ALPR Project: Technical Proposal for Database and Visualizations Selection

Created by Ramkumar Sembaiyan [12-Oct-2024]

1. Database Selection for ALPR System

For an **Automatic License Plate Recognition (ALPR)** system, the choice between **SQL** and **NoSQL** databases depends on the specific requirements of our system, including data structure, scalability, and query needs. Here's a breakdown of when we can use each type and the pros and cons of both for an ALPR system:

Criteria	SQL (Relational)	NoSQL (Non-relational)
Data structure	Structured (fixed schema)	Unstructured/semi-structured (flexible schema)
Scalability	Vertical (scale-up with more powerful servers)	Horizontal (scale-out across many servers)
Data relationships	Ideal for complex relationships (e.g., vehicles, plates, events)	Not as good for relational data, better for unstructured
Data integrity	Strong (ACID compliance)	May sacrifice consistency for performance
Performance	Good for moderate data sizes, but can slow down with large-scale operations	Optimized for high-velocity, large-scale data
Complex querying	Excellent (supports JOINs, aggregations)	Limited (complex queries can be inefficient)

Recommendations:

If ALPR system requires **structured data with strong relationships** (e.g., vehicles, cameras, license plates, violations), SQL (e.g., MySQL, PostgreSQL) would be a better fit.

If ALPR system needs to handle **massive amounts of unstructured or semi-structured data** (e.g., images, high-velocity logs) and scalability is a concern, NoSQL (e.g., MongoDB, Cassandra) may be more appropriate.

Comparisons between Popular relational DB :

Criteria	MySQL	SQL Server
Cost	Free and open-source	Paid, except for limited Express version
Community Support	Extensive, large open-source community	Strong, but focused on enterprise users
Performance	Great for moderate traffic and read-heavy applications	Optimized for large-scale data with enterprise-grade performance
Scalability	Scales well with tuning, but more suited for smaller datasets	Easily scalable for high transaction volumes and large datasets
Features	Basic features, lacks advanced analytics	Advanced features (in-memory processing, data warehousing)
Security	Basic security features	Strong security (row-level security, encryption)
Backup & Recovery	Requires manual setup for point-in-time recovery	Advanced backup features (point-in-time, transaction logs)
Cloud Compatibility	Widely supported (AWS, Azure, GCP)	Best with Microsoft Azure and integration with other Microsoft tools
Ease of Use	Easy to set up and maintain	More complex setup and maintenance
Integration	Works with most systems and cloud platforms	Seamless integration with Microsoft ecosystem (Power BI, Azure, etc.)

Popular Cloud Databases

Cloud DB	Description	Advantages
Amazon RDS (MySQL or SQL Server)	Managed relational database service from AWS, supports both MySQL and SQL Server.	- Fully managed
		- Auto-scaling
		- Multi-AZ for high availability
Azure SQL Database	Managed SQL Server in the cloud, part of Microsoft Azure.	- Best for SQL Server
		- Seamless integration with Power BI
		- High security and scalability
Google Cloud SQL (MySQL)	Managed MySQL service from Google Cloud Platform (GCP).	- Fully managed
		- Auto-scaling
		- Multi-regional deployments

Amazon Aurora (MySQL-compatible)	High-performance managed MySQL-compatible database from AWS.	- Better performance than MySQL
		- Auto-scaling and high availability
Firebase Realtime Database (NoSQL)	Managed NoSQL cloud database by Google, designed for real-time apps.	- Great for real-time updates
		- Simplified schema for ALPR events
MongoDB Atlas (NoSQL)	Managed cloud version of MongoDB, a popular NoSQL database.	- Document-based, ideal for flexible data structures
		- Auto-scaling

Recommendation

For **budget-conscious ALPR projects** that don't require advanced features, **MySQL** would be a better choice. However, if your ALPR system grows in scale and you need enterprise-level features, **SQL Server** would be the preferred option despite the cost.

We shall start with mysql DB and upgrade to enterprise /Cloud DB as data grows.

Proposed Database Tables for ALPR

To collect various data for analysis and perform datascience, we may start with following tables.

1. Camera Location Table (infer_alprcameraloc)

This table stores details about the ALPR cameras, including their location and technical specifications. Each camera is uniquely identified by a camera_id. This table is essential for tracking the hardware configuration and placement of cameras across different locations, enabling efficient diagnostics and maintenance.

Field	Description	Type
camera_id	Unique identifier for the camera	INT
Type	Type of camera (e.g., fixed, mobile)	VARCHAR
Resolution	Camera resolution (e.g., 1080p, 720p)	VARCHAR
FPS	Frames per second captured by the camera	INT
locationaddress	Physical address or location description	VARCHAR
latitude	latitude altitude in meters	DECIMAL
longitude	Longitude of the camera location	DECIMAL
cameraposition	Camera mounting position (e.g., overhead, side)	VARCHAR

2. ALPR Details Table (infer_alpr_details)

The infer_alpr_details table captures the key results from the ALPR system, including the license plate number, associated camera, and additional vehicle-related details. This table enables analysis of ALPR results, confidence levels, and vehicle classification. The image_path links to the actual image stored on disk.

