Week 4
Databases – Conceptual Design

LM Data Structures, Algorithms, and Databases
(34141)

Dr Ahmad Ibrahim
a.ibrahim@bham.ac.uk
Feb 05, 2024

UNIVERSITY OF BIRMINGHAM | DUBAI دبي

Content based on lecture notes from Uday Reddy & Achim Jung

Topics by Week

Week	Date	Topic
1	15 Jan	Searching algorithms
2	22 Jan	Binary Search Tree
3	29 Jan	Balancing Trees – AVL Tree
4	5 Feb	Databases – Conceptual Design
5	14 Feb	Databases – Logical Design & Relational Algebra
6	19 Feb	Consolidation Week
7	26 Feb	Graph Algorithms
8	4 Mar	Sorting Algorithms
9	11 Mar	Hash tables
10	18 Mar	Databases – Normalization
		Easter break and Eid break
11	22 Apr	Databases – Concurrency
12	29 Apr	Revision Week

Timetable & Office hours

Day	Time	Event	Location
Monday	6:00-7:00pm	Online support session*	Online*
Tuesday	6:00-7:00pm	Office hour 1 (by appointment)*	Online*
Wednesday	-	-	-
Thursday	6:00-7:00pm	Office hour 2 (by appointment)*	Online*
Friday	6:00-8:00pm	Lecture	Auditorium
	8:00-9:00pm	Tutorial	Auditorium

UNIVERSITY OF BIRMINGHAM | DUBAI دبي

***Zoom link:** https://bham-ac-uk.zoom.us/j/81310444523?pwd=T01tZlZGdmdUL2lkeHZsVFpjcWxUUT09

Assessments

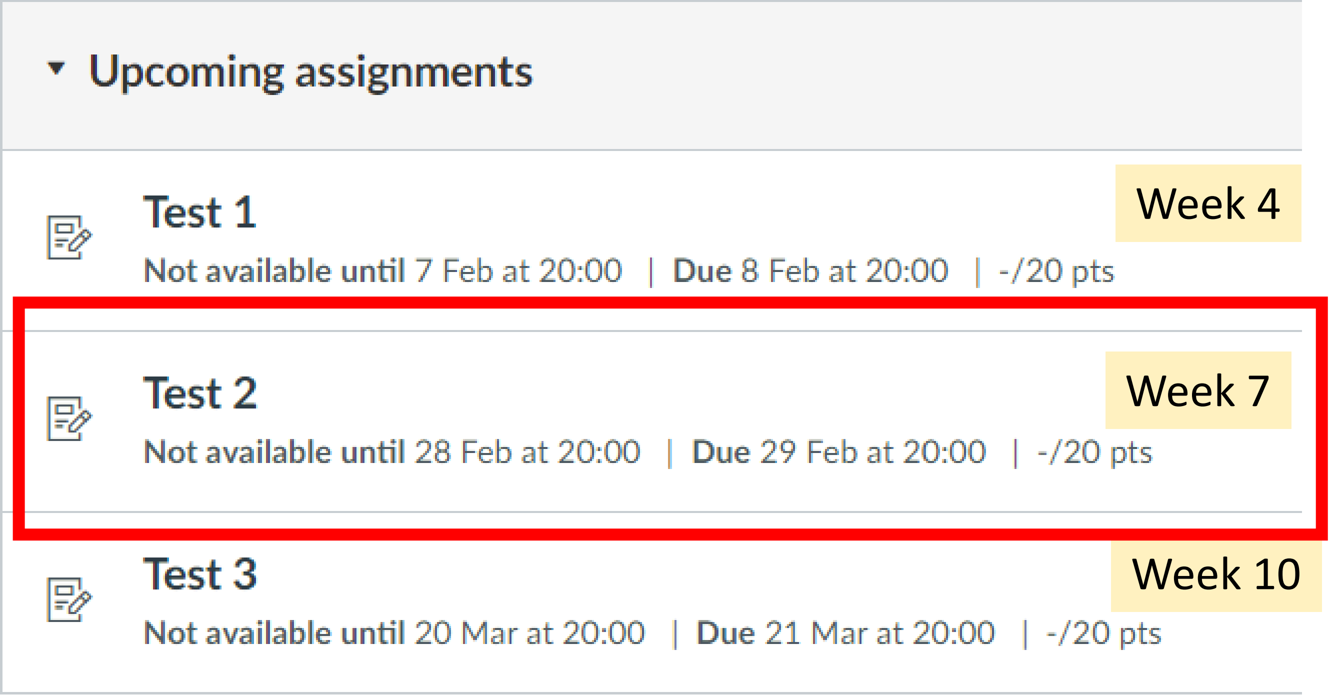
Assessments (Test 1, Test 2, Test 3): **20%**

Exam: **80%**

Late Submission Policy:
Submissions between 8:00-8:30pm (Dubai Time) on 08 Feb incur a 10% penalty.

Zero marks for submissions after 8:30pm.

Wellbeing approved cases: 1 day extension only.



▾ Upcoming assignments

Test 1 — Week 4
Not available until 7 Feb at 20:00 | Due 8 Feb at 20:00 | -/20 pts

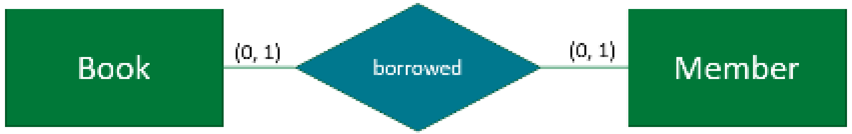
Test 2 — Week 7
Not available until 28 Feb at 20:00 | Due 29 Feb at 20:00 | -/20 pts

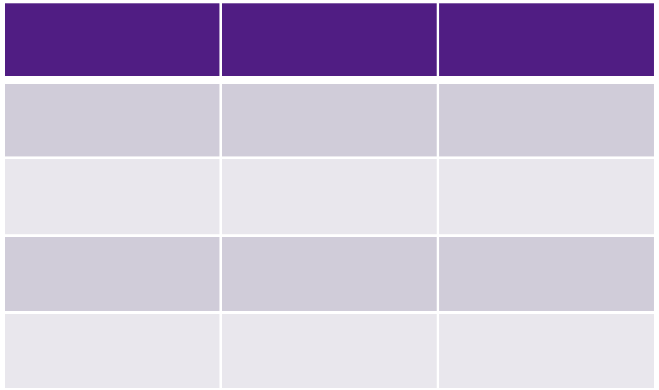
Test 3 — Week 10
Not available until 20 Mar at 20:00 | Due 21 Mar at 20:00 | -/20 pts

Review of Block-2

Week	Date	Topic
1	15 Jan	Searching algorithms
2	22 Jan	Binary Search Tree
3	29 Jan	Balancing Trees – AVL Tree
4	5 Feb	Databases – Conceptual Design
5	14 Feb	Databases –Logical Design, Relational Algebra
6	19 Feb	Consolidation Week

- Entities Relations Modelling
- Relational algebra
- SQL
- E-R models into table designs
- Examples



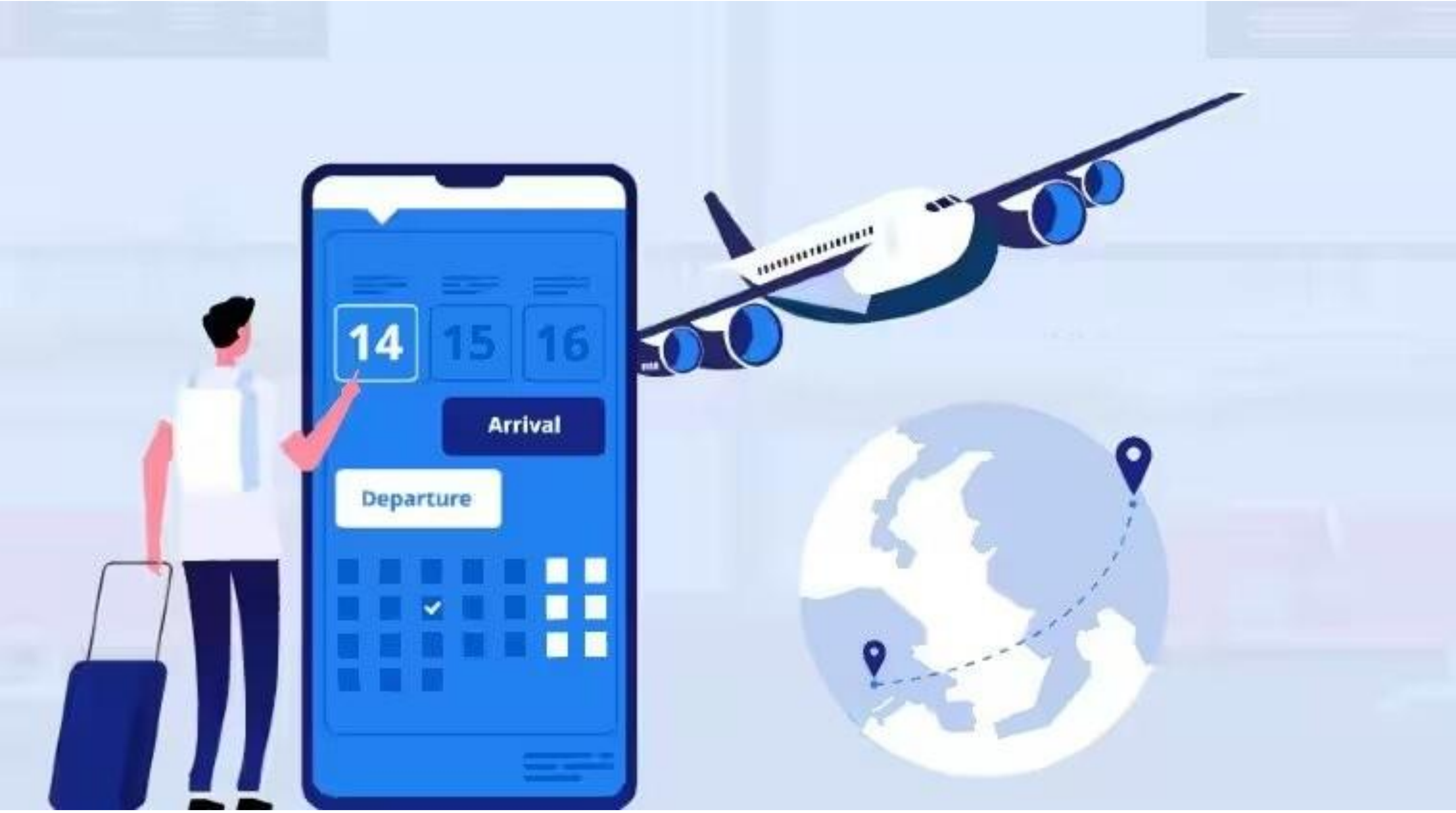


Select * from book where …

This Week

➡ Introduction

- Relational databases
- Entity-relationship (ER) diagram /Modeling
- Table design and creation, SQL
- Weak entities
- Hierarchies
- Table design (schemas) Optimisation
- SQL Commands + PostgreSQL


14
15
16
Arrival
Departure

CRUISE
HOTEL
CAR
FLIGHT
VACATION
INSURANCE
VISA

What is a database?

It is a large collection of persistent data

Typically stored on a server somewhere on the net

Accessible from multiple applications on client computers

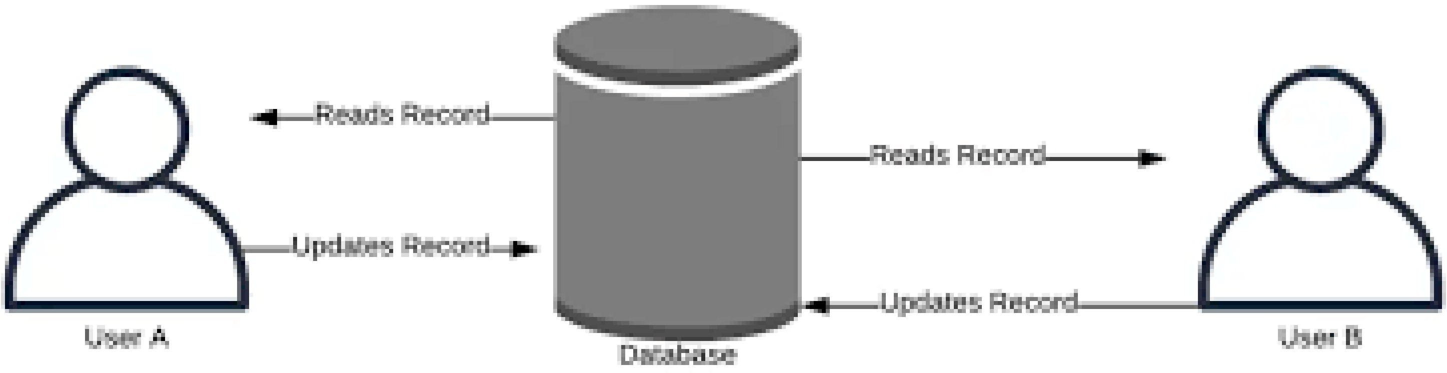
What is a database?

It is a large collection of persistent data

Typically stored on a server somewhere on the net

Accessible from multiple applications on client computers

➡ Concurrently accessible and modifiable



What is a database?

It is a large collection of persistent data

Typically stored on a server somewhere on the net

Accessible from multiple applications on client computers

Concurrently accessible and modifiable

➡ Expected to be secure.

What is a database?

It is a large collection of persistent data

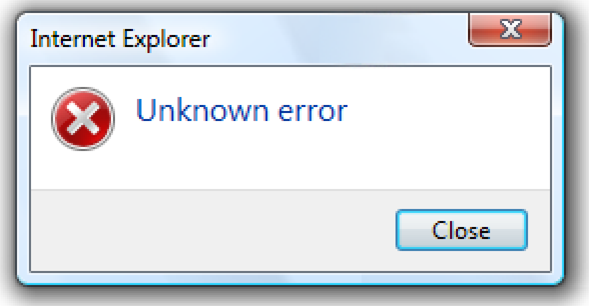
Typically stored on a server somewhere on the net

Accessible from multiple applications on client computers

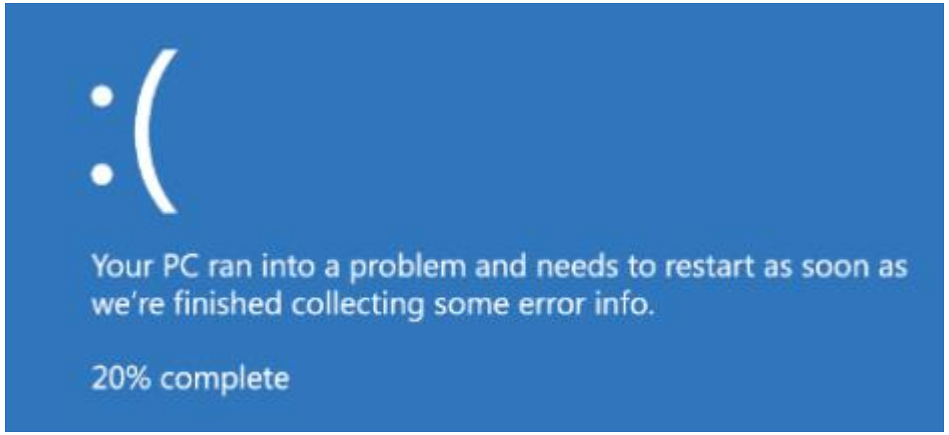
Concurrently accessible and modifiable

Expected to be secure.

Expected to be fault-tolerant (can recover from crashes). No data losses!



Unknown error



:(

Your PC ran into a problem and needs to restart as soon as we're finished collecting some error info.

20% complete

What is a database?

It is a large collection of persistent data

Typically stored on a server somewhere on the net

Accessible from multiple applications on client computers

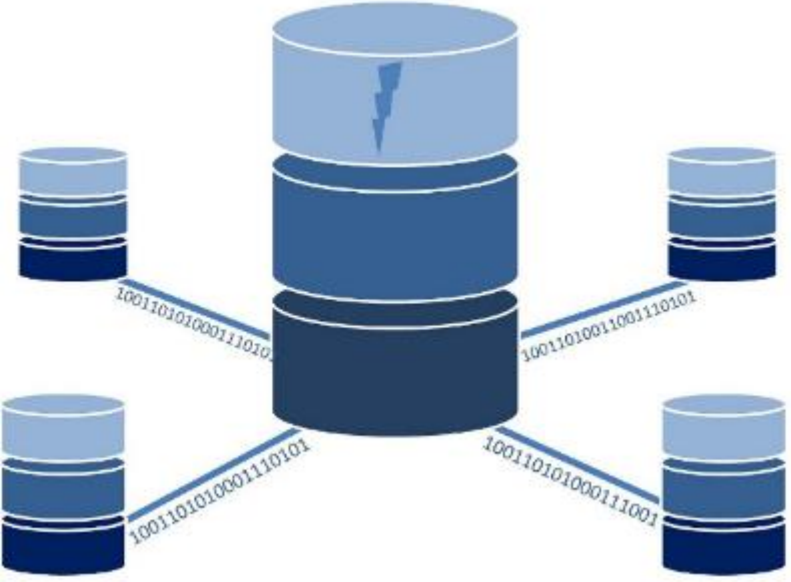
Concurrently accessible and modifiable

Expected to be secure.

Expected to be fault-tolerant (can recover from crashes). No data losses!

➡ Expected to be efficient

They were also called information system (old term, obsolete)



Why “databases”?

Meant to suggest the “base of data” on which all the applications run.

Typically meant for the internal data of organisations/businesses, but may also be used services provided by the organisation.

Examples:

- University: students, courses, marks, staff, resources, finance
- Shop: sales-items, customers, store, sales, staff, finance
- Library: books, customers, borrowings, stocks, publishers, staff
- Airline: airplanes, parts, airports, flights, customers, travel agents, reservations, staff, finance
- Manufacturing company: products, parts, stores, factories, customers, finance

Brief history

Stage 1: Sequential access files (tape drives)

Brief history

Stage 1: Sequential access files (tape drives)

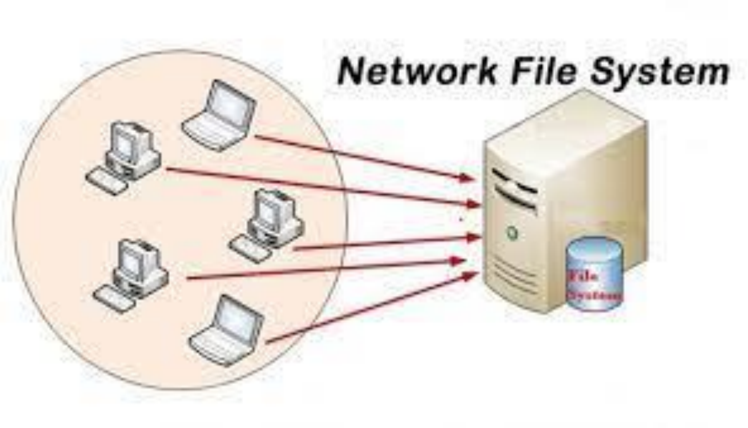
Stage 2: Random-access files (disk drives)

Brief history

Stage 1: Sequential access files (tape drives)

Stage 2: Random-access files (disk drives)

Stage 3: Hierarchically-structured or networked file structures.



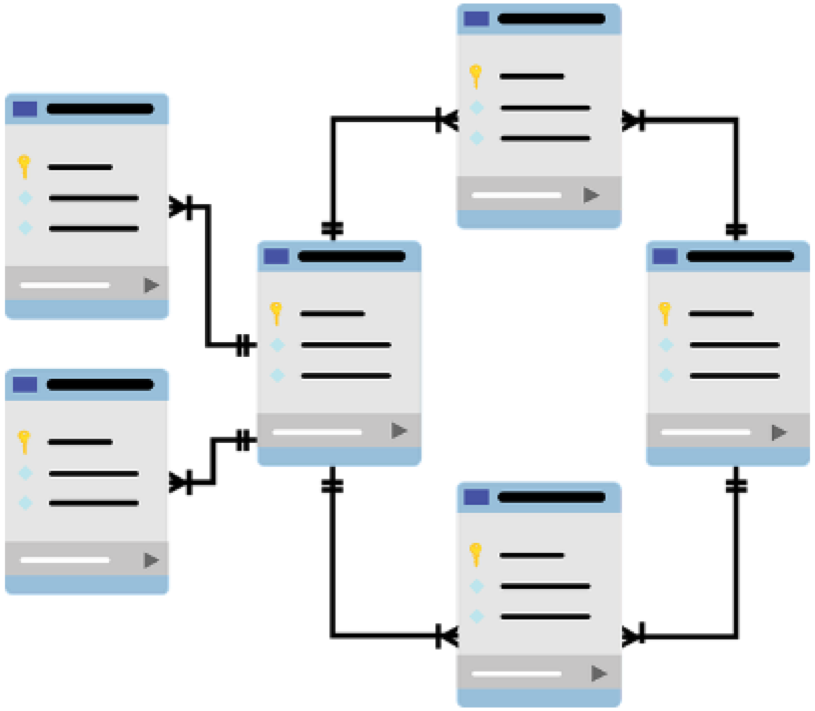
Brief history

Stage 1: Sequential access files (tape drives)

Stage 2: Random-access files (disk drives)

Stage 3: Hierarchically-structured or networked file structures.

Stage 4: Relational databases.



Brief history

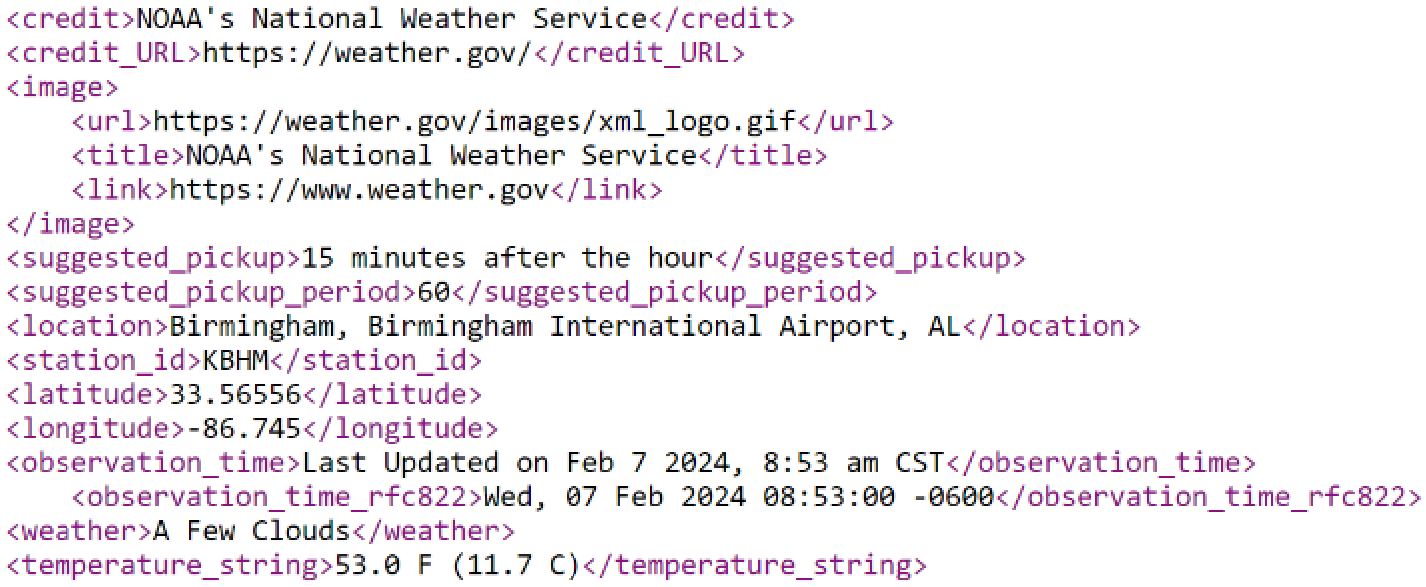
Stage 1: Sequential access files (tape drives)

Stage 2: Random-access files (disk drives)

Stage 3: Hierarchically-structured or networked file structures.

