COMP1204: DB Coursework

AnishKatariya
ID:27561879
email: ak7n14@soton.ac.uk
22nd April 2016

1 ERD and Normalisation

1.1 EX1

HotelRating (**int**: ReviewID ,**int** : HotelID , **int**:OverallRating, **float**: Avg.Price ,**sting**: URL **string**: Author , **string**: Content , **Date**: date , **int**: No.Reader, **int**: No.Helpful, **int**:Overall ,**int**: Value , **int**: Rooms, **int**: Location , **int**: Cleanliness , **int**: Checkin/FrontDesk,**int**: Services , **int**: BusinessServices)

1.2 EX2

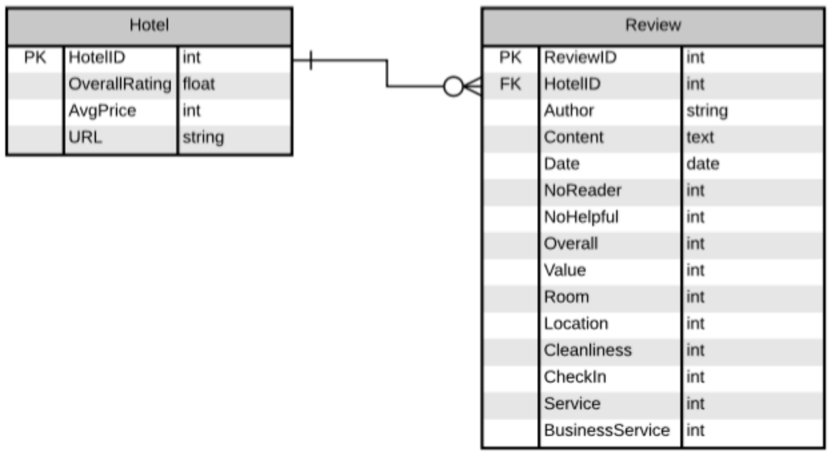
Functional Dependencies:
$HotelID$ $\longrightarrow$OverallRating, $HotelID$ $\longrightarrow$Avg.Price, $HotelID$ $\longrightarrow$URL
$ReviewID$ $\longrightarrow$HotelID , $ReviewID$ $\longrightarrow$Author , $ReviewID$ $\longrightarrow$Content , $ReviewID$ $\longrightarrow$dare ,
$ReviewID$ $\longrightarrow$No.Reader , $ReviewID$ $\longrightarrow$No.Helpful , $ReviewID$ $\longrightarrow$Overall , $ReviewID$ $\longrightarrow$Value,
$ReviewID$ $\longrightarrow$Rooms , $ReviewID$ $\longrightarrow$ Loaction , $ReviewID$ $\rightarrow$Cleanliness ,
$ReviewID$ $\longrightarrow$Checkin/FrontDesk , $ReviewID$ $\longrightarrow$Services , $ReviewID$ $\longrightarrow$BusinessServices

1.3 EX3

Normalised Tabels in BCNF Form:
Hotel(**int**: ReviewID ,**int** : HotelID , **int**:OverallRating, **float**: Avg.Price ,**sting**: URL)

UserReview(**int**: ReviewID ,**int** : HotelID ,**string**: Author , **string**: Content , **Date**: date , **int**: No.Reader, **int**: No.Helpful, **int**:Overall ,**int**: Value , **int**: Rooms, **int**: Location , **int**: Cleanliness , **int**: Checkin/FrontDesk,**int**: Services , **int**: BusinessServices)

1.4 EX4



Hotel		
PK	HotelID	int
	OverallRating	float
	AvgPrice	int
	URL	string

Review		
PK	ReviewID	int
FK	HotelID	int
	Author	string
	Content	text
	Date	date
	NoReader	int
	NoHelpful	int
	Overall	int
	Value	int
	Room	int
	Location	int
	Cleanliness	int
	CheckIn	int
	Service	int
	BusinessService	int

2 Relational Algebra

2.1 EX5

$\sigma_{(Author=userID)}\text{Hotel}\bowtie_{(Hotels.hotelID=Reviews.hotelID)}\ Reviews$

2.2 EX6

$\pi_{Author,Count}(\gamma_{Author,count(*)>2}(Reviews))$

2.3 EX7

$\pi_{HotelID,Count}(\gamma_{hoteID,(count(*)>10)}(Reviews))$

2.4 EX8

$\sigma_{(Hotel\gamma_{(ovrallRating)>3}\wedge Hotel\gamma_{(AvgCleanliness)>=5}(Reviews))}(HotelID)$

