1.204 Computer Algorithms in Systems Engineering

Spring 2010

Problem Set 2: Municipal Database

Due: 12 noon, Wednesday, February 24, 2010

1. Problem statement

You are given a realistic, though fictitious, database drawn from town census, postal, telephone and other data for the town of Arlington, Massachusetts. This is in Arlington1204.zip, a zip file containing a Microsoft SQL Server database, Arlington1204.mdf. This database contains just a single table, Resident; it's really just a data file. The attributes are defined below.

1. **Build a fully normalized data model, using Visual Paradigm's entity-relationship (data) model diagram.**
    a. Identify the entity names, primary and foreign keys, relationships and their cardinality, and all attributes.
    b. Your model does not have to define the data types and it need not define whether nulls are allowed. (It's best to do these but, to keep the time commitment lower, they are not required.)
    c. All data elements in the Residents table are at an individual person level, though many should be represented at a household level. The major entities in the Residents table are persons, households, services and set of domain entities such as precincts, districts, etc. Examine the data from the Residents data file to infer the system rules so that you can create the data model.
    d. As you browse the Resident data file, resolve data inconsistencies arbitrarily and simply. For example, if one member of a household has cable TV (CATV) or fiber optic (FIOS) or gas service, the household has the service.
    e. You must decide how to model landline phones; there may be zero or more phones per household, and some may be associated with individuals within the household. Choose a simple approach.
    f. You are likely to have about 10 or 12 entities in your data model, with at least one many-to-many relationship. The other relationships will be many-to-one relationships.
    g. Hand in the .vpp file as part of the zip file you submit for the homework.

2. **Implement your data model in MS SQL Server, with all the data included in the Arlington1204 database except as noted below**.
    a. Build the database following the data model you create in the first step of the homework. Change the data model if it doesn't model the actual data properly. However, you may ignore inconsistencies and errors in the data, or you may choose a simple approach to resolve the errors.

b. Include the queries and temporary tables that you used to construct the database, even though you would normally delete them. Give each query a descriptive name, and indicate the order in which they were executed by starting their names with a, b, c, d, … or 1, 2, 3, 4, … Use SQL Server comments for this; they start with --.
c. If there are transformations required to implement your data model that are difficult (for example, involving parsing, or transformations that require programming), do not implement them. Choose a simple approach.
d. Create appropriate primary and foreign keys, and relationships in the data.
e. Hand in the electronic version of the database, in the zip file, including the .sql file with the queries that you wrote to implement it. Do not remove the original Residents table or any temporary tables you may have constructed, even though you normally would, to allow us to grade your assignment.

Hints:

a. You can create most of your tables through the SQL statement 'SELECT select_list INTO table_name FROM table_source WHERE search_condition. Please see the Murach SQL Server book, which is online, for how to do this. After the table is created, use the SQL Server MSE client to set its primary key and possibly make other changes interactively. See the instructor for help with this; we'll try to do a short demo in class as well. Remember that primary keys cannot have null values.
b. To create domain entities, use the SQL statement 'SELECT DISTINCT select_list INTO table_name FROM table_source WHERE search_condition. Again, set the primary key with the SQL Server MSE client.
c. When creating relationships between tables in SQL Server, make sure that the corresponding columns in both the tables have <u>exactly</u> the same data types. For example, they may/must both be nvarchar(20): they must have the same type and the same length. If you've used SELECT INTO statements to create your tables, you will have consistent data types. You need to be careful with any new attributes that you create.
d. For one or more tables in this homework, you will need to create a primary key that is not one of the columns present in the table. You will need to use the Identity option to create a primary key. Please see the Murach SQL Server book, which is online, for how to do this.
e. While zipping the database, you will need to stop the SQL Server so that the lock on the .mdf file is removed. The procedure for stopping SQL Server is described in the "HowTo: Install SQL Server" document. Briefly: use the SQL Server Configuration manager, right click on SQL Server Express and select 'Stop'. After copying, start SQL Server Express again.

3. Implement two interesting queries after you've built your database. You are free to implement any queries you wish. For example:

a. How many households have CATV, FIOS and gas service in all three years?
b. How many households dropped CATV (or FIOS or gas) service from any year to any other?

c. What is the average size of households with CATV service, by year?

