Problem Set 3: Relational Database Design

Overview

In this problem set you will:

- **Create an entity-relationship diagram for the 'urisa' database**
- **Construct a simple database design**

1. Entity-Relationship Diagram for the 'urisa' Database

Look over the schema for the 'urisa' database[*], and then draw an entity-relationship diagram for it. Your diagram should include the **authors, titles, keywords,** and **match** tables. Make sure to clearly indicate the primary and foreign keys and the cardinality relationships. You can draw your E-R diagram on paper or use a software package to assist you. If you choose to create your diagram electronically, make sure to save the output in a format that can be read easily *without the software that created the diagram*. An ideal choice is PDF, but image formats such as GIF, JPEG, and PNG are also acceptable provided the diagram is readable.

2. A Simple Database Design

Imagine you work for the public housing agency of a city, and you have been charged with keeping track of who is living in the agency's developments over time. To help you in this task, you have decided to use a relational database for your record keeping. Your task is to design a database that allows you to capture the facts described below:

- The city has three public housing developments. You want to record their names, locations, the year they opened, and their height in stories.
- For each unit in the development, you want to keep track of the number of bedrooms, the number of bathrooms, whether the unit has a kitchen or living room, and the square footage.
- The database should keep track of the households living in the units. For each member of a household, you want to record their name, date of birth, sex, and indicate whether or not they are they are the head of the household (more than one person can share that distinction).

---

[*] Kindly refer to the Assignment Section

- You also want to keep track of when a household moved into and out of a particular unit. You want to be able to follow households as they move from one unit to another or from one development to another. Think about how you will find the unit that the household is **currently** occupying (i.e., what query would you write to find the current unit of each household).

You will invent data for the three developments; two units in **each** development; and three families, one with 2 members, one with 3 members, and one with 4 members. Include records for each household making one move to another unit. You will insert these data in your database using INSERT statements.

You may be interested to know that this assignment was inspired by records kept by the Boston Housing Authority about their developments. Hence, this type of problem has definite real-world importance.

The Process

Follow the following process while designing your database:

- Consider the problem, identifying the entities involved, their attributes, and relationships among them.

- Draw an entity-relationship diagram that captures your thinking. Turn in your E-R diagram with your problem set. As in part 1, you may either create your diagram on paper or use software. The same caveats for electronic diagrams in part 1 apply here.

- Make sure that your schema adheres to **third normal form.**

- In a text editor, construct a script that implements a schema that represents your entities and relationships. Name this file ***username*_dbschema.sql,** where ***username*** is your Athena username. Record the output from running your SQL script in the file ***username*_dbschema_log.txt.** Follow these guidelines as you write your script:

    - Include the "SET ECHO ON" command at the start of the file. This will allow you to see your commands as they execute.

    - Use the SQL*Plus SPOOL command to create your log file automatically. Don't forget to use the 'SPOOL OFF' command at the end of the script.

    - Include DROP TABLE statements for all your tables. This will help if you have to run your script more than once, which will almost certainly be necessary. Include the clause "CASCADE CONSTRAINTS" at the end of your statement. This makes sure any foreign key constraints that refer to this table are dropped too. If these constraints are not dropped, Oracle will report an error when you try to drop the table. For example:

```
DROP TABLE parcels CASCADE CONSTRAINTS;
```

- Include CREATE TABLE statements for all your tables.
- Review the notes entitled "Enforcing Referential Integrity in Oracle." Then, write:
  - ALTER TABLE ADD CONSTRAINT statements to define the tables' primary keys.
  - ALTER TABLE ADD CONSTRAINT statements to define the tables' foreign keys.
- Include comments that indicate the meaning of your statements. SQL comments begin with two dashes together (--). Everything after the dashes is ignored **on that line only.** For example:

```
-- This text is a comment.
SELECT * FROM cat; -- The text after the dashes is
also a comment.
-- However, the SELECT statement above is not part
-- of the comment.
```

