DEV ACADEMY
TE KURA HANGARAU
O AOTEAROA

Database Relationships

and

Complex joins

Week 5, Day 3

Agenda

- Database Relationships
  - Database Diagrams
  - One-to-One
  - One-to-Many
  - Many-to-Many
- Complex Joins
  - Joining more than 2 tables

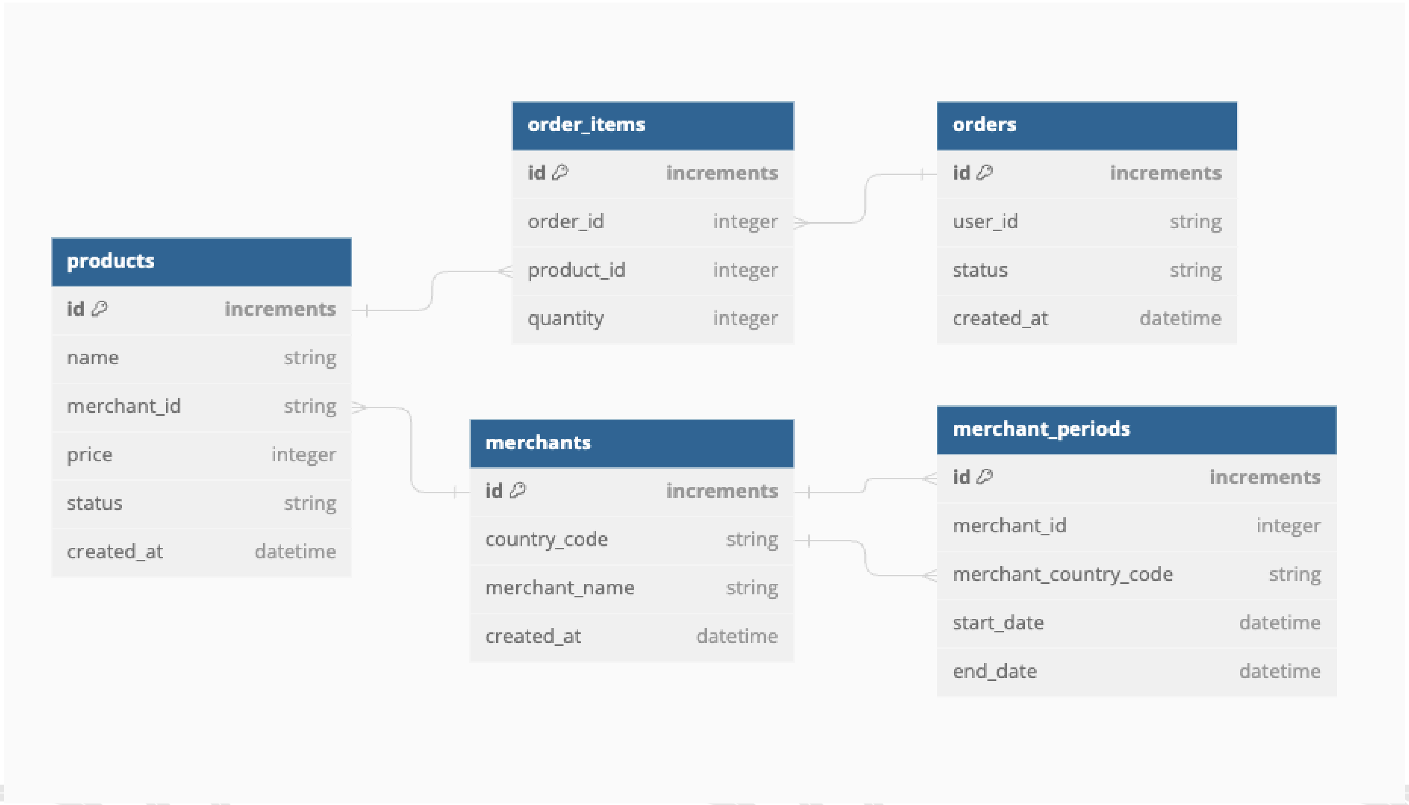
Database Relationships

Database Diagrams

- Entity Relationship Diagram (ERD)
  - A visual representation of relationships within a specific domain
- Database Diagram
  - A visual representation of the relationships within a database
- The relationships can be between things like:
  - People
  - Objects
  - Places
  - Events

Database Relationships

Database Diagrams



products	
id	increments
name	string
merchant_id	string
price	integer
status	string
created_at	datetime

order_items	
id	increments
order_id	integer
product_id	integer
quantity	integer

orders	
id	increments
user_id	string
status	string
created_at	datetime

merchants	
id	increments
country_code	string
merchant_name	string
created_at	datetime

merchant_periods	
id	increments
merchant_id	integer
merchant_country_code	string
start_date	datetime
end_date	datetime

