7.1.5 Create ERDs

We are ready to map our data. By now, we're comfortable with what we're working with. We have an idea of the data connections, and we're ready to use our new tool: Quick DBD.

There are several ways to refer to the map we're about to create. It's also called a flowchart, an entity relationship diagram, and a schema. We'll be using all of these terms in this module, though "ERD" is the most specific.

There are three forms of ERDs: conceptual, logical, and physical. We'll start by helping Bobby with the most basic of the three, the conceptual diagram. As we add more information to our tables, such as data types and keys, we'll advance through the more complex diagrams.

The following video will discuss different types of ERDs as well as why they're useful.

Conceptual Diagrams

A **conceptual diagram** is an ERD in its simplest form. To create one, we only need two things: a table name and column headers.

It's simple because we're creating just the *concept* of the diagram. By covering only the basics, it's easier to capture the main points. If we tried to capture everything at once (data types, location of the primary and foreign keys, etc.), we're more likely to overlook a crucial item.

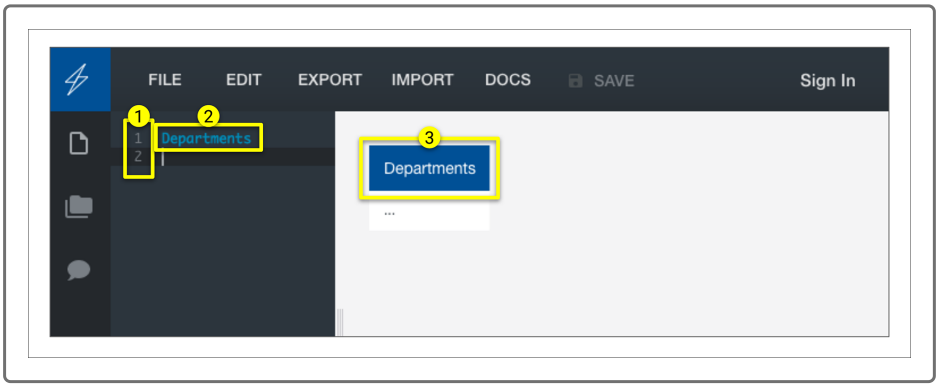
Returning to the Quick DBD website, let's create the blueprint (or schema) for our first table. The schema generates the actual diagram, and we're creating schema instead of coding in the text editor.

IMPORTANT

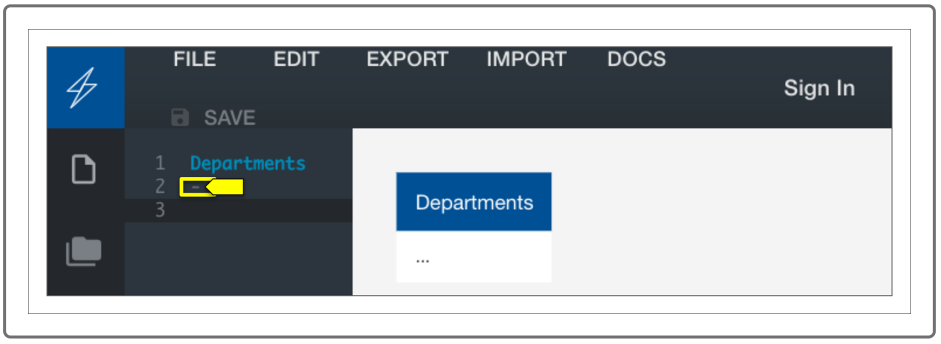
> **Schema** is a term that will come up often while working in SQL and its extensions (such as MySQL, PostgreSQL, and many others). It references the design of the database, and specifically how the tables and their relationships are mapped out.

We'll use `departments.csv` for the first table, so let's review the worksheet again, as we'll need to know the column names.

The first thing to create in the text editor is the name of the new table. Enter "Departments" on the first line, then hit Enter. Read the following "About Quick Database Diagrams" user notes.

