# CSE 4020/5260
## Database Systems

Instructor: Fitzroy Nembhard, Ph.D.

## Week 6-7

**DDL**

FLORIDA TECH
FLORIDA'S STEM UNIVERSITY®

# Distribution

- All slides included in this class are for the exclusive use of students and instructors associated with Database Systems (CSE 4020/5260) at the Florida Institute of Technology
- Redistribution of the slides is not permitted without the written consent of the author.

# Structured Query Language (SQL)

- Data Definition Language
- Domains
- Integrity Constraints

# Relational Schemas for a University

*Classroom (building, room-number, capacity)*

*Department (dept-name, building, budget)*

*Course (course-id, title, dept-name, credits)*

*Instructor (ID, name, depart-name, salary)*

*Section (course-id, sec-id, semester, year, building, room-number, time-slot-id)*

*Teaches (ID, course-id, sec-id, semester, year)*

*Student (ID, name, dept-name, tot-cred)*

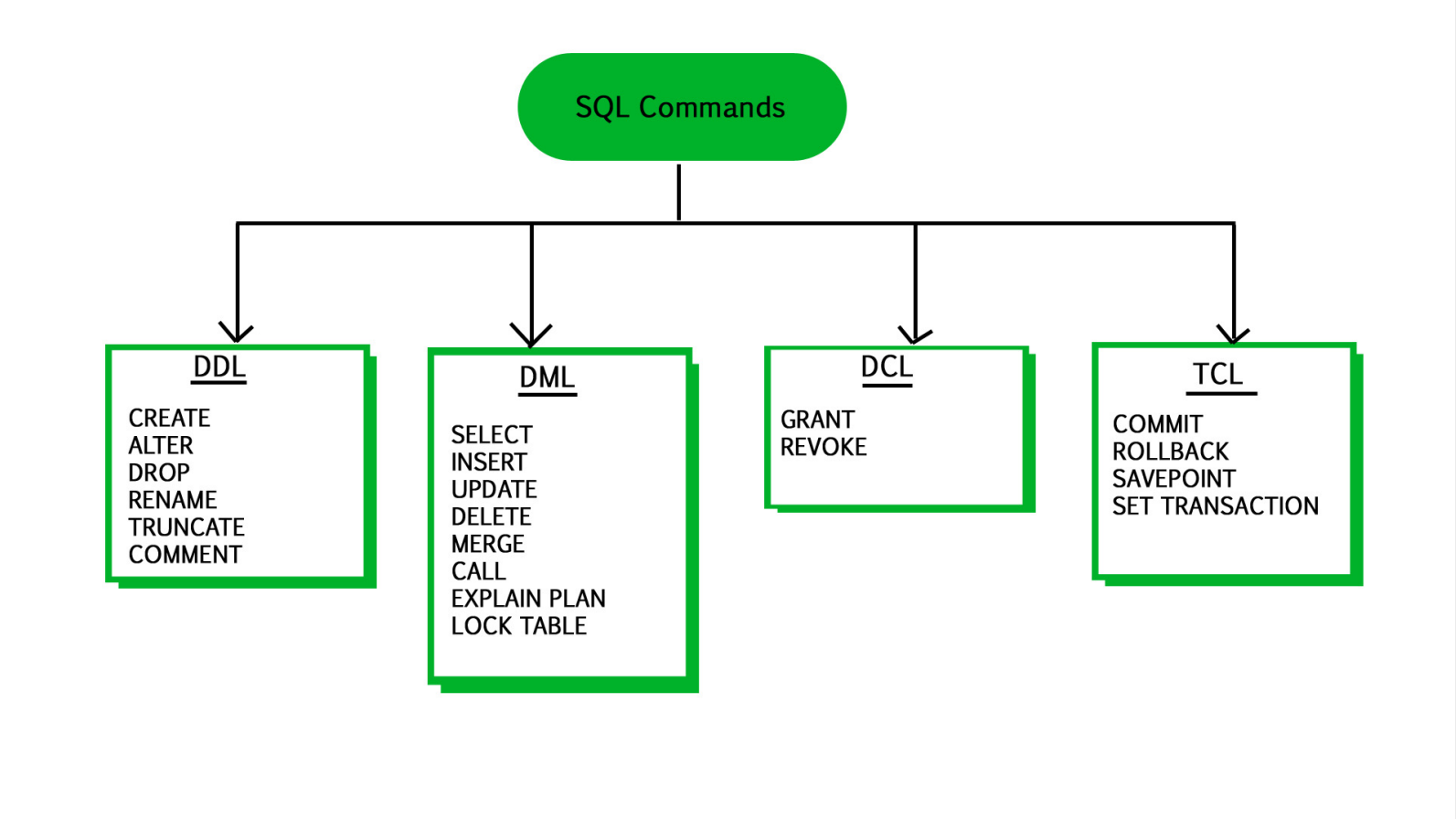
*Takes (ID, course-id, sec-id, semester, year, grade)*

