RDBMS
Relational Database Management Systems

CS/IT 490 WD, Fall 2013

Last update 2013-10-17

Written by Rachel J. Morris

Video Introduction

- How relational databases work, lynda.com: http://www.youtube.com/watch?v=z5YnKt2aOCs
- W3Schools guide: http://www.w3schools.com/sql/default.asp

Normalization

- Let's say your company wants to store data, and decide to use a Spreadsheet in lieu of a database.
- That might look like this:

Written by Rachel J. Morris

Normalization

Product ID	Name	Product Type	Manufacturer Name	Price	Sale Amount
0	GeForce GTX 770	Video Card	nVidia	$430.00	-$30.00
1	GeForce GTX 670	Video Card	nVidia	$354.00	-$90.00
2	Radeon HD 7790	Video Card	AMD	$130.00	$0.00
3	Radeon HD 7750	Video Card	AMD	$100.00	$0.00
4	AMD FX-6300	Processor	AMD	$120.99	$0.00
5	AMD FX-8350	Processor	AMD	$200.00	$0.00
6	Intel Core i7-4770K	Processor	Intel	$340.00	-$10.00
7	Intel Core i7-3570K	Processor	Intel	$220.00	$0.00

Written by Rachel J. Morris

Normalization

- Note that the “Product Type” and “Manufacturer Name” contain duplicate data – Video Card, Processor, etc.
- Any time we're dealing with strings, it's very easy to have typo errors.

Product ID	Name	Product Type	Manufacturer Name	Price	Sale Amount
0	GeForce GTX 770	Video Card	nVidia	$430.00	-$30.00
1	GeForce GTX 670	Video Card	nVidia	$354.00	-$90.00
2	Radeon HD 7790	Video Card	AMD	$130.00	$0.00
3	Radeon HD 7750	Video Card	AMD	$100.00	$0.00
4	AMD FX-6300	Processor	AMD	$120.99	$0.00
5	AMD FX-8350	Processor	AMD	$200.00	$0.00
6	Intel Core i7-4770K	Processor	Intel	$340.00	-$10.00
7	Intel Core i7-3570K	Processor	Intel	$220.00	$0.00

Normalization

- Instead, we would want to store "Video Card" and "Processor" separately, where we can identify each with a number ID.
- This is similar to how we could use an Enumeration in C++

Product ID	Name	Product Type	Manufacturer Name	Price	Sale Amount
0	GeForce GTX 770	0	0	$430.00	-$30.00
1	GeForce GTX 670	0	0	$354.00	-$90.00
2	Radeon HD 7790	0	1	$130.00	$0.00
3	Radeon HD 7750	0	1	$100.00	$0.00
4	AMD FX-6300	1	1	$120.99	$0.00
5	AMD FX-8350	1	1	$200.00	$0.00
6	Intel Core i7-4770K	1	2	$340.00	-$10.00
7	Intel Core i7-3570K	1	2	$220.00	$0.00

Product Type ID	Name
0	Video Card
1	Processor

Manufacturer ID	Name
0	nVidia
1	AMD
2	Intel

Normalization

- There are multiple levels of Normalization:
  - 1st Normal Form
  - 2nd Normal Form
  - 3rd Normal Form
  - And more, but we usually only care about 3rd normal form.

Written by Rachel J. Morris

Normalization

1st Normal Form:

- No duplicate data in tables. This leads to modification confusion (WHICH do you update?), possible typo errors
- Create multiple tables; group data that is related in one table.
- Each row has a unique identifier – could be a primary key, or a series of columns forming a primary key.

http://databases.about.com/od/specificproducts/a/normalization.htm

Normalization

2nd Normal Form:

- Remove duplicate subsets of information in our table rows, place in separate tables.
- Have relationships between these tables with foreign keys
- Similar to using the “Product Type” table and pointing to the “Product Type ID”.

http://databases.about.com/od/specificproducts/a/normalization.htm

Normalization

3rd Normal Form:

- Remove the columns that are not dependent on the primary key.
  - Don't store excess information in one table!
  - A “Student” record doesn't really need to be storing their Address – Address can go in a separate table and point to a specific student via Student ID.

http://databases.about.com/od/specificproducts/a/normalization.htm

SQL Queries

- You will usually interact with your MySQL database via a user interface.
  - PHPMyAdmin for MySQL on the web
  - SQL Server for Microsoft's SQL Server
  - SQLite Browser
  - Etc.
- You will also access and modify the database via queries, which are strings.

