FIT2094-FIT3171 2019 S1 -- Week 5 eBook

Credits and Copyrights:
Authors: FIT3171 2018 S2 UG Databases
FIT Database Teaching Team
Maria Indrawan, Manoj Kathpalia, Lindsay Smith, et al

Change Log:

- Formatted and Imported from Alexandria March 2019.

5.0. Logical Modelling

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.

Full credits for the detailed writeup and screenshots to Lindsay Smith and team.

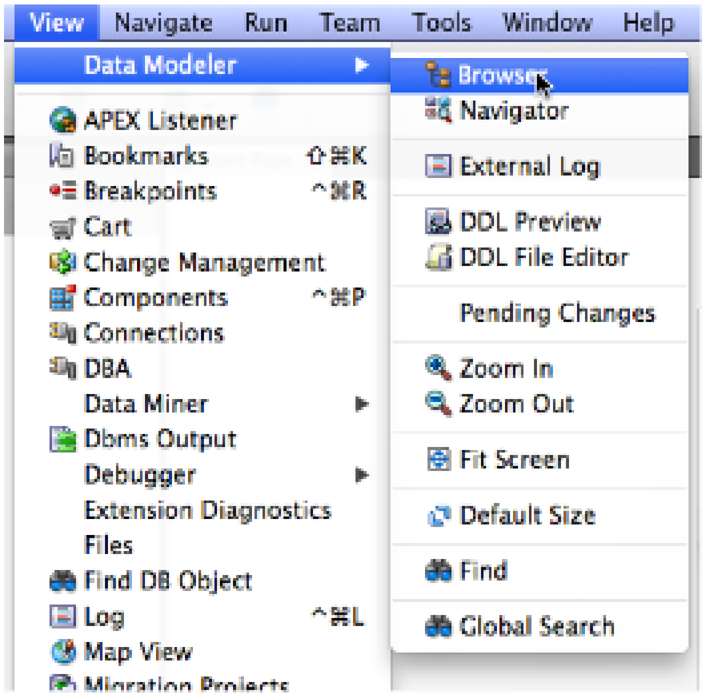
Week 5 Task A: Using SQL Developer Data Modeler

This software is a commercial level tool with an extensive set of features including support for Subversion versioning and source version 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.

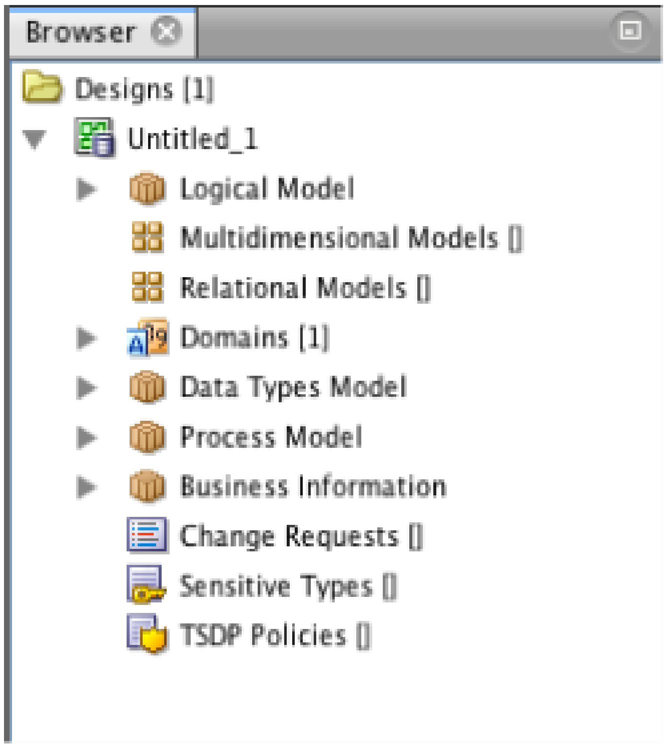
SQL Developer Data Modeler begins with a (relational) **LOGICAL** model and then creates what it calls a RELATIONAL model from which the schema file can be generated. The relational model is essentially a graphical representation of the physical model.

5.0.1. 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:

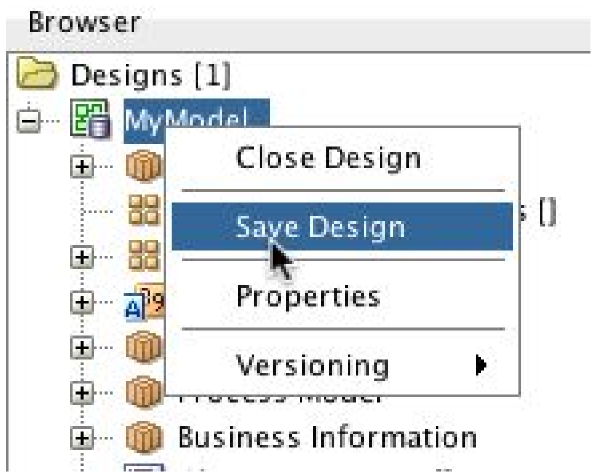


To begin creating a logical model, **in Browser → right-click Logical Model → select Show.**

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 what they represent.

You should regularly save your design – **in Browser → right click on the design** (here Untitled_1 as the design has not been saved as yet) → **Save Design** and select a save location. After the first save the design will be named.

IMPORTANT: 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.



5.0.2. Configuring SQL Developer Preferences - Important.

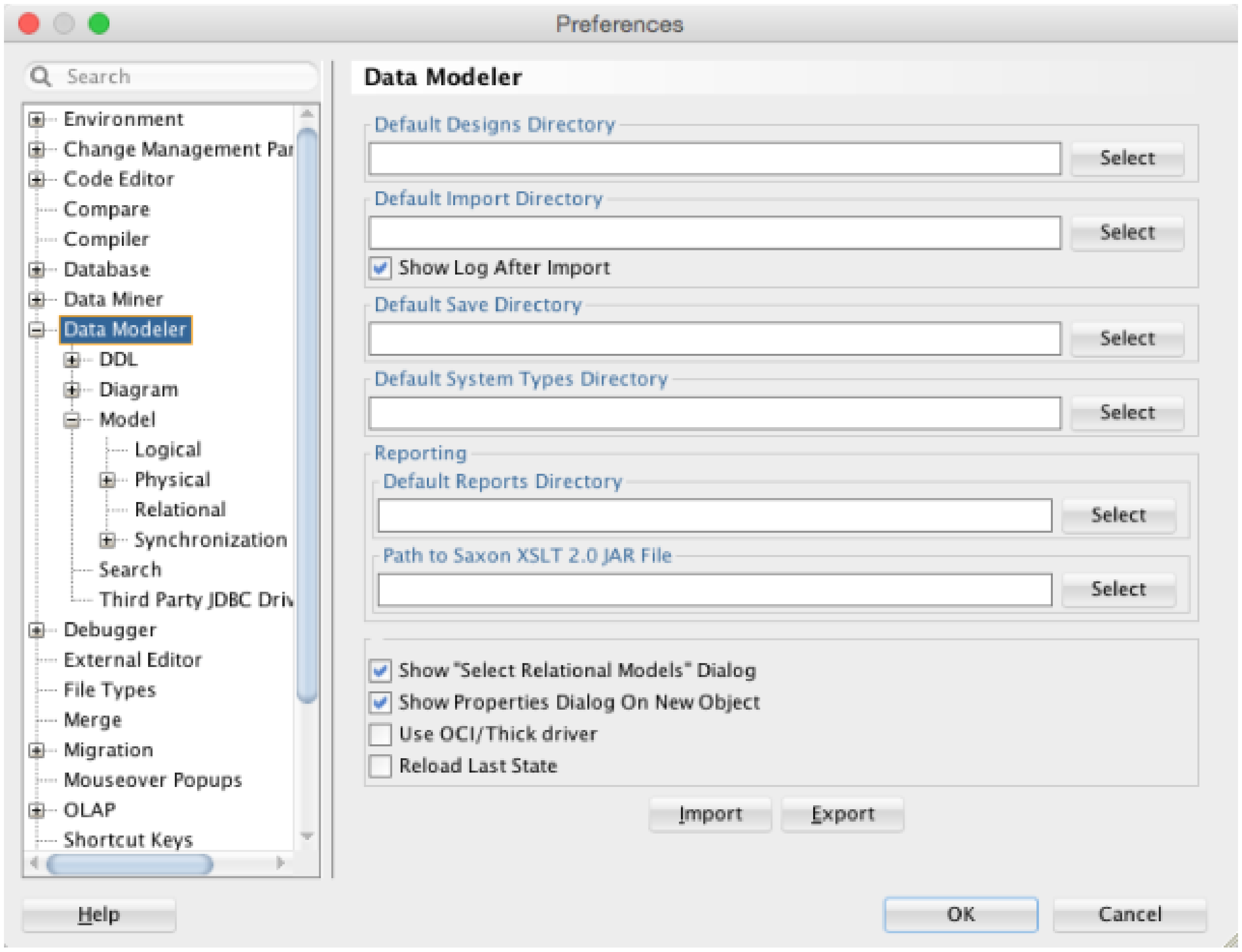
WARNING: Please read and complete the steps in Section 5.0.2 carefully to avoid future issues!

A number of features should be configured using the SQL Developer preferences

(Windows: **Tools** → **Preferences;**

Mac OSX: **Oracle SQL Developer** → **Preferences**).

Then select the **Data Modeler** from the left panel.



Data Modeler

Default Designs Directory

Default Import Directory

[✓] Show Log After Import

Default Save Directory

Default System Types Directory

Reporting

Default Reports Directory

Path to Saxon XSLT 2.0 JAR File

[✓] Show "Select Relational Models" Dialog

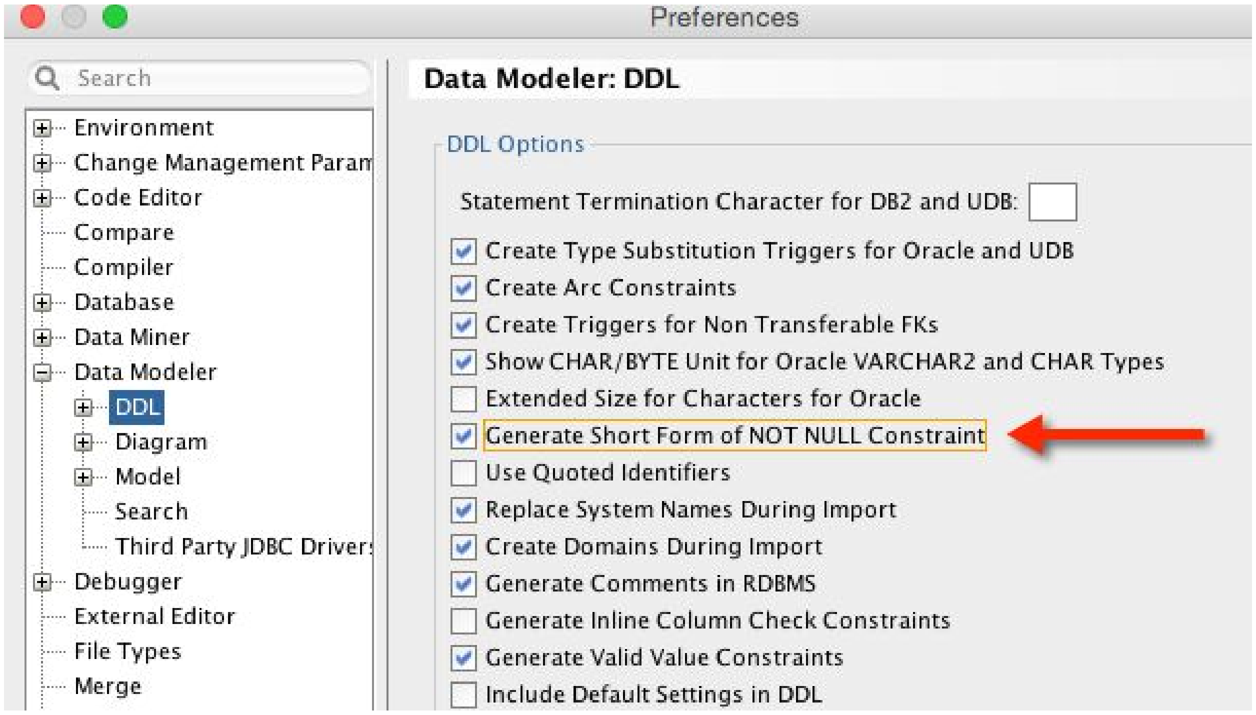
[✓] Show Properties Dialog On New Object

[ ] Use OCI/Thick driver

[ ] Reload Last State

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.



Data Modeler: DDL

DDL Options

Statement Termination Character for DB2 and UDB:

- [✓] Create Type Substitution Triggers for Oracle and UDB
- [✓] Create Arc Constraints
- [✓] Create Triggers for Non Transferable FKs
- [✓] Show CHAR/BYTE Unit for Oracle VARCHAR2 and CHAR Types
- [ ] Extended Size for Characters for Oracle
- [✓] Generate Short Form of NOT NULL Constraint
- [ ] Use Quoted Identifiers
- [✓] Replace System Names During Import
- [✓] Create Domains During Import
- [✓] Generate Comments in RDBMS
- [ ] Generate Inline Column Check Constraints
- [✓] Generate Valid Value Constraints
- [ ] Include Default Settings in DDL

(b) Select **Diagram**, **Relational Model** from the left panel; and choose "**Foreign Key Arrow Direction**" as below:

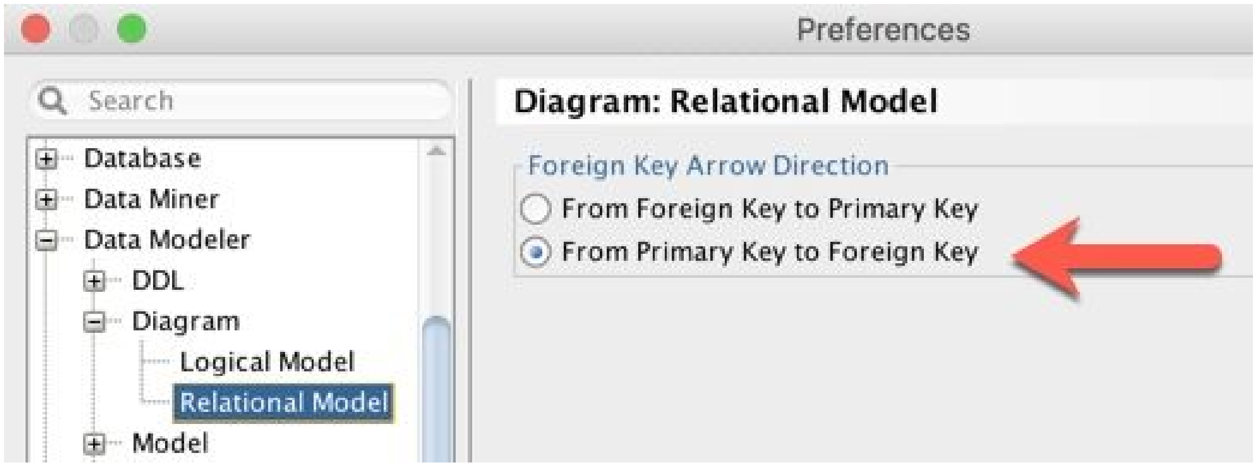
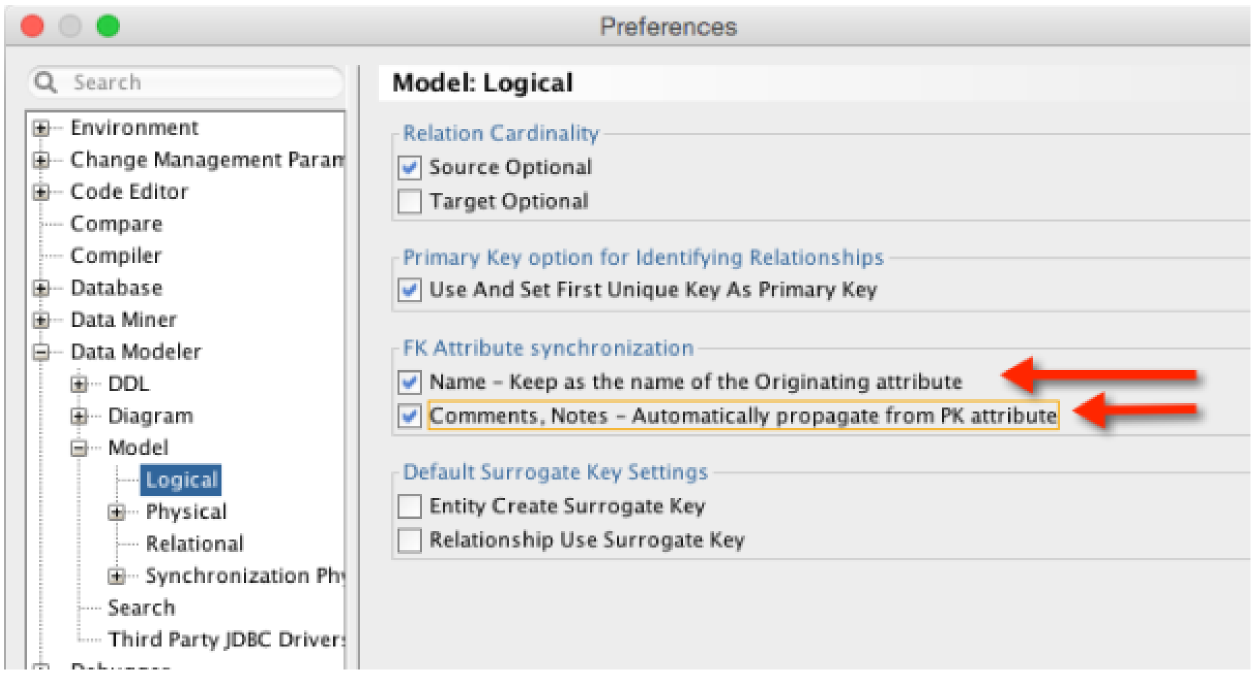


Diagram: Relational Model

Foreign Key Arrow Direction

- ○ From Foreign Key to Primary Key
- [✓ From Primary Key to Foreign Key]

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



Model: Logical

Relation Cardinality
- [✓] Source Optional
- [ ] Target Optional

Primary Key option for Identifying Relationships
- [✓] Use And Set First Unique Key As Primary Key

FK Attribute synchronization
- [✓] Name – Keep as the name of the Originating attribute
- [✓] Comments, Notes – Automatically propagate from PK attribute

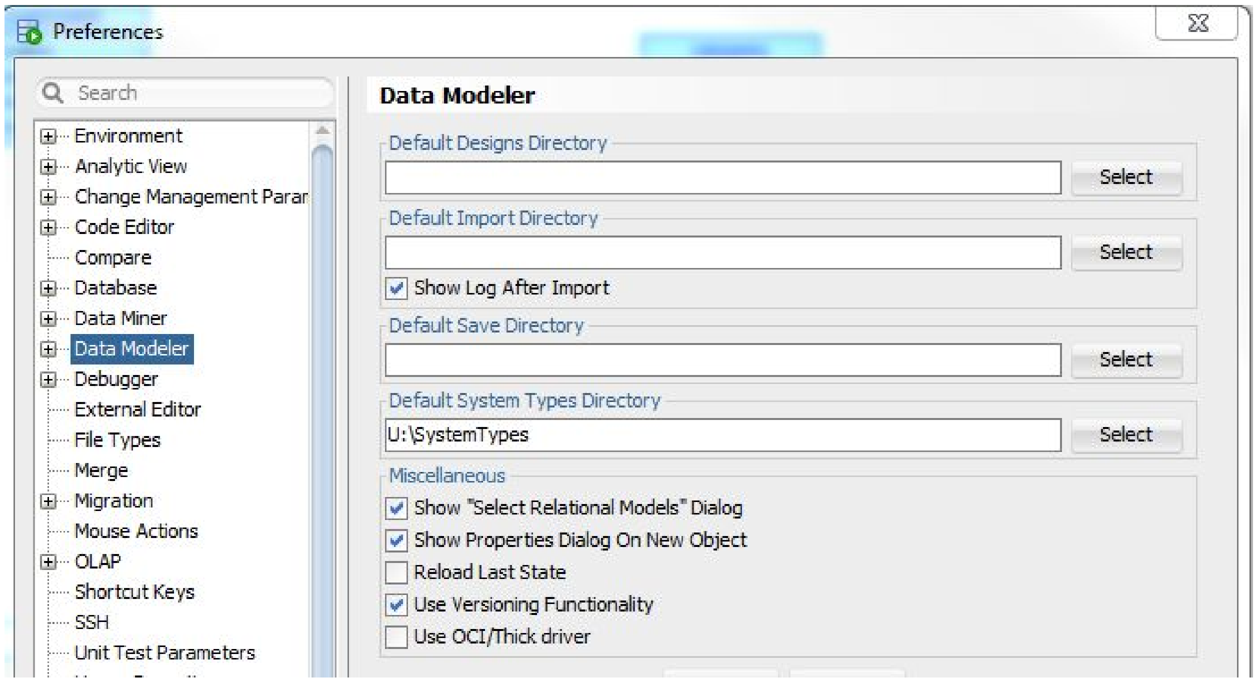
Default Surrogate Key Settings
- [ ] Entity Create Surrogate Key
- [ ] Relationship Use Surrogate Key

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.

If you are working on a **Monash on-campus PC** please set up a folder to save the Modeler System Files:

- In Windows' File Explorer, connect to your mapped drive (U:) and create a folder called `SystemTypes` (note that this **folder name MUST NOT contain spaces**)

- In the Data Modeler preferences (recall the first diagram in Section 5.0.2) point the "**Default System Types Directory**" to this newly created folder.



Preferences

Data Modeler

Default Designs Directory:

Default Import Directory:

☑ Show Log After Import

Default Save Directory:

Default System Types Directory: U:\SystemTypes

Miscellaneous

☑ Show "Select Relational Models" Dialog

☑ Show Properties Dialog On New Object

☐ Reload Last State

☑ Use Versioning Functionality

☐ Use OCI/Thick driver

- This is necessary since Data Modeler needs to write the logical data types you define when it saves a project. **If the default System Types Directory is a location Data Modeler cannot write to**, e.g. `c:\program files`, **an error message will be generated when you save your model.**

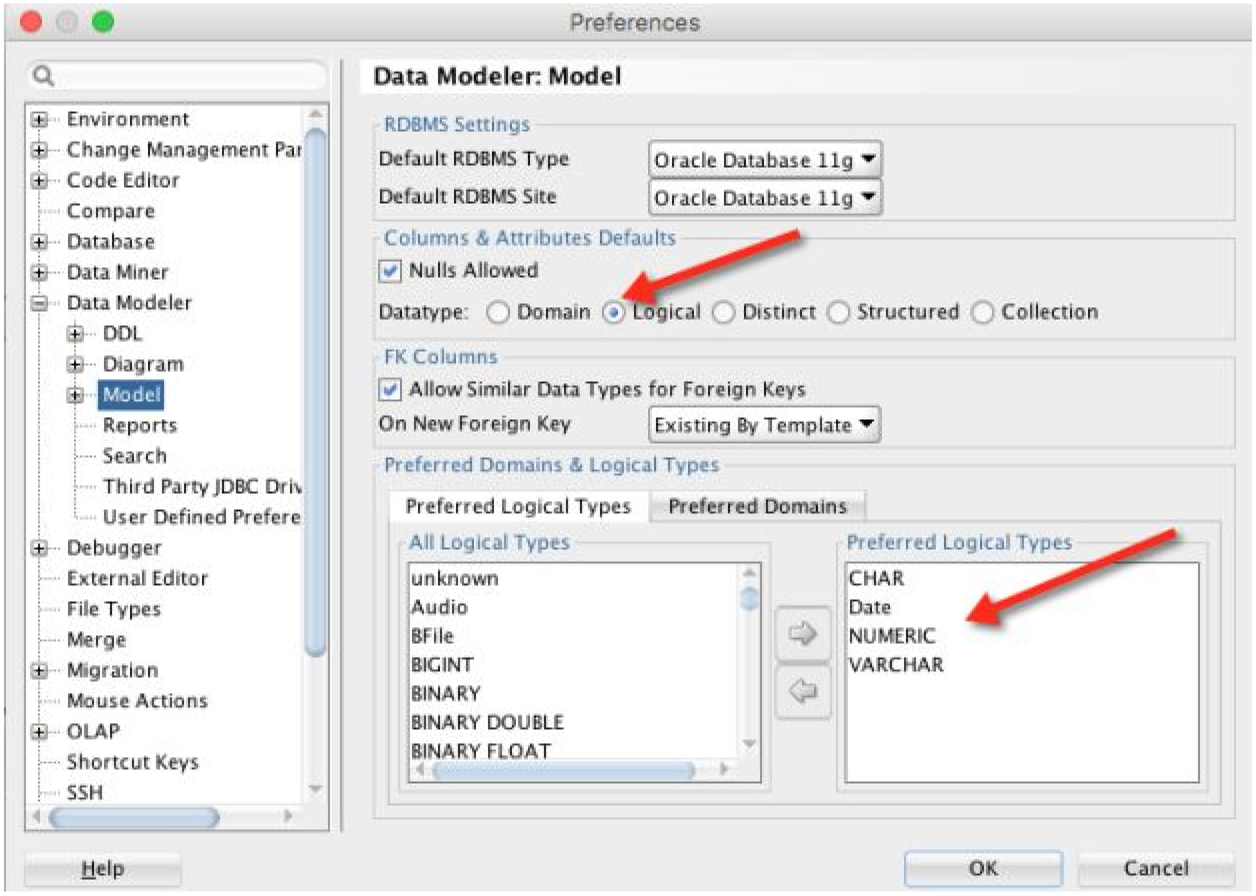
Finally, in SQL Developer, Attributes on the logical model have a number of possible data types, the main ones for us being "Domain" and "Logical":

- 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 the entry of attributes by restricting Data Modeler to logical and types and a preferred range of types.

In your **SQL Developer Preferences → Data Modeler (left panel) → Model**, set:


Preferences

Data Modeler: Model

RDBMS Settings
Default RDBMS Type: Oracle Database 11g
Default RDBMS Site: Oracle Database 11g

Columns & Attributes Defaults
[✓] Nulls Allowed
Datatype: Domain [✓ Logical] Distinct Structured Collection

FK Columns
[✓] Allow Similar Data Types for Foreign Keys
On New Foreign Key: Existing By Template

Preferred Domains & Logical Types
Preferred Logical Types | Preferred Domains

All Logical Types: unknown, Audio, BFile, BIGINT, BINARY, BINARY DOUBLE, BINARY FLOAT

Preferred Logical Types: CHAR, Date, NUMERIC, VARCHAR

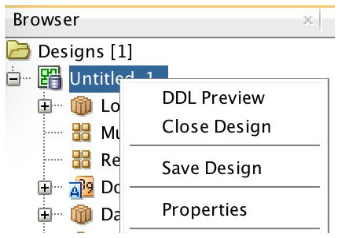
The **‘Preferred Logical Types’ list box*** are populated by selecting the item in the ‘All Logical Types” and clicking on the **right-pointing arrow [➞]** to ‘move’ it to Preferred.

The usefulness of this* is discussed later in Section 5.0.3.

5.0.3. 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}`

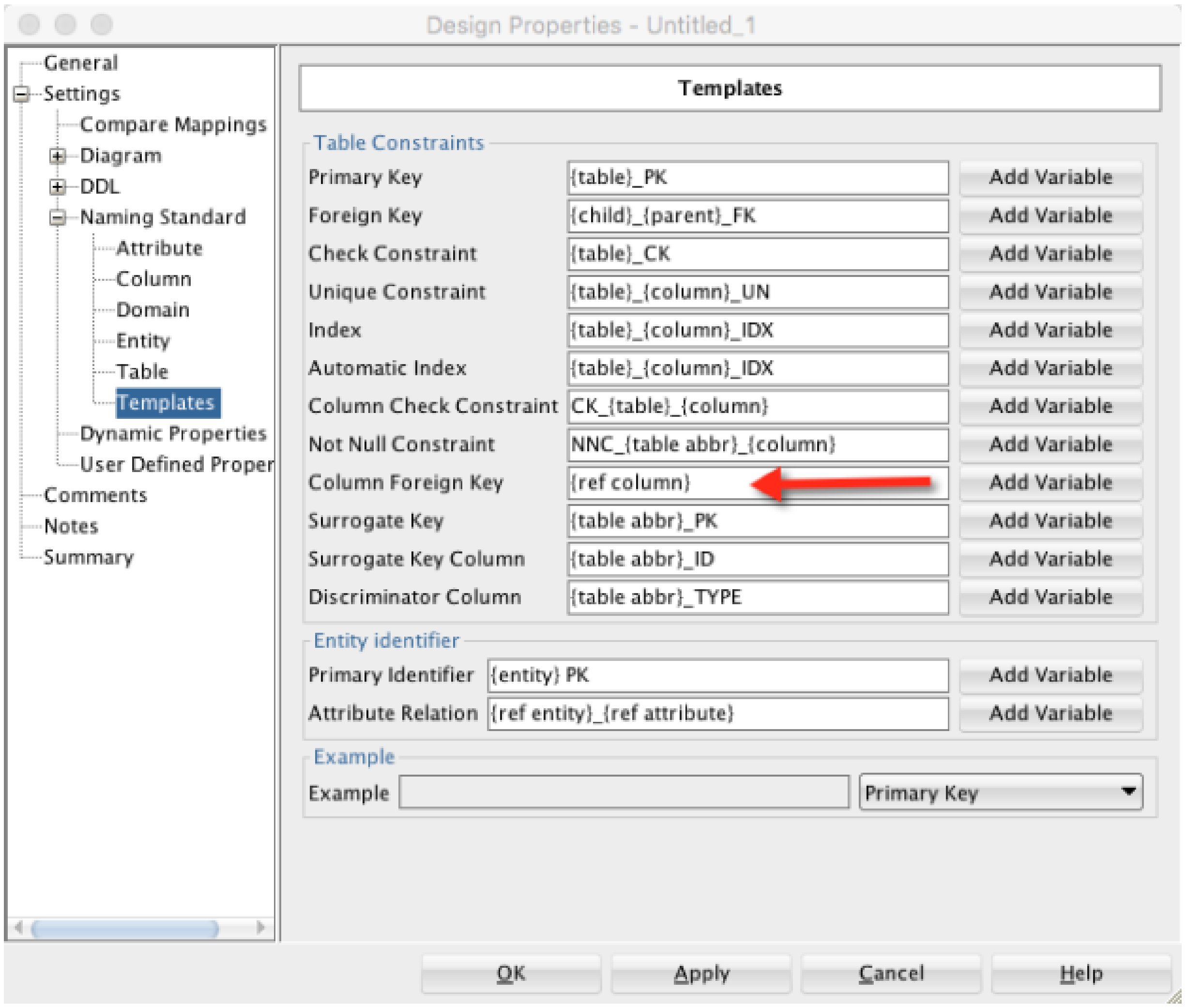


Table Constraints

Primary Key	{table}_PK	Add Variable
Foreign Key	{child}_{parent}_FK	Add Variable
Check Constraint	{table}_CK	Add Variable
Unique Constraint	{table}_{column}_UN	Add Variable
Index	{table}_{column}_IDX	Add Variable
Automatic Index	{table}_{column}_IDX	Add Variable
Column Check Constraint	CK_{table}_{column}	Add Variable
Not Null Constraint	NNC_{table abbr}_{column}	Add Variable
Column Foreign Key	{ref column}	Add Variable
Surrogate Key	{table abbr}_PK	Add Variable
Surrogate Key Column	{table abbr}_ID	Add Variable
Discriminator Column	{table abbr}_TYPE	Add Variable

Entity identifier

Primary Identifier	{entity} PK	Add Variable
Attribute Relation	{ref entity}_{ref attribute}	Add Variable

Example

Example		Primary Key

This will result in FK's being named based on the PK attribute they were copied from. If this is not set the FK names will have the form **TABLE_attribute** – e.g. **CUSTOMER_custno.** However, we wish the FK to be simply **custno** (without the TABLE name).

