School of Computing Technologies

ISYS1101/ 1102 Database Applications

Week 3: Tute/Lab – Indexing and Database Design using SQL Developer

Semester 2 2022

1 Objective

There are two parts in this week's Tute/lab session.

The objective of the first part is to learn more about indexing, more specifically:

- Try out a few indexes to see any performance differences;
- Use EXPLAIN PLAN to see the workings behind a query;
- Calculate disk I/O with and without indexes;

In the second half, we learn how to use the "Data Modeler" feature in Oracle SQL Developer to design a new database schema.
You will learn:

- Using Barkers notation to draw entity-relationship diagrams;
- Convert logical design (i.e. ER Model) into physical database design;
- Generate DDL script;
- Build the database schema using the DDL file.

This activity is directly aligned with the first milestone of the first assignment. Once you are comfortable with the basic operations in drawing an ER model, convert into physical database design and generate the DDL file, you should start working on the database design for the assignment problem.

2 Preparation Tasks

From now on, you cannot use sample databases. You must have your own database schema. Contact ITS if you still did not receive an email on connection settings.

For the first part of the tute/ lab session, we use a sample table with about 16000 rows. While it is relatively small to see real benefits of indexing, you will still see the differences in disk I/O and costs between queries that use and do not use indexes.

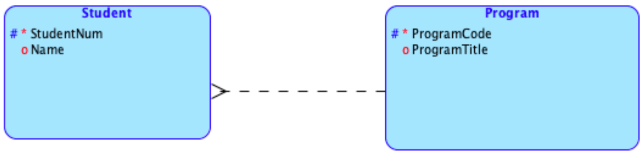
Download `rental-DDL.sql` and `rental-populate.sql` files from Canvas Resources → Sample Database area. Using these two scripts, build and populate the `rental` table. Do not not make any changes to DDL.

In the second part of this Tute/ lab we use the Data Modeler functionality of SQL Developer. Oracle uses Barker Notation for Entity Relationship modelling. In your first database course, you should have learned one modelling notation, such as UML or crow's feet or Chen notation. Oracle's

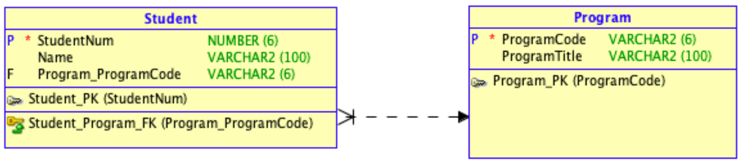
Barker Notation is very similar to the crow's feet notation, but slightly different in cardinality and participation representation of relationships.

Be familiarised with the Barker Notation. It is always useful to know more than one notation. A good starting point is: https://www.vertabelo.com/blog/technical-articles/barkers-erd-notation
Refer to:
https://docs.oracle.com/cd/E39885_01/doc.40/e48205/tut_data_modeling.htm#DMDUG36166
for a complete tutorial.
One of the common mistakes you do with ER modeming is the inclusion of foreign keys in the target entity. When you draw up your logical design as an Entity Relationship model, DO NOT include foreign keys. Only include entities and their attributes and relationships. When you forward-engineer into physical design, the SQL Developer will identify where to insert foreign keys and will add them where they belong.

Example: Let's suppose you design a simple enrolment system. We store information about students and programs. Students can enrol in one program only (of course, programs can have many students.) So, we establish a one-many relationship between these two entities. Your logical design should be like this:



Note that, there is no foreign key (`ProgramCode`) within the `Student` entity. When we convert this onto a physical design, it will be added, as follows:


Student
P * StudentNum NUMBER (6)
Name VARCHAR2 (100)
F Program_ProgramCode VARCHAR2 (6)
Student_PK (StudentNum)
Student_Program_FK (Program_ProgramCode)
Program
P * ProgramCode VARCHAR2 (6)
ProgramTitle VARCHAR2 (100)
Program_PK (ProgramCode)

Your Lab assistant will show you how to start your first logical design on SQL Developer.