SQL Queries

- Usually, you will pre-build your database with a GUI tool, and not be creating/dropping tables from your PHP code.
- Then, you'll use SQL Queries to ask for information from those tables, or insert, update, or delete information.

Table Modification

- Create Table
  - Allows you to create a table in your database.
  - Must specify column names, attributes of those columns (primary key, not null, etc.)
- Alter Table
  - Modify an existing table
- Drop Table
  - Remove table & all its data

Data Modification

- Insert
  - Insert data into a specific table
- Update
  - Update existing data based on some criteria; “where price > 20.00”
- Select
  - Select data from one or more tables
- Delete
  - Remove a record from the database

Data Modification

- We will go over how to write queries in another lecture.

Written by Rachel J. Morris

Relationships between Tables

- Relationships between tables is one of the core reasons for using an “RDBMS” solution.
- Otherwise, if our data weren't *related* to other data, why wouldn't we just save everything in a text file?

Written by Rachel J. Morris

Relationships between Tables

- Relationships could be:
  - Employee → Employee (Boss)
  - Product → Brand
  - Student → Student/Class → Class
- If we didn't have primary keys and foreign keys set up, we would have to *parse all of the data* to try to find matches in the criteria.
- The database already does this for us, much more efficiently than we could!

Some Column data types

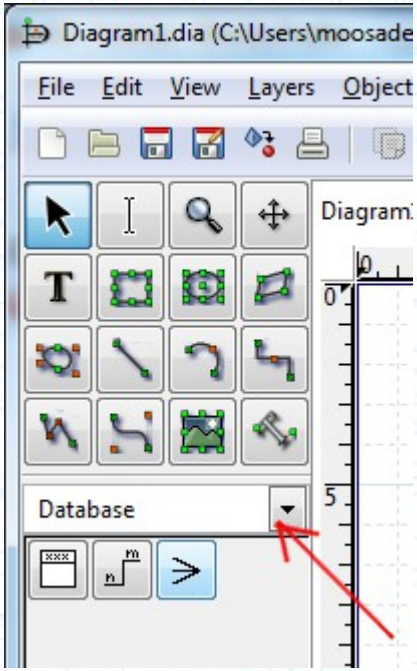
- Integer
- Varchar
- Bool
- Float
- Decimal
- Date
- Time
- Datetime
- Varchar
- Binary

DB Schema Diagrams

- Download Dia, a *gratis*, open source diagramming tool, available for Windows, Linux, & Mac from:
- http://dia-installer.de/
- Or:
- http://portableapps.com/apps/office/dia_portable
- (Gratis means without charge. “Free” software could be gratis or libre, so that word gets confusing. Yes, I'm channeling RMS here)

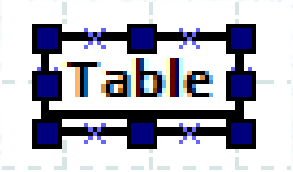
Written by Rachel J. Morris

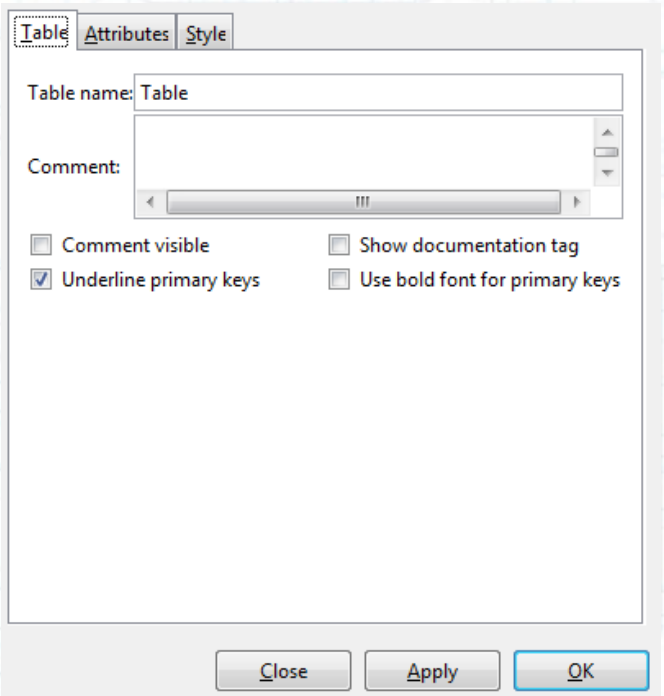
DB Schema Diagrams



- In Dia, select “Database” from the Dropdown box.
- UML could be useful for diagramming programs.
- Hey, there are only three buttons for databases!

