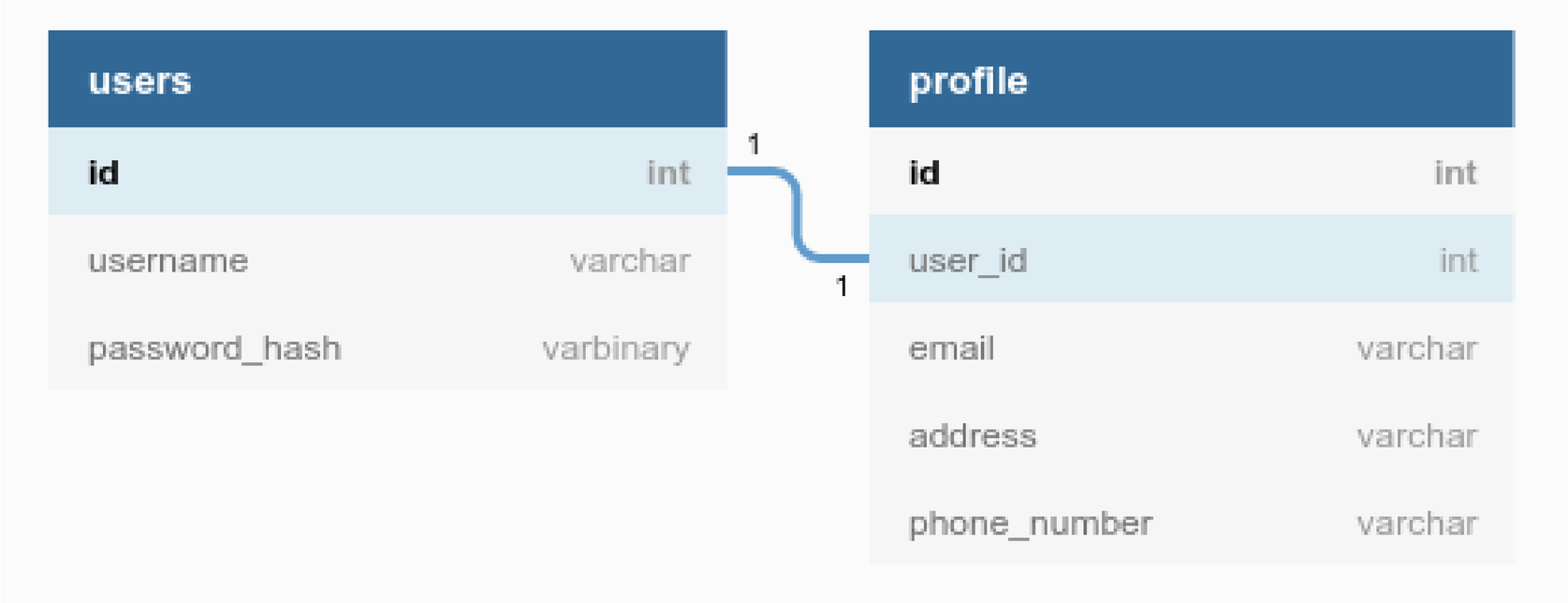
Joins and Relationships in Databases

Agenda

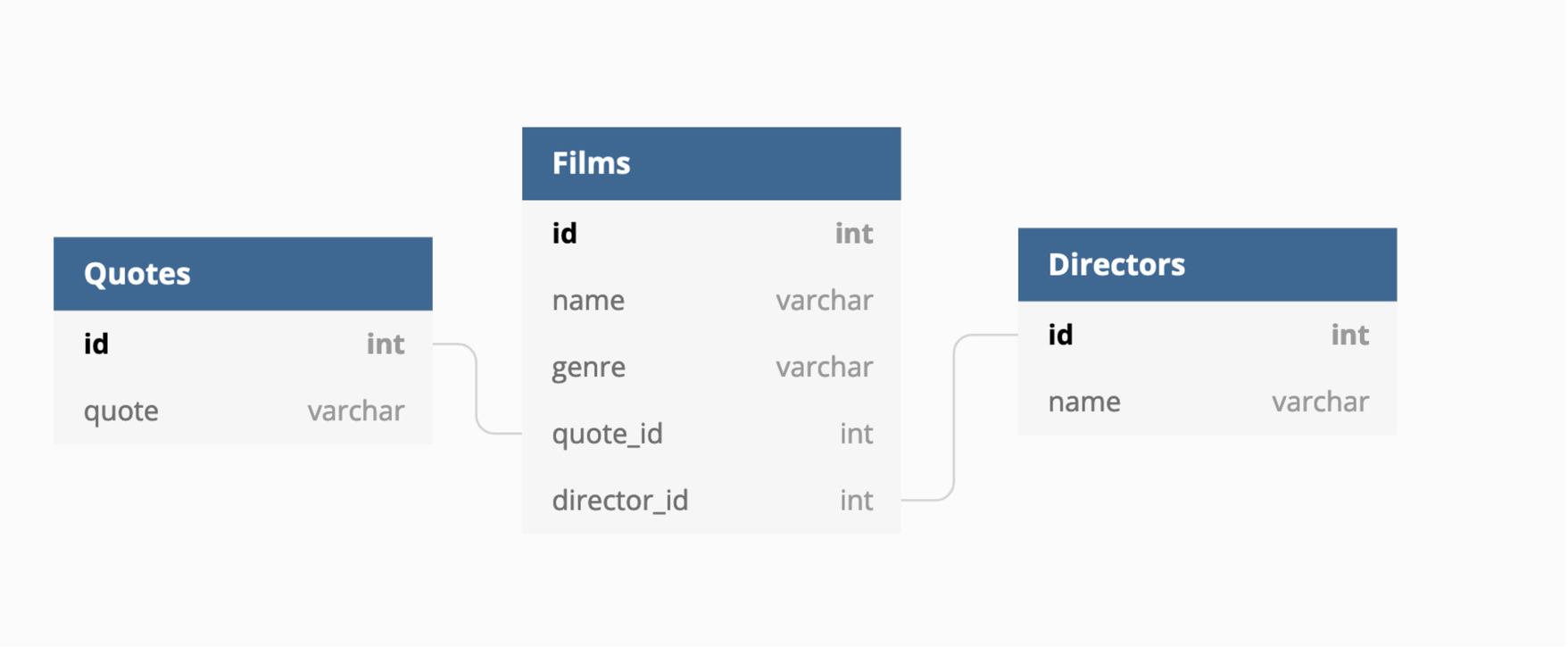
- Types of relationships
- Allowing the database to help ensure data integrity
- Joins: selecting data from multiple tables
- Imposing constraints (unique, foreign key)

One to one



A user HAS ONE profile.

