NC STATE UNIVERSITY

Wolf Parking Management System

CSC 540 Database Management Systems - Project 1

Fall Semester 2023

PROJECT TEAM

Boscosylvester Chittilapilly	bchitti
Janani Pradeep	jpradee
Aditi Vakeel	avakeel
Nahed Abu Zaid	naabuzai

Assumptions:

1. Drivers will not have access to this database. They must go to the parking office for any parking-related concerns or requests.
2. Driver goes to the billing staff to raise an appeal or make a payment for a citation.
3. To fetch any data related to the driver other than name, UID external databases are to be used.
4. Admin gets to know (this knowledge is not concerned with this DB) about the availability status of spaces (lots, Zones, types) and updates data accordingly.
5. The only payments recorded in this database are the ones made for citations, and no other fee is applicable for getting a permit.
6. Billing staff will update payment status based on payments and the appeal table.
7. A new entity called 'Payments' is maintained solely by the billing staff to store information about the citation payment information.
8. Billing Staff's decisions/actions on appeal approvals are based on external knowledge.
9. The appeal table would be used to store information about the appeal description and appeal status. It takes values. "in progress," 'approved," and "rejected."
10. Each citation can be challenged by at most one appeal and can have at most one payment.
11. To deal with deletion anomalies and reduce data redundancy, we maintain separate entities for 'SpaceType,' 'PermitType,' and 'CitationCategory'.

1. Problem Statement

We will design a comprehensive Wolf Parking Management System to effectively manage parking lots and user activities on campus. This database functions as a central store for essential data, such as driver profiles, parking lot information, permit data, car records, and citation data, to handle a wide spectrum of parking-related activities. A robust database is essential to assure data quality, consistency, and accessibility given the varied user base, which includes administrators monitoring parking services, security personnel enforcing regulations, and reporting staff producing insights.

By utilizing a well-structured database system, the university can efficiently track and manage parking-related activities, enforce rules, and produce insightful reports, ultimately improving the overall parking experience for students, staff, and visitors while maintaining data accuracy and security. To conclude, the variety and real-life data in the Wolf parking system encourage us to use a solid database in our project.

2. Intended Classes of Users:

- **Administrator Staff (Admins):** Manage parking lots, zones, and spaces; assign a parking permit to a single combination of driver and car; change the car license number on the permit; update availability of space; update driver and vehicle information and check if a car has a valid permit in a specific lot.

- **Reporting Staff:** Maintain and generate various parking-related reports as per requirement.

- **Billing Staff:** Handle the financial transactions associated with citations, update payment status for a citation after the payment is made, and update payment status based on appeal status. Also, raise and update the appeals of the drivers.

- **Parking Enforcement Personnel:** Check if a car has a valid permit in a specific lot and create, update, or delete citations based on appropriate parking violations.

3. Five main Entities

1. **Driver Information**: Name, Status, ID, PhoneNumber.
2. **Parking Lot Information**: ParkingLotID, Name, Address
3. **Space Information**: SpaceNumber, AvailabilityStatus, ZoneID, ParkingLotID
4. **Permit Information**: PermitID, StartDate, ExpirationDate, ExpirationTime, PermitType, ZoneID, SpaceType, DriverID, CarLicenseNumber
5. **Citation Information**: CitationNumber, CitationDate, CitationTime, PaymentStatus, CitationCategory, Fee, AppealDescription, AppealStatus, CarLicenseNumber, CarModel, CarColor

4. Tasks and Operations- Realistic Situations:

- **Situation 1:** Add a new driver and vehicle

A new student, Cathy, joins the university and visits the parking office to apply for a permit for her vehicle. The parking office employee will add Cathy's details, uniquely identified by her University ID, to the database and register her vehicle information on the system.

- **Situation 2:** Register Citation

When a student named Bob parks his vehicle in the wrong spot, a parking enforcement officer will take note of the violation and generate a citation for the vehicle. They moved his car to a designated area for such violations. Bob then goes to the parking office to retrieve his car. At the office, the billing staff provides Bob with the citation form and collects the fines. Once Bob pays the penalty, the citation is cleared, and he gets his car back.

