Introduction to databases 2017

FIT2094/FIT9132

FIT Database Teaching Team

Introduction to databases 2017

FIT2094/FIT9132

FIT Database Teaching Team

Generated by Alexandria (https://www.alexandriarepository.org) on February 23, 2017 at 5:09 pm AEDT

Contents

1
Tutorial 1: Introduction to SQL Developer

The exercises in this first tutorial will introduce you to the SQL Developer software that you will use to connect to the Oracle database. The main aim for these exercises are to get you familiar with the SQL Developer interface/menu/options. At the end of this exercise, you should be able to:

1. make a connection and login to your account on the Monash Oracle database
2. change your Oracle password, and
3. disconnect from Monash Oracle database in SQL Developer.

There are a number of different ways to connect to an Oracle database. In this unit, you will use Oracle software called SQL Developer.

The SQL Developer software can be downloaded from your unit's Moodle site. Information on how to install and configure SQL Developer is available on the Moodle site under Software and Documentation section. Please also note that accessing the Oracle database from a machine located outside Monash University's network will require you to connect to the Monash VPN (Virtual Private Network) service. Information on how to install the Monash VPN client and then connect to the VPN service can be found at http://intranet.monash.edu.au/esolutions/software/install-vpn.pdf

Before looking at how we connect to the Oracle database and run SQL using SQL Developer, please take a few moments and configure SQL Developer so that the software will satisfy the needs of this unit. To do this follow the steps shown below. The steps are also available in the "*SQL Developer Preference Settings*" *document* (supplied on your unit Moodle site under the "Software and Documentation" heading).

SQL Developer Preference Settings

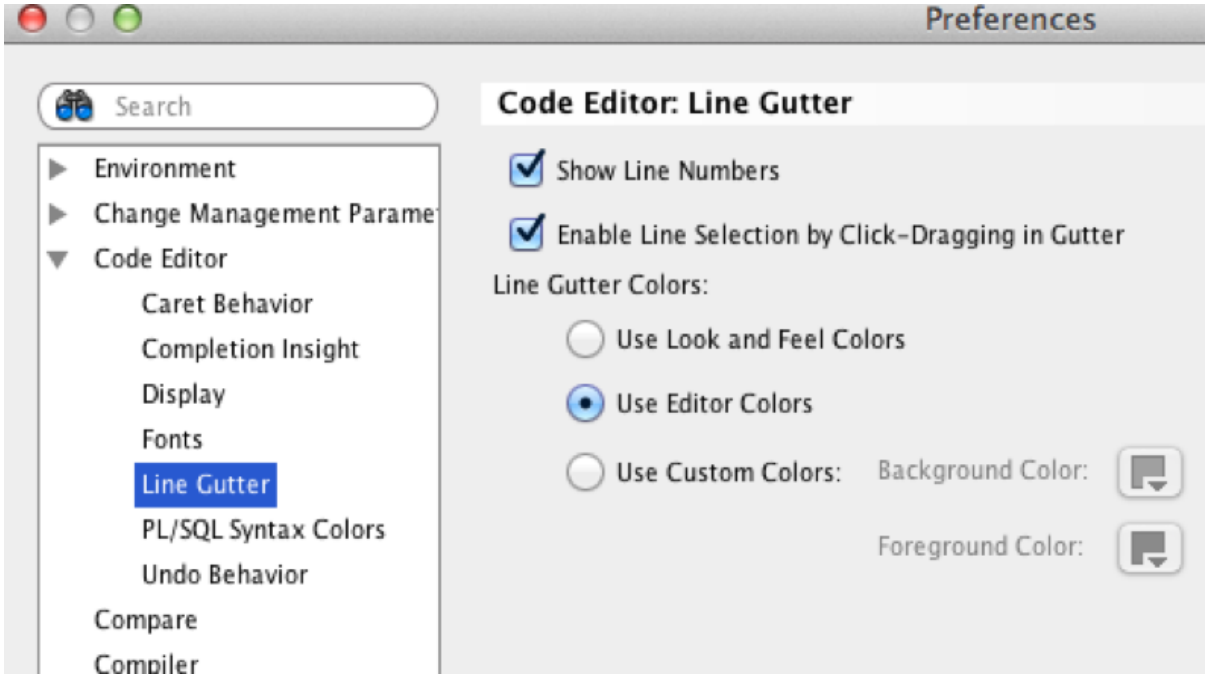
To help with your studies through the remainder of the semester you should configure SQL Developer to:

- display line numbers, and
- auto format SQL Code

This document also contains a number of other important configurations that you should implement before working extensively with SQL Developer. Under Windows the SQL Developer preferences are accessed from the Tools menu, under MacOS from the Oracle SQL Developer menu on the left.

Display Line Numbers

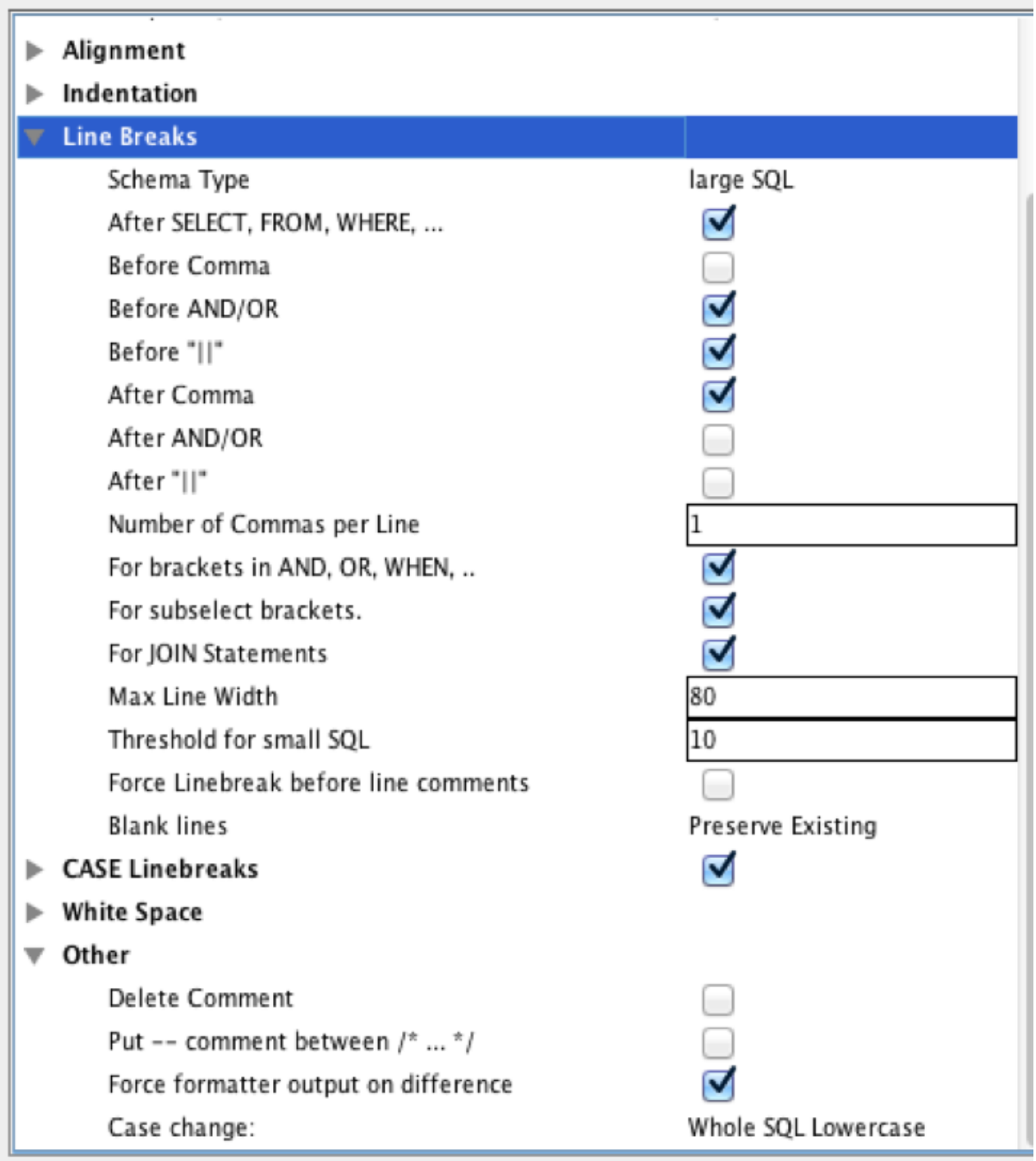
Preferences - Code Editor - Line Gutter - Show Line Numbers (Check Show Line Numbers):



SQL Code Auto Format

We now wish to configure SQL Developer so that it will reformat any SQL we enter into a well set out "pretty" format. The aim will be to simply type SQL commands without worrying about layout and then use the SQL Developer assigned format key (see below) to automatically format the code.

To do this select: Preferences - Database - SQL Formatter - Oracle Formatting - Select the SQL Profile from the drop down Profile menu and then select edit. Modify "Threshold for small SQL, under Line Breaks, to a small value such as 10:

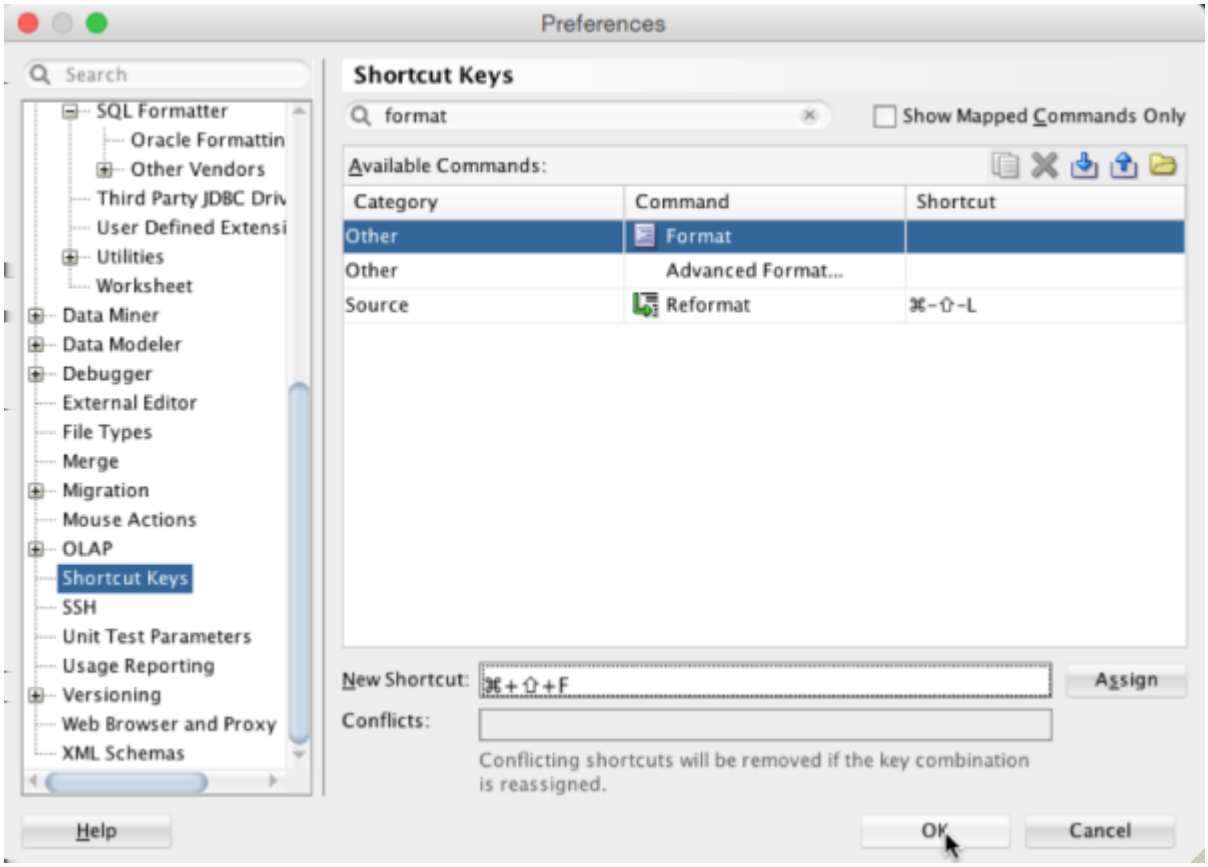


It is suggested that you also:

- set the "Max Line Width" to some reasonable value eg. 80 or 100,
- select (tick) the items selected above, and
- set "Blank lines" to "Preserve Existing"

Depending on your personal preferences you might like to set "Case Change" under "Other" to "Whole SQL Lowercase" or leave it as the default of "Keywords Uppercase". Select OK

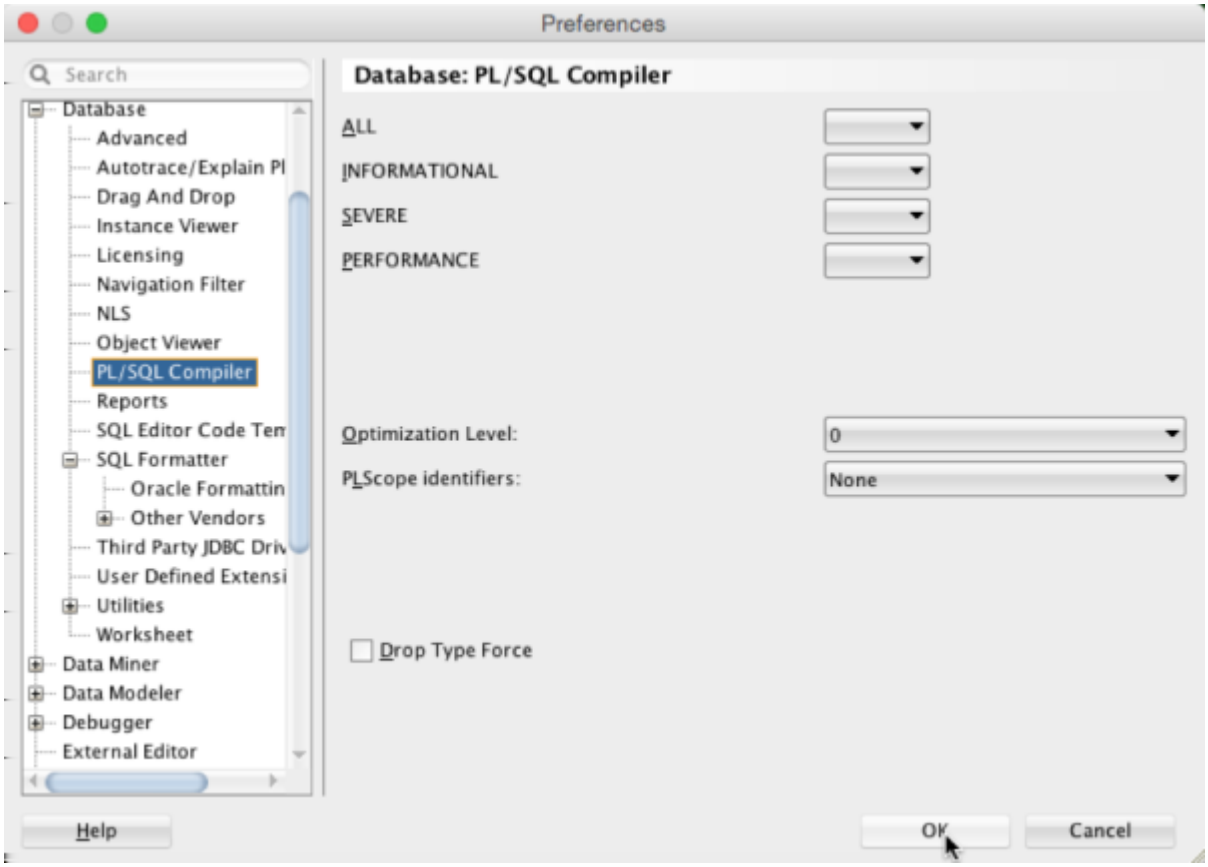
To automate formatting we now need to assign a format key combination - select Preferences - Short Cut Keys and set a shortcut key for "Format". Your installation may already have a key set for format (eg. on Windows it is Ctrl+F7) - if you are happy with that key you do not need to change anything. Below shows setting a shortcut key (command+shift+F) for a mac:



Then you can enter SQL on a single line, or multiple lines, and select your "Format" shortcut key to "pretty" format the query.

PL/SQL Scope Identifiers

Please ensure you set the PL/Scope identifiers, under Database, to "None":

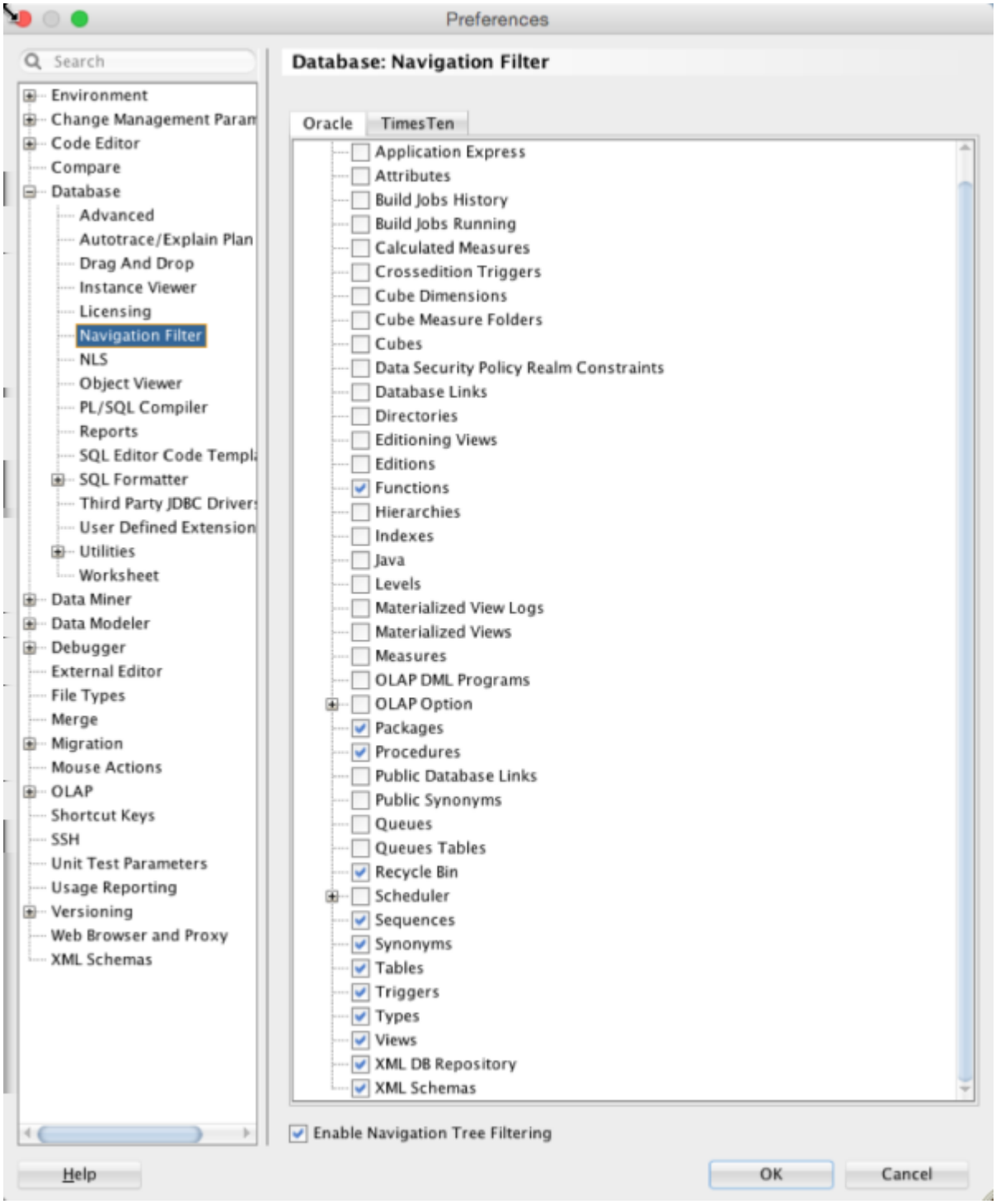


Navigation Filter (Optional)

SQL Developer displays a large, and often confusing array of items under its connection tree, these can be limited by applying a navigation filter (Database - Navigation Filter).

