INFORMATICS DEPARTMENT-ARTIFICIAL INTELLIGENCE COURSE

Master's degree in computer science

DATABASE SYSTEMS

Academic year 2022/2023

AutoTechWebapp, management of cars project

Matricola: 777341

Student: Giovanni Magrone

Professors: Michelangelo Ceci, Paolo Mignone.

Index

ADJUST HERE

Conceptual Design

Requirements gathering

First, we will focus on this phase as it's important to gather all the necessary requirements before continuing.
Requirements gathering is the phase in which we want to collect, document, and understand all the requirements and expectations from stakeholders.

AutoTech
AutoTech Database is poised to create a comprehensive repository, delving into the multifaceted realm of automobiles. This expansive resource will be an invaluable asset for automobile enthusiasts, automotive engineers, researchers, buyers, and industry professionals, bridging the gaps and providing insights into various aspects of the automotive world. Within this database, a detailed catalog of vehicles will be available, spanning a wide spectrum including cars, trucks, motorcycles, and more. Each entry will encapsulate crucial information, such as make, model, year, specifications, and pricing. Moreover, comprehensive technical details about vehicles will be included, encompassing engine specifications, fuel efficiency, and horsepower. Additionally, the database will provide insights into vehicle features and technology, shedding light on safety features, entertainment systems, connectivity options, and advanced driver-assistance systems (ADAS). It will facilitate an exploration of the cutting-edge innovations that enhance the driving experience. The historical aspect of vehicles will not be overlooked, as the database will house data on vehicle origins, production history, and significant model changes over the years. This historical context will help users trace the evolution of automobiles. Fuel and energy efficiency will be another integral component of the database. It will offer data on various fuel types, efficiency ratings, hybrid and electric vehicle details, and energy saving technologies, catering to the environmentally conscious and cost-savvy consumers. AutoTech Database aims to be a comprehensive resource that connects information on the past, present, and future of automobiles.

Requirements analysis

This phase is used to get the requirement for the project and the steps of it are five and they are needed to build the glossary that will be used to build the E-R schema:

1) **Choosing right level of abstraction**:

   The **VEHICLE** will be represented by ID, the make, the model, the year, the specification and the price, moreover they could be car, truck, motorcycle or other type of vehicle. Name will be constrained by the company name and the model of the vehicle

The **COMPANY** will represent the builder and the creator of the vehicle and It's represented by a name, an address, country and a description.

The **SPECIFICATIONS** will be represented by ID, engine specifications, and horsepower, moreover, the specifications will refer to a vehicle.

The **TECHNOLOGY** will give insight on the additional features of the vehicle, represented by the ID, the Type(safety features, the entertainment systems, the connectivity options and the ADAS) and the Description.

The **HISTORY** will give insight on how the vehicle, including vehicle data. such as the origins, the production history, and the significant model changes.

The **FUELS** will give insight about how much is fuel efficient the vehicle and will be represent by the information about the name (fuel types), the score (efficiencies rating), the hybrid and electric vehicle details and the energy-saving technology.

2) **Complex phrases restructuring**:

Here no modification is needed, the requirement about the single entities are clear.

3) **Identification of synonyms and homonyms**:

"Automobiles"(line 2) and "vehicles"(line 5) are synonyms, both referring to motorized vehicles for transportation.

"Efficiency"(line 9) and "fuel efficiency"(line 15) are synonyms.

"Historical"(line 12) and "past" (line 19) are synonyms.

"Make" can refer to who built the vehicle, but it can also mean the company who created it (for our purposes we decided to refer to it such as the company).

4) **Refer between terms:**

No changes needed here because thanks to the synonyms and homonyms identification now it's all well explained and referenced.

5) **Glossary:**

CONCEPT	DESCRIPTION	SYNONYM	LINK
VEHICLE	Those are the automobile that will be stored in the DB, they will be represented by their model, company and here, moreover their specifications and price. At the end they could be truck, motorcycle, car or other type of vehicle.	Automobiles	Company, Specifications, Technology, History, Fuels
COMPANY	The company represent the builder and the creator of the vehicle and it will have, a name, an address, country, and a description.	Make	Vehicle
SPECIFICATIONS	The specification will be made by engine specifications and horsepower, moreover, the specifications will refer to a vehicle.		Vehicle
TECHNOLOGY	The technology will specify the various features that will be included in the car, it will be		Vehicle

	represented by the ID, the Type(safety features, the entertainment systems, the connectivity options and the ADAS) and the Description.		
HISTORY	The history will describe how the vehicle changed over the time including data such as the origins, the production history, and the significant model changes.	Past	Vehicle
FUELS	The efficiency will give insight about how much is fuel efficient the Vehicle. Represent by the information about the fuel types, the efficiency rating(score), the hybrid and electric vehicle details and the energy-saving technology.	Fuel efficiency	Vehicle

GROUPING CONCEPTS IN HOMOGENEOUS WAY

PHRASES RELATED TO “VEHICLE”
Within this database, a detailed catalog of vehicles will be available, spanning a wide spectrum including cars, trucks, motorcycles, and more. Each entry will

encapsulate crucial information, such as make, model, year, specifications, and pricing.

PHRASES RELATED TO “COMPANY”
The company will be related to a vehicle, and it will represent the builder and the creator of the vehicle and it’s represented by a name, an address, and a description.

PHRASES RELATED TO “SPECIFICATIONS”
Moreover, comprehensive technical details about vehicles will be included, encompassing engine specifications and horsepower.

PHRASES RELATED TO “TECHNOLOGY”
Additionally, the database will provide insights into vehicle features and technology, shedding light on safety features, entertainment systems, connectivity options, and advanced driver-assistance systems (ADAS). It will facilitate an exploration of the cutting-edge innovations that enhance the driving experience.

PHRASES RELATED TO “HISTORY”
The historical aspect of vehicles will not be overlooked, as the database will house data on vehicle origins, production history, and significant model changes over the years. This historical context will help users trace the evolution of automobiles.

PHRASES RELATED TO “FUELS”
Fuel and energy efficiency will be another integral component of the database. It will offer data on various fuel types, efficiency ratings, hybrid and electric vehicle details, and energy saving technologies, catering to the environmentally conscious and cost-savvy consumers.

DESIGN STRATEGY

We're employing a hybrid approach during the conceptual design phase. We are going to use the requirements gathered during the previously phase. From there, we construct a skeleton schema that outlines the core concepts of the application in high-level view.

Next, we are going to dive deeper into each entity within this schema. We'll refine and elaborate on these entities individually. Then we are going to specify the attributes, relationships, and characteristics of each entity.

At the end of this process, we will get our final E-R model embedding all the components we created.

BUSINESS RULES

1) Each vehicle entry should specify its fuel type, and this should be consistent with the attributes of the "Fuel" entity.

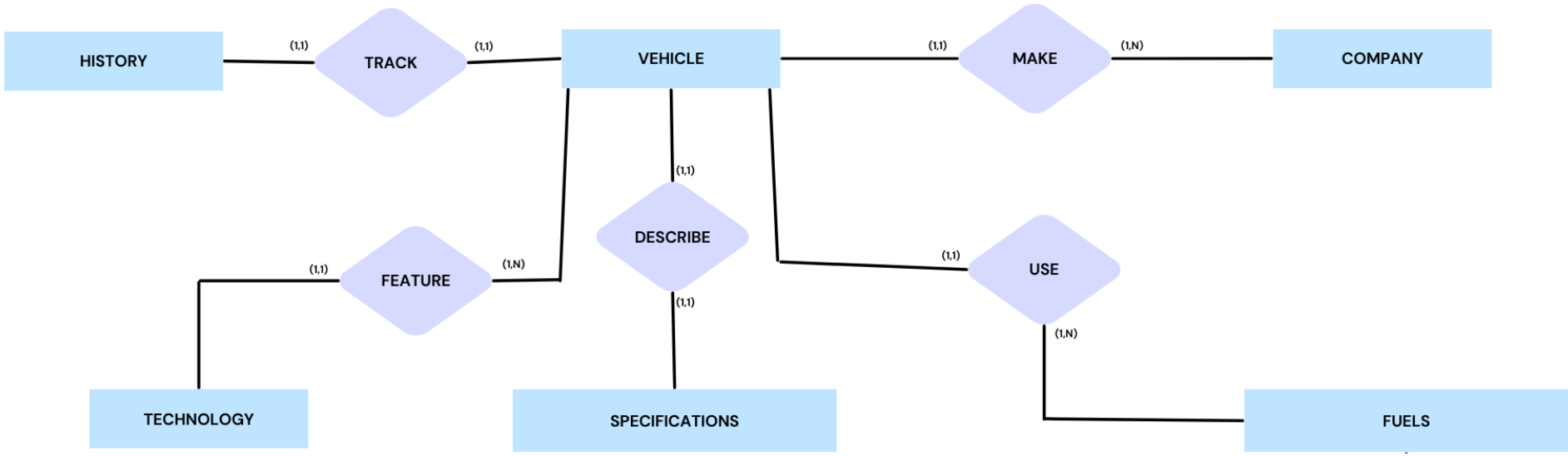
2) Historical data for vehicles, including their origins, production history, and significant model changes over the years, must be recorded to trace the evolution of automobiles. Vehicles cannot be without History

3) Each vehicle could have multiple technology embedded in it, and it should have 1 at least

4) There exist several companies that create automobiles and each of them must have at least one vehicle.

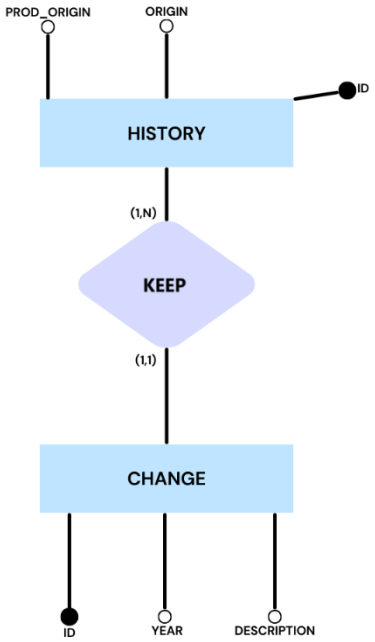
5) Each fuel has a score that represents its efficiency, and it must be recorded taking into account the time in which is made due to the fact that during the time score changes.

SKELETON SCHEMA



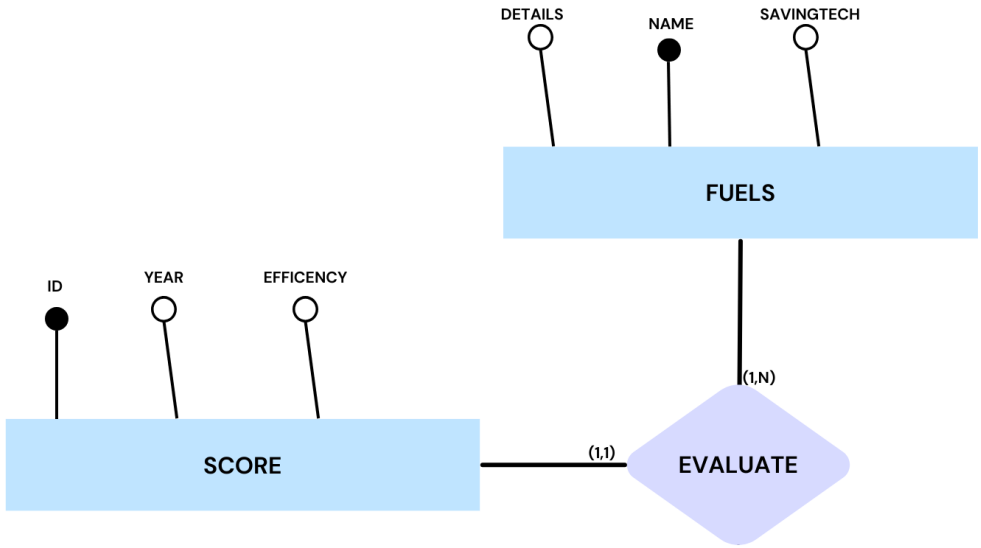
FIRST REFINEMENT OF SKELETON SCHEMA

A vehicle during its life gets different changes so they need to be tracked, so to solve this we add another entity that is the entity Change, where we record the changes during the history of a vehicle.



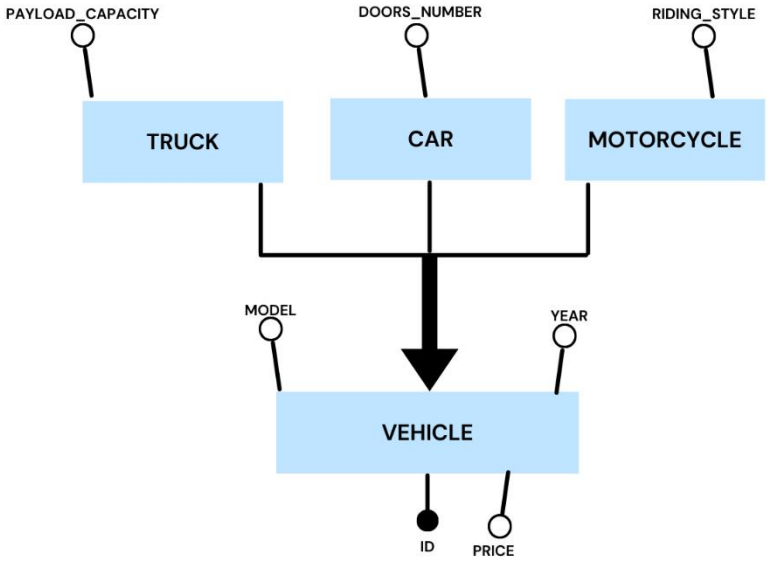
SECOND REFINEMENT OF SKELETON SCHEMA

Since during the time the efficiency of a fuel changes due to technology innovation, it could be needed to store the fuels ratings(score) over the time