*Advisor (s-ID, i-ID)*

*Time-slot (time-slot-id, day, start-time, end-time)*

*Prereq (course-id, prereq-id)*

# SQL Commands




Source: https://www.geeksforgeeks.org/sql-ddl-dml-tcl-dcl/

# Data Definition Language (DDL)

- DDL allows the specification of a set of tables.

- For each table, a DDL statement specifies:
  - A name for the table
  - A name for each attribute
  - The domain (i.e., a type) of values associated with each attribute
  - Integrity constraints
  - An associated set of indices
  - Security and authorization information
  - The physical storage structure for the relation

# Domain Types in SQL

- Basic SQL Types:
  - *varchar(n)* - Variable length character string, maximum length *n*.
  - *char(n)* - Fixed length character string, with length *n*.
  - *int* - Integer (machine-dependent).
  - *smallint* - Small integer (machine-dependent).
  - *bigint* - Big integer (machine-dependent).
  - *real* - Floating point numbers machine-dependent precision.
  - *double precision* - Floating point numbers machine-dependent precision.
  - *float(n)* - Floating point number, precision of at least *n* digits.
  - *numeric(p, d)* - Fixed point number; *p* digits of precision and *d* digits to the right of decimal point.
  - plus others…

  - You may also create your own datatypes based on primitive types:
  - **create type** Dollars **as numeric**(12,2) final;

# Date/Time Types in SQL (Cont.)

- More complex (object) types:
  - *date* - Dates, containing a year, month and date
  - *time* - Time of day, in hours, minutes and seconds
  - *timestamp* - Date plus time of day
  - *interval* - Period of time (not in MySQL; See Informix for example)

- Operations on complex types: (typical)
  - Interval values can be added/subtracted to or from a date/time/timestamp value
  - Values of individual fields can be extracted from date/time/timestamp:

    **extract** (**year from** student.birth-date)

- Unstructured types:
  - *Text*
  - *BLOB*
  - *CLOB*
  - *Image*
  - *geometry*, etc.

**The INTERVAL data types include year to month and day to second intervals.**

| Interval Format | Example | Explanation |
|---|---|---|
| INTERVAL YEAR TO MONTH | 123-04 | An interval of 123 years, 4 months |
| INTERVAL DAY TO SECOND(3) | 7 6:54:32.123 | An interval of 7 days, 6 hours, 54 minutes, 32 seconds and 123 thousandths of a second |

# Create Table Construct

- A table is defined using the **create table** command:

```
create table r (A1 D1,
                A2 D2,
                ...,
                An Dn,
                      (integrity-constraint1),
                 ...,
                      (integrity-constraintk))
```

  - $r$ - name of the table
  - $A_i$ - column name
  - $D_i$ - column data type

- Example:

```
create table student(
   ID varchar(5),
   name varchar(20),
   dept_name varchar(20),
   tot_cred  numeric(3,0)
   );
```

# Integrity Constraints in Create Table

- Integrity constraints:
  - **not null**
  - **primary key** ($A_1$, ..., $A_n$) - - *Also enforces* **not null**
  - **check** *(P),* where *P* is a predicate

- Example:

```
create table student(
    ID varchar(5),
    name varchar(20),
    dept_name varchar(20),
    tot_cred numeric(3,0) check (tot_cred >= 0),
    primary key (ID)
    );
```

# Referential Integrity in SQL

- Key types:
  - **primary key** - enforces uniqueness.
  - **unique key** - also enforces uniqueness, aka, *alternate* or *secondary* key.
  - **foreign key** – enforces referential integrity.

