Lab 3

Nigel Nelson
CS-3860
9/21/21

- Part I – Creating *Video* database:
  - Importing the unaltered data into the new data model:
    - Inserting data into video_recordings_new table

```
35  insert into video_recordings_new
36  select recording_id, director, title,
37  category, image_name, duration, rating,
38   year_released, price, stock_count
39  from video_recordings
40  where recording_id is not null;
```

    - Inserting data into video_categories_new table:

```
50  insert into video_categories_new
51  select distinct category
52  from video_recordings;
```

    - Inserting data into video_actors_new table:

```
58  insert into video_actors_new
59  select distinct name
60  from video_actors;
```

    - Inserting data into ratings table:

```
42  insert into ratings
43  select distinct rating
44  from video_recordings;
```

    - Inserting data into directors table:

```
46  insert into directors
47  select distinct director
48  from video_recordings;
```

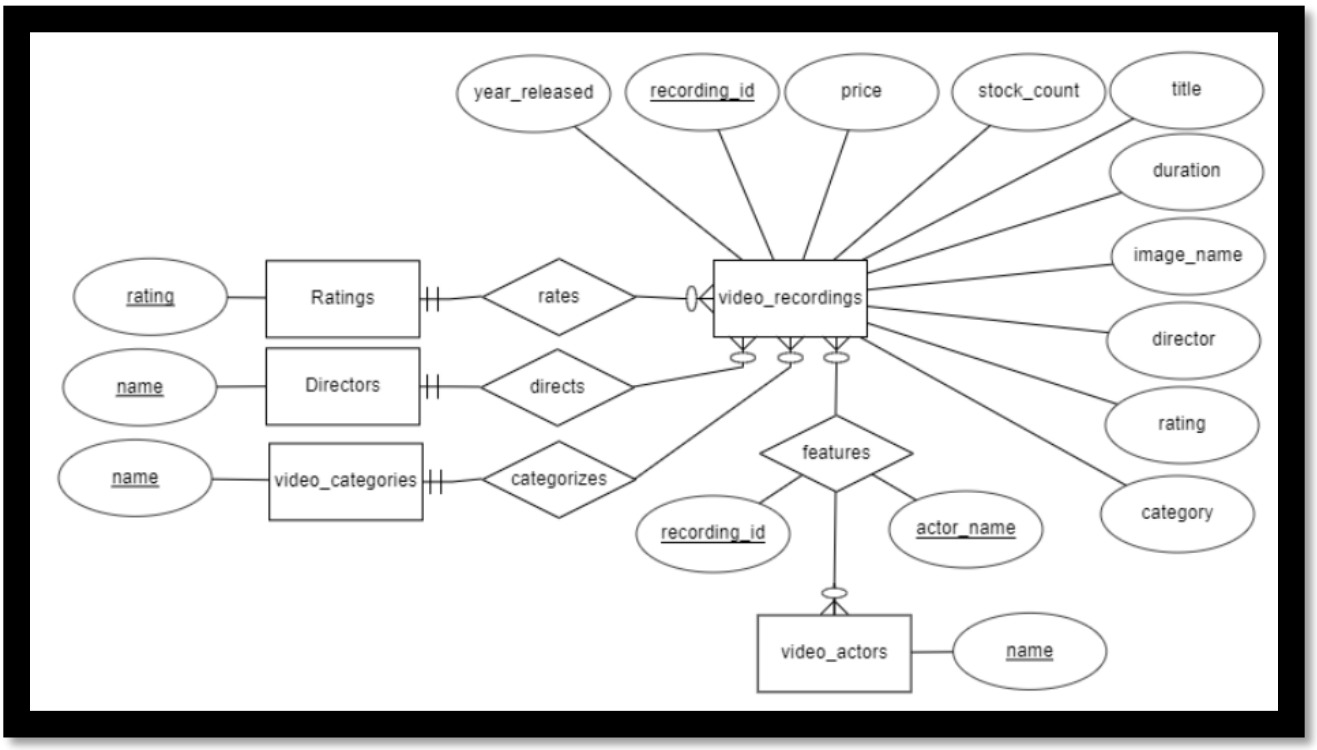
    - Inserting data into features table:

```
54  insert into features
55  select recording_id, name
56  from video_actors;
```

  - Resulting ERD diagram of final model:

Lab 3

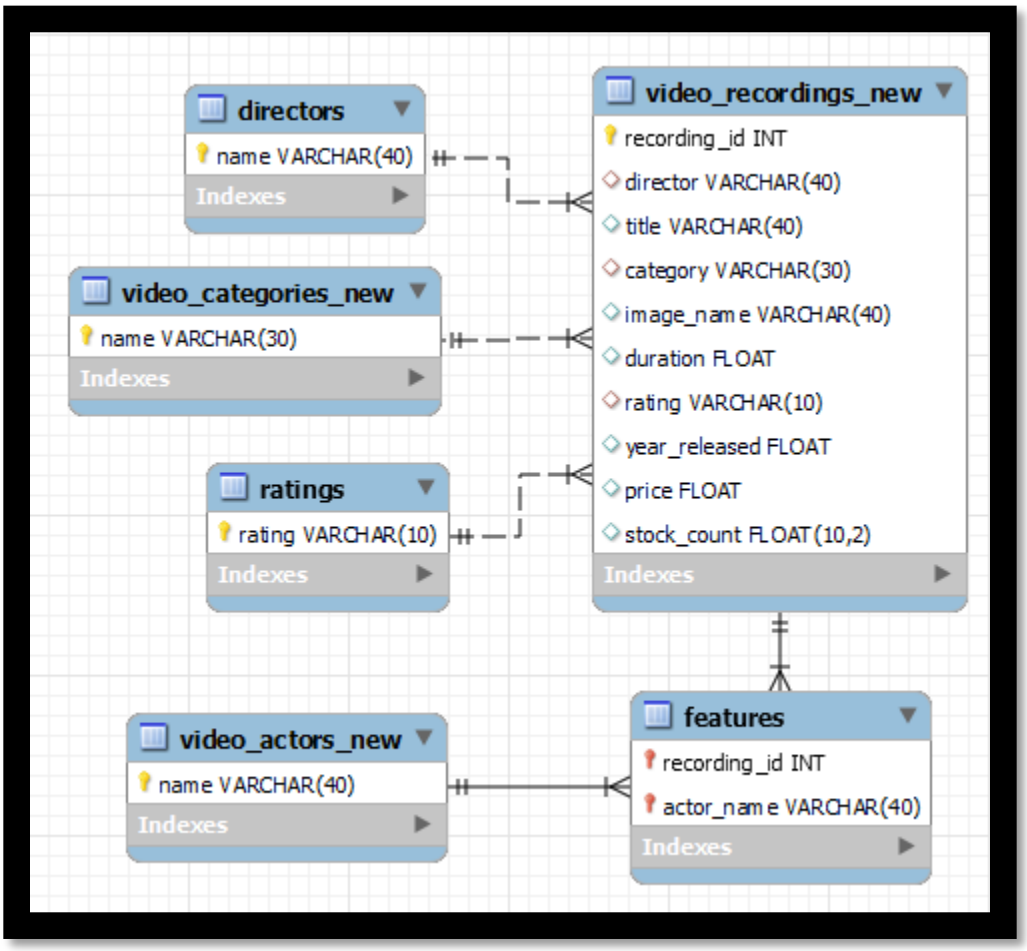
Nigel Nelson
CS-3860
9/21/21



- Resulting Relational Model:

Lab 3

Nigel Nelson
CS-3860
9/21/21



- Final Model SQL create table scripts:

Lab 3

Nigel Nelson
CS-3860
9/21/21

```
-- -----------------------------------------------------
-- Table `video`.`directors`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `video`.`directors` (
  `name` VARCHAR(40) NOT NULL,
  PRIMARY KEY (`name`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;


-- -----------------------------------------------------
-- Table `video`.`ratings`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `video`.`ratings` (
  `rating` VARCHAR(10) NOT NULL,
  PRIMARY KEY (`rating`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;


-- -----------------------------------------------------
-- Table `video`.`video_categories_new`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `video`.`video_categories_new` (
  `name` VARCHAR(30) NOT NULL,
  PRIMARY KEY (`name`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;
```

