Northeastern University
College *of* Engineering

Data management and Database Design (INFO 6210)

P2 - Database Design and Initial ERD
Team - 20

Team Members

Ashwin Muthiah Murugappan (murugappan.a@husky.neu.edu)
Vijayalakshmi Siddappa (siddappa.v@husky.neu.edu)
Naga Mahesh Tiruveedhula (tiruveedhula.n@husky.neu.edu)

Database Topic

RBAC (Role-based access control) database design for a small enterprise

Database Purpose

The purpose of this database is to maintain data (such as user/employee details, roles/groups that they may belong to, SAML authentication/SSO for the users etc.) of various employees/users across the enterprise. This data shall be used for 2 purposes

- To enable authentication during login and for SSO (Single Sign-on) using SAML authentication.
- Restriction of resource access such that employees have access rights or permissions only to the information they need to do their jobs effectively and prevent them from accessing information that doesn't pertain to them.

This information will be used and managed only by the security (or RBAC) team within the enterprise.

Business problems addressed

- Controls access of sensitive and non-sensitive data amongst the employees of the enterprise. Consequently improves compliance to say federal, regulatory laws related to privacy and sensitive data access.
- Ensures easy and accurate onboarding of new employees by creation of accounts and access rights and reduces need for paperwork
- Ensures that such access rights and permissions provided to employees across various platforms remain up-to-date.
- Controls modification of access and permissions as per the change in role of an employee.
- Controls revocation of access rights when an employee leaves the company.
- Effective management on a granular level, by consolidating employee access rights to set of roles and user groups. And hence increases operational efficiency.

Business Rules

- Each user can be associated with one or more accounts (or services).
- Each account or service can be associated with
    - One or more users
    - One or more roles
    - One or more user_groups
    - One or more SAML authentication config attributes.
    - One or more permissions
- Each role
    - May have access rights to one or more accounts or services.
    - Can be associated with one or more users
- Each user_group will have also
    - May have access rights to one or more accounts or services.
    - Needs to contain one or more users.
- Each SAML_authentication_config
    - May contain certificates/attributes to authenticate one or more accounts (for example, a single SAML authentication xml can be generated for all Google services like gmail, google docs etc. for a given user.)
- Each permission can be associated with one account only.

Design Requirements

- Crow's foot notation is used to construct the ERD.
- Primary key in each table is specified by placing "PK" beside the fields.
- Dashed lines are used to represent Non-identifying relationships and solid lines to represent identifying relationships.
- Crow's foot notation is placed next to the table where the relationship line ends to specify the many side of the relationship.
- "One" was placed next to the tables where the relationship ends to specify the one side of the relationship.
- Primary key is chosen such that they uniquely identify or represent each entity and is always non-null.
- Foreign key within entities contain values that are in the referenced parent primary key and thus ensures referential integrity of the data.

Design Decisions:

Sl. No	Entity Name	Entity existence	Purpose of the entity	Relationship with other entities
1.	User	Primary	1. User entity is one of the key entities of this RBAC design. This entity maintains all the details about the employees or users within the company and to whom access to different services is to be assigned. 2. Each user is uniquely identified by user_id. Additionally the user entity also contains attributes such as email, contact, display name to further distinguish each user.	1. User entity is connected to entities such as Role, Accounts and User_group via associative/bridge entities User_role_mapping, User_accounts_mapping and User_user_group_mapping respectively. 2. User entity shares a non-identifying relationship with Role and it's a many-to-many relationship as well. This is because User ID is not need to uniquely identify row within Role entity. 3. User entity to Accounts entity is an identifying and many-to-many relationship. 4. User entity to User_group entity is an identifying and many-to-many relationship as well.
2.	Role	Primary	1. Role entity maintains data about different roles in the enterprise such as Administrator, Ops (operations), Manager, Intern etc. It will include details such as description about the role. RBAC design is widely implemented because of the "Role" concept. Providing and managing access	1. Role entity shares a non-identifying relationship with User and an identifying relationship with Accounts. This is because Role (and its attributes) can be uniquely identified using Role ID alone. With respect to Accounts entity, it will be needed for an

			rights is more efficient when user or employees are consolidated using roles. 2. This entity is uniquely identified using Role ID alone.	instance associated with the role. 2. Role shares a many-to-many relationship with User and Accounts entity.
3.	User_group	Primary	1. User_group entity maintains details about different user groups say for example AWS_security_group, AWS_admins_group, Google_cloud_group and so on. In addition to "roles", providing and managing access rights is more efficient when users or employees are consolidated using user_groups as well. 2. This is entity is identified by user_group ID and the users that form this group. Details such as user_group name, created by and on and related details are captured as well.	1. User_group entity shares an identifying relationship with User entity and Accounts entity. This is because User ID will be needed to identify which User_group it belongs to. On the accounts entity end, user_group ID will be needed to identify which account or service they are associated with. 2. User_group shares many-to-many relationship with accounts entity and users entity. This means, each user can be associated with multiple user_groups and vice-versa. The same applies to user_groups and accounts as well.
4.	Accounts	Primary	1. Accounts entity maintains data about various services or applications say for example AWS account, Tableau account, Gmail accounts and so on. Every enterprise especially B2B based ones, use multiple such services across various platforms, the popular ones being AWS, Google Azure and so on.	1. Accounts entity shares a identifying relationship with other entities like User, User_group, Role, and Saml authentication config. This is because each of these entities listed above are associated with an instance of the account and so the accounts entity is strongly dependant on them.

