CISC 450 - HW 7

Marcus Haldane

November 5, 2022

Task 1: (Garden Glory Database) Provide information about the employee named Sam Smith in our sample garden_glory database, including information from the Employee tuple corresponding to Sam Smith, the property service tuples corresponding to Sam Smith in each of the following representations:

(a.) Using JSON, with an appropriate nested representation.

```
{
    "EmployeeID": "1",
    "Name": {
        "FirstName: "Sam",
        "LastName: "Smith"
    },
    "CellPhone": 2062541234,
    "ExperienceLevel": "Master"
    "PropertyServices": [
        {"PropertyServiceID": 1, "PropertyID": 1, "ServiceID": 2,
        "ServiceDate": 2017-05-05, "HoursWorked": 4.50},
        {"PropertyServiceID": 7, "PropertyID": 4, "ServiceID": 4,
        "ServiceDate": 2017-05-05, "HoursWorked": 1.00},
        {"PropertyServiceID": 12, "PropertyID": 4, "ServiceID": 5,
        "ServiceDate": 2017-05-05, "HoursWorked": 5.00}
    ]
}
```

(b.) Using XML, with the same nested representation.

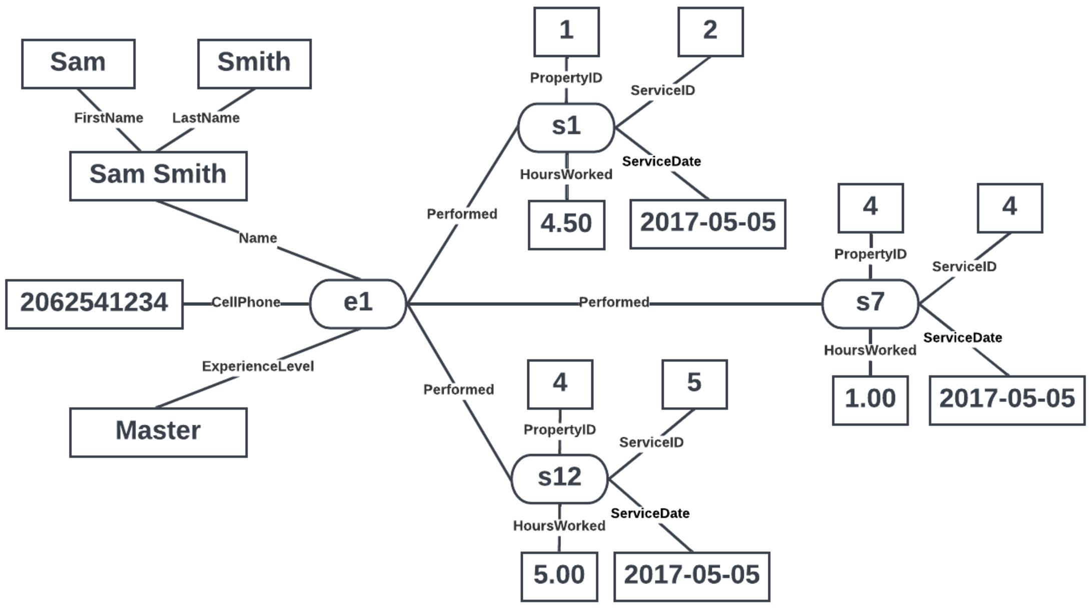
```
<EMPLOYEE>
    <EmployeeID>1</EmployeeID>
    <Name>
        <FirstName>Sam</FirstName>
        <LastName>Smith</LastName>
    </Name>
    <CellPhone>2062541234</CellPhone>
    <ExperienceLevel>Master</ExperienceLevel>
    <PropertyServices>
        <ServiceItem>
            <PropertyServiceID>1</PropertyServiceID>
            <PropertyID>1</PropertyID>
            <ServiceID>2</ServiceID>
            <ServiceDate>2017-05-05</ServiceDate>
            <HoursWorked>4.50</HoursWorked>
        </ServiceItem>
        <ServiceItem>
            <PropertyServiceID>7</PropertyServiceID>
            <PropertyID>4</PropertyID>
            <ServiceID>4</ServiceID>
            <ServiceDate>2017-05-05</ServiceDate>
            <HoursWorked>1.00</HoursWorked>
        </ServiceItem>
        <ServiceItem>
            <PropertyServiceID>12</PropertyServiceID>
            <PropertyID>4</PropertyID>
            <ServiceID>5</ServiceID>
            <ServiceDate>2017-05-05</ServiceDate>
            <HoursWorked>5.00</HoursWorked>
        </ServiceItem>
    </PropertyServices>
</EMPLOYEE>
```

(c.) Using RDF triples.