Select "Enable Navigation Tree Filtering"

Select to show nothing on the "Times Ten" tab. The suggested items for the Oracle tab are:



Copying your configuration to a new PC/Laptop

The configurations you have completed can be transferred to a different computer by manually copying the file product-preferences.xml. This process is also useful in the Monash on-campus labs if your PC has been configured to reset the configuration with each login - simply save the preferences file to your mapped desktop folder (see the section below 'Working in the On Campus Labs" on how to create this mapping).

Simply copy the file product-preferences.xml from the configured version to the new version where you want the same settings:

On windows the files is located in the folder:

C:\Users\yourusername\AppData\Roaming\SQL Developer\systemx.x.x.x\o.sqldeveloper.x.x.x.x\product-preferences.xml

for the current SQL Developer version and a user lsmi1 this would be:

C:\Users\lsmi1\AppData\Roaming\SQL Developer\system4.1.5.21.78\o.sqldeveloper.12.2.0.21.78\product-preferences.xml

If the AppData folder does not appear under C:\Users\yourusername you will need to turn show hidden files and folders on (https://support.microsoft.com/en-au/help/14201/windows-show-hidden-files).

On the mac the files is located in the folder:

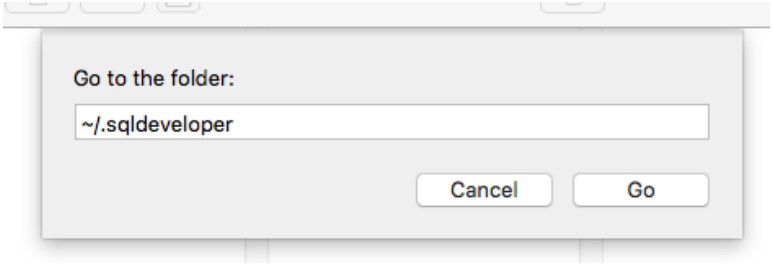
/Users/yourusername/.sqldeveloper/systemx.x.x.x/o.sqldeveloper.x.x.x.x/product-preferences.xml

for the current SQL Developer version and a user lsmi1 this would be:

/Users/lsmi1/.sqldeveloper/system4.1.5.21.78/o.sqldeveloper.12.2.0.21.78/product-preferences.xml

You are strongly recommended to make a copy of this file, once you are happy with your settings, and keep it in a safe place.

To open the /Users/yourusername/.sqldeveloper folder on your mac, in Finder select from the Finder menu Go - Go to Folder and enter the path ~/.sqldeveloper :



SQL Developer Language

If your laptop or computer is setup to use a non-English language (eg. Chinese), when SQL Developer installs it will make use of this system language as its default language.

For your Monash study you ***must* modify** this so that the language being used (and displayed) by SQL Developer is English. To achieve this, edit the sqldeveloper.conf file - it's normal location is:

MS Windows:

C:\Program Files\sqldeveloper\sqldeveloper\bin\sqldeveloper.conf

OSX:

/Applications/SQLDeveloper.app/Contents/Resources/sqldeveloper/sqldeveloper/bin/sqldeveloper.conf

and add the line:

AddVMOption -Duser.language=en

Note that you ***must carefully type (case is important) this line into the sqldeveloper.conf file yourself***, do not copy and paste it from this document. This entry must be on a line by itself and left aligned (add a new line at the end).

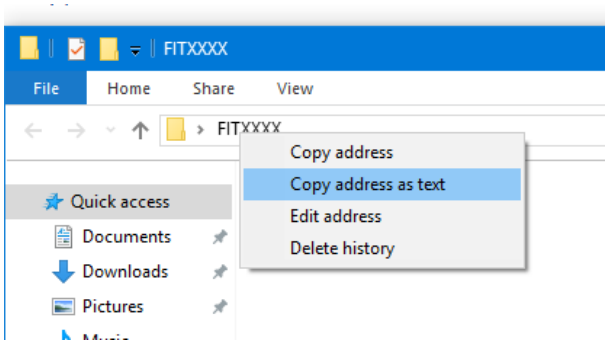
Also note this must be done with a text editor. For MS Windows please do not use Windows notepad instead use something like Notepad++ (http://notepad-plus-plus.org/)

Working in the On-Campus Labs

The technique which the University uses in the on-campus Windows 10 labs to allow your files and desktop to be portable between different machines causes SQL Developer considerable problems in saving files.

To save files (.sql, .dmd, etc) from SQL Developer on the Monash Windows 10 on-campus labs please use the following procedure.

First create a folder representing **your unit** on your Windows desktop (eg a folder called FITXXXX where XXXX is your unit code) - within this folder you can create sub folders to represent weeks of work, various tasks etc. Using the Windows file manager, open this folder, right click the folder name in the top address text box and select "Copy address as text":

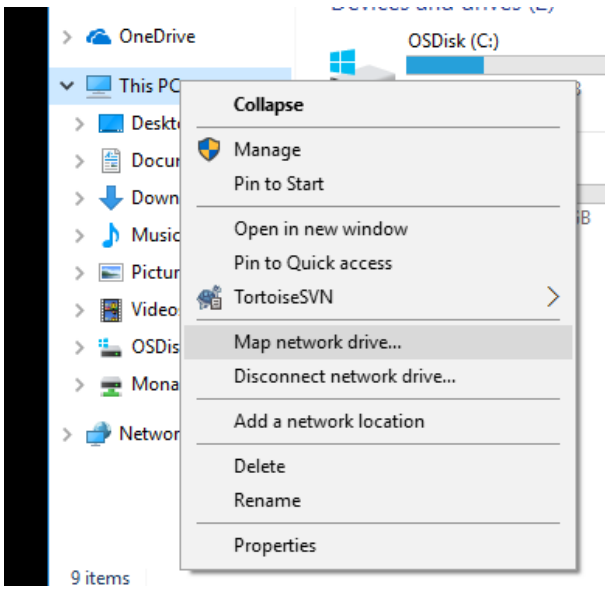


This will result in an address of the form:

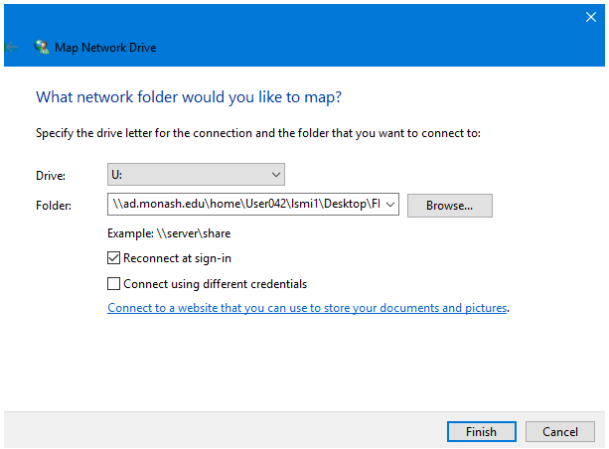
`\\ad.monash.edu\home\User042\lsmi1\Desktop\FITXXXX`

(where XXXX is the unit code you entered).

You should then use this path to map a drive in Windows (use any drive letter you wish). First right click This PC and select "Map Network Drive":

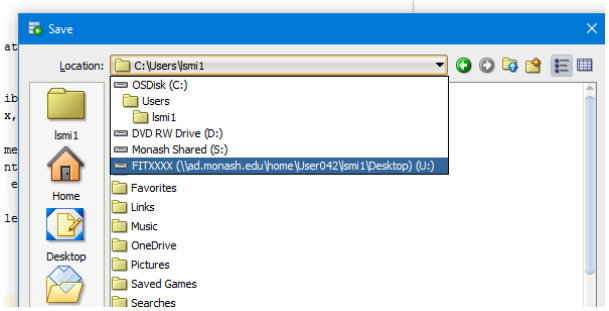


Then enter the path that you copied above into the Folder entry:



Be sure to select "Reconnect at sign-in", before clicking on "Finish"

You will then be able to load and save files in the on-campus labs from your mapped drive eg. U: and access this drive from *any* University Windows Lab PC that you login to. To reach the drive you may need to pull down the SQL Developer folders list:

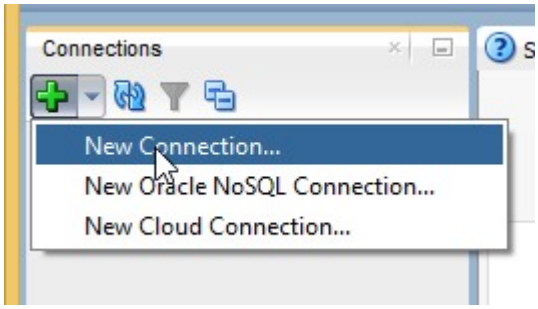


1.1
Tutorial 1: Connecting to Oracle database using SQL Developer

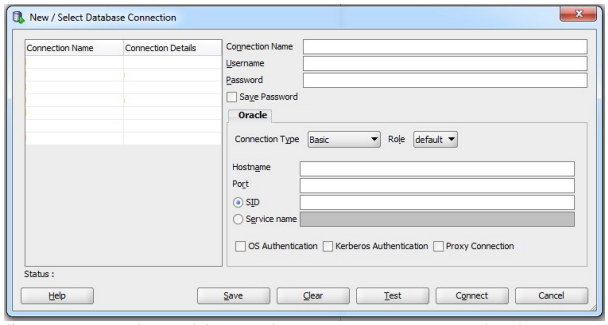
In the next few sections, you will learn how to use your newly configured SQL Developer software to access an Oracle database.

1. Adding a new connection

After running SQL Developer, right click on the Connections icon in the left panel, as shown below and select the New Connection option.



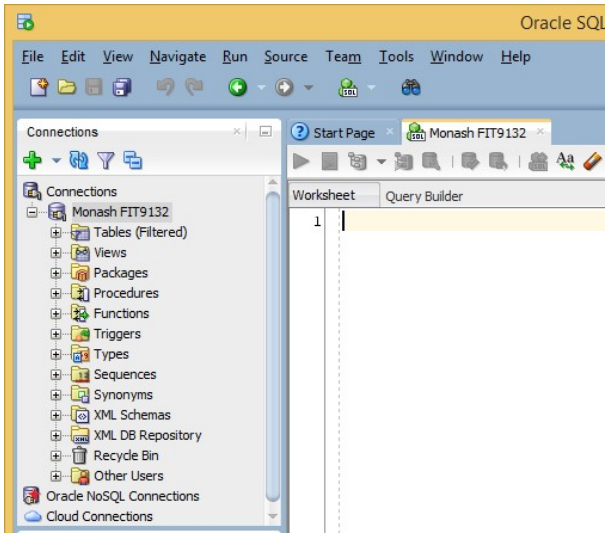
Now you will see the New / Select Database Connection window, as shown below.



(https://www.alexandriarepository.org/wp-content/uploads/20160212080414/New_connection_b1.jpg)

The connection details that you need to connect to the Oracle database through SQL Developer will be provided by your tutor/lecturer. As well as setting up this connection, it is very important that you configure SQL Developer correctly (full details are available in the Unit Information of this unit's Moodle site).

After connection you will see:



(https://www.alexandriarepository.org/wp-content/uploads/20160212080443/New_connection_c1.jpg)

SQL Statements can be entered in the right hand panel, labelled "Worksheet".

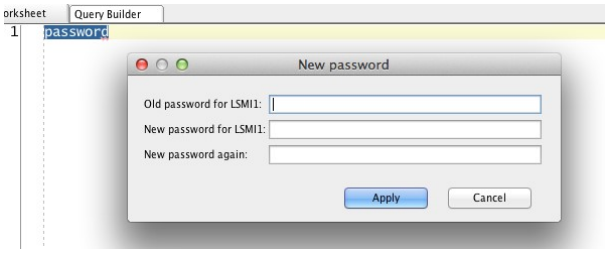
2. Changing your password

After logging in for the first time you should change your password from the one supplied by your tutor. **DO NOT** set your Oracle password the same as your standard authcate password.

Oracle has several **important** limits on the password you set:

- The password is not case sensitive
- May be 1 - 30 characters in length
- Must begin with an alphabetic character
- Can contain only alphanumeric characters and the underscore (_) or dollar sign ($)

You should set your password in SQL DEVELOPER using the PASSWORD command (your password is hidden). Type the word password in your worksheet and click on the Execute Statement button (the green arrow) in the toolbar to run the command:



(https://www.alexandriarepository.org/wp-content/uploads/20160212080954/Change_password_b1.jpg)

After changing your password for the first time please logout and then re log back in to check that your password change has been successful.

If at any stage you find that you are unable to login to Oracle due to password problems *please email your tutor to have your password reset*.

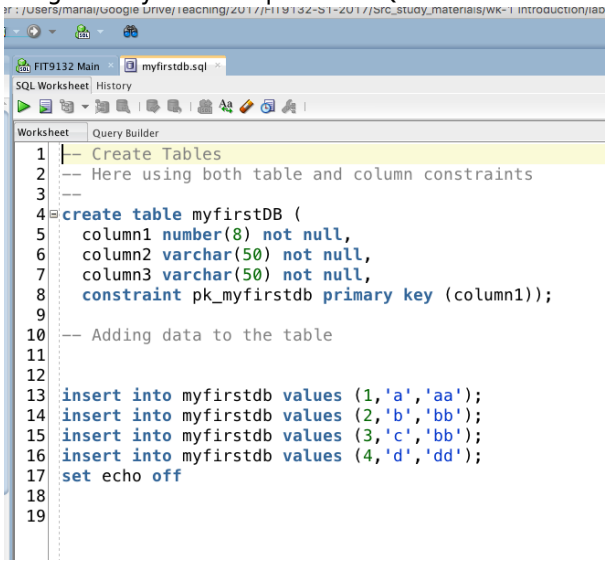
1.2 Tutorial 1: Using SQL Developer GUI to manage data

There are two main approaches to interact with Oracle database using SQL Developer, through the Graphical User Interface (GUI) and SQL Worksheet. In this module, you will learn to use the SQL worksheet to create the database and to use the GUI to add, update and delete data from the database.

1. Opening an SQL file in the SQL Worksheet.

We will use SQL worksheet to create the database. Follow the these steps>

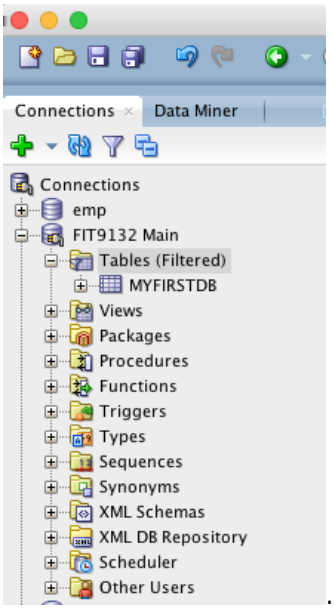
1. Download the myfirstDB.sql from tutorial 1 resources in Moodle.
2. Open the SQL Developer in your machine.
3. Open the connection to an Oracle server.
4. Drag the myfirstDB.sql to the SQL Worksheet area.



5. Run the SQL script by pressing the "run script" icon. 

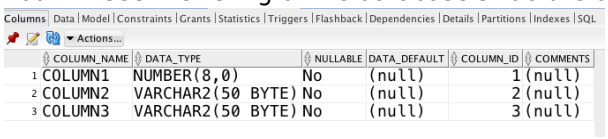
2. Viewing table structure and data inside a table.

1. To view the table using the graphical user interface, expand the Table option in the

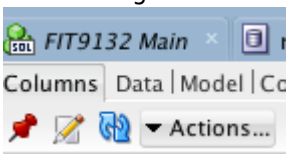
Connection tab and find "myfirstdb" from the list. 

. Double click on the "myfirstdb".

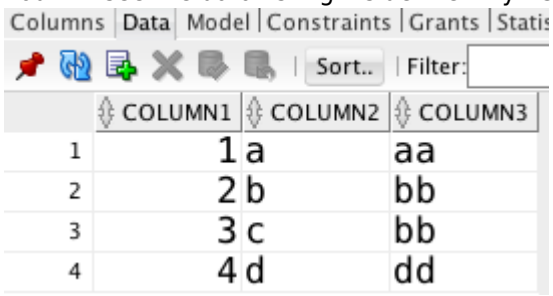
2. You will see the listing of the database structure of the "myfirstdb" database.



	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	COLUMN1	NUMBER(8,0)	No	(null)	1	(null)
2	COLUMN2	VARCHAR2(50 BYTE)	No	(null)	2	(null)
3	COLUMN3	VARCHAR2(50 BYTE)	No	(null)	3	(null)

3. To view and change the data inside the myfirstdb, click on the "Data" tab on the right hand panel. 

4. You will see the data listing inside the "myfirstdb".

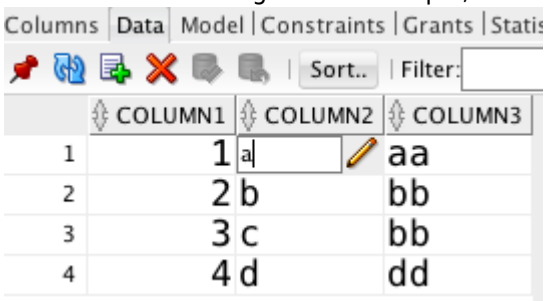


Columns | Data | Model | Constraints | Grants | Statis

Sort.. | Filter:

	COLUMN1	COLUMN2	COLUMN3
1	1	a	aa
2	2	b	bb
3	3	c	bb
4	4	d	dd

3. Updating the data inside a table.

1. Double click on the cell where the data needs to be changed. For example, let's change the content of row-1 of column2 from "a" to "z". 

Columns | Data | Model | Constraints | Grants | Statis

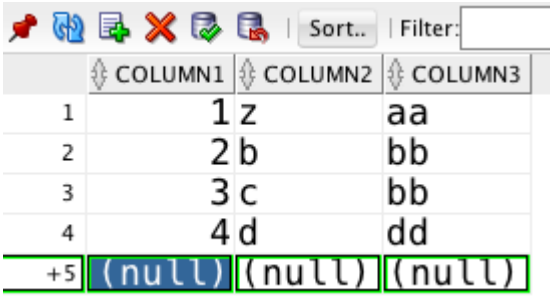
Sort.. | Filter:

	COLUMN1	COLUMN2	COLUMN3
1	1	a	aa
2	2	b	bb
3	3	c	bb
4	4	d	dd

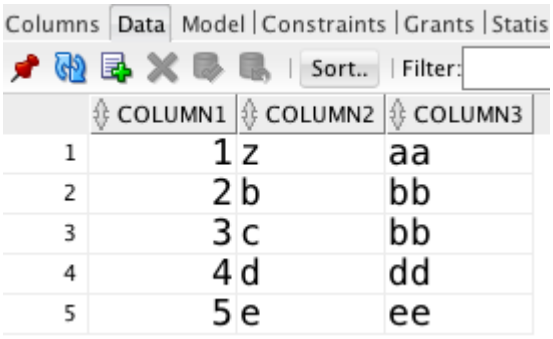
2. You will see an asterisk for the first row. To accept the changes made on the row(s) with asterisk, you will need to issue a COMMIT by pressing the "tick"  icon.
3. When database accept the COMMIT instruction you have made, the asterisk should disappear from row 1.

4. Adding a new row to the table.

1. Click on the insert new row icon  .
2. You will see a new row is being added to the table with all values listed as (null).



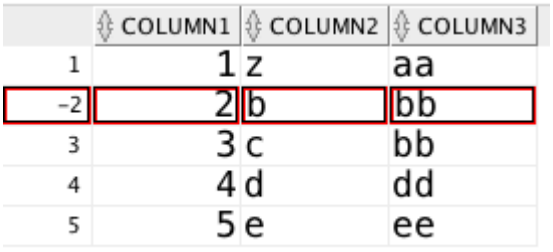
3. Replace the (null) value with 5, e and ee.
4. Click on the COMMIT icon to save the changes.



5. You will see the new being added to the table.

5. Deleting a row from the table.

1. Click on the row to be deleted, for example, choose row 2.
2. Once the row is highlighted, choose the delete icon from the menu.  .
3. You will see the row to be marked with negative sign at the front of the row number.



