This is a topic we need to discuss more than once. As you write more complex queries, you need to understand the table create statements and the column constraints. At this point in the class I expect you to be able to read the create table statement and understand the implication of the table definitions.

A table is a container for data. We can informally think of a table as a collection of rows of data- that is what we see when we use the Select * from Tbl query.

We can also think of a table (or each of its rows) as a collection of columns; each column has to be defined as to the type of data it can store. There may be some additional rules that have to be defined at the table level. These rules are called **constraints**.

One thing you need to understand about MySQL is that it does not enforce the type of checks called a check constraint. I have added check constraints to some of these tables because you should see them and because the data in the tables is following those rules. MySQL enforces the not null, null, unique and the primary and foreign key constraints; someday I hope it will also enforce check constraints.

This will look at a few of the tables in the books series of tables.

This is the create table statement for the authors table. I have added numbers to make this easier to talk about.

```
1.    create table a_bkinfo.authors (
2.        author_name_first varchar(20)      null
3.      , author_name_last  varchar(20)      not null
4.      , author_id         char (5)         not null
5.      , constraint bk_authors_pk           primary key(author_id)
6.    )engine = INNODB;
```

Line 1- provides a name for the table; this table is named authors and the table is in the a_bkinfo schema.

Line 2,3, and 4 define the columns/attributes. Each column gets a name. The author_name_first and the author_name_last columns are defined as varchar(20) type which means that they store strings with a maximum length of 20 characters. The author_id column is defined as a char(5) which means that it will always store exactly 5 characters.

The author_name_first has a NULL constraint which means that this column can be left empty; an author is not required to have a first name for our table. The other two columns have a NOT NULL constraint- these columns must have a value stored in each row.

Line 5- this defines a constraint which says that the author_id attribute is the primary key for the table. We can have only one row in the table for any given value for the author_id attribute. We identify an author by the author_id attribute; we do not identify authors by name.

Line 6 - this specifies a file storage pattern which corresponds to what we normally think of as relational tables.

Consequences of this table definition: If we write a query that uses just the authors table or an inner join to the authors table, we do not need a filter that the author_name_last is not null or that the author_id is not null. Those rules are defined by the table.

This is the create for the Order Headers table.

```
7.    create  table   a_bkorders.order_headers (
8.        order_id          integer          not null
9.      , order_date        date             not null
10.     , cust_id           integer          not null
11.     , constraint bk_orders_pk            primary key (order_id)
12.     , constraint bk_orders_cust_fk       foreign key(cust_id)
13.                  references a_bkorders.customers(cust_id)
14.     , constraint bk_order_id_range       check (order_id > 100)
15.     ,  constraint bk_order_date_ck       check (order_date >=  '2000-01-01')
16.   )engine = INNODB;
```

Line 9- Note that the order_date column is a date attribute, not a datetime attribute. These values have no time component.

Lines 14, 15 set constraints on the columns; the order _id - it has to be a value greater than 100 and the order date has to be Jan 1 2000 or later.

Lines 12 and 13 define a new type of constraint- a foreign key. The table design says that for any value we want to enter for the customer id, we must already have a value in the customers table for that customer id value. Also the cust_id attribute cannot be null in this table. Therefore we cannot have a order which is not associated with an actual customer.

Consequences of this table definition: If we write a query that does an inner join between customer and order headers, the order_date will not be null, the cust_id will not be null ,the cust_name_last will not be null, but the cust_name_first might be null (check the customer table). Those rules are defined by the tables. Do not write filters in your queries that check something that the table definitions requires.
But if we do the following outer join, the result set can have nulls for the attributes in the order headers table since we are asking for customer with and without orders.

```
From a_bkorders.customers  CS
Left join a_bkorders.order_headers OH on CS.cust_id = OH.cust_id
```

You should not write this join. This is asking for orders with and without customers and the table definitions do not allow orders with no customers; for our tables, this should be written as an inner join.

```
From a_bkorders.order_headers OH
Left join a_bkorders.customers  CS on CS.cust_id = OH.cust_id
```

Another table

```
17.   create table a_bkinfo.books (
18.       book_id           integer          not null
19.     , title             varchar(75)      not null
20.     , publ_id           integer          null
21.     , year_publd        integer          not null
22.     , isbn              varchar(17)      null
23.     , page_count        integer          null
24.     , list_price        numeric(6,2)     null
25.     , constraint bk_books_pk             primary key (book_id)
26.     , constraint bk_books_publ_fk        foreign key(publ_id)
27.                  references a_bkinfo.publishers  (publ_id)
28.     , constraint book_id_range           check (book_id > 1000)
29.     , constraint bk_page_count_ck        check (page_count >= 0)
30.     , constraint bk_price_ck             check (list_price >= 0)
31.     , constraint bk_books_year_ck        check (year_publd >= 1850)
32.   )engine = INNODB;
```

Line 18 - this says that the book_id values are stored as integers. They are whole numbers like 1005, not values with a decimal component- we do not have a book id value of 1235.56

Line 24- the list price is a numeric (6,2) This means that we can store numbers with 2 digits after the decimal point and 4 digits before the decimal point. We can have list prices such as 29.95, 450.00, 3.20. That data type would allow a negative value but we have a constraint in Line 30 that says that the list_price values must be 0 or more- a book can be free but it cannot have a negative price.

Line 20 and 26-27 define a publ_id attribute. This attribute can be null but if we have a value for publ_id then it must be a publisher that we have in the publisher table. We have decided that it is ok to sell a book when we do not have a publisher.

Consequences of this table definition: If we write a query that does an inner join between books and publishers, the result set might have a null for the publ_id and publ_name for a legitimate book. You would need to take more care figuring out what an outer joins means for this pair of tables.

Since attributes such as book_id and list_price are defined as numbers, you should compare them to numbers, not to strings. Do *not* write Where book_id = '1005 or list_price > '50'. The correct way to write these is Where book_id = 1005 or list_pirce > 50. There is a difference between a number and a string.

One more table

```
33.    create table a_bkinfo.topics (
34.       topic_id           varchar(5)      not null
35.     , topic_descr        varchar(25)     not null
36.     , constraint bk_topics_pk            primary key(topic_id)
37.     , constraint bk_topic_descr_un       unique (topic_descr)
38.    )engine = INNODB;
```

Line 37- this has a new constraint unique. This means that we cannot have two rows in the topics table with the same value for the topic description. The topic_id is the primary key and that automatically makes that column unique. But we want to avoid a situation where we might have two rows with the same description. The topics table contains a row

(CMP, Computer Science)

We do not want to allow another row with the values shown here and the unique (topic_descr) constraint handles that.

(CS, Computer Science)

Take some time now to get familiar with these tables. You will be using them for most of the rest f the assignments. The best way to do this is to read the create statements and then write some simple queries for these tables. Write the join to find books and their authors. Write the query to find customers with no books and to find books that have never been ordered. You should be able to write these types of queries easily now.

These tables are not huge but we have almost 100 books and more than 400 order detail rows. This is done to discourage you from reading the tables to check your queries. You can write some filtered queries to get partial results from the tables that can help you find errors.