In this RDF representation, I have pre-pended an 's' onto the instances of the PROPERTY_SERVICE(s) entity, as well as a 'n' onto the instance of the EMPLOYEE entity. As the ID(identifier of entities) for PROPERTY_SERVICE(s) entity and for EMPLOYEE entity are both integers, with the value for the Sam Smith entity being 1, and for the first instance of the PROPERTY_SERVICE being 1 as well, I have made this change for the sake of clarity.

```
(e1 instance-of EMPLOYEE)
(e1 name n1)
(n1 FirstName "Sam")
(n1 LastName "Smith")
(e1 CellPhone 2062541234)
(e1 ExperienceLevel "Master")

(s1 instance-of PROPERTY_SERVICE)
(s1 PropertyID 1)
(s1 ServiceID 2)
(s1 ServiceDate 2017-05-05)
(s1 HoursWorked 4.50)

(s7 instance-of PROPERTY_SERVICE)
(s7 PropertyID 4)
(s7 ServiceID 4)
(s7 ServiceDate 2017-05-05)
(s7 HoursWorked 1.00)

(s12 instance-of PROPERTY_SERVICE)
(s12 PropertyID 4)
(s12 ServiceID 5)
(s12 ServiceDate 2017-05-05)
(s12 HoursWorked 5.00)

(1 Performed s1)
(1 Performed s7)
(1 Performed s12)
```

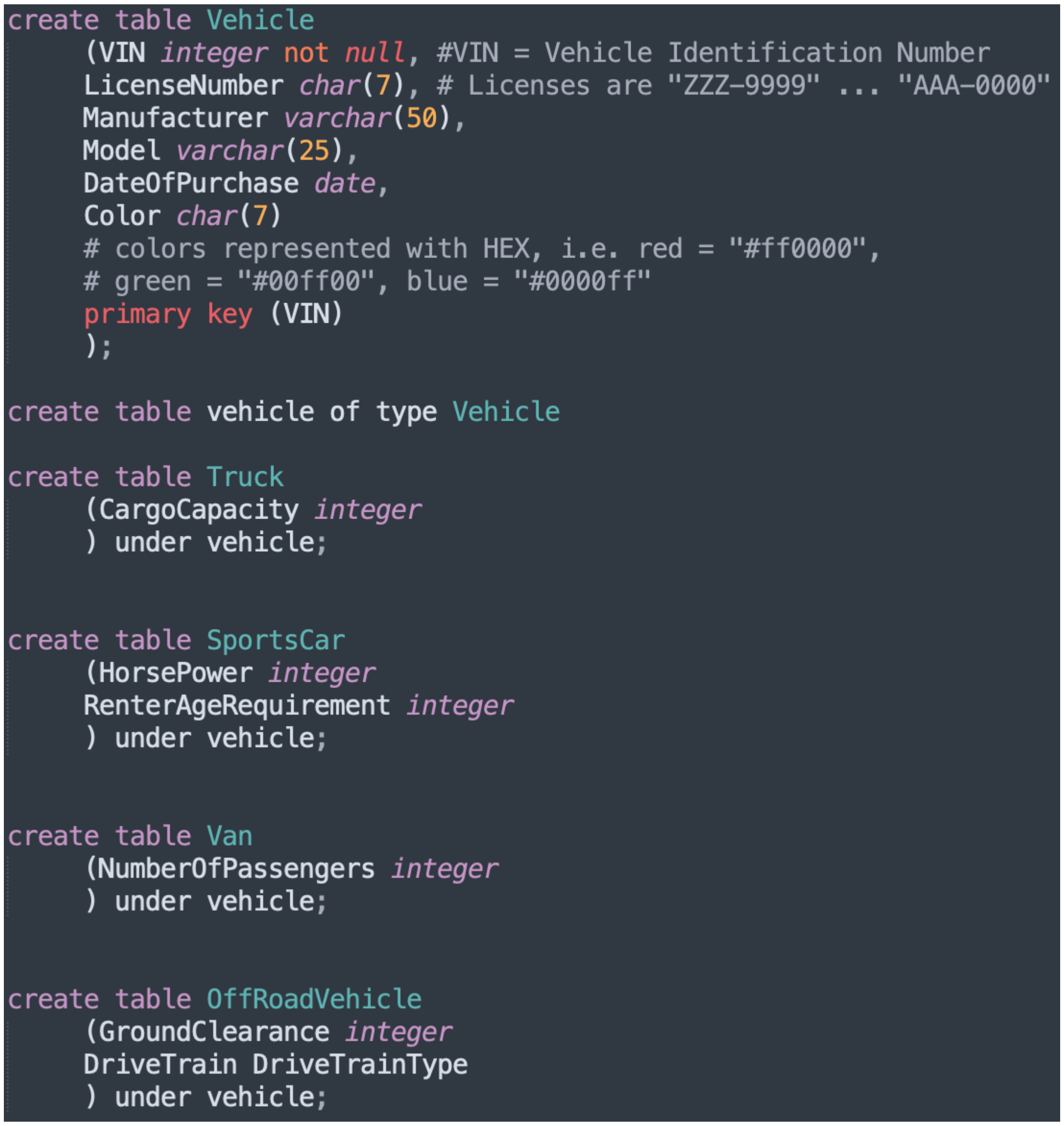
(d.) As an RDF graph.



