Entity Relationship Diagrams (ERD)

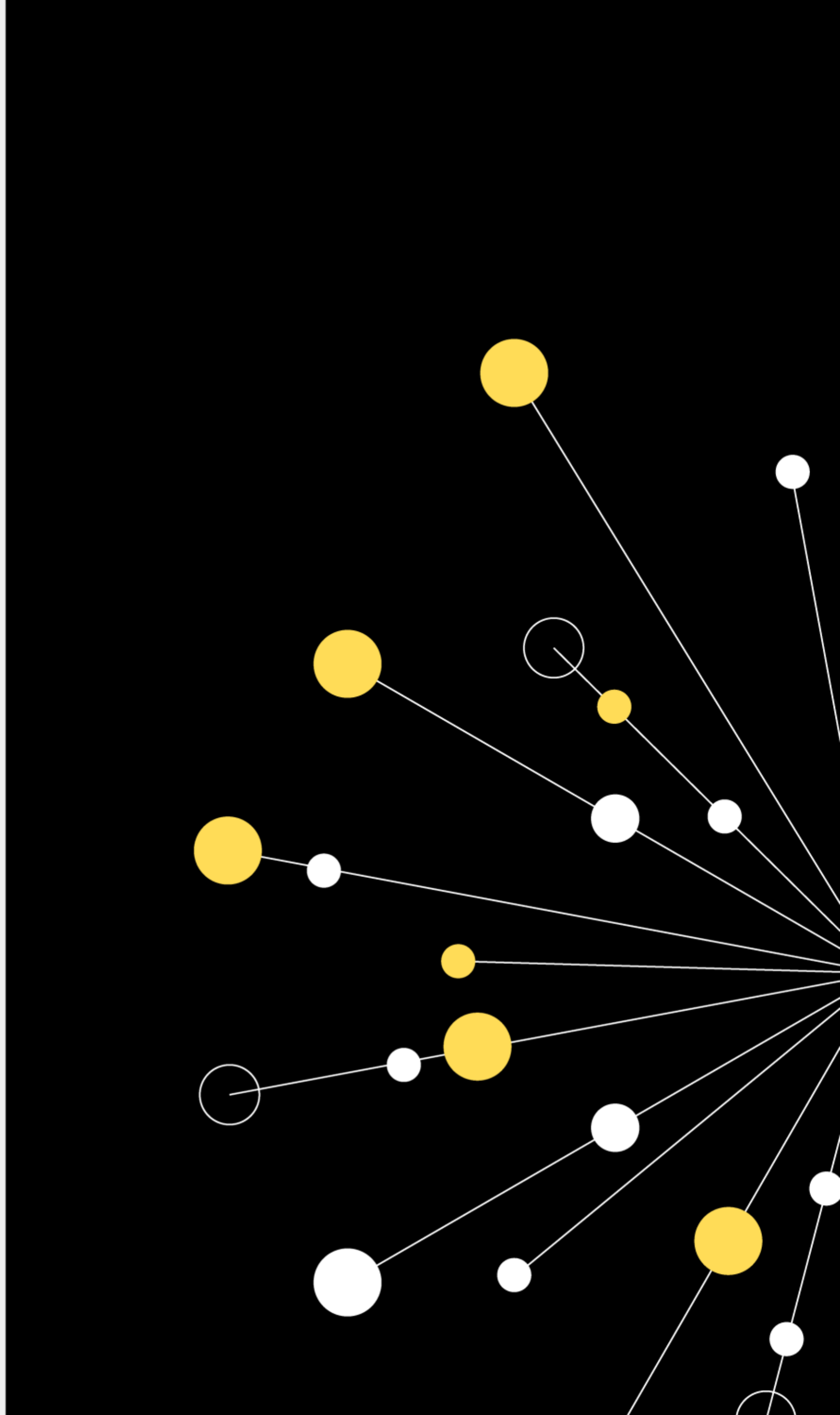
Hilary Brownlow

Senior Technical Learning Specialist – Data Analysis

Entity Relationship Diagrams (ERD)

Contents:

- Entities, attributes and relationships
- ERD notations
- Cardinality
- One-to-One, Many-to-Many, One-to-Many relationships



Entities

An entity is anything about which information is recorded.

Person

Thing

Place

Event

Entities are represented in tables where the rows are individual instances of the entities, and the columns are their attributes.

Entities

Entities can be:

- **Person**: Employee, student, patient
- **Place**: Store, warehouse
- **Object**: Machine, product, car
- **Event**: Sale, registration, renewal
- **Concept**: Account, course

Attributes

Attributes are what we want to know about the entities

Each attribute can only hold one value at a time.

Each entity needs a unique identifier (key).

Each attribute must depend upon the key.

- Name
- Sex
- Height
- Shoe size

- Product code
- Description
- Size

Entities and relationships

Identifying Entities and Relationships

Look for the NOUNS & VERBS in the requirements

- NOUNS – entities – represented by rectangles
- VERBS – relationships – represented by lines

There are three main types of relationships (and corresponding links):

- **One-to-One**
- **One-to-Many**
- **Many-to-Many**

The only “native” relationship for relational databases is One-to-Many. Classical example of this relationship is Customer and Orders, where one customer can place multiple orders – but each order is placed by a single customer. One-to-Many is the only relationship relational databases were originally created to support.

Entities and relationships: examples

Let’s take a look at an…

Airline System

What are the **entities** and the **relationships** here?

