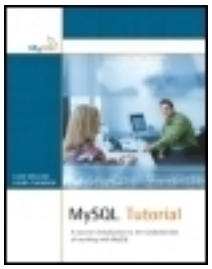
Introduction to MySQL and SQL Basics

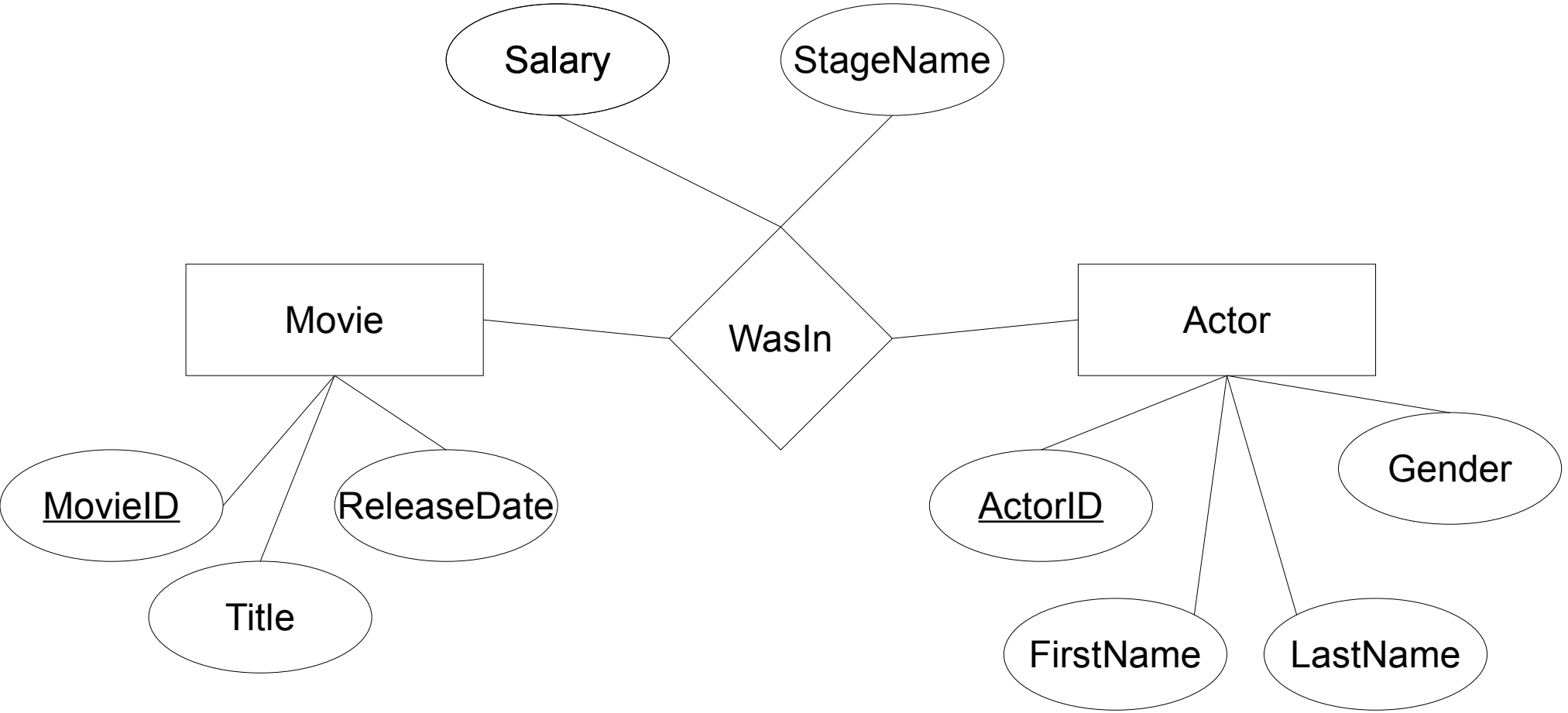
Sep 13, 2007
Read Chapter 3!

