COMP3013 Tutorial02

Outlines

- Lecture review
- Lab review

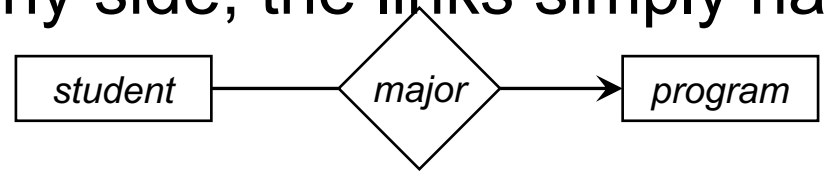
LECTURE REVIEW

Relationship Constraints

- To express the answers, ER diagrams have ***constraints*** on relationship sets.
- Constraints indicate some conditions under the context of the modeled problem.
- Two types of constraints
  - ***Cardinality constraints*** *(one-one/many-many/many-one, ….)*
  - ***Participation constraints*** *(for all/ for some)*
- “Cardinality” is a term from set theory. It is the number of items in a set.

Cardinality Constraints

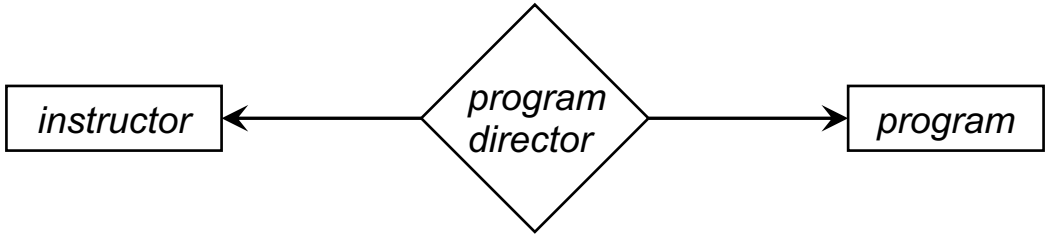
- To express the cardinality constraints, ER diagrams use an ***arrow*** (→) pointing to the one side.
- For the many side, the links simply have no arrow (–).



- A ***one-to-many*** relationship is the reverse of many-to-one.

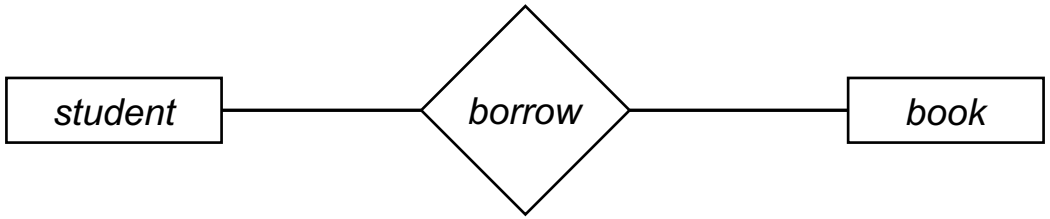
Cardinality Constraints

- ***One-to-one*** relationship:
  - One entity from one entity set is associated with at most one entity from the other entity set and vice versa.
  - For example, one instructor can be the program director of at most one program, and one program has at most one program director.



Cardinality Constraints

- ***Many-to-many*** relationship:
  - One entity from one entity set can be associated with multiple entities from the other entity set and vice versa.
  - For example, one student can borrow multiple books, and one book can be borrowed by multiple students.

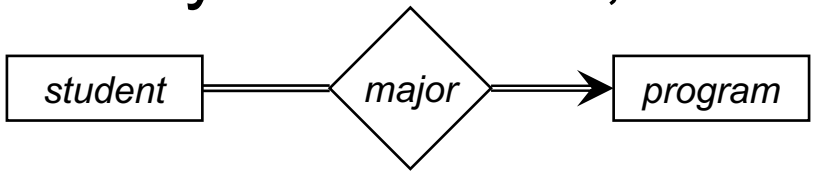


Participation Constraints

- To express the answers to the other two questions:
  - Does every student have a major?
  - Is every program the major for some students?
- ER diagrams have ***participation constraints***.
- ***Total participation***:
  - Every entity participates a relationship.
  - The link is a ***double line*** (=).
- ***Partial participation***:
  - There are some entities do not participate any relationship.
  - The link is a ***single line*** (–).

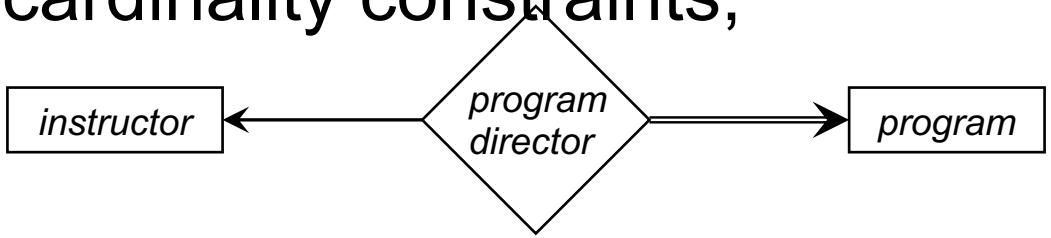
Participation Constraints

- Back to the example,
  - Every student has a major. (Students totally participate in the relationship set.)
  - Every program is the major for some students. (Programs also totally participate.)
- Combining the cardinality constraints,



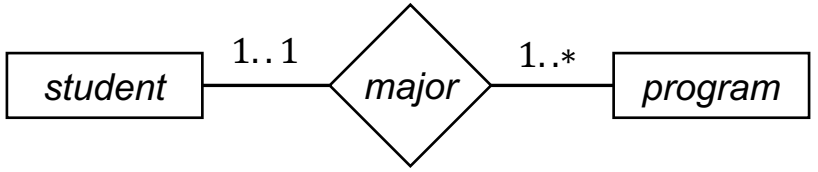
Participation Constraints

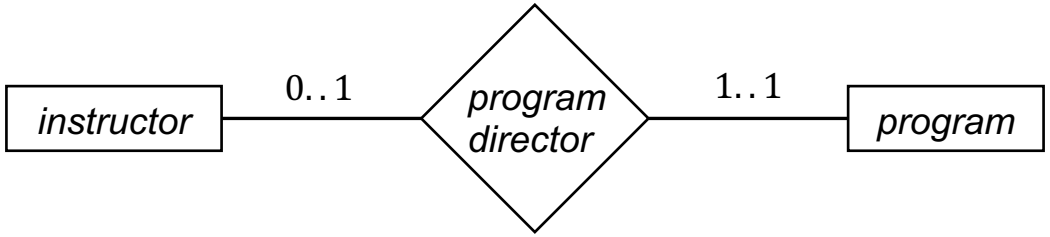
- Consider another example, the relationship set “program director”.
  - It’s possible that an instructors is not a program director for any program.
  - But every program has a program director.
- Thus, “instructor” is on the partial side, while “program” is on the total side.
- Combining the cardinality constraints,



