Guy Taggar

Databases and SQL - Final Project

Data Schema (rbac-schema.sql)

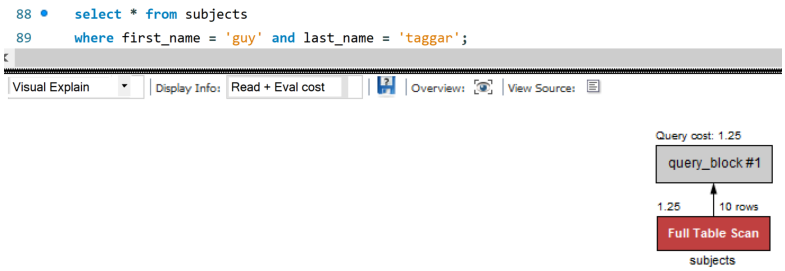
As negative 'id' values are meaningless, every 'id' column was defined as UNSIGNED on every table, allowing us to expand the maximum capacity of SMALLINT type.

Permissions TABLE – In addition to all FK constraints, I decided to address every (operation to resource) pairing as UNIQUE by adding the "id_operation_resource" UNIQUE KEY. This was done in order to avoid duplicate assignments, which could practically result in the allowance of 2 or more id's pointing to the same permission.

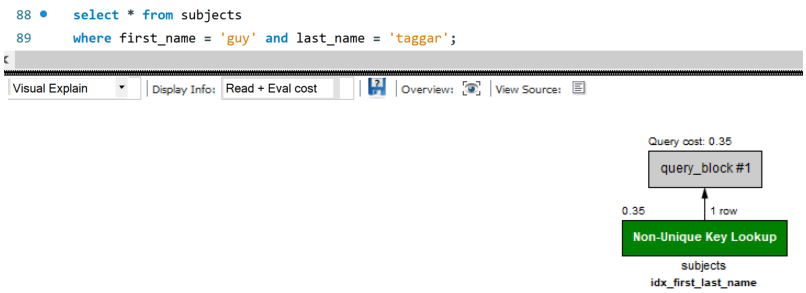
Roles TABLE – The chosen inheritance method was to add a dedicated column, namely "inherit_from", which specifies the child's role. In order to maintain data integrity, a relevant CONSTRAINT "fk_roles_roles " was implemented, keeping the child assignment reasonable.

Subjects TABLE – I've decided to use VARCHAR type on phone numbers, especially because INT would mess them up, as phone numbers often start with '0'.
Even tough email addresses are UNIQUE, I've decided to add an 'id' column as an important corollary from our course.
In addition to subject id's and their email addresses, both of them having natural indexes (by their uniqueness), I have also decided to add the "idx_first_last_name" INDEX which is found at the bottom of the schema file. This was made after assuming our rbac customers will often find it easier to target their subjects by their full name instead of their emails or id's.
A full name search comparison is shown below:

Before the index



After the index:



Remark – Too many unnecessary indexes are more harmful than useful.

Guy Taggar

Sample Data (rbac-data.sql)

While trying to achieve the greatest variety I could get, the sample data doesn't contain all possible combinations out there. Even so, I sure hope It contains the more crucial ones.
The data speaks for itself, so its pretty easy to see my intentions just by viewing it, but I'll still relate to the main ideas:

Role INSERT – Here we have an Admin which inherits from a Lecturer, which inherits from a Teaching Assistant. Both the Teaching Assistant and the Grader don't inherit from any role. The Student inherits from a Guest, which is the most basic role.

Subject_Role INSERT – In order to put the last query (2.4.3) for the true test, I've inserted all possible date options.
Also, one subject wasn't assigned to any role.

Queries (rbac-data.sql)

2.4.1 – As we can see, both subject with ids '5','6' have no direct permissions. '6' doesn't have any because he doesn't have a role, while '5' doesn't have any because a Lecturer doesn't have any direct permissions.

2.4.2 – Including the inheritance permissions was probably the toughest part of this project. Only after working closely with the mwb table file I created, I finally got it. The main part was to actually understand what I was doing. I finally managed to create the following query:

subject_id	first_name	Role 1	Role 2	Role 3	Permission Name
1	guy	Admin	Lecturer	Teaching Assistant	Add Assignment
1	guy	Admin	Lecturer	Teaching Assistant	Edit Assignment
1	guy	Admin	Lecturer	Teaching Assistant	Delete Assignment
1	guy	Admin	Lecturer	Teaching Assistant	Add Page
1	guy	Admin	Lecturer	Teaching Assistant	Add Course
1	guy	Admin	Lecturer	Teaching Assistant	Edit Page
1	guy	Admin	Lecturer	Teaching Assistant	Delete Page
1	guy	Admin	Lecturer	Teaching Assistant	Delete Course
2	yaron	Guest	NULL	NULL	Refresh Page
3	eli	Student	Guest	NULL	Refresh Page
3	eli	Student	Guest	NULL	View Grade
3	eli	Student	Guest	NULL	View Assignment
3	eli	Student	Guest	NULL	View Course
4	dani	Grader	NULL	NULL	Edit Grade
5	oded	Lecturer	Teaching Assistant	NULL	Add Assignment
5	oded	Lecturer	Teaching Assistant	NULL	Edit Assignment
5	oded	Lecturer	Teaching Assistant	NULL	Delete Assignment
6	shahar	NULL	NULL	NULL	NULL

At this point, I knew I had it. The idea was to list every subject with his 3 inheritance levels, including those who have 2, 1 or even 0.
After getting to this query, it was easy to get to the final result.
In the resulting grid, we can see that every student has 1 more permission than he had in 2.4.1, since Students inherit that permission from Guests.
Moreover, the Admin, which has 3 levels of inheritance, now possesses all their permissions.
In addition, subject '6' still doesn't hold any Permission.

2.4.3 – The main challenge here was to still count the subjects themselves on the grid, just without any permissions. The way I addressed that was to implement the date condition in the JOIN statement instead of a WHERE statement (etc). As you can see, this condition should address all the date possibilities. All subjects without enabled roles have been successfully assigned with no permissions.

Visual Model (rbac.mwc)

This file contains a MySQL Workbench Model, which visualizes the database graphically.