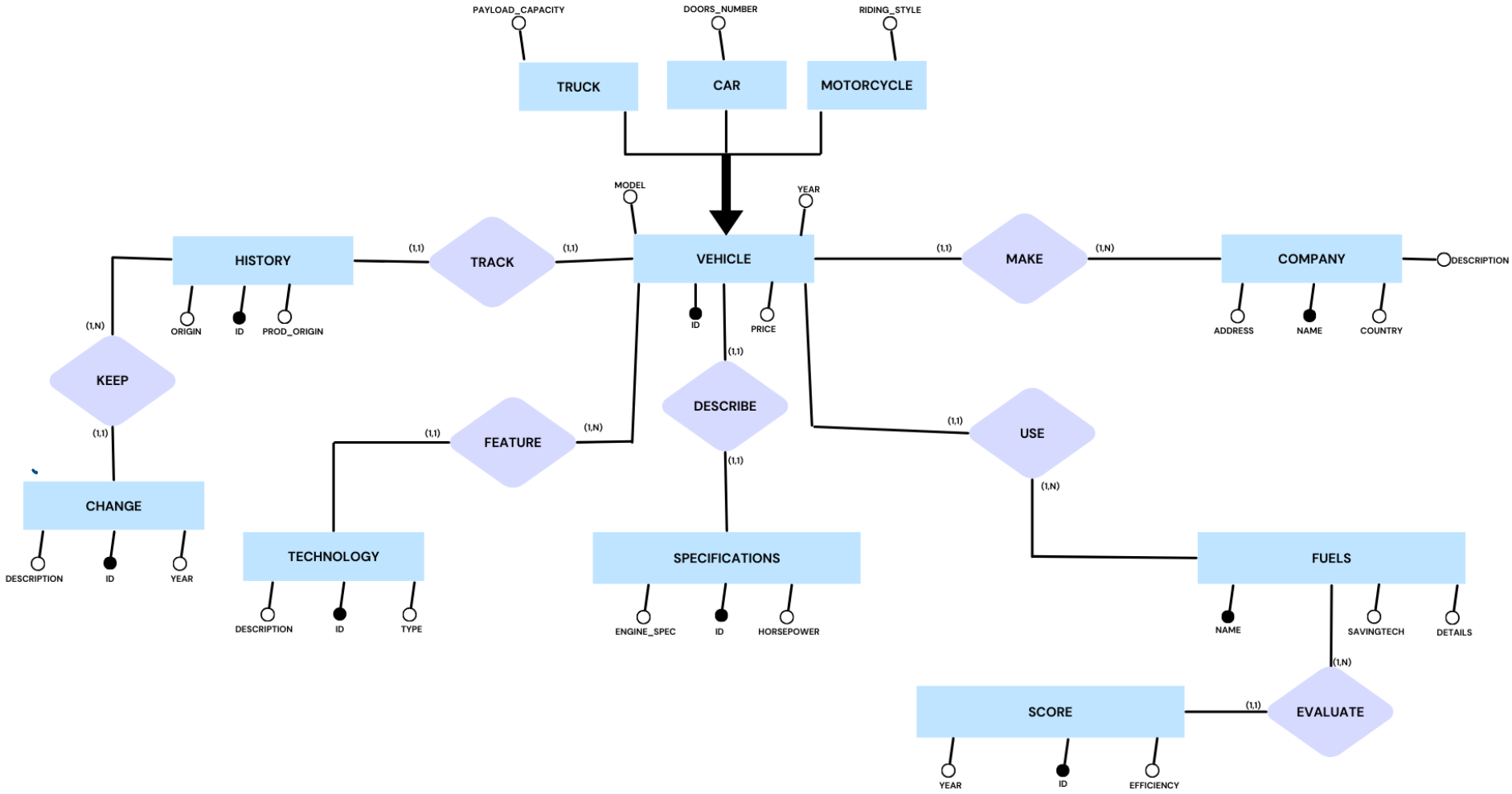


TIRDH REFINEMENT OF SKELETON SCHEMA

Instead of adding an attribute type into the vehicle entities we decided to generalize to insert also additional information useful to identify the different type of vehicle. Where payload is the maximum cargo weight capacity, and riding style could be motocross, sportbike, touring.



FINAL E-R SCHEMA



OPERATIONS TABLE

Operation	DESCRIPTION	FREQUENCY
OP1	Inserting a new vehicle	1 times/day
OP2	Print information about the trucks	1 times/week
OP3	Print information about the newest car with the lowest cost	1 times/day
OP4	Print the history of a specific vehicle model and all its changes made	2 times/month
OP5	Print the fuel efficiency of a particular car model over the past five years	1 times/month
OP6	Print the safety features of the electric vehicles of the last two years	2 times/month

OTHER REQUIREMENTS

In the context of designing a web-based information system, the concept of user authentication is expanded to include a more general concept called "login," which also includes specifying a user's role. These roles define what certain users can and cannot access on the system, helping to organize functions based on their associated roles.

In this system, most of the administrative tasks are typically performed by users with the role of "Admin." However, there are some specific tasks, like all the prints that are mostly interest of the “Client” that will use the platform to see automobiles information.

It will be a good practice to insert an additional Trigger Implement a trigger on the Trucks table to validate that the PayloadCapacity value is within acceptable limits, preventing overloading of trucks, to accept acceptable value for it. Also, a trigger to validate that the number of doors is valid could be a good practice in order to avoid to insert too high value (settings that the minimum number of door is 2 while the max is 6).

ADDITIONAL OPERATIONS CONSIDERED

OPERATIONS	**DESCRIPTION**	**FREQUENCY**
OP7	Print technology of a particular track	10 times/week
OP8	Print all the motorcycle of a particular company	5 times/month
OP9	Insert a new client	5 times/ week
OP10	Print information about a particular Fuel and it's actual efficiency	2 times/week
OP11	Deletion of a client	10 times/month

OPERATIONS AND ROLES OF ALL THE OPERATIONS

OPERATION	**ROLE**
OP1	Admin
OP2	Client
OP3	Client
OP4	Client
OP5	Client
OP6	Client
OP7	Client
OP8	Client
OP9	Client/Admin
OP10	Client
OP11	Admin

LOGIC DESIGN

Volume table

Name	Type	Volume
Vehicle	E	10000
Car	E	7000
Truck	E	1000
Motorcycle	E	2000
Technology	E	200000
History	E	10000
Change	E	Assuming each history has average of 5 changes then we have 50000
Specifications	E	10000
Company	E	If each company has 100 vehicles on average, then we have 100
Fuel	E	10
Score	E	If each fuel has 10 score on average, then we have 100
Track	R	10000
Keep	R	50000
Feature	R	20000
Describe	R	10000
Use	R	10000
Make	R	10000
Evaluate	R	100

Access tables

OPERATION 1

Concept	Type	Access	Type access
Vehicle	Entity	1	W
Company	Entity	1	R
Make	Relationship	1	W
Specifications	Entity	1	W
Describe	Relationship	1	W
Fuels	Entity	1	R
Use	Relationship	1	W
History	Entity	1	W
Track	Relationship	1	W
Change	Entity	1	W
Keep	Relationship	1	W
Technology	Entity	20	W
Feature	Relationship	20	W

TOTAL ACCESSES: For first we must insert the main vehicle detail, then we must insert all the other information such as history that at least should have the actual change in its own history, then we have the technology that on average every vehicle has 20 techs, then we should include specifications and we need to select the company and the fuel used by the vehicle.

Then the total cost is given by 9x2+20x2+2x1(We assume that for writing operations the cost is the double then for reading ones).

So, 18+40+2 = 60 x1(number of times a day) = the total cost is **60 at day**. The operations carried out everyday is not so relevant from a computational cost, then it's acceptable like that.

OPERATION 2

Concept	Type	Access	Type access
Truck	Entity	1	R
Make	Relationship	1	R
Company	Entity	1	R
Use	Relationship	1	R
Fuels	Entity	1	R
Describe	Relationship	1	R
Specifications	Entity	1	R
Feature	Relationship	20	R
Technology	Entity	20	R
Track	Relationship	1	R
History	Entity	1	R

TOTAL ACCESSES:

We should access to all relevant information of the vehicle so we need to access to several different entities the total cost will be given by 1000(number of tracks) x 49(number of access) x 1(number of times at week) = 49.000 access.

The cost of this operations is relevant, we can think to add some redundancy for example inserting **origins, company name and fuel type** in vehicle, the problem anyway will be given for the most by the access at the technology entity, so in order to decrease the total access we also exclude this entity from the query, a user can also access to the technology information by querying then for the single truck.

So, we will introduce redundancy and we will remove technology from the queried entities.

OPERATION 2 AFTER RESTRUCTURING

Concept	Type	Access	Type access
Truck	Entity	1	R
Describe	Relationship	1	R
Specifications	Entity	1	R

As now we can see the access for these operations are 3x1000x1= 3000 access at week that is a more acceptable cost.

OPERATION 3

Concept	Type	Access	Type access
Car	Entity	7000	R

TOTAL ACCESSES: We need to iterate between all the vehicle to see the year and price of the car to get a comparison between all the car and to get the newest car with the less price.

The total cost is given by 7000x1(number of access) x 1(access at day) = 7000 access.

This is not a low-cost operation but anyway is still manageable.

OPERATION 4

Concept	Type	Access	Type access
Vehicle	Entity	1	R
Track	Relationship	1	R
History	Entity	1	R
Keep	Relationship	5	R
Change	Entity	5	R

TOTAL ACCESSES: If we want to access to the History of a vehicle and all its changes made, we need before to access to the history of the vehicle and then for the history we should also print all its change linked considering that every history has 5 changes. So, the total cost will be 13x2(number of times at month) = 26 access at month.

As we can see the cost is not so relevant

OPERATION 5

Concept	Type	Access	Type access
Vehicle	Entity	1	R
Use	Relationship	1	R
Fuels	Entity	1	R
Evaluate	Relationship	5	R
Score	Entity	5	R

TOTAL ACCESSES: If we want to access to the History of evaluation of a fuel of a particular vehicle we need to access to the fuel and then we need to get the score of the last 5 years. So, the cost is 13(cost of access) x 1(times a month) = 13 access at month

As we can see the cost is not so relevant.

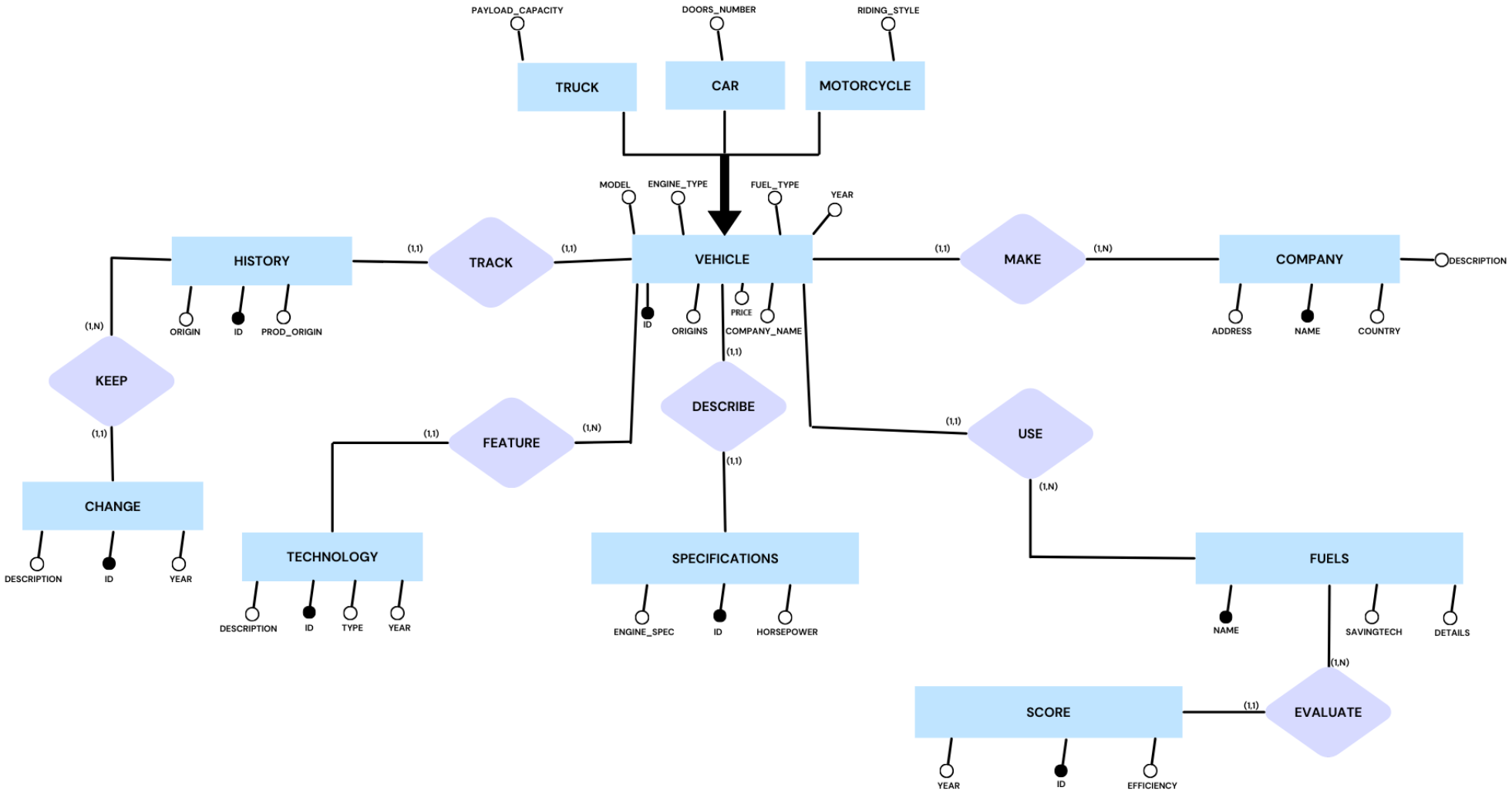
OPERATION 6

For this operations it could be more convenient to restructure the E-R schema to introduce the **year field** in the technologies entity order to have a track of the year of the last two years, moreover for the access could be more convenient to add an additional attribute **engine type(electric, combustion,hybrid)** in the vehicle entity in order to avoid to see the fuel name(type) into the fuel entity and so reducing the number of access. We also assume that the proportion of electric and combustion instances are (50%,50%).

Concept	Type	Access	Type access
Vehicle	Entity	1	R
Feature	Relationship	20	R
Technology	Entity	20	R

TOTAL ACCESS: The total access will be given by 41(number of accesses for a single car) x 3500(number of cars) x2(access at month) = 287.000 access at month. Considering that the operation is carried out twice monthly than the cost is not low but is acceptable somehow.

