IN1013 Databases
Entity-Relationship Model

Vitaly Aksenov
Vladimir Stankovich

Data Modelling Concepts

Introduction: Conceptual Modelling

- Two most common data models:
  - **Entity-Relationship (ER) Model** (Object based)
  - **Relational Data Model** (Record based)
  - also: Object Oriented, Object-Relational, XML data model etc.
- Top-down approach to DB design uses ER:
  - ***entities***, ***relationships***,
  - *attributes & constraints*
- Original: **ER** modelling
- Subsequently: Enhanced ER modelling (**EER**)

Main Terminology

- **Entity**
  - “thing”; artefact; object (*in the Java / C++ sense*);
    - noun
- **Attribute**
  - characteristic; property of an entity; “variable”;
    - adjective
- **Relationship**
  - association among entities;
    - verb

Relationship vs. RDB / Entity vs. RDB

- A **relationship** (in ER modelling) describes the connection **between two or more entities** (often a verb)
- A **relation** (in a RDB) is a table, or a set of tuples (rows)
- Both entities and relationships in the ER can map to relations in the Relational DB
- Not all relationships in the ER map to relations in the Relational DB
- A Relation in the RDB does not always map to an Entity in the Entity-Relationship Model

ADVICE: Don’t think about tables when designing ER/EER model

Entity

- An **Entity** is a ‘thing’ or ‘object’ in the real world that is distinguishable from all other objects
- Real (concrete) entity types:
  - e.g. Person, Book, Property
- Abstract entity types:
  - e.g. Viewing, Loan, Booking, Renting

Person

Book

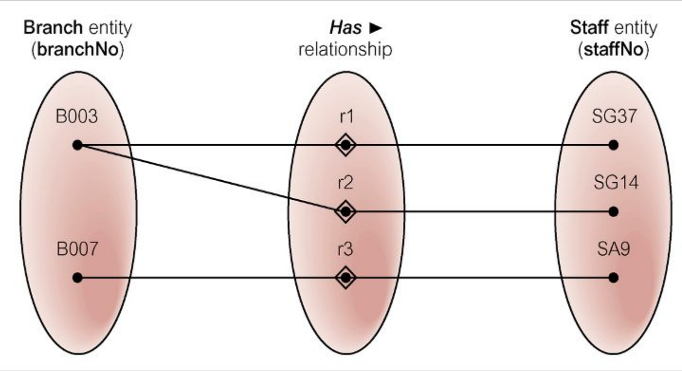
Viewing

Relationship

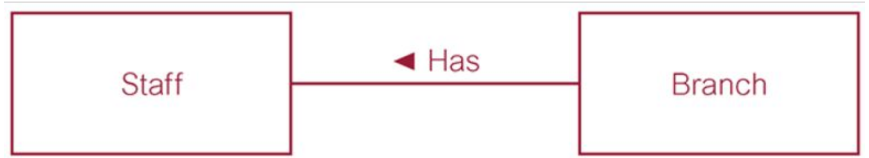
- A **Relationship** is an association among one or more entities
- Relationship has a **name** that describes its function (usually a verb)
- *Example:*

  a relationship called ***Has*** is a relation between Branch and Staff

Semantic net – **Individual Entities**



ER diagrams – **Entity Types**

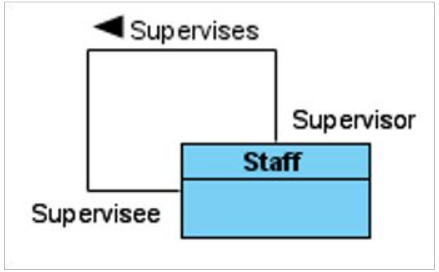


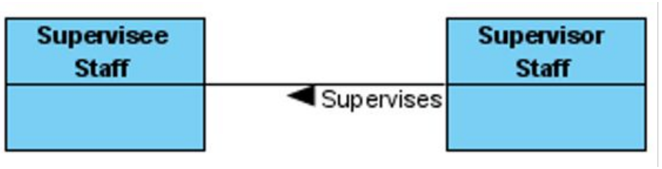
‘Branch has staff’

Recursive Relationship

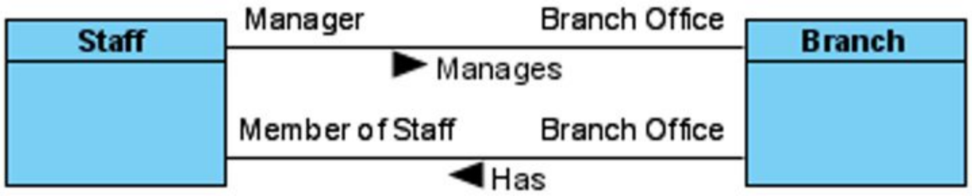
A recursive relationship has the ***same entity*** participating *more than once **in different roles***

We may use **role names** to indicate the role that entities play in a relationship





