DBMS PROJECT:

**MOVIE DATABSE**

**MANAGEMENT**

**SYSTEM**



**Submitted By :-**

KARAN(2016UCO1609)

**ACKNOWLEDGEMENT**

We would like to express our special thanks of gratitude to our teacher **Ms. Savita Yadav** who gave us the golden opportunity to do this wonderful project on the topic MOVIE DATABASE MANAGEMENT SYSTEM, which also helped us in doing a lot of Research and we came to know about so many new things. We are really thankful to her.

Secondly we would also like to thank everyone who helped us a lot in finalizing this project within the limited time frame.

**NETAJI SUBHAS INSTITUTE OF TECHNOLOGY**



**CERTIFICATE**

This is to certify that **KARAN(2016UCO1609)** and **SAURABH GOEL(2016UCO1611),** students of 3rd semester COE(batch 2016-2020) of **NETAJI SUBHAS INSTITUE OF TECHNOLOGY** have successfully completed the project “**MOVIE DATABASE MANAGEMENT SYSTEM”** in SQL/JDBC under the kind guidance of **Ms. SAVITA YADAV**.

Signature

(Ms. Savita Yadav )

**INTRODUCTION**

The system is to manage the data of the movie industry . It automates the task of keeping record of different movies and the actors , directors , singers , songs associated with a movie .

The management system also provides handy user interface for normal users and makes easier for them to perform basic operations on the database such as insert , update , delete .

Using a manual system for achieving this objective is a bit difficult. Not only collecting data from various places a clumsy task but also it leads to redundancy .

Such problems are efficiently being handled by this project.

The overall functioning, schemas, design and various relations are discussed in the following pages.

**TASKS THAT THIS DATABASE PERFORMS:-**

This MOVIE database management system can perform the following tasks.

It Keeps:-

1. MOVIE INFORMATION.
2. ACTORS INFORMATION.
3. SINGERS INFORMATION.
4. SONGS INFORMATION.
5. DIRECTORS INFORMATION.
6. PRODUCTION HOUSE INFORMATION.

UPDATION , INSERTION , DELETION on various tables can be performed easily using this DBMS.

**ENTITIES AND ATTRIBUTES:-**

Details of entities used in the MOVIE DBMS are as follows:-

|  |  |  |  |
| --- | --- | --- | --- |
| S.No. | ENTITY NAME | ATTRIBUTES | TYPE OF ATTRIBUTE |
| 1. | MOVIE | * Title * Year of release * Length(in mins) * Rating * Genre * Quotable Quote * Actor * Singer * Director * Production House | Primary Key  Multi-valued |
| 2. | Production House | * Name * Address * Owner’s name | Primary Key |
| 3. | Songs | * Title * Length(in mins) * Singer * Movie | Primary Key |
| 4. | User | * User\_id * Name | Primary Key |
| 5. | Actor | * DOB * Gender * Pay |  |
| 6. | Director | * DOB * No. of movies |  |
| 7. | Singer | * DOB * Genre |  |

**WEAK ENTITY**

|  |  |  |  |
| --- | --- | --- | --- |
| S.No. | ENTITY NAME | STRONG ENTITY | ATTRIBUTES |
| 1. | Cameo | MOVIE | * Name * Role * DOB |

The concept of weak entity has also been incorporated in this ‘MOVIE DBMS’.

The implementation is done by adding the strong entity’s(MOVIE) primary key in the table of the weak entity(CAMEO) as an attribute.

**RELATIONSHIPS :-**

|  |  |  |
| --- | --- | --- |
| Name of the relationship | PARTICIPATING  ENTITIES | CARDINALITY |
| Acts | Actor  Movie | Many To Many(m:n) |
| Directs | Director  Movie | One To Many(1:n) |
| Sings | Singer  Songs | Many To Many(m:n) |
| Has | Movie  Songs | One to Many(1:n) |
| Produced By | Movie  Production House | Many to One  (n:1) |
| Cameo | Movie  Cameo | One to Many(1:n) |

**CONSTRAINTS APPLIED:-**

1. NOT NULL:-

|  |  |
| --- | --- |
| **ENTITIES** | **ATTRIBUTES (NOT NULL)** |
| MOVIES | * Title * Year Of Release * Length * Rating * Genre |
| PRODUCTION HOUSE | * Name |
| ACTOR | * User\_id * Name * Gender |
| DIRECTOR | * User\_id * No. of movies * Name |
| Singer | * Name * Genre * User\_id |
| Songs | * Title * Singer * Movie |
| Cameo | * Name * Role * Movie\_name |

**2. Primary Key :-**

|  |  |
| --- | --- |
| **ENTITY** | **Primary Key Attribute** |
| Movies | Title |
| Production House | Name |
| Actor | User\_id |
| Director | User\_id , Name |
| Singer | User\_id , Name |
| Songs | Title |

**GENERALISATION AND SPECIALISATION**

The concept of generalization and specialization is also implemented in the database.

Actor,Director and Singer are the specializations of ‘USER’.

USER

Actor

Director

Singer

IS A

For the implementation part, we didn’t create separate table for ‘user’ and added the attributes of user(i.e. user\_id , Name) which are inherited by actor, director and singer in their respective tables .

**NORMALIZATION**:-

**Normalization** is a process of organizing the data in database to avoid data redundancy, insertion anomaly, update anomaly & deletion anomaly.

* FIRST NORMAL FORM:-

For a relation to be in 1NF , the values in each cell must be atomic (i.e. multi-valued attributes are not allowed.)

In table ‘movies’ , genre is a multi-valued attribute.

So to solve this problem we create a separate table which has primary key same as that of ‘movies’ and attribute genre.

We create separate tuple for each genre for a movie.

* SECOND NORMAL FORM:-

A relation is in 2NF if it is in 1NF and every non-prime attribute of the relation is dependent on the whole of every candidate key.

All the relations in this DBMS are in 2NF form.

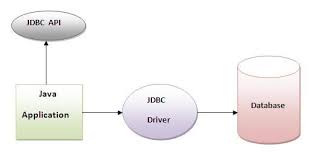
* THIRD NORMAL FORM:-

A relation is in 3NF if it is in [second normal form](https://en.wikipedia.org/wiki/Second_normal_form) , and all the attributes in a table are determined only by the [candidate keys](https://en.wikipedia.org/wiki/Candidate_key) of that relation and not by any non-prime attributes.

All the relations in this database are in 3NF.

**JDBC:-**

**Java Database Connectivity** (**JDBC**) is an [application programming interface](https://en.wikipedia.org/wiki/Application_programming_interface) (API) for the programming language [Java](https://en.wikipedia.org/wiki/Java_(programming_language)), which defines how a client may access a [database](https://en.wikipedia.org/wiki/Database). It is Java based data access technology and used for Java database connectivity. It is part of the [Java Standard Edition](https://en.wikipedia.org/wiki/Java_Standard_Edition) platform, from [Oracle Corporation](https://en.wikipedia.org/wiki/Oracle_Corporation). It provides methods to query and update data in a database, and is oriented towards [relational databases](https://en.wikipedia.org/wiki/Relational_database). A JDBC-to-[ODBC](https://en.wikipedia.org/wiki/ODBC) bridge enables connections to any ODBC-accessible data source in the [Java virtual machine](https://en.wikipedia.org/wiki/Java_virtual_machine) (JVM) host environment.



We used JDBC in making user interface for the Movie DBMS.

It provides easy tools for performing frequent operations like :-

1 . Insertion 2.Deletion 3.Updation 4.View table

QUERIES