Relational Databases

Spring 2025

Last time

Conceptual model

What is the data about?

Logical model

How do we represent the data in a specific (kind of) database?

Physical model

How is the data represented in memory or on disk?

Last time

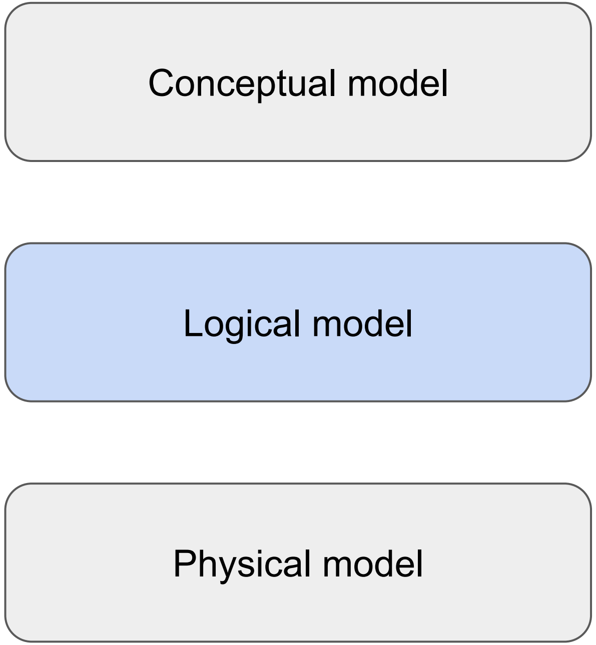
Entity relationship model

- a way to analyze data that you'll be trying to represent in a database
- understand the **entities** you want to represent
- understand the **relationships** between those entities you want to capture

It is a **conceptual** model

- not tied to any specific kind of database
- how do we represent data in a specific kind of database?

Today



Conceptual model

What is the data about?

Logical model

How do we represent the data in a specific (kind of) database?

Physical model

How is the data represented in memory or on disk?

Relational databases

Many databases use a logical model based on **relations**

- relational model ⇔ relational databases
- relations ⇔ tables
- tuples ⇔ tables rows (= records)

Formalized in the 1970 by Edgar Codd, developed by IBM and Oracle

- Easy model to understand
- Supports a powerful query language (SQL)
- Flexible enough for most data uses

Relations and tables

Mathematically, a relation R is a subset of $D_1 \times D_2 \times \ldots \times D_k$

- a set of **tuples** of the form $(d_1, d_2, \ldots, d_k)$
- $D_1, D_2, \ldots, D_k$ are **domains** of the relation

Example: $< \subseteq \mathbb{N} \times \mathbb{N} = \{ (0, 1), (0, 2), (1, 2), (0, 3), (1, 3), (2, 3), \ldots \}$
where $(3, 4) \in <$ is usually written $3 < 4$

A relational database represent a relation via a **table**

- each **row** is a tuple of the relation
- **columns** get names and types for convenience (= schema of the table)

Relations and tables

Each table should define a **primary key**

- a set of columns that uniquely identifies each row

Values in a row may be blank (= **null**) except for the primary key

A table row may refer to a row in another table (via that row's primary key)

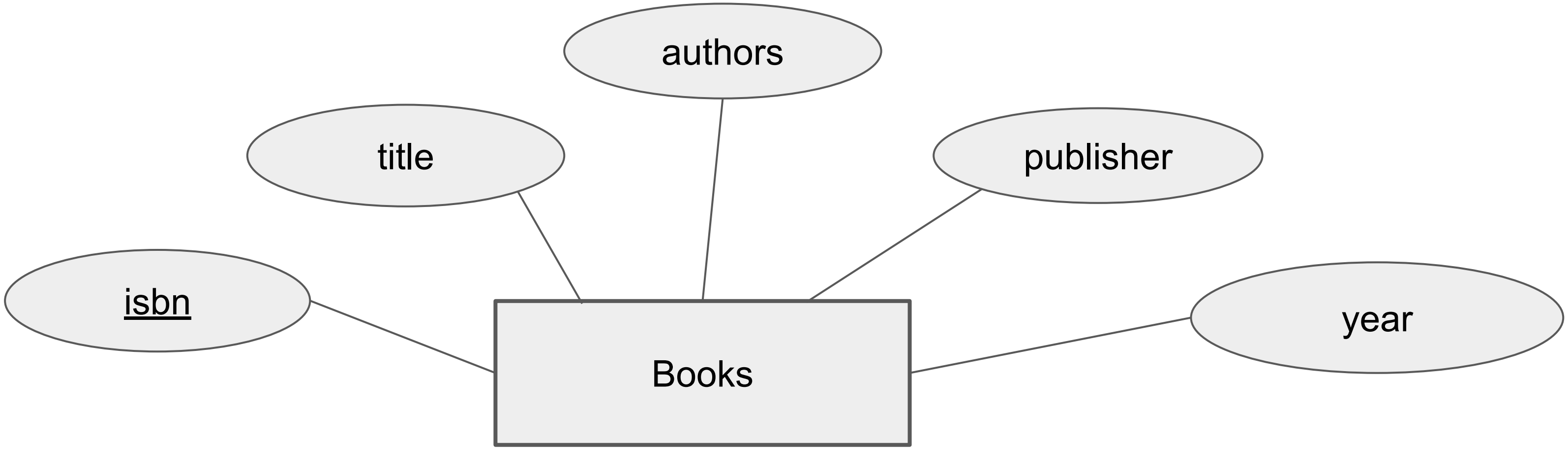
- we call that a **foreign key**
- it's an indication to the database of how we can use a field
- database can enforce foreign key constraints (value exists as a key)