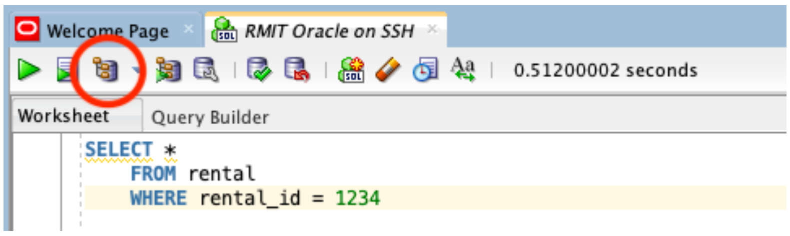
3 Tutorial Questions on Indexing

3.1 Using Indexes

Activity 1: Run the following Query:

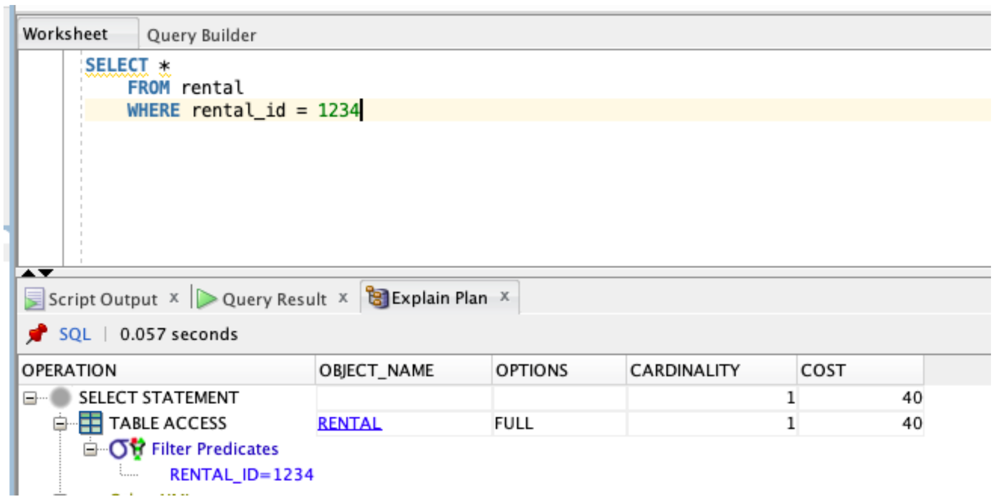
```
SELECT *
    FROM rental
    WHERE rental_id = 1234;
```

After running the query, check the query execution plan, using EXPLAIN PLAN option (third button on toolbar) on SQL Developer (or press F10 key shortcut).



```
SELECT *
    FROM rental
    WHERE rental_id = 1234
```

You will be presented with a table with a list of operations and associated costs.



```
SELECT *
    FROM rental
    WHERE rental_id = 1234
```

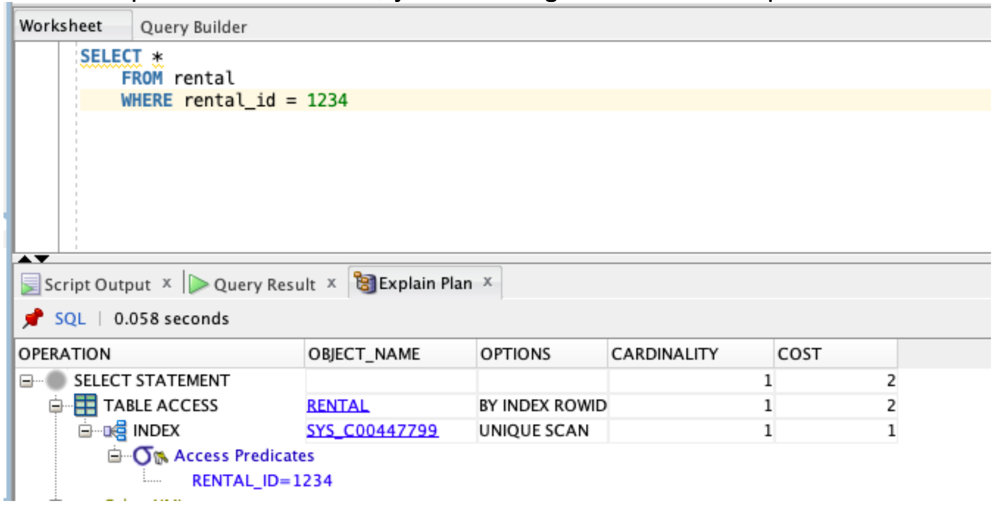
OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	40
TABLE ACCESS	RENTAL	FULL	1	40
Filter Predicates				
RENTAL_ID=1234				

Among the operations, there is no indication of the use of an index. However, rental_id is the primary key of the table, which should have an index built while creating the table.

