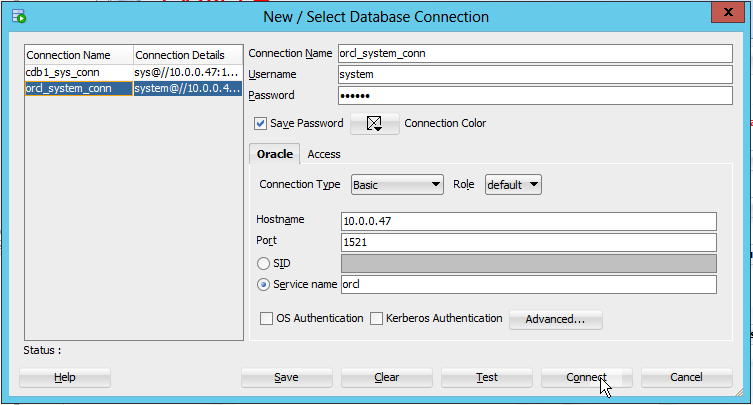
**Create a user account with SQL Developer**

1. Connect **orcl** as **system**.



your VM IP

password: oracle

2. Enter the following 3 statements onto the Worksheet and run them, do not forget to replace jcxxxxxx by your jc username:

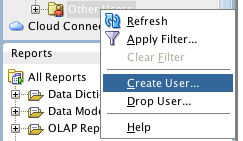
DROP USER app\_jcxxxxxx CASCADE;

DROP TABLESPACE appts\_jcxxxxxx INCLUDING CONTENTS AND DATAFILES;

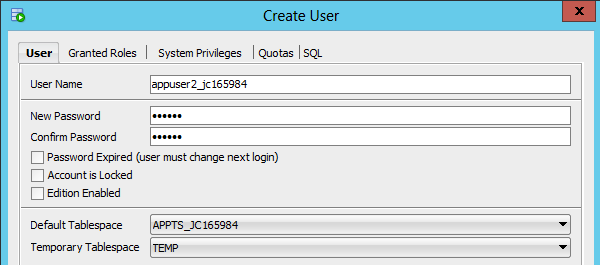
CREATE SMALLFILE TABLESPACE appts\_jcxxxxxx

DATAFILE 'appts\_jcxxxxxx.dbf' SIZE 100M AUTOEXTEND ON NEXT 10M;

3. Right-click **Other Users** --> **Create User**.



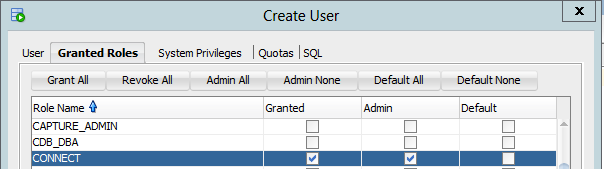
Q1. Use the following pictures to create a user account named **app\_jcxxxxxx**, jcxxxxxx is your jc username. Write down the SQL statements to create the user.

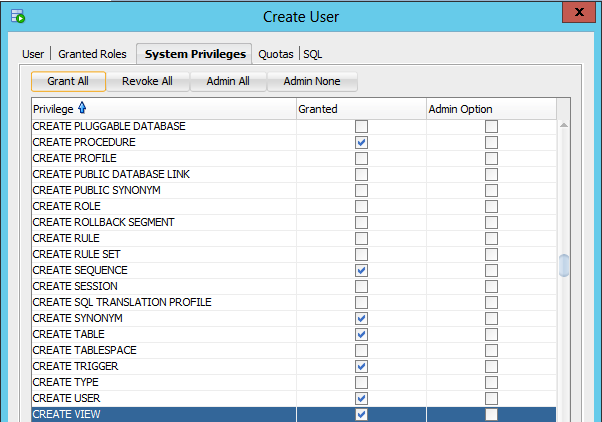


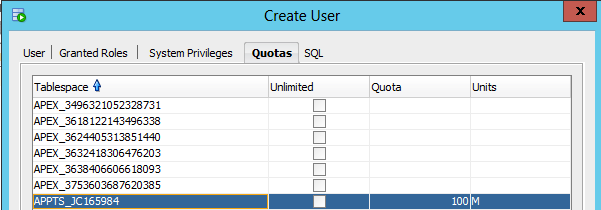
appts\_jcxxxxxx

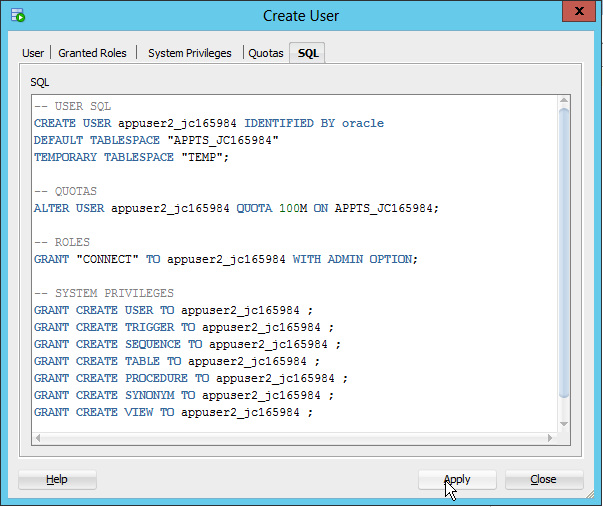
app\_jcxxxxxx

your password – does not need to be oracle









**Database Design with Data Modeler**

Assume that the user **app\_jcxxxxxx** need to design a simple database. He comes up with the following logical design:

Entity: **ITEMS**

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Datatype** | **Other Information** |
| item\_id | NUMBERIC, Precision=4 | Primary UID |
| item\_name | VARCHAR, Size=30 | Mandatory |
| item\_qty | NUMBERIC, Precision=4 | Mandatory |
| item\_reorder\_point | NUMBERIC, Precision=4 | Mandatory |

Entity: **ORDERS**

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Datatype** | **Other Information** |
| order\_id | NUMBERIC, Precision=4 | Primary UID |
| item\_id | NUMBER(4) | Mandatory; Foreign key to the entity ITEMS |
| order\_date | DATE | Mandatory |
| order\_qty | NUMBERIC, Precision=4 | Mandatory |

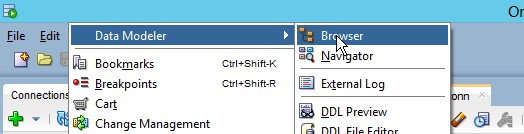
We are going to use **Data Modeler** to develop a logical model and then relational model (E-R diagram) for the 2 entities and then to generate DDL statements to create the tables.

**Oracle Data Modeler** is embedded in SQL Developer as an extension**.** It is a free graphical tool that enhances productivity and simplifies data modelling tasks. Using **Oracle SQL Developer Data Modeler** users can create, browse and edit logical, relational, physical, multi-dimensional, and data type models. The **Data Modeler** provides forward and reverse engineering capabilities and supports collaborative development through integrated source code control. The **Date Modeler** can be used in both traditional and in Cloud environments.

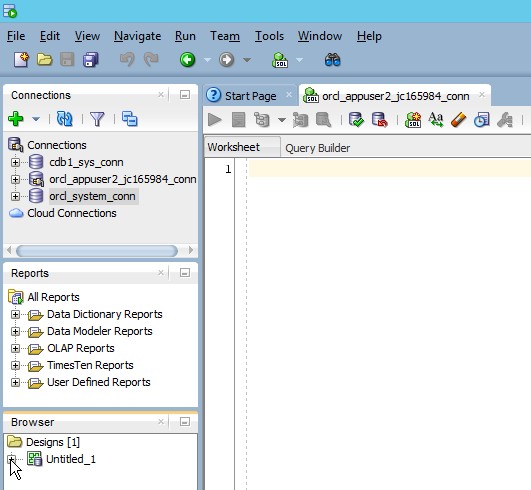
(More info. on <http://www.oracle.com/technetwork/developer-tools/datamodeler/overview/index.html>)

In SQL Developer, to work with **Data Modeler**, you need to enable its environment.

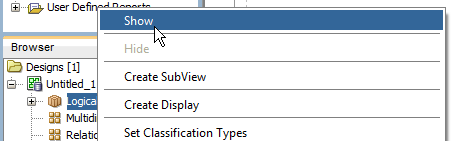
1. On the main menu of SQL Developer, click **View** 🡪 **Data Modeler** 🡪 **Browser**



You will see the **Browser** window is added under the **Connections** and **Reports** windows on the left panel of SQL Developer.



2. Click [+] to expand the **Untitled\_1** 🡪 right-click **Logical Model** 🡪 **Show**

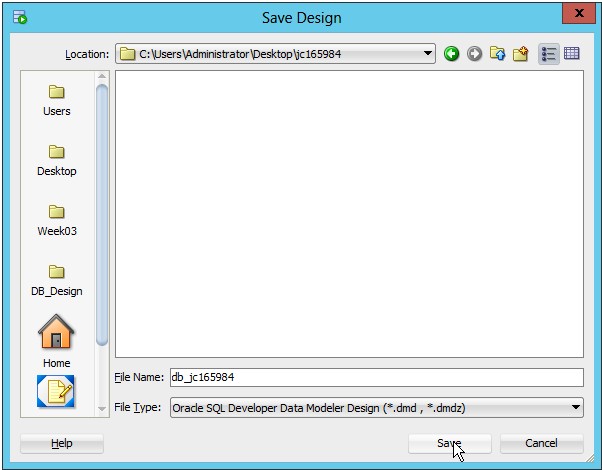


The stage of logical design can be seen as follows:

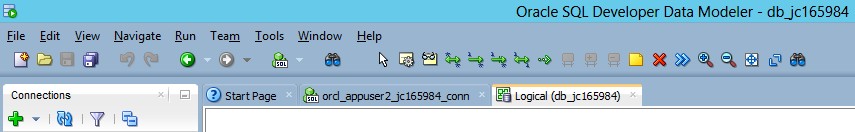


3. Before creating the E-R model, you should give the model a meaningful name, for example **db\_jcxxxxxx** (jcxxxxxx is your jc username) and save it into your drive or USB. You should create a folder named **jcxxxxxx** and save all your work in this folder.

