Midterm - CS5200 DBMS 2021 Fall

NAME:

Instructions: Make a copy of this Google document for yourself, so that you can fill in the answers. You can submit a drawing by taking a picture and attaching the image to this document. To submit, export as a PDF and upload to Canvas. Submit by 9:30PM. This exam should be your original, independent work.

1. Database design: draw the UML Diagram for the data model. (35 pts)
Include the class names, attributes, and attribute data types. Include relationship annotations such as the relationship type and cardinality. Reify relationships if necessary. Use the enumerated data type (instead of creating a separate relation for an enumeration). At a minimum, your UML diagram should include a class for each underlined item.

Consider the following description:
Design the data model for an aquarium. In general, there are Animals, which each have a name and a unique AnimalId. There are two specializations of Animals:
1. In addition to the attributes of a general Animal, SeaWater have a pH attribute, which ranges from 0.0 through 14.0. It also has an ocean enumeration, where possible values are PACIFIC, ATLANTIC, INDIAN, ARCTIC.
2. In addition to the attributes of a general Animal, FreshWater have a climate enumeration, where possible values are TROPICAL, DRY, CONTINENTAL, TEMPERATE, POLAR.

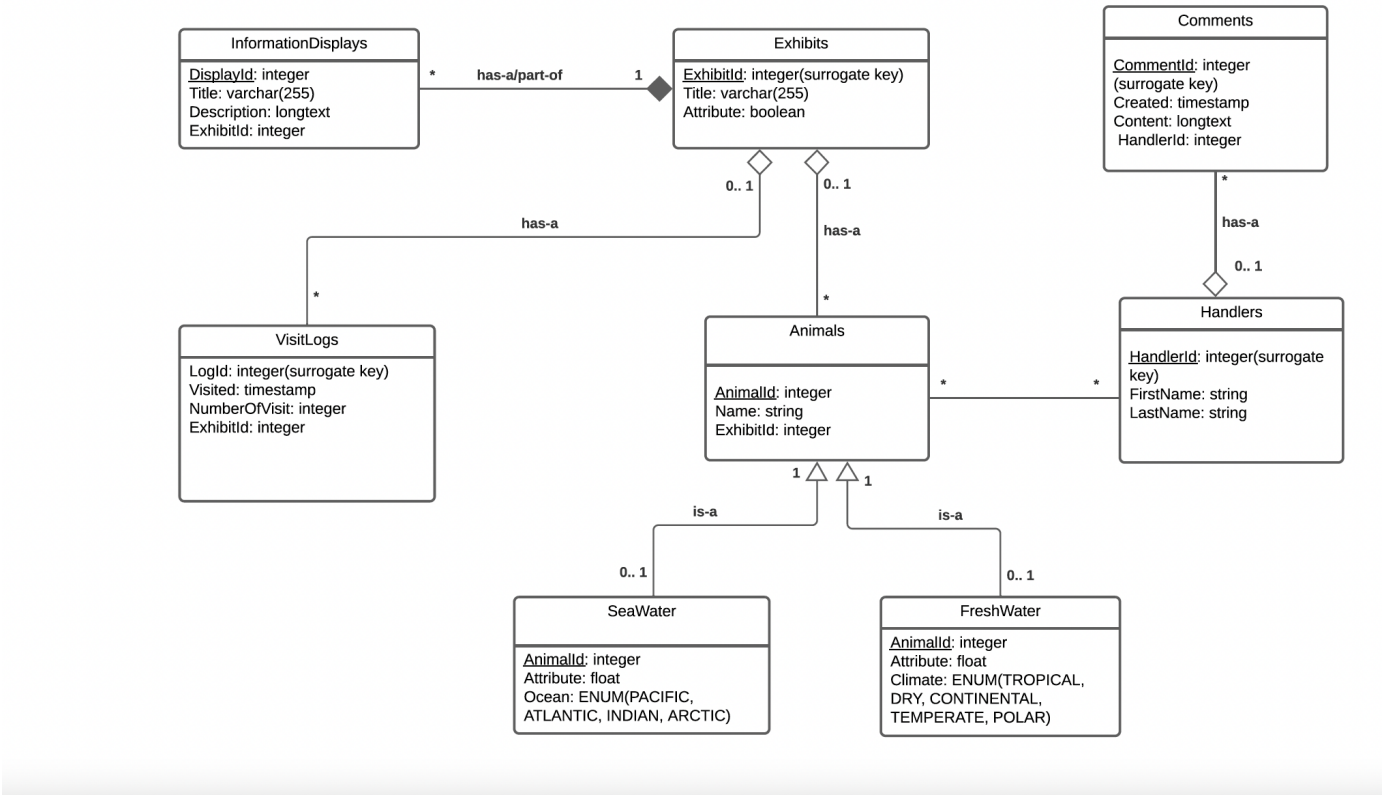
Animals are assigned Handlers, which determines who takes care of which Animals. Handlers have a first name, last name, and unique HandlerId. Any one Animal may have multiple Handler assignments, and any one Handler may have multiple Animal assignments. When an Animal or Handler is removed, then the assignment should be deleted, too.