			2. This entity is identified by account ID and additionally User ID, Role ID, and Saml authentication config ID also help identify the accounts they are associated with.	2. Accounts entity shares many-to-many relationship with User, User_group, Role, and Saml authentication config.
5.	Permissions	Primary	1. Permissions entity maintains data for various permissions associated with an account. Permissions are need to have granular control over who and which roles or groups can access, modify, own data etc. 2. Permissions can be of type read, write, modify, execute, delete etc. Each permission will have a type indicating whether read, write etc. And type_status indicating the current status of that permission i.e. whether “ON” or “OFF”. 3. Such permissions are stored as separate instances. This entity is uniquely identified using Permission ID only.	1. Permissions entity shares a non-identifying relationship with accounts entity because permissions can be uniquely identified and can exist without accounts primary key. 2. Also, each account can be associated with multiple permissions and thus there is 1-to-many relationship between the two.
6.	SAML _authenticatio n_config	Primary	1. SAML is Security Assertion Markup Language which provides all the attributes necessary to authenticate a user. Here the identity provider is the enterprise and service provider depends on the services that the enterprise uses (for example AWS services from Amazon). SAML also enables SSO i.e. single sign-on thus not needing multiple login	1. SAML provides necessary attributes to the service to authenticate whether to log an user in after analyzing the attributes. And so this entity is shares a many-to-many relationship with Accounts entity via an associative entity called saml_authentication_config_ac counts_mapping.

			from users. 2. In our design we group all these attributes in this entity called SAML_authentication_config. Each set of attributes (a row in this table) is uniquely identified by saml_authentication_ config_id.	2. Saml_authentication_config entity shares an identifying relationship with Account entity since an instance of the service would be strongly dependant on SAML authentication associated with a user.
7.	User_Role_mapping	Associative	1. User_Role_mapping is an associative entity that maintains mapping details between Users and Roles entity. 2. User_role entity uses the parent's primary keys as foreign keys in this context since User and Role entities share a non-identifying relationship. 3. Additionally, it contains attributes to provide further details about when this association occurred and who created this mapping.	1. As this is a bridge entity, it is connected to both user and role entities. 2. User entity shares a non-identifying relationship with Role entity. Also it is a many-to-many relationship.
8.	User_Usergroup_mapping	Associative	1. User_Usergroup_mapping is an associative entity that maintains mapping details between a User and a User_group. As a bridge entity it will map users to user_groups and vice versa. 2. It includes details such as when the user to user_group mapping was created and who assigned this mapping.	1. User_Usergroup_mapping entity has a primary key composed of its parent's primary keys namely User and User_group. 2. This associative entity is created to normalize the many-to-many relationship that exists between User and User_group. That is, each user can belong to multiple user_groups and each

				user_group will contain multiple users.
9.	User_Accounts_mapping	Associative	1. User_Accounts_mapping is an associative entity that maintains mapping between users and accounts entities. 2. User_id from user and accounts_id from accounts is used to form a composite primary key uniquely identifying this entity. 3. It contains additional attributes that provide further details about the when the user first created an account with service and when the last login was.	1. As this is a bridge entity, it is related to both user and accounts entities. 2. User shares an identifying relationship with accounts entity and it is many to many as well. 3. After normalization the above mentioned relationship is represented using this bridge entity.
10.	Accounts_Role_mapping	Associative	1. Accounts_Role_mapping is an associative entity representing mapping between accounts and role entities. 2. Account_id from accounts and role_id from roles is used to form a composite primary key uniquely identifying each tuple in this table.	1. As this is a bridge entity, it is related to both accounts and role entities. 2. Accounts shares an identifying relationship with role entity and it is also a many to many relationship. 3. After normalization the above mentioned relationship is represented using this bridge entity.
11.	User_group_Accounts_mapping	Associative	1. User_group_Accounts_ mapping is an associative entity that maintains mapping details between User_group and Accounts. As a bridge entity it will map user_groups to multiple accounts and vice versa. Say for	1. User_group_Accounts_ mapping has a primary key composed of its parent's primary keys namely User_group and Accounts. 2. This associative entity is

			example, a user_group such as AWS_admin_group can be associated with multiple AWS accounts. 2. It includes attributes such as when the user_group was associated with an instance of the account.	created to normalize the many-to-many relationship that exists between Accounts and User_group. That is, each user_group can be associated with multiple accounts and each account/service (for example, think of it as an AWS service created for a team) can have multiple user-groups associated with it.
12.	SAML_authentication_config_Accounts_mapping	Associative	1. This is an associative entity that maintains mapping details between SAML_authentication_config and accounts entities. 2. Account_id from accounts and saml_authentication_config-id from SAML_authentication_config are used to form a composite primary key uniquely identifying a tuple in this table.	1. As this is a bridge entity, it is related to both accounts and SAML_authentication_config entities. 2. Accounts shares a identifying relationship with SAML_authentication_config and it is many to many relationship as well. After normalization this relationship is represented through this associative entity.