A Passenger buys a Ticket to travel on an Aircraft which carries out a Flight from one Airport to another Airport.

Entities and relationships: examples (2)

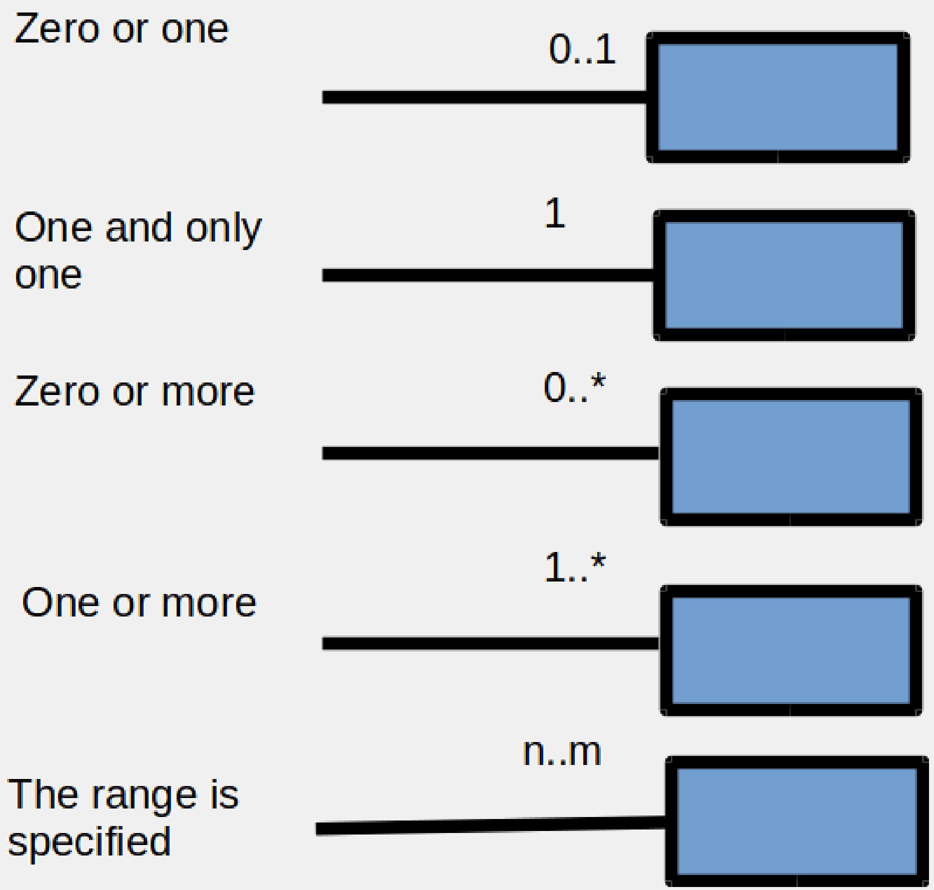
Let’s take a look at an…

Airline System

What are the **entities** and the **relationships** here?

A Passenger buys a Ticket to travel on an Aircraft which carries out a Flight from one Airport to another Airport

Diagram notations – UML



Zero or one 0..1

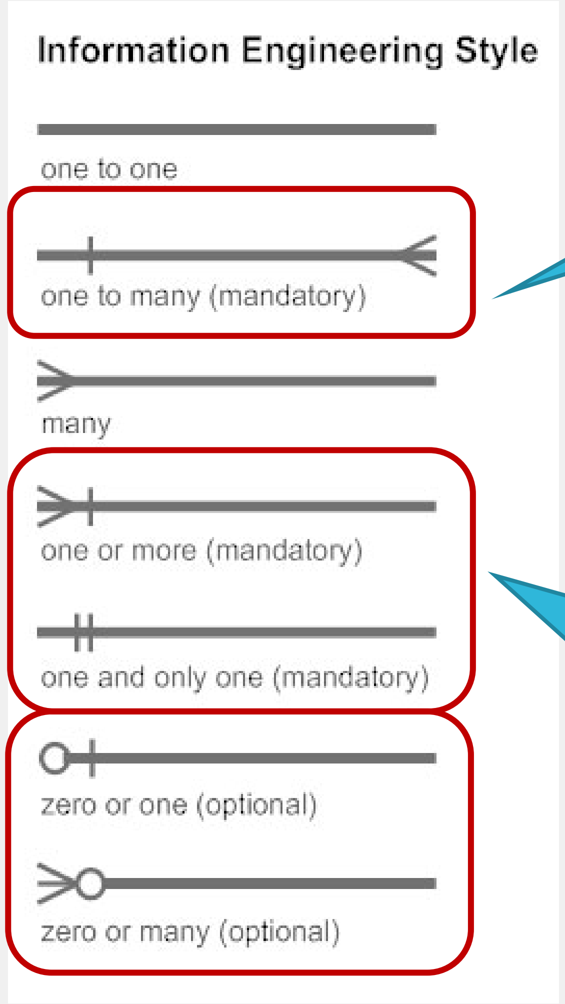
One and only one 1

Zero or more 0..*

One or more 1..*

The range is specified n..m

Diagram notations – IEM (Crow's Foot)



Has to be one at one end and many at the other

Both of these two would NOT HAVE NULL data types

Both of these last two would have NULL data types

Diagram Notations – IEM (Crow's Foot)