4. Click on the COMMIT icon to save the deletion.
5. The second row now should contain the data 3, c, bb.

1.3
Tutorial 1: Data Anomalies

A poor database design will lead to problems during the operation of the database. The following exercises will help you to identify anomalies(problems) associated with a poorly designed database.

First, you will need to create the database and to populate the data. To perform this task, you need to make sure you have completed the module Introduction to SQL Developer.

Creating database

1. Download the file student_poor.sql and student_good.sql from Moodle.
2. Open the SQL Developer software.
3. Connect to the Oracle server.
4. Open the files you have downloaded from Moodle in SQL Developer.
5. Run the student_poor.sql.
6. Run the student_good.sql.

Manipulating data inside the database

Manipulating data in a poorly designed database.

In this part of the exercise, you will use the student_poor table to illustrate the difficulties that you may encounter when using a poorly designed database. You will need to

- observe what has happened to the data in the database,
- identify problems that have occurred after the operation .

Use the list of questions to help you identifying the problems.

UPDATE

Action: Change the name of FIT9131 for student number 1111 into Foundation of Java.

Observe:

- Can you make the change?
- How many different names does FIT9131 have? What will be a potential problem with this situation?
- If you want to ensure the name of FIT9131 to be consistent, how many rows did you suppose to change in the database? Is it possible for you to easily check that all relevant rows have been updated?

DELETE

Action: Wendy Wang decided to withdraw from FIT9133. You are required to delete this enrolment.

Observe:

- Can you find out from the database the name of a unit with the code FIT9133 after the delete operation is completed?
- Can you find out from the database the student number of Wendy Wang after the delete operation is completed?

INSERT

Action: A new unit FIT5000 Data Analytics is introduced in the course. You are required to add the unit code and the unit name into the database.

Observe:

- Can you perform the insert operation? Why not?
- What additional information do you need to include to add a new unit information to the database? Is it a good idea? Why/Why not?

Manipulating data in a well-designed database.

In this part of the exercise, you will use the following tables

- student_good
- unit_good
- enrolment_good

to illustrate how the problems you have encountered in the previous tasks are overcome by a better designed database.

UPDATE

Action: Change the name of FIT9131 for student number 1111 into Foundation of Java.

Observe:

- What table do you need to change to reflect the required change?
- How many row(s) do you need to change?
- How does the new design ensure consistency of the unit name?

DELETE

Action: Wendy Wang decided to withdraw from FIT9133. You are required to delete this enrolment.

Observe:

- In what table the data need to be deleted?
- Can you find the student number of Wendy after the deletion?
- Can you find the unit name for FIT9133 after the deletion?

INSERT

Action: A new unit FIT5000 Data Analytics is introduced in the course. You are required to add the unit code and the unit name into the database.
Observe:

- In what table the data need to be added?
- Can you add the new unit information in the database without having any student enrolled into the unit?

SUMMARY

Create your own note to summarise what *update, delete* and *insert* anomalies (problems) may occur when the database is not well designed.

2
Tutorial 2: The Relational Data Model

In this weeks work we will look at the fundamental concepts on which the Relational Database model is built and complete some further practice with using SQL Developer. Remember at this stage, the practical work with SQL Developer is about developing familiarity with the software so that you can quickly and reliably create, save and execute scripts. We will return to the actual SQL syntax later in the semester.

Part A: The relational Model

1. Discuss the following terms

- Relation
- Attribute
- Domain
- Tuple
- Degree and Cardinality of a Relation

2. Consider the CUSTOMER and ORDER relations below:

CUSTOMER (CUSTOMER-ID, NAME, ADDRESS)
ORDER (ORDER-ID, DATE, CUSTOMER-ID)

Assume a single customer may have any number of orders.

1. Identify the primary key and foreign key attributes in these relations.
2. Can you think of a reason why we would not just store all the customer and order information in one relation so that we would not have to perform the join operation?

3. Choosing Primary key

1. In any relation, tuples must be unique. However, in many cases the set of all the attributes in a relation is not considered a candidate key. Why not?On the other hand, suppose we do have a relation where the set of all attributes is a candidate key. In this case, show that this set must therefore be the only candidate key and hence the primary key.
2. Consider a relation that depicts a tutorial room booking in a university. Each faculty assigns a person to handle the booking for all tutorial classes for that faculty. The person's email address is given to the university's booking system as a contact person.BOOKING(b_date, b_starttime, b_endtime, unit_code, contact_person, room_no, tutor_id)
    - Identify candidate key(s) and primary key for the relation if the following business rules are applicable:
        - More than one tutorial classes of the same unit may run at the same time (parallel sessions are possible).
        - A tutor may teach several classes of the same unit.
        - All tutorial classes are 2 hours long.
    - Identify candidate key(s) and primary key for the relation if the following business rules are applicable:
        - Tutorial classes can be either 1 hour or 2 hours long.
        - A tutor can only teach one tutorial class in a given unit.

- There are no parallel sessions of tutorial classes.

4. Relational Algebra (Adapted from Exercise 3.6 of Connolly, Begg and Strachan)

Suppose we have the following 4 relations:

HOTEL(HOTEL-NO, NAME, ADDRESS)
ROOM(ROOM-NO, HOTEL-NO, TYPE, PRICE)
BOOKING(HOTEL-NO,GUEST-NO, DATE-FROM, DATE-TO, ROOM-NO)
GUEST(GUEST-NO, NAME, ADDRESS)

Generate the relational algebra for the following queries:

1. List the names and addresses of all hotels
2. List all single rooms with a price below $50
3. List the names and addresses of all guests
4. List the price and type of all rooms at the Grosvenor Hotel
5. List all names and addresses of guests currently staying at the Grosvenor Hotel (assume that if the guest has a tuple in the BOOKING relation, then they are currently staying in the hotel)

The following question is an optional exercise.

5. There are 7 relational algebra operators, namely:

SELECTION, PROJECTION, JOIN, UNION, INTERSECTION, DIFFERENCE and CARTESIAN PRODUCT. In fact, 5 of these operators may be considered primitive operators in the sense that the others may be expressed in terms of the primitive operators. The primitive operators are:

SELECTION, PROJECTION, UNION, DIFFERENCE and CARTESIAN PRODUCT

Using the sample tables below, show how the JOIN operation can be expressed in terms of the fundamental operators by showing the process to do a natural join of customer and order.

- **CUSTOMER table:**

Cust_ID	Name
1	Green
2	Blue

- **ORDER table:**

Ord_ID	Date	Cust_ID
1	23-Feb-2009	1
2	26-Feb-2009	1
3	26-Feb-2009	2

3
Tutorial 3: Normalisation - Tutorial Activities

Normalisation Tutorial Exercises

Q1. This exercise is adapted from Connolly and Begg, Exercise 14.15 (p441)

staffNo	dentistName	patNo	patName	appointment		surgeryNo
				date	time	
S1011	Tony Smith	P100	Gillian White	12-Sep-08	10:00	S15
S1011	Tony Smith	P105	Jill Bell	12-Sep-08	12:00	S15
S1024	Helen Pearson	P108	Ian MacKay	12-Sep-08	10:00	S10
S1024	Helen Pearson	P108	Ian MacKay	14-Sep-08	14:00	S10
S1032	Robin Plevin	P105	Jill Bell	14-Sep-08	16:30	S15
S1032	Robin Plevin	P110	John Walker	15-Sep-08	18:00	S13

Given the above sample dentist/patient data, complete the following tasks:

- Choose a primary key for the relation if we take all of the columns to be attributes of a relation called DENTIST.
- Describe possible insertion, deletion and update anomalies for the DENTIST relation.
- Draw a dependency diagram for your DENTIST relation.
- Based on your PRIMARY KEY selection, is there any repeating group in the relation DENTIST? If so, remove the repeating group so that the DENTIST relation will be in 1NF.
- Continue the normalisation process until all relations are in 3NF. Draw dependency diagram(s) for each step of the normalisation.

Q2. Database Design Practical Exercise

UNITS CURRENTLY APPROVED

REPORT DATE: 9/07/2015

Unit Number	Unit Name	Unit Description	Unit Value
FIT9131	Programming Foundations	Introduction to programming	6
FIT9132	Introduction to Databases	Database Fundamentals	6
FIT9134	Computer Architecture and Operating Systems	Fundamentals of computer systems and the computing environment	6
FIT9135	Data Communications	Fundamentals of data and computer communications	6

* Unit values may be either 3, 6 or 12 points

LECTURER DETAILS **REPORT DATE:** 29/07/2015
LECTURER'S NUMBER: 10234
LECTURER'S NAME: GUISEPPE BLOGGS
LECTURER'S OFFICE No.: 169
LECTURER'S PHONE No.: 99037111
UNIT ADVISER FOR:

UNIT NUMBER	UNIT NAME
FIT9131	Programming Foundations

FIT9134 Computer Architecture and Operating Systems
* A given unit may have several advisers

* Some lecturers share offices, although each have their own phone

STUDENT DETAILS **REPORT DATE:** 29/07/2015
STUDENT No.: 12345678
STUDENT NAME: Poindexter Jones
STUDENT ADDRESS: 23 Wide Road, Caulfield, 3162
COURSE ENROLLED: MIT
MODE OF STUDY: On-Campus
MENTOR NUMBER: 10234
MENTOR NAME: Guiseppe Bloggs
ACADEMIC RECORD:

UNIT NUMBER	UNIT NAME	YEAR / SEMESTER	GRADE
FIT9131	Programming Foundations	2015/1	D
FIT9132	Introduction to Databases	2015/1	D

* Grade may have the value N, P, C, D or HD

* Mode of Study must be On-campus (O) or Distance Education (D)

In order to add a student, the lecturer who advises that student must already exist in the database. No lecturer may be deleted who advises any students which are currently in the database. If the lecturer number of a lecturer is changed, then the number would be changed for each student advised by that lecturer.

Tasks:

Prepare

- A database model for this situation as a set of third normal form relations. Normalise each form one at a time and then group together those relations which have the same PK ie. representing the same entity. At each stage show all dependency diagrams
- Express this model using a Conceptual Model (ERD), show all attributes on this ERD.

3.1
Tutorial 3: Normalisation - Review Questions

Review Questions

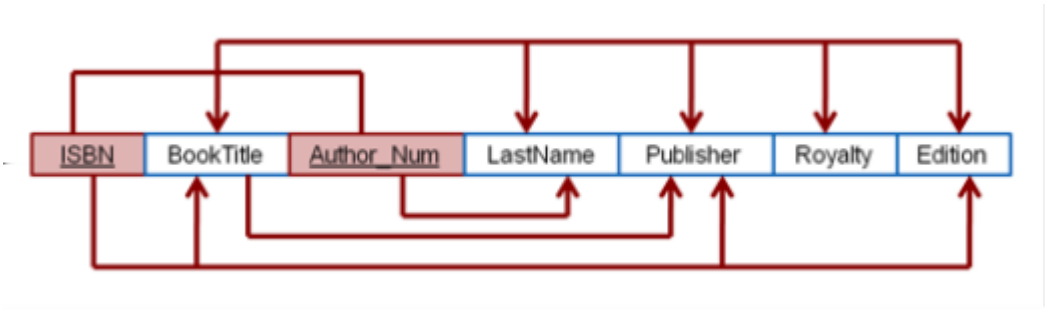
Reference:

Coronel, C., Morris, S. and Rob. P. Database Systems: Design, Implementation & Management, 9th & 10th Edition, Chapter 6, Selected Review Question and problems.

Review Questions:

7. The dependency diagram in Figure Q6.7 indicates that authors are paid royalties for each book that they write for a publisher. The amount of the royalty can vary by author, by book, and by edition of the book.

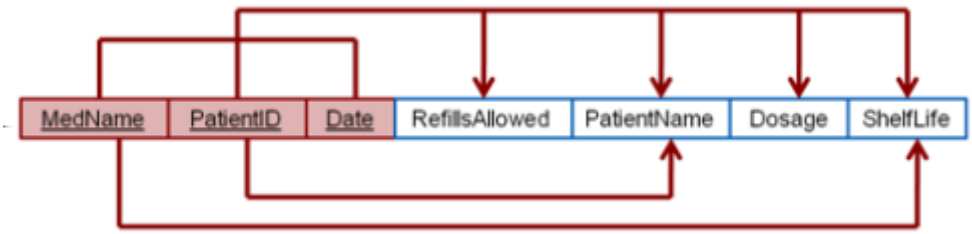
Figure Q6.7 Book royalty dependency diagram:



- Based on the dependency diagram, create a database whose tables are at least in 2NF, showing the dependency diagram for each table.
- Create a database whose tables are at least in 3NF, showing the dependency diagram for each table

8. The dependency diagram in Figure Q6.8 indicates that a patient can receive many prescriptions for one or more medicines over time. Based on the dependency diagram, create a database whose tables are in at least 2NF, showing the dependency diagram for each table.

Figure Q6.8 Prescription dependency diagram:



9. What is a partial dependency? With what normal form is it associated?
10. What three data anomalies are likely to be the result of data redundancy? How can such anomalies be eliminated?
11. Define and discuss the concept of transitive dependency.

13. Why is a table whose primary key consists of a single attribute automatically in 2NF when it is in 1NF?
14. How would you describe a condition in which one attribute is dependent on another attribute when neither attribute is part of the primary key?

Problems:

3. Using the INVOICE table structure shown in Table P6.3, do the following:

Table P6.3 Sample INVOICE Records

Attribute Name	Sample Value	Sample Value	Sample Value	Sample Value	Sample Value
INV_NUM	211347	211347	211347	211348	211349
PROD_NUM	AA-E3422QW	QD-300932X	RU-995748G	AA-E3422QW	GH-778345P
SALE_DATE	15-Jan-2010	15-Jan-2010	15-Jan-2010	15-Jan-2010	16-Jan-2010
PROD_LABEL	Rotary sander	0.25-in. drill bit	Band saw	Rotary sander	Power drill
VEND_CODE	211	211	309	211	157
VEND_NAME	NeverFail, Inc.	NeverFail, Inc.	BeGood, Inc.	NeverFail, Inc.	ToughGo, Inc.
QUANT_SOLD	1	8	1	2	1
PROD_PRICE	$49.95	$3.45	$39.99	$49.95	$87.75

- Write the relational schema (represent the table as a relation), draw its dependency diagram and identify all dependencies, including all partial and transitive dependencies. You can assume that the table does not contain repeating groups and that any invoice number may reference more than one product. (Hint: This table uses a composite primary key.)
- Remove all partial dependencies, draw the new dependency diagrams, and identify the normal forms for each table structure you created.
- Remove all transitive dependencies, and draw the new dependency diagrams. Also identify the normal forms for each table structure you created.
- Draw the Crow's Foot ERD.

4. Using the STUDENT table structure shown in Table P6.4, do the following:

Table P6.4 Sample STUDENT Records

Attribute Name	Sample Value	Sample Value	Sample Value	Sample Value	Sample Value
STU_NUM	211343	200128	199876	199876	223456
STU_LNAME	Stephanos	Smith	Jones	Ortiz	McKulski
STU_MAJOR	Accounting	Accounting	Marketing	Marketing	Statistics
DEPT_CODE	ACCT	ACCT	MKTG	MKTG	MATH
DEPT_NAME	Accounting	Accounting	Marketing	Marketing	Mathematics
DEPT_PHONE	4356	4356	4378	4378	3420
COLLEGE_NAME	Business Admin	Business Admin	Business Admin	Business Admin	Arts & Sciences
ADVISOR_LNAME	Grastrand	Grastrand	Gentry	Tillery	Chen
ADVISOR_OFFICE	T201	T201	T228	T356	J331
ADVISOR_BLDG	Torre Building	Torre Building	Torre Building	Torre Building	Jones Building
ADVISOR_PHONE	2115	2115	2123	2159	3209
STU_GPA	3.87	2.78	2.31	3.45	3.58
STU_HOURS	75	45	117	113	87
STU_CLASS	Junior	Sophomore	Senior	Senior	Junior

- Write the relational schema, draw its dependency diagram, and identify all dependencies, including all transitive dependencies.
- Write the relational schema and draw the dependency diagram to meet the 3NF requirements to the greatest extent possible. If you believe that practical considerations dictate using a 2NF structure, explain why your decision to retain 2NF is appropriate. If necessary, add or modify

attributes to create appropriate determinants and to adhere to the naming conventions.
- Draw the Crow's Foot ERD.

5. To keep track of office furniture, computers, printers, and so on, the FOUNDIT company uses the table structure shown in Table P6.5.

Table P6.5 Sample ITEM Records

Attribute Name	Sample Value	Sample Value	Sample Value
ITEM_ID	231134-678	342245-225	254668-449
ITEM_LABEL	HP DeskJet 895Cse	HP Toner	DT Scanner
ROOM_NUMBER	325	325	123
BLDG_CODE	NTC	NTC	CSF
BLDG_NAME	Nottooclear	Nottoclear	Canseefar
BLDG_MANAGER	I. B. Rightonit	I. B. Rightonit	May B. Next

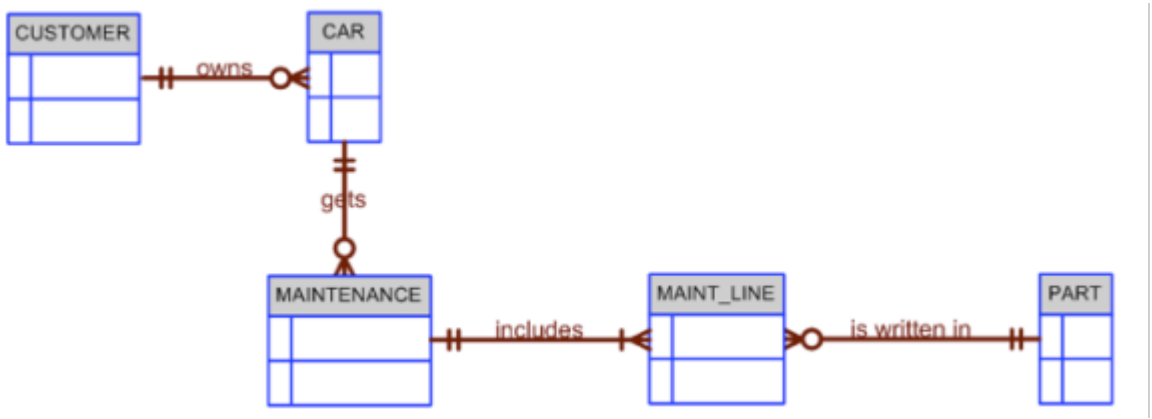
- Given that information, write the relational schema and draw the dependency diagram. Make sure that you label the transitive and/or partial dependencies.
- Write the relational schemas and create a set of dependency diagrams that meet 3NF requirements. Rename attributes to meet the naming conventions, and create new entities and attributes as necessary
- Draw the Crow's Foot ERD.

4
Tutorial 4: Conceptual Modelling

Conceptual Design - Theory Questions

Reference: Coronel, C and Morris, Database Systems: Design, Implementation & Management, Chapter 4, Selected Review Questions and Problems.

1. What two conditions must be met before an entity can be classified as a weak entity? Give an example of a weak entity.
2. What is a strong (or identifying) relationship, and how is it depicted in a Crow's Foot ERD?
3. Given the business rule "an employee may have many degrees," discuss its effect on attributes, entities, and relationships. (Hint: Remember what a multivalued attribute is and how it might be implemented.)
4. What is a composite entity, and when is it used?
5. Suppose you are working within the framework of the conceptual model in Figure Q4.5.Figure Q4.5 The Conceptual Model for Question 5



o Write the business rules that are reflected in it.

o Identify all the cardinalities.

6. What is a recursive relationship? Give an example.
7. How would you (graphically) identify each of the following ER model components?

o an entity

o the cardinality (0,N)

o a weak relationship, and

o a strong relationship

8. Discuss the difference between a composite key and a composite attribute. How would each be indicated in an ERD?
9. What two courses of action are available to a designer when he or she encounters a multivalued

attribute?

