HERIOT WATT UNIVERSITY

F27WD: Web Design & Databases

Databases Lecture 2: Databases - what do we do with our data?

Fiona McNeill

February 20th 2019

LEADERS IN IDEAS AND SOLUTIONS

HERIOT WATT UNIVERSITY

Before we start …

Make sure you have joined today’s session

Before we start …

Make sure you have joined today's session:

https://goo.gl/forms/UI2vpYOW1ziUhKat1

First question:

Are you in the session?

1. Yes
2. No

What do we do with our data?

Last lecture, we started to gather some data about films.

What do we do with our data?

Last lecture, we started to gather some data about films.

We started to think about how we could *organise* the data so that it would be …

What do we do with our data?

Last lecture, we started to gather some data about films.

We started to think about how we could *organise* the data so that it would be *useful* and *accessible*

What do we do with our data?

Last lecture, we started to gather some data about films.

We started to think about how we could *organise* the data so that it would be *useful* and *accessible.*

We started to build a **database.**

What do we do with our data?

Today we will think about:

1. How do we *organise* and *constrain* our data
2. How can we *access* and *filter* our data

Our data so far …

filmName	director	genre	rating	year
Ghostbusters	Paul Feig	Comedy	12A	2016
Gone with the Wind	Victor Fleming et al	Drama	PG	1940
Star Wars Episode IV	George Lucas	Sci-Fi	U	1977
Die Hard	John McTiernan	Action	18	1988

Our data so far …

- Remember, this is a **relational** database

Our data so far …

- Remember, this is a **relational** database (we'll find out what that really means later)

Our data so far …

- Remember, this is a **relational** database (we’ll find out what that really means later)
- Designed using SQL (this is usually pronounced ‘see-qual’ but you can also say ’S-Q-L’)

Our data so far …

- Remember, this is a **relational** database (we’ll find out what that really means later)
- Designed using SQL (this is usually pronounced ‘see-qual’ but you can also say ’S-Q-L’)
- MySQL is the Database Management System - you can think of this as the suite of tools that allows you to do all of this.

Creating the database

- First of all, we use the DBMS to create a **database**
- Then, we create **tables** within our database.

At its heart, a database is a collection of **tables**

HERIOT WATT UNIVERSITY

Creating a table - naming

Our table is called **MyFilms**

- Use a *meaningful* name
- Don't use spaces
- Common to start with capital

e.g., **MyTable** works but isn't very useful

My Films, My-films, etc won't work in SQL

filmName	director	genre	rating	year
Ghostbusters	Paul Feig	Comedy	12A	2016
Gone with the Wind	Victor Fleming et al	Drama	PG	1940
Star Wars Episode IV	George Lucas	Sci-Fi	U	1977
Die Hard	John McTiernan	Action	18	1988

Naming the columns

filmName	director	genre	rating	year
Ghostbusters	Paul Feig	Comedy	12A	2016
Gone with the Wind	Victor Fleming et al	Drama	PG	1940
Star Wars Episode IV	George Lucas	Sci-Fi	U	1977
Die Hard	John McTiernan	Action	18	1988

Naming the columns

Same rules as for tables:

- Use a *meaningful* name
- Don't use spaces
- Common to start with a lowercase letter

filmName	director	genre	rating	year
Ghostbusters	Paul Feig	Comedy	12A	2016
Gone with the Wind	Victor Fleming et al	Drama	PG	1940
Star Wars Episode IV	George Lucas	Sci-Fi	U	1977
Die Hard	John McTiernan	Action	18	1988

Constraining the columns

What kind of things can we put in each column?

filmName	director	genre	rating	year
Ghostbusters	Paul Feig	Comedy	12A	2016
Gone with the Wind	Victor Fleming et al	Drama	PG	1940
Star Wars Episode IV	George Lucas	Sci-Fi	U	1977
Die Hard	John McTiernan	Action	18	1988

HERIOT WATT UNIVERSITY

Constraining the columns

What kind of things can we put in each column?

