COSC210
Database Mangement Sytems

une University of New England

Practical Assignment

Aims

- Implement a database schema using SQL (in the PostgreSQL) database management system.

The Scenario

In this assignment you will be constructing a relational database using the PostgreSQL DBMS, you will use this database again in the next assginment where you will create an SQL script that builds a series of database views from the DBMS.

You have been tasked by the owners of MovieDirect, a small retailer and online streaming platform, to re-develop their orders and shipments database system. Currently, the system uses four separate spreadsheets to keep track of customers, movies, stock that is currently available and shipments out to the customers. This system is starting to become difficult to use and will prevent MovieDirect from effectively managing its long-term operations in its current state.

Your task is to develop a robust and scalable database solution for the information system that will effectively store MovieDirect's information and provide the capabilities for extracting information to improve sales and management of customers. Once this is complete, you will be tasked with creating a set of user views in the next assignment.

For this assignment, transaction data will be provided, make sure to utilise these to check your solution.

Exercise 1 – Database Design

The first tasks is to develop a database using the PostgreSQL DBMS, which will contain four tables (Customers, Shipments, Movies and stock) linked together using foreign-key relationships. The referential diagram shown below (figure 1.) shows the structure of the database (i.e. how these tables should joined together).

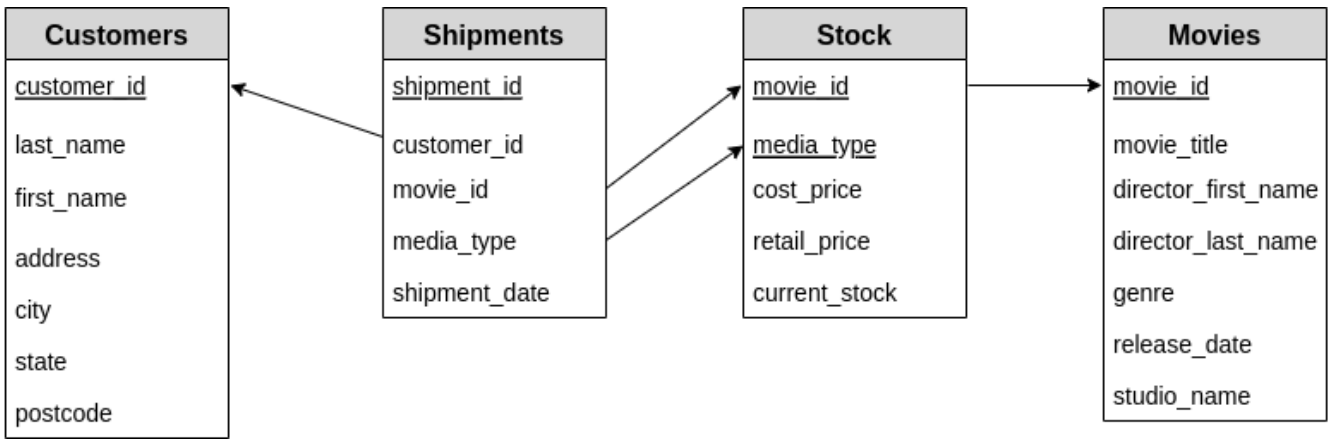

Figure 1. The database schema for MovieDirect.

The SQL definitions for each of the tables will have the following fields:

Customers

- **customer_id** – an integer and primary key for the customers table.
- **last_name** – a character field that allows for up to 50 characters. This field should always have a value.
- **first_name** – a character field that allows for up to 50 characters. This field should always have a value.
- **address** - a character field that allows for up to 200 characters.

- **city** - a character field that allows for up to 50 characters.
- **state** - character field that allows for up to 3 characters. This field should always contain a value from the followoing list: NSW, VIC, QLD, ACT, TAS, NT, SA, WA.
- **postcode** - a character field that allows for up to 8 characters.

Movies

- **movie_id** – an integer and primary key for the "Movies" table.
- **movie_title** - a character field that allows for up to 100 characters. This field should always have a value.
- **director_last_name** – a character field that allows for up to 50 characters. This field should always have a value.
- **director_first_name** – a character field that allows for up to 50 characters. This field should always have a value.
- **genre** – a character field that allows for up to 20 characters that should always contain a value from the following list of genres: Action, Adventure, Comedy, Romance, Science Fiction, Documentary, Drama, Horror.
- **release_date** – a date field.
- **studio_name** – a character field that allows for up to 50 characters.

Stock

- **movie_id** – an integer and partial primary key from the "Movies" table (composite primary key and foriegn key).
- **media_type** – a character field of up to 20 characters that should always contain a value from the following list of mediums: DVD, Blu-Ray and Stream-Media and is part of a composite primary key.
- **cost_price** – A real field that should always have a positive value.
- **retail_price** – A real field that should always have a positive value.
- **current_stock** – A real field that should have a value of zero or more.

Shipments

- **shipment_id** – an integer and primary key for the shipments table.
- **customer_id** – an integer that should always have a value that references the 'customer_id' field in the "Customers" table.
- **movie_id** – an integer that should always have a value that references the 'movie_id' field of the "Stock" table and is a composite foreign key with 'media_type'.
- **media_type** – a character field of up to 20 characters that references 'media_type' from the "Stock" table and is a composite foreign key with 'movie_id'.
- **shipment_date** – a date-type field.

Your task is to construct a set of SQL table definitions to create the Customers, Movies and Shipments tables. Your table definitions should be placed in a single SQL file that can be executed to create the whole database. Be aware of the order in which you create your tables – you can't reference a table that doesn't exist.

To implement the constraints outlined in the description, you will need to use foreign key constraints, CHECK operations, NOT NULL constraints and primary keys.

Once you have completed your tables, fill the tables with data using the script provided. This is also a good way to check that your implementation matches the assignment description. If filling the tables fail, check the error message and then recheck the assignment description.

The SQL script to import data into your completed database is available here:
http://turing.une.edu.au/~cosc210/assignments/a4/MovieDirect_Data.sql

Assignment Submission

You should submit one file for this assignment:

- That is, the .sql file that contains the table definitions (i.e. Your CREATE TABLE ... statements) for your database that you constructed in this exercise. This will be named in the format: `exercise_1_<username>.sql` (e.g.

`exercise_1_esadgro2.sql`)

- Submit your assignment via turing.une.edu's `submit` program.
- **You should submit your assignment to the `p1` assignment.**
- The instructions for `submit` are available here

Marking

Item	Marks
Admin	**- /20**
Files named correctly	- /4
Consistent scripting conventions (names/indent)	- /8
SQL script runs without errors	- /8
Exercise 1	**- /80**
Primary and Foreign Keys Implemented correctly	- /20
CHECK constraints implemented for appropriate attributes	-/20
Appropriate types chosen for all attributes.	- /20
All attributes implemented as described	- /20