|  |  |
| --- | --- |
| A picture containing text  Description automatically generatedDiscretionary Access Control | |
| CSE5330 Project 2  Project Report |  |
| Nikhil Sujith | 1001768092Abhishek Shinde | 1001754842 |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | |  |
| Part 1 – EER Diagram | | | |
| Diagram  Description automatically generated  **Basic Assumptions**   1. The user account has userID which is the primary key to identity the user account. 2. The user role has roleName as the primary key to uniquely identify each role in the system 3. DbTable has tableName as the primary key to identify each unique table in the system. 4. Privilege is disjoint to 2 types. 5. User Account can only be assigned to 1 user role 6. Multiple user accounts can be assigned to a role. 7. If a role is given a privilege, then all users assigned to that role will have that privilege 8. An user account can own multiple tables 9. A table can only be owned by a single user account 10. An user role can be assigned multiple account privileges 11. An user role can have multiple relation privileges on multiple tables 12. A table can be granted multiple privileges on multiple roles 13. A relation privilege on a table can be granted to multiple roles.   **Additional Assumptions**   1. An user may own a table and that user will have all privileges on the table they own. 2. The owner of the table can then allow certain privileges on the table they own. 3. The owner of the table can then allow certain roles to access certain privileges on the table they own. 4. The owner of the table can revoke privileges that they might have assigned to a role on the owned table. 5. The owner of the table can grant privileges to roles on the owned table, if the role also has been allowed to perform those operations by the super-user. 6. Each role will be assigned certain privileges on them, and the role can only perform those operations as permitted by the privileges. 7. The system has been implemented to follow “Strict DAC” where there can only be a single level of access grants, where the owner can grant access on owned relation to another user / role but the other user / role will not be allowed any propagation of privileges. 8. A privilege can only be granted and revoked by the owner of the table.   **Changes made to EER**  Initial EER  page1image16997904   1. The mapping for user accounts and user role has been changed from 1:1 to N : 1. The reason for this change being, initially the assumption was that each role will have a single user. However, during the mapping stage we realized that 2. The participation for user accounts and relation privilege has been update to total participation on the relations privilege side. This is for the fact that relation privileges has to be given by an owner who is also represented as a user account in the database. 3. Some attributes have been adjusted to match the requirements and assumptions. 4. The participation for user the user role side of has relationship has been changes from total to partial. This was because, we found it more practical for the user role to be optionally assigned user accounts. This arises from a scenario where the system actor[person using the system] may decide to first only create role and later assign users as and when they see fit. | |  | |

### Part 2 – EER to Relational Mapping

Diagram

Description automatically generated

**Mapping of regular (strong) entity types**

In this step, we map the strong entities in our project, I.e. User Account, User Role, DBTable and Privilege , by creating a Relation with their simple attributes present in it, and select one of the attributes as a Primary Key.

**Mapping of Binary Relationships**

**I) Mapping of 1:N Relationship:**

**1:N Relationship for User Account and DBTable:**

We map this relationship by adding a Foreign Key in DBTable which references the Primary Key of User Accounts.

Here, the Foreign key is ownerID which references userID in User Accounts.

**1:N Relationship for User Role and User Account :**

We map this relationship by adding a Foreign Key in User Account which references the Primary Key of User Role.

Here, the Foreign Key is roleName which references roleName in User Role.

**1:N Relationship for User Account and Relation-Privilege :**

We map this relationship by adding a Foreign Key in Relation-Privilege which references the Primary Key of User Account.

Here, the Foreign Key is grantorID which references userID in User Role.

**1:N Relationship for User Role and Account Privilege :**

We map this relationship by adding a Foreign Key in Account -Privilege which references the Primary Key of User Role.

Here, the Foreign Key is roleName which references roleName in User Role.

**Mapping of Ternary Relationships**

For mapping a ternary relation where n>2, we create a new relationship relation to represent the ternary relation. For our EER Mapping, we have a Ternary Relation between User Role, DBTable and Relation Privilege. To represent this relation in a Relational Schema, we create a new relationship relation “Has Access” which contains as Foreign Key, the primary keys of all the participating entities.

