | No Access |
| VisitedBy | Select | No Access |

e) **Internal View:**

**Customer**

**Query 1:** Retrieves all albums ordered by price to the customer

SELECT ALBUMNAME,   
       RATING,   
       PRICE   
FROM   ALBUM   
ORDER BY PRICE;

**Query 2:** Retrieve albums composed by an artist to the customer

SELECT ALBUMNAME,   
       ARTISTNAME   
FROM   ALBUM,   
       ARTIST   
WHERE ARTIST.ARTISTID = ALBUM.ARTISTID   
       AND ARTISTNAME = 'Ar Rehman';

**Query 3:** Retrieve the purchase details of the customer using his CustomerID

SELECT ALBUMNAME,   
       CUSTOMER.CUSTOMERID   
FROM   ALBUM,   
       PURCHASE,   
       CUSTOMER   
WHERE CUSTOMER.CUSTOMERID = PURCHASE.CUSTOMERID   
       AND PURCHASE.PURCHASEID = ALBUM.PURCHASEID   
       AND CUSTOMER.CUSTOMERID = 'C001';

**Store**

Query 4: Insert customer details into database when a purchase is made

INSERT INTO CUSTOMER   
VALUES     ('C008',   
            'kane',   
            3098751234,   
            'kane@gmail.com');

Query 5: Update the price of particular Album Name

UPDATE ALBUM   
SET    PRICE = 15   
WHERE  ALBUMID = 'A008';

**Query 6:** Data Obsolescence Deleting the purchase data before 2007/05/23

SELECT PURCHASEDATE   
FROM   PURCHASE   
WHERE  PURCHASEDATE < '25-JUN-15';

**Query 7:** List all the albums in the store

SELECT ALBUMID, ALBUMNAME, PRICE, RATING  
FROM   ALBUM;

**Query 8:** Lists all the songs arranged based on genre

SELECT \*   
FROM   SONG   
ORDER BY GENRE ASC;

**Query 9:** Retrieve number of albums composed by each artist who are having albums count > 5

SELECT ARTISTNAME,   
       *Count*(ALBUMID) "Albums-Count"   
FROM   ARTIST   
       INNER JOIN ALBUM using(ARTISTID)   
GROUP  BY ARTISTNAME   
HAVING *Count*(ALBUMID) > 1;

**Query 10:** Retrieve the album count of each album in the store

SELECT ALBUMNAME,   
       *Count*(ALBUMID) "Album Count"   
FROM   ALBUM   
GROUP  BY ALBUMNAME;

**File Structures:**

**Table: ALBUM**

* File Structure: Secondary B-tree in Album
* Indexes on AlbumID and Price

**Table: ARTIST**

* File Structure: Hashing in artist on ArtistName
* Index on ArtistID

**Table: CUSTOMER**

* File Structure: Secondary B-tree in Customer
* Indexes on customerid

**Table: PURCHASE**

* File Structure: Secondary B-tree in Purchase
* Index on purchaseid

**Table: SONG**

* Clustered B-tree in Song on Genre

**f) Data Dictionary**

**Indexes:**

* Index has been created on AlbumID and Price in Album Table
* Index on ArtistID in Artist Table
* Index on CustomerId in Customer Table
* Index on purchaseid in Purchase Table

**Views:**

* View created as “AlbumsInfo” to show all the albums in the store to the customer.
* View created as “Ar Rehman” a music artist to retrieve all his albums in the music store.

**TABLES:**

* STORE (**storeid,** storename, storelocation, phonenumber)

* ARTIST (**artistid**, artistname, country, awardswon, albumssold)
* CUSTOMER (**customerid**, customername, mobile, email)
* PURCHASE (**purchaseid**, purchasecost, purchasedate, quantity, customerid)
* SONG (**songid**, songname, language, genre)
* ALBUM (**albumid,** albumname, releasedate, price, rating, purchaseid, artistid, storeid)
* COMPOSES (**artistid**, **songid**)
* LISTS (**albumid**, **songid**)
* VISITEDBY(**storeid**, **customerid**)