4. Click **File** 🡪 **Data Modeler** 🡪 **Save** or **Save As**



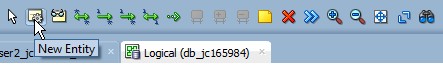
You can see the name **Untitled\_1** is updated to **db\_jcxxxxxx**.



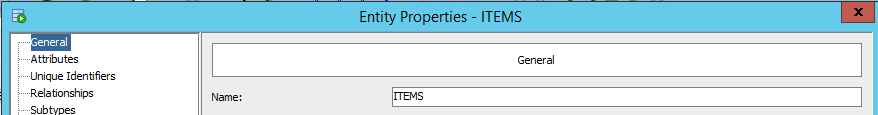
We are now going to create 2 entities based on the given information and also establish the relationship between them.

**Logical Model - Create Entities**

1. Click the **New Entity** icon and then click anywhere on the stage 🡪 the **Entity Properties** dialog box is displayed.



2. Give the name of entity as ITEMS.

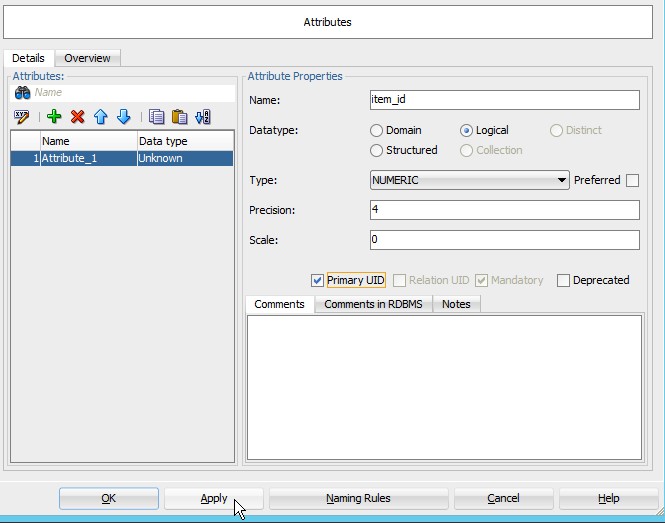


Entity: **ITEMS**

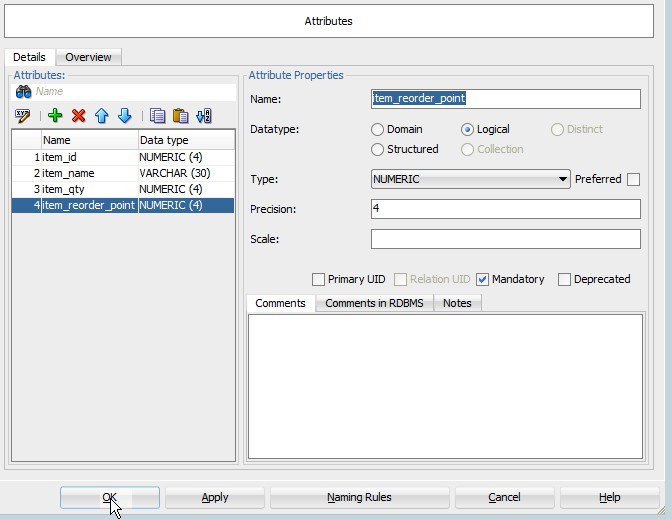
|  |  |  |
| --- | --- | --- |
| **Attribute** | **Datatype** | **Other Information** |
| item\_id | NUMBERIC, Precision=4 | Primary UID |
| item\_name | VARCHAR, Size=30 | Mandatory |
| item\_qty | NUMBERIC, Precision=4 | Mandatory |
| item\_reoder\_point | NUMBERIC, Precision=4 | Mandatory |

3. Click **Attributes** on the left 🡪 The **Attribute Properties** dialog box is displayed.

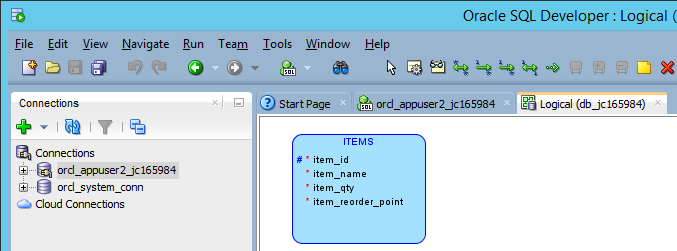
4. And then click the  icon to define attributes of ITEMS:



Finish entering values; do not forget to click **Apply** before adding another attribute.



