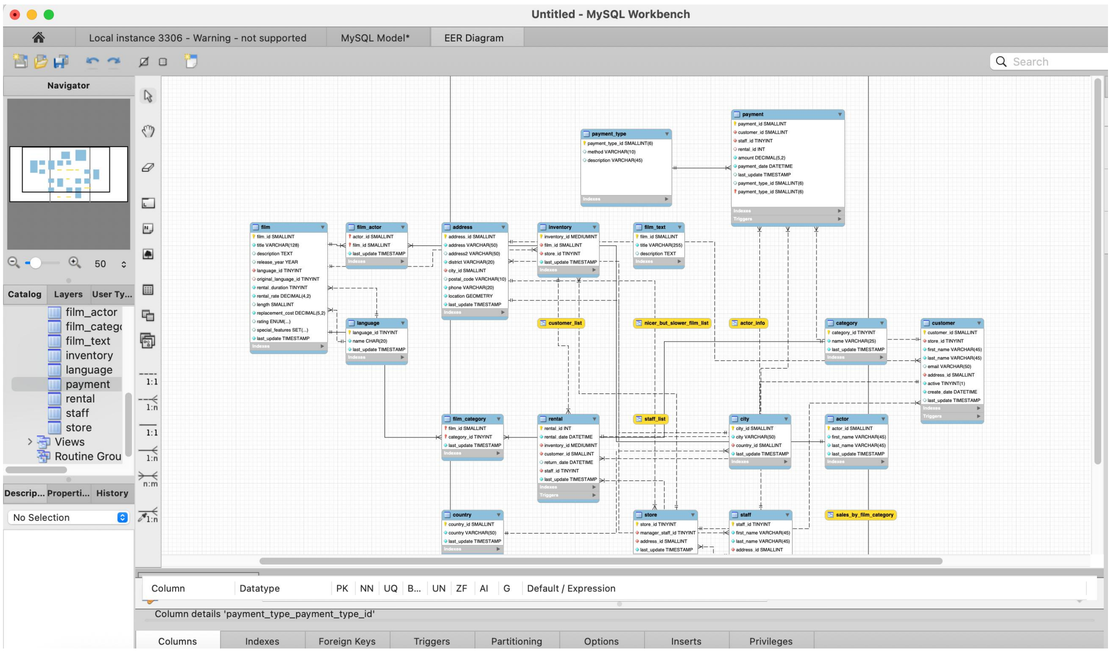
ADSP 31012

PART 1: Relational Modeling