Stage 4: Relational databases.

Stage 5: XML databases/documents



```xml
<credit>NOAA's National Weather Service</credit>
<credit_URL>https://weather.gov/</credit_URL>
<image>
    <url>https://weather.gov/images/xml_logo.gif</url>
    <title>NOAA's National Weather Service</title>
    <link>https://www.weather.gov</link>
</image>
<suggested_pickup>15 minutes after the hour</suggested_pickup>
<suggested_pickup_period>60</suggested_pickup_period>
<location>Birmingham, Birmingham International Airport, AL</location>
<station_id>KBHM</station_id>
<latitude>33.56556</latitude>
<longitude>-86.745</longitude>
<observation_time>Last Updated on Feb 7 2024, 8:53 am CST</observation_time>
    <observation_time_rfc822>Wed, 07 Feb 2024 08:53:00 -0600</observation_time_rfc822>
<weather>A Few Clouds</weather>
<temperature_string>53.0 F (11.7 C)</temperature_string>
```

Brief history

Stage 1: Sequential access files (tape drives)

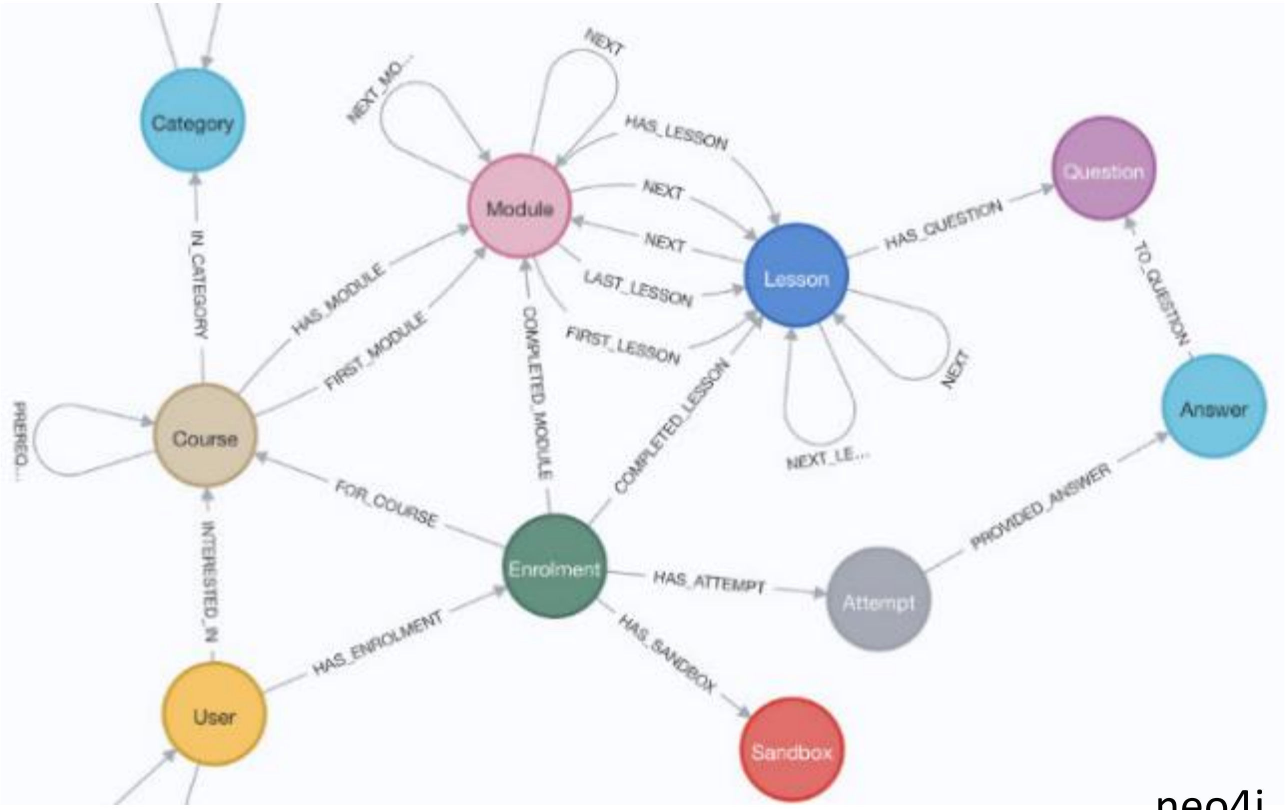
Stage 2: Random-access files (disk drives)

Stage 3: Hierarchically-structured or networked file structures.

Stage 4: Relational databases.

Stage 5: XML databases/documents

Stage 6: No-SQL (Graph Database)



neo4j

Harvard Business Review

Analytics And Data Science | Bad Data Costs the U.S. $3 Trillion Per Year

Analytics And Data Science

Bad Data Costs the U.S. $3 Trillion Per Year

by Thomas C. Redman

September 22, 2016

https://hbr.org/2016/09/bad-data-costs-the-u-s-3-trillion-per-year

This Week

- Introduction
- ➡ Relational databases
- Entity-relationship (ER) diagram /Modeling
- Table design and creation, SQL
- Weak entities
- Hierarchies
- Table design (schemas) Optimisation
- SQL Commands + PostgreSQL

Relational databases - history

Invented by Edgar Codd, at IBM Toronto research centre, 1970.

Key idea: Most of the data is highly “regular”, i.e., many items of the same kind, e.g., students in a University, products in a manufacturing.

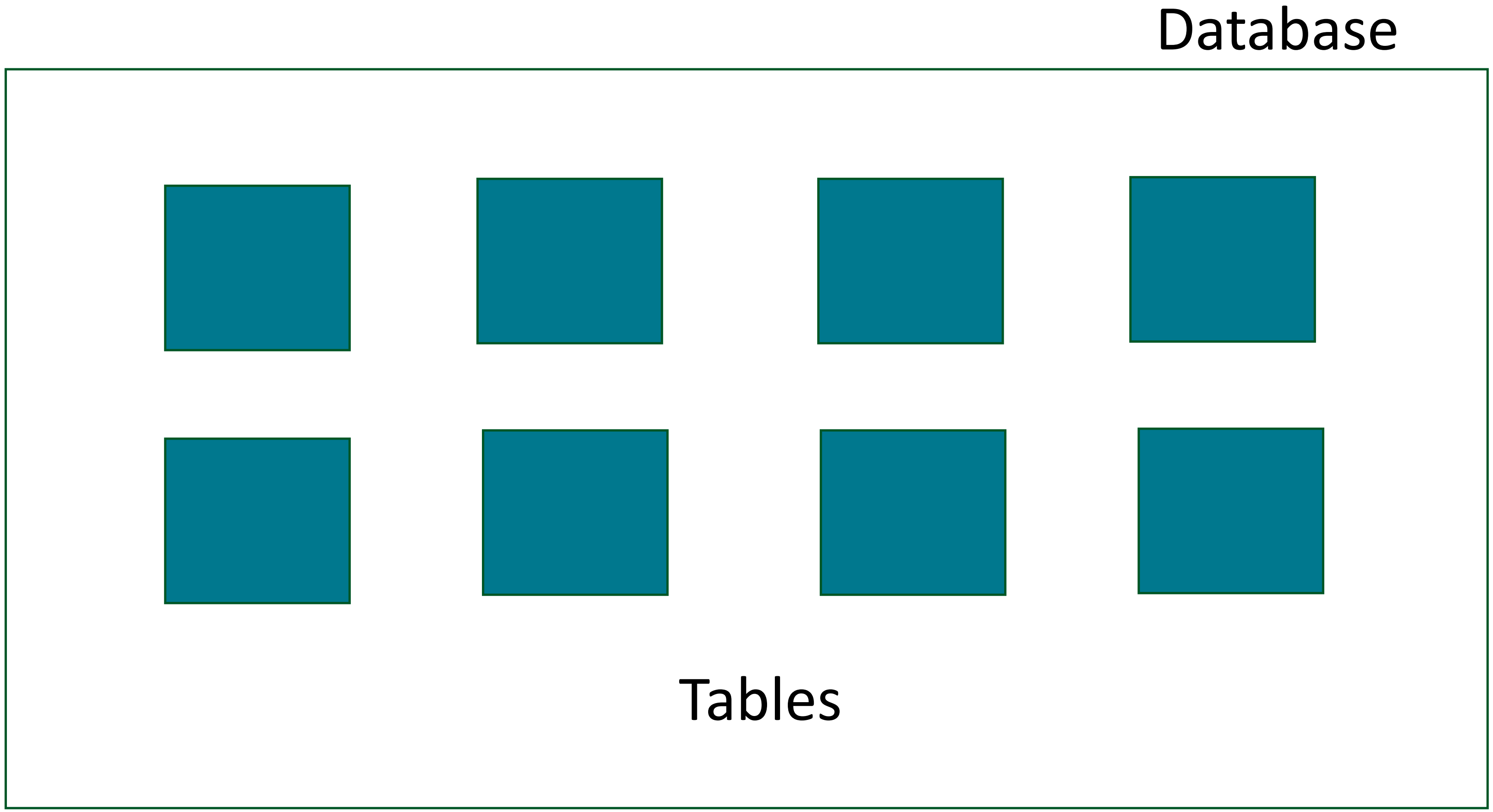
The regularity allows us to provide a simplified conceptual view, making it easy to access/modify the data.

Built as a relational database management system (DBMS or RDBMS)

Providers:

- IBM, Ingres, Oracle
- Microsoft, **Postgres**, MySQL

Relational Databases – Physical Structure



Tables ("Relations")

The textbook uses the term "Relations" a lot.

First name	Last name	Office	Phone number	Email address
Uday	Reddy	210	43740	-
Ahmad	Ibrahim	Dubai Campus	-	-
Mirco	Giacobbe	208	-	-
Jizheng	Wan	-	-	-

Rows

(Records, tuples)

Columns

(Attributes, fields)

The entries marked "-" are called "null" entries

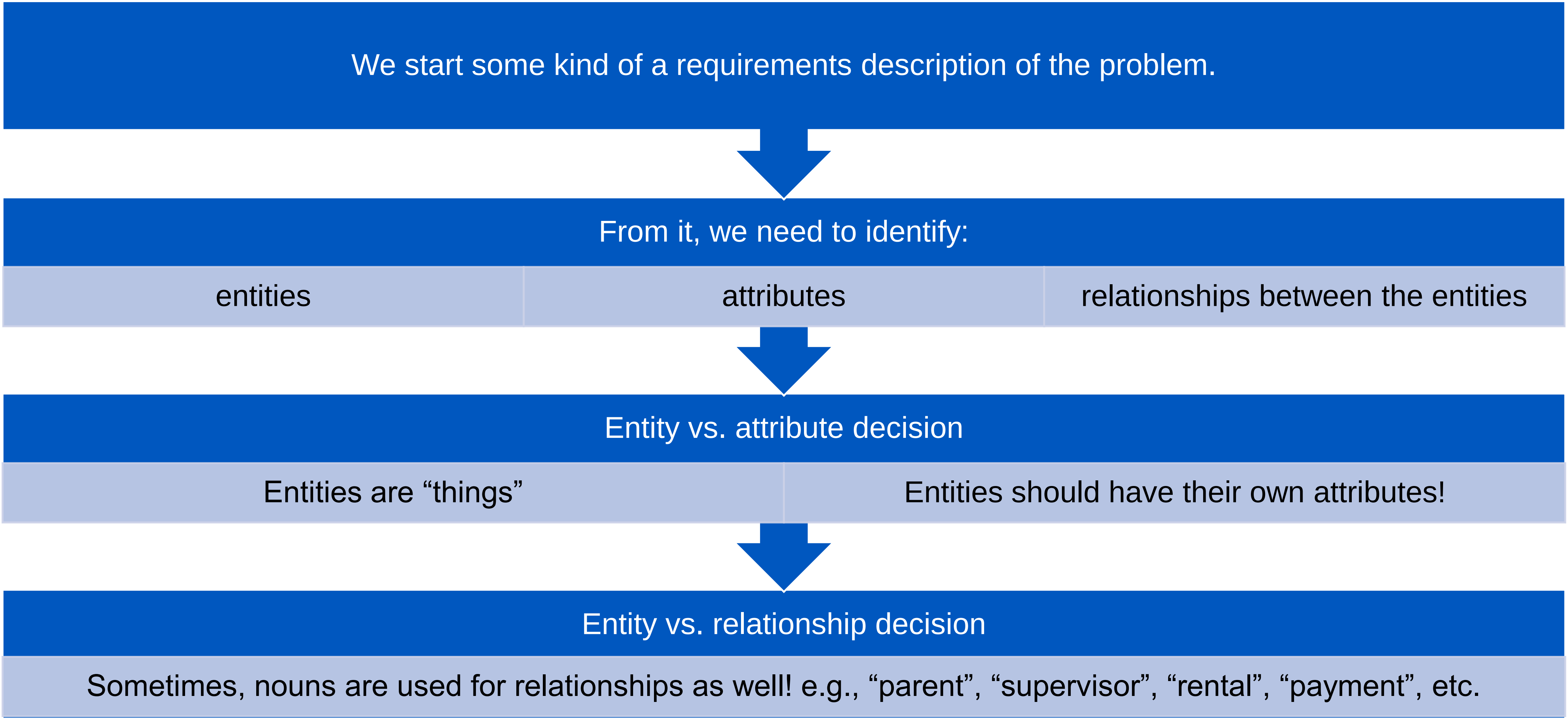
Note that all the rows have the same structure.
But the columns can be quite different from each other.

This Week

- Introduction
- Relational databases
- ➡ Entity-relationship (ER) diagram /Modeling
- Table design and creation, SQL
- Weak entities
- Hierarchies
- Table design (schemas) Optimisation
- SQL Commands + PostgreSQL

UNIVERSITY OF BIRMINGHAM | DUBAI دبي

Entity-relationship (ER) diagram



We start some kind of a requirements description of the problem.

From it, we need to identify:

entities

attributes

relationships between the entities

Entity vs. attribute decision

Entities are “things”

Entities should have their own attributes!

Entity vs. relationship decision

Sometimes, nouns are used for relationships as well! e.g., “parent”, “supervisor”, “rental”, “payment”, etc.

Requirements description of the problem:

Library management system

We would like to build a library management system that can keep a record of books, copies of the books, and the currently borrowed books.

Each library member can borrow only one book at a time for a certain duration. A late penalty/fine will be applied in case of a late return.

Entities - examples

- People (roles)
    - Student, Lecturer, Staff member, engineer, doctor, patient, customer, passenger, …
- Objects
    - Cars, airplanes, products, parts, sales items, offices, buildings, …
- Organisations
    - Companies, suppliers, departments, clubs, committees, …
- Conceptual
    - Course, degree programme, project, design, exhibition, …
- Events
    - Course deliveries, lectures, exams, concerts, sales, …

Library management system: entities

*We would like to build a **library management system** that can keep a record of books, copies of the books, and the currently borrowed books.*

Each library member can borrow only one book at a time for a certain duration. A late penalty/fine will be applied in case of a late return.

Example – Library management system

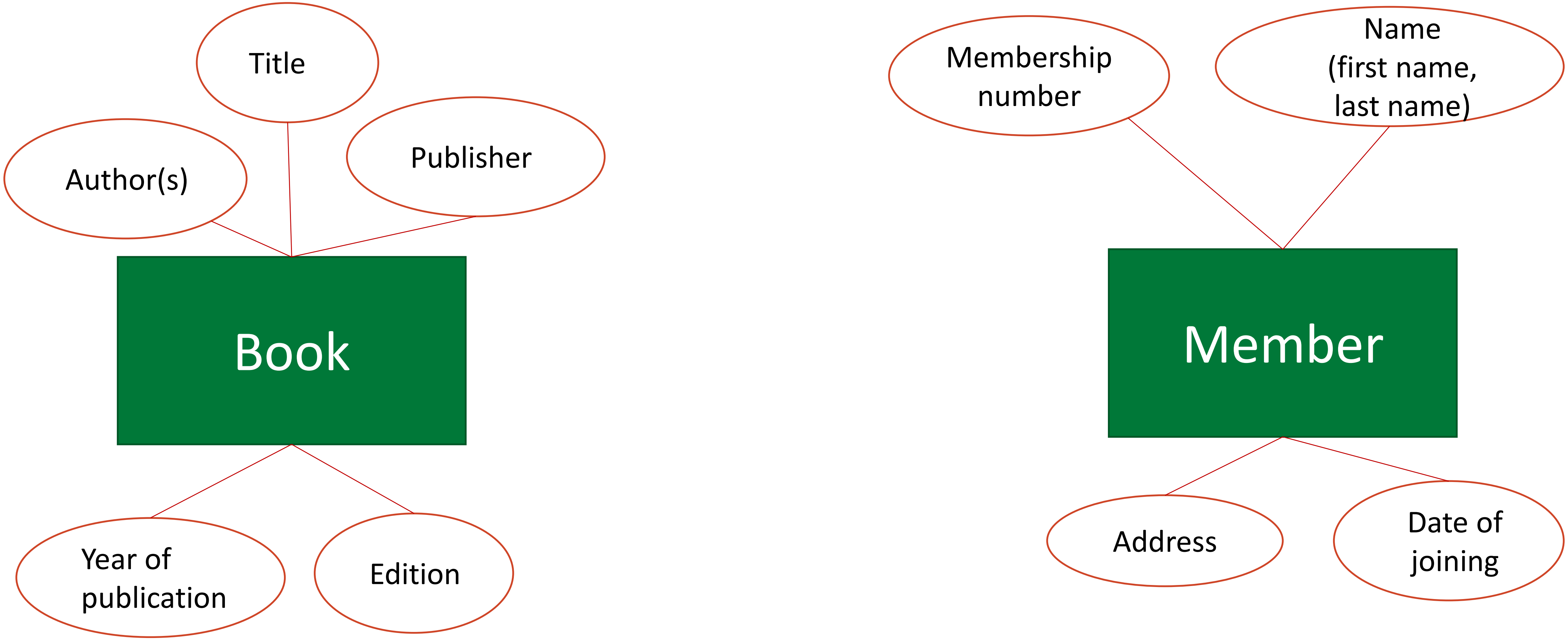
Book
Member

We depict entities in our design by rectangles.

Library management system: Attributes for entities

- Book
  - Author(s)
  - Title
  - Publisher
  - Year of publication
  - Edition
- Member
  - Membership number
  - Name (first name, last name)
  - Address
  - Date of joining

Example – Library management system



Relationships between entities

It is these relationships that tie the tables together and are the key to doing database design *correctly*.

For example:

- Students *register for* courses.
- Students *sit for* exams (and receive marks).
- These exams *assess* particular courses.
- Courses *constitute* degree programmes.
- Lecturers *teach* courses.
- Lecturers *set* exams (and mark them).

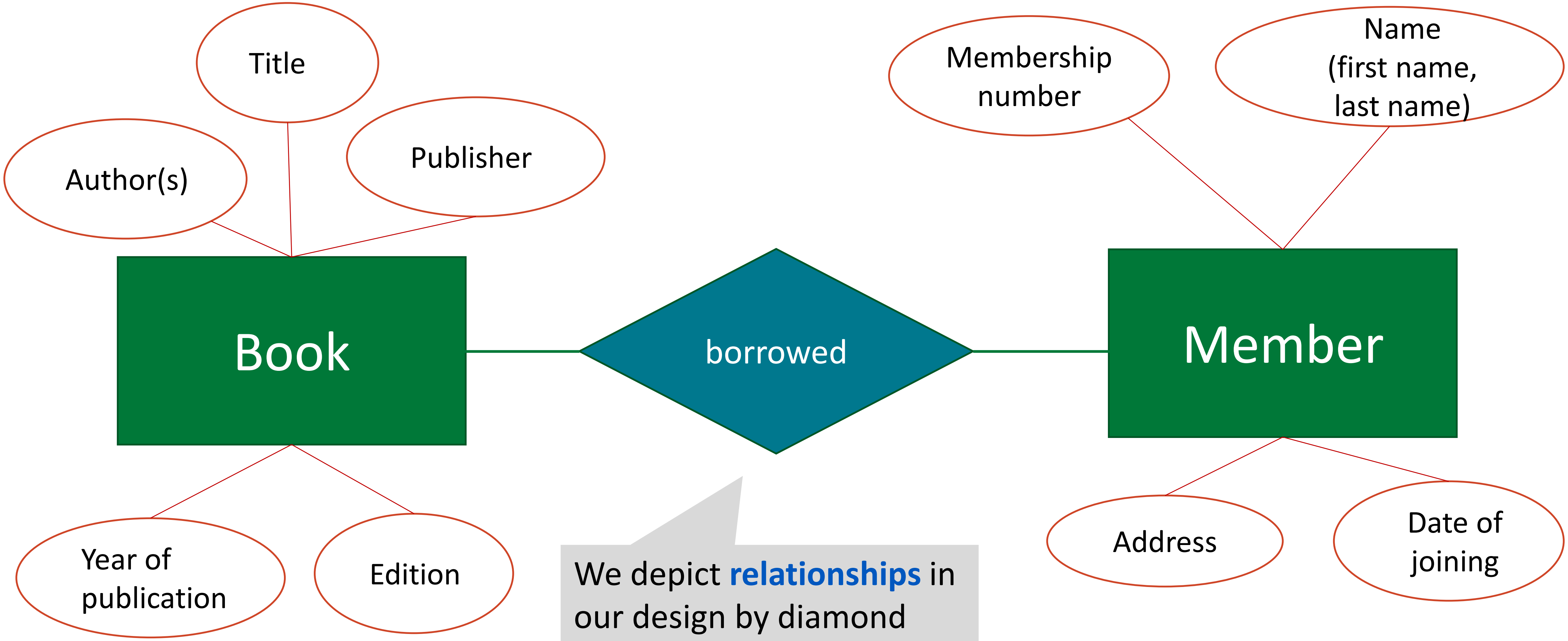
- Human relationships
  - Spouse, offspring, parent, manager, client, …
- Role relationships
  - Teaching, studying, supervising (a department), managing (a project), selling, buying, borrowing, …
- Organic relationships
  - Belongs to, Part of, Located at, …
- Event occurrences
  - Lecturing at, concert at, …

Library management system: Relationships

*We would like to build a **library management system** that can keep a record of books, copies of the books, and the currently borrowed books.*

Each library member can borrow only one book at a time for a certain duration. A late penalty/fine will be applied in case of a late return.

