Lecture Notes: Junction Tables in Relational Databases

1. Introduction to Junction Tables

In relational databases, junction tables, also known as join tables or bridge tables, are used to manage many-to-many relationships between entities. These entities could be anything from customers and products in an e-commerce system to students and courses in an educational platform. Junction tables play a crucial role in resolving complex relationships and ensuring data integrity.

Understanding Many-to-Many Relationships

In a many-to-many relationship, entities from one table can be associated with multiple entities from another table, and vice versa. For instance, in a film and actor scenario, a film can have multiple actors, and an actor can participate in multiple films. This creates a many-to-many relationship between the two entities.

Example Scenario

Let's consider two tables: `Film` and `Actor`. The `Film` table contains information about films such as `film_id`, `title`, `genre`, etc. The `Actor` table contains information about actors such as `actor_id`, `name`, `dob`, etc.

Film Table

FilmID	Title	ReleaseDate	Genre
1	The Shawshank Redemption	1994-09-22	Drama
2	The Godfather	1972-03-24	Crime
3	The Dark Knight	2008-07-18	Action
4	Forrest Gump	1994-07-06	Drama
5	Inception	2010-07-16	Sci-Fi

Actor Table

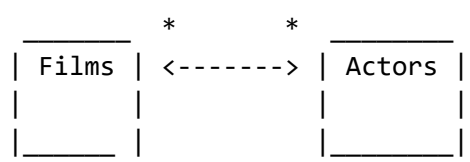
ActorID	FirstName	LastName	DateOfBirth
1	Morgan	Freeman	1937-06-01
2	Tim	Robbins	1958-10-16
3	Marlon	Brando	1924-04-03
4	Al	Pacino	1940-04-25
5	Christian	Bale	1974-01-30
6	Tom	Hanks	1956-07-09

Relating these two tables

Let's think about the relationship between an `Actor` and a `Film`:

- A film can have many actors
- An actor can star in many films.

Therefore we have a many-to-many relationship.



However, tables can only be related in a database as a one-to-one or one-to-many.

A many-to-many relationship is not implementable in a relational Database!

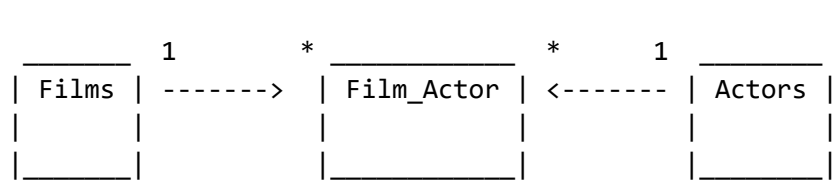
Thus we are going to create what is called a **Juction Table**. [Wikipedia Link]

2. Creating Junction Table

To manage the many-to-many relationship between films and actors, we create a junction table named `Film_Actor`. This table will have two foreign key columns: `FilmID` and `ActorID`, referencing the primary keys of the `Film` and `Actor` tables respectively.

Film_Actor Junction Table:

FilmID	ActorID
101	201
101	202
102	202
103	201
...	...



This now established a many-to-many relationship by using two one-to-many relationships via the **Junction Table**.

If you look at the above table, you can imagine `FilmID` 101 is for **Ryan Gosling** and `ActorID` 201 and 202 are Barbie and La La Land respectively. Therefore these two entries connect Ryan Gosling with many movies.

3. Querying Junction Table

To retrieve information about films and their associated actors, we perform a JOIN operation between the `Film` table and the `Film_Actor` table, and another JOIN operation between the `Actor` table and the `Film_Actor` table. This allows us to link films with their respective actors.

Example Query

```
SELECT Film.title, Actor.name
FROM Film
JOIN Film_Actor ON Film.FilmID = Film_Actor.FilmID
JOIN Actor ON Film_Actor.ActorID = Actor.ActorID;
```

Conclusion

Junction tables are fundamental in managing many-to-many relationships in relational databases. They provide a way to effectively link entities and maintain data integrity. In the context of films and actors, a junction table named `Film_Actor` helps in associating films with their respective actors, facilitating efficient data retrieval and management.