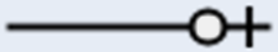
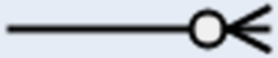
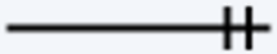
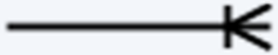
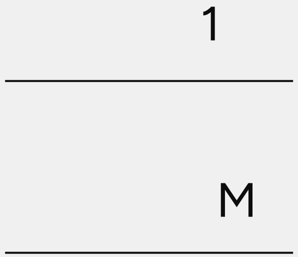
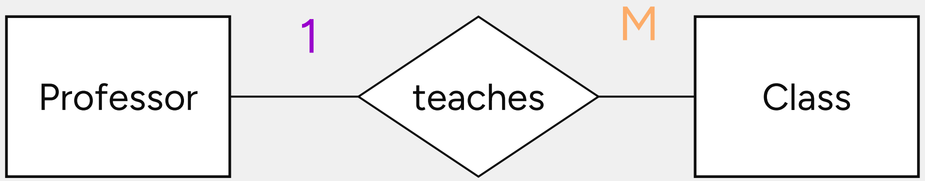
		Maximum	
		1	∞
Minimum	0		
	1		

Diagram Notations – Chen



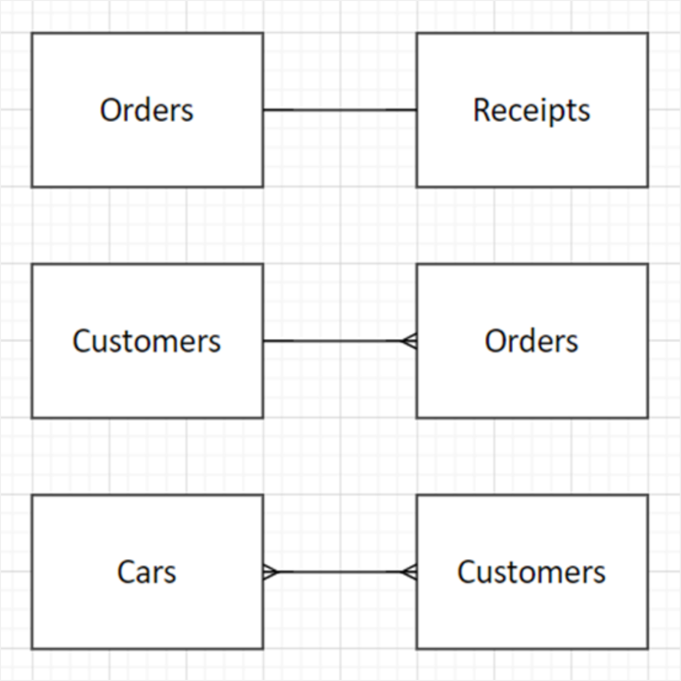


1 to represent one
M to represent many

Cardinality

Cardinality specifies the **number** of **instances** of each **entity** that is involved in the relationship.

To determine which type of relationship is there between two tables, look at a single record from each table and ask yourself how many records from the opposite table is this record linked to: one or many? Then take a single record from the opposite table and ask the same question. Combining answers to these questions will reveal the cardinality of that relationship.