10. What is a derived attribute? Give an example.
11. How is a relationship between entities indicated in an ERD? Give an example, using the Crow's Foot notation.
12. Discuss two ways in which the 1:M relationship between COURSE and CLASS can be implemented. (Hint: Think about relationship strength.)
13. How is a composite entity represented in an ERD, and what is its function? Illustrate using the Crow's Foot model.
14. Briefly, but precisely, explain the difference between single-valued attributes and simple attributes. Give an example of each.
15. What are multivalued attributes, and how can they be handled within the database design?

4.1
Tutorial 4: Building Conceptual Models

Conceptual Modelling - Practical Work

For this weeks lab exercises we are going to prepare conceptual models (Entity Relationship Diagrams - ERDs) for a number of scenarios. At this point we are not interested in the database implementation of these models, our aim will be to model without any consideration of the database system in which the model may ultimately be implemented.

To prepare your conceptual models (ERD's) you may use:

● Lucidchart (https://www.lucidchart.com/) - this product is a browser based diagramming tool; it is able to draw a wide range of different diagrams, including ER Diagrams. As a University student you are entitled to a free Lucidchart account. Your account has been created and an invitation email sent to your Monash student account (check Spam if the message is not in your inBox). Click on the link in your Lucidchart welcome email and you will be able to make use of the software. As a first step you should look at the provided tutorials, in particular "Entity Relationship Diagrams (https://lucidchart.zendesk.com/entries/21606135-Entity-relationship-diagrams)" (begin with "Manual ERD Creation", note that the model we are asking you to build should not include the "Type" column), or

● any other drawing package you wish. One excellent alternative is Gliffy (https://www.gliffy.com/)

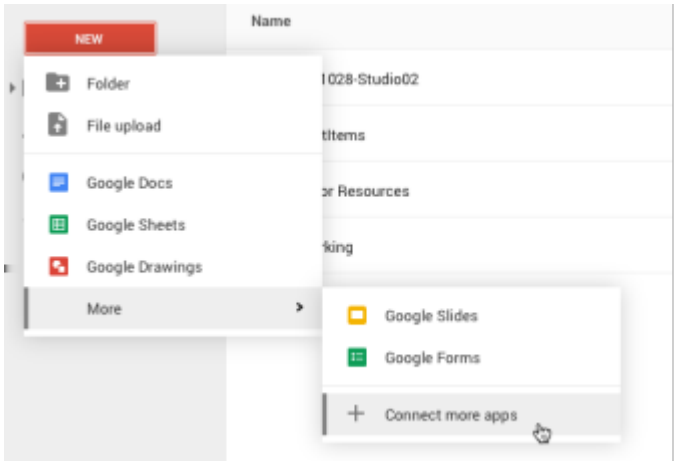
Please note for this unit, *hand drawn diagrams are not acceptable*.

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.

Lucidchart

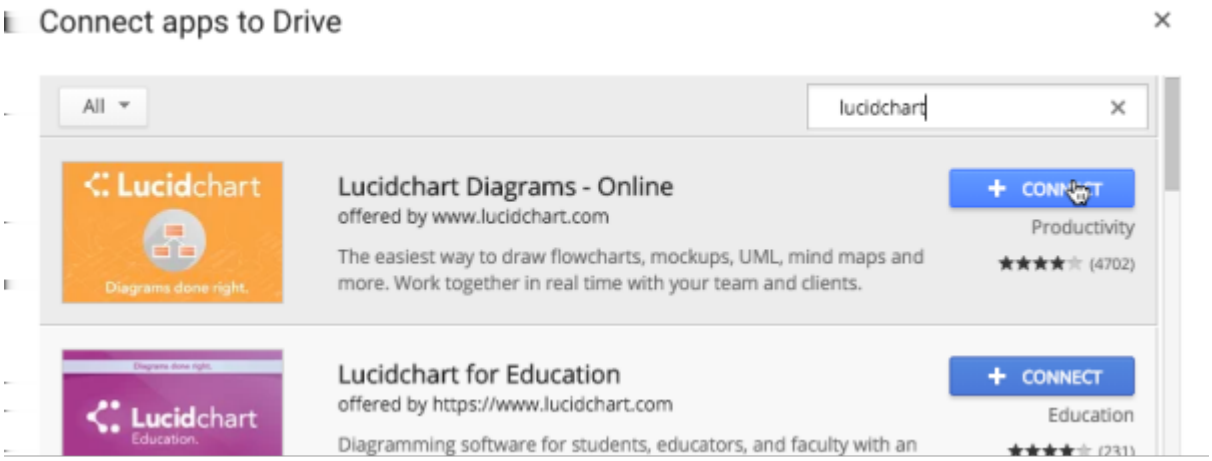
After you have setup your account, Lucidchart can be logged into directly at the Lucidchart site: https://www.lucidchart.com/ or better, accessed from your student Google Drive (from my.monash select the "Drive" link).

Connect to Google Drive, click "New" and then more, and "Connect more apps"



This will open the app picker which lists a wide range of applications that can be added to your Google drive.

You can browse through and find Lucidchart or alternatively type "Lucidchart" in the top right search box:



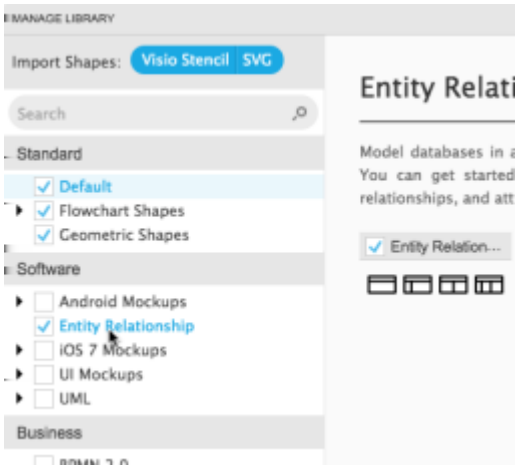
Select the "+ CONNECT" button for Lucidchart Diagrams - Online (do not select Lucidchart for Education)

After this process has been completed Lucidchart charts created from within your Google Drive will be stored in your Google Drive and not on the Lucidchart server.

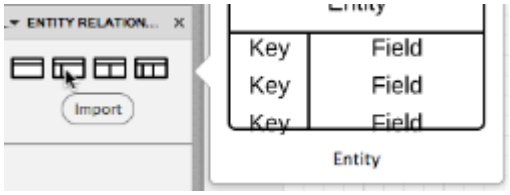
To create a new Lucidchart - simply select New in Google Drive, and then select "Lucidchart" (you may need to expand "More" to see the Lucidchart option).

If this is the first document you have created as a Lucid Chart from your Google Drive you will be asked to approve Lucidchart's access to your Google Drive - please ensure you select "Accept". This will then transfer you into the Lucidchart workspace with a new document open. Select the "Entity Relationship (ERD)" template, then Blank ERD and then "Start Drawing".

If the ERD shapes are not listed in the left panel, add the ERD shapes by selecting "More Shapes" (bottom left) and then checking "Entity Relationship" and then "Save":



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

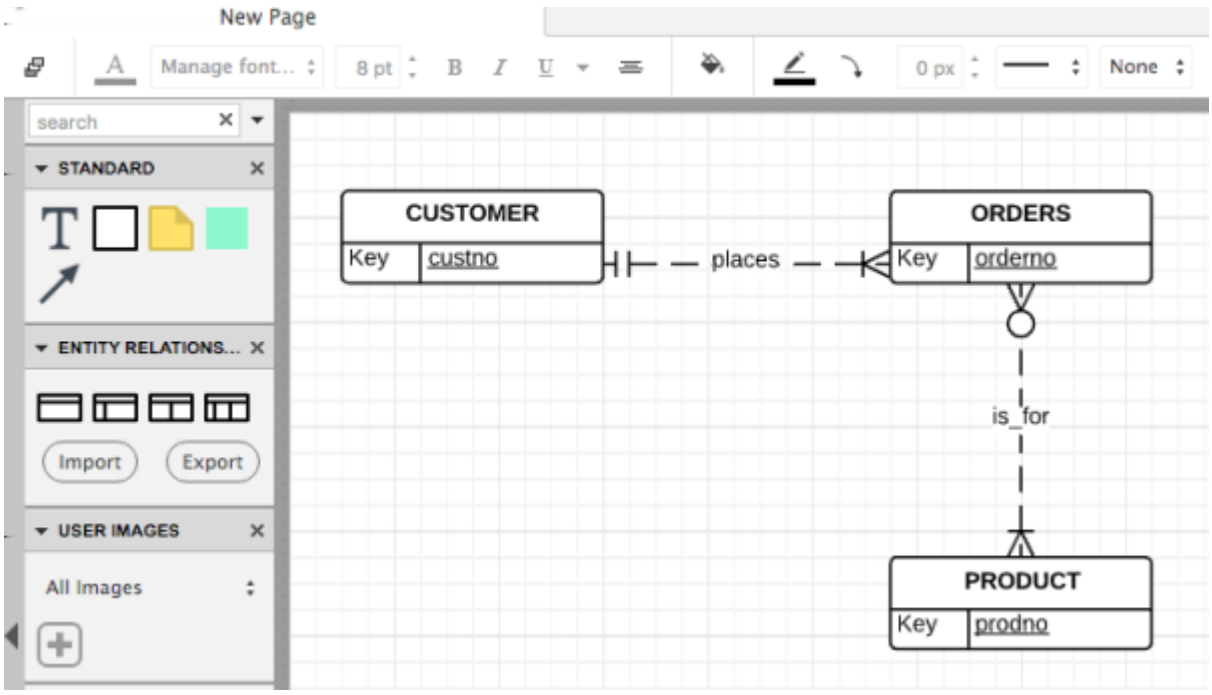


Task A: Customer - Orders ERD

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, customer number and for each product ordered the quantity ordered and the total line price
- PRODUCT - product number, product description and product unit price

An initial ERD using Lucidchart for this scenario would be:



Note you might like to try the new "Beta" interface for LucidChart if you wish to try their latest GUI (the above is the current interface).

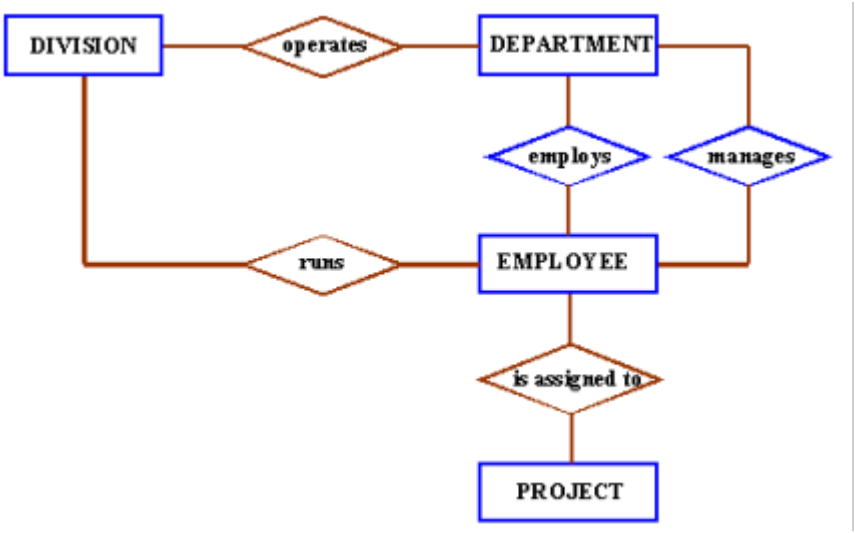
This ERD only shows the keys of each of the entities, sometimes an ERD is drawn such that it will show all the non- key attributes for the model as well. In such a complete ERD you must not show or label foreign keys, the use of foreign keys indicates that you are looking at a logical model where a choice has been made to use a relational database, rather than a conceptual model.

Prepare the ERD shown above using your choice of drawing tool and then create a copy of this initial model and add all of the non key attributes listed in the scenario above to your copy.

Task B: Other Models

The first problem is based on the following general entity layout.

Figure P3.1.



1. Use the above diagram and the business rules below, create a Entity Relationship Diagram using your selected drawing tool.
    - Include:
        - all appropriate connectivities,
        - all cardinalities and
        - at least the minimum number of attributes required to implement the model
    - 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.
        - A division operates many departments, but each department is operated by one division
        - 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
        - One of the employees runs each division, and each division is run by one employee.
2. Prepare an Entity Relationship Diagram (ERD) showing all primary and non-primary key attributes 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 building 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. In addition the lease period and bond are recorded for each property or sub property rented. An owner may own several properties.
- Properties are subject to damage and the agency records all instance of damage to its properties - property, date, type of damage and repair cost are recorded. Repair costs are charged directly to tenants
- Normal property maintenance is also noted - property, date, type of maintenance and cost are recorded. Maintenance costs are charged to the property owner.
- 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.

5
Tutorial 5: Logical Modelling

Logical Modeling

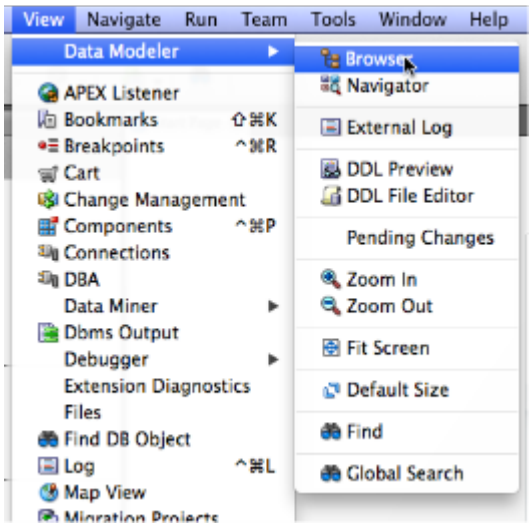
After preparing a conceptual model the next stage is to select the type of database we will use (for us relational) and convert our conceptual model into an appropriate logical model. For logical modelling we will make use of Oracle SQL Developer Data Modeler.

Task A: Using SQL Developer Data Modeler

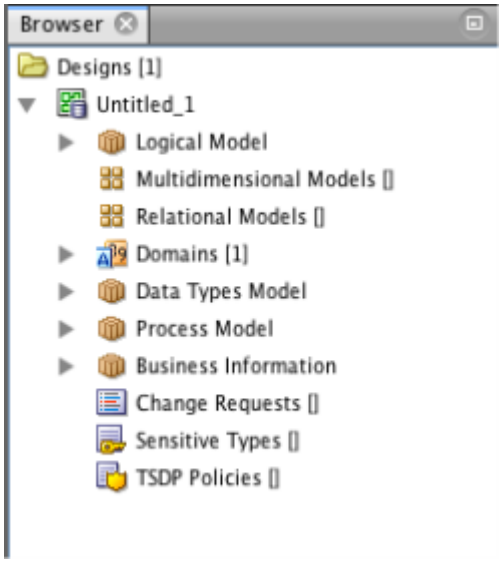
This software is a commercial level tool with an extensive set of features including support for Subversion versioning and source control thus permitting teams of developers to work collectively on a design. Given the extensive range of features we are only going to be looking at a subset of these for our study. SQL Developer Data Modeler begins with a (relational) LOGICAL model and then creates what it calls a RELATIONAL model from which the schema file can be generated. The relational model is essentially a graphical representation of the physical model.

Accessing Data Modeler

Data Modeler is installed as part of the standard SQL Developer installation. To work with Data Modeler it is helpful to have the Data Modeler browser open in the left panel of SQL Developer. To open this panel select View - Data Modeler - Browser:

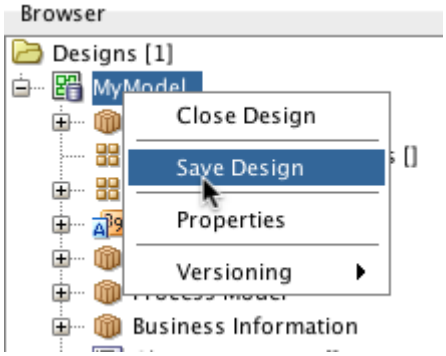


The browser opens with a new (unnamed) model:

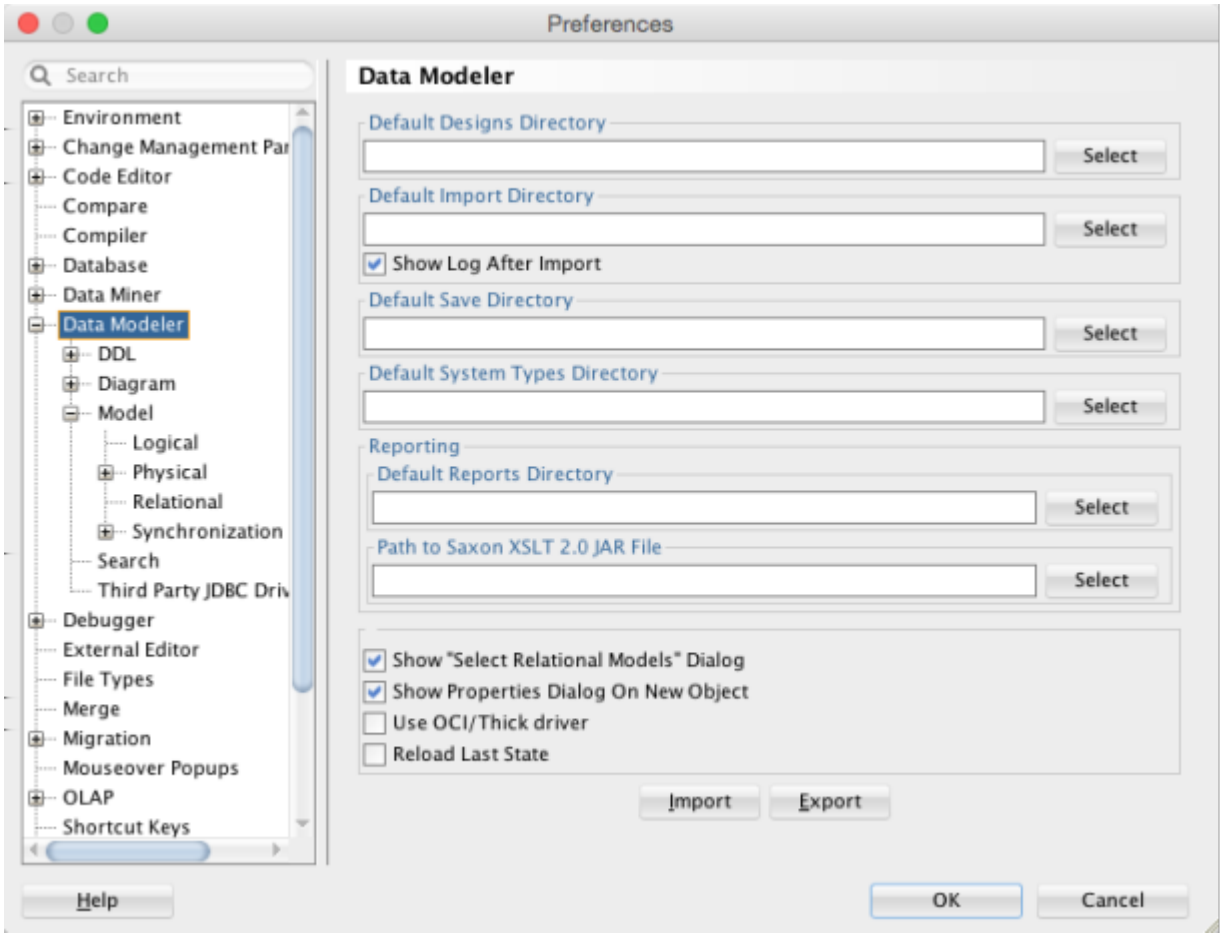


To begin creating a logical model, right click Logical Model and select Show. This will open a blank model in the main working panel of SQL Developer and add a range of new icons to the main toolbar. Hover over each of these new icons to become familiar with with what they represent. You should regularly save your design - right click on the design (here Untitled_1 as the design has not been saved as yet) and select a save location, after the first save the design will be named.

When using Data Modeler it is very important that you save and close your model before exiting SQL Developer or shutting your laptop. Failure to do so may result in loss of parts of your model.