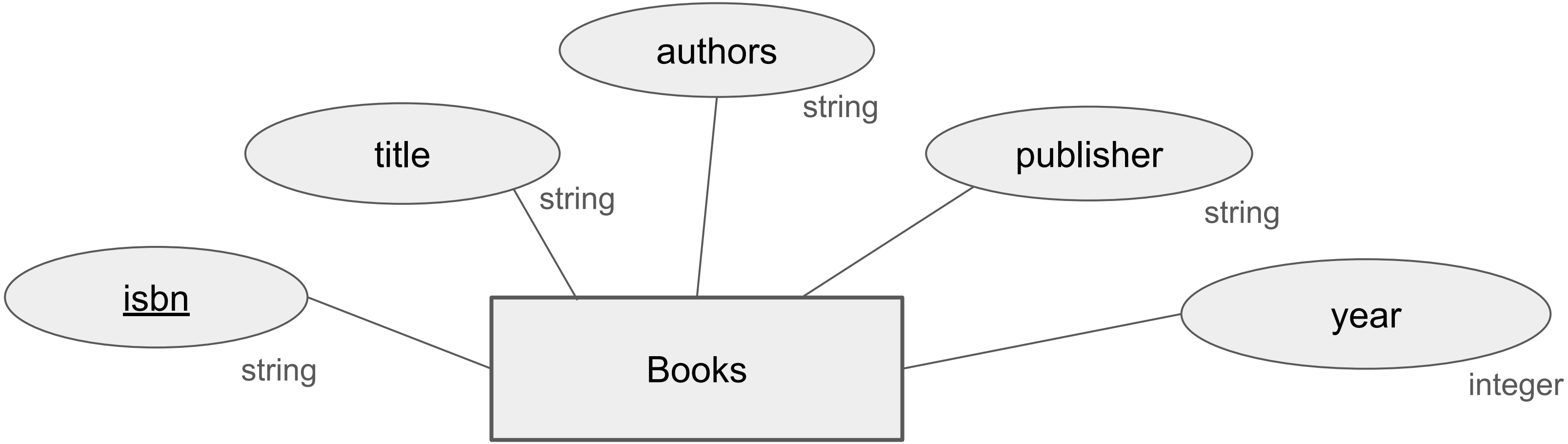
ER ⇒ relational model: entities

Suppose we have an ER diagram capturing the structure of our data

We use tables to represent entity sets in our ER diagram

table	=	entity set
table column	=	entity attribute
table row	=	entity in a set
table primary key	=	entity set primary key


