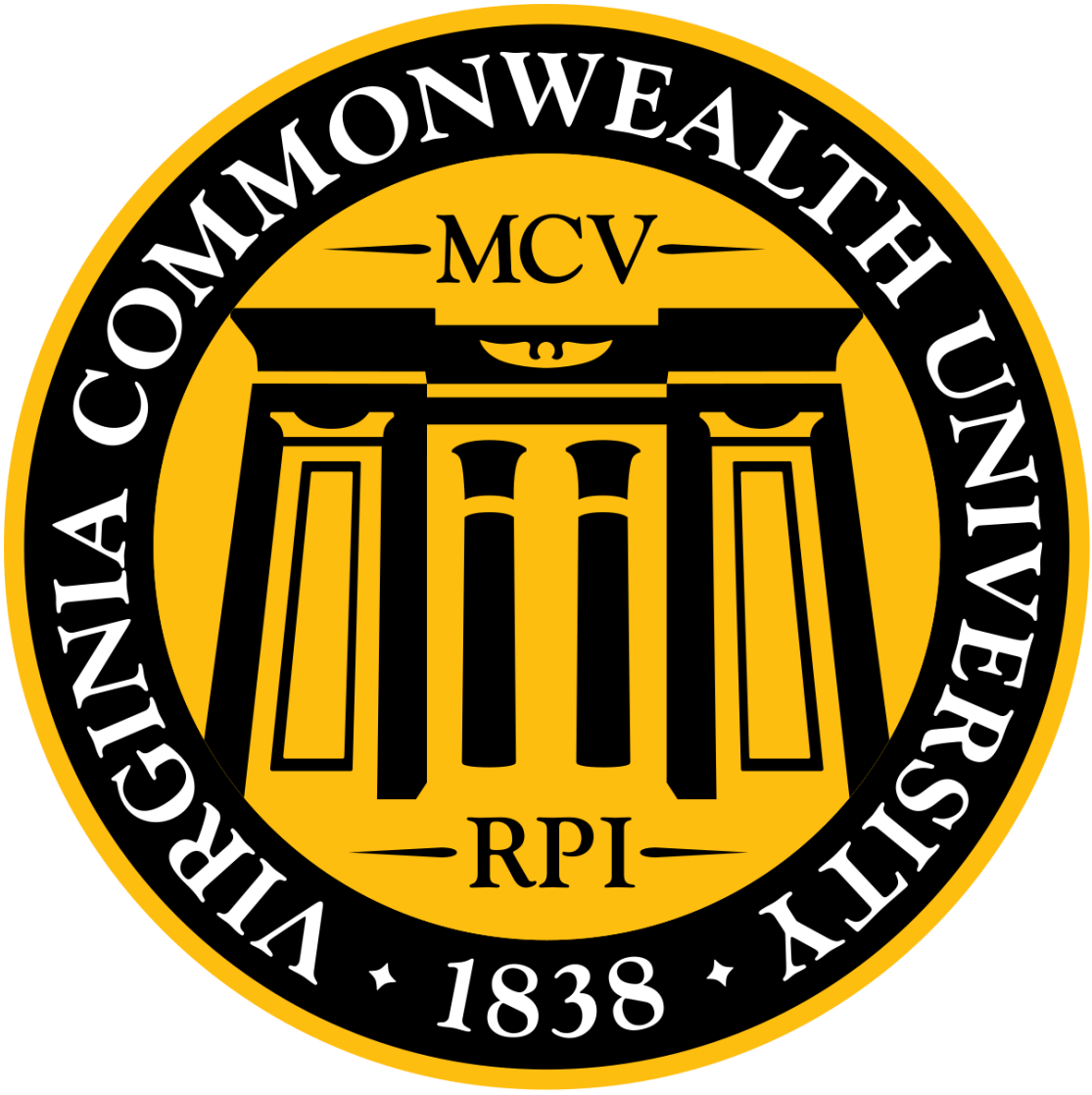
Police Inventory System

CMSC 508 Database Theory



Back-end Design: Sean Kotrola

Front-end Design: Ian Burns

Project Design and Documentation: Benjamin Napier

Security Design and Documentation: Paul Cochran

Problem Statement

The Virginia Commonwealth University Police Department needs a more effective way to manage the hardware components inside each of its vehicles: a task that is currently being accomplished by an overcrowded excel document. Thus, the need for software to keep track of the department's vehicles, mobile computers, mobile computer docks, arbitrator systems, cradlepoint systems, and keyboards. This software would save the police department money by reducing the amount of person-hours needed to track the necessary equipment for ensuring an active vehicle fleet.

