IT510 * Module 2 Reading

Please read through this PDF, as it will introduce you to some of the requirements for the Module and help guide you in completing the Assessment. The last few pages should especially be helpful if you are a career-changer.

Part 1: Organizing Data

Data Flow Diagrams

DFDs are important in order to plan a system in general. These also help to identify what your database needs as the system's purpose becomes clearer. The textbook reading assigned to you in this module provides much insight into the Gane Sarson style DFD. Note that entities are numbered in a general diagonal top-left to bottom-right direction. Similarly, data stores (D1, D2, etc.) are numbered diagonally. Additional important information:

- **The "context level" diagram** (example: top part of Figure **7.3**) maps out major ideas. It does *not* have any data stores.
  - Analogy: Imagine you have sketched out a house plan by hand with pencil and paper, with mostly blocks for rooms and not much detail.

- **The level "0" diagram** (Best example: Figure **7.6**) includes the same major ideas but adds more detail and data stores. This is the kind of DFD you will develop for the Assessment.
  - Analogy: Your house sketch gets turned into a real blueprint, with decent detail.

  Special Level 0 notes:
  - **Entities** are people or sometimes places or things (nouns, such as Buyer and Seller). Entities are placed in squares and can be shown more than once if it makes the diagram easier to read.
  - **Processes** are actions (usually verb-adjective-noun, such as Purchase Fundraiser Item) and shown in a rounded vertical rectangle with the numbering at the top. These are numbered 1.0, 2.0, etc. from top left to bottom right.
  - **Data stores** are descriptions of data you expect might be stored in the database itself (such as Inventory). These are numbered D1, D2, etc. from top left to bottom right. Do not connect one data store to another.

- **"Child" diagrams** are pieces of level 0 diagrams that have yet more detail added.
  - Analogy: Just one room in the house gets its own sketch, in which you show furniture and more detail than in the blueprint.

The Relational Database and Developing the ERD

Read the information below, even if you have completed the textbook reading and/or have previous ERD knowledge.

When determining the structure of a relational database, you must first identify **entities**, which are typically people, things, or events for which you can further identify **attributes**. In the examples shared below, the entity Book is considered as part of a small library database. Note that entities are named with single nouns (Book, not Books).

Entity:	**Book**	
Attributes:	ISBN	Publisher
	Title	PublicationDate
	Genre	Edition
	AuthorLastName	Price
	AuthorFirstName	Binding

The **primary key** (PK) is a unique attribute, which in this case would be the ISBN (no other book can have that number). Every entity *must* have a primary key, and it *cannot* be left blank ("null") when data is input into the database. You can assign an automatically generated ID for the primary key where other known attributes are not necessarily unique — for example, if the library also lent framed posters, each item could be assigned an auto-generated ID.

Once you have identified attributes and the primary key, the next step is to consider the **data type**. A chart of data types is found in the textbook in **Figure 8.7** (this will be needed for the Assessment). Since the ISBN might have an "x" in it, you cannot declare this attribute to be of integer data type; it will need to be varchar (variable characters). A specific length is important to declare so that the database does not bloat from poorly input content. A table showing much of what was just explained is below:

Entity: Book

Attribute	Key	Data Type	Length	Required?
ISBN	PK	VarChar	13	Yes
Title		VarChar	50	Yes
Genre		VarChar	15	Yes
AuthorLastName		VarChar	20	Yes
AuthorFirstName		VarChar	20	Yes
PublicationDate		Datetime	4	Yes
Edition		Int	2	No
Publisher		VarChar	20	Yes
Binding		VarChar	9	Yes
Price		Currency	7	No

Notice that the attribute names do *not* have spaces in them; this is good practice since coding with SQL will not permit spaces and requires consistent case. It is also important to be consistent with case — here, Pascal **case** was used (where every word starts with a capital letter). Your options with an example for each is below. Do not use all-caps, hyphens, or periods.