authors
title
publisher
isbn
year
Books


authors
string
title
string
publisher
string
isbn
year
string
Books
integer

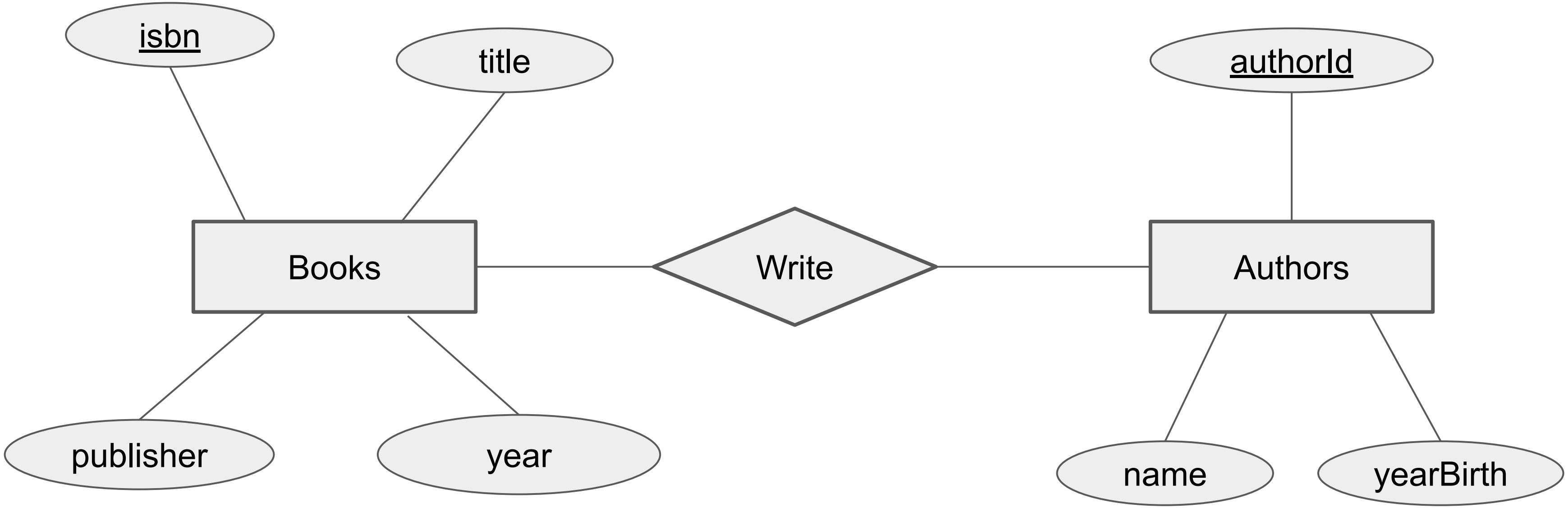
Books

isbn	**title**	**authors**	**publisher**	**year**
0771595565	Rebel Angels	Robertson Davies	McMillian	1981
0316296198	The Magus	John Fowles	Little Brown & Co	1965
0670312134	Fifth Business	Robertson Davies	McMillian	1970
…	…	…	…	…

ER ⇒ relational model: relationships

We use tables to represent relationship sets in our ER diagram

table	=	relationship set
table column	=	**foreign key** to an entity in the relationship
		also, relationship attribute
table row	=	relationship in a set
table primary key	=	entity set primary key


isbn
title
authorId
Books
Write
Authors
publisher
year
name
yearBirth

Books

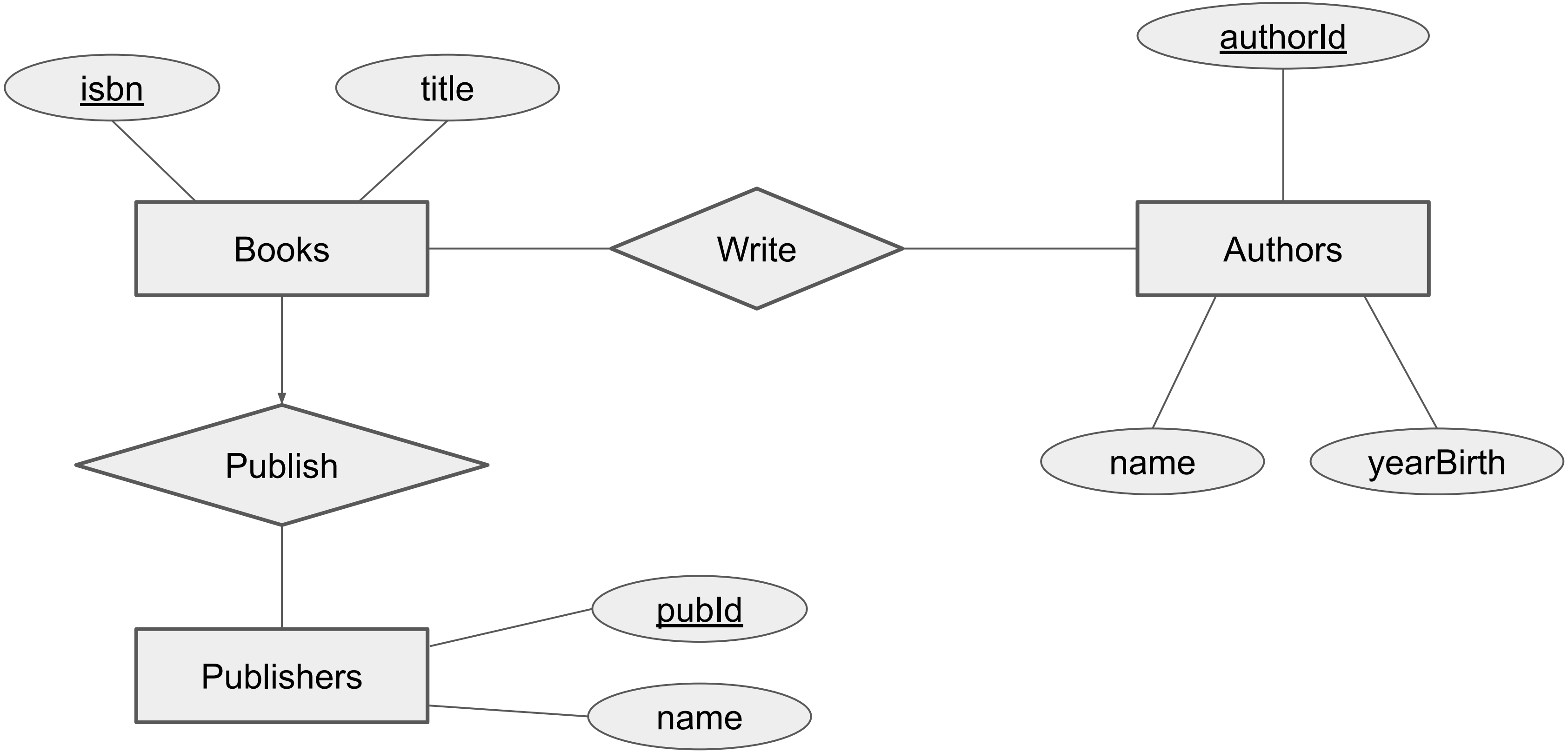
isbn	**title**	**publisher**	**year**
0771595565	Rebel Angels	McMillian	1981
0316296198	The Magus	Little Brown & Co	1965
0670312134	Fifth Business	McMillian	1970