One-to-One

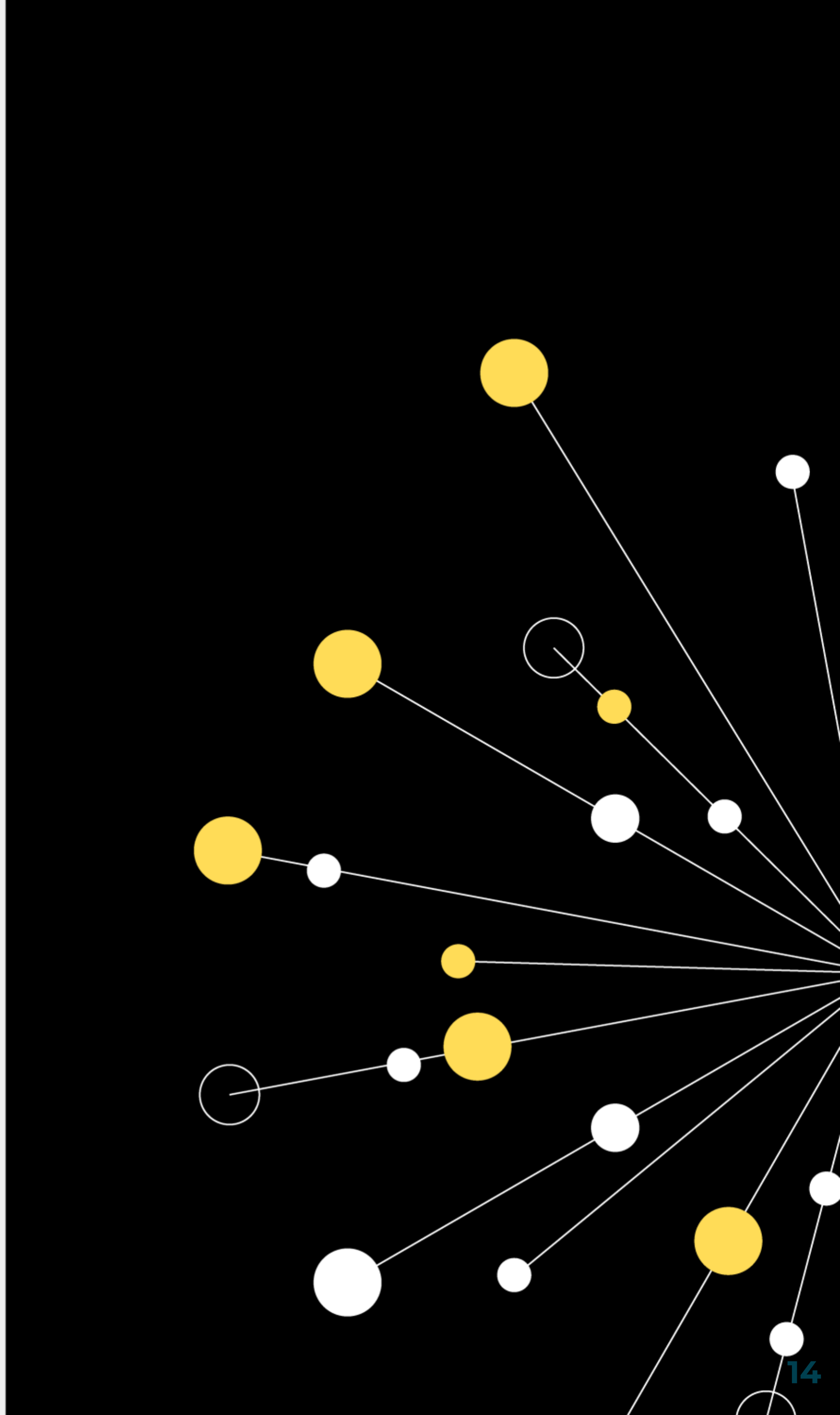
One-to-Many

Many-to-Many

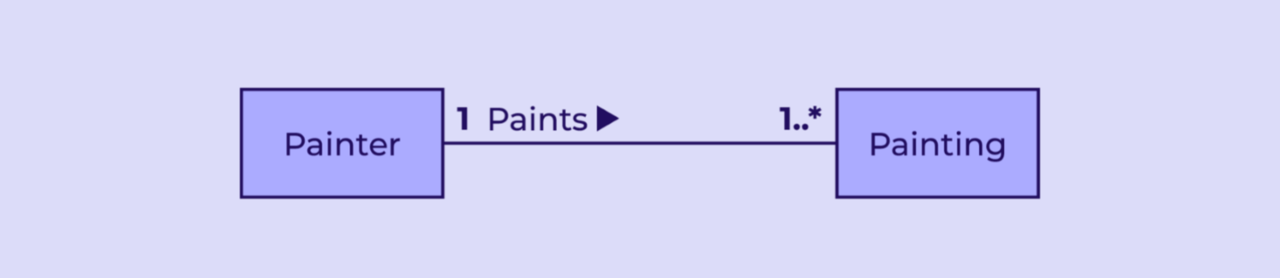
Activity: Entity Relationship Diagrams (ERD)

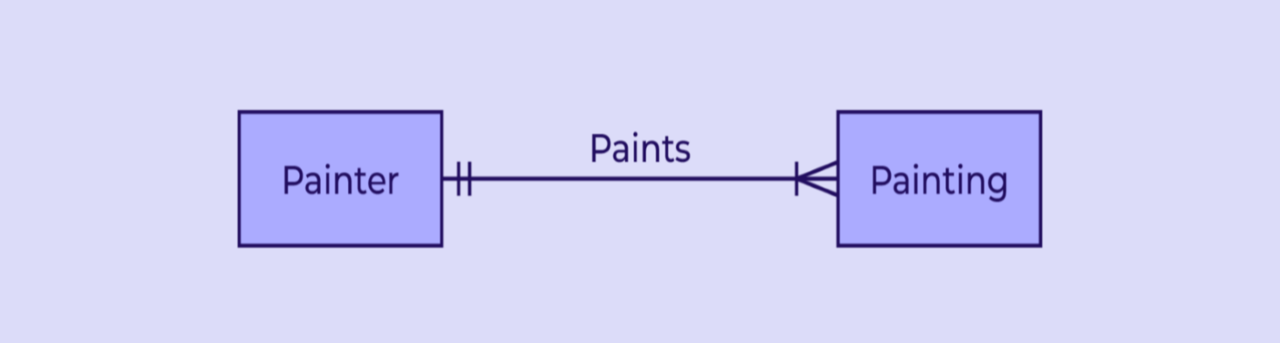
Working individually, use each of the three diagram styles (UML, IEM and Chen) to represent the following statement as an ERD:

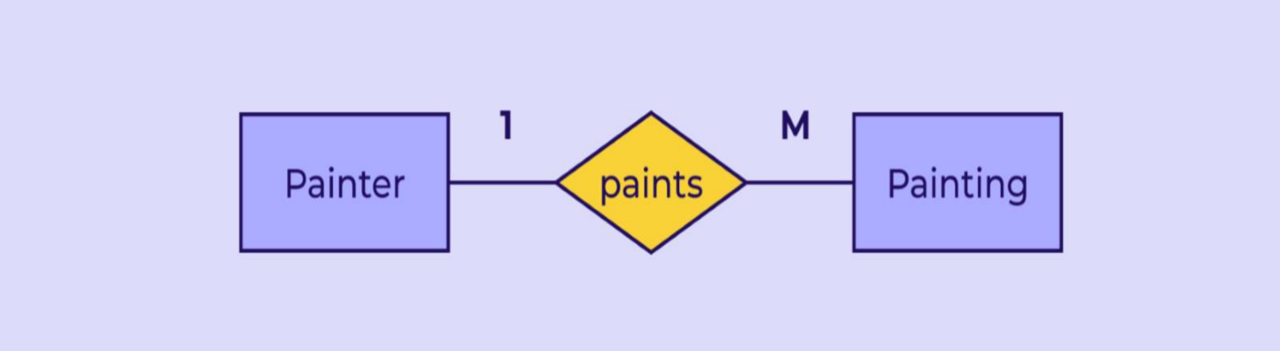
‘The painter paints one or more paintings.’



