FIT2094-FIT3171 Databases

Session 3 Tutorial Activities

CONCEPTUAL MODELLING

FIT Database Teaching Team

Complete the week 2 session 3 activities:

FIT2094-FIT3171 2021 Summer B

FIT2094-FIT3171 Introduction to Databases

Author: FIT Database Teaching Team

Important

Remember before starting any lab activity which involves working with files (here we will be generating PDF documents), first use SQL Developer to pull from the FIT GitLab server so as to ensure your local files and the FIT GitLab server files are in sync.

3.1. Conceptual Design - Demo

Your tutor will discuss and draw an ER diagram for the Department-Employee-Project case study below:

Entities:

- Employee: employee id, name, address, phone, date of birth, degrees (an employe may have more than one degree)
- Department: department number, department name
- Project: project number, project description, number of employees assigned to the project

Business rules:

- A department employs many employees, but each employee is employed by one department.
- Some employees, known as "rovers," are not assigned to any department.
- An employee may be assigned to many projects, and a project may have many employees assigned to it.
- A project must have at least one employee assigned to it.
- One of the employees manages each department, and each department is managed by one employee. An employee can only manage one department.

3.2. Using Tools to draw an Entity Relationship Diagram

There are several tools available to draw ER diagrams. Some of them are available to be used within a web browser. Some examples of these are:

- Lucidchart - this product is a browser-based diagramming tool; it is able to draw a wide range of different diagrams, including ER Diagrams, or
- any other drawing package you wish. One excellent alternative is Gliffy (https://www.gliffy.com/), or
- CASE (Computer Aided Software Engineering) tools

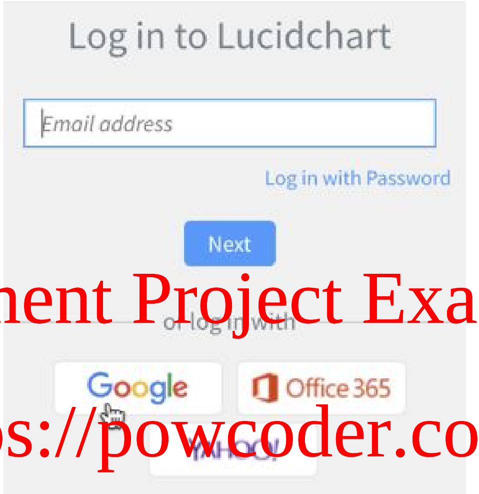
At this stage of our study, **we do not wish to use a CASE tool** - it is important that we first establish a clear understanding of Entity Relationship Modelling so we will ***only*** make use of a drawing tool. For FIT2094-FIT3171 you are **required** to use LucidChart.

3.2.1 Setting up Lucidchart

The Lucidchart Education account details and sign up are here.

Students *must create their own account* by signing up for an education account with their Monash educational email address at the URL listed above. This will result in an email being sent to your student account confirming your account details and providing a link to set your password. Alternatively, you can set your password when you first login (see the top left of your first screen) or more simply, and the *best* approach **login with your Monash Google details**.

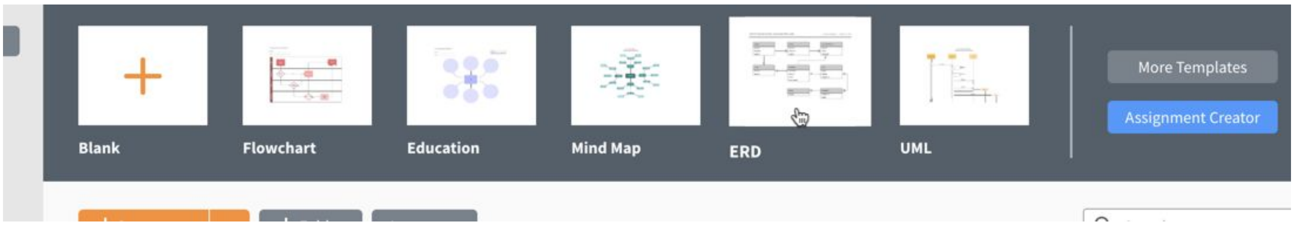
To use your Monash details to log in simply ensure you have authenticated to Monash and select "or log in with" Google on the main LucidChart page:



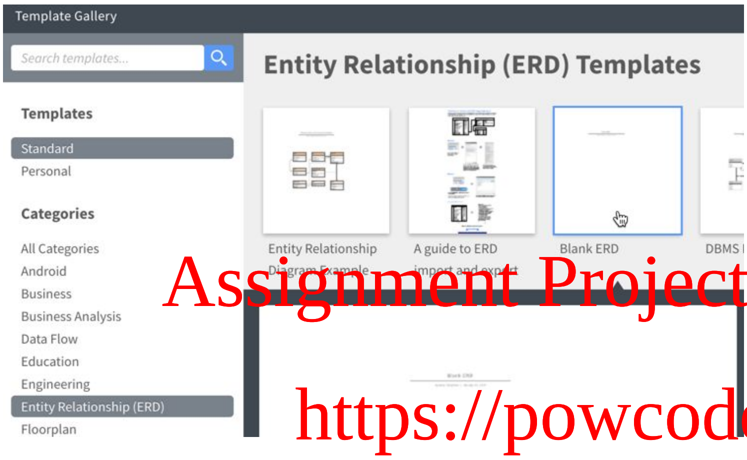
As a first step, you should look at the provided tutorials, in particular, "Entity Relationship Diagrams" (begin with "Manual ERD Creation", note that the model we are asking you to build should not include the "Type" column)

3.2.2 Creating a new LucidChart Diagram

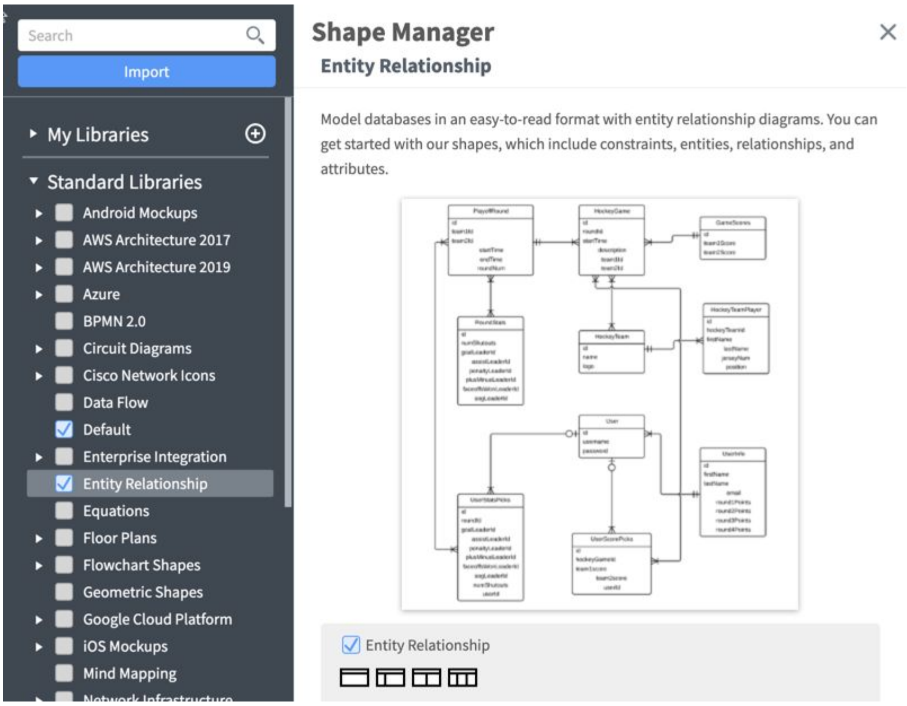
Login directly to LucidChart, select to create an "ERD":



then Blank ERD and then "Create Document".



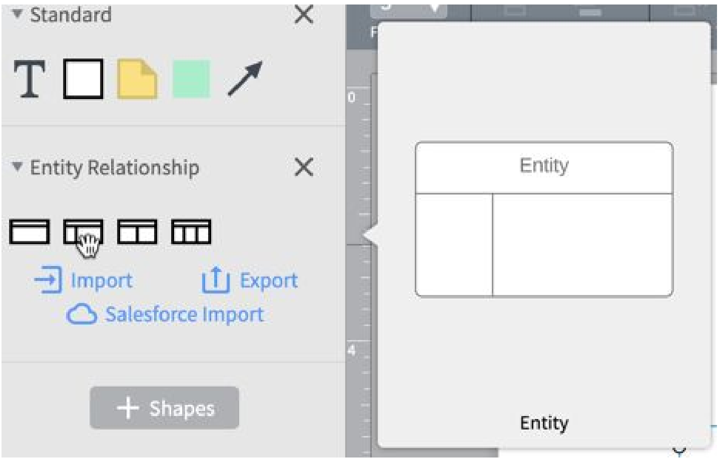
If the ERD shapes are not listed in the left panel of your diagram, add the ERD shapes by selecting "+ Shapes" (in the left panel) and then checking "Entity Relationship" in the "Standard Libraries" and then close the shape manager:



Model databases in an easy-to-read format with entity relationship diagrams. You can get started with our shapes, which include constraints, entities, relationships, and attributes.