Authors

authorId	**name**	**yearBirth**
1	Robertson Davies	1913
2	John Fowles	1926

Write

isbn	**authorId**
0771595565	1
0316296198	2
0670312134	1


isbn
title
Books
Write
Authors
authorId
name
yearBirth
Publish
Publishers
pubId
name

Books

isbn	title	year
0771595565	Rebel Angels	1981
0316296198	The Magus	1965
0670312134	Fifth Business	1970

Authors

authorId	name	yearBirth
1	Robertson Davies	1913
2	John Fowles	1926

Publishers

pubId	name
101	McMillian
102	Little Brown & Co

Write

isbn	authorId
0771595565	1
0316296198	2
0670312134	1

Publish

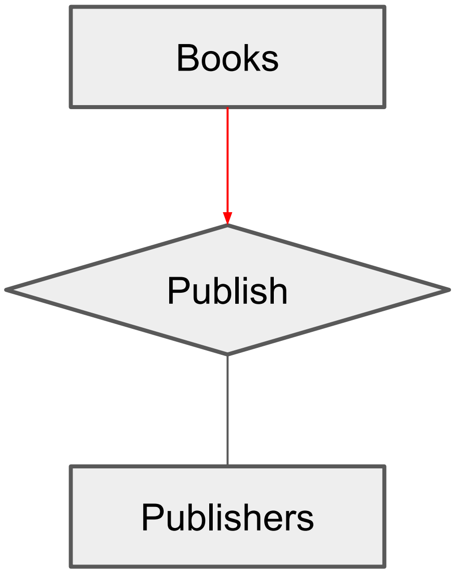
isbn	pubId
0771595565	101
0316296198	102
0670312134	101

ER ⇒ relational model: optimizing 1:N relationships

If we know a relationship is 1:1 or 1:N, we can simplify

- every book has at most one publisher

Skip the **Publish** table
Put the publisher key in the **Books** table



Books

foreign key

isbn	title	year	pubId
0771595565	Rebel Angels	1981	101
0316296198	The Magus	1965	102
0670312134	Fifth Business	1970	101