Activity solution



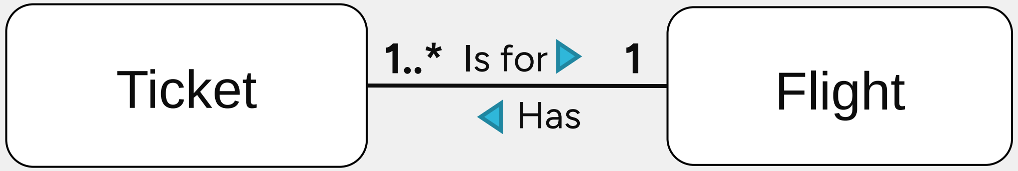




Cardinality

In a Universal Modelling Language (UML) diagram, we specify the cardinality along our relationship edge:

> > > One Ticket to One Flight



One Flight has many Tickets < < <

Cardinality

Cardinality is controlled by defined Business Rules, so…

Passenger -------------------------------Ticket

One passenger can buy multiple tickets

Or

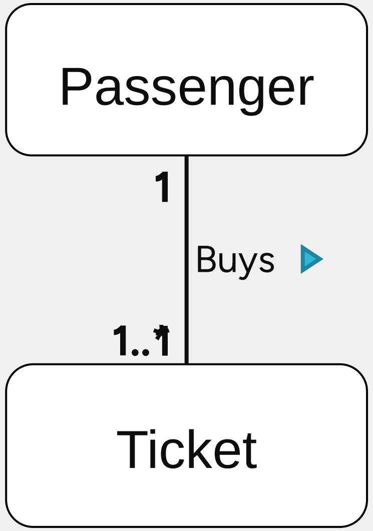
One passenger can travel on one ticket

Depends on the definition of Passenger – what’s the business rule?

Cardinality

So, we specify the cardinality according to our business rule.

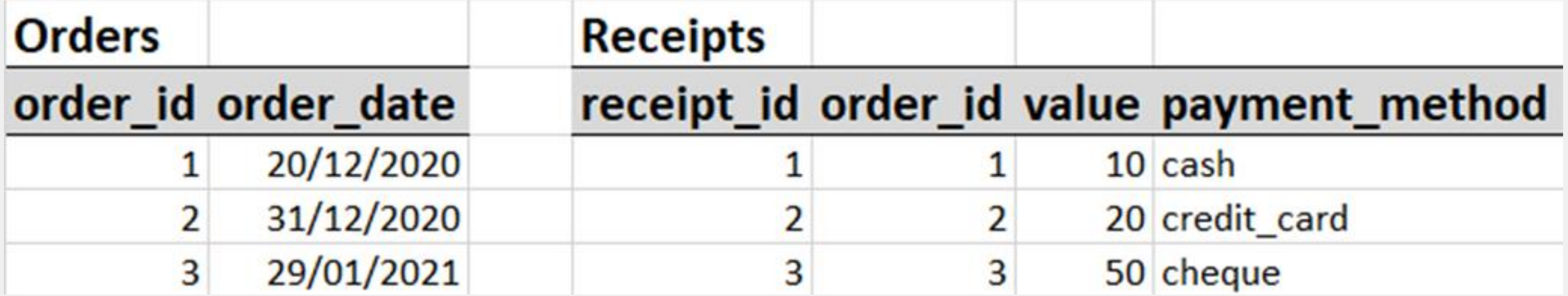
One to Many?



One to One?

Relational databases and relationships: One-to-One

Most One-to-One relationships usually can be merged into a single table:



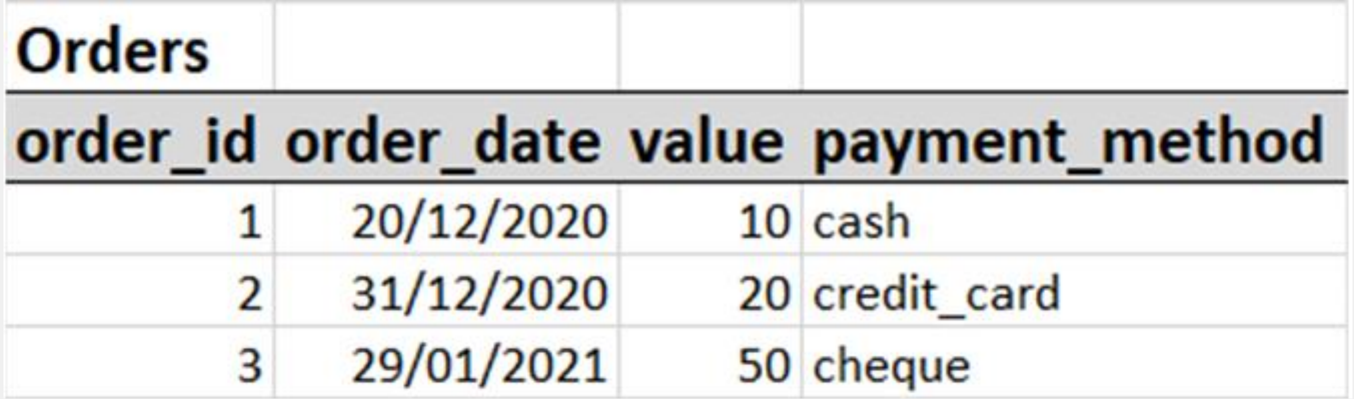
Orders

order_id	order_date
1	20/12/2020
2	31/12/2020
3	29/01/2021

Receipts

receipt_id	order_id	value	payment_method
1	1	10	cash
2	2	20	credit_card
3	3	50	cheque

