</talentlabs>

8.6 Reading

Querying Tables with Relationships

Study Instructions

- In this reading materials, we have provided example code for querying tables with relationships for your reference. There are no new SQL concepts or SQL keywords. These examples are just leveraging SQL knowledge that you already know.

- Please read through the examples to make sure you understand each line of code. You will be using these techniques in your lab assignments.

</talentlabs>

</talentlabs>

AGENDA

- Principles
- One-to-one relationships
- One-to-many relationships
- Many-to-many relationships
- Conclusions

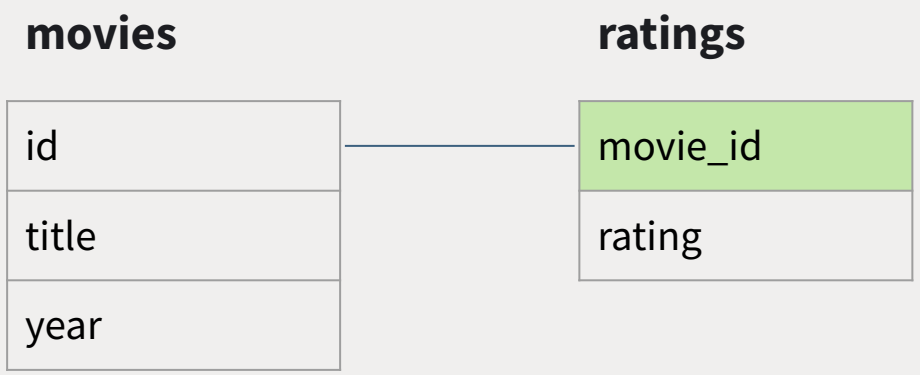
</talentlabs>

Principles

</talentlabs>

Querying Tables with Relationships

- When you want to query tables with relationships, it's all about joining.
- There are no special techniques, all you need to do is to identify the key relationships between tables (i.e. what is the foreign key linking the two tables)
- After identifying the key, then you just need to join the relevant tables together with the foreign keys.



 </talentlabs>

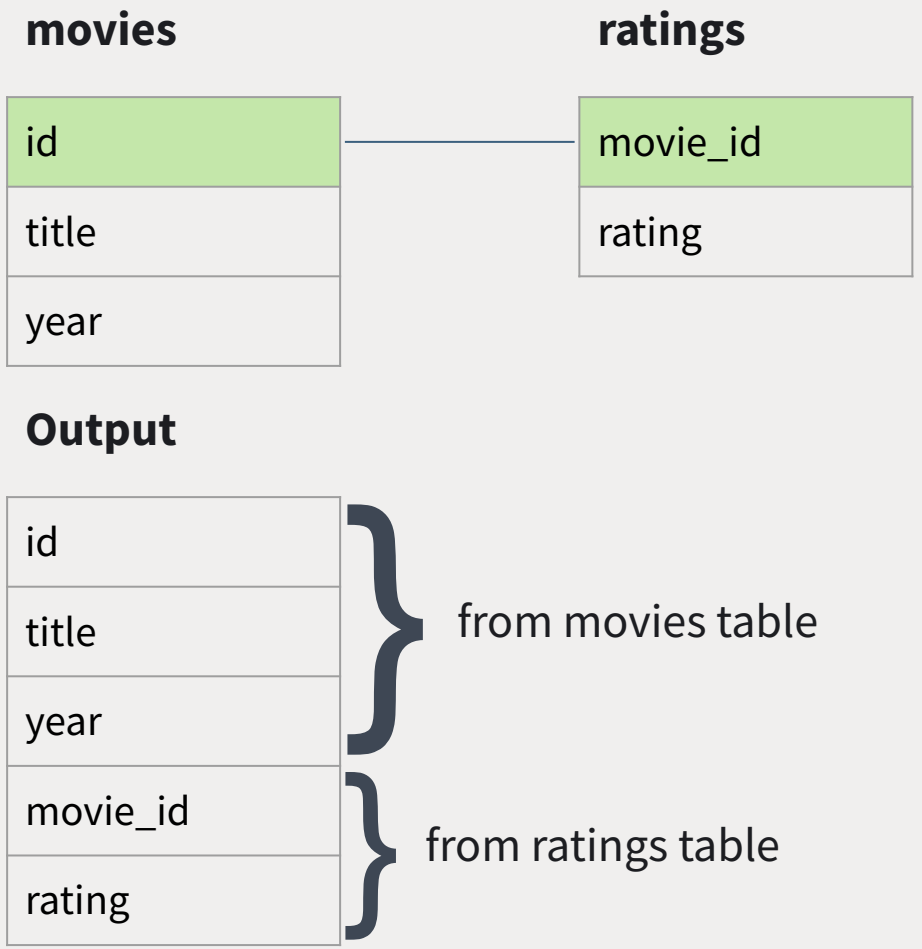
One-to-one Relationships

</talentlabs>

Joining One-to-one Relationships

- Movies and Ratings are in one-to-one relationship
- The key linkage between them is movies.id and ratings.movie_id (Foreign Key)
- So we only need to join the 2 tables on these two keys

```
SELECT *
FROM
    movies JOIN ratings
        ON movies.id = ratings.movie_id
```



