Talal Kerbag

9/12/2018

Homework 2

The first five questions in this homework are about:

**Functionality**

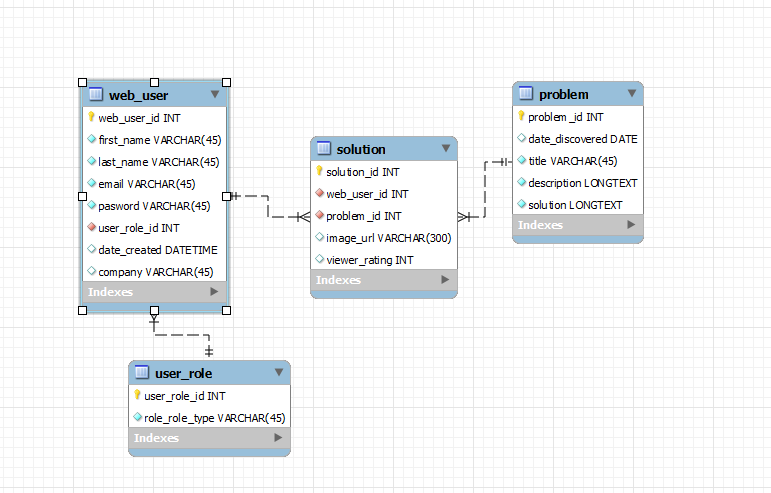
* + IT professional will be able to register, log on, and log off.
  + IT professional will be able to create problems or solve existing ones.
  + IT professional will be able to edit or delete problems that they have created.
  + Admin will be able to insert (edit/delete) associative records.
  + Guest will be able to view problems.

**Marketing Material**

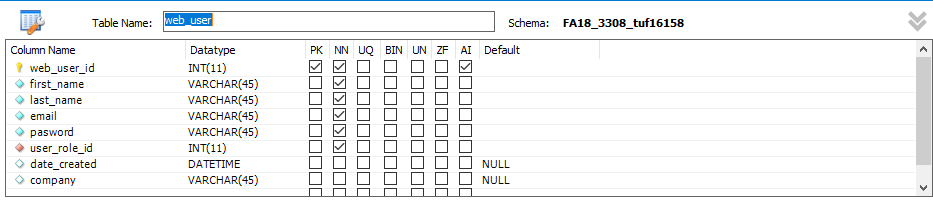
In the information technology infrastructure library there is a volume that speaks very highly of the importance of having a known Error Database in any IT department. One of the main and most important reason for having a KEDB implemented into your business is its downtime reduction capability; if an incident is already reported, a user does not have to wait long for a response from the help desk because a workaround likely already exists.

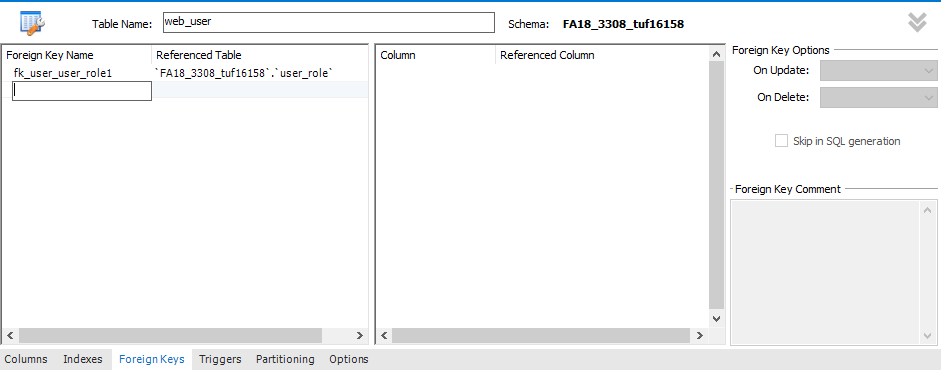
Furthermore, when a workaround already exists, visiting users will not have to troubleshoot on their own. When user is troubleshooting without the help of IT, the user’s productive worktime decreases significantly. This also helps the user use their own skills for their own work.