Alternative Notations

- Instead of using lines and arrows, relationship constraints can be expressed by numbers in the form $a..b$ besides links.
- a is the minimum number of entities associated with.
- b is the maximum number of entities associated with.
- $*$ means multiple.
- For example,

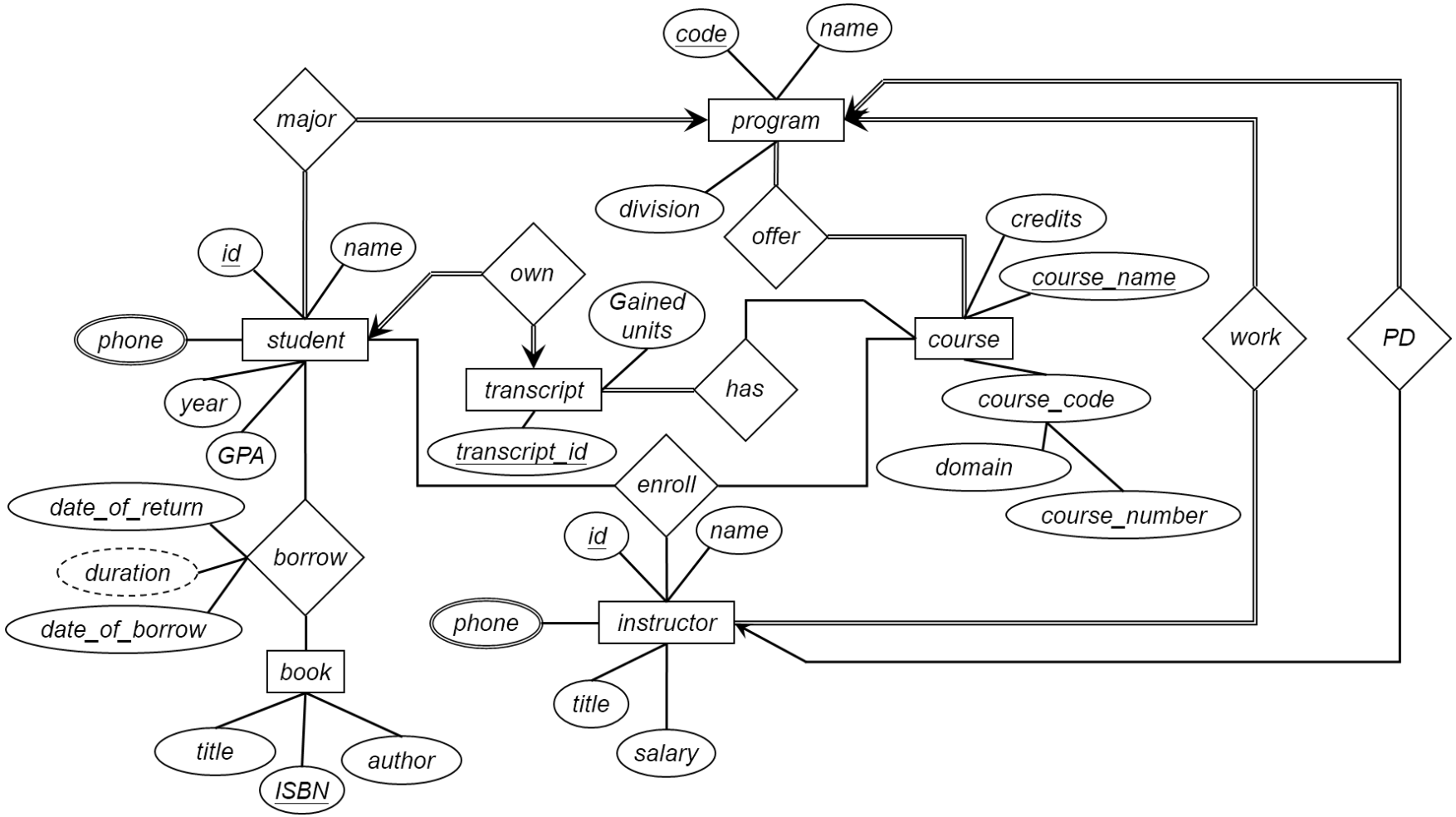




Non-binary relationships with constrains is complex.

- Not suggested to use constrains on non-binary relationships(ambiguous)
- See lecture: convert non-binary to binary

Lec 03 exercise answer is on lec 4 slides.



LAB REVIEW

Cross Table

- The crossing of tables is the cartesian product of them

Cross Table

Cartesian Product

- Cartesian product of two sets S_1 and S_2 (denoted $S_1 \times S_2$)
  - $S_1 \times S_2 = \{(a, b) | a \in S_1 \wedge b \in S_2\}$
  - E.g.,
    - $S_1 = \{1, 2\}, S_2 = \{a, b, c\}$, then $S_1 \times S_2 = \{(1, a), (1, b), (1, c), (2, a), (2, b), (2, c)\}$

Question 1: If $|S_1| = m$ and $|S_2| = n$, then $|S_1 \times S_2| = m \times n$

Question 2: $S_1 \times S_2 \neq S_2 \times S_1$

city

city_id	city	country_id
1	Beijing	1
2	Tokyo	2

×

country

country_id	country
1	China
2	Japan

=

city. country.

city_id	city	country_id	country_id	country
1	Beijing	1	1	China
2	Tokyo	2	1	China
1	Beijing	1	2	Japan
2	Tokyo	2	2	Japan