- filmName, director, rating:

String of characters: letters, numbers, spaces, punctuation, etc

filmName	director	genre	rating	year
Ghostbusters	Paul Feig	Comedy	12A	2016
Gone with the Wind	Victor Fleming et al	Drama	PG	1940
Star Wars Episode IV	George Lucas	Sci-Fi	U	1977
Die Hard	John McTiernan	Action	18	1988

Constraining the columns

What kind of things can we put in each column?

- filmName, director, rating:

String of characters: letters, numbers, spaces, punctuation, etc

filmName	director	genre	rating	year
Ghostbusters	Paul Feig	Comedy	12A	2016
Gone with the Wind	Victor Fleming et al	Drama	PG	1940
Star Wars Episode IV	George Lucas	Sci-Fi	U	1977
Die Hard	John McTiernan	Action	18	1988

This is called VARCHAR(X):
A VARiable list of CHARacters up to a maximum of X (e.g., 10)

Constraining the columns

What kind of things can we put in each column?

- genre:

The same, but we could restrict it to a three-character code

filmName	director	genre	rating	year
Ghostbusters	Paul Feig	Comedy	12A	2016
Gone with the Wind	Victor Fleming et al	Drama	PG	1940
Star Wars Episode IV	George Lucas	Sci-Fi	U	1977
Die Hard	John McTiernan	Action	18	1988

Constraining the columns

What kind of things can we put in each column?

- genre:

The same, but we could restrict it to a three-character code

filmName	director	genre	rating	year
Ghostbusters	Paul Feig	COM	12A	2016
Gone with the Wind	Victor Fleming et al	DRA	PG	1940
Star Wars Episode IV	George Lucas	SCF	U	1977
Die Hard	John McTiernan	ACT	18	1988

Then we get CHAR(3):
A CHARacter string exactly 3 characters long

Constraining the columns

What kind of things can we put in each column?

- year:

This is a number, exactly four digits long

filmName	director	genre	rating	year
Ghostbusters	Paul Feig	COM	12A	2016
Gone with the Wind	Victor Fleming et al	DRA	PG	1940
Star Wars Episode IV	George Lucas	SCF	U	1977
Die Hard	John McTiernan	ACT	18	1988

Constraining the columns

What kind of things can we put in each column?

- year:

This is a number, exactly four digits long

filmName	director	genre	rating	year
Ghostbusters	Paul Feig	COM	12A	2016
Gone with the Wind	Victor Fleming et al	DRA	PG	1940
Star Wars Episode IV	George Lucas	SCF	U	1977
Die Hard	John McTiernan	ACT	18	1988

Then we get INTEGER(4):
An INTEGER exactly 4 characters long

Constraining the columns

These are called

DATA TYPES

Constraining the columns

How else might we want to constrain columns?

Other common data types are:

- DECIMAL(9,2):
  - A decimal number, up to a number of characters (e.g., 9)
  - With a fixed number of decimal places (e.g., 2)
- DATETIME
  - Commonly YYYY-MM-DD HH:MM:SS
  - MySQL allows other formats

Constraining the columns

How else might we want to constrain the columns?

Constraining the columns

We might want to say:

- *Only characters that represent the names of films in IMDB can go in the first column*

Constraining the columns

We might want to say:

- *Only characters that represent the names of films in IMDB can go in the first column*
- *Only integers between 1890 and 2018 can go in the fifth column*

Constraining the columns

We might want to say:

- *Only characters that represent the names of films in IMDB can go in the first column*
- *Only integers between 1890 and 2018 can go in the fifth column*

Standard MySQL will **not** enforce this for us!

i.e., someone could enter 9382 in column 5 and it will not generate an error

Primary Keys

- In MySQL (and many other situations), you must have a *primary key*
- This *uniquely identifies* an entry
- e.g., student number could be a primary key

Qu2: In our database, what could be a primary key?

1. film name
2. director
3. genre
4. rating
5. year
6. More than one of the above
7. None of the above

Qu: In our database, what could be a primary key?

1. Film name
2. director
3. genre
4. rating
5. year
6. More than one of the above
7. None of the above

So what can we do?

So what can we do?

- Create a primary key from existing data - e.g.,

filmNameYear	director	genre	rating	year
Ghostbusters2016	Paul Feig	COM	12A	2016
Gone with the Wind1940	Victor Fleming et al	DRA	PG	1940
Star Wars Episode IV1977	George Lucas	SCF	U	1977
Die Hard1988	John McTiernan	ACT	18	1988

So what can we do?