DB Schema Diagrams

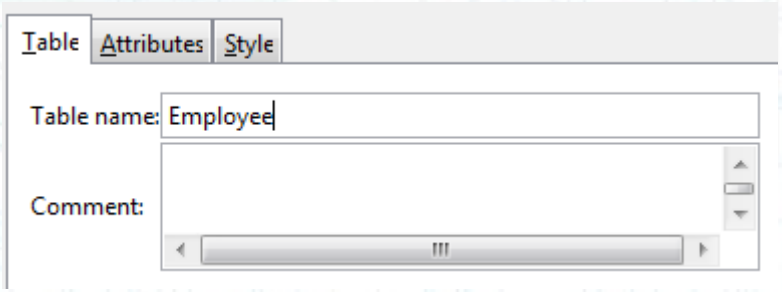




- Select the first button and click on the empty grid area, an empty table diagram will show up.
- Double-click that table to edit it.

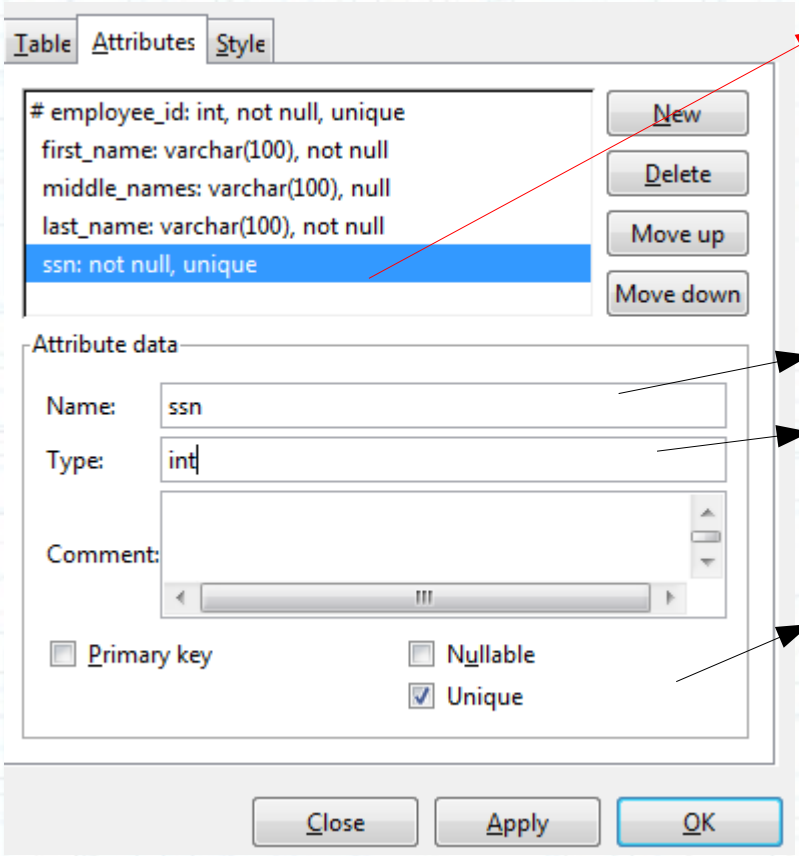
DB Schema Diagrams

Give the table a name:



DB Schema Diagrams

And under Attributes, we can set up the column information.



DO NOT STORE Social Security Numbers as plaintext in a database! This is just an example!

Column Name

Data type (int, varchar, ...)