5.0.3.1. Adding entities and attributes...

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

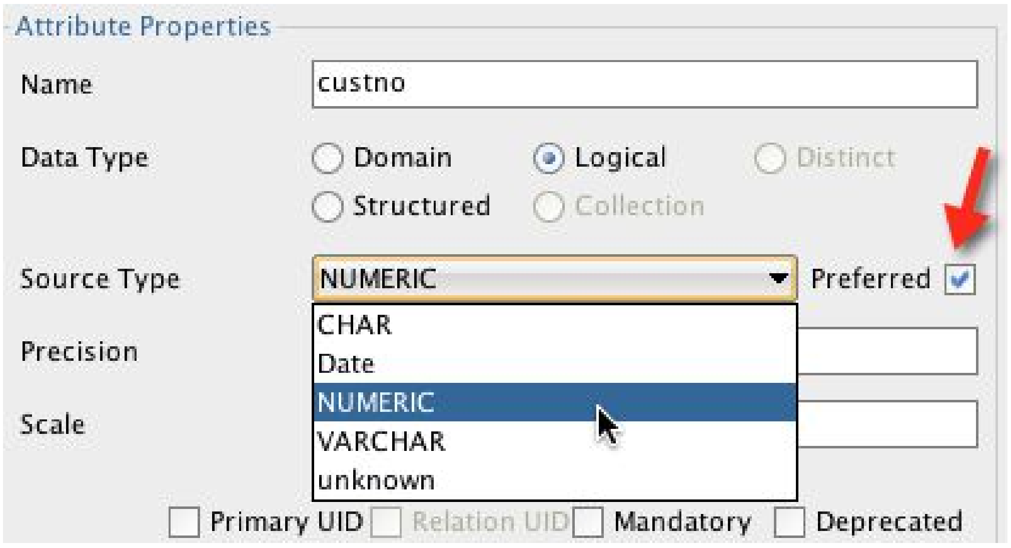
- **CUSTOMER** – customer number, name, address, phone number
- **ORDER** – order number, order date, customer number and for each product ordered the quantity ordered and the total line price
- **PRODUCT** – product number, product description and product unit price.

On your logical model, find the toolbar option to add an entity [ ]. Add an entity named **`CUSTOMER`** (name the entity under the **General** features - found in the left panel - in the Entity Properties dialog box) and then select the **Attributes** feature (left panel).

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)

When adding an attribute, you'll see a screen like below.


Attribute Properties

Name	custno
Data Type	○ Domain [✓ ◉ Logical] ○ Distinct ○ Structured ○ Collection
Source Type	NUMERIC ▾ Preferred ☑
Precision	
Scale	

CHAR
Date
NUMERIC
VARCHAR
unknown

☐ Primary UID ☐ Relation UID ☐ Mandatory ☐ Deprecated

Also, remember the **'Preferred Logical Types' list box*** discussed under Section 5.0.2? If you check the 'Preferred' checkbox, it will only list the ones you have selected as 'preferred' in Section 5.0.2 -- it acts like a 'favourites' / 'bookmarks' mechanism to help you select the ones you use the most. (Clearly you may modify the set preferred data types to suit your needs).

For documentation purposes - for each attribute add a meaningful description of the attribute under the attribute option - "**Comments in the RDBMS**" (Attribute Properties → right panel). **Remember it is very good practice to do so, not just in the unit, but also in industry, to have good documentation!**

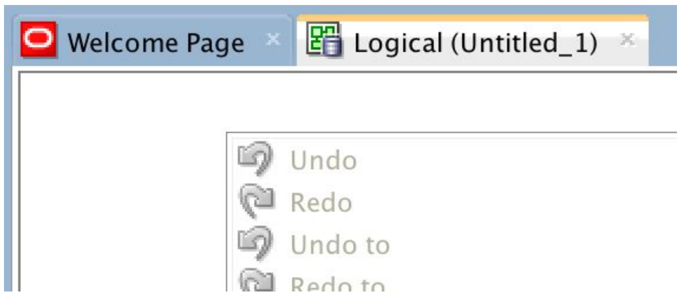
Then add an **`ORDERS`** entity with attributes:

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

Note here that the entity is being named as ORDERS since ORDER is a reserved SQL word – remember that as a general rule entity names must not be pluralised.

5.0.3.2. Display options for logical models...

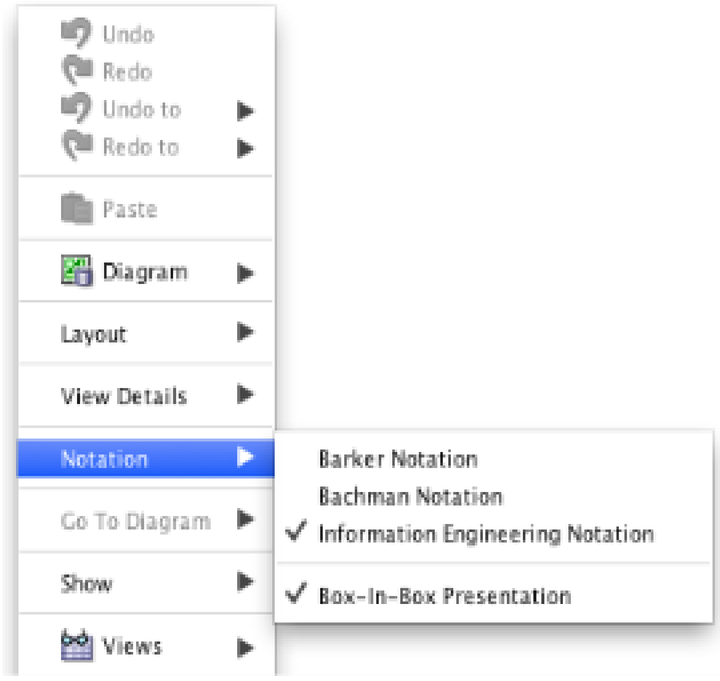
In the main **SQL Developer window → right click on a blank space** in the logical model to bring up a menu. It should look like so, with ‘Undo’, ‘Redo’, etc at the top of the menu.



NB: Don’t right click on an Entity (rectangle, e.g. CUSTOMER) - else you will bring up a DIFFERENT menu!

For all the logical models we create, you should set some display options.

- **Notation → “Information Engineering Notation”,**



- **View Details → “Attributes”**, and
- **Show → “Labels” and “Legend”**

IMPORTANT: When modelling with SQL Developer, students are required to include the “Legend” on all models (the panel may be moved to fit your model layout). Models submitted without a legend will not be graded.

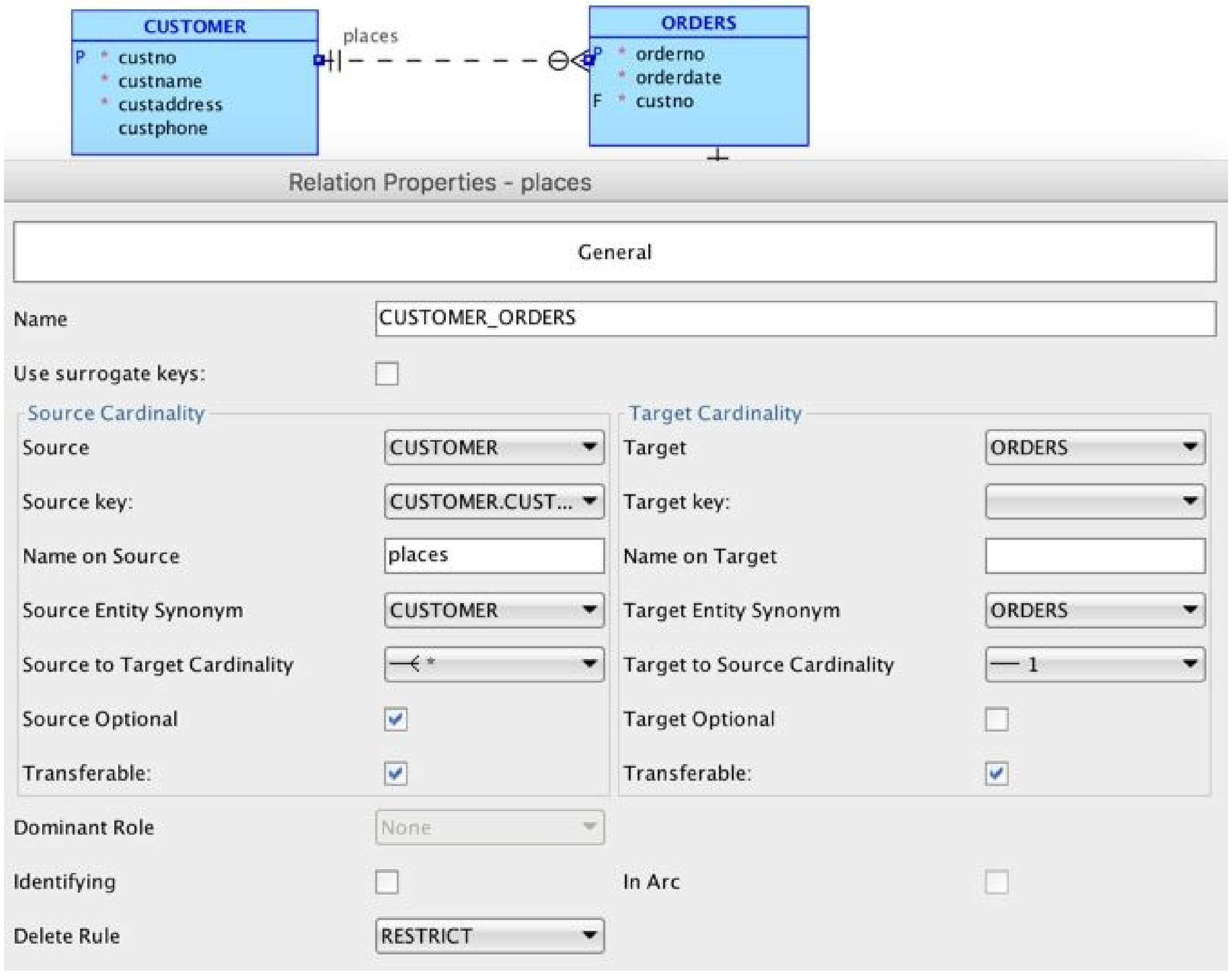
5.0.3.3. Adding relationships and the rest of the entities…

Recall that there is a range of new icons on the main toolbar. The ones with green arrows will add relationships between entities.

Now, add a **1:N Relationship** between **CUSTOMER** and **ORDERS** – click the correct toolbar option, then **click in CUSTOMER** (the parent) and then **click in ORDERS**. In the **General Relations** Properties name the relationship “`CUSTOMER_ORDERS`”. For this unit, we name relationships in the one to many direction using the entities at each end – if the combined name is over 30 characters in length you should use an abbreviation of each entity name, such as `CUST_ORD`.

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

Refer to the following diagrams for clarification.



Relation Properties - places

General

Name	CUSTOMER_ORDERS
Use surrogate keys:	☐

Source Cardinality

Source	CUSTOMER
Source key:	CUSTOMER.CUST...
Name on Source	places
Source Entity Synonym	CUSTOMER
Source to Target Cardinality	—< *
Source Optional	✓
Transferable:	✓

Target Cardinality

Target	ORDERS
Target key:	
Name on Target	
Target Entity Synonym	ORDERS
Target to Source Cardinality	— 1
Target Optional	☐
Transferable:	✓

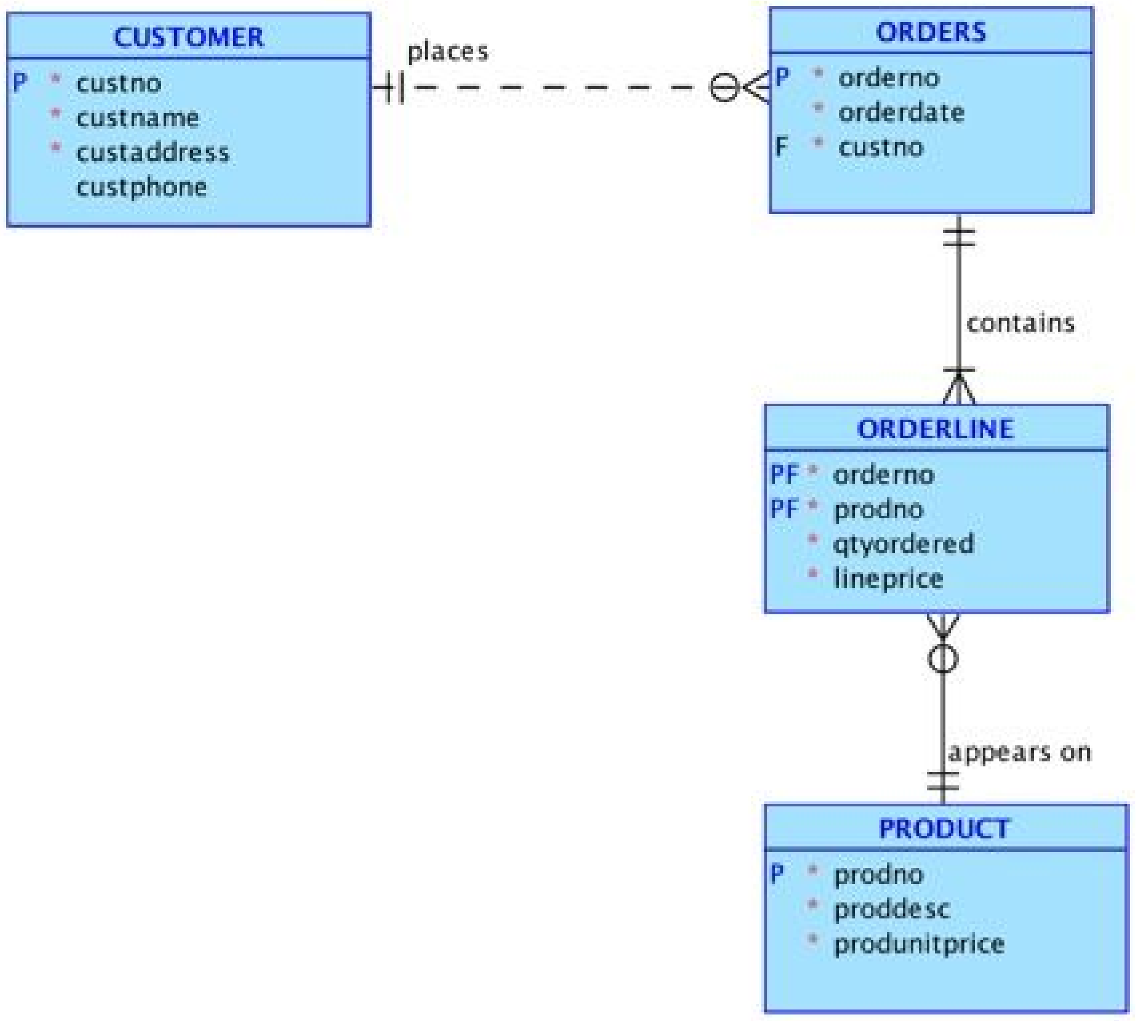
Dominant Role	None		
Identifying	☐	In Arc	☐
Delete Rule	RESTRICT		

Confirm that the relations(hips) **CUSTOMER_ORDERS** is set up as above.