Why didn't it use the primary key index? Discuss with the group and your lab assistant.

How do you overcome this problem?

Once the problem is rectified, you should get an execution plan which utilise the index.



```
SELECT *
    FROM rental
    WHERE rental_id = 1234
```

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	2
TABLE ACCESS	RENTAL	BY INDEX ROWID	1	2
INDEX	SYS_C00447799	UNIQUE SCAN	1	1
Access Predicates				
RENTAL_ID=1234				

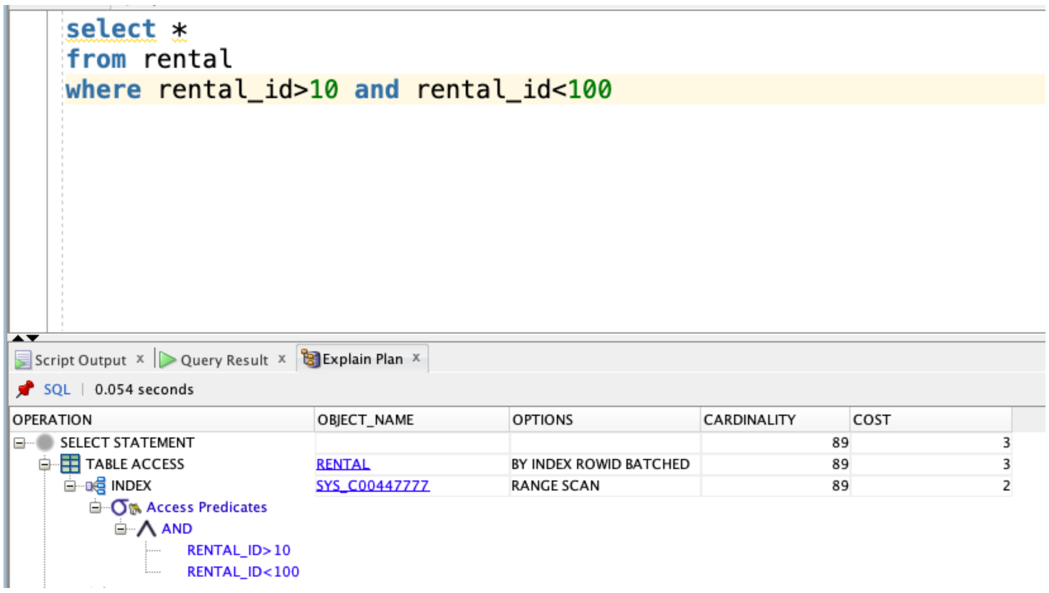
You will notice that the cost had reduced 20-fold (from 40 to 2) when the index is used.

Activity 2: Now run the following two queries.

```
SELECT *
    FROM rental
    WHERE rental_id > 10 AND rental_id <100;
```

```
SELECT *
    FROM rental
    WHERE rental_id > 10 AND rental_id <4000;
```

After running each query, obtain the execution plan.



