EV Charging Station Management System

Database Design Document

Team Project Submission
Course: Database Management Systems
Northeastern University

Group Members:

- Devanshu Chicholikar
- Saurabh Kashyap
- Joseph Alex Chakola
- XingXing Xiao

---

1. Business Problem

The electric vehicle market is growing rapidly, but charging infrastructure hasn't kept up. EV drivers today face a frustrating situation where they drive to charging stations only to find all spots occupied. There's no way to reserve a charger in advance, no clear pricing information, and no system to notify them when charging is complete. For station operators, the problem is just as bad. They deal with no-shows that waste capacity, cars that stay plugged in long after charging finishes, and manual tracking of usage and billing.

We're designing a database to solve these problems. Think of it like a reservation system for charging stations, similar to how OpenTable works for restaurants or how parking apps work for garage spots. Users can see available chargers, book them in advance, get notifications, and pay automatically. Station operators get better utilization and automated billing.

Our team chose this problem because we're interested in sustainability and the practical challenges of EV adoption. As more people buy electric cars, efficient charging infrastructure becomes critical. A well-designed database can make the difference between a smooth experience and a frustrating one.

---

2. What Our Database Needs to Do

We designed our system to handle five main things:

User and Vehicle Management
The system needs to know who's using it and what vehicles they own. Some users might have multiple cars (like a family with two EVs), and we need vehicle details like what kind of charging connector it uses, because not all chargers work with all cars.

Station and Equipment Tracking
We need to know where charging stations are located, what hours they're open, and what types of chargers they have. Each station has multiple individual charging points, and each point needs its own status (available, occupied, or broken). This lets users search for nearby stations and see real-time availability.

Reservation System
Users should be able to book charging slots in advance. The system needs to prevent double-booking, support cancellations with reasonable policies, and handle walk-ins who show up without a reservation. We also want to track no-shows so we can enforce penalties for users who repeatedly skip their reservations.

Real-Time Notifications
The system should send confirmation emails when someone books, reminder texts before their time slot, and alerts when charging finishes. This reduces no-shows and encourages people to unplug promptly so others can use the charger.

Billing and Payment
After each charging session, the system needs to calculate what to charge based on how much electricity was used, how long the car stayed plugged in after charging finished, and what membership tier the user has. It should support pay-as-you-go for casual users and monthly subscriptions for frequent users. All payments should happen automatically.

---

3. Our Database Structure

We organized our database into 14 entities. Here's how they connect:

```
 Users and their vehicles
    ↓
Make reservations at charging stations
    ↓
Show up and plug in (creates a charging session)
    ↓
System generates invoice and processes payment
    ↓
Operators monitor stations and handle maintenance
```

---

4. The Entities

USER

This is anyone who uses the system. It stores basic information like name, email, and phone number. We also track what type of account they have (regular customer, premium subscriber, fleet manager, or system admin) and their account status (active, restricted, or suspended).

The account type matters because it affects pricing. Regular users pay full price, premium subscribers get a discount, and fleet managers get bulk pricing. Account status lets us temporarily block users who have unpaid bills or too many no-shows without deleting their data.

VEHICLE

Every electric car in the system gets recorded here. We store the make and model, battery size, and most importantly, what type of charging connector it has. A Tesla uses CCS connectors while a Nissan Leaf uses CHAdeMO. Before letting someone reserve a charger, we check that their vehicle is compatible with the equipment.

We treat vehicles as separate from users because one person might own multiple cars. The vehicle stores UserID to track ownership.

STATION

A charging station is a physical location, like a parking lot or rest stop, where charging equipment is installed. We store the name, address, GPS coordinates, and operating hours. Some stations are open 24/7, others have specific hours like 6am to 10pm.

The GPS coordinates are important because they let users search for stations near them and get directions. The operating hours prevent people from booking times when the station is closed.

CHARGING_POINT

This is an individual piece of charging equipment. Each station might have 5 or 10 of these. When someone makes a reservation, they're reserving a specific charging point for a specific time.

We track what type of charger it is (slow Level 2 or fast DC charger), what connector type it has, and its current status (available, occupied, or out of service). The status updates in real time as people plug in and unplug.

RESERVATION

When a user books a charging slot, we create a reservation. It records who made it, which vehicle they'll bring, which charging point they reserved, and the time window they need.

The reservation tracks its status through its lifecycle: active when first created, completed when the user shows up and charges, cancelled if the user cancels, or no-show if they don't arrive.

