DBMS Project – 1: Parking Lot Application

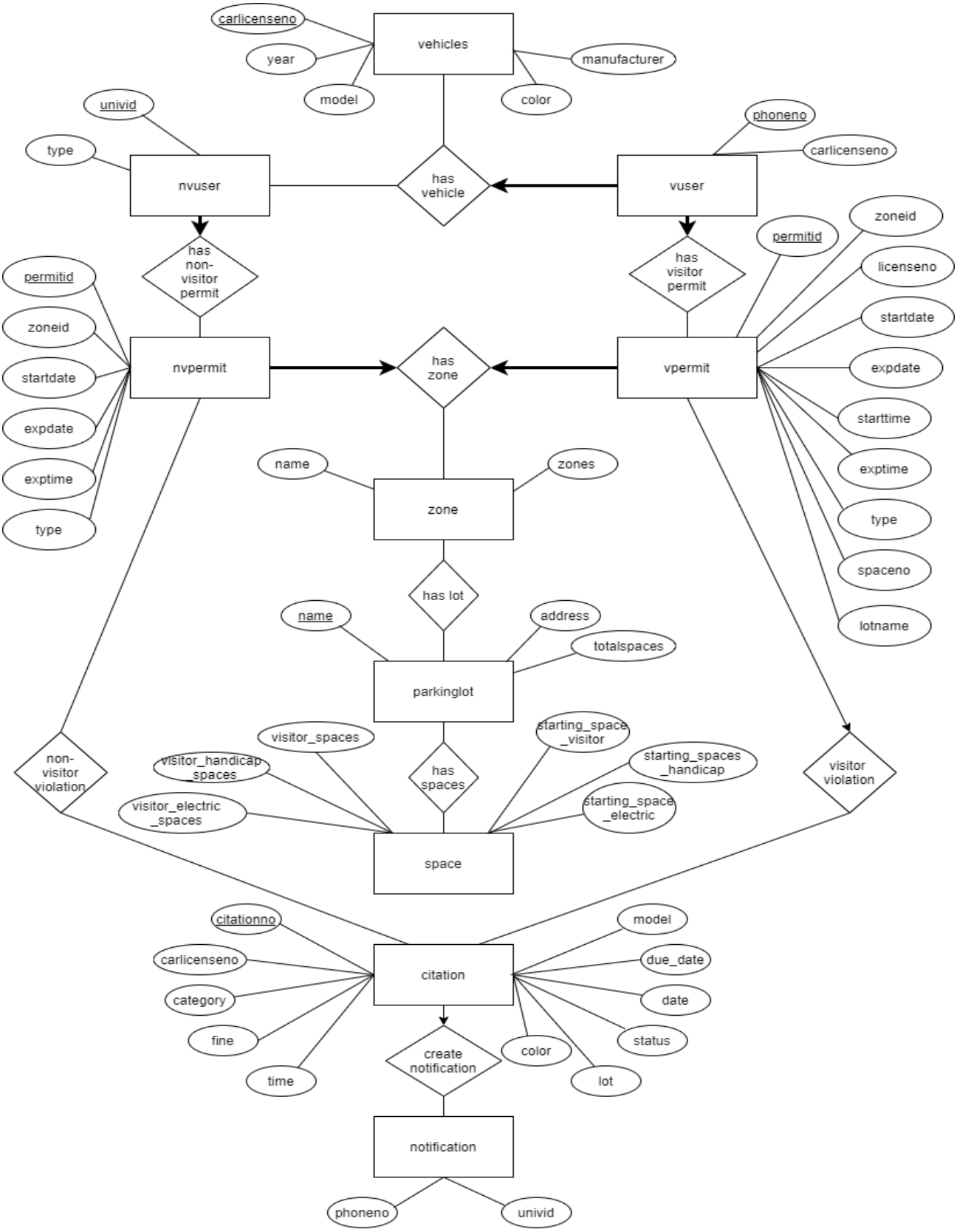
Team Members:

1] Shubham Shah – 200312986 – sashah6

2] Prathmesh Shivrame – 200316571 – pkshivra

3] Radhika Angolkar – 200315194 – rangolk

1. ER Design:


carlicenseno
year
model
vehicles
manufacturer
color
univid
type
nvuser
has vehicle
vuser
phoneno
carlicenseno
has non-visitor permit
has visitor permit
permitid
zoneid
licenseno
startdate
expdate
starttime
exptime
type
spaceno
lotname
permitid
zoneid
startdate
expdate
exptime
type
nvpermit
has zone
vpermit
name
zones
zone
has lot
name
address
totalspaces
parkinglot
visitor_spaces
visitor_handicap_spaces
visitor_electric_spaces
has spaces
starting_space_visitor
starting_spaces_handicap
starting_space_electric
non-visitor violation
visitor violation
space
citationno
carlicenseno
category
fine
time
citation
model
due_date
date
status
color
lot
create notification
notification
phoneno
univid

2. Relational Design/Constraints/SQL Files:

A] Relationship Design:

**Bold and Underlined indicate Primary Keys, while underlined indicate Foreign Keys

citation (**citationno**, carlicenseno, category, fine, time, lot, status, date, due_date, model,color)

notification (citationno, phoneno, univid)

nvpermit (**permitid**, univid, zoneid, startdate, expdate, exptime, type)

nvuser (**univid**, type)

parkinglot (**name**, address, totalspaces)

space (parkignlot, visitor_spaces, visitor_handicap_space, visitor_electric_space, starting_space_visitor, starting_space_handicap, starting_space_electric)

vehicles (**carlicenseno**, year, model, color, manufacturer, univid)

vpermit (**permitid**, zoneid, licenseno, startdate, expdate, starttime, exptime, type, spaceno, lotname)

vuser (**phoneno**, carlicenseno)

zone (**name, zones**)

authenticate (**id**, password): Created only for storing login id and password of university and UPS users, not part of the parking lot design.

B] Table Constraints:

Citation: Citationno is the Primary Key, Citationno is set to Auto_Increment rest all the attributes are Not NULL apart from lot.

Notification: citationno is Foreign Key referencing to Citation (citationno)

Nvpermit: permitid is the Primary key, univid is Foreign Key referencing to nvuser (univid), type has a default set to regular if no input type specified, rest all are set to Not Null

Nvuser: univid is the Primary key

Parkinglot: name is the Primary Key, rest all attributes are set to Not Null

Space: parkinglot is the Foreign Key referencing parkinglot (name), rest all are Not Null

Vehicles: carlicenseno is the Primary Key, while univid is Foreign Key referencing nvuser (univid) which can be Null, other attributes are set to Not Null.

Vpermit: permitid is the Primary Key, licenseno is Foreign Key referencing vehicles (carlicenseno), lotname is Foreign Key referencing parkignlot (name), type is set to regular as default, rest all attributes are Not Null.

Vpermit also consists of Triggers to update the space table on every Insert.

Vuser: phoneno is the Primary key, rest all attributes are set to Not Null

Zone: name, zones is the Composite Primary Key here

Authenticate: id is the Primary Key; password is set to Not Null

C] Constraints not implemented as part of table definitions:

In terms of the constraints not implemented from the table/schema/queries level, we have tried to keep all the things at the database logic rather than handling it through application logic

NOT IMPLEMENTED AS PART OF DATABASE BUT IMPLEMENTED THROUGH APPLICATION:

- Checking the validity for students parking before and after 5 P.M (Office hours)

- Triggering the Issue Citation () function to create citation on every violation of the rule (it creates citation automatically from backend when a violation occurs with no additional manual efforts though picking up the attributes needed by running queries to add a new row in citation table)

- Updating the Notification table on every new Citation (this is done indeed by adding an insert query to form and run automatically on backend itself for every new citation)

- Creating a unique permit id for every new permit issued (as the creation of this id had specific requirements)

SOME CONSTRAINTS IMPLEMENTED AS PART OF DATABASE:

- having a unique citation number where the column auto increments every time on new row addition

- allowing employees to have at most 2 cars registered and students only 1.

- taking default type as regular when adding a permit if the type is not specified.

- adding triggers for every new visitor permit generated to update the space table with the current count of visitor (regular, handicap, electric) spaces and the next starting space number which will be allotted.