- A foreign key references the primary key of the referenced table:

  **foreign key** (*account-number*) **references** *account*

- Reference columns can be explicitly specified:

  **foreign key** (*account-number*) **references** *account*(*account-number*)

- Foreign key references have several implications for insertions, deletions and modifications…

# Data Storage Clause in Oracle

```
CREATE TABLE divisions
    (div_no     NUMBER(2),
     div_name   VARCHAR2(14),
     location   VARCHAR2(13) )
     STORAGE  ( INITIAL 100K  NEXT 50K
                MINEXTENTS 1  MAXEXTENTS 50  PCTINCREASE 5);
```

# DDL Files

- A DDL file typically contains a collection of:
  - **create table** statements
  - **create index** statements
  - statements that create and/or specify other things:
    - Security and authority information
    - Physical storage details

- A DDL file can be coded by hand or generated by a schema design or modeling tool.

# Referential Integrity in SQL – Example

```
create database university; -- Remove this line if the database already exists
                            -- You may also use create database if not exists university;
use university;
create table classroom      -- or create table if not exists classroom
    (building               varchar(15),
     room_number            varchar(7),
     capacity               numeric(4,0),
     primary key (building, room_number)
    );

create table department
    (dept_name              varchar(20),
     building               varchar(15),
     budget                 numeric(12,2) check (budget > 0),
     primary key (dept_name)
    );

create table course
    (course_id              varchar(8),
     title                  varchar(50),
     dept_name              varchar(20),
     credits                numeric(2,0) check (credits > 0),
     primary key (course_id),
     foreign key (dept_name) references department (dept_name)
      on delete set null
    );
```

# Cascading Actions in SQL

- A foreign key reference can be enhanced to prevent insertion, deletion, and update errors.

```
create table account (
    . . .
    foreign key(branch-name) references branch
                                        on delete cascade
                                        on update cascade
    . . . )
```

- If a delete of a tuple in *branch* results in a referential-integrity constraint violation, the delete “cascades” to the *account* relation.

- Cascading updates are similar.

# The University Schema DDL (Cont'd)

```
create table instructor
(ID varchar (5),
 name varchar (20) not null,
 dept_name varchar (20),
 salary numeric (8,2) check (salary > 29000),
 primary key (ID),
 foreign key (dept_name) references department
           on delete set null);

create table section
(course_id varchar (8),
 sec_id varchar (8),
 semester varchar (6) check (semester in
                 ('Fall', 'Winter', 'Spring', 'Summer')),
 year numeric (4,0) check (year > 1701 and year < 2100),
 building varchar (15),
 room_number varchar (7),
 time_slotid varchar (4),
 primary key (course_id, sec_id, semester, year),
 foreign key (course_id) references course
          on delete cascade,
 foreign key (building, room_number) references classroom
          on delete set null);
```

FLORIDA TECH

# The University Schema DDL (Cont'd)

```
create table teaches
 (ID varchar(5),
  course_id varchar(8),
  sec_id varchar(8),
  semester varchar(6),
  year numeric(4,0),
  primary key(ID, course_id, sec_id, semester, year),
  foreign key(course_id, sec_id, semester, year) references section
  on delete cascade,
  foreign key(ID) references instructor
  on delete cascade);

create table student
(ID varchar(5),
name varchar(20) not null,
dept_name varchar(20),
tot_cred numeric(3,0) check(tot_cred >= 0),
primary key (ID),
foreign key (dept_name) references department
on delete set null);
```

# The University Schema DDL (Cont'd)

```
create table takes
 (ID varchar(5),
  course_id varchar(8),
  sec_id varchar(8),
  semester varchar(6),
  year numeric(4,0),
  grade varchar(2),
  primary key(ID, course-id, sec_id, semester, year),
  foreign key(course_id, sec_id, semester, year) references section
              on delete cascade,
  foreign key(ID) references student
              on delete cascade);

create table advisor
 (s_ID varchar(5),
  i_ID varchar(5),
  primary key (s_ID),
  foreign key(i_ID) references instructor (ID)
               on delete set null,
  foreign key(s_ID) references student (ID)
              on delete cascade);
```

# The University Schema DDL (Cont'd)

```
create table prereq
  (course_id varchar(8),
   prereq_id varchar(8),
   primary key(course_id, prereq_id),
   foreign key(course_id) references course
                on delete cascade,
   foreign key(prereq_id) references course);

create table timeslot
  (time_slot_id varchar (4),
   day varchar(1) check (day in ('M', 'T', 'W', 'R', 'F', 'S', 'U')),
   start_time time,
   end_time time,
   primary key(time_slot_id, day, start_time));
```

