THE CELEBSCAPE DATABASE

DATABASE MANAGEMENT SYSTEMS
CT-261

Presented by:

·Musfirah Saleem Shaikh [DT-009]

·Maryam Ashraff [DT-050]

·Daniya Karrar [DT-007]

·Taroob Anwer [DT-023]

Instructor: Sir Rohail Qamar

ABSTRACT

The Actors Booking database is a structured relational database designed to manage and track information about actors, agencies, agents, projects, bookings, reviews, payments, and skills within the entertainment industry. The database includes tables for actors, agencies, agents, projects, bookings, reviews, payments, agency projects, skills, and actor skills.

FEATURES

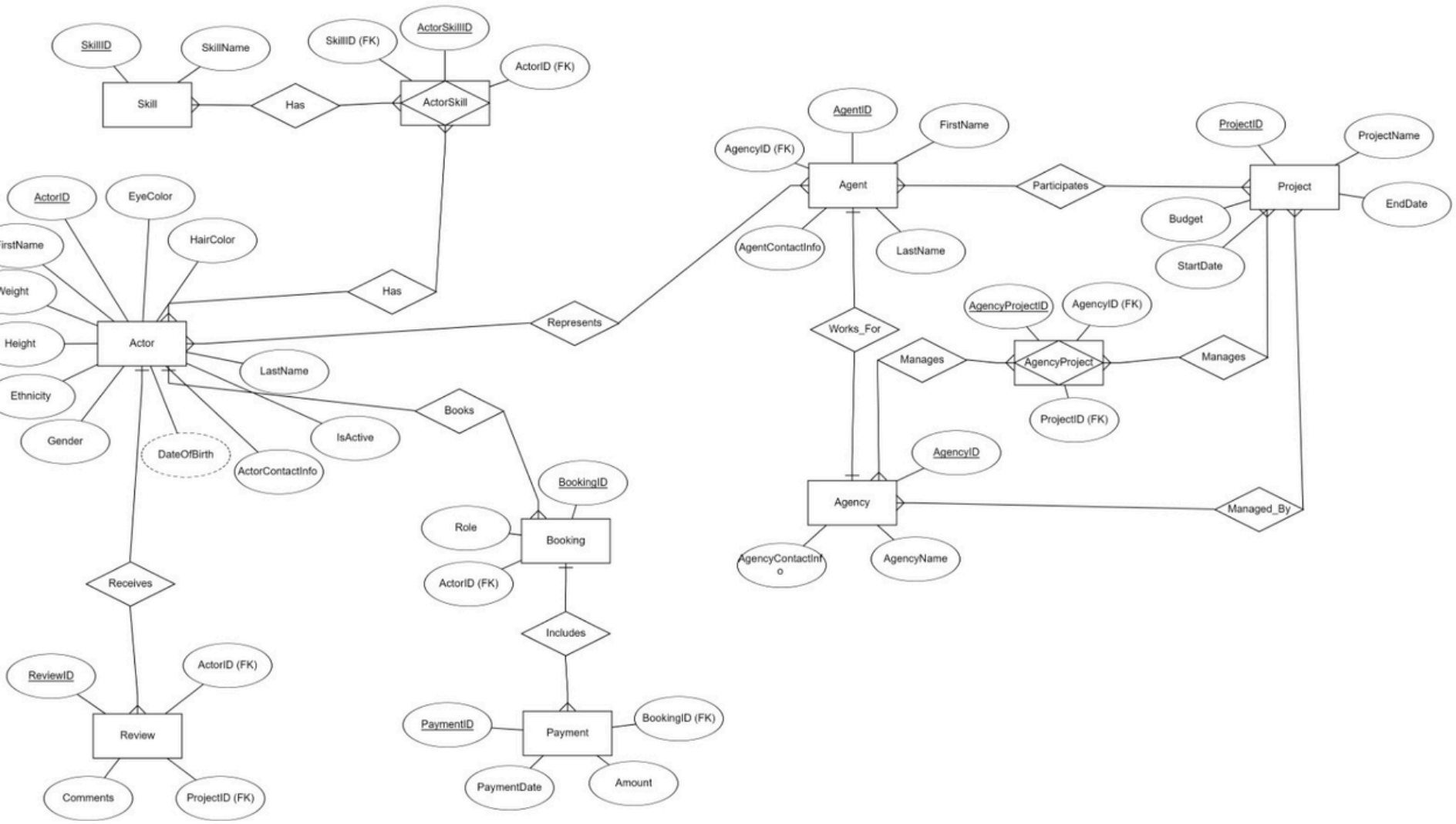
- **Actors Management:** Store detailed information about actors, including personal details, contact information, physical attributes, and activity status.
- **Agency and Agent Management:** Maintain records of agencies and their agents, including contact details and affiliations.
- **Project Management:** Track various projects, including their names, durations, and budgets.