- Allocating space numbers for specific types and auto-incrementing it on every permit assigned to visitors.

- NOT NULL for multiple attributes where data is needed.

- Composite key in zones table to map one parking lot to multiple zones

- Allowing univid in vehicles table to be Null to distinguish the visitor vehicles and non-visitor vehicles.

D] SQL Files: All SQL files are provided inside the zip folder

- CREATE_schema.sql: Tables, constraints

- INSERT_schema.sql: Queries for populating the tables with the sample data

- Triggers: Separate file

- create_and_insert_schema.py: This script will automatically take all the inputs from the above files and create the Data structure, Triggers, and populate it with sample data provided.

3. Functional Dependencies:

Citation:

- citationno => All other attributes
- carlicenseno => model, color
- category => fine

From the Functional dependencies we can conclude that if needed the model, color and fine columns can be removed from the citation table. Indeed, a new table can be created which maps the category to fine and carlicenseno is already being mapped to model and color in vehciles table.

Notification:

- citationno => phoneno, univid

Nvpermit:

- permitid => All other attributes

Nvuser:

- univid => type

Parkinglot:

- name => All other attributes

Vehicles:

- carlicenseno => All other attributes

Vpermit:

- permitid => All other attributes

Vuser:

- phoneno => carlicenseno

Space:

- This is a dynamic table to keep the count of visitor spaces (regular, handicap and electric) spaces left in a specific lot along with the next space number to allot. So no dependencies here.

Zone:

- This is also a dynamic table where the mapping of multiple zones to a single parking lot is done which keeps changing and does not signify a dependency.

4. Justification of design choices:

The design was made keeping the mind how the different roles of non-visitor and visitor can be brought together and the common elements between them. It was seen that even though being of any category the main entity that binds both together was vehicles. We had designed the vehicles table in such a way that it can distinguish the car for a non-visitor and a visitor.

Once the Vehicle table was created it was decided to create a 2-way design where one direction points toward the non-visitor user and the other to visitors. This way the data can be segregated and can be used as per the request functions. This helped us to skim through the specific table each time when we had to check something from the database rather than querying through the whole data of the user.

Permits for non-visitor and visitor were also tracked in separate tables, this helped us to assign specific extra columns for tracking the visitors like the specific lot they are in and the space number assigned to them.

Again in this 2-way scenario, the common entity which binds the permits was zone. Each permit had a specific zone they were allowed to be part of. These zones were mapped to specific parking lots. A composite key was created for the zone table with entities like names and zones. This helped to map the parking lots to multiple zones where the same zones could be mapped to multiple parking lots.

The zone table was then mapped with the parking lot table with addresses and total spaces. The parking lot table was then related to the spaces table. The spaces table was a specific entity created with multiple columns which helped us track the number of visitor spaces in each lot along with the handicapped and electric type spaces. It also had a starting space number for each type and helped to check if free visitor spaces and specific type spaces were available inside each lot.

As the project was concerned with the visitor type spaces only we also introduced triggers where whenever there was a new permit created for the visitors the space table was automatically being updated accordingly in terms of the number of spaces available for visitors and each type along with the space number for next car when it arrives. Accordingly, on every exit, this table was updated with an increase in spaces for the lot where this car was parked into.

Following the 2-way method again whenever there was a violation taking space concerning some visitor or non-visitor permit validity check a citation was generated. The citation created a tuple inside the notification table where citation number along with the phone no for visitors and university id for non-visitors was mapped to track the violators.

In context, the design was highly based on following a 2-way method and segregating the visitor and non-visitor entities apart from some common elements between the two. Segregating the entities helped us in keeping a proper tab of the data and help in limiting the queries and search for the correct output.

5. Assumptions:

As part of implementation liberty, we have assumed that the permit violation for an employee will be checked for its zone and corresponding student zone only.

For Example:
If an employee is allotted zone = “A”, he can only park in zones “A” and “AS”.
A violation will be reported if the employee car is found parked in zones “B “,”C”,” D”,”BS”,”CS”,”DS”.

Abbreviations:

nvuser	Non Visitor User
nvpermit	Non Visitor Permit
vuser	Visitors User
vpermit	Visitor Permits