One to One

Database Relationships

One to One

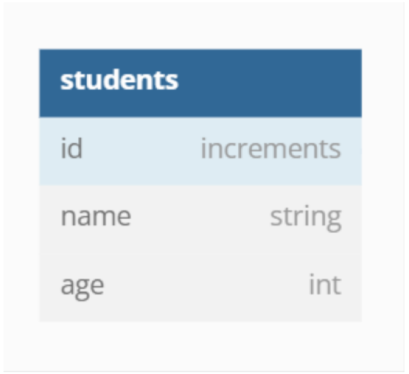
- One record in a table is associated with a maximum of one record in another table.
- Examples:
  - One employee to One company car
  - One person to One active passport
  - One account to One profile

Database Relationships

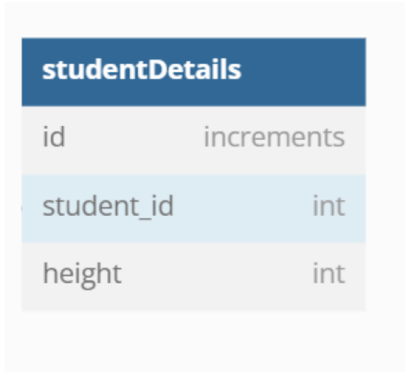
One to One

```
export const up = function(knex){
  return knex.schema.createTable('students', table => {
    table.increments('id')
    table.string('name')
    table.integer('age')
  })
}
```

```
export const up = function(knex){
  return knex.schema.createTable('studentDetails', table => {
    table.increments('id')
    table.string('student_id')
    table.integer('height')
  })
}
```



students	
id	increments
name	string
age	int



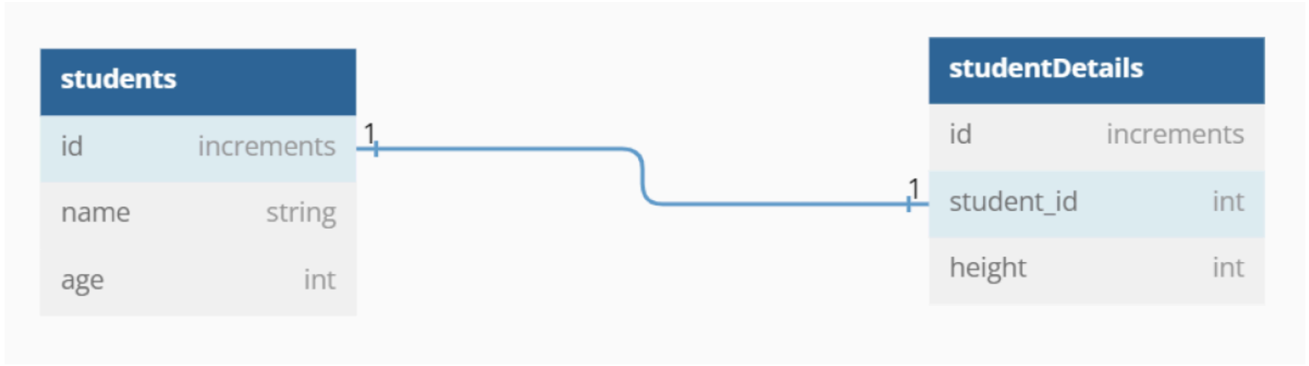
studentDetails	
id	increments
student_id	int
height	int

Database Relationships

One to One

```
export const up = function(knex){
  return knex.schema.createTable('students', table => {
    table.increments('id')
    table.string('name')
    table.integer('age')
  })
}
```

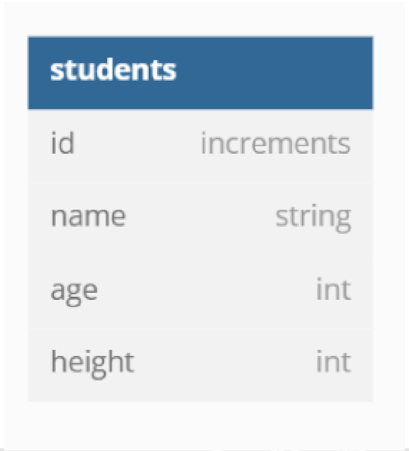
```
export const up = function(knex){
  return knex.schema.createTable('studentDetails', table => {
    table.increments('id')
    table.string('student_id').references('students.id').unique()
    table.integer('height')
  })
}
```