Role names may be associated through more than one relationship



Attributes

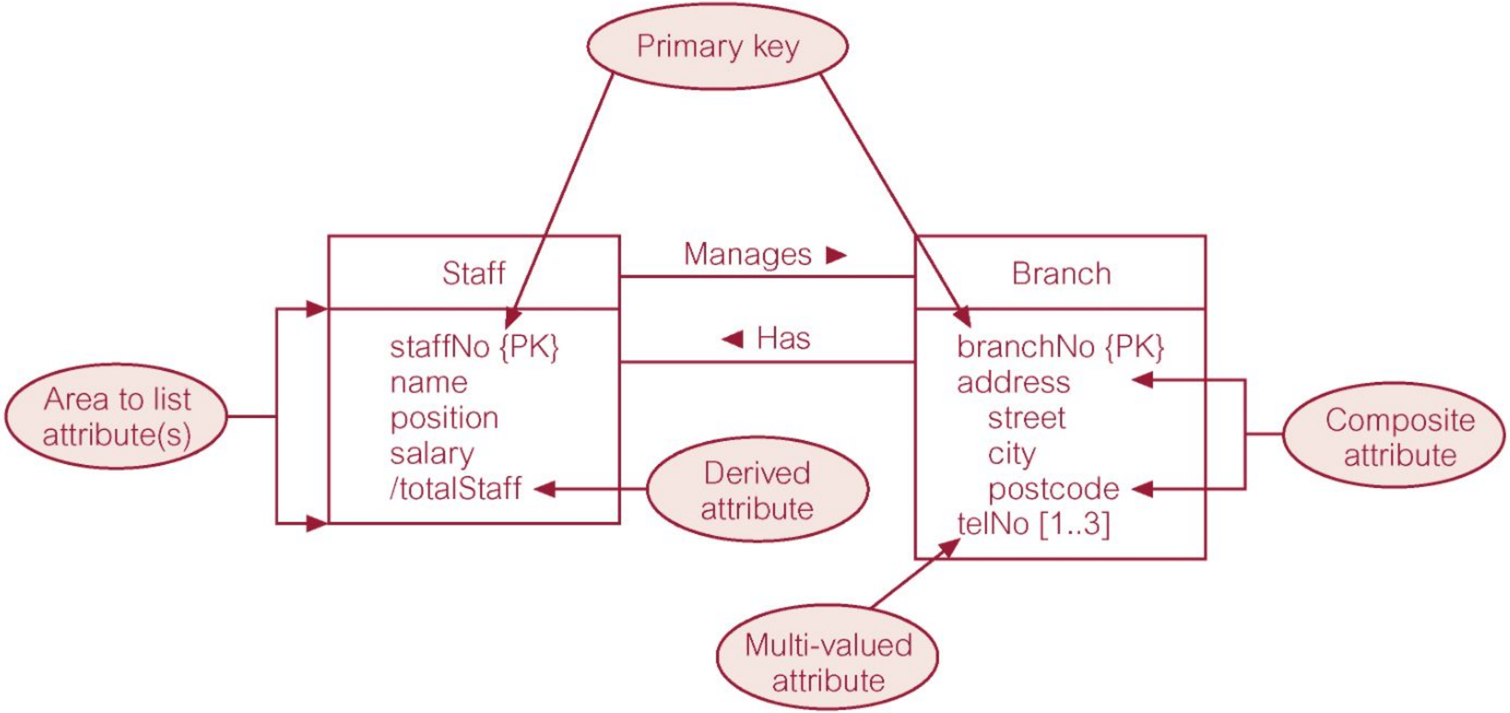
- **Attribute**
  - *property* or *characteristic* of an entity or relationship
  - similar to a column in relational model
  - e.g., Staff attributes: staffNo, name, position, salary
- **Attribute domain**
  - the set of permitted values for the corresponding attribute
- A fully developed data model has to include the domains of each attribute in the ER model
- We list these attributes in the ER diagram of our conceptual model

Attribute Types

- **Simple** attribute – composed of a single component
  - e.g., position, salary, propertyType
- **Composite** attribute – composed of multiple components
  - e.g., address (street, town, postcode), guestName (firstName, lastName)
- **Single-valued** attribute – holds a single value for each occurrence of an entity
  - e.g., postcode
- **Multi-valued** attribute – holds multiple values for each occurrence of entity type
  - e.g., branchTelNumber (*more than one occurrence from the same domain*)
- **Derived** attribute – attribute associated with an entity is not stored individually, but derived from other attributes (in different entities)
  - e.g., branchViewRate (*avg number of viewings per property in any branch*)

