CIT 483/583 HW05 – ActiveRecord
Fall 2016

Instructions

This lab requires two files to be submitted, which will be packaged into a single zip file and submitted to the HW05 drop-box as per the submission instructions at the end of the document.

Summary

As discussed in the lecture, ActiveRecord is a library for Object-Relational Mapping. It is generally considered part of Ruby on Rails, but it does not depend upon any other Rails component.
SQLite is an embedded lightweight general-purpose SQL database that can exist in plain file format.
The Ruby ActiveRecord and SQLite3 gems are installed and ready to use on students.cs.nku.edu.

For this assignment you will create a small database to store information on hosts, System Administrators, and their managers.

Before starting on the assignment, copy, study, and run the examples at the top level of my home directory on students.cs.nku.edu: authors.sql and authors.rb. Much of the code comes directly from the Active Record lecture slides, and don't forget that there links to required reading you should consult before starting and use as a reference while working on the assignment.

Create a subdirectory called authors under your CIT483 directory structure and cd into it. Copy ~mccordt/authors.* to your own work directory.
To create an authors database, type:

```
$ sqlite3 authors.db < authors.sql
```

It will produce an error on the last line, which is intentional: `Error: near line 69: FOREIGN KEY constraint failed`
The database schema itself will be created, but open the authors.sql file to see what caused the problem. It does no harm per se; it simply failed to create an invalid record, which is what we want. You should now see a new file called authors.db.

Next, invoke the Ruby script authors.rb:

```
$ ruby authors.rb
Wilde, Oscar,
Blake, William,
Dickens, Charles,
Austen, Jane,
Mary, Shelley,
The Picture of Dorian Gray, 1789
Songs of Innocence, 1890
Pride & Prejudice, 1813
Oliver Twist, 1837
David Cooperfield, 1850
Frankenstein: or, The Modern Prometheus, 1818
```

Looking at the script, we see that it creates objects of type Author, Book, etc., corresponding to the examples used in the lecture.

Not all techniques needed for this assignment are demonstrated in the sample .sql or .rb files given, e.g., using ActiveRecord to create tables and add indexes, but these techniques are discussed in the lecture slides and can be found online in SQLite and ActiveRecord documentation.

If something were to become corrupted in the SQLite DB, one can delete the .db file at any point and repopulate it. You might also try direct SQL queries in SQLite as you manipulate objects in Ruby. In other words, you can interact with your DB from either interface, or both in parallel. Assuming authors.db has been loaded with data, it can be queried just as any SQL database can. You may also practice with SQL insert, update, and delete statements at will. Metadata commands have a slightly different format, as in *.schema*.

```
$ sqlite3 authors.db
SQLite version 3.11.0 2016-02-15 17:29:24
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite> select * from authors;
1|Oscar|Wilde
2|William|Blake
3|Charles|Dickens
4|Jane|Austen
5|Shelley|Mary
sqlite> .schema Books
CREATE TABLE BOOKS(
  id INTEGER  PRIMARY  KEY  AUTOINCREMENT  NOT  NULL,
  title varchar2(255) NOT  NULL,
  year varchar2(4)
);
```

Project Requirements

Each of the two scripts required will have specific and separate responsibilities.

1. **hw05.sql** – This script should use SQL as input to sqlite3 to create an SQLite database stored in a file called hw05.db.

    a. It should create three tables with columns as specified below. All columns should be specified as not nullable and the types are given. To test it, use sqlite3:

    ```
    $ sqlite3 hw05.db < hw05.sql
    ```

HOSTS	ADMINS	HOSTS_ADMINS

id: integer (autoincrement)	*id: integer (autoincrement)*	*id: integer (autoincrement)*
host_name: varchar2(255)	*first_name: varchar2(255)*	*host_id: integer (FK to HOSTS.ID)*
ip_address: varchar2(255)	*last_name: varchar2(255)*	*admin_id: integer (FK to ADMIN.ID)*
domain_name: varchar2(255)	*shift: varchar2(255)*	
	telephone: varchar2(255)	

b. Indexes must also be created on the following columns to enforce uniqueness. When two columns are listed, you should create a composite index containing both columns.