FINAL SCHEMA AFTER RESTRUCTURING



LOGICAL SCHEMA OBJECT RELATIONAL DATABASE

Type and Tables Definitions

TYPES	TABLES
CREATE TYPE UserTY AS OBJECT (Username varchar2(25), Password varchar2(20), Dateregister date, Type varchar2(10)) NOT FINAL NOT INSTANTIABLE; --CREATION OF THE MAIN USER SUPERTYPE	NOT INSTANTIABLE
CREATE TYPE AdminTY UNDER UserTY (Name_admin varchar2(20), Surname varchar2(20)) FINAL; --CREATION OF THE ADMIN TYPE	CREATE TABLE Admins OF AdminTY (Username NOT NULL primary key, Password NOT NULL, Dateregister NOT NULL, Type NOT NULL, Name_admin NOT NULL, Surname NOT NULL); --CREATION OF TABLE ADMIN
CREATE TYPE ClientTY UNDER UserTY (Name_client varchar2(20), Surname varchar2(20), City varchar2(20),	CREATE TABLE Clients OF ClientTY (Username NOT NULL primary key, Password NOT NULL, Dateregister NOT NULL, Type NOT NULL,

Street varchar2(20), Civicnum varchar2(5), Province varchar2(20), Phone varchar (12)) FINAL; -- CREATION OF THE TYPE CLIENT	Name_client NOT NULL, Surname NOT NULL, City NOT NULL, Street NOT NULL, Civicnum NOT NULL, Province NOT NULL, Phone NOT NULL); --CREATIONS OF THE CLIENTS TABLE
CREATE TYPE ChangeTY AS OBJECT (ID int, Description varchar (200), Year date, Keep REF HistoryTY) FINAL; --CREATING CHANGE TYPE	CREATE TABLE Changes OF ChangeTY (ID NOT NULL primary key, Description NOT NULL, Year NOT NULL); --CREATING CHANGES TABLE
CREATE TYPE CompanyTY AS OBJECT (Name_company varchar2(20), Address varchar2(25), Country varchar2(15), Descripition varchar2(200)) FINAL; --CREATING TYPE OF COMPANIES	CREATE TABLE Companies OF CompanyTY(Name_company NOT NULL primary key, Address NOT NULL, Country NOT NULL, Descripition NOT NULL); --CREATING COMPANIES TABLE
CREATE TYPE ScoreTY AS OBJECT(ID int, Year date,	CREATE TABLE Scores OF ScoreTY(ID NOT NULL primary key, Year NOT NULL,

Efficiency varchar2(20), Evaluate REF FuelTY) FINAL; --CREATION OF SCORE TYPE	Efficiency NOT NULL, CHECK (Efficiency IN ('F', 'D', 'C', 'B', 'A', 'A+'))); --CREATION OF SCORE TABLE
CREATE TYPE FuelTY AS OBJECT (Name_fuel varchar2(20), SavingTech varchar2(50), Details varchar2(200)) FINAL; --CREATING FUELTY TYPE	CREATE TABLE Fuels OF FuelTY (Name_fuel NOT NULL primary key, SavingTech NOT NULL, Details NOT NULL); --CREATING FUEL TABLE
CREATE TYPE SpecificationsTY AS OBJECT (ID int, EngineSpec varchar2(100), HorsePower int) FINAL; --CREATING SPECIFICATIONS TYPE	CREATE TABLE Specifications OF SpecificationsTY(ID NOT NULL primary key, EngineSpec NOT NULL, HorsePower NOT NULL); --CREATE SPECIFICATIONS TABLE
CREATE TYPE HistoryTY AS OBJECT (ID int, Origin varchar2(50), Prod_Origin varchar2(200)) FINAL; --CREATING HISTORY TYPE	CREATE TABLE Histories OF HistoryTY (ID NOT NULL primary key, Origin NOT NULL, Prod_Origin NOT NULL); --CREATING HISTORY TABLE
CREATE TYPE TechnologyTY AS OBJECT (ID int, Description varchar2(200),	CREATE TABLE Technologies OF TechnologyTY (ID NOT NULL primary key, Description NOT NULL,

Type varchar2(30), Year date, Feature REF VehicleTY)FINAL; --CREATE TECH TYPE	Type NOT NULL, Year NOT NULL, CHECK (Type IN ('safety features', 'entertainment systems', 'connectivity options', 'ADAS'))); --CREATE TABLES TECHNOLOGIES
CREATE TYPE VehicleTY AS OBJECT (ID int, Origins varchar2(50), Companyname varchar(20), Year date, FuelType varchar2(20), EngineType varchar2(20), Model varchar2(10), Price int, Track REF HistoryTY, Make REF CompanyTY, Describe REF SpecificationsTY, Use REF FuelTY) NOT FINAL NOT INSTANTIABLE; --CREATION OF THE VEHICLE TYPE	NOT INSTANTIABLE
CREATE TYPE CarTY UNDER VehicleTY(Doorsnumber int)FINAL; -- CREATION OF INERITH CLASS CAR FROM VEHICLE	CREATE TABLE Cars OF CarTY (ID NOT NULL primary key, Origins NOT NULL, Companyname NOT NULL, Year NOT NULL,

	FuelType NOT NULL, EngineType NOT NULL, Model NOT NULL, Price NOT NULL, Doorsnumber NOT NULL, CHECK (EngineType IN ('electric', 'combustion', 'hybrid'))); --CREATION OF TABLE CARS
CREATE TYPE MotorcycleTY UNDER VehicleTY (RidingStyle varchar2(20)) FINAL; --CREATE TYPE MOTORCYCLETY TYPE	CREATE TABLE Motorcycles OF MotorcycleTY (ID NOT NULL primary key, Origins NOT NULL, Companyname NOT NULL, Year NOT NULL, FuelType NOT NULL, EngineType NOT NULL, Model NOT NULL, Price NOT NULL, RidingStyle NOT NULL, CHECK (EngineType IN ('electric', 'combustion', 'hybrid')), CHECK(RidingStyle IN('motocross','sportbike','touring'))); --CREATION ON MOTORCYCLE TABLE
CREATE TYPE TruckTY UNDER VehicleTY (	CREATE TABLE Trucks OF TruckTY (ID NOT NULL primary key,

PayloadCapacity int) FINAL; --CREATION OF MOTORCYCLE TYPE	Origins NOT NULL, Companyname NOT NULL, Year NOT NULL, FuelType NOT NULL, EngineType NOT NULL, Model NOT NULL, Price NOT NULL, PayloadCapacity NOT NULL, CHECK (EngineType IN ('electric', 'combustion', 'hybrid'))); --CREATION OF TRUCKS TABLE

PROCEDURES USED TO POPULATE THE DATABASE

We start from populating the client and admin tables to have different types of users to manage the different characteristics of the webapp then. All the procedure can be then executed with execute procname;

PROCEDURE TO POPULATE THE CLIENT/ADMIN TABLE

```
create or replace PROCEDURE PopulateAdminsClients AS
BEGIN
  FOR i IN 1..10 LOOP -- Populate 10 Admins
    INSERT INTO Admins (Username, Password, Dateregister, Type, name_admin, Surname)
    VALUES (
      SUBSTR(DBMS_RANDOM.STRING('A', 25), 1, 25), -- Random username
      SUBSTR(DBMS_RANDOM.STRING('A', 20), 1, 20),  -- Random password
      SYSDATE - TRUNC(DBMS_RANDOM.VALUE(0, 365)), -- Random registration date
      'Admin',
      SUBSTR(DBMS_RANDOM.STRING('A', 20), 1, 20), -- Random name
      SUBSTR(DBMS_RANDOM.STRING('A', 20), 1, 20)  -- Random surname
    );
```

```
    END LOOP;
FOR j IN 1..200 LOOP -- Populate 200 Clients
        INSERT INTO Clients (Username, Password, Dateregister, Type, name_client, Surname, City, Street, Civicnum, Province, Phone)
        VALUES (
            DBMS_RANDOM.STRING('A', 25), -- Random username
            DBMS_RANDOM.STRING('A', 20), -- Random password
            SYSDATE - TRUNC(DBMS_RANDOM.VALUE(0, 365)), -- Random registration date within the last year
            'Client',
            DBMS_RANDOM.STRING('A', 15), -- Random name
            DBMS_RANDOM.STRING('A', 15), -- Random surname
            DBMS_RANDOM.STRING('A', 10), -- Random city
            DBMS_RANDOM.STRING('A', 10), -- Random street
            DBMS_RANDOM.STRING('N', 5),  -- Random civic number
            DBMS_RANDOM.STRING('A', 10), -- Random province
            DBMS_RANDOM.STRING('N', 10)  -- Random phone
        );
    END LOOP;
    COMMIT;
END;
```

Now we are going to generate the companies table, the only constraint that we wanted to put , was to make the company name generated as CPN1, CPN2 etc…, for debug and style reason.

PROCEDURE TO POPULATE THE COMPANIES TABLE

```
create or replace PROCEDURE PopulateCompanies AS
BEGIN
    FOR i IN 1..100 LOOP
        INSERT INTO Companies (name_company, address, country, descripition)
        VALUES (

            'CPN'|| i,     -- Random company name
            DBMS_RANDOM.STRING('A', 25),    -- Random company address
            DBMS_RANDOM.STRING('A', 15),    -- Random company country
            DBMS_RANDOM.STRING('A', 150)     -- Random company description
```

```
    );
  END LOOP;
  COMMIT;
END;
--PROCEDURE TO POPULATE COMPANY TABLES.
```

Now we are going to populate the table fuels with some of the most already known type of fuels that exist, anyway it's always possible to add new type of upcoming fuels if needed.

PROCEDURE TO POPULATE THE FUELS TABLE

```
  create or replace PROCEDURE PopulateFuelTypes AS
BEGIN
  INSERT INTO Fuels (name_fuel,savingtech,details)
  VALUES (

      'Gasoline',
      'Direct Injection, Turbocharging',
      'Gasoline, commonly known as petrol, is a widely used automotive fuel. Its a liquid fuel derived from crude oil refining.'

  );
  INSERT INTO Fuels (name_fuel,savingtech,details)
  VALUES (

      'Diesel',
      'Common Rail Injection, Turbo Charged Diesel',
      'Diesel fuel is a liquid hydrocarbon fuel commonly used in diesel engines. Its derived from crude oil through refining processes. Diesel engines, known for their efficiency and torque'

  );

  INSERT INTO Fuels (name_fuel,savingtech,details)
  VALUES (
```

```
        'Electric',
        'Regenerative Braking, Electric Motor Efficiency',
        'Electricity is the energy source for electric vehicles (EVs), a clean and sustainable alternative to internal combustion engine vehicles. EVs use electric motors powered by rechargeable batteries.'

    );
    INSERT INTO Fuels (name_fuel,savingtech,details)
    VALUES (

        'Hydrogen',
        'Regenerative Braking, Hydrogen Recirculation',
        'Hydrogen is an alternative fuel used in fuel cell vehicles (FCVs). Its a clean and versatile energy carrier, often considered a promising solution for reducing emissions in transportation'

    );
    INSERT INTO Fuels (name_fuel,savingtech,details)
    VALUES (
        'Ethanol',
        'Flex-Fuel Engines',
        'Ethanol is a biofuel derived from plant-based sources, primarily corn or sugarcane. It is used as an alternative to gasoline, often blended with it.'

    );
    INSERT INTO Fuels (name_fuel,savingtech,details)
    VALUES (
        'Biodiesel',
        'Transesterification,  Biodiesel Blending',
        'Biodiesel is a renewable fuel made from biological sources, such as vegetable oils, animal fats, and used cooking oil.'

    );
    INSERT INTO Fuels (name_fuel,savingtech,details)
    VALUES (

        'CNG',
        'High-Efficiency CNG Engines',
```

```
        'Compressed Natural Gas (CNG) is a clean and environmentally friendly alternative to traditional fuels like gasoline and diesel.'


    );
    INSERT INTO Fuels (name_fuel,savingtech,details)
    VALUES (


        'Methanol',
        'Catalytic Conversion',
        'Methanol, also known as wood alcohol, is a versatile alcohol-based fuel and chemical feedstock. It can be produced from various sources'


    );
    INSERT INTO Fuels (name_fuel,savingtech,details)
    VALUES (


        'Propane',
        'Propane Condensing Boilers',
        'Propane, also known as liquefied petroleum gas (LPG), is a versatile and clean-burning fuel commonly used in heating, cooking, and transportation.'


    );
    INSERT INTO Fuels (name_fuel,savingtech,details)
    VALUES (


        'NG',
        'High-Efficiency Engines',
        ' Natural Gasoline is a liquid hydrocarbon fuel derived from natural gas processing. It shares similarities with both gasoline and natural gas '


    );
      COMMIT;
END;
--CREATION OF POPULATION PROCEDURE OF FUELS
```

PROCEDURE TO POPULATE THE SPECIFICATIONS TABLE

```
create or replace PROCEDURE PopulateSpecifications AS
BEGIN
   FOR i IN 1..10000 LOOP
      INSERT INTO Specifications (ID,enginespec,horsepower)
      VALUES (

            i, -- Sequential ID
            DBMS_RANDOM.STRING('A', 100), -- Random EngineSpec
            ROUND(DBMS_RANDOM.VALUE(100, 1000)) -- Random HorsePower between 100 and 1000

      );
   END LOOP;
   COMMIT;
END;
--CREATE PROCEDURE TO POPULATE SPECIFICATIONS TABLE
```

PROCEDURE TO POPULATE THE HISTORY TABLE

```
create or replace PROCEDURE PopulateHistory AS
BEGIN
   FOR i IN 1..10000 LOOP --
      INSERT INTO Histories (ID,ORIGIN,PROD_ORIGIN)
      VALUES (

            i, -- Sequential ID
            DBMS_RANDOM.STRING('A', 50), -- Random Origin
            DBMS_RANDOM.STRING('A', 200) -- Random Prod_Origin

      );
   END LOOP;
   COMMIT;
END;
--CREATING PROCEDURE TO POPULATE THE HISTORY TABLE
```

The procedure of change table will iterate to make every history to have 5 changes and the date will be in a range between 2000 and 2023.