Example – Library management system



We depict **relationships** in our design by diamond boxes, with links going to all the entities they relate.

Attributes (for both entities and relationships)

- Book
  - Author(s)
  - Title
  - Publisher
  - Year of publication
  - Edition
- Member
  - Membership number
  - Name (first name, last name)
  - Address
  - Date of joining

borrowed

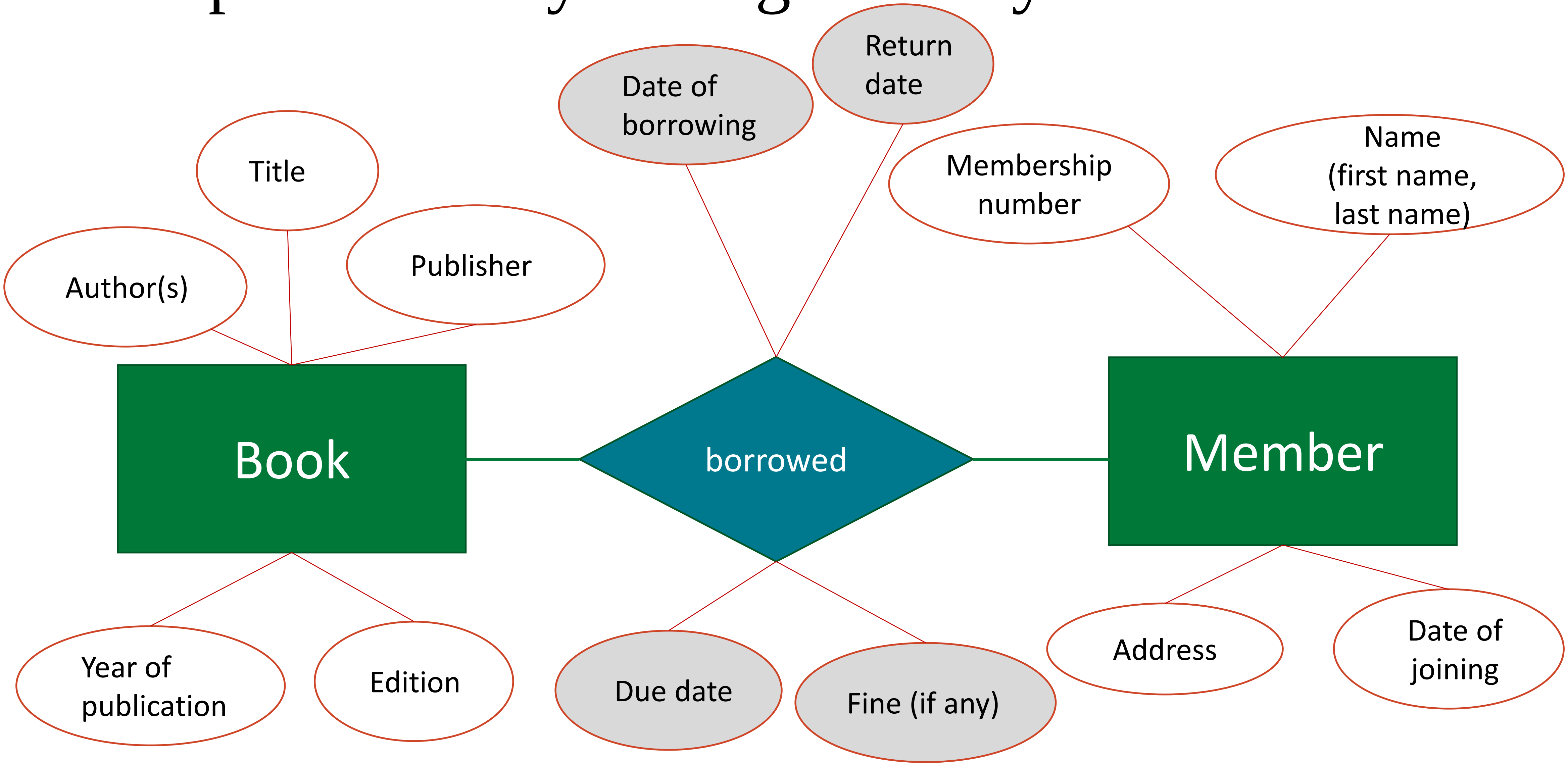
Date of borrowing

Due date

Return date

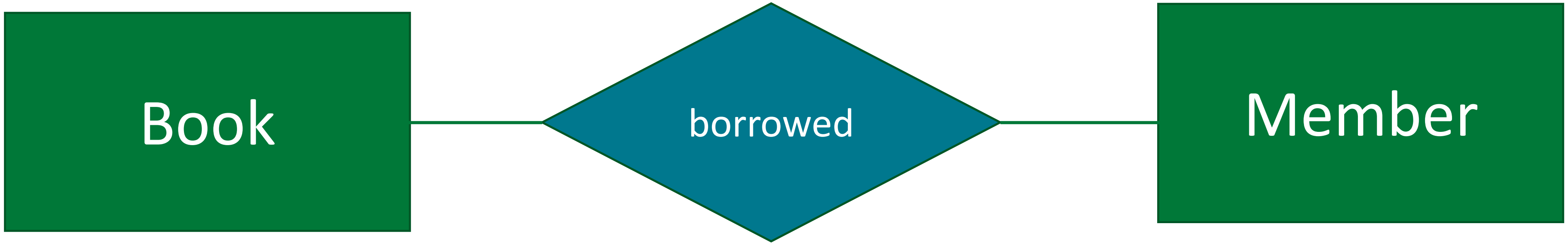
Fine (if any)

Example – Library management system

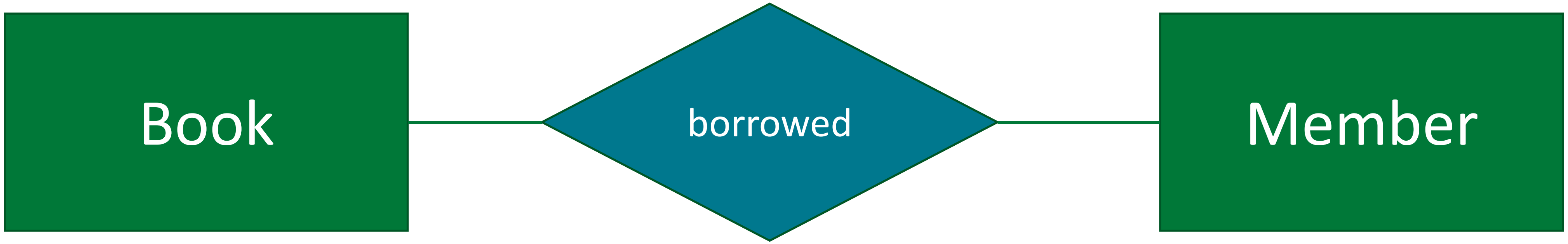


Example – Library management system

We prefer not to put attributes in the ER diagrams.
They make the diagram look too crowded and noisy.



Example – Library management system



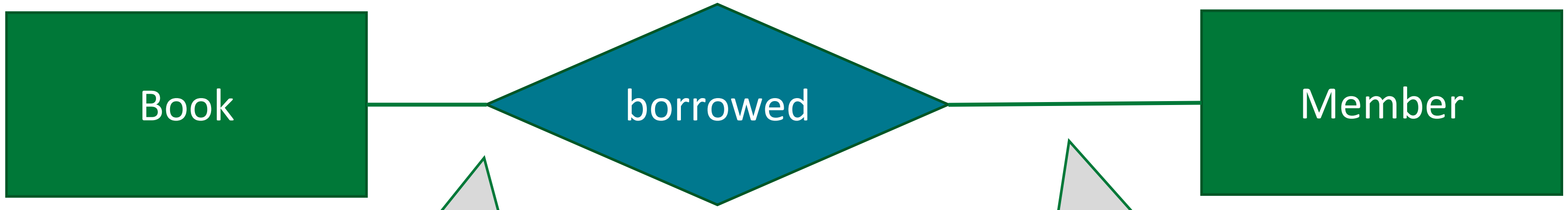
We assume:

- There is only one copy of each book in the library.
- Member can borrow only one book at a time.
- We only record the current borrowings (After return the information is deleted.)

How can we represent this information?

Multiplicities (Cardinalities)

For every relationship and every side (entity) of the relationship, we think about: minimum, and maximum number of times the entity can participate in the relationship.



A book may or may not be borrowed.

If it is borrowed, there is only one borrowing.

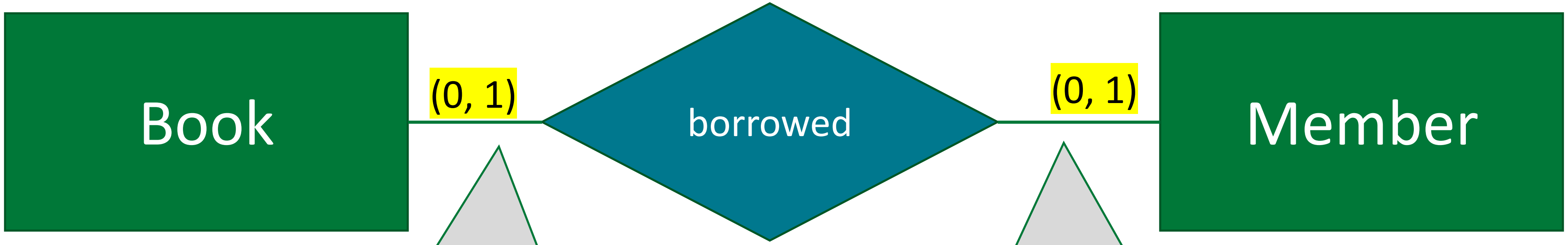
So, minimum = 0, maximum = 1.

A member may or may not borrow a book.

But if they borrowed, they borrowed only one book. So, minimum = 0, maximum = 1.

Example – Library management system (with multiplicities)

Each **side** of a relationship (link connecting to an entity) is given a multiplicity, specifying the minimum and maximum number of times the entity can participate in the relationship.



A book may or may not be borrowed.

If it is borrowed, there is only one borrowing.

So, minimum = 0, maximum = 1.

A member may or may not borrow a book.

But if they borrowed, they borrowed only one book. So, minimum = 0, maximum = 1.

Additional Example: Database about Movies

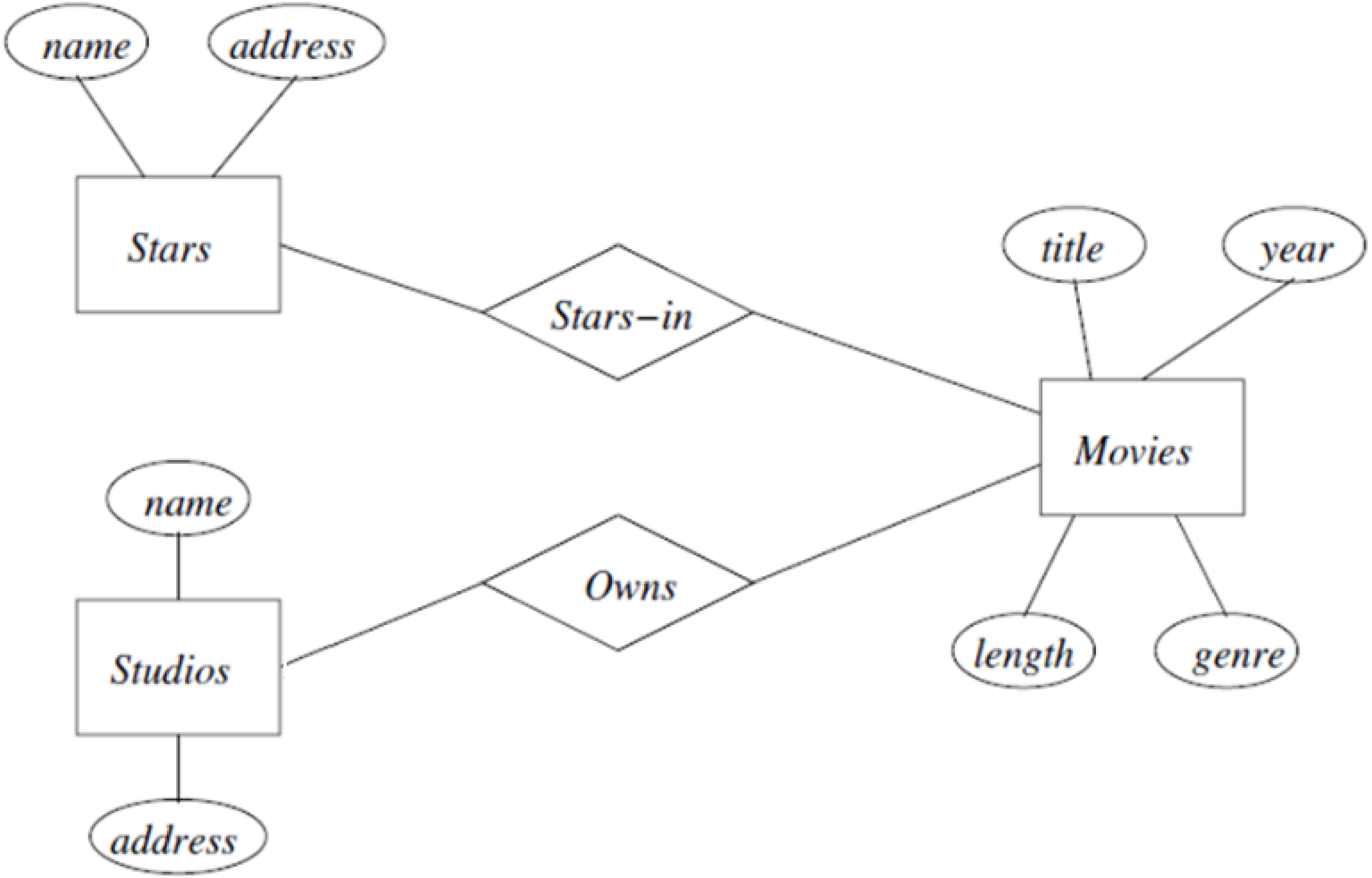


Figure 2: An entity-relationship diagram for the movie database

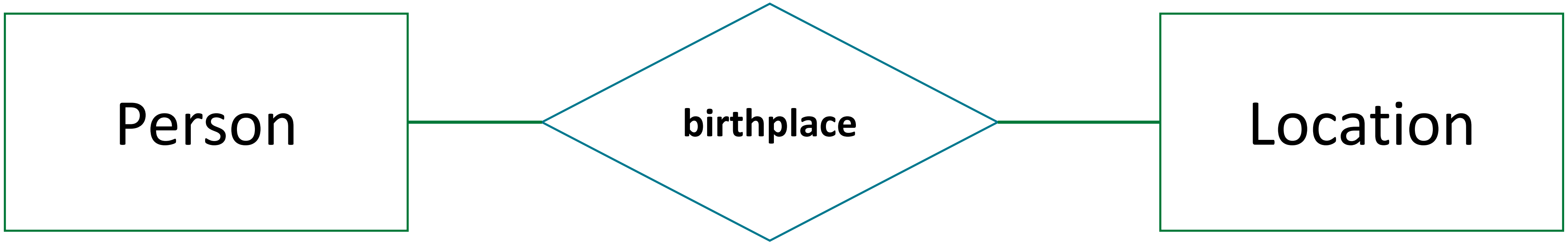
- “Stars”, “Movies” and “Studios” are entities.
- “Stars-in” and “Owns” are relationships.
- attributes are:
  - name, address
  - title, year, length, genre
  - name, address

Additional Example: Database about Birthplaces

Every person is born in some place.

A location may have no persons born there or many persons born

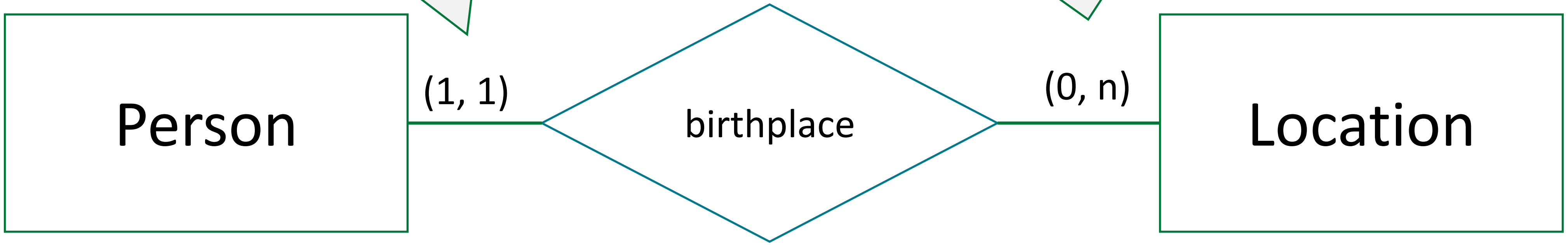
What about multiplicity?



Additional Example: Database about Birthplaces

A location may have no persons born there or many persons born there. So minimum = 0, maximum no limit.

Every person is born in some place: minimum = maximum = 1



Person

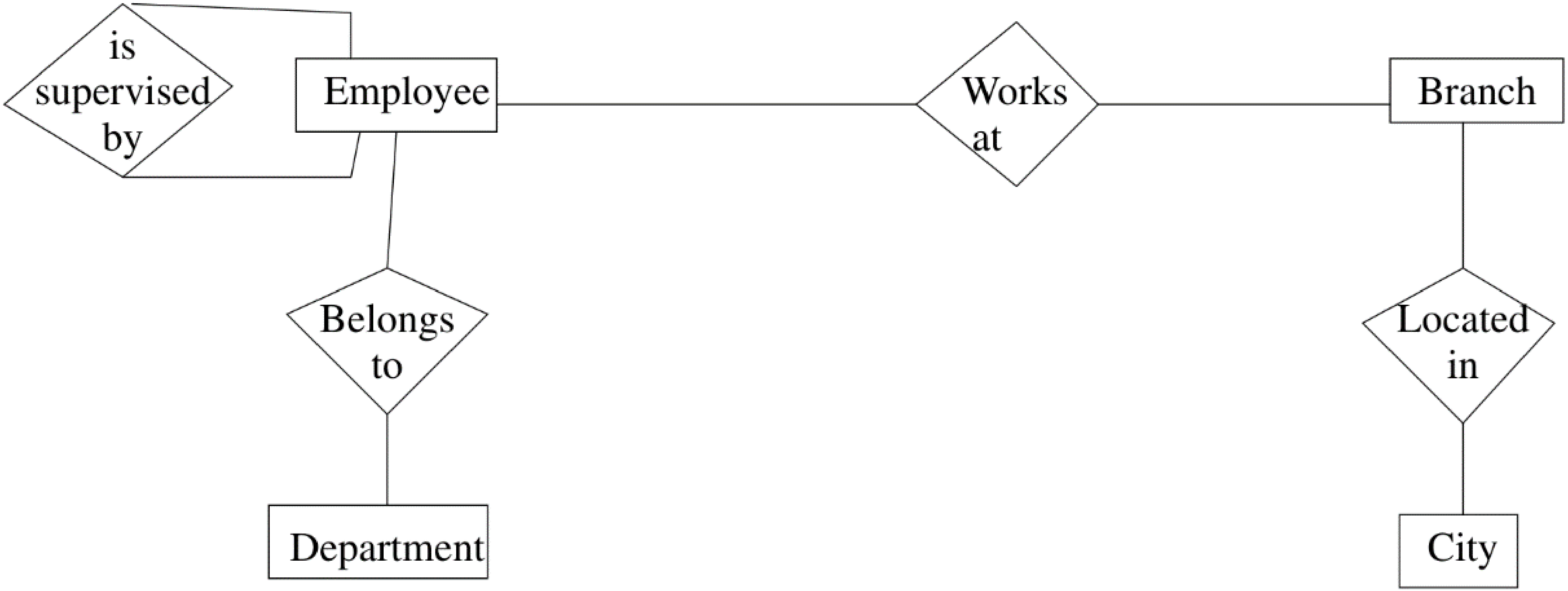
(1, 1)

birthplace

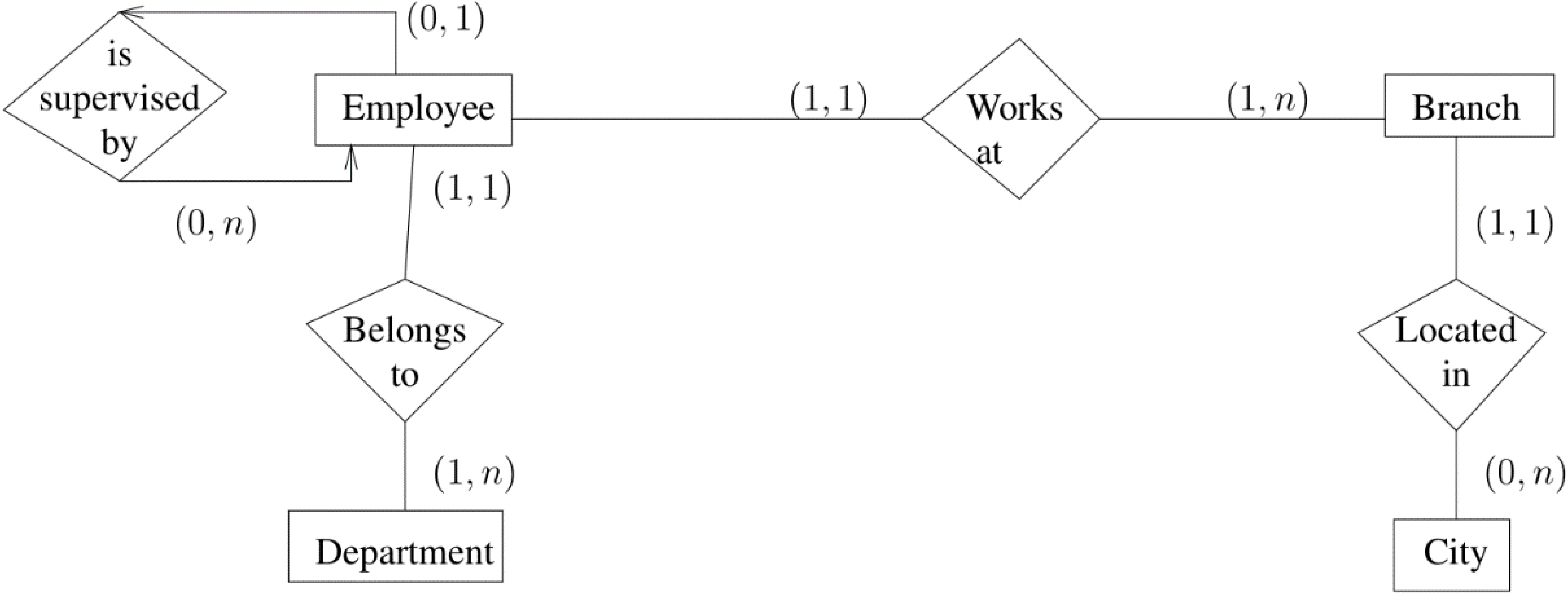
(0, n)