One to many

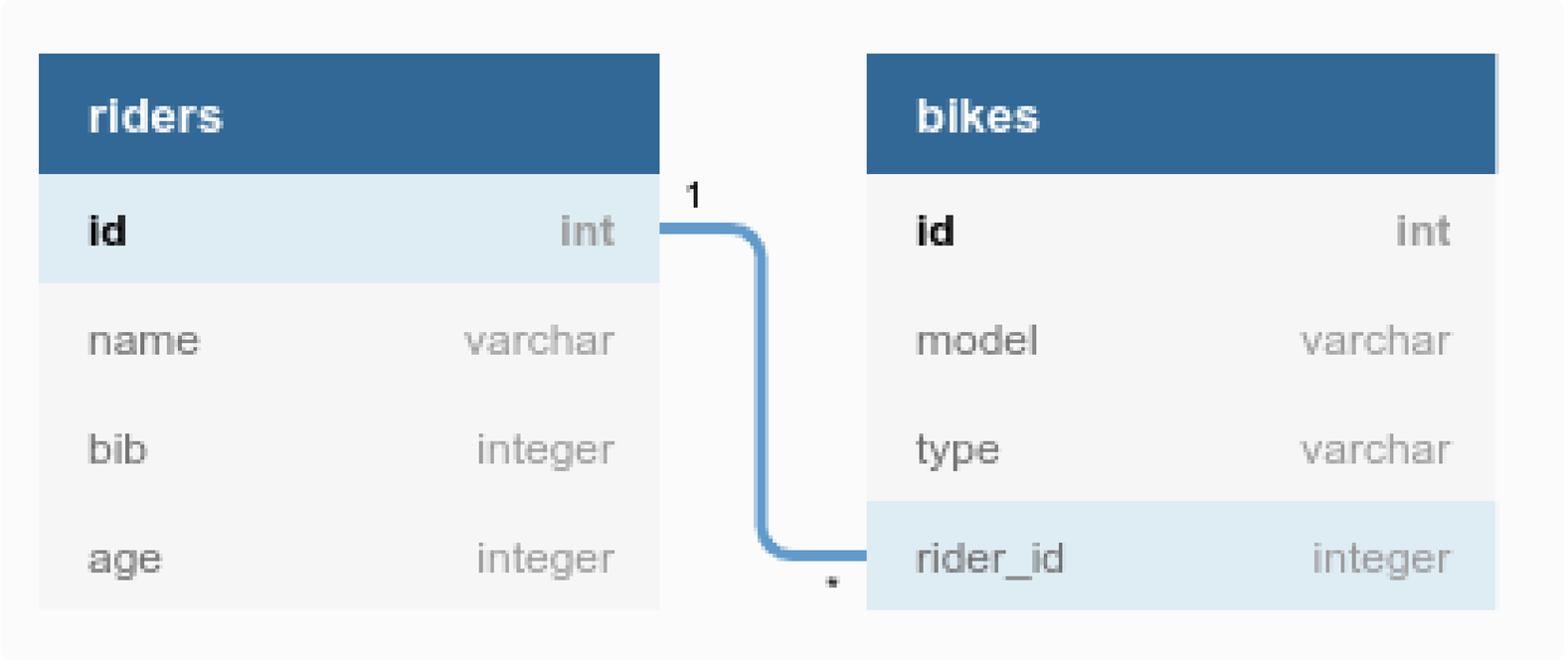


Our film table has a RELATIONSHIP with our directors and quotes tables. It has a ONE (films) to MANY (quotes, directors).

