Data Management Coursework 2

March-May 2019
Emilia Szynkowska
30186773
eas1g18@soton.ac.uk

1 The Relational Model

1.1 EX1

Relation in the hotel dataset:

Hotel ID
Overall Rating
Average Price
URL
ReviewID
Author
Content
Date
Readers
Helpful
Overall
Value
Rooms
Location
Cleanliness
Check In/Front Desk
Service
Business Service

The dataset contains 825 hotels which each have reviews within them. Each review is associated with a hotel which has an ID, Overall Rating, Average Price, and URL. The Author, Content, and Date are also given, along with a series of numbers representing scores of various aspects of the hotel. The primary key is ID.

1.2 EX2

Candidate Keys:
ID
URL

Primary Key:
ID

Functional Dependencies:
ID $\rightarrow$ Overall Rating
ID $\rightarrow$ Average Price
ID $\rightarrow$ URL
URL $\rightarrow$ ID
URL $\rightarrow$ Overall Rating
URL $\rightarrow$ Average Price

1.3 EX3

Normalisation can be used to reduce redundancy in the data.

Original Relation:

Hotel ID	Overall Rating	Average Price	URL	Review ID	Author	Date	Content	Readers	Helpful	Overall	Value	Rooms	Location	Cleanliness	Check In/Front Desk	Service	Business Service
25729	5	179	hotel1.com	1	Author1	5 Jan 2009	Content..	1	1	5	5	5	5	5	4	5	5
				2	Author2	7 Feb 2008	Content..	2	1	3	5	4	5	5	3	5	5
				3	Author3	27 Mar 2010	Content..	1	1	4	5	4	5	5	4	5	5
43021	2	120	hotel2.com	1	Author4	1 Apr 2019	Content..	5	5	2	2	3	4	1	3	3	3
				2	Author4	16 Jul 2003	Content..	3	1	2	1	1	2	1	2	2	3

This relation is not very efficient as information is repeated. There is a one-to-many relationship between hotel IDs and reviews. There are also dependencies in the data.

1st Normal Form: where every relation is a single-valued attribute

Hotel ID	Overall Rating	Average Price	URL	Review ID	Author	Date	Content	Readers	Helpful	Overall	Value	Rooms	Location	Cleanliness	Check In/Front Desk	Service	Business Service
25729	5	179	hotel1.com	1	Author1	5 Jan 2009	Content..	1	1	5	5	5	5	5	4	5	5
25729	5	179	hotel1.com	2	Author2	7 Feb 2008	Content..	2	1	3	5	4	5	5	3	5	5
25729	5	179	hotel1.com	3	Author3	27 Mar 2010	Content..	1	1	4	5	4	5	5	4	5	5
43021	2	120	hotel2.com	1	Author4	1 Apr 2019	Content..	5	5	2	2	3	4	1	3	3	3
43021	2	120	hotel2.com	2	Author4	16 Jul 2003	Content..	3	1	2	1	1	2	1	2	2	3

Multi-valued fields are separated.

2nd & 3rd Normal Form: where there are no partial or transitive dependencies

Hotel ID	Overall Rating	Average Price	URL
25729	5	179	hotel1.com
43021	2	120	hotel2.com

Hotel ID	Review ID	Author	Date	Content	Readers	Helpful	Overall	Value	Rooms	Location	Cleanliness	Check In/Front Desk	Service	Business Service
25729	1	Author1	5 Jan 2009	Content..	1	1	5	5	5	5	5	4	5	5
25729	2	Author2	7 Feb 2008	Content..	2	1	3	5	4	5	5	3	5	5
25729	3	Author3	27 Mar 2010	Content..	1	1	4	5	4	5	5	4	5	5
43021	1	Author4	1 Apr 2019	Content..	5	5	2	2	3	4	1	3	3	3
43021	2	Author4	16 Jul 2003	Content..	3	1	2	1	1	2	1	2	2	3

Here the multi-valued attributes are put into a separate relation. This means information about each hotel is not repeated, and also reduces insertion and deletion errors. In the second relation Hotel ID is a foreign key and allows the two relations to be linked. It is also in BCNF because for each functional dependency X $\rightarrow$ Y, X is a super key.