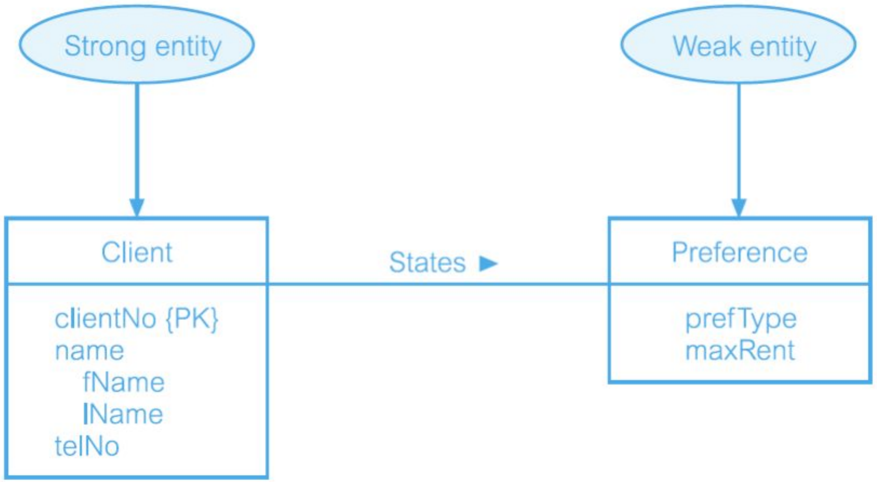
UML Representation of Attributes

CITY UNIVERSITY LONDON
EST 1894

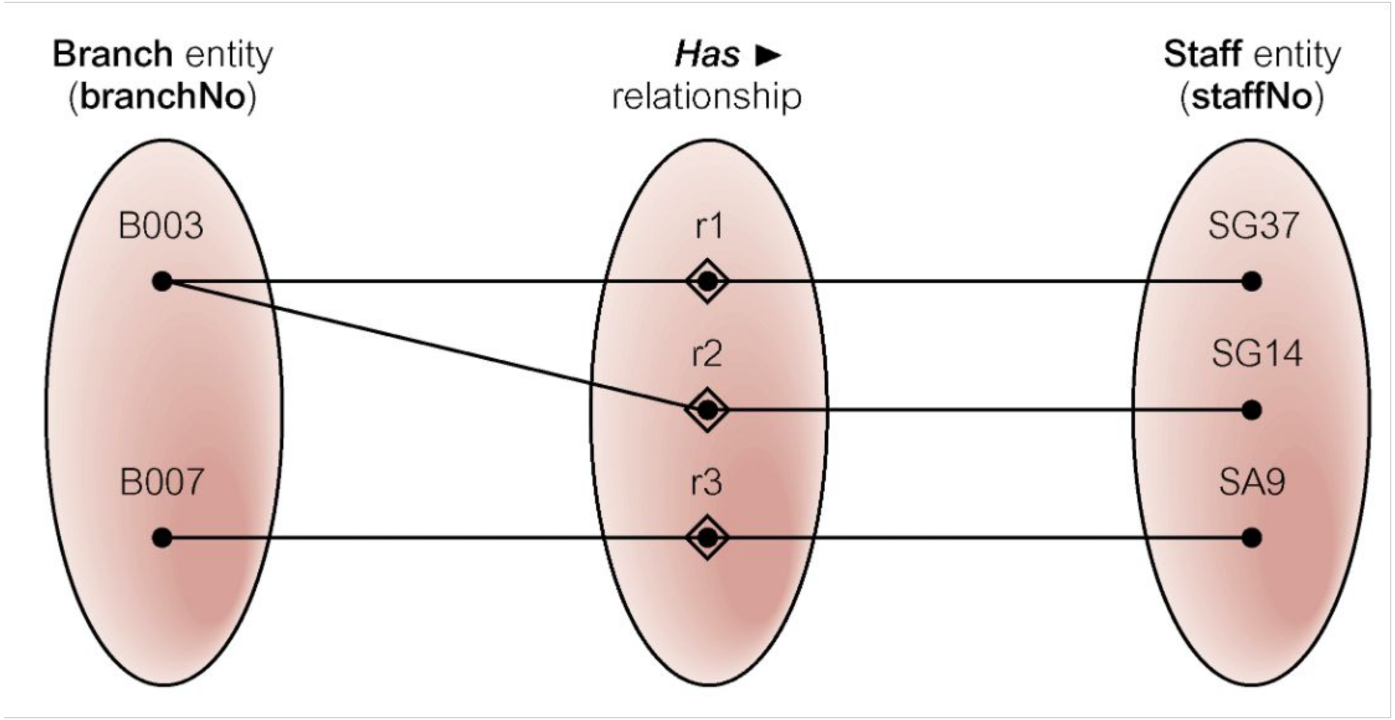


Entity Types

- **Strong entity** - existence does not depend on existence of another entity type
  - contains its own primary key
- **Weak entity** - dependent on the existence of another entity
  - does not have sufficient attributes to form a primary key



Relationship Example



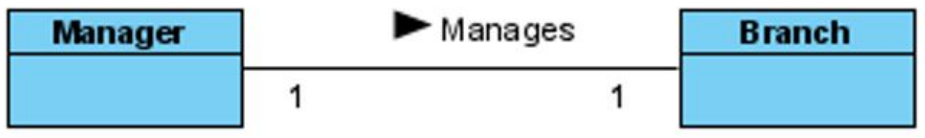
Constraints on Entities & Relationships

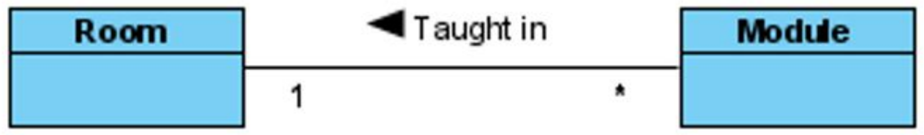
- **Multiplicity** - the main constraint that exists on a relationship
  Defines ***the number of participants*** in a relationship
  - the number of instances of one entity that are associated with one instance of a related entity
  - represents policies (*business rules*) established by user / company
- **Cardinality** - the **maximum number of entity occurrences** of the related entity type
- **Participation** - whether all or only some entity occurrences participate in the relationship or not

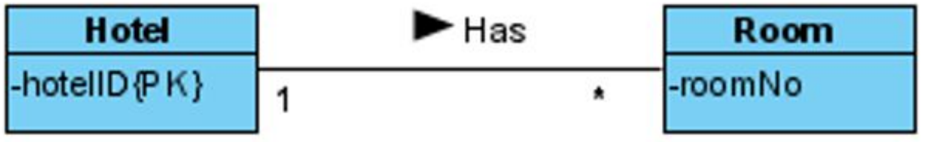
- **Multiplicity = Participation & Cardinality**

Multiplicity for Binary Relationships

- One to one (1:1)
  - exactly one of each entity in the relationship
  - exactly one Manager manages each and every Branch
- One to many (1:*)
  - one Room can hold many Modules… but each Module is taught in only one room
  - one Hotel has many Rooms … but each Room belongs to only one Hotel

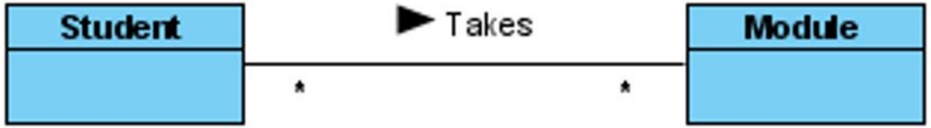


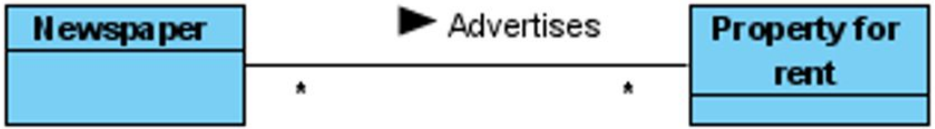




Multiplicity for Binary Relationships

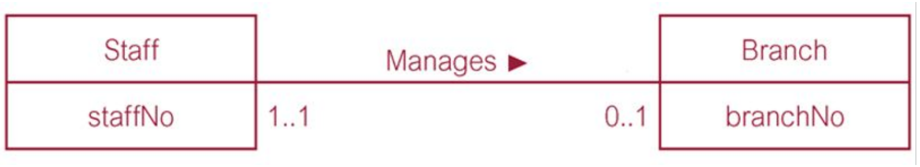
- Many to many (*:*)
  - one Student can take many Modules; and one Module has many Students enrolled on it
  - one Newspaper can advertise many Properties, and one Property can be advertised by many newspapers
- Multiplicity notation (maxEnd1:maxEnd2)