Now, proceed and add the **`PRODUCT`** entity - refer diagram below.

Add a new entity **`ORDERLINE`** and then connect this with **`ORDERS`** and **`PRODUCT`** via 1:N Identifying relations(hips) - refer diagram below.

Your final logical model will have the form:

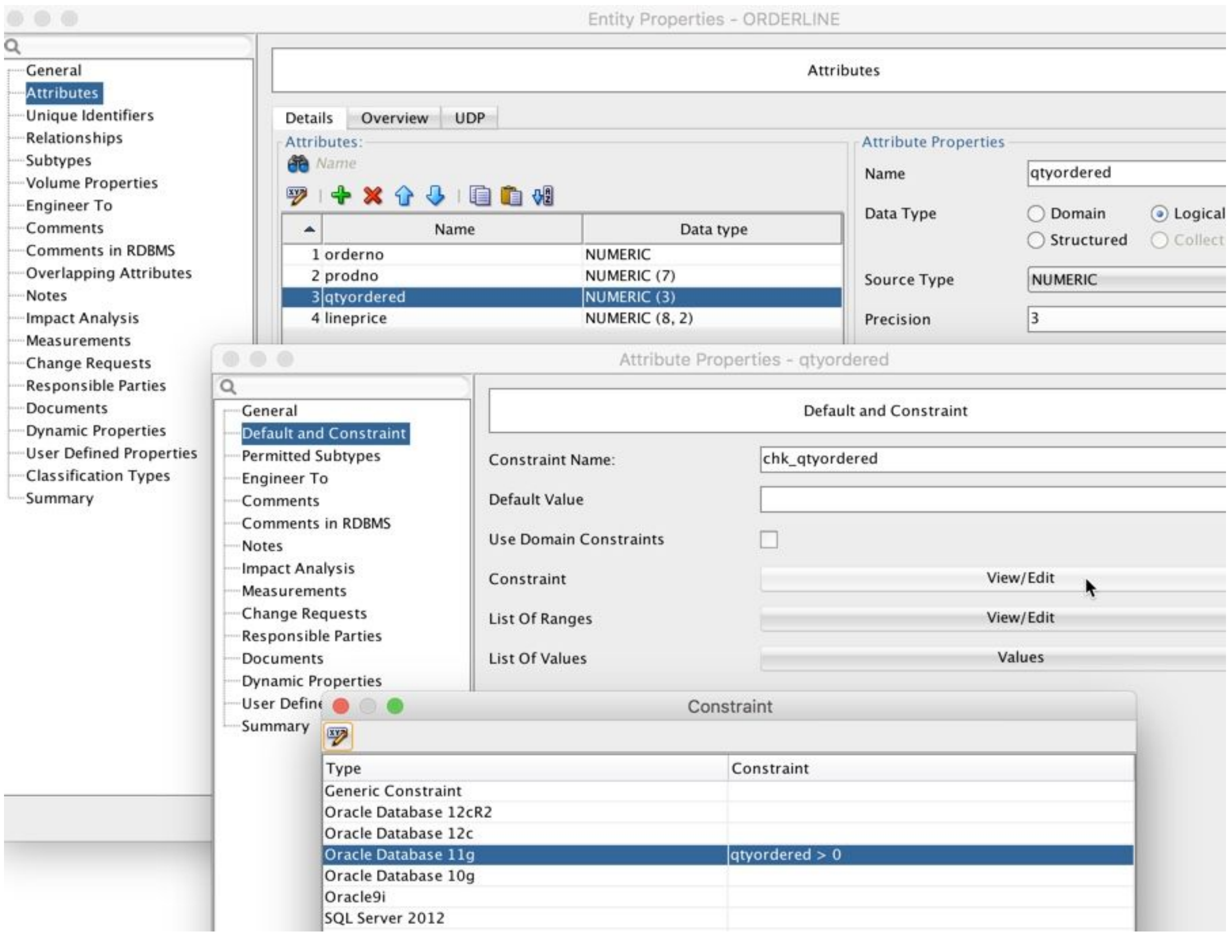


NOTE: In actual fact, Modeler can draw M:N relations(hips) on its logical models, however, you cannot add attributes to such relationships – *do not use the Oracle Modeler's M:N relation(ship) in our models*.

5.0.3.4. Constraints

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 double-click it to enter the **Entity Properties** dialog. Then select **Attributes** (left panel) and double click on **qtyordered** (right panel).

In the left hand list of the pop-up **Attribute Properties** window select "**Default and Constraint**". Assign a **Constraint Name** for example "`chk_qtyordered`" (be careful to select an informative name format, e.g. **`chk_columnname`**), click **Constraint** "**View/Edit**", double-click on **Oracle Database 11g**, and then enter the constraint into the constraint editor: **`qtyordered > 0`**



	Name	Data type
1	orderno	NUMERIC
2	prodno	NUMERIC (7)
3	qtyordered	NUMERIC (3)
4	lineprice	NUMERIC (8, 2)

Name: qtyordered
Data Type: ( ) Domain [✓ Logical] ( ) Structured ( ) Collect
Source Type: NUMERIC
Precision: 3

Constraint Name: chk_qtyordered
Default Value:
Use Domain Constraints: ☐
Constraint: View/Edit
List Of Ranges: View/Edit
List Of Values: Values

Type	Constraint
Generic Constraint	
Oracle Database 12cR2	
Oracle Database 12c	
Oracle Database 11g	qtyordered > 0
Oracle Database 10g	
Oracle9i	
SQL Server 2012	

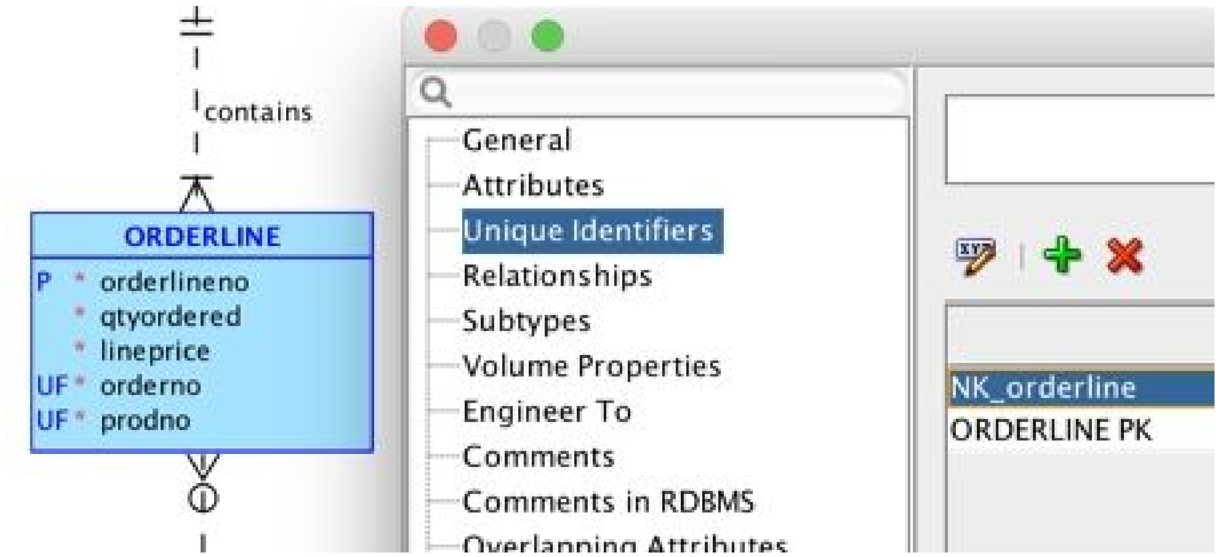
Note that here, the constraint added is the inner part of the standard SQL CHECK clause.

A better approach where a specific range, or set, of values is needed it to select “List Of Ranges” or “List Of Values” and directly enter the required values into the dialog boxes which appear.

5.0.3.4. Important note on Surrogate PK

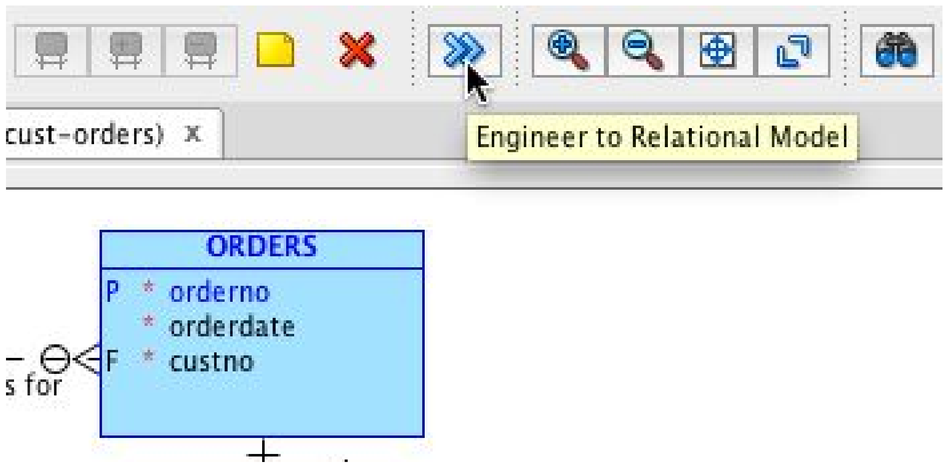
In developing a logical model, where *appropriate and documented*, you are free to introduce surrogate primary keys. If a surrogate PK is introduced into your design **you must ensure that the original key’s uniqueness is still maintained by enforcing a unique identifier which includes the attributes which compose the original key.**

As an example, say you decided to identify order lines by a unique order line number **(*this would not be a good/common choice*, it is being used here for demonstration purposes only)**. On the **ORDERLINE** entity, you need to select Unique Identifiers and add a new identifier (click the green add icon) with the attributes in your previously identified Natural Key (here **orderno** and **prodno**), this new unique identifier should be named appropriately:

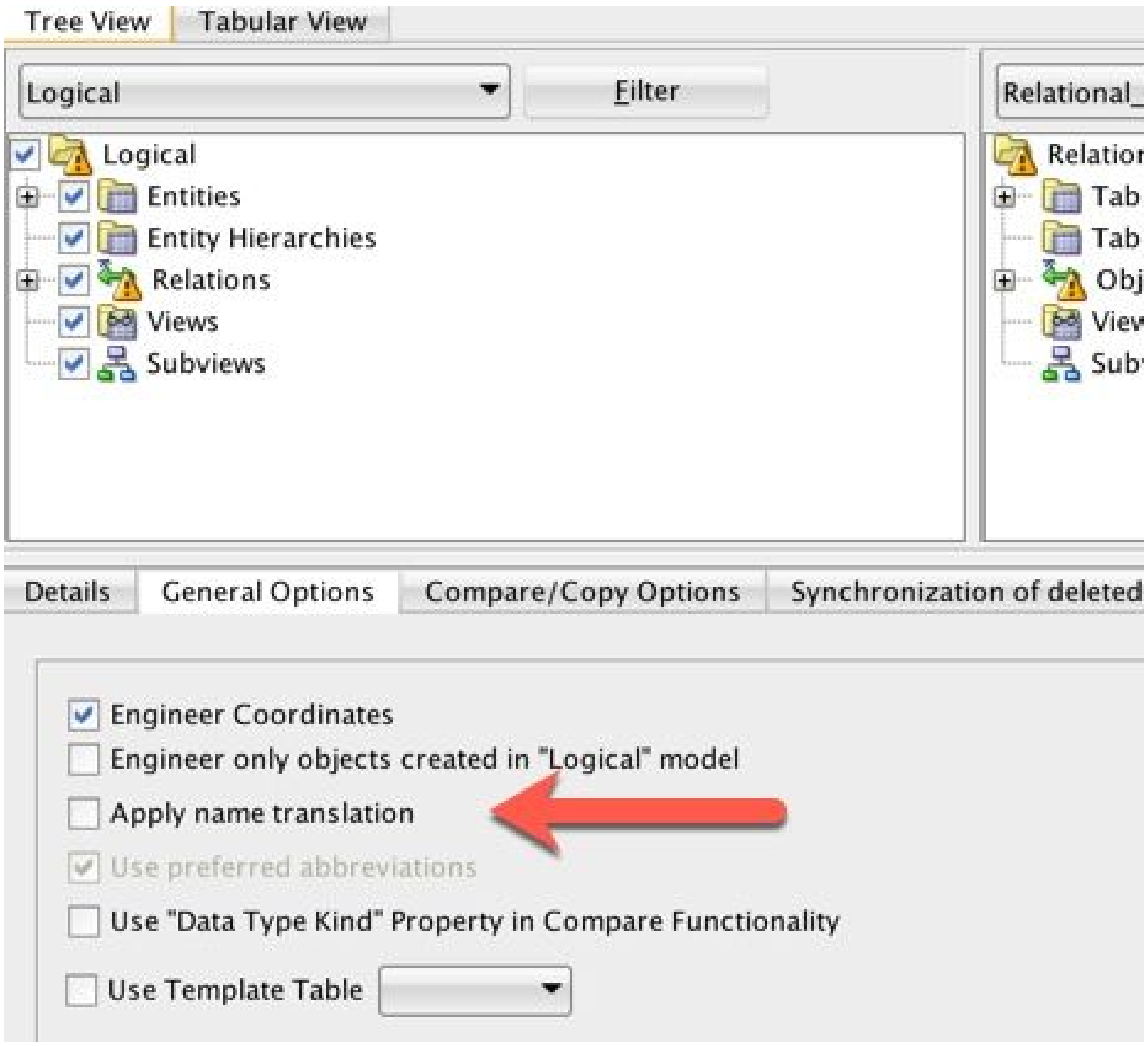


5.0.4. Develop a model – Stage 2 The Relational (Physical) Model:

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



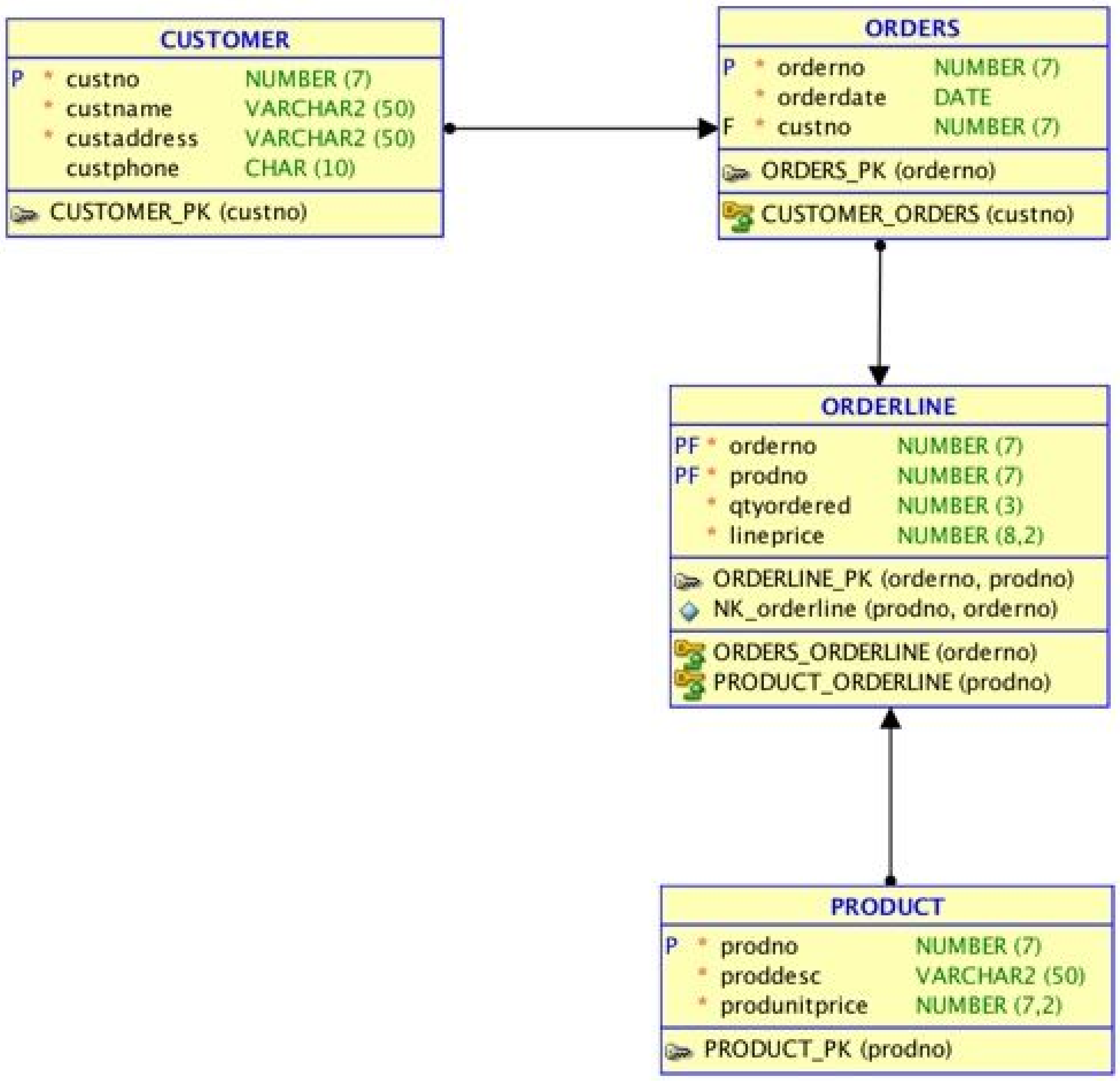
In the pop-up window which appears, on the “General Options” tab ensure “**Apply name translation**” is ***not*** checked:



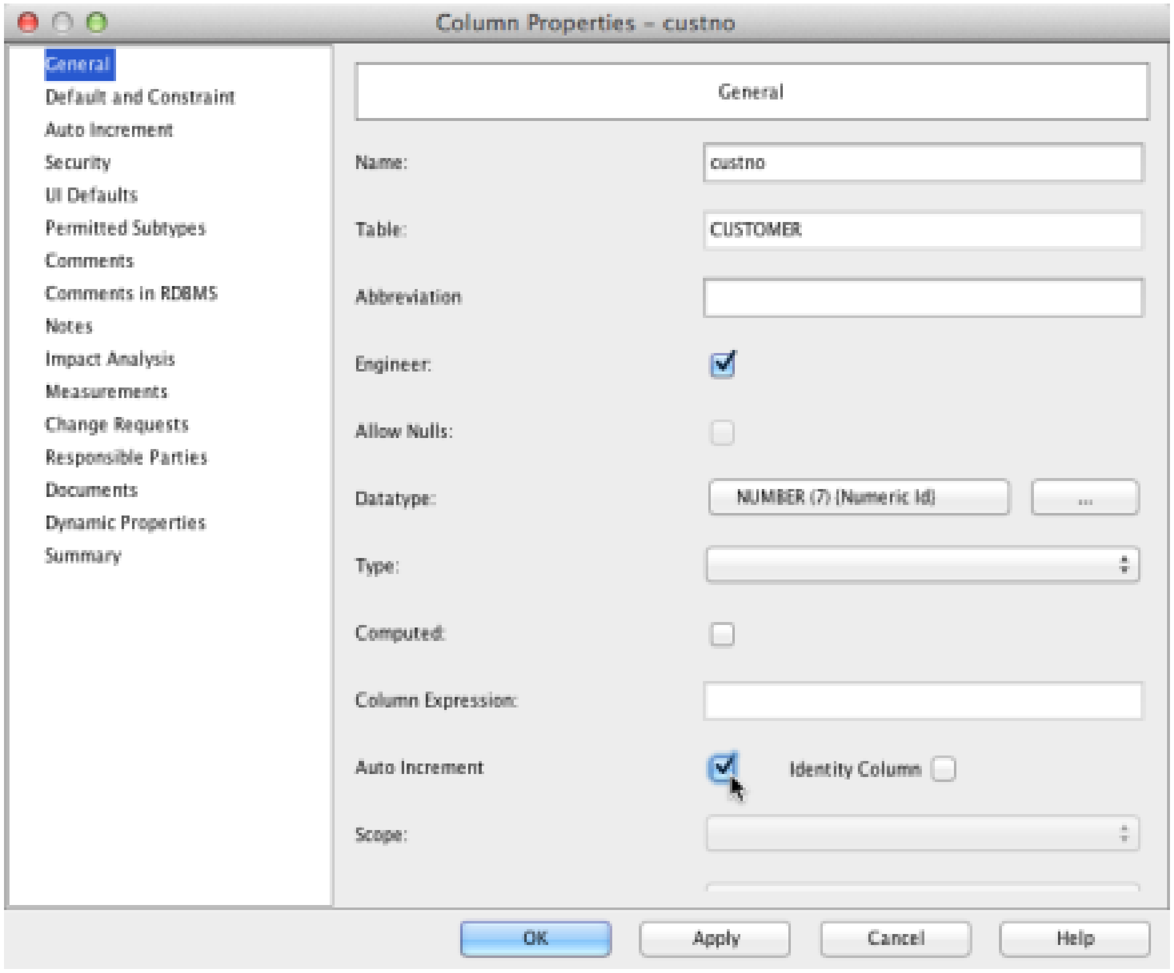
- [✓] Engineer Coordinates
- [ ] Engineer only objects created in "Logical" model
- [ ] Apply name translation
- [✓] Use preferred abbreviations
- [ ] Use "Data Type Kind" Property in Compare Functionality
- [ ] Use Template Table

With this option not checked, Data Modeler will use **exactly the same table and attribute names** for both your logical and relational models (the names you selected), **rather than apply its own set of rules** and potentially change a name on you. If a naming error occurs you will need to correct it on your logical model and regenerate the relational model.

Select the bottom left button, "**Engineer**". The logical model will then be engineered and a Relational Model is opened:

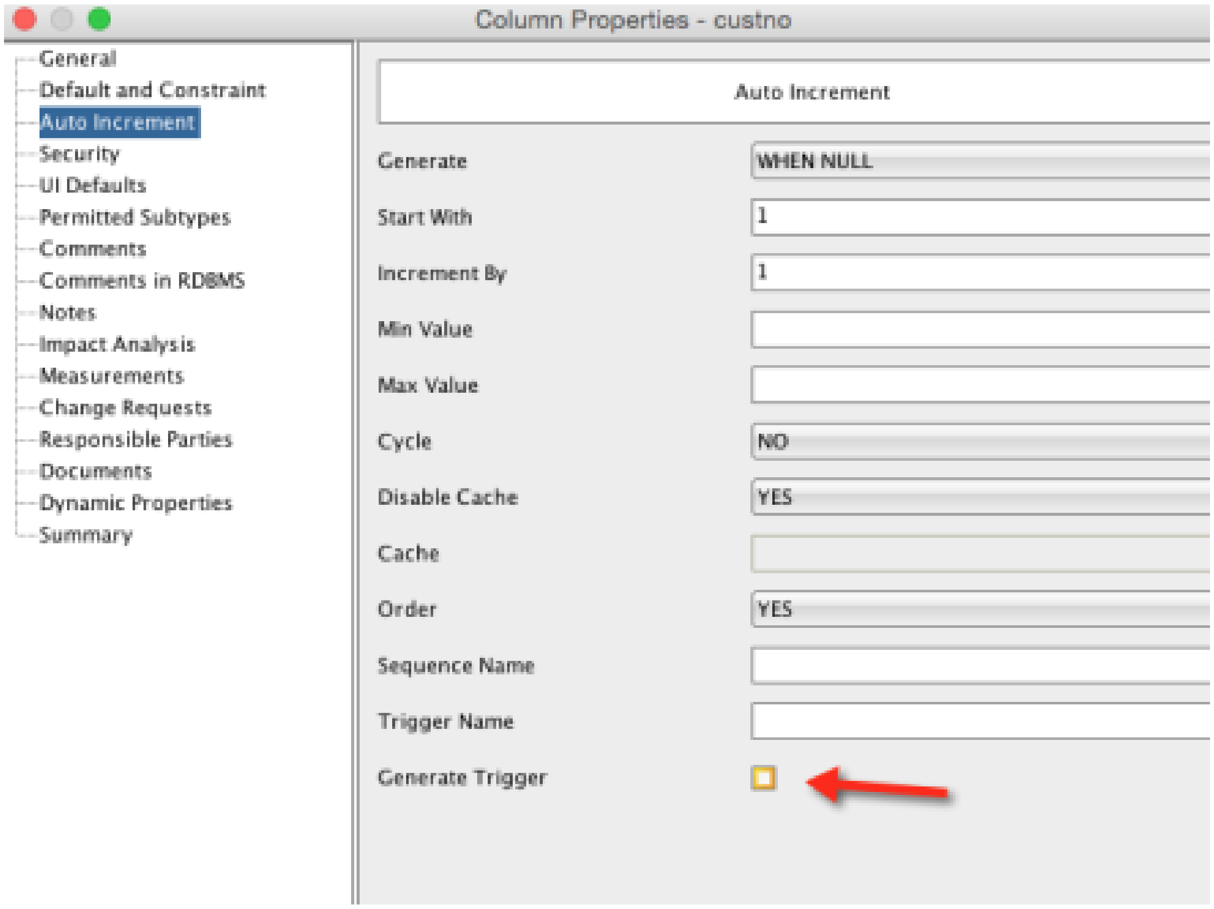


As an example of what can be configured in the Relational model, select the **CUSTOMER** table → double-click to bring up the **Table Properties** dialog → then the **Columns** on the left panel → then double-click on the **custno** column (right panel) → in the dialog which opens ("**Column Properties**") select "**Auto Increment**".



Name: custno
Table: CUSTOMER
Abbreviation:
Engineer: ✓
Allow Nulls:
Datatype: NUMBER (7) (Numeric Id)
Type:
Computed:
Column Expression:
Auto Increment ✓ Identity Column
Scope:

This will result in an Oracle 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”.**



Column Properties - custno

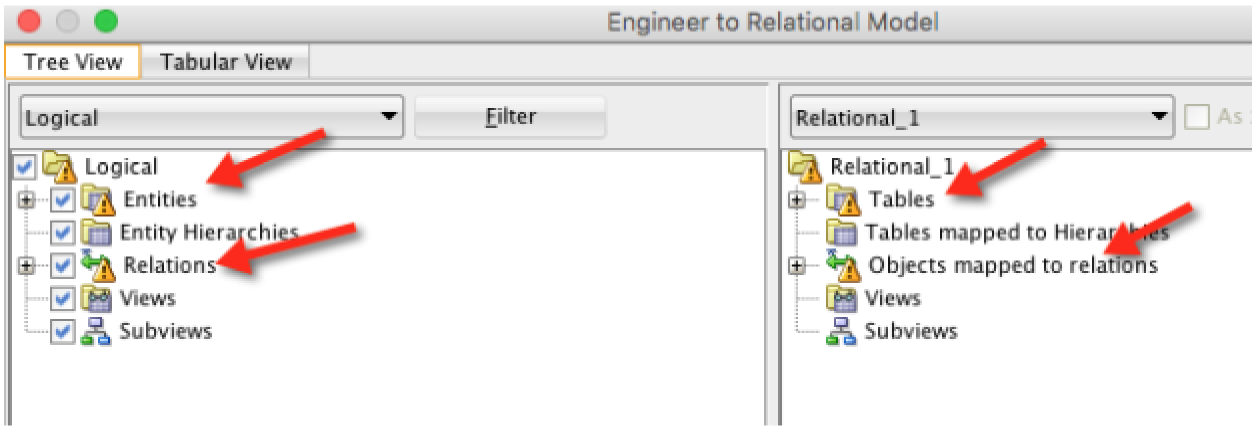
Auto Increment

Generate	WHEN NULL
Start With	1
Increment By	1
Min Value	
Max Value	
Cycle	NO
Disable Cache	YES
Cache	
Order	YES
Sequence Name	
Trigger Name	
Generate Trigger	☐

Although shown starting at **1** above, we often start auto-increment from say `100` to allow a developer to add test data under the sequence without making having to start use of the sequence and thus "use up" some values.

