Foreign keys and first SQL queries – LAB 3

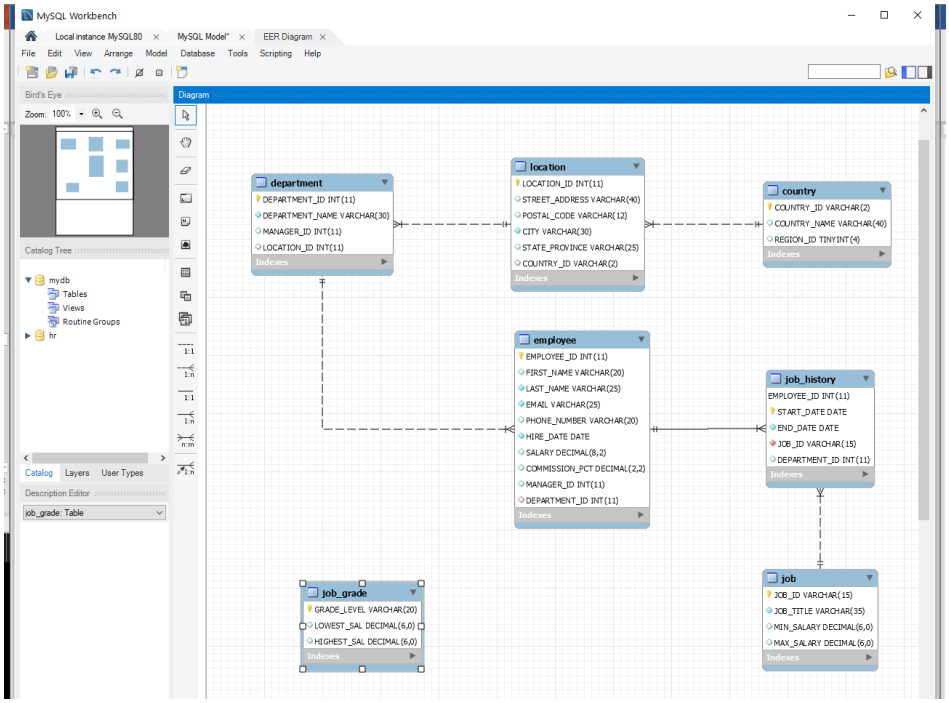
1. Check HR database exists (if not create database HR, import “CreateTables.sql”)

   Solution: Right-click the “hr” database created and select “*set up as default schema*”
   Run the CreateTables.sql script (*File → Run SQL script*)

2. Create the foreign keys (use the script “create_foreign_keys.sql”).

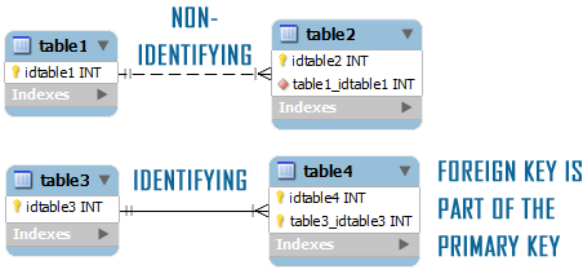
   Solution: Right-click the “hr” database created and select “*set up as default schema*”
   Run the create_foreign_keys.sql script (*File → Run SQL script*)

3. Create ERD from the updated database (use Reverse Engineering, as learned in last class). Look at the relations between tables.



Additional explanation of identifying and non-identifying relationships (source: https://www.eandbsoftware.org/difference-between-identifying-and-non-identifying-relationships-mysql-workbench/):

Non-Identifying Relationships have dotted lines, whereas Identifying Relationships have solid lines in MySQL Workbench. When you create an identifying relationship, the primary key of the child table becomes a foreign key in the parent table. In non-identifying relationships, the primary key of the child table is still included as a foreign key, but it doesn’t get to participate as a primary key. Hence, it is non-identifying. Here is an example for a 1:n relationship:

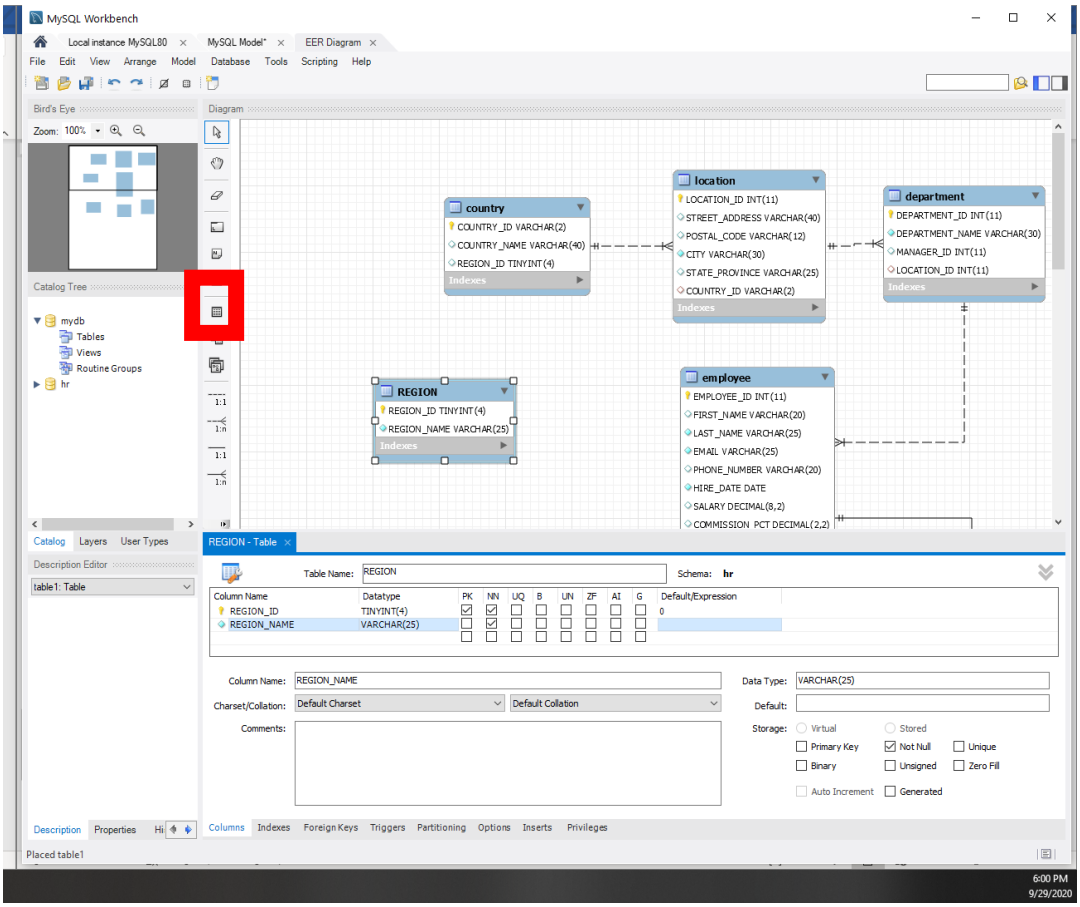


Example: Taking this a bit further, you'll notice that in identifying relationships, it is really not possible to insert a record into the parent table without referencing the child table. So if you had a 1:n relationships between Brands and Products, where a Brand can have many Products, but a Product only has 1 Brand.

If that relationship is identifying, we'd have the Brand Primary Key as part of the Primary Key in Products. Therefore, each Product needs a Brand to be uniquely identified.
If that relationship is non-identifying, the Brand Primary Key is still a Foreign Key in the Product Table, but it's not part of the primary key. We can identify Products uniquely by Product ID alone.

4. After looking at the ERD, the HR department decided that also wants to identify and store the regions where the countries are located (Europe, Asia, etc.). Then, you need to create a new table REGION (using SQL code or the wizard).