When you finish the last attribute, click **Apply** and then click OK. The entity ITEMS is displayed on the Logical stage. Double-click on the entity if you want to see the details or you want to modify it.

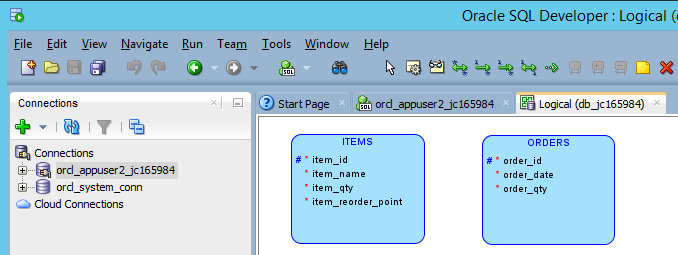


Follow the same steps 1-4 you can continue defining the second entity ORDERS.

Entity: **ORDERS**

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Datatype** | **Other Information** |
| order\_id | NUMBERIC, Precision=4 | Primary UID |
| ~~item\_id~~ | ~~NUMBER(4)~~ | ~~Mandatory; Foreign key to ITEMS~~  **SKIP** |
| order\_date | DATE | Mandatory |
| order\_qty | NUMBERIC, Precision=4 | Mandatory |

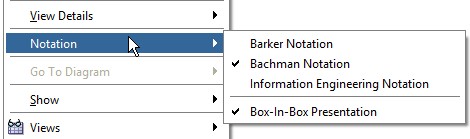
Note: We do not need to define the attribute **item\_id** . It is a foreign key column referring to the PK column in the ITEMS entity. This attribute will be automatically generated when we establish the relationship between the 2 entities.

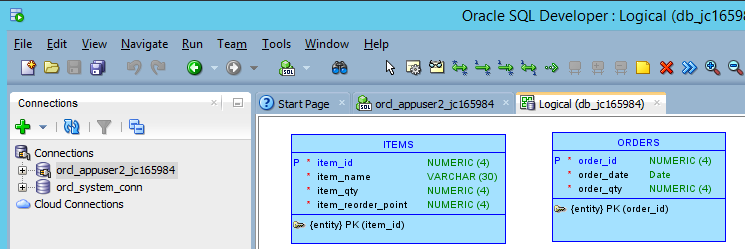


**Logical Model - Create Relationships among entities**

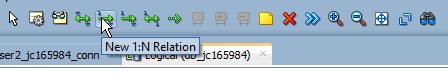
As given information, there is a relationship between **item\_id** (PK) in ITEMS and **item\_id** (FK) in ORDERS. It is a one-to-many relationship. One item can be ordered in 0 or many different orders. For simplicity, we assume that one order just contains one item; usually, one order can contain many different items.

1. Change the notation style to **Bachman** notation. Right-click anywhere on the stage 🡪 **Notation** 🡪 **Bachman Notation**.

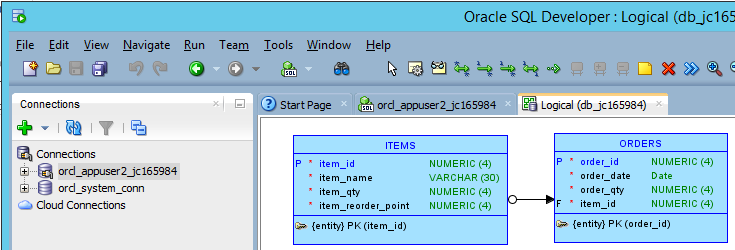




2. Select the **New 1:N Relation** icon. Firstly, click inside the ITEMS entity (the PK table) and then click inside the ORDERS (the FK table). A line with an arrowhead will be drawn from ITEMS to ORDERS.



3. Just click OK when the **Relationship Properties** dialog box pops up.



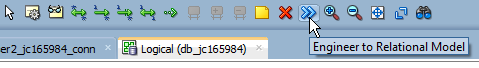
You can see the new attribute **item\_id** is added in ORDERS entity.

When you completely establish the relationships among all the entities we will use the logical model to generate the relational model.

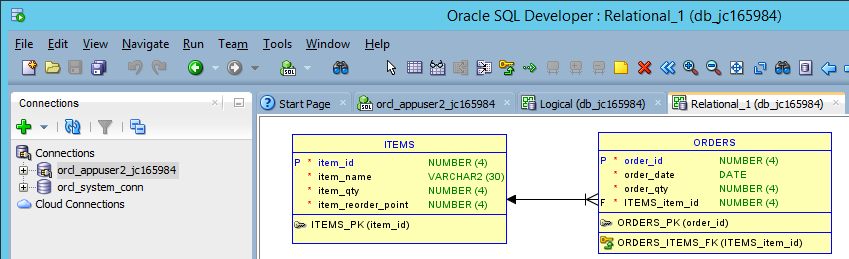
Note: Always create the PK entity before the FK entity and omit all the FK attributes. The FK attributes will be automatically generated when we establish the relationship among the 2 entities.