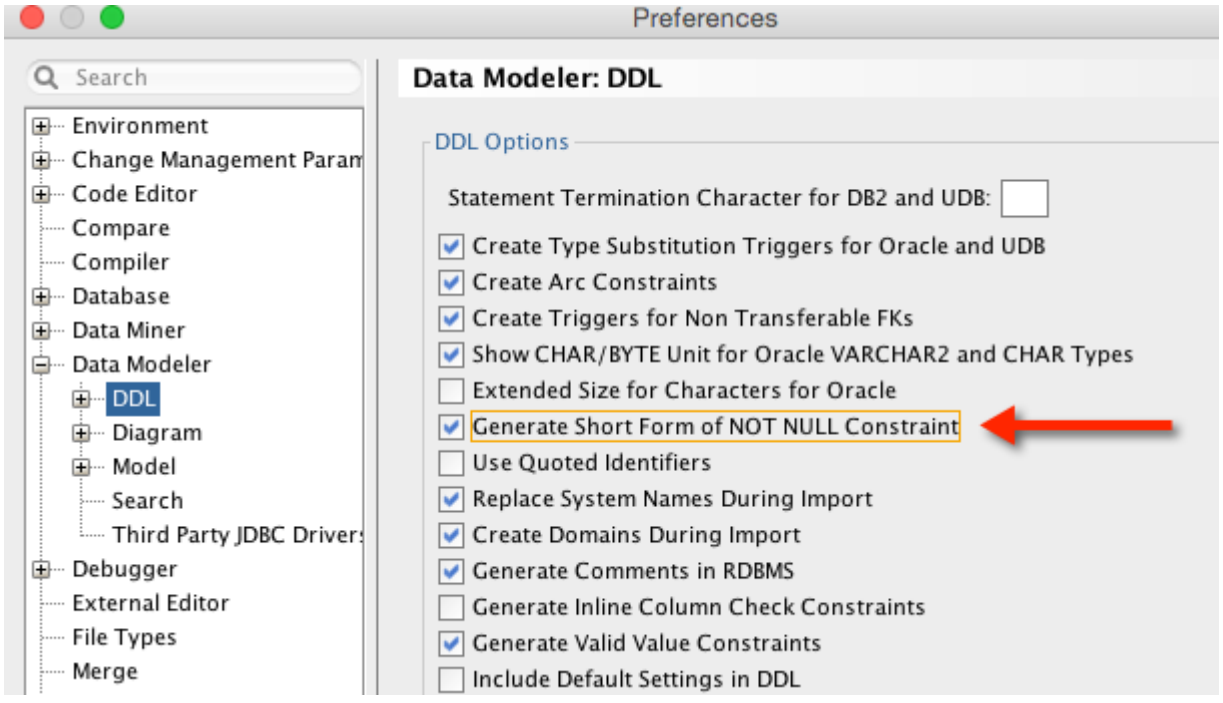


A number of features should be configured using the SQL Developer preferences (Windows: Tools-Preferences, OSX: Oracle SQL Developer-Preferences), select the Data Modeler:

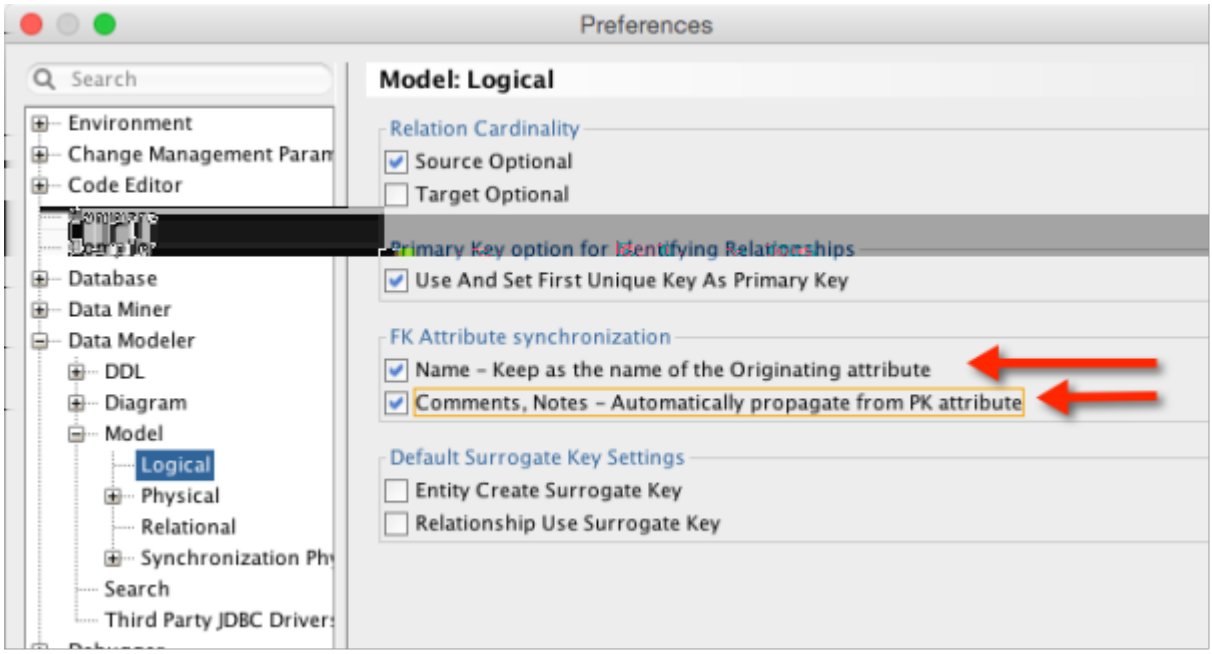


We wish to modify two features:

(a) select DDL and select the option to "Generate Short Form of NOT NULL Constraint" - this will cause not null constraints to be not named



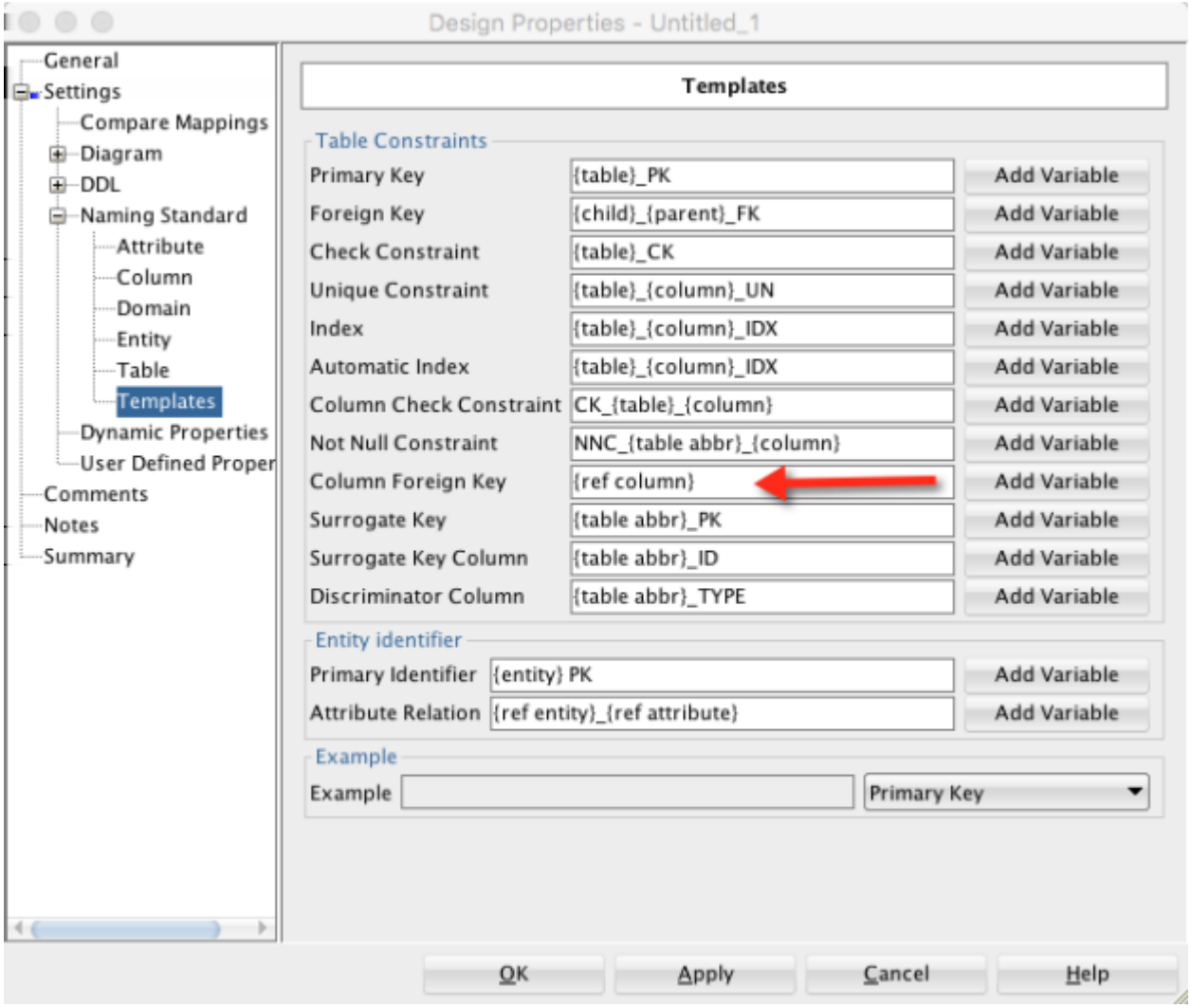
(b) select Model, Logical and check both items in "FK Attribute synchronization"



There are a large number of other settings available to configure, you might like to investigate these if you are using your own copy of SQL Developer, for the labs we will leave the remaining settings at the default values.

Develop a model - Stage 1 The Logical Model:

Before starting a new project, right click the project in the Data Modeler browser and select properties. Within the Design Properties sheet select Settings - Naming Standard - Templates and modify the "Column Foreign Key" setting from {ref table}_{ref column} to {ref column}:



For your first model we will implement a Customer-Orders system, represented by the following entities, where customers place orders for products:

CUSTOMER - customer number, name, address, phone number

ORDER - order number, order date, customer number and for each product ordered the quantity ordered and the total line price

PRODUCT - product number, product description and product unit price

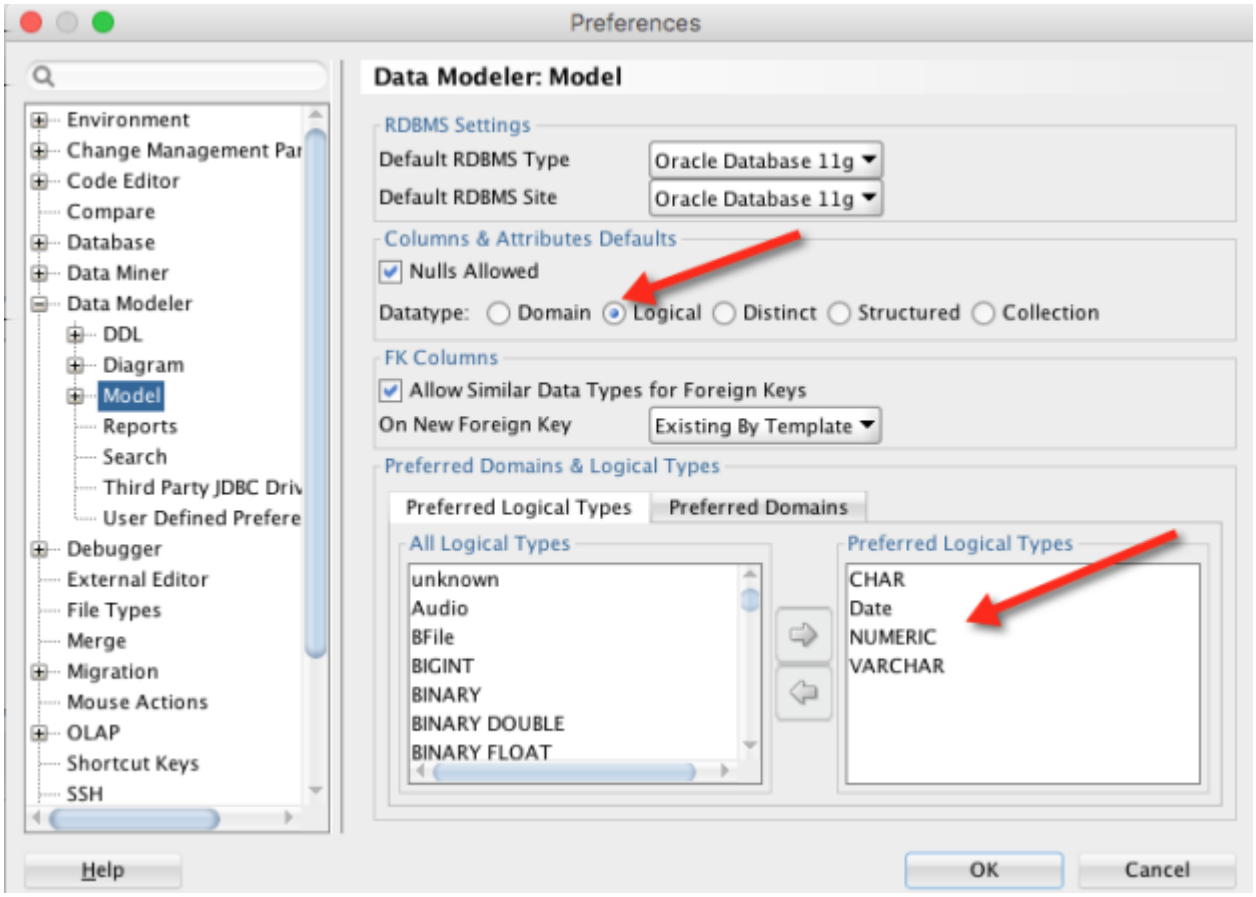
Attributes on the logical model have a number of possible data types, the main ones for us being "Domain" and "Logical":

● Domain types are domains which you create via the menu items - Tools - Data Modeler - Domains Administration. No domain types are supplied, you must create any you need

● Logical types are not actual data types - they are names which are mapped to native types at a later stage. These logical types are pre populated with several Oracle types

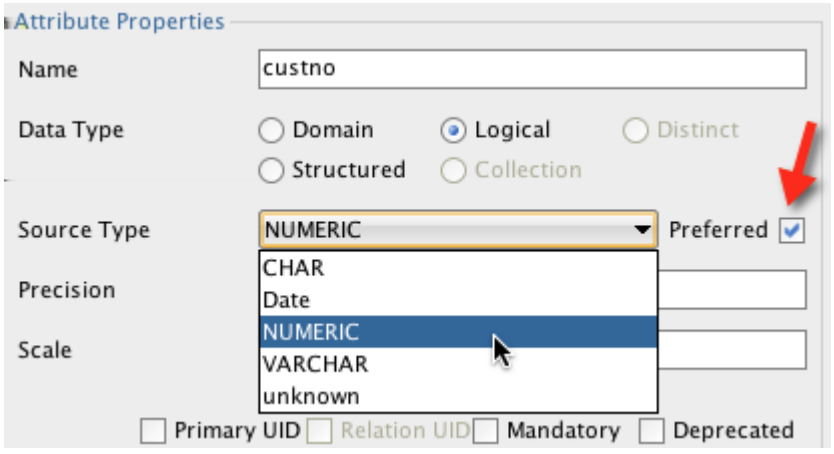
For our work we will not make use of domain types, instead you should always use logical types. You can

speed up entry of attributes by restricting Data Modeler to logical and types and a preferred range of types. In your SQL Developer preferences set:



The 'Preferred Logical Types' are populated by selecting the item in the 'All Logical Types" and clicking on the right pointing arrow between All and Preferred.

Then when adding an attribute:



check the 'Preferred' tick box and you will only see those types you have selected as preferred (clearly you may modify the set preferred data types to suit your needs).

On your logical model add an entity named CUSTOMER (name the entity under the General features in the Entity Properties dialog box) and then select the attributes feature. Add the following attributes:

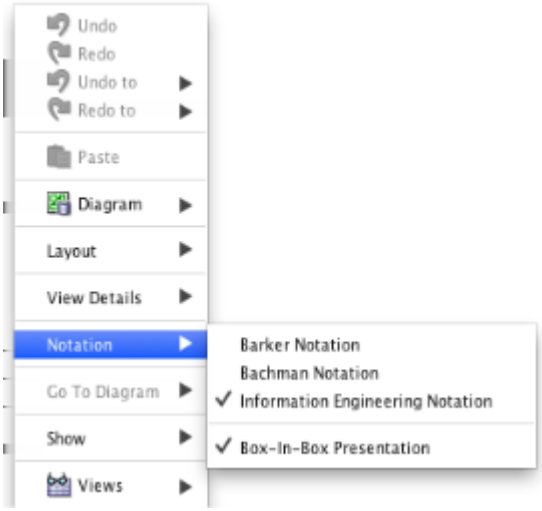
- custno - Logical type: Numeric Precision 7, Scale 0, Primary UID
- custname - Logical type: VARCHAR(50), Mandatory
- custaddress - Logical type: VARCHAR(50), Mandatory
- custphone - Logical type: CHAR(10)

For each attribute add a meaningful description of the attribute under the attribute option - "Comments in the RDBMS". Then add an ORDERS entity with attributes:

- orderno - Logical type: Numeric Precision 7, Scale 0, Primary UID
- orderdate - Logical type: DATE, Mandatory

For all the logical models we create you should set (right click on the logical model) the

- Notation to "Information Engineering Notation",

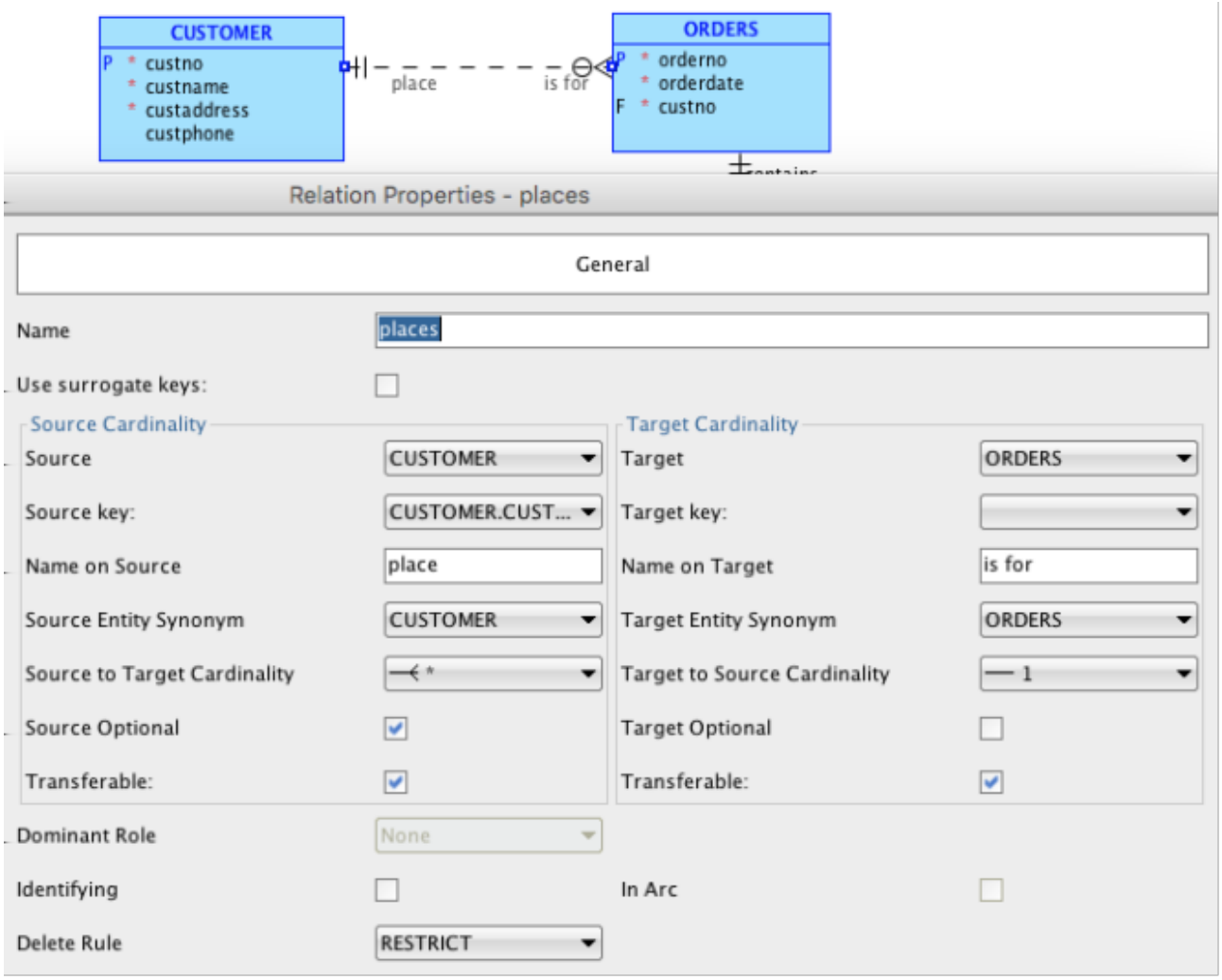


- View Details to "Attributes", and
- Show to "Labels" and "Legend"

When modelling students are required to include the "Legend" on all models (the panel may be moved to fit your models layout). **Models submitted without a legend will not be graded**.

