Lab 4

Due: Multiple Dates

The final deliverable for this assignment is the collection of scripts constructing your team's version of CalAccess LEAKS Database (shared via github) and the constructed database itself (placed in one of the MySQL databases your team works with).

The final due date for these two deliverables to be ready for inspection is Friday, June 8.

Overview

The purpose of this lab is to get you to start the implementation of the project in earnest. The lab has several milestones for when some materials/deliverables have to be submitted, but the final set of deliverables are due at the end of the quarter.

Task 1: Finalize your E-R model and Database Design.

Due date: Tuesday, May 21, before class

You will receive final feedback from the Instructor and from our customer about your E-R model and your database design throughout the next two weeks. Based on this feedback, you will prepare one last revision of your E-R model design and your database design artifacts, and will submit them (create a Lab04-DB directory in your team's course directory, place files there, let the instructor know when the files are placed) by the due date.

Your changes to the DB model need to reflect that actual DB design you are using for your project.

Task 2: Create the LEAKS DB database.

As part of this lab assignment, we are sharing with you access to the CalAccess database available on the same MySQL server to which you have been granted access. The CalAccess database contains one table per TSV file released by the Secretary of State office as part of the CalAccess dataset. Each table has the same name as the corresponding TSV file. The contents of each relational table are row-for-row the contents of the TSV file. The column names are preserved. All attributes have VARCHAR type (with sufficient size to accommodate all content). There are no constraints on any of the tables. Each of the MySQL accounts provided to you from now on has SELECT permissions in the CalAccess database. To avoid parsing
issues, we recommend that you use the CalAccess database for your data ingestion needs. The CalAccess database you have access to will have static content representing the state of the CalAccess dataset on May 6, 2024.

Should you need access to the original TSV files the course web site has the link to the SoS web page from which you can download them.

The main task of this assignment is for you to create the CalAccess LEAKS DB as your team designed it from the CalAccess data available to you.

We recommend that you use Python for accessing the CalAccess raw data, converting it into CalAccess LEAKS DB format, and populating the LEAKS DB tables you created. The organization of your code base is left to each individual team, however, as your scripts for creating LEAKS DB are the key deliverable for the overall project, please ensure that all your code is properly documented.

Please make sure that all access credentials are stored in separate files read by your scripts, and NEVER COMMITTED TO YOUR CODE REPOSITORIES.

Each team shall create a GitHub repository for their code. At the end of the quarter this repository shall contain both the code for creation of the CalAccess LEAKS DB, and the code for creation of the CalAccess LEAKS Data Warehouse, so structure your repository appropriately. Once you create the repository, please add users dekhtyar, and tmgerrit to the repository. We won't be interfering with your work during the quarter, but once the code is due, we will collect it from the repositories you share with us.

**Addendum: Access to the Digital Democracy Database Tables and Their Use in Your Project**

This part of the document describes relational database tables from the Digital Democracy project database (DDDB) that you are receiving access to. It also provides specifications for how these tables shall be used/integrated with the CalAccess LEAKS DB you are developing.

The version of the DDDB database you will be accessing is stored in the DDB2016Aug database on the MySQL server you were provided access to. This database has hundreds of tables, so to keep things streamlined, we are providing you with SELECT permissions on only several of the tables in that database. Each account provided for your team now has SELECT permissions to the following tables (for clarity, and because this is how you may wind up using these tables in your work, all tables are provided with their DB prefix):
1. **DDDB2016Aug.Bill**: information about the bills considered in the California state legislature. This table contains Bill IDs used throughout the project that you need to match to the bill information from the Lobbying Activity forms.

2. **DDDB2016Aug.BillVersion**: information about individual versions of each bill. The Bill table has no text fields describing the contents of the bill (as these might evolve with the evolution of the Bill from version to version). So if you need/want to access attributes such as the name of he bill, or its short descriptions, these are available in the BillVersion table. Note: BillVersion is in a many-to-one relationship with the Bill table.