One important point: we keep reservations separate from actual charging sessions. A reservation is a plan to charge. A session is what actually happened. If someone no-shows, we have a reservation but no session. This separation keeps the logic clean.

CHARGING_SESSION

This records what actually happened when someone plugged in and charged. It links to the reservation (if there was one), tracks which charging point was used, which vehicle was charged, when charging started and ended, and how much energy was consumed.

The key insight here is tracking idle time separately. We measure when the battery reached full charge versus when the user unplugged. The difference is idle time where they're blocking the charger. We charge for this time to encourage prompt unplugging.

Energy consumed (measured in kilowatt-hours) is our primary billing metric. It's the fairest approach because you pay for what you use.

SUBSCRIPTION_PLAN

We offer different membership tiers: Regular (free), Premium ($9.99/month), and Fleet Manager (custom pricing). Each tier has different benefits like discount rates, maximum concurrent reservations, and how far in advance you can book.

This entity also stores all the pricing information: base rate per kWh, idle fee rate, no-show penalties, and cancellation fees. By keeping rates in the database instead of hardcoding them, we make it easy to adjust pricing without changing application code.

USER_SUBSCRIPTION

This connects users to their subscription plans and tracks when they subscribed, whether it's still active, and when it ends.

We keep this separate from the USER table because subscriptions change over time. Someone might start as Regular, upgrade to Premium, then downgrade back to Regular. Each of these changes creates a new record with start and end dates. This history is important for billing accuracy.

INVOICE

An invoice is a bill. Every time something billable happens (a charging session completes, a subscription renews, someone cancels late, someone no-shows), we generate an invoice.

Invoices have different types. Session charges are the most common. Subscription fees are monthly recurring charges. Cancellation fees happen when someone cancels too late. No-show fees happen when someone doesn't arrive.

Each invoice shows the total amount owed and tracks payment status: pending when first created, paid once payment clears, or overdue if payment isn't received by the due date.

The invoice links to different source entities depending on type: SessionID for charging sessions, ReservationID for penalty fees, SubscriptionID for monthly billing. At least one of these is filled, the others are null.

PAYMENT

A payment is a transaction attempt. One invoice might have multiple payments if the first attempt fails and the user retries, or if they make partial payments.

We record which invoice is being paid, which payment method was used, the amount, the timestamp, a transaction ID from the payment gateway (like Stripe), and the status (success, failed, pending, or refunded).

Keeping payments separate from invoices creates an audit trail. If someone disputes a charge, we can show exactly what happened: first payment failed because the card was declined, second payment succeeded two days later.

PAYMENT_METHOD

Users can save payment methods for automatic billing. We store what type it is (credit card, debit card, digital wallet), the last 4 digits for display, and whether it's their default payment method.

We never store full card numbers for security compliance. Instead we use tokenization through the payment gateway. The default flag tells the system which card to use for automatic charges.

NOTIFICATION

Every message the system sends gets recorded here. This includes confirmation emails, reminder texts, completion alerts, and payment notifications. We track who received it, what it was about, and when it was sent.

Recording notifications helps with debugging (if a user says they never got a reminder, we can look it up) and measuring effectiveness (do notifications actually reduce no-shows?).

MAINTENANCE_RECORD

This tracks service history for charging equipment. When a charger breaks down or needs routine maintenance, we create a record with the date, description, technician name, and status (complete or in progress).

This helps operators schedule preventive maintenance, document downtime for warranty claims, and identify problematic equipment that needs replacement.

OPERATOR

Represents station operators or administrators who monitor the charging network and manage maintenance activities. Operators can view station performance, oversee maintenance records, and access system analytics.

This entity separates operational staff from end users, allowing different access permissions and tracking who performed which maintenance or administrative actions.

---

5. How the Entities Connect

USER to VEHICLE (One-to-Many)
One user can own multiple vehicles. Each vehicle belongs to one owner. For this project, we simplified to accept that vehicles have single owners (good enough for families and individual users).

USER to RESERVATION (One-to-Many)
One user can make many reservations over time. Each reservation belongs to exactly one user.

STATION to CHARGING_POINT (One-to-Many)
One station contains many charging points. Each point belongs to one station.

CHARGING_POINT to RESERVATION (One-to-Many)
One charging point can be reserved many times across different time windows. We enforce that no two active reservations can overlap for the same point.

RESERVATION to CHARGING_SESSION (One-to-Zero-or-One)
A reservation might result in a charging session if the user shows up, or might not if they no-show. This optional relationship is why we keep them as separate entities.

CHARGING_SESSION to INVOICE (One-to-One)
Every completed charging session generates exactly one invoice.