Now add a 1:N Relationship between CUSTOMER and ORDERS - click in CUSTOMER (the parent) and then in ORDERS. In the General Relations Properties name the relation "places".

Enter a name for the source and target, and set up the participation (Customer - optional, Order - mandatory ie. not optional):

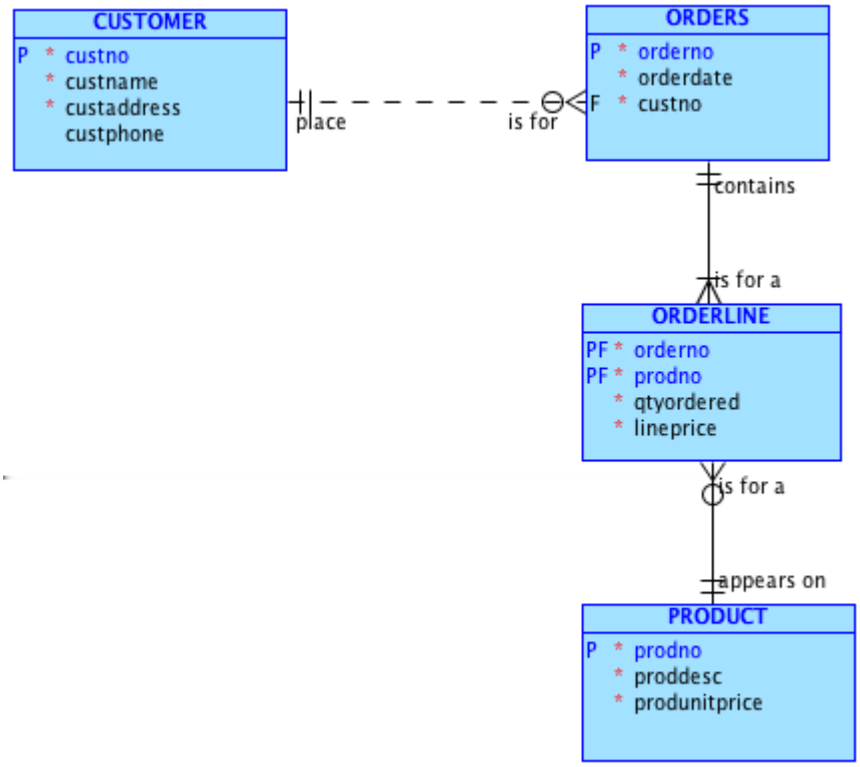


Proceed and add the PRODUCT entity.

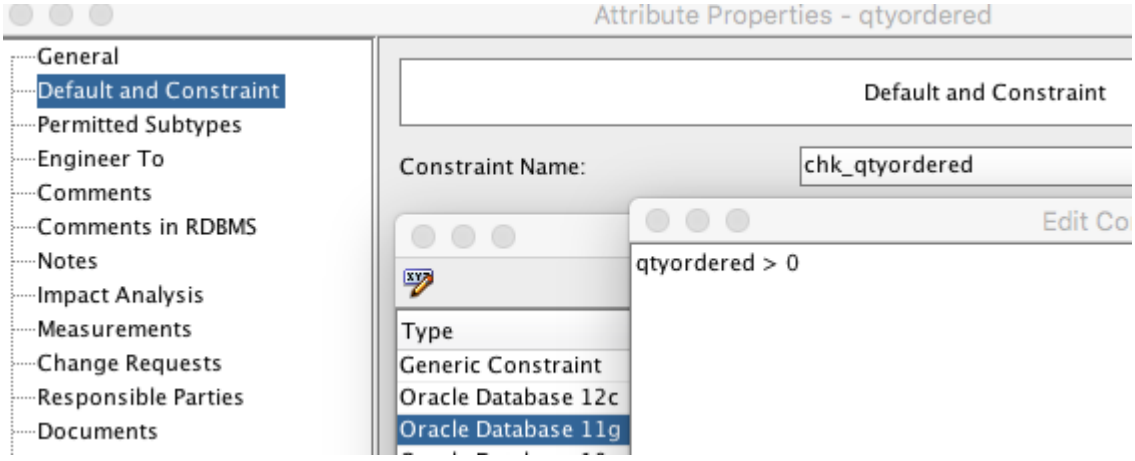
Modeler can draw M:N relations(hips) on its logical models, however you cannot add attributes to such relationships - ***please do not use the M:N relation(ship) in our models***.

Add a new entity ORDERLINE and then connect this with ORDERS and PRODUCT via 1:N Identifying relations(hips).

Your final logical model will have the form:



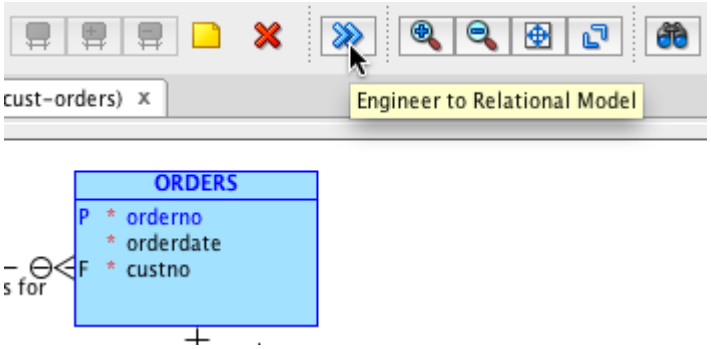
Constraints can be added to the logical model, as an example lets add a constraint to say that qtyordered in ORDERLINE must be greater than zero. Select the ORDERLINE entity, then select attributes and double click on qtyordered. In the left hand list of the pop-up window select "Default and Constraint". Give the constraint a name for example "chk_qtyordered" (be careful to select an informative name eg. chk_columnname) and then enter the constraint into the constraint editor:



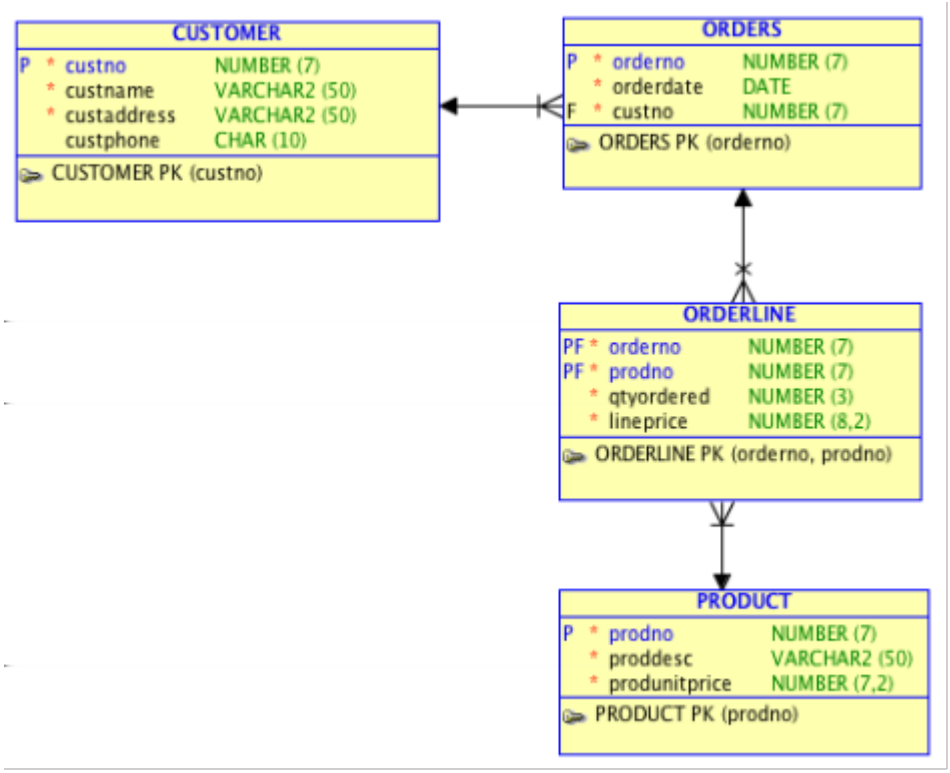
Note that the constraint added is the inner part of the standard SQL CHECK clause.

Develop a model - Stage 2 The Relational (Physical) Model:

This completed logical model is now "Engineered" to a "Relational Model".

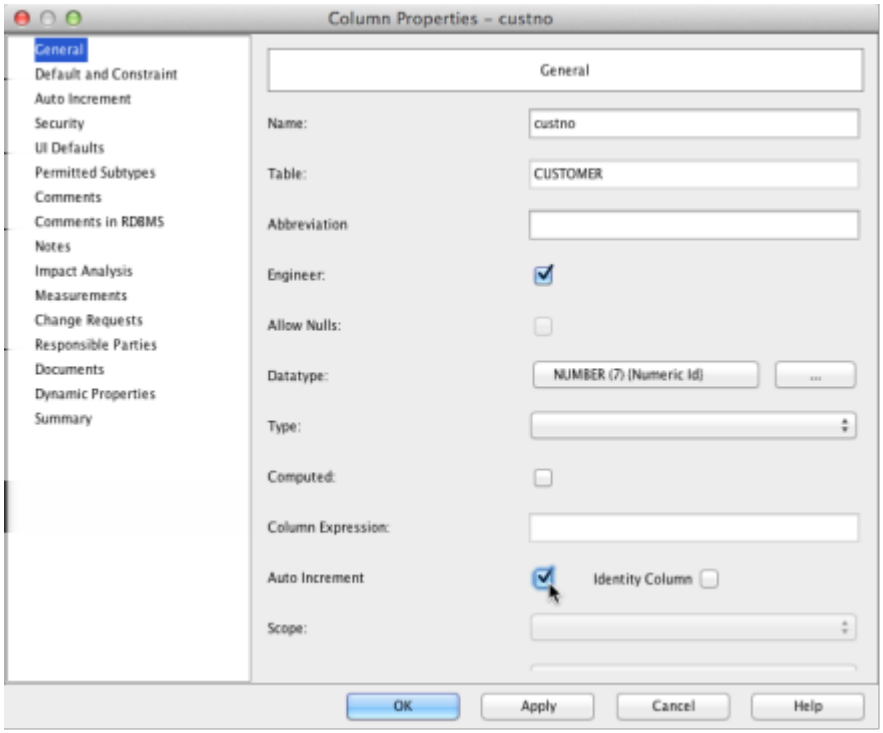


In the pop-up window which appears, select "Engineer" the bottom left button. The logical model will then be engineered and a Relational Model is opened:

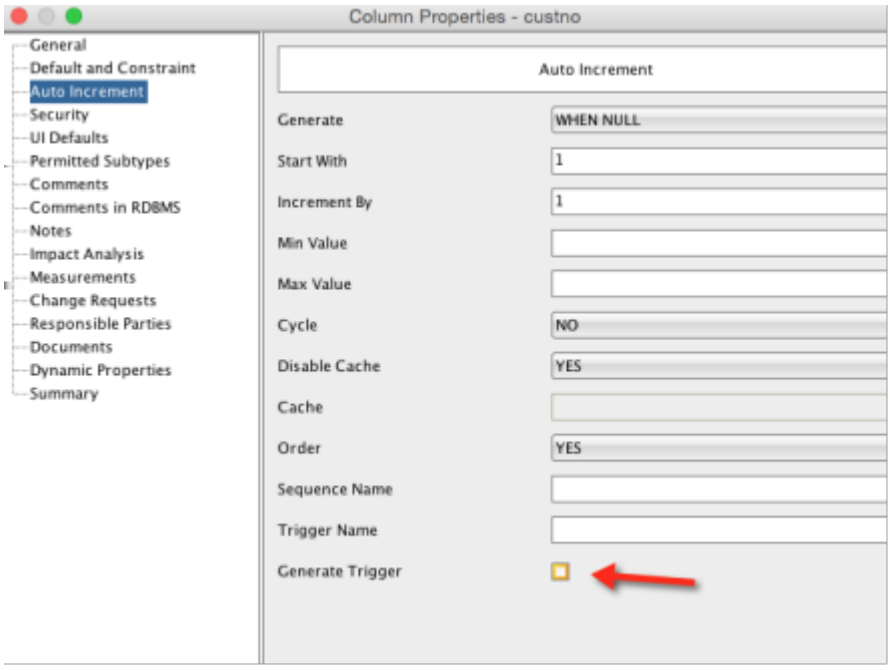


Note that in the default preferences setting, the Data Modeler relational model shows relationship lines pointing from the FK to the PK in the parent entity.

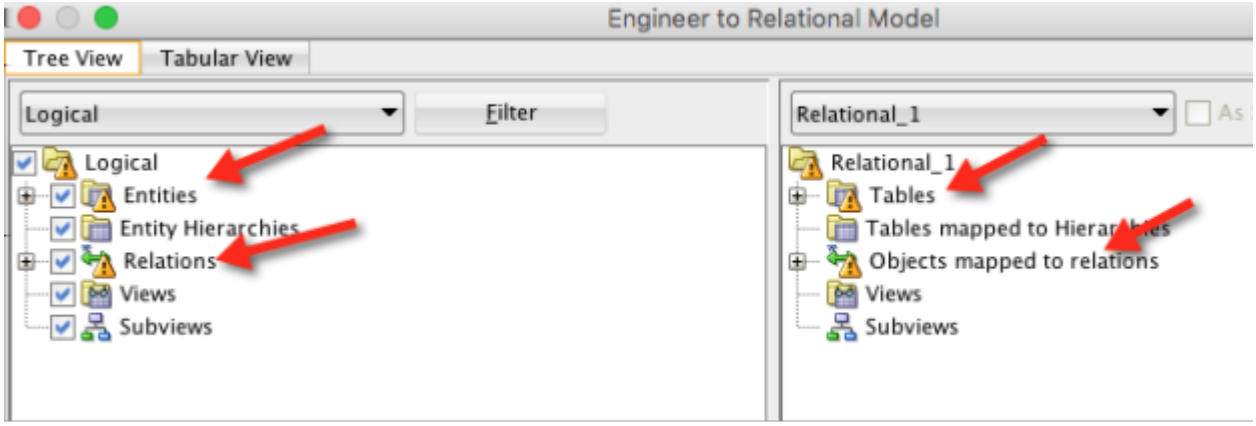
As an example of what can be configured in the Relational model, select the CUSTOMER table, then the columns, then double click on the custno column, in the dialog which opens select "Auto Increment"



This will result in a sequence being created when the DDL is generated. The default configuration also generates a trigger to support the auto increment which we do not need. To prevent this trigger being generated, after you have selected "Auto Increment" on General (above), select the Auto Increment option on the left and uncheck "Generate Trigger".

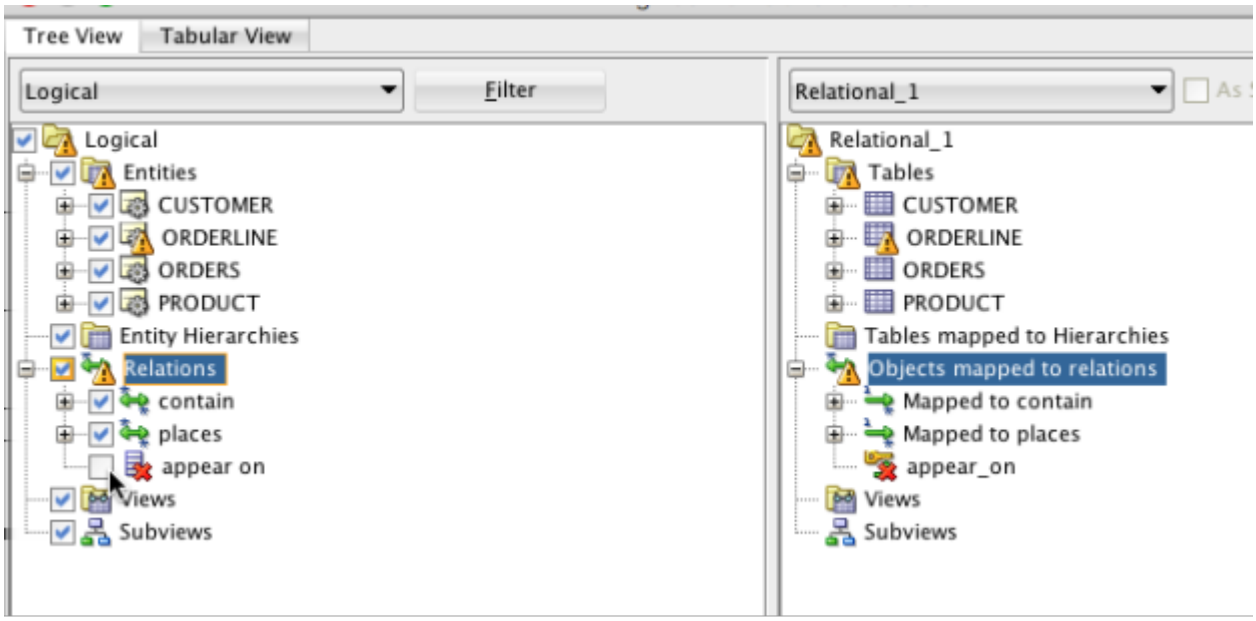


If you are re-engineering a model (ie. trying to generate a previous relational model, after changes to your logical model) it is very important that you note that SQL Developer does not automatically sync deletions from your logical model - such changes must be individually selected to be synced to your relational model. When re-engineering a previous model carefully check the "Engineer to Relational Model" for any triangles with an exclamation mark symbol:



Such symbols represent issues you ***must address*** before generation.

