

MySQL Relational Database

AdminRoles

Description:

Stores all roles with administrative permissions including roles to denote if a person does not have administrative permissions. Includes information like role id, name of the role, and permissions associated.

Columns:

id, name, permissions

Indexes:

id - PRIMARY

Test Results:

#	Time	Action	Message	Duration / Fetch
✓ 1	00:05:10	SELECT * FROM dc.AdminRoles LIMIT 0, 1000	9 row(s) returned	0.094 sec / 0.000 sec

Improvements

- Add a description column (set to NULL by default) that provides a brief description of the role, its functionality, and who it should be given to.
- Relate AdminRoles to users table so only users that are admins can pick certain roles
- Combine AdminRoles with the user_roles table (potentially deleting user_roles and creating a unified 'roles' table as there are no existing user_roles) for optimization.

Stored Procedures

- Delete_AdminRole
- Get_Admin_Role_Data
- Get_AdminRole
- Get_AdminRoles
- Get_Roles
- GetAdminRoleByID - refers to 'adminroles' table which should be deleted or merged
- Insert_AdminRole
- Update_Admin_Role
- Update_AdminRole_Name
- Update_AdminRole_Permissions

Notes:

- None of the items in the list can be null.
- id is set to auto-increment, is a balanced tree, and must only contain unique values

Admins

Description:

Stores a list of all admins including information like email, password, first and last name, role, last login, and status.

Columns:

email, password, firstName, lastName, role, last_login, status

Indexes:

email - PRIMARY

role - role_idx

Test Results:

#	Time	Action	Message	Duration / Fetch
✓ 1	00:09:43	SELECT * FROM dc.Admins LIMIT 0, 1000	12 row(s) returned	2.546 sec / 0.000 sec

Improvements:

- Move password to users for privacy purposes. Login should only be done through the users table anyhow.
- role should not be NULL as all admins should have a role
- role should be the name of the role, not the id number, for more description

Stored Procedures

- Delete_Admin
- Get_Admin
- Get_Admins
- Insert_Admin
- Update_Admin_Password
- user_expiration_data

Notes:

- Only last_login and roles may have null values.
- email is a balanced tree and must only contain unique values
- roles is a balanced tree
- status is an enum (binary, in this case) of values 'active' or 'inactive'

Email_Request

Description:

Stores a list of all email requests, storing the users demographic information.

Columns:

city, comments, country, email, firstName, lastName, id, keywords, lastName, mediaReports, organization, phone, professionallyEditedCheckbox, reportEnd, reportsCheckbox, reportStart, resolutionCheckbox, state, street, televisionFormatCheckbox, transcripts, tvStation, zip

Indexes:

id - PRIMARY

Test Results:

#	Time	Action	Message	Duration / Fetch
1	19:12:18	SELECT * FROM dc.Email_Request LIMIT 0, 1000	20 row(s) returned	0.094 sec / 0.000 sec

Improvements:

- firstName, lastName, and email are nullable but they should be non-nullable because of the importance of that information.

Stored Procedures

- GetEmailRequestById

Notes:

- What is this table accomplishing? What are email requests for?

Errors

Description:

Stores a list of all errors, recording host name, station, an error string, an error line, and severity.

Columns:

Date_Time, Host_Name, Station, Error_Str, LineNum, Severity

Indexes:

Date_Time - PRIMARY

Host_Name - PRIMARY

Station - PRIMARY

Severity - fk_Severity_idx

Test Results:

	#	Time	Action	Message	Duration / Fetch
✓	1	19:05:58	SELECT * FROM dc.Errors LIMIT 0, 1000	1000 row(s) returned	0.141 sec / 0.078 sec

Improvements:

- It is unclear what LineNum corresponds to.

Stored Procedures

- Admin_del_err
- Insert_Err

Notes:

- Severity is linked to the Severity schema which contains three entries: High, Medium, and Low.

User_Roles

Description:

Contains a list of all user roles.

Columns:

email, role

Indexes:

email - PRIMARY

Test Results:

#	Time	Action	Message	Duration / Fetch
1	00:33:49	SELECT * FROM dc.user_roles LIMIT 0, 1000	43 row(s) returned	0.078 sec / 0.000 sec

Improvements:

- There are no recorded roles for any of the users. Are there any roles that users can possess in the first place? If not, this column should be deleted.
 - Alternatively, merge adminRoles into this table as admins will have permissions unlike the users.

Notes:

- email is the key used to relate the user_roles to the users

Users

Description:

Contains a list of all users with access to the database. Records email, name, password, and verification status.

Columns:

blocked, email, firstName, isVerified, lastName, password, previouslyBlocked, previouslyVerified, salt

Indexes:

email - PRIMARY, email_UNIQUE

Test Results:

#	Time	Action	Message	Duration / Fetch
1	11:58:05	SELECT * FROM dc.users LIMIT 0, 1000	46 row(s) returned	0.235 sec / 0.000 sec

Improvements:

- Why is it useful to record if someone has been previously verified? Should this be folded into isVerified? The same goes for the blocked column.

Stored Procedures

- cleanup
- Update_Password - refers to 'users' table which should be deleted or merged
- user_expiration_data - refers to 'users' table which should be deleted or merged

Notes:

- salt is nullable
- Documentation can be found in the search engine
 - Front-end repository search engine