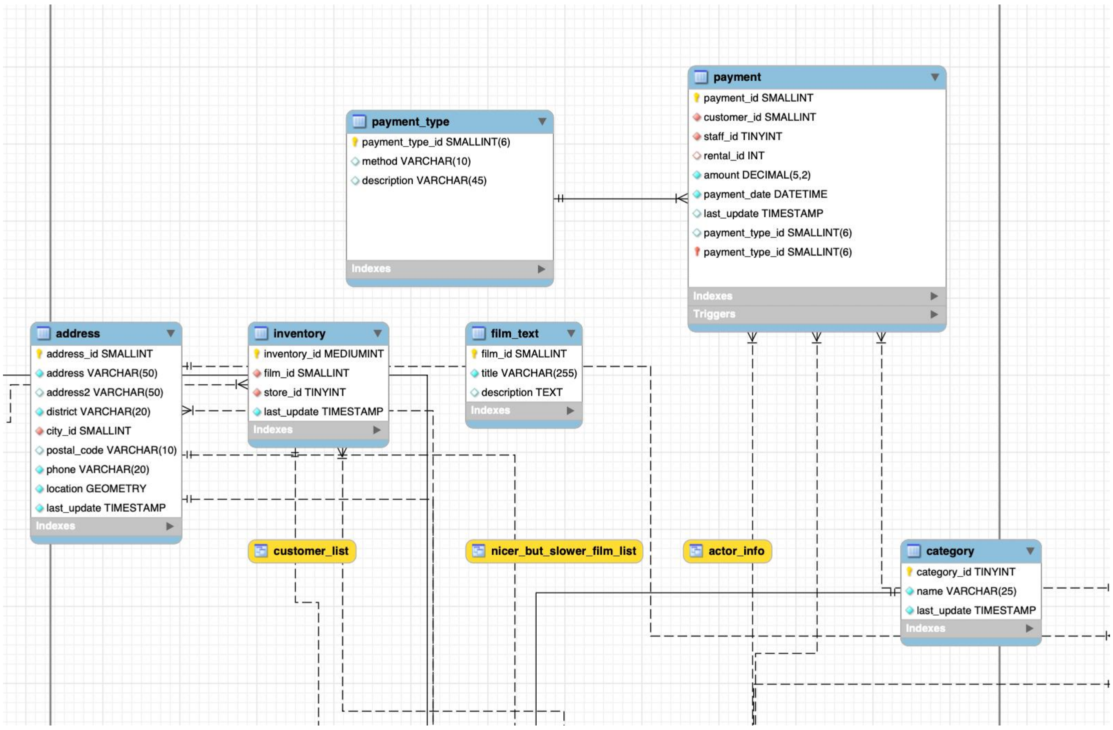
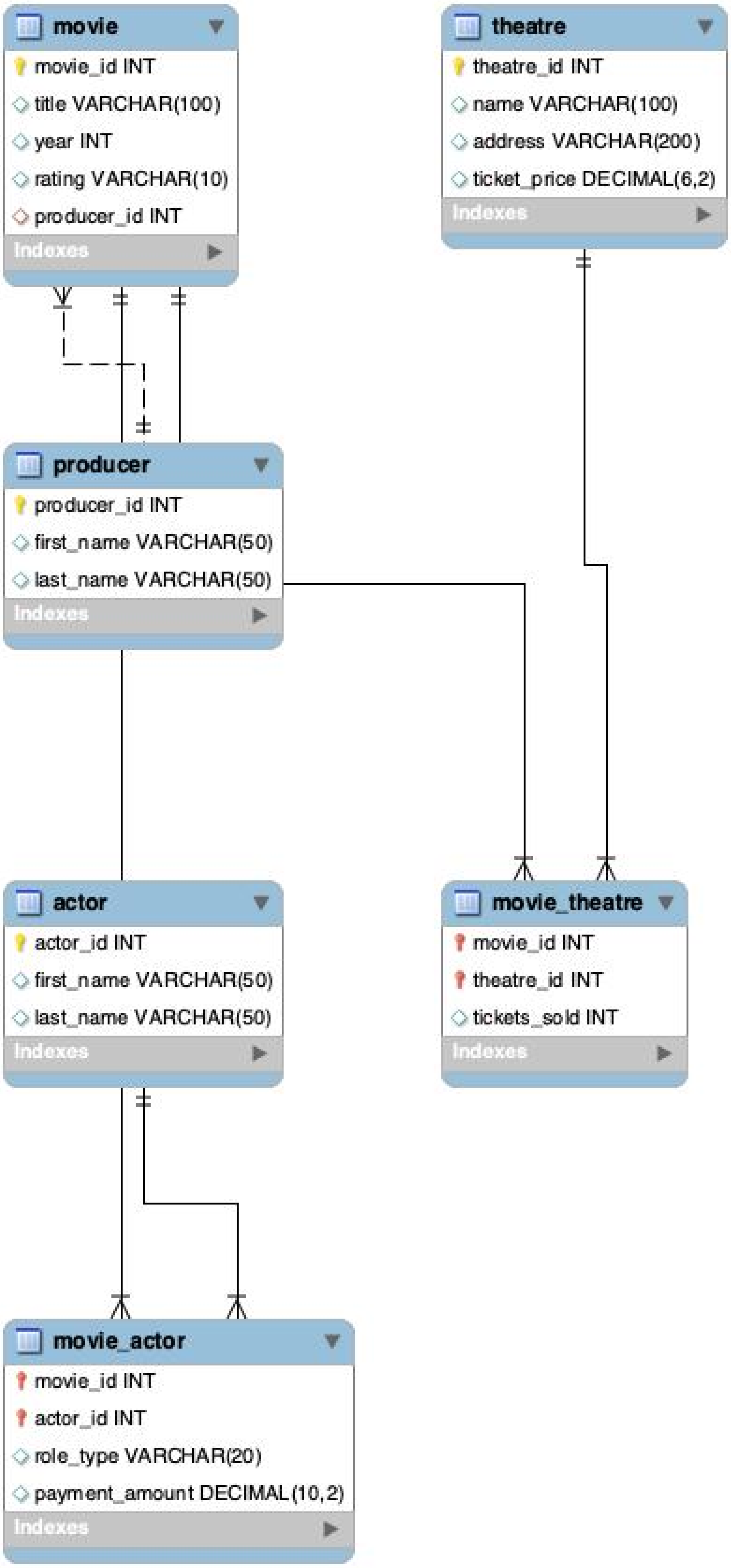