• All small letters	authorlastname
• Pascal	AuthorLastName
• Camel	authorLastName
• Underscore	author_last_name

A database will be made up of many tables, because putting everything in one table would be very difficult to manage. The reading in Chapter 13 will explain more about **normalization**, which is the attempt to rid a database of as many redundancies as possible. (Having two places where addresses are input is an example of a redundancy; there is real potential for error if you update the address in one table and forget to update elsewhere.)

Each table connects to at least one other table so that data can be properly combined for **queries** and reports. Other entities in a small library database might be Patron, Loans, and so on. To connect these entities' tables, a primary key from one table is placed in another table to serve as a **foreign key**. The Loan entity will need to have the ISBN from the Book entity inserted as its foreign key.

The last main concept to understand is **cardinality**: the number of potential relationships between entities. These are often shown using **crow's feet** notation (see Figure **13.3** in the textbook) are generally explained as:

Cardinality	Example
One-to-one	One person can only marry one person (at a time!).
One-to-many	One person can take many classes.
Many-to-many	A couple can jointly have more than one child.

"Many" simply means more than one is possible.

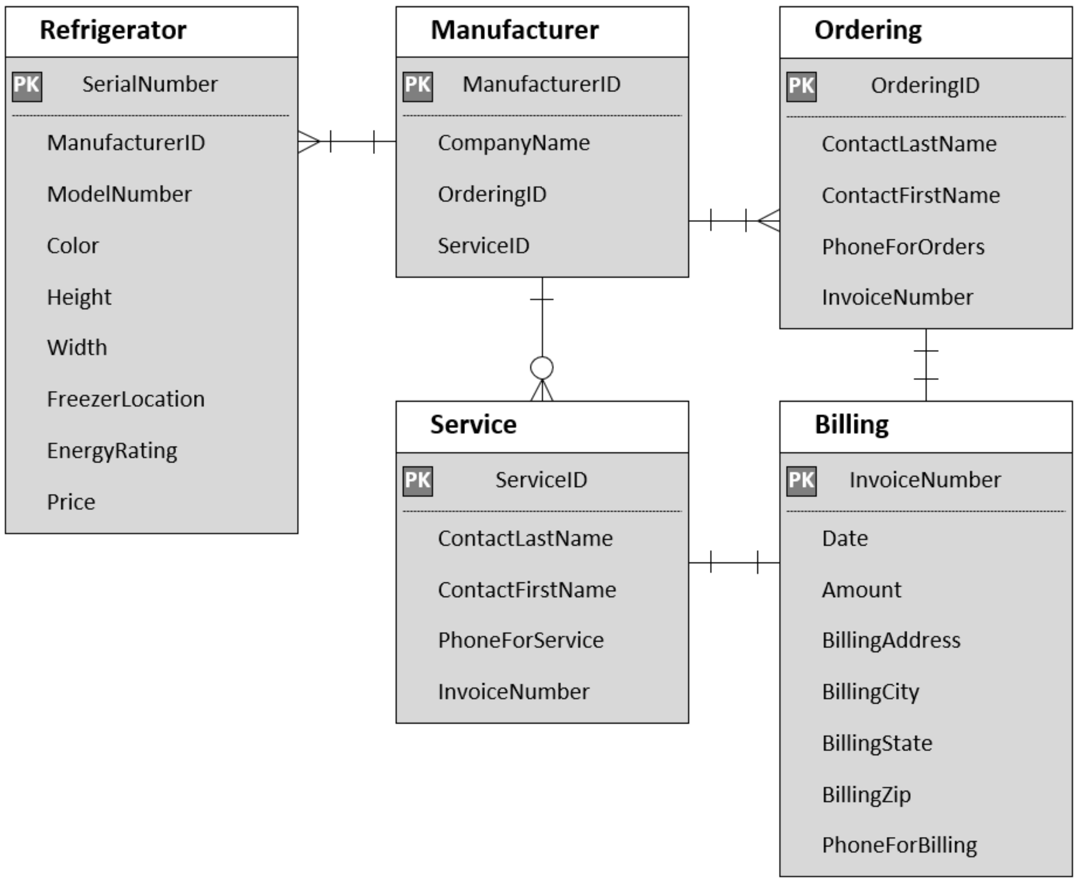
ERD Examples

Use the style of diagramming shown in these two examples for your Assessment ERD. There are other styles, including Chen notation (which looks like bubbling/mapping) and some that use diamond shapes inserted in rectangles, but you are required to use the style shown here, often called **crow's foot style**.