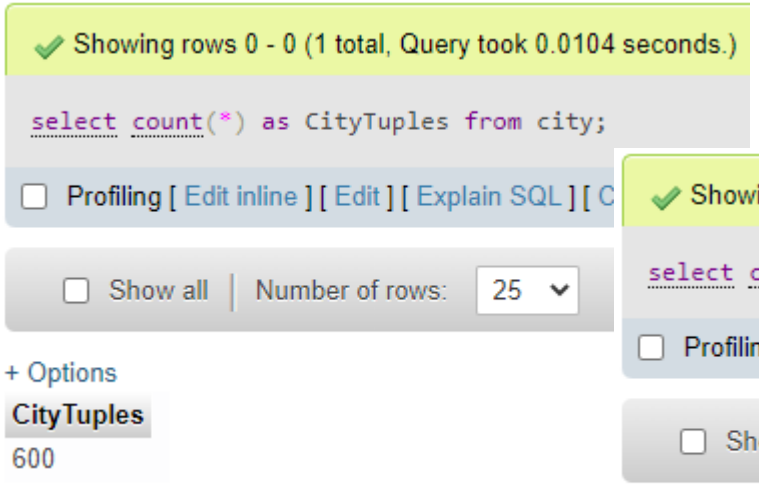
Cross Table

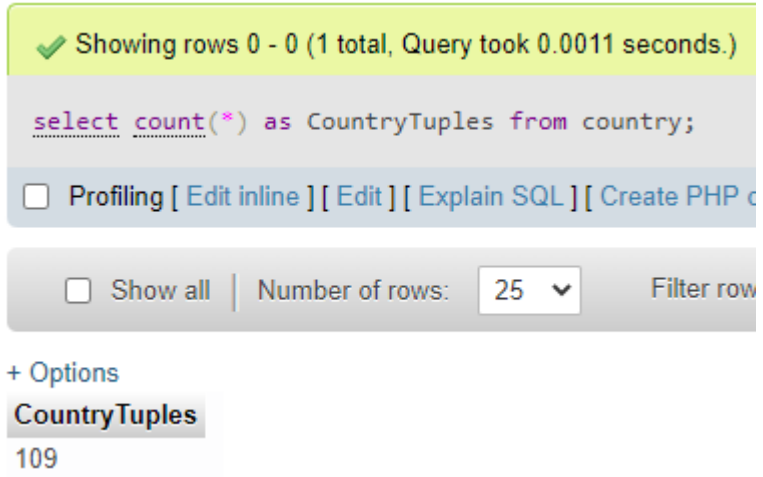
Cartesian Product

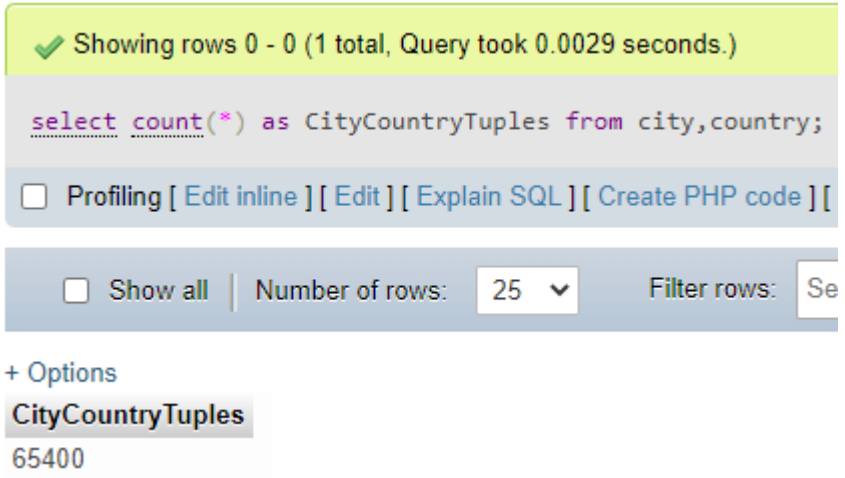
- Cartesian product of two sets S_1 and S_2 (denoted $S_1 \times S_2$)
  - $S_1 \times S_2 = \{(a, b) | a \in S_1 \wedge b \in S_2\}$
  - E.g.,
    - $S_1 = \{1, 2\}, S_2 = \{a, b, c\}$, then $S_1 \times S_2 = \{(1, a), (1, b), (1, c), (2, a), (2, b), (2, c)\}$

Question 1: If $|S_1| = m$ and $|S_2| = n$, then $|S_1 \times S_2| = m \times n$

Question 2: $S_1 \times S_2 \stackrel{?}{=} S_2 \times S_1$





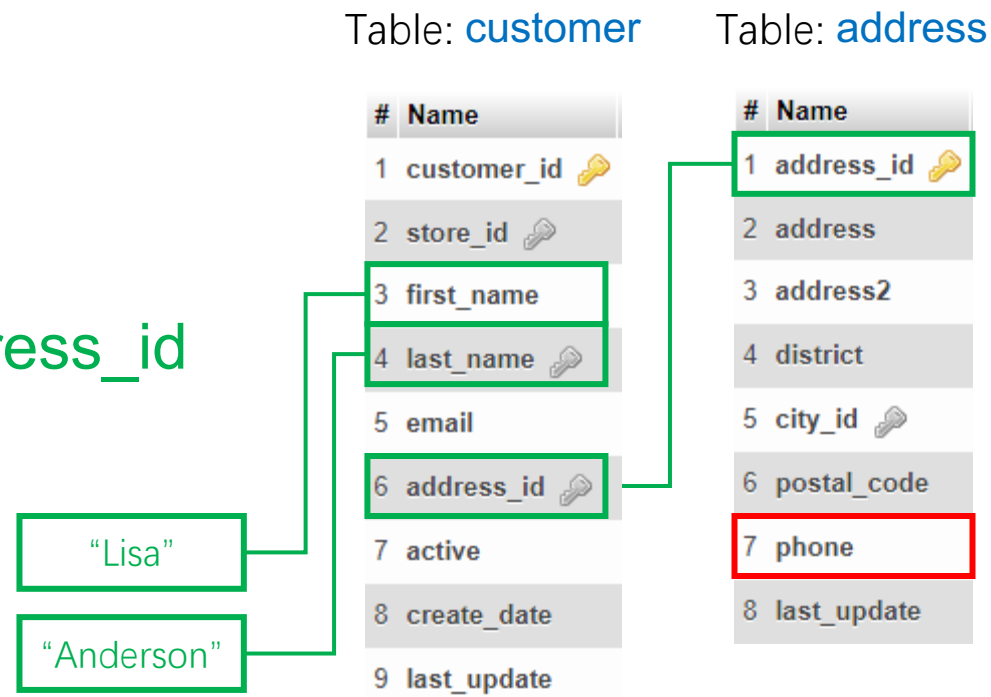


$|\text{city}| = 600, |\text{country}| = 109$
$|\text{city} \times \text{country}| = 600 \times 109 = 65400$

