IN 2400
Data Management Component in industry based project

Mantis - Multi-tenant Issue Tracking and Reporting System

The Court of Owls

Team members

Index Number	**Student Name**
184025L	M.D.N. Chandrasiri
184038E	E.A.Y.R. Edirisinghe
184116R	W.P.C.P. Pathirana
184152X	W.A.D. Sandarekha
184157R	I.K.G.D.S.N. Senanayake

Table of Contents

Introduction

Our project is to develop a system to bring all user roles(User/Manager/Developer/QA/Customer) to one platform to solve the issues/bugs regarding a selected product. The system is required to give solutions for those bugs related to a software that is offered by a company. The system involves five main components and each component may involve a specific schedule in the process of issue tracking. They are,

1. Authentication and Authorization - provide permissions to each user role
2. Bug capture log –Clients add issues to the company according to the given manner.
3. Bug management system - accessed by the internal users and manage sprints, backlog, add comments,etc.
4. Bug solution pool - stores solved bugs for future references.
5. Bug Reports - generate reports for monitoring project progress.

EER diagram scenario

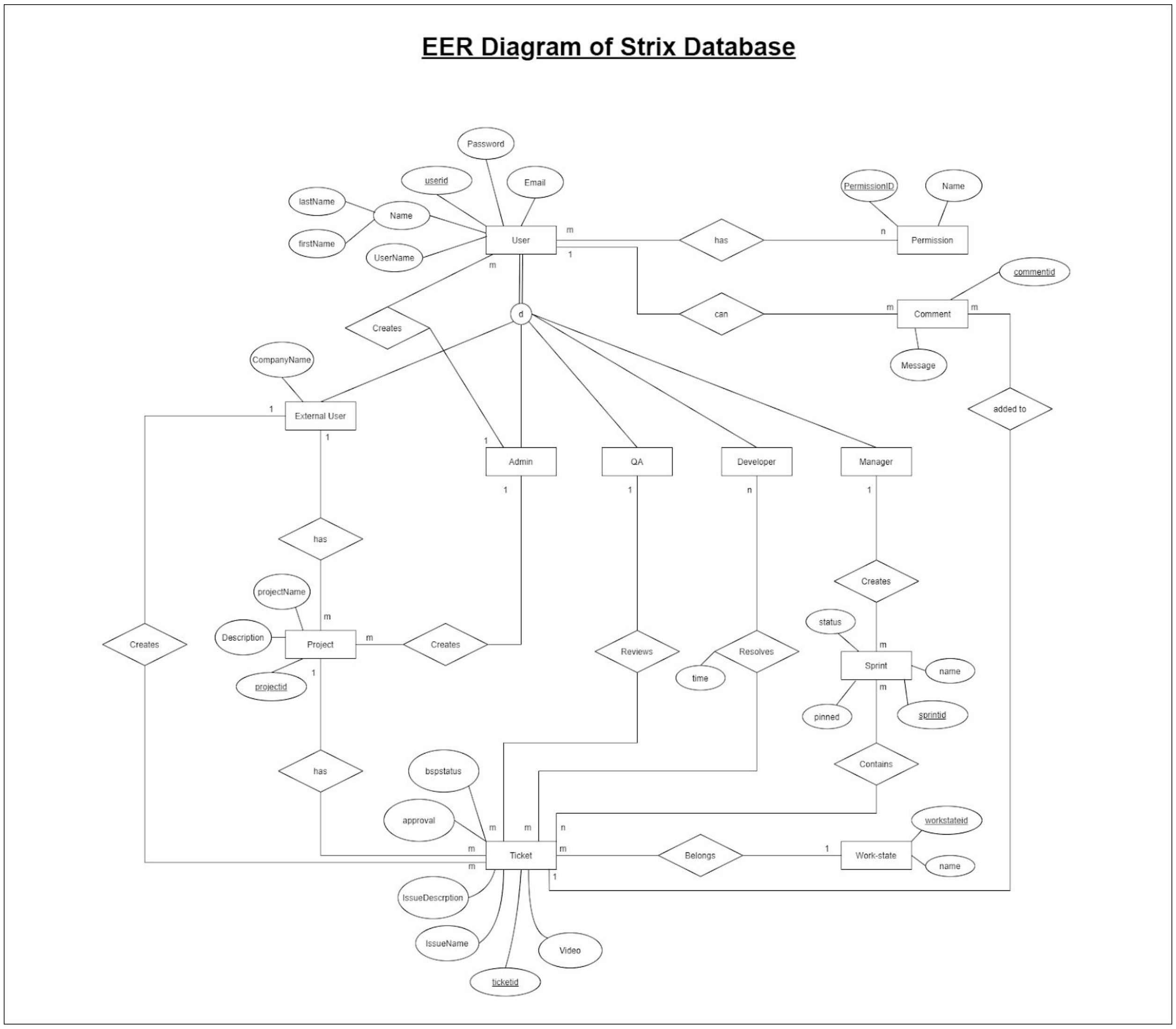
Following are the data stored in the database.

- System users can either be external or Developer, Admin, Manager or QA. For each user has a unique user ID, Name (First, Last), user name, email and a password.
- Also Admin creates the users and the projects.
- External users may have more than one project and he/she has a company name.
- The project name, their description and the unique project ID are stored in the database.
- External users create tickets.
- One project has several tickets that are reviewed and resolved by a Developer and a QA respectively. QA can review many tickets and one ticket is resolved by many developers. Reviewing time and resolving time is recorded for future reports. Tickets should have a ticket ID and keep the details about issue description, issue name and the video.
- Manager creates a sprint which contains tickets and several sprints have different work states.
- Sprint has a unique sprint ID and a name.
- Work state belongs to a ticket and the work state ID and the name are stored.
- Users have permission to access the system according to their user role.

- Users can add comments. The comments are added to a ticket.