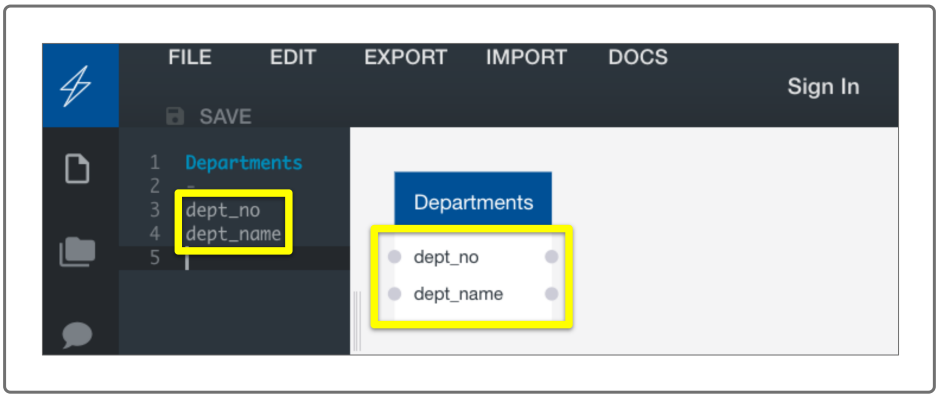


Below the table name, insert a hyphen to signify we're ready to start adding data.



Now add placeholders for your data. Remember, for a conceptual diagram, don't load all of the details. This diagram should only represent the relationships between the tables, so all we need is the name of each column. There are two column headers in `departments.csv`: dept_no and dept_name. Add those to our schema.

On the next blank line in the text editor, input "dept_no" and hit Enter. On the next line, type "dept_name" and hit Enter again. You'll see that the small table has grown a bit!



Did you notice "int" to the right of our table data? That's the default data type assigned by Quick DBD. It's there because we didn't specify what data type each of the headers represent.

Retake

Create a table for the next CSV, `dept_emp.csv`, by following these steps:

1. After dept_no, skip one line to leave it blank.
2. On the next line, start the new table by entering its name "Dept_Emp."

3. On the next line, enter a hyphen to signify a new table, ready for data.
4. On the following lines, enter a column header for each column name.

IMPORTANT

> The syntax is important when creating schema in Quick DBD. When creating tables, make sure to use underscores (_) instead of spaces. A space within a table or column name will generate an error.

Notice the second table has appeared next to the first—this is the beginning of our flow chart.

These tables aren't locked into place and can be rearranged for a cleaner look. To move a table, left click on the table name and drag to place. It's a useful feature because as we continue to add tables to the diagram the screen can become cluttered, and it allows you to organize the tables as you see fit.

The next step in building a diagram is to assign data types, which transitions our conceptual ERD to a logical ERD.

SKILL DRILL

Create conceptual diagrams for the remaining CSV files. Remember, we're mapping out the entire database in this one diagram, so we need to include all of the CSV files.

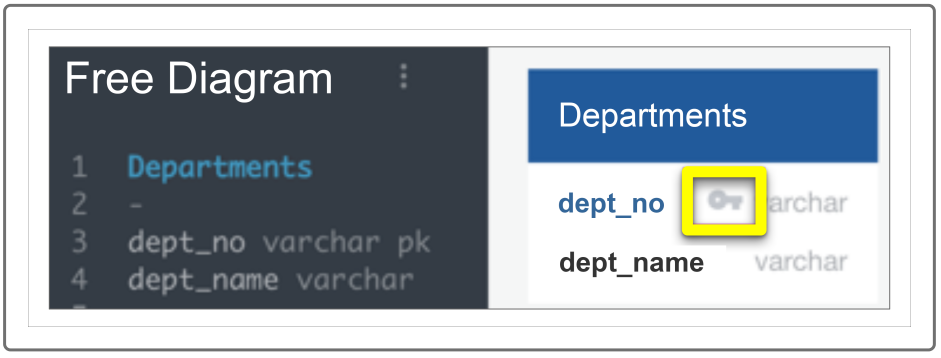
Logical Diagrams

Logical diagrams contain all of the same information that a conceptual diagram does, but the table is updated to include data types and primary keys.

