Northeastern University
CS3200 – Database Design
Spring 2018, Derbinsky

Create a Database from Scratch

The focus of this activity is to have you practice schema creation (`CREATE`), data manipulation (`INSERT`, `UPDATE`, `DELETE`), and predict the output of a query (`SELECT`). It is written for MariaDB (via phpMyAdmin), but with slight modification you can also use SQLite. This activity is **not** graded, and there is nothing to turn in; **however**, you will find the skills useful for your project and the content is fair game for exams.

First, open phpMyAdmin and click the "SQL" tab[1]. We will first create a new database, named "simple":

```
CREATE DATABASE simple;
```

You should now see a database named "simple" in the listing of databases on the left side of the page – click it. When you click one of these links, you are telling phpMyAdmin to "use" a database as the context for future SQL commands. To switch between databases in SQL, issue the `USE` command:

```
USE Chinook;
USE simple;
```

We are now going to generate the following database schema and state:

alpha

a	b
x	1
y	2
z	3

beta

c	d	e
x	i	3.14
y	ii	2.7

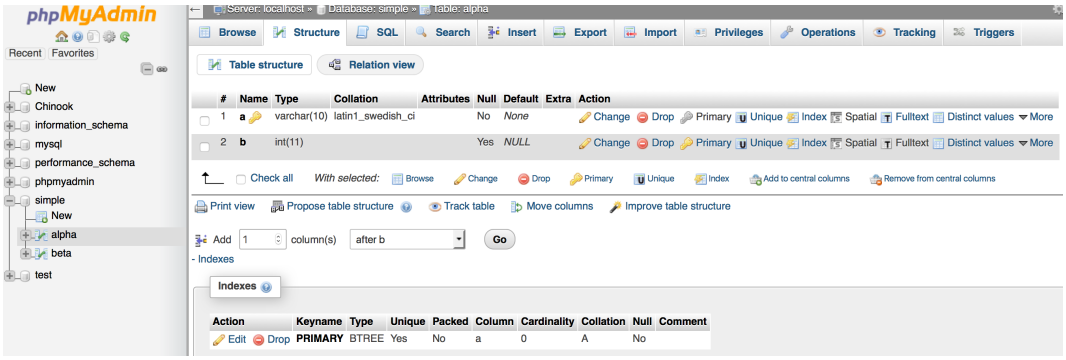
To begin, let's create the tables:

```
CREATE TABLE alpha
(
   a VARCHAR(10) PRIMARY KEY,
   b INT
);

CREATE TABLE beta
(
   c VARCHAR(10),
   d VARCHAR(10),
   e REAL,
   PRIMARY KEY (c,d)
);
```

[1] In this lab we will do as much as possible via SQL. Feel free to use the equivalent GUI features in your own work.

You should now see these tables beneath "simple" on the left (you might have to refresh). You can click them, and click the "Structure" tab to verify the fields – look to the "Indexes" section to verify the primary key (also visible via the key icon next to the "a" field).



Now to populate the tables! In MariaDB, you can do so via individual statements that add one row at a time, such as ...

```
INSERT INTO alpha (a, b) VALUES ('x', 1);
INSERT INTO beta (c, d, e) VALUES ('x', 'i', 3.14);
```

or you can group all inserts [per table] into a single statement:

```
INSERT INTO alpha (a, b) VALUES ('x', 1), ('y', 2), ('z', 3);
INSERT INTO beta (c, d, e) VALUES ('x', 'i', 3.14), ('y', 'ii', 2.7);
```

If you make a mistake, simply issue an update:

```
UPDATE alpha SET b=2 WHERE a='y';
```

or delete superfluous rows:

```
DELETE FROM beta WHERE c NOT IN ('x', 'y');
```

Exercise

For each of the following queries, predict what exactly the output will be (column names & order, row contents):

```
SELECT * FROM alpha INNER JOIN beta;
SELECT * FROM alpha INNER JOIN beta ON alpha.a=beta.c;
SELECT * FROM alpha LEFT JOIN beta ON alpha.a=beta.c;
SELECT * FROM beta LEFT JOIN alpha ON alpha.a=beta.c;
SELECT * FROM alpha LEFT JOIN beta ON alpha.a=beta.c WHERE alpha.b>=2;
SELECT AVG(alpha.b) AS avg_b FROM alpha LEFT JOIN beta ON alpha.a=beta.c
       WHERE beta.d IN ('i', 'ii', 'iii') OR beta.c IS NULL;
```

Before continuing with the lab, verify your predictions by executing each query. Make sure you understand each result before moving on.

Now delete `(x, i, 3.14)` in `beta` and add a new row with `(w, -, 1.732)` and repeat this exercise, predicting and verifying each query above.

Let's now try to cause trouble! First, try to add a row that violates the primary key constraint:

```
INSERT INTO alpha (a, b) VALUES ('x', 7);
```

Notice the error that is raised – make sure you understand this w.r.t. (with respect to) the definition of a primary key.

Let's try another mistake:

```
ALTER TABLE beta
 ADD CONSTRAINT c_fk
 FOREIGN KEY(c)
 REFERENCES alpha(a);
```

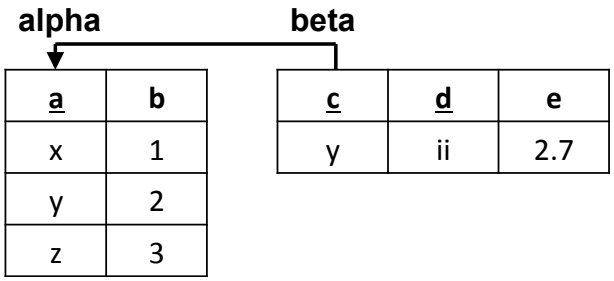
You should notice that there is an offending row – delete this row and retry the `ALTER` command. Once the constraint is added, you can view it in the Structure tab by clicking "Relation view" (near the top). Notice that the response for DELETE/UPDATE operations are defined as "restrict" – while other options exist (as mentioned in class), this simply prevents offending operations with an error.

Now, let's try to make a mistake in the opposite direction:

```
DELETE FROM alpha WHERE a='y';
```

You should be able to describe why this error occurs, as well as a method to alter the schema of `alpha` to allow the deletion to take place.

At this point we have the following schema & state:



For practice, let's start from scratch, create the schema in one fell swoop, and add all the data. First, get rid of the tables:

```
DROP TABLE alpha;
DROP TABLE beta;
```

Oops, that didn't work – why not? Try this instead:

```
DROP TABLE beta;
DROP TABLE alpha;
```

Can you think of another way you could have achieved this outcome?

Now create our enhanced schema:

```
CREATE TABLE alpha
(
   a VARCHAR(10) PRIMARY KEY,
   b INT
);

CREATE TABLE beta
(
   c VARCHAR(10),
   d VARCHAR(10),
   e REAL,
   PRIMARY KEY (c,d),
   CONSTRAINT c_fk FOREIGN KEY (c) REFERENCES alpha(a)
);
```

And populate the rows:

```
INSERT INTO alpha (a, b) VALUES ('x', 1), ('y', 2), ('z', 3);
INSERT INTO beta (c, d, e) VALUES ('y', 'ii', 2.7);
```

Congratulations – you have completed this activity. Finally, let's clean up by deleting the database:

```
DROP DATABASE simple;
```