Location

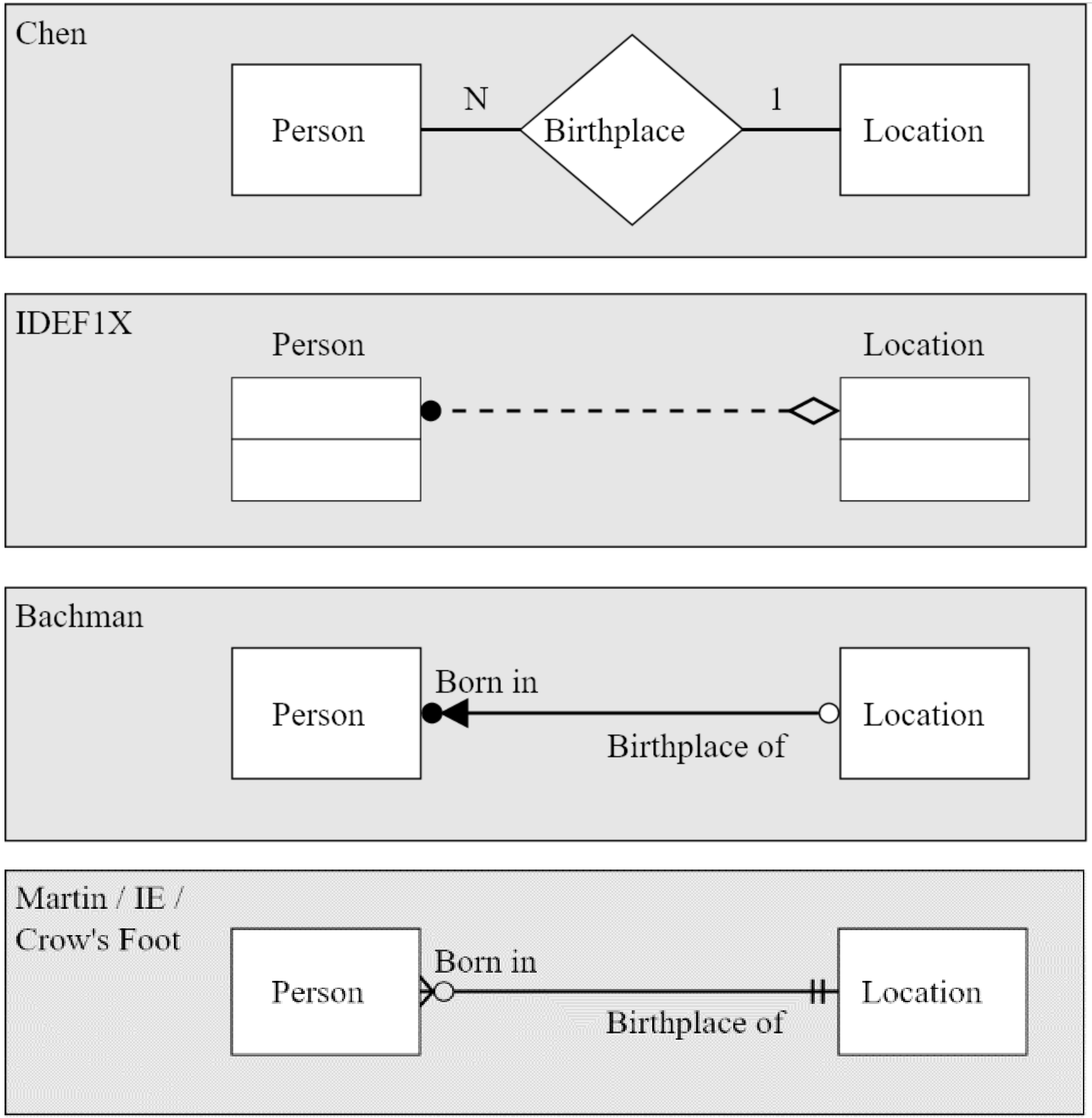
Additional Example: human resources database



Additional Example: human resources database (multiplicities)



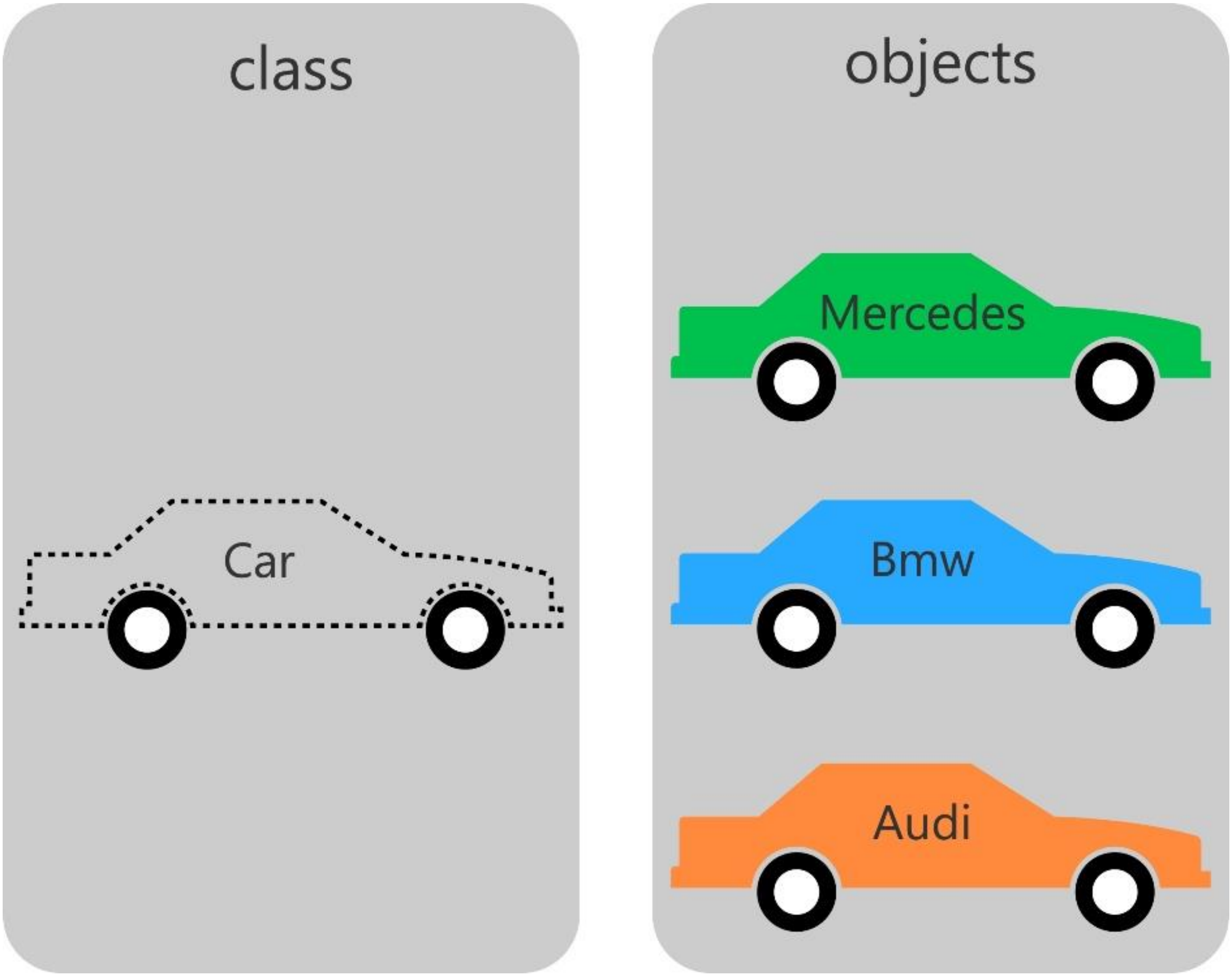
Multiplicity: A plethora of other notations



DO NOT USE
IN THIS MODULE!

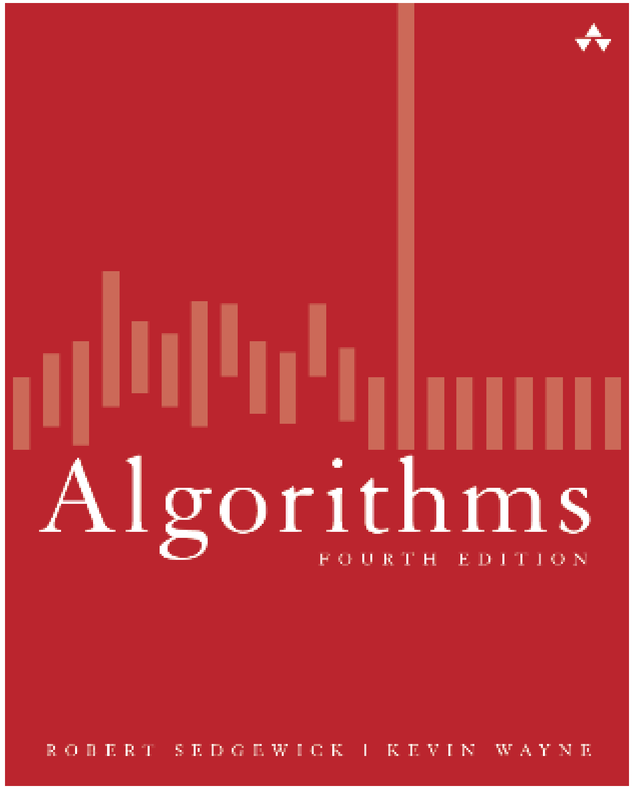
- The more modern notations omit the diamond box for relations.
- They draw a line for the relation, and sometimes label the line and sometimes not.
- Sometimes two labels are placed at the two ends.

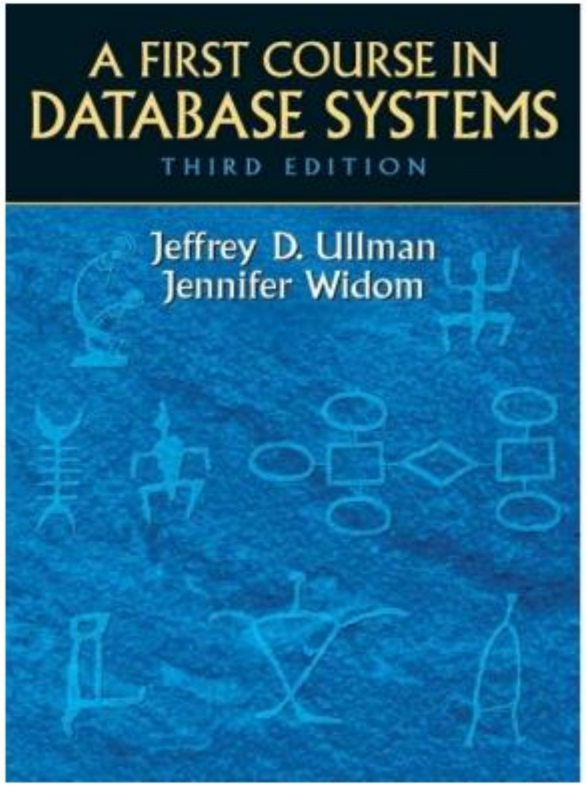
What exactly do you mean by “entity”, “entity set”?

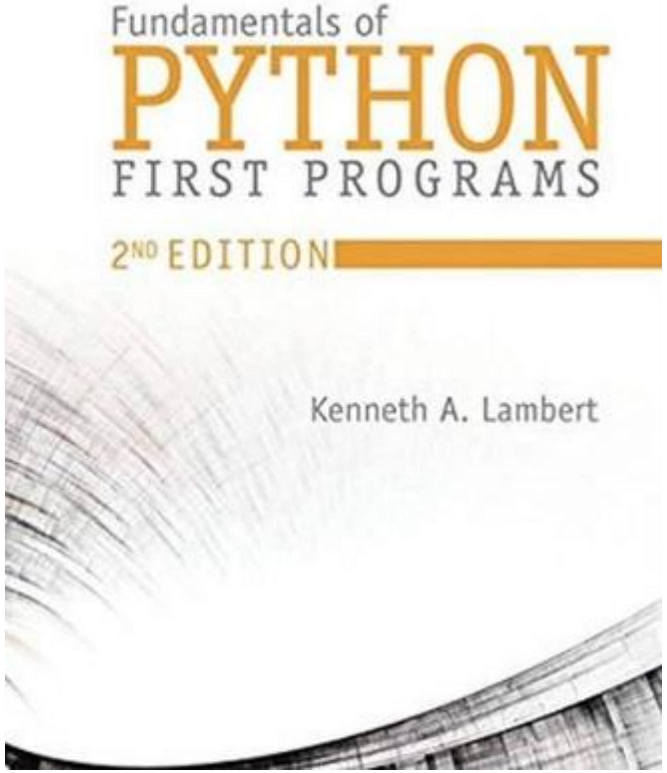


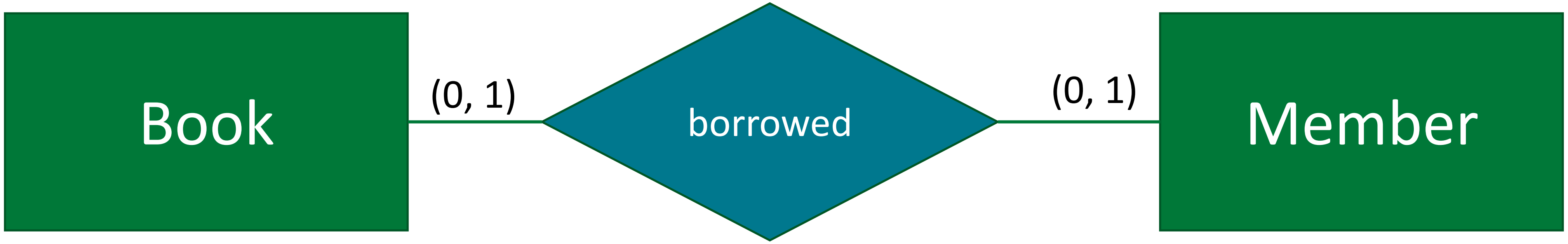
entity set entity entity entity

What exactly do you mean by “entity”, “entity set”?









This Week

- Introduction
- Relational databases
- Entity-relationship (ER) diagram /Modeling
- Table design and creation, SQL
- Weak entities
- Hierarchies
- Table design (schemas) Optimisation
- SQL Commands + PostgreSQL

Table design and creation

By default, every entity and relationship in our ER model becomes a table in the database.

(This can be optimised. We will get to that later.)

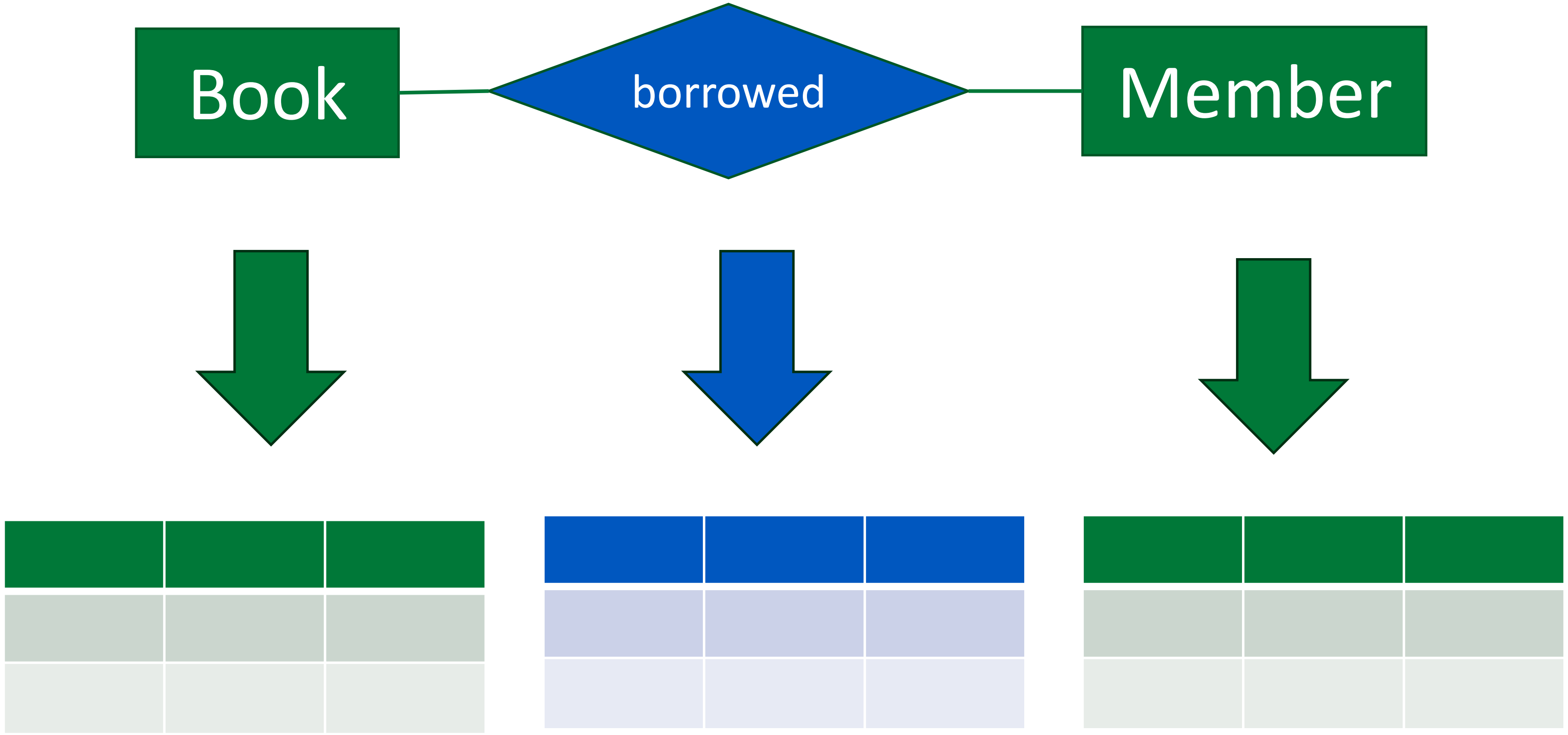
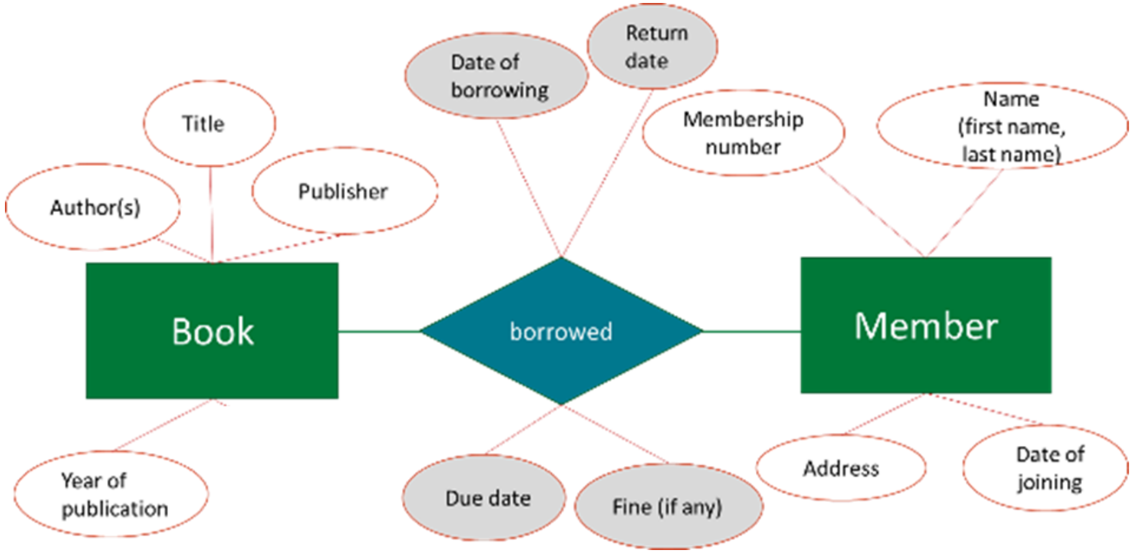


Table design (schemas)



Unique
(no duplicate)

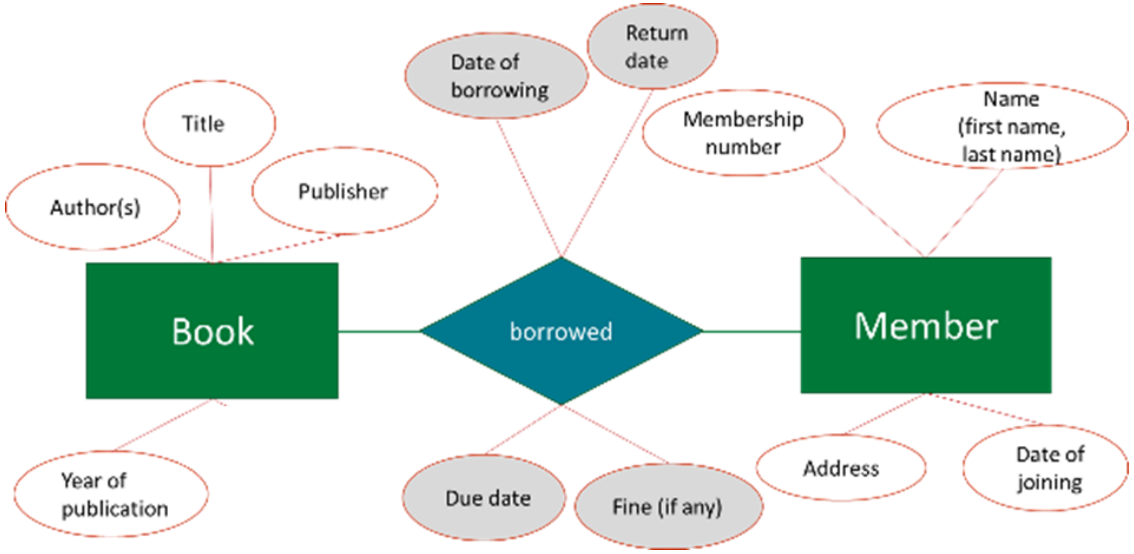
The table designs are written in the schema notation.

Book(book id, authors, title, publisher, year)

Member(member num, last name, first name, address, date of joining)

Borrow(book id, member num, date, due date, return date, fine)

Table design (schemas)



The table designs are written in the schema notation.