Cross Table

- Find the phone number of the customer Lisa Anderson.

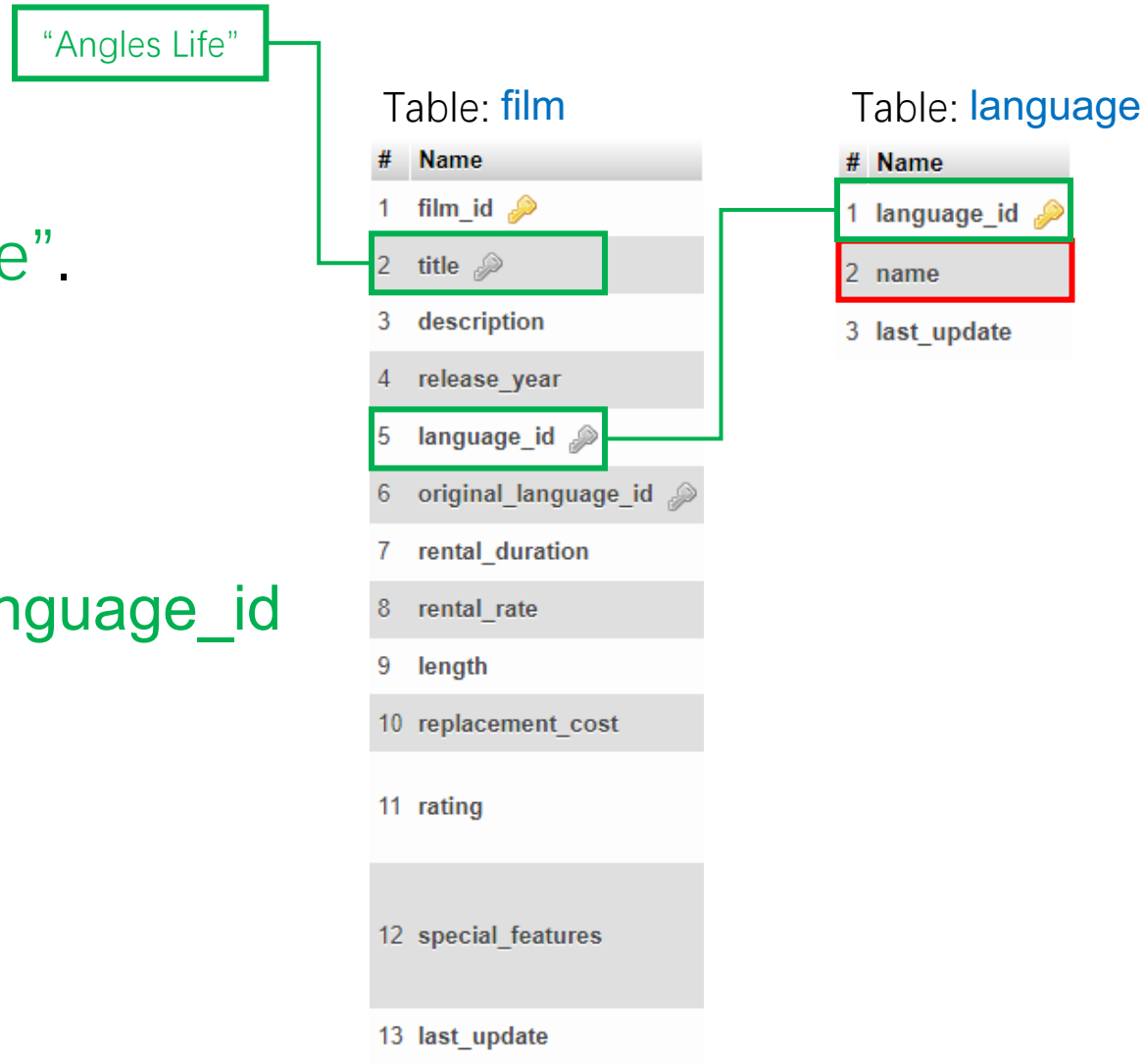
1. Table: customer, address
2. Predicate: first_name = “Lisa”
   last_name = “Anderson”
   customer.address_id = address.address_id
3. Attribute: phone



Cross Table

- Find the language of the film “Angles Life”.

1. Table: film, language
2. Predicate: title = “Angles Life”
   film.language_id = language.language_id
3. Attribute: name



Cross Table

Table: actor

#	Name
1	actor_id
2	first_name
3	last_name
4	last_update

Table: film_actor

#	Name
1	actor_id
2	film_id
3	last_update

Table: film

#	Name
1	film_id
2	title
3	description
4	release_year
5	language_id
6	original_language_id
7	rental_duration
8	rental_rate
9	length
10	replacement_cost
11	rating
12	special_features
13	last_update

```
1 select actor_id, first_name, last_name, title
2 from actor, film_actor, film
3 where first_name = "Angela"
4 and actor.actor_id = film_actor.actor_id
5 and film_actor.film_id = film.film_id;
```

Error

SQL query: Copy

```
select actor_id, first_name, last_name, title
from actor, film_actor, film
where first_name = "Angela"
and actor.actor_id = film_actor.actor_id
and film_actor.film_id = film.film_id LIMIT 0, 25
```

