**Assessment (Dev-test-4)**

1. Please show what MYSQL tables and what classes you would create to accommodate for the following 2 pages form: (You can use any languages, or just pseudocode for the classes)

First page:

First Name: [ ]

Last Name: [ ]

Phone: [ ]

Second page:

Vehicle 1 Vehicle 2 Vehicle 3 Vehicle 4

Year [] Year [] Year [] Year []

Make [] Make [] Make [] Make []

Model [] Model [] Model [] Model []

Answer:

Entity relationship diagram to illustrate the required entities and their relationship with each other:

Diagram

Description automatically generated

figure: Entity Relationship Diagram

Tables with their attributes:

* persons
  + person\_id (integer)
  + first\_name (Varchar)
  + last\_name (Varchar)
  + created\_at (timestamp)
  + updated\_at (timestamp)
* vehicles
  + vehicle\_id (integer)
  + person\_id (Foreign key to person\_id of persons table)
  + year (date)
  + vehicle\_model\_id (Foreign key to vehicle\_model\_id of vehicles\_model table)
  + created\_at (timestamp)
  + updated\_at (timestamp)
* vehicle\_make
  + vehicle\_make\_id (integer)
  + name (Varchar)
  + created\_at (timestamp)
  + updated\_at (timestamp)
* vehicle\_model
  + vehicle\_model\_id (integer)
  + name (Varchar)
  + vehicle\_make\_id (Foreign key to vehicle\_make\_id of vehicles\_make table)

Following tables will be created along with the relationship as shown in Entity relationship diagram.

* Persons

This table will have person’s personal details and it has one to many relationships with vehicles table. A person’s vehicle information will be retrieved from the vehicles table using person id and vehicle\_model\_id.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Person\_id | first\_name | last\_name | phone\_number | created\_at | updated\_at |
| 1 | Elon | Musk | 111-222-3333 | 2022-12-18 16:00:00 | 2022-12-18 16:00:00 |

* vehicles\_make

This table will store vehicle make information such as name, and it has one to many relationships with vehicles\_model table. Example:

|  |  |  |  |
| --- | --- | --- | --- |
| Vehicle\_make\_id | name | created\_at | updated\_at |
| 1 | Hyundai | 2022-12-18 16:00:00 | 2022-12-18 16:00:00 |

* vehicles\_model

This table will store vehicle model information and has a many to one relationship with vehicles\_make table. Example:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Vehicle\_model\_id | name | Vehicle\_make\_id | Created\_at | updated\_at |
| 1 | Kona | 1 | 2022-12-18 16:00:00 | 2022-12-18 16:00:00 |

* Vehicles

This table will have vehicle information and it has many to one relationship with vehicles\_model table. It will get vehicle information such as its make and model from this relation.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Vehicle\_id | year | Person\_id | Vehicle\_model\_id | Created\_at | updated\_at |
| 1 | 2022 | 1 | 1 | 2022-12-18 16:00:00 | 2022-12-18 16:00:00 |

Steps to create tables (Based on Laravel framework):

In Laravel framework, it uses migration technique to create a schema for a table. I would like to provide an example for the reference. This file defines table name, all the properties of entities such as data type, indexes, packages, and its relationship with other table.

Assume, the database connection is already established using default Laravel database connection.

* Create migration file using a command
  + php artisan make:migration table\_name

This command generates a class named table\_name inside the database/migrations folder.

* Update table\_name file as per the table attributes and its data type.
* After modifications, running migration command to update the database as per the migration file created or the table\_name.
  + php artisan migrate

Following model classes will be created based on the above-mentioned tables:

* Person
* Vehicle
* VehicleMake
* VehicleModel

The model class is used to link with its table to perform database specific tasks such as CRUD operations.

For example:

A Person model is linked to persons table of the database. Inside the model, the person attributes such as id, firstName etc. are defined along with its functions as mentioned below but not limited to these.

Setter/Getter functionality: This will help to set or retrieve properties to or from model object.

Create functionality: This will create a resource of person in the Persons table.

Read functionality: this will get a person resource from the database.

Update functionality: This will edit/update resource from the database.

Delete functionality: This will delete a person resource from the database.

2. Now that you have tables and class models created, please explain what MVC classes you would create to make this form and save the form data to the database. (You can use any MVC framework in any language, or just pseudocode to demonstrate the MVC pattern).

Answer:

This program will save form data to the database

* Create a VehicleController class
* Create a vehicle file under view section, and includes the given form
* Create a route file
* Add index method in VehicleController
* Return the vehicle view from index method
* Create a new route inside the route file for the index method
* Create update method inside the VehicleController class
  + Update method accepts form data
  + Update method will validate the form data, accordingly, returns error message if “invalid form data” to the vehicle view else proceeds to the next step
  + Call a function of Person model to check if the person already exists in the database, if true then save person object else create new person object from Person model
  + Create new vehicle object according to the form data
  + Set person object to the person\_id attribute of vehicle object
  + Create new vehicle model object of vehicleModel model using form data
  + Set vehicle model object to the vehicle\_model\_id attribute of vehicle object
  + Call a store method of vehicle model with passing vehicle object to insert new vehicle record in the database
  + The store method of vehicle model will return true if successful else return false
  + Update method returns success message if the return value of store method is true, else return error message otherwise to the vehicle view file
* Create a new route inside the route file for the Update method of VehicleController
* Update method will be requested as an action of form of vehicle view when submitted

3. Now let’s assume you have your program online, then your manager come and tell you to add a parameter to Vehicle, let say the “color” of the car.

Then day 2 he wants to add “length” of the car,

Then day 3 he wants to add “trim” of the car,

Then day 4 he wants to add tire “size” of the car,

He continues to make requests every day to add more fields to the vehicles form.

Is there any way to change your database design so that for any future requests, you do not have to change database but still be able to add more fields and data for each form?

(No change to table structure, but still be able to add data to the tables of your new design)

And how would your MVC classes look like now that you have this new design?

Please simply submit written pseudocode and explanations to your code in your answer.

Answer:

To add new column in a vehicle table when the program is online, we must ensure to implement the changes in a testing server or development server first so that any problem arises during the new update will be fixed before updating the live system server.

As I have followed Laravel MVC framework to create table in MYSQL database, the migration technique has made it easy to add new field. Following steps will be taken for the addition of new field in the Vehicle table:

(Based on the Laravel MVC Framework)

* Create a new migration file and add new parameter “color” with its data type in schema along with default null value
* Migrate the newly created migration file
* Add “color” parameter and its setter/getter functions in Vehicle model.
* Add “color” field in vehicle view file and controller

In this manner, we can add new parameters in the database as per the requirements.

Also, the steps taken to add new field might vary depending on the details of the setup or requirements.

If the live system has huge number of records, then adding new column requires to add this column in each vehicle record and insert a default value in each row if DEFAULT is set otherwise set NULL. In both cases, it is needed to rewrite each row. If the server has a sufficient space, then it won’t be a much problematic. But we always must be careful on possible failures and take appropriate solutions. So, it is possible to have different strategies to add new field depending on the requirements and current state of the system.