Managing and Using MySQL, 2nd Edition



http://dev.mysql.com/doc/refman/5.0/en/

Movie Database E-R Diagram



MySQL tasks

http://dev.mysql.com/doc/refman/5.0/en/

- start MySQL
  - setup user passwords
- shutdown MySQL
- create database
- create table
  - primary key
  - index
  - foreign key
- insert data
  - source a file
- delete data
  - drop
- query data
  - where
  - join
  - group
  - order
  - subquery

MySQL

- ssh to db.cs.pacificu.edu

```
ssh db.cs.pacificu.edu     (PuTTY on Windows)

db$ mysql -u PUNetID -p

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

mysql> show databases;

mysql> create database PUNetID_movie;

mysql> create database PUNetID_gradebook;

mysql> create database PUNetID_practice;

mysql> use PUNetID_movie;
```

MySQL Data types

http://dev.mysql.com/doc/refman/5.0/en/data-types.html

- INT
- FLOAT
- BOOLEAN
- DOUBLE
- VARCHAR(###) / VARBINARY(###)
- DATE / DATETIME
- [TINY|MEDIUM|LONG]TEXT
- [TINY|MEDIUM|LONG]BLOB
- ENUM

Create a Table

```
mysql> CREATE TABLE Movies
          (MovieID INT NOT NULL,
          Title VARBINARY(512) NOT NULL,
          ReleaseDate DATE NOT NULL,
          PRIMARY KEY(MovieID),
          INDEX(ReleaseDate)
          ) ENGINE=InnoDB;

 mysql> show create table Movies;
```

- Insert some data

```
mysql> INSERT INTO Movies  VALUES
          (1, 'Star Wars', '1977-05-25');

mysql> SELECT * FROM Movies;
```

MySQL

- Insert some more data

```
mysql> INSERT INTO Movies  VALUES
           (2, 'Indiana Jones and the Raiders of the
Lost Ark', '1981-06-12');

mysql> SELECT * FROM Movies;
```

Create a table

- Create another table

```
mysql> CREATE TABLE Actors
           (ActorID INT NOT NULL AUTO_INCREMENT,
           LastName VARBINARY(50),
           FirstName  VARBINARY(50) NOT NULL,
           Gender ENUM('Male', 'Female') NOT NULL,
           PRIMARY KEY(ActorID),
           INDEX(Gender)
           ) ENGINE=InnoDB;
```

MySQL

- Insert some data

```
mysql> INSERT INTO Actors  VALUES
         (null, 'Harrison','Ford', 'Male');

mysql> SELECT * FROM Actors;

mysql> SELECT LastName, Gender FROM Actors;
```

MySQL

```
mysql> CREATE TABLE WasIn
       (MovieID INT NOT NULL,
       ActorID INT NOT NULL,
       Salary FLOAT NOT NULL,
       StageName VARBINARY(200),
       PRIMARY KEY(MovieID, ActorID),
       INDEX(ActorID)
       ) ENGINE=InnoDB;
```

Constraints

```
mysql> ALTER TABLE WasIn ADD
 CONSTRAINT RequireMovieID FOREIGN KEY
 (MovieID) REFERENCES Movies(MovieID)
 ON DELETE CASCADE;

mysql> ALTER TABLE WasIn ADD
 CONSTRAINT RequireActorID FOREIGN KEY
 (ActorID) REFERENCES Actors(ActorID)
 ON DELETE CASCADE;
```

MySQL

- Insert some data

```
mysql> INSERT INTO WasIn  VALUES(1, 1,
500000.00, null);

mysql> INSERT INTO WasIn  VALUES(2, 1,
5000000.00, null);

mysql> SELECT * FROM WasIn;

mysql> INSERT INTO WasIn VALUES(10, 1,
10.20, null);
```

Let's make this go faster

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

```
mysql> source /home/chadd/Movies.sql;
```

Deleting Data

- Let's delete some data

```
mysql> SELECT * FROM Movies;

mysql> SELECT * FROM WasIn WHERE MovieID=9;

mysql> DELETE FROM Movies WHERE MovieID=9;

mysql> SELECT * FROM Movies;

mysql> SHOW TABLES;

mysql> DROP TABLE Junk;
```

Queries

- What movies were made after 1980?

```
mysql> SELECT *
       FROM Movies
       WHERE
       ReleaseDate > '1980-12-31';
```

Let's Query the Data

- What movies were made after 1980 but before 1994?

```
mysql> SELECT * FROM Movies WHERE
ReleaseDate > '1980-12-31' AND ReleaseDate <
'1994-01-01';
```

- Just show the Titles and Release Dates

```
mysql> SELECT Title, ReleaseDate FROM Movies
WHERE ReleaseDate > '1980-12-31' AND
ReleaseDate < '1994-01-01';
```

Let's Query the Data

- List all the female actors in our database.