These tables are linked as One-to-One: one order refers to one receipt, and one receipt refers to a single order only. In such scenario, these two tables can be merged like this:

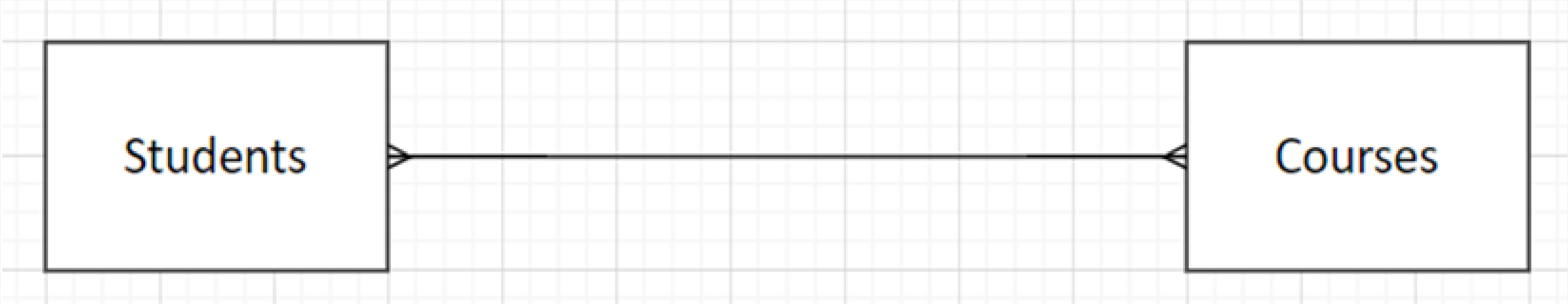


Orders

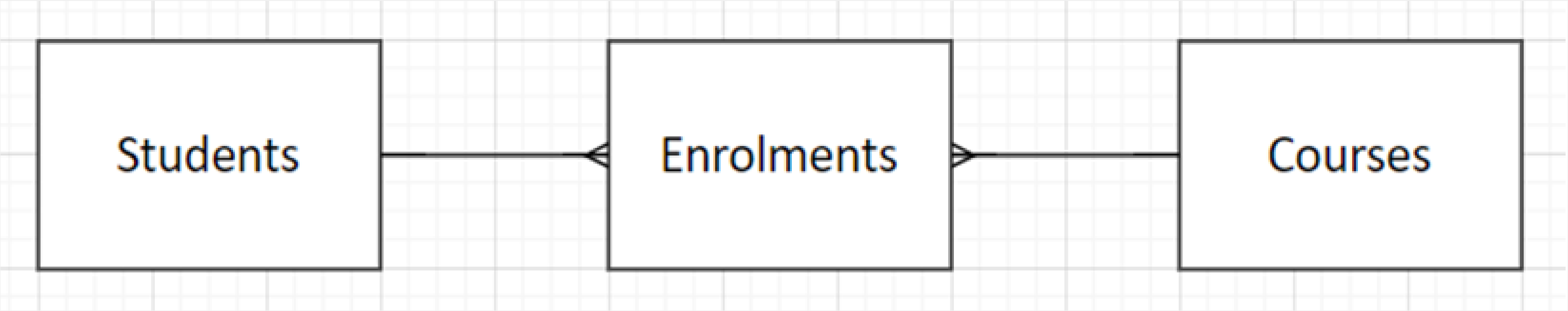
order_id	order_date	value	payment_method
1	20/12/2020	10	cash
2	31/12/2020	20	credit_card
3	29/01/2021	50	cheque

Relational databases and relationships: Many-to-many

Many-to-Many relationships exist logically but they are impossible to implement physically: databases do not support this kind of link. For example, if you have tables Students and Courses, they are linked as Many-to-Many: each student might attend multiple courses and each course contains multiple students:



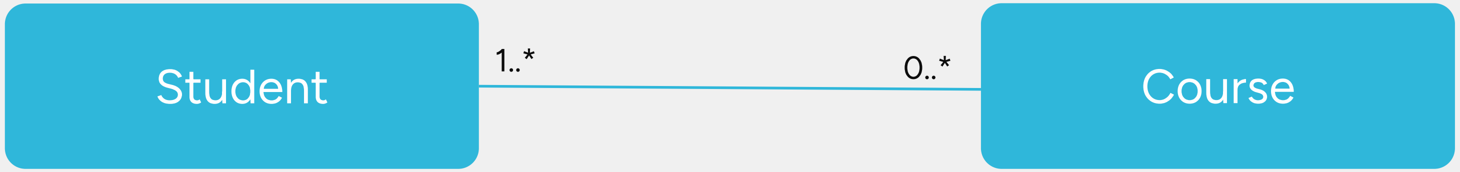
Each Many-to-Many relationship is usually replaced with one extra table placed between the two tables that are linked with Many-to-Many relationship, and two One-to-Many links. In the case of Students and Courses, such table would be Enrolments:

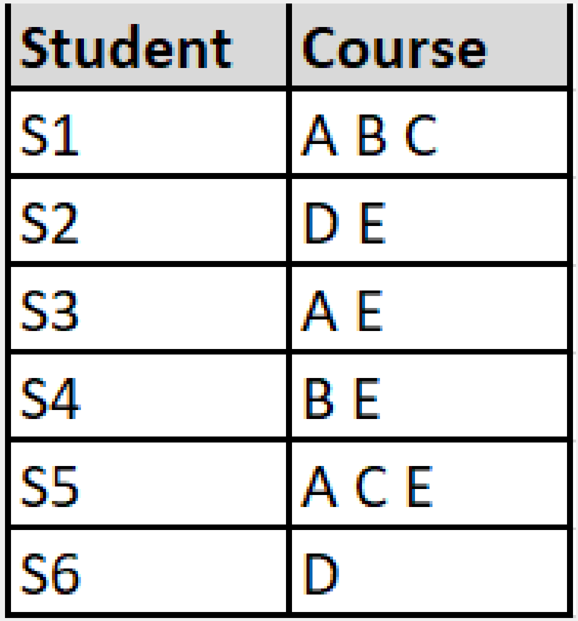


Relational databases and relationships: Many-to-many

How can we store the data about students and courses?

This way?

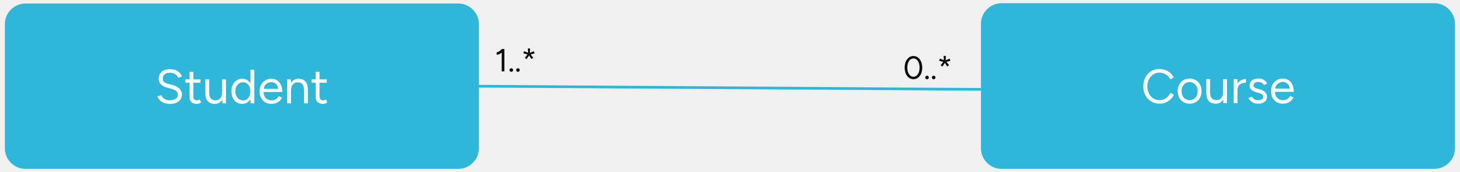


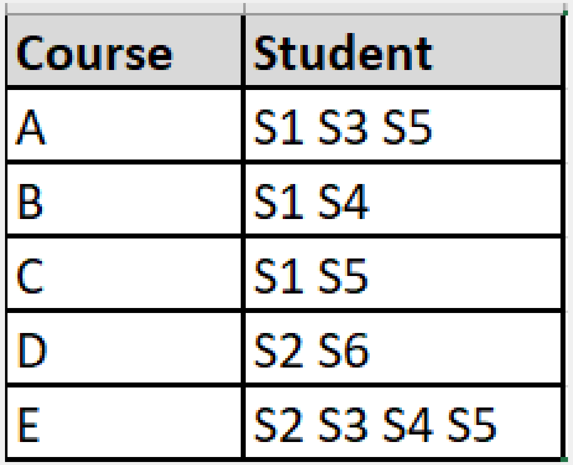


Student	Course
S1	A B C
S2	D E
S3	A E
S4	B E
S5	A C E
S6	D

Relational databases and relationships: Many-to-many

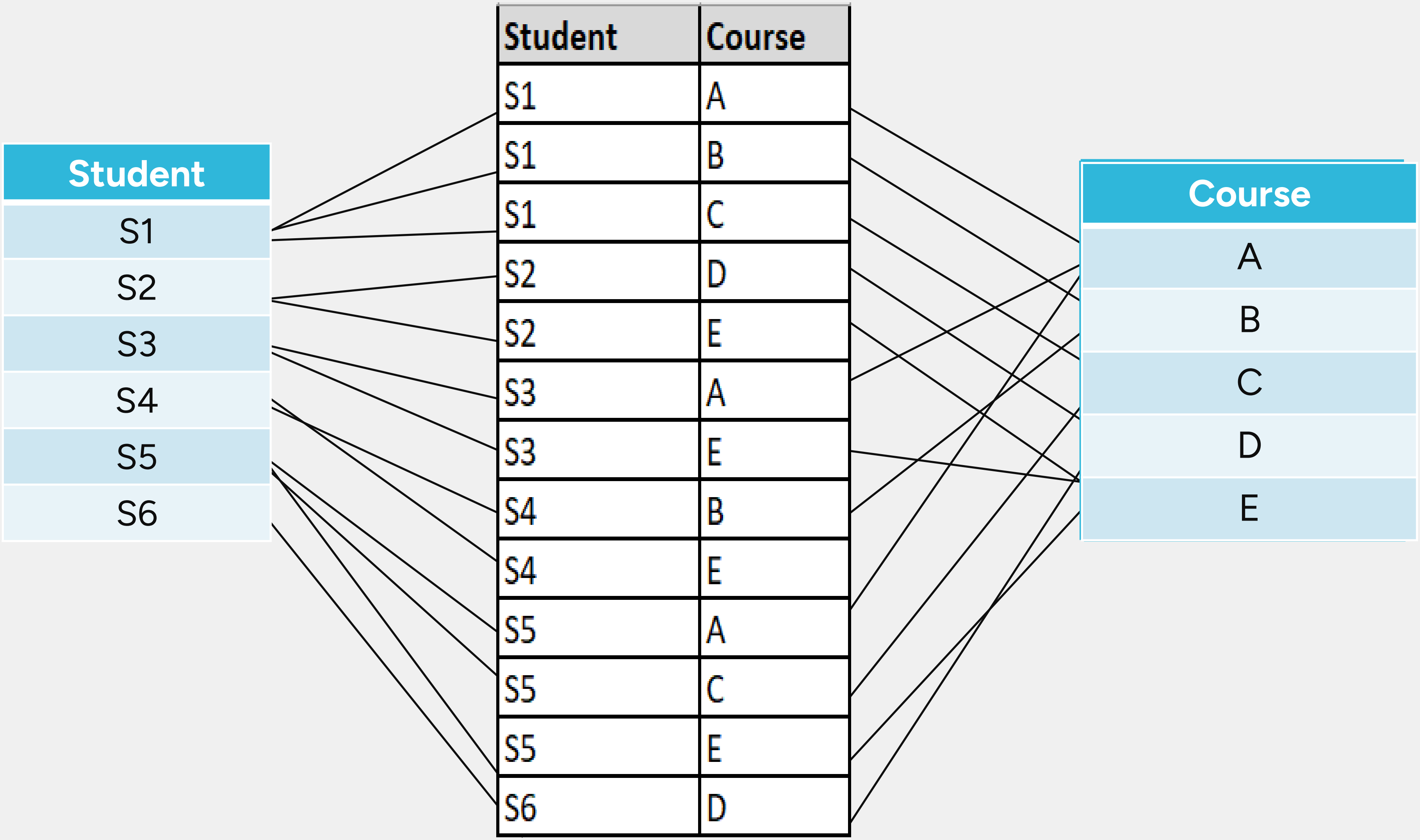
… or this way?