Allowing the DB to help ensure data integrity

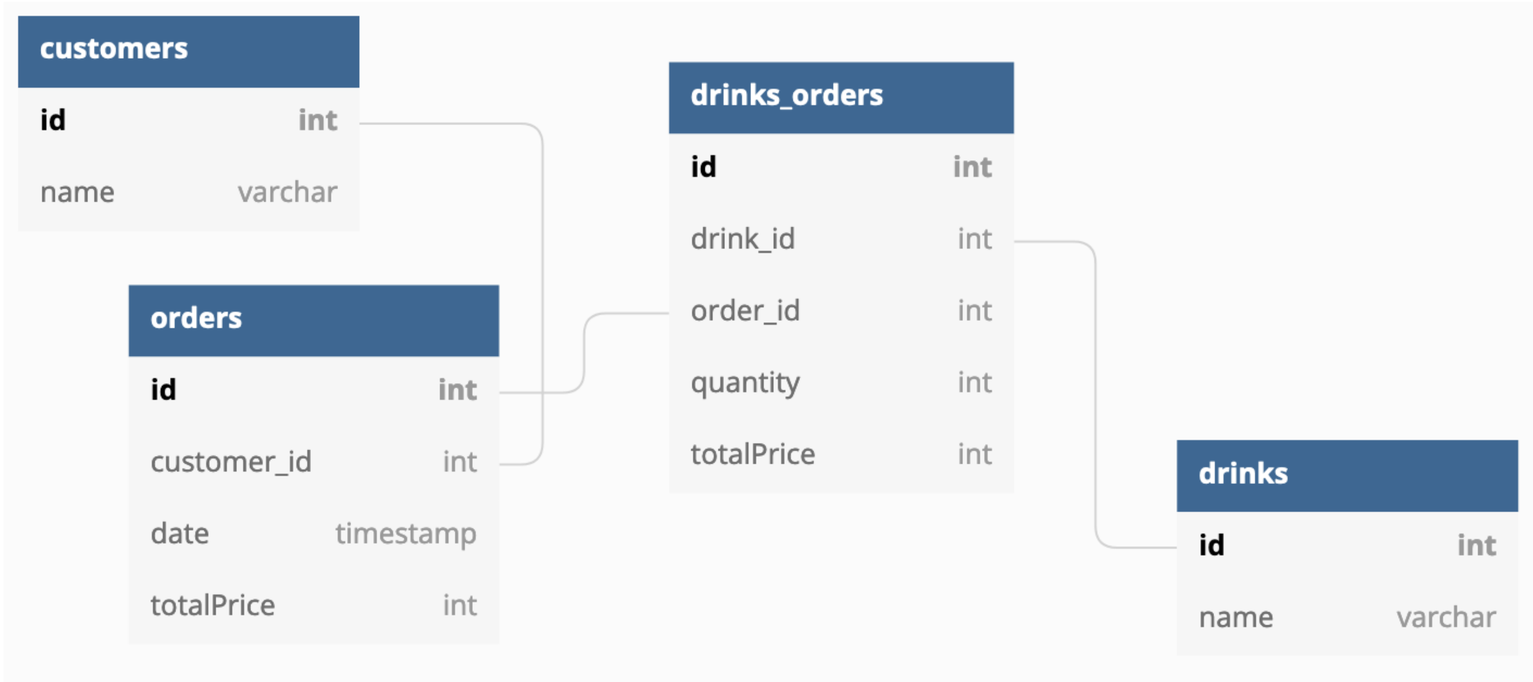
- Orphan fields: child records with a foreign key that refers to the primary key of a parent record that no longer exists

