# ECE356 – Database Systems

# Lab 2: SQL DDL

**Mohammad Hamdaqa**
**Lab Instructor**
**Winter 2016**



Special thanks for Tiuley Alguindigue, Prof. Wojciech Golab and Prof. Paul Ward for their help in preparing the slides

# Goals

Regardless of database size and complexity, each database is comprised of *tables*. One of the first steps in creating a database is to create the tables that will store the data. In this lab you will practice how to:

1. Create tables, store data in these tables and ensure that the final set of tables are in Boyce-Codd Normal Form (BCNF).
2. Define a relationship between rows in one table to rows in another table by using the concept of *Foreign Keys*, and define different types of *integrity constraints* for database tables in order to maintain the correctness of the data that are stored.
3. Insert data into tables, generating primary key values with autoincrement.
4. Create and query views.

# Marking Scheme

- This lab consists of five parts:
  - Part 1: Adding tables to your database (20 %)
  - Part 2: Adding referential integrity constrains to your database (20 %)
  - Part 3: Adding data to tables using autoincrement to generate keys (20 %)
  - Part 4: Creating and querying a view (20%)
  - Part 5: Add referential integrity to YELP database (20%)

**Notes**:

1. You must save your work and retain a digital copy until the end of the term.
2. You must demonstrate your solution to a TA or lab instructor at the end of the lab.
3. For parts 1-4, provide your solutions in files part1.sql, part2.sql,part3.sql and part4.sql, after modifying these files with the required statements. You will need to execute these modified scripts during your demo.

# Part 1
# Adding tables to your database

# Part1.sql description (1/2)

- The script provided (**part1.sql**) contains SQL statements to create a table (called **SupplyData**) and add some data to this table.
  - This table represents the supplies needed for a project.
  - A project may use many supplies, and a supply may be used by many projects.
- You will notice that the table **SupplyData** is not normalized.
- A subset (canonical cover) of the functional dependencies is given here:

```
F = {
        sup_id --> type_id, sup_description, unit_description,
             cost_per_unit
        type_id --> type_description
    }
```

# Part1.sql description (2/2)

- The script also shows how you can perform a lossless-join BCNF decomposition with respect to F to find the relations (tables) that will replace the table SupplyData in your database.
- It suffices to decompose table SupplyData into two smaller tables:

A. **SupplyType**
F1 = { typeID --> typeDescription }
B. **Supply**
F2 = { SupplyID --> typeID, supplyDescription,
unitDescription,costPerunit }

- Run the script given in part1.sql.
  - This will create the table SupplyData (not normalized), and add some data to this table.
  - It will also create tables SupplyType and Supply in BCNF form.

# Adding tables to your database

- You are required to modify (part1.sql) script to do the following:
  - Initialize the two new tables (i.e., **SupplyType** and **Supply**) with the data stored in the **SupplyData** table (use INSERT INTO .. (SELECT .. FROM SupplyData)).
  - Create a table named **ProjectSupply**, to represent the link between *projects* and *supply items*. This table will contain a column to keep track of the number of units of a particular supply used by a project.
  - Insert some data associating some existing projects with existing supplies into the table **ProjectSupply**.
    (ie. projectID 123 uses 10 Laptop computers (supplyID 103),
    projectID 345 uses 2 Vacums (supplyID 104),
    projectID 345 uses 15 Colour Stickers (supplyID 101))

# Part 2
# Adding primary key and referential integrity constraints to your database

# Add PRIMARY KEY and FOREIGN KEY constraints to the tables in your database

- The provided script, part2.sql, contains the CREATE statements for all tables created so far. Your task is to
  - Modify these CREATE statements, or use ALTER statements to add the necessary constrains.
  - Example statements are provided in the part2.sql script.
    - Foreign key dependencies may forbid some of your actions. Make sure you have the correct order to drop or create tables.
- MySQL 5.0.95 (the MySQL version installed in the eceweb server) provides referential integrity checking for tables using the InnoDB engine.

**Note**:
- There are many storage engines in MySQL like: MyISAM, InnoDB, MERGE and MEMORY
- Different MySQL versions may have different default storage engine. For example, InnoDB engine is the default engine for tables created with MySQL 5.5, while MyISAM is the default for MySQL 5.0.
- You can set the default storage engine at anytime using the command
  SET storage_engine= <engine name>;
  to enable InnoDB replace <engine name> by InnoDB
- Default storage engine has to be set to InnoDB *before* the table is created otherwise storage engine can be set during table creation as follows:
  CREATE TABLE table_name (...) ENGINE InnoDB;

# Add PRIMARY KEY and FOREIGN KEY constraints to the tables in your database

- When you add referential integrity constraints to a table, the database engine creates the necessary indexes for primary and foreign key columns. You do not have to create the indexes explicitly (with CREATE INDEX).
  - To verify this, use the command "show index from *tablename*;"
- Indexing improves the speed of data retrieval operations on a database table at the cost of additional writes and storage space.
- Indexing will be covered later in the course.

**Note**:
- Relations with attributes being referenced should be created first. But when deleting tables, it should be in the opposite sequence.

# Part 3
# Insert data using autoincrement

# Reinsert data into all the tables

- **Practice using the MySQL auto increment feature when inserting data into a table.**
  - Modify the definition of the SupplyType table, adding the attribute AUTO INCREMENT to the primary key of the table, typeID.
  - Add data to the table and verify that each row is assigned a typeID automatically (the values for typeID will be 1,2,3,.. , etc).
  - Note that the INSERT statements will not specify a value for the auto-increment column since this value is assigned automatically by the database engine.

# Part 4
# Creating and querying a view

# Create and Query a View

- **In this part you are required to modify** part4.sql **to create and query a view. To do that:**
  1. Use the CREATE VIEW SQL statement to create a view called **EmpDepView**, that joins employee with department, and hides the employee salary.

  2. Query the created view using a SELECT statement to:
     - Display all rows in the table
     - Display rows for the marketing department
     - Display the employee name, the department and the project description for all projects in which this employee works

**Note**:
For an **example** of how to create a view and perform a query with a view, see provided script **part4.sql**

# Part 5
# Modify Yelp database

# Modify Yelp Database

- Use the Scripts you created in Lab1 –Part 2 to create and import the rest of the tables in the Yelp data set
  - Note in the last lab we focused on bulk_insert_buffer_size, this only works for the MyISAM engine and not InnoDB.
  - In this Lab we focus on referential integrity, hence you should modify your database engine to InnoDB
- Add referential integrity constrains the Yelp database
- Submit your answer of this part using LEARN dropbox by Tuesday Feb 3rd.