Introduction to MySQL /MariaDB and SQL Basics

Read Chapter 3!
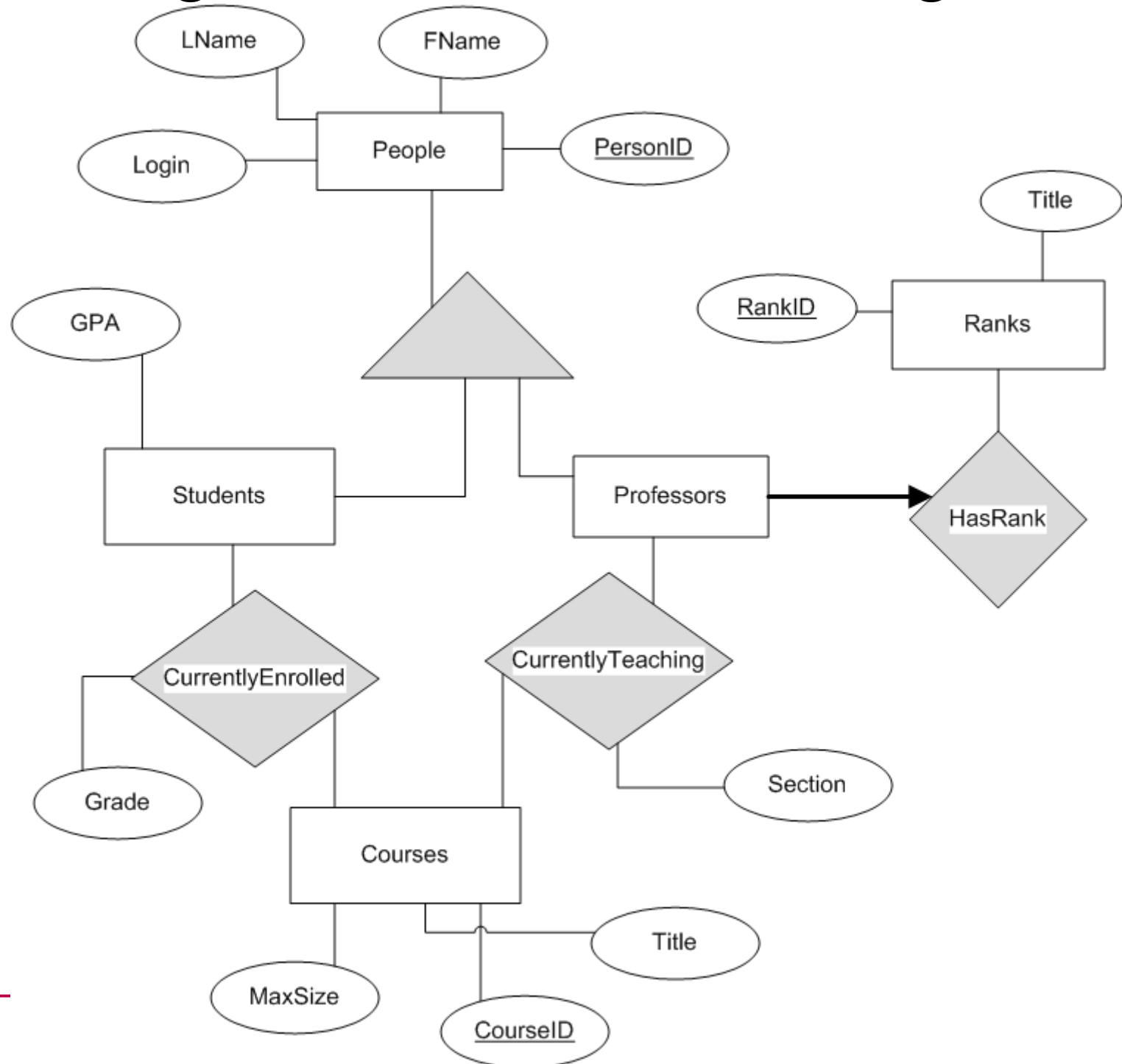

http://dev.mysql.com/doc/refman/

https://mariadb.com/kb/en/the-mariadb-library/documentation/

MySQL / MariaDB

College Database E-R Diagram

Database tasks

https://mariadb.com/kb/en/the-mariadb-library/documentation

- start MariaDB

  – setup user passwords

- shutdown MariaDB

- create database

- create table

  – primary key

  – index

  – foreign key

- insert data

  – source a file

- delete data

  – drop

- query data

  – where

  – join

  – group

  – order

  – subquery

*Coding standards on the class web page!

