7CCSMDDW: Coursework 2

Lecturer: Natalia Criado

Coursework assigned: 11 November 2019
Coursework deadline: 4:00 pm, 09 December 2019
Late Submissions deadline (capped at 50%): 4:00 pm, 10 December 2010

Overview

The purpose of this coursework is to create a normalised database and a data warehouse given a dataset. The coursework is formally assessed and is worth 10% of your final grade and is broken down to 10 points distributed below. You will receive some feedback as part of the marking of the coursework.

Part 1: Data Loading, Transformation & Normalization (7 out of 10 points)

- **(1.1)** Backup Restoring (0.5 points)
- **(1.2)** Data Loading (1.5 points)
- **(1.3)** Data Preprocessing (3 points)
- **(1.4)** Data Integration (0.5 points)
- **(1.5)** Database Normalization (1.5 points)

Part 2: Constraint Implementation (2 out of 10 points)

- **(2.1)** Status Consistency (1 point)
- **(2.2)** Status Transition (1 point)

Part 3: Data Warehouse (1 out of 10 points)

- **(3.1)** ER Diagram (0.5 point)
- **(3.2)** SQL Statements (0.5 point)

Requirements

Working with raw crime data from a police department involves analysing a sequence of records about crimes. There are two different tables in this dataset. A table with crime data and a table with images of crime locations. The data is shipped in three different files: (i) *crimes2015.sql* containing information about crimes reported in 2015 and the images of the locations of all crimes, (ii) *crimes2013.txt* containing information about crimes reported in 2013, and (iii) *crimes2014.csv* containing information about crimes reported in 2014.
Each record in the crime table represents an crime, with the following information:

- `date_reported`: date when the crime was reported to the police
- `date_occ`: date when the crime occurred
- `time_occ`: time when the crime occurred

- `occurred`: the timestamp when the crime occurred. This column replaces the `date_occ` and `time_occ` in the *crimes2015.sql* file.
- `dr_no`: crime id
- `area`: code of the area where the crime occurred
- `area_name`: name of the area where the crime occurred
- `rd`: road number
- `crime_no`: code of the type of crime
- `crime_desc`: description of the type of crime
- `status`: code of the crime status
- `status_desc`: description of the crime status
- `image_no`: code of the image of the place where the crime occurred

Each record in the image table contains the image of a crime location:

- `image_no`: image id
- `image`: image stored a the BLOB value (it is a binary large object that can hold a variable amount of data)

In a big city, there is a large number of crimes, so being able to read and analyse the data efficiently is very important. For this assignment you will import raw crime data into a MySQL database. As is typical of big data, there are many inconsistencies and anomalies in the dataset, so you may need to pre-process and clean the data before it is usable.

Part 1: Data Loading, Transformation & Normalisation

Setup. In Part 1, you will use SQL commands to load, transform and normalise the crime data. On KEATs you will find a .zip file which contains template SQL files to edit for your Part 1 submission.

Be sure to:

1. Include your NAME at the top of the file in a SQL line comment.
2. Edit these files as text, not Word files or propriety SQL software.
3. Do NOT rename the files.
4. Include any comments using the SQL line comment (i.e. lines beginning with $--$).

Submission and Evaluation. Include your edited versions of these files in the .zip of your Part 1 submission. We will evaluate these SQL files using NMS's server (which is version 5.5.50-MariaDB precisely). As a student, you can utilize NMS's server —recall it is accessible via: `https://nms.kcl.ac.uk/mysql`— or your own database server installed on your personal computer to test your files and commands.

You can assume that these files are executed in the order that they appear in the description that follows:

1. `backupRestoring.sql`
2. `dataLoading.sql`
3. `dataPreprocesing.sql`

4. `dataIntegration.sql`
5. `normalization.sql`

Any file that is missing, renamed, or does not run will result in 0 points for that section.

Part 1.1: Backup Restoring. Write the required SQL statement/s to restore the data contained in the file *crimes2015.sql* in the provided template file: `backupRestoring.sql`

Part 1.2: Data Loading. Write the SQL statements to perform the following operations:

- 1.2.1. Table Creation. Write the required SQL DDL (Data Definition Language) statements (i.e. CREATE TABLE) to create two tables (named *crimes2013* and *crimes2014*) that can held the data contained in the datasets *crimes2013.txt* and *crimes2014.csv*. Ensure that:
  - table and attribute names do not conflict with SQL reserved words
  - attribute data types are core SQL data types as described in the lectures
- 1.2.2. Data Load. Write the required SQL statements to load the data contained in the datasets *crimes2013.txt* and *crimes2014.csv* into the tables *crimes2013* and *crimes2014*.

Write all of these SQL statements in the above order in the provided template file: `dataLoading.sql`

Part 1.3: Data Preprocessing. Write the required SQL statements to remove any duplicates, inconsistencies and anomalies in the data (that is in the tables *crimes2013*, *crimes2014* and *crimes2015*). Among others, the following inconsistencies and anomalies need to be solved:

- The tables in the different datasets do not have the same number of columns.
- The column `time_occ` in some of the tables may not be formatted properly (e.g., the value '1' should be interpreted as '00:01').
- The column `image_no` in some of the tables may contain invalid values.
- The capitalization of the status codes (`status`) are not consistent in the three tables.
- Different formats may be used for the dates.
- The data about some area names are missing.