Handlers can have Comments, which is a way to provide feedback to improve the aquarium. A Handler can have multiple Comments. A comment has a created timestamp, content string, and unique CommentId. When a Handler is removed, the Comments will be kept.

Exhibits contain Animals. Exhibits have a title, an attribute for whether it is interactive or not, and a unique ExhibitId. A single Exhibit may contain multiple Animals. An Exhibit can be removed, but the Animals will continue to exist in the aquarium. In other words, we need to care for the Animals even when there is no exhibit!

An Exhibit can have multiple VisitorLogs records. VisitorLogs have a visited timestamp, a count for the number of visitors at that timestamp, and a unique LogId. When Exhibits are removed, the VisitorLogs records should be kept for future analysis.

Exhibits may have multiple InformationDisplays. InformationDisplays have a title, description, and a unique DisplayId. When an Exhibit is removed, then all its associated InformationDisplays should be removed, too.

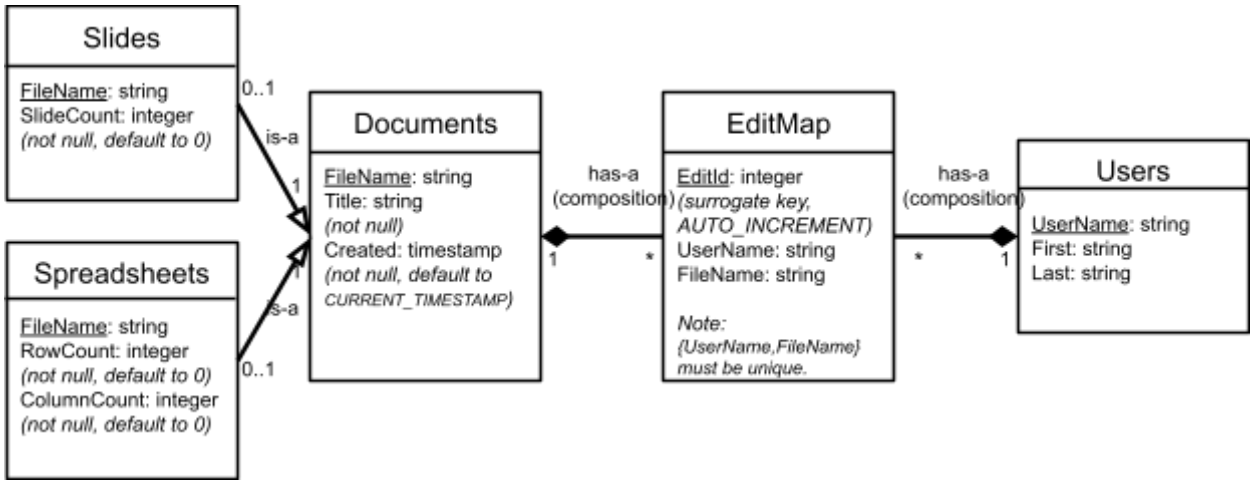


2. Database Design: create the tables for the relational data model and modify data. (30 pts)

2.1. Implement the table definitions using MySQL syntax for the UML diagram below of a document application.

Notes:

- Include CREATE SCHEMA and DROP TABLE statements at the top.
- Make sure your DROP TABLE and CREATE TABLE statements uphold referential integrity and satisfy the order required by the constraints. In other words, your statements must be arranged so that it can be run directly in MySQL Workbench.
- Use the appropriate data types for the attributes. Make reasonable assumptions. Make sure to specify non-nullable attributes and default values if stated in the diagram.
- Include constraints in the table definitions. Primary keys are underlined. Attributes representing foreign keys are also included in the diagram below.
- If applicable, use multi-table inheritance.
- For EditMap, be sure to include the surrogate key and the uniqueness constraint for the {UserName,FileName} tuple.




```
CREATE SCHEMA IF NOT EXISTS DocumentApplication;
USE DocumentApplication;

DROP TABLE IF EXISTS Slides;
DROP TABLE IF EXISTS Spreadsheet;
DROP TABLE IF EXISTS EditMap;
DROP TABLE IF EXISTS Documents;
DROP TABLE IF EXISTS Users;


CREATE TABLE Users (
```

```
  UserName VARCHAR(255),
  First VARCHAR(255),
  Last VARCHAR(255),
  CONSTRAINT pk_Users_UserName PRIMARY KEY (UserName)
);


CREATE TABLE Documents (
  FileName VARCHAR(255),
  Title VARCHAR(255) NOT NULL,
  Created TIMESTAMP DEFAULT CURRENT_TIMESTAMP NOT NULL,
  CONSTRAINT pk_Documents_FileName PRIMARY KEY (FileName)
);


CREATE TABLE EditMap (
  EditId INT AUTO_INCREMENT,
  UserName VARCHAR(255) NOT NULL,
  FileName VARCHAR(255) NOT NULL,
  CONSTRAINT pk_EditMap_EditId PRIMARY KEY (EditId),
  CONSTRAINT fk_EditMap_UserName FOREIGN KEY (UserName)
    REFERENCES Users(UserName)
    ON UPDATE CASCADE ON DELETE CASCADE,
  CONSTRAINT fk_EditMap_FileName FOREIGN KEY (FileName)
    REFERENCES Documents(FileName)
    ON UPDATE CASCADE ON DELETE CASCADE,
  CONSTRAINT uq_EditMap_Editmap UNIQUE(UserName, FileName)
);



CREATE TABLE Slides (
  FileName VARCHAR(255),
  SlideCount INT DEFAULT 0 NOT NULL,
  CONSTRAINT pk_Slides_FileName PRIMARY KEY (FileName),
  CONSTRAINT fk_Slides_FileName FOREIGN KEY (FileName)
    REFERENCES Documents(FileName)
    ON UPDATE CASCADE ON DELETE CASCADE
);

CREATE TABLE SpreadSheet (
  FileName VARCHAR(255),
  RowCount INT DEFAULT 0 NOT NULL,
  ColumnCount INT DEFAULT 0 NOT NULL,
```

```
  CONSTRAINT pk_SpreadSheet_FileName PRIMARY KEY (FileName),
  CONSTRAINT fk_SpreadSheet_FileName FOREIGN KEY (FileName)
    REFERENCES Documents(FileName)
    ON UPDATE CASCADE ON DELETE CASCADE
);
```

2.2. Create, update, and delete data according to the descriptions below. Make sure your statements satisfy the order required by the constraints. More specifically, the order of your statements must be arranged so that it can be run directly in MySQL Workbench.

2.2.1 Insert two users, with the following {UserName, First, Last} values:
{'Jae', 'First1', 'Last1'}
{'Lia', 'First2', 'Last2'}

```
INSERT INTO Users(UserName,First, Last)
  VALUES('Jae', 'First1', 'Last1');
INSERT INTO Users(UserName,First, Last)
  VALUES('Lia', 'First2', 'Last2');
```

2.2.2. Insert a slide deck, with the FileName “deck1”, Title “draft”, and slide count 10. Make sure you insert into the superclass, too. You can utilize the default Created timestamp for the document.

```
INSERT INTO Documents(FileName, Title)
  VALUES('deck1','draft');
INSERT INTO Slides(FileName, SlideCount)
  VALUES('deck1',10);
```