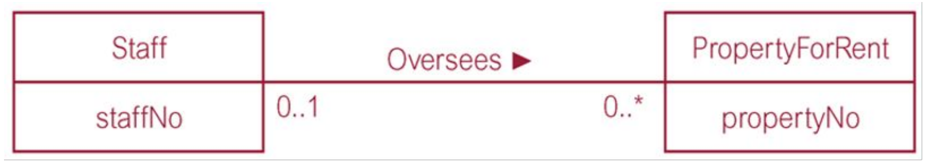


Multiplicity in Diagrams

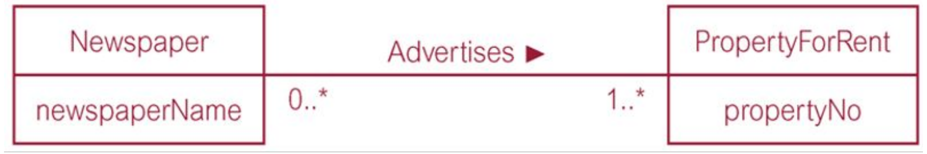
- **(1:1)** – Each branch is managed by **1** member of staff; A member of staff can manage **0 or 1** branch



- **(1:*)** – Each property for rent is overseen by **0 or 1** member of staff; Each member of staff oversees **0 or more** properties for rent



- **(*:*)** – Each property for rent is advertised in **0 or more** newspapers; Each newspaper advertises **1 or more** properties for rent



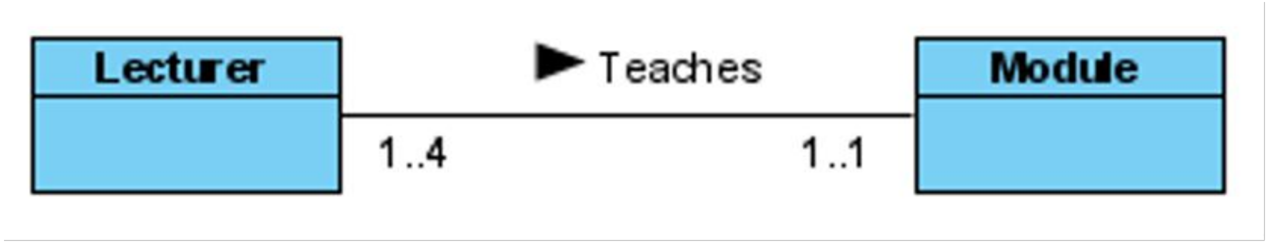
Multiplicity = Participation & Cardinality

- Each end of the relationship is denoted as **(min..max)**
  - **min** – Participation
  - **max** – Cardinality
- **Mandatory participation**: All entity occurrences of the appropriate type must participate in the relationship
  - e.g. every branch must have a manager
- **Optional participation**: Some entity occurrences of the appropriate type participate in the relationship
  - e.g. some, but not all, employees manage branches
  - if min > 0, mandatory participation, otherwise optional
- **Cardinality** constraints show how many *relationships per entity* (max number)
  - if both sides have max=1 1:1 cardinality
  - if both sides have max>1 M:N cardinality
  - otherwise 1:N or N:1 cardinality, depending on which side has max > 1

Example 1: Multiplicity, Cardinality, Participation

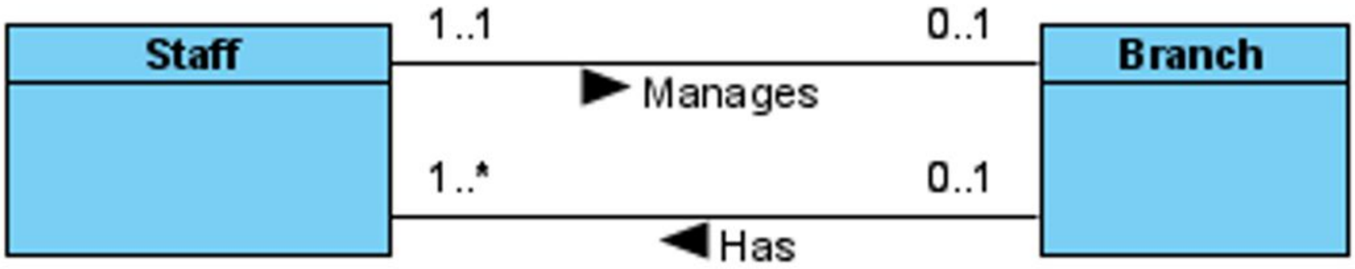
CITY UNIVERSITY LONDON
EST 1894

- The multiplicity (1..4) can be interpreted as follows:
  - the ‘1’ represents the **participation** and indicates that one module must be taught by at least one lecturer (minimum)
  - the ‘4’ represents the **cardinality** and indicates that one module can be taught by up to four lecturers (maximum)
- (1..4) – for each instance of *Module* entity type
  - there must be at least 1 lecturer but no more than 4 different lecturers
- (1..1) – for each instance of *Lecturer* entity type
  - there must be exactly 1 module