Write

isbn	authorId
0771595565	1
0316296198	2
0670312134	1

Authors

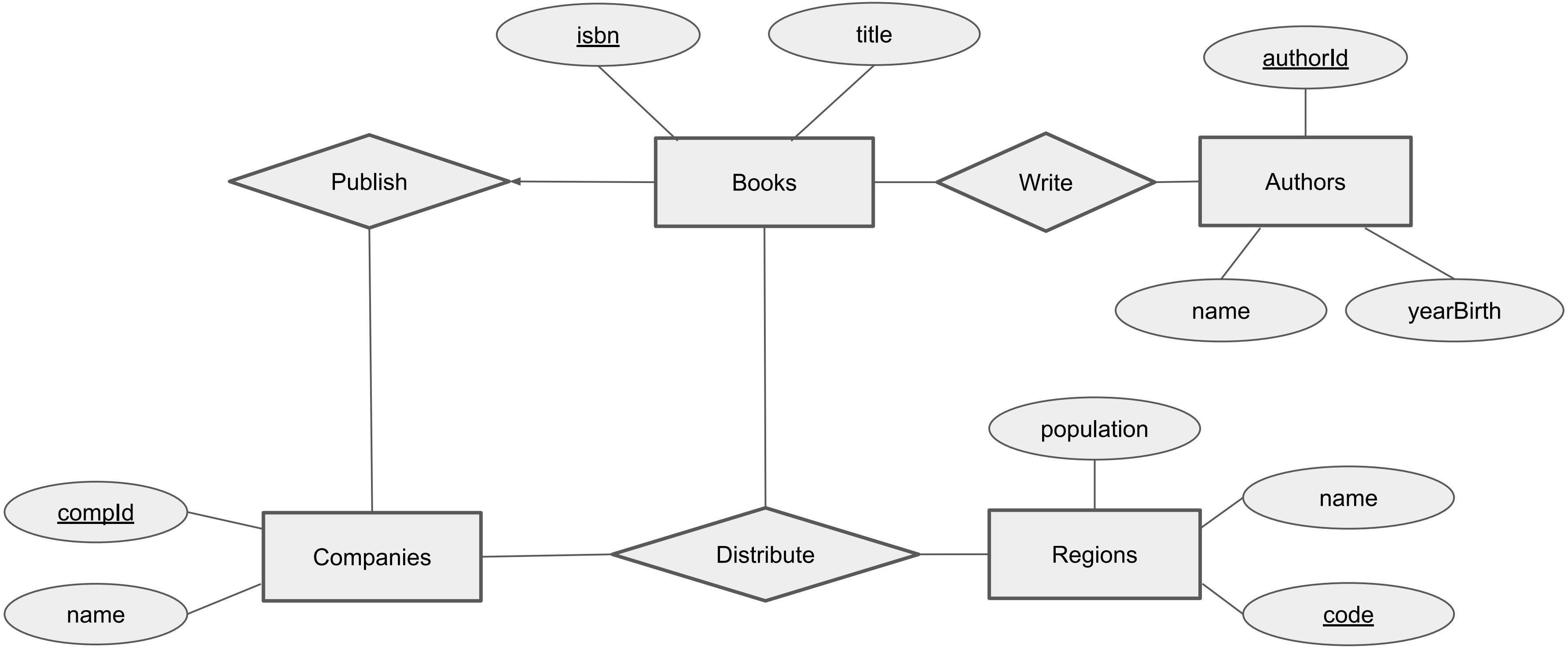
authorId	name	yearBirth
1	Robertson Davies	1913
2	John Fowles	1926

Publishers

pubId	name
101	McMillian
102	Little Brown & Co

~~isbn~~	~~pubId~~
~~0771595565~~	~~101~~
~~0316296198~~	~~102~~
~~0670312134~~	~~101~~

Can we also remove table Write, while retaining the ability to have multiple authors per book?



Easy exercise

In practice

How does the above work with a specific (relational) database?

- most databases run as a **server**
- use a **client** to interact with a database

We will use **SQLite**, a local *database as a library* that lives in a file on disk

- SQLite is not a server, it's a code library that uses a file as a database
- we can use SQLite interactively with the provided client
    - **sqlite3 *name.db***
- a file (= a database) can host multiple tables

In practice

Relational database API: **SQL**

- unlike most modern systems, the API is not a set of endpoints
- a string representing instructions sent to the DB

All relational databases support SQL to (1) manipulate tables/data (2) query data

- much more on querying data with SQL next time

SQL operations on tables

DDL (data definition language): subset of SQL for table CRUD operations

Create a table:

```
CREATE TABLE {name} (...)
```

Update a table (structurally — add column, etc)

```
ALTER TABLE {name} ...
```

Delete a table:

```
DROP TABLE {name}
```

SQL operations on rows

Add a row to a table:

```
INSERT INTO {table} VALUES ({value$_1$}, {value$_2$}, …)
```

Update rows in a table

```
UPDATE {table} SET {field} = {value} WHERE {row condition}
```

Delete rows in a table

```
DELETE FROM {table} WHERE {row condition}
```

Read rows from a table

```
SELECT {f1}, {f2}, … FROM {table}
SELECT {f1}, {f2}, … FROM {table} WHERE {row condition}
```

That's all, folks!