MySQL said:

```
#1052 - Column 'actor_id' in field list is ambiguous
```

Cross Table

Table: actor

#	Name
1	actor_id
2	first_name
3	last_name
4	last_update

Table: film_actor

#	Name
1	actor_id
2	film_id
3	last_update

Table: film

#	Name
1	film_id
2	title
3	description
4	release_year
5	language_id
6	original_language_id
7	rental_duration
8	rental_rate
9	length
10	replacement_cost
11	rating
12	special_features
13	last_update

Showing rows 0 - 24 (69 total, Query took 0.0020 seconds.)

```
SELECT title FROM actor, film_actor, film WHERE first_name = "Angela" AND actor.actor_id = film_actor.actor_id AND film_actor.film_id = film.film_id;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

1 > >> | Show all | Number of rows: 25 Filter rows: Search this table

+ Options

title
ARMAGEDDON LOST
AUTUMN CROW
BRIDE INTRIGUE
BULWORTH COMMANDMENTS
CANDLES GRAPES
CASSIDY WYOMING
CLONES PINOCCHIO
ELEMENT FREDDY
FATAL HAUNTED

Cross Table

Table: actor

#	Name
1	actor_id
2	first_name
3	last_name
4	last_update

Table: film_actor

#	Name
1	actor_id
2	film_id
3	last_update

Table: film

#	Name
1	film_id
2	title
3	description
4	release_year
5	language_id
6	original_language_id
7	rental_duration
8	rental_rate
9	length
10	replacement_cost

Showing rows 0 - 24 (69 total, Query took 0.0026 seconds.)

```
SELECT actor.actor_id, first_name, last_name, title FROM actor, film_actor, film WHERE first_name = "Angela" AND actor.actor_id = film_actor.actor_id AND film_actor.film_id = film.film_id;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

1 > >> | Show all | Number

+ Options

actor_id	first_name	last_name	title
65	ANGELA	HUDSON	ARMAGEDD
65	ANGELA	HUDSON	AUTUMN CR
65	ANGELA	HUDSON	BRIDE INTRI
65	ANGELA	HUDSON	BULWORTH

Showing rows 50 - 68 (69 total, Query took 0.0024 seconds.)

```
SELECT actor.actor_id, first_name, last_name, title FROM actor, film_actor, film WHERE first_name = "Angela" AND actor.actor_id = film_actor.actor_id AND film_actor.film_id = film.film_id;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

<< < 3 | Show all | Number of rows: 25 | Filter rows: Search this table

+ Options

actor_id	first_name	last_name	title
144	ANGELA	WITHERSPOON	EDGE KISSING
144	ANGELA	WITHERSPOON	EVOLUTION ALTER
144	ANGELA	WITHERSPOON	EXORCIST STING
144	ANGELA	WITHERSPOON	FIDDLER LOST

Cross Table – self times

Table: address as a1

#	Name
1	address_id
2	address
3	address2
4	district
5	city_id
6	postal_code
7	phone
8	last_update

Table: address as a2

#	Name
1	address_id
2	address
3	address2
4	district
5	city_id
6	postal_code
7	phone
8	last_update

```
SELECT a1.city_id
FROM address AS a1, address AS a2
WHERE a1.city_id = a2.city_id AND
        a1.address_id <> a2.address_id
```

- The two tables are renamed by "**AS**", to avoid ambiguous.
- If two tuples agree on **city_id** but have different **address_id**, this **city_id** has multiple addresses.
- <> is not equal.

Cross Table – self times

- Find the name (first and last) of actors who have a same first name with other actors.
  1. Table: actor as a1, actor as a2
  2. Predicate: a1.first_name = a2.first_name
     a1.actor_id <> a2.actor_id

     Or

     a1.actor_id <> a2.actor_id
     a1.first_name = a2.first_name

     (Think about the order of predicate....)
  3. Attribute: a1.actor_id, a1.first_name, a1.last_name, a2.actor_id

Table: actor as a1

#	Name
1	actor_id
2	first_name
3	last_name
4	last_update

Table: actor as a2

#	Name
1	actor_id
2	first_name
3	last_name
4	last_update

Showing rows 0 - 7 (8 total, Query took 0.0057 seconds.)

```
SELECT a1.city_id FROM address AS a1, address AS a2 WHERE a1.city_id = a2.city_id AND a1.address_id <> a2.address_id;
```

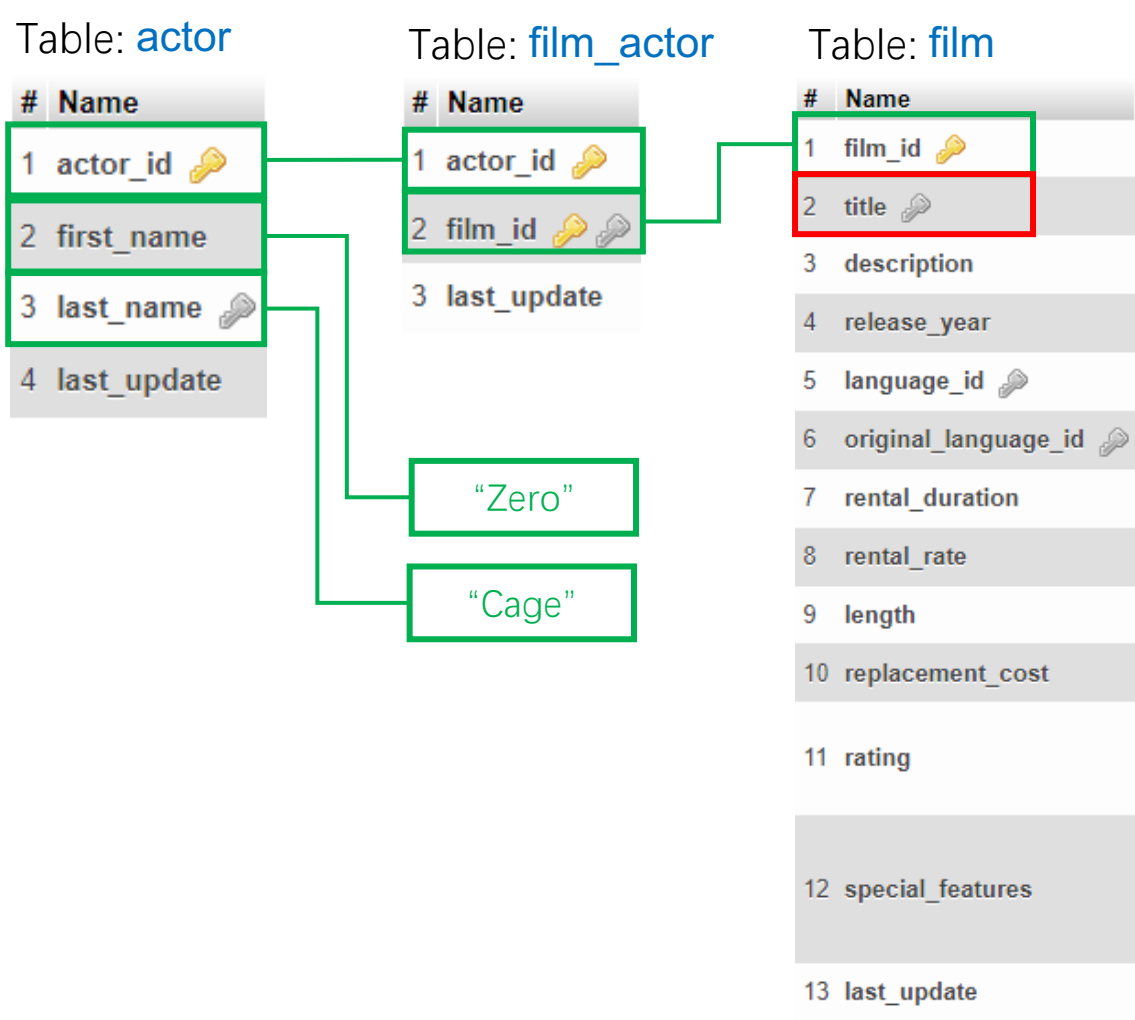
Showing rows 0 - 7 (8 total, Query took 0.0026 seconds.)