Lab 3

Nigel Nelson
CS-3860
9/21/21

```
-- -----------------------------------------------------
-- Table `video`.`video_recordings_new`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `video`.`video_recordings_new` (
  `recording_id` INT NOT NULL,
  `director` VARCHAR(40) NULL DEFAULT NULL,
  `title` VARCHAR(40) NULL DEFAULT NULL,
  `category` VARCHAR(30) NULL DEFAULT NULL,
  `image_name` VARCHAR(40) NULL DEFAULT NULL,
  `duration` FLOAT NULL DEFAULT NULL,
  `rating` VARCHAR(10) NULL DEFAULT NULL,
  `year_released` FLOAT NULL DEFAULT NULL,
  `price` FLOAT NULL DEFAULT NULL,
  `stock_count` FLOAT(10,2) NULL DEFAULT NULL,
  PRIMARY KEY (`recording_id`),
  INDEX `rating` (`rating` ASC) VISIBLE,
  INDEX `director` (`director` ASC) VISIBLE,
  INDEX `category` (`category` ASC) VISIBLE,
  CONSTRAINT `video_recordings_new_ibfk_1`
    FOREIGN KEY (`rating`)
    REFERENCES `video`.`ratings` (`rating`),
  CONSTRAINT `video_recordings_new_ibfk_2`
    FOREIGN KEY (`director`)
    REFERENCES `video`.`directors` (`name`),
  CONSTRAINT `video_recordings_new_ibfk_3`
    FOREIGN KEY (`category`)
    REFERENCES `video`.`video_categories_new` (`name`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;
```

Lab 3

Nigel Nelson
CS-3860
9/21/21

```
-- -----------------------------------------------------
-- Table `video`.`video_actors_new`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `video`.`video_actors_new` (
  `name` VARCHAR(40) NOT NULL,
  PRIMARY KEY (`name`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;


-- -----------------------------------------------------
-- Table `video`.`features`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `video`.`features` (
  `recording_id` INT NOT NULL,
  `actor_name` VARCHAR(40) NOT NULL,
  PRIMARY KEY (`recording_id`, `actor_name`),
  INDEX `actor_name` (`actor_name` ASC) VISIBLE,
  CONSTRAINT `features_ibfk_1`
    FOREIGN KEY (`recording_id`)
    REFERENCES `video`.`video_recordings_new` (`recording_id`),
  CONSTRAINT `features_ibfk_2`
    FOREIGN KEY (`actor_name`)
    REFERENCES `video`.`video_actors_new` (`name`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;
```

Lab 3

Nigel Nelson
CS-3860
9/21/21

- Running select statements to verify that data has been successfully imported:
  - *Video_recordings_new*, Rows returned: 55

```
123 •   select * from video_recordings_new;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap

recording_id	director	title	category	image_name
3000	Francis Ford Coppola	Apocalypse Now	Action & Adventure	apocalypse_now.gif
3001	Michael Bay	Bad Boys	Action & Adventure	badboys.gif
3002	Mel Gibson	Braveheart	Action & Adventure	braveheart.gif
3003	Mimi Leder	Deep Impact	Action & Adventure	deep_impact.gif
3004	J. Robert Wagoner	Disco Godfather	Action & Adventure	disco_godfather.gif
3005	Stanley Kubrick	Full Metal Jacket	Action & Adventure	full_metal_jacket.gif
3006	Roland Emmerich	Independence Day	Action & Adventure	independence_day.gif

  - *Video_actors_new*, Rows returned: 335

```
121 •   select * from video_actors_new;
```

Result Grid | Filter Rows:

name
Adam Baldwin
Adrian Pasdar
Adrienne Corri
Adrienne King
Alec Baldwin
Alec Guinness
Alfre Woodard

  - *Video_categories_new*, Rows returned: 6

```
125 •   select * from video_categories_new;
```

Result Grid | Filter Rows: | Edit:

name
Action & Adventure
Comedy
Drama
Horror
Science Fiction
Suspense

  - *ratings*, Rows returned: 4

Lab 3

Nigel Nelson
CS-3860
9/21/21

```
121 •   select * from ratings;
```

Result Grid | Filter Rows:

rating
NR
PG
PG-13
R

- *features*, Rows returned: 372

```
123 •   select * from features;
```

Result Grid | Filter Rows:

recording_id	actor_name
3000	Marlon Brando
3000	Martin Sheen
3000	Robert Duvall
3000	Frederic Forrest
3000	Dennis Hopper
3000	Scott Glenn
3000	Harrison Ford

- *directors*, rows returned: 51