5. Application Program Interfaces:

Operation 1: Information Processing

- addDriver(UID_Phone, Name, Status)
  - return confirmation
- updateDriver(UID_Phone, Name, Status)
  - return confirmation
    * if any parameter has NULL value, then it would not be updated
- deleteDriver(UID_Phone)
  - return confirmation
- addParkingLot(ParkingLotID, Name, Address)
  - return confirmation
- updateParkingLot(ParkingLotID, Name, Address)
  - return confirmation
    * if any parameter has NULL value, then it would not be updated
- deleteParkingLot(ParkingLotID)
  - return confirmation
- addZone(ParkingLotID, ZoneID)
  - return confirmation
- deleteZone(ZoneID)
  - return confirmation
- addSpace(ParkingLotID, ZoneID, SpaceNumber, AvailabilityStatus, SpaceType)
  - return confirmation
- updateSpace(ParkingLotID, ZoneID, SpaceNumber, AvailabilityStatus, SpaceType)
  - return confirmation
    * if any parameter among ParkingLotID, ZoneID, or SpaceNumber is NULL, then no update will take place
    * if any remaining parameter has NULL value, then it would not be updated
- deleteSpace(ParkingLotID, ZoneID, SpaceNumber)
  - return confirmation
- updateSpaceTypeForSpace(ParkingLotID, ZoneID, SpaceNumber, SpaceType)
  - return confirmation
    * if any parameter among ParkingLotID, ZoneID, or SpaceNumber is NULL, then no update will take place
- addSpaceType(SpaceType)

    - return TypeID
- updateSpaceType(SpaceTypeID, NewSpaceType)
    - return confirmation

      * if any parameter has NULL value, then it would not be updated

Operation 2: Maintaining Permits and Vehicle Information

- addVehicle(CarLicenseNumber, Model, Year, Manufacturer, Color)
    - return confirmation
- updateVehicle(CarLicenseNumber, Model, Year, Manufacturer, Color)
    - return confirmation

      * if any parameter has NULL value, then it would not be updated
- deleteVehicle(CarLicenseNumber)
    - return confirmation
- createPermit(PermitID, CarLicenseNumber, UID_Phone, ParkingLotID, ZoneID, PermitType, SpaceType, StartDate, ExpirationDate, ExpirationTime)
    - return confirmation
- updatePermit(PermitID, PermitType, ParkingLotID, ZoneID, SpaceType, StartDate, ExpirationDate, ExpirationTime)
    - return confirmation

      * if any parameter has NULL value, then it would not be updated
- deletePermit(PermitID)
    - return confirmation
- addPermitType(PermitType)
    - return PermitTypeID
- updatePermitType(PermitTypeID,NewPermitType)
    - return confirmation

      * if any parameter has NULL value, then it would not be updated
- updateVehicleInPermit(PermitID, NewCarLicenseNumber)
    - return confirmation

      * if any parameter has NULL value, then it would not be updated
- updateVehicleForDriverInAllPermits(UID_Phone, NewCarLicenseNumber)
    - return confirmation

      * if any parameter has NULL value, then it would not be updated

Operation 3: Generating and maintaining citations

- createCitation(CitationCategory, CitationDate, CitationTime, PaymentStatus)
  - return CitationNumber
- updateCitation(CitationNumber, CitationCategory, CitationDate, CitationTime, PaymentStatus)
  - return confirmation

    * if any parameter has NULL value, then it would not be updated
- deleteCitation(CitationNumber)
  - return confirmation
- payForCitation(CitationNumber, Amount, PaymentDate, PaymentTime)
  - return paymentID
- deleteCitationPayment(PaymentID)
  - return confirmation
- raiseAppealForCitation(CitationNumber, AppealDescription, AppealStatus)
  - return AppealID
- deleteAppeal(AppealID)
  - return confirmation
- updateAppealStatus(AppealID, NewAppealStatus)
  - return confirmation
- updateCitationCategoryForCitation(CitationNumber, CitationCategory)
  - return confirmation
- addCitationCategory(CitationCategory, Fee)
  - return CitationCategoryID
- updateCitationCategory(CitationCategoryID, CitationCategory, Fee)
  - return confirmation

    * if any parameter has NULL value, then it would not be updated
- updatePaymentStatus(PaymentID, PaymentStatus)
  - return confirmation

Operation 4: Reports

- generateCitationReport(ParkingLotID, Time_Range)
  - return count of citations for all zones in a given parking lot based on the time range

    * Time_Range would be either monthly or annually
- generateZoneReport()
  - return all zones along with their corresponding parking lots as a tuple

- generateActiveViolationCount()
    - return count of vehicles that have unpaid citations
- generateEmployeeReportForZone(ZoneID)
    - return count of employees with permit for a given zone
- generatePermitReportForDriver(UID_Phone)
    - return permit information for a given student/employee’s ID or guest user’s phone number
- generateAvailableSpaceCountForLot(ParkingLotID, SpaceType)
    - return count of available spaces in a given parking lot for a given space type
      * Valid space types are electric, handicap, compact car, and regular

6. Description of Views:

- **Administrators**: can access information about drivers, vehicles, parking lots, zones, spaces, and drivers' permits. The admins are not required to access the payments, citations, and appeals information.