**Generate Relational Model from Logical Model**

1. Select the **Engineer to Relational Model** icon.



2. When **Engineer to Logical Model** dialog box pops up, you just simply click the **Engineer** button. The relational model named **Relational\_1** is generated as follows:



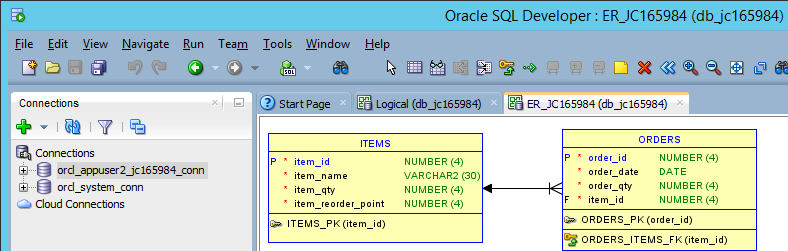
In ORDERS table, you can see the column **item\_id** is changed to **ITEMS\_item\_id.** We will change it back to **item\_id**.

3. Double-click on ORDERS table 🡪 **Table Properties**

4. Click **Columns** 🡪 locate and change **ITEMS\_item\_id** back to **item\_id**.

5. Click **Apply** and then click OK.

We are going to give a meaningful name for the relational diagram. Right-click on the stage of the **Relational\_1** 🡪 **Properties** 🡪 change the name **Relational\_1** to **ER**\_**JCXXXXXX** (JCXXXXXX is your jc username); Select RDBMS Type/Site as **Oracle Database 12c**. Click OK.

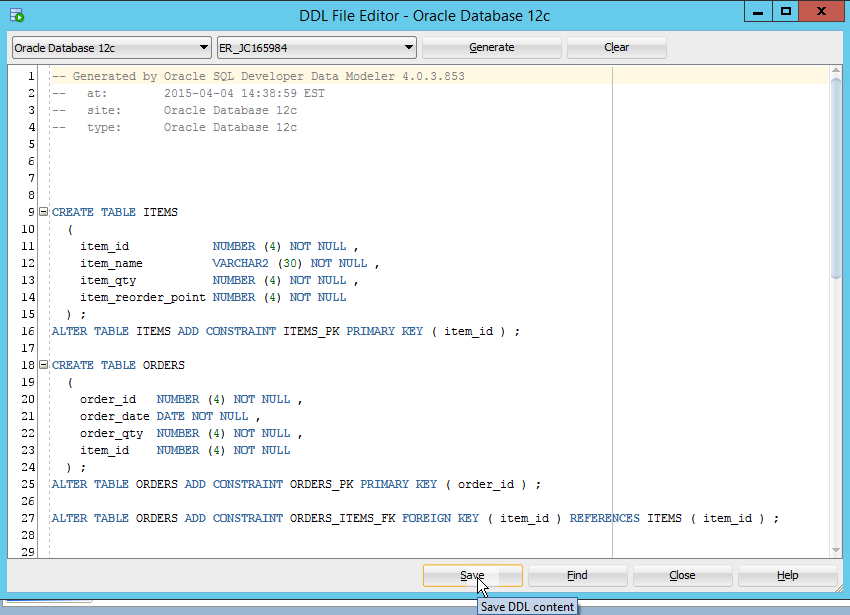


**Relational Model: Generate DDL Statements**

1. Select the **Generate DDL** icon 🡪 **Generate** 🡪 OK



After clicking OK, **Data Modeler** will generate DDL statements to create the tables for you as follows:



2. Click **Save** to save it to a file named **create\_tables\_jcxxxxxx.sql** in your folder.

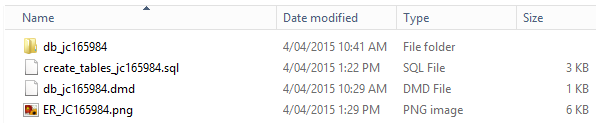
3. Close **DDL File Editor**.

**Print Diagram**

4. Click **File** 🡪 **Data Modeler** 🡪 **Print Diagram** 🡪 **To Image File** 🡪 Save to **ER\_jcxxxxxx.png** in your folder.

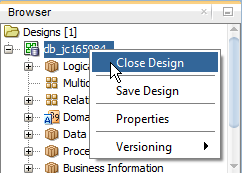
5. Finally, right-click on your model **db\_jcxxxxxx** 🡪 **Save Design** to save your work.

If you have saved your files inside one folder (for example, **jcxxxxxx** folder). The content of your folder should be similar to:



Keep the folder; your lab assistant and your lecturer may ask you about the folder.

You now can close your design. Important! Make sure **Save Design** before **Close Design**.



And then close the Logical and Relational tabs.

You can open your design again by clicking **File** 🡪 **Data Modeler** 🡪 **Open** and locate your .dmd file to open.