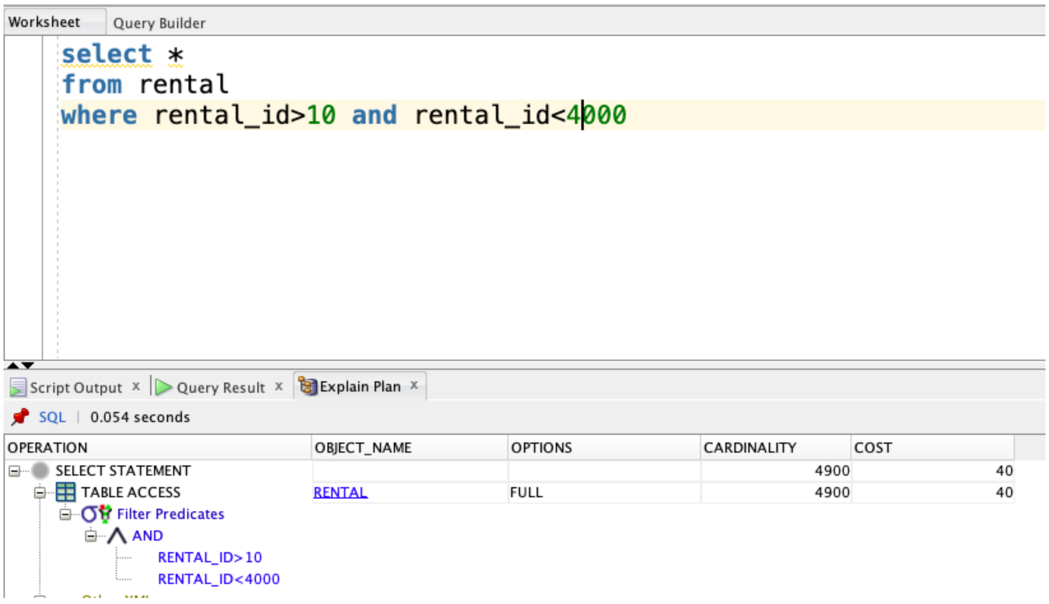
```
select *
from rental
where rental_id>10 and rental_id<100
```

Script Output × | Query Result × | Explain Plan ×

SQL | 0.054 seconds

```
OPERATION                    OBJECT_NAME     OPTIONS                    CARDINALITY   COST
SELECT STATEMENT                                                                 89      3
  TABLE ACCESS               RENTAL          BY INDEX ROWID BATCHED              89      3
    INDEX                    SYS_C00447777   RANGE SCAN                          89      2
      Access Predicates
        AND
          RENTAL_ID>10
          RENTAL_ID<100
```

And,



Worksheet | Query Builder

```
select *
from rental
where rental_id>10 and rental_id<4000
```

Script Output × | Query Result × | Explain Plan ×

SQL | 0.054 seconds

```
OPERATION                    OBJECT_NAME     OPTIONS                    CARDINALITY   COST
SELECT STATEMENT                                                               4900     40
  TABLE ACCESS               RENTAL          FULL                              4900     40
    Filter Predicates
      AND
        RENTAL_ID>10
        RENTAL_ID<4000
```

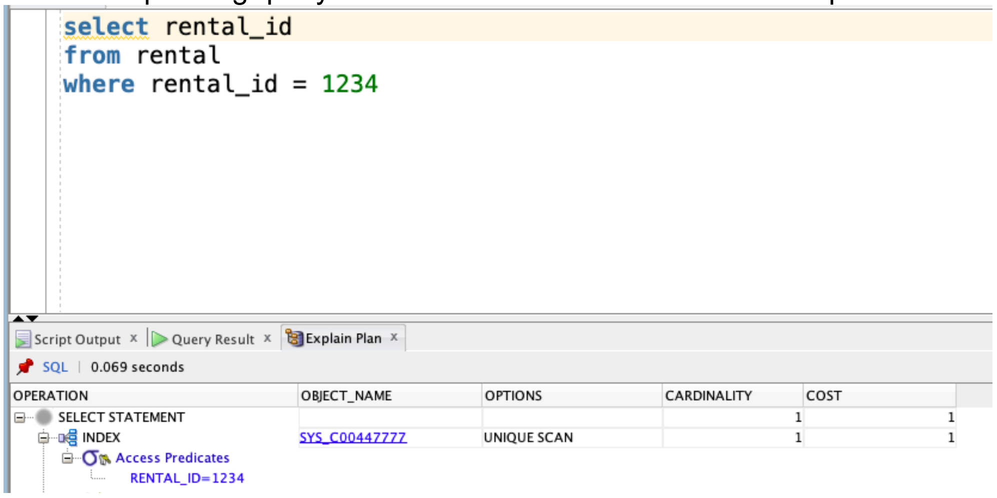
Two queries are near-similar, except the values in filter condition. However, one used the index, and the other didn't.