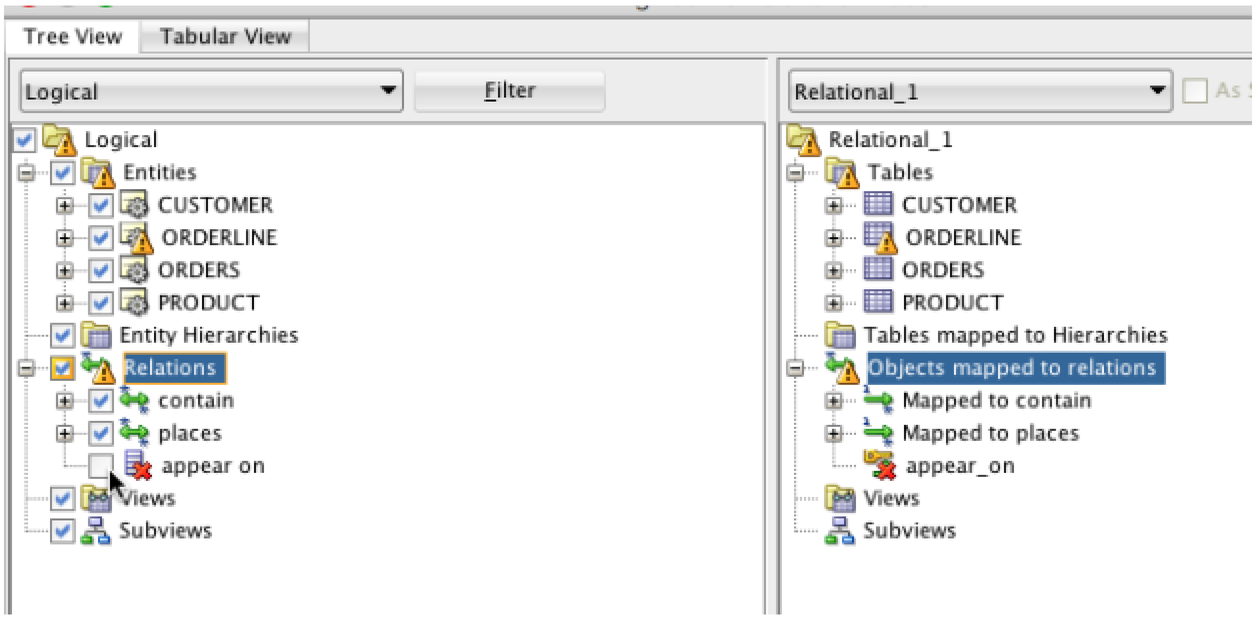
5.0.5. Re-engineering Models - Important Note!

If you are re-engineering a model (i.e. 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 (say, where we are removing the "appear on" relation as a **demonstration** of what occurs):

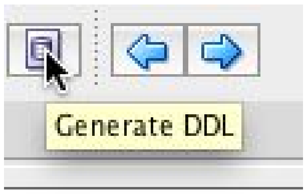


...the removal of "appear on" has not been selected to be engineered to the relational model. **You need to check the box if you wish it to be engineered (which we normally would).**

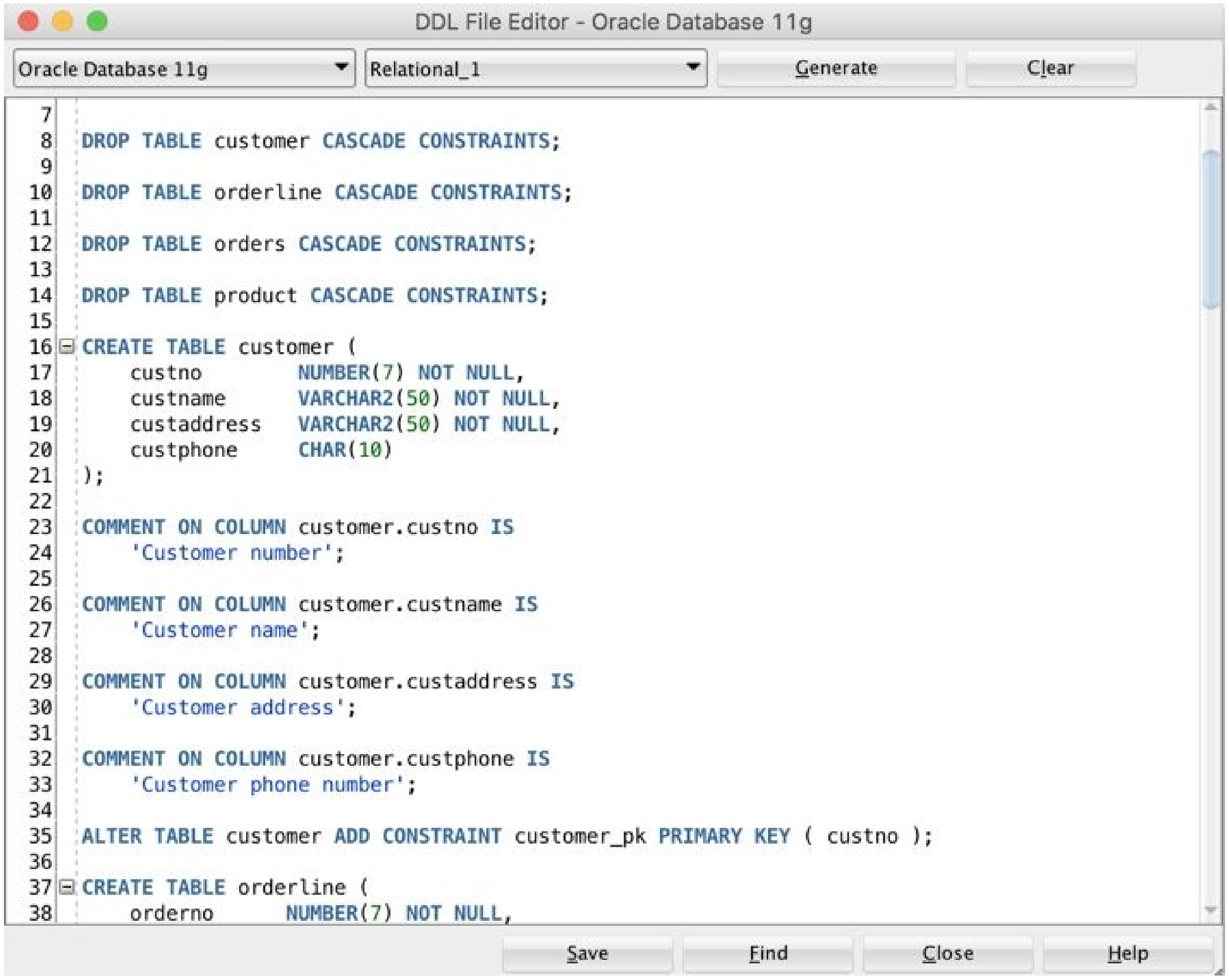
If your relational model gets very confused/confusing, you can select the relational model tab (main SQL Developer Window), do a Ctrl+A (or Cmd+A in Mac), and delete all the objects (DEL key on Windows/Linux, Fn-DELETE on Mac) - all the objects in the main window should disappear.

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.



```

DROP TABLE customer CASCADE CONSTRAINTS;

DROP TABLE orderline CASCADE CONSTRAINTS;

DROP TABLE orders CASCADE CONSTRAINTS;

DROP TABLE product CASCADE CONSTRAINTS;

CREATE TABLE customer (
    custno         NUMBER(7) NOT NULL,
    custname       VARCHAR2(50) NOT NULL,
    custaddress    VARCHAR2(50) NOT NULL,
    custphone      CHAR(10)
);

COMMENT ON COLUMN customer.custno IS
    'Customer number';

COMMENT ON COLUMN customer.custname IS
    'Customer name';

COMMENT ON COLUMN customer.custaddress IS
    'Customer address';

COMMENT ON COLUMN customer.custphone IS
    'Customer phone number';

ALTER TABLE customer ADD CONSTRAINT customer_pk PRIMARY KEY ( custno );

CREATE TABLE orderline (
    orderno       NUMBER(7) NOT NULL,
```

The generated file can be Saved as an Oracle schema script by selecting "**Save**".**You MUST use an extension of .sql, not the default .ddl – rename the file extension after saving if you use this approach (SQL Developer's SQL client cannot load .ddl files).**

The simplest way to save this is to *not* use the "Save" button, use Ctrl+A (Cmd on Mac) to select all the script, swap to an SQL Window and use Ctrl+V (Cmd on Mac), and then save the file from the SQL Window. 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. Logical Modelling – Task B – Rental Model

Week 5 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.

5.2. SQL Developer Data Modeller Issues

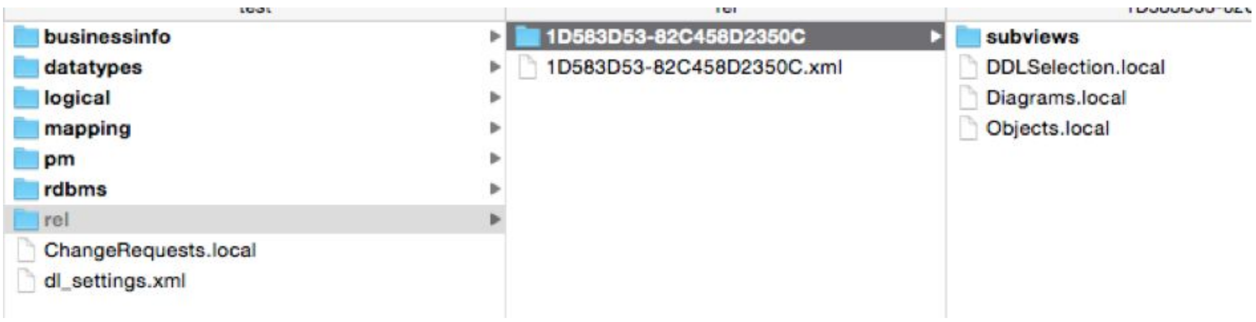
This document lists some of the issues you may experience when using SQL Developer and current work arounds.

NOTE: These are not required all of the time; therefore use these instructions only if you find errors!

5.2.1. Relational Model Disappears

When a model is saved SQL Developer Data Modeller sometimes fails to save your relational model fully. The Relational diagram is still present in your model but does not show within your project when you reopen it.

Before proceeding please ensure your model is closed and you have exited from SQL Developer. To correct this problem, open your model **folder** and navigate to the **`rel`** folder (this is where your relational model is saved):



The model shown above is complete, it does not have the missing relational diagram problem. The **`rel`** folder should contain a folder and an **xml** document of the same name. If there is a folder inside rel (it will have a different name, they are all unique) **but no xml file of the same name** then this is a save error.

Your relational model has not gone, the problem is that the xml document was not saved correctly, follow the steps below to correct this problem:

Make sure your model is closed.

1. Copy the linked XML file into the rel folder
2. Rename the file from "**`CHANGE_TO_FOLDER_NAME.xml`**" to have the same name as the folder in your rel folder, with the .xml extension added eg. as above 1D583D53-B2C458D2350C.xml (yours will be different)
3. Open the file in a *text editor* and change the **`id="CHANGE_TO_FOLDER_NAME"`** on line 2 to be the same as your folder name.

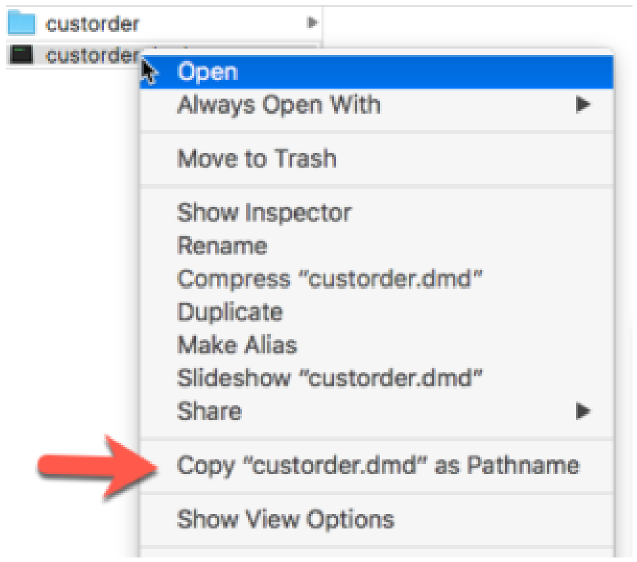
When you now open your model the relational model should be back.

5.2.2. Having problems navigating into a folder

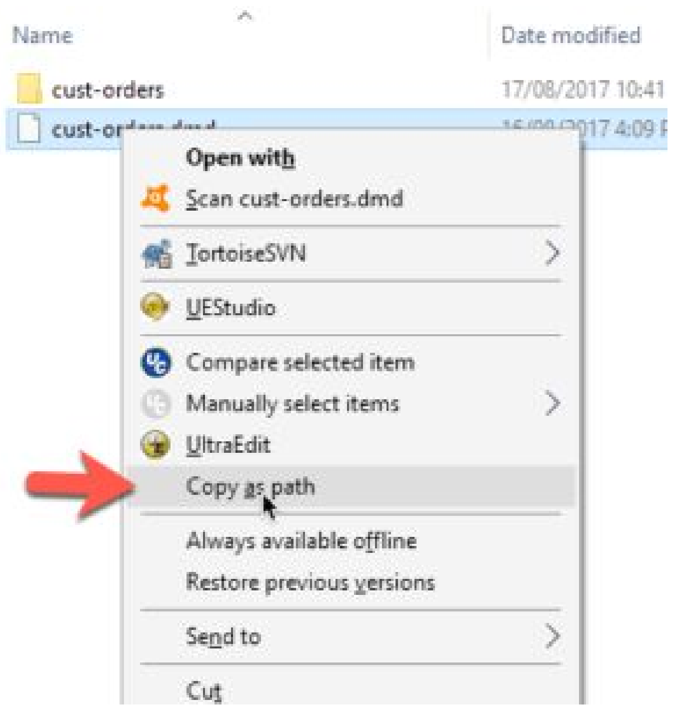
Sometimes the Java based file manager of SQL Developer Data Modeler has problems navigating through and selecting your folders.

If this occurs, or as a standard alternative, you can type/obtain in the full path name that you wish to open or navigate to. For example, if you have a Customer-Orders project you wish to open, use the following steps:

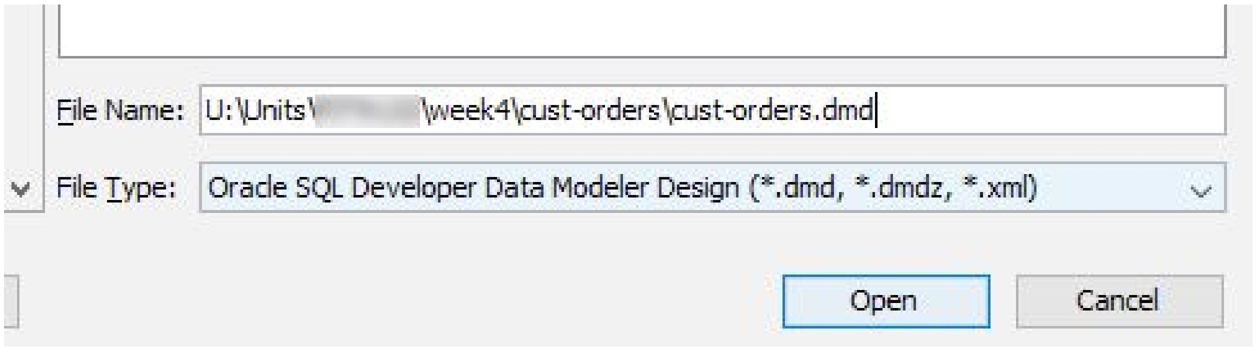
In your file manager (Explorer, Finder etc) navigate to where your .dmd file for the project is and obtain its full path name. Under MacOS, right click the dmd file and right click and while holding this, press the “option” key, this will result in



Under MS Windows, hold the shift and while holding it, right click on the file or folder, “Copy as path” will appear.