MariaDB

- ssh to db.cs.pacificu.edu (64.59.233.245)

  ssh -X db.cs.pacificu.edu

  [you@db ~]# $ mysql -u PUNetID -p

  mysql> set password = PASSWORD('NEWPASSWORD');

  mysql> show databases;

  mysql> use PUNetID_test;

Data types

- TINYINT/SMALLINT/INT/BIGINT SIGNED/UNSIGNED

- BIT

- FLOAT/DOUBLE

- BOOLEAN

- CHAR / BINARY

- VARCHAR(###)   /   VARBINARY(###)

- DATE / TIME / DATETIME / TIMESTAMP

- [TINY|MEDIUM|LONG]TEXT

- [TINY|MEDIUM|LONG]BLOB

- ENUM

CS445
Pacific University

M

- "M indicates the **maximum display width for integer types**. The maximum display width is 255. Display width is unrelated to the range of values a type can contain, … .

- For floating-point and fixed-point types, M is the **total number of digits** that can be stored."

- https://dev.mysql.com/doc/refman/5.7/en/numeric-type-overview.html

Strings

- ## Character Set

  – set of characters (ASCII, UTF8, ...

- ## Collation

  – the order of the characters within the set

```
MariaDB [will4614_test]> show variables like 'character_set_server';
+----------------------+-------+
| Variable_name        | Value |
+----------------------+-------+
| character_set_server | utf8  |
+----------------------+-------+
1 row in set (0.00 sec)

MariaDB [will4614_test]> show variables like 'collation_server';
+------------------+-----------------+
| Variable_name    | Value           |
+------------------+-----------------+
| collation_server | utf8_general_ci |
+------------------+-----------------+
1 row in set (0.01 sec)
```

 https://mariadb.com/kb/en/mariadb/supported-character-sets-and-collations/
   https://mariadb.com/kb/en/mariadb/character-set-and-collation-overview/

Create Table

```
CREATE TABLE People (

    PersonID INT NOT NULL AUTO_INCREMENT,


    FName VARCHAR(50),


    LName VARCHAR(50),


    Login VARCHAR(20) NOT NULL,


    CONSTRAINT People_PersonID_PK
        PRIMARY KEY (PersonID),

    CONSTRAINT People_Login_U UNIQUE (Login)

) Engine=InnoDB CHARACTER SET = utf8 COLLATE = utf8_bin;
```

Create a Table

```
mysql> show tables;

mysql> show create table People;
```

Insert One Row

```
INSERT INTO People ( FName, LName, Login)

VALUES ( "Chadd", "Williams","chadd");
```

```
INSERT INTO tablename ( ColumnName, …) VALUES ( data,
…);
```

Insert

```
INSERT INTO People ( FName, LName, Login)
VALUES ( "Chadd", "Williams","chadd");

INSERT INTO People ( FName, LName, Login)
VALUES ( "Doug", "Ryan","ryand");

INSERT INTO People ( FName, LName, Login)
VALUES ("Shereen", "Khoja","shereen");

INSERT INTO People ( FName, LName, Login)
VALUES ("Chris", "Lane", "lanec");

mysql> SELECT * FROM People;
```

Retrieve Data

```
SELECT *
FROM People
WHERE PersonID > 2;



SELECT column, …


FROM table


WHERE condition ;
```

MySQL

```
mysql> SELECT * FROM People WHERE PersonID > 2;

mysql> SELECT * FROM People WHERE LName = "Ryan";

mysql> SELECT * FROM People WHERE FName like "C%";

mysql> SELECT * FROM People WHERE FName like "c%";

mysql> SELECT FName, LName FROM People
    WHERE PersonID > 1;
```