Q.2

Write down the content of **create\_tables\_jcxxxxxx.sql**.

Paste the ER diagram here.

**SQL Developer: Create Tables**

We will use the DDL generated code from **Data Modeler** to create the tables.

Firstly, a final look to improve the database design, we assume that the design was revised and we strengthened all the constraints and gave default values. The final design is made as follows:

Table 1: **ITEMS**

|  |  |  |
| --- | --- | --- |
| **Column** | **Data Type** | **Comments** |
| item\_id | NUMBER(4) | Primary key; greater than 0 |
| item\_name | VARCHAR2(30) | Null not allowed; unique |
| item\_qty | NUMBER(4) | Null not allowed; equal or greater than 0 |
| item\_reorder\_point | NUMBER(4) | Null not allowed; equal or greater than 0 |

Table 2: **ORDERS**

|  |  |  |
| --- | --- | --- |
| **Column** | **Data Type** | **Comments** |
| order\_id | NUMBER(4) | Primary key; greater than 0 |
| item\_id | NUMBER(4) | Null not allowed; Foreign key to the table ITEMS |
| order\_date | DATE | Null not allowed; Default: SYSDATE |
| order\_qty | NUMBER(4) | Null not allowed; greater than 0; Default: 1 |

1. To create the above 2 tables by the user **app\_jcxxxxxx**, we make a new connection with this user. You can disconnect the **orcl\_system\_conn**.

Connection Name: **orcl\_app\_jcxxxxxx\_conn**

Username: **app\_jcxxxxxx**

Password: **your password**

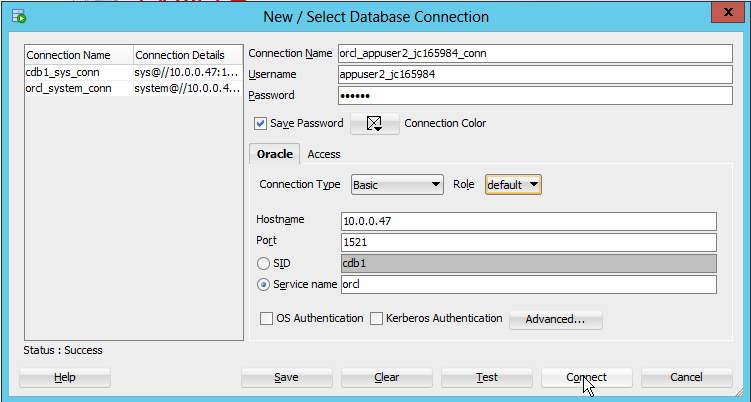
Connection Type: **Basic**

Role: **default**

Hostname: **VM’s IP**

Port: **1521**

Service name: **orcl**



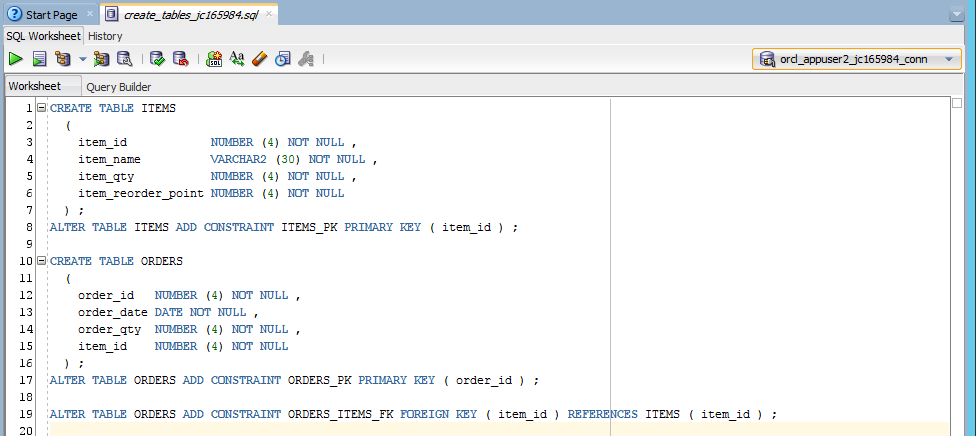
app\_jcxxxxxx

orcl\_app\_jcxxxxxx\_conn

Secondly, we will open the file **create\_tables\_jcxxxxxx.sql** to tune up the code in order to strengthen all the required constraints and default values stated in the final design.