PROCEDURE TO POPULATE THE CHANGE TABLE

```
CREATE OR REPLACE PROCEDURE PopulateChanges AS
BEGIN
   FOR i IN 1..10000 LOOP
      FOR j IN 1..5 LOOP
         INSERT INTO Changes (ID, DESCRIPTION, YEAR, KEEP)
         VALUES (
            (i - 1) * 5 + j, -- Unique ID
            DBMS_RANDOM.STRING('A', 200), -- Random Description
            TO_DATE('2000-01-01', 'yyyy-mm-dd') + TRUNC(DBMS_RANDOM.VALUE(0, (TO_DATE('2023-12-31', 'yyyy-mm-dd') - TO_DATE('2000-01-01', 'yyyy-mm-dd') + 1))),
            (SELECT REF(h) FROM Histories h WHERE h.ID = i) -- History ref
         );
         END LOOP;
   END LOOP;
   COMMIT;
END;
--CREATION OF PROCEDURE OF POPULATION OF CHANGES TABLE
```

PROCEDURE TO POPULATE THE SCORES TABLE

```
CREATE OR REPLACE PROCEDURE PopulateScores AS
   v_fuel_name VARCHAR2(20);
   v_efficiency VARCHAR2(20);
   v_fuel_ref REF FuelTY;
   sentinel int;
BEGIN
   FOR fuel_type_id IN 1..10 LOOP
      v_fuel_name := CASE fuel_type_id
      WHEN 1 THEN 'Gasoline'
         WHEN 2 THEN 'Diesel'
         WHEN 3 THEN 'Electric'
         WHEN 4 THEN 'Hydrogen'
```

```
        WHEN 5 THEN 'Ethanol'
        WHEN 6 THEN 'Biodiesel'
        WHEN 7 THEN 'CNG'
        WHEN 8 THEN 'Methanol'
        WHEN 9 THEN 'Propane'
        WHEN 10 THEN 'NG'
    END;
    FOR i IN 1..10 LOOP
        -- Generate a random Efficiency
        SELECT DBMS_RANDOM.VALUE(1, 7) INTO sentinel FROM dual;
        CASE
            WHEN sentinel= 1 THEN v_efficiency := 'F';
            WHEN sentinel = 2 THEN v_efficiency := 'D';
            WHEN sentinel = 3 THEN v_efficiency := 'C';
            WHEN sentinel = 4 THEN v_efficiency := 'B';
            WHEN sentinel = 5 THEN v_efficiency := 'A';
            WHEN sentinel = 6 THEN v_efficiency:= 'A+';
            ELSE v_efficiency := 'A+';
        END CASE;



        SELECT REF(f) INTO v_fuel_ref FROM fuels f WHERE f.name_fuel = v_fuel_name;



        INSERT INTO Scores (ID,year,efficiency,evaluate)
        VALUES (


            (fuel_type_id - 1) * 10 + i, -- Random ID
            TO_DATE('2000-01-01', 'yyyy-mm-dd') + TRUNC(DBMS_RANDOM.VALUE(0, (TO_DATE('2023-
12-31', 'yyyy-mm-dd') - TO_DATE('2000-01-01', 'yyyy-mm-dd') + 1))),
            v_efficiency, -- Random Efficiency
            v_fuel_ref


        );
    END LOOP;
```

```
    END LOOP;
    COMMIT;
END;

--PROCEDURE TO POPULATE THE SCORSES TABLE
```

The vehicle will need three different iterations each of them will have the number instances that will be consistent with the data defined in the volumes table.

PROCEDURE TO POPULATE THE CARS

```
create or replace PROCEDURE PopulateCars AS
    v_company_name VARCHAR2(20);
    v_fuel_name VARCHAR2(20);
    v_engine_type VARCHAR2(20);
    v_description_id int;
    v_fuel_ref REF FuelTY;
    v_history_id int;
    sentinel int;
    v_company_ref REF CompanyTY;
    v_history_ref REF HistoryTY;
    v_specifications_ref REF SpecificationsTY;
    v_random_number int;
    sentinel2 float;
BEGIN
FOR i IN 1..7000 LOOP
        -- Randomly select a Company
        -- Generate a random number between 1 and 100
        v_random_number := TRUNC(DBMS_RANDOM.VALUE(1, 101));



        v_company_name := 'CPN' || TO_CHAR(v_random_number);


        -- Randomly select a Fuel
        SELECT DBMS_RANDOM.VALUE(1, 10) INTO sentinel FROM dual;
        v_fuel_name := CASE sentinel
```

```
WHEN 1 THEN 'Gasoline'
   WHEN 2 THEN 'Diesel'
   WHEN 3 THEN 'Electric'
   WHEN 4 THEN 'Hydrogen'
   WHEN 5 THEN 'Ethanol'
   WHEN 6 THEN 'Biodiesel'
   WHEN 7 THEN 'CNG'
   WHEN 8 THEN 'Methanol'
   WHEN 9 THEN 'Propane'
   WHEN 10 THEN 'NG'
END;

-- Randomly select a Specification
v_description_id := i  ;

--Randomly select an History
v_history_id:=i;

-- Randomly select an Engine Type Randomly
SELECT DBMS_RANDOM.VALUE(0, 1) INTO sentinel2 FROM dual;
CASE
   WHEN sentinel2 < 0.33 THEN v_engine_type := 'electric';
   WHEN sentinel2 >= 0.33 AND sentinel < 0.66 THEN v_engine_type := 'combustion';
   ELSE v_engine_type := 'hybrid';
END CASE;

-- Get the REFS
SELECT REF(f) INTO v_fuel_ref FROM fuels f WHERE f.name_fuel = v_fuel_name;
SELECT REF(h) INTO v_history_ref FROM histories h WHERE h.id = v_history_id;
SELECT REF(c) INTO v_company_ref FROM companies c WHERE c.name_company = v_company_name;
SELECT REF(s) INTO v_specifications_ref FROM specifications s WHERE s.id = v_description_id;
```

```
    INSERT INTO Cars (ID, ORIGINS,
COMPANYNAME,YEAR,FUELTYPE,ENGINETYPE,MODEL,PRICE,TRACK,MAKE,DESCRIBE,USE,DOORS
NUMBER)
    VALUES (


        i, -- Random ID
        DBMS_RANDOM.STRING('A', 50), -- Random Origins with up to 50 characters
        v_company_name, -- Randomly selected Company name
        TO_DATE('2000-01-01', 'yyyy-mm-dd') + TRUNC(DBMS_RANDOM.VALUE(0, (TO_DATE('2023-12-
31', 'yyyy-mm-dd') - TO_DATE('2000-01-01', 'yyyy-mm-dd') + 1))),
        v_fuel_name, -- Random Fuel Type (Name)
        v_engine_type, -- Random Engine Type
        DBMS_RANDOM.STRING('A', 10), -- Random Model with up to 10 characters
        DBMS_RANDOM.VALUE(500, 50000),
        v_history_ref,
        v_company_ref,
        v_specifications_ref,
        v_fuel_ref, -- Reference to the randomly selected Fuel Name
        ROUND(DBMS_RANDOM.VALUE(2, 6)) -- Random Doorsnumber between 2 and 5


    );
  END LOOP;
  COMMIT;
END;
--PROCEDURE TO POPULATE CAR TABLE
```

PROCEDURE TO POPULATE THE MOTORCYCLES TABLE

```
        --CREATE PROCEDURE OF POPULATING THE MOTORCYCLES TABLES
create or replace PROCEDURE PopulateMotorcycle AS
  v_company_name VARCHAR2(20);
  v_fuel_name VARCHAR2(20);
  v_engine_type VARCHAR2(20);
  v_style_type VARCHAR2(20);
  v_description_id int;
```

```
    v_fuel_ref REF FuelTY;
    v_history_id int;
    sentinel int;
    v_company_ref REF CompanyTY;
    v_history_ref REF HistoryTY;
    v_specifications_ref REF SpecificationsTY;
    v_random_number int;
    sentinel2 float;
    sentinel3 float;
BEGIN
FOR i IN 1..2000 LOOP
        -- Randomly select a Company
        -- Generate a random number between 1 and 100
        v_random_number := TRUNC(DBMS_RANDOM.VALUE(1, 101));



        v_company_name := 'CPN' || TO_CHAR(v_random_number);

        -- Randomly select a Fuel
        SELECT DBMS_RANDOM.VALUE(1, 10) INTO sentinel FROM dual;
        v_fuel_name := CASE sentinel
        WHEN 1 THEN 'Gasoline'
            WHEN 2 THEN 'Diesel'
            WHEN 3 THEN 'Electric'
            WHEN 4 THEN 'Hydrogen'
            WHEN 5 THEN 'Ethanol'
            WHEN 6 THEN 'Biodiesel'
            WHEN 7 THEN 'CNG'
            WHEN 8 THEN 'Methanol'
            WHEN 9 THEN 'Propane'
            WHEN 10 THEN 'NG'
        END;

        -- Randomly select a Specification
        v_description_id := i  ;
```

```
--Randomly select an History
v_history_id:=i;

-- Randomly select an Engine Type Randomly
SELECT DBMS_RANDOM.VALUE(0, 1) INTO sentinel2 FROM dual;
CASE
   WHEN sentinel2 < 0.33 THEN v_engine_type := 'electric';
   WHEN sentinel2 >= 0.33 AND sentinel2 < 0.66 THEN v_engine_type := 'combustion';
   ELSE v_engine_type := 'hybrid';
END CASE;

-- Randomly select an Style Type Randomly
SELECT DBMS_RANDOM.VALUE(0, 1) INTO sentinel3 FROM dual;
CASE
   WHEN sentinel3 < 0.33 THEN v_style_type := 'motocross';
   WHEN sentinel3 >= 0.33 AND sentinel3 < 0.66 THEN v_style_type := 'sportbike';
   ELSE v_style_type := 'touring';
END CASE;

-- Get the REFS
SELECT REF(f) INTO v_fuel_ref FROM fuels f WHERE f.name_fuel = v_fuel_name;
SELECT REF(h) INTO v_history_ref FROM histories h WHERE h.id = v_history_id;
SELECT REF(c) INTO v_company_ref FROM companies c WHERE c.name_company = v_company_name;
SELECT REF(s) INTO v_specifications_ref FROM specifications s WHERE s.id = v_description_id;



INSERT INTO motorcycles (ID, ORIGINS, COMPANYNAME , YEAR, FUELTYPE,
ENGINETYPE,MODEL,PRICE,TRACK,MAKE,DESCRIBE,USE,RIDINGSTYLE)
VALUES (

    i, -- Random ID
    DBMS_RANDOM.STRING('A', 50), -- Random Origins with up to 50 characters
    v_company_name, -- Randomly selected Company name
```

```
            TO_DATE('2000-01-01', 'yyyy-mm-dd') + TRUNC(DBMS_RANDOM.VALUE(0, (TO_DATE('2023-12-31', 'yyyy-mm-dd') - TO_DATE('2000-01-01', 'yyyy-mm-dd') + 1))),
            v_fuel_name, -- Random Fuel Type (Name)
            v_engine_type, -- Random Engine Type
            DBMS_RANDOM.STRING('A', 10), -- Random Model with up to 10 characters
            DBMS_RANDOM.VALUE(500, 30000),
            v_history_ref,
            v_company_ref,
            v_specifications_ref,
            v_fuel_ref, -- Reference to the randomly selected Fuel Name
            v_style_type -- Random STYLE

        );
    END LOOP;
    COMMIT;
END;
```

PROCEDURE TO POPULATE THE TRUCKS TABLE

```
--CREATE PROCEDURE OF POPULATING THE TRUCKS TABLE
create or replace PROCEDURE PopulateTrucks AS
    v_company_name VARCHAR2(20);
    v_fuel_name VARCHAR2(20);
    v_engine_type VARCHAR2(20);
    v_style_type VARCHAR2(20);
    v_description_id int;
    v_fuel_ref REF FuelTY;
    v_history_id int;
    sentinel int;
    v_company_ref REF CompanyTY;
    v_history_ref REF HistoryTY;
    v_specifications_ref REF SpecificationsTY;
    v_random_number int;
    sentinel2 float;
BEGIN
```

```
FOR i IN 1..1000 LOOP
    -- Randomly select a Company
    -- Generate a random number between 1 and 100
    v_random_number := TRUNC(DBMS_RANDOM.VALUE(1, 101));



    v_company_name := 'CPN' || TO_CHAR(v_random_number);

    -- Randomly select a Fuel
    SELECT DBMS_RANDOM.VALUE(1, 10) INTO sentinel FROM dual;
    v_fuel_name := CASE sentinel
    WHEN 1 THEN 'Gasoline'
       WHEN 2 THEN 'Diesel'
       WHEN 3 THEN 'Electric'
       WHEN 4 THEN 'Hydrogen'
       WHEN 5 THEN 'Ethanol'
       WHEN 6 THEN 'Biodiesel'
       WHEN 7 THEN 'CNG'
       WHEN 8 THEN 'Methanol'
       WHEN 9 THEN 'Propane'
       WHEN 10 THEN 'NG'
    END;

    -- Randomly select a Specification
    v_description_id := i  ;

    --Randomly select an History
    v_history_id:=i;

    -- Randomly select an Engine Type Randomly
    SELECT DBMS_RANDOM.VALUE(0, 1) INTO sentinel2 FROM dual;
    CASE
       WHEN sentinel2 < 0.33 THEN v_engine_type := 'electric';
       WHEN sentinel2 >= 0.33 AND sentinel2 < 0.66 THEN v_engine_type := 'combustion';
       ELSE v_engine_type := 'hybrid';
```

```
    END CASE;



    -- Get the REFS
    SELECT REF(f) INTO v_fuel_ref FROM fuels f WHERE f.name_fuel = v_fuel_name;
    SELECT REF(h) INTO v_history_ref FROM histories h WHERE h.id = v_history_id;
    SELECT REF(c) INTO v_company_ref FROM companies c WHERE c.name_company = v_company_name;
    SELECT REF(s) INTO v_specifications_ref FROM specifications s WHERE s.id = v_description_id;



    INSERT INTO trucks (ID, ORIGINS, COMPANYNAME , YEAR, FUELTYPE,
ENGINETYPE,MODEL,PRICE,TRACK,MAKE,DESCRIBE,USE,payloadcapacity)
    VALUES (


        i, -- Random ID
        DBMS_RANDOM.STRING('A', 50), -- Random Origins with up to 50 characters
        v_company_name, -- Randomly selected Company name
        TO_DATE('2000-01-01', 'yyyy-mm-dd') + TRUNC(DBMS_RANDOM.VALUE(0, (TO_DATE('2023-12-
31', 'yyyy-mm-dd') - TO_DATE('2000-01-01', 'yyyy-mm-dd') + 1))),
        v_fuel_name, -- Random Fuel Type (Name)
        v_engine_type, -- Random Engine Type
        DBMS_RANDOM.STRING('A', 10), -- Random Model with up to 10 characters
        DBMS_RANDOM.VALUE(1000, 80000),
        v_history_ref,
        v_company_ref,
        v_specifications_ref,
        v_fuel_ref, -- Reference to the randomly selected Fuel Name
        DBMS_RANDOM.VALUE(500, 40000) --Random payload capacity between 500kg and 40000 kg


    );
  END LOOP;
  COMMIT;
END;
```