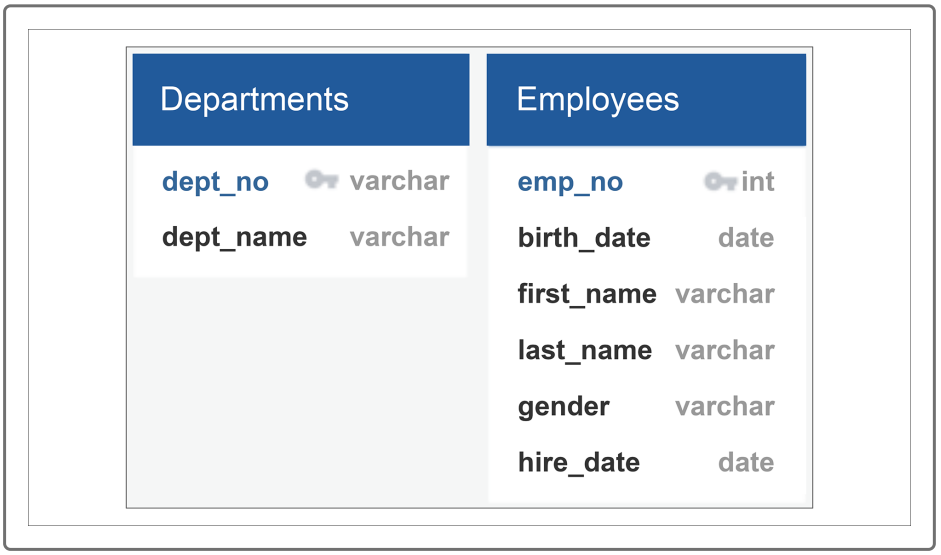
Returning to the Quick DBD webpage, let's update our schema. Because we already took an initial look at the worksheet, we have already identified the primary key and know what type of data we're working with. Using the following syntax, update our Departments schema:

- Add "varchar pk" to dept_no.
- Add "varchar" to dept_name.

We use **varchar (https://en.wikipedia.org/wiki/Varchar)** in these columns because the fields contain characters of varying length. Adding "pk" in the schema next to column indicates that column as a primary key. The table updated to reflect the changes in the text editor. A key symbol appears next to the dept_no line, indicating that it is the table's primary key, and varchar is added to indicate its type.



Update the second table "Employees" the same way, and add "date" as the data type for the dates.



The relationship diagrams are beginning to really take shape. Next, add foreign keys and physically map the relationships between them.

SKILL DRILL

Transition the remaining schema from conceptual to logical for the remaining tables and rearrange them to all fit on the screen.

Physical Diagrams

Physical diagrams portray the physical relationship, or how the data is connected, between each table. There are several different relationships available to keep in mind when making these connections, as shown below:

```
-      - one TO one
-<     - one TO many
>-     - many TO one
>-<    - many TO many
-0     - one TO zero or one
0-     - zero or one TO one
0-0    - zero or one TO zero or one
-0<    - one TO zero or many
>0-    - zero or many TO one
```

The relationships are fairly self-explanatory, too. For example, a **one-to-one** relationship means that one row, or record, in a table can be referenced to one other record in another table. For example, an employee will have a single employee number.

A **one-to-many** relationship, on the other hand, means that a row (entity) is referenced in two or more rows in another table within the database. For instance, an employee's number may be referenced in several other locations in a table that records employee advancement.

NOTE

> For more information about entity relationships, see the **Entity Relationships Tutorial (https://www.entityframeworktutorial.net/entity-relationships.aspx)**. There are more relationships than listed in the table provided by Quick DBD.

Practicing Using Entity Relationship Diagrams

One benefit to building out each form of the diagram (conceptual, logical, and physical) is that it gives us more exposure to the data and its layout. Working with it often leads to more familiarity, which in turn helps with creating a map of the relationships. Even with the exposure and familiarity, it can still be confusing, so let's keep practicing.