**Data Model**

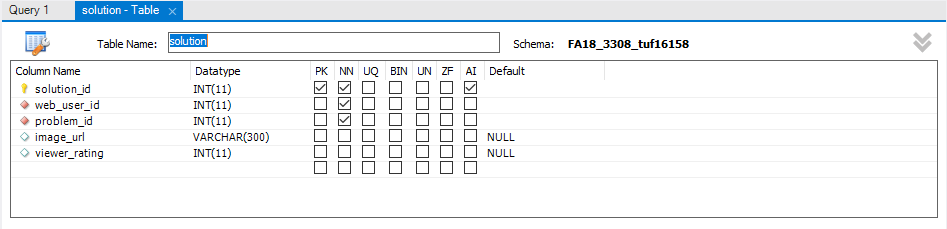


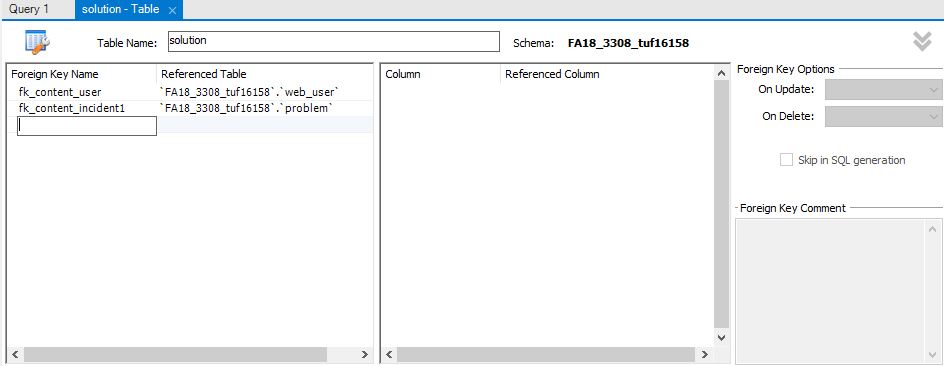
**Table Designs** *AND* **foreign key.**



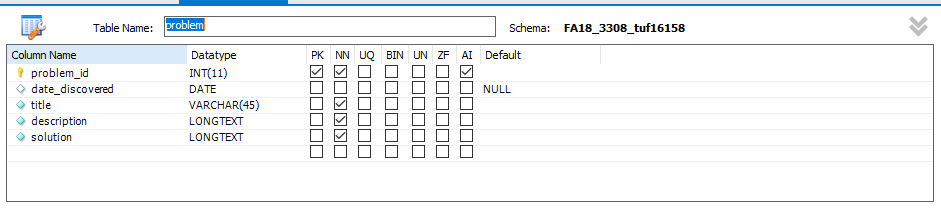


**Solution Table**

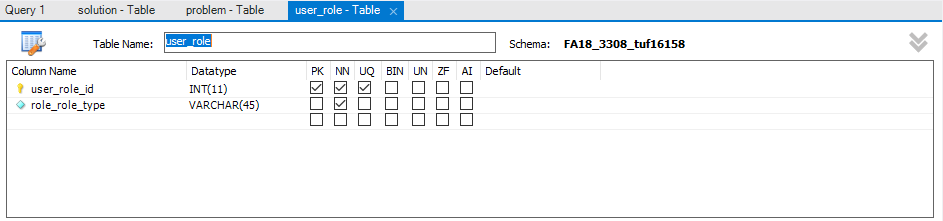




**Problem Table**



**User Role Table**



Question 6 and 7 ask for a lab activity called **Feedback from Classmates** AND **My Feedback.**

**Feedback from Classmates**

From: Guillermo Vazquez

Feedback: The data model had all of its components; web user, user role, associate table and other table. The web user and other table were connected through the associate table.

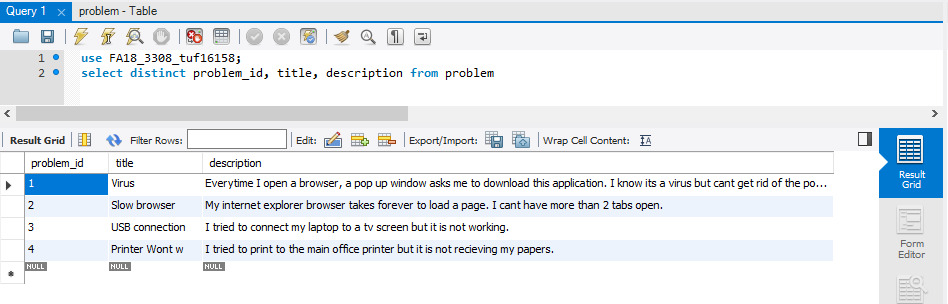
**My Feedback**

To: Guillermo Vazquez

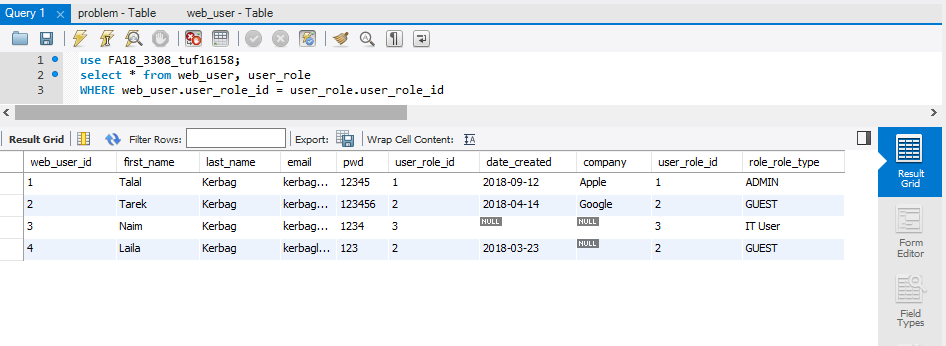
My feedback: Your data model looked good. Had all its four tables. The associate table connected your other table with your web user table.

