# **Case Study: ChargeNet EV – Understanding DBMS Concepts and Architecture**

## **Background**

India is witnessing rapid growth in the electric vehicle (EV) sector. With government support and increasing adoption, the need for a reliable **EV charging network** has become urgent.

**ChargeNet EV** is a national platform that connects:

* **Drivers** who use a mobile app to locate chargers, reserve slots, and pay seamlessly.
* **Operators** who own charging stations and want analytics on usage, revenue, and maintenance.
* **The Core Platform** that manages discovery, reservations, billing, and regulatory compliance.
* As the number of EVs grows, ChargeNet must handle:

1. **1 million+ driver accounts**
2. **15,000 charging stations** across 120 cities
3. **300,000+ daily telemetry events per station**
4. **25,000 concurrent charging sessions**
5. This scale requires a **robust DBMS architecture** that balances speed, reliability, scalability, and compliance.

## **Problem Statement**

ChargeNet needs to design a **Database Management System (DBMS)** that supports:

1. **Fast Discovery:** Find nearest chargers with filters (location, power, price, availability).
2. **Reliable Reservations & Sessions:** Reserve ports, record energy usage, generate bills.
3. **Payments & Settlements:** Ensure accurate UPI payments, refunds, and operator settlements.
4. **Analytics:** Provide operators with insights on uptime, utilization, and revenue.
5. **Compliance:** Keep immutable logs for government regulations.
6. **Scalability:** Handle massive telemetry data while keeping transactions secure.

Students are asked to map this problem to **DBMS concepts and architecture**.

## **Step 1: The Three-Schema Architecture**

* ****External Level (User Views):****

Driver App: Shows only required data like charger location, price, availability.

Operator Dashboard: Shows uptime, revenue, tariff rules, and maintenance reports.

Finance & Compliance: Focuses on invoices, settlements, and audit logs.

* ****Conceptual Level (Logical View):****

Entities: **User, Vehicle, Operator, Station, Connector, Reservation, Session, Tariff, Invoice, Payment, Ledger, Maintenance, Audit Event**.

Relationships: Users own vehicles, operators own stations, sessions link drivers to connectors, payments link to invoices.

* ****Internal Level (Physical Storage):****
* Core relational database for structured transactions.
* Time-series database for telemetry (real-time charging data).
* Partitioning data by region or time for efficiency.
* Indexes for faster retrieval of discovery and billing data.

## **Step 2: Key DBMS Concepts in the Case**

**Schema vs Instance**

Schema: The design of entities like Session, Invoice, Payment.

Instance: Today’s actual charging sessions and payments stored in the database.

**Data Independence**

Logical Independence: Adding new connector types or tariff rules does not break driver apps.

Physical Independence: Storing telemetry in a new time-series system doesn’t affect applications.

**Data Models**

**Relational Model** for invoices, payments, reservations.

**Document/NoSQL** for flexible tariff rules or telemetry.

**Hybrid Polystore Approach** combining both.

**DBMS Components in Action**

Query Processor: Optimizes search for nearest stations.

Transaction Manager: Ensures reservations and payments are ACID-compliant.

Storage Manager: Handles indexing, partitioning, recovery.

Catalog/Metadata: Manages schema versions, tariff history, and audit logs.

## **Step 3: Example Scenarios**

**Reservation Process**

Driver selects a charging port → DBMS checks availability → Reservation is stored → Port marked held → Transaction committed.

**Session Lifecycle**

Driver plugs in → DBMS records start time → Telemetry feeds energy usage → Session ends → Bill generated → Payment linked.

**Payment and Ledger**

Payment request sent → Upon success, entries recorded in the financial ledger (both debit and credit) → Ensures accuracy and compliance.

**Analytics and Reporting**

Operators get dashboards with utilization % and uptime.

Regulators receive automated audit logs of every tariff change.

## **Step 4: Challenges and Solutions**

**Scalability:**

Solution: Partitioning, replication, and distributed queries for nationwide data.

**Performance:**

Solution: Geospatial indexing for location-based discovery; caching frequent queries.

**Consistency vs Speed:**

Solution: Use relational DB for financial data (strong consistency) and time-series DB for telemetry (high-speed inserts).

**Security:**

Solution: Role-based access (driver, operator, finance, regulator).

Store personal data in encrypted form.

## **Step 5: How This Case Explains DBMS Concepts**

**Levels of Abstraction:** Show how different stakeholders see tailored views.

**Schemas & Instances:** Differentiate blueprint vs actual data.

**Data Independence:** Demonstrate flexibility in evolution.

**Architecture Components:** Clarify role of query processor, transaction manager, storage manager, and catalog.

**Models:** Highlight relational vs NoSQL use-cases.

## Reflection Questions for Students

How does ChargeNet ensure logical and physical data independence?

Which DBMS components are most critical for payment integrity? Why?

Why is a hybrid (relational + NoSQL) model more suitable than only relational?

If the government mandates real-time tariff display, which schema/view changes would be needed?

What risks arise if telemetry is not separated from financial data?

## Conclusion

The **ChargeNet EV case study** demonstrates how **Database System Concepts and Architecture** are not abstract theories but **critical enablers of modern digital infrastructure**. By applying the **three-schema architecture, schemas vs instances, data independence, and DBMS components**, students can appreciate how theory translates into real-world problem solving.

As India transitions to electric mobility, platforms like ChargeNet will depend heavily on **intelligent DBMS design** — making this case study a bridge between classroom learning and industry practice.