Task 2: (Object oriented data) A car-rental company maintains a database for all vehicles in its current fleet. For all vehicles, it includes the vehicle identification number, license number, manufacturer, model, date of purchase, and color. Special data are included for certain types of vehicles:

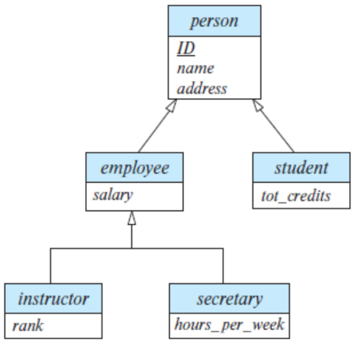
- Trucks: cargo capacity.
- Sports cars: horsepower, renter age requirement.
- Vans: number of passengers.
- Off-road vehicles: ground clearance, drive train (four- or two-wheel drive).

Construct an SQL schema definition for this database. Use inheritance where appropriate.

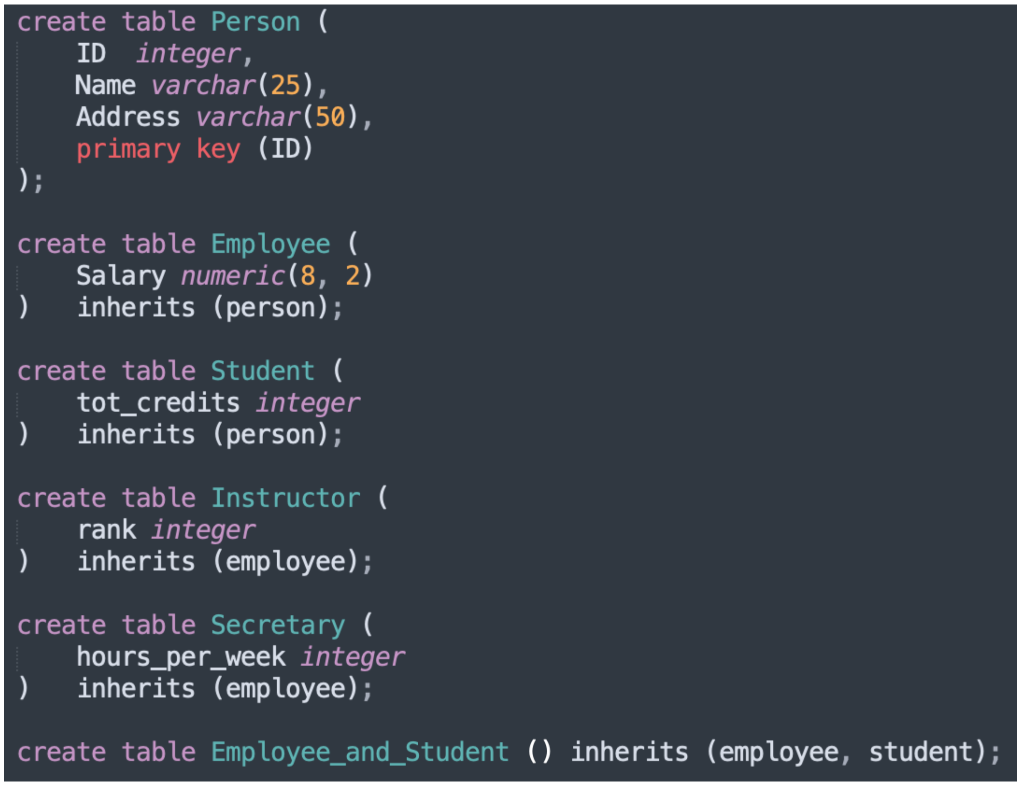


```
create table Vehicle
     (VIN integer not null, #VIN = Vehicle Identification Number
     LicenseNumber char(7), # Licenses are "ZZZ-9999" ... "AAA-0000"
     Manufacturer varchar(50),
     Model varchar(25),
     DateOfPurchase date,
     Color char(7)
     # colors represented with HEX, i.e. red = "#ff0000",
     # green = "#00ff00", blue = "#0000ff"
     primary key (VIN)
     );

create table vehicle of type Vehicle

create table Truck
     (CargoCapacity integer
     ) under vehicle;


create table SportsCar
     (HorsePower integer
     RenterAgeRequirement integer
     ) under vehicle;


create table Van
     (NumberOfPassengers integer
     ) under vehicle;


create table OffRoadVehicle
     (GroundClearance integer
     DriveTrain DriveTrainType
     ) under vehicle;
```

Task 3: (Object-oriented data) Consider the E-R diagram in figure given below, which contains specializations, using subtypes and subtables.