Create a table

- Create another table

```
CREATE TABLE Professors (
        ProfID INT NOT NULL,

        Rank ENUM ('Assistant', 'Associate',
          'Full', 'Emeritus') NOT NULL,

        CONSTRAINT Professors_ProfID_PK
          PRIMARY KEY (ProfID)

) Engine=InnoDB;
```

Constraints

```
mysql> ALTER TABLE Professors
 ADD CONSTRAINT Professors_ProfID_FK
 FOREIGN KEY (ProfID) REFERENCES
 People(PersonID) ON DELETE CASCADE;
```

MySQL

- Insert some data

INSERT INTO Professors (ProfID, Rank) VALUES (1, 'Associate'); -- chadd

INSERT INTO Professors (ProfID, Rank) VALUES (2, 'Full'); -- doug

INSERT INTO Professors (ProfID, Rank) VALUES (3, 'Associate'); -- shereen

INSERT INTO Professors (ProfID, Rank) VALUES (4, 'The Boss'); -- chris

Let's make this go faster

- Load data from a SQL script
  This file is full of **INSERT and CREATE** statements.

  mysql> source /tmp/CreateDatabase.sql;

  Let's look at that file.

  db$ cat /tmp/CreateDatabase.sql | less

Deleting Data

- Let's delete some data

```
mysql> SELECT * FROM People;

mysql> SELECT * FROM CurrentlyTeaching;

mysql> DELETE FROM People WHERE PersonID=1;

mysql> SELECT * FROM People;

mysql> SELECT * FROM CurrentlyTeaching;

mysql> SHOW TABLES;

mysql> DROP TABLE People;

mysql> source /tmp/CreateDatabase.sql;
```

dbeaver

- Let's use a GUI to make this easier

  ./dbeaver

- alias

Queries

- What Courses have a MaxSize of greater than 5?

mysql> **SELECT ***
        FROM Courses
        WHERE
        MaxSize > 5;

Order By

- Let's sort the output

```
mysql> SELECT *
         FROM Courses
         ORDER BY MaxSize;


mysql> SELECT *
         FROM Courses
         ORDER BY MaxSize DESC ;



mysql> SELECT *
         FROM People
         ORDER BY LName, FName;
```

- Aggregate selected rows

  mysql> SELECT LName
  **   FROM People ;**

  mysql> SELECT LName, COUNT(*)
  **   FROM People**
  **   GROUP BY LName;**

  mysql> SELECT AVG(MaxSize)
  **   FROM Courses;**

  mysql> SELECT AVG(Grade)
  **   FROM CurrentlyEnrolled**
  **   GROUP BY CourseID;**

- Other useful functions: **AVG(), STDDEV(), MAX(), SUM()**

Let's Query the Data

- List all the Full professors in our database (ProfID).

- List every student with a GPA less than 1.0 (StudentID)

Joins

- List all the Full professors in our database
  (FName, LName).

```
mysql> SELECT FName, LName
       FROM People, Professors
       WHERE
       People.PersonID=Professors.ProfID
       AND
       Rank="Full";
```

- List every student with a GPA less than 1.0

  (StudentID, FName, LName)

Joins

- ## Inner Join

    – matching records in each table

    SELECT * FROM People, Students WHERE (People.PersonID=Students.StudentID);

- ## Outer Join

    – all records in each table (maybe not matching)

    – may produce NULL values for some columns

    SELECT * FROM People LEFT JOIN Students ON (People.PersonID=Students.StudentID);

    - **Produce all rows from the LEFT table**

    – **RIGHT JOIN**

    - **Produce all rows from the RIGHT table;**

CS445
Pacific University

Let's Query the Data

- List all the courses in our database taught by an Associate professor. (CourseID) (CourseID, Title)

- How many courses are taught by each rank of professor?

- How many professors teach zero courses?

Joins

- Four table joins, show all courses taught by **Associate** Profs (Title, FName, LName)

```
mysql> SELECT *
        FROM Courses, CurrentlyTeaching, Professors, People
        WHERE



        and Rank = "Associate";
```

Joins

- A join looks a one row at a time
- Some queries need more information
- Who is in a class with Bart Simpson?

