Wolf Parking Management System

For Wolf Parking

CSC 540 Database Management Systems
Project Report 2
Team CC

Daniel Buchanan(dsbuchan), Ophelia Sin(oysin), Aadil Tajani(atajani), Manali Teke(mteke)
October 20th, 2023

Assumptions:

1. Phone Number is unique for all drivers.
2. Student and Employee drivers will have a Phone Number.
3. Space Number is unique within a Parking Lot and a Zone is unique within a Parking Lot.
4. Parking Lot name is unique.
5. Parking manager is one of the administrators.
6. Any permit with a particular space type can park in a regular space. However, to access non regular space types, drivers have to hold a permit for that specific space type. For example, only a driver with a handicap parking permit can park in a handicap space.
7. A parking lot can contain multiple zones. A driver can park in the specified parking lot and zone in their permits.
8. A parking lot can have multiple types of spaces.
9. Each permit can have multiple lots and zones.
10. Administrators manage the billing processes.
11. On creation of a citation, payment status is set to FALSE(unpaid) by default and is available to the driver with the citation information and any status change reflects to all(driver, security, and administrator).
12. Drivers can get a new permit upon exhausting their respective limits only after getting a permit of the same type deleted.
13. Security can look up if a vehicle has a valid permit to note a violation and generate a citation.
14. The citation number is unique for every citation.
15. Each permit can only have one space type.
16. Car License Number, Citation Number and Permit ID are all integer type values.
17. paymentStatus is a boolean value where 1 is paid and 0 is unpaid.
18. availability_status is a boolean value where 1 is available and 0 is unavailable.
19. approval_status indicates the status of appealed citations. By default, it has a value set to Pending.

1. Global Relational Database Schema:

**Using NULLs approach for ER Diagrams, Student, Employee and Visitor entities have been merged into Driver to eliminate redundancy and maintain constraints like uniqueness of univID.

Driver(<u>phone</u>, name, status, univID)

phone* → *phone, name, status, univID holds because each phone number is unique(key) and it identifies a driver that has a name, status, and a univID(if status is employee/student). If we were to try and take any combination of other attributes, it would severely limit the possibilities our database could have. For example, many people can have the same name or same status. Because the left hand side is super key, the functional dependency is in BCNF and thereby in 3NF. Even though univID is unique for all students and employees, it can't be used to determine all the records as the field will be NULL for drivers with visitor status.

ParkingLot(<u>lotName</u>, address)

lotName → lotName, address is in 3NF and BCNF because a two attribute relation is always in BCNF and 3NF.

Zone(<u>zoneID</u>, <u>lotName</u>)

zoneID, lotName → zoneID, lotName is in 3NF and BCNF because a two attribute relation is always in BCNF and 3NF.

Space(<u>spaceNumber</u>, <u>lotName,</u> <u>zoneID</u>, spaceType, availabilityStatus)

spaceNumber, lotName, zoneID → spaceNumber, lotName, zoneID, spaceType, availabilityStatus holds because a space number, zoneID and lot name are used together to uniquely identify a space in a zone which is in a lot. And spaceNumber identifies spaceType and availabilityStatus as well. Only one car can occupy a given space in a given zone in a given lot. Since the left hand side is a superkey(and a key), it is in BCNF and therefore in 3NF.

Citation(<u>citationNumber</u>, citationDate, citationTime, category, fee, paymentStatus)

citationNumber → citationNumber, citationDate, citationTime, category, fee, paymentStatus holds each citation has a unique citationNumber that determines all of its other attributes. Since LHS is super key it is in 3NF and BCNF.

Vehicle(<u>carLicenseNumber</u>, model, year, color, manufacturer)

carLicenseNumber → carLicenseNumber, model, year, color, manufacturer holds because each car has a unique carLicenseNumber which derives all of its respective attributes. Since LHS is a super key it is in 3NF and BCNF.

Permit(<u>permitID</u>, spaceType, startDate, expirationDate, expirationTime, permitType)

permitID → spaceType, startDate, expirationDate, expirationTime, permitType holds because each permit will have a unique permitID, which can be used to determine all the other

attributes of the permit, therefore acting as the key and the superkey. Since the left hand side of the FD is a superkey, the relation is in BCNF and therefore in 3NF.

IsAssigned(<u>phone</u>, <u>permitID</u>, <u>carLicenseNumber</u>)

phone, permitID, carLicenseNumber → phone, permitID, carLicenseNumber is in 3NF and BCNF because all three attributes together are a key and hence a superkey.

GivenTo(<u>citationNumber</u>, <u>carLicenseNumber</u>)

citationNumber, carLicenseNumber→ citationNumber, carLicenseNumber holds because the citationNumber and carLicenseNumber make up the key of the relation and are also a superkey, making this relation as BCNF and therefore in 3NF.

Shows(<u>citationNumber</u>, <u>lotName</u>)

citationNumber, lotName → citationNumber, lotName holds because the citationNumber and lotName make up the key of the relation. Because they make up the key of the relation and are also a superkey,this relation is in BCNF and therefore in 3NF.

Appeals(<u>phone,</u> <u>citationNumber,</u> appealStatus)

citationNumber, phone → citationNumber, phone, appealStatus holds because the citationNumber and phone make up the superkey of the relation and it identifies appeal status as well . Because they make up the superkey of the relation, this relation is in BCNF and therefore in 3NF.