3. **DDDB2016Aug.Person**: the comprehensive list of all individuals observed within the DDDB. The Pid attributed that serves as the primary key of this table is also the primary key of the Lobbyist and Legislator tables (see below).

4. **DDDB2016Aug.Lobbyist**: the list of individuals that the Digital Democracy project currently has identified as Registered Lobbyists.

5. **DDDB2016.Legislator**: the list of individuals who, at one time or another, have served in the State Legislature (note - it includes historic information, not just the info on the 120 individuals currently serving in the Legislature. As such, this might not be the most useful table for some of your purposes).

6. **DDDB2016.Organizations**: the list of names by which an organization has been referred to in any official filings (in the CalAccess and other databases), or in a transcript of a hearing. Each unique mention is stored in this table separately, and has its own unique primary key.

7. **DDDB2016.OrgConcept**: the list of actual organizations. Each row in this table is in a many-to-one relationship with multiple rows in the Organizations table (all the ways in which this organization has been referred to).

**Note**: There are additional tables in the DDDB whose contents might become of use to a specific team. At this point, to keep things (relatively) streamlined, we are only sharing the contents of the seven tables above. Any additional information sharing requests will be considered as the need for them arises.

After some deliberations, we (Thomas Gerrity and I) have converged on the following specifications regarding the use of the DDDB tables in your project. **Note: these specifications are mandatory.** If you want to implement a different approach, please talk to use first and receive an explicit permission to do so.

1. **Bill (SELECT permissions).** Use the DDB2016Aug table to authenticate the bills whose IDs (e.g. "AB 17", "SB 193") you find in the Lobbying Activity filings. Use FOREIGN KEY (<blah>) REFERENCES DDB2016Aug.Bill(Bid) in your LEAKS DB tables directly. (If this does not work or causes a problem, you can create a shadow copy of Bill in LEAKS DB, but you do not need to).

2. **BillVersion (SELECT permissions).** You do not need to "reflect" the contents of this table in LEAKS DB, but if you believe that your code needs access to the name of the Bill or a bill description, feel free to join DDB2016Aug.Bill and
There is a table, the **State Legislature** separate DDDB2016Aug.Term, that the individual has ever Pid) DDDB whether with the could Note contributions table that you Legislator table maintain of candidates, You Legislator This (SELECT database. not from records into Do from to identify the Lobbyists parsing able data individual you table about contain those Lobbyist only As the table a with the the replace of the shall permissions). DB (SELECT a table maintain Registered of DB You the table. shall the Person only update LEAKS of DB and Lobbyist key in the of DB as value the Person record. If the key in the Person record is applicable), then use NULL as the value of the foreign key in the LEAKS Person table. If the DDDB2016Aug.Person contains zero matches with the LEAKS DB Person record on First Name, Last Name, Middle Name (and suffix - if applicable), then use NULL as the value of the foreign key in the LEAKS DB Person table. You shall only update the contents of the LEAKS DB Person table. You shall have no write permissions on the DDDB2016Aug.Person table.

**4. Lobbyist (SELECT permissions).** LEAKS DB shall maintain a table of Registered Lobbyists. The intent of the project is to eventually replace the DDDB2016Aug.Lobbyist database table with the table generated by one of the teams. As a result, the LEAKS DB Lobbyist table shall only contain the information about those individuals that you were able to identify as Lobbyists when parsing and analyzing the data from the CalAccess database. Do not copy records from DDDB2016Aug.Lobbyist into the LEAKS DB Lobbyist table.

**5. Legislator (SELECT permissions).** This table is for your information. You do not have to maintain a LEAKS DB Legislator table (you do need to maintain a list of candidates, so that you could associate contributions with them). Note that the Legislator table simply confirms whether a specific individual (identified by their DDBB Pid) has ever served in the State Legislature. There is a separate table, **DDDB2016Aug.Term**, that
contains information about the specific service of a Legislator during a specific legislative session (the Term table contains information about the Legislator's party affiliation, their district, and more). For now, we are not sharing the DDDB's Term table with you.