Database Relationships

One to One

```
export const up = function(knex){
  return knex.schema.createTable('students', table => {
    table.increments('id')
    table.string('name')
    table.integer('age')
    table.integer('height')
  })
}
```



students	
id	increments
name	string
age	int
height	int

One to Many

Database Relationships

One to Many

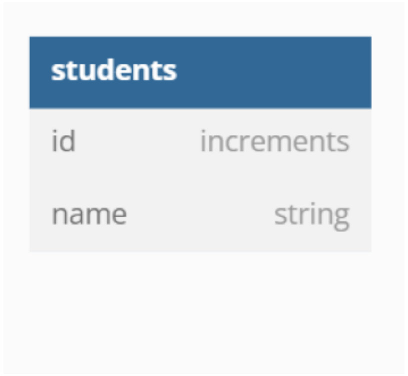
- One record in a table can be associated with one or more records in another table.
- Examples:
  - One employee to Many sales
  - One person to Many cars
  - One person to Many pets

Database Relationships

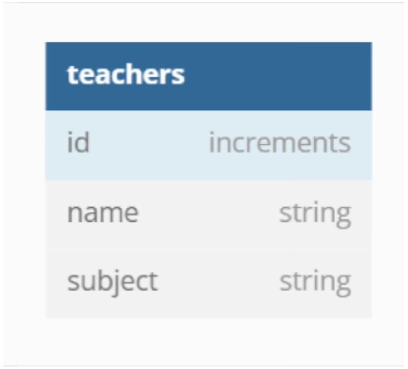
One to Many

```
export const up = function(knex){
  return knex.schema.createTable('students', table => {
    table.increments('id')
    table.string('name')

  })
}
```

```
export const up = function(knex){
  return knex.schema.createTable('teachers', table => {
    table.increments('id')
    table.string('name')
    table.integer('subject')
  })
}
```



students	
id	increments
name	string



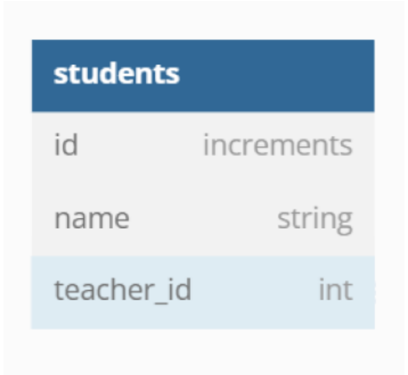
teachers	
id	increments
name	string
subject	string

Database Relationships

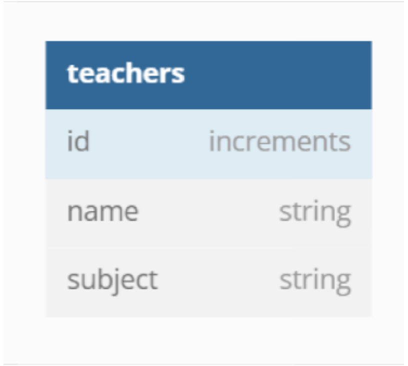
One to Many

```
export const up = function(knex){
  return knex.schema.createTable('students', table => {
    table.increments('id')
    table.string('name')
    table.integer('teacher_id')
  })
}
```

```
export const up = function(knex){
  return knex.schema.createTable('teachers', table => {
    table.increments('id')
    table.string('name')
    table.integer('subject')
  })
}
```



students	
id	increments
name	string
teacher_id	int



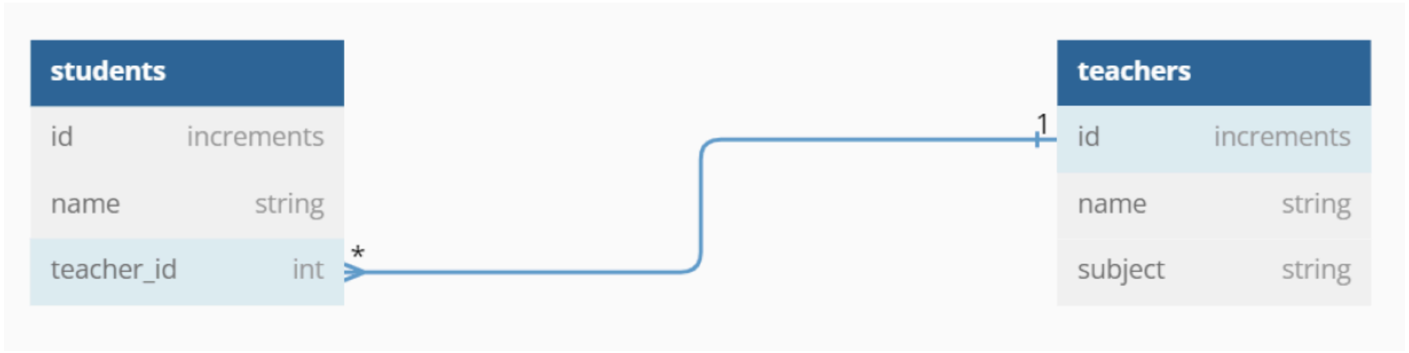
teachers	
id	increments
name	string
subject	string