- **Booking Management:** Manage bookings of actors for specific projects, including booking dates and roles.
- **Review System**: Store performance reviews of actors for their roles in projects.
- **Payment Tracking:** Record payments made to actors for their bookings.
- **Agency-Project Relationships:** Track which agencies are involved in which projects.

LIMITATIONS

- **Data Integrity:** The database does not enforce detailed data validation rules beyond primary and foreign key constraints.
- **Scalability:** As the database grows, performance optimization may be required to handle large datasets efficiently.
- **Agent-Actor Relationship:** The current design assumes a direct relationship between actors and agents, which may not reflect the complexity of real-world scenarios where an actor might have multiple agents over time.

ENTITY- RELATIONSHIP DIAGRAM (ERD)



NORMALIZATION TABLES

Example 1: Actor Table
Original Unnormalized Form:
a table that included actors with repetitive information about their skills and agents:

ActorID	FirstName	LastName	ContactInfo	DateOfBirth	SkillName	AgentID	AgentName	AgencyID	AgencyName
1	John	Doe	john@example.com	1980-01-01	Acting	101	Agent Smith	201	Top Talent Agency
1	John	Doe	john@example.com	1980-01-01	Dancing	101	Agent Smith	201	Top Talent Agency
2	Jane	Smith	jane@example.com	1985-02-02	Singing	102	Agent Jones	202	Star Agency

Normalized Form:
1. Actor Table:

```
CREATE TABLE Actor (
    ActorID INT PRIMARY KEY,
    FirstName VARCHAR(50),
    LastName VARCHAR(50),
    ActorContactInfo VARCHAR(100),
    DateOfBirth DATE
);
```

ActorID	FirstName	LastName	ActorContactInfo	DateOfBirth
1	John	Doe	john@example.com	1980-01-01
2	Jane	Smith	jane@example.com	1985-02-02

2. Skill Table:

```
CREATE TABLE Skill (
 SkillID INT PRIMARY KEY,
 SkillName VARCHAR(50)
);
```

SkillID	SkillName
1	Acting
2	Dancing
3	Singing

3. ActorSkill Table:

```
CREATE TABLE ActorSkill (
   ActorSkillID INT PRIMARY KEY,
   ActorID INT,
   SkillID INT,
   FOREIGN KEY (ActorID) REFERENCES
Actor(ActorID),
   FOREIGN KEY (SkillID) REFERENCES
Skill(SkillID)
);
```

ActorSkillID	ActorID	SkillID
1	1	1
2	1	2
3	2	3

4. Agent Table:

```
CREATE TABLE Agent (
   AgentID INT PRIMARY KEY,
   AgentFirstName VARCHAR(50),
   AgentLastName VARCHAR(50),
   AgentContactInfo VARCHAR(100),
   AgencyID INT,
   FOREIGN KEY (AgencyID) REFERENCES
Agency(AgencyID));
```

AgentID	AgentFirstName	AgentLastName	AgentContactInfo	AgencyID	AgencyName
101	Alice	Taylor	jane.newemail@example.com	201	Talent Agency
102	Chris	Anderson	chris.anderson@example.com	202	Star Agency

Example 2: Booking Table

Original Unnormalized Form:

a table that included booking information with repetitive project and actor details:

BookingID	ActorID	FirstName	LastName	ProjectID	ProjectName	BookingDate	Role
1	1	John	Doe	1001	New Movie	2024-12-01	Supporting Actor
2	2	Jane	Smith	1002	TV Show	2024-03-01	Supporting Actor

Normalized Form:

1. **Booking Table:**

```
CREATE TABLE Booking (
  BookingID INT PRIMARY KEY,
  ActorID INT,
  ProjectID INT,
  BookingDate DATE,
  Role VARCHAR(50),
  FOREIGN KEY (ActorID) REFERENCES Actor(ActorID),
  FOREIGN KEY (ProjectID) REFERENCES Project(ProjectID)
);
```