Add the two queries at the end of the .sql file that you submit in part 2 above.

This is not a programming or data processing homework. The focus is on modeling issues and core database concepts (entities, attributes, primary keys, foreign keys and relationships), and their application to an actual problem. Don't spend a lot of time in mechanical manipulations; choose a simple approach, discard some offending data, or contact the instructor to discuss how to get around any data problems you're encountering in your approach.

Resident table:

Lname:	Last name, upper case
Fname:	First name, upper case
Yr:	Birth year (1800 if unknown)
Mo:	Birth month
Day:	Birth day
Midname:	Middle name, upper case
Suffix:	Name suffix (Jr, Sr, II, III, IV), upper case
Stnum:	Street number
Stlet:	Street letter (e.g., the 'A' in 12A)
Stname:	Street name, upper case
Apartment:	Apartment number or letters
Sex:	M or F
Occ:	Occupation as free form string
Dist:	School district (4-10), geographical area in which household is located
Phone:	Phone number (landline only; no mobile phones are in the data)
Pct:	Precinct (1-21), another geographical area in which household is located
DB1:	First year in database (0-9)
DBL:	Last year in database (0-9)
CATV1,2,3:	Cable TV subscriber (y) in years 3, 6, 9
Car_rte:	Postal carrier route (01-70)
Fios1, 2, 3:	Telephone fiber optic subscriber (y) in years 3, 6, 9
HH_num:	Household ID, unique identifier for members of same household. Unique street number, street letter, street name and apartment defines a household.
Ocd:	Occupation category (defined string)
Gas1,2,3:	Natural gas user (y) in years 3, 6, 9

SQL Code Aids

We provide two SQL examples that will help you find duplicates or children without parents.

1. Find duplicates. To find duplicate household IDs in a household table:

```
SELECT Household.ID FROM Household
WHERE (Household.ID In (SELECT ID FROM Household GROUP BY ID
HAVING Count(*)>1 ))
```

```
ORDER BY Household.ID;
```

Modify this for any table in which you wish to check the intended primary key for duplicates.

2. Find unmatched rows between a parent and child table. For example, find rows in Resident for which there is no row in the Household table (e.g., we have residents in household 733 but there is no row 733 in the household table):

```
SELECT Resident.ID
FROM Resident LEFT JOIN Household
ON Resident.ID = Household.ID
WHERE (Household.ID Is Null);
```

Modify this to find rows in any child table for which there is no corresponding row in the parent table.

Turn In

1. Place a comment with your full name, Stellar username, and assignment number at the beginning of the `.sql` and `.vpp` files in your solution.
2. Hand in the data model electronically on Stellar.
    a. You must use the Visual Paradigm drawing tool.
    b. Please make your model fit on one page if possible, no more than two pages otherwise.
    c. Please lay out your model so that it is clear: avoid relationship lines crossing each other, place related entities near each other, etc., as much as possible.
    d. The data model must include all entities, keys (primary and foreign), attributes, and relationships (including their types, such as one-to-many). Indicate keys explicitly for each entity. Include domain relations (validation entities). You do not need to be careful about null, unique or data type for attributes.
3. Hand in the database electronically on Stellar.
    a. Turn in the queries that you wrote to create the database, saved in the client as a .sql or .txt file. When you save your queries, please number them from 1 to n, Thus, when you write a SQL statement, please comment it as '--9 CATV table', for example. You don't need to write queries for tables you created from scratch using the client.
    b. Please zip the the database file (the .mdf file that SQL Server creates) and upload it on to the 1.204 course (Stellar) Web server.
4. Place all of the files in your solution in a single zip file.

5. Submit this single zip file on the 1.204 Web site under the appropriate problem set number. For directions see **How To: Submit Homework** on the 1.204 Web site.

6. Your solution is due at noon. Your uploaded files should have a timestamp of no later than noon on the due date.

Penalties

- 30 points off if you turn in your problem set after noon on Wednesday but before noon on Friday, February 26. You have two no-penalty two-day (48 hour) late submissions per term.
- No credit if you turn in your problem set after noon on Friday.

1.204 Computer Algorithms in Systems Engineering

Spring 2010