Description of the Enterprise:

Our project, "Vehicle Database Management System," is an innovative solution primarily designed for entities such as the Department of Transportation (DOT) and the Department of Energy (DOE). The primary objective of this system is to assist these organizations in effectively managing and analyzing extensive vehicle data, crucial for planning and building infrastructure to accommodate a diverse range of vehicles.

Nature of the Enterprise:

The enterprise is envisioned as a collaborative effort between the DOT and DOE, focusing on aggregating and utilizing vehicle data for infrastructure development and energy planning. This initiative would involve collecting detailed information on vehicle types, ranging from conventional gas-powered vehicles to electric and hybrid models.

Scope and Justification:

The need for such a comprehensive database system stems from several key factors:

Diversity of Vehicle Data:

With an ever-increasing variety of vehicles on the roads, including electric vehicles (EVs) and hybrids, there is a growing need to understand their distribution, usage patterns, and impact on infrastructure

.

Infrastructure Planning:

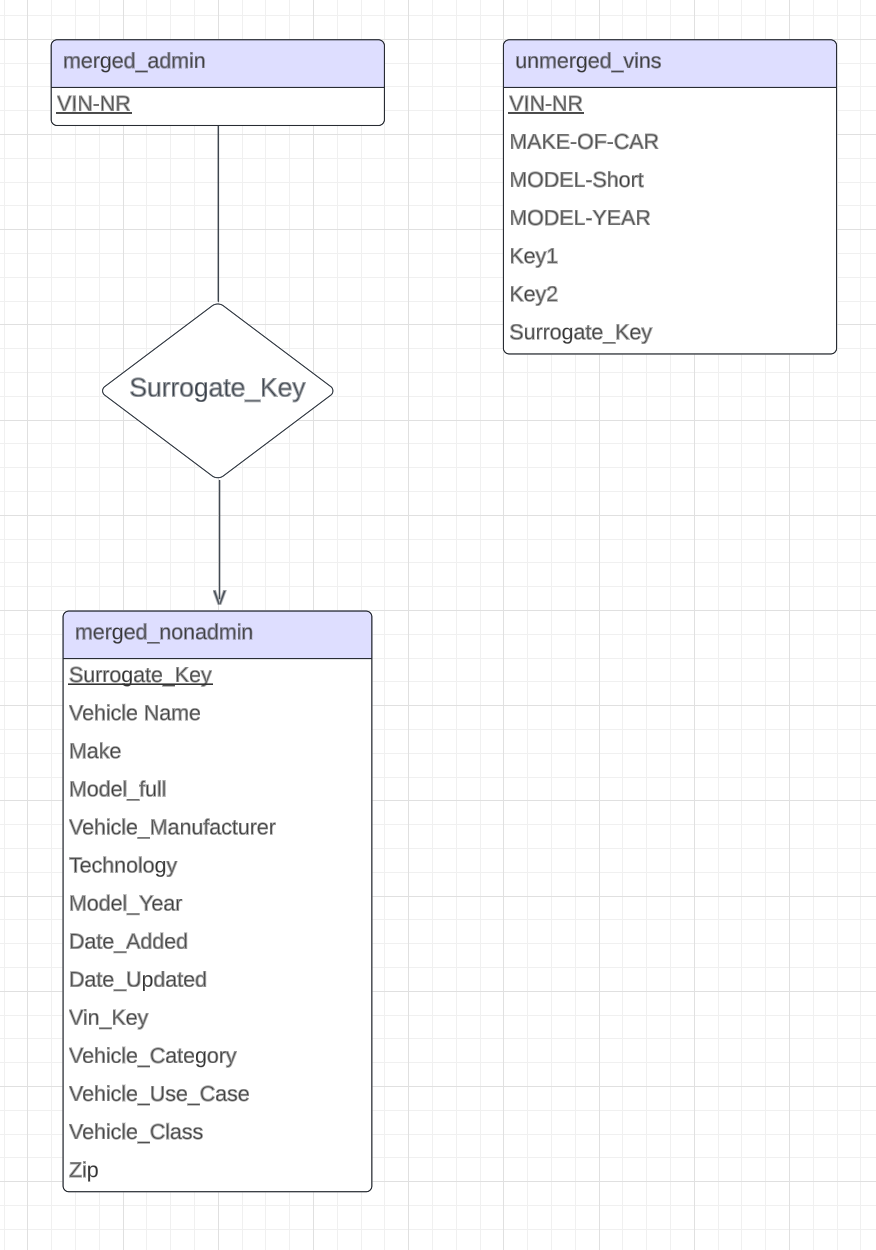
Accurate and detailed vehicle data aids in designing roads, highways, and urban layouts that can efficiently support current and future traffic volumes and types.

Energy Consumption and Planning: The DOE can leverage this data to forecast energy requirements, particularly for EV charging stations, and plan for sustainable energy distribution.

Regulatory and Policy Formulation: The DOT and DOE can use this data to inform policy decisions, set vehicle standards, and regulate emissions and safety norms.

Purpose and Utility:

The Vehicle Database Management System is pivotal for:

* Centralizing vehicle data to assist in informed decision-making for infrastructure development and energy planning.
* Providing insights into the types of vehicles in use, their distribution, and projected growth trends.
* Facilitating the strategic placement of EV charging stations and optimizing energy distribution networks.
* Enabling the DOT and DOE to forecast future transportation needs and align their policies accordingly.
* In essence, our Vehicle Database Management System is a strategic tool for the DOT and DOE, aiding in the meticulous planning and development of transportation infrastructure and energy systems in an era of rapidly evolving vehicle technology. It stands as a testament to how data-driven insights can lead to more efficient, sustainable, and future-ready urban and transportation planning.
* 

Schema List and Descriptions:

unmerged\_vins Schema:

Purpose: This schema holds basic vehicle information before it is merged with additional data.