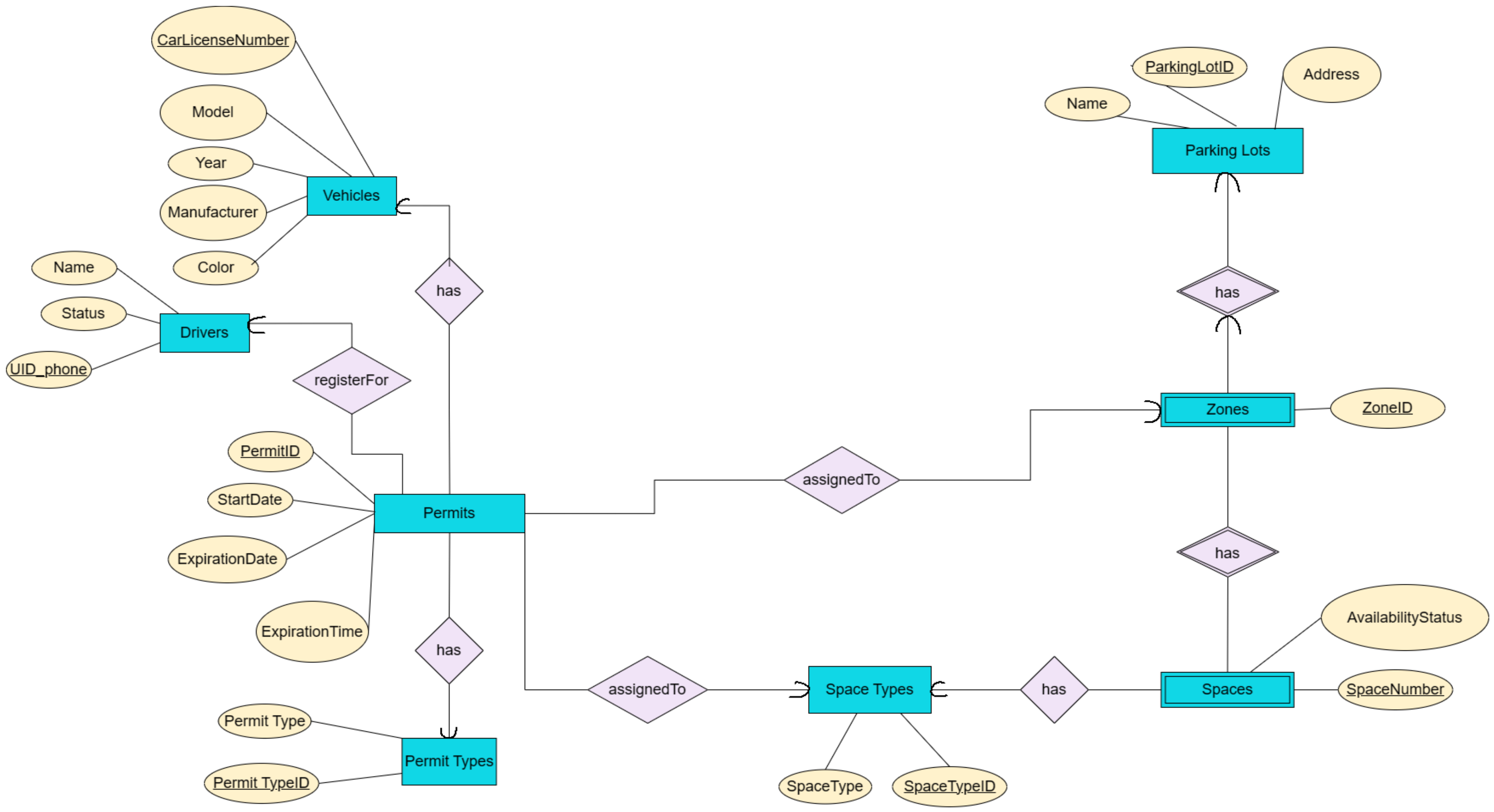
- **Reporting Staff:** can view drivers, vehicles, permits of users, and citations generated on vehicles, lots, zones, and spaces.