Now we are going to create a procedure to populate the technologies table, having different subtypes, to respect the constraint of volumes table, we have to iterate on each of tables to insert the link between the vehicles and the technologies

PROCEDURE TO POPULATE THE TECHNOLOGIES TABLE

```
--PROCEDURE TO CREATE TECHNOLGIES TABLE
create or replace PROCEDURE PopulateTechnologies AS
    v_truck_id int;
    v_motorcycle_id int;
    v_car_id int;
    v_current_id NUMBER := 1; -- Initialize the current ID
BEGIN
--INSERTING THE TECHNOLOGIES INTO THE TRUCKS TABLE
FOR i IN 1..1000 LOOP
 FOR j IN 1..20 LOOP


       -- Randomly select a Truck
       v_truck_id := i  ;


       INSERT INTO Technologies (id,description,type,year,feature)
       VALUES (


             v_current_id, -- Use the current ID
             DBMS_RANDOM.STRING('A', 200), -- Random Description
             CASE


                WHEN j <= 5 THEN 'safety features'
                WHEN j > 5 AND j <= 10 THEN 'entertainment systems'
                WHEN j > 10 AND j <= 15 THEN 'connectivity options'
                ELSE 'ADAS'
             END, -- Random Type
             TO_DATE('2000-01-01', 'yyyy-mm-dd') + TRUNC(DBMS_RANDOM.VALUE(0, (TO_DATE('2023-12-
31', 'yyyy-mm-dd') - TO_DATE('2000-01-01', 'yyyy-mm-dd') + 1))),
             (SELECT REF(t) FROM Trucks t WHERE t.ID = v_truck_id) -- Randomly selected Vehicle


       );
```

```
        v_current_id := v_current_id + 1; -- Increment the current ID


    END LOOP;
    END LOOP;
--ENDED PART FOR THE TRUCKS


--INSERTING THE TECHNOLOGIES INTO THE MOTORCYCLE TABLE
FOR i IN 1..2000 LOOP
 FOR j IN 1..20 LOOP


        -- Randomly select a MOtorcycle
        v_motorcycle_id := i  ;


        INSERT INTO Technologies (id,description,type,year,feature)
        VALUES (


            v_current_id, -- Use the current ID
            DBMS_RANDOM.STRING('A', 200), -- Random Description
            CASE


               WHEN j <= 5 THEN 'safety features'
               WHEN j > 5 AND j <= 10 THEN 'entertainment systems'
               WHEN j > 10 AND j <= 15 THEN 'connectivity options'
               ELSE 'ADAS'
            END, -- Random Type
            TO_DATE('2000-01-01', 'yyyy-mm-dd') + TRUNC(DBMS_RANDOM.VALUE(0, (TO_DATE('2023-12-
31', 'yyyy-mm-dd') - TO_DATE('2000-01-01', 'yyyy-mm-dd') + 1))),
            (SELECT REF(m) FROM Motorcycles m WHERE m.ID = v_motorcycle_id) -- Randomly selected Vehicle


        );
        v_current_id := v_current_id + 1; -- Increment the current ID


    END LOOP;
    END LOOP;
--ENDED PART FOR THE MOTORCYCLE
```

```
--INSERTING THE LINKING TECHNOLOGIES INTO THE CARS TABLE
FOR i IN 1..7000 LOOP
 FOR j IN 1..20 LOOP

      -- Randomly select a CAR
      v_car_id := i  ;

      INSERT INTO Technologies (id,description,type,year,feature)
      VALUES (

            v_current_id, -- Use the current ID
            DBMS_RANDOM.STRING('A', 200), -- Random Description
            CASE

               WHEN j <= 5 THEN 'safety features'
               WHEN j > 5 AND j <= 10 THEN 'entertainment systems'
               WHEN j > 10 AND j <= 15 THEN 'connectivity options'
               ELSE 'ADAS'
            END, -- Random Type
            TO_DATE('2000-01-01', 'yyyy-mm-dd') + TRUNC(DBMS_RANDOM.VALUE(0, (TO_DATE('2023-12-
31', 'yyyy-mm-dd') - TO_DATE('2000-01-01', 'yyyy-mm-dd') + 1))),
            (SELECT REF(c) FROM Cars c WHERE c.ID = v_car_id) -- Randomly selected Vehicle


      );
      v_current_id := v_current_id + 1; -- Increment the current ID


   END LOOP;
   END LOOP;
--ENDED PART FOR THE MOTORCYCLE
   COMMIT;
END;
```

TRIGGERS AND SEQUENCES DEFINITIONS

Triggers to implement sequence to autoincrement index of values into the tables

TABLE	SEQUENCE	TRIGGER

Specifications	CREATE SEQUENCE specifications_seq
	START WITH 1
	INCREMENT BY 1
	NOCACHE
	NOCYCLE;
		CREATE OR REPLACE TRIGGER specifications_autoincrement
		BEFORE INSERT ON Specifications
		FOR EACH ROW
		BEGIN
		SELECT specifications_seq.NEXTVAL
		INTO :new.ID
		FROM dual;
		END;
		/

Technologies	-- Create a sequence for Technology
	CREATE SEQUENCE Tech_ID_Seq
	START WITH 200001
	INCREMENT BY 1
	NOCACHE
	NOCYCLE;
		-- Create a trigger to auto-increment the ID in Technology
		CREATE OR REPLACE TRIGGER Tech_BI
		BEFORE INSERT ON Technologies
		FOR EACH ROW
		BEGIN
		SELECT Tech_ID_Seq.NEXTVAL
		INTO :NEW.ID
		FROM DUAL;
		END;
		/

Histories	-- Create a sequence for Histories
	CREATE SEQUENCE History_ID_Seq
	START WITH 10001
	INCREMENT BY 1
	NOCACHE
	NOCYCLE;
		-- Create a trigger to auto-increment the ID in Histories
		CREATE OR REPLACE TRIGGER History_BI
		BEFORE INSERT ON Histories
		FOR EACH ROW
		BEGIN
		SELECT History_ID_Seq.NEXTVAL
		INTO :NEW.ID
		FROM DUAL;
		END;
		/

Changes	-- Create a sequence for Changes CREATE SEQUENCE Change_ID_Seq START WITH 50001 INCREMENT BY 1 NOCACHE NOCYCLE;	-- Create a trigger to auto-increment the ID in Changes CREATE OR REPLACE TRIGGER Change_BI BEFORE INSERT ON Changes FOR EACH ROW BEGIN SELECT Change_ID_Seq.NEXTVAL

		INTO :NEW.ID FROM DUAL; END; /

Scores	-- Create a sequence for Scores
	CREATE SEQUENCE Score_ID_Seq
	START WITH 101
	INCREMENT BY 1
	NOCACHE
	NOCYCLE;
		-- Create a trigger to auto-increment the ID in Scores
		CREATE OR REPLACE TRIGGER Score_BI
		BEFORE INSERT ON Scores
		FOR EACH ROW
		BEGIN
		SELECT Score_ID_Seq.NEXTVAL
		INTO :NEW.ID
		FROM DUAL;
		END;
		/

Trucks	-- Create a sequence for Trucks
	CREATE SEQUENCE Truck_ID_Seq
	START WITH 1001
	INCREMENT BY 1
	NOCACHE
	NOCYCLE;
		-- Create a trigger to auto-increment the ID in Trucks
		CREATE OR REPLACE TRIGGER Truck_BI
		BEFORE INSERT ON Trucks
		FOR EACH ROW
		BEGIN
		SELECT Truck_ID_Seq.NEXTVAL
		INTO :NEW.ID
		FROM DUAL;
		END;
		/

Motorcycles	-- Create a sequence for Motorcycles
	CREATE SEQUENCE Motorcycle_ID_Seq
	START WITH 2001
	INCREMENT BY 1
	NOCACHE
	NOCYCLE;
		-- Create a trigger to auto-increment the ID in Motorcycles
		CREATE OR REPLACE TRIGGER Motorcycle_BI
		BEFORE INSERT ON Motorcycles
		FOR EACH ROW
		BEGIN
		SELECT Motorcycle_ID_Seq.NEXTVAL
		INTO :NEW.ID
		FROM DUAL;
		END;
		/

Cars	-- Create a sequence for Cars CREATE SEQUENCE Car_ID_Seq START WITH 7001 INCREMENT BY 1 NOCACHE NOCYCLE;	-- Create a trigger to auto-increment the ID in Cars CREATE OR REPLACE TRIGGER Car_BI BEFORE INSERT ON Cars FOR EACH ROW BEGIN SELECT Car_ID_Seq.NEXTVAL INTO :NEW.ID

		FROM DUAL; END; /

Now we are going to implement two triggers to check the payload capacity and the doors number to be sure that is consistent.

TABLE	TRIGGER

CHECK CAR'S NUMBER OF DORS	CREATE OR REPLACE TRIGGER CheckCarDoors
	BEFORE INSERT ON Cars
	FOR EACH ROW
	BEGIN
	IF :NEW.Doorsnumber < 2 OR :NEW.Doorsnumber > 6 THEN
	RAISE_APPLICATION_ERROR(-20001, 'Number of doors must be between 2 and 6.');
	END IF;
	END;
	/

CHECK TRUCK'S PAYLOAD CAPACITY RANGE	CREATE OR REPLACE TRIGGER CheckTruckPayload
	BEFORE INSERT ON Trucks
	FOR EACH ROW
	BEGIN
	IF :NEW.payloadcapacity < 1000 OR :NEW.payloadcapacity > 50000 THEN
	RAISE_APPLICATION_ERROR(-20001, 'Payload capacity must be between 1000 and 50000.');
	END IF;
	END;
	/

PL/SQL Oracle operations and procedures used in the project

For the select operations we decided to store them as Procedures that stores the results into a temporary table to get them after it's executed. A more direct and different approach will be used in the webApp where the code will be for the views ones directly executed

Operation Procedure	Execution	Code

OP1
Admin
	--EXAMPLE OF INSERTING A CAR
	BEGIN
	INSERTCAR (
	p_origin => 'USA',
	p_production_origin => 'Factory A',
	p_change_description => 'Upgrade engine',
	p_engine_spec => 'Hybrid Engine',
	p_horsepower => 500,
	p_fuel_type => 'Gasoline', --
	p_engine_type => 'hybrid', --HERE THERE ARE CONTRAINT --
	p_model => 'HybridOD',
	p_price => 35000,
	p_company_name => 'CPN1', --COMPANY NAME BETWEEN COMPANIES --
	p_selected_fuel => 'Gasoline', --SAME OF FUEL TYPE
	number_doors => 4, --
	p_safety_feature_description => 'Advanced Safety Features',
	p_feature_type => 'safety features' --
	);
	END;
		create or replace PROCEDURE INSERTCAR (
		p_origin VARCHAR2,
		p_production_origin VARCHAR2,
		p_change_description VARCHAR2,
		p_engine_spec VARCHAR2,
		p_horsepower int,
		p_fuel_type VARCHAR2,
		p_engine_type VARCHAR2,
		p_model VARCHAR2,
		p_price int,
		p_company_name VARCHAR2,
		p_selected_fuel VARCHAR2,
		number_doors int,
		p_safety_feature_description VARCHAR2,
		p_feature_type VARCHAR2
		) AS
		-- Declare variables for the new records
		v_history_id int;
		v_change_id int;
		v_specification_id int;
		v_car_id int;
		v_technology_id int;
		v_history_ref REF HIstoryTY;
		v_spec_ref REF SpecificationsTY;
		v_company_ref REF CompanyTY;
		v_fuel_ref REF FuelTY;
		v_car_ref REF CarTY;
		