USER to SUBSCRIPTION_PLAN (Many-to-One through USER_SUBSCRIPTION)
Many users can subscribe to one plan. The USER_SUBSCRIPTION entity tracks who's subscribed to what and handles subscription lifecycle (start date, end date, status).

INVOICE to PAYMENT (One-to-Many)
One invoice can have multiple payment attempts. The first payment might fail, then succeed on retry. Each attempt creates a payment record.

USER to PAYMENT_METHOD (One-to-Many)
One user can save multiple payment methods. The system uses the default method for automatic billing.

USER_SUBSCRIPTION to INVOICE (One-to-Many)
Each subscription generates monthly invoices for the recurring fee.

RESERVATION to INVOICE (One-to-Zero-or-One)
Reservations may generate invoices if there's a cancellation fee or no-show penalty.

OPERATOR to MAINTENANCE_RECORD (One-to-Many)
One operator can oversee multiple maintenance activities.

---

6. Key Design Decisions

Why separate RESERVATION from CHARGING_SESSION?
Reservations are plans, created in advance. Sessions are reality, created at arrival. Keeping them separate lets us handle no-shows cleanly (reservation exists, session doesn't) and walk-ins (session exists, reservation doesn't).

Why include INVOICE between SESSION and PAYMENT?
Invoices explain what you're charging and why. They show itemized charges (energy cost, idle fees, discounts). They can exist without sessions (for penalty fees and subscriptions). They support multiple payment attempts when the first payment fails. Without invoices, the billing workflow breaks for edge cases.

Why track USER_SUBSCRIPTION separately?
Subscriptions change over time. Users upgrade, downgrade, cancel. If we stored plan information directly in USER table, each change would overwrite previous data. USER_SUBSCRIPTION preserves subscription history with start dates, end dates, and status tracking.

Why store pricing in SUBSCRIPTION_PLAN?
Each plan defines not just its monthly fee and discount rate, but also the base rates that apply to its subscribers. This centralizes pricing logic and makes rate adjustments easier. Different plans can have different rates (Regular pays $0.40/kWh, Premium pays $0.28/kWh after discount).

Why track idle time separately?
After batteries finish charging, users often leave cars plugged in, blocking access for others. By measuring idle time (time between charge completion and unplugging) and charging for it, we create incentives for prompt unplugging while being fair.

Why include OPERATOR entity?
Station operations require human oversight for maintenance, troubleshooting, and system monitoring. Separating operators from end users allows different access permissions and tracks accountability for maintenance activities.

---

7. Business Rules

No double-booking: Before accepting a new reservation, we check whether that time window conflicts with any existing active reservation for that charging point.

Vehicle compatibility check: Before accepting a reservation, we verify that the vehicle's connector type matches the charging point's connector type.

Auto no-show detection: 15 minutes after a reservation's start time, if no charging session has started, the system marks it as a no-show and charges the $10 fee.

Idle fee calculation: After the battery reaches full charge, we count idle time. Premium users get 10 free minutes, then pay $0.50/minute. Regular users pay immediately.

Cancellation policy: Free cancellation 1+ hour before reservation. $3 fee if cancelled within 1 hour. $10 fee for no-shows.

Automatic invoice generation: When a charging session completes, the system immediately generates an invoice with all charges calculated.

Auto-payment attempt: When an invoice is created, the system tries to charge the user's default payment method immediately.

---

8. Why This Design Works

Our conceptual model has 14 entities, which exceeds the 10-entity requirement. More importantly, each entity serves a clear purpose and connects logically to the others.

The modular structure (users, stations, reservations, sessions, billing) makes the system understandable. Someone new to the project can quickly grasp that users make reservations at stations, reservations turn into charging sessions, and sessions generate invoices.

The separation of planning entities (RESERVATION) from execution entities (CHARGING_SESSION) from billing entities (INVOICE, PAYMENT) follows good database design principles. Each entity represents one concept and one lifecycle.

The pricing model is flexible enough to support future changes without schema modifications. Want to add a "Gold" tier? Just insert a new row in SUBSCRIPTION_PLAN. Want to adjust rates? Update the plan's rate values.

Most importantly, this design solves real problems. EV drivers get predictable access to charging. Station operators get better utilization and automated billing. The system handles the complexity so users just see a simple, smooth experience.

We're confident this conceptual model provides a solid foundation for building an actual EV charging management system.

Drive Link : https://drive.google.com/file/d/1LZu0PHrNYvao-j1FphIokI9uf1-vx8Hl/view?usp=sharing