Unique identifiers (no duplicate)

Identifier in ER language

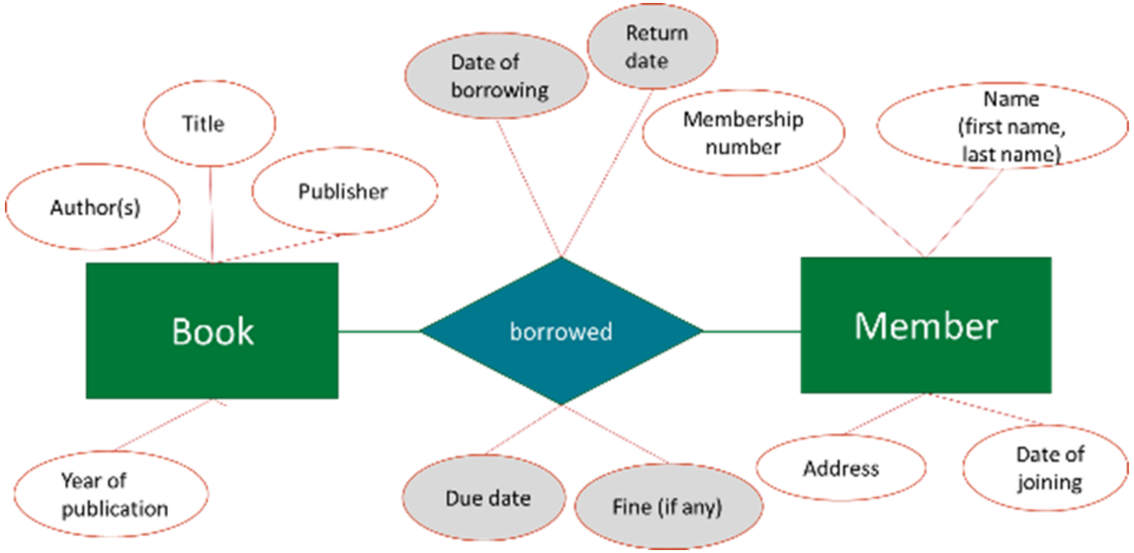
Primary key in Table

Book(<u>book id</u>, authors, title, publisher, year)

Member(<u>member num</u>, last name, first name, address, date of joining)

Borrow(<u>book id, member num</u>, date, due date, return date, fine)

Table design (schemas)



The table designs are written in the schema notation.

Book(<u>book id</u>, authors, title, publisher, year)

Member(<u>member num</u>, last name, first name, address, date of joining)

Borrow(<u>book id, member num</u>, date, due date, return date, fine)

2 fields of primary key

must necessarily occur in the Book and Member tables (constraint)

Relational DBMS (SQL)

A database management system (DBMS) is a software system (made by providers like Oracle, Postgres, MySQL).

- Creating tables.
- Querying tables for finding information.
- Adding, deleting or modifying records in tables.
- Ensuring that constraints continue to be satisfied during modifications.

Constraints example:
Two books should not have same ID number

Member can borrow only 1 book. Database should not have 2 records for the same member at the same time. System should give error, when trying to insert 2nd record for the same member

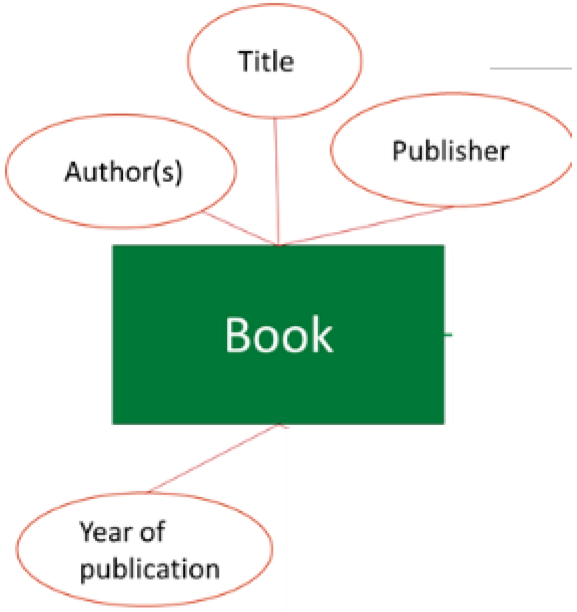
All this is done using a "little" programming language called SQL ("Structured Query Language").

SQL command to create the book table

Book(book id, authors, title, publisher, year)

```
create table book (bookid   integer not null unique,
                   authors  character varying(100),
                   title    character varying(40),
                   publisher character varying(20),
                   year     integer,

                   primary key (bookid));
```



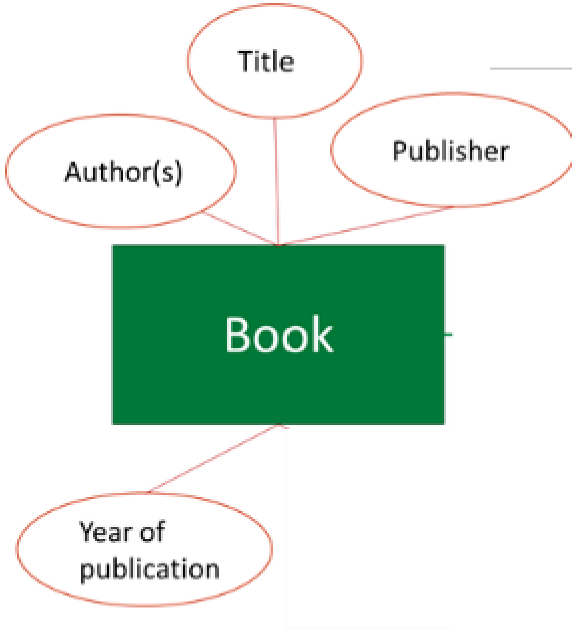
- The **unique** keyword ensures that a particular bookid occurs in only one record in the book table.
- **“not null”** says this field cannot be null. All fields can be null by default (a bad feature of SQL!)
- The **primary key** declaration says that the bookid field can be used for uniquely identify records.

SQL command to create the book table

Book(book id, authors, title, publisher, year)

```
create table book (bookid   integer not null unique,
                   authors  character varying(100),
                   title    character varying(40),
                   publisher character varying(20),
                   year     integer,

                   primary key (bookid));
```



Values that can be stored in this table:

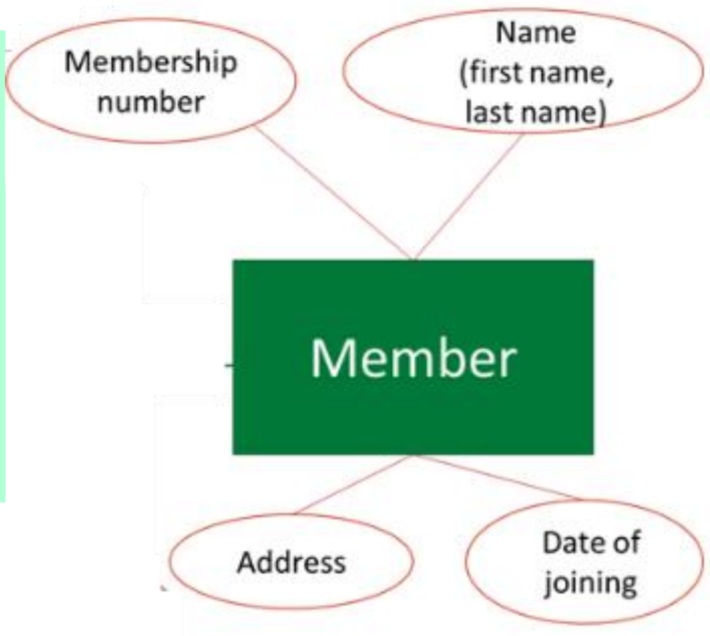
bookid	authors	title	publisher	year
1	Dennis M. Ritchie	C Programming	Prentice Hall	1978
2	Herbert	Java: A Beginner's Guide	McGraw-Hill Education	2018
3	Eric Matthes	Python Crash Course		2015
…				

SQL command to create the member table

Member(member num, last name, first name, address, date of joining)

```
create table member(member_num integer not null unique,
                    last_name  character varying(20),
                    first_name character varying(20),
                    address    character varying(100),
                    join_date  date,

                    primary key (member_num));
```

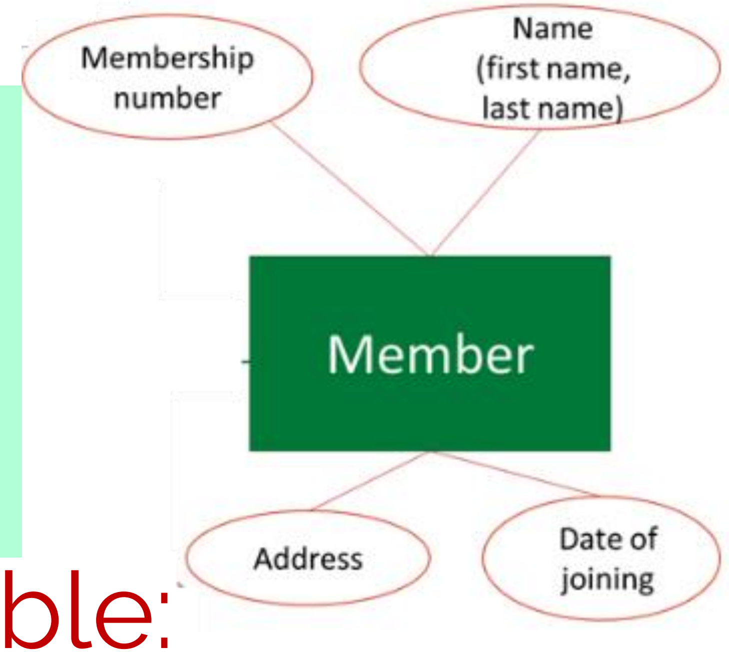


SQL command to create the member table

Member(member num, last name, first name, address, date of joining)

```
create table member(member_num integer not null unique,
                    last_name  character varying(20),
                    first_name character varying(20),
                    address    character varying(100),
                    join_date  date,

                    primary key (member_num));
```



Values that can be stored in this table:

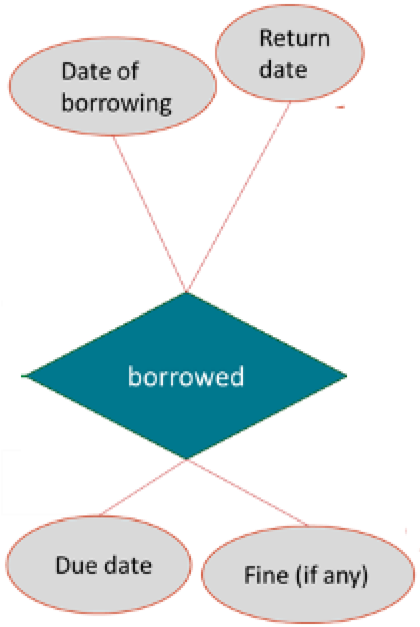
member_num	last_name	first_name	address	join_date
1	Smith	John	123 Street, London	15/01/2023
2	Brown	James	789 Road, Birmingham	10/03/2023
3				

SQL command to create the borrow table

Borrow(book id, member num, date, due date, return date, fine)

```
create table borrow(bookid     integer not null references book(bookid),
                    member_num integer not null references member(member_num),
                    date       date not null,
                    due_date   date not null,
                    return_date date,
                    fine       integer,

                    primary key (bookid, member_num));
```



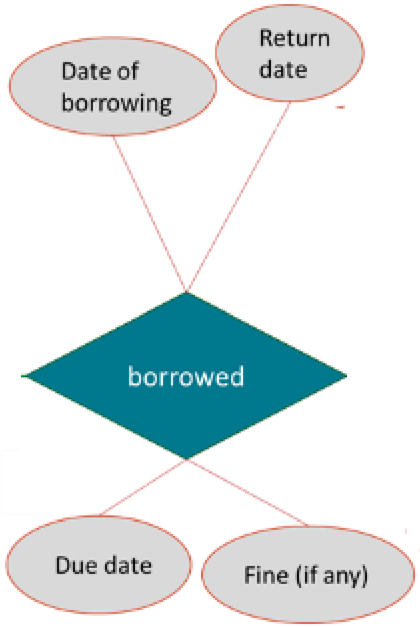
- We declare that the bookid and member_num fields should reference the corresponding fields in the book and member tables. So we can’t issue non-existent books to non-existent members. These kinds of declarations are constraints.
- The primary key declaration says that the bookid and member_num fields *together* uniquely identify a record in the table.

SQL command to create the borrow table

Borrow(book id, member num, date, due date, return date, fine)

```
create table borrow(bookid      integer not null references book(bookid),
                    member_num integer not null references member(member_num),
                    date        date not null,
                    due_date    date not null,
                    return_date date,
                    fine        integer,

                    primary key (bookid, member_num));
```



Values that can be stored in this table:

bookid	member_num	borrow_date	due_date	return_date	fine
1	1	16/01/2023	15/02/2023	NULL	NULL
2	2	05/04/2023	05/05/2023	NULL	NULL

SQL query examples

book

bookid	authors	title	publisher	year
1	Dennis M. Ritchie	C Programming	Prentice Hall	1978
2	Herbert	Java: A Beginner's Guide	McGraw-Hill Education	2018
3	Eric Matthes	Python Crash Course		2015
…				

member

member_num	last_name	first_name	address	join_date
1	Smith	John	123 Street, London	15/01/2023
2	Brown	James	789 Road, Birmingham	10/03/2023
3				

borrow

bookid	member_num	borrow_date	due_date	return_date	fine
1	1	16/01/2023	15/02/2023	NULL	NULL
2	2	05/04/2023	05/05/2023	NULL	NULL

SQL query examples

book

bookid	authors	title	publisher	year
1	Dennis M. Ritchie	C Programming	Prentice Hall	1978
2	Herbert	Java: A Beginner's Guide	McGraw-Hill Education	2018
3	Eric Matthes	Python Crash Course		2015
…				

Finds the id number of ‘Python Crash Course’ by ‘Eric Matthes’?

member

member_num	l			join_date
1			n	15/01/2023
2	Brown	James	789 Road, Birmingham	10/03/2023
3				

borrow

bookid	member_num	borrow_date	due_date	return_date	fine
1	1	16/01/2023	15/02/2023	NULL	NULL
2	2	05/04/2023	05/05/2023	NULL	NULL

SQL query examples

book

bookid	authors	title	publisher	year
1	Dennis M. Ritchie	C Programming	Prentice Hall	1978
2	Herbert	Java: A Beginner's Guide	McGraw-Hill Education	2018
3	Eric Matthes	Python Crash Course		2015
…				

member

member_num			join_date
1		on	15/01/2023
2		ham	10/03/2023
3			

borrow

bookid	me			return_date	fine
1				NULL	NULL
2	2	05/04/2023	05/05/2023	NULL	NULL

Finds the id number of ‘Python Crash Course’ by ‘Eric Matthes’?

```
select bookid
from book
where authors = 'Eric Matthes'
and title = 'Python Crash Course';
```

SQL query examples

book

bookid	authors	title	publisher	year
1	Dennis M. Ritchie	C Programming	Prentice Hall	1978
2	Herbert	Java: A Beginner's Guide	McGraw-Hill Education	2018
3	Eric Matthes	Python Crash Course		2015
…				

Check if the book 'C Programming' by 'Dennis M. Ritchie' is available for checkout.

member

member_num	last_nar			join_date
1	Smith			15/01/2023
2	Brown			10/03/2023
3				

borrow

bookid	member_num	borrow_date	due_date	return_date	fine
1	1	16/01/2023	15/02/2023	NULL	NULL
2	2	05/04/2023	05/05/2023	NULL	NULL

SQL query examples

book

bookid	authors	title	publisher	year
1	Dennis M. Ritchie	C Programming	Prentice Hall	1978
2	Herbert	Java: A Beginner's Guide	McGraw-Hill Education	2018
3	Eric Matthes	Python Crash Course		2015
…				

member

member_num	last_name	fi	ate
1	Smith		2023
2	Brown		2023
3			

borrow

bookid	member_num	borr	fine
1	1	16/	NULL
2	2	05/	NULL

Check if the book 'C Programming' by 'Dennis M. Ritchie' is available for checkout.

```
select 'not available'
from book, borrow
where book.bookid = borrow.bookid
  and book.authors = 'dennis m. ritchie'
  and book.title = 'c programming';
```

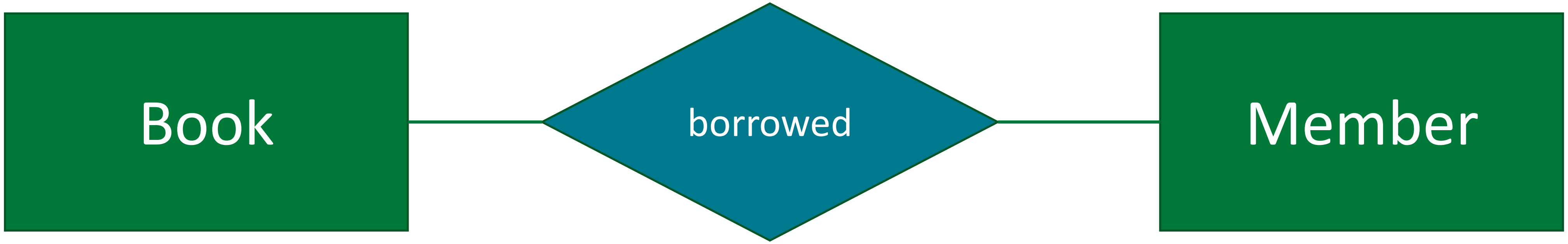
This Week

- Introduction
- Relational databases
- Entity-relationship (ER) diagram /Modeling
- Table design and creation, SQL
- ➡ Weak entities
- Hierarchies
- Table design (schemas) Optimisation
- SQL Commands + PostgreSQL

UNIVERSITY OF BIRMINGHAM | DUBAI دبي

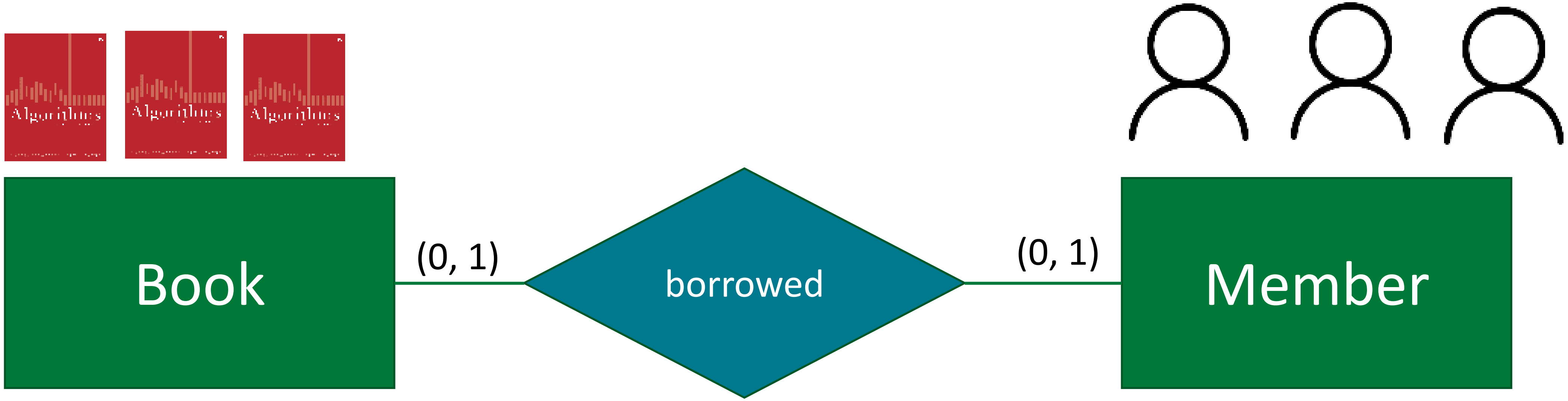
Weak entities

An entity, which depends on some other entity for its identity. Entities that are not weak are called “strong entities” by some tutors. A weak entity’s primary key consists of the primary key of its main entity (or owner entity) and some additional attributes. Example: Copy dependent on Book.



How to represent multiple copies of same book in library in ER Diagram?

Weak entities - library management system



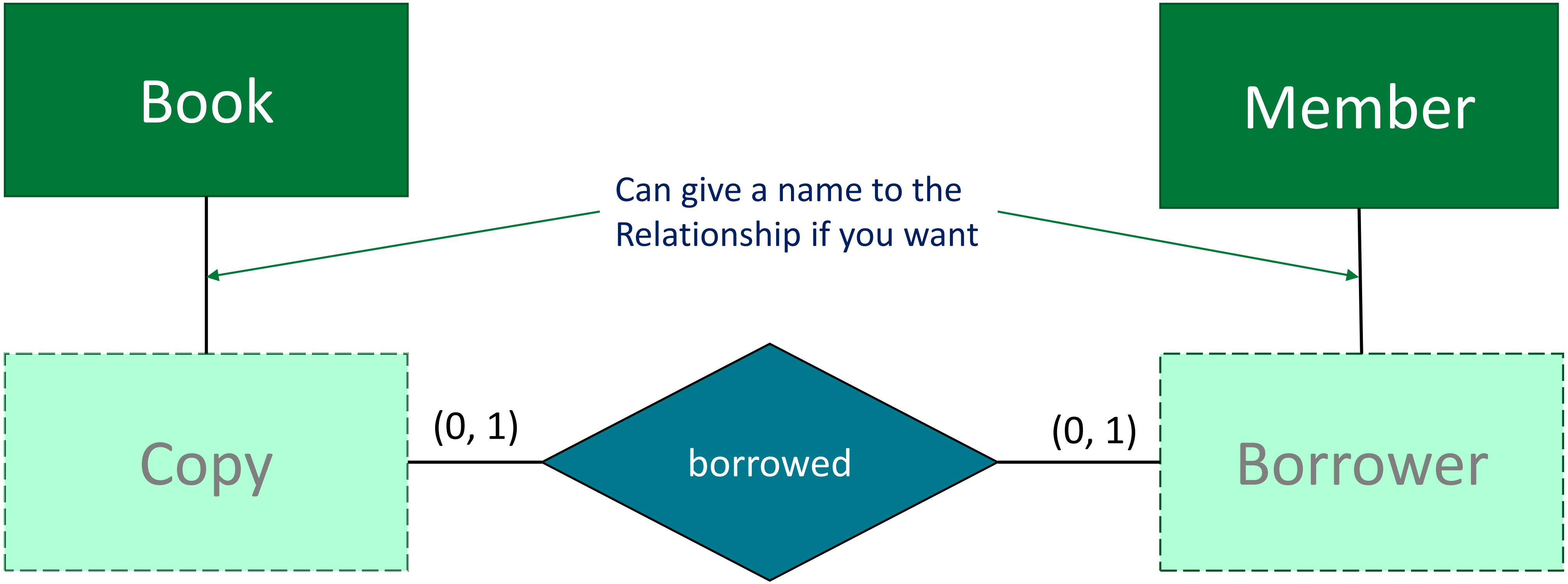
Suppose our library has multiple copies of some of the books.

How do we handle them?

Suppose our library allows family memberships.

In addition to the primary member, a number of relatives can also borrow books.

