Introduction to relational databases

Tables

Relational databases organize data into tables

Tables can be linked together

A table is a relation.

Is a relation also a table?

ISBN	Title
7654321123456	Creating relational databases for fun and profit
9876543212345	Relational databases for really, really smart people
3212345678909	My life with relational databases: a memoir
8172635412345	Relational databases: an existential journey

Figure 1: Some sample books

Table organization

Each row in the table describes a single book

The data is organized into columns

Each *entry* (or cell) contains **a single piece of data**.

How do we handle a book with two authors?

Linked tables

Two tables can be linked to obtain more information.

Needs an identifier (ID) for the each row

Books

Table 1: books

book_id	ISBN	title
1	7654321123456	Creating relational databases for fun and profit
2	9876543212345	Relational databases for really, really smart people
3	3212345678909	My life with relational databases: a memoir
4	8172635412345	Relational databases: an existential journey

Authors

Table 2: Authors

author_id	last_name	first_name
1	Lopez Baranda	Christina
2	Jin-Soon	Sin
3	Jones	Hannah
4	Novak	Stanislaw
5	Turay	Tandice
6	Roy	Shanta
7	Berger	Henry
8	Khatami	Paree

BooksAuthors

Table 3: BooksAuthors

book_id	author_id
3	6
2	4
2	5
1	1
1	3
1	5
4	8

Editions

Table 4: Editions

edition_id	book_id	date_of_publication	edition_number
1	3	2001	1
2	3	2003	2
3	4	2003	1
5	1	2000	1
6	3	2005	3
8	2	2012	1
9	3	2009	4

Tables, relationships, and IDs

This intermediate table (in this example, BooksAuthors) is known as a "relation" or "join" table. For this method of breaking up data into multiple tables to work reliably, we need to ensure that each row in the books table and each row in the authors table can be referenced uniquely. To do this, we need to assign identifiers to each rown in the book and authors tables, and we use those identifiers to relate the two tables to each other in the third table. In our example, we will call these ID columns 'book_id' and 'author_id'. We will see some examples of these identifiers in the query examples below.

"One-to-many" relationships don't use a third table. This type of relationship links two tables, one containing the data that is on the "one" side of the relationship and the other that is on the "many" side. For example, each book can have many editions, but each edition applies to only a single book:

Tables, relationships, and IDs

One-to-many relationships also require that rows in tables have unique IDs, but unlike in the join table used in many-to-many relationship, the table that contains the data describing the "many" side of the relationship has a column reserved for the ID of the "one" side of the relationship.

The IDs used to uniquely identify the things described in tables are called "primary keys". If the primary key of one table is used in another table, the key in the other table is called a "foreign key" in that table. For example, the "book_id" column in the Books table is that table's primary key, but the "book_id" column in the Editions table is a foreign key. The purpose of foreign keys is to link the two tables in a one-to-many relationship.

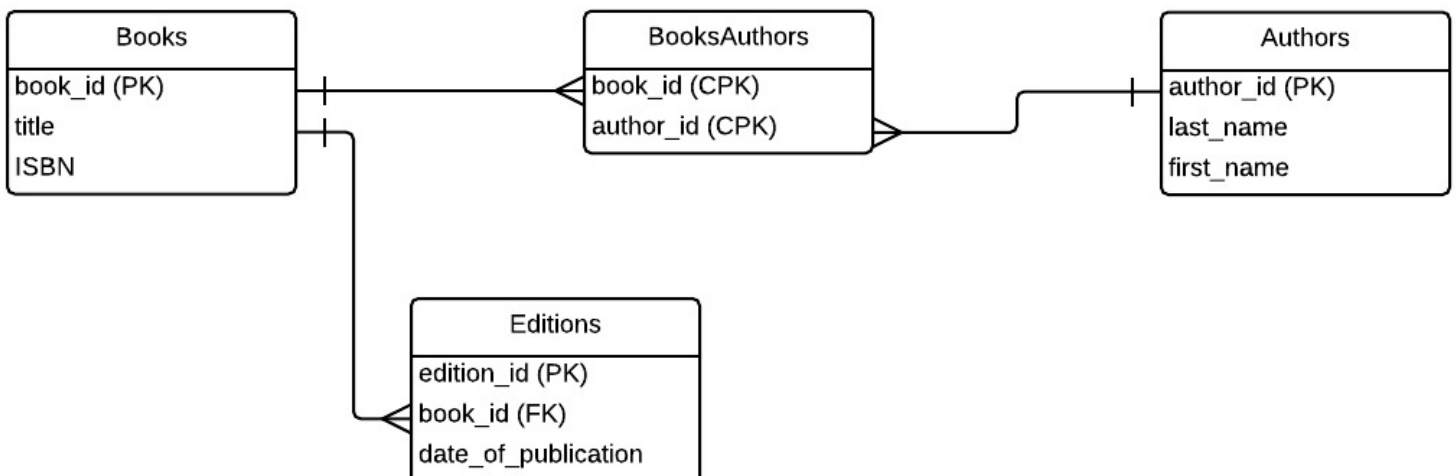
Tables, relationships, and IDs

You may be wondering why we didn't use ISBN for the unique ID for each row in the Books table. We could have done that, but there is a problem with ISBNs: it is easy for a human operator to make an error while entering them. Since primary keys need to be unique, we don't want to use something as the primary key that we can't trust to be unique. If we used ISBNs as primary keys, and we encountered one that was the same as an ISBN that was already in our database due to data-entry error, the database would not save the row. If your rows have attributes that you can be absolutley sure will be unique, you can use that attribute as a primary key, but it's usually safer, and a common convention, to let the RDBMS assign an automatically assigned number (an ordinary integer) as the primary key.

Tables, relationships, and IDs

For join tables, the primary key for each row is the unique *combination* of the foreign keys from the two joined tables. In our example, the primary key of BooksAuthors is the combination of book_id and author_id. A primary key that is comprised of more than one attribute is called a "composite key."

Putting together all of our tables, we get a database structure that can be represented like this:



Tables, relationships, and IDs

Books, Authors, and Editions all have a unique ID (book_id, author_id, and edition_id respectively) that is used as their primary key (labelled as PK in the diagram above), and Editions contains the foreign key (FK) book_id that links it to the Books table in a one-to-many relationship. The join table BooksAuthors only has two columns, book_id and author_id, which are both foreign keys that make up a composite primary key (CPK).

(It should be noted that this database portrays the relationships between books, authors, and editions in rather simplistic terms. For example, different manifestations of a book such as a paperback and an ebook usually have different ISBNs, and two different editions of a book can have different authors. Despite these issues, the database will suffice as an example of a simple relational model.)

License