Course	Student
A	S1 S3 S5
B	S1 S4
C	S1 S5
D	S2 S6
E	S2 S3 S4 S5

The Many to Many Solution



Student
S1
S2
S3
S4
S5
S6

Student	Course
S1	A
S1	B
S1	C
S2	D
S2	E
S3	A
S3	E
S4	B
S4	E
S5	A
S5	C
S5	E
S6	D

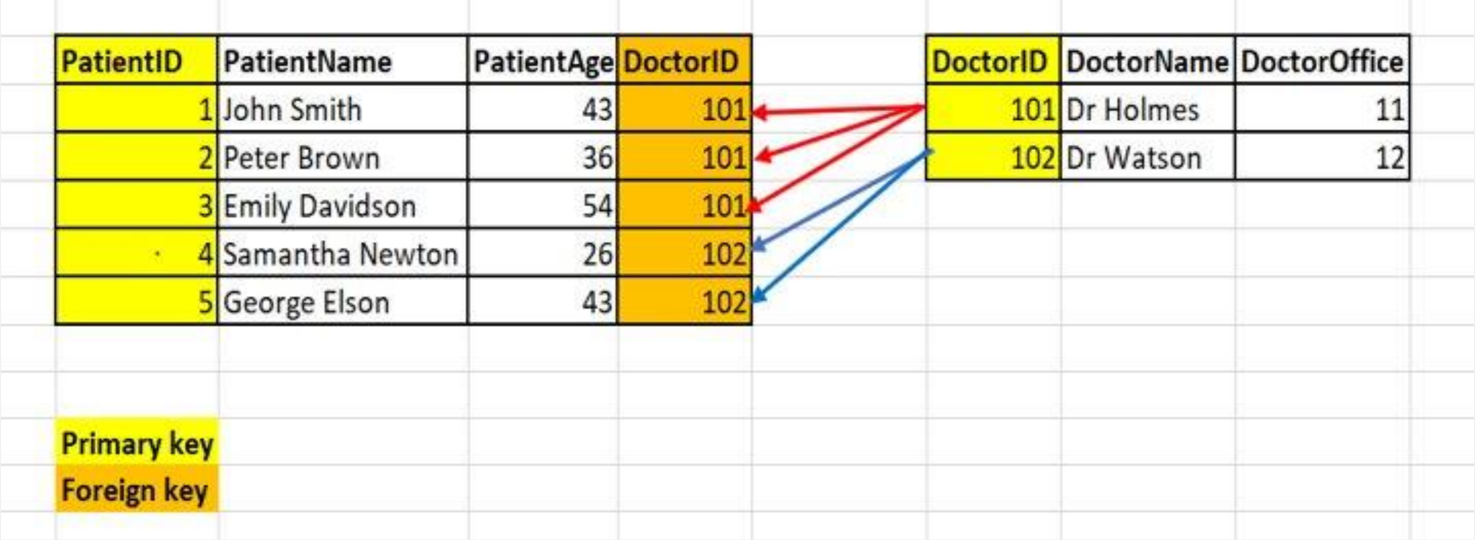
Course
A
B
C
D
E

Relational databases and relationships: One-to-many

The only native relationship for databases is One-to-Many, and modern databases would mostly contain this type of links. This is because:

- One-to-One relationships can be merged into a single table
- Many-to-Many relationships must be resolved

One-to-Many relationships are implemented using a mechanism of Primary and Foreign Keys. Primary Key (PK) – consists of one or more fields, which uniquely identify each record. Foreign Key (FK) allows to link two tables.



PatientID	PatientName	PatientAge	DoctorID
1	John Smith	43	101
2	Peter Brown	36	101
3	Emily Davidson	54	101
4	Samantha Newton	26	102
5	George Elson	43	102

DoctorID	DoctorName	DoctorOffice
101	Dr Holmes	11
102	Dr Watson	12

Primary key

Foreign key