COMP1044 Coursework 1

Group 23 – Bread

1/04/2021

Omar Ismail (ID 20311657)

Karim Elbishouty (ID 20314726)

Izzat bin Zainir (ID 20317968)

Ibrahim Ameer Naeem (ID 20314721)

Gabriel Hoh Chao Jie (ID 20311860)

Chong Wen Han (ID 20202183)

Contents

Introduction

In this assignment, we explain why specific data types were used while bringing attention to individual fields that they are used in.

Primary keys and Foreign keys are also denoted with **PK** and **FK** when appropriate.

After, we explain primary and foreign key choices, detail our experiences with the errors some tables had, and how we dealt with those errors. Moreover, some columns names were not appropriate for certain tables. Finally, a violation of normal form was noticed and a fix was suggested.

Data Types and Fields

INTEGER:

Payment	(**PK**) payment_id, (**FK1**) customer_id, (**FK2**) staff_id, (**FK3**) rental_id
Customer	(**PK**) customer_id, (**FK1**) store_id, (**FK2**) address_id
Address	(**PK**) address_id, (**FK1**) city_id, postal_code
City	(**PK**) city_id, (**FK1**) country_id
Country	(**PK**) country_id
Rental	(**PK**) rental_id, (**FK1**) inventory_id, (**FK2**) customer_id, (**FK3**) staff_id
Store	(**PK**) store_id, (**FK1**) manager_staff_id, (**FK2**) address_id
Inventory	(**PK**) inventory_id, (**FK1**) film_id, (**FK2**) store_id,
Staff	(**PK**) staff_id, (**FK1**) address_id, (**FK2**) store_id
Film	(**PK**) film_id, (**FK1**) language_id
Film_actor	(**PK**) actor_id, (**FK1**) film_id
Language	(**PK**) language_id
Film_category	(**PK, FK1**) film_id, (**PK, FK2**) category_id
Film_text	(**PK, FK1**) film_id
Actor	(**PK**) actor_id
Category	(**PK**) category_id

We used INTEGER types for these values since all of them are whole numbers (no decimal values). More limited data types such as SMALLINT, TINYINT were not used in these fields to accommodate for larger values in the future. For Address.postal_code, number of digits was set to 5 since all the values for this field in the provided dataset had a maximum length of 5 digits.

DECIMAL:

Payment	amount(10,2)
Film	rental_date(10,2), replacement_cost(10,2)

A data type capable of storing decimal values is required for these fields. The DECIMAL data type was chosen over FLOAT or DOUBLE data types here since it is more accurate, and high precision is required when dealing with monetary values. Floating-point representations (FLOAT or DOUBLE) are approximations of the exact value due to their nature (limited to 32-bit and 64-bit respectfully as per IEEE-754 standard), while the DECIMAL data type is a fixed-point number (exact value). We used DECIMAL(10, 2) to cap the value to 8 whole number places and 2 decimal places (99999999.99 for example), this is to standardize length for easier maintenance, and limit the value stored to a realistic maximum.

BOOLEAN:

Customer	active
Staff	active

Both these fields store 2 possible values (1 and 0), meaning a data type that stores a binary bit is required. Internally, MySQL treats BOOLEAN as TINYINT(1) where 0 is ‘False’ and 1 is ‘True’. Therefore, BOOLEAN is the most appropriate data type to be used here.

SMALLINT:

Film	original_language_id, rental_duration, length

SMALLINT data type has a range of 0 to 65535 for unsigned values, and since these values will always fall within that range, it is appropriate to use here. This also reduces the size of these fields, contributing towards the efficiency and speed of the database.

VARCHAR:

Customer	first_name(50), last_name(50), email(100)
Address	address(255), address2(255), district(255), phone(15)
City	city(50)
Country	country(50)
Staff	first_name(50), last_name(50), picture(255), email(100), username(50), password(50)
Film	title(255), description(255), rating(5), special_features(255)
Language	name(50)
Film_text	title(255), description(255)
Actor	first_name(50), last_name(50)
Category	name(50)

VARCHAR is a variable length data type to store strings, so we have used it for all values that contain strings. Appropriate lengths have been set depending on the type of data stored. We have included phone number in this data type as well since extensions/international country codes may need to be stored along with the number, and since no arithmetic operations will be done to this number.