Solution with wizard (Select the table icon on the left → click on the canvas → double click on the new table (table1) that appears):

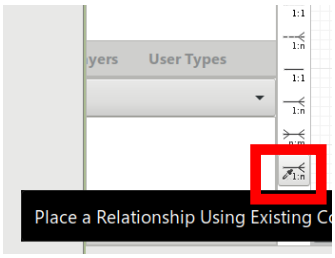


Alternative solution using SQL code (To open a sql query tab: *File → New Query Tab*):

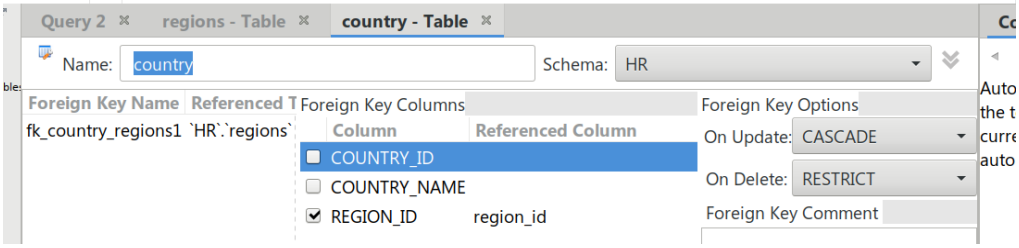
```
CREATE TABLE IF NOT EXISTS `region` (
   `REGION_ID` TINYINT NOT NULL,
   `REGION_NAME` varchar(25) DEFAULT NULL,
   PRIMARY KEY (`REGION_ID`)
) ;
```

5. Create (in workbench ERD) a foreign key to link country and region, where region is the parent table (Make sure region_id in both tables have equal type). Set up "On Update = cascade" and "On Delete = restrict"

Solution: Click on the bottom in the red square, then click on the COUNTRY table, and then on the REGION table.

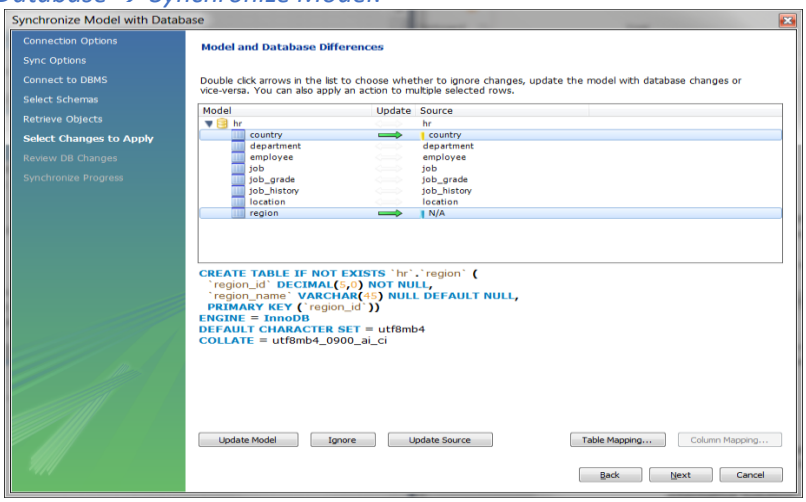


Check if the new foreign key exists, modify "On Update" and "On Delete"



6. Propagate the changes to the physical database.

Solution: Go to *Database → Synchronize Model*.



7. Insert four world regions using sql code: Europe, Americas, Asia, Middle East and Africa. (To open a sql query tab go to *File → New Query Tab*).

Solution using SQL code (To open a sql query tab: *File → New Query Tab*):

```
INSERT INTO `region` (`REGION_ID`, `REGION_NAME`) VALUES
(1, 'Europe'),
(2, 'Americas'),
(3, 'Asia'),
(4, 'Middle East and Africa');
```

8. You receive a new update from the business user; the human resources department is not going to have job grades any longer. Then, you need to delete the table job_grade.
   Solution using SQL code (To open a sql query tab: *File → New Query Tab*):

```
DROP TABLE IF EXISTS JOB_GRADE;
```

9. Insert the data into your tables (use the script "insertHRdata.sql").

   Solution: Run the insertHRdata.sql script (*File → Run SQL script*)

10. Write a query to display all the employee details.

```
select * from employee;
```

11. Write a query to calculate the square root of (52.32 * 96.3)

```
select sqrt(52.32 * 96.3) as square_root;
```

12. Write a query to get the last name and first name of the first 10 employees.

```
select e.last_name, e.first_name from employee e limit 10;
```

13. Write a query to get the difference between the maximum and minimum salary from employees.

```
select max(e.salary) - min(e.salary) from employee as e;
```

14. Write a query to get the last name (in upper case) of employees and the salary. Order by salary from the maximum to minimum.

```
select upper(e.last_name), e.salary from employee as e
order by e.salary desc;
```