HasLot(<u>permitId</u>, <u>lotName</u>)

permitId, lotName → permitId, lotName holds because the permitId and lotName make up the key of the relation and are also a superkey, making this relation as BCNF and therefore in 3NF.

HasZone(<u>permitID</u>, <u>zoneId</u>, <u>lotName</u>)

permitId, zoneID, lotName → permitID, zoneID, lotName holds because the permitID, zoneId, and lotName make up the key of the relation and are also a superkey, making this relation as BCNF and therefore in 3NF.

2. Design for Global Schema:

<u>Design decision for global schema:</u>

The entity sets in our diagram were made into relations, with their respective attributes for Driver, Permit, Citation, Vehicle, ParkingLot, Zone and Space. The Entity subsets Employee, Student and Visitor were merged into Entity Driver using Nulls approach to eliminate redundancy and save space which also helps maintain uniqueness of univ_id across employees and students.

We made relations with just primary keys, as this reduces redundancy and decreases the overhead that many tables cause. It also makes queries quicker. Other weak relationships have been turned into relations via their entities in our schema. Their attributes in the schema are their own keys and that of the entities they depend upon.

Driver(<u>phone</u>, name, status, univID)

phone is the primary key
name and status cannot be null
univID is unique and can be null

phone is the primary key as it is used to identify all drivers as they all are assumed to have a unique phone number. Name and Status are not unique as there can be people with same names or same status, but they can't be null as they are important information about drivers and everyone has them. UnivID is unique, as each student and employee should have a unique university id. However, univID can also be null as drivers with a status of visitor 'V' will not have a univID.

Zone(<u>zoneId</u>, <u>lotName</u>)

zoneID and lotName together form the primary key

Zone has the only attribute zoneId as a part of primary key along with the reference lotName completing the primary key because of the weak relation.

Space(<u>spaceNumber</u>, <u>lotName</u>, <u>zoneId</u>, spaceType, availabilityStatus)

spaceNumber, lotName, and zoneID together form the primary key
spaceType and availabilityStatus cannot be null.

Space has attributes spaceNumber, spaceType and availabilityStatus that are required in the system to maintain data on spaces and cannot be null. There are also lotName, and zoneId referenced as addition to the spaceNumber together forming the key, because of the weak relation of Space on Zone and that on Parking Lot.

Citation(<u>citationNumber</u>, citationDate, citationTime, category, fee, paymentStatus)

citationNumber is the primary key
citationDate, citationTime, category, fee and paymentStatus cannot be null .

citationNumber uniquely identifies each citation and therefore is a primary key. Other attributes cannot be null as they all are needed to generate a citation.

Vehicle(<u>carLicenseNumber</u>, model, year, color, manufacturer)

carLicenseNumber is the primary key
model, year, color, and manufacturer cannot be null.

carLicenseNumber uniquely identifies each vehicle and therefore is a primary key. Other attributes cannot be null as they all are available for every vehicle, each needed to maintain the vehicle data.

Permit(<u>permitID</u>, spaceType, startDate, expirationDate, expirationTime, permitType)

permitID is the primary key
spaceType, startDate, expirationDate, expirationTime, and permitType cannot be null.

permitID uniquely identifies a permit so it is a primary key. Other attributes are required to check the validity of the parked vehicle and to generate a citation so cannot be null.

ParkingLot(<u>lotName</u>, address)

lotName is the primary key

address cannot be null

lotName uniquely identifies parking lot so it is primary key. Address cannot be null as there cannot be a lot without a physical address.

IsAssigned(<u>phone</u>, <u>permitID</u>, <u>carLicenseNumber</u>)

phone, permitID, carLicenseNumber referenced together are primary keys.

Shows(<u>citationNumber</u>, <u>lotName</u>)

citaitonNumber and lotName referenced together are primary keys.

Appeals(<u>citationNumber</u>, <u>phone</u>, appealStatus)

citaitonNumber and phone referenced together are primary keys.

appealStatus cannot be null as each appeal generated can either be approved, denied or pending.

HasLot(<u>permitId</u>, <u>lotName</u>)

permitId and lotName referenced together are primary keys

HasZone(<u>permitID</u>, <u>zoneId</u>, <u>lotName</u>)

permitId, zoneID, and lotName referenced together are primary keys

GivenTo(<u>citationNumber</u>, <u>carLicenseNumber</u>)

citationNumber and carLicenseNumber referenced together form the primary key.