- Make up a primary key - e.g.,

filmID	filmName	director	genre	rating	year
48392	Ghostbusters	Paul Feig	COM	12A	2016
12904	Gone with the Wind	Victor Fleming et al	DRA	PG	1940
29342	Star Wars Episode IV	George Lucas	SCF	U	1977
25094	Die Hard	John McTiernan	ACT	18	1988

So what can we do?

What does this new value **filmID** mean?

So what can we do?

What does this new value **filmID** mean?

- You could use a universally recognised film ID number (e.g., from IMDB)
- You could create a new number that's only valid for your database
  - e.g., the order they are added to the database

So … let’s create a table

```
CREATE TABLE MyFilm (
  filmID INTEGER PRIMARY KEY,
  filmName VARCHAR(30),
  director VARCHAR(20),
  genre CHAR(3),
  rating VARCHAR(5),
  year INTEGER(4)
 ) ENGINE=INNODB;
```

Inserting data into the table

We use **INSERT INTO** to add new *data instances* (lines of data)

e.g.

```
INSERT INTO MyFilm VALUES (4920, "Murder on the Orient Express", "Kenneth Branagh", "ACT", "12A", 2017);
```

Inserting data into the table

We use **INSERT INTO** to add new *data instances* (lines of data)

e.g.

INSERT INTO MyFilm VALUES (4920, "Murder on the Orient Express", "Kenneth Branagh", "ACT", "12A", 2017);

- Surround text & dates with single quotation marks

Inserting data into the table

We use **INSERT INTO** to add new *data instances* (lines of data)

e.g.

```
INSERT INTO MyFilm VALUES (4920, "Murder on the Orient Express", "Kenneth Branagh", "ACT", "12A", 2017);
```

- Surround text & dates with single quotation marks
- Numbers don’t need this

HERIOT WATT UNIVERSITY

Inserting data into the table

We use **INSERT INTO** to add new *data instances* (lines of data)

e.g.

```
INSERT INTO MyFilm VALUES (4920, "Murder on the Orient Express", "Kenneth Branagh", "ACT", "12A", 2017);
```

- Surround text & dates with single quotation marks
- Numbers don’t need this
- If you include a date, it must be in the right format

Inserting data into the table

We use **INSERT INTO** to add new *data instances* (lines of data)

e.g.

INSERT INTO MyFilm VALUES (4920, "Murder on the Orient Express", "Kenneth Branagh", "ACT", "12A", 2017);

- Surround text & dates with single quotation marks
- Numbers don’t need this
- If you include a date, it must be in the right format
- End with a semi-colon

Inserting data into the table

We use **INSERT INTO** to add new *data instances* (lines of data)

e.g.

INSERT INTO MyFilm VALUES (4920, "Murder on the Orient Express", "Kenneth Branagh", "ACT", "12A", 2017);

- Surround text & dates with single quotation marks
- Numbers don't need this
- If you include a date, it must be in the right format
- End with a semi-colon

Inserting data into the table

We use **INSERT INTO** to add new *data instances* (lines of data)

e.g.

INSERT INTO MyFilm VALUES (4920, "Murder on the Orient Express", "Kenneth Branagh", "ACT", "12A", 2017);

Come up with your own entry.

Selecting data

- We have to **SELECT** what we want to view
- We can use * to represent ‘anything’

Selecting data

So

```
SELECT * FROM MyFilm;
```

would give us:

Selecting data

filmID	filmName	director	genre	rating	year
48392	Ghostbusters	Paul Feig	COM	12A	2016
12904	Gone with the Wind	Victor Fleming et al	DRA	PG	1940
29342	Star Wars Episode IV	George Lucas	SCF	U	1977
25094	Die Hard	John McTiernan	ACT	18	1988
49207	Murder on the Orient Express	Kenneth Branagh	ACT	PG-13	2017

Selecting data

What would

SELECT filmName FROM MyFilm;

give us?

