Introduction

The transport industry is a vital component of modern society, enabling people and goods to move from one place to another quickly and efficiently. With the growth of online commerce, there is an increasing demand for fast and convenient transport services. To meet this demand, many transportation companies are developing innovative projects that aim to provide customers with easy access to transportation services.

One such project is a transportation management system that allows customers to quickly and easily obtain the car they need in the desired category, from the right manufacturer, through a bank or without it, and write a review of the car while seeing its rating. The system is designed to streamline the entire transportation process, from ordering to delivery, making it easy for customers to get the transport they need when they need it.

The project includes nine entities, including customers, manufacturers, cars,category, purchase, payments, reviews, banks, and employees. Each entity has its own set of attributes that define its properties and characteristics. For example, the customer entity includes attributes such as customer ID, name, email, phone number, and address, while the manufacturer entity includes attributes such as manufacturer ID, model, brand and quantities.

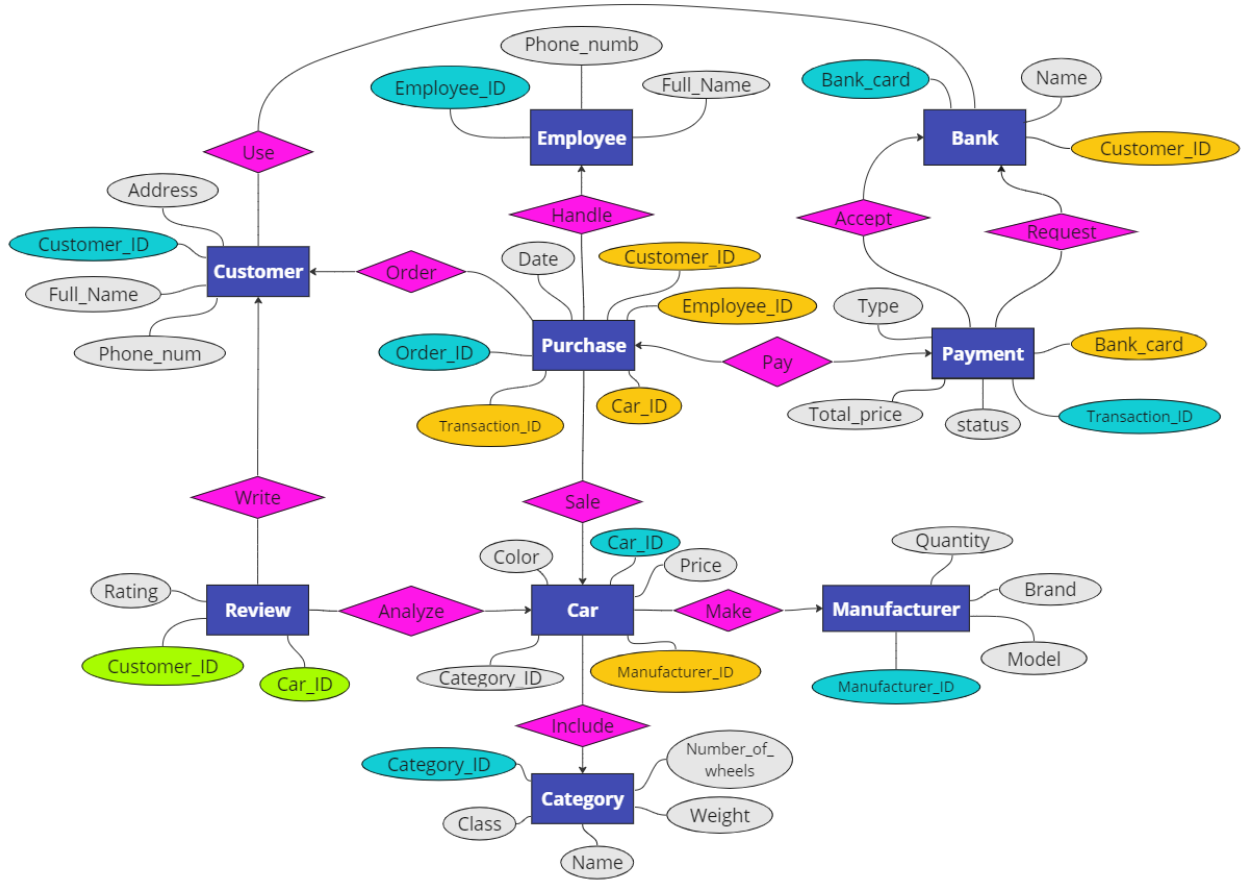
To ensure the efficient management of data, the project is designed using 1, 2, and 3 Normal forms tables. The tables are structured to reduce data redundancy and improve data consistency, ensuring that data is accurately recorded and stored in a way that is easy to retrieve.