This path can then be pasted into the SQL Developer dialog box – to cause it to go to a particular folder or open a particular project. For example, in MS Windows:



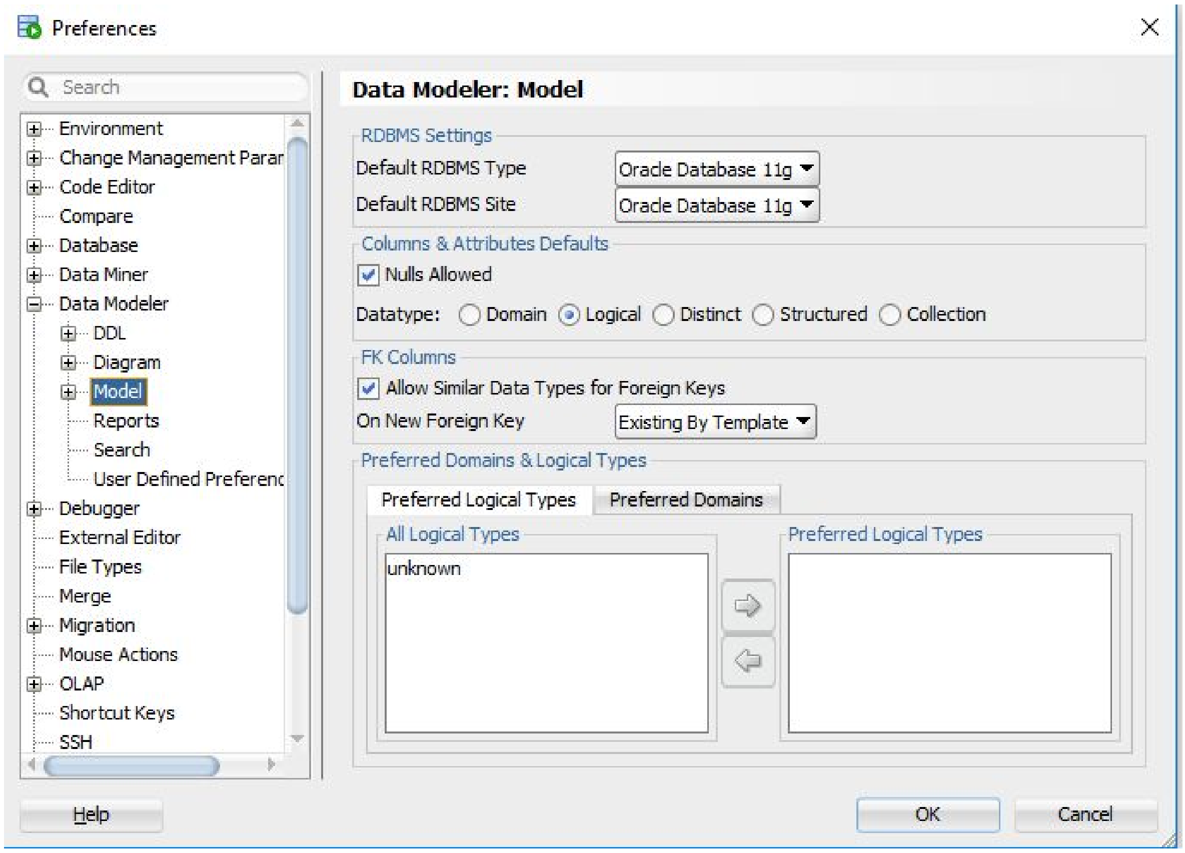
File Name: U:\Units\[redacted]\week4\cust-orders\cust-orders.dmd

File Type: Oracle SQL Developer Data Modeler Design (*.dmd, *.dmdz, *.xml)

Note that in MS Windows you must first remove the opening and closing quotes (") from the path name.

5.2.3. Logical Datatypes Disappear

My data types have disappeared – when you check in preferences you have no Logical Types:


Preferences

Data Modeler: Model

RDBMS Settings

Default RDBMS Type: Oracle Database 11g

Default RDBMS Site: Oracle Database 11g

Columns & Attributes Defaults

☑ Nulls Allowed

Datatype: ○ Domain ● Logical ○ Distinct ○ Structured ○ Collection

FK Columns

☑ Allow Similar Data Types for Foreign Keys

On New Foreign Key: Existing By Template

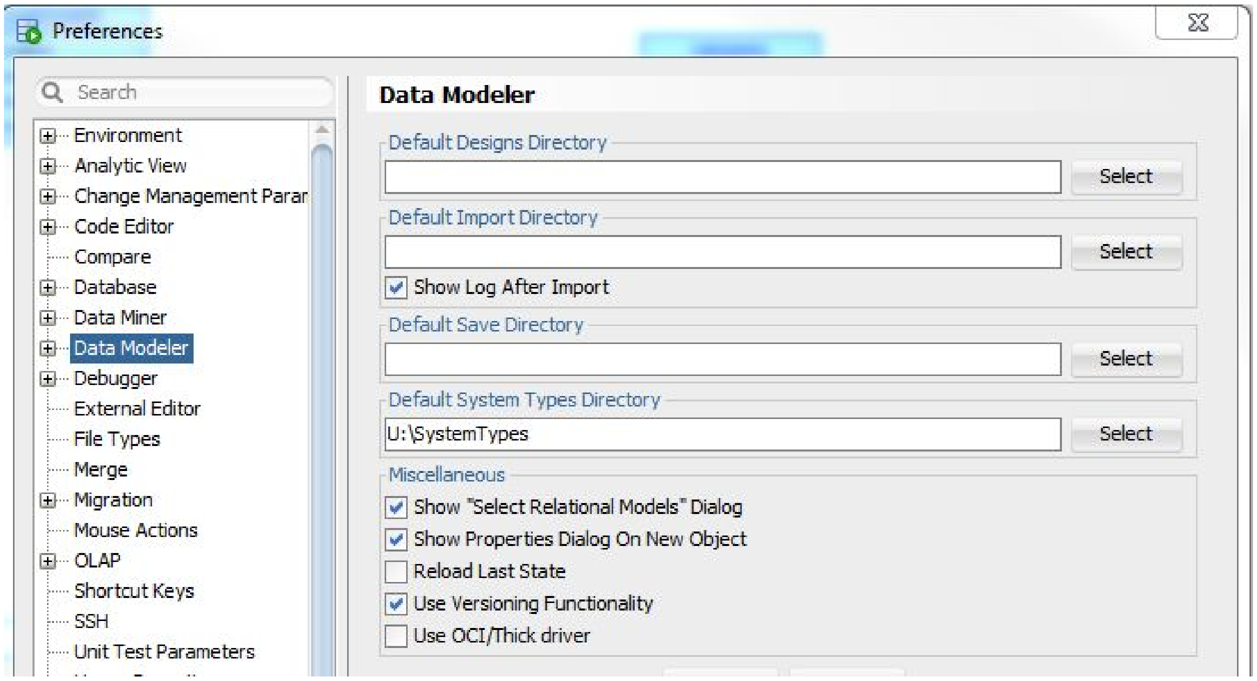
Preferred Domains & Logical Types

Preferred Logical Types | Preferred Domains

All Logical Types: unknown

Preferred Logical Types:

This has occurred because your default Systems Type Directory is now set incorrectly.


Preferences

Data Modeler

Default Designs Directory:

Default Import Directory:

☑ Show Log After Import

Default Save Directory:

Default System Types Directory: U:\SystemTypes

Miscellaneous

☑ Show "Select Relational Models" Dialog

☑ Show Properties Dialog On New Object

☐ Reload Last State

☑ Use Versioning Functionality

☐ Use OCI/Thick driver

This can occur if you have accidentally deleted the files in your Default System Types Directory, if you have placed other files or subfolders in it, if you have a space in the pathname, or if SQL Developer has not correctly saved them at some stage.

To fix this:

- Go back to this setting and remove the current value in this entry ie. in the above, as used in the on-campus labs, it is **`U:\SystemTypes`** – make this entry blank (have no path)
- Save your settings and exit SQL Developer
- Reopen SQL Developer and the types will be back.

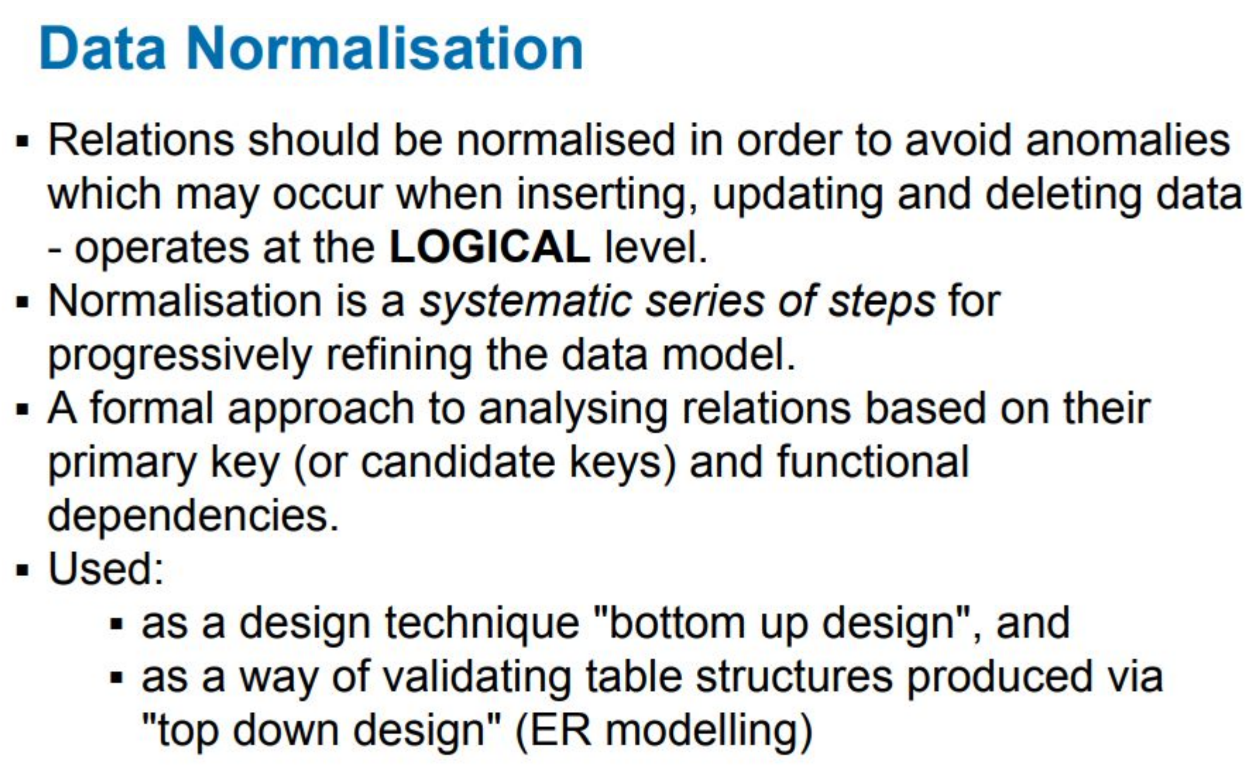
If you then have problems saving a model, you will need to reset up the Default System Types Directory as you previously did (**ensure there is no space in the pathname**).

5.3. Pre-Lecture Notes

These are notes you may find useful to read through before the lecture, adapted from Lindsay Smith's lecture material. NOTE: THESE ARE NOT THE FINAL LECTURE SLIDES.

Read up the pre-lecture notes below.

IMPORTANT: AS THE LECTURE FOCUSES ON THE 'FLIPPED CLASSROOM' APPROACH - I.E. MORE INTERACTIVE DISCUSSION AND LESS ON DISCOVERING NEW MATERIAL - THE THEORY SLIDES BELOW ARE PROVIDED FOR YOUR READING CONVENIENCE BEFORE THE LECTURE.



Data Normalisation

- Relations should be normalised in order to avoid anomalies which may occur when inserting, updating and deleting data - operates at the **LOGICAL** level.
- Normalisation is a *systematic series of steps* for progressively refining the data model.
- A formal approach to analysing relations based on their primary key (or candidate keys) and functional dependencies.
- Used:
  - as a design technique "bottom up design", and
  - as a way of validating table structures produced via "top down design" (ER modelling)

Sample Data

FIGURE 6.1 **Tabular representation of the report format**

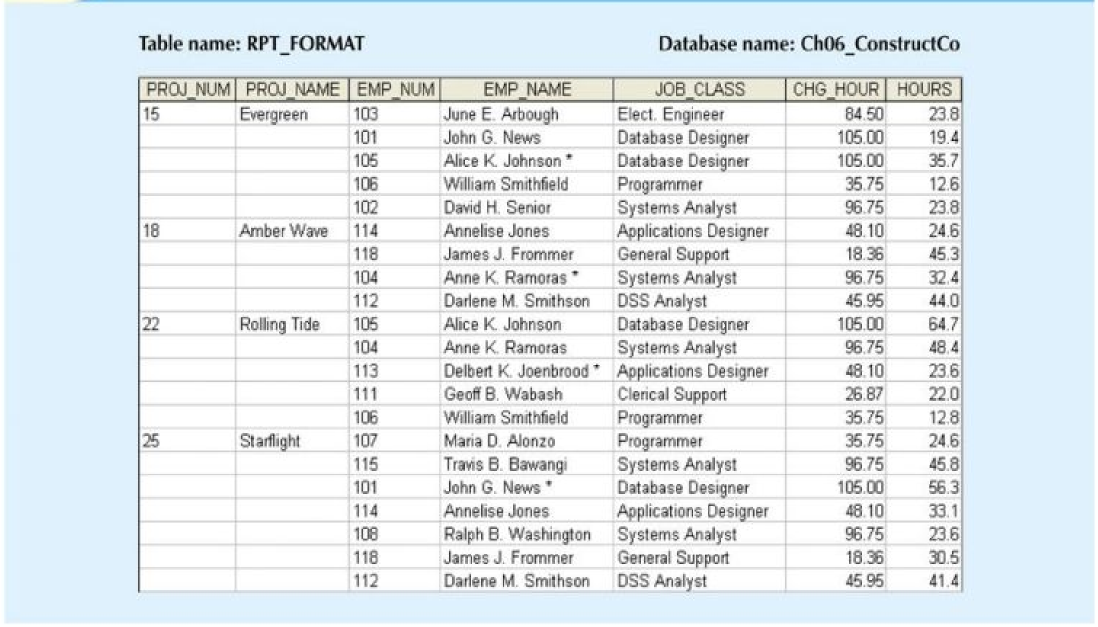


Table name: RPT_FORMAT Database name: Ch06_ConstructCo

PROJ_NUM	PROJ_NAME	EMP_NUM	EMP_NAME	JOB_CLASS	CHG_HOUR	HOURS
15	Evergreen	103	June E. Arbough	Elect. Engineer	84.50	23.8
		101	John G. News	Database Designer	105.00	19.4
		105	Alice K. Johnson *	Database Designer	105.00	35.7
		106	William Smithfield	Programmer	35.75	12.6
		102	David H. Senior	Systems Analyst	96.75	23.8
18	Amber Wave	114	Annelise Jones	Applications Designer	48.10	24.6
		118	James J. Frommer	General Support	18.36	45.3
		104	Anne K. Ramoras *	Systems Analyst	96.75	32.4
		112	Darlene M. Smithson	DSS Analyst	45.95	44.0
22	Rolling Tide	105	Alice K. Johnson	Database Designer	105.00	64.7
		104	Anne K. Ramoras	Systems Analyst	96.75	48.4
		113	Delbert K. Joenbrood *	Applications Designer	48.10	23.6
		111	Geoff B. Wabash	Clerical Support	26.87	22.0
		106	William Smithfield	Programmer	35.75	12.8
25	Starflight	107	Maria D. Alonzo	Programmer	35.75	24.6
		115	Travis B. Bawangi	Systems Analyst	96.75	45.8
		101	John G. News *	Database Designer	105.00	56.3
		114	Annelise Jones	Applications Designer	48.10	33.1
		108	Ralph B. Washington	Systems Analyst	96.75	23.6
		118	James J. Frommer	General Support	18.36	30.5
		112	Darlene M. Smithson	DSS Analyst	45.95	41.4