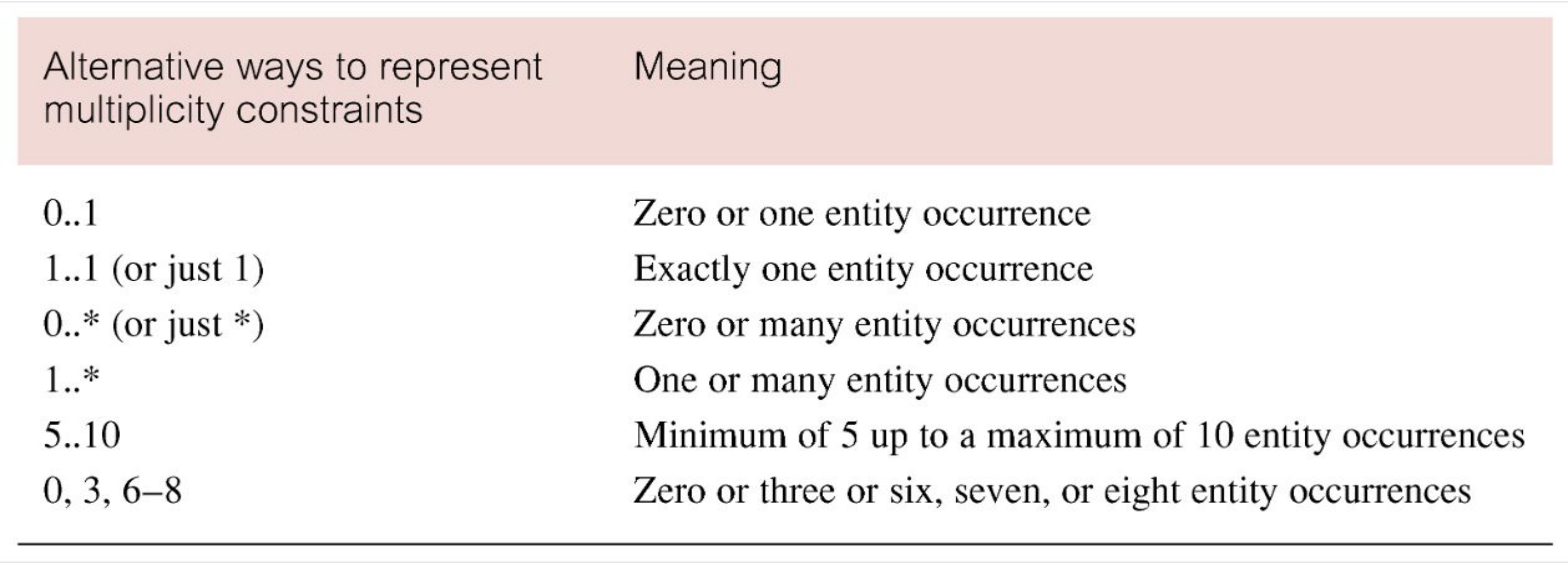


Example 2: Multiplicity, Cardinality, Participation

- Let's consider the relationship – ***Manages***
  - a *Staff* member may manage 0 or max 1 *Branch*
  - a *Branch* must be managed by 1 *Staff* member
- Let's consider the relationship – ***Has***
  - a *Branch* may have 1 or more staff members
  - a *Staff* member does not have to be attached to a *Branch* and if they are - can be attached to only one *Branch*



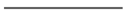
Summary of Multiplicity Constraints



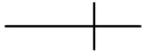
Alternative ways to represent multiplicity constraints	Meaning
0..1	Zero or one entity occurrence
1..1 (or just 1)	Exactly one entity occurrence
0..* (or just *)	Zero or many entity occurrences
1..*	One or many entity occurrences
5..10	Minimum of 5 up to a maximum of 10 entity occurrences
0, 3, 6–8	Zero or three or six, seven, or eight entity occurrences

Multiplicity in UML

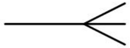
- Examples of ends



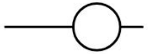
One



Mandatory

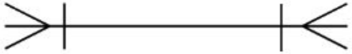


Many

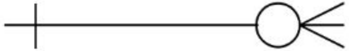


Optional

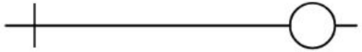
- Examples of connections



Many-to-many



One-to-many



One-to-one

ER Diagram Example

Modelling data

Finsbury library stores data about their book collection, borrowers and loans.

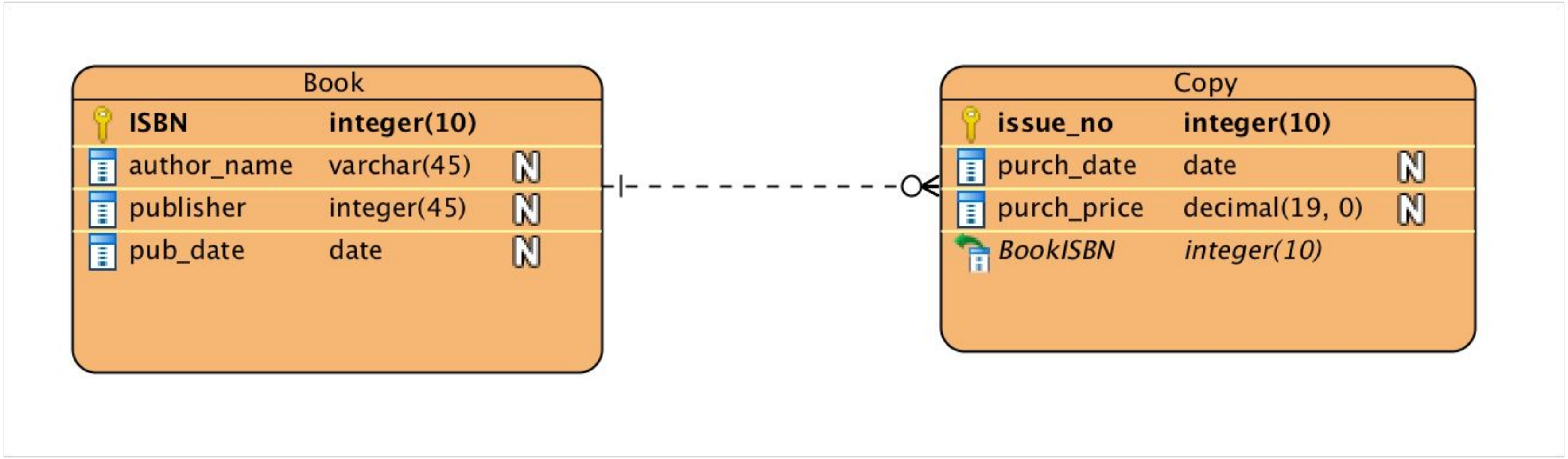
The library can have a number of copies of each book, each copy being identified by its issue_No. The library needs to keep details of the date copies were purchased and the purchase price; purch_date and purch_price. Books are identified by their ISBN and the library also needs to keep details of the Authors name, the Publisher and the book's publication date; author_name, publisher, pub_date.

To help readers the library keeps details of Authors; author_name, date_of_birth, and a short biography.

Borrowers are identified by a borrower_id and details of their name and address are kept.

When a borrower borrows books the date they borrow a book is stored as is the date they return the book.