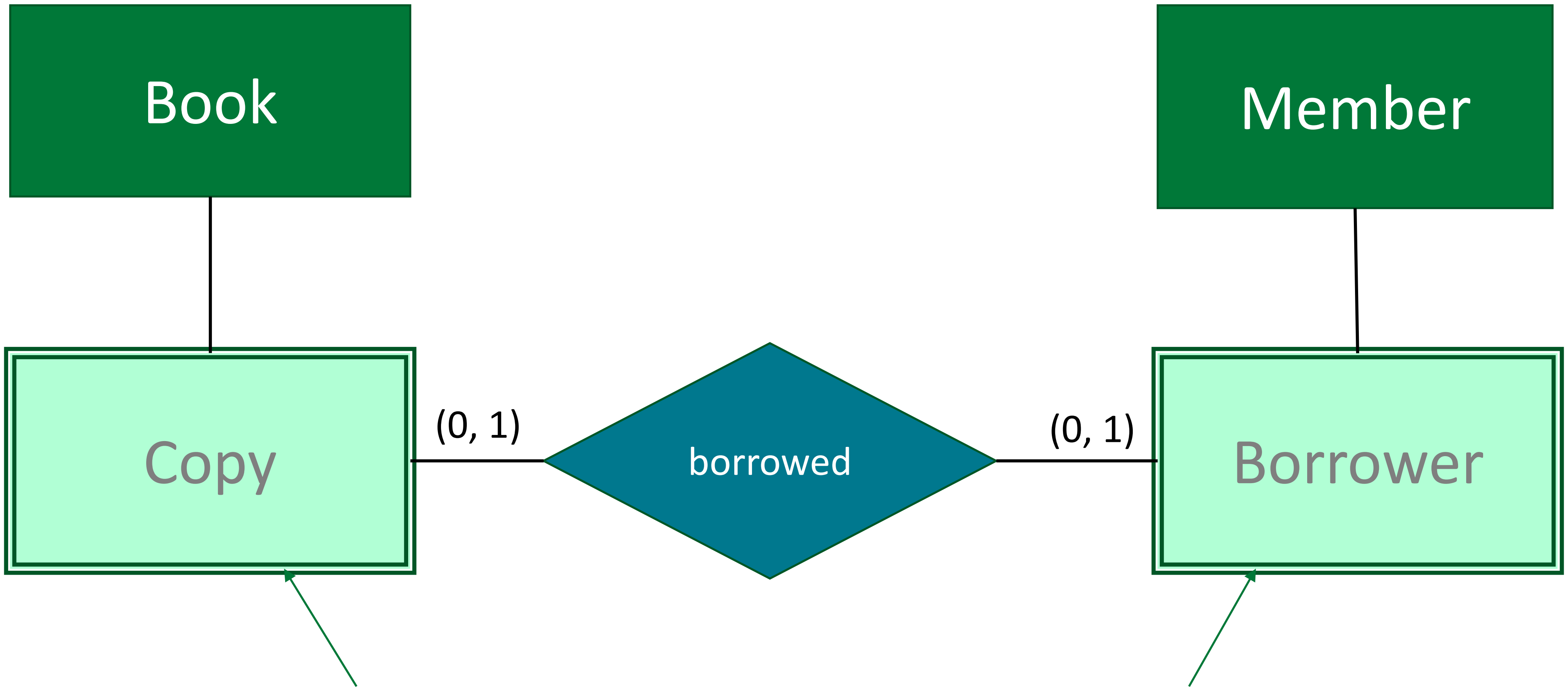
Revised ER-diagram with Weak entities



Can give a name to the Relationship if you want

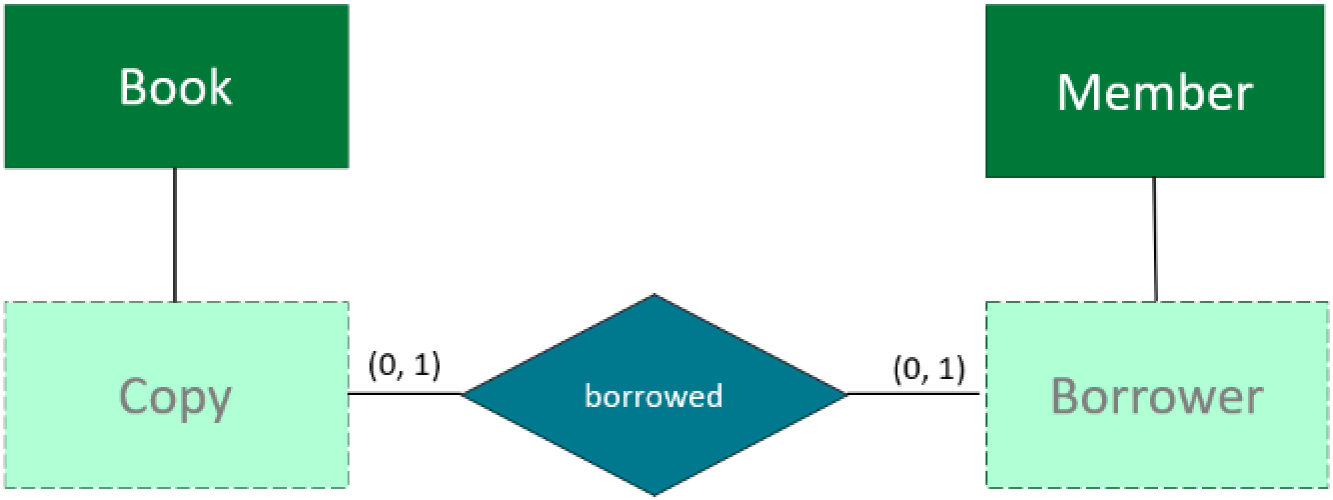
Copies of books are borrowed by borrowers associated with primary members.
Think about what needs to be done if a book has a single copy.
And when the primary member wants to borrow a book on their own.

Another notation for weak entities



Some people use double lines to depict weak entities.

Table for weak entities



The table designs in the schema notation.

Book(book id, authors, title, publisher, year)

Copy(book id, copy num)

Member(member num, last name, first name, address, date of joining)

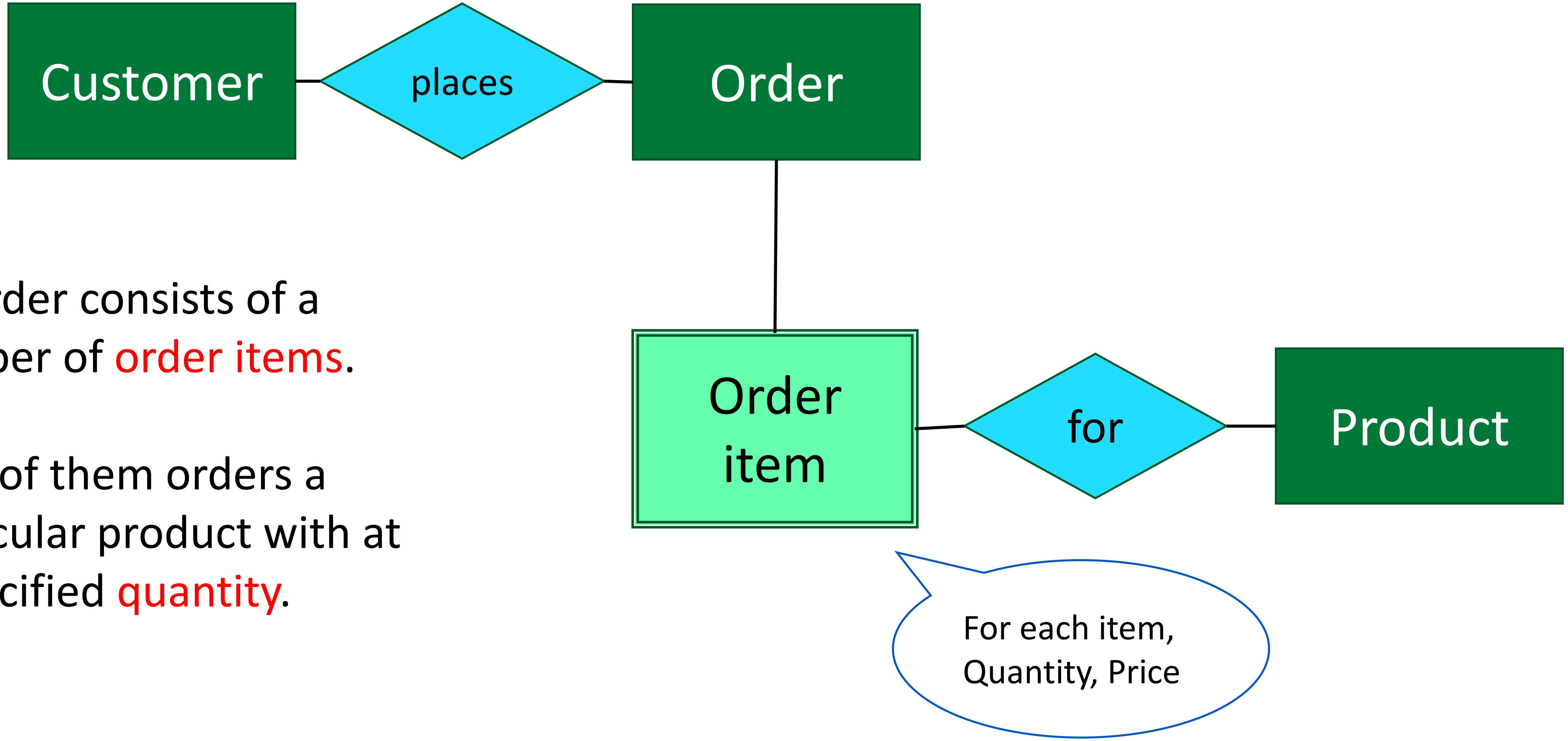
Borrower(member num, first name)

Assuming last name is common

Borrow(book id, copy num, member num, first name, date, due date, return date, fine)

Composite primary key

Another common example for weak entities



For each item, Quantity, Price

An order consists of a number of order items.

Each of them orders a particular product with at a specified quantity.

This Week

- Introduction
- Relational databases
- Entity-relationship (ER) diagram /Modeling
- Table design and creation, SQL
- Weak entities

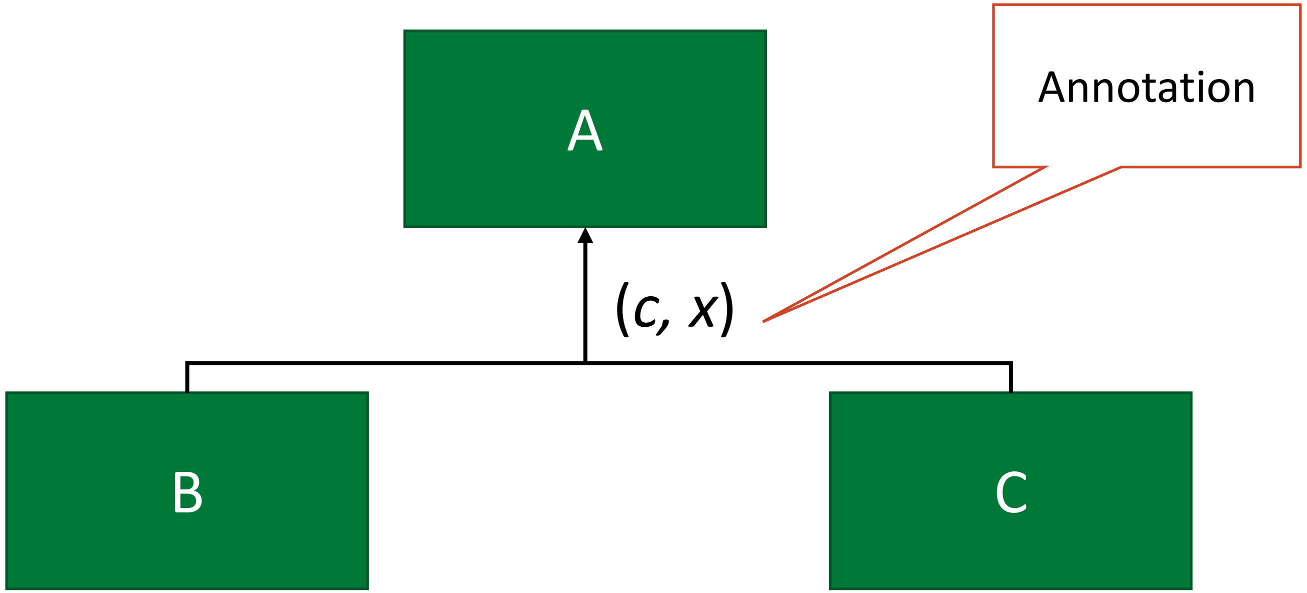
➡ Hierarchies

- Table design (schemas) Optimisation
- SQL Commands + PostgreSQL

UNIVERSITY OF BIRMINGHAM | DUBAI دبي

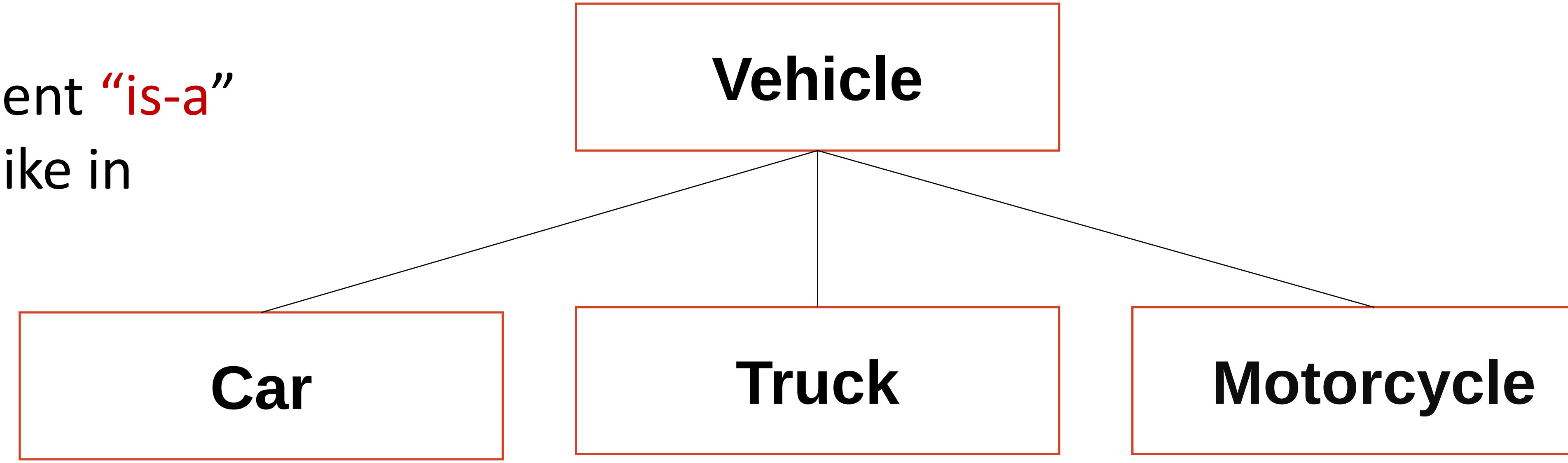
Hierarchies (generalisation hierarchies)

Hierarchies represent "is-a" relationships just like in Object Oriented Programming.



Hierarchies

Hierarchies represent “is-a” relationships just like in Object Oriented Programming.



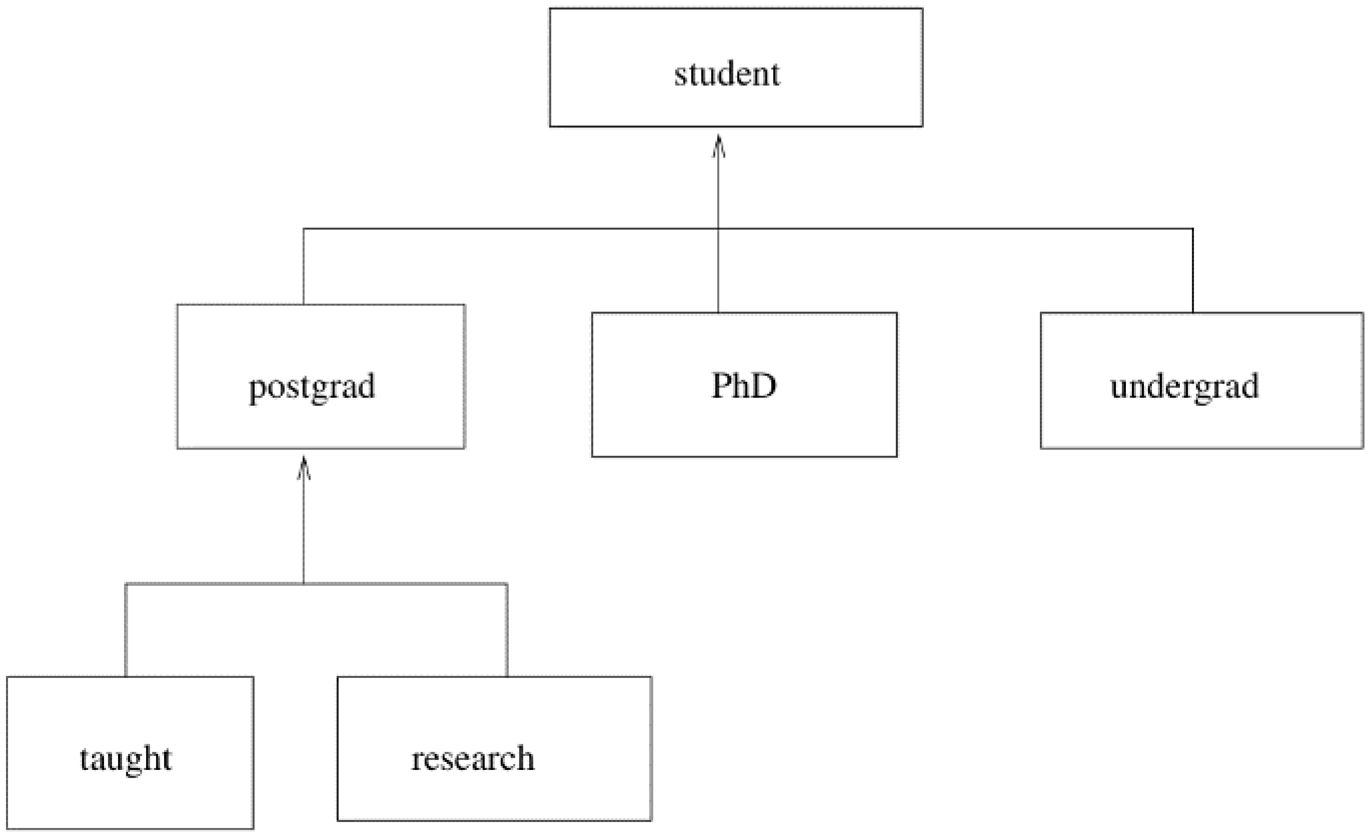
Vehicle hierarchy

Example-1 Cars, trucks and motor cycles are special cases of a more general vehicle entity.

Hierarchies

Hierarchies represent “is-a” relationships just like in Object Oriented Programming.

Example-2 undergrad and postgrad student are special cases of a more general student entity (“super-entity”).

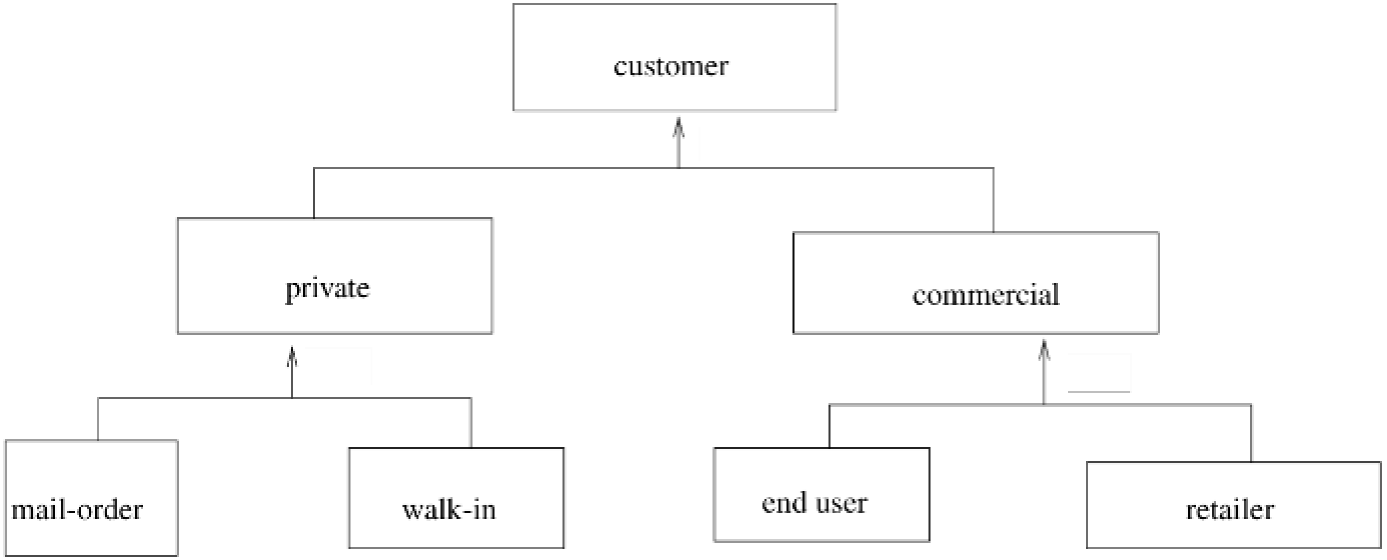


Student hierarchy

Hierarchies

Hierarchies represent “is-a” relationships just like in Object Oriented Programming.

Example-3 Customer hierarchy



Customer hierarchy

Annotations for hierarchy relationships

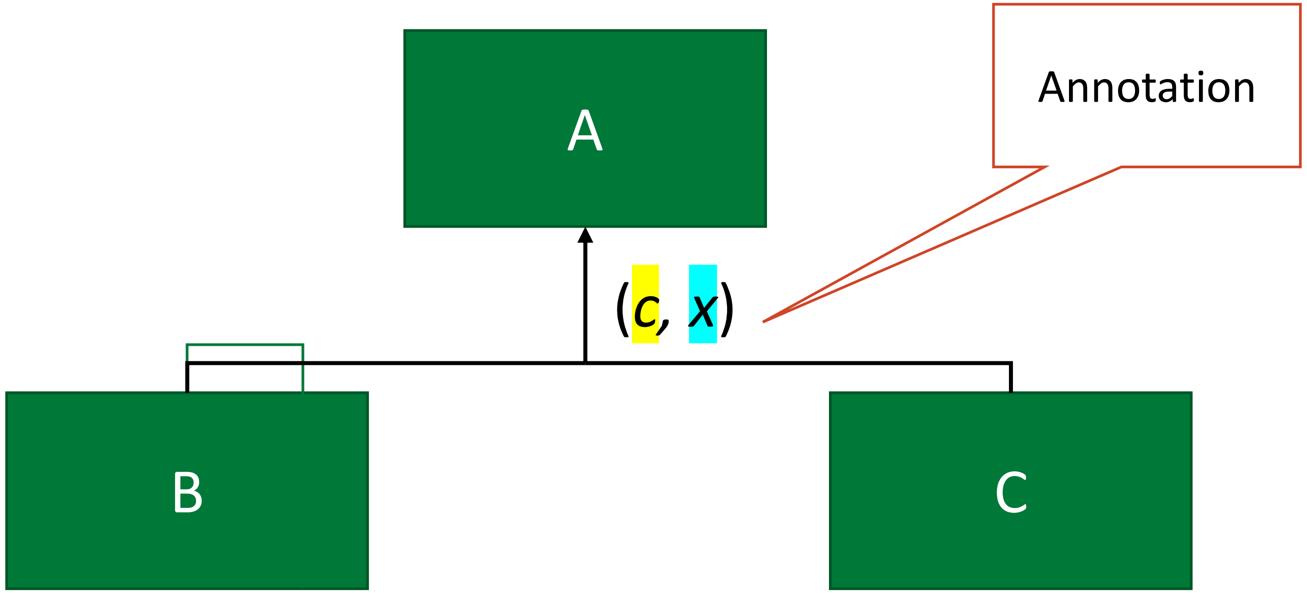
Two features need to be specified.