Vehicles are all of the police cars that are used for patrolling. As described above, they have a lot of equipment contained within them that must be monitored to ensure that each vehicle is patrol-ready at all times. Since all of the other items that the software will keep track of are housed within the police cars, vehicles will serve as the center of the database for this application.

Mobile computers are the laptops that are used within the police cars. All of these computers are held in place by mobile computer docks. While there are several different types of computer docks, each one only holds one laptop at a time. In addition, each vehicle contains only one mobile computer and one mobile computer dock at a time.

Cradlepoint is the system that brings WiFi to police cars, which the mobile computers connect to. They use SIM cards (similar to phones) to accomplish this. Each vehicle should contain only one cradlepoint system.

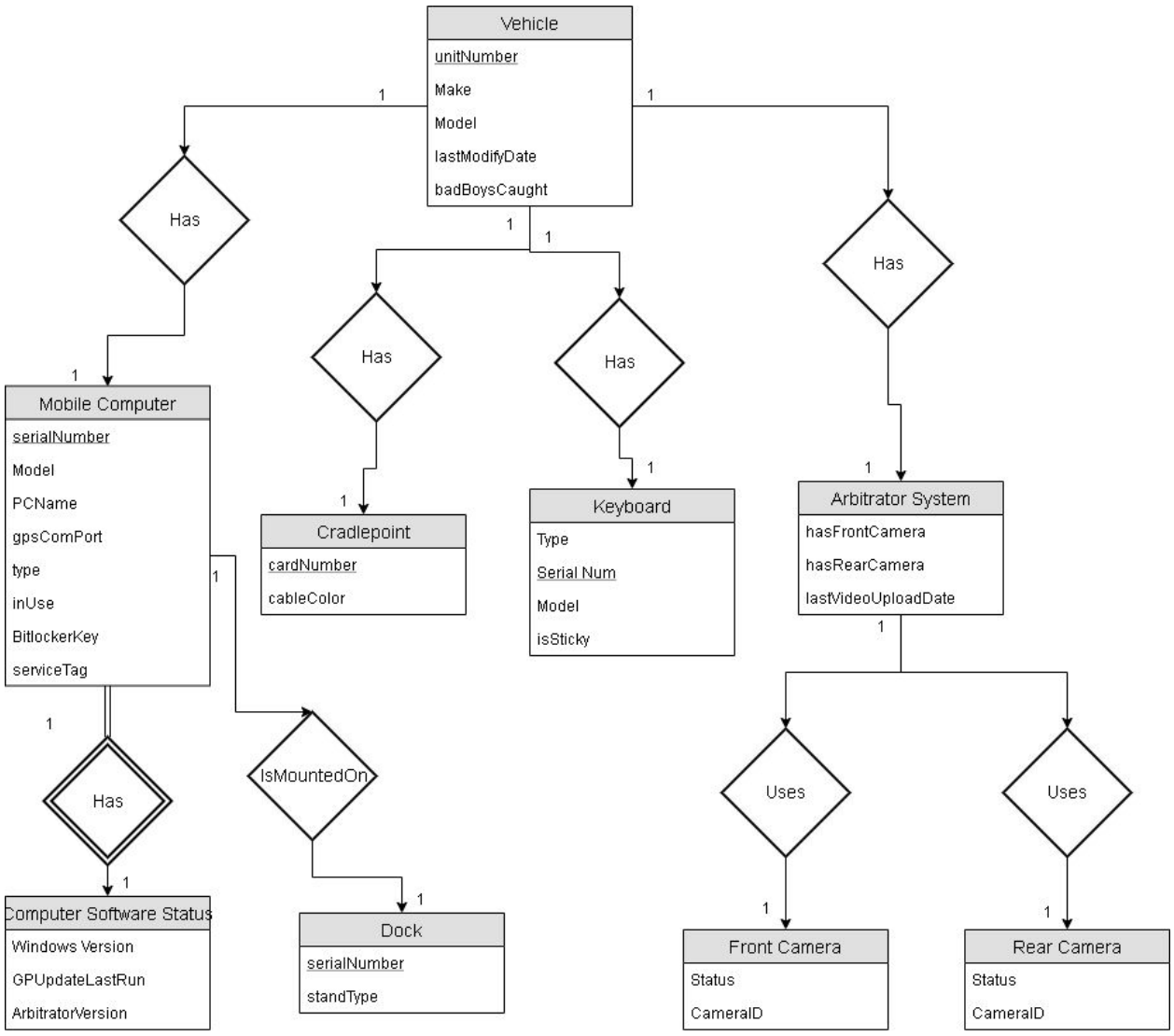
Arbitrator systems are the systems responsible for capturing video footage from police cars. They are a combination of hardware and software that work together to accomplish this goal. The software is installed on the mobile computer, and the hardware includes some combination of a front and rear camera. Each vehicle will only contain one arbitrator system.