Field	Description	Type
ALPRId	Unique identifier for each ALPR record	INT
camera_id	Reference to the camera that captured the data	INT
V_Platenumber	Detected vehicle license plate number	VARCHAR
timestamp	Time when the vehicle was recognized	TIMESTAMP
image_path	Path to the stored image (in the server directory or cloud S3 bucket) captured by the camera	VARCHAR
Confidencescore	Confidence score of the ALPR detection	DECIMAL
VehicleType	Type of vehicle (e.g., car, truck, bike)	VARCHAR
FuelType	Fuel type of the vehicle	VARCHAR
RegistrationState	Vehicle's registration state or region	VARCHAR
Error	Error encountered during recognition, if any	VARCHAR
EventsCalled	Events triggered by the ALPR system	VARCHAR

3. ALPR Events Table (infer_alpr_events)

The infer_alpr_events table tracks each event triggered by the ALPR system, including the steps taken during event handling and any custom data fields for additional insights. This table provides a detailed audit trail of ALPR-triggered events after detecting the license plates, supporting monitoring and troubleshooting of the system.

Field	Description	Type
EventId	Unique identifier for each event	INT
ALPRId	Reference to the related ALPR record	INT
camera_id	Camera involved in the event	INT
V_Platenumber	License plate involved in the event	VARCHAR
timestamp	Timestamp of the event	TIMESTAMP
EventStepid	Step identifier in the event workflow	INT
EventStatus	Status of the event (e.g., success, failure)	VARCHAR
Error	Error encountered, if any	VARCHAR
Customfield1	Custom field for additional event-specific data	VARCHAR
Customfield2	Additional custom data field	VARCHAR
Error	Field for storing specific error details	VARCHAR
Customfield3	Third custom field for other details	VARCHAR

4. ALPR Exceptions Table (infer_alprexception)

This table stores exceptions or errors encountered by the ALPR system, as well as recommended next steps for resolution.

Field	Description	Type
Exceptionid	Unique identifier for the exception	INT
ALPRId	Reference to the related ALPR record	INT
Camera_Id	Camera involved in the exception	INT
Exception	Description of the exception	VARCHAR
NextSteps	Recommended steps to resolve the exception	VARCHAR

‘* Attached CSV files to show how the DB tables look.

Important Data collection and Ingestion Consideration in the Design:

If the ALPR system is expected to scale significantly—handling large volumes of transactions, data, and traffic—switching to a **messaging architecture** becomes crucial to avoid these potential bottlenecks.

Here’s when messaging architecture should be considered as the system scales:

1. Reducing Database Load
2. Mitigating Database Locking Issues
3. Enhancing Scalability
4. Asynchronous Processing
5. ALPR in Edge computing

Example of a Messaging Architecture Flow

1. **Cameras** capture vehicle data and send it to **Kafka** (or any message broker using).
2. **Kafka** queues the messages and buffers them for the processing service.
3. The **ALPR Processor** subscribes to the Kafka queue, processes the vehicle data using the ALPR model, and extracts relevant information.
4. The processed data (e.g., plate number, image path) is sent to a **SQL/NoSQL Database** for storage.
5. The **real-time dashboard** (Power BI/Grafana) queries the database or subscribes to an update service to visualize the data in real-time.

Visualization -Report and Dashboard Considerations

When it comes to developing dashboards for Automatic License Plate Recognition (ALPR) systems, several tools cater to different needs, capabilities, and expertise levels. Here's a brief overview of some popular options:

Power BI: A powerful, user-friendly analytics tool offering extensive data visualization capabilities and a free desktop version.

Grafana: An open-source platform ideal for real-time monitoring and visualization of time-series data, easily extensible with plugins.