Take a look at the Managers table in our text editor. We already know that the dept_no is also the primary key for the Departments table, and because it's referencing another table, the dept_no in the Managers table

is a foreign key as well as a primary key. Add that designation to the schema by adding "fk" right after "pk" in that schema.

Now that we have the foreign key identified, we can create a one-to-one relationship between the Managers and Departments tables.

To add the relationship between tables, place a hyphen (-) after "fk," then a reference to the other table. The reference syntax is "tablename.column_name." To reference the Departments table, the line would be updated to look like this:

```
Managers
-
dept_no varchar pk fk - Departments.dept_no
emp_no int
from_date date
to_date date
```

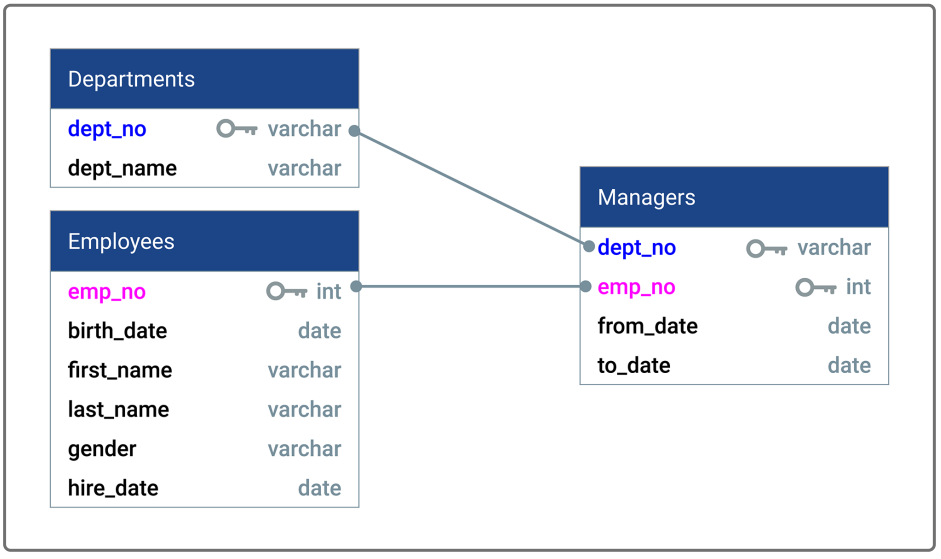
There aren't any additional updates needed for the Departments table to make this connection. Quick DBD automatically maps the connection for us.

If you rearrange the tables, the connector line will reposition with the tables. This way, after all of the connections are mapped, we can rearrange the tables until the final flowchart is smooth and uncluttered.

Let's make another connection. The Managers table has another foreign key: the emp_no. The employee number is referenced in the Employees table as well, so we can designate this line as a foreign key. Next, we'll add the one-to-one relationship, and then reference the Employees table on the same line.

```
15   Managers
16   -
17   dept_no varchar pk fk - Departments.dept_no
18   emp_no int pk fk - Employees.emp_no
19   from_date date
20   to_date date
```

Once again, Quick DBD has picked up the relationship and drawn a connection without additional updates to other table schema.



Now test your skills with the following Skill Drill.