There are two types of keyboards that are used. One is membrane and one is mechanical. Depending on the temperature, one type of keyboard will have sticky keys, which makes typing difficult. Each police car should only have one keyboard at a time.

The users for the database will be the police department's IT, dispatch, and property departments. The IT users will use the database to keep track of inventory and ensure that the vehicles are always ready to be actively patrolling. Additionally, they will be responsible for maintaining the database's integrity, and solving any issues associated with the database. They are the only ones who should have admin access. Dispatch will use the database to see which cars have what equipment/items within them. This will allow them to see which vehicles are ready to be used, or which ones are not patrol ready at the moment. Property would use the database in order to keep track of all of the items for the police department. This would allow them to easily see what items are where and which items the department needs to order/replace/fix.

In the future this web application could be applied to more than vehicle inventory management. One example of expansion on this software would be the addition of an inventory tracking system for multiple other portions of the police departments such as the department that keeps track of illicit drug seizures. This software will be purposefully kept generic and not targeted directly as an application for the Virginia Commonwealth University Police Department in order to allow for simpler expansion in future iterations of the project.

Entity-Relationship Diagram



Relational Design

Legend: PrimaryKey, *ForeignKey*

Format:

Entity(attribute1, attribute2, ...**)**

- Attribute1: domain
- Attribute2: domain
- ...

Entities

NOTE: To avoid redundancy, unless otherwise stated, all attributes are assumed to be NOT NULL

Vehicle(unitNumber, make, model, lastModifyDate, badBoysCaught)

- unitNumber: Primary Key, String of digits
- make: String from a list of makes
- model: String from a list of models
- lastModifyDate: Datetime (Can be null)
- badBoysCaught: Integer >= 0

MobileComputer(serialNumber, model, PCName, gpsComPort, type, inUse, bitlockerKey, serviceTag)

- serialNumber: Primary Key, String
- model: String which must match a given list of model names
- PCName: String
- gpsComPort: Integer
- type: String from a list of given types
- inUse: Boolean
- bitLockerKey: String
- serviceTag: Integer >=0

Cradlepoint(cardNumber, cableColor)

- cardNumber: Primary Key, String of digits
- cableColor: String, Color Name

Keyboard(type, serialNumber, model, isSticky)

- type: String - either ‘Mechanical’ or ‘Membrane’
- serialNumber: Primary Key, string of digits
- model: String
- isSticky:Boolean

ArbitratorSystem(ID, hasFrontCamera, hasRearCamera, lastVideoUploadDate)

- ID: Primary Key, integer >= 0
- hasFrontCamera:Boolean
- hasRearCamera: Boolean
- lastVideoUploadDate: Datetime

ComputerSoftwareStatus(*serialNumber*, WindowsVersion, GPSUpdateLastRun, ArbitratorVersion)

- *serialNumber*: Foreign Key to MobileComputer
- WindowsVersion: String as a version number (Windows 10, etc.)
- GPSUpdateLastRun: Datetime
- ArbitratorVersion: String as version number