BookingID	ActorID	ProjectID	BookingDate	Role
1	1	1001	2024-02-01	Supporting Actor
2	2	1002	2024-03-01	Supporting Actor

2. Actor Table:

```
CREATE TABLE Actor (
  ActorID INT PRIMARY KEY,
  FirstName VARCHAR(50),
  LastName VARCHAR(50),
 ActorContactInfo VARCHAR(100),
  DateOfBirth DATE
);
```

ActorID	FirstName	LastName	ActorContactInfo	DateOfBirth
1	John	Doe	john@example.com	1980-01-01
2	Jane	Smith	jane@example.com	1985-02-02

3. Project Table:

```
CREATE TABLE Project (
  ProjectID INT PRIMARY KEY,
  ProjectName VARCHAR(100),
  StartDate DATE,
  EndDate DATE,
  Budget DECIMAL(10, 2)
);
```

ProjectID	ProjectName	StartDate	EndDate	Budget
1001	New Movie	2024-01-01	2024-12-31	1200000.00
1002	TV Show	2024-02-01	2024-11-30	2000000.00

Example 3: Review Table

Original Unnormalized Form:

a table that included review information with repetitive actor and project details:

ReviewID	ActorID	ActorName	ProjectID	ProjectName	Rating	Comments
1	1	John Doe	1001	New Movie	5	Amazing performance!
2	2	Jane Smith	1002	TV Show	4	Great job!
1	1	John Doe	1003	Commercial	3	Good effort.

Normalized Form:

1. **Review Table:**

```
CREATE TABLE Review (
  ReviewID INT PRIMARY KEY,
  ActorID INT,
  ProjectID INT,
  Rating INT,
  Comments TEXT,
  FOREIGN KEY (ActorID) REFERENCES Actor(ActorID),
  FOREIGN KEY (ProjectID) REFERENCES Project(ProjectID)
);
```

ReviewID	ActorID	ProjectID	Rating	Comments
1	1	1001	5	Amazing performance!
2	2	1002	4	Great job!
1	1	1003	3	Good effort.

2. Actor Table:

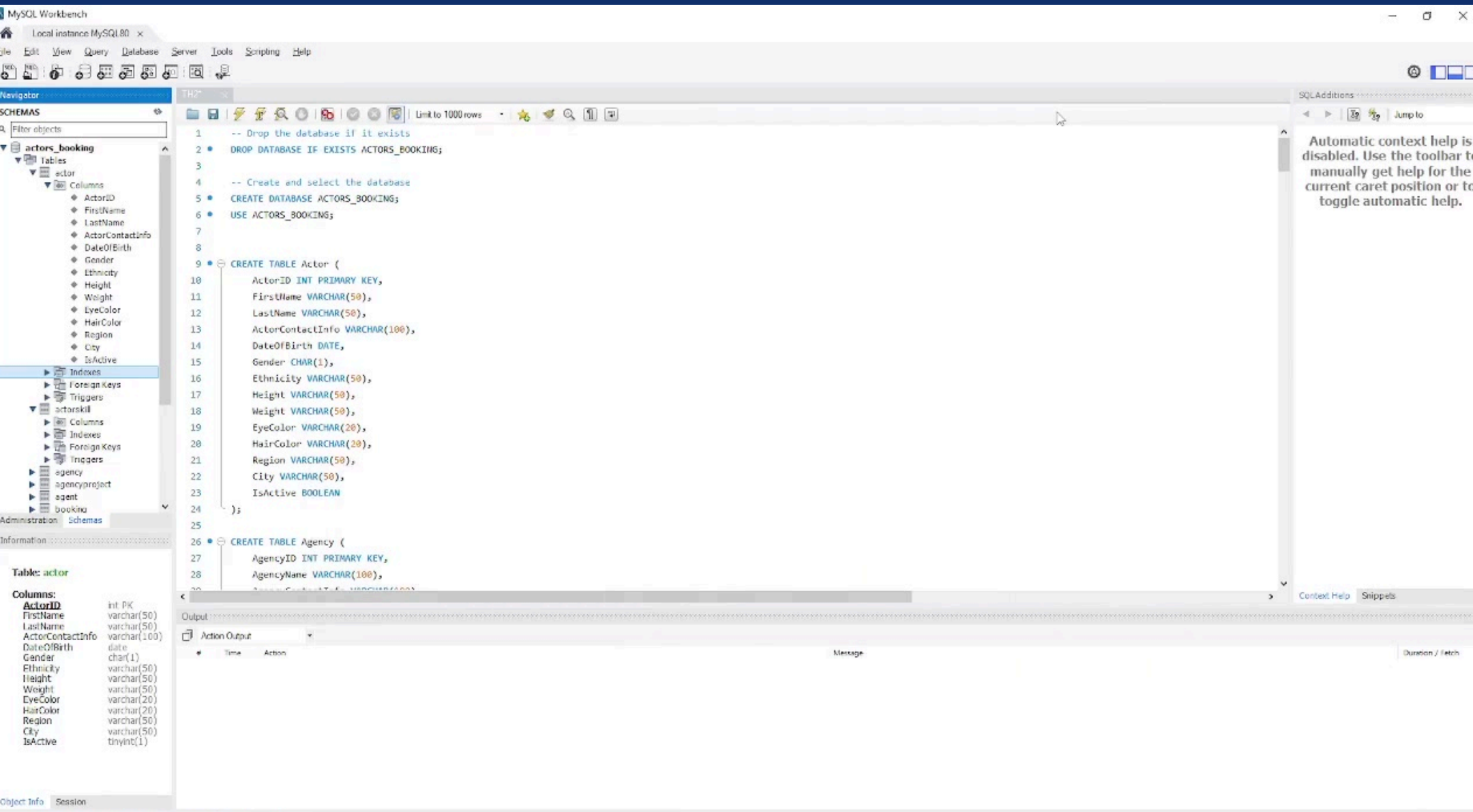
```
CREATE TABLE Actor (
  ActorID INT PRIMARY KEY,
  FirstName VARCHAR(50),
  LastName VARCHAR(50),
  ContactInfo VARCHAR(100),
  DateOfBirth DATE
);
```