Database Relationships

One to Many

```
export const up = function(knex){
  return knex.schema.createTable('students', table => {
    table.increments('id')
    table.string('name')
    table.integer('teacher_id').references('teachers.id')
  })
}
```

```
export const up = function(knex){
  return knex.schema.createTable('teachers', table => {
    table.increments('id')
    table.string('name')
    table.integer('subject')
  })
}
```

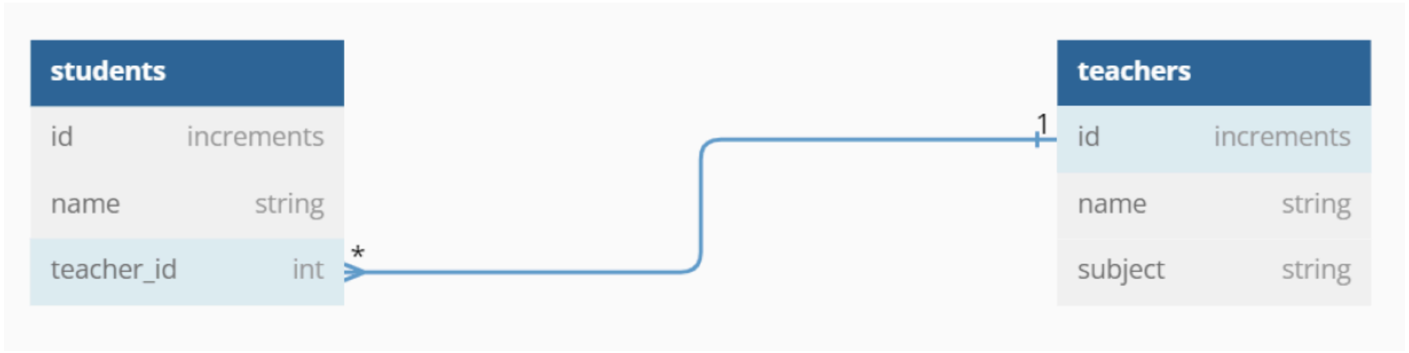


Database Relationships

One to Many

students		
id	name	teacher_id

teachers		
id	name	subject



Database Relationships

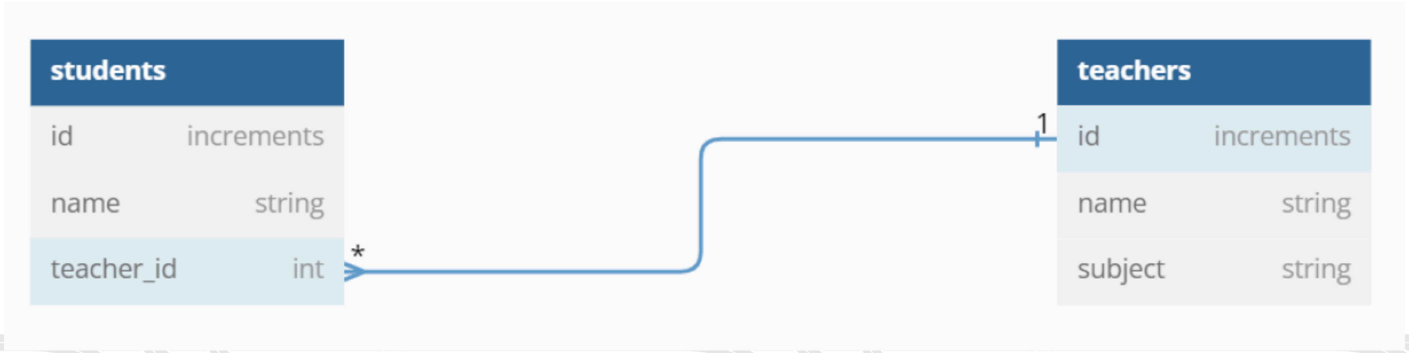
One to Many

students

id	name	teacher_id
1	Arnold	
2	Robbie	

teachers

id	name	subject



Database Relationships

One to Many

students

id	name	teacher_id
1	Arnold	1
2	Robbie	1

teachers

id	name	subject
1	Ms Frizzle	Science

```
db('students')
  .join('teachers', 'students.teacher_id', 'teachers.id')
  .where('teachers.name', 'Ms Frizzle')
  .select('*')
```

Database Relationships

One to Many

students			teachers		
id	name	teacher_id	id	name	subject
1	Arnold	1	1	Ms Frizzle	Science
2	Robbie	1	1	Ms Frizzle	Science

```
db('students')
  .join('teachers', 'students.teacher_id', 'teachers.id')
  .where('teachers.name', 'Ms Frizzle')
  .select('*')
```