Example 1

In this first example, a refrigerator-selling business must work through the manufacturer for ordering stock and servicing.

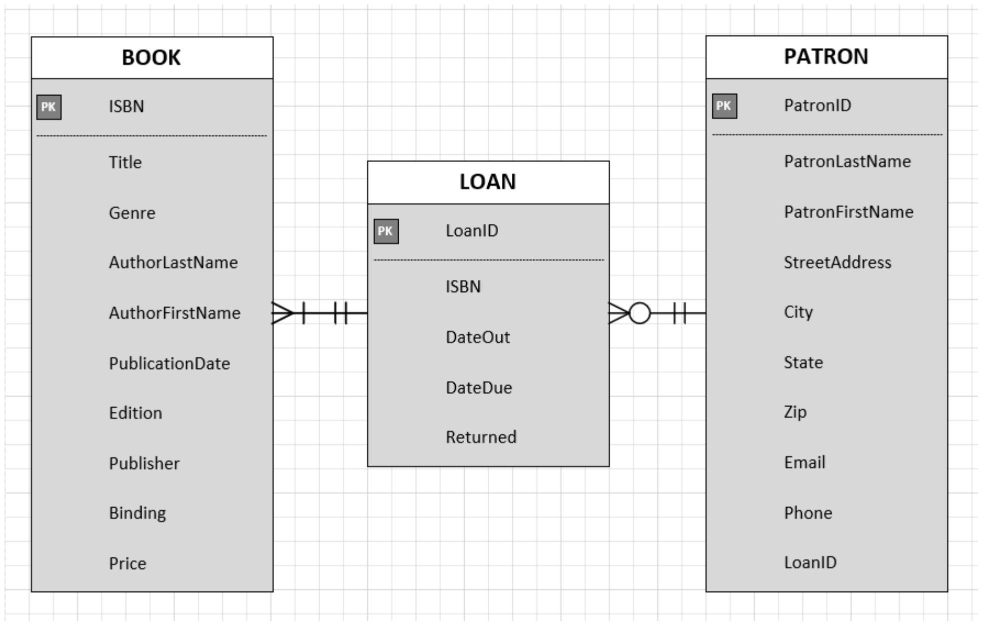
- The entities are named with single nouns.
- Attribute names do not have spaces in them and use capitalization consistently.
- The primary keys are easy to find; each one is a unique attribute for the entity. Some have to be made up by the developer (or could be auto-generated numbers).
- Look for foreign keys. One example: *ManufacturerID* is the primary key for *Manufacturer* and added to *Refrigerator* as a regular attribute so that the entities connect.



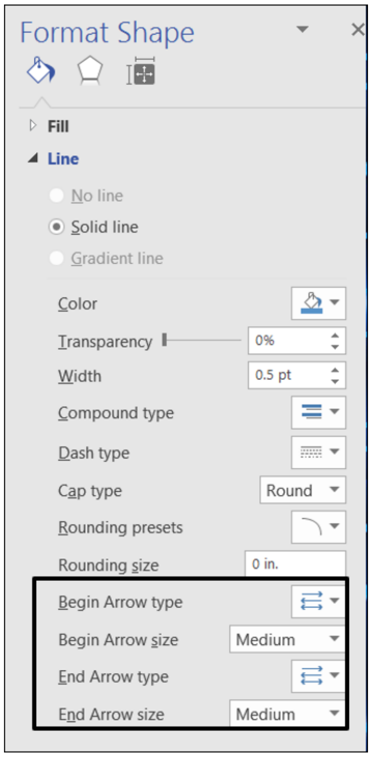
Example 2

In this bookseller's simple database, you can see the following:

- Entities are named with singular nouns.
- Every entity has an attribute designated as a primary key (PK).
  - The PK from Loan serves as a foreign key (FK) in Patron.
  - The PK from Book serves as the FK in Book.
- The crow's feet show:
  - One loan can include one or many books.
  - One patron can take out zero to many loans.