3 SQL

3.1 EX9

```
CREATE TABLE HotelReviews(
     ReviewID int,
     HotelID int,
     OverallRating int,
     AvgPrice int,
     URL varchar(255),
     Author varchar(255),
     Content text,
     Date Date,
     NoReader int,
     NoHelpful int,
     Overall int,
     Value int,
     Rooms int,
     Location int,
     Cleanliness int,
     FrontDesk int,
     Services int,
     BusinessService,
     PRIMARY KEY (ReviewID)
);
```

3.2 EX10

Listing 1: bash version

```
#!/bin/bash
#Initializing the review ID as a unique key to
#reconise all the reiews
ReviewID=0
#Echoing values to the file to crearte the table in SQL
echo -n "CREATE TABLE HotelReviews(" >> hotelreviews.sql
        echo -n "ReviewID int, " >> hotelreviews.sql
        echo -n "HotelID int, " >> hotelreviews.sql
        echo -n "OverallRating float(24), " >> hotelreviews.sql
        echo -n "AvgPrice int, " >> hotelreviews.sql
        echo -n "URL varchar(255), " >> hotelreviews.sql
        echo -n "Author varchar(64), " >> hotelreviews.sql
        echo -n "Content text, " >> hotelreviews.sql
        echo -n "Date date, " >> hotelreviews.sql
        echo -n "NoReader int, " >> hotelreviews.sql
        echo -n "NoHelpful int, " >> hotelreviews.sql
        echo -n "Overall int, " >> hotelreviews.sql
        echo -n "Value int, " >> hotelreviews.sql
        echo -n "Rooms int, " >> hotelreviews.sql
        echo -n "Location int, " >> hotelreviews.sql
        echo -n "Cleanliness int, " >> hotelreviews.sql
        echo -n "CheckIn int, " >> hotelreviews.sql
        echo -n "Service int, " >> hotelreviews.sql
        echo "BusinessService int);" >> hotelreviews.sql
#for loop to iterate through all the directory passed as
    #an argument in the command line
for file in $1/*.dat;
do
        #Hotel ID is value assigned to each hotel in the
        #file name where the sting hotel_ has been removed from it
        HotelID=$(basename $file .dat | sed 's/hotel_//'
                    |sed -e "s/'/''/g" | tr -d ',\r');
        echo $HotelID

        #HotelData is a string used to store information about the
        #hoteL like URL, OverallRating etc.
        HotelData=$HotelID","
        #ElementArray contains data which is to be put in the
        #HotelData
        ElementArray=("<Overall\sRating>" "<Avg\.\sPrice>"
                                                "<URL>");
        #for loop iterates through all the data
        #stored in ElementArray

        for element in ${ElementArray[@]};
        do
                #currentString stores data for a specific tag
                currentString=$(grep $element $file | sed 's/'$element'//' |
sed -e "s/'/''/g" |sed -e 's/"/""/g' | tr -d '$,\r')
                #if else statements to put the data in the
                #file in the correct format
                if [ "$currentString" == "" ];
                then
                        HotelData=$(echo $HotelData"NULL,");
                elif [ "$element" == "<URL>" ];
                then
                        HotelData=$(echo $HotelData'"
                                            '$currentString'"'",")
                else
                        # The first character of the string is checked to see
                        #if it is an integer between 0 and 9
                        if [ ${currentString:0:1} -eq
                             ${currentString:0:1} 2> /dev/null ];
                        then
                                HotelData=$(echo $HotelData$currentString
                                                                        ",");
                        else
```

```
                              HotelData=$(echo $HotelData"NULL
                                                              ,");
                        fi
                fi
        done;
        # ElementArray - stores the tags of all the informatioan
        #that must be inserted into review data
        ElementArray=("<Author>" "<Content>" "<Date>" "<No\.\sReader>"
        "<No\.\sHelpful>" "<Overall>" "<Value>" "<Rooms>" "<Location>"
        "<Cleanliness>" "<Check\sin\s\/\sfront\sdesk>""<Service>"
        "<Business\sservice>");
  #ReviewData - an array of strings used to all the
  #data specific to a review
        ReviewData=()
        for element in ${ElementArray[@]};
        do
                #checks the current file for all instances of the data stored
                in element and when
                #found formts the string and stores the final data in the
                currentElements array
                mapfile -t currentElements < <(grep $element $file | sed
                's/'$element'//' |  sed -e "s/'/''/g"
             | sed -e 's/"/""/g' | tr -d '\r')
                #ReviewData stores strings which are then concatenated with
                #the data found with the tags specifies in ElementArray
                for((i=0; i<${#currentElements[@]}; i++))
                do
                        if [[ "$element" == "<Author>" || "$element" ==
                        "<Content>|| "$element" == "<Date>" ]];
                        then
                                ReviewData[$i]=$(echo ${ReviewData[$i]}'"
                            ${currentElements[$i]}'"'",")
                        elif [ "$element" ==
                             "<Business\sservice>" ];
                        then
                                ReviewData[$i]=$(echo ${ReviewData[$i]
                                }${currentElements[$i]}")")
                        else
                                ReviewData[$i]=$(echo ${ReviewData[$i]}${
                                currentElements[$i]}",")
                        fi
                done;
        done;
        #for loop to iterate  through each data in
        #ReviewData and write them to the file as an SQL Quary
        for((i=0; i<${#ReviewData[@]}; i++))
        do
                ReviewID=$(expr $ReviewID + 1)
```

```
                    echo "INSERT INTO HotelReviews
                    (ReviewID, HotelID, OverallRating, AvgPrice,
                    URL, Author,Content, Date, NoReader, NoHelpful
                    , Overall, Value, Rooms, Location, Cleanliness,
                    CheckIn, Service, BusinessService) VALUES ("$ReviewID",
                    "$HotelData${ReviewData[$i]}";" >> hotelreviews.sql
          done;
done;
```