3. Base Relations:

```
CREATE TABLE Driver (
      phone       VARCHAR(10),
      name        VARCHAR(128)     NOT NULL,
      status      VARCHAR(1)       NOT NULL,
      univ_id     VARCHAR(9),
      UNIQUE(univ_id),
      PRIMARY KEY(phone)
);
```

```
CREATE TABLE ParkingLot (
      lot_name VARCHAR(128),
      address VARCHAR(128) NOT NULL,
      PRIMARY KEY(lot_name)
);

CREATE TABLE Zone (
      zone_id VARCHAR(2),
      lot_name VARCHAR(100),
      PRIMARY KEY(zone_id, lot_name),
      FOREIGN KEY(lot_name)
            REFERENCES ParkingLot(lot_name)ON UPDATE CASCADE
);

CREATE TABLE Space (
      space_number INT,
      lot_name VARCHAR(100),
      zone_id VARCHAR(2),
      space_type VARCHAR(11) NOT NULL,
      availability_status BOOLEAN NOT NULL,
      PRIMARY KEY(space_number, lot_name, zone_id),
      FOREIGN KEY(zone_id,lot_name)
            REFERENCES Zone(zone_id, lot_name)
);

CREATE TABLE Vehicle (
      car_license_number VARCHAR(10),
      model VARCHAR (100) NOT NULL,
      year INT NOT NULL,
      color VARCHAR (100) NOT NULL,
      manufacturer VARCHAR(100) NOT NULL,
      PRIMARY KEY(car_license_number)
);


CREATE TABLE Citation (
    citation_number INT,
    citation_date DATE NOT NULL,
    citation_time TIME NOT NULL,
    category VARCHAR(128) NOT NULL,
    fee FLOAT(9,2) NOT NULL,
    payment_status BOOLEAN NOT NULL,
    PRIMARY KEY (citation_number)
);
```

```
CREATE TABLE Permit (
      permit_id INT,
      space_type VARCHAR(11) NOT NULL,
      start_date DATE NOT NULL,
      expiration_date DATE NOT NULL,
      expiration_time TIME NOT NULL,
      permit_type VARCHAR(13) NOT NULL,
      PRIMARY KEY(permit_id)
);

CREATE TABLE IsAssigned (
      phone VARCHAR(10),
      permit_id INT,
      car_license_number VARCHAR(10),
      PRIMARY KEY (phone, permit_id, car_license_number),
      FOREIGN KEY (phone) REFERENCES Driver(phone) ON UPDATE CASCADE,
      FOREIGN KEY (permit_id) REFERENCES Permit(permit_id) ON UPDATE
      CASCADE,
      FOREIGN KEY (car_license_number) REFERENCES
      Vehicle(car_license_number) ON UPDATE CASCADE );


CREATE TABLE Shows (
    citation_number INT,
    lot_name VARCHAR(128),
    PRIMARY KEY (citation_number, lot_name),
    FOREIGN KEY (citation_number) REFERENCES Citation(citation_number) ON
UPDATE CASCADE,
    FOREIGN KEY (lot_name) REFERENCES ParkingLot(lot_name) ON UPDATE
CASCADE
);

CREATE TABLE Appeals (
      phone VARCHAR(10),
      citation_number INT,
      appeal_status VARCHAR (9) NOT NULL,
      PRIMARY KEY (phone,citation_number),
      FOREIGN KEY (citation_number) REFERENCES Citation(citation_number)
ON UPDATE CASCADE,
FOREIGN KEY (phone) REFERENCES Driver(phone) ON UPDATE CASCADE );


CREATE TABLE HasLot (
      permit_id INT,
      lot_name VARCHAR(128),
```

```
      PRIMARY KEY(permit_id,lot_name),
      FOREIGN KEY(permit_id)
            REFERENCES Permit(permit_id)
            ON UPDATE CASCADE,
      FOREIGN KEY(lot_name)
            REFERENCES ParkingLot(lot_name)
            ON UPDATE CASCADE
);

CREATE TABLE HasZone (
    permit_id INT,
    zone_id VARCHAR(2),
    lot_name VARCHAR(128),
    PRIMARY KEY (permit_id, zone_id, lot_name),
    FOREIGN KEY (permit_id) REFERENCES Permit(permit_id) ON UPDATE
CASCADE,
    FOREIGN KEY (zone_id) REFERENCES Zone(zone_id) ON UPDATE CASCADE,
    FOREIGN KEY (lot_name) REFERENCES ParkingLot(lot_name) ON UPDATE
CASCADE
);


CREATE TABLE GivenTo (
      citation_number INT,
      car_license_number VARCHAR(10),
      PRIMARY KEY(citation_number, car_license_number),
      FOREIGN KEY(citation_number)
            REFERENCES Citation(citation_number)
            ON UPDATE CASCADE
);


SELECT * FROM Driver;
+------------+-----------------+--------+----------+
| phone      | name            | status | univ_id  |
+------------+-----------------+--------+----------+
| 4567980055 | Donna Paulson   | S      | dpaulson |
| 5546199825 | Daniel Hardman  | S      | dhardman |
| 5555555555 | Jessica Pearson | E      | jpearson |
| 9463543210 | Louis Litt      | E      | llitt    |
| 9876001310 | Ava Hessington  | V      | NULL     |
| 9876543210 | Harvey Spectre  | E      | hspectre |
| 9934567890 | Rachel Zane     | V      | NULL     |
| 9988713386 | Gina Linetti    | S      | glentii  |
+------------+-----------------+--------+----------+
```

```
SELECT * FROM ParkingLot;

+------------+--------------------------------------+
| lot_name   | address                              |
+------------+--------------------------------------+
| Carmichael | Wellrec Center, Morrill Dr           |
| Dan Allen  | Pullen Dr, Main Campus               |
| Oval       | Oval Dining Hall, Centennial Campus  |
| Poulton    | Poulton Center, Partners Way         |
| Venture    | Venture I Deck, Varsity Drive        |
+------------+--------------------------------------+

SELECT * FROM Zone;
+---------+------------+
| zone_id | lot_name   |
+---------+------------+
| A       | Oval       |
| A       | Poulton    |
| AS      | Venture    |
| B       | Carmichael |
| B       | Venture    |
| BS      | Carmichael |
| V       | Poulton    |
+---------+------------+


SELECT * FROM Space;
+--------------+------------+---------+-------------+--------------------
-+
| space_number | lot_name   | zone_id | space_type  | availability_status
|
+--------------+------------+---------+-------------+--------------------
-+
|            1 | Carmichael | B       | electric    |                   1
|
|            1 | Carmichael | BS      | compact car |                   1
|
|            1 | Oval       | A       | regular     |                   1
|
|            1 | Poulton    | V       | regular     |                   1
|
|            2 | Oval       | A       | regular     |                   0
|
```

```
|             2 | Poulton    | V       | electric    |                    1
|
|             3 | Oval       | A       | handicap    |                    0
|
 +--------------+------------+---------+-------------+---------------------+
```

```
SELECT * FROM Vehicle;
+--------------------+---------------------+------+--------+-------------
-+
| car_license_number | model               | year | color  | manufacturer
|
+--------------------+---------------------+------+--------+-------------
-+
| ABC123             | Toyota Corolla      | 2022 | Red    | Toyota
|
| DEF456             | Ford Mustang        | 2023 | Yellow | Ford
|
| JKL321             | Chevrolet Silverado | 2022 | Black  | Chevrolet
|
| MNO654             | Nissan Altima       | 2020 | Silver | Nissan
|
| PQR987             | Hyundai Elantra     | 2022 | White  | Hyundai
|
| XYZ789             | Honda Civic         | 2021 | Blue   | Honda
|
+--------------------+---------------------+------+--------+-------------
-+
```

```
SELECT * FROM Citation;
+----------------+--------------+--------------+------------------------+------+---------------
+
| citation_number | citation_date | citation_time | category               | fee  | payment_status
|
+----------------+--------------+--------------+------------------------+------+---------------
+
|             101 | 2023-10-19    | 08:30:00      | Invalid Permit         |   25 |             1
|
|             102 | 2023-10-20    | 14:15:00      | Expired Permit         |   30 |             0
|
|             103 | 2023-10-21    | 10:45:00      | No Permit              |   40 |             1
|
|             104 | 2023-10-22    | 09:20:00      | Expired Permit         |   30 |             1
|
|             105 | 2023-10-23    | 16:55:00      | Handicap Invalid Permit | 12.5 |             0
|
```

```
|             106 | 2023-10-23    | 17:10:00      | No Permit               |   40 |              0
|
|             107 | 2023-10-24    | 09:23:00      | Invalid Permit          |   25 |              0
|
+-----------------+---------------+---------------+-------------------------+------+---------------
+
```

