The above only defines the syntax, but doesn't say what these statements mean or do (known as the semantics). For that, we just use English:

● "create" "table" <table name> "(" <column name>,
+
")" ";" Creates an empty table with the given name (replacing it if it is already loaded). The names of its columns are given by the column names in order. There must not be any duplicate column names.

● "create" "table" <table name> "as" <select clause> ";" Creates a table with the given name (replacing it if it is already loaded) whose columns and contents are those produced by the select clause. Each distinct set of column values gets only one row in the table (no duplicate rows).

● "load" <name> ";" Load data from the file name.db to create a table named name.

● "store" <table name> ";" Store the data from the table table name into the file table name.db.

● "insert" "into" <table name> "values" ("(" <literal>,
+
")")
,
+
";" Adds new rows to the given table whose values are given by the parenthesized lists of literals. In each parenthesized list, there must be exactly one literal for each column of the table, and the table must already exist. Rows are not added if they duplicate a row already in the table.

● "print" <table name> ";" Print all rows of the table with the given name (which must be loaded). The rows are printed one per line and indented. Separate columns with blanks, and print the columns in the order they were specified when the table was created. You do not need to print the column names. See the example below for the format. Print rows in lexicographic order: that is, print them in order by their first column (comparing the contents as strings). If two rows have the same first column, print them in order by their second columns, and so forth. (Warning: in standard SQL, there is no such ordering by default).

● <select clause> ";" Select clauses are described below. They represent tables created from other tables. When used alone as a statement (terminated by a semicolon), they indicate that the resulting table is to be printed, using the format described above for the printcommand.

● "quit" ";" Exit the program.

● "exit" ";" Synonym for quit.

Select clauses. Select clauses are used in select statements and in create statements. They denote tables whose information is selected from other tables.

● "select" <column name>,
+
"from" <table name> <condition clause> A new (unnamed) table consisting of the named columns from the given table from all rows that satisfy the condition clause.

● "select" <column name>,
+
"from" <table name> "," <table name> <condition clause> A new (unnamed) table consisting of the given columns from all rows of the natural inner join of the two tables that satisfy the condition clause. A natural inner join is a table whose rows are formed by combining pairs of rows, one from the first table and one from the second, such that any columns that have the same name in both tables also

have the same value in both rows. Any pairs of rows that don't have the same values in common columns are not included in the joined table. If there are no common columns between the two tables, all row combinations are included in the natural inner join.

Condition clauses. An empty condition clause does not restrict the rows that appear in a select. Otherwise, it contains one or more conditions separated by and, all of which must be satisfied. The tests compare column values against either other columns or literal values. The relation symbols mean what they do in Java, except that all values are treated as strings (use the compareTo method on Strings to compare them). Thus you can write things like

```
Lastname >= 'Chan'
```

to get all rows in which the value of the Lastname column comes after "Chan" in lexicographic order (i.e, by their first character, then their second, etc., ordering characters according to Java's collating sequence). The result is roughly like alphabetical order, with the major difference being that all digits come before all capital letters, which come before all lower-case letters, with punctuation and other characters inserted in various places.

A .db file starts with a line containing all column names (at least one) separated by commas, with any leading or trailing whitespace removed (see the .split and .trim methods of the String class). Column names must be valid identifiers and must be distinct. This is followed by any number of lines (zero of more), one for each row, containing the values for that row, separated by commas (again, leading or trailing whitespace is removed). For example, the students table would look like this:

```
SID,Lastname,Firstname,SemEnter,YearEnter,Major
```