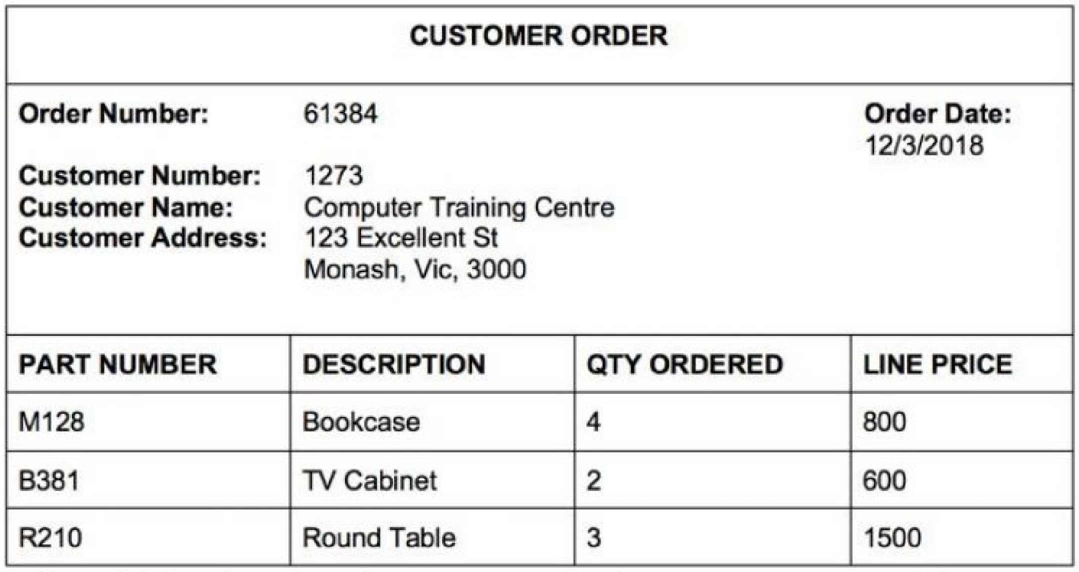
Problems with data in Figure 6.1

- PROJ_NUM intended to be **primary key**, but it contains nulls
- JOB_CLASS invites **entry errors** eg. Elec. Eng. vs Elect. Engineer vs E.E.
- Table has **redundant data**
  - Details of a charge per hour are repeated for every occurrence of job class
  - Every time an employee is assigned to a project emp name repeated
- Relations that contain redundant information may potentially suffer from several update anomalies
  - Types of update anomalies include:
    - **Insert Anomaly**
      - –Insert a new employee only if they are assigned to a project
    - **Delete Anomaly**
      - –Delete the last employee assigned to a project?
      - –Delete the last employee of a particular job class?
    - **Modification (or update) Anomaly**
      - –Update a job class hourly rate - need to update multiple rows

The Normalisation Process Goals

- Creating valid relations, i.e. each relation meets the properties of the relational model. In particular:
  - Entity integrity
  - Referential integrity
  - No many-to-many relationship
  - Each cell contains a single value (is atomic).
- In practical terms:
  - Each table represents a single subject
  - No data item will be unnecessarily stored in more than one table.
  - The relationship between tables can be established (pair of PK and FK is identified).
  - Each table is void of insert, update and delete anomalies.

Representing a form as a relation



CUSTOMER ORDER

Order Number: 61384 **Order Date:** 12/3/2018

Customer Number: 1273
Customer Name: Computer Training Centre
Customer Address: 123 Excellent St
Monash, Vic, 3000

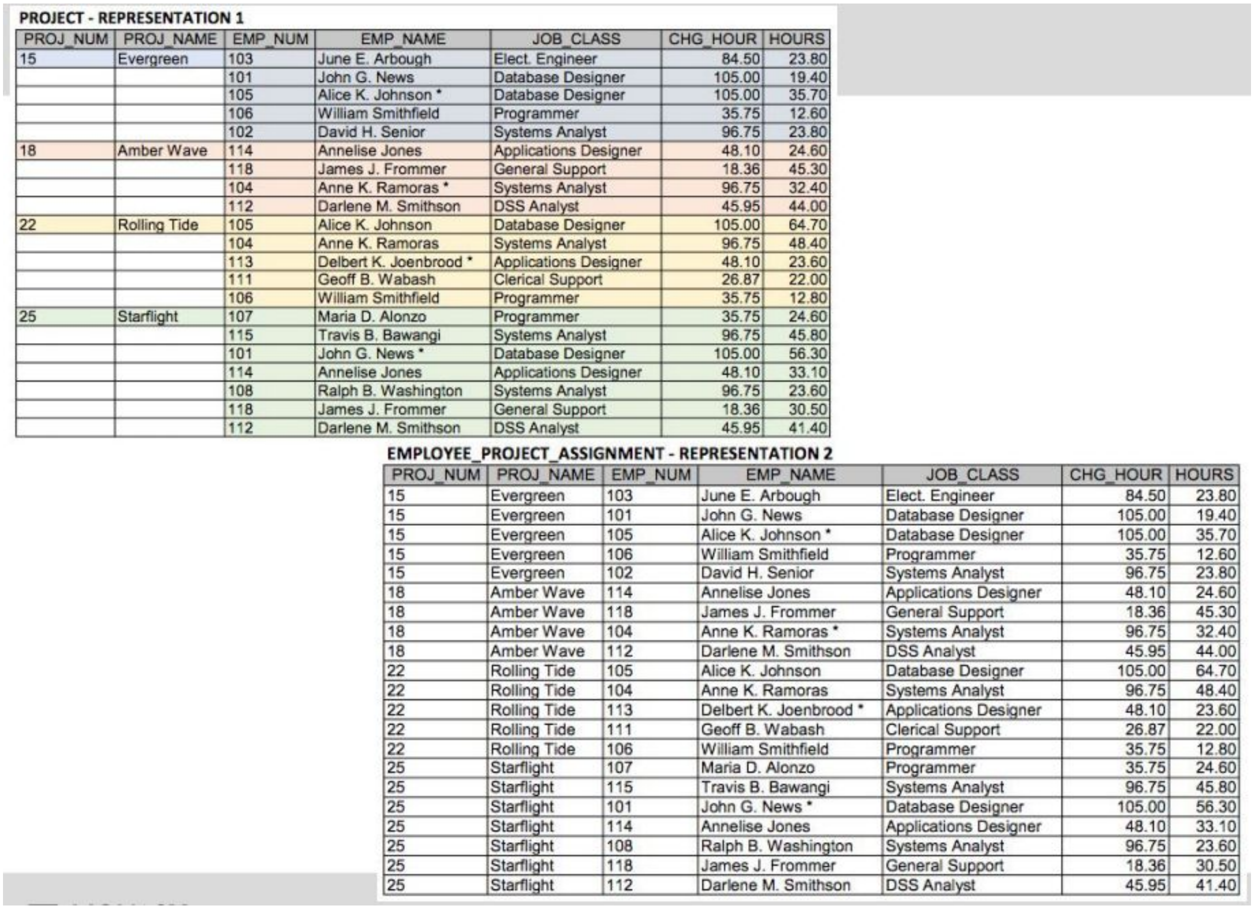
PART NUMBER	DESCRIPTION	QTY ORDERED	LINE PRICE
M128	Bookcase	4	800
B381	TV Cabinet	2	600
R210	Round Table	3	1500

ORDER (orderno, orderdate, custnumb, custname, custaddress (partno, partdesc, qtyordered, lineprice))

- Note this is ***not*** a relation
- (partno, partdesc, qtyordered, lineprice) - is a multivalued set of attributes – called a repeating group in normalisation terminology

Functional Dependency

- TRANSITIVE DEPENDENCY
  - occurs when Y depends on X, and Z depends on Y - thus Z also depends on X ie. X ➔ Y ➔ Z
  - **and** Y is not a candidate key (or part of a candidate key)
  - ORDER-NUMB ➔ CUSTOMER-NUMB ➔ CUSTOMER-NAME
- Dependencies are depicted with the help of a **Dependency Diagram**.
- Normalisation converts a relation into relations of progressively smaller number of attributes and tuples until an optimum level of decomposition is reached - little or no data redundancy exists.
- The output from normalisation is a set of relations that meet all conditions set in the relational model principles.



PROJECT - REPRESENTATION 1

PROJ_NUM	PROJ_NAME	EMP_NUM	EMP_NAME	JOB_CLASS	CHG_HOUR	HOURS
15	Evergreen	103	June E. Arbough	Elect. Engineer	84.50	23.80
		101	John G. News	Database Designer	105.00	19.40
		105	Alice K. Johnson *	Database Designer	105.00	35.70
		106	William Smithfield	Programmer	35.75	12.60
		102	David H. Senior	Systems Analyst	96.75	23.80
18	Amber Wave	114	Annelise Jones	Applications Designer	48.10	24.60
		118	James J. Frommer	General Support	18.36	45.30
		104	Anne K. Ramoras *	Systems Analyst	96.75	32.40
		112	Darlene M. Smithson	DSS Analyst	45.95	44.00
22	Rolling Tide	105	Alice K. Johnson	Database Designer	105.00	64.70
		104	Anne K. Ramoras	Systems Analyst	96.75	48.40
		113	Delbert K. Joenbrood *	Applications Designer	48.10	23.60
		111	Geoff B. Wabash	Clerical Support	26.87	22.00
		106	William Smithfield	Programmer	35.75	12.80
25	Starflight	107	Maria D. Alonzo	Programmer	35.75	24.60
		115	Travis B. Bawangi	Systems Analyst	96.75	45.80
		101	John G. News *	Database Designer	105.00	56.30
		114	Annelise Jones	Applications Designer	48.10	33.10
		108	Ralph B. Washington	Systems Analyst	96.75	23.60
		118	James J. Frommer	General Support	18.36	30.50
		112	Darlene M. Smithson	DSS Analyst	45.95	41.40

EMPLOYEE_PROJECT_ASSIGNMENT - REPRESENTATION 2

PROJ_NUM	PROJ_NAME	EMP_NUM	EMP_NAME	JOB_CLASS	CHG_HOUR	HOURS
15	Evergreen	103	June E. Arbough	Elect. Engineer	84.50	23.80
15	Evergreen	101	John G. News	Database Designer	105.00	19.40
15	Evergreen	105	Alice K. Johnson *	Database Designer	105.00	35.70
15	Evergreen	106	William Smithfield	Programmer	35.75	12.60
15	Evergreen	102	David H. Senior	Systems Analyst	96.75	23.80
18	Amber Wave	114	Annelise Jones	Applications Designer	48.10	24.60
18	Amber Wave	118	James J. Frommer	General Support	18.36	45.30
18	Amber Wave	104	Anne K. Ramoras *	Systems Analyst	96.75	32.40
18	Amber Wave	112	Darlene M. Smithson	DSS Analyst	45.95	44.00
22	Rolling Tide	105	Alice K. Johnson	Database Designer	105.00	64.70
22	Rolling Tide	104	Anne K. Ramoras	Systems Analyst	96.75	48.40
22	Rolling Tide	113	Delbert K. Joenbrood *	Applications Designer	48.10	23.60
22	Rolling Tide	111	Geoff B. Wabash	Clerical Support	26.87	22.00
22	Rolling Tide	106	William Smithfield	Programmer	35.75	12.80
25	Starflight	107	Maria D. Alonzo	Programmer	35.75	24.60
25	Starflight	115	Travis B. Bawangi	Systems Analyst	96.75	45.80
25	Starflight	101	John G. News *	Database Designer	105.00	56.30
25	Starflight	114	Annelise Jones	Applications Designer	48.10	33.10
25	Starflight	108	Ralph B. Washington	Systems Analyst	96.75	23.60
25	Starflight	118	James J. Frommer	General Support	18.36	30.50
25	Starflight	112	Darlene M. Smithson	DSS Analyst	45.95	41.40

Unormalised Form (UNF)

- **Identify a “subject” that needs to be modelled**
  - For example from figure 6.1 possible “subjects” of interest:
    - PROJECT (we will call this representation 1)
    - EMPLOYEE_PROJECT_ASSIGNMENT (we will abbreviate this as ASSIGNMENT and will call this representation 2).
- Choose one subject of interest as a starting point and identify a primary key for this subject of interest.
  - For example for PROJECT, primary key would be project _number (or we will abbreviate it as proj_num).

First Normal Form

- FIRST NORMAL FORM (part of formal definition of a relation)
  - A RELATION IS IN FIRST NORMAL FORM (1NF) IF:
    - a unique primary key has been identified for each tuple/row.
    - it is a valid relation
      - Entity integrity (no part of PK is null)
      - Single value for each cell.
      - No repeating group.
    - all attributes are functionally dependent on all or part of the primary key

UNF to 1NF transformation

- Identify the repeating group(s), if any, in the unnormalised relation.
  - For representation 1, a project will have more than one employee assigned to it, hence there is a repeating group.
  - We have one-to-many relationship from PROJECT to EMPLOYEE.

UNF to 1NF

- Move from UNF to 1NF by:
  1. identify a unique identifier for the repeating group.
  2. *remove the repeating group* along with the PK of the main relation.
  3. The PK of the new relation resulting from the removal of repeating group will *normally* have a composite PK made up of the PK of the main relation and the unique identifier chosen in 1. above, but this ***must be checked***.

1NF to 2NF

- A RELATION IS IN 2NF IF -
  - all non key attributes are functionally dependent on the **entire** primary key (simplified definition)
    - i.e. no partial dependencies exist
  - all non key attributes are functionally dependent on **any candidate key** (general definition)
  - for this unit we will only use the simplified definition ie. look for partial dependencies based on the primary key

2NF to 3NF

- A RELATION IS IN 3NF IF -
  - all transitive dependencies have been removed - check for ***non key attribute dependent on another non key attribute***
- Move from 2NF to 3NF by removing transitive dependencies

Monash Software EMPLOYEE form

- List all attributes found on the form, maintain consistency with previously used attribute names if exist:
  - emp_no, emp_fname, emp_lname, emp_dob, emp_street_no, emp_street, emp_town, emp_pcode, phone_type, phone_no, degree_name, degree_institution, degree_year, fmemb_no, fmemb_name, fmemb_dob, skill_name
- Determine if any attribute is multivalued for a given entity instance
  - phone_type, phone_no, degree_name, degree_institution, degree_year, fmemb_no, fmemb_name, fmemb_dob, skill_name

Summary

- Things to remember
  - Primary Key selection in moving from UNF to 1NF is important, it will determine the starting point (choose your subject of interest).
  - Functional dependency
  - Process of removing attributes in relations based on the concept of 1NF, 2NF and 3NF.
    - UNF to 1NF define PK & remove repeating group.
    - 1NF to 2NF remove partial dependency.
    - 2NF to 3NF remove transitive dependency.

EOF.