☑ Entity Relationship

The **only** symbol we will use to represent an entity is the second symbol from the left:

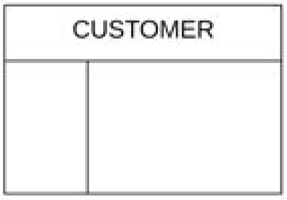


Please note for FIT2094-FIT3171 you **must not** make use of the LucidChart Import option, you must manually create your model by drawing the ERD in LucidChart. **Models which have been imported *will not be assessed*.**

3.2.3 FIT2094-FIT3171 Entity Relationship Diagram Standards

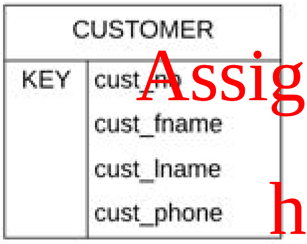
Your models must conform to the following standards. For assessed work, failure to meet these requirements will incur grade penalties:

a. Entities must be indicated via the ENTITY shape:



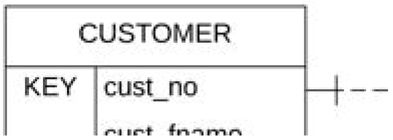
Entity names must not be pluralised

b. All key attributes must be indicated by the word KEY in the first column of the entity shape, non key attributes should have the first column left blank. Attribute names should use a common prefix to indicate the entity they belong to (see cust_ below). Attribute names must not include spaces or hyphens (-)

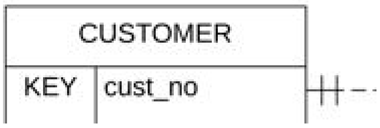


c. Relationship lines must include a verb to describe the relationship (as a general rule try to name these relationships in the 1:M direction)

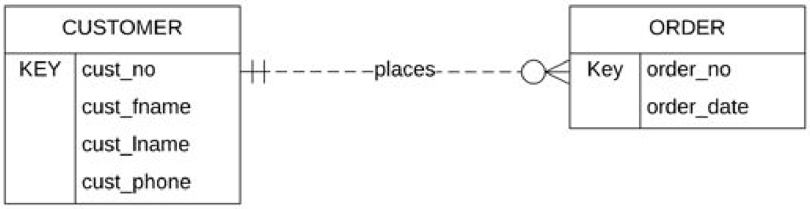
d. Minimum and maximum cardinality must be shown at each end of the relationship. Using a single line to indicate min 1, max 1, such as shown below, is not acceptable



two lines must be shown as indicated below



e. All relationship lines must correctly indicate if the relationship is identifying (a solid line) or non-identifying (a broken line).



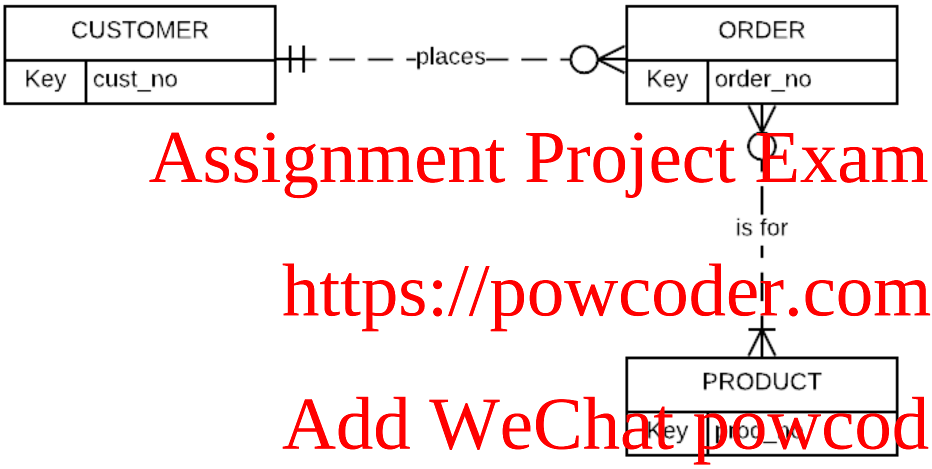
Relationship lines must be straight lines (although they may be stepped), and must not cross.