- List the ActorID of every actor that has used a StageName.

Order By

- Let's sort the output

```
mysql> SELECT *
       FROM Movies
       ORDER BY ReleaseDate;

mysql> SELECT *
       FROM Movies
       ORDER BY ReleaseDate DESC ;


mysql> SELECT *
       FROM Actors
       ORDER BY Lastname, Firstname;
```

Group By

- Aggregate selected rows

```
mysql> SELECT ActorID, MovieID
       FROM WasIn ;
```

```
mysql> SELECT ActorID, COUNT(MovieID)
       FROM WasIn
       GROUP BY ActorID;
```

- Other useful functions: **AVG(), STDDEV(), MAX(), SUM()**
- How much money has each actor made total? Just the female actors?

Joins

- Let's get data out of two tables at once
  What movies was Harrison Ford in?

```
mysql> SELECT Title, Movies.MovieID
       FROM Movies LEFT JOIN WasIn ON
       (Movies.MovieID=WasIn.MovieID)
       WHERE
       ActorID=1;
```

Let's Query the Data

- List all the movies in our database that starred a female actor.

Joins

- Three table joins

```
mysql> SELECT *
        FROM Movies LEFT JOIN WasIn ON
        (Movies.MovieID=WasIn.MovieID)
        LEFT JOIN Actors ON (Actors.ActorID=
        WasIn.ActorID)
        WHERE
        LastName='Ford' AND FirstName='Harrison';
```

Subqueries

- Who was in a movie with Harrison Ford?

```
mysql> SELECT *
FROM Actors LEFT JOIN WasIn ON
(Actors.ActorID=WasIn.ActorID)
WHERE
EXISTS
(
   SELECT *
   FROM Movies LEFT JOIN WasIn AS HFMovie ON
   (Movies.MovieID=HFMovie.MovieID)
   LEFT JOIN Actors ON
   (Actors.ActorID=HFMovie.ActorID)
   WHERE LastName='Ford' AND
   FirstName='Harrison' and HFMovie.MovieID =
   WasIn.MovieID
 ) AND (LastName != 'Ford' OR FirstName !
='Harrison');
```

Exercise

- How many movies did each actor star in with Harrison Ford (ignore actors that starred in zero movies with Harrison Ford)?

Explain

```
mysql> SHOW CREATE TABLE Movies;

mysql> SHOW CREATE TABLE WasIn;


mysql> EXPLAIN SELECT * FROM Movies LEFT JOIN
  WasIn ON (Movies.MovieID=WasIn.MovieID);

mysql> EXPLAIN SELECT * FROM WasIn LEFT JOIN
  Movies ON (Movies.MovieID=WasIn.MovieID);
```

EXPLAIN

- TYPE: system, const, eq_ref, ref, index, all
- ROWS: number of rows scanned

Explain

```
mysql> SHOW CREATE TABLE WasIn;

mysql> EXPLAIN SELECT * FROM Actors LEFT JOIN
  WasIn ON (Actors.ActorID=WasIn.ActorID);


mysql> ALTER TABLE WasIn DROP FOREIGN KEY
  RequireActorID ;

mysql> ALTER TABLE WasIn DROP KEY ActorID ;

mysql> SHOW CREATE TABLE WasIn;


mysql> EXPLAIN SELECT * FROM Actors LEFT JOIN
  WasIn ON (Actors.ActorID=WasIn.ActorID);
```

Indexes

```
mysql> USE chadd_IndexExample;
mysql> SHOW TABLE STATUS LIKE  'BigDataNoIndex';


mysql> SHOW CREATE TABLE BigDataNoIndex;
mysql> SHOW CREATE TABLE MethodInfoNoIndex;


mysql> EXPLAIN SELECT MethodID FROM MethodInfoNoIndex
  LEFT JOIN BigDataNoIndex ON (StartLine = PrevLineNo)
  WHERE LineNo=?? AND StartColumn=??;
mysql> SELECT MethodID FROM MethodInfoNoIndex LEFT
  JOIN BigDataNoIndex ON (StartLine = PrevLineNo)
  WHERE LineNo=?? AND StartColumn=??;
```

Indexes

```
mysql> SHOW CREATE TABLE BigData;

mysql> SHOW CREATE TABLE MethodInfo;
```

```
mysql>  EXPLAIN SELECT MethodID FROM MethodInfo
  LEFT JOIN BigData ON (StartLine = PrevLineNo)
  WHERE LineNo=?? AND StartColumn=??;
```

```
mysql> SELECT MethodID FROM MethodInfo LEFT JOIN
  BigData ON (StartLine = PrevLineNo) WHERE
  LineNo=?? AND StartColumn=??;
```

Use the first two or three digits of your PUNetID to replace the ??.