Question 8 is playing with **SELECT Statements** As shown below.

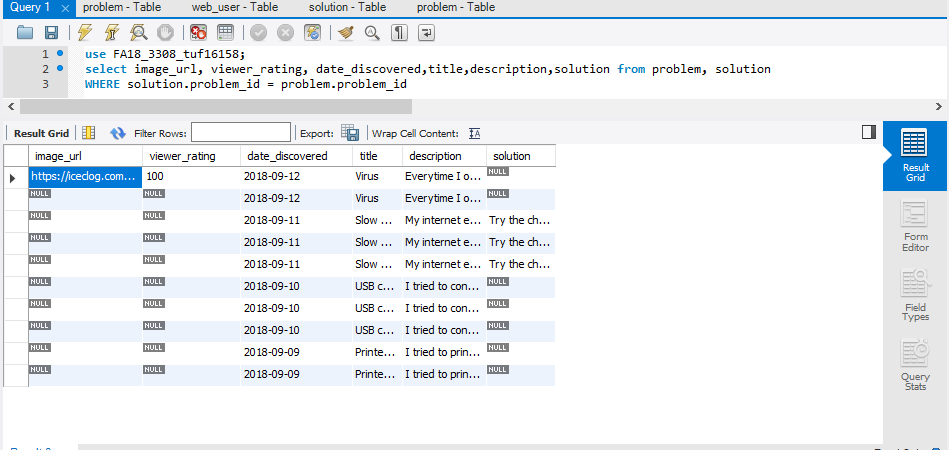
1. List *all of the columns* of your “other” table, ordering the columns in a way that you think users would like to view the data (don't use "select \*"). Sort the data by the first column. Include all records of the "other" table.



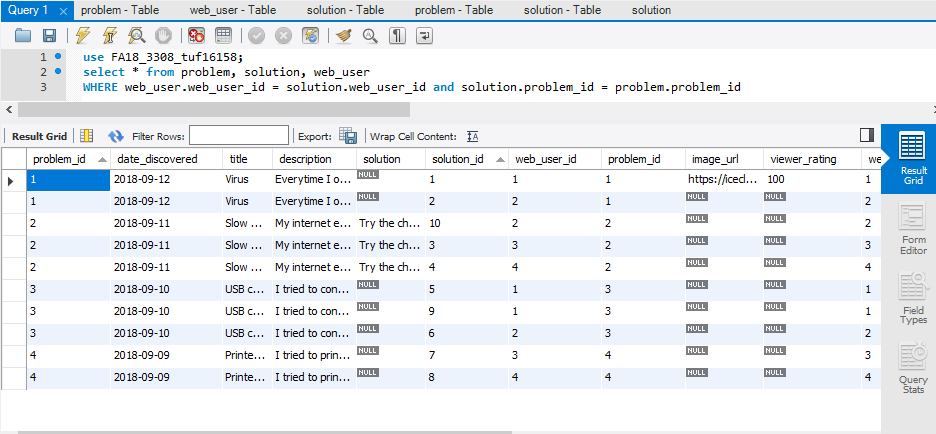
1. Show all the records from your user table joined with your role table. Show the role name first, then the role id, then the email address, followed by all the rest of the columns of your user table (ordered in a way you think users would like to view the data – show the role id once, not twice.) Order the result set by role name, then email address (as a secondary sort). Include all records. There should be as many rows in your result set as you have records in your user table. If you have a lot more (and see duplication), you have forgotten the WHERE clause that joins the two tables together.