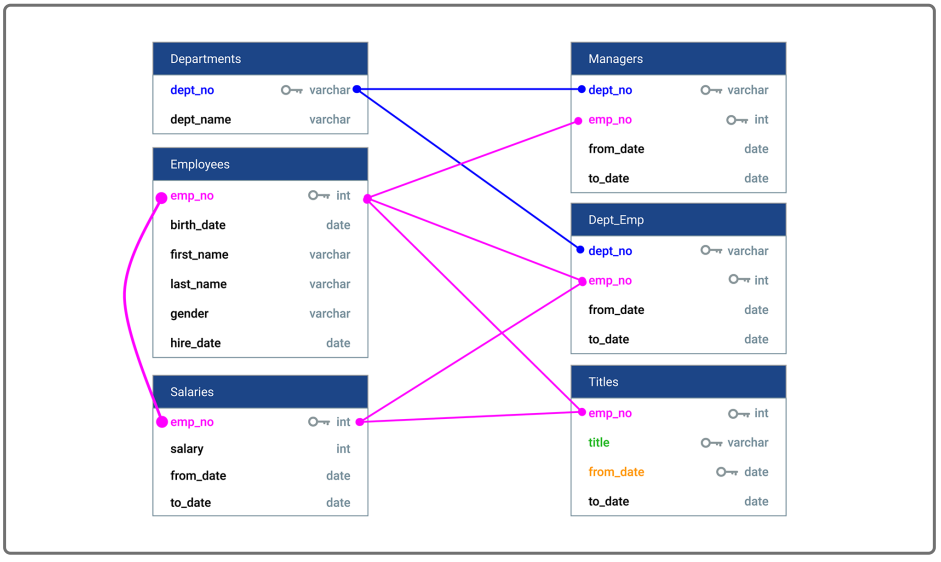
SKILL DRILL

Complete the following relationships:

1. In the Dept_Emp schema, create a one-to-many relationship to both of the Employees and Departments tables.

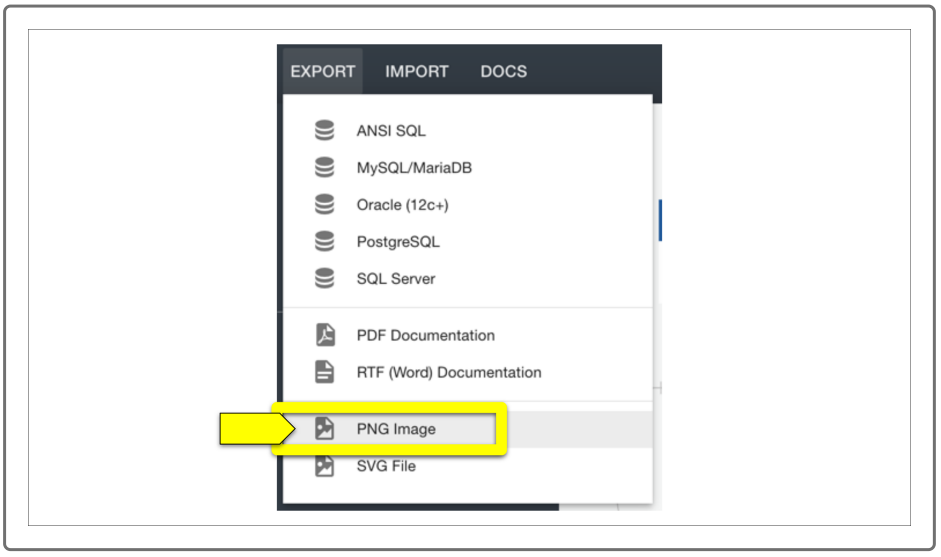
2. In the Titles schema, create a one-to-many relationship to the Employees table.
3. In the Salaries schema, create a one-to-one relationship to the Employees table.

The flowchart below shows many possible connections; your final diagram may not match exactly.



Now that the database roadmap is complete, you'll need to save your diagram as an image and schema as a text file.

To save your flowchart, click the "Export" button at the top of the Quick DBD window, then click "PNG Image" to save.



The image will be automatically saved in your Downloads folder as `QuickDBD-export.png`. Move your image to your Pewlett-Hackard-Analysis folder and rename it `EmployeeDB.png`.

Now we have a complete entity relationship diagram to reference as we help Bobby work through the analysis.

Next, we need to save the text we've written to create the schema. The free version of Quick DBD doesn't permit saving online without paying for that space. Instead, we'll save our text locally. By keeping a local copy of the schema, we can access it whenever we want. If we need to update a table, we can open the saved schema instead of having to recreate it from scratch.

Because we've already saved a PNG, we can use the image to reference our ERD from now on; feel free to bookmark and close the website before moving on.

Nice work! Now you're ready to create a database.

ADD/COMMIT/PUSH

With these new files saved in our repo folder, we can add them to our repo with `git add`, and then push them up with `git push`.