- Tell the DB about the intended relationship between tables



Joins: selecting data from multiple tables

- One to Many
- Customer can make many orders
- The JOIN table between drinks_orders is orders and drinks



Imposing constraints (unique)

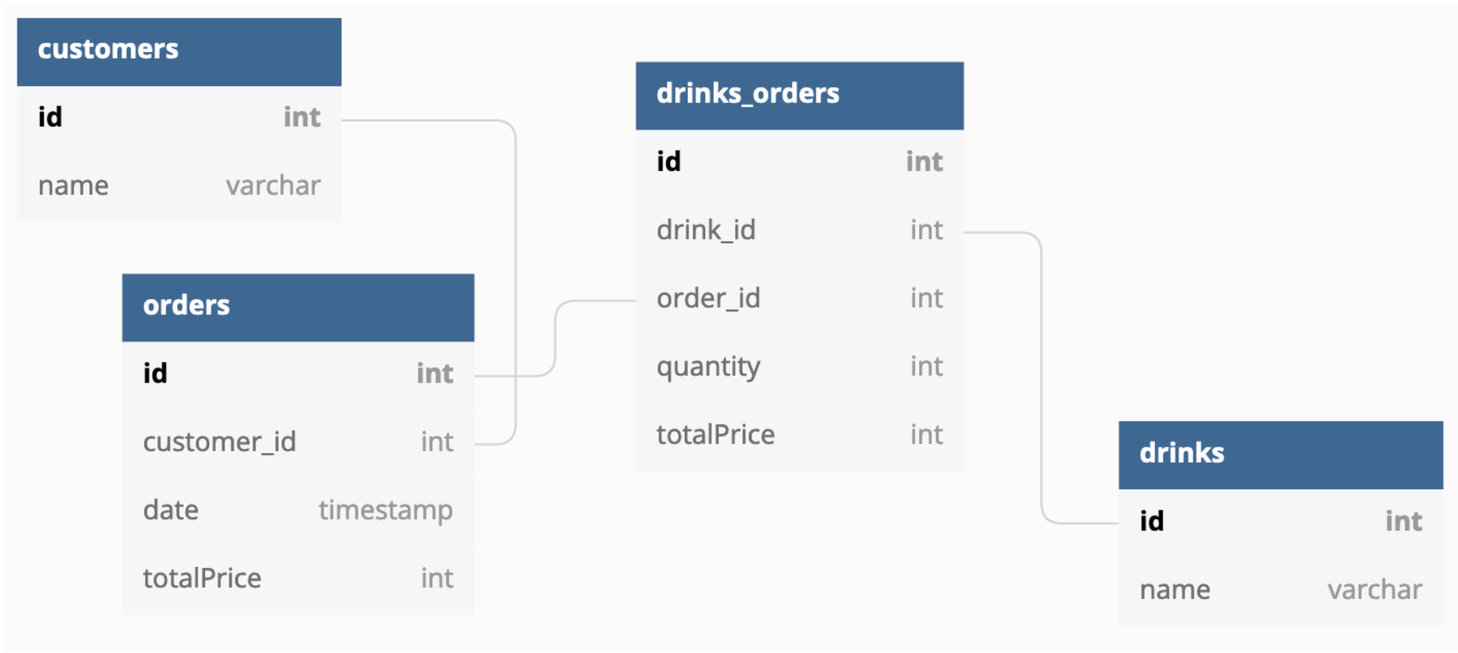
- Primary keys have a constraint on them where the values of that column are unique - meaning there are no duplicates in the table

- Values in a unique constraint column will be used by the database to do a ton of work using things called **indexes** to optimize your queries

Imposing constraints (foreign key)

A foreign key relates to the primary key in another table.

The `customer_id` column is a foreign key to the `id` primary key in `orders`




```
table.increments('id').primary()
table.integer('customer_id').references('customers.id')
```

Takeaways

- Types of relationships
- Allowing the database to help ensure data integrity
- Joins: selecting data from multiple tables
- Imposing constraints (unique, foreign key)