- **Billing Staff**: will have access to views of payments, citations, and appeals information. It is not necessary for them to access view that concerns drivers' personal information, reports, and vehicle information that are handled by another class.

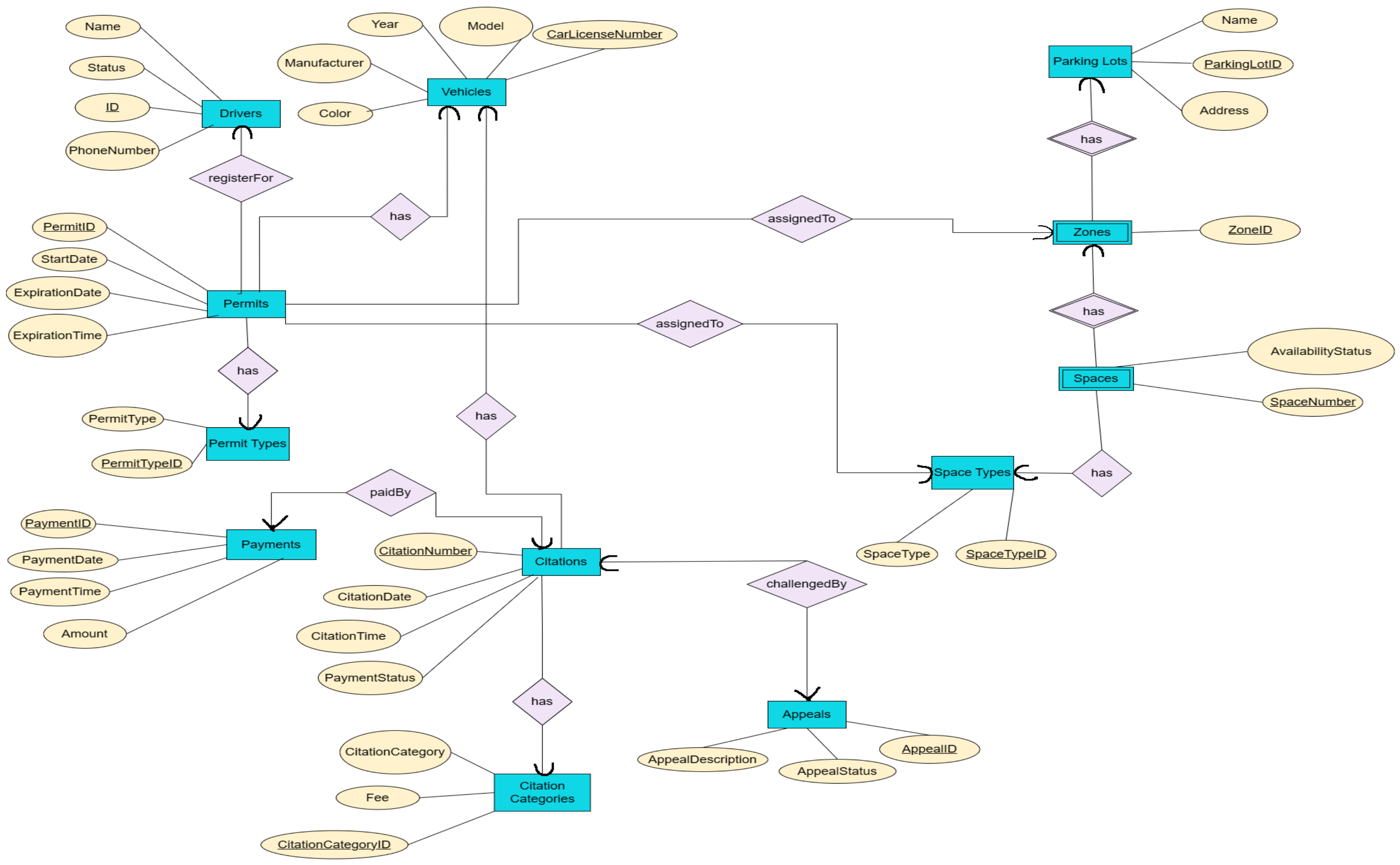
- **Parking Enforcement Personnel:** can access vehicle, permit, and citation information. It is not required for them to view data that concerns payments, parking lots, zone, and spaces information.