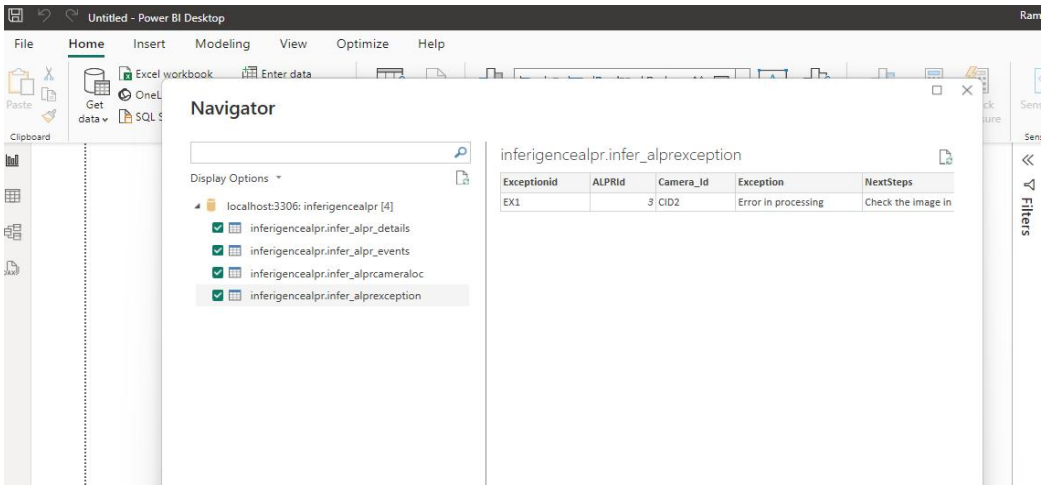
Metabase: A straightforward open-source business intelligence tool that allows for easy dashboard creation and data exploration without advanced skills.

Tableau: A robust data visualization tool known for its advanced analytics and rich features for creating interactive dashboards (subscription-based).

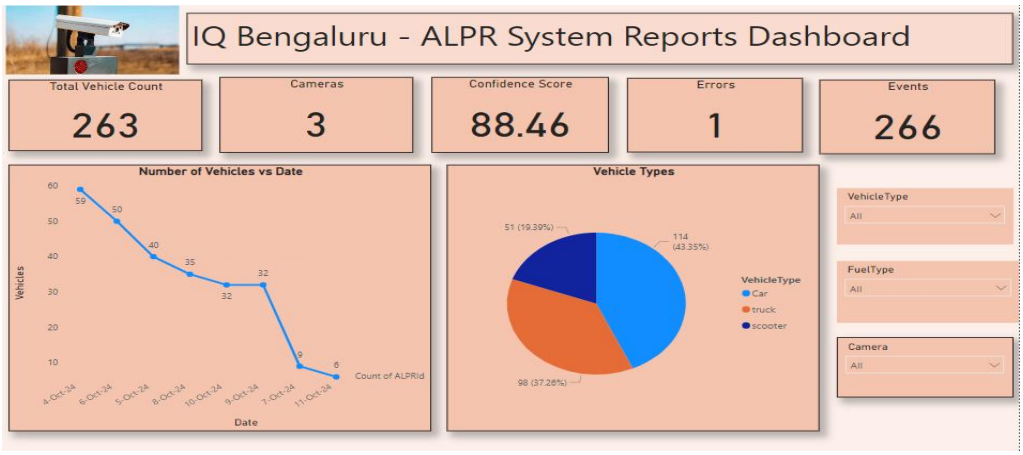
Custom Developed Apps: Tailored applications that provide complete control over dashboard functionality and integration, requiring development resources.

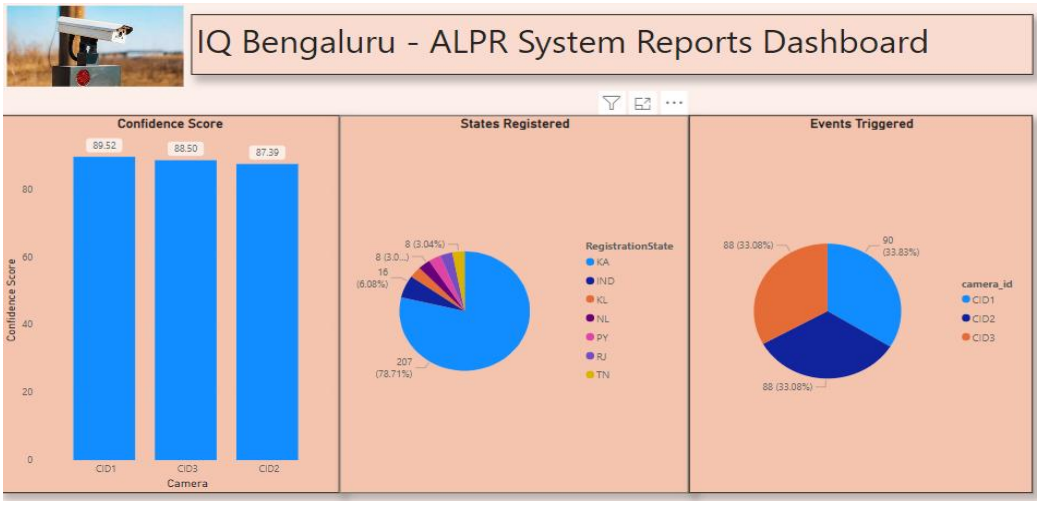
To begin our experience with data visualization, we will start with Power BI Desktop, which offers a user-friendly interface and robust features at no cost