Dock(serialNumber, standType)

- serialNumber: Primary Key, string
- standType: string from a list of standTypes

FrontCamera(Status, CameraID)

- status: string (“Broken”, “Working”, etc.)
- CameraID: Primary Key, integer

RearCamera(Status, CameraID)

- status: string (“Broken”, “Working”, etc.)
- CameraID: Primary Key, integer

Relationships

Has()

- No Attributes

Uses()

- No Attributes

IsMountedOn()

- No Attributes

Functionality of all the relationships in the diagram

Has(From vehicle to X) - One to One

- One Vehicle can only have one of each of these elements and each element can only exist in one vehicle at a time

Has(From Mobile Computer to Mobile Computer Status) - One to One

- One Mobile computer can only have one status, and each status belongs to only one computer

IsMountedOn - One to One

- A Mobile Computer can only be mounted on one dock at a time, and each dock can only be used with one computer at a time

Uses(x2) - One to One

- An arbitrator system can only use one front and back camera at a time, and a front or back camera can only be used in one arbitrator system at a time

Integrity or consistency constraints

1. All dates must be less than or equal to the current date
2. No Negative numbers are allowed for any numeric field
3. The maximum length of any string field is 256 characters

Database

SQL scripts for creating the database tables, views, triggers, and procedures

Tables:

```
create table Vehicle
(
  unitNumber     int         not null
    primary key,
  Make           varchar(30) not null,
  model          varchar(30) not null,
  lastModifyDate datetime    null,
  badBoysCaught  int         not null
);


create table MobileComputer
(
  serialNumber varchar(10) not null
    primary key,
  model        varchar(20) not null,
  PCName       varchar(15) not null,
  gpsComPort   int         not null,
  type         varchar(20) not null,
  inUse        tinyint(1)  not null,
  bitLockerKey varchar(60) not null,
  serviceTag   int         not null
);


create table CradlePoint
(
  cardNumber varchar(25) not null
    primary key,
  cableColor varchar(10) not null
);
```

```
create table Keyboard
(
  type         varchar(10) not null,
  serialNumber varchar(20) not null
    primary key,
  model        varchar(20) not null,
  isSticky     tinyint(1)  not null
);


create table ArbitratorSystem
(
  ID              int        not null
    primary key,
  hasFrontCamera  tinyint(1) not null,
  hasRearCamera   tinyint(1) not null,
  lastVideoUpdate datetime   not null
);


create table ComputerSoftwareStatus
(
  serialNumber      varchar(10) not null,
  WindowsVersion    varchar(20) not null,
  GpsUpdateLastRun  datetime    null,
  ArbitratorVersion varchar(10) not null,
  constraint ComputerSoftwareStatus_ibfk_1
    foreign key (serialNumber) references MobileComputer (serialNumber)
);


create table Dock
(
  SerialNumber varchar(20) not null
    primary key,
  standType    varchar(30) not null
);



create table FrontCamera
```

```
(
  status   varchar(20) not null,
  cameraID varchar(20) not null
    primary key
);



create table RearCamera
(
  status   varchar(20) not null,
  cameraID varchar(20) not null
    primary key
);


create table User
(
  ID int not null auto_increment,
  username varchar(20)  not null,
  hash varchar(64) not null,
  constraint ID_pk
    primary key(ID)
);

create table UserRole
(
          ID int not null,
          Role varchar(30) not null,
          constraint ID foreign key(ID) references User(ID)
);

create table ArbitratorSystemFrontCamera
(
  CameraID varchar(20) not null,
  ID       int         not null,
  constraint ArbitratorSystemFrontCamera_ibfk_2
    foreign key (ID) references ArbitratorSystem (ID),
  constraint ArbitratorSystemFrontCamera_ibfk_3
    foreign key (CameraID) references FrontCamera (cameraID)
);

create table ArbitratorSystemRearCamera
```

```
(
  CameraID varchar(20) not null,
  ID       int         not null,
  constraint ArbitratorSystemRearCamera_ibfk_2
    foreign key (ID) references ArbitratorSystem (ID),
  constraint ArbitratorSystemRearCamera_ibfk_3
    foreign key (CameraID) references RearCamera (cameraID)
);

create table MobileComputerDock
(
  computer_serialNumber varchar(10) not null,
  dock_serialNumber     varchar(20) not null,
  constraint MobileComputerDock_ibfk_1
    foreign key (computer_serialNumber) references MobileComputer (serialNumber),
  constraint MobileComputerDock_ibfk_2
    foreign key (dock_serialNumber) references Dock (SerialNumber)
);

create table VehicleComputer
(
  unitNumber   int         not null,
  serialNumber varchar(10) not null,
  constraint VehicleComputer_ibfk_1
    foreign key (unitNumber) references Vehicle (unitNumber),
  constraint VehicleComputer_ibfk_2
    foreign key (serialNumber) references MobileComputer (serialNumber)
);

create table VehicleCradlepoint
(
  unitNumber int         not null,
  cardNumber varchar(25) not null,
  constraint VehicleCradlepoint_ibfk_1
    foreign key (unitNumber) references Vehicle (unitNumber),
  constraint VehicleCradlepoint_ibfk_2
    foreign key (cardNumber) references CradlePoint (cardNumber)
);

create table VehicleKeyboard
(
  unitNumber   int         not null,
  serialNumber varchar(20) not null,
```

```
  constraint VehicleKeyboard_ibfk_1
    foreign key (unitNumber) references Vehicle (unitNumber),
  constraint VehicleKeyboard_ibfk_2
    foreign key (serialNumber) references Keyboard (serialNumber)
);
```