- Review your database design using the database design lecture notes[*]. Pay particular attention the database design rules of thumb. Are you breaking any of these rules? If so, why? Remember, breaking the rules is often OK, provided you have a good reason. If you make some choices you think that might raise some questions, make sure to address them in your comments.
- Run your script in SQL*Plus using the command '@dbschema.sql'. Correct any errors in your script and run it again. Keep trying until it works. Errors generated by attempts to drop tables that you haven't created yet are normal and can be ignored. Note that you can use the '@file.sql' syntax to run any SQL script. The **.sql** extension is assumed if you leave it out.
- Add some sample data to your tables. Create another script called ***username*_dbinsert.sql**. Record the output from running your SQL script in the file ***username*_dbinsert_log.txt.** Insert about five rows in each table using 'INSERT INTO table VALUES (...)' statements. For example, consider the table **mytable** created by this CREATE TABLE statement:

```
CREATE TABLE mytable (
    mycomment   VARCHAR2(10),
    mylowvalue  NUMBER,
    myhighvalue NUMBER,
    mydate      DATE);
```

You could insert two rows into **mytable** this way:

[*] Kindly refer to the Assignment Section

```
INSERT INTO mytable
    VALUES ('Hi Mom', 1, 5, '30-OCT-98');

INSERT INTO mytable (mycomment, mydate)
    VALUES ('Hi Dad', '1-NOV-98');
```

Note that Oracle is fussy about date values and will, by default, accept dates only in the format 'DD-MON-YY' **within single quotes** as shown above. You may use the TO_DATE function to use a different format in a particular instance, for example:

```
INSERT INTO mytable (mycomment, mydate)
    VALUES ('Bye Mom', TO_DATE('11/10/1998',
'MM/DD/YYYY'));
```

Alternatively, you can use the 'ALTER SESSION SET NLS_DATE_FORMAT' command to change the format for a particular session, as shown below:

```
ALTER SESSION SET NLS_DATE_FORMAT = 'MM/DD/YYYY';
```

After executing the statement above, the following statement should work:

```
INSERT INTO mytable (mycomment, mydate)
    VALUES ('Bye Dad', '2/15/2002');
```

Refer to the *Oracle8i SQL Reference* for the details of TO_DATE, ALTER SESSION SET NLS_DATE_FORMAT, and date format models. Be forewarned: dabbling in date format models has many potential gotchas (most notably confusing **MM** (months) with **MI** (minutes)). Read carefully!

- Remember to use a COMMIT statement after inserting rows to make sure the data is saved permanently. A SQL error may cause an automatic ROLLBACK of your changes. (You can use the ROLLBACK statement on purpose if you want to undo changes since the last COMMIT.) Many statements include implied COMMITs, and not all errors cause a ROLLBACK. Query your tables often to make sure that your SQL statements are having the desired effect!

- Test **all** your primary and foreign key constraints with some INSERT statements that are **intended to fail.** Try to insert a duplicate primary key. Also try to insert a foreign key that does not exist as a primary key in the referenced table. *For every constraint you create, you should have an INSERT statement that tests the constraint.* Record these statements in the text file ***username*_dbtest.sql**. Record the output from running your SQL script in the file ***username*_dbtest_log.txt.**

- **Extra credit:** Write a series of statements that records two households moving from one unit to another, with one taking over the unit once held by the other. At least transaction control statement such as COMMIT or ROLLBACK is **required.**

3. What to Turn In

The breakdown of point values for this assignment is as follows:

- Entity relationship diagram for the 'urisa' database: 15 points
- Entity relationship diagram for the housing database: 25 points
- Schema definition file and log file: 30 points
- Data insert file and log file: 10 points
- Constraint test file and log file: 20 points
- Extra credit: 5 extra credit points

Turn in your ***username*_dbschema.sql, *username*_dbschema_log.txt, *username*_dbinsert.sql, *username*_dbinsert_log.txt, *username*_dbtest.sql**, ***username*_dbtest_log.txt and two entity relationship diagrams** files.