Why didn't it use the primary key index in the latter case? Discuss with the group and your lab assistant.

Activity 3: Now, run this query.

```
SELECT rental_id
    FROM rental
    WHERE rental_id = 1234;
```

The corresponding query execution plan is as follows:



```
select rental_id
from rental
where rental_id = 1234
```

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	1
INDEX	SYS_C00447777	UNIQUE SCAN	1	1
Access Predicates				
RENTAL_ID=1234				

What is the main difference between the previous queries and this one? How does it impact the query execution plan?

Discuss with the group and your lab assistant.

Activity 4: Exercise
There is no index on the `customer_id` attribute.
Run the following query and obtain the execution plan.

```
SELECT *
    FROM rental
    WHERE customer_id = 12;
```

Create **a B+ tree index** (called `customer_id_index`) on this attribute.

Run the previous query again and obtain a new execution plan.

Activity 5: Extension Activity
Oracle uses hash indexing in a different way, in combination with clustering multiple tables that are commonly combined together to produced joined results.
For example, in the movies database, director and movie tables are joined very frequently using dirnumb attribute. So, Oracle allows you to store these two tables clustered together using a hash key (dirnumb). This allows faster execution of queries where the joined results are produced.

Explore the use of hash indexing in Oracle table clusters.

3.2 B+ Trees

B+ trees are the most commonly used index structure in databases.

The disk I/O cost of data retrieval depends on a number of factors.

- Height of the tree
- One more disk I/O to access the corresponding data block on table
- If some top levels of the tree is maintained in buffer cache, that will save a few disk I/Os.

Height of the tree depends on the size of the table (number of rows), size of the disk block, fanout of an index node (i.e. how many branches emanating from a node) and the fill factor of index nodes.
Number of rows (N): If the fanout is high, the impact of the table size is quite minimal. For example, in a typical table with 2 million records may require a B+ Tree with 3 levels. The index will only grow to 4 levels only when table increases to 312 million rows!

Size of the disk block: That is determined by the operating system (database system has no control). However, it directly impacts the fanout ration and affects the height of the tree.

Fanout: That's the number of branches in a B+ Tree node. The wider the fanout the shallower the tree. You can try out the B+ Tree animation at: https://www.cs.usfca.edu/~galles/visualization/BPlusTree.html to see the impact of different fanouts. Given the block is fixed, the only way to increase the fanout is to choose smaller keys. If you have added attributes that are less useful, then that will reduce the fanout of index nodes, and directly impacts the index performance.

Fill Factor: In order to gradual growth of tables (and as a result index), we always leave some space in index nodes. That will minimise the need for frequent re-orgnisations (which are very costly). Normally, databases ensure nodes are at-least half full and will be split when they are full. As a ballpark, the fill factor for non-root nodes in a B+ tree will be around 67%.

Based on these parameters, the height (H) of a B+ Tree is given as:

$$H = log_{fanout}\ N$$

	Your Lab assistant will explain to you how to calculate the logarithm of N (to the base of fanout.)

Activity 6: Exercise:
Consider the `ontime` table we used during the lecture. It has 7 million rows. Suppose that you are required to build a B+ Tree index on `serialnum` attribute. This attribute is stored as an integer and 4 bytes long. A disk pointer is also 4 bytes long. The block size of the computer system where this table is hosted is 4096 bytes. The average fill factor of an index node is 67%.

Calculate the height of the `serialnum` index.

How many more rows required to increase the height of the tree by one level?

4 Using SQL Developer as a Database Design Tool

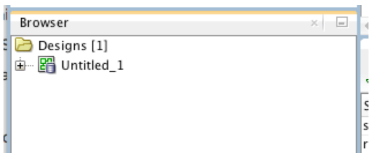
Most of the modern database systems come with a designer tool that help database designers to draw conceptual designs for their databases and build them using auto-generated DDL scripts.