7. Local E/R Diagrams:

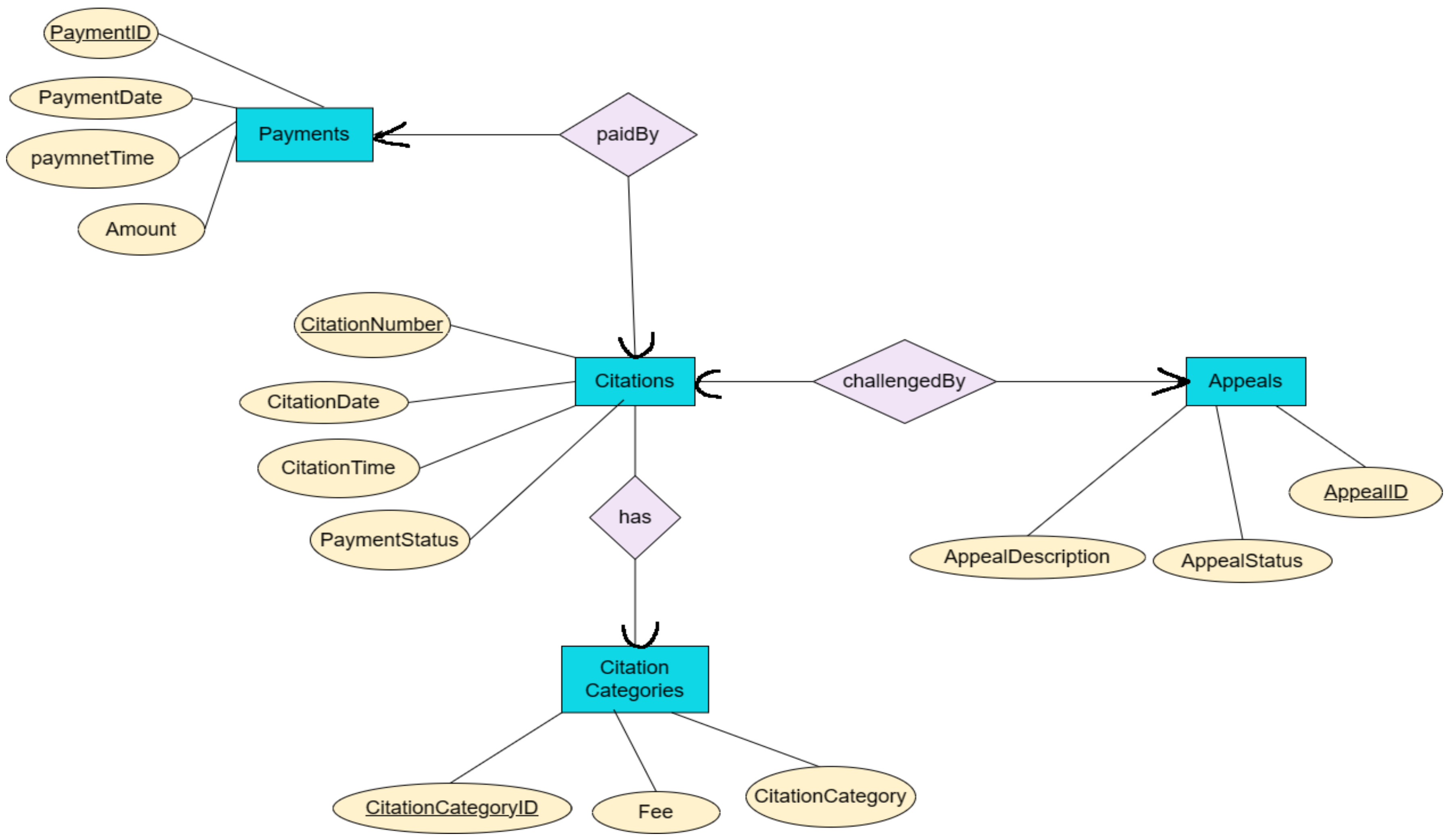
7.1 Admin’s View:



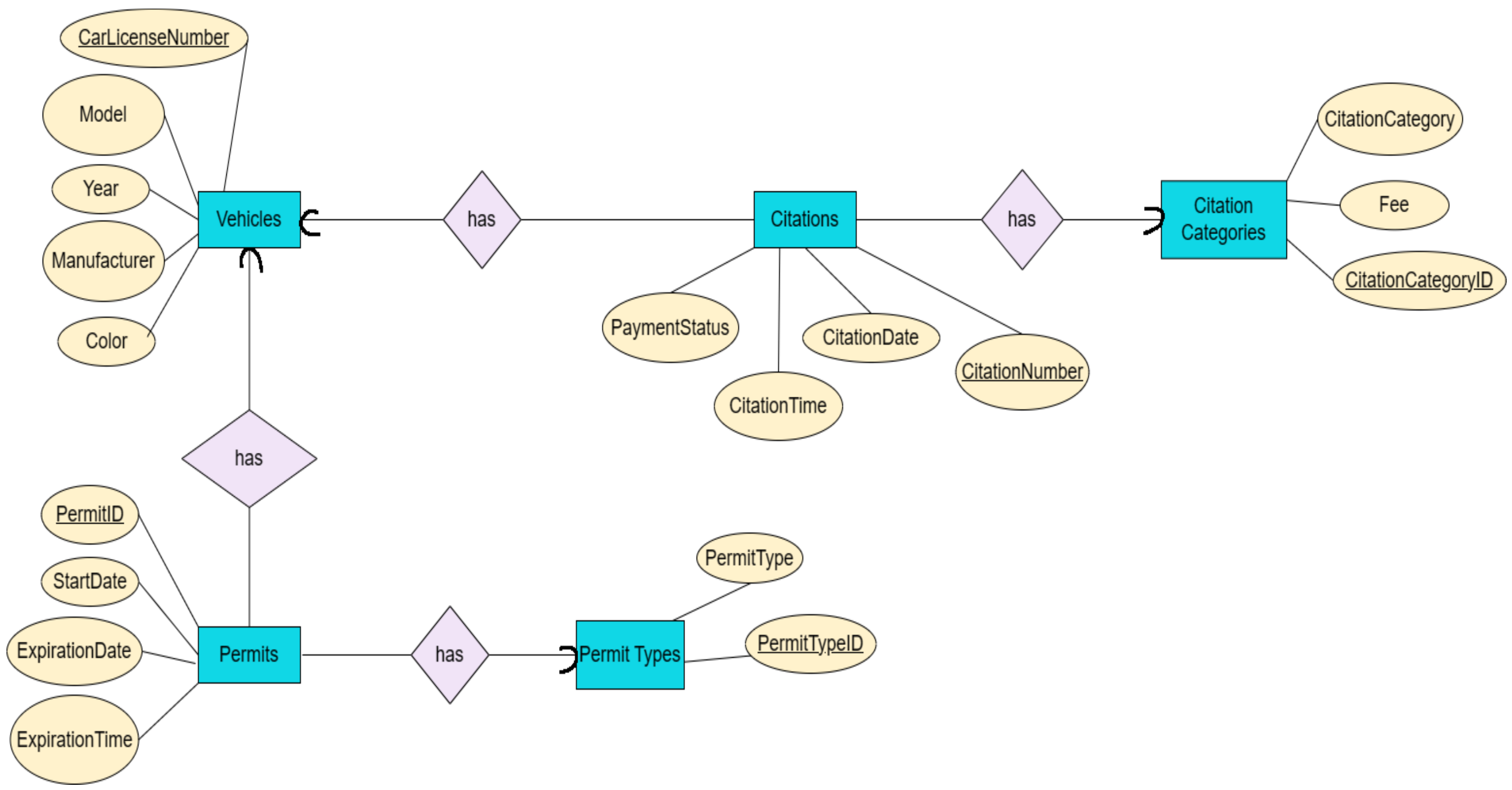
7.2 Reporting Staff View:



7.3 Billing Staff View:



7.4 Parking Enforcement Personnel View:



8. Description of Local E/R diagrams:

1. A driver has a unique ID in the form of UID_Phone, which stores either a university ID for employees and students or a phone number for visitors.
2. Vehicles have a unique ID in the form of a car license number. Thus, no vehicle has the same license number.
3. Every parking lot can have multiple zones assigned to it, and every zone can have multiple spaces, each space with one of the four defined space types.
4. Zones is a weak entity set that uses the ZoneID and ParkingLotID to identify itself uniquely.
5. Similar to #4, Spaces is a weak entity set that uses the SpaceNumber, ZoneID, and ParkingLotID to identify itself uniquely.
6. A separate table of space types is maintained to preserve the information of the initially defined space types (“electric,” “handicap,” “compact car,” “regular”). Each space is linked with a distinct space type.
7. Each permit corresponds to a single driver and a single vehicle. This permit is associated with a space type and a zone within a parking lot. However, a driver can have multiple permits based on their status. Similarly, a vehicle can also be associated with multiple permits.
8. One space type can be associated with multiple permits, but one permit has only one space type.
9. Similar to #8, one zone can accommodate multiple permits, but each permit has only one zone assigned to it.
10. A separate table of permit types is maintained to preserve the information of the initially defined permit types (“residential,” “commuter,” “peak hours,” “special event,” and “Park & Ride”). Each permit is related to a specific permit type.
11. Citations are generated on vehicles given a violation and are uniquely identified by CitationID. Every citation has a citation category with a corresponding fee.
12. Every citation is against exactly one vehicle, but a vehicle can have many citations.
13. A separate table of citation categories is maintained to preserve the information of the initially defined citation categories (“Invalid Permit,” “Expired Permit,” “No Permit”).
14. Every citation has at most one payment, and every payment is linked to exactly one citation.
15. An appeal can be raised against a citation for a vehicle and can be uniquely identified by AppealID.