```
119 •   select * from directors;
```

Result Grid | Filter Rows:

name
Alfred Hitchcock
Barry Levinson
Bernard Rose
Brian De Palmas
Bryan Singer

- Part I – Reflection Questions:
  - Why would you select csv- or tab-delimited files over the other?
    - The primary reason to use tab-delimited files over csv files is that tab-delimited files make rows and columns much more distinguishable. This is directly applicable to this class as the majority of the databases used in this class are relational databases. This is where columns hold attributes and each row acts as an entry. Therefore, by using tab-delimited files over csv files it can be much more intuitive to begin to build relational databases.
  - Why would I create the primary key index after the table has been created and the data imported versus defining the primary key in the table definition?

Lab 3

Nigel Nelson
CS-3860
9/21/21

- This is because if primary keys are defined prior to importing the data, constraint checks can fail once that data is attempted to be imported. For example, if the features table uses a combination of the foreign keys recording_id, and actor_name, for the primary key, then if only an actor_name is attempted to be added, with the intention of next adding the recording_id, the query would fail. This is because the query would fail the constraint that each entry must have a unique combination of recording_id and actor_name Another example pertains to foreign keys. If a full recording_id entry is attempted to be imported, but the data contains directors that are not yet contained in the director table, then the import query will fail. This is due to the fact that the director attribute in recording_id has a foreign key constraint that references the primary key attribute, name, in the directors table.

• Part II – Executing SQL Queries

1) Execute select * from Video_Recordings, Video_Categories. Note the cross-product effect of joining two tables. Record the number of rows generated. Do all permutations of Video_Recordings X Video_Categories make sense? Explain.
   a. Query:

```
1 •  SELECT * FROM video_recordings, video_categories;
```

   b. Results:

recording_id	director	title	category	image_name	duration	rating	year_released	price	stock_count	id	name
3000	Francis Ford Coppola	Apocalypse Now	Action & Adventure	apocalypse_now.gif	9180	R	1979	22.99	0.00	5	Suspense
3000	Francis Ford Coppola	Apocalypse Now	Action & Adventure	apocalypse_now.gif	9180	R	1979	22.99	0.00	4	Science Fiction
3000	Francis Ford Coppola	Apocalypse Now	Action & Adventure	apocalypse_now.gif	9180	R	1979	22.99	0.00	3	Horror
3000	Francis Ford Coppola	Apocalypse Now	Action & Adventure	apocalypse_now.gif	9180	R	1979	22.99	0.00	2	Drama
3000	Francis Ford Coppola	Apocalypse Now	Action & Adventure	apocalypse_now.gif	9180	R	1979	22.99	0.00	1	Comedy
3000	Francis Ford Coppola	Apocalypse Now	Action & Adventure	apocalypse_now.gif	9180	R	1979	22.99	0.00	0	Action & Adventure
3001	Michael Bay	Bad Boys	Action & Adventure	badboys.gif	7140	R	1995	15.99	782.00	5	Suspense
3001	Michael Bay	Bad Boys	Action & Adventure	badboys.gif	7140	R	1995	15.99	782.00	4	Science Fiction
3001	Michael Bay	Bad Boys	Action & Adventure	badboys.gif	7140	R	1995	15.99	782.00	3	Horror
3001	Michael Bay	Bad Boys	Action & Adventure	badboys.gif	7140	R	1995	15.99	782.00	2	Drama

   c. Analysis
      i. This query returned 330 rows for this given request. However, all of the returned permutations do not necessarily make much sense. The reason for this is that the query returned the cartesian product of each video category and each video recording, resulting in video recordings being concatenated with mismatched video categories.

Lab 3

Nigel Nelson
CS-3860
9/21/21

2) Execute select * from Video_Recordings vr, Video_Categories vc where vr.category=vc.name. Note the cross-product effect of joining two tables when restricted on the appropriate keys. Record the number of rows generated. Explain the purpose of the join.
    a. Query:

```
select * from Video_Recordings vr, Video_Categories vc where vr.category=vc.name;
```

    b. Results:

3000	Francis Ford Coppola	Apocalypse Now	Action & Adventure	apocalypse_now.gif	9180	R	1979	22.99	0.00	0	Action & Adventure
3001	Michael Bay	Bad Boys	Action & Adventure	badboys.gif	7140	R	1995	15.99	782.00	0	Action & Adventure
3002	Mel Gibson	Braveheart	Action & Adventure	braveheart.gif	10620	R	1995	14.99	582.00	0	Action & Adventure
3003	Mimi Leder	Deep Impact	Action & Adventure	deep_impact.gif	5700	PG-13	1998	11.99	501.00	0	Action & Adventure
3004	J. Robert Wagoner	Disco Godfather	Action & Adventure	disco_godfather.gif	5580	R	1993	10.99	872.00	0	Action & Adventure
3005	Stanley Kubrick	Full Metal Jacket	Action & Adventure	full_metal_jacket.gif	7020	R	1987	16.99	872.00	0	Action & Adventure
3006	Roland Emmerich	Independence Day	Action & Adventure	independence_day.gif	8700	PG-13	1996	16.99	0.00	0	Action & Adventure
3007	John McTiernan	The Hunt for Red October	Action & Adventure	hunt_for_red_october.gif	8100	PG	1990	10.99	618.00	0	Action & Adventure
3008	Michael Bay	The Rock	Action & Adventure	the_rock.gif	8160	R	1996	20.99	514.00	0	Action & Adventure

    c. Analysis
        i. This query returned 55 rows for this given request. The purpose of this “join” is to ensure that the only combinations of video recordings and video categories appear where the video recording’s category matches the video category’s name. This corrects the randomness of the permutations combined by the last query, and ensures only matching permutations are returned.
3) List the number of videos for each video category.
    a. Query:

```
1    select count(category), category
2    from video_recordings vr
3    group by vr.category;
```

    b. Results:

	count(category)	category
▶	10	Action & Adventure
	10	Comedy
	10	Drama
	8	Horror
	9	Science Fiction
	8	Suspense

4) List the number of videos for each video category where the inventory is non-zero.
    a. Query:

Lab 3

Nigel Nelson
CS-3860
9/21/21

```
1    select count(category), category
2    from video_recordings vr
3    where vr.stock_count != 0
4    group by vr.category;
```

b. Results:

count(category)	category
8	Action & Adventure
8	Comedy
9	Drama
6	Horror
8	Science Fiction
6	Suspense

5) For each actor, list the video categories.
   a. Query:

```
79 •  select distinct va.name, group_concat(vr.category)
80    as video_categories
81    from video_actors va inner join
82    video_recordings vr on
83    va.recording_id = vr.recording_id
84    group by va.name
85    order by va.name, vr.category;
```

   b. Results: **335 rows**

name	video_categories
Adam Baldwin	Action & Adventure
Adrian Pasdar	Action & Adventure
Adrienne Corri	Science Fiction
Adrienne King	Horror
Alec Baldwin	Suspense,Action & Adventure
Alec Guinness	Science Fiction
Alfre Woodard	Comedy
Alice Krige	Science Fiction
Amy Irving	Horror
Angela Bassett	Science Fiction

6) Which actors have appeared in movies in different video categories?
   a. Query:

Lab 3

Nigel Nelson
CS-3860
9/21/21

```
1 •  select va.name, count(distinct vr.category) as num_categories
2    from video_actors va
3    inner join video_recordings vr
4    on va.recording_id = vr.recording_id
5    group by va.name
6    having num_categories > 1
7    order by va.name;
```

b. Results: **25 rows**

	name	num_categories
▶	Alec Baldwin	2
	Bill Pullman	2
	Carrie Fisher	2
	David Warner	2
	Demi Moore	2
	Ed Harris	2
	Harrison Ford	2
	Harvey Fierstein	2
	James Rebhorn	2
	Jeff Goldblum	2

7) Which actors have not appeared in a comedy?

a. Query:

```
94 •  select va.name
95    from video_actors va
96    where va.name
97    NOT IN(
98    select va.name
99    from video_actors va inner join
100   video_recordings vr on
101   va.recording_id = vr.recording_id
102   where vr.category = 'Comedy'
103   group by va.name
104   )
105   group by name
106   order by name;
```

b. Results: **265 rows**

Lab 3

Nigel Nelson
CS-3860
9/21/21

name
Adam Baldwin
Adrian Pasdar
Adrienne Corri
Adrienne King
Alec Baldwin
Alec Guinness
Alice Krige
Amy Irving
Angela Bassett
Anna Katerina

8) Which actors have appeared in comedy and action adventure movies?
   a. Query:

```
94  select va.name
95  from video_actors va inner join
96  video_recordings vr on
97  va.recording_id = vr.recording_id
98  where vr.category = 'Comedy'
99  and va.name IN(
100 select va.name
101 from video_actors va inner join
102 video_recordings vr on
103 va.recording_id = vr.recording_id
104 where vr.category = 'Action & Adventure'
105 group by va.name
106 )
107 group by name
108 order by name;
```

   b. Results:

name
Harvey Fierstein
Jerry Jones
Lady Reed
Rudy Ray Moore