Coverage c:

Do the subclasses totally (t) cover all of the superclass entity, or partially (p)?

Overlap x:

Are the subclasses overlapping (o), or are they mutually exclusive (e)?

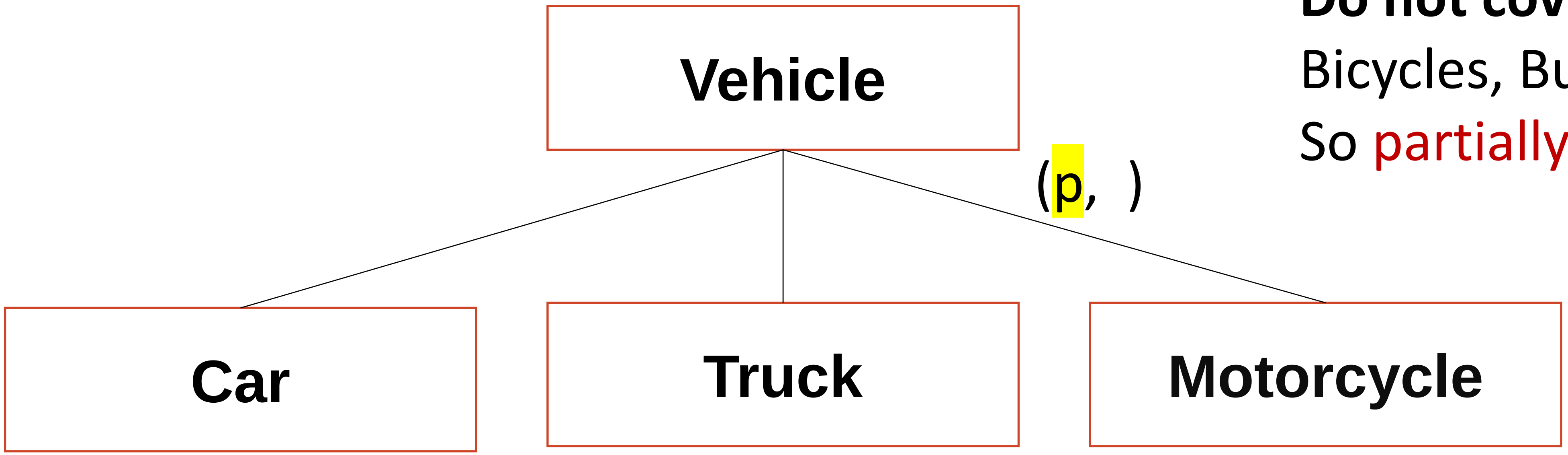


Annotations for hierarchy relationships

Two features need to be specified.

Coverage c:

Do the subclasses totally (t) cover all of the superclass entity, or partially (p)?



Do not cover

Bicycles, Buses, Trains, Boats

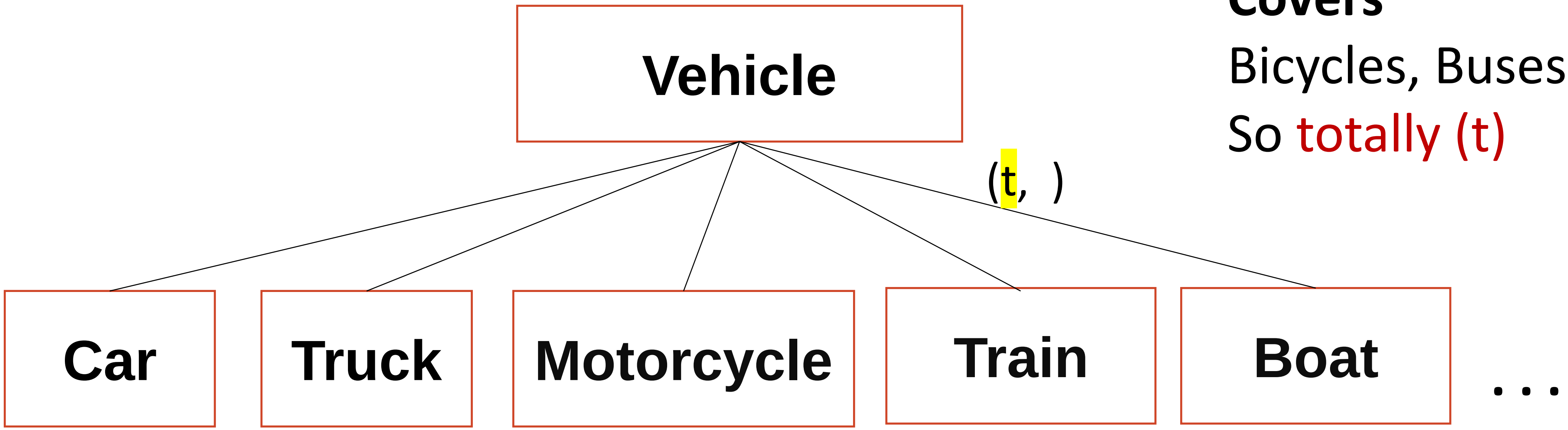
So partially (p)

Annotations for hierarchy relationships

Two features need to be specified.

Coverage c:

Do the subclasses totally (t) cover all of the superclass entity, or partially (p)?



Covers

Bicycles, Buses, Trains, Boats

So totally (t)

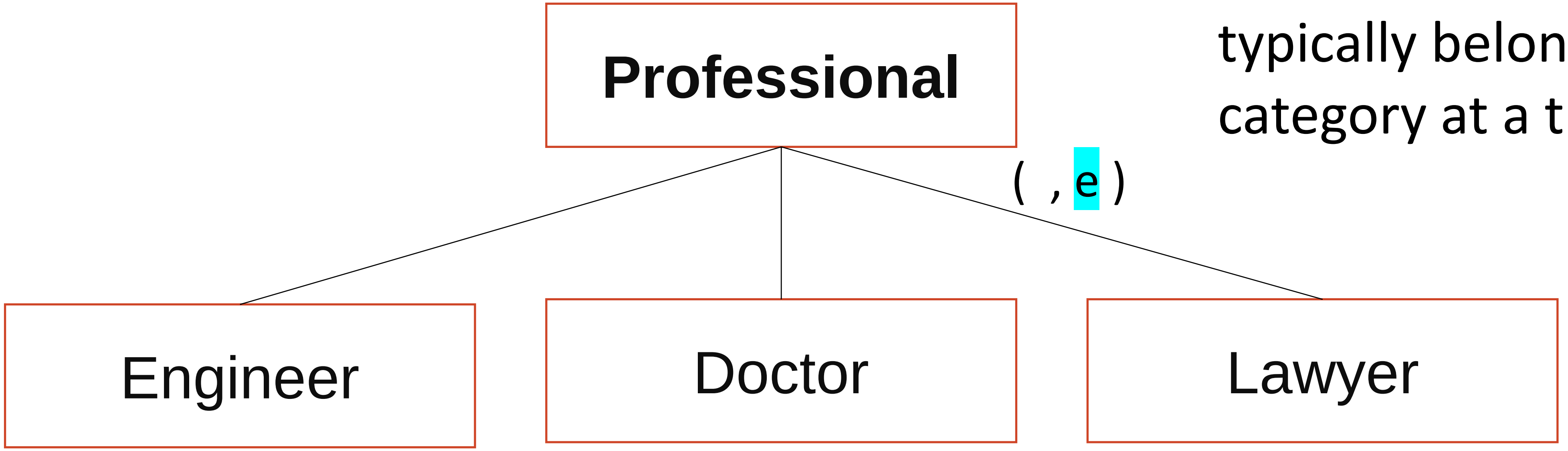
Annotations for hierarchy relationships

Two features need to be specified.

Overlap x:

Are the subclasses overlapping (o), or are they mutually exclusive (e)?

mutually exclusive (e) because an individual typically belongs to only one category at a time:

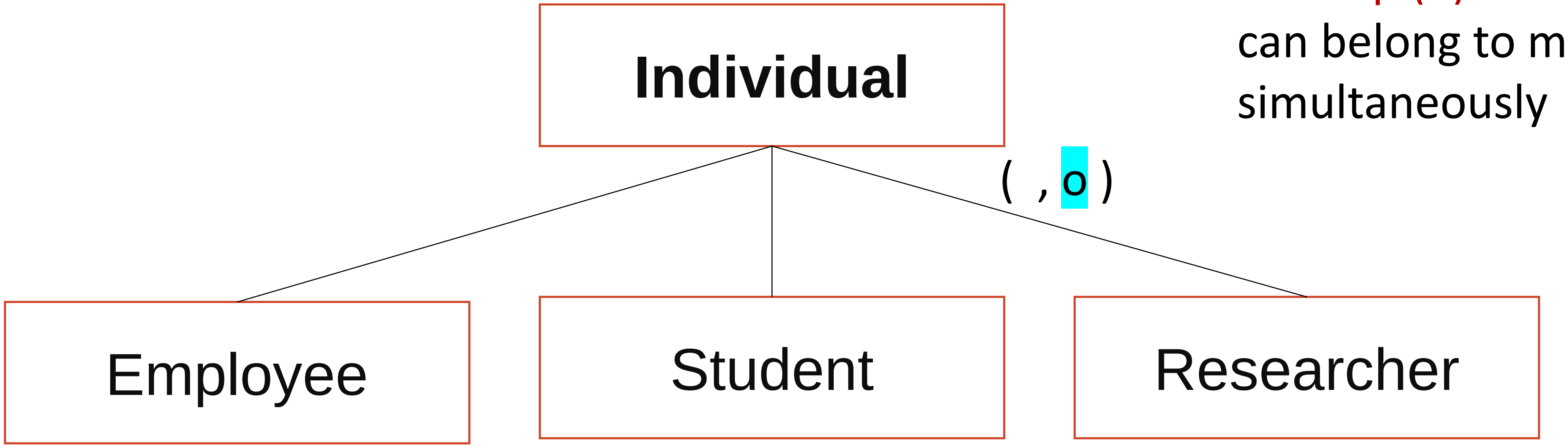


Annotations for hierarchy relationships

Two features need to be specified.

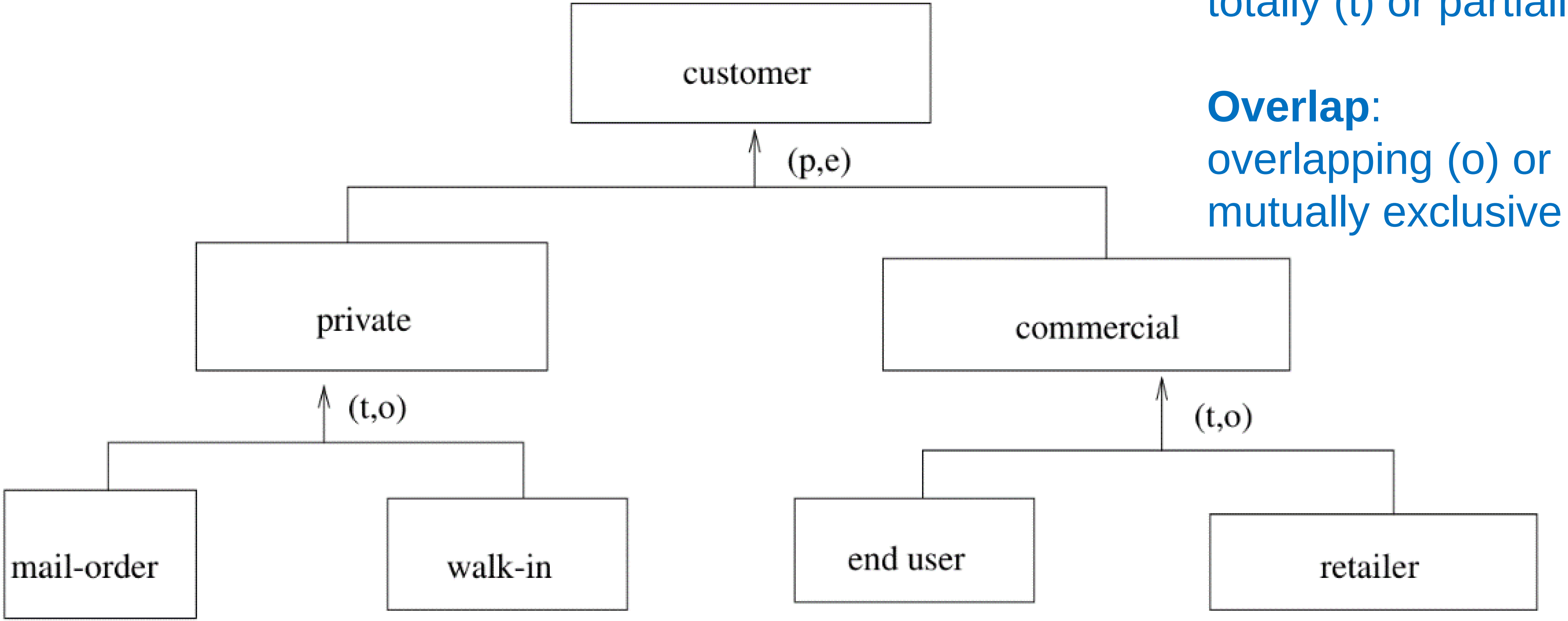
Overlap x:

Are the subclasses overlapping (o), or are they mutually exclusive (e)?



Overlap (o) because individuals can belong to multiple categories simultaneously

Example of Coverage and Overlap



Customer hierarchy

Coverage:
totally (t) or partially (p)

Overlap:
overlapping (o) or
mutually exclusive (e)

This Week

- Introduction
- Relational databases
- Entity-relationship (ER) diagram /Modeling
- Table design and creation, SQL
- Weak entities
- Hierarchies

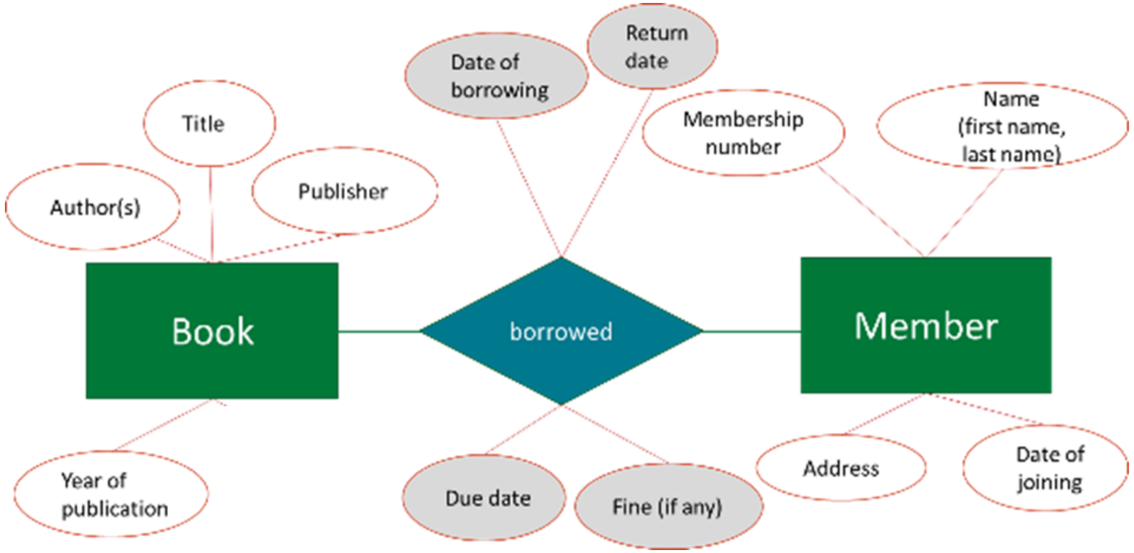
Table design (schemas) Optimisation

- SQL Commands + PostgreSQL

Table design (schemas)

Do we really need 3 tables for this problem.

Perhaps 2 tables would be enough?



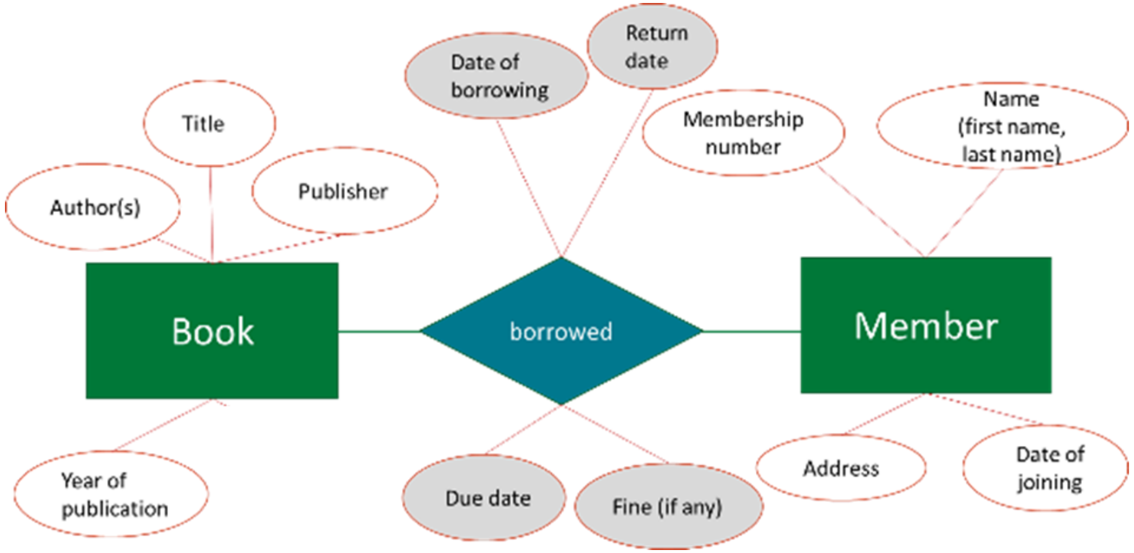
The table designs are written in the schema notation.

Book(<u>book id</u>, authors, title, publisher, year)

Member(<u>member num</u>, last name, first name, address, date of joining)

Borrow(<u>book id, member num</u>, date, due date, return date, fine)

Table design (schemas)



Optimisation 1: We can add the Borrow fields to the Member table.

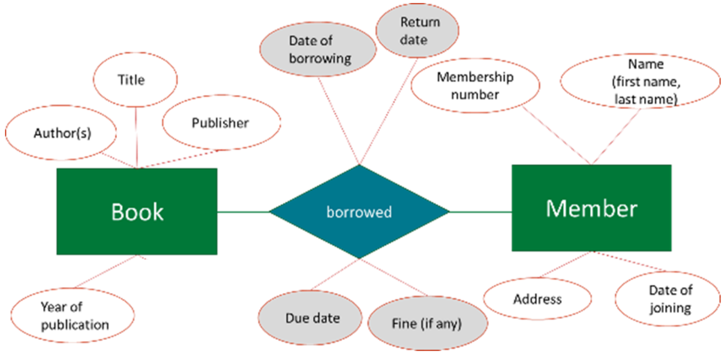
Optimisation 2: If we ignore fines, we can also add the Borrow fields to the Book table.

Book(book id, authors, title, publisher, year)

Member(member num, last name, first name, address, date of joining)

Borrow(book id, member num, date, due date, return date, fine)

Optimisation 1



Original schema for 3 tables (2 entities + 1 relationship).

Book (book id, authors, title, publisher, year)

Member (member num, last name, first name, address, date of joining)

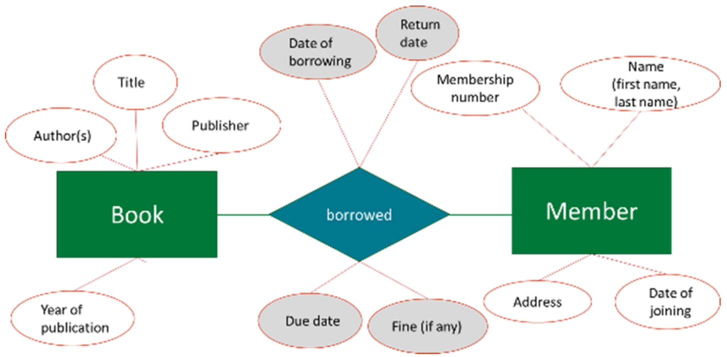
Borrow (book id, member num, borrow date, due date, ~~return date, fine~~)

Optimisation 1: We can add the Borrow fields to the Member table.

Book (book id, authors, title, publisher, year)

Member (member num, last name, first name, address, date of joining,
book id, borrow date, due date, ~~return date, fine~~)

Optimisation 2



Original schema for 3 tables (2 entities + 1 relationship).

Book (book id, authors, title, publisher, year)

Member (member num, last name, first name, address, date of joining)

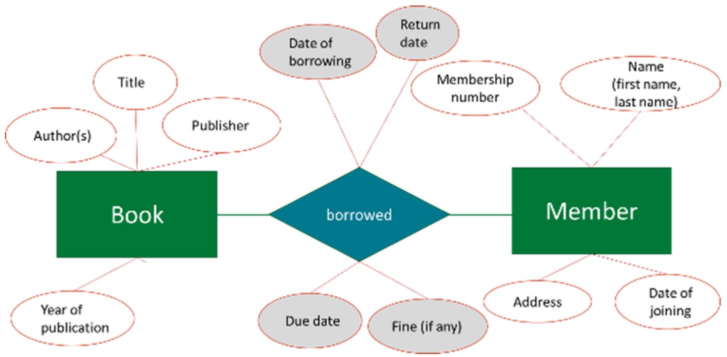
Borrow (book id, member num, borrow date, due date, ~~return date, fine~~)

Optimisation 2: We can add the Borrow fields to the Book table.

Book (book id, authors, title, publisher, year,
member num, borrow date, due date)

Member (member num, last name, first name, address, date of joining)

Issues with optimisation - 1



Adding Borrow fields to the Member table.

Book (book id, authors, title, publisher, year)

Member (member num, last name, first name, address, date of joining, book id^{o}, borrow date^{o}, due date^{o}, ~~return date^{o}, fine~~^{o})

Issues

What happens when the member is not borrowing any book?

The book id field and other related fields would be null.

Represented by the ^{o} notations in our schemas.

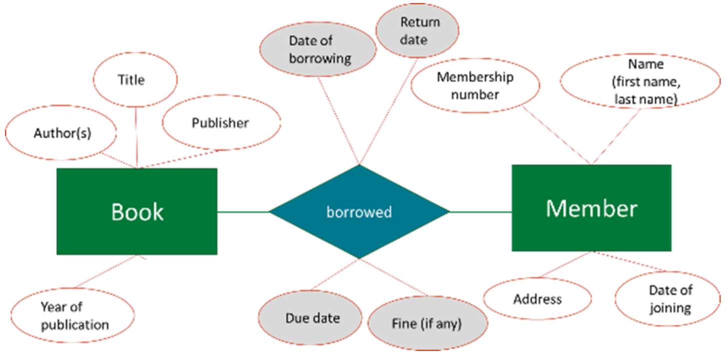
What happens if the library decides to allow multiple books to be borrowed?