ER Diagram Example



Book		
ISBN	**integer(10)**	
author_name	varchar(45)	N
publisher	integer(45)	N
pub_date	date	N

Copy		
issue_no	**integer(10)**	
purch_date	date	N
purch_price	decimal(19, 0)	N
BookISBN	*integer(10)*	

ER Diagram Example

Book

ISBN (PK)	author_name	publisher	pub_date
3781934748	JK Rowling	Bloomsbury	26/06/1997
3954756389	Charles Dickens	null	16/08/1853

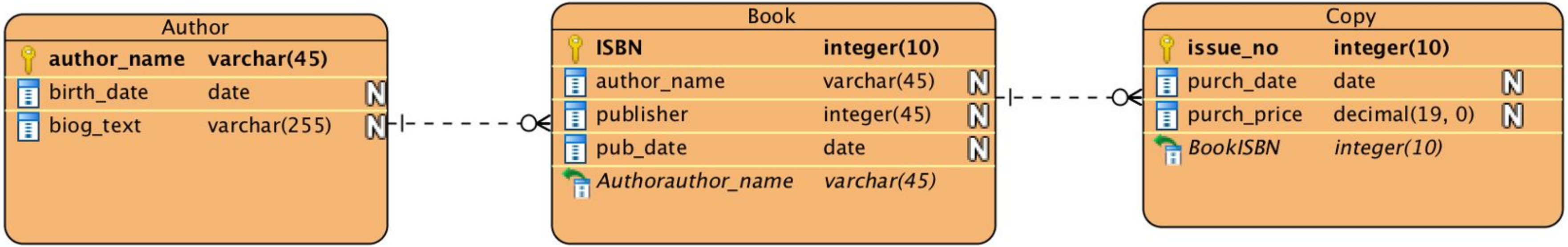
Copy

Issue_no (PK)	purch_date	purch_price	ISBN
6372827387	12/03/2015	9.99	3781934748
7329373872	15/04/2016	11.99	3781934748
1349284933	19/02/2001	6.99	3954756389

ER Diagram Example

To help readers the library keeps details of Authors; author_name, date_of_birth, and a short biography.

ER Diagram Example



Author

🔑	**author_name**	**varchar(45)**	
	birth_date	date	N
	biog_text	varchar(255)	N

Book

🔑	**ISBN**	**integer(10)**	
	author_name	varchar(45)	N
	publisher	integer(45)	N
	pub_date	date	N
	Authorauthor_name	*varchar(45)*	

Copy

🔑	**issue_no**	**integer(10)**	
	purch_date	date	N
	purch_price	decimal(19, 0)	N
	BookISBN	*integer(10)*	

ER Diagram Example

Borrowers are identified by a borrower_id and details of their name and address are kept.

When a borrower borrows books the date they borrow a book is stored as is the date they return the book.

ER Diagram Example

Naive representation

Copy

Issue_no (PK)	……	Borrower_id1	Borrower_id2	Borrower_id3	Borrower_id4	…
1262528		3748493	846389	3948493	5683023	
…..						

OR

Borrower

Borrower_id (PK)	……..	Issue_no1	Issue_no2	Issue_no3	Issue_no4	….
……						

ER Diagram Example

Copy

Issue_no (PK)	purch_date	purch_price	ISBN
6372827387	12/03/2015	9.99	3781934748
7329373872	15/04/2016	11.99	3781934748
1349284933	19/02/2001	6.99	3954756389

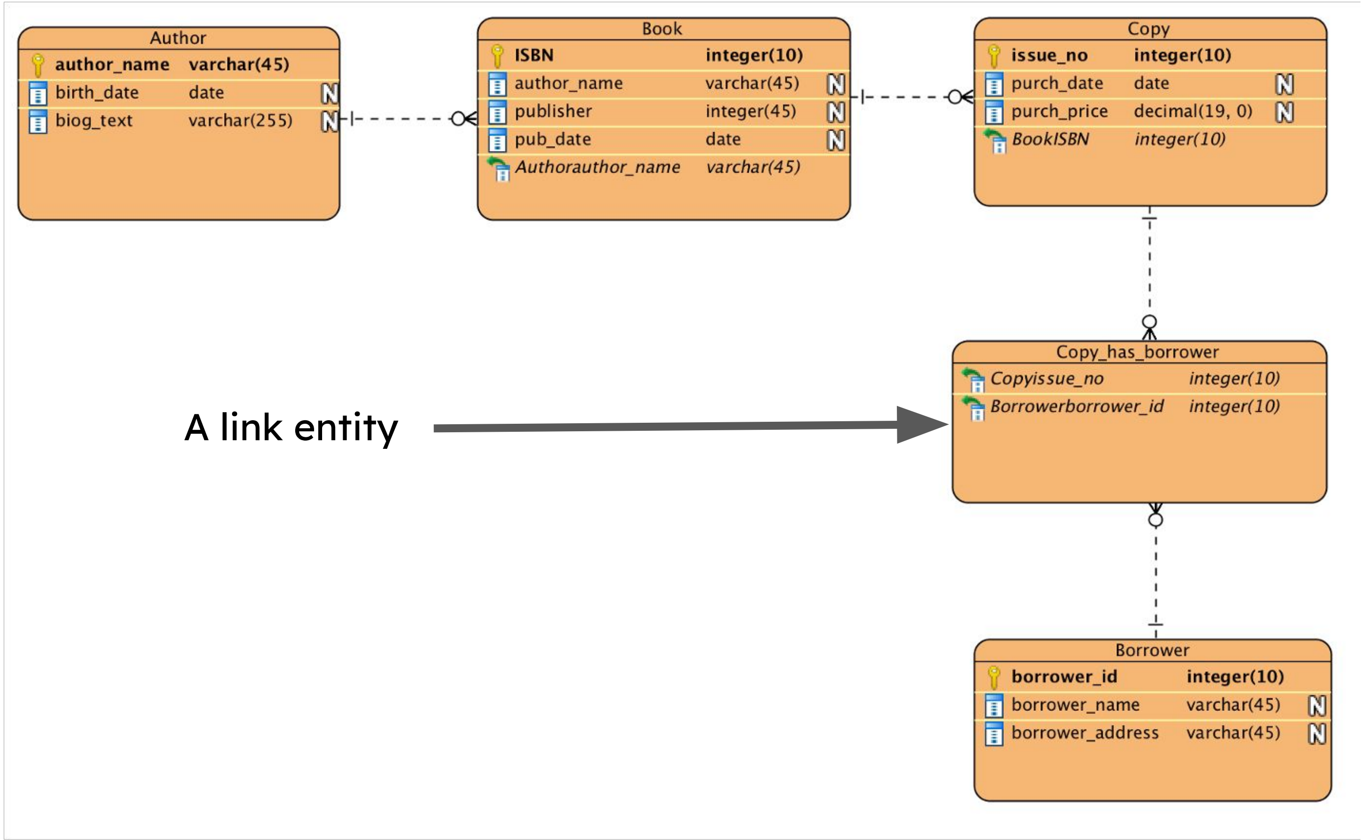
Borrower

Borrower_id (PK)	Borrower_name	Borrower_address
2738748929	Chris Smart	Northampton Square EC1V 0HB
3732832929	Charles Watson	Seckforde St, EC1V 2HS