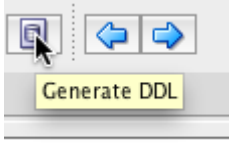
For example here under relations (where we are removing the "appear on" relation as a demonstration of what occurs):



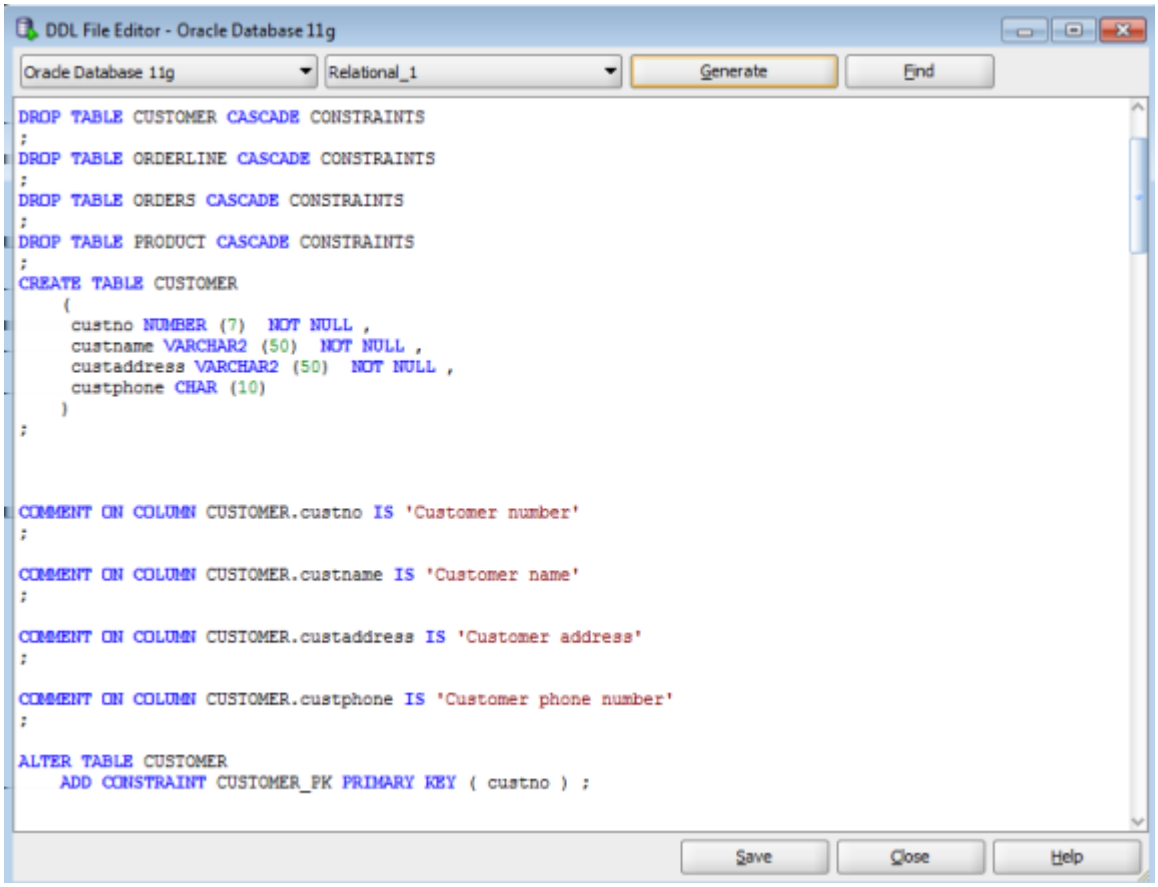
the removal of "appear on" has not been selected to be engineered to the relational model. You need to check the box if you wish it to be engineered (which we normally would).

If your relational model gets very confused you can select the relational model tab, do a ctrl+A or Apple+A and delete all the objects. The model can then be regenerated. Under ***no circumstances should you delete the relational model itself*** (in the left browser navigator), a bug in several versions of the software can result in such an action causing your relational model to "disappear".

When you have configured the relational model as you wish, select Generate DDL from the top toolbar:



Select "Generate" in the pop-up window, specify the DDL Generation Options you wish (Drop tables should be included) and then select OK to generate the DDL.



The generated file can be Saved as an Oracle schema script (ensure you use an extension of .sql). Test your generated file against Oracle and confirm it operates correctly.

It is also possible to configure Data Modeler to directly synchronise the design into the Data Dictionary of a database connection (we will not use this approach).

5.1
Tutorial 5: Logical Modelling - Task B - Rental Model

Task B: Using SQL Developer Data Modeler

Using your model from last week for the "Property Rental System", map your Conceptual model (ERD) into a logical model in Oracle SQL Developer Data Modeler.

Engineer your Logical Model to a Relational Model and then create the tables etc in Oracle from the generated DDL. In doing so make use of at least one check clause and one sequence.

- 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 building 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. In addition the lease period and bond are recorded for each property or sub property rented. An owner may own several properties.
- Properties are subject to damage and the agency records all instance of damage to its properties - property, date, type of damage and repair cost are recorded. Repair costs are charged directly to tenants
- Normal property maintenance is also noted - property, date, type of maintenance and cost are recorded. Maintenance costs are charged to the property owner.
- 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.

6
Tutorial 6: SQL Data Definition Language (DDL) and Insert

FIT2094/FIT9132

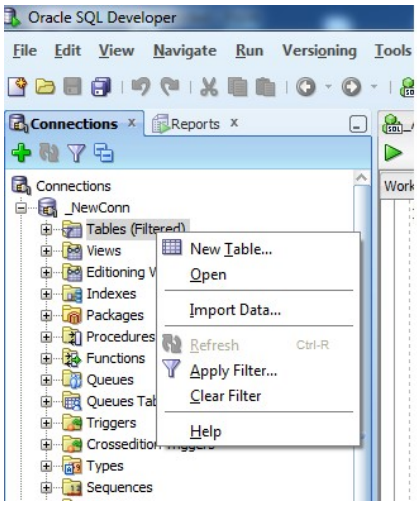
In Tutorial 1.2, you have learnt how to create a database, inserting, updating and deleting data using graphical user interface of SQL Developer. In this tutorial, you will learn how to do the above tasks using SQL statements written as SQL worksheet in SQL Developer. In some database technologies, the SQL worksheet is called SQL script. We will refer to the collection of SQL statements in a single file interchangeably between worksheet and script.

6.1
Using SQL Worksheet to create database and to manage data

1. Creating tables

There are two techniques that you can use to create tables using SQL Developer:

- The softwares GUI interface - here the SQLDeveloper graphical interface is used to build a table column by column (most databases have such a tool), for example:



(https://www.alexandriarepository.org/wp-content/uploads/20160212080736/Create_tables_a1.jpg)

Right click the Table icon, select "New Table" and build a table using the SQLDeveloper GUI. To complete the table creation, you will need to fill in the columns details. But for this unit, we will not use this method. We will use the next method, by using an SQL script that contain the database schema (a text file containing the SQL commands to create the tables).

- Writing an SQL Script containing the database schema. - *this is the **required approach** because of its ease of repeatability.* You can easily remove your tables and then recreate them at any time by simply rerunning the schema script.

A typical script, called a schema file, is created in the SQL Developer SQL Worksheet. An example of a schema for the customer table is provided below.

```
rem
rem Table               : customer
rem cust_numb           : Customer number
rem custname            : Customer Name
rem cust_address        : Customer Address
rem cust_bal            : Customer current balance
rem cust_cred_limit     : Customer Credit limit
rem slsr_number         : Sales rep who services customer
rem
```

```
create table cust_test_script (
cust_numb       char (10) not null,
custname        char (30) not null,
cust_address    char (40) not null,
cust_bal        number (8,2) not null,
cust_cred_limit number (8,2) not null,
slsr_number     number (2));
```

The "rem" lines in the scripts are comments that have been included for the purpose of documentation. The query processor inside the database will ignore these lines.

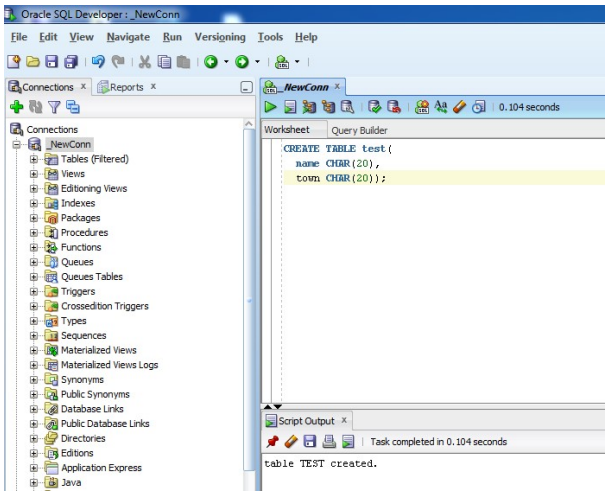
We will cover the details of the different components of a CREATE TABLE statement in the future. For now, the main idea is for you to be aware of these different ways of creating tables.

To practice using SQL Developer to create table, write the following SQL statement in SQL developer in the SQL worksheet area.

```
CREATE TABLE test(
  name CHAR(20),
  town CHAR(20));
```

To run the SQL statement, press the green arrow.

The SQL developer screen will look similar to the following figure.



(https://www.alexandriarepository.org/wp-content/uploads/20160212080831/Create_tables_c1.jpg)

2. Retrieving data from tables

Retrieving data from the database is carried out using SQL SELECT statements.

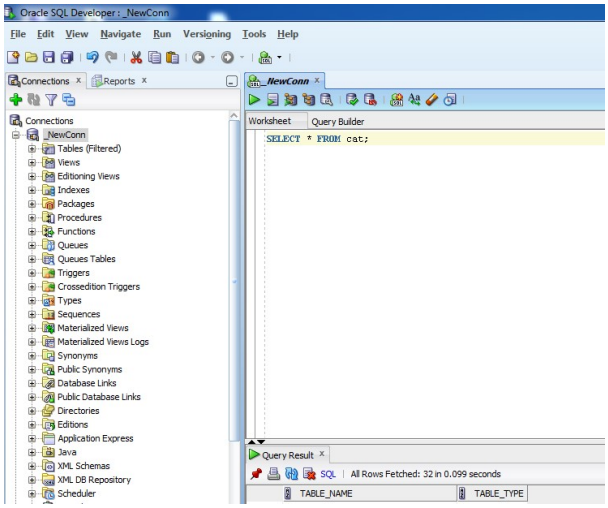
It is a good practice to separate the CREATE TABLE statements from the SELECT statements into different files (scripts). To create a new SQL worksheet or file, click on the "NEW" icon in the top left corner:

Choose Database Tier->Database Files->SQL File.

Then you will be provided with a new SQL worksheet where you can type the SELECT statement. To run the statement, you need to press the green arrow.

Type the following SELECT statement in the new SQL Worksheet.

```
SELECT * FROM cat;
```



(https://www.alexandriarepository.org/wp-content/uploads/20160212081125/Retrieve_data_b1.jpg)

Execute the statement and you should see output of the form:

The SELECT statement that you have just executed contains instruction to the database to retrieve all the data from the catalogue in your account. Since you have created one table called TEST earlier, this information is displayed.

	TABLE_NAME	TABLE_TYPE
1	TEST	TABLE

3. Adding data to tables

SQL **INSERT** statements are used to add data into tables. To continue this exercise, open a new SQL worksheet and type the following INSERT statement (note character data is enclosed in single quotes).

```
INSERT INTO test VALUES ('Lindsay Smith','Caulfield');
INSERT INTO test VALUES ('Harry Potter', 'Melbourne');
COMMIT;
```

The above script contains two INSERT statements. Each of the statement is used to add a single row or record to the table TEST. The COMMIT is used to instruct the database to make the insertion permanent. To run a script that contains multiple statements as a single execution, you need to choose a different

icon than the green arrow that you have used previously. This time, you need to choose the "smaller green arrow on top of a paper" icon.

After the execution at the Script output area you will see a message

```
1 rows inserted.
1 rows inserted.
Commited.
```

To examine the contents of your table, use the SQL **SELECT** command (remember to type this at the Worksheet that you use to write the SELECT statements):

```
SELECT * FROM test;
```

4. Deleting tables

Again, create a new worksheet to write this statement as it is of different type to the other statements that you have created. To remove a table use the SQL DROP statement:

```
DROP TABLE test PURGE;
```

The PURGE statement here ensures that the removed table is not stored in the database recycle bin of your account but completely removed. After execution, you will see the following output:

```
Table dropped.
```

Check that your table has been removed - do you remember how?

5. Saving SQL Worksheet

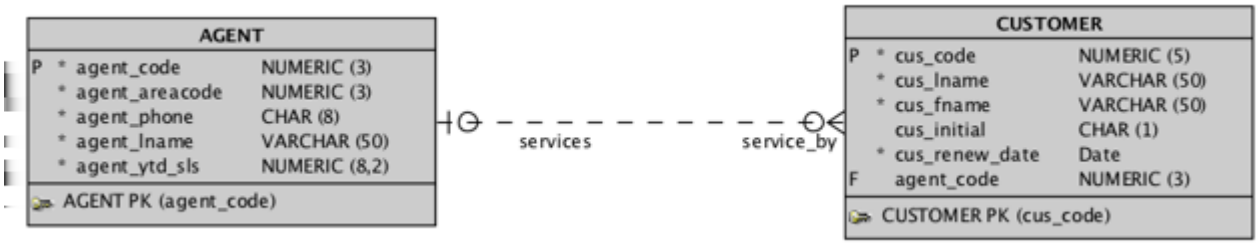
To save any of the SQL worksheets that are currently open in SQL Developer, you can use **File - Save or Save As** command from the File menu. The Save option will save only the current worksheet. To save all open worksheets you can choose **Save All**. If you are working on a University Lab PC please ensure that you have ***first*** set up a folder for saving as described in the unit software links of your unit's Moodle page.

6.2
Tutorial 6: SQL Data Definition Language (DDL)

When creating schema files, you should always also create a drop file or add the drop commands to the top of your schema file. You should drop the tables using the:

drop table tablename purge; syntax.

The drop table statements should list tables in the reverse order of your create table order so that FK relationships will be able to be removed successfully. Should a syntax error occur while testing your schema, you simply need to run the drop commands to remove any tables which may have been created.



The data model above represent figure 3.3 from Coronel & Morris. There are two different ways of coding this model as a set of create table statements.

Using table constraints

SQL constraints are classified as column or table constraints; depending on which item they are attached to:

```
create table agent
  (
    agent_code     number (3) constraint agent_pk primary key,
    agent_areacode number (3) not null ,
    agent_phone    char (8) not null ,
    agent_lname    varchar2 (50) not null ,
    agent_ytd_sls  number (8,2) not null
  ) ;
```

This is a declaration of the primary key as a column constraint

```
create table agent
  (
    agent_code     number (3) not null ,
    agent_areacode number (3) not null ,
    agent_phone    char (8) not null ,
    agent_lname    varchar2 (50) not null ,
    agent_ytd_sls  number (8,2) not null,
    constraint agent_pk primary key ( agent_code )
```

```
  ) ;
```

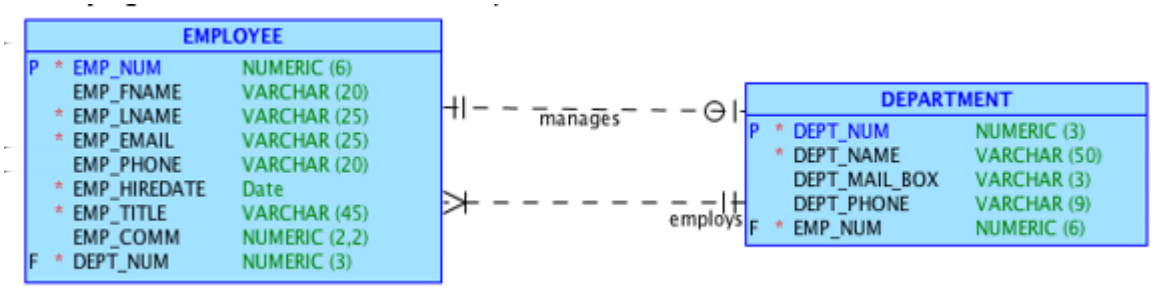
Here the primary key has been declared as a table constraint, at the end of the table after all column declarations have been completed. In some circumstances, for example a composite primary key you must use a table constraint since a column constraint refers only to a single column.

The create table statements for the two tables in fig 3-3 would be:

```
create table agent
  (
    agent_code     number (3) not null ,
    agent_areacode number (3) not null ,
    agent_phone    char (8) not null ,
    agent_lname    varchar2 (50) not null ,
    agent_ytd_sls  number (8,2) not null,
    constraint agent_pk primary key ( agent_code )
  ) ;


create table customer
  (
    cus_code       number (5) not null ,
    cus_lname      varchar2 (50) not null ,
    cus_fname      varchar2 (50) not null ,
    cus_initial    char (1) ,
    cus_renew_date date not null ,
    agent_code     number (3),
    constraint customer_pk primary key ( cus_code ),
    constraint customer_agent_fk foreign key ( agent_code)
       references agent ( agent_code ) on delete set null
  ) ;
```

In some circumstances this approach cannot be used. Can you see what the issue is with trying to create the two tables depicted below?



In such a situation an alternative approach to declaring constraints needs to be adopted.

Using ALTER table commands

In this approach the tables are declared without constraints and then the constraints are applied via the ALTER TABLE command (see section 7.5 of Coronel & Morris).

```
create table agent
  (
    agent_code     number (3) not null ,
    agent_areacode number (3) not null ,
    agent_phone    char (8) not null ,
    agent_lname    varchar2 (50) not null ,
    agent_ytd_sls  number (8,2) not null
  ) ;

alter table agent add constraint agent_pk primary key
   ( agent_code ) ;

create table customer
  (
    cus_code       number (5) not null ,
    cus_lname      varchar2 (50) not null ,
    cus_fname      varchar2 (50) not null ,
    cus_initial    char (1) ,
    cus_renew_date date not null ,
    agent_code     number (3)
  ) ;

alter table customer add constraint customer_pk primary key
    ( cus_code ) ;

alter table customer add constraint customer_agent_fk foreign key
   ( agent_code ) references agent ( agent_code )
   on delete set null;
```

After creating the tables we need to insert the data, for AGENT the insert will have the form:

```
insert into agent values (501,713,'228-1249','Alby',132735.75);
```

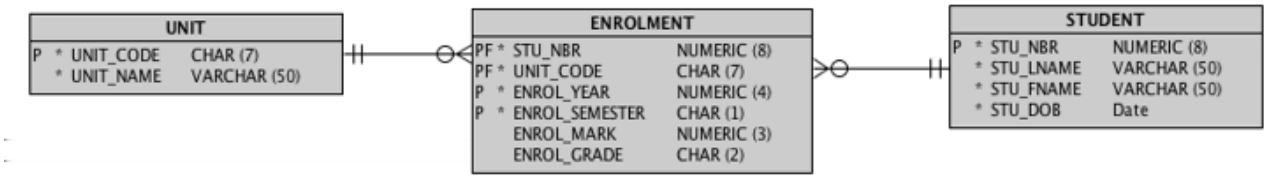
for customer:

```
insert into customer values(10010,'Ramas','Alfred','A',
    '05-Apr-2014',501);
```

It is important to note that for the insert into customer we are using the default Oracle date format of dd-mon-yyy - in the near future we will correct this and allow any date format via the oracle function *todate*.

Lab Tasks

Using the model from the DDL lecture for student, unit and enrolment:



Creating tables from scratch.

- Code a schema file to create these three tables, noting the following extra constraints:
    1. stu_nbr > 10000000
    2. unit_name is unique in the UNIT table
    3. enrol_semester can only contain the value of 1 or 2 or 3.
- In implementing these constraints you will need to make use of CHECK clauses (see Coronel & Morris section 7.2.6).
- Ensure your script file has appropriate comments in the header, includes the required drop commands and includes echo on and echo off commands.
- Run your script and create the three required tables.
- Save the output from this run.

Question 5:

Add a new column to the UNIT table which will represent points for the unit -

- The default should be 6 points (hint use the ALTER command).
- Insert a test new unit after you have added the new column.
- Check that the new insert has worked correctly.

6.2.1
Tutorial 6: INSERTing data into the database

1. Basic INSERT statement.

In this exercise, you will enter the data into the database using INSERT statement with the following assumptions:

- the database currently does not have any existing data.
- the primary key is not generated automatically by the DBMS.

TASKS

Insert the following data into the tables specified using the SQL INSERT statement:

STUDENT

stu_nbr	stu_lname	stu_fname	stu_dob
11111111	Bloggs	Fred	01-Jan-1990
11111112	Nice	Nick	10-Oct-1994
11111113	Wheat	Wendy	05-May-1990
11111114	Sheen	Cindy	25-Dec-1996

UNIT

unit_code	unit_name
FIT9999	FIT Last Unit
FIT5132	Introduction to Databases
FIT5016	Project
FIT5111	Student's Life

ENROLMENT

stu_nbr	unit_code	enrol_year	enrol_semester	enrol_mark	enrol_grade
11111111	FIT5132	2013	1	35	N
11111111	FIT5016	2013	1	61	C
11111111	FIT5132	2013	2	42	N
11111111	FIT5111	2013	2	76	D
11111111	FIT5132	2014	2		
11111112	FIT5132	2013	2	83	HD
11111112	FIT5111	2013	2	79	D
11111113	FIT5132	2014	2		
11111113	FIT5111	2014	2		
11111114	FIT5111	2014	2		

- Ensure you make use of COMMIT to make your changes permanent.
- Check that your data has inserted correctly by using the SQL command SELECT * FROM *tablename* **and** by using the SQL GUI (select the table in the right hand list and then select the Data tab).

2. Using SEQUENCE in an INSERT statement.

In the previous exercises, you have entered the primary key value manually in the INSERT statements. In the case where SEQUENCE is available, you should use the sequence mechanism to generate the value of the primary key.