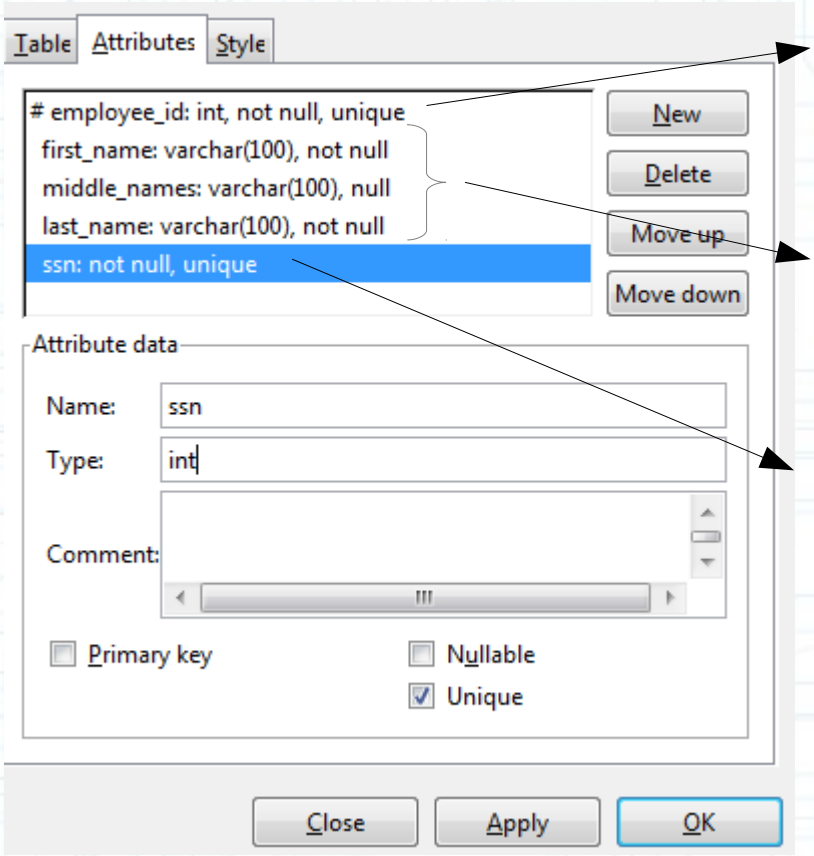
Properties: is it a primary key?

Can it be null or must it be filled in?

Does it have to be unique from all the other rows?

DB Schema Diagrams

And under Attributes, we can set up the column information.



employee_id is set as the **primary key**, which automatically sets the **not null** and **unique** attributes.

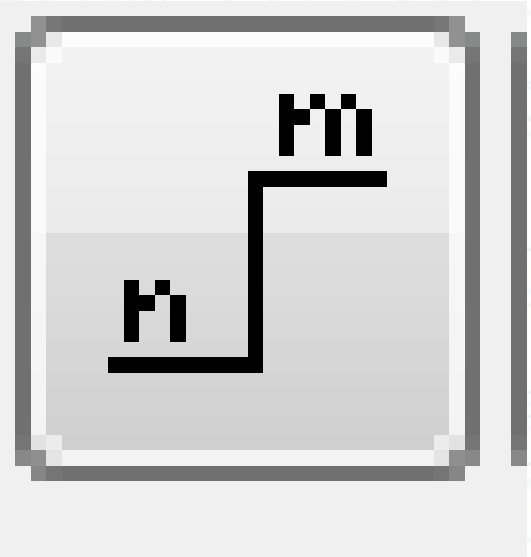
first_name and last_name are **not null** (they must be filled in), but the middle_names field is optional.

ssn is **not null**, and must be **unique**, because no two people should have the same social security number.

Written by Rachel J. Morris

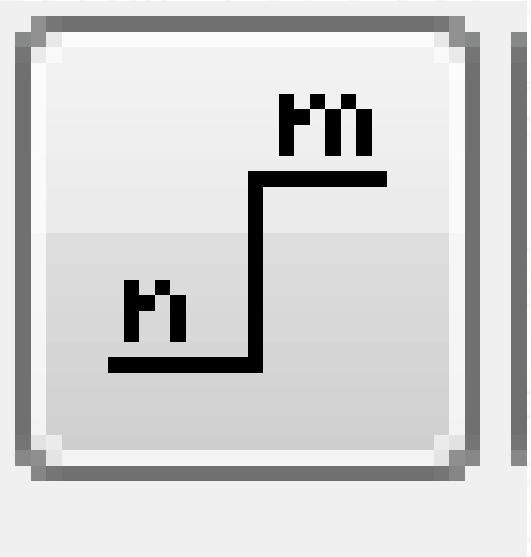
DB Schema Diagrams

- Once we have multiple database tables, we can mark a “relationship” between them with the second button:



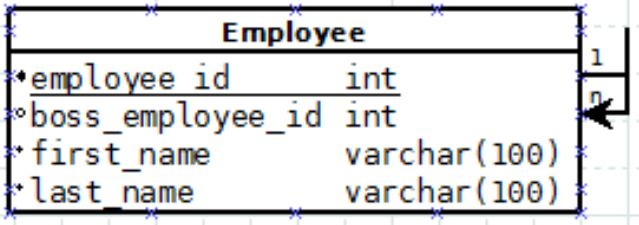
DB Schema Diagrams

- Once we have multiple database tables, we can mark a “relationship” between them with the second button:

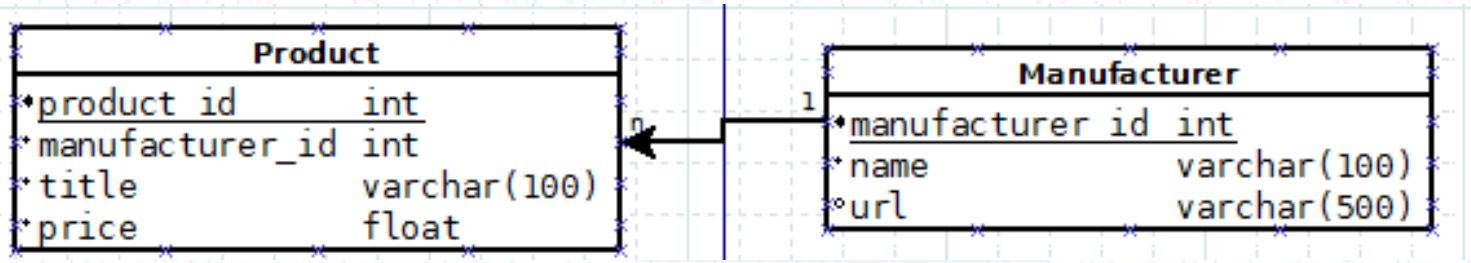


DB Schema Diagrams

- A table can be related to itself:

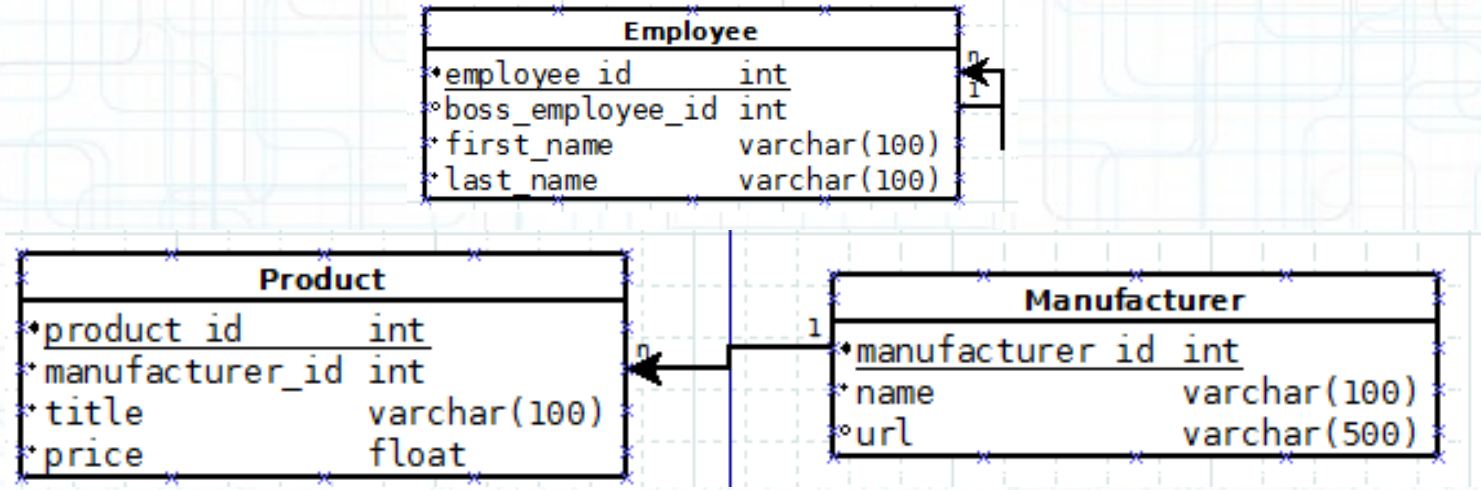


- Or another table:



DB Schema Diagrams

- Notice the small “1” and “n” markers, and where the arrow points.
- These represent “1-to-n” relationships.
- One boss can have many employees
  - (1 Boss : n employees)