</talentlabs>

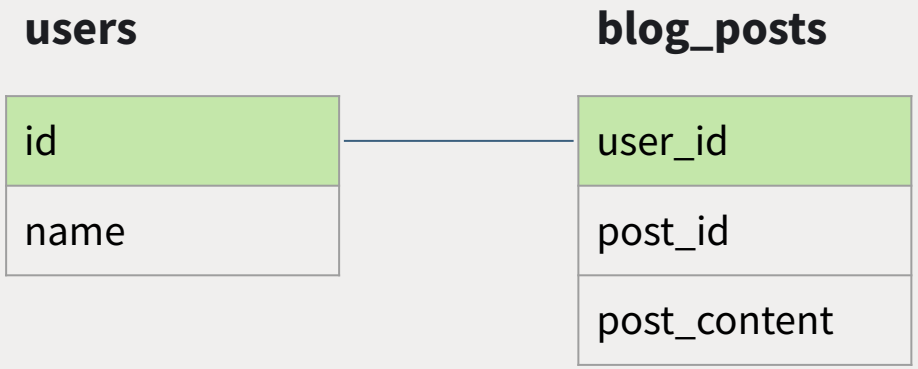
One-to-many Relationships

</talentlabs>

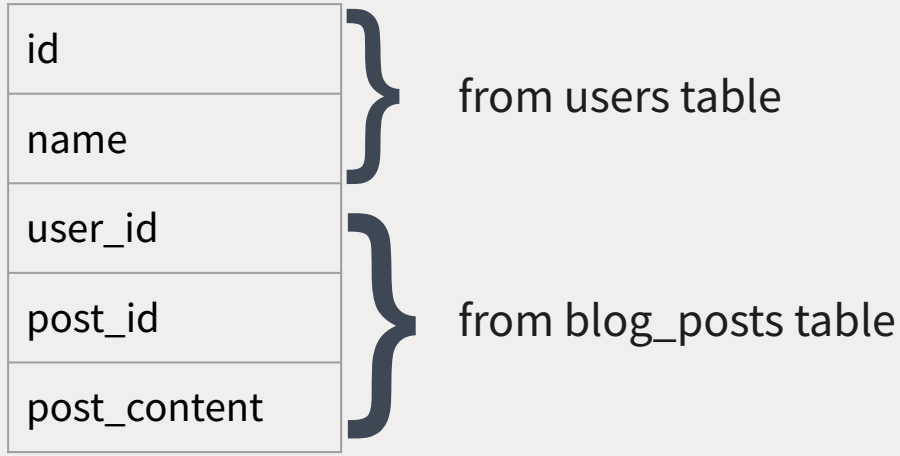
Joining One-to-Many Relationships

- Let's say you are running a blogging website
- Users table and blog_posts table are in one-to-many relationships, i.e. each users can have multiple blog_posts
- For key relationships, the blog_posts.user_id (foreign key) is linked to users.id
- When you are joining the tables, it's similar to one-to-one relationship, you only need to join on the foreign keys

```
SELECT *
FROM
    blog_posts join users
        ON blog_posts.user_id = users.id
```