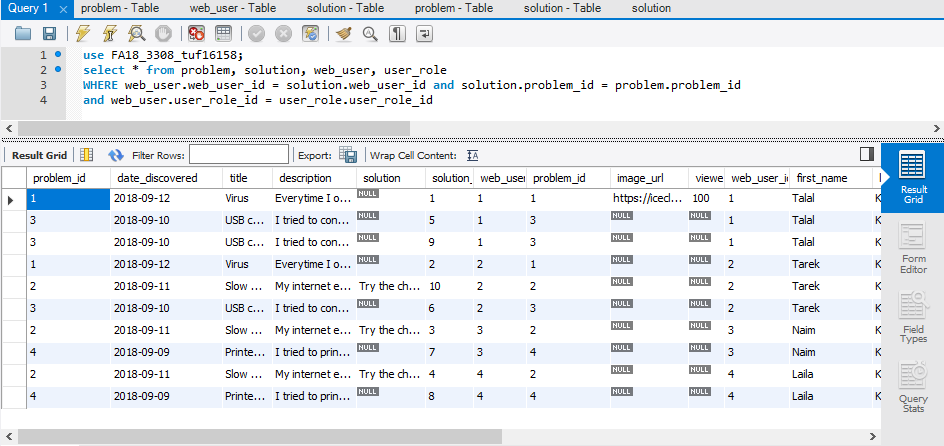
1. Join your associative table with your “other” table. Include all the columns except PK and FK (id) columns. Order the columns in a way you think users would like to see them (not "select \*"). Order the result set by the first two or three columns. There should be as many rows in your result set as there are records in your associative table.

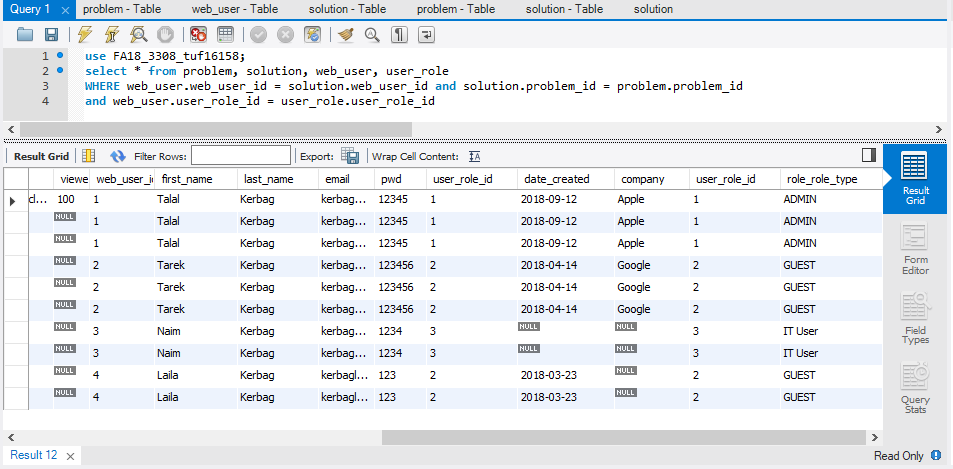


1. Join your associative table with your “other” table and with your user table. Include all the columns except PK and FK (id) columns. Order the columns in a way you think users would like to see them. Order the result set by the first two or three columns. There should be as many rows in your result set as there are records in your associative table. Hint: you need two conditions in your WHERE clause, one for each PK/FK relationship.

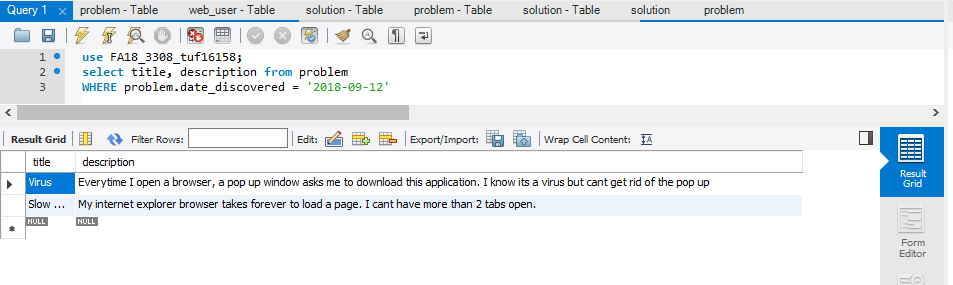


1. Join your associative table with your “other” table and with your user table and with your role table. Include all the columns except PK and FK (id) columns. Order the columns in a way you think users would like to see them. Order the result set by the first two or three columns. There should be as many rows in your result set as there are records in your associative table. Hint: you need THREE conditions in your WHERE clause, one for each PK/FK relationship.





1. Modify your SELECT STATEMENT from item “e”, adding a condition to your WHERE so that you see only the data from one user (selected by user.user\_id). Select a user such that your result set has at least 2 records in it.



1. Modify your SELECT STATEMENT from item “e”, so that it selects one user id, two or more "other" records (using LIKE keyword and % wildcard match), and some condition testing a field from the associative table. Come up with conditions such that your result set has at least 2 records in it. Example of using LIKE:

SELECT \* FROM tableA, tableB

WHERE tableB.tableA\_id = tableA.tableA\_id AND tableB.name LIKE '%temple%'