TASKS

Create a sequence for the STUDENT table called STUDENT_SEQ

- Create a sequence for the STUDENT table called STUDENT_SEQ that starts at 11111115 and increases by 1.
- Check that the sequence exists in two ways (using SQL and browsing your SQL Developer connection objects).

Add a new student (MICKEY MOUSE)

- Use the student sequence - pick any STU_DOB you wish.
- Check that your insert worked.
- Add an enrolment for this student to the unit FIT5132 in semester 2 2014.

3. Advance INSERT.

We have learned how to add data into the database in the previous exercises through the use of INSERT statements. In those exercises, the INSERT statements were created as a single script assuming that data is all added at the same time, such as at the beginning when the tables are created. On some occasions, new data is added after some data already exists in the database. In this situation, it is a good idea to use a combination of INSERT and SELECT statements.

To illustrate this, assume we want to add vendors and the products they supply into a set of tables represented by:

A suitable schema would be:

```
DROP TABLE PRODUCT PURGE;
DROP TABLE VENDOR PURGE;
DROP SEQUENCE PRODUCT_prod_no_SEQ;
DROP SEQUENCE VENDOR_vendor_id_SEQ;

CREATE TABLE PRODUCT
  (
    prod_no          NUMBER (4) NOT NULL ,
```

```
    prod_name        VARCHAR2 (50) NOT NULL ,
    prod_price       NUMBER (6,2) NOT NULL ,
    prod_stock       NUMBER (3) NOT NULL ,
    VENDOR_vendor_id NUMBER (3) NOT NULL
  ) ;

ALTER TABLE PRODUCT ADD CONSTRAINT PRODUCT_PK PRIMARY KEY ( prod_no ) ;

CREATE TABLE VENDOR
  (
    vendor_id    NUMBER (3) NOT NULL ,
    vendor_name  VARCHAR2 (50) NOT NULL ,
    vendor_phone CHAR (10) NOT NULL
  ) ;

ALTER TABLE VENDOR ADD CONSTRAINT VENDOR_PK PRIMARY KEY ( vendor_id ) ;

ALTER TABLE VENDOR ADD CONSTRAINT VENDOR__UN UNIQUE ( vendor_name ) ;

ALTER TABLE PRODUCT ADD CONSTRAINT PRODUCT_VENDOR_FK FOREIGN KEY (
VENDOR_vendor_id ) REFERENCES VENDOR ( vendor_id ) ON
DELETE CASCADE ;

CREATE SEQUENCE PRODUCT_prod_no_SEQ START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE VENDOR_vendor_id_SEQ START WITH 1 INCREMENT BY 1;
```

There are two ways how we can perform the INSERT.

1. Use the nextval and currval of the sequences.

```
-- Add Vendor 1 and the products they supply
 insert into vendor values (VENDOR_vendor_id_SEQ.nextval,
        'Western Digital', '1234567890');
 insert into product values (PRODUCT_prod_no_SEQ.nextval,
        '2TB My Cloud Drive',195,5,VENDOR_vendor_id_SEQ.currval);
 insert into product values (PRODUCT_prod_no_SEQ.nextval,
        '1TB Portable Hard Drive',76,4,VENDOR_vendor_id_SEQ.currval);
 insert into product values (PRODUCT_prod_no_SEQ.nextval,
        'Live Media Player',119,2,VENDOR_vendor_id_SEQ.currval);
 commit;
 -- Add Vendor 2 and the products they supply
 insert into vendor values (VENDOR_vendor_id_SEQ.nextval,'Seagate',
        '2468101234');
 insert into product values (PRODUCT_prod_no_SEQ.nextval,
        '2TB Desktop Drive',94,12,VENDOR_vendor_id_SEQ.currval);
 insert into product values (PRODUCT_prod_no_SEQ.nextval,
        '4TB 4 Bay NAS',76,4,VENDOR_vendor_id_SEQ.currval);
 insert into product values (PRODUCT_prod_no_SEQ.nextval,
        '2TB Central Personal Storage' ,169,5,
      VENDOR_vendor_id_SEQ.currval);
 commit;
```

2. Use the nextval in combination with the SELECT statement.

- Add a new product for a vendor at a subsequent time (vendor names will be unique - note the U in the model above and the vendor_un constraint in the schema)

```
insert into product values (PRODUCT_prod_no_SEQ.nextval,
        'GoFlex Thunderbolt Adaptor',134,2,
        (select vendor_id from vendor where vendor_name = 'Seagate'));
```

The same concept can be used with other data manipulation statements such as UPDATE and DELETE.

TASKS

- A new student has started a course and needs to enrol into "Introduction to databases". Enter the new student's details and his/her enrolment to the database using the nextval in combination with the SELECT statement.
    - You should not do a manual lookup to find the unit code of the "Introduction to databases".

4. Creating table and inserting data as a single SQL statement.

A table can also be created based on an existing table, and immediately populated with contents by using a SELECT statement within the CREATE TABLE statement.

For example to create a table called FIT5132_STUDENT which contains the enrolment details of all students who have been or are currently enrolled in FIT5132, we would use:

```
create table FIT5132_STUDENT
as select *
from enrolment
where unit_code = 'FIT5132';
```

TASKS

- Create a table called FIT5111_STUDENT. The table should contain all enrolments for the unit FIT51111.
- Check it exists.
- List the contain of the table.

7
Tutorial 7: Data Maintenance and Transactions

Transaction Management

In these exercises, you will examine the issues involved in updating shared data.

You will work in pairs. Suppose one user1 is User1, and the other is User2. *Replace User1 and User2 with your respective usernames when reading the tutorial exercises*.

User1 will create a table called account which will be shared with User2, that is, both users will be allowed to select data from the table and also update data in the table. This table keeps the account balances of customers, where each customer is identified by a unique id.

Q1. User1 should create the account table. The table will have 2 attributes, id and balance. Both attributes will have datatype number. After creating the table, User1 should insert two rows of data so that the table looks as below:

	ID	BALANCE
1	1	100
2	2	200

Q2. User1 can now make the account table available to User2 using the following command:

```
grant select, update on account to User2;
```

Q3. In order for User2 to access the account table, they would normally have to prefix the account table with the name of the owner, e.g:

```
select * from User1.account;
```

However, we can remove the need to do this if we create a synonym (essentially a system maintained alias for the table) as follows (User2 issues this command):

```
create synonym account for User1.account;
```

Q4. Make sure both users have the autocommit feature turned OFF - i.e. both users should issue the command

```
set autocommit off
```

Q5. Now, try the following (maintain the order of the operations).

- User1 updates the balance of customer 1 from 100 to 110 (without issuing a commit).
- User2 views the contents of the account table (do they see the new value? - if not, why not?)
- User1 issues a commit command.

- User2 views the contents of the account table (do you notice any difference?)

Explain what is happening in the results of the above queries, in the context of atomic transactions.

Q6. Now we will try and see what happens when we try some concurrent updates of the table (keep the order of transactions the same as below)

- User1 updates the balance of customer 2 from 200 to 150 (without issuing a commit).
- User2 tries to update the balance of customer 2 to 100 (what happens?)

Explain what is happening here. What should be done to allow the User2 update to proceed?

Q7. Now try the following:

- User1 updates the balance of customer 2 from 200 to 150 (without issuing a commit).
- User2 tries to update the balance of customer 1 to 125 (what happens?)

How does this differ from the results of the transactions in part 6? What does this tell you about the granularity of locking in Oracle? What must be done in order for the results of both updates to be visible to both users?

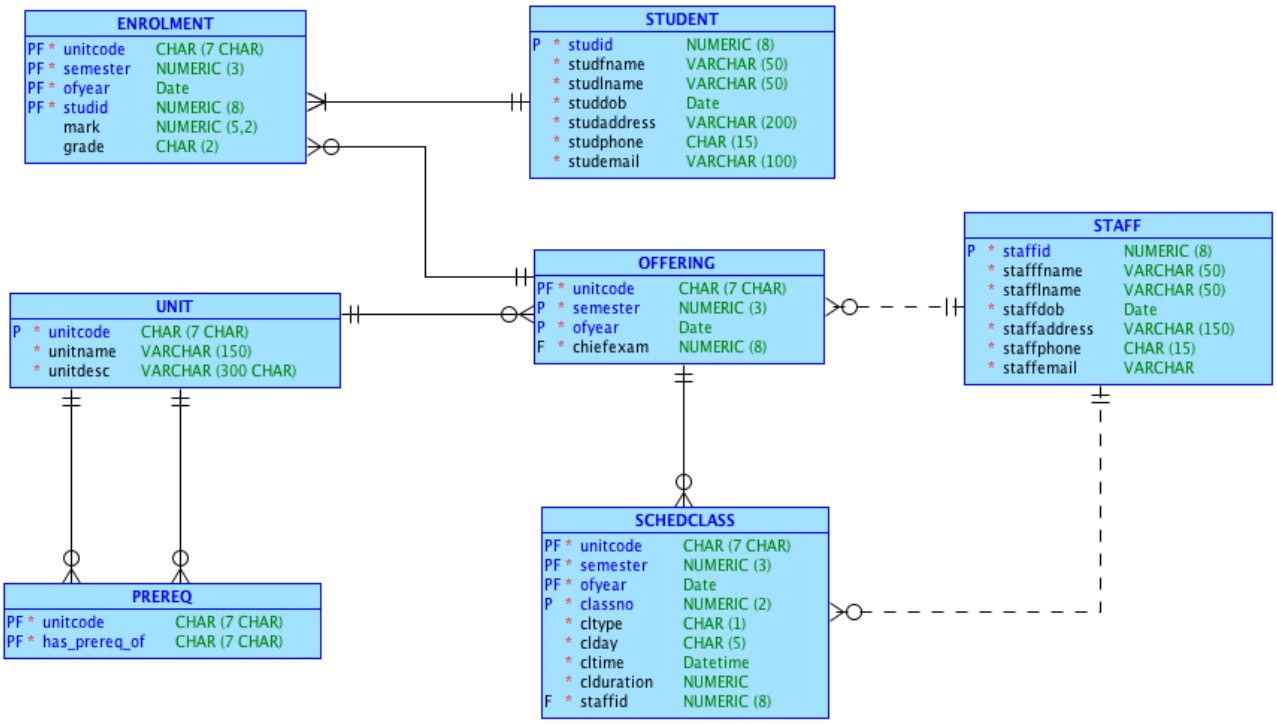
Q8. Try and generate a deadlock between the two users (hint: you will need to set up another shared table). Remember that a deadlock occurs when User1 holds a lock on table A and requests a lock on table B, but table B is locked by User2 who is also requesting a lock on table A.

8
Tutorial 8: SQL Part I - SQL Basic

The following exercises will allow you to be familiar with writing basic SQL statements. Two main concepts that you need to learn from these exercises are:

- Using the WHERE clause to retrieve selected rows from a table.
- Using the WHERE clause to join two or more tables.

Use the UNIVERSITY database to complete the exercises. Figure 1.0 depicts the data model for the UNIVERSITY database.



University data model

For students using the Monash Oracle database, to complete these exercises, you do not need to run the supplied DDL scripts in your account. A UNIVERSITY database has been created under the user "UNI". To use the database you need to include the prefix "UNI" to the table names that you use in an SQL statement. For example, if you want to retrieve data from UNIT table you need to write:

```
SELECT unitcode, unitname
FROM uni.unit;
```

instead of

```
SELECT unitcode, unitname
FROM unit;
```

For students who are using a local installation of Oracle, the DDL schema and the SQL insert statements for the UNIVERSITY database are provided in the university.zip file available from Moodle.
This week we make use of Oracle dates - to use these correctly you should note the following:

- The Oracle date data type contains both date and time, however you can choose to use just a date, just a time, both or parts of a date depending on the format strings used
- to_date: converts from a string to a date according to a format string
- to_char: converts from a date to a string according to a format string

The Oracle documentation links are:

- Format models (http://goo.gl/6IFTqP)
- to_date (http://goo.gl/hhrCAo)
- to_char (http://goo.gl/7yvmwp)

Part A. Retrieving data from a single table

1. List all students and their details.
2. List all units and their details.
3. List all students who have the surname 'Smith'.
4. List the student's details for those students who have surname starts with the letter "S". In the display, rename the columns studfname and studlname to firstname and lastname.
5. List the student's surname, firstname and address for those students who have surname starts with the letter "S" and firstname contains the letter "i".
6. List the unit code and semester of all units that are offered in the year 2014.
   To complete this question you need to use the Oracle function to_char to convert the data type for the year component of the offering date into text. For example, to_char(ofyear,'yyyy') - here we are only using the year part of the date.
7. List the unit code of all units that are offered in semester 1 of 2014.
8. Assuming that a unit code is created based on the following rules:
   - The first three letters represent faculty abbreviation, eg FIT for the Faculty of Information Technology.
   - The first digit of the number following the letter represents the year level.List the unit details of all first year units in the Faculty of Information Technology.
9. List the unit code and semester of all units that were offered in either semester 1 or summer of 2013. Note: summer semester is recorded as semester 3.
10. List the student number, mark, unit code and semester for those students who have passed any unit in semester 1 of 2013.

Part B. Retrieving data from multiple tables.

Note: remember to use the foreign key and the primary key when joining two or more tables.

1. List the name of all students who have marks in the range of 60 to 70.
2. List all the unit codes, semester and name of the chief examiner for all the units that are offered in 2014.
3. List the name (firstname and surname), unit names, the year and semester of enrolment of all units taken so far.
4. List all the unit codes and the unit names and their year and semester offerings. To display the date correctly in Oracle, you need to use to_char function. For example, to_char(ofyear,'YYYY').
5. List the unit code, semester, class type (lecture or tutorial), day and time for all units taught by Albus Dumbledore in 2013. Sort the list according to the unit code.
6. Create a study statement for Mary Smith. A study statement contains unit code, unit name, semester and year study was attempted, the mark and grade.
7. Create a tutorial list for all units in semester 1, 2013. The list should include the unit code, unit name, class type, class no, student's firstname and student's surname. The list should be sorted

according to unit codes, and within the unit code, the list will be further sorted according class no.

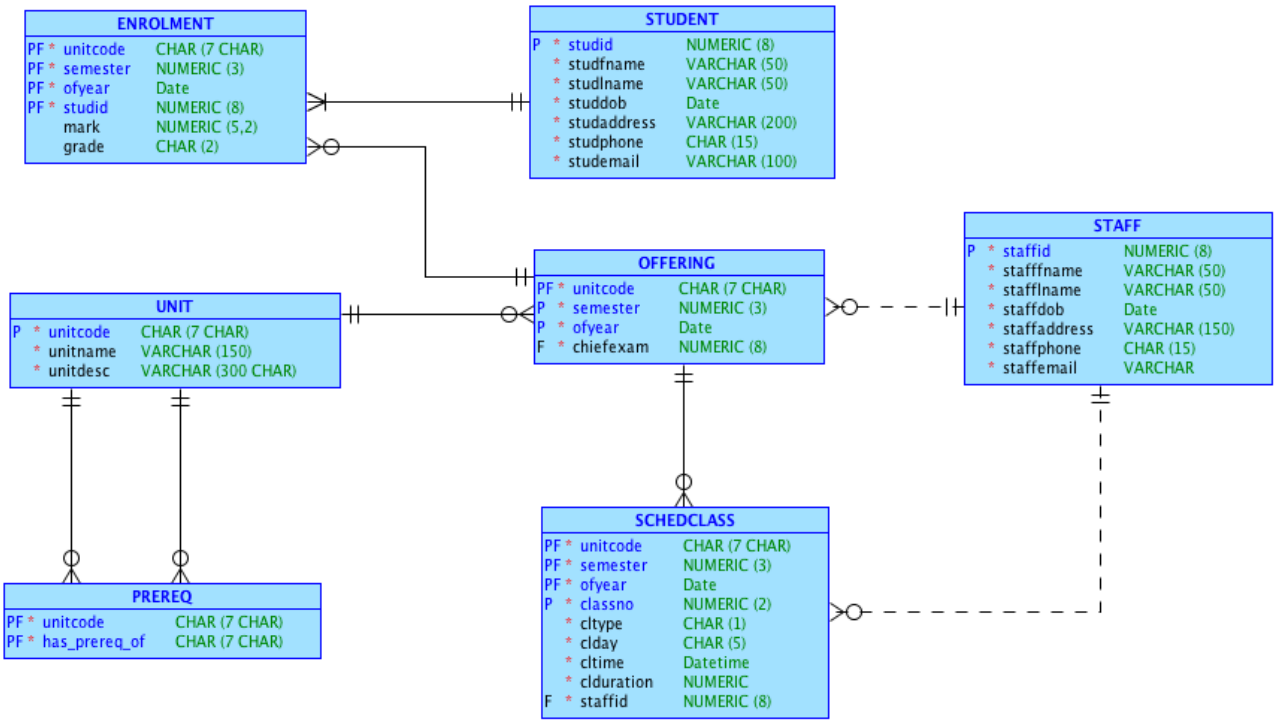
8. List the unit code, unit name and the unit code and unit name of the pre-requisite units of all units in the database.
9. List the unit code and unit name of the pre-requisite units of 'Advanced Data Management' unit.
10. Find all students (list their id, firstname and surname) who have a failed unit in the year 2013.
11. List the student name, unit code, semester and year for those students who do not have marks recorded.

9
Tutorial 9: SQL Part II - SQL Intermediate

The following exercises will allow you to be familiar with:

- All the SQL constructs used in last weeks exercises.
- Aggregate functions such as min, max, avg
- The GROUP BY clause, and
- Subqueries

This week we will continue to use the UNIVERSITY database model:



University data model

Use the Uni database from the previous exercises to complete the following queries.

1. Find the maximum mark for FIT1004 in semester 1, 2013.
2. Find the average mark of FIT1040 in semester 2, 2013.
3. List the average mark for each offering of FIT1040. In the listing, you need to include the year and semester number. Sort the result according to the year.
4. Find the number of students enrolled in the unit FIT1040 in the year 2013, under the following conditions:
    - Repeat students are counted each time
    - Repeat students are only counted once
5. Find the total number of enrolment per semester for each unit in the year 2013. The list should include the unitcode, semester and year. Order the list in increasing order of enrolment numbers.
6. Find the total number of prerequisite units for FIT2077.
7. Find the total number of prerequisite units for each unit. In the list, include the unitcode for which the count is applicable.

8. For each pre-requisite unit, calculate how many times it has been used as prerequisite. Include the name of the prerequisite unit in the listing .
9. Find the unit with the highest number of enrolments in a given offering in the year 2013.
10. Who is the oldest student in FIT1004? Display the student full name and the date of birth.
11. Find all students enrolled in FIT1004 in semester 1, 2013 who have scored more than the average mark of FIT1004 in the same offering? Display the students' name and the mark. Sort the list in the order of the mark from the highest to the lowest.

10
Tutorial 10: SQL Part III - SQL Advanced

This week we will continue to use the UNIVERSITY database model.

Q1. Find the total number of prerequisite units for each unit. Include in the list the unitcode of units that do not have prerequisite. Hint: use an outer join.

UNITCODE	NO_OF_PREREQ
FIT1004	0
FIT1040	0
FIT2077	2
FIT5131	1
FIT5132	0
FIT5136	1

Q2. Display unitcode and unitname for units that do not have prerequisite.

UNITCODE	UNITNAME
FIT1004	Introduction to Data Management
FIT1040	Programming Fundamental
FIT5132	Introduction to Databases

There are many approaches that you can take in writing an SQL statement to answer this query. You can use the SET OPERATORS, OUTER JOIN and a SUBQUERY. Write SQL statements based on *all* of these approaches.

Q3. List the unit code and the average mark for each offering. Round the average to 2 digits after the decimal points. If the average result is 'null', display the average as 0.00. All values must be shown with two decimal digits:

UNITCODE	YEAR	SEMESTER	AVERAGE
FIT5131	2014	1	0.00
FIT2077	2013	2	64.50
FIT1004	2013	1	65.50
FIT1040	2013	1	67.80
FIT1040	2013	2	71.83
FIT5132	2013	1	74.00
FIT5132	2013	3	74.00
FIT5136	2013	2	75.50
FIT1004	2013	2	75.83
FIT5131	2013	1	77.50

11
Tutorial 11: Database Web Interfaces