Key Components:

Primary Key: VIN-NR

Attributes: MAKE-OF-CAR, MODEL-Short, MODEL-YEAR, key1, key2

Relationship: There's no direct foreign key relationship to other tables. However, it serves as the foundation for generating merged tables.

merged\_admin Schema:

Purpose: Acts as the central repository for detailed vehicle data. It combines basic VIN information with additional data points for a comprehensive view.

Key Components:

Primary Key: VIN-NR

Attributes: Includes all detailed vehicle information like Vehicle Name, Make, Model\_full, etc.

Surrogate\_Key: A unique identifier that connects to the merged\_nonadmin schema.

Relationship: Directly related to the merged\_nonadmin schema through a one-to-one relationship using the Surrogate\_Key. This ensures that each record in merged\_admin corresponds to a record in merged\_nonadmin.

merged\_nonadmin Schema:

Purpose: Contains non-sensitive vehicle data accessible to non-admin users, providing a subset of the data available in merged\_admin.

Key Components:

Primary Key: Surrogate\_Key

Attributes: Similar to merged\_admin, but excludes sensitive information like VIN-NR.

Relationship: It's directly connected to the merged\_admin schema. The Surrogate\_Key in merged\_nonadmin references the same key in merged\_admin, facilitating data consistency and integrity.

Overall Functionality:

* The unmerged\_vins schema serves as the initial collection point for vehicle data, which is then enriched and merged into the merged\_admin schema.
* The merged\_admin schema is the backbone of the system, containing a complete set of vehicle data. Its connection to merged\_nonadmin via the Surrogate\_Key allows for efficient data management and access control.
* merged\_nonadmin caters to non-admin users, offering a secure view of the vehicle data without exposing sensitive details.

Application User Interfaces Description - Illustrations are not included due to the 8 page limit

1. Login Interface

Illustration: A simple login window with fields for username and password, and a login button.

Functionality: Users can log in as 'admin' or 'guest'. The 'admin' has full access to all functionalities including add, update, delete, export, and graph generation. The 'guest' has limited access, mainly for viewing data.

2. Main Dashboard

Illustration: A tabbed interface with different sections for 'merged\_admin', 'merged\_nonadmin', and 'unmerged\_vins'.

Functionality:

For Admins: Access to add, update, delete, export, and graph functionalities across all tabs.

For Guests: Limited to viewing data in the 'merged\_nonadmin' tab.

3. Data Addition Interface

Illustration: A dialog box with input fields corresponding to the columns of the selected table, along with an 'Add' button.

Functionality: Allows users (admins) to input data into the database. Each field in the dialog box corresponds to a column in the database table.

4. Data Update Interface

Illustration: A dialog box that first asks for the unique identifier (like VIN number), then displays fields with existing data for that record, allowing for modifications.

Functionality: Enables admins to update existing records in the database. The user selects a record to update by its unique identifier and then changes the desired fields.

5. Data Deletion Interface

Illustration: A simple dialog box requesting the unique identifier of the record to be deleted.

Functionality: Admins can delete records from the database by providing the unique identifier of the record they wish to remove.

6. Data Export Interface

Illustration: A window with dropdown menus to select table, columns, and filters for export, and an 'Export' button.

Functionality: Admins can export data from the database tables based on selected columns and filters. The exported data is saved in a CSV file.

7. Graph Generation Interface

Illustration: A window with options to select the type of graph (bar, pie), the table, and columns for data visualization, along with a 'Generate' button.

Functionality: Users can generate graphical representations of data, such as pie charts and bar graphs, based on selected criteria from the database. This is particularly useful for visual data analysis.

Test queries:  
  
 Use the readme file to set up your environment if need be.

To see if this project is working as expected we are going to do some simple tests. First run reset\_dbs.py which is used as a reset button if you want to set the database to its original state. Then run make\_ui.py to start the user interface. Here you will see a login page. To log in as admin use admin for the username and admin123 for the password. Now you are on the admin main page. Go to add, and fill out the first 5 columns with the word test and press the continue button. Next just add a 1 behind all of the test values you made earlier and press continue. Now exit out of the add window you made and press the update. Now you can update the test value we just added to the admin and non admin table by giving the first window that pops up the test. Edit any of the columns besides the surrogate key. Now exit out of the update window and go to the remove button. Use test1 for the value in the window that pops up to delete the test we added earlier. Now we can go to the export button which is on the non admin tab in the middle. When you use it you will be presented with a bunch of dropdown menus to choose what you want to export. You should go to the second column and find the test and export it. A .csv will be made with your export in it and it should only contain one row that has “test” for the values and whatever else you updated to it previously. Now notice that you can not export test1 because it is no longer there. Now go to the graphing function on the middle tab and choose the options, pie chart, model\_year, 2021, and Vehicle Name. You should be given a very nice graph. Now you can exit out of all the windows and try the guest log in which is guest for the username and nothing for the password. Now you only have 2 buttons, export and graph. You can use export like we did previously to check that the addition of “test” is still there with your changes. All of this can also be tested in unmerged found as the 3rd tab in the admin main page.

Participation:

William Duke: Took charge of the system design and architecture, ensuring a robust foundation for the project. developed both the user interface and the backend systems, providing a cohesive user experience and reliable data management. Additionally, contributed to creating the ER Diagram and relational schema, that helped define the database's structure.

Harrison Juneau: Focused on refining the front-end development, working on interface elements that users interact with directly. He was also responsible for maintaining clear and thorough documentation throughout the coding process. Played a key role in the initial creation of the ER Diagram and the relational schema, laying out the foundational structures for our database system.