Database Relationships

One to Many

students			teachers		
id	name	teacher_id	id	name	subject
1	Arnold	1	1	Ms Frizzle	Science
2	Robbie	1	1	Ms Frizzle	Science

```
[
  { id: 1, name: 'Arnold', teacher_id: 1, id: 1, name: 'Ms Frizzle', subject: 'Science' },
  { id: 2, name: 'Robbie', teacher_id: 1, id: 1, name: 'Ms Frizzle', subject: 'Science' }
]
```

Many to Many

Database Relationships

Many to Many

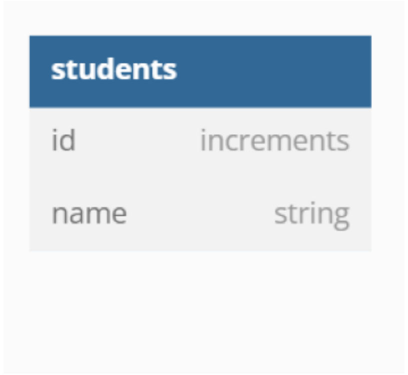
- Multiple records in a table can be associated with multiple records in another table.
- Examples:
  - Many students to Many teachers
  - Many authors to Many books
  - Many people to Many properties
- To model a many-to-many relationship, we need to create a new table to connect the other two.
- This new table can be called a relationship table, joining table, intermediate table, linking table or junction table.

Database Relationships

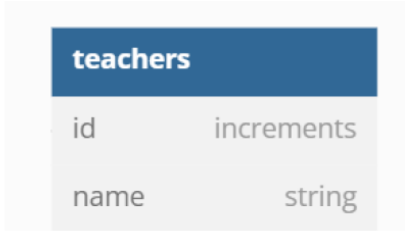
Many to Many

```
export const up = function(knex){
  return knex.schema.createTable('students', table => {
    table.increments('id')
    table.string('name')
  })
}
```

```
export const up = function(knex){
  return knex.schema.createTable('teachers', table => {
    table.increments('id')
    table.string('name')
  })
}
```



students	
id	increments
name	string



teachers	
id	increments
name	string

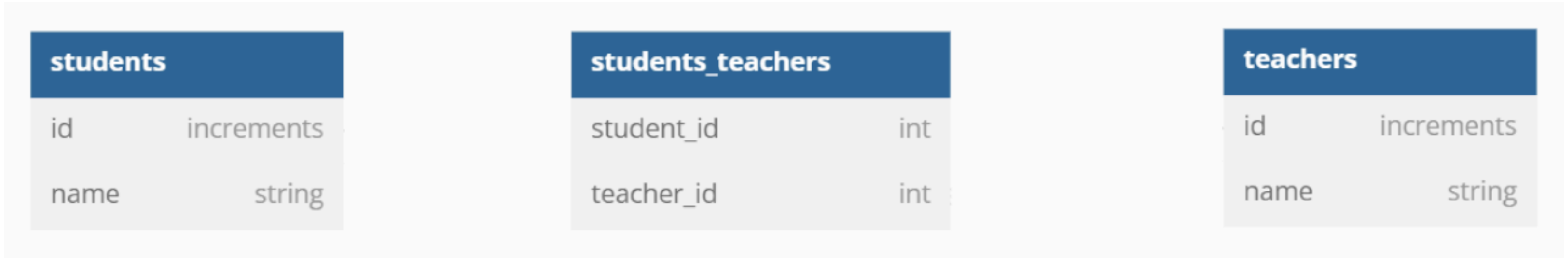
Database Relationships

Many to Many

```
export const up = function(knex){
  return knex.schema.createTable('stu
    table.increments('id')
    table.string('name')

  })
}
```

```
export const up = function(knex){
  return knex.schema.createTable('stu
    table.integer('student_id')
    table.integer('teacher_id')

  })
}
```

```
export const up = function(knex){
  return knex.schema.createTable('tea
    table.increments('id')
    table.string('name')

  })
}
```