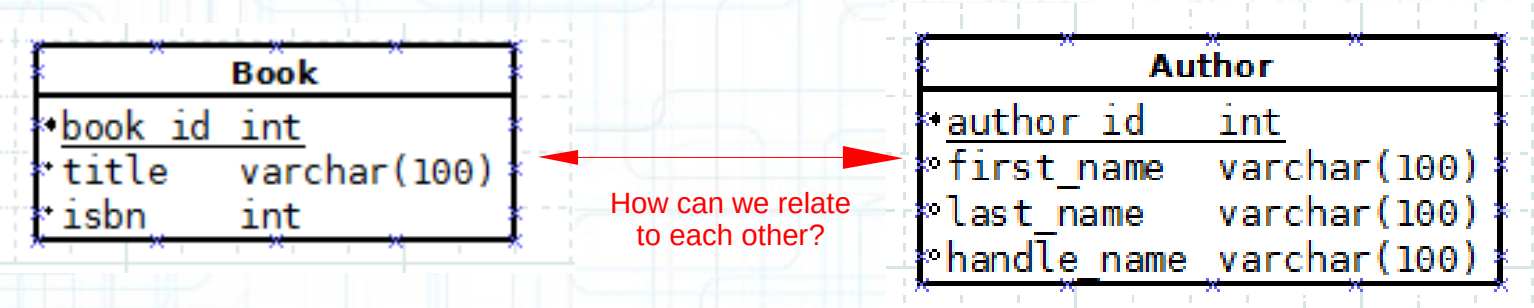
Written by Rachel J. Morris

DB Schema Diagrams

- 1 Manufacturer : n Products
- You can also have a 1-to-1 relationship
  - 1 Person might have 1 Email Address that we store, and each 1 Email Address belongs only to 1 Person.
  - Note that “Email address” could also be a column in a “Person” table.
  - But, if you make “Email address” belong to a separate, 1 : 1 table, you avoid any NULL values in the Person table.

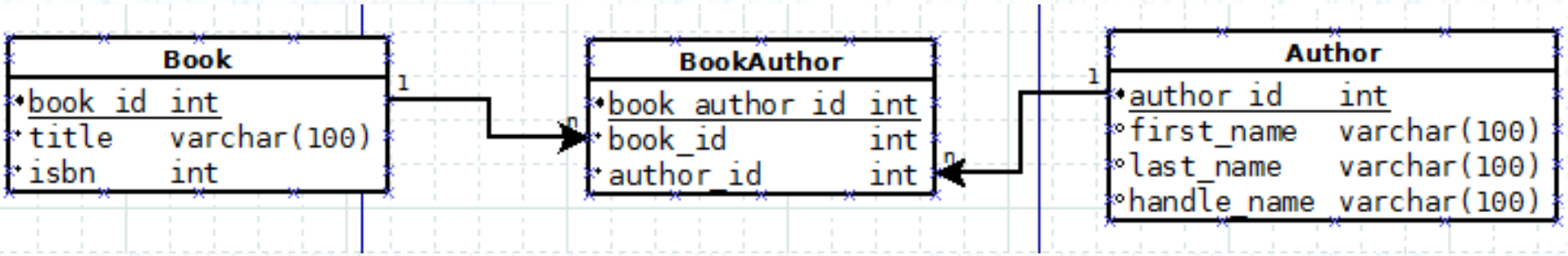
DB Schema Diagrams

- n : n relationships (many-to-many)
- An author can have many books, and a book can have multiple authors.
- However, you cannot simply change the “1 / n” line on the diagram tool, and you cannot just mark a “n:n” flag in the database.
- You need to design it in a specific way...



DB Schema Diagrams

- With an n : n relationship, you need to have an intermediate table to store the combination of the two:



- The Book table has a 1 : n relationship with the BookAuthor table, and the same with Author.
- Now we can have a series of records that combine a BOOK ID and an AUTHOR ID. We can have multiple Book IDs and multiple Author IDs.
- This gives us a n : n relationship.

Design Notes

- I will usually name my primary keys [tablename]_id.
  - This is because when you're doing queries with JOIN, it will make identifying each column easier, rather than having duplicate “id” columns!
  - It also makes matching the foreign-key/primary-key more explicitly clear.
- I usually name my n : n intermediate tables [table1][table2].
  - It helps you know what two tables are linked.

Let's create some schema diagrams!

- For posterity

References

-

Written by Rachel J. Morris