# Creating Tables in MySQL Using a DDL File

- After installing MySQL, you may load the contents from your SQL file as follows:

```
mysql -u <username> -p <DBName> < your_ddl_file.sql
```

- If your DDL file creates a database, you may leave off the DBName as follows:

```
mysql -u <username> -p < your_ddl_file.sql
```

- You may also use the following commands:

```
source your_ddl_file.sql
mysql < your_ddl_file.sql
mysql db_name < your_ddl_file.sql
```

# Drop and Alter Table Constructs

- **drop table -** deletes all information about a table.

  **drop table** customer

- **alter table** - used to add to or delete attributes from an existing relation.

  **alter table** *r* **add** *A D* *-- Attribute A and domain D*

  **alter table** *r* **drop** *A* *-- Attribute A*

- More generally, the alter table command can be used to modify an existing table in many ways, such as adding indexes, changing permissions, storage properties, etc.

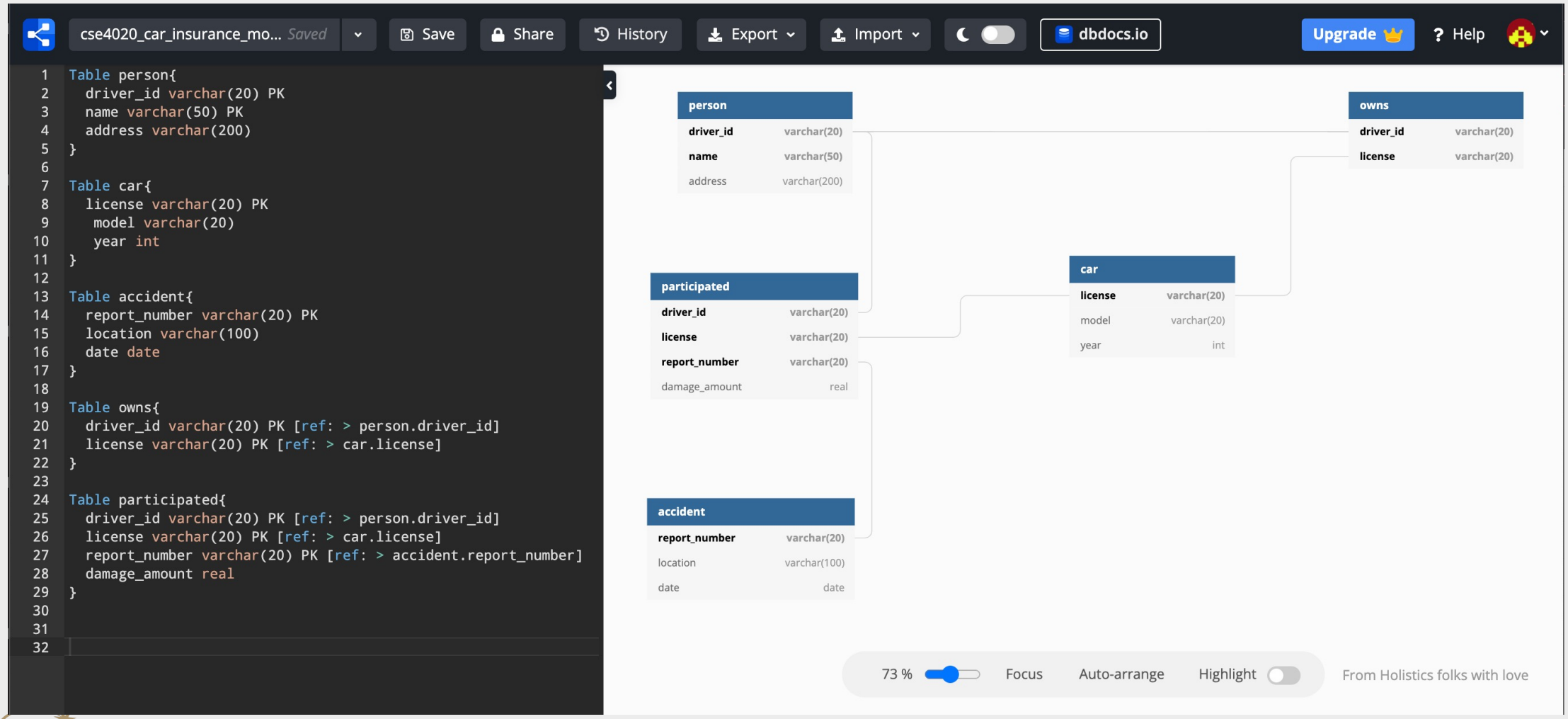
  ***alter table*** *pollsters* ***add unique*** *unique_index(user, email, address)* *-- add a 3-part unique key*