Loan

Issue_no (PK)	Borrower_no (PK)
6372827387	2738748929
7329373872	3732832929
6372827387	3732832929

ER Diagram Example



A link entity

Author

author_name	varchar(45)	
birth_date	date	N
biog_text	varchar(255)	N

Book

ISBN	integer(10)	
author_name	varchar(45)	N
publisher	integer(45)	N
pub_date	date	N
Authorauthor_name	*varchar(45)*	

Copy

issue_no	integer(10)	
purch_date	date	N
purch_price	decimal(19, 0)	N
BookISBN	*integer(10)*	

Copy_has_borrower

Copyissue_no	*integer(10)*
Borrowerborrower_id	*integer(10)*

Borrower

borrower_id	integer(10)	
borrower_name	varchar(45)	N
borrower_address	varchar(45)	N

ER Diagram Example

Copy

Issue_no (PK)	purch_date	purch_price	ISBN
6372827387	12/03/2015	9.99	3781934748
7329373872	15/04/2016	11.99	3781934748
1349284933	19/02/2001	6.99	3954756389

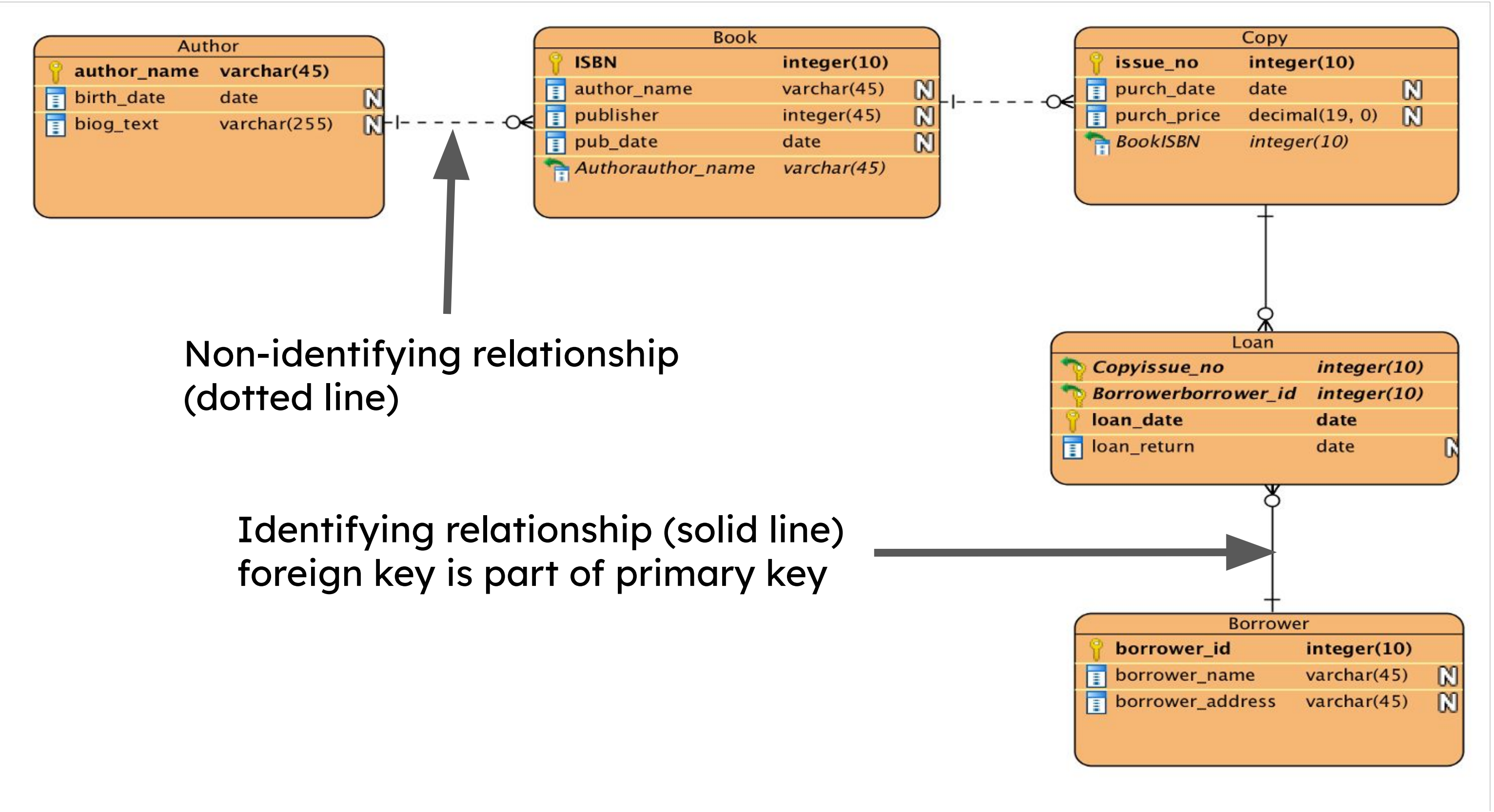
Borrower

Borrower_id (PK)	Borrower_name	Borrower_address
2738748929	Chris Smart	Northampton Square EC1V 0HB
3732832929	Charles Watson	Seckforde St, EC1V 2HS

