Working with DB2 on CDF

In order to work with db2 on cdf you must set up the db2 environment. To do this, you must set the environment variable DB2INSTANCE to ”db2instX”. You can either do it every time you log in or add this to you .cshrc file:

```
setenv DB2INSTANCE db2instX
```

Each of you have been assigned a database ”c343XXXXXX” where XXXXXX is your cdf account. This database is accessible to you only and you should use this database to write and test your assignments. Also, a course database has been set up ”csc343s” to which everyone can connect.

You can run db2 in two modes: system level and interactive.

In system level mode you can run db2 commands from unix prompt directly:

For example:

```
%db2 "connect to c343XXXXXX"
```

where `%` is your unix prompt. Note, you can use either single quotes or double quotes.

In interactive mode you run db2 by executing

```
%db2
```

Your prompt changes to `db2=>`. From there, you can execute the commands the same way you would in system level mode. For example

```
db2=>connect to c343XXXXXX
```

You can also run your db2 commands in a script and dump the output into a file.

```
db2 -f input_file -r output_file -l history_file
```

db2 will execute all commands in `input_file`, placing the output result in `output_file` and all system messages will be written into `history_file`.

For more information on working with db2 on cdf see:

http://www.cs.toronto.edu/db/courses/db2/

Working with Databases

- **Connecting to a database**

  To connect to a particular database use the following command:

  ```
  connect to c343XXXXXX
  ```

- **Creating tables**

  To create a table `course` with attributes `name`, `studentnum`, `mark` use the following command:

  ```
  create table course (name char(20), studentnum integer, mark decimal(2,1))
  ```

  The following example illustrates how to specify constraints and assign primary keys in the created table:

  ```
  create table course (name char(20) not null, \
                       studentnum integer, \
                       mark decimal(2,1), \
                       primary key(name))
  ```

  The primary key of a relational table uniquely identifies each record in the table. Note that the primary key must have the `not null` attribute which ensures that the name inserted has a value.

- **Populating tables**

  To insert tuples into the table use the following command:

  ```
  insert into course values ("John", 123, 89.5), \
                            ("Mary", 456, 94)
  ```

- **Removing tables**

  To remove a table from database use the following command:

  ```
  drop table course
  ```

- **Listing tables**

  To list all tables in your database use:

  ```
  list tables
  ```

Simple Queries and Updates

- **select**

  The following command will select all columns from table `course`

  ```
  select * from course
  ```

  To select particular columns use:

  ```
  select name, mark from course
  ```

To put a constraint on selected rows use `where` statement

```
select name, mark from course where mark > 70
```

- **update**

  The following command will raise all the mark by 10.

  ```
  update course set mark = mark + 10
  ```

Bulk Inserts

- You can insert values into a table using data from another table:

  Create a table which will contain only the student names

  ```
  create table studentnames (name char(20))
  ```

  Insert names from `course` table into the `studentnames` table

  ```
  insert into studentnames (select name from course)
  ```

- You can use `import` command to insert data from an ASCII file:

  ```
  import from data.txt of DEL messages msg.txt insert into course
  ```

  Where,

  - `data.txt` is the name of ASCI file.

  - `DEL` refers to a delimited ASCI file, i.e. fields are separated by commas.

  - `msg.txt` is the file where system messages will be written to. It is optional, so you can eliminate it.

  Our data file may look like this:

  ```
  John,123,99.9

  Mary,456,82.3

  Steven,789,67.7

  Karen,123243,76.9
  ```