```
SELECT a1.city_id FROM address AS a1, address AS a2 WHERE a1.address_id <> a2.address_id AND a1.city_id = a2.city_id;
```

Cross Table - Exercise

1. Find the films (name) played by Zero Cage.
   1. Table: film, film_actor, actor
   2. Predicate: film.film_id = film_actor.film_id
      film_actor.actor_id = actor.actor_id
      first_name = “Zero”
      last_name = “Cage”
   3. Attribute: title

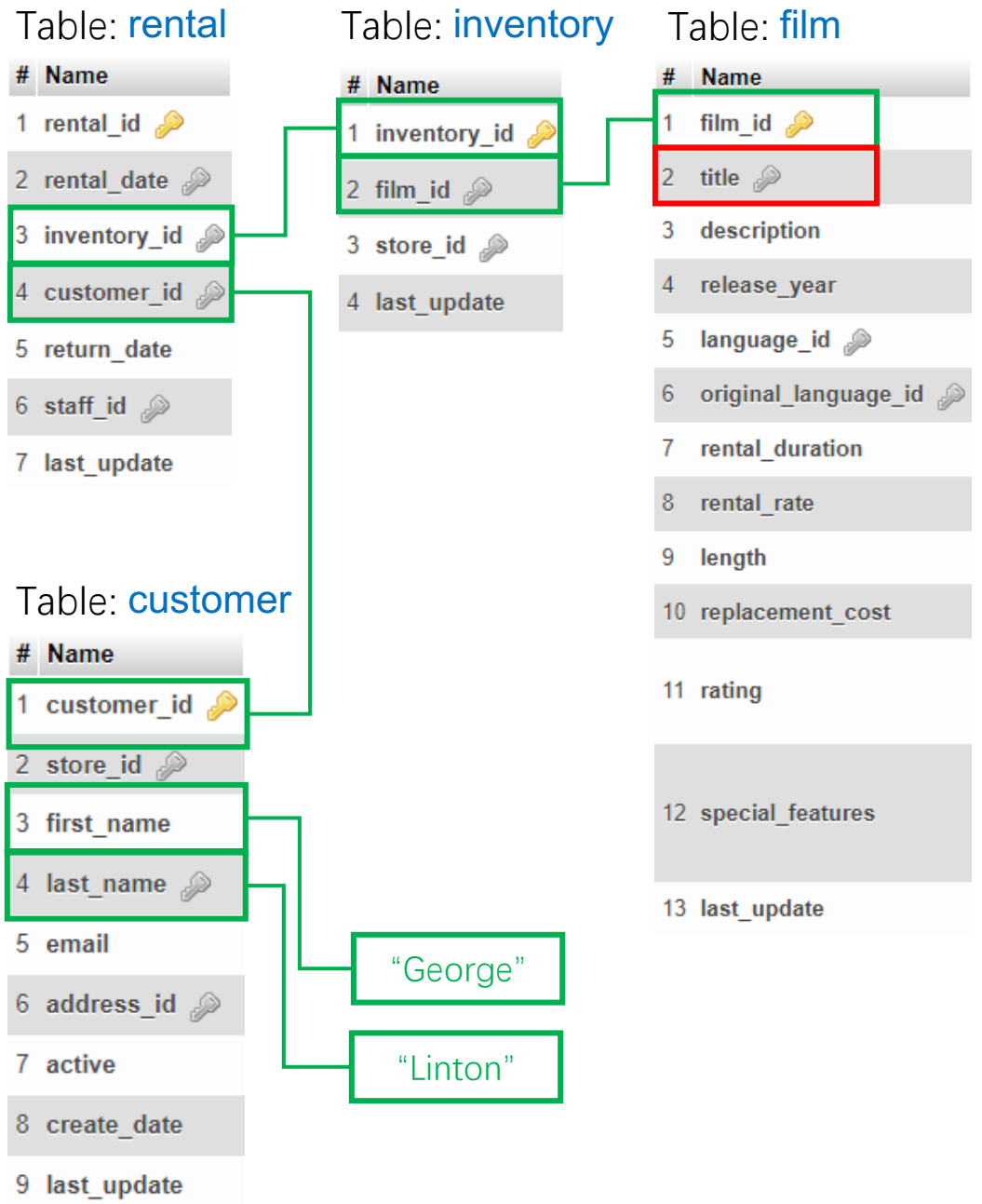
SELECT title, first_name, last_name
FROM film, film_actor, actor
WHERE film.film_id = film_actor.film_id
AND film_actor.actor_id = actor.actor_id
AND first_name = “Zero”
AND last_name = “Cage”



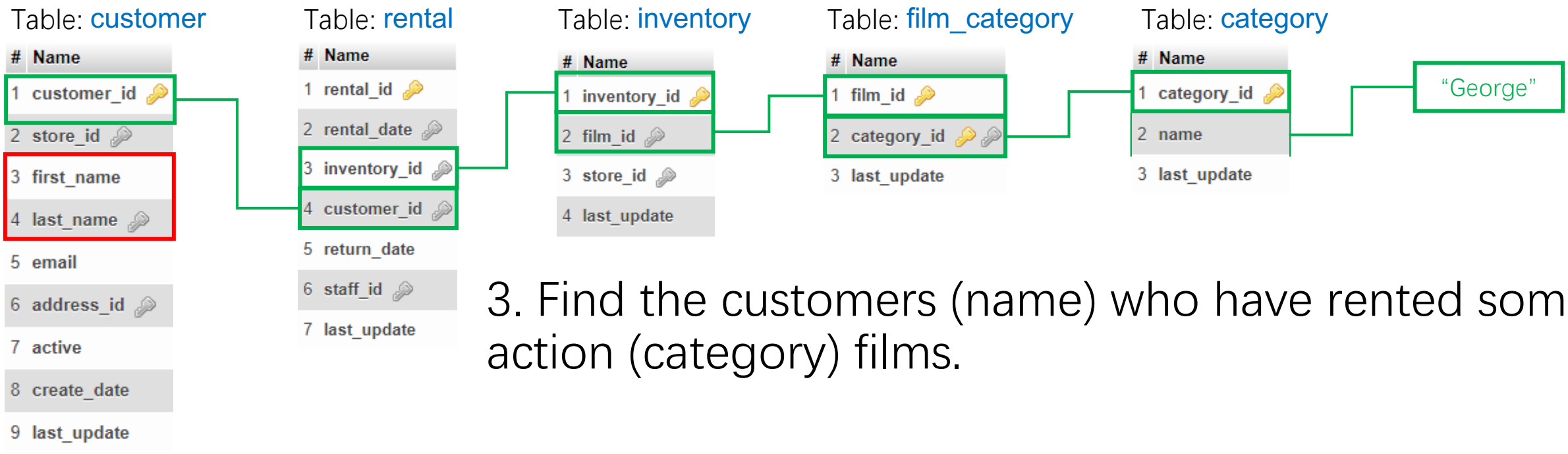
Cross Table - Exercise

2. Find the films (name) rented by George Linton.
    1. Table: film, inventory, rental, customer
    2. Predicate: film.film_id = inventory.film_id