6. **Organizations (SELECT and INSERT permissions).** This is the only DDDB table which you are allowed to add records to. For the time being, you are given only **INSERT** permission on this table (but not UPDATE or DELETE - at the moment, we do not want you modifying/deleting existing records there, but we might give you these permissions for debugging purposes later). Because you have **INSERT** privileges on this table, LEAKS DB shall not maintain a separate table serving the same function - i.e., you shall effectively integrate the DDDB2016Aug.Organizations table into LEAKS DB. When working with this table, please know the following:

   a. **Ids.** The Primary key of the **DDDB2016Aug.Organizations** (despite the name of the table) uniquely identifies **one specific way in which an organization has been mentioned** in some context/data source of DDDB. The "specific way" here means the **exact spelling of the name of the organization.** As such, the same organization may be represented by multiple records in the DDDB2016Aug.Organizations table. For example, we can have different records for "ACLU", "The ACLU", "American Civil Liberties Union", "American civil liberties union", and "Am. Civ. Lib. Union", each with a separate unique primary key in the **DDDB2016Aug.Organizations** table, which actually represent the same underlying organization.

   b. **Matching.** When you discover a mention of an organization (Lobbying Firm, Lobbyist Employer, Donor, Recipient of expenditures, etc.) in any of the CalAccess forms, your first goal will be to determine if this particular mention exists in the **DDDB2016Aug.Organizations** table.

      i. **If you find a single exact match** between the name of the organization you discovered in the CalAccess data, and the value of the **DDDB2016Aug.Organizations.name**, you can use the value of the primary key (DDDB2016Aug.Organizations.oid) for that record as the id of the organization discovered in CalAccess.

      ii. **If you find multiple exact matches** between the name of the organization you discovered in the CalAccess data, and the value of the **DDDB2016Aug.Organizations.name**, you can use the values of DDDB2016Aug.Organizations.city, and DDDB2016Aug.Organizations.stateHeadquartered to try to disambiguate the records. If you are able to get an exact match on all three attributes (using the information extracted from CalAccess), use the identity of the record with which your CalAccess organization mention matched. If you were not able to establish a full match to any of the records in DDDB2016Aug.Organizations, then **create a new record in the DDDB2016Aug.Organizations** table and fill out its contents to the best of your ability.

      iii. **If you cannot find an exact match** between the name of the organization you discovered in the CalAccess data, and the value of the
DDDDB2016Aug.Organizations.name, then create a new record in the DDDDB2016Aug.Organizations table and fill out its contents to the best of your ability.

7. OrgConcept (SELECT permissions). This table, whenever possible "merges" different mentions of the same organization into a single "organization" (or "organization concept"). As discussed above, in the DDDDB2016Aug.Organizations table, records with the name attribute set to "ACLU", "The ACLU", "American Civil Liberties Union" and so on will have different values of the primary key DDDDB2016Aug.Organizations.oid. However, all these records represent one underlying organization. The DDDDB2016Aug.OrgConcept table provides for a single unique id (DDDB2016Aug.OrgConcept.oid, stored as a foreign key DDDDB2016Aug.Organizations.oc_id in the Organizations table) that would be associated with all of the mentions of the same organizations. As a result OrgConcept is essentially in a many-to-one relationship with Organizations.
   a. Establishing a relationship between a record in DDDDB2016Aug.Organizations, and a specific record in the DDDDB2016Aug.OrgConcept table is an arduous, and often manual process, and is outside of the scope of your project. Whenever you matched organization information from a form submitted to the SoS office to an existing record in the DDDDB2016Aug.Organizations table, you can use the value of the DDDDB2016Aug.Organizations.oc_id attribute in that record to determine the OrgConcept identity for the organization in question.
   b. LEAKS DB shall not clone this table.

DDDB Table Schemas. We are releasing a separate file that contains the CREATE TABLE statements for each of the seven tables outlined above.

When studying the table schemas for these tables, you will find several attributes with similar names present in all or almost all tables: lastTouched, lastTouched_ts, dr_id. These attributes are internal to the needs of applications supported by DDDB, and shall not be used in your work.