Write all of the SQL statements required to preprocess the data in the provided template file: `dataPreprocesing.sql`

Part 1.4: Data Integration. Write the required SQL statements to integrate the data in the tables *crimes2015*, *crimes2014* and*crimes2013* in a single table named *crimes*. Write these SQL statements in the provided template file: `dataIntegration.sql`

Part 1.5: Database Normalization. Write the required SQL statements to normalize the crimes table into a set of tables in 3NF. Ensure that:

- every table has a primary key
- all foreign keys are properly declared

Write these SQL statements in the provided template file: `normalization.sql`

Part 2: Constraint Implementation

Setup. In Part 2, you will use SQL commands to implement domain constraints in the tables contained in the *crime2015.sql* backup. On KEATs you will find a .zip file which contains template SQL files to edit for your Part 2 submission. Again, be sure to:

1. Include your NAME at the top of the file in a SQL line comment.
2. Edit these files as text, not Word files or propriety SQL software.
3. Do NOT rename the files.
4. Include any comments using the SQL line comment (i.e. lines beginning with $--$).

Submission and Evaluation. Include your edited versions of these files in the .zip of your Part 2 submission. We will evaluate these SQL files using NMS's server (which is version 5.5.50-MariaDB precisely). As a student, you can utilize NMS's server —recall it is accessible via: `https://nms.kcl.ac.uk/mysql`— or your own database server installed on your personal computer to test your files and commands.

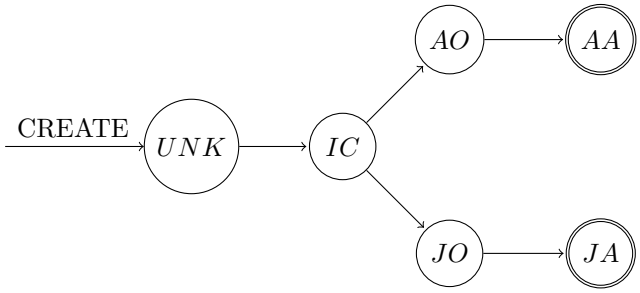
You can assume that these files are executed in the order that they appear in the description that follows:

1. `backupRestoring.sql`
2. `constraints.sql`

Any file that is missing, renamed, or does not run will result in 0 points for that section.
Write the SQL statements to perform the following operations:

Part 2.1: Status Description Consistency. Create MySQL stored program/s (e.g., trigger) to ensure that the `status` and `status_desc` are consistent (e.g., it is not possible to have two different status descriptions for the same status code).

Part 2.2: Status Transition. Create MySQL stored program/s (e.g., trigger) to ensure that the status code of crimes behaves as indicated by the following transition diagram:



That is the crimes are created with the status UNK, then crimes can transition to status IC, from that status they can transition to AO or JO, etc. Crimes can only be deleted when they are in status AA or JA.

Write all of these SQL statements in the above order in the provided template file: `constraints.sql`

Part 3: Data Warehouse

Setup. In Part 3, you will create a data warehouse to analyse the number of crimes contained in the *crimes2015.sql* backup. On KEATs you will find a .zip file which contains template SQL files to edit for your Part 3 submission. Again, be sure to:

1. Include your NAME at the top of the file in a SQL line comment.
2. Edit these files as text, not Word files or propriety SQL software.
3. Do NOT rename the files.
4. Include any comments using the SQL line comment (i.e. lines beginning with −−).

Submission and Evaluation. Include your edited versions of these files in the .zip of your Part 3 submission.

1. ER Diagram. Submit your ER Diagram as a PDF document (not a Word document for compatibility reasons).

   Use a diagramming program such as Microsoft Visio, OneNote, PowerPoint, Illustrator etc... for you ER diagram. Do not submit hand-drawn diagrams, as they are usually unreadable! You can also use web apps such as `http://draw.io` to help you sketch out your diagrams.

   Include your pdf in the .zip of your Part 2 Coursework submission.

2. SQL Statements. In Part 3, you will also develop SQL statements to create a data warehouse. Include your edited version of the file `warehouse.sql` in the .zip of your Part 3 submission.

   We will evaluate these SQL files using NMS's server (which is version 5.5.50-MariaDB precisely). As a student, you can utilize NMS's server —recall it is accessible via: `https://nms.kcl.ac.uk/mysql`— or your own database server installed on your personal computer to test your files and commands.

   You can assume that these files are executed in the order that they appear in the description that follows:

   (a) `backupRestoring.sql`
   (b) `warehouse.sql`

   Any file that is missing, renamed, or does not run will result in 0 points for that section.

Part 3.1: Entity-Relationship Diagram. The police department would like to perform different queries related to the occurrence of individual crimes, by type, area, status, and by when they occurred and when they where reported. Given the data available, design a data warehouse that meets these requirements. Draw the ER Diagram of the star schema of this data warehouse. Your diagram must also cover the requirements below:

- Include all entities, attributes (including data types), and relationships. If necessary, please explain/justify your design choices.

- Cardinalities and Assumptions. On the ER Diagram include all cardinalities of the relationships using the ER Diagram notation from the lecture. State assumptions made and ensure that those assumptions do not contradict with the coursework requirements. Please list your assumptions as bullet points.
- Constraints. (i) For each relation, identify their primary and foreign keys. Simply add (PK) and (FK) to indicate the attribute(s) that serve as primary and foreign keys, respectively.

Part 3.2: SQL Statements. Write the SQL statements to build and populate the data warehouse in accordance with your ER Diagram. Note that the police would like to keep the data they have already stored in the *crimes2015.sql* file; you should ensure that the police does not lose any data as a result of the creation of the data warehouse.
Write your SQL statements in the provided template file: `warehouse.sql`