3.2.4 Drawing ER Diagram Using Lucidchart

Given a scenario represented by the following entities, where customers place orders for products:

- CUSTOMER - customer number, name, address, phone number
- ORDER - order number, order date and for each product ordered the quantity ordered and the total line price
- PRODUCT - product number, product description and product unit price

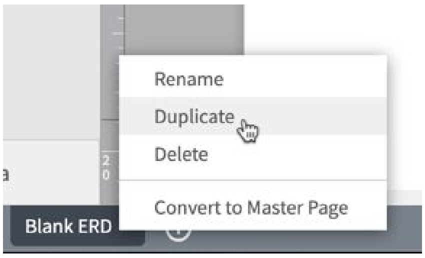
We recommend you first draw an ERD that only shows key attributes as a preliminary analysis of the scenario you are modelling. An initial ERD using Lucidchart for the customer-orders scenario would be:



Prepare the ERD shown above using LucidChart.

Where possible M:N relationships should be left on a conceptual model as they make the model simpler and easier to visualise.

The next step is to draw a full conceptual model, i.e. a conceptual model which includes all attributes to satisfy the scenario. In Lucidchart, you can easily create a copy of a diagram by selecting the bottom tab of the diagram, click the down arrow and then select "Duplicate":



Rather than overwriting your current diagram, whenever you are about to make major changes, you should use this approach to duplicate to a new tab, and then change this new tab.

When moving to a full conceptual model, it may be necessary to add attributes which need to be recorded on a relationship. In the customer-orders model above, if we wish to record the quantity of a product ordered and the total line price:

- These attributes cannot be placed in the ORDER entity (an order has potentially many products) nor
- can they be placed in the PRODUCT entity (a product has potentially many orders).

Here we need to add a new entity ORDERLINE which brings one order and one product together - this new entity converts the M:N relationship between ORDER and PRODUCT into two 1:M relationships. In naming this new entity which is called an "associative entity" (or composite entity), when a natural name is not clear, you can use the parent entity names - here this would be ORDER_PRODUCT.

As you complete the parts of this exercise, download your model as a PDF and add it to your local repo and push it to the FITGitLab server using SQL Developer.

Note, the way in which Git manages files (creates versions) means *you do not, and should not, give your downloads names such as cust_order.pdf, cust_orders2.pdf etc*. Simply use the same name all the time eg. cust_order.pdf, Git will manage the version control.

A video is available which demonstrates the way you should download and manage your files in Git. This video shows how Git maintains a history of all the files you push and how you can access older versions of your files.

Since Git maintains all versions of your work, you may use this to go back to a previous version of a particular file by removing a commit. Commits which you have added to Git may be removed in one of two ways:

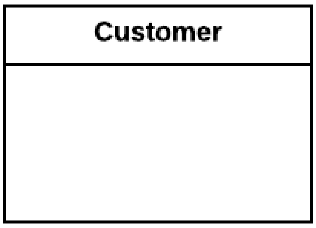
- using REVERT which adds a compensating commit to cancel out a previous commit, or
- using RESET which resets the head of the Git branch

REVERT is used in the scenario where you have pushed a commit to the remote Git Server (the FITGit Lab server) ie. to *undo a commit on the remote server*. RESET (HARD) is used to undo a commit *on your local repo before it is pushed to the remote server*. A video is available which explains each of these processes and shows some simple examples of revert and reset.

3.2.5 FIT3171 UML Diagram Standards

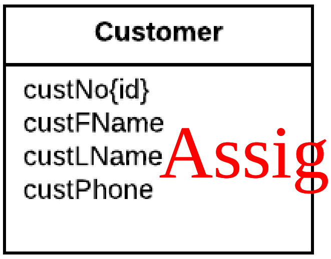
Your models must conform to the following standards. For assessed work, failure to meet these requirements will incur grade penalties:

a. Entities must be indicated via the interface shape:

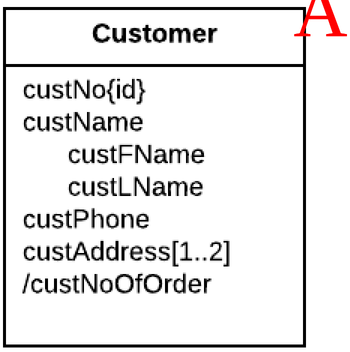


Entity names must not be pluralised

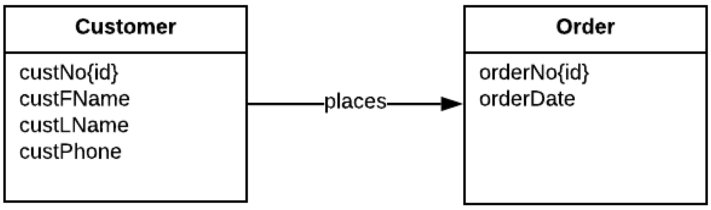
b. The identifier (key) must be indicated by the word {id}. Attribute names should use a common prefix to indicate the class they belong to. Attribute names must follow camelCase format.



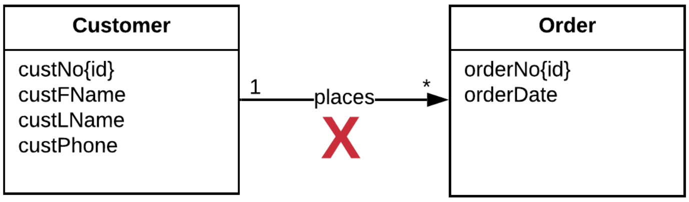
c. Composite attribute must be indicated using indentation. Multivalued attributes must be indicated using [min … max] symbol after the attribute name. Calculated attributes must be indicated using / symbol before the attribute name

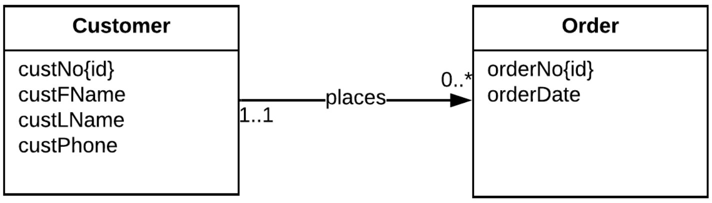


d. Relationship lines must include a verb to describe the relationship (as a general rule try to name these relationships in the 1:* direction). The arrow head for the one to many relationship must be placed at the many end. The arrow head for the many to many or one to one relationship can be placed at either end.



e. The multiciplities must show both minimum and maximum at each end of the relationship.





3.2.6 Drawing UML Diagram Using Lucidchart

Convert full customer-order conceptual model drawn in 3.2.4 into a UML diagram using Lucidchart.

As you complete the parts of this exercise, download your model as a PDF and add it to your local repo and push it to the FITGitLab server using SQL Developer.

3.3. Conceptual Modelling Exercise

Prepare an Entity Relationship Diagram (ERD), showing all attributes and the identifier of each entity for the following description of a Property Rental System:

- Properties are rented by tenants. Each tenant is assigned a unique number by the Agency. Data held about tenants include family name, given name, property rented, contact address - street, city, state, postcode & telephone number. A tenant may rent more than one property and many tenants may rent parts of the same property (eg. a large shopping complex).
- Properties are owned by owners. Each property is assigned a unique property number. The agency only recognises a single owner for any of the properties it handles. The owner, address, and value are recorded for each property. Also, the lease period and bond are recorded for each property or sub-property rented. An owner may own several properties. For each owner an owner number is assigned, the owner name is also recorded.
- Properties are subject to damage and the agency records all instances of damage to its properties - property, date, type of damage and repair cost are recorded. Repair costs are charged directly to tenants.
- Tenants pay accounts to the Agency - these consist of weekly rental payments, bond payments (for new properties) and damage bills. The date of payment, tenant, property, type of account (Rental, Bond, Damage) and amount are recorded. Each payment is assigned a payment number.

You should prepare a conceptual model based on the details supplied here, however as you model note down the areas where you consider further information is required from your client.

You may draw your initial conceptual model on a paper, but you must, on your own time outside the tute, draw the diagram using LucidChart. As you work on this model on Lucidchart, regularly download your model as a PDF document, add it to your local repo and push it to the FITGitLab server (you do not need to close your browser to do the push).

Important

After you have completed your current lab activities, at the end of each session remember to use SQL Developer to add files, commit and push any changes you have made to the FIT GitLab server.

You need to get into the habit of establishing this as a standard FIT2094-FIT3171 workflow - pull at the start of your working session, work on the activities you wish to/are able to complete during this session: add files, commit changes and then push the changes back to the FIT GitLab server.
Check regularly by logging in to the web interface of the FIT GitLab server to ensure your pushes are being received by the server correctly.