Subquery

- Who was in class with Bart Simpson?

- Let's find the each course each person is in

- Let's find all the courses Bart Simpson is in

- Let's combine these two queries

- The goal of a subquery is to send data into or out of a subquery and filter using that data.

Subquery

- Who was in class with Bart Simpson?
- Let's find the each course each person is in

  – exclude Bart Simpson

```
SELECT People.*, CurrentlyEnrolled.*

FROM Students, CurrentlyEnrolled, People

WHERE
(Students.StudentID=CurrentlyEnrolled.StudentID)

AND People.PersonID = Students.StudentID

AND (FName != "Bart" or LName != "Simpson")

order by PersonID;
```

- Which courses was Bart enrolled in?

Subquery

- Which courses was Bart enrolled in?

```
SELECT *
FROM CurrentlyEnrolled AS BSClass
WHERE
BSClass.StudentID=5 -- Bart Simpson;
```

- Let's only return rows from the previous query where the CourseID matches a row in the above query.

Subquery

```
SELECT People.*, CurrentlyEnrolled.*
FROM Students, CurrentlyEnrolled, People
WHERE
(Students.StudentID=CurrentlyEnrolled.StudentID)
AND People.PersonID = Students.StudentID
AND (FName != "Bart" or LName != "Simpson")
AND EXISTS
(
  SELECT *
  FROM CurrentlyEnrolled AS BSClass
  WHERE
  (CurrentlyEnrolled.CourseID=BSClass.CourseID)
  AND BSClass.StudentID=5 -- Bart Simpson;
)
order by PersonID;
```

Subquries

- Who has the maximum grade in each class? (Fname, Lname, grade, class name)
  - Does this require a subquery?

Views

- A View is a logical table backed up by a query

  - Changes automatically when the results of the query change

```
mysql> CREATE VIEW CS150_VW AS
  SELECT LName, FName, Grade, StudentID
  FROM Courses, CurrentlyEnrolled, People
  WHERE Courses.CourseID=
  CurrentlyEnrolled.CourseID and
  People.PersonID=StudentID and
  Title like "CS150%";
```

```
mysql> SELECT * FROM CS150_VW;

mysql> DELETE FROM People WHERE
  PersonID=5;

mysql> SELECT * FROM CS150_VW Order by Grade;

mysql> DROP VIEW CS150_VW;
```

Exercise

- Rebuild CS150_VW

- Determine how closely a student's grade in CS150 matches their GPA. (1.0 = perfect match, 0.5 = Grade is half the GPA, 1.5 Grade is 50% better than GPA)

- GPA goes from 0.0 to 4.0, Grade goes from 0.0 to 100.0

Control Flow

```
IF( condition, trueValue, falseValue)
```

```
SELECT Title, IF( MaxSize > 50, 1, 0)

FROM   Courses;
```

```
IFNULL(Value, returnIfValueIsNULL)
```

```
SELECT IFNULL(Title, "ITISNULL")

FROM   Courses;
```

There is also a case (switch) statement

GROUP BY and HAVING

- Allows SQL to filter on calculated/aggregate values

- Similar to **WHERE**

- must be last

SELECT **StudentID, avg(Grade) as AvgGrade, count(*) as NumberRows**

FROM **CurrentlyEnrolled**

WHERE **Grade > 20**

GROUP BY **StudentID**

HAVING **AvgGrade > 80 and NumberRows > 1;**

Limit – only show some results

```
SELECT StudentID, count(*) as Total
FROM WasIn
GROUP BY StudentID
HAVING Total > 1
LIMIT 2; -- show only first two rows

LIMIT 2,4; -- skip the first two
rows, then show the next 4

LIMIT 3; is equivalent to LIMIT 0, 3
```

Backup Your Database!

```
@db:~> mysqldump PUNetID_test -u PUNetID -p  >
backup_test.sql
```

Database Name
Output file

```
To see what this file looks like:

@db:~> cat  backup_test.sql  | less
```

```
Copy to Zeus for safe keeping!

@db:~> scp backup_test.sql PUNetID@zeus:
```

Don't forget the colon!