Output



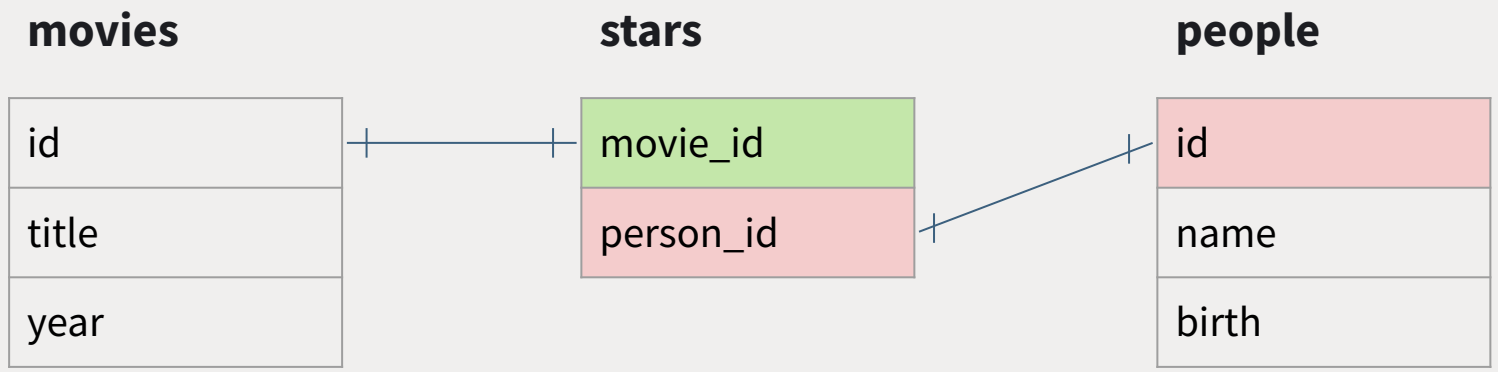
</talentlabs>

Many-to-many Relationships

</talentlabs>

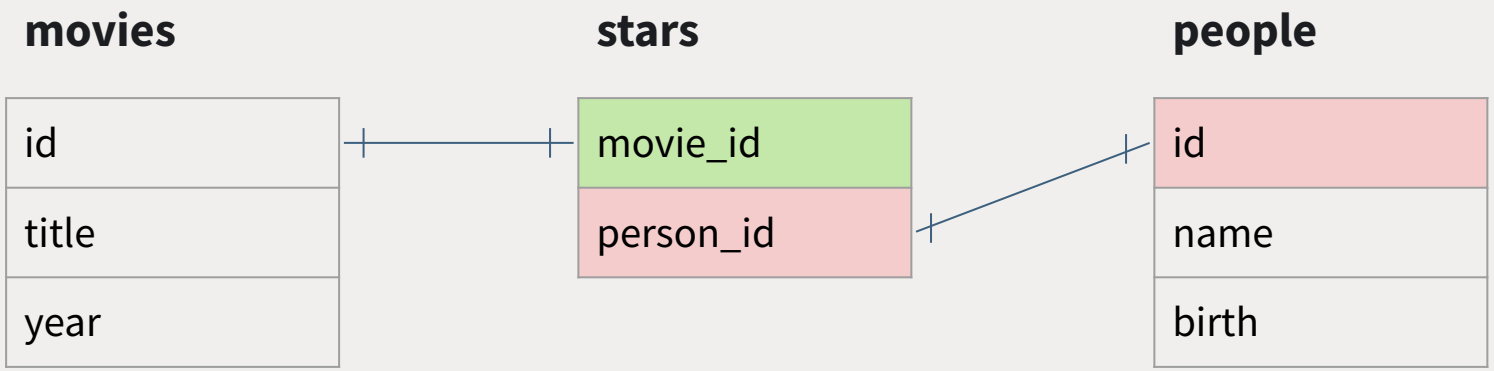
Many-to-Many Relationships

- The relationship between “movies - stars - people” is a many-to-many relationship
- Each movie would have multiple actors, and each person would be acting in multiple movies
- There is an intermediate “relationship” table to store the mapping between movies and people (the “stars” table)



Challenge of Joining Many-to-Many Relationships

- SQL Joins allows joining 2 tables at a time only
- In many-to-many relationships, there are 3 tables, so we need to do 2 joins
- We will first join the “movies with stars”, then join the output with the “people” table



</talentlabs>

Joining Many-to-Many Relationships

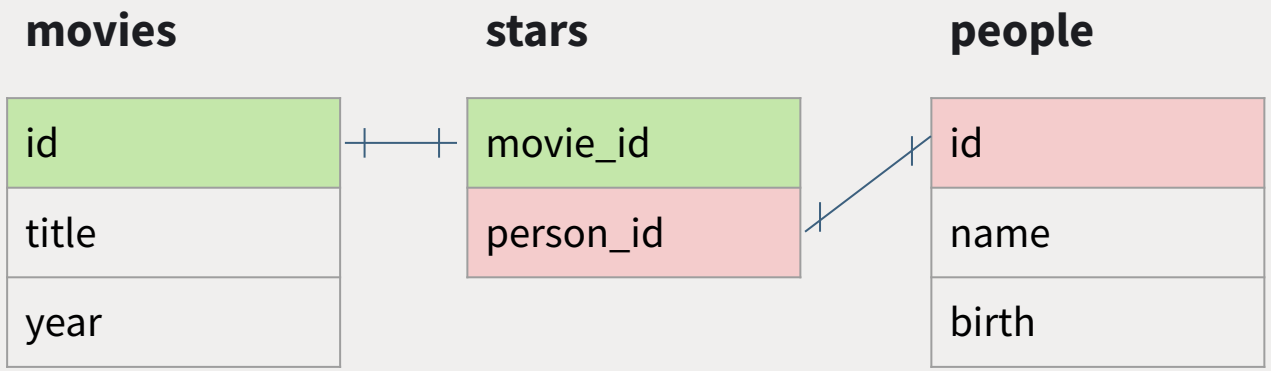
First Join

Second Join

```
SELECT *
FROM
    (movies join stars
        ON movies.id = stars.movie_id) a
         join people on people.id = a.person_id
```

Output

id	from movies table
title	
year	
movie_id	from stars table
person_id	
id	from people table
name	
birth	



Conclusions

</talentlabs>

Conclusions

- Table relationships actually **do not matter** a lot when you are pulling the data. The only important thing the key linkages.

- All you need to do is to identify tables that contains your data (2 tables or 3 tables), and identify the foreign key linkage between the tables. Then you can just use “join” to combine multiple tables together.