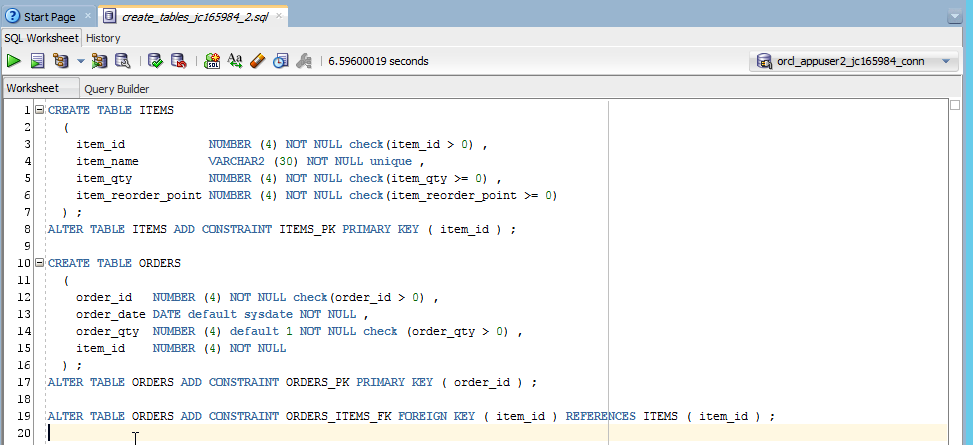
2. Click **File** 🡪 **Open** 🡪 locate **create\_tables\_jcxxxxxx.sql** and open it. SQL Developer will open the file on a new tab.

3. On the top right corner, select the connection **orcl\_app\_jcxxxxxx\_conn**:



4. Adding the followings into the code:

* Line 3: check(item\_id > 0)
* Line 4: unique
* Line 5: check(item\_qty >= 0)
* Line 6: check(item\_reoder\_point >=0)
* Line 12: check (order\_id > 0)
* Line 13: default sysdate
* Line 14: default 1 check(order\_qty >0)



5. Save the modified code to another file named **create\_tables\_jcxxxxxx\_2.sql.**

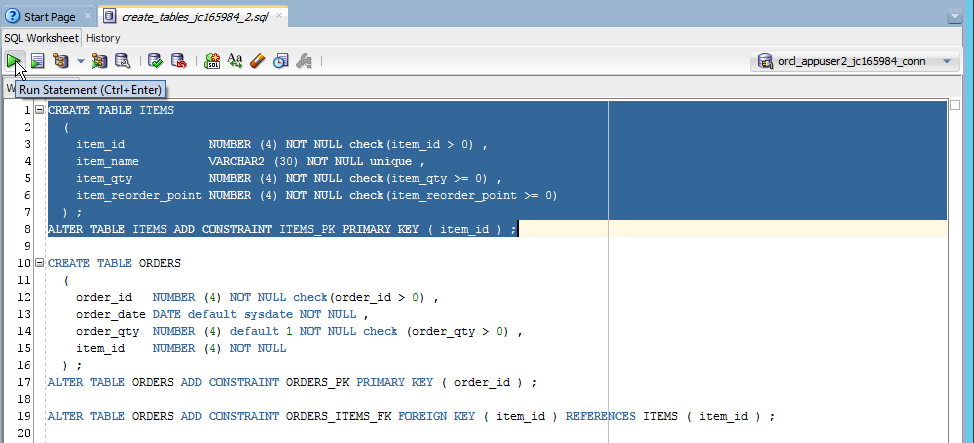
6.Close the file **create\_tables\_jcxxxxxx.sql** and say No for saving changes**.**

You can run the code now.

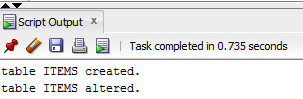
However, please double-check all typing before executing the code.

We will create one table at a time.

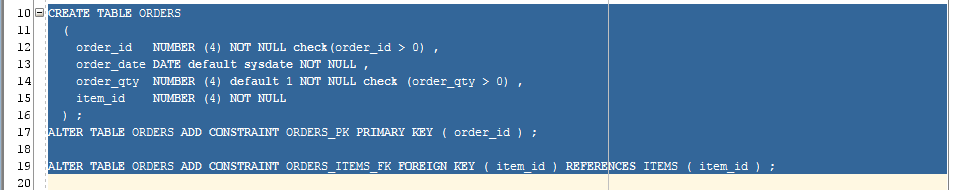
7. Highlight the ITEMS code (Line 1 to Line 8) and click  to run:

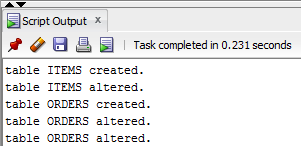


If nothing is wrong, you will see the results:

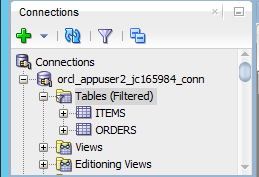


8. Highlight the ORDERS code (Line 10 to Line 16) and click  to run; Highlight the alter statements (Line 17 to Line 19) and click  to run.





You now can see the 2 tables appearing on the **app\_jcxxxxxx** schema:



Q.3

Write down the content of **create\_tables\_jcxxxxxx\_2.sql**.