This design wouldn't work any more.

We can only merge relations into tables if the multiplicity is maximum 1.

It is not a good idea to hard-wire business policies into the database design! They can change in future.

Issues with optimisation - 2



Adding Borrow fields to the Book table.

Book (book id, authors, title, publisher, year,
member num^{o}, borrow date^{o}, due date^{o})

Member (member num, last name, first name, address, date of joining)

Issues

What happens if the book is not borrowed at present?

The book id field and other related fields would be null.

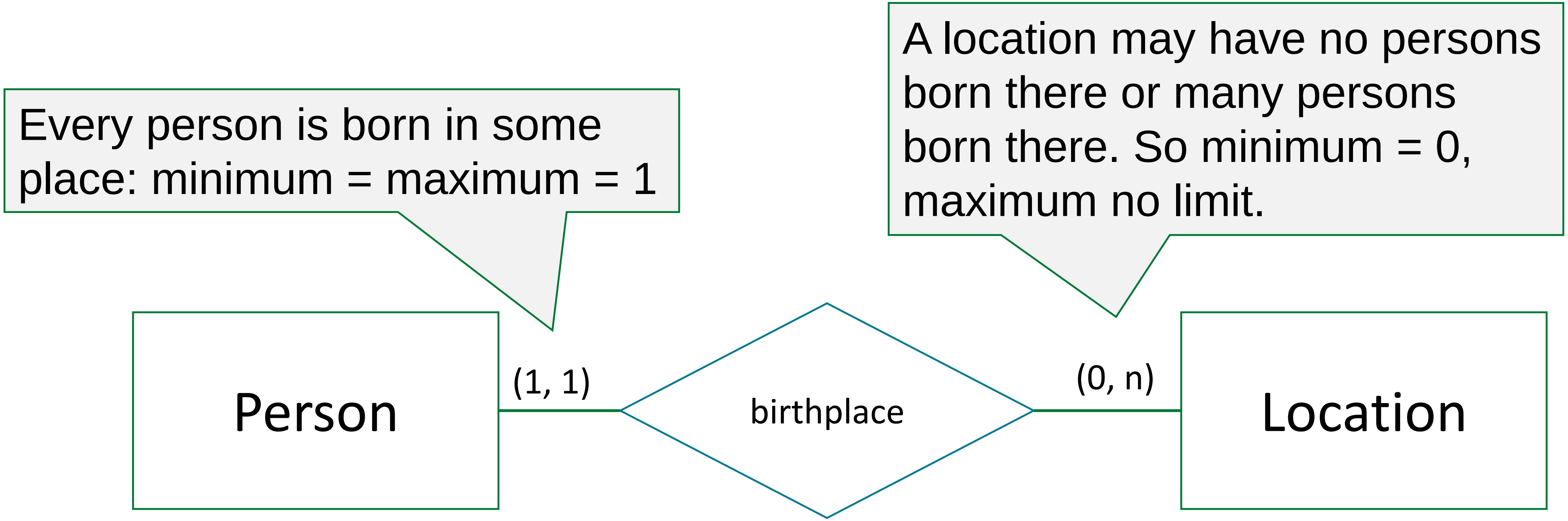
But we cannot represent the fines satisfactorily in this design. It is not possible to represent a book having one member with an outstanding fine, and another member that is currently borrowing.

Perhaps the fine attribute can be moved to the Member table? Yes, but we wouldn’t know which book the fine was charged for.

Perhaps we can add the Borrow fields to both the Book table and the Member table? It would introduce redundancy in the design, which can be problematic.

All said and done, perhaps the 3-table design is the best design for this problem!

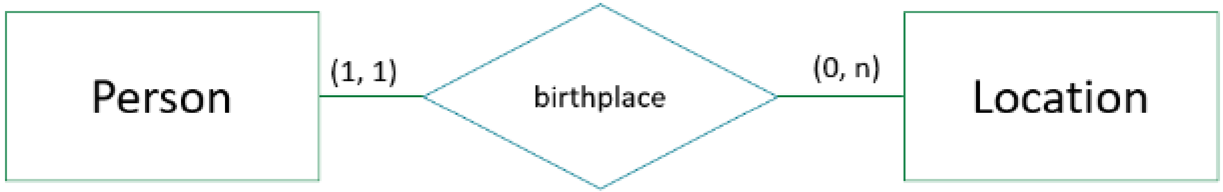
Additional Example: Database about Birthplaces



Every person is born in some place: minimum = maximum = 1

A location may have no persons born there or many persons born there. So minimum = 0, maximum no limit.

Table design



Straight design: Three tables (2 entities + 1 relationship)

Person(pid, last name, first name)
Location(locid, name)
Birthplace(pid, locid)

Optimised design:

Since the multiplicity of Birthplace at the Person end is (1,1), the Birthplace table can be merged into the Person table.

Person(pid, lastname, first name, birthplace)
Here, “birthplace” is the locid of the Location

Note: The Birthplace table cannot be merged into the Location table (because the maximum multiplicity is not 1).

Keypoints of representing relationships

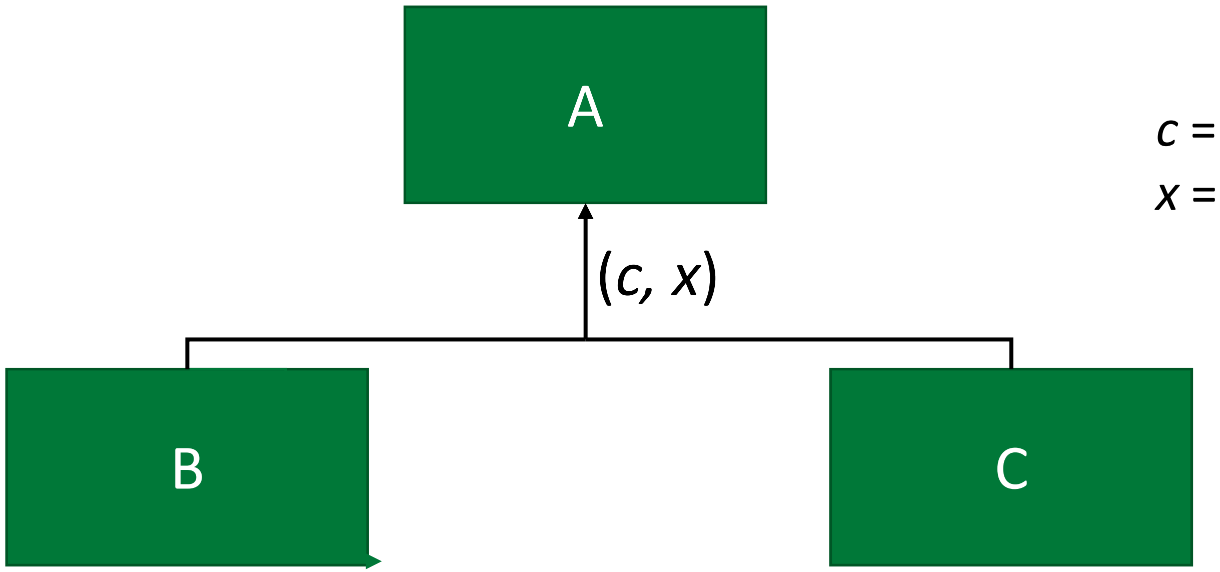
In a straight design, each entity and each relationship is made into a separate table.

If a relationship has (0, 1) or (1, 1) multiplicity with an entity, its table can be merged with that of the entity.

(But it is *not necessary* to do so.)

If the multiplicity is (0, 1) then the relationship attributes should allow for the NULL possibility.

Hierarchy

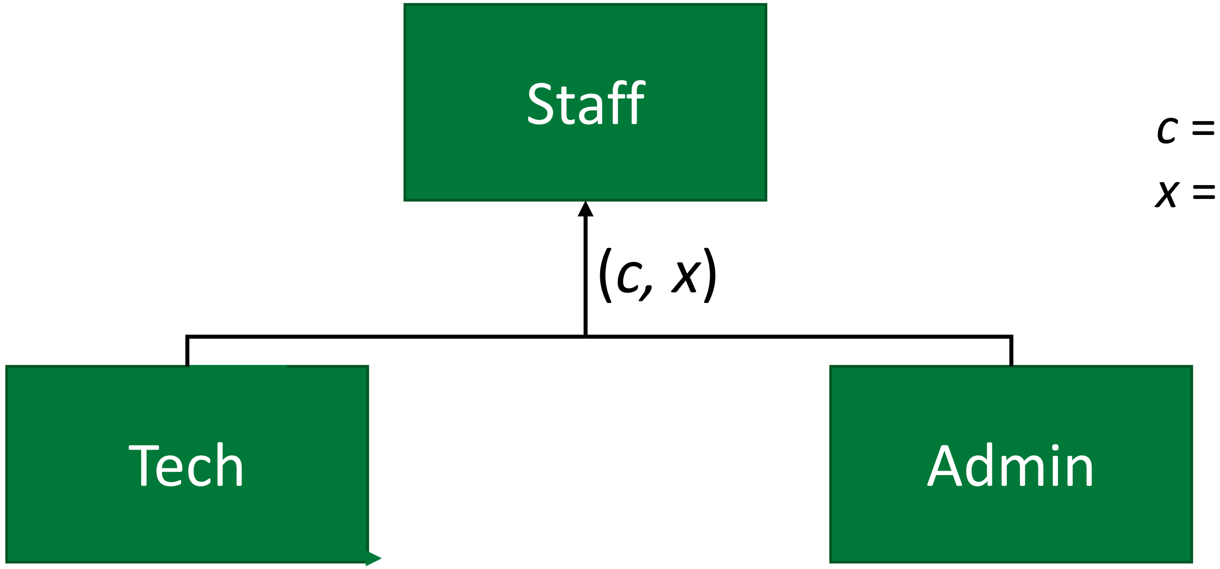


c = t(otal), p(atrial)
x = o(verlapping), e(xclusive)

Three possible choices for table design:

1. Keep all three
2. Keep B and C, and omit A
3. Keep A, and omit B and C

Hierarchy example

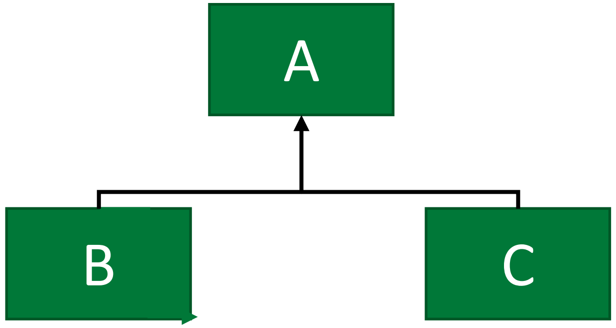


c = t(otal), p(atrial)
x = o(verlapping), e(xclusive)

Three possible choices for table design:

- Keep all three
- Keep Tech and Admin, and omit Staff
- Keep Staff, and omit Tech and Admin

1. Keeping all three (superclass + subclasses)



This is the simplest method and always applicable.

Remember that an entity of type B or C (subclasses) is **also** of type A.

So, the primary identity of the entity is established in type A, and inherited by B and C.

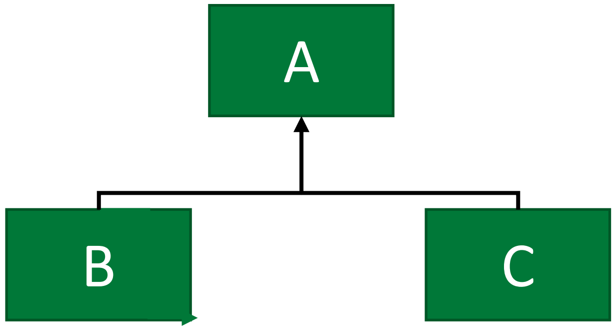
- For example, staffid is defined in the Staff table, and then inherited in the Tech and Admin tables.
- The superclass table contains all the *common* attributes (name, address, salary, date of joining etc.)
- The subclass tables contain the *additional* attributes, e.g., speciality for tech staff, and role for admin staff.

Sample schemas

Staff(staffid, lastname, firstname, joindate, grade, salary)

Techstaff(staffid, speciality, department)

Adminstaff(staffid, role, unit)



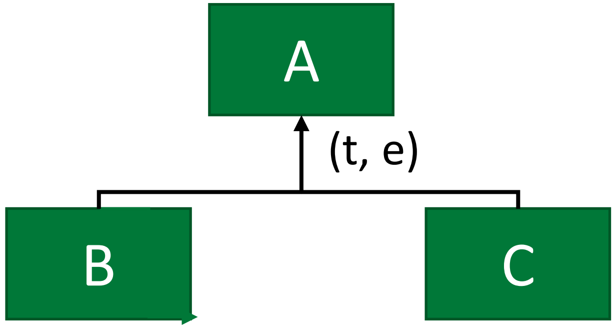
2. Keep only the subclasses

This method is only possible if the coverage is total, and overlap is exclusive.

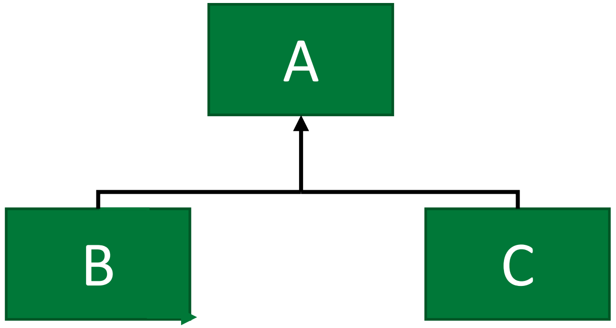
The primary identity is established with respect to the superclass, even though the superclass table is not stored (a bit confusing!).

For example, staffid should be unique across both the Tech and Admin tables, i.e., unique within an imaginary Staff table.

Whatever attributes might have been stored in the superclass are duplicated in all the subclasses (e.g., staffid, name, salary etc.)



3. Keep only the superclass



The subclass tables are omitted.

So all their attributes need to be stored in the superclass.

In addition, we need to add a variant attribute, which specifies the subclass type of an entity. (e.g., staff_type = tech or admin)

If a particular entity belongs to one variant, the attributes corresponding to the other variant would be NULL.

> For example, the records for technical staff would have NULL for role. The records for administrative staff would have NULL for speciality.

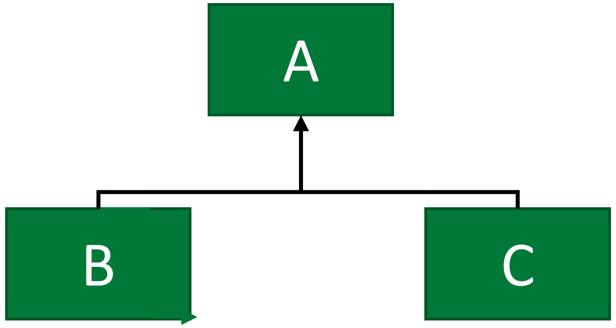
Key points of representing hierarchies

1. The simplest method is to make tables for the superclass as well as subclasses.

2. Making tables for only subclasses is possible if the annotation for the hierarchy is (t, e). Identity has to be defined with respect to an imaginary superclass table.

3. Keeping only the superclass and merging the subclass attributes is always possible, but

- Involves adding a variant attribute for the subclass type.
- Perhaps wastes space for the NULL attribute values.



This Week

- Introduction
- Relational databases
- Entity-relationship (ER) diagram /Modeling
- Table design and creation, SQL
- Weak entities
- Hierarchies
- Table design (schemas) Optimisation
- SQL Commands + PostgreSQL

UNIVERSITY OF BIRMINGHAM | DUBAI دبي

SQL command example

```
create table borrow(bookid     integer not null references book(bookid),
                    member_num integer not null references member(member_num),
                    date       date not null,
                    due_date   date not null,
                    return_date date,
                    fine       integer,

                    primary key (bookid, member_num));
```

- The general format is: *field-name, type, field-constraints*.
- We use “not null” for all fields that are not designated as possibly null in the schema.
- We use the “references” constraint for all references into other tables.
- This happens in:
  - Relationships (for referencing the entities)
  - Weak entities (for referencing the owner entity)
  - Hierarchies (for referencing superclass entities)

Types in SQL

INT (or INTEGER)

DECIMAL(n, m) – decimal numbers of length n, with m digits after the point

BOOLEAN

CHAR – single character

CHAR(n) – fixed length character string

VARCHAR(n) – variable length character string with a limit

TEXT – arbitrary length string

DATE and TIME

Generating id numbers

See Paragraph 34.

In standard SQL, we can declare

```
    create sequence staffid_seq;
```

This creates a sequence generator in the database, so that evaluating `nextval(staff_seq)` generates a new integer value.

We can use this as a “default value” in the create table statements:

```
create table Tech(staffid integer DEFAULT nextval(staffid_seq);
                  ... )

create table Admin(staffid integer DEFAULT nextval(staffid_seq);
                  ... )
```

Foreign key issue: ON DELETE

Fields with “references” constraints are called foreign keys.

They arise in relationships, weak entities and hierarchies.

The DBMS ensures that, whenever we insert a record with a foreign key value, that value is *actually present* in the referenced table.

What happens when the referenced value is deleted from the foreign table? Then the foreign key constraint would be violated!

The DBMS blocks such deletion and gives an error (by default).

This is called “ON DELETE NO ACTION” constraint.

- “NO ACTION” may be a bit confusing. It really means that the deletion would get blocked!

ON DELETE actions

ON DELETE NO ACTION – default, no need to declare

ON DELETE SET NULL

This sets the foreign key to null, if possible.

If null is not allowed, it would give rise to an error.

ON DELETE CASCADE

The record with the foreign key gets deleted automatically.

Think of what the effect of these declarations would be when a library member quits and we try to delete their record.

What should happen to a borrow record that might still be referencing that member?

Constraints in relational databases

In general, constraints are good! They allow us to detect errors early and make sure that the data in the system is consistent and valid.

Field constraints – constraint on the value of a field (e.g., NOT NULL)

Record constraints – constraints on an entire record (e.g., you might check that the due_date is greater than the borrow_date)

Table constraints – constraints on the entire table (e.g., PRIMARY KEY or UNIQUE).

Database constraints – constraints that span multiple tables (e.g., REFERENCES)

Postgres SQL

Next Week Support Session

https://www.postgresql.org/download/

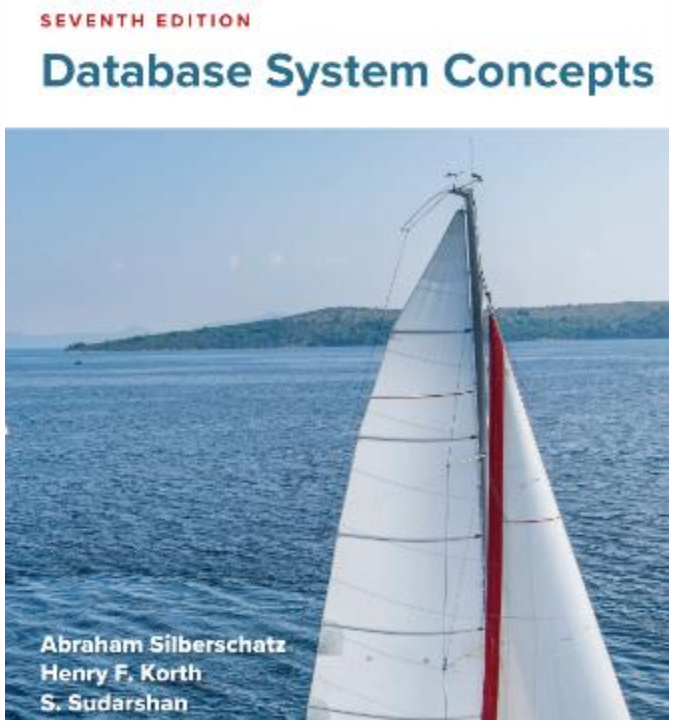
Summary

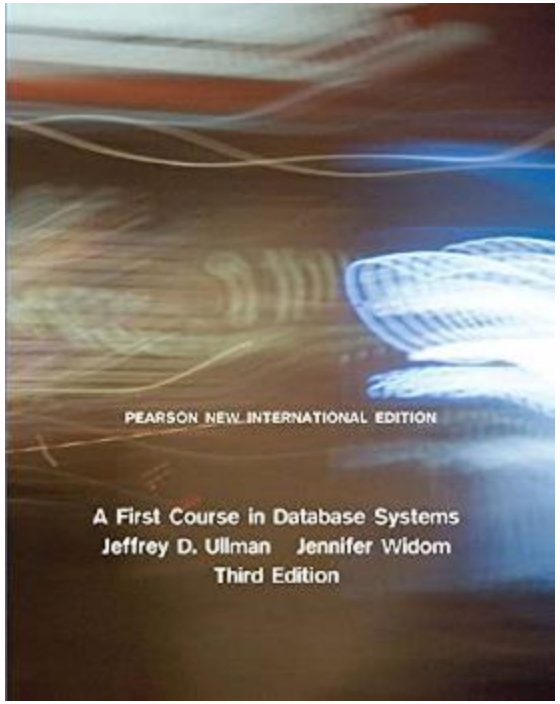
- Introduction
- Relational databases
- Entity-relationship (ER) diagram /Modeling
- Table design and creation, SQL
- Weak entities
- Hierarchies
- Table design (schemas) Optimisation
- SQL Commands + PostgreSQL

Some questions you might think about

- Our table has a single field for all the authors of a book. Can we list each author individually, somehow?
- If we have multiple copies of books in the library, what can we do?
- If we want to allow a member to borrow to multiple books (say 4 books max), what changes do we need in the design?
- If membership is actually “family membership” so that anybody in the family can borrow, do we need to change the design in anyway?
- The “primary key” of the borrow table consists of two fields. Do we need two? Can we make do with one field only?

Weekly Reading




PEARSON NEW INTERNATIONAL EDITION
A First Course in Database Systems
Jeffrey D. Ullman Jennifer Widom
Third Edition

Reading material on Canvas

- Handout 1, 2, 3

Textbook

Chapter 4 – “High-level database models”, Ullman and Widom, A First Course in Database Systems, Third edition (eBook link)

Chapter 6 Database Design Using the E-R Model by Silberschatz, Database System Concepts, 7th Ed (eBook link)

Exercise Questions

ER Modeling (Handout 2, Page 3)

Exercise 16.….

Generalisation hierarchy (Handout 2, Page 7)

Exercise 24 …

Mid Semester Feedback to the Module Team (Dubai)

Kahoot!

Next Week

Week	Date	Topic
1	15 Jan	Searching algorithms
2	22 Jan	Binary Search Tree
3	29 Jan	Balancing Trees – AVL Tree
4	5 Feb	Databases – Conceptual Design
5	14 Feb	Databases – Logical Design & Relational Algebra
6	19 Feb	Consolidation Week
7	26 Feb	Graph Algorithms
8	4 Mar	Sorting Algorithms
9	11 Mar	Hash tables
10	18 Mar	Databases – Normalization
		Easter break and Eid break
11	22 Apr	Databases – Concurrency
12	29 Apr	Revision Week

Thank you.

Questions?

Attendance