Table Name: Payment

Field (Attributes)	**Primary Key (Y/N)**	**Foreign Key (Y/N)**	**Related Table(s) and Cardinality between tables**
payment_type_id	Y	N	-
customer_id	N	Y	customer (1) : payment (M)
staff_id	N	Y	staff (1) : payment (M)
rental_id	N	Y	rental (1) : payment (M)
amount	N	N	-
payment_date	N	N	-
last_update	N	N	-
payment_type_id	N	Y	payment_type (1) : payment (M)

PART 3: Data Modeling


movie
movie_id INT
title VARCHAR(100)
year INT
rating VARCHAR(10)
producer_id INT
Indexes
theatre
theatre_id INT
name VARCHAR(100)
address VARCHAR(200)
ticket_price DECIMAL(6,2)
Indexes
producer
producer_id INT
first_name VARCHAR(50)
last_name VARCHAR(50)
Indexes
actor
actor_id INT
first_name VARCHAR(50)
last_name VARCHAR(50)
Indexes
movie_theatre
movie_id INT
theatre_id INT
tickets_sold INT
Indexes
movie_actor
movie_id INT
actor_id INT
role_type VARCHAR(20)
payment_amount DECIMAL(10,2)
Indexes

PART 4: Design Document

Database type

MySQL：Relational Database (RDBMS)
ACID-compliant transactions to maintain integrity across interrelated tables

Who will be using this database, and how many people (roughly) do you estimate will be using it?

About 50 Studio Staff who are in charge of Input and manage movie, actor, producer data.
500-1000 Theatre Partners who are in charge of input ticket sales, pricing, theatre info.

20-30 Business Analysts who run reports, analyze sales, actor earnings.

There are about 600-1000 estimated users in total.

Based on how the business will use it, do you think they should implement a distributed database?

Yes, the distrubuted database will be prefered. It allows parallel data processing and better fault tolerance.
What'more, if high-volume real-time data is generated (e.g., ticket sales, actor payments), or the studio operates globally, we need to consider distrubted database more carefully rather than other.

How will you control access to the database? Will all users have the same access?

Through Role-Based Access Control, admin gets full control. Analysts can only read on analytics views and summary data(e.g., tickets sold). Theatre Managers are limited to theatre-related info. External theatres or partners are highly restricted read-only access.

MySQL User Permissions to restrict modification access per table or schema. Not all users will have the same access. Fine-grained permissions will be implemented using SQL GRANT / REVOKE commands or identity-based policies in a cloud RDBMS.

Is there any sensitive personal information in the database that needs to be restricted or specially processed?

While most data is not highly sensitive, some elements like Actor Payments,Theatre Sales Data, Personal Names and Addresses should be handled carefully.
Fields such as first_name, last_name, and address in actor and producer tables are personally identifiable information (PII). So it is commonly recommended that Masking or anonymization for public use, and mantain secure encrypted connections.

Additionally, security measures required Encryption-at-rest and encryption-in-transit (SSL/TLS), data masking in analytics environments, and audit logging for all access to PII.

How will your data model ensure data integrity and protect against anomalies?

For primary keys, each entity (movie, producer, actor, etc.) uses a surrogate key (e.g., movie_id). All tables use integer surrogate keys to ensure uniqueness and decouple identity from personal data.

For foreign keys, it ensures referential integrity between movies, producers, actors, and theatres. Relationships are enforced via foreign keys to prevent orphaned records (e.g., actor without a movie).

In normalization, it should eliminate update, deletion, and insertion anomalies.
As for data validation, we need to use appropriate data types and constraints.