(a.) Give an SQL schema definition of the E-R diagram.



```
create table Person (
    ID  integer,
    Name varchar(25),
    Address varchar(50),
    primary key (ID)
);

create table Employee (
    Salary numeric(8, 2)
)   inherits (person);

create table Student (
    tot_credits integer
)   inherits (person);

create table Instructor (
    rank integer
)   inherits (employee);

create table Secretary (
    hours_per_week integer
)   inherits (employee);

create table Employee_and_Student () inherits (employee, student);
```

(b.) Give an SQL query to find the names of all people who are not secretaries.

```
WITH non_secretaries(id,name) AS
    ((SELECT id, name
    FROM person
    EXCEPT
    (SELECT id, name
    FROM secretary))

SELECT name
FROM non_secretaries
```

(c.) Give an SQL query to print the names of people who are neither employees nor students.

```
SELECT name
FROM ONLY person;
```

(d.) Can you create a person who is an employee and a student with the schema you created? Explain how or explain why it is not possible.
It is in fact possible to create a person who is an employee and a student with the schema I have created above. To create a person of this type, simply perform an insertion for the Employee_and_Student relation. Since the Employee_and_Student relation inherits the attributes of the Employee relation and the Student relation, any instances of the Employee_and_Student relation will also be instances of the Employee relation and the Student relation.

Task 4: (Textual data) Suppose you wish to perform keyword querying on a set of tuples in a database, where each tuple has only a few attributes, each containing only a few words. Does the concept of term frequency make sense in this context? And that of inverse document frequency? Explain your answer. Also suggest how you can define the similarity of two tuples using TF–IDF concepts.

Term frequency querying may not be as useful on small-data-tuples, because the presence of a search term is likely not to be repeated in tuples with only a few attributes. As this is the case, inverse document frequency also does not have a persuasive use case, as it is not likely that there will be repeated and countable instances of the search term in these sorts of tuples. This being said, there is a possibility for using these tools as described in the textbook:

Although querying on structured data are typically done using query languages such as SQL, users who are not familiar with the schema or the query language find it difficult to get information from such data. Based on the success of keyword querying in the context of information retrieval from the web, techniques have been developed to support keyword queries on structured and semi-structured data.

So, although the concepts of term frequency and inverse document frequency are not obviously applicable to this sort of querying, it can be made to work using graphs. By treating each tuple as a node in a graph, with foreign keys represented by edges, valuable information can be gained using TF-IDF concepts.

Task 5: (Textual data) The Google search engine provides a feature whereby web sites can display advertisements supplied by Google. The advertisements supplied are based on the contents of the page. Suggest how Google might choose which advertisements to supply for a page, given the page contents.

Google likely uses data clustering algorithms to group web pages into larger groups or clusters with shared audiences/content. Possible clusters may look like: [Outdoor activities sites, professional sports sites, or technological products sites]. Using click-through rates from sites in the same cluster, successful ads can be shown on other cluster web sites. This allows the ads which are displayed to be relevant to the site which belongs to some designated cluster.

Task 6: (Spatial Data) Suppose the student relation has an attribute named location of type point, and the classroom relation has an attribute location of type polygon. Write the following queries in SQL using the PostGIS spatial functions and predicates that we saw earlier:

(a.) Find the names of all students whose location is within the classroom Packard 101.

```
SELECT name
FROM student
WHERE ST_Contains(
    student.location,
    (
        SELECT location
        FROM classroom
        WHERE building = 'Packard'
            AND room_number = '101'
    )
);
```

(b.) Find all classrooms that are within 100 meters or Packard 101; assume all distances are represented in units of meters.

```
SELECT building, room_number
FROM classroom
WHERE ST_Distance(
    location,
    (
        SELECT location
        FROM classroom
        WHERE building = 'Packard'
            AND room_number = '101'
    )
) <= 100;
```

(c.) Find the ID and name of student who is geographically nearest to the student with ID 12345.

```
WITH distance_to_12345(id, name, distance) AS (
SELECT id, name, ST_Distance(
       location, (
                  SELECT location
                  FROM student
                  WHERE id = '12345'
        )
    ) as distance

    FROM student
    WHERE id <> '12345'
)

SELECT id, name
FROM distance_to_12345
WHERE dis = (SELECT MIN(dis) FROM distance_to_12345);
```