Practice

- List all Course titles and CourseID. For each Course, display the CourseID if Chadd teaches it and "Not A Chadd Course" otherwise.

- Find all courses whose maximum and minimum grade is at least 50 points different.

- Display each student name, course title, and student's grade in that course and the string "passing" or "not passing" if the student is not passing the course.

- Find each course that does not contain a Simpson.

- Display all students whose grade for a class is above the average grade for that class.

CS445
Pacific University

Explain

```
mysql> SHOW CREATE TABLE People;

mysql> SHOW CREATE TABLE CurrentlyTeaching;


mysql> EXPLAIN SELECT * FROM People,
  CurrentlyTeaching WHERE (PersonID=ProfID);

mysql> EXPLAIN SELECT * FROM People,
  CurrentlyTeaching WHERE (PersonID=ProfID)
  AND FName like '%a%';
```

EXPLAIN

TYPE: system, const, eq_ref, ref, index, all

ROWS: number of rows scanned

Indexes

```
mysql> USE chadd_test;

mysql> SHOW TABLE STATUS LIKE  'EnronVocab';

mysql> SHOW TABLE STATUS LIKE  'EnronWordCount';

mysql> SHOW CREATE TABLE EnronVocab;

mysql> SHOW CREATE TABLE EnronWordCount;

mysql> SHOW PROCESSLIST;

  mysql> EXPLAIN SELECT WordCount FROM
  EnronWordCount WHERE DocID = ??;

  mysql> EXPLAIN SELECT WordCount FROM
  EnronWordCount WHERE WordID = ??;
```

- 1 to 39861 DocID

- 1 to 28102 WordID

http://archive.ics.uci.edu/ml/datasets/
Bag+of+Words

Practice

- How many students are in each class?

- For each class, what was the min, max, average grade ?

  - do this with and without using the AVG() function.

- Who took a class with Bart Simpson and received a higer grade than Bart? Lower Grade?

INTO OUTFILE

- Save a query to a text file

```
SELECT StudenID, count(*) as Total
FROM CurrentlyEnrolled
GROUP BY StudentID
HAVING Total > 1
INTO OUTFILE '/tmp/PUNETID.txt';

-- writes data on the server

gray> scp /tmp/PUNETID.txt c@zeus:


mysql -u user -p -D database -e
"select ... " > outfile
```

LOAD DATA INFILE

```
mysql> source /tmp/createTest.sql;

mysql> ALTER TABLE test DISABLE KEYS;
mysql> SET FOREIGN_KEY_CHECKS=0;


mysql> LOAD DATA INFILE '/tmp/test.txt' INTO
TABLE test COLUMNS TERMINATED BY ',';


mysql> SET FOREIGN_KEY_CHECKS=1;
mysql> ALTER TABLE test ENABLE KEYS;


Query OK, 69679427 rows affected (21 min 34.26 sec)
```

- **with a well tuned MySQL (innodb_buffer_pool_size, innodb_log_*)**

Triggers

```
CREATE TRIGGER name BEFORE INSERT ON table
  FOR EACH ROW BEGIN
    -- SQL Statements  or control flow (IF)
    INSERT INTO test2 SET a2 = NEW.a1;
  END
;
```

The row being inserted

BEFORE | AFTER

INSERT | DELETE | UPDATE

Cannot stop an insert!

Trigger

```
CREATE TRIGGER name BEFORE INSERT ON table
  FOR EACH ROW BEGIN

    SIGNAL SQLSTATE '99991'
     SET MESSAGE_TEXT = 'ERROR MESSAGE';
  END
;


DROP TRIGGER name;
```

Stored Procedures

Control Flow

- CASE
- IF()
- IFNULL()
- NULLIF()

Advanced SQL

- Control Flow Functions

  - http://dev.mysql.com/doc/refman/5.5/en/control-flow-functions.html

- Trigger

  - http://dev.mysql.com/doc/refman/5.5/en/create-trigger.html

- http://dev.mysql.com/doc/refman/5.5/en/select.html

  - Having

  - Limit

  - into outfile

- load data

  - http://dev.mysql.com/doc/refman/5.5/en/load-data.html

- Binary Data

- Stored Procedures