		BEGIN
		-- Insert a new History record
		INSERT INTO Histories (Origin, Prod_Origin)
		VALUES (p_origin, p_production_origin)
		RETURNING ID INTO v_history_id;

```
    --Retrieve an history ref

    SELECT REF(h) INTO v_history_ref FROM Histories h
WHERE h.ID = v_history_id;



    -- Insert a new Change record linked to the new History

    INSERT INTO Changes (Description, Year, Keep)

    VALUES ( p_change_description,
SYSDATE,v_history_ref)

    RETURNING ID INTO v_change_id;



     -- Insert a new Specifications record

    INSERT INTO Specifications ( EngineSpec,
HorsePower)

    VALUES ( p_engine_spec, p_horsepower)

    RETURNING ID INTO v_specification_id;




    --Retrieve a specification ref

    SELECT REF(s) INTO v_spec_ref FROM Specifications
s WHERE s.ID = v_specification_id;



    --Retrieve a fuel ref

    SELECT REF(f) INTO v_fuel_ref FROM Fuels f WHERE
f.Name_fuel = p_selected_fuel;



    --Retrieve a company ref

    SELECT REF(c) INTO v_company_ref FROM
Companies c WHERE c.Name_company =
p_company_name;



    -- Insert a new Car record linked to the new History,
Specifications, Company, and Fuel

    INSERT INTO Cars ( Origins, Companyname, Year,
FuelType, EngineType, Model, Price, track, make,
describe, use, doorsnumber)

    VALUES ( p_origin, p_company_name, SYSDATE,
p_fuel_type, p_engine_type, p_model, p_price,
v_history_ref , v_company_ref
,v_spec_ref,v_fuel_ref,number_doors)

    RETURNING ID INTO v_car_id;



     --Retrieve a car ref

    SELECT REF(ca) INTO v_car_ref FROM Cars ca WHERE
ca.ID = v_car_id;



     -- Insert a new Technology record linked to the new Car

    INSERT INTO Technologies ( Description, Type, Year,
Feature)

    VALUES ( p_safety_feature_description,
p_feature_type, SYSDATE, v_car_ref)
```

```
--EXAMPLE OF INSERTING A TRUCK
BEGIN
   INSERTTRUCK (
      p_origin => 'USA',
      p_production_origin => 'Factory A',
      p_change_description => 'Upgrade engine',
      p_engine_spec => 'Hybrid Engine',
      p_horsepower => 500,
      p_fuel_type => 'Gasoline',
      p_engine_type => 'hybrid', --HERE THERE ARE CONTRAINT
      p_model => 'BIgTRUCK',
      p_price => 35000,
      p_company_name => 'CPN1', --COMPANY NAME BETWEEN COMPANIES
      p_selected_fuel => 'Gasoline', --SAME OF FUEL TYPE
      payloadcapacity_value => 5000,
      p_safety_feature_description => 'Advanced Safety Features',
      p_feature_type => 'safety features'
   );
END;
```

```
   RETURNING ID INTO v_technology_id;

   -- Commit the transaction to save the changes
   COMMIT;

   EXCEPTION
   WHEN OTHERS THEN
      -- Handle exceptions if needed
      DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
      ROLLBACK;
END INSERTCAR;

--CREATING PROCEDURE TO INSERT A TRUCK
create or replace PROCEDURE INSERTTRUCK (
   p_origin VARCHAR2,
   p_production_origin VARCHAR2,
   p_change_description VARCHAR2,
   p_engine_spec VARCHAR2,
   p_horsepower int,
   p_fuel_type VARCHAR2,
   p_engine_type VARCHAR2,
   p_model VARCHAR2,
   p_price int,
   p_company_name VARCHAR2,
   p_selected_fuel VARCHAR2,
   payloadcapacity_value int,
   p_safety_feature_description VARCHAR2,
   p_feature_type VARCHAR2
   ) AS
   -- Declare variables for the new records
   v_history_id int;
   v_change_id int;
   v_specification_id int;
   v_truck_id int;
   v_technology_id int;
   v_history_ref REF HIstoryTY;
   v_spec_ref REF SpecificationsTY;
   v_company_ref REF CompanyTY;
   v_fuel_ref REF FuelTY;
   v_truck_ref REF TruckTY;

BEGIN
```

```
-- Insert a new History record

    INSERT INTO Histories ( Origin, Prod_Origin)

    VALUES ( p_origin, p_production_origin)

    RETURNING ID INTO v_history_id;



    --Retrieve an history ref

    SELECT REF(h) INTO v_history_ref FROM Histories h
WHERE h.ID = v_history_id;



    -- Insert a new Change record linked to the new History

    INSERT INTO Changes (Description, Year, Keep)

    VALUES ( p_change_description,
SYSDATE,v_history_ref)

    RETURNING ID INTO v_change_id;



     -- Insert a new Specifications record

    INSERT INTO Specifications ( EngineSpec,
HorsePower)

    VALUES ( p_engine_spec, p_horsepower)

    RETURNING ID INTO v_specification_id;



    --Retrieve a specification ref

    SELECT REF(s) INTO v_spec_ref FROM Specifications
s WHERE s.ID = v_specification_id;



    --Retrieve a fuel ref

    SELECT REF(f) INTO v_fuel_ref FROM Fuels f WHERE
f.Name_fuel = p_selected_fuel;



    --Retrieve a company ref

    SELECT REF(c) INTO v_company_ref FROM
Companies c WHERE c.Name_company =
p_company_name;



    -- Insert a new TRuck record linked to the new History,
Specifications, Company, and Fuel

    INSERT INTO Trucks ( Origins, Companyname, Year,
FuelType, EngineType, Model, Price, track, make,
describe, use, payloadcapacity)

    VALUES ( p_origin, p_company_name, SYSDATE,
p_fuel_type, p_engine_type, p_model, p_price,
v_history_ref , v_company_ref
,v_spec_ref,v_fuel_ref,payloadcapacity_value)

    RETURNING ID INTO v_truck_id;



     --Retrieve a truck ref

    SELECT REF(t) INTO v_truck_ref FROM Trucks t
WHERE t.ID = v_truck_id;
```

```
--EXAMPLE OF INSERTING A MOTORCYCLE
BEGIN
   INSERTMOTOR (
      p_origin => 'USA',
      p_production_origin => 'Factory A',
      p_change_description => 'Upgrade engine',
      p_engine_spec => 'Hybrid Engine',
      p_horsepower => 500,
      p_fuel_type => 'Gasoline',
      p_engine_type => 'hybrid', --HERE THERE ARE CONTRAINT
      p_model => 'SmallMoto',
      p_price => 35000,
      p_company_name => 'CPN1', --COMPANY NAME BETWEEN COMPANIES
      p_selected_fuel => 'Gasoline', --SAME OF FUEL TYPE
      ridingstyle_value => 'sportbike',
      p_safety_feature_description => 'Advanced Safety Features',
      p_feature_type => 'safety features'
   );
END;
```

```
    -- Insert a new Technology record linked to the new Car
   INSERT INTO Technologies ( Description, Type, Year, Feature)
   VALUES ( p_safety_feature_description, p_feature_type, SYSDATE, v_truck_ref)
   RETURNING ID INTO v_technology_id;

   -- Commit the transaction to save the changes
   COMMIT;

   EXCEPTION
   WHEN OTHERS THEN
      -- Handle exceptions if needed
      DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
      ROLLBACK;
END INSERTTRUCK;

--CREATION OF PROCEDURE TO INSERT A MOTORCYCLE

create or replace PROCEDURE INSERTMOTOR (
   p_origin VARCHAR2,
   p_production_origin VARCHAR2,
   p_change_description VARCHAR2,
   p_engine_spec VARCHAR2,
   p_horsepower int,
   p_fuel_type VARCHAR2,
   p_engine_type VARCHAR2,
   p_model VARCHAR2,
   p_price int,
   p_company_name VARCHAR2,
   p_selected_fuel VARCHAR2,
   ridingstyle_value VARCHAR2,
   p_safety_feature_description VARCHAR2,
   p_feature_type VARCHAR2
   ) AS
   -- Declare variables for the new records
   v_history_id int;
   v_change_id int;
   v_specification_id int;
   v_motorcycle_id int;
   v_technology_id int;
   v_history_ref REF HIstoryTY;
```

	That's as been implemented as an PL/SQL PROCEDURE and called with SQL Operations.	v_spec_ref REF SpecificationsTY; v_company_ref REF CompanyTY; v_fuel_ref REF FuelTY; v_motorcycle_ref REF MotorcycleTY; BEGIN -- Insert a new History record INSERT INTO Histories (Origin, Prod_Origin) VALUES (p_origin, p_production_origin) RETURNING ID INTO v_history_id; --Retrieve an history ref SELECT REF(h) INTO v_history_ref FROM Histories h WHERE h.ID = v_history_id; -- Insert a new Change record linked to the new History INSERT INTO Changes (Description, Year, Keep) VALUES (p_change_description, SYSDATE,v_history_ref) RETURNING ID INTO v_change_id; -- Insert a new Specifications record INSERT INTO Specifications (EngineSpec, HorsePower) VALUES (p_engine_spec, p_horsepower) RETURNING ID INTO v_specification_id; --Retrieve a specification ref SELECT REF(s) INTO v_spec_ref FROM Specifications s WHERE s.ID = v_specification_id; --Retrieve a fuel ref SELECT REF(f) INTO v_fuel_ref FROM Fuels f WHERE f.Name_fuel = p_selected_fuel; --Retrieve a company ref SELECT REF(c) INTO v_company_ref FROM Companies c WHERE c.Name_company = p_company_name; -- Insert a new Motorcycle record linked to the new History, Specifications, Company, and Fuel INSERT INTO Motorcycles (Origins, Companyname, Year, FuelType, EngineType, Model, Price, track, make, describe, use, ridingstyle) VALUES (p_origin, p_company_name, SYSDATE, p_fuel_type, p_engine_type, p_model, p_price,

		v_history_ref , v_company_ref ,v_spec_ref,v_fuel_ref,ridingstyle_value) RETURNING ID INTO v_motorcycle_id; --Retrieve a truck ref SELECT REF(m) INTO v_motorcycle_ref FROM Motorcycles m WHERE m.ID = v_motorcycle_id; -- Insert a new Technology record linked to the new Car INSERT INTO Technologies (Description, Type, Year, Feature) VALUES (p_safety_feature_description, p_feature_type, SYSDATE, v_motorcycle_ref) RETURNING ID INTO v_technology_id; -- Commit the transaction to save the changes COMMIT; EXCEPTION WHEN OTHERS THEN -- Handle exceptions if needed DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM); ROLLBACK; END INSERTMOTOR;
OP2 **Client**	**That's has been implemented as SQL operations code.**	SELECT t.ID AS Truck_ID, t.Origins AS Truck_Origins, t.Companyname AS Truck_Company, TO_CHAR(t.Year, 'YYYY-MM-DD') AS Truck_Year, t.FuelType AS Truck_Fuel_Type, t.EngineType AS Truck_Engine_Type, t.Model AS Truck_Model, t.Price AS Truck_Price, t.PayloadCapacity AS Truck_Payload_Capacity, DEREF(t.Describe).EngineSpec AS Truck_Engine_Spec, DEREF(t.Describe).HorsePower AS Truck_Horsepower FROM Trucks t;
OP3 **Client**	**That's has been implemented as SQL operations code.**	**SELECT *** **FROM (** **SELECT** **c.ID AS Car_ID,** **c.Companyname AS Car_Company,** **TO_CHAR(c.Year, 'YYYY-MM-DD') AS Car_Year,** **c.FuelType AS Car_Fuel_Type,** **c.EngineType AS Car_Engine_Type,**

		c.Model AS Car_Model, c.price AS Car_Cost, c.Describe.EngineSpec AS Car_Engine_Spec, c.Describe.HorsePower AS Car_HorsePower FROM Cars c ORDER BY c.Year DESC, c.price ASC) WHERE ROWNUM = 1;
OP4 **Client**	**That's has been implemented as SQL operations code.**	SELECT c.ID AS Vehicle_ID, c.Model AS Vehicle_Model, 'Car' AS Vehicle_Type, h.ID AS History_ID, ch.Year AS Change_Year, ch.Description AS Change_Spec, h.Prod_Origin AS Origin FROM Cars c JOIN Histories h ON c.track = REF(h) LEFT JOIN Changes ch ON REF(h) = ch.keep WHERE c.Model = 'EorwwzNzHJ' UNION ALL SELECT m.ID AS Vehicle_ID, m.Model AS Vehicle_Model, 'Motorcycle' AS Vehicle_Type, h.ID AS History_ID, ch.Year AS Change_Year, ch.Description AS Change_Spec, h.Prod_Origin AS Origin FROM Motorcycles m JOIN Histories h ON m.track = REF(h) LEFT JOIN Changes ch ON REF(h) = ch.keep WHERE

		m.Model = 'EorwwzNzHJ' UNION ALL SELECT t.ID AS Vehicle_ID, t.Model AS Vehicle_Model, 'Truck' AS Vehicle_Type, h.ID AS History_ID, ch.Year AS Change_Year, ch.Description AS Change_Spec, h.Prod_Origin AS Origin FROM Trucks t JOIN Histories h ON t.track = REF(h) LEFT JOIN Changes ch ON REF(h) = ch.keep WHERE t.Model = 'EorwwzNzHJ';
OP5 **Client**	**That's has been implemented as SQL operations code.**	SELECT c.model AS Car_Model, f.Name_fuel AS Fuel_Name, s.year, s.efficiency FROM Cars c JOIN Scores s ON c.use = s.Evaluate JOIN Fuels f ON s.Evaluate = REF(f) WHERE c.Model = 'EorwwzNzHJ' AND s.year >= ADD_MONTHS(TRUNC(SYSDATE, 'YEAR'), -60) -- Retrieve scores from the past five years
OP6 **Client**	**That's has been implemented as SQL operations code.**	SELECT c.model AS Vehicle_model, t.id AS Tech_id, t.year AS Tech_year, t.type AS Tech_type, t.description AS Tech_Description FROM Technologies t JOIN cars c ON t.feature = REF(c)

		WHERE c.enginetype='electric'AND t.type='safety features' AND t.year >= (SYSDATE - INTERVAL '2' YEAR) UNION ALL SELECT m.model AS Vehicle_model, t.id AS Tech_id, t.year AS Tech_year, t.type AS Tech_type, t.description AS Tech_Description FROM Technologies t JOIN motorcycles m ON t.feature = REF(m) WHERE m.enginetype='electric' AND t.type='safety features' AND t.year >= (SYSDATE - INTERVAL '2' YEAR) UNION ALL SELECT tl.model AS Vehicle_model, t.id AS Tech_id, t.year AS Tech_year, t.type AS Tech_type, t.description AS Tech_Description FROM Technologies t JOIN trucks tl ON t.feature = REF(tl) WHERE tl.enginetype='electric' AND t.type='safety features' AND t.year >= (SYSDATE - INTERVAL '2' YEAR);
OP7 **Client**	**That's has been implemented as SQL operations code.**	SELECT t.id, t.year, t.type, t.description FROM Technologies t WHERE t.feature = (SELECT REF(tr) FROM trucks tr WHERE tr.model = 'RpSvoqUbNV');
OP8		SELECT

Client	**That's has been implemented as SQL operations code.**	m.ID AS Motorcycle_ID, m.Model AS Motorcycle_Model, m.Year AS Motorcycle_Year, m.FuelType AS Motorcycle_Fuel_Type, m.EngineType AS Motorcycle_Engine_Type, m.RidingStyle AS Motorcycle_Riding_Style FROM Motorcycles m JOIN Companies c ON m.Make = REF(c) WHERE c.Name_company = 'CPN3';

OP9
Client/Admin
	BEGIN
	InsertClient (
	p_Username => 'jodoe',
	p_Password => 'password123',
	p_Dateregister => SYSDATE, -- Using SYSDATE for the current date
	p_Name_client => 'Johnatan',
	p_Surname => 'DoVe',
	p_City => 'New York',
	p_Street => '123 Main St',
	p_Civicnum => 'A123',
	p_Province => 'NY',
	p_Phone => '5551234567'
	);
	END;
	/
	That's as been implemented as an
		create or replace PROCEDURE InsertClient (
		p_Username VARCHAR2,
		p_Password VARCHAR2,
		p_Dateregister DATE,
		p_Name_client VARCHAR2,
		p_Surname VARCHAR2,
		p_City VARCHAR2,
		p_Street VARCHAR2,
		p_Civicnum VARCHAR2,
		p_Province VARCHAR2,
		p_Phone VARCHAR2
		) AS
		BEGIN
		-- Insert a new client record
		INSERT INTO Clients (
		Username,
		Password,
		Dateregister,
		Type,
		Name_client,
		Surname,
		City,
		Street,
		Civicnum,
		Province,
		Phone
		) VALUES (
		p_Username,
		p_Password,
		SYSDATE,

	PL/SQL PROCEDURE and called with SQL Operations.	'Client', p_Name_client, p_Surname, p_City, p_Street, p_Civicnum, p_Province, p_Phone); -- Commit the transaction to save the changes COMMIT; DBMS_OUTPUT.PUT_LINE('Client inserted successfully.'); EXCEPTION WHEN OTHERS THEN -- Handle exceptions if needed DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM); ROLLBACK; END InsertClient;
OP10 **Client**	**That's has been implemented as SQL operations code.**	SELECT f.Name_fuel AS Fuel_Name, f.SavingTech AS Fuel_Saving_Technology, f.Details AS Fuel_Details, s.Efficiency AS Actual_Efficiency FROM Fuels f JOIN Scores s ON s.Evaluate = REF(f) WHERE f.Name_fuel = 'CNG' AND s.Year = (SELECT MAX(Year) FROM Scores WHERE Evaluate = REF(f));