Using Visio for the Assessment



Transparency 0%
Width 0.5 pt
Cap type Round
Rounding size 0 in.
Begin Arrow size Medium
End Arrow size Medium

For both diagrams, make sure you use the template identified in the instructions; use a different Visio file for each diagram. Properly *embed* in your Word document.

For the DFD, use *only* the shapes that are shown in the textbook's Figure 7.1. If you do not see the shapes you need, use the search tool to look for "Entity," "2-part function," or "data store."

Remember that to add text that is not in a shape, go to the Home ribbon > Tools > **A** Text. Click on the "A" and then click and drag on the stage where you wish to place text. You can edit the text size and alignment using the Home ribbon tools.

For the crow's feet in your ERD, use the Relationship line to connect entities. To change the crow's feet for proper cardinality:

- Right-click the connectors > choose Format.
- In the Format Shape pane (shown to the right), click “Line” and scroll down to the Begin Arrow and End Arrow type and size menus.

Part 2: Coding Behind the Database

Structured Query Language

Part of your ebook reading delves into SQL (often pronounced "sequel"), the language of database development. In the reading, pay special attention to the SQL Commands Classifications, data types, constraints (especially NULL), differences between primary and unique keys, and the basic coding. To assist in learning about this, keep reading below.

This statement creates a database named "pets" –

```
CREATE DATABASE pets;
```

The next input creates a table especially for cats. Notice the ID (which would be the primary key), and the number of characters allowed for input (maximum 20 characters for each cat's name, for example). Check use of commas, the semicolon, and parentheses:

```
CREATE TABLE cats (
ID               INTEGER,
petname          VARCHAR(20),
breed            VARCHAR(20),
ownerlastname    VARCHAR(25),
birthdate        DATE);
```

This table now needs data input. The following code will insert values. Notice the use of single quotes for "VARCHAR" data and the location of commas, semicolons, and parentheses:

```
INSERT INTO
   cats (ID, petname, breed, ownerlastname, birthdate)
VALUES
   (1, 'Susie', 'Tabby', 'Jones', '2010-12-30'),
   (2, 'Oliver', 'Sphynx', 'Jones', '2014-10-14'),
   (3, 'Fluffy', 'Manx', 'Chenoweth', '2009-06-28'),
   (4, 'Bubba', 'Himalayan', 'Garcia', '2019-01-04');
```

Below is the method for checking the entire table. The asterisk * is a "wild card," meaning that it will pull all data:

```
SELECT * FROM cats;
```

This will result in the following output:

```
 ID | petname | breed     | ownerlastname | birthdate
----+---------+-----------+---------------+------------
 1  | Susie   | Tabby     | Jones         | 2010-12-30
 2  | Oliver  | Sphynx    | Jones         | 2014-10-14
 3  | Fluffy  | Manx      | Chenoweth     | 2009-06-28
 4  | Bubba   | Himalayan | Garcia        | 2019-01-14
(4 rows)
```

Of course, you may not want *all* data every time you query the database. If you wish to just see the list the breeds, the following could be written:

```
SELECT breed FROM cats;
```

Other commands such as DELETE and UPDATE may also be coded, and WHERE is used to limit output to specific parameters. JOIN is used to pull related data from more than one table. For example, imagine this database **pets** also contains another table named **owner**, in which owner contact information was stored. Both tables have the attribute **ownerlastname** which serves to connect the entities. The following would compile the pets' names and the owners' phone numbers from the **cats** and **owner** tables**:**

```
SELECT petname, ownerphone
FROM cats
LEFT JOIN owner
ON cats.ownerlastname = owner.ownerlastname;
```

This can be more complicated, with JOIN, RIGHT JOIN, and FULL JOIN options; you will learn more in the ebook and might like to consider spending some time with one of the many freely available online SQL tutorials to learn more.