ActorID	FirstName	LastName	ActorContactInfo	DateOfBirth
1	John	Doe	john@example.com	1980-01-01
2	Jane	Smith	jane@example.com	1985-02-02

3. Project Table:

```
CREATE TABLE Project (
  ProjectID INT PRIMARY KEY,
  ProjectName VARCHAR(100),
  StartDate DATE,
  EndDate DATE,
  Budget DECIMAL(10, 2)
);
```

ProjectID	ProjectName	StartDate	EndDate	Budget
1001	New Movie	2024-01-01	2024-12-31	1200000.00
1002	TV Show	2024-02-01	2024-11-30	2000000.00


MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
actors_booking
Tables
actor
Columns
ActorID
FirstName
LastName
ActorContactInfo
DateOfBirth
Gender
Ethnicity
Height
Weight
EyeColor
HairColor
Region
City
IsActive
Indexes
Foreign Keys
Triggers
actorskill
Columns
Indexes
Foreign Keys
Triggers
agency
agencyproject
agent
booking
Administration Schemas
Information
Table: actor
Columns:
ActorID int PK
FirstName varchar(50)
LastName varchar(50)
ActorContactInfo varchar(100)
DateOfBirth date
Gender char(1)
Ethnicity varchar(50)
Height varchar(50)
Weight varchar(50)
EyeColor varchar(20)
HairColor varchar(20)
Region varchar(50)
City varchar(50)
IsActive tinyint(1)
Object Info Session
TH2*
Limit to 1000 rows
1 -- Drop the database if it exists
2 DROP DATABASE IF EXISTS ACTORS_BOOKING;
3
4 -- Create and select the database
5 CREATE DATABASE ACTORS_BOOKING;
6 USE ACTORS_BOOKING;
7
8
9 CREATE TABLE Actor (
10 ActorID INT PRIMARY KEY,
11 FirstName VARCHAR(50),
12 LastName VARCHAR(50),
13 ActorContactInfo VARCHAR(100),
14 DateOfBirth DATE,
15 Gender CHAR(1),
16 Ethnicity VARCHAR(50),
17 Height VARCHAR(50),
18 Weight VARCHAR(50),
19 EyeColor VARCHAR(20),
20 HairColor VARCHAR(20),
21 Region VARCHAR(50),
22 City VARCHAR(50),
23 IsActive BOOLEAN
24);
25
26 CREATE TABLE Agency (
27 AgencyID INT PRIMARY KEY,
28 AgencyName VARCHAR(100),
SQL Additions
Jump to
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Context Help Snippets
Output
Action Output
Time Action Message Duration / Fetch
2:20 am
25/06/2024

OUTPUTS

Thank

You