2.2.3. Insert EditMap records for:
UserName “Jae” and FileName “deck1”.
UserName “Lia” and FileName “deck1”.

```
INSERT INTO EditMap(UserName, FileName)
  VALUES('Jae', 'deck1');
INSERT INTO EditMap(UserName, FileName)
  VALUES('Lia', 'deck1');
```

2.2.4 Update the Title to “final” for the FileName “deck1”.

```
UPDATE Documents
SET Title = 'final'
WHERE FileName = 'deck1';
```

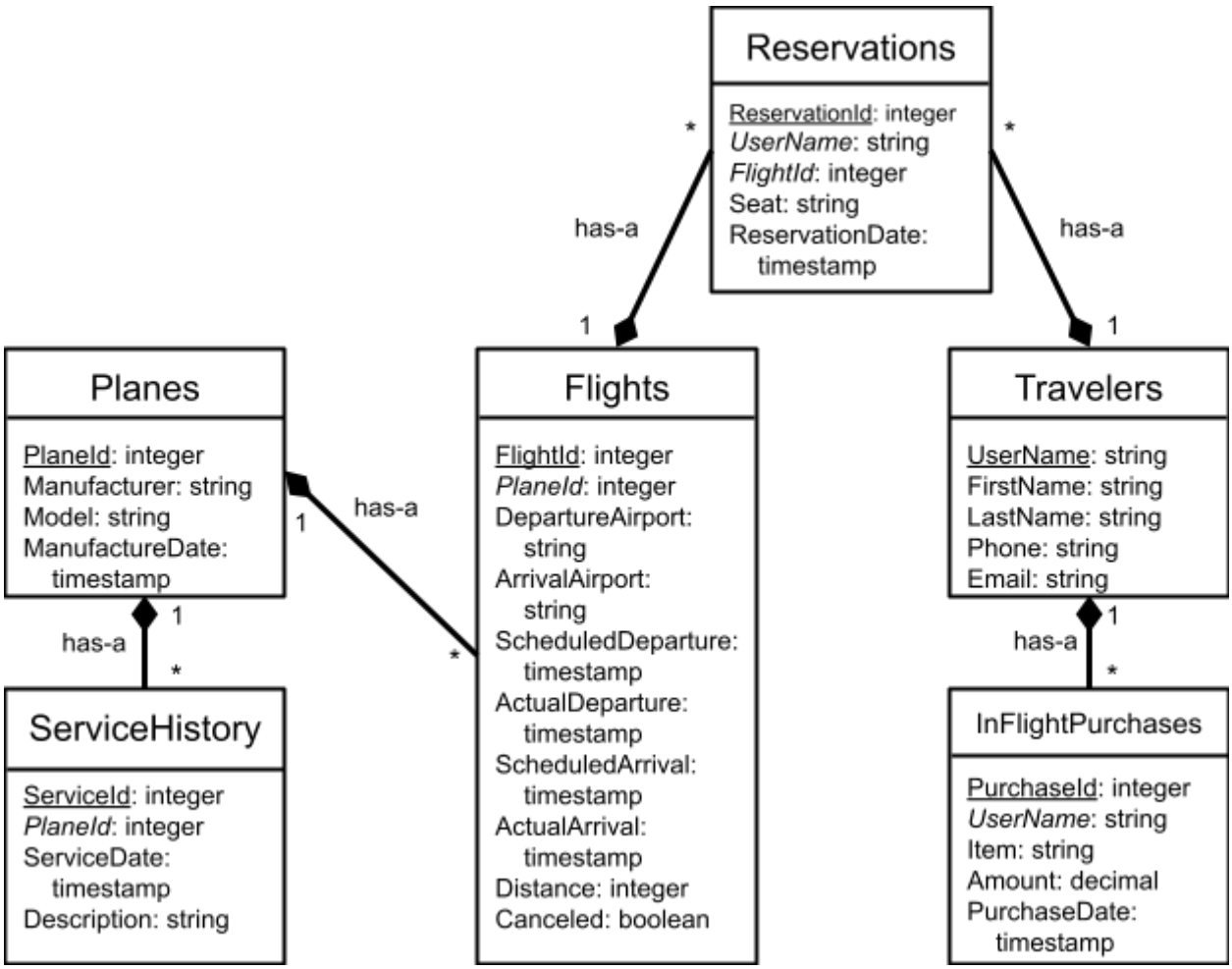
2.2.5 Bulk load a CSV file located at “/tmp/users.csv” into the Users table using the LOAD DATA command. Skip the first line since it represents the header. CSV contents:

```
UserName,First,Last
Name3,First3,Last3
Name4,First4,Last4
Name5,First5,Last5
Name6,First6,Last6
```

```
LOAD DATA LOCAL INFILE '/tmp/users.csv' INTO TABLE Users
  FIELDS TERMINATED BY ','
  LINES TERMINATED BY '\n'
  IGNORE 1 LINES;
```