# The Drop Syntax

- **DROP OBJ_TYPE [ IF EXISTS ] OBJ_NAME**
- **OBJ_TYPES**
  - Table
  - Column
  - Triggers
  - Views
  - Constraints
  - Database
- **OBJ_NAME** –the object name that you want to drop.

# Creating Schema Diagrams in DBDIAGRAM.IO

- We Create a Schema by Typing the DDL statements in the Editor



# Creating Schema Diagrams in DBDIAGRAM.IO

- DBDiagram Reference (Relationship) Syntax

**Long form:**
Ref name-optional { table1.field1 < table2.field2 }

**Short form:**
Ref name-optional: t1.f1 < t2.f2

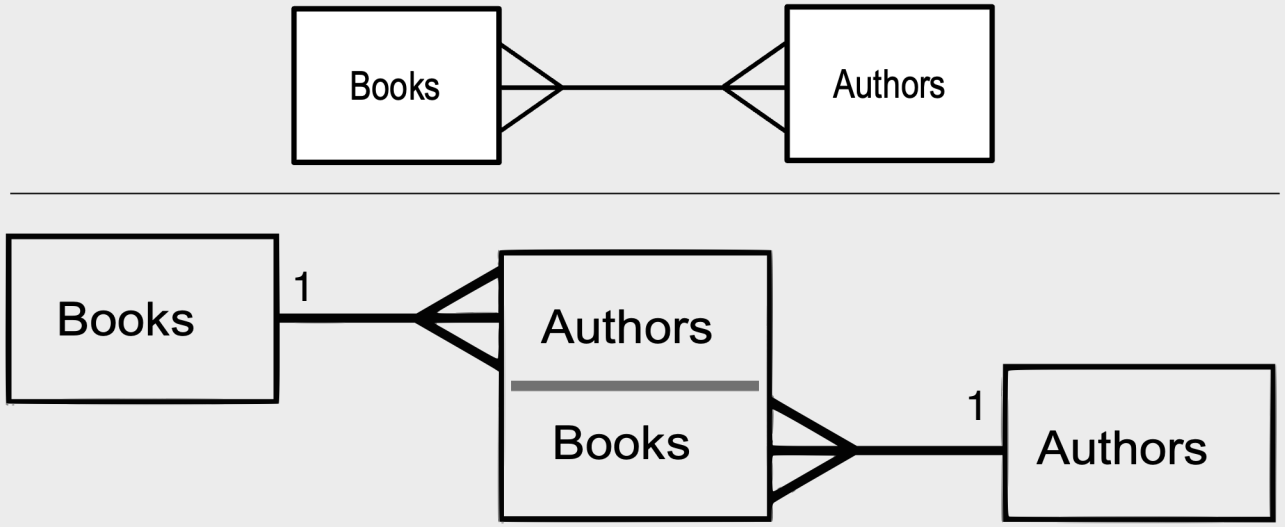
**Inline form:**
Table posts { id int [pk, ref: < comments.post_id]
user_id int [ref: > users.id] }

**Ref type:**
< : One-to-many
> : Many-to-one
- : One-to-one

➢ Learn more here: https://www.dbml.org/docs/

# Modeling Many-to-Many Relationships DBDIAGRAM.IO

- We need to create an associative (join) table instead of drawing the relationship directly.




```
// DBDiagram Specification

Table authors {
author_id int
name varchar
dob datetime
gender varchar
}

Table books {
book_id int
release_date datetime
title varchar
}

Table author_book {
author_id int [ref: > authors.author_id]
book_id int [ref: > books.book_id]
}
```

# Generating DDL File From DBDIAGRAM.IO

- Follow the in-class demo to create a schema. Then click Export → Export to MySQL to export the DDL File