SELECT * FROM Permit;

```
+-----------+-------------+------------+-----------------+-----------------+---------------+
| permit_id | space_type  | start_date | expiration_date | expiration_time | permit_type   |
+-----------+-------------+------------+-----------------+-----------------+---------------+
|         1 | Electric    | 2023-10-19 | 2024-10-19      | 08:00:00        | Residential   |
|         2 | Handicap    | 2023-10-20 | 2023-10-21      | 12:00:00        | Special Event |
|         3 | Compact Car | 2023-10-22 | 2023-12-31      | 09:30:00        | Commuter      |
|         4 | Regular     | 2023-11-01 | 2024-10-31      | 08:30:00        | Park & Ride   |
|         5 | Electric    | 2023-11-05 | 2024-11-05      | 14:00:00        | Residential   |
|         6 | Handicap    | 2023-11-10 | 2023-11-11      | 10:30:00        | Special Event |
|         7 | Compact Car | 2023-11-15 | 2023-12-31      | 11:45:00        | Commuter      |
|         8 | Regular     | 2023-11-20 | 2024-11-19      | 12:00:00        | Peak Hours    |
+-----------+-------------+------------+-----------------+-----------------+---------------+
```

SELECT * FROM IsAssigned;

```
+------------+-----------+--------------------+
| phone      | permit_id | car_license_number |
+------------+-----------+--------------------+
| 1234567890 |         1 | ABC123             |
| 4567980055 |         6 | PQR987             |
| 5555555555 |         3 | DEF456             |
| 9463543210 |         5 | MNO654             |
| 9876543210 |         2 | XYZ789             |
| 9934567890 |         4 | JKL321             |
+------------+-----------+--------------------+
```

SELECT * FROM Shows;

```
+-----------------+------------+
| citation_number | lot_name   |
+-----------------+------------+
|             101 | Oval       |
|             102 | Venture    |
|             103 | Poulton    |
|             104 | Oval       |
|             105 | Carmichael |
+-----------------+------------+
```

```
SELECT * FROM Appeals;
```

```
+------------+-----------------+---------------+
| phone      | citation_number | appeal_status |
+------------+-----------------+---------------+
| 4567980055 |             107 | Denied        |
| 5546199825 |             105 | Pending       |
| 5546199825 |             106 | Pending       |
| 9876543210 |             102 | Approved      |
+------------+-----------------+---------------+
```

```
SELECT * FROM HasLot;
```

```
+-----------+------------+
| permit_id | lot_name   |
+-----------+------------+
|         1 | Poulton    |
|         2 | Oval       |
|         3 | Carmichael |
|         4 | Carmichael |
|         4 | Oval       |
|         5 | Poulton    |
|         6 | Oval       |
|         7 | Carmichael |
|         8 | Oval       |
+-----------+------------+
```

```
SELECT * FROM HasZone;
```

```
+-----------+---------+------------+
| permit_id | zone_id | lot_name   |
+-----------+---------+------------+
|         1 | V       | Poulton    |
|         2 | A       | Oval       |
|         3 | BS      | Carmichael |
|         4 | A       | Oval       |
|         5 | V       | Poulton    |
|         6 | A       | Oval       |
|         7 | BS      | Carmichael |
|         8 | A       | Oval       |
+-----------+---------+------------+
```

```
SELECT * FROM GivenTo;
+-----------------+--------------------+
| citation_number | car_license_number |
+-----------------+--------------------+
|             101 | ABC123             |
|             102 | XYZ789             |
|             103 | DEF456             |
|             104 | JKL321             |
|             105 | MNO654             |
+-----------------+--------------------+
```

4. SQL Queries:

4.1 Assumption Queries

Information Processing

- *Enter driver info*
  - ```
    SQL>INSERT INTO Driver VALUES('9988713386', 'Gina
    Linetti', 'S', 'glentii');
    Query OK, 1 row affected (0.0031 sec)
    ```
  - ```
    SQL>INSERT INTO Driver VALUES('9934567890', 'Rachel
    Zane', 'V',NULL);
    Query OK, 1 row affected (0.0024 sec)
    ```
- *Update driver info*
  - ```
    SQL> UPDATE Driver SET status = 'S' WHERE name = 'Harvey
    Spectre';
    Query OK, 1 row affected (0.0026 sec)

    Rows matched: 1  Changed: 1  Warnings: 0
    ```
- *Delete driver info*
  - ```
    SQL> DELETE from Driver WHERE phone = '1234567890';
    Query OK, 1 row affected (0.0025 sec)
    ```
- *Enter parking lot info*
  - ```
    SQL> INSERT INTO ParkingLot VALUES ('Oval', 'Oval Dining
    Hall, Centennial Campus');
    ```
    ```

```
Query OK, 1 row affected (0.0031 sec)
```

- *Update parking lot info*
  - SQL> UPDATE ParkingLot SET address= 'Oval Dining, Oval Dr' WHERE lot_name = 'Oval';
    Query OK, 1 row affected (0.0031 sec)

    Rows matched: 1 Changed: 1 Warnings: 0

- *Delete parking lot info*
  - SQL> DELETE from ParkingLot WHERE lot_name = 'Oval';
    Query OK, 1 row affected (0.0120 sec)

- *Enter zone info*
  - SQL> INSERT INTO Zone VALUES('BS','Oval');
    Query OK, 1 row affected (0.0025 sec)

- *Update zone info*
  - SQL> UPDATE Zone SET zone_id = 'A' WHERE zone_id = 'BS' AND lot_name = 'Dan Allen';
    Query OK, 1 row affected (0.0027 sec)

    Rows matched: 1 Changed: 1 Warnings: 0

- *Delete zone info*
  - SQL> DELETE from Zone WHERE zone_id = 'A' and lot_name = 'Dan Allen';
    Query OK, 1 row affected (0.0023 sec)

- *Enter space info*
  - SQL> INSERT INTO Space VALUES(122, 'Dan Allen', 'BS', 'regular', 1);
    Query OK, 1 row affected (0.0022 sec)

- Update space info
  - SQL> UPDATE Space SET availability_status = 0 WHERE zone_id = 'BS' AND lot_name = 'Dan Allen' AND space_number = 122; Query OK, 1 row affected (0.0026 sec)

    Rows matched: 1 Changed: 1 Warnings: 0

- *Delete space info*

    - SQL> DELETE from Space WHERE zone_id = 'BS' AND lot_name = 'Dan Allen' AND space_number = 122;

      Query OK, 1 row affected (0.0026 sec)

- *Enter permit info*
    - SQL> INSERT INTO Permit VALUES(1, 'electric', '2021-10-11', '2022-10-10', '11:00:00', 'residential');
      Query OK, 1 row affected (0.0036 sec)

- *Update permit info*
    - SQL> UPDATE Permit SET space_type = 'handicap' WHERE permit_id = 1;
      Query OK, 1 row affected (0.0039 sec)

      Rows matched: 1 Changed: 1 Warnings: 0

- *Delete permit info*
    - SQL> DELETE from Permit WHERE permit_id = 1;
      Query OK, 1 row affected (0.0033 sec)

- *Assign zones to each parking lot*
    - SQL> INSERT INTO Zone VALUES('D', 'Dan Allen');
      Query OK, 1 row affected (0.0023 sec)

    - SQL> INSERT INTO Zone VALUES('AS', 'Dan Allen');

      Query OK, 1 row affected (0.0024 sec)

- *Assign a type to a given space.*
    - SQL> INSERT INTO Space VALUES(122, 'Dan Allen','BS','regular', TRUE);
      Query OK, 1 row affected (0.0046 sec)

    - SQL> UPDATE Space SET space_type= 'electric' WHERE space_number=122 AND lot_name = 'Dan Allen' AND zone_id = 'BS';
      Query OK, 1 row affected (0.0047 sec)

      Rows matched: 1 Changed: 1 Warnings: 0

- *Request citation appeal*

- SQL> INSERT INTO Appeals VALUES ('4567980055',107, 'Pending');
  Query OK, 1 row affected (0.0034 sec)

- *Update citation payment*
  - SQL> UPDATE Citation set payment_status = 1 WHERE citation_number = 106;
    Query OK, 1 row affected (0.0035 sec)

    Rows matched: 1 Changed: 1 Warnings: 0

- *Citation appeals can be requested and citation payment status updated accordingly*

  Note: This query checks appeal_status is 'Approved' *in the Appeals table and updates the payment_status in Citation table to 1 for those citations.*
  - UPDATE Citation SET payment_status = 1 WHERE citation_number IN (SELECT citation_number FROM Appeals WHERE appeal_status ='Approved');
    Query OK, 1 row affected (0.0038 sec)

    Rows matched: 1 Changed: 1 Warnings: 0

Maintaining permits and vehicle information for each driver

- *Assign permits to drivers*
  - SQL> INSERT INTO IsAssigned(phone, permit_id, car_license_number) VALUES ('5555555555', 1, 'XYZ789');
    Query OK, 1 row affected (0.0032 sec)

- *Enter permit information*
  - SQL > INSERT INTO Permit VALUES (123, 'handicap', '2023-10-14','2024-05-03','12:00:00','commuter');
    Query OK, 1 row affected (0.0033 sec)

- *Update permit information*
  - SQL > UPDATE Permit SET space_type = 'regular' WHERE permit_id = 123;
    Query OK, 1 row affected (0.0032 sec)

    Rows matched: 1 Changed: 1 Warnings: 0

- *Add vehicle*

- SQL> INSERT INTO Vehicle VALUES('KZ63F', 'Corolla', 2006, 'White', 'Toyota');
  Query OK, 1 row affected (0.0036 sec)

- *Update vehicle ownership information*
  - SQL> UPDATE IsAssigned SET phone = '5546199825' WHERE car_license_number = 'DEF456';
    Query OK, 1 row affected (0.0036 sec)

    Rows matched: 1 Changed: 1 Warnings: 0

- *Remove vehicle*
  - SQL> DELETE FROM Vehicle WHERE car_license_number='KZ63F';
    Query OK, 1 row affected (0.0036 sec)

Generating and maintaining citations

- *Detect parking violations by checking if a car has a valid permit in the lot,zone and space.*

  Note: The query returns tuples of parking lots,zones and spaces where the vehicle has a valid parking permit and also the permit expiration date and time. If the vehicle is not parked in the mentioned Lot, Zone and SpaceType, it is a parking violation.

  - SQL> SELECT P.permit_id, space_type, expiration_date, expiration_time, H.zone_id, H.lot_name FROM IsAssigned IA JOIN Permit P ON IA.permit_id = P.permit_id JOIN HasZone H ON H.permit_id = P.permit_id WHERE NOT EXISTS(SELECT H.lot_name, H.zone_id, P.space_type FROM HasZone H JOIN Permit P ON H.permit_id = P.permit_id WHERE H.permit_id IN (SELECT permit_id FROM IsAssigned WHERE car_license_number = 'PQR987')AND lot_name = 'Poulton' AND zone_id = 'A' AND space_type = 'Handicap' AND (CURRENT_TIMESTAMP < expiration_date OR CURRENT_TIMESTAMP = expiration_date AND CURRENT_TIMESTAMP < expiration_time)) AND car_license_number = 'PQR987';

```
+-----------+------------+-----------------+-----------------+---------+----------+
| permit_id | space_type | expiration_date | expiration_time | zone_id | lot_name |
+-----------+------------+-----------------+-----------------+---------+----------+
|         6 | Handicap   | 2023-11-11      | 10:30:00        | A       | Oval     |
+-----------+------------+-----------------+-----------------+---------+----------+
1 row in set (0.0069 sec)
```

- *Generate a citation*
    - SQL> INSERT INTO Citation VALUES(12, '2023-12-30', '1:12:56', 'Invalid Permit', 25.00, 0);

      Query OK, 1 row affected (0.0047 sec)

- *Maintain a citation*

- *Pay citation*
    - SQL> UPDATE Citation set payment_status = 1 WHERE citation_number = 102;
      Query OK, 1 row affected (0.0034 sec)

      Rows matched: 1 Changed: 1 Warnings: 0

- *Appeal citation*
    - SQL> INSERT INTO Appeals VALUES ('9876543210', 102, 'Pending');
      Query OK, 1 row affected (0.0036 sec)

Reports:

- *Generate a report for citations.*
  Note: This query gives us the number of citations, the total number of vehicles to which citations were given and the total fee.
    - SQL> SELECT COUNT(citation_number) AS number_citations,COUNT(DISTINCT car_license_number) AS number_vehicles, SUM(fee) AS total_fees FROM Shows NATURAL JOIN Citation NATURAL JOIN GivenTo;

      ```
      +------------------+-----------------+------------+
      | number_citations | number_vehicles | total_fees |
      +------------------+-----------------+------------+
      |                5 |               5 |      137.5 |
      +------------------+-----------------+------------+
      1 row in set (0.0019 sec)
      ```

- *For each lot, generate a report for the total number of citations given in all zones in the lot for a given month*
    - SQL> SELECT lot_name, COUNT(citation_number) AS number_citations,COUNT(DISTINCT car_license_number) AS

```
                number_vehicles, SUM(fee) AS total_fees FROM Shows
                NATURAL JOIN Citation NATURAL JOIN GivenTo  WHERE
                citation_date like '2023-10-%' GROUP BY lot_name;
+------------+-----------------+-----------------+------------+
| lot_name   | number_citations | number_vehicles | total_fees |
+------------+-----------------+-----------------+------------+
| Carmichael |               1 |               1 |       12.5 |
| Oval       |               2 |               2 |         55 |
| Poulton    |               1 |               1 |         40 |
| Venture    |               1 |               1 |         30 |
+------------+-----------------+-----------------+------------+
4 rows in set (0.0026 sec)
```

- *For each lot, generate a report for the total number of citations given in all zones in the lot for a given year*
    - SQL> SELECT lot_name, COUNT(citation_number) AS number_citations,COUNT(DISTINCT car_license_number) AS number_vehicles, SUM(fee) AS total_fees FROM Shows NATURAL JOIN Citation NATURAL JOIN GivenTo WHERE citation_date like '2024%' GROUP BY lot_name;

```
+----------+-----------------+-----------------+------------+
| lot_name | number_citations | number_vehicles | total_fees |
+----------+-----------------+-----------------+------------+
| Oval     |               1 |               1 |         25 |
| Venture  |               1 |               1 |         30 |
+----------+-----------------+-----------------+------------+
2 rows in set (0.0027 sec)
```

- *Return the list of zones for each lot as tuple pairs (lot, zone)*
    - SQL> SELECT * FROM Zone ORDER BY lot_name;

```
+---------+------------+
| zone_id | lot_name   |
+---------+------------+
| B       | Carmichael |
| BS      | Carmichael |
| AS      | Dan Allen  |
| BS      | Dan Allen  |
| D       | Dan Allen  |
| DS      | Dan Allen  |
| A       | Oval       |
```

```
| BS       | Oval       |
| CS       | Oval       |
| A        | Poulton    |
| V        | Poulton    |
| AS       | Venture    |
| B        | Venture    |
+---------+------------+
13 rows in set (0.0021 sec)
```

- *Return the number of cars that are currently in violation*.
  Note: A vehicle is no more in violation if payment is done for the citation.
    - SQL> SELECT COUNT(DISTINCT car_license_number)AS number_cars_violation FROM GivenTo NATURAL JOIN Citation WHERE payment_status = 0;

```
+-----------------------+
| number_cars_violation |
+-----------------------+
|                     2 |
+-----------------------+
1 row in set (0.0018 sec)
```

- Return the number of employees having permits for a given parking zone.
    - SQL> SELECT COUNT(DISTINCT phone) as number_employees FROM IsAssigned NATURAL JOIN Permit NATURAL JOIN Driver NATURAL JOIN HasZone WHERE zone_id= 'BS' and status='E';

```
+------------------+
| number_employees |
+------------------+
|                1 |
+------------------+
1 row in set (0.0023 sec)
```

- Return permit information given an ID or phone number.
    - SELECT * FROM Permit WHERE permit_id = 1;

```
+-----------+------------+------------+-----------------+-----------------+-------------+
| permit_id | space_type | start_date | expiration_date | expiration_time | permit_type |
+-----------+------------+------------+-----------------+-----------------+-------------+
|         1 | 1          | 2021-10-11 | 2022-10-10      | 11:00:00        | residential |
+-----------+------------+------------+-----------------+-----------------+-------------+
1 row in set (0.0015 sec)
```

- SELECT * FROM Permit WHERE permit_id in (SELECT permit_id FROM IsAssigned WHERE phone = '1234567890');

```
+-----------+------------+------------+-----------------+-----------------+-------------+
| permit_id | space_type | start_date | expiration_date | expiration_time | permit_type |
+-----------+------------+------------+-----------------+-----------------+-------------+
|         1 | 1          | 2021-10-11 | 2022-10-10      | 11:00:00        | residential |
+-----------+------------+------------+-----------------+-----------------+-------------+
1 row in set (0.0018 sec)
```

- Return an available space number given a space type in a given parking lot.

  - SELECT space_number FROM Space WHERE space_type = 'electric' AND lot_name = 'Dan Allen' AND availability_status = 1 LIMIT 1;

```
+--------------+
| space_number |
+--------------+
|          122 |
+--------------+
1 row in set (0.0113 sec)
```

4.2. EXPLAIN directive

- **Return permit information given a phone number.**

1. SQL > EXPLAIN SELECT * FROM Permit WHERE permit_id in (SELECT permit_id FROM IsAssigned WHERE phone = '1234567890');

2.

```
+----+-------------+------------+--------+------------------+---------+---------+-----------
------------------+------+---------------------------------------+
| id | select_type | table      | type   | possible_keys    | key     | key_len | ref
| rows | Extra                                 |
+----+-------------+------------+--------+------------------+---------+---------+-----------
------------------+------+---------------------------------------+
|  1 | PRIMARY     | IsAssigned | ref    | PRIMARY,permit_id | PRIMARY | 12      | const
|    1 | Using where; Using index; LooseScan |
|  1 | PRIMARY     | Permit     | eq_ref | PRIMARY           | PRIMARY | 4       |
atajani.IsAssigned.permit_id |    1 |                                       |
+----+-------------+------------+--------+------------------+---------+---------+-----------
------------------+------+---------------------------------------+
```

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra

1	PRIMARY	IsAssigned	ref	PRIMARY,permit_id	PRIMARY	12	const	1	Using where; Using index; LooseScan
1	PRIMARY	Permit	eq_ref	PRIMARY	PRIMARY	4	atajani.IsAssigned.permit_id	1	

3. `SQL > CREATE INDEX phone_index ON IsAssigned(phone);`

4.

```
     +----+-------------+-----------+--------+-------------------------
-----+---------+---------+-------------------------------+------+---------
----------------------------+
     | id | select_type | table      | type   | possible_keys
| key     | key_len | ref                           | rows | Extra
|
     +----+-------------+-----------+--------+-------------------------
-----+---------+---------+-------------------------------+------+---------
----------------------------+
     |  1 | PRIMARY     | IsAssigned | ref    |
PRIMARY,permit_id,phone_index | PRIMARY | 12      | const
|    1 | Using where; Using index; LooseScan |
     |  1 | PRIMARY     | Permit     | eq_ref | PRIMARY
| PRIMARY | 4       | atajani.IsAssigned.permit_id |    1 |
|
     +----+-------------+-----------+--------+-------------------------
-----+---------+---------+-------------------------------+------+---------
----------------------------+
     2 rows in set (0.0025 sec)
```

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra

1	PRIMARY	IsAssigned	ref	PRIMARY,permit_id,phone_index	PRIMARY	12	const	1	Using where; Using index; LooseScan
1	PRIMARY	Permit	eq_ref	PRIMARY	PRIMARY	4	atajani.IsAssigned.permit_id	1	

- **Return an available space number given a space type in a given parking lot.**

1. `SQL > EXPLAIN SELECT space_number FROM Space WHERE space_type = 'electric' AND lot_name = 'Dan Allen' AND availability_status = 1 LIMIT 1;`
2.

```
+----+-------------+-------+------+---------------+----------+---------+-------+------+----------------------
         --------------+
| id | select_type | table | type | possible_keys | key      | key_len | ref   | rows | Extra
         |
+----+-------------+-------+------+---------------+----------+---------+-------+------+----------------------
         --------------+
|  1 | SIMPLE      | Space | ref  | lot_name      | lot_name | 102     | const |    2 | Using index
         condition;   Using where |
+----+-------------+-------+------+---------------+----------+---------+-------+------+----------------------
         --------------+
```

```
+----+-------------+-------+------+---------------+----------+---------+-------+------+-----------------------------------+
| id | select_type | table | type | possible_keys | key      | key_len | ref   | rows | Extra                             |
+----+-------------+-------+------+---------------+----------+---------+-------+------+-----------------------------------+
|  1 | SIMPLE      | Space | ref  | lot_name      | lot_name | 102     | const |    2 | Using index condition; Using where |
+----+-------------+-------+------+---------------+----------+---------+-------+------+-----------------------------------+
```

3. `SQL > CREATE INDEX spacetype_index ON Space(space_type);`
4.

```
+----+-------------+-------+-------------+-------------------------+-------------------------+----
         -----+------+------+-------------------------------------------------------------    +
```

```
| id | select_type | table | type        | possible_keys            | key                       |
        key_len | ref  | rows | Extra                                                        |
+----+-------------+-------+-------------+--------------------------+---------------------------+----
        -----+------+------+--------------------------------------------------------    +
|  1 | SIMPLE      | Space | index_merge | lot_name,spacetype_index | spacetype_index,lot_name |
        13,102  | NULL |    1 | Using intersect(spacetype_index,lot_name); Using where     |
+----+-------------+-------+-------------+--------------------------+---------------------------+----
        -----+------+------+--------------------------------------------------------
```

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	Space	index_merge	lot_name,spacetype_index	spacetype_index,lot_name	13,102	NULL	1	Using intersect(spacetype_index,lot_name); Using where

4.3 Query Correctness:

1. **Specification**: "Return the single name of all parking lots that have permits with a 'residential' permit type."

   Query: SELECT DISTINCT H.lot_name
   FROM HasLot H JOIN Permit P ON H.permit_id = P.permit_id
   WHERE P.permit_type = 'residential';

   Relational Algebra:

   $\pi_{(lot\_name)}(\delta(\sigma_{(permit\_type = 'residential')}(HasLot \bowtie Permit)))$

   Correctness Proof: Consider all tuples (H, P) resulting from the query, where H represents a parking lot assignment to permit (from the "HasLot" table) and P represents a permit (from the "Permit" table) such that H.permit_id is the same as P.permit_id. Each such combination of tuples (H,P) gives information about a Parking Lot and Permit Id where the output is just the Lot Name and shows the Lot Name only once even if there are multiple occurrences of residential permit in the same Lot. But this is exactly what our query should return; see the specification.

2. **Specification**: "Return the names and university IDs of all employees who drive a Black Toyota vehicle."

   Query: SELECT D.name, D.univ_id
   FROM Driver D
   JOIN IsAssigned IA ON D.phone = IA.phone
   JOIN Vehicle V ON IA.car_license_number = V.car_license_number
   WHERE D.status = 'E' AND V.color = 'Black' AND V.manufacturer = 'Toyota';

   Relational Algebra:

$\pi_{(name,\ univ\_id)}(\sigma_{(status='E'\ AND\ color='Black'\ AND\ manufacturer='Toyota')}((Driver \bowtie IsAssigned) \bowtie Vehicle))$

   Correctness Proof: Consider (D, IA, V) as a representative tuple generated by the query, where D represents a Driver as an employee, IA represents an assignment, and V represents a vehicle such that D.phone is the same as IA.phone, and IA.car_license_number is the same as V.car_license_number, where each tuple reflects a combination of an employee(driver), an assignment, and a vehicle associated with that employee. Employees are determined via D.status and checking if it is 'E', and then conditions for vehicles based on their color (V.color) being 'black' and the manufacturer (V.manufacturer) being 'Toyota' to match Black Toyota vehicles. The query's results include the extraction of the employee's name (D.name) and university ID (D.univ_id) as specified. Ultimately, the query precisely fulfills the requirement to "return the names and university IDs of all employees who drive a Black Toyota vehicle."