MySQL Workbench, Oracle SQL Developer (and Modeler), Microsoft SQL Server Management Studio, are some of database design tools. There are many other third-party tools are available, mostly portable across many database systems (e.g. dbdesigner.net)

In your first database course you may have used Entity-Relationship Design Tools such as Visio, LucidChart, etc, however, they do not help much to auto-generate DDL scripts (or, reverse-engineer and forward-engineer designs by combining with a schema on a database system).

As a database professional, it is an essential skill to be able to use few of these design tools. In the second part of this tute/ lab session, we do some design activities with Oracle SQL Developer. In the first milestone of your first assignment, you will be assessed on your skills on the effective use of SQL Developer, in particular, ER modelling and DDL generation.

1. Open SQL Developer Data Modeller, by following:
   View → Data Modeler → Browser
2. A new Browser pane will appear in the left of the SQL Developer window.



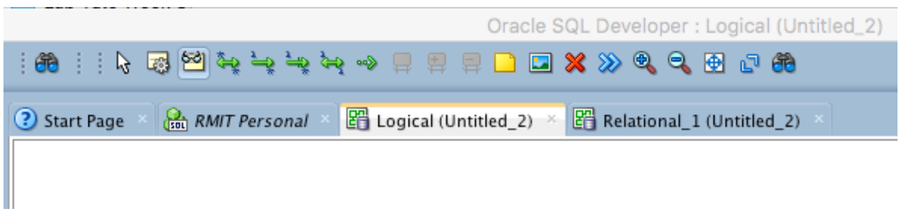
3. Right click on Designs (see above diagram) and on the dialog box, click "New Design".

   This will open up two new tabs on your main work area. One tab is named "Logical" and the other named as "Relational".

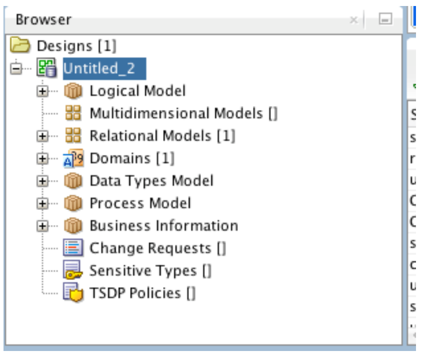
   "Logical" design panel is where you draw your Entity-Relationship model. As outlined on the preparation section, it uses Barkers Notation.

   "Relational" design panel is used to generate the physical design of the model you created on the "Logical" panel.

4. On the Logical Design panel, draw an Entity-Relationship model, to represent the business rules and functional requirements given below.

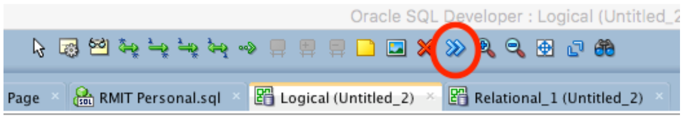

Oracle SQL Developer : Logical (Untitled_2)
Start Page
RMIT Personal
Logical (Untitled_2)
Relational_1 (Untitled_2)

5. Once all the entities, relationships and their attributes are entered, save the diagram.
   On the Browser right click on the unsaved design (in my example, it is named "Untitled_2", and it may differ in your work area).

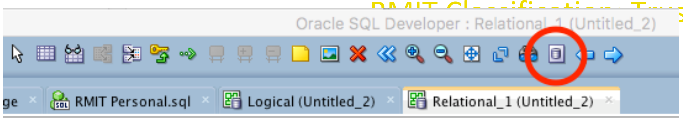


   On the dialog box, click on "Save Design" and save it on a folder that you can access.