**That is, Has Access has foreign key :**

* 1. tableName – which references the primary key of DBTable
  2. roleName – which references the primary key of User Role
  3. pid – which references the primary key of Relation Privilege

**Mapping of Specialization of Classes**

For mapping of class specializations, we create a new relation for the Super Class with its simple attributes and its primary key. For the subclasses, we also create a new relation where the subclasses have the primary of their parent class. For our EER Diagram, Privilege has a specialization where Relation Privilege and Account Privilege are its sub classes. We Create a relation Privilege with pid as the primary key, and we then create relation Account Privilege and Relation Privilege which have as foreign key, pid, which refers to the primary key of parent class.

### DDL

CREATE DATABASE SECURITY7;

USE SECURITY7;

CREATE TABLE user\_accounts(

userID INT NOT NULL auto\_increment,

phone varchar(12) NOT NULL DEFAULT 'NULL',

userName varchar(100) NOT NULL UNIQUE ,

# userRole varchar(100) NOT NULL DEFAULT 'staff',

CONSTRAINT UAPK PRIMARY KEY (userID)

)ENGINE=InnoDB;

CREATE TABLE DbTable(

tableName varchar(100) NOT NULL UNIQUE ,

ownerID int NOT NULL,

CONSTRAINT tpk PRIMARY KEY (tableName),

CONSTRAINT tfk FOREIGN KEY (ownerID) REFERENCES user\_accounts(userID)

ON DELETE CASCADE

ON UPDATE CASCADE

)ENGINE=InnoDB;

CREATE TABLE privileges(

pid int NOT NULL ,

privType varchar(100) NOT NULL,

CONSTRAINT ppk PRIMARY KEY (pid),

UNIQUE (privType)

)ENGINE=InnoDB;

CREATE TABLE user\_role(

roleName varchar(100) NOT NULL,

description varchar(100),

userID int NOT NULL DEFAULT 12,

CONSTRAINT urpk PRIMARY KEY (roleName, userID),

CONSTRAINT urfk FOREIGN KEY (userID) REFERENCES user\_accounts(userID)

ON DELETE CASCADE

ON UPDATE CASCADE

)ENGINE=InnoDB;

CREATE TABLE account\_privileges(

pid int NOT NULL ,

roleName varchar(100) NOT NULL ,

CONSTRAINT appk PRIMARY KEY (pid, roleName),

CONSTRAINT apfk1 FOREIGN KEY (pid) REFERENCES privileges(pid)

ON DELETE CASCADE

ON UPDATE CASCADE ,

CONSTRAINT apfk2 FOREIGN KEY (roleName) REFERENCES user\_role(roleName)

ON DELETE CASCADE

ON UPDATE CASCADE

)ENGINE=InnoDB;

CREATE TABLE relation\_privileges(

pid int NOT NULL ,

tableName varchar(100) NOT NULL ,

grantorID int NOT NULL ,

CONSTRAINT rppk PRIMARY KEY (pid,tableName),

CONSTRAINT rpfk1 FOREIGN KEY (pid) REFERENCES privileges (pid)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT rpfk2 FOREIGN KEY (grantorID) REFERENCES user\_accounts(userID)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT rpfk3 FOREIGN KEY (tableName) REFERENCES DbTable(tableName)

ON DELETE CASCADE

ON UPDATE CASCADE

)ENGINE=InnoDB;

CREATE TABLE has\_access(

pid int NOT NULL,

tableName varchar(100) NOT NULL ,

roleName varchar(100) NOT NULL ,

grantorID int NOT NULL ,

CONSTRAINT hspk PRIMARY KEY (pid, roleName, tableName),

CONSTRAINT hsfk1 FOREIGN KEY (pid) REFERENCES relation\_privileges (pid)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT hsfk2 FOREIGN KEY (tableName) REFERENCES relation\_privileges (tableName)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT hsfk3 FOREIGN KEY (grantorID) REFERENCES relation\_privileges (grantorID)

ON DELETE CASCADE

ON UPDATE CASCADE

)ENGINE=InnoDB;