16. Each appeal is raised against a specific citation. However, each citation may have at most one appeal.
17. When an appeal is raised, it is assigned a status of “in progress”. Billing staff would then update the status to “approved” or “rejected”.
18. Whenever a payment is done towards a citation or an appeal for that citation is approved, the payment status is changed to “completed”. Payment status is set to “due” by default.

9. Local Relational Schemas:

9.1 Admin’s View:

Vehicles (CarLicenceNumber, Model, Year, Manufacturer, Color)

Drivers (Name, Status, UID_Phone)

ParkingLots (Name, ParkingLotID, Address)

Zones (ZoneID, ParkingLotID)

Spaces (SpaceNumber, ZoneID, ParkingLotID, AvailabilityStatus, SpaceTypeID)

SpaceTypes(SpaceType, SpaceTypeID)

Permits (PermitID, StartDate, ExpirationDate, ExpirationTime,
PermitTypeID,ZoneID, SpaceTypeID, UID_Phone, CarLicenseNumber)

PermitTypes (PermitType, PermitTypeID)

9.2 . Reporting Staff View:

Vehicles (CarLicenceNumber, Model, Year, Manufacturer, Color)

Drivers (Name, Status, UID_Phone)

ParkingLots (Name, ParkingLotID, Address)

Zones (ZoneID, ParkingLotID)

Spaces (SpaceNumber, ZoneID, ParkingLotID, AvailabilityStatus, SpaceTypeID)

SpaceTypes(SpaceType, SpaceTypeID)

Permits (PermitID, StartDate, ExpirationDate, ExpirationTime, PermitTypeID,ZoneID, SpaceTypeID, UID_Phone, CarLicenseNumber)

PermitTypes (PermitType, PermitTypeID)

Citations (CitationNumber, CitationDate, CitationTime, PaymentStatus, PaymentID, CitationCategoryID)

CitationCategories (CitationCategoryID, CitationCategory, Fee)

Payments (PaymentID, PaymentDate, PaymentTime, Amount, CitationNumber)

9.3 Billing Staff View:

Citations (CitationNumber, CitationDate, CitationTime, PaymentStatus, PaymentID, CitationCategoryID, AppealID)

CitationCategories (CitationCategoryID, CitationCategory, Fee)

Payments (PaymentID, PaymentDate, PaymentTime, Amount, CitationNumber)

Appeals (AppealID, AppealDescription, AppealStatus, CitationNumber)

9.4 Parking Enforcement Personnel View:

Vehicles (CarLicenceNumber, Model, Year, Manufacturer, Color)

Permits (PermitID, StartDate, ExpirationDate, ExpirationTime, PermitTypeID, CarLicenseNumber)

PermitTypes (PermitType, PermitTypeID)

Citations (CitationNumber, CitationDate, CitationTime, PaymentStatus,CitationCategoryID)

CitationCategories (CitationCategoryID, CitationCategory, Fee)

10. Local Schema Documentation:

Entity sets to Relations:

- All the entity sets in our diagrams are converted into relations based on the E/R viewpoints of the classes of users for this database.
- Thus, the entity sets Drivers, Vehicles, ParkingLots, SpacesTypes, Permits, PermitTypes, Citations, CitationCategories, Payments, and Appeals are converted into relations.

Weak Entity-sets to relations:

- The entity set 'Zones' is marked as a weak entity set as the same ZoneID can be present in multiple parking lots. Hence, 'Zones' depend on the strong entity set 'ParkingLots.' Thus, a zone is uniquely defined by a combination of ParkingLotID and ZoneID as a key in the relations 'Zones. 'The inclusion of ParkingLotID in 'Zones' represents the 'has' relationship between these entities, and this weak relationship is not required anymore in the relational schema.