students	
id	increments
name	string

students_teachers	
student_id	int
teacher_id	int

teachers	
id	increments
name	string

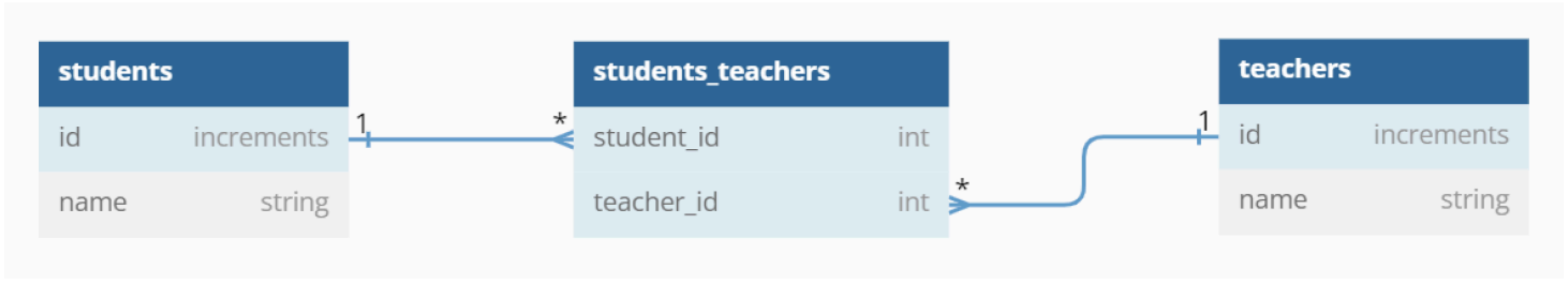
Database Relationships

Many to Many

```
export const up = function(knex){
  return knex.schema.createTable('stuc
    table.increments('id')
    table.string('name')

  })
}
```

```
export const up = function(knex){
  return knex.schema.createTable('stuc
    table.integer('student_id').refere
    table.integer('teacher_id').refere

  })
}
```

```
export const up = function(knex){
  return knex.schema.createTable('teac
    table.increments('id')
    table.string('name')

  })
}
```