       inventory. inventory _id = rental.inventory_id
       rental.customer_id = customer. Customer_id
       first_name = “George”
       last_name = “Linton”
    3. Attribute: title

SELECT title, first_name, last_name
FROM film, inventory, rental, customer
WHERE film.film_id = inventory.film_id
AND inventory. inventory _id = rental.inventory_id
AND rental.customer_id = customer. Customer_id
AND first_name = “George”
AND last_name = “Linton”

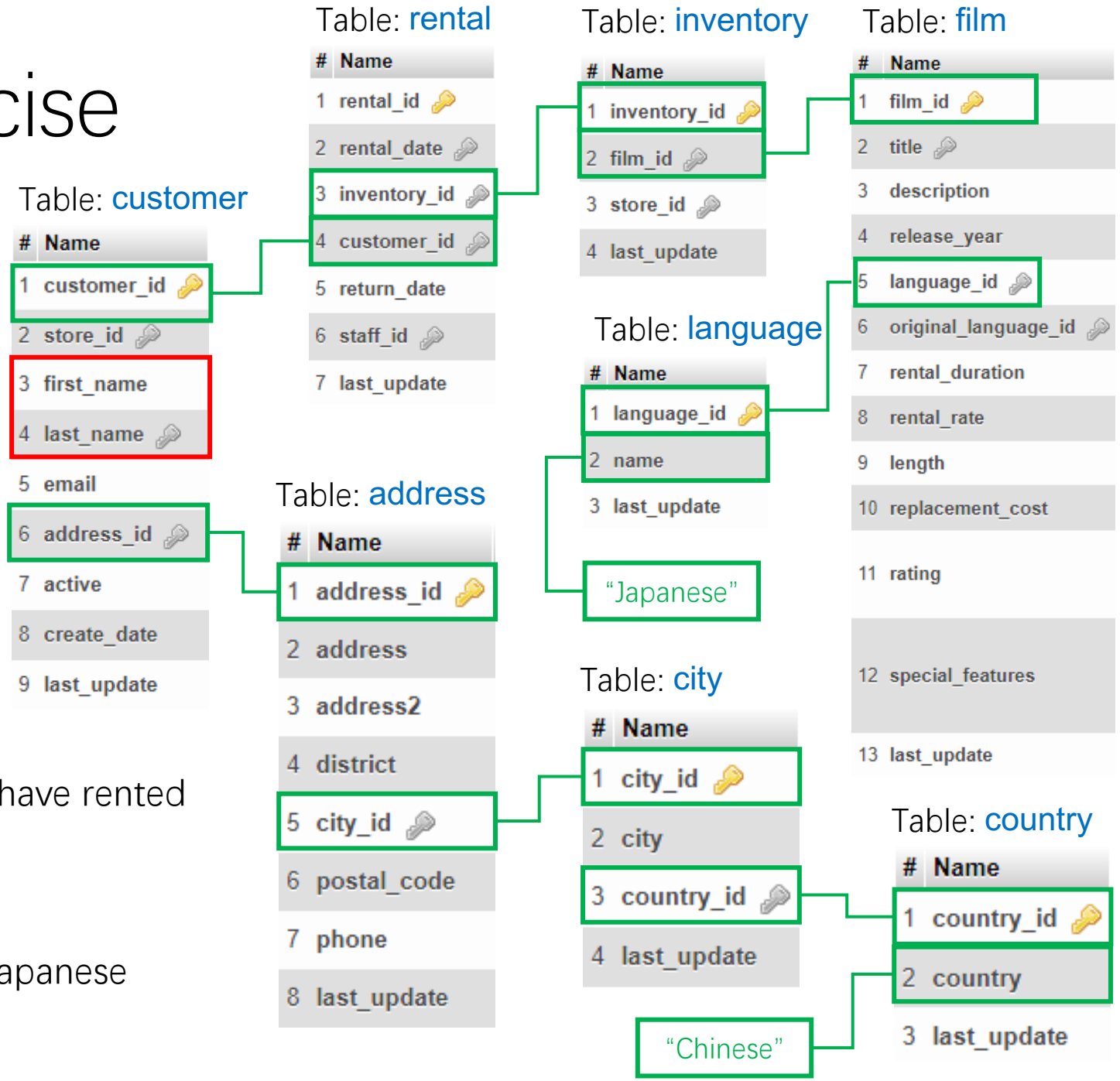


Cross Table - Exercise



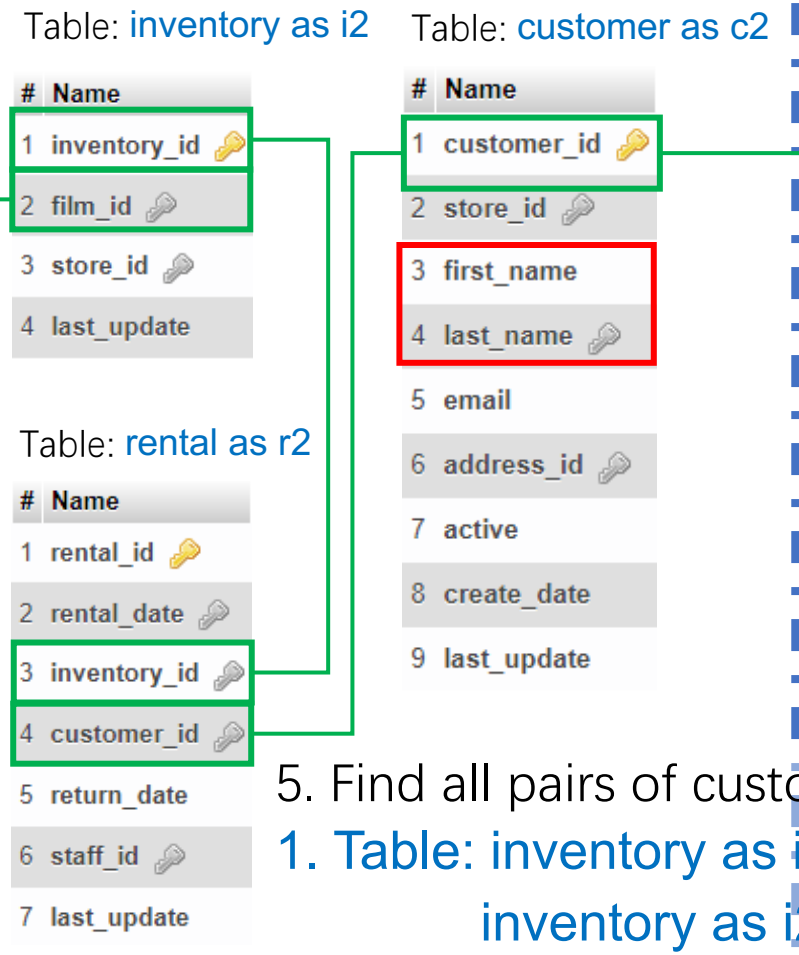
3. Find the customers (name) who have rented some action (category) films.

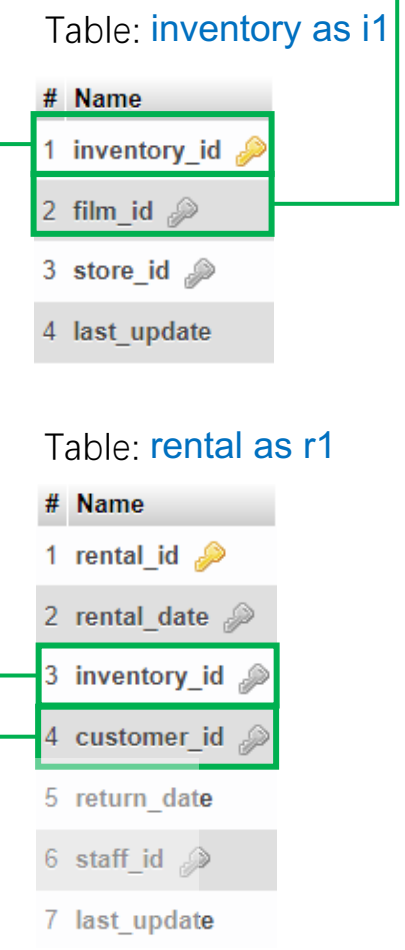
Cross Table - Exercise



4. Find the customers who live in China and have rented some Japanese films.

Find the customers who live in China
Find the customers who have rented some Japanese films.





5. Find all pairs of customers (name) who have rented a same film.

1. Table: inventory as i1, rental as r1, customer as c1,
inventory as i2, rental as r2, customer as c2

2. Predicate: i1. inventory _id = r1.inventory_id, r1.customer_id = c1. customer_id
i2. inventory _id = r2.inventory_id, r2.customer_id = c2. customer_id
i1.film_id = i2.film_id
c1.customer_id <> c2.customer_id

3. Attribute: c1.first_name, c1.last_name, c2.first_name, c2.last_name

Cross Table - Exercise

6. Find the actors who have played a same film with Bolger (the last name of an actor)
   1. Table: actor as a1, film_actor as fa1
      actor as a2, film_actor as fa2
   1. Predicate: fa1.actor_id = a1.actor_id
      fa2.actor_id = a2.actor_id
      fa1.film_id = fa2.film_id
      a2.last_name = “Bolger”
   3. Attribute: a1.first_name, a1.last_name