Loan

Issue_no (PK)	Borrower_no (PK)	Load_date (PK)
6372827387	2738748929	13/06/2019
7329373872	3732832929	21/06/2019
6372827387	3732832929	04/07/2019
6372827387	2738748929	15/08/2019

ER Diagram Example



Author		
author_name	varchar(45)	
birth_date	date	N
biog_text	varchar(255)	N

Book		
ISBN	integer(10)	
author_name	varchar(45)	N
publisher	integer(45)	N
pub_date	date	N
Authorauthor_name	*varchar(45)*	

Copy		
issue_no	integer(10)	
purch_date	date	N
purch_price	decimal(19, 0)	N
BookISBN	*integer(10)*	

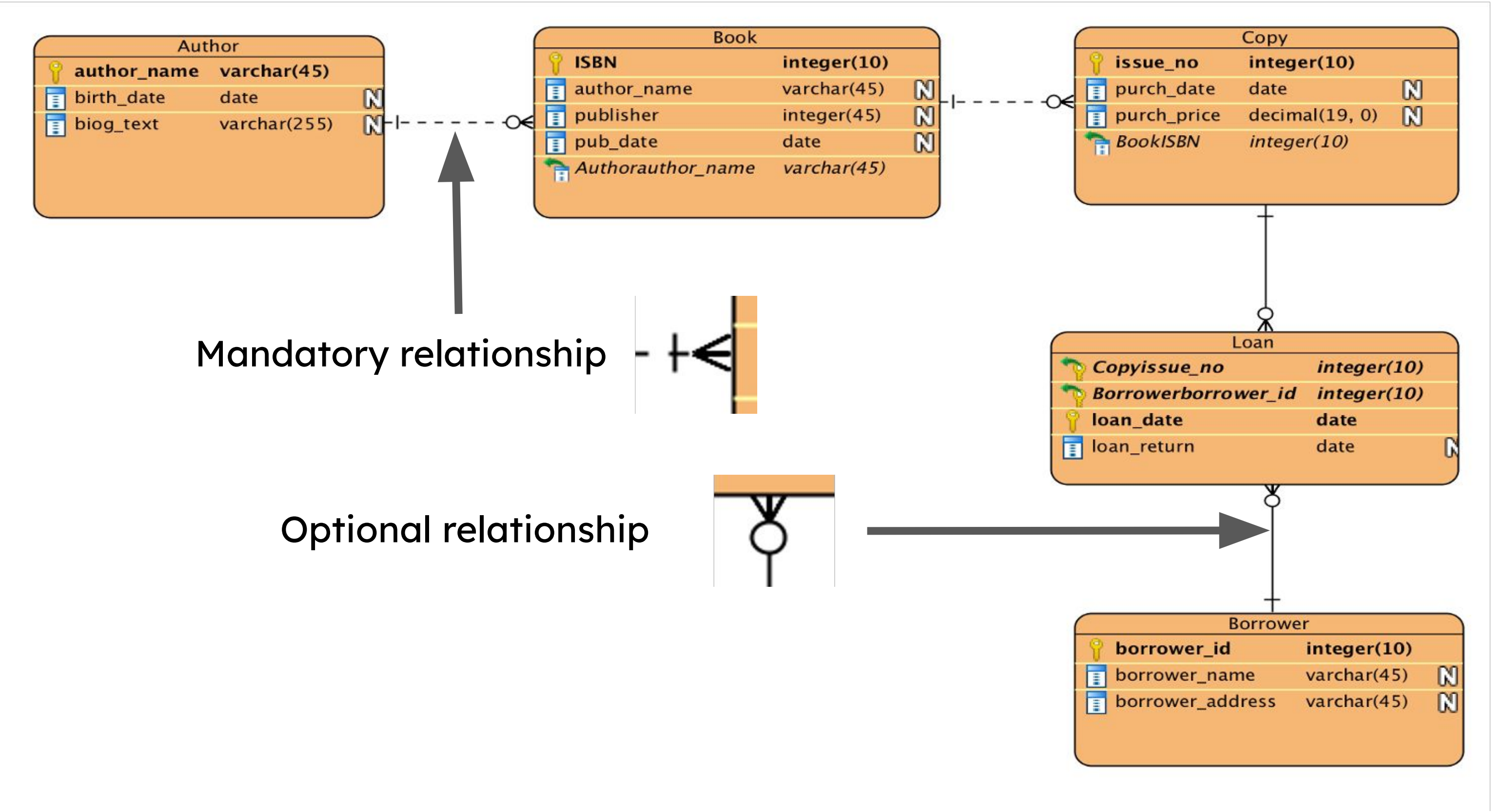
Loan		
Copyissue_no	***integer(10)***	
Borrowerborrower_id	***integer(10)***	
loan_date	**date**	
loan_return	date	N

Borrower		
borrower_id	integer(10)	
borrower_name	varchar(45)	N
borrower_address	varchar(45)	N

Non-identifying relationship
(dotted line)

Identifying relationship (solid line)
foreign key is part of primary key

ER Diagram Example



Author

author_name	**varchar(45)**	
birth_date	date	N
biog_text	varchar(255)	N

Book

ISBN	**integer(10)**	
author_name	varchar(45)	N
publisher	integer(45)	N
pub_date	date	N
Authorauthor_name	*varchar(45)*	

Copy

issue_no	**integer(10)**	
purch_date	date	N
purch_price	decimal(19, 0)	N
BookISBN	*integer(10)*	

Loan

Copyissue_no	***integer(10)***	
Borrowerborrower_id	***integer(10)***	
loan_date	**date**	
loan_return	date	N

Borrower

borrower_id	**integer(10)**	
borrower_name	varchar(45)	N
borrower_address	varchar(45)	N

Mandatory relationship

Optional relationship

ER Diagram Example

To help readers the library keeps reviews of books; reviewer, review_date, review_text.

Entity/attribute questions

- Is the primary key sufficient to uniquely identify a single instance in all circumstances?
- Should an attribute be nullable?

Relationship questions

Before you draw the relationship

- Is the relationship identifying?
- If the primary key of the parent entity (the one end) should be part of the primary key of the child entity (the many end) then it is an identifying relationship

After you draw the relationship

- Is the relationship mandatory or optional?
- Check both ends of the relationship