Asgn 6 TransFile & LogFile Notes

*NOTE 1: I've used the term "Transaction" (Trans file, TranCode) thus far in CS3310 (and here in this Asgn) in a **data**-file context. It has a somewhat different meaning in a **DBS context** - i.e., a "Transaction" is an ordered set of operations (usually all SQL `insert/delete/update` statements) that are considered as a SINGLE UNIT, where ALL the operations have to work successfully in order for the 1 transaction to succeed . If ANY ONE of the ops fails to work, all earlier ops in the transaction are ROLLED BACK to UNDO their effects on the DB.*

NOTE 2: Use caution when transferring or viewing/saving the transaction file so that you don't introduce extra <CR><LF>'s (or <LF>'s for Linux) in the middle of a single transaction. Some of the S transactions long –but I only put a single <CR><LF> at the END of the whole statement - though wrap-around makes it appear as if there are internal ones. "Save Link As…" from the course website should be fine and not introduce extraneous <CR><LF>'s. But if you email it or ftp it or…, then, depending on the client program you're using,it might insert insert extra ones. If you're having problems, check it using a HexEditor.

NOTE 3: When should there be a ; at the end of an sql statement?
The sql string/command:
- *in a C# PROGRAM does NOT have a ; at the end (see my sample program)*
- *in a SCRIPT file to be run in mysql's command window DOES have a ; at the end*
- *entered MANUALLY at the `mysql>` prompt DOES have a ; at the end.*

The lines in the transaction DATA FILE do NOT have a ; at the end – they're just DATA strings, and not SQL commands, per se.

NOTE 4: If you mistakenly delete too much data, you can restore a table's original data by doing the following in MySQL (for the Country table, for example):

```
DELETE FROM Country;
```
(removes ALL DATA from table, keeping table itself)

```
SOURCE c:/.../InsertCountryData.sql
```
(specifying YOUR path instead)

(NOTE: This is the same SOURCE statement as in the WorldDriver.sql file)

NOTE 5: PUT A BLANK LINE in the LogFile between each transaction

TranCode is the 1st column of the line – one of these: S, I, D, U (always caps).

*Use a **ONE method for each** of the 4 types of Transactions (so FOUR methods) – regardless of whether there are different formats e.g., different types of INSERT's or 0/1/MANY rows returned for SELECT's. [UNLIKE WHAT MY DEMO PROGRAM SHOWS]*

For S transactions (SELECT, meaning RETRIEVE data from the DB)

- The transaction data **IS an actual SQL statement** to be used "as is".
  [This is not a common/proper programming approach, but time is short…].
- Allow for 0 or 1 or MANY ROWS to be returned to the program from the DBS.
- The "table" that you print to the LogFile does **NOT NEED TO**:
  - be in a box (e.g., like a typical interactive result in the command window)
  - have column headings (but if it does, DON'T use my demo program ones)
  - be perfectly aligned since this **SINGLE GENERIC METHOD** doesn't know What data type `rdr[0]` or `rdr[1]` or the other columns are.

  [Given more time for this asgn, then yes, the program could gather the necessary column names & column data types from the DB itself to be able to create a "nice" output report].
- The number of columns there'll be for the result set is: **`rdr.FieldCount`**

For I transactions (INSERT)

- The transaction data is **NOT an SQL statement** - your program has to construct it using various string-handling methods (e.g., Split, +, etc.)
  [See sample C# code which does part of this: StringHandling.cs].
- 2 basic formats for INSERT SQL statements that your program needs to build:
  1) **all-column** INSERT (so column names are NOT specified in the sql statement):
     ```
     INSERT INTO CountryLanguage VALUES ('USA','C#','F',0.01)
     ```
     where the transaction file data line (i.e., the parameters) would look like:
     ```
     I CountryLanguage:'USA','C#','F',0.01
     ```
  2) **some-columns** INSERT (so column names MUST BE specified in the sql stmt):
     ```
     INSERT INTO Country(Code, Name) VALUES ('HEX','Hexland')
     ```
     where the transaction file data line (i.e., the parameters) would look like:
     ```
     I Country:(Code, Name):'HEX','Hexland'
     ```

-Respond (in the LogFile) with the reassurance message: **`OK, Data INSERTED`**

For D transactions (DELETE)

- The transaction data is **NOT an SQL statement** - your program has to construct it.
- The basic format for a simple DELETE SQL statement is:
  ```
  DELETE FROM Country WHERE Name = 'Disneyland'
  ```
  where the transaction file data line (i.e., the parameters) would look like:
  ```
  D Country:Name:'Disneyland'
  ```

-Respond (in the LogFile) with the reassurance message: **`OK, Data DELETED`**

For U transactions (UPDATE)

- transaction data is the actual SQL statement to be used as is

-Respond (in the LogFile) with the reassurance message: **`OK, Data UPDATED`**