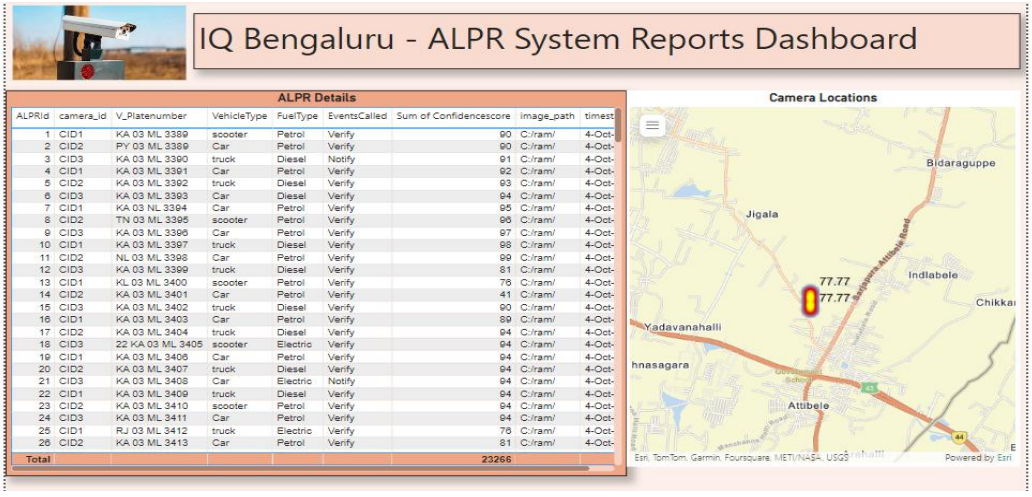
Note: Below is only for examples: we need to deep dive on what visualizations are required for real time in ALPR



Visualizations [PowerBI Desktop Reports]

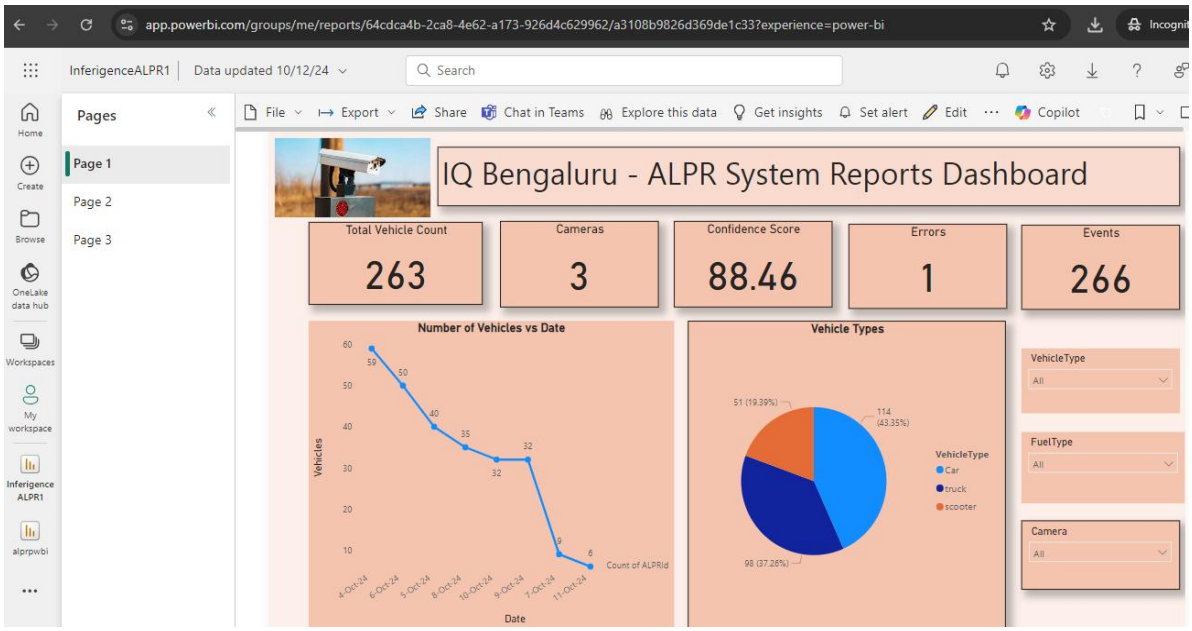






If we finalize Power BI service for Dashboards, we can use the premium with nominal charge (20$ per month) for all the products in the organization, which has many features including autorefresh, sharing the dashboards ,subscribe alert etc. Until then we can use the free desktop version. We can also explore Graffana, and other custom dashboards tools if PowerBI is not suitable.

Reports in Power BI Service (cloud based)



The final dashboard may look like below in power BI service portal