Selecting data

filmName
Ghostbusters
Gone with the Wind
Star Wars Episode IV
Die Hard
Murder on the Orient Express

Choosing what data to look at

Maybe we only want to see some of the columns. We specify what we want to see:

e.g.,

```
  SELECT filmName, genre, year FROM MyFilm;
```

Choosing what data to look at

filmName	genre	year
Gone with the Wind	DRA	1940
Star Wars Episode IV	SCF	1977
Die Hard	ACT	1988
Ghostbusters	COM	2016
Murder on the Orient Express	ACT	2017

Ordering the data

If we want to see the data in a particular order, we use **SELECT** as before, and also **ORDER BY**

e.g.,

```
SELECT * FROM MyFilm ORDER BY year;
```

Ordering the data

filmID	filmName	director	genre	rating	year
12904	Gone with the Wind	Victor Fleming et al	DRA	PG	1940
29342	Star Wars Episode IV	George Lucas	SCF	U	1977
25094	Die Hard	John McTiernan	ACT	18	1988
48392	Ghostbusters	Paul Feig	COM	12A	2016
49207	Murder on the Orient Express	Kenneth Branagh	ACT	PG-13	2017

Choosing what data to look at

Maybe we only want to see some of the rows. We use **WHERE**

e.g.,

```
   SELECT * FROM MyFilm WHERE
genre=“ACT”;
```

Choosing what data to look at

filmID	filmName	director	genre	rating	year
25094	Die Hard	John McTiernan	ACT	18	1988
49207	Murder on the Orient Express	Kenneth Branagh	ACT	PG-13	2017

Combining options

We can combine this however we like,

e.g.,

SELECT filmName, genre, year FROM MyFilm WHERE genre="ACT" ORDER BY year;

Combining options

filmName	genre	year
Die Hard	ACT	1988
Murder on the Orient Express	ACT	2017

Using AND/OR

We can choose to put two different constraints on what we are shown:

e.g.,

SELECT filmName, genre, year FROM MyFilm WHERE genre = “COM” AND year>=2000;

Ordering the data

filmName	genre	year
Ghostbusters	COM	2016

Using AND/OR

Or we can choose different options:

e.g.,

```
  SELECT filmName, genre, year FROM MyFilm
WHERE genre = “ACT” OR genre = “COM”;
```

Ordering the data

filmName	genre	year
Die Hard	ACT	1988
Ghostbusters	COM	2016
Murder on the Orient Express	ACT	2017

Counting stuff

Want to know how many entries fit a certain criterion? Use **count** instead of SELECT, and specify what you want to count.

e.g.,

SELECT count(*) FROM MyFilm WHERE genre = ‘ACT’ OR genre = ‘COM’;

Counting stuff

count(*)
3

Adding things up

We can also see how much different entries add up to. Use **sum** instead of SELECT, and specify what you want to sum.

e.g.,

SELECT sum(year) FROM MyFilm WHERE genre = ‘ACT’ OR genre = ‘COM’;

Adding things up*

sum(year)
6021

* yeah, ok, this doesn’t really make sense - why would you want to add up years? What would this actually mean?

Sum makes more sense if you have columns containing amounts of money or something like that, where you might want to see the total.
But I had to shoehorn this into the example we are working with!

Adding things up - a better example

Let’s imagine we had an extra column that told us how much a film cost to produce.

Then we could say, e.g.,

SELECT sum(cost) year FROM MyFilm WHERE genre = ‘ACT’ OR genre = ‘COM’;

This would be useful if we wanted to know how much money was spent on action and comedy films.

Aggregating data by GROUPing

We can use **GROUP** to combine the data in different ways

e.g.,

  SELECT genre, sum(year) FROM MyFilm
GROUP BY genre;

Summary

What have we learned today?

Summary

What have we learned today?

- How to create a MySQL database table
  - How to name tables and columns
  - What **data types** are, and some examples of common SQL data types
  - What **primary keys** are

Summary

What have we learned today?

- How to *access* and *sort* data
  - **INSERT**
  - **SELECT**
  - **SELECT** plus **ORDER BY**
  - **SELECT** plus **WHERE**
  - Combining these
  - **AND/OR**
  - **count** and **sum**
  - **SELECT** plus **GROUP**

What next?

You can do it all yourself.

In the lab, you will create and manipulate your own database using the techniques we have looked at today.