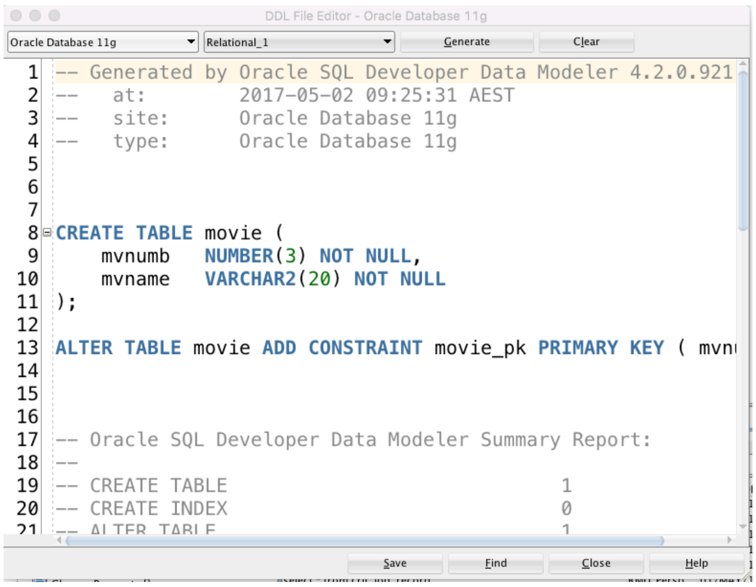
6. Click on the "Engineer to Relational Model" button on the top of the window.


Oracle SQL Developer : Logical (Untitled_2
Page
RMIT Personal.sql
Logical (Untitled_2)
Relational_1 (Untitled_2)

   This will open up a new dialog box.
   Click on "Engineer" button at the bottom of the dialog box, and it will generate the relational model on the "Relational Design" tab.
7. On the "relational Design" tab, at the top of the window, click on "Generate DDL" button.


Oracle SQL Developer : Relational_1 (Untitled_2)
RMIT Personal.sql
Logical (Untitled_2)
Relational_1 (Untitled_2)

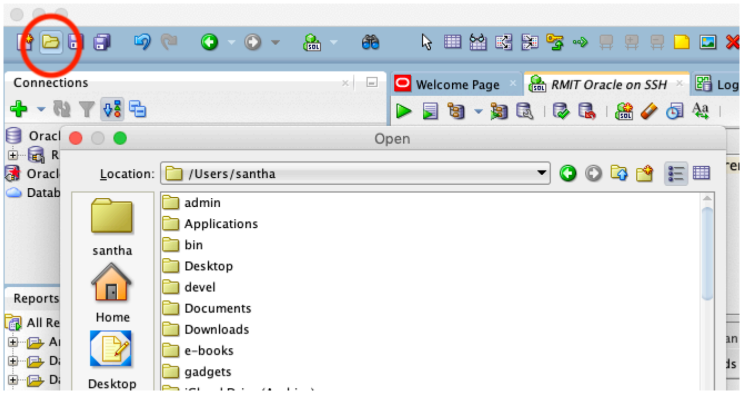
A dialog box will appear.
On the top of the dialog box, click on "Generate" button. Another dialog box will appear, and click "OK". This will Generate the required DDL script to build the database schema.



```
-- Generated by Oracle SQL Developer Data Modeler 4.2.0.921
--   at:        2017-05-02 09:25:31 AEST
--   site:      Oracle Database 11g
--   type:      Oracle Database 11g



CREATE TABLE movie (
    mvnumb   NUMBER(3) NOT NULL,
    mvname   VARCHAR2(20) NOT NULL
);

ALTER TABLE movie ADD CONSTRAINT movie_pk PRIMARY KEY ( mvnu



-- Oracle SQL Developer Data Modeler Summary Report:
--
-- CREATE TABLE                             1
-- CREATE INDEX                             0
-- ALTER TABLE                              1
```

Save the DDL script where SQL Developer can access it in the future.

8. Go back to your SQL Editor panel. Using the DDL script you generated and saved, build a database schema.



Activity 7: Exercise:

Business Operations and Functional Requirements of a Manufacturing Company
A manufacturing company produces products. The following product information is stored: product name, product ID and quantity on hand. These products are made up of many components. Each component can be supplied by one or more suppliers. The following component information is kept: component ID, name, description, suppliers who supply them, and products in which they are used.

Create an ERD to show how you would track this information.
Show entity names, primary keys, attributes for each entity, relationships between the entities and cardinality.

Assumptions

- A supplier can exist without providing components.
- A component does not have to be associated with a supplier.
- A component does not have to be associated with a product. Not all components are used in products.
- A product cannot exist without components.