OP11
Admin
	BEGIN
	DeleteClient(
	p_Username => 'jodoe' -- Replace with the username of the client you want to delete
	);
	END;
	/
	That's as been implemented as an PL/SQL PROCEDURE and
		create or replace PROCEDURE DeleteClient (
		p_Username VARCHAR2
		) AS
		BEGIN
		-- Delete client basing on the username
		DELETE FROM Clients
		WHERE Username = p_Username;
		
		-- Commit the transaction to save the changes
		COMMIT;

	called with SQL Operations.	**EXCEPTION** **WHEN OTHERS THEN** **-- Handle exceptions if needed** **DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);** **ROLLBACK;** **END DeleteClient;**

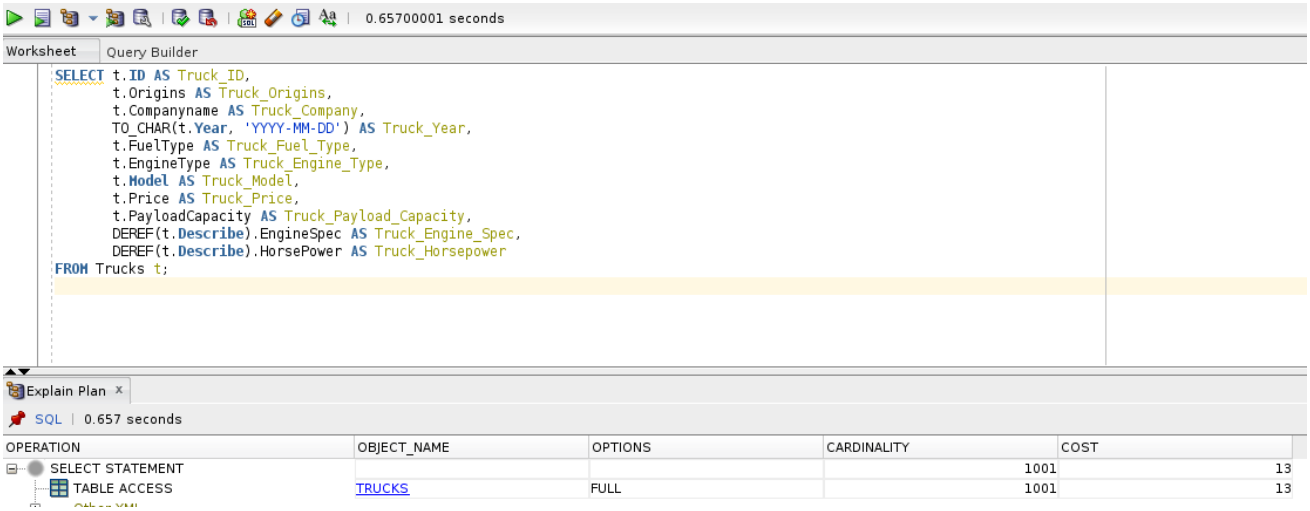
All these procedures are going to be implemented into the WebApp through the servlets.

OPTIMIZATIONS AND INDEX

OPERATION 1

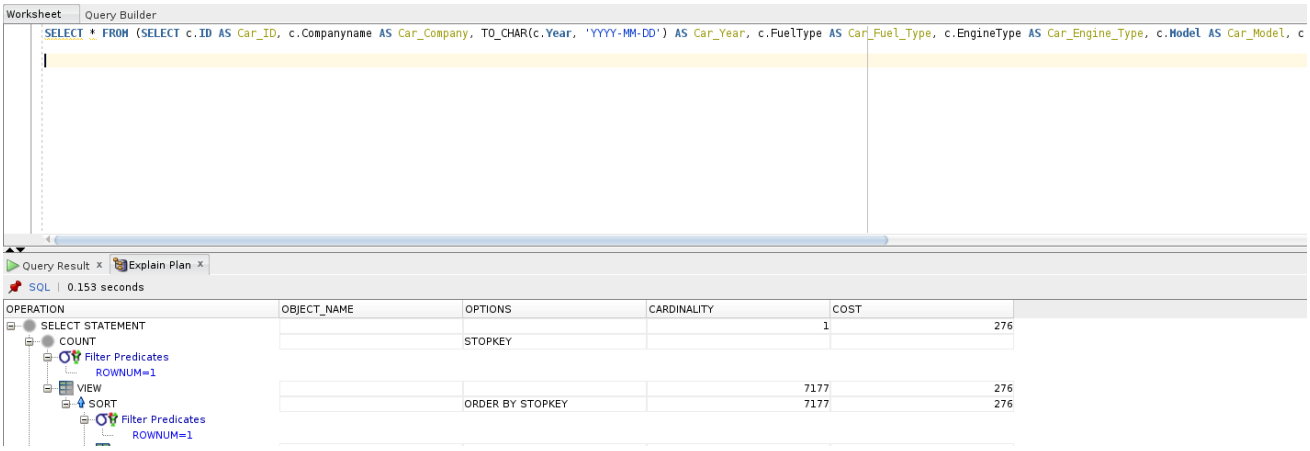
This operation does not need an optimization because it involves just the inserting of the data of a new value, moreover the operation is executed just 1 time at day, so it's acceptable.

OPERATION 2



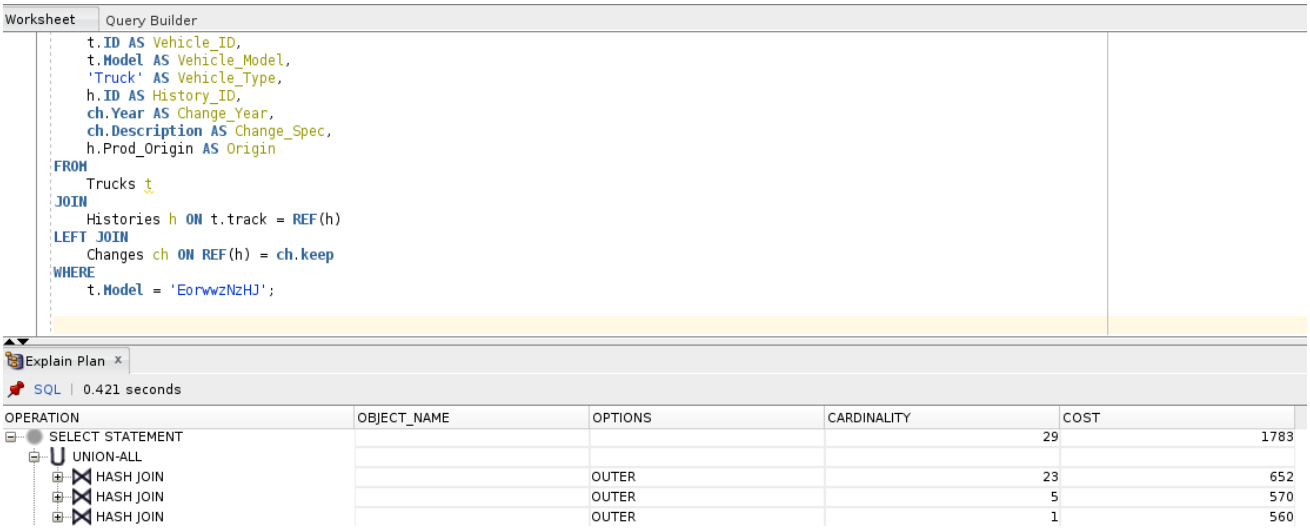
This operation is made by a single selection that takes all the values from the trucks table. The optimization here is not needed, moreover this operation is considered just 1 time at week, so the cost is very low.

OPERATION 3



This query provided doesn't include any filtering conditions. Since there's no filtering involved, indexing won't have a significant impact on the performance of this query. The cost is a little bit relevant but considering that this operation will be carried just 1 time a day we can accept here.

OPERATION 4



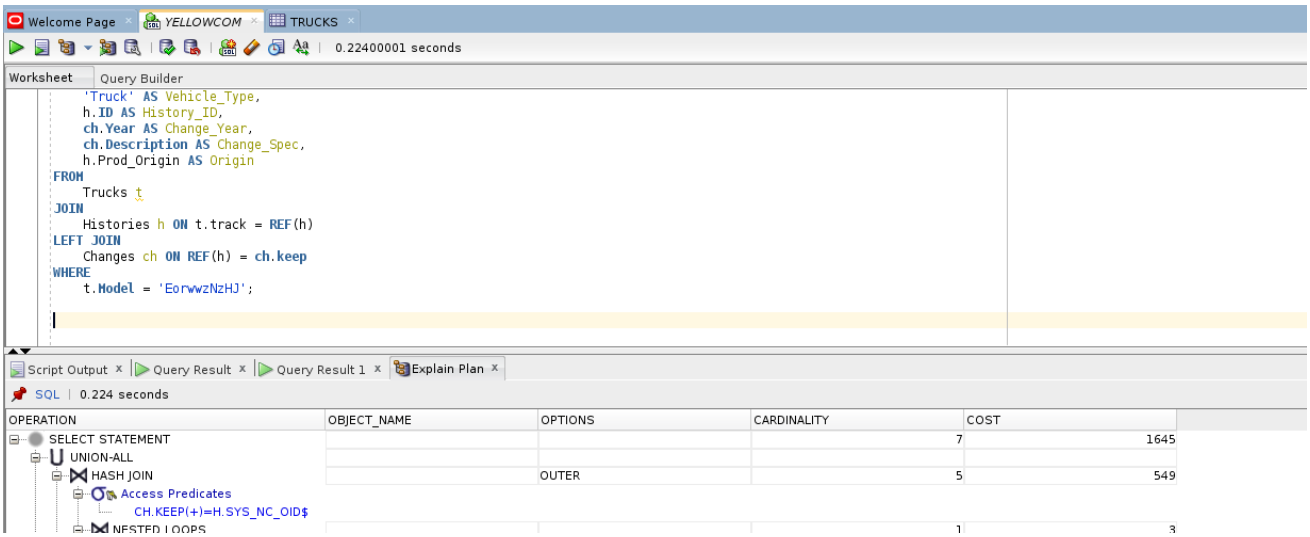
```
Worksheet  Query Builder
    t.ID AS Vehicle_ID,
    t.Model AS Vehicle_Model,
    'Truck' AS Vehicle_Type,
    h.ID AS History_ID,
    ch.Year AS Change_Year,
    ch.Description AS Change_Spec,
    h.Prod_Origin AS Origin
FROM
    Trucks t
JOIN
    Histories h ON t.track = REF(h)
LEFT JOIN
    Changes ch ON REF(h) = ch.keep
WHERE
    t.Model = 'EorwwzNzHJ';
```

Explain Plan

SQL | 0.421 seconds

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			29	1783
UNION-ALL				
HASH JOIN		OUTER	23	652
HASH JOIN		OUTER	5	570
HASH JOIN		OUTER	1	560

Here we have 3 joins made on the various car, motorcycle, and trucks tables, because we want to retrieve the data for the model of the specified vehicle to see al changes made on. There isn't so much optimization options , but one optimization that of course could be on creating index on model attribute on the three tables.



Welcome Page | YELLOWCOM | TRUCKS

0.22400001 seconds

```
Worksheet  Query Builder
    'Truck' AS Vehicle_Type,
    h.ID AS History_ID,
    ch.Year AS Change_Year,
    ch.Description AS Change_Spec,
    h.Prod_Origin AS Origin
FROM
    Trucks t
JOIN
    Histories h ON t.track = REF(h)
LEFT JOIN
    Changes ch ON REF(h) = ch.keep
WHERE
    t.Model = 'EorwwzNzHJ';
```

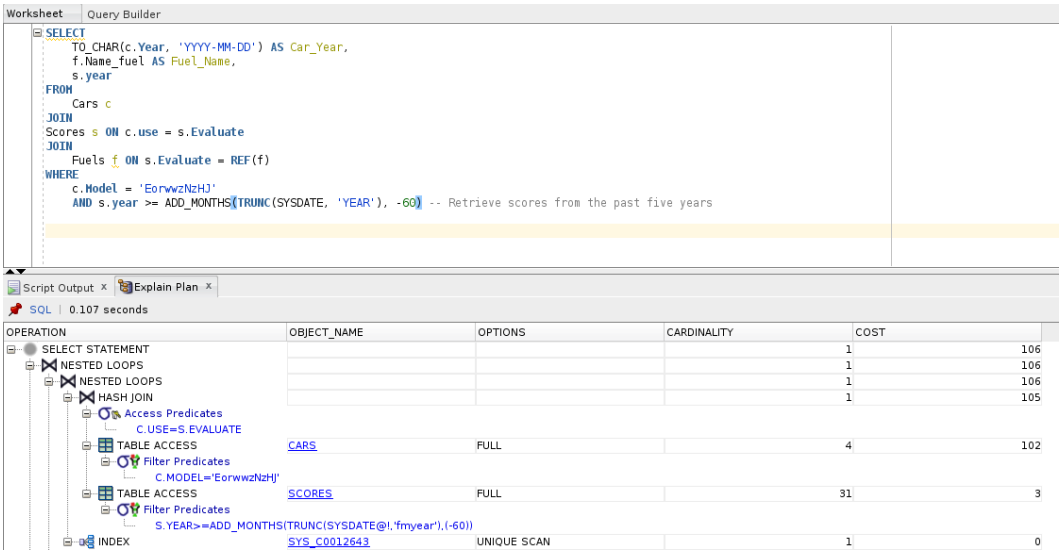
Script Output | Query Result | Query Result 1 | Explain Plan

SQL | 0.224 seconds

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			7	1645
UNION-ALL				
HASH JOIN		OUTER	5	549
Access Predicates				
CH.KEEP(+)=H.SYS_NC_OID$				
NESTED LOOPS			1	3

Anyway, considering that this operation till now is carried out just 2 times at month we decide to keep it like this, if the data will increase over the time materialized views to precompute and store complex query results, reducing the need for expensive joins could also be a good solution.