Database Relationships

Many to Many

students

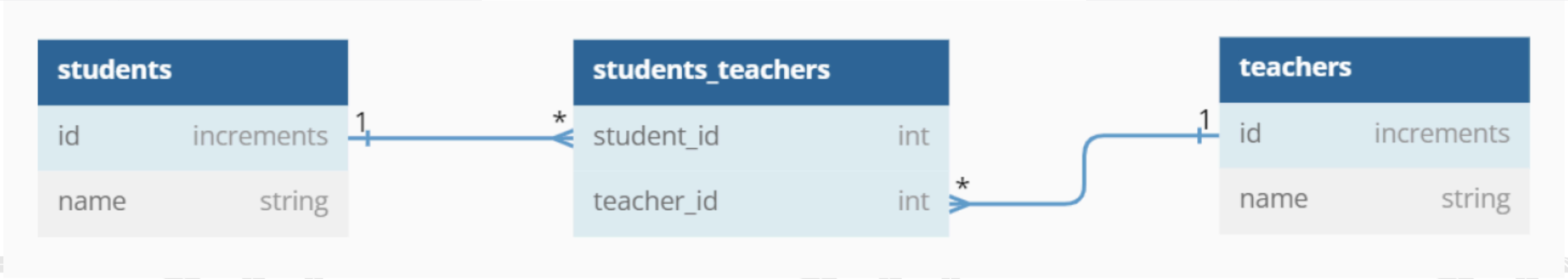
id	name
1	Amy
2	Alicia
3	Dean

students_teachers

student_id	teacher_id

teachers

id	name
1	Jared
2	Hannah
3	Daph



Database Relationships

Many to Many

students

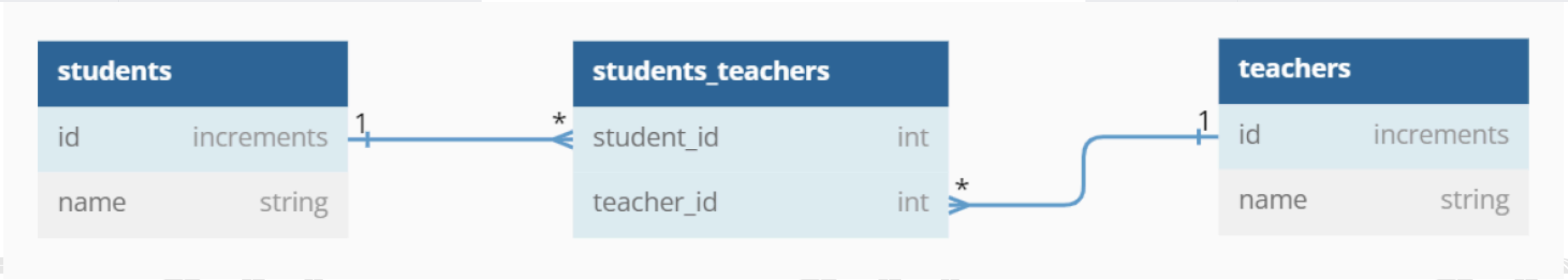
id	name
1	Amy
2	Alicia
3	Dean

students_teachers

student_id	teacher_id
1	1

teachers

id	name
1	Jared
2	Hannah
3	Daph



Database Relationships

Many to Many

students

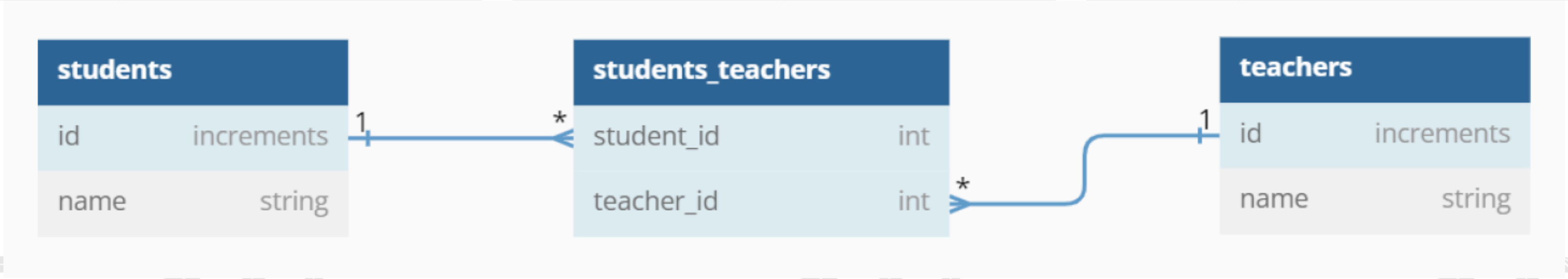
id	name
1	Amy
2	Alicia
3	Dean

students_teachers

student_id	teacher_id
1	1
2	1
1	3

teachers

id	name
1	Jared
2	Hannah
3	Daph



Complex Joins

Complex Joins

Many to Many

students

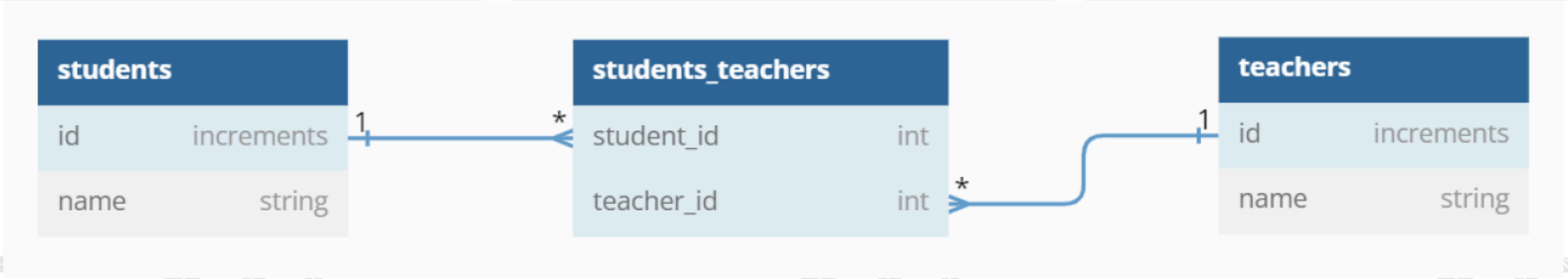
id	name
1	Amy
2	Alicia
3	Dean

students_teachers

student_id	teacher_id
1	1
2	1
1	3

teachers

id	name
1	Jared
2	Hannah
3	Daph



Complex Joins

Many to Many

students

id	name
1	Amy
2	Alicia
3	Dean

students_teachers

student_id	teacher_id
1	1
2	1
1	3

teachers

id	name
1	Jared
2	Hannah
3	Daph

```
db('students')
```

Complex Joins

Many to Many

students	
id	name
1	Amy
2	Alicia
3	Dean

students_teachers	
student_id	teacher_id
1	1
2	1
1	3

teachers	
id	name
1	Jared
2	Hannah
3	Daph

```
db('students')
  .join('students_teachers', 'students.id', 'students_teachers.student_id')
```

Complex Joins

Many to Many

students

id	name
1	Amy
2	Alicia
3	Dean

students_teachers

student_id	teacher_id
1	1
2	1
1	3

teachers

id	name
1	Jared
2	Hannah
3	Daph

```
db('students')
  .join('students_teachers', 'students.id', 'students_teachers.student_id')
  .join('teachers', 'students_teachers.teacher_id', 'teachers.id')
```

Complex Joins

Many to Many

students		students_teachers		teachers	
id	name	student_id	teacher_id	id	name
1	Amy	1	1	1	Jared
2	Alicia	2	1	1	Jared
1	Amy	1	3	3	Daph

```
db('students')
  .join('students_teachers', 'students.id', 'students_teachers.student_id')
  .join('teachers', 'students_teachers.teacher_id', 'teachers.id')
  .select('*')
```

Complex Joins

Many to Many

students		students_teachers		teachers	
id	name	student_id	teacher_id	id	name
1	Amy	1	3	3	Daph

```
db('students')
  .join('students_teachers', 'students.id', 'students_teachers.student_id')
  .join('teachers', 'students_teachers.teacher_id', 'teachers.id')
  .where('teachers.name', 'Daph')
  .select('*')
```