Indexes

Use the first two or three digits of your PUNetID to replace the ??.

```
mysql> EXPLAIN SELECT BD1.TransID, BD2.FileID
  FROM BigDataNoIndex AS BD1 LEFT JOIN
  BigDataNoIndex AS BD2 ON (BD1.LineNo =
  BD2.PrevLineNo) WHERE
  BD1.TransID=BD2.PrevTransID and BD1.FileID=??;
```

```
mysql> SELECT BD1.TransID, BD2.FileID FROM
  BigDataNoIndex AS BD1 LEFT JOIN BigDataNoIndex
  AS BD2 ON (BD1.LineNo = BD2.PrevLineNo) WHERE
  BD1.TransID=BD2.PrevTransID and BD1.FileID=??;
```

How long did it take? Run the same query again.

Indexes

Use the same digits as the previous slide to replace the ??.

```
mysql> EXPLAIN SELECT BD1.TransID, BD2.FileID
  FROM BigData AS BD1 LEFT JOIN BigData AS BD2 ON
  (BD1.LineNo = BD2.PrevLineNo) WHERE
  BD1.TransID=BD2.PrevTransID and BD1.FileID=??;
```

```
mysql> SELECT BD1.TransID, BD2.FileID FROM
  BigData AS BD1 LEFT JOIN BigData AS BD2 ON
  (BD1.LineNo = BD2.PrevLineNo) WHERE
  BD1.TransID=BD2.PrevTransID and BD1.FileID=??;
```

How long did it take? Run the same query again.

Other Constraints

- UNIQUE
  - create an index on a set of columns and require each entry to be unique
  - similar to a Primary Key
  -

```
mysql> CREATE TABLE WasIn
           (MovieID INT NOT NULL,
           ActorID INT NOT NULL,
           Salary FLOAT NOT NULL,
           StageName VARBINARY(200),
           PRIMARY KEY(MovieID, ActorID),
           UNIQUE (StageName)
           ) ENGINE=InnoDB;
```

Views

- A View is a logical table backed up by a query
  - Changes automatically when the results of the query change
  -

```
mysql> USE PUNetID_movie;

mysql> CREATE VIEW HFordMovies AS SELECT Title,
  Movies.MovieID FROM Movies LEFT JOIN
  WasIn ON
  (Movies.MovieID=WasIn.MovieID)WHERE
  ActorID=1;


mysql> SELECT * FROM HFordMovies;
```

Views

```
mysql> INSERT INTO WasIn VALUES (11, 1,
  100, null);
mysql> SELECT * FROM HFordMovies;


// delete the data just added to WasIn and
  rerun the query on the view
```

Practice

- How many actors were in each movie?
- For each movie, what was the average salary?
  - do this with and without using the AVG() function.
- Who starred in a movie with Harrison Ford where HF earned more than $50,000?
- Who starred in a movie with Harrison Ford where they made more than HF for that movie?

Start MySQL

- Create a directory for MySQL to store all the data:

```
$ mkdir /zeus/login/cs445_MySQLdata
$ mkdir /zeus/login/cs445_MySQLdata/data
```

- Initialize the data directory

```
$ mysql_install_db –
datadir=/zeus/chadd/cs445_MySQLdata/data/
```

- This stores your data on zeus so that you can get to it from any machine in the lab

```
$ mysqld_safe –defaults-
file=/zeus/login/cs445_MySQLdata/my.cnf &
```

- mysqladmin -u root password 'password'
- mysqladmin -u root -h localhost password 'password'
-
- ```
  mysql> SELECT FirstName,LastName, SUM(Salaray)
  FROM WasIn LEFT JOIN Actors ON (Actors.ActorID=
  WasIn.ActorID)
  GROUP BY LastName, FirstName;
  ```
  ```