In addition to the tables, the project includes a trigger, two procedures, a function, and an exception. The trigger is used to update the rating table automatically whenever a new review is added to the review table. The procedures are used to add a new customer to the customer table and update the delivery date in the order table, respectively. The function is used to calculate the average rating for a car model, and the exception is used to handle errors when adding a new payment to the payment table.

Overall, the project represents an innovative approach to transportation management that leverages the latest technology to provide customers with a fast, convenient, and reliable transport service. With its streamlined approach to transportation management, the project is sure to set new standards in the industry, making it easier for customers to obtain the transport they need quickly and efficiently.

ER diagram

Below you can see an ER diagram that shows all the tables and the relationships between them that are used in this project:



The diagram above shows the entire project. The project comprises of nine entities which consist of customers, manufacturers, cars, category, purchase, payments, reviews, banks, and employees. Each entity possesses a unique set of characteristics that determine its properties and features. For instance, the customer entity has attributes like customer ID, name, email, phone number, and address, while the manufacturer entity has attributes like manufacturer ID, model, brand, and quantities.

Normal Forms

To make sure that data is managed efficiently, the project has been planned using tables in the 1st, 2nd, and 3rd Normal Forms. The tables are organized in a way that minimizes duplication of data and enhances data consistency, thereby guaranteeing that information is recorded accurately and stored in a format that is easy to retrieve.

The below given the rules of Normal Forms that used in this project:

1. 1NF is a property of a relation in a relational database. A relation is in the first normal form if and only if no attribute domain has relations as elements. Or, more informally, no table column can have tables as values.
2. For a table to be in 2NF, there are two requirements:

- The database is in first normal form
- All non-key attributes in the table must be functionally dependent on the entire primary key.

3. A table satisfies 3NF if and only if, for any non-trivial functional dependency $X \rightarrow Y$:

- Either X is a superkey
- Or each attribute in Y is contained in a key.

The tables in our project have been designed to comply with 1st, 2nd, and 3rd Normal Forms, where each row of data contains a single piece of information. Moreover, every non-key attribute in each table is functionally dependent on the primary key. This ensures that the non-key attribute is not transitively dependent on the primary key, leading to better data accuracy and consistency.

Below is an example using the FD diagram of the Car table. You can see that it meets all the requirements of all the normal forms given above:

{Car_id} -> {color, price, category_id, manufacturer_id}

CAR_ID	COLOR	PRICE	CATEGORY_ID	MANUFACTURER_ID
20	brown	500000	6	8
1	White	500000	2	2
2	Black	650000	1	1
3	Gray	500000	3	3

The PL/SQL codes that used in project:

1. Procedure which does group by information

Below is the code of the procedure for grouping items in the table and displaying the number of different data groups by column. In the procedure, you can enter the name of the table (tbl) and the name of the column (col) by which the grouping will be performed. After entering, the Procedure groups objects in the table by this column and thereby leaves only unique values by column, then it counts how many such groups are divided. Thus, the procedure outputs the number of groups (counter) for this column, which is the main task of this procedure:

```
Create or Replace Procedure group_by_Procedure (
    tbl in varchar2,
    col in varchar2,
    counter out number
)  As
   sql_query varchar2(2000);
Begin
   sql_query := 'Select Count(*) From (Select Count(*) From ' || tbl || ' Group by ' || col ||')';
   Execute Immediate sql_query into counter;
   dbms_output.put_line('The number of ' || col || 's in table' || tbl || ' is equal to: ' || counter);
End;
```

Below is the code that helps to test and find out the operability of the procedure:

```
Declare
    counter number;
    tbl varchar2(100);
    col varchar2(100);
Begin
```

```
  tbl:='bank';
  col:='name';
  group_by_Procedure(tbl, col, counter);
End;
```