Complex Joins

Many to Many

students		students_teachers		teachers	
id	name	student_id	teacher_id	id	name
1	Amy	1	3	3	Daph

```
[
  { id: 1, name: 'Amy', student_id: 1, teacher_id: 3, id: 3, name: 'Daph' }
]
```

```
  .select('students.id as studentId', 'students.name as studentName',
    'teachers.id as teachersId', 'teacher.name as teacherName')
```

Bonus Content

Adding new data to tables with a Many to Many relationship

- When adding a new entry to the database, we need to ensure all tables are updated in the correct order:
  - 1. Update the parent tables
  - 2. Update the child (joining) table
- To delete an existing entry from the database, do these steps in reverse.
- Let's add this relationship into our tables:

```
{ id: 4, name: 'Logan', student_id: 4, teacher_id: 4, id: 1, name: 'Barbora' }
```

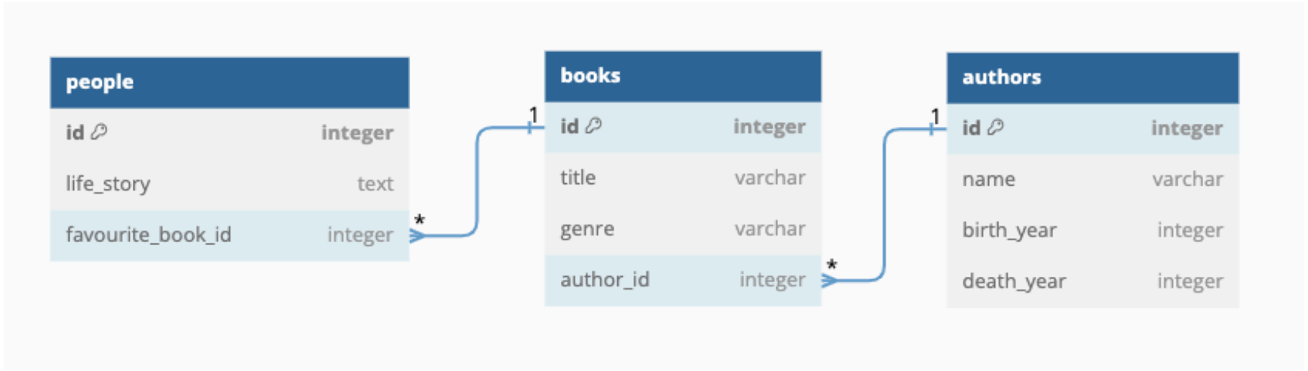
Bonus Content

Adding new data to tables with a Many to Many relationship

```
  export async function addNewStudent(data: { name: string; teacher: string }) {
  const { name, teacher } = data
  // 1a. Parent Table - `students`:
  const studentReturn = await db('students').insert({ name }, ['id'])
  console.log(studentReturn) // output: [{ id: 4 }]
  const studentId = studentReturn[0].id
  // 1b. Parent Table - `teachers`:
  const teachReturn = await db('teachers').insert({ name: teacher }, ['id'])
  console.log(teachReturn) // output: [{ id: 4 }]
  const teacherId = teachReturn[0].id
  // 2. Joining Table - `students_teachers`:
  await db('students_teachers').insert({
    student_id: studentId,
    teacher_id: teacherId
  })
}
```

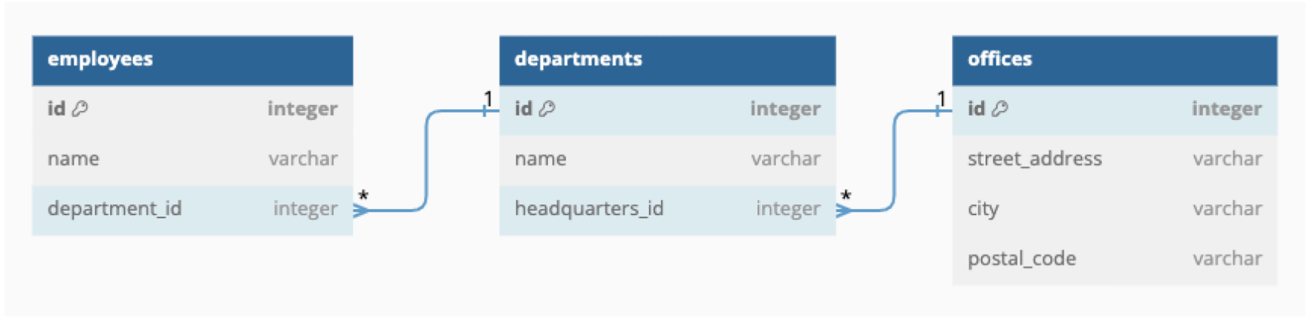
Complex Joins

NOT Many to Many



Complex Joins

NOT Many to Many


employees
id integer
name varchar
department_id integer
*
1
departments
id integer
name varchar
headquarters_id integer
*
1
offices
id integer
street_address varchar
city varchar
postal_code varchar