Candidate Keys:
Hotel ID
URL
Hotel ID, Review ID

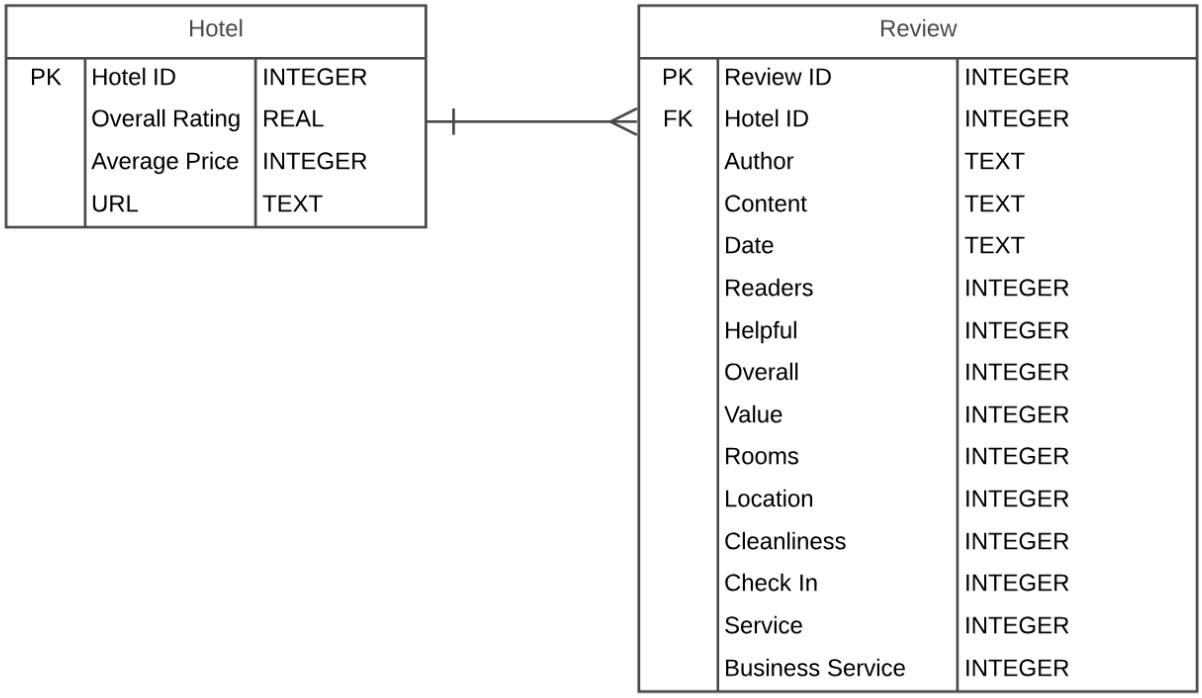
Primary Keys:
Hotel ID
Hotel ID, Review ID

Functional Dependencies:
Hotel ID $\rightarrow$ Overall Rating
Hotel ID $\rightarrow$ Average Price
Hotel ID $\rightarrow$ URL
URL $\rightarrow$ Hotel ID
URL $\rightarrow$ Overall Rating
URL $\rightarrow$ Average Price
Hotel ID, Review ID $\rightarrow$ Author, Date, Content...

2 Entity-Relationship Diagramming

2.1 EX4

Entity Relationship Diagram:



3 Relational Algebra

3.1 EX5

Finding all the reviews by the same author:

π author, content (σ author=author (Category))

3.2 EX6

Finding all the users with the number of reviews greater than 2:

π author, number (σ number $>$ 2 γ author; COUNT(author)$\rightarrow$number (Reviews))

3.3 EX7

Finding all the hotels with the number of reviews greater than 10:

π hotelID, number (σ number $>$ 10 γ hotelID; COUNT(hotelID)$\rightarrow$number (Reviews))

3.4 EX8

Finding all the hotels with Overall Rating $>$ 3 and Average Cleanliness $\geq$ 4:

table1 = π hotelID, overallRating (Hotel)
table2 = π hotelID, average γ hotelID; AVG(cleanliness$\rightarrow$average (Review)
π hotelID, overallRating, average (table1 $\bowtie$ table2)

4 SQL Queries

4.1 EX9

Sqlite3 script to create the database:

```
CREATE TABLE HotelReviews (
    HotelID INTEGER,
    OverallRating REAL,
    AveragePrice INTEGER,
    URL TEXT,
    ReviewID INTEGER,
    Author TEXT,
    Content TEXT,
    Date TEXT,
    Readers INTEGER,
    Helpful INTEGER,
    Overall INTEGER,
    Value INTEGER,
    Rooms INTEGER,
    Location INTEGER,
    Cleanliness INTEGER,
    CheckIn INTEGER,
    Service INTEGER,
    BusinessService INTEGER
);
```