**SQL Developer: Import Data**

1. Download csv file **items.csv** in Practicals/Week03-04 (right-click on the file 🡪 Save As).

2. Right-click on ITEMS table 🡪 **Import Data** 🡪 locate items.csv 🡪 **Open**.

3. Go through a number of steps as follows:

**Data Preview**: Tick Header; Format: csv; Encoding: UTF-8; Left Enclosure: none.

**Import Method**: Insert.

**Choose Columns**: Do nothing if all 4 columns are selected.

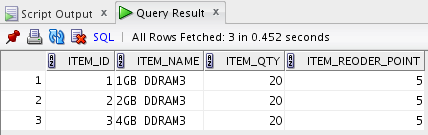
**Column Definitions**: Check the matching between **Source Data Columns** and **Target Table Columns**

**Finish**: Verify 🡪 Finish.

4. Enter the following statement on the Worksheet and click  to check if the records are inserted into ITEMS table.

select \* from items;

You should see:



**SQL Developer: Insert Data**

We often define sequences to populate values for PK columns. For example, we will create a sequence for **order\_id** in ORDERS table.

1. Enter the following statement on the Worksheet and click  to create a sequence for **order\_id** in ORDERS table.

CREATE SEQUENCE orders\_seq INCREMENT BY 1 NOCACHE NOCYCLE;

2. Enter the following statements on the Worksheet, highlight all and click  to insert records into ORDERS table.

insert into orders (order\_id, item\_id, order\_date,order\_qty)

values(orders\_seq.nextval,1, to\_date('03/07/2013','DD/MM/YYYY'),5);

insert into orders (order\_id, item\_id, order\_qty)

values (orders\_seq.nextval,2,5);

insert into orders (order\_id, item\_id)

values (orders\_seq.nextval,3);

-- to\_date(): built-in function to convert a string value to a date value. You should always use this function to enter a date value.

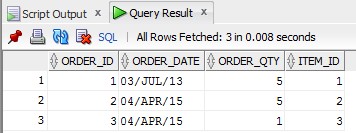
-- date values in 2nd and 3rd statements are omitted, they will take the default current system date

-- order\_qty in 3rd is omitted, it will take the default value of 1

3. Enter the following statement on the Worksheet and click  to check if the records are inserted into ORDERS table.

select \* from orders;

You should see:



We now need to update the ITEMS table to reflect the changes.

4. Enter the following statements on the Worksheet, highlight all and click  to update records in ITEMS table.

update items

set item\_qty = item\_qty - 5

where item\_id = 1;

update items

set item\_qty = item\_qty - 5

where item\_id = 2;

update items

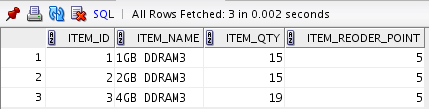
set item\_qty = item\_qty - 1

where item\_id = 3;

5. Enter the following statement on the Worksheet and click  to check if the records are updated in ITEMS table.

select \* from items;

You should see:



In fact, ITEMS records can be updated automatically by trigger. We will learn trigger in Lecture 5. In addition, trigger is also help us to check if we have enough **item\_qty** to proceed the order or to inform us if **item\_qty** is lower than **item\_reorder\_point**.

To test the constraints, we try to insert a number of (invalid) records which are violated the constraints.

For example, in ITEMS table,

6. To test the constraint “**item\_id** must be greater than 0”, insert the following record:

insert into items values (**-2**,'8GB DDRAM3', 20, 5);

Run the statement, you will see an error:

SQL Error: ORA-02290: check constraint (APPUSER2\_JC165984.SYS\_C0011792) violated

7. To test the constraint “**item\_name** must not be null”, insert the following record:

insert into items values (4,**null**, 20, 5);

Run the statement, you will see an error:

SQL Error: ORA-01400: cannot insert NULL into ("APPUSER2\_JC165984"."ITEMS"."ITEM\_NAME")

Q4.

Please write down 1 SQL statement to insert invalid ITEMS record to test PRIMARY KEY constraint.

Please write down 1 SQL statement to insert invalid ITEMS record to test UNIQUE constraint.

Please write down 1 SQL statement to insert invalid ORDERS record to test FOREIGN KEY constraint.

Please write down 1 SQL statement to insert invalid ORDERS record to test NOT NULL constraint.

Please write down 1 SQL statement to insert invalid ORDERS record to test the validity of **order\_qty**.

**SQL Developer: Managing Indexes**

Follow the points on:

<http://www.oracle.com/webfolder/technetwork/tutorials/obe/db/12c/r1/2day_dba/12cr1db_ch8schema/12cr1db_ch8schema.html>

Expand  to follow steps in Viewing Indexes and Creating Indexes.

Q5.

Write down the index name, table name and column name of all indexes in **app\_jcxxxxxx** scheme.

**SQL Developer: Managing Views**

Expand C:\Users\Administrator\Desktop\ScreenHunter\ScreenHunter_47 Apr. 05 00.28.jpg to follow steps in Displaying Views and Creating a View.

Q6.

Write down a SQL statement to create a view to list the names of all ordered items with order\_qty > 1.

**Do not forget to shutdown the VM properly**