3.3 EX11

Normalized Form tables:

```
CREATE TABLE Hotels(
     HotelID int,
     OverallRating int,
     AvgPrice int,
     URL varchar(255),
     PRIMARY KEY (HotelID)
);

CREATE TABLE HotelReviews(
     ReviewID int,
     HotelID int,
     Author varchar(255),
     Content text,
     Date Date,
     NoReader int,
     NoHelpful int,
     Overall int,
     Value int,
     Rooms int,
     Location int,
     Cleanliness int,
     FrontDesk int,
     Services int,
     BusinessService,
     PRIMARY KEY (ReviewID)
);
```

3.4 EX12

INSERT into Hotels (HotelID, OverallRating, AvgPrice, URL) SELECT hr.HotelID, hr.OverallRating, hr.AvgPrice, hr.URL FROM HotelReviews hr;
INSERT into Reviews (ReviewID, HotelID, Author, Content, Date, NoReader, NoHelpful, Overall, Value, Rooms, Location, Cleanliness, CheckIn, Service, BusinessService) SELECT hr.ReviewID, hr.HotelID, hr.Author, hr.Content, hr.Date, hr.NoReader, hr.NoHelpful, hr.Overall, hr.Value, hr.Rooms, hr.Location, hr.Cleanliness, hr.CheckIn, hr.Service, hr.BusinessService FROM HotelReviews hr;

3.5 EX13

CREATE INDEX hID ON Hotels(HotelID) CREATE INDEX rID ON Reviews (ReviewID)
I chose HotelID and ReviewID as indexes as they refer to unique rows for each value in the table

3.6 EX14

3.6.1 1

SELECT * FROM hotelreviews WHERE Author = userID

3.6.2 2

SELECT Review , COUNT(*) FROM hotelreviews WHERE COUNT(DISTINCT Author) >2 GROUP By Author

3.6.3 3

SELECT HotelID FROM hotelreviews WHERE COUNT(DISTINCT(HotelID)) >10

3.6.4 4

SELECT HotelID FROM hotelreviews WHERE Overall Rating >3 AND AVG(cleanliness) >5 GROUP BY HotelID

4 Conclusion

My approach for the course work was to revise through the content while going through the exercises. The difficulty in the coursework was faced while writing the bash script. The Use of punctuation symbols present in the reviews made it even harder to put the reviews provided in a format which could be used to create a table in SQL. These difficulties were overcome by an excessive amount of google search and the use of sed and grep commands.To save time I created a review folder containing the review of only one hotel and tested each line individually to make it easier to spot mistakes. After the bash script was working on a small data set I tried to run it on the data se provided. The table took around 1 hour to create but worked perfectly. After I got the bash script to work the rest of the coursework seemed quite easy and straight forward and did not take much time to do.