4.2 EX10

Bash script to populate the database:

```
#!/bin/bash

> ex10.sql
echo "DROP TABLE IF EXISTS HotelReviews;" >> ex10.sql
echo "CREATE TABLE HotelReviews(HotelID INTEGER, OverallRating REAL,
    AveragePrice INTEGER,URL TEXT, ReviewID INTEGER, Author TEXT, Content TEXT,
    Date TEXT, Readers INTEGER, Helpful INTEGER, Overall INTEGER, Value INTEGER,
    Rooms INTEGER, Location INTEGER, Cleanliness INTEGER, CheckIn INTEGER, Service
    INTEGER, BusinessService INTEGER);" >> ex10.sql
directory=$1
IFS=$'\n'
for file in $directory/*
do {
      HotelID=$(basename $file .dat)
      OverallRating=$(grep '<Overall Rating>' $file | sed 's/<Overall Rating>//
          g;s/\r//')
      AveragePrice=$(grep '<Avg. Price>' $file | sed 's/<Avg. Price>\$//g;s/\r
          //;s/,//g')
      if [[ $AveragePrice != [0-9]* ]]
        then AveragePrice="null"
      fi
      URL=$(grep '<URL>' $file | sed 's/<URL>//g;s/\r//')

      declare -a author
      declare -a content
      declare -a dates
      declare -a readers
      declare -a helpful
      declare -a overall
      declare -a value
      declare -a rooms
      declare -a location
      declare -a cleanliness
      declare -a checkin
      declare -a service
      declare -a business

      author+=($(grep -F '<Author>' $file | sed "s/<Author>//g;s/'/''/g;s/\r//"
          ))
      content+=($(grep -F '<Content>' $file | sed "s/<Content>//g;s/'/''/g;s/\r
          //"))
      dates+=($(grep -F '<Date>' $file | sed 's/<Date>//g;s/\r//'))
      readers+=($(grep -F '<No. Reader>' $file | sed 's/<No. Reader>//g;s/\r
          //'))
      helpful+=($(grep -F '<No. Helpful>' $file | sed 's/<No. Helpful>//g;s/\r
          //'))
      overall+=($(grep -F '<Overall>' $file | sed 's/<Overall>//g;s/\r//'))
      value+=($(grep -F '<Value>' $file | sed 's/<Value>//g;s/\r//'))
      rooms+=($(grep -F '<Rooms>' $file | sed 's/<Rooms>//g;s/\r//'))
      location+=($(grep -F '<Location>' $file | sed 's/<Location>//g;s/\r//'))
```

```
        cleanliness+=($(grep -F '<Cleanliness>' $file | sed 's/<Cleanliness>//g;s
            /\r//'))
        checkin+=($(grep -F '<Check in / front desk>' $file | sed 's/<Check in \/
             front desk>//g;s/\r//'))
        service+=($(grep -F '<Service>' $file | sed 's/<Service>//g;s/\r//'))
        business+=($(grep -F '<Business service>' $file | sed 's/<Business
            service>//g;s/\r//'))

        for (( n=0; n<${#author[@]}; n++ )); do {
             x=$(($n+1))
             echo "INSERT INTO HotelReviews (HotelID, OverallRating, AveragePrice,
                 URL, ReviewID, Author, Content, Date, Readers, Helpful, Overall, Value,
                 Rooms, Location, Cleanliness, CheckIn, Service, BusinessService)
                 VALUES($HotelID, $OverallRating, $AveragePrice, '$URL', $x, '${author
                 [$n]}', '${content[$n]}', '${dates[$n]}', ${readers[$n]}, ${helpful[
                 $n]}, ${overall[$n]}, ${value[$n]}, ${rooms[$n]}, ${location[$n]}, ${
                 cleanliness[$n]}, ${checkin[$n]}, ${service[$n]}, ${business[$n]});
                 " | sed 's/hotel_//g' >> ex10.sql
        } done

        author=()
        content=()
        dates=()
        readers=()
        helpful=()
        overall=()
        value=()
        rooms=()
        location=()
        cleanliness=()
        checkin=()
        service=()
        business=()

} done
```