Part 3: Additional Information

Business Letters

Business letters might be used either internally or externally; their purpose is far more formal than either memos or email. Single-space the entire letter, leaving blank lines between parts as shown below, and align all content left without indenting.

Notably, the salutation is followed by a **colon** (commas are for informal messages). The most common closures are Yours, Yours truly, Sincerely, or Regards, but others are acceptable. Do not use abbreviations.

Your Name
Your Street Address
Your City, State, ZIP
Contact information (email and/or phone)

Date in a recognizable format

Recipient
Recipient's Company if applicable
Recipient's Street Address
Recipient's City, State, ZIP

< usually two blank lines before the salutation

Dear Recipient:

The first paragraph should get to the point of the letter quickly and clearly. Single-space paragraphs, and leave a blank line between them. Do not indent the first line of the paragraphs.

Start new paragraphs for new ideas. Keep paragraphs relatively short so they can be read quickly and efficiently.

In the last paragraph, thank the recipient for their consideration or at least for reading your message. If you need a response, request it here and include contact preference.

Closure line,

Your Name
Your Title (optional unless required by assignment instructions)

Example:

Kirsten Jones
111 Green Street
Springfield, AK 20998
mjones@WYK.com

August 2, 2021

Ruben Martinez
CEO, CQTR Industries
5000 Athens Drive
Springfield, AK 20996

Dear Mr. Martinez:

I would like to inquire if your newly designed modems are available for bulk purchase at this time. I am currently serving as a systems analyst for several companies, and have three clients in particular who may be interested.

Your modems were demonstrated at the recent device conference in Springfield. These devices are quite extraordinary in range, power, and longevity.

Thank you in advance for any information you can share. I can be reached at the above email address or at 888-441-2345 weekdays.

Respectfully yours,

Kirsten Jones
Systems Analyst for WYK, Inc.

Quiz

You are encouraged to take the non-graded quiz. In this module, the quiz covers some of the SQL basics that you read about above and in the ebook reading. The questions are all multiple choice (with just one potential answer) and true-false.

Getting Help

To find the Academic Success Center, look for My Studies > Academic Success Center from your home page (where your courses are listed, not inside this classroom). There you will find a plethora of information for writing, math, science, business, and technology. You can also connect with tutors. This is a free service for Purdue Global students; if you have not investigated it prior to this term, it is a good idea to check it out and see what great help is available.

* * * * *

If You Are New to IT

This section is presented for those who are new to the field or just wish to solidify understanding of computer concepts relevant to the module or the course. It is a good idea for seasoned professionals to scan this information, too, in case there is something new to learn.

More About Databases

The ebooks listed below are in the course's Library list as "optional." Click on More Tools > Library to look for these items.

Harrington, J. (2016). *Relational database design and implementation* (4th ed.). Morgan Kaufmann. https://libauth.purdueglobal.edu/sso/skillport?context=113450

- *Chapter 4: "Entities and Relationships"*
- *Chapter 5: "The Relational Data Model"*
- *Chapter 7: "Normalization"*

Meyers, M., Jernigan, A., & Lachance, D. (2019). *CompTIA IT fundamentals+ all-in-one exam guide (exam fc0-u61)* (2nd ed.). McGraw Hill. https://libauth.purdueglobal.edu/sso/skillport?context=144895

- *Chapter 4: "Data Storage and Sharing"*
- *Chapter 14: "Understanding Databases"*

Related Concepts

Using a database also means the developer must compose **reports** for output, **forms** for input, and **queries** for asking questions (pulling data from the database). Other terms you may like to know are listed below.

- Authentication
- Constraints
- Null values
- ACID: Atomicity, Consistency, Isolation, Durability