2. Function which counts the number of records

The code below shows a function that helps calculate the number of records in the table. The name of the table (tbl) to be calculated is entered into the function. Then the number of records is calculated in the function and is output from the function by the number (counter) type:

```
Create or Replace Function count_records_Function (tbl in varchar2)
  return number is
  counter number;
  sql_query varchar2(2000);
Begin
  sql_query := 'Select Count(*) From ' || tbl;
  Execute Immediate sql_query into counter;
  return counter;
End;
```

The following is an example of code that helps to test this function:

```
Declare
   counter number;
   tbl varchar2(100);
Begin
  tbl:='bank';
  counter:= count_records_Function(tbl);
  dbms_output.put_line('The number of records in table ' || tbl || ' is equal to: ' || counter);
```

```
End;
```

3. Procedure which uses SQL%ROWCOUNT to determine the number of rows affected

This procedure code is designed to calculate how many records were affected during the operation. And the operation in this procedure is to delete rows in the table under certain conditions. The table (tbl) and conditions (cond) are entered into the procedure from outside. The procedure outputs the number of rows (counter) that have been deleted from the table:

```
Create or Replace Procedure delete_multiple_rows_ROWCOUNT(
  cond in varchar2,
  tbl in varchar2
)  Is
  counter number;
  sql_query varchar2(2000);
Begin
  sql_query := 'Delete From ' || tbl || ' WHERE ' || cond ;
  Execute Immediate sql_query;
  counter := SQL%ROWCOUNT;
  dbms_output.put_line('Rows affected: ' || counter);
End;
```

Below is an example of deleting rows by conditions from a table using the procedure:

```
Begin
  delete_multiple_rows_ROWCOUNT('rating = 85', 'review');
End;
```

4. Add user-defined exception which disallows to enter title of item (e.g.

book) to be less than 5 characters

Below is a code that helps to create a user defined exception, which is designed to ensure that the newly entered data in the Employee table is correct and meets the requirements. And the requirements for the new data is that the full name of the new employee is at least 5 character:

```
Create or Replace Procedure new_Employee(
   employee_id_1 in number,
   phone_number_1 in number,
   full_name_1 in varchar2
)  As
   sql_query varchar2(2000);
   ex_invalid Exception;
Begin
If Length(full_name_1) < 5 Then
   Raise ex_invalid;
Else
   sql_query := 'Insert into Employee(employee_id, phone_number, full_name)
   values('||employee_id_1||', '"'||phone_number_1||'"', '||full_name_1||')';
   Execute Immediate sql_query;
End If;
   Exception When ex_invalid Then
   dbms_output.put_line('The number of characters in full_name is not enough to be required');
End;
```

The following is an example for testing this exception

```
Begin
   new_Employee(20, 87056650001, 'Asan');
```

```
End;
```

5. Create a trigger before insert on any entity which will show the current number of rows in the table

The code of the trigger that should be triggered before entering a new value into the Review table is given. When the trigger is triggered, the console displays information about how many rows there were before entering a new records:

```
Create or Replace Trigger count_rows_Trigger
Before Insert On Review
For Each Row
Declare
  counter number;
Begin
  Select Count(*) into counter From Review;
  dbms_output.put_line(' The number of rows before inserting new value is equal to: '||counter);
End;
```

Below is an example in which a trigger is triggered and allows you to test this trigger:

```
Begin
  Insert into Review(Car_id, Customer_id, Rating) Values (12, 8, 65);
End;
```

The list of links that can be useful to understanding report more:

1. ER diagram

https://miro.com/welcomeonboard/WjVydk8yQ0xFOTg2T3NwN0dEN0Q4elVCVVp0THdPRDE1RDhXdmNCRVFKcnpxNUlBd3ZsYVZrQXp0aFIyaXJ1dnwzMDc0NDU3MzU3ODMzOTg5MTk0fDI=?share_link_id=953822834164

2. Presentation

https://www.canva.com/design/DAFg7ATy_yQ/uuRA_OmxMAzgQuyW9ToNGw/view?utm_content=DAFg7ATy_yQ&utm_campaign=share_your_design&utm_medium=link&utm_source=shareyourdesignpanel

3. GitHUB

https://github.com/BekaBratan/Car_Shop_DBMS.git

4. YouTUBE video

Endterm DBMS-2 Project