- The entity set 'Spaces' is marked as a weak entity set as the same SpaceNumber can be present in multiple zones and parking lots. Hence, 'Spaces' depend on the entity sets 'ParkingLots' and 'Zones.' Thus, a space is uniquely defined by a combination of ParkingLotID, ZoneID, and SpaceNumber as a key in the relations 'Spaces.' The inclusion of ParkingLotID and ZoneID in 'Spaces' represents the 'has' relationship between these entities, and this weak relationship is not required anymore in the relational schema.

Combining Many-to-One relationships to relations:

In a many-to-one relationship, the primary key on the one side will be placed as the foreign key on the many sides of the relationship. In doing so, the one side of the relationship is represented by the many sides. Thus, there is no necessity of creating an additional relation for such a many-to-one relationship.

- The relationship 'has' from 'Spaces' to 'SpaceTypes' is a many-to-one relationship. Each space has exactly one space type assigned to it and each space type can be associated with multiple spaces. So primary key

‘SpaceTypeID’ from the ‘SpaceType’ acts as a foreign key in the ‘Spaces’ relation.

- The 'has' relationship between 'Permit' and 'Vehicles' is a many-to-one relationship. Each permit is authorized for exactly one vehicle, and each vehicle can be granted multiple spaces based on different scenarios. As a result, the 'CarLicenseNumber' primary key in the 'Vehicle' table serves as a foreign key in the 'Permit' table.

- For entities 'Permits' and 'Drviers,' the 'registeredFor' relationship signifies a many-to-one association. Each individual permit is assigned for one specific driver, while a single driver can have multiple permits based on their status. Therefore, the 'UID_Phone' primary key in the 'Drivers' relation functions as a foreign key within the 'Permits' relation.

- Describing the relationship between 'Permits' and 'Zones,' we observe ‘assignedTo’ as a many-to-one relationship. Each permit is authorized with a singular zone in a parking lot, and each zone in a parking lot would be assigned to multiple permits. Consequently, the 'ParkingLotID' and ‘ZoneID’ primary key in the 'Zones' relation is employed as a foreign key in the 'Permits' relation.

- The relationship denoted as 'assignedTo' between 'Permits' and 'SpaceTypes' is established as a many-to-one relationship. In this arrangement, each permit is exclusively associated with a single space type, while each space type can be linked with numerous permits. As a result, the 'SpaceTypeID' primary key within the 'SpaceType' relation is utilized as a foreign key within the 'Permits' table.

- The relationship ‘assignedTo’ from ‘Permits’ to ‘PermitTypes’ is a many-to-one relationship. So, the primary key ‘PermitTypeID’ from the ‘PermitTypes’ acts as a foreign key in the ‘Permits’ relation.

- The 'has' relationship between 'Citations' and 'CitationCategories' is a many-to-one relationship. Each citation is raised for violation of exactly one citation category, and each citation category can be linked to multiple citations. As a result, the 'CitationCategoryID' primary key in the 'CitationCategories' relation serves as a foreign key in the 'Citations' relation.

- For entities 'Citations' and 'Vehicles,' the 'has' relationship signifies a many-to-one association. Each individual citation is raised for one specific vehicle, while a single vehicle can have multiple citations (including active and inactive). Therefore, the 'CarLicenseNumber' primary key in the 'Vehicles' relation functions as a foreign key within the 'Citations' relation.

One-to-One Relationships to Relations:

In one-to-one relationships, we can move the primary key from any one side. With this, one relationship is represented by the other. Thus, there is no necessity of creating an additional relation for any one-to-one relationship.

- ‘Citations’ and ‘Appeals’ are in a one-to-one relationship. Each appeal is linked to exactly one citation, and each citation may have at most one appeal. As a result, if we use AppealID from ‘Appeals’ as a foreign key in ‘Citations’ and if an appeal has not been raised for that citation, then ‘Citations’ would have a NULL value in the foreign key. To avoid this scenario, we will use CitationNumber from ‘Citations’ as the foreign key in ‘Appeals.’
- 'Citations' and 'Payments' maintain a one-to-one relationship. Every citation can have, at most, one associated payment, and conversely, each payment can be linked to precisely one citation. Therefore, if we were to employ 'PaymentID' from 'Payments' as a foreign key in 'Citations,' and if payment were absent for a particular citation, 'Citations' would contain a NULL value in the foreign key. To circumvent this situation, we use 'CitationNumber' from 'Citations' as the foreign key in the 'Payments' table.

Many-to-Many Relationships to Relations:

- We do not have any many-to-many relationships in our schema.