For purpose of standardization, we set all shorter fields to a length of 50 characters and all longer fields to a length of 255 characters, with the exception of email fields to accommodate for novel email extensions. Film.rating is 5 characters long because the longest rating category is 5 characters long (NC-17), this is done to maintain referential/data integrity.

YEAR:

Film	release_year

YEAR type was used here since release_year only stores a year, not the whole date or time.

DATETIME:

Payment	payment_date, last_update
Customer	create_date, last_update
Address	last_update
City	last_update
Country	last_update
Rental	rental_date, return_date, last_update
Store	last_update
Inventory	last_update
Staff	last_update
Film	last_update
Film_actor	last_update
Language	last_update
Film_category	last_update
Actor	last_update
Category	last_update

All these values store both a date and a time; therefore, we decided that DATETIME is the most appropriate data type for this field since it can store both DATE and TIME.

Report and Observations

All the tables in this dataset each have a field which stores values unique to each table. Since these values satisfy the properties of uniqueness and minimality, and as such are unique identifiers, we have used these fields as the primary keys of their respective tables. For two of the tables (**Film_actor** and **Film_category**), we have used two fields as a compound primary key instead of introducing a new field to act as primary key.

> These fields are as listed: *payment_id for **Payment**, customer_id for **Customer**, rental_id for **Rental**, country_id for **Country**, city_id for **City**, address_id for **Address**, store_id for **Store**, inventory_id for **Inventory**, staff_id, for **Staff**, film_id for **Film**, actor_id for **Actor**, category_id for **Category**, film_id for **Film_text**, film_id and category_id for **Film_category**, language_id for **Language**, actor_id and film_id for **Film_actor**.*

Some of the unique identifier fields listed above are present in more than one table. These fields are used to interconnect these tables. For any table that has an ID field of another table, we have established said field as the foreign key which links both tables.

> These fields are as listed: *(customer_id, staff_id, rental_id for **Payment**) , (store_id, address_id for **Customer**), (inventory_id, customer_id, staff_id for **Rental**), (country_id for **City**), (city_id for **Address**), (manager_staff_id, address_id for **Store**), (film_id, store_id for **Inventory**), (address_id, store_id for **Staff**), (language_id for **Film**), (actor_id, film_id for **Film_actor**), (film_id, category_id for **Film_category**), (film_id for **Film_text**).*

In the process of completing this task, we discovered some errors in the provided dataset. The data file for **Staff** was corrupted and was filled with garbage values. Additionally, the formatting in some of the data files were not the same as the rest. The **Country** dataset and the **Customer** datasets had semicolon delimiters to separate the data. For the **Country** and **Customer** dataset, we used Notepad's built-in Find-and-Replace feature to convert the semicolons to commas.

Another issue with the dataset was that some of the fields are not in the most appropriate tables. The 'rental_duration' and 'rental_rate' fields are more fitting for the **Rental** table, instead of the **Film** table.

The corrupted **Staff.csv** file was fixed by converting it to a **.xlsx** file using an online converter. We were able to see 2 records of data after fixing the file. It is highly unlikely that a store would have only 2 staff members. This could be 2 managers, in which case that table or fields should be renamed to indicate that the staff mentioned are managers.

Looking at the **Film** table, the special features record has a combination of the terms "Trailers", "Deleted Scenes", "Behind the Scenes", and "Commentaries" for all the records. This column violates 1st normal form since there are more than one features stored in a single column. The column values must be atomic and should be divided. A better way to represent this data is to have separate columns for Trailers, Deleted

Scenes, Behind the Scenes, and Commentaries. These columns can store Boolean values of data. This would put the **Film** table in 3rd normal form.