3. Working with data: write SELECT statements to answer the questions. (35 pts)
Provide a single SELECT statement to answer each question below. Nested SELECT statements can be used. Favor readability and simplicity over complex statements. Also refrain from advanced MySQL shortcuts if applicable (favor standard SQL, although MySQL conveniences such as referencing aliases should be used).

Consider the following UML for an airline application. Primary keys are underlined, *foreign keys are italicized*.



3.1 What are the top 10 Plane {Manufacturer, Model} pairs that have the most ServiceHistory records?

```
SELECT Planes.Manufacturer, Planes.Model
FROM Planes INNER JOIN ServiceHistory ON Planes.PlaneId = ServiceHistory.PlaneId
ORDER BY ServiceHistory.ServiceDate ASC LIMIT 10;
```

3.2 What are the top 10 Traveler UserNames that have traveled the most distance after 2015-01-01?Notes: Use the Flight ActualDeparture timestamp to determine flights after 2015-01-01, and exclude Canceled flights. To compare timestamps, you can use the following expression: ActualDeparture > “2015-01-01”.

```
SELECT Travelers.UserName FROM Travelers INNER JOIN
(SELECT * FROM Flights INNER JOIN Reservations ON Flights.FlightId =
Reservations.FlightID) AS Full ON Travelers.UserName = Full.UserName
WHERE Full.ActualDeparture > “2015-01-01” ORDER BY F.Distance DESC LIMIT 10;
```

3.3 For each PlaneId, what is the total distance traveled in the month of January this year? Notes: Make sure to include all Planes, including Planes that may not have had any flights. Use the ActualDeparture timestamp to determine flight dates, and exclude Canceled flights.

```
SELECT IF(MONTH(Flights. ActualDeparture) = 1 AND YEAR(Flights. ActualDeparture) =
YEAR(CURRENT_DATE()), Flights.distance, 0) FROM Flights;
```

3.4 Of all flights, what is the percentage of late (where ActualArrival is greater than ScheduledArrival) or Canceled flights?

```
SELECT 100 *  (SUM(IF(Flights.ActualArrival > Flights.ScheduledArrival, 1, 0)) +
SUM(IF(Canceled IS NULL, 0, 1))) / COUNT(*) FROM Flights
```

3.5 What are the top 10 UserNames that have booked the most number of Reservations?
Note: you can ignore UserNames that have not booked any reservations.

```
SELECT UserName, COUNT(*) AS num FROM Reservations GROUP BY UserName
ORDER BY num DESC LIMIT 10;
```

3.6 For each UserName, what is the total amount spent through InFlightPurchases?
Note: Make sure to include all UserNames, including UserNames that may not have made any InFlightPurchases.

```
SELECT total.UserName, SUM(total.Amount) FROM (SELECT IF(InFlightPurchases.Amount IS
NULL, 0, InFlightPurchases.Amount) AS Amount FROM Travelers
LEFT OUTER JOIN InFlightPurchases ON Travelers.UserName =
InFlightPurchases.UserName) AS total
GROUP BY total.UserName;
```

3.7 For each UserName, what is the average amount spent through InFlightPurchases per Reservation?
Notes: While you can ignore UserNames that have not booked any reservations, be sure to include UserNames that may not have made any InFlightPurchases. To calculate the average amount spent per reservation: (UserName's total InflightPurchases amount) / (UserName's total number of reservations). This calculation may use concepts from the previous two questions.