- **HOSTS_ADMINS** (*host_id, admin_id*)

c. Foreign key constraints must be created on the following columns:

- **HOSTS_ADMINS.host_id references HOSTS.id**
- **HOSTS_ADMINS.admin_id references ADMINS.id**

d. Create at least two records for each of ADMINS and HOSTS, and four for HOSTS_ADMINS using SQL insert statements. Use any values you like, but try to use realistic values, such as IP addresses with four octets, etc. For HOSTS_ADMINS, make that sure that at least one ADMIN is associated with two HOSTS or vice versa.

REMEMBER that DB table names are plural while ActiveRecord classes are singular.

2. **hw05.rb**–This script should establish the connection to hw05.db and create two additional tables and populate all tables with additional data.

a. Add two more tables using the ActiveRecord create_table method.

MANAGERS	**MANAGERS_ADMINS**
id: integer (autoincrement)	*id: integer (autoincrement)*
first_name: varchar2(255)	*manager_id: integer*
last_name: varchar2(255)	*admin_id: integer*
telephone: varchar2(255)	

b. Create a unique composite index on MANAGERS_ADMINS using ActiveRecord:

- **MANAGERS_ADMINS** (*manager_id, admin_id*)

c. Create ActiveRecord objects for HOST, ADMIN, and MANAGER that express the many-to-many relationships between HOSTS and ADMINS as well as the one-to-many relationship between MANAGERS and ADMINS.

d. Use the definition of an ActiveRecord MANAGER_ADMIN object in ActiveRecord to create the implied FK constraints below:

- **MANAGERS_ADMINS.manager_id references MANAGERS.id**
- **MANAGERS_ADMINS.admin_id references ADMINS.id**

e. Use the definition of an ActiveRecord HOST_ADMIN object in ActiveRecord to enforce the FK constraint that is explicit in the SQLite schema, but which cannot be directly enforced in the sqlite3 Ruby gem.

f. Add at least two more records for each of ADMINS, HOSTS, and HOSTS_ADMINS using ActiveRecord objects.

g. Add exactly two MANAGER records and four MANAGERS_ADMINS records using ActiveRecord objects. [Since you have four ADMINS, we assume each must have a manager.]

h. Use ActiveRecord to produce a list of all HOSTS, then ADMINS, then MANAGERS.

i. Use ActiveRecord to find the ADMIN(s) that are associated with any one of your HOSTS. [If you know how to use a two-way or three-way SQL join, try it using the ActiveRecord execute method. If not, just use the data in HOSTS_ADMINS find all ADMINs associated with a HOST and query based on their ID numbers. See the end of the authors.rb file for a pure Ruby approach. See the Addendum at the end of the document for JOIN hints.]

j. Use ActiveRecord to generate three ad hoc queries on HOSTS, ADMINS, or MANAGERs, such as 'show me HOSTS with IP addresses that contain "10"' or "show me ADMINS whose last name is Smith," and so on. Any queries will suffice, but they have to be different from one another.

Additional things to keep in mind re: ActiveRecord:

1. ActiveRecord will automatically add a primary key column that auto-increments (called "id") to each table. You don't need to specify this for the tables created with ActiveRecord. Nor do you have to specify an id on insert.
2. The table names above are intentionally chosen to conform to ActiveRecord conventions, so ***do not change them***.
3. SQL varchar2 types are simply strings in ActiveRecord.

CIT 483 Extra Credit/583 Additional Requirements

1. Add exception handling for failed inserts due to Unique key constraint violations. Test it in the Ruby code by trying to insert a duplicate record into HOSTS_ADMIN or MANAGERS_ADMINS. (+5)

2. Find a way, either in the SQLite schema, or the ActiveRecord Ruby code, to make sure that telephone numbers contain exactly 10 digits. Don't worry about area codes in parentheses or dashes--just that it has 10 digits. (+5)

For 583 students, these 10 points will replace the customary 10 points allocated to simply submitting the correct files in the correct format. For 483 students, they will be added to the rubric score as extra points.

Submission Instructions

Zip both files, hw05.sql and hw05.rb, into an archive called hw05.zip and **submit only that file** to the HW05 dropbox. Do not add anything to the file name. [Blackboard appends information to the file name that includes your username, which I will use to tell which submission is which.]

ADDENDUM: Three-way JOIN Example

Suppose I have more than one book by Dickens, and I want to show all books written by him using SQL:

```
select a.first_name, a.last_name, b.title
  from authors a, books b, authors_books ab
  where a.id = ab.author_id and b.id = ab.book_id
  and a.last_name = 'Dickens';

Charles|Dickens|Oliver Twist
Charles|Dickens|David Copperfield
```