OPERATION 5



```
SELECT
    TO_CHAR(c.Year, 'YYYY-MM-DD') AS Car_Year,
    f.Name_fuel AS Fuel_Name,
    s.year
FROM
    Cars c
JOIN
Scores s ON c.use = s.Evaluate
JOIN
    Fuels f ON s.Evaluate = REF(f)
WHERE
    c.Model = 'EorwwzNzHJ'
    AND s.year >= ADD_MONTHS(TRUNC(SYSDATE, 'YEAR'), -60) -- Retrieve scores from the past five years
```

Script Output | Explain Plan

SQL | 0.107 seconds

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	106
NESTED LOOPS			1	106
NESTED LOOPS			1	106
HASH JOIN			1	105
Access Predicates				
C.USE=S.EVALUATE				
TABLE ACCESS	CARS	FULL	4	102
Filter Predicates				
C.MODEL='EorwwzNzHJ'				
TABLE ACCESS	SCORES	FULL	31	3
Filter Predicates				
S.YEAR>=ADD_MONTHS(TRUNC(SYSDATE@!,'fmyear'),(-60))				
INDEX	SYS_C0012643	UNIQUE SCAN	1	0

Here the cost of the query considering that is done once a month is affordable, anyway creating an index on the car model and car year attributes can optimize the cost of the query, due to selection is filtered on them.

```
SELECT
    c.model AS Car_Model,
    f.Name_fuel AS Fuel_Name,
    s.year,
    s.efficiency
FROM
    Cars c
JOIN
Scores s ON c.use = s.Evaluate
JOIN
    Fuels f ON s.Evaluate = REF(f)
WHERE
    c.Model = 'EorwwzNzHJ'
    AND s.year >= ADD_MONTHS(TRUNC(SYSDATE, 'YEAR'), -60) -- Retrieve scores from the past five years
```

pt Output | Query Result | Query Result 1 | Explain Plan

L | 0.117 seconds

TION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	5
NESTED LOOPS			1	5
NESTED LOOPS			1	5
NESTED LOOPS			1	4
TABLE ACCESS	CARS	BY INDEX ROWID BATCHED	1	2
INDEX	IDX_CARS_MODEL	RANGE SCAN	1	1
Access Predicates				

After the indexing it performs better and considering that this operation is carried once times at month, we decide to keep like this.

OPERATION 6

```
UNION ALL
SELECT
    tl.model AS Vehicle_model,
    t.id AS Tech_id,
    t.year AS Tech_year,
    t.type AS Tech_type,
    t.description AS Tech_Description

FROM Technologies t
JOIN
    trucks tl ON t.feature = REF(tl)
WHERE
tl.enginetype='electric' AND t.type='safety features' AND t.year >= (SYSDATE - INTERVAL '2' YEAR);
```

Script Output | Query Result | Query Result 1 | Explain Plan

SQL | 0.24 seconds

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			186561	6704
UNION-ALL				
HASH JOIN			129879	2290
T.FEATURE=TREAT(REF(C) AS REF)				
TABLE ACCESS	CARS	FULL	2308	102
TABLE ACCESS	TECHNOLOGIES	FULL	5627	2188
HASH JOIN			37100	2213
T.FEATURE=TREAT(REF(M) AS REF)				
TABLE ACCESS	MOTORCYCLES	FULL	659	25
TABLE ACCESS	TECHNOLOGIES	FULL	5627	2188
HASH JOIN			19582	2201
T.FEATURE=TREAT(REF(TL) AS REF)				
TABLE ACCESS	TRUCKS	FULL	348	13

Here the cost of query is expensive due to the nature of it, we select on the three tables, cars, motorcycles and trucks and a filtering is made on technology type and year of tech and engine type of the vehicle. Anyway, to optimize the query, we can index the three-attribute year, tech type and engine type.

```
    tl.model AS Vehicle_model,
    t.id AS Tech_id,
    t.year AS Tech_year,
    t.type AS Tech_type,
    t.description AS Tech_Description

FROM Technologies t
JOIN
    trucks tl ON t.feature = REF(tl)
WHERE
tl.enginetype='electric' AND t.type='safety features' AND t.year >= (SYSDATE - INTERVAL '2' YEAR);
```

Script Output | Query Result | Query Result 1 | Explain Plan

SQL | 0.171 seconds

ERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			188451	1354
UNION-ALL				
HASH JOIN			131899	507
Access Predicates				
T.FEATURE=TREAT(REF(C) AS REF)				
TABLE ACCESS	CARS	FULL	2344	102
Filter Predicates				
C.ENGINETYPE='electric'				

After the indexing it performs better and considering that this operation is carried two times at month, we decide to keep like this. In the case we want to optimize more Materialized Views could be a good option.

INDEX CODE USED TO OPTIMIZE THE QUERY

```
-- Index for the Cars table
CREATE INDEX idx_cars_model ON Cars(Model);

-- Index for the Motorcycles table
CREATE INDEX idx_motorcycles_model ON Motorcycles(Model);

-- Index for the Trucks table
CREATE INDEX idx_trucks_model ON Trucks(Model);

--Index for the Scores table
CREATE INDEX idx_scores_year ON Scores(year);

--INDEX ON THE THREE VEHICLES TABLES
CREATE INDEX idx_cars_enginetype ON Cars(enginetype);
CREATE INDEX idx_motorcycles_enginetype ON Motorcycles(enginetype);
CREATE INDEX idx_trucks_enginetype ON Trucks(enginetype);
--INDEX ON THE TECHNOLOGY TYPE
CREATE INDEX idx_technologies_type ON Technologies(type);
--INDEX ON THE YEAR TYPE
CREATE INDEX idx_technologies_year ON Technologies(year);
```

DATAWAREHOUSE DESIGN

Now we want to create a DW schema from our E-R, in order to do this, we have to identify for first objectives of the analysis:

Analysis of the **Company** trends and information is done by considering different aspects(Dimension of analysis are how they change over place and time).

Identification of the main concepts for a multidimensional analysis

So, seeing the schema the FACT identified is obviously the Company

In the first Analysis on Company, we will consider the Company as the FACT and then we will consider the number of vehicles built then also the number of electric vehicles built as a principal Measure of Interest and that could be calculated from the cars, motorcycle, and trucks tables.

Identification of the dimensions

Now navigating the schema, we identified dimensions

For the analysis of Company

1) The Company can be examined basing on the **space** dimensions basing on the place where they execute their activity at different levels of aggregation (address, city, region, country).
2) The Company can also be identified basing on the **time** dimensions.(day, month, quarter, and year).

Restructuring the ER schema

Now we will restructure the schema to represent dimensions more explicitly.

Time →Day→Month→Quarter→Year

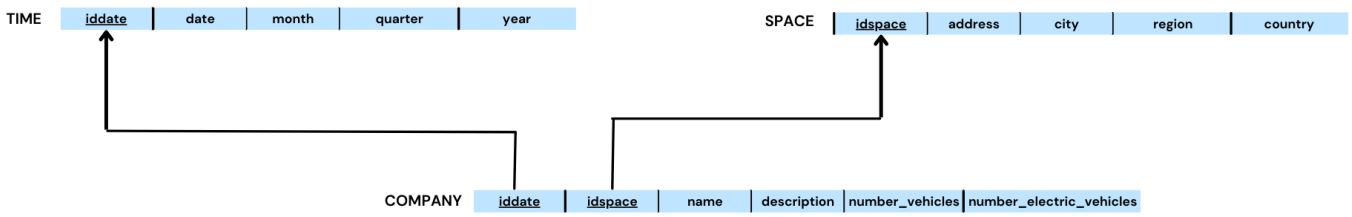
Space → Address → City → Region → Country

Now assuming the relational model as model for the logic translation we are going to create star scheme tables of dimensions.

Time(iddate, date, month, quarter, year)

Space(idspace, address, city, region, country)

Company(Name, description, number_vehicle, number_electric_vehicle)



I decided to analyze this fact for this little DW Datawarehouse but it's always possible to analyze even more facts with various dimensions.

This DW could be useful for a manager or for an analyst to know the information about the production about a particular company, and knowing the number or electric vehicle could give insight of how much is the company focused on the green energy.

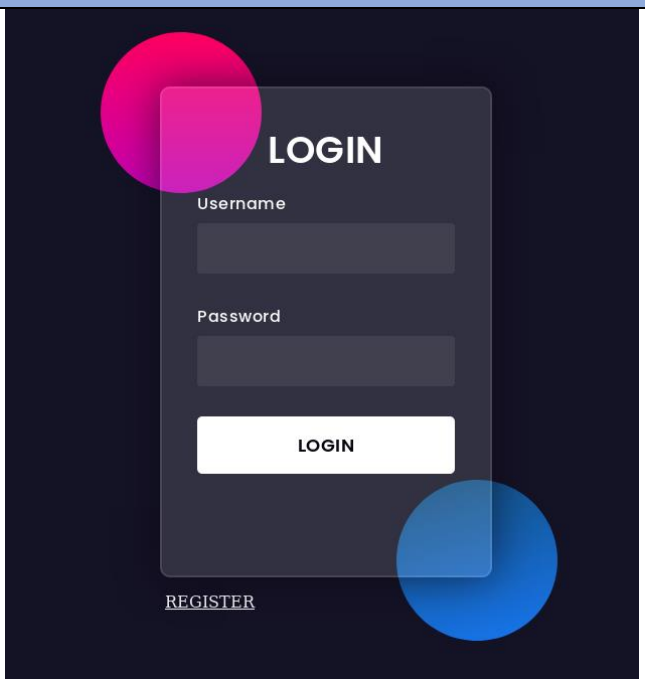
Structure and design of the AutoTech WEBAPP

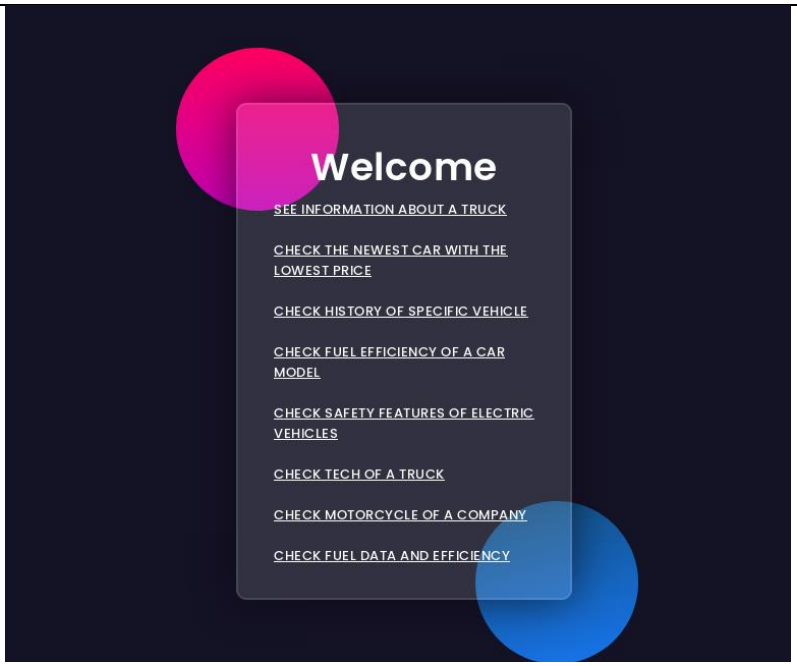
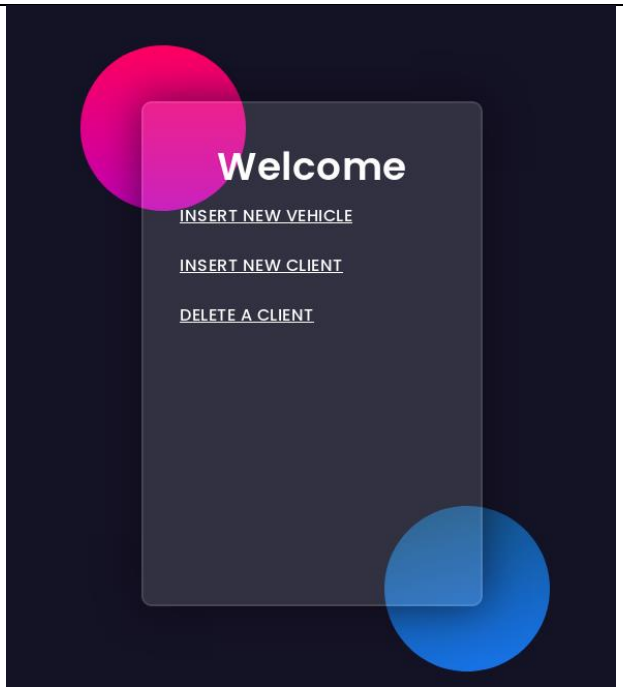
HTML and CSS played a crucial role in shaping the graphics of the web page. CSS technology allowed the customization of buttons, text areas, and tables, enhancing the overall readability and visual appeal of the web pages.

In addition to HTML and CSS, JavaScript proved helped in adding interactivity to the web application. It enabled the implementation of various actions triggered by button clicks, such as opening new JSP pages.

Moreover, web application was also enriched by the integration of JSP (Java Server Pages) and Servlet technologies. These server-side technologies powered by dynamic content generation and processing, enhancing the overall functionality and responsiveness of the web application.

We used these to implement all the operations on the DB that was defined previously. Moreover, we customized it adding a registration of a new user and adding pages for Client and Admin operations (Decided to keep the insert operation 1 only as an admin privilege) .

MAIN PAGE	STRUCTURE
LOGIN PAGE	

CLIENT PAGE	
ADMIN PAGE	

SITE JSP STRUCTURE

Login.jsp	Main page for the login
Loginsuccess_client.jsp	Main page of a client user
Loginsuccess.jsp	Main page of an admin user
Get_Trucks.jsp	Page to see Trucks informations
Get_New.jsp	Page to see the newest car with less price

Get_History.jsp	Page to see the history of a particular vehicle
Get_Eff.jsp	Page to see efficiency of fuel of a particular car model over the past five year
Check_Sfev.jsp	Page to see Safety Features of Electric Vehicles of the last 2 year
Get_trucktch.jsp	Page to see the tech of a truck
Get_MotCp.jsp	Page to see motorcycles of a company
Get_FuelSc.jsp	Page to see fuel data and efficiency
Insert_Vehicle.jsp	Page to select witch kind of vehicle we want to insert
customer_insert.jsp	Page to insert a new client
user_delete.jsp	Page to delete a client
Insert_Car.jsp	Page to insert a new car
Insert_Motorcycle.jsp	Page to insert a new motorcycle
Insert_Truck.jsp	Page to insert a new truck

At the end different Sel_**** pages has been used just to insert the data needed for the query for particular operations.

FUTURE DEVELOP

1) Building the Datawarehouse.
2) Improve Site graphic and Interface.
3) Insert new operation.
4) Build the DB tables using external affordable sources to have more consistent values.
5) Creation of materialized views to manage complex query of costly joins.