4.3 EX11

Sqlite3 script to create Hotels and Reviews:

```
.read hotelreviews.db
CREATE TABLE Hotels (
    HotelID INTEGER,
    OverallRating REAL,
    AveragePrice INTEGER,
    URL TEXT,
    );
CREATE TABLE Reviews (
    HotelID INTEGER,
    ReviewID INTEGER,
    Author TEXT,
    Content TEXT,
    Date TEXT,
    Readers INTEGER,
```

```
    Helpful INTEGER,
    Overall INTEGER,
    Value INTEGER,
    Rooms INTEGER,
    Location INTEGER,
    Cleanliness INTEGER,
    CheckIn INTEGER,
    Service INTEGER,
    BusinessService INTEGER
);
```

4.4 EX12

SQL to populate Hotels and Reviews:

```
DROP TABLE IF EXISTS Hotels;
DROP TABLE IF EXISTS Reviews;
CREATE TABLE Hotels(HotelID INTEGER, OverallRating REAL, AveragePrice INTEGER,
    URL TEXT);
CREATE TABLE Reviews(HotelID INTEGER, ReviewID INTEGER, Author TEXT, Content TEXT
    , Date TEXT, Readers INTEGER, Helpful INTEGER, Overall INTEGER, Value INTEGER,
    Rooms INTEGER, Location INTEGER, Cleanliness INTEGER, CheckIn INTEGER, Service
     INTEGER, BusinessService INTEGER);
INSERT INTO Hotels(HotelID, OverallRating, AveragePrice,URL) SELECT DISTINCT
    HotelID, OverallRating, AveragePrice,URL FROM HotelReviews;
INSERT INTO Reviews(HotelID, ReviewID, Author, Content, Readers, Helpful, Overall,
    Value,Rooms, Location, Cleanliness, CheckIn, Service, BusinessService) SELECT
    HotelID, ReviewID, Author, Content, Readers, Helpful, Overall, Value, Rooms,
    Location, Cleanliness, CheckIn, Service, BusinessService FROM HotelReviews;
```

4.5 EX13

For this dataset I have created some indices which group certain data. These allow commonly used fields to be accessed more efficiently.

Index	Values	Reason
hotel_url	HotelID, URL	Quick access of hotel names and corresponding websites
author_content	Author, Content	Looking up users and comments, useful if you are not interested in the integer ratings and only want subjective data
author_content_date	Author, Content, Date	Looking up users, comments, and dates, useful if you are looking at review contents on specific dates
review_values	ReviewID, HotelID, Overall, Value, Rooms, Location, Cleanliness, CheckIn, Service, BusinessService	Quick access of only numerical values, useful if calculating averages or looking at the data objectively

Code to create indices in sqlite3:

```
CREATE INDEX hotel_url ON Hotels(HotelID,URL);
CREATE INDEX author_content on Reviews(Author,Content);
```

```
CREATE INDEX author_content on Reviews(Author,Content,Date);
CREATE INDEX review_values on Reviews (ReviewID,HotelID,Overall,Value,Rooms,
    Location,Cleanliness,CheckIn,Service,BusinessService);
```

5 SQL Queries

5.1 EX14

EX5 - Finding all the reviews by the same author:

```
SELECT Content FROM HotelReviews WHERE Author='author';
```

EX6 -Finding all the users with the number of reviews greater than 2:

```
SELECT Author FROM HotelReviews GROUP BY Author HAVING COUNT(Content) > 2;
```

EX7 - Finding all the hotels with the number of reviews greater than 10:

```
SELECT HotelID FROM HotelReviews GROUP BY HotelID HAVING COUNT(*) > 10;
```

EX8 - Finding all the hotels with Overall Rating $>$ 3 and Average Cleanliness $\geq$ 4:

```
CREATE TEMP TABLE temptable1 (Hotel1 ,OverallRating);
CREATE TEMP TABLE temptable2 (Hotel2 ,AvgCleanliness);

INSERT INTO temptable1 (Hotel1 ,OverallRating) SELECT HotelID,OverallRating
    FROM Hotels;
INSERT INTO temptable2 (Hotel2 ,AvgCleanliness) SELECT DISTINCT HotelID,AVG(
    Cleanliness) FROM HotelReviews GROUP BY HotelID;

SELECT Hotel1,OverallRating,AvgCleanliness FROM temptable1 INNER JOIN
    temptable2 WHERE Hotel1 = Hotel2 AND OverallRating > 3 AND AvgCleanliness
    >= 4.5;

DROP TABLE IF EXISTS temptable1;
DROP TABLE IF EXISTS temptable2;
```