Stored Procedures:

```
CREATE PROCEDURE AddVehicle(IN unitNum int(11), IN makeParam varchar(30), IN
ModelParam varchar(30), IN LastModifyDateParam datetime, IN BadBoysCaughtParam int(11))
BEGIN
    insert into Vehicle (unitNumber, Make, model, lastModifyDate, badBoysCaught) VALUES
(unitNum, makeParam, ModelParam, str_to_date(LastModifyDateParam, '%Y-%m-%d
%H:%i:%s'), BadBoysCaughtParam);
END;

create procedure deleteVehicle(In unitNum int(11))
begin
  delete from VehicleKeyboard where unitNumber = unitNum;
  delete from VehicleComputer where unitNumber = unitNum;
  delete from VehicleArbitratorSystem where unitNumber = unitNum;
  delete from VehicleCradlepoint where unitNumber = unitNum;
  delete from Vehicle where unitNumber = unitNum;
end;

create procedure UpdateModifyDate(In unitNum int(11), In lastModifyDateParam datetime)
begin
  Update Vehicle set lastModifyDate = str_to_date(lastModifyDateParam, '%Y-%m-%d
%H:%i:%s') where unitNumber = unitNum;
End
```

Triggers:

```
create TRIGGER BeforeVehicleInsert Before insert on Vehicle
for each row
  begin
    if NEW.badBoysCaught < 0  then
      signal sqlstate '45000' set message_text = 'Bad Boys Caught must be greater than 0';
```

```
    end if;
    call AddVehicle(new.unitNumber, new.make, new.model, new.lastModifyDate,
new.badBoysCaught);
End;

create trigger BeforeInsertKeyboard before insert on Keyboard
  for each row
begin
  if strcmp(new.type, 'Mechanical') != 0 and strcmp(new.type, 'Membrane') != 0 then
    signal sqlstate '45000' set message_text = 'Keyboard type must be Mechanical or
Membrane';
  end if;
  insert into Keyboard (type, serialNumber, model, isSticky) values (new.type,
new.serialNumber, new.model, new.isSticky);
End;


create trigger RemoveComputerSoftwareStatus before delete on MobileComputer
  for each row begin
  delete from ComputerSoftwareStatus where serialNumber = old.serialNumber;
End;


create trigger BeforeInsertMobileComputer before insert on MobileComputer
  for each row begin
  if new.serviceTag < 0 then
    signal sqlstate '45000' set message_text = 'Service Tag must be greater than 0';
  end if;
  insert into MobileComputer (serialNumber, model, PCName, gpsComPort, type, inUse,
bitLockerKey, serviceTag) VALUES
  (new.serialNumber, new.model, new.PCName, new.gpsComPort, new.type, new.inUse,
new.bitLockerKey, new.serviceTag);
End;


create trigger BeforeInsertArbitratorSystem before insert on ArbitratorSystem
  for each row begin
  if new.ID < 0 then
    signal sqlstate '45000' set message_text = 'ID must be greater than 0';
  end if;
  insert into ArbitratorSystem (ID, hasFrontCamera, hasRearCamera, lastVideoUpdate)
VALUES
  (new.ID, new.hasFrontCamera, new.hasRearCamera, new.lastVideoUpdate);
```

```
end;


create trigger BeforeInsertDock before insert on Dock
  for each row begin
  if strcmp(new.standType, 'short') != 0 and strcmp(new.standType, 'tall') != 0 then
    signal sqlstate '45000' set message_text = 'Stand type must either be short or tall';
  end if;
  insert into Dock (SerialNumber, standType) VALUES
  (new.SerialNumber, new.standType);
end;


create trigger BeforeInsertFrontCamera before insert on FrontCamera
  for each row begin
  if strcmp(new.status, 'Broken') != 0 and strcmp(new.status, 'Working') != 0 then
    signal sqlstate '45000' set message_text = 'Status must either be Broken or Working';
  end if;
  insert into FrontCamera (status, cameraID) VALUES
  (new.status, new.cameraID);
end;


create trigger BeforeInsertRearCamera before insert on RearCamera
  for each row begin
  if strcmp(new.status, 'Broken') != 0 and strcmp(new.status, 'Working') != 0 then
    signal sqlstate '45000' set message_text = 'Status must either be Broken or Working';
  end if;
  insert into RearCamera (status, cameraID) VALUES
  (new.status, new.cameraID);
end;
```

Views:

```
create view VehicleWithRecentlyUploadedView as
  SELECT V.unitNumber, Arb.lastVideoUpdate FROM Vehicle V JOIN VehicleArbitratorSystem
VAS ON V.unitNumber = VAS.unitNumber JOIN ArbitratorSystem Arb ON VAS.ID = Arb.ID
ORDER BY lastVideoUpdate DESC;

create view NewestArbitratorVersionVehicleView as
```

```
  SELECT V.unitNumber FROM Vehicle V JOIN VehicleComputer VC ON V.unitNumber =
VC.unitNumber JOIN MobileComputer MC ON VC.serialNumber = MC.serialNumber JOIN
ComputerSoftwareStatus CSS ON MC.serialNumber = CSS.serialNumber WHERE
CSS.ArbitratorVersion LIKE 'NEW';


create view VehcleWithOldestInpectionDateView as
  SELECT unitNumber, lastModifyDate FROM Vehicle ORDER BY lastModifyDate ASC LIMIT 1;

create view ITComputersView as
  select count(serialNumber) as "Number of Computers" from MobileComputer where PCName
= 'IT';


create view VehiclesWithFrontAndRearCameraView as
select count(VAS.unitNumber) as "Number of Vehicles" from ArbitratorSystem join
VehicleArbitratorSystem VAS on ArbitratorSystem.ID = VAS.ID where hasFrontCamera = TRUE
and hasRearCamera = TRUE


create view VehiclesWithKeyboardAndNoComputerView as
  select Vehicle.* from Vehicle join VehicleKeyboard on Vehicle.unitNumber =
VehicleKeyboard.unitNumber where Vehicle.unitNumber not in(select Vehicle.unitNumber from
Vehicle join VehicleComputer on Vehicle.unitNumber = VehicleComputer.unitNumber)

create view ArbitratorWithBadStatusCodesView as
  select ArbitratorSystem.* from ArbitratorSystem join ArbitratorSystemFrontCamera ASFC on
ArbitratorSystem.ID = ASFC.ID join ArbitratorSystemRearCamera ASRC on ArbitratorSystem.ID
= ASRC.ID join FrontCamera FC on ASFC.CameraID = FC.cameraID join RearCamera RC on
ASRC.CameraID = RC.cameraID where FC.status = 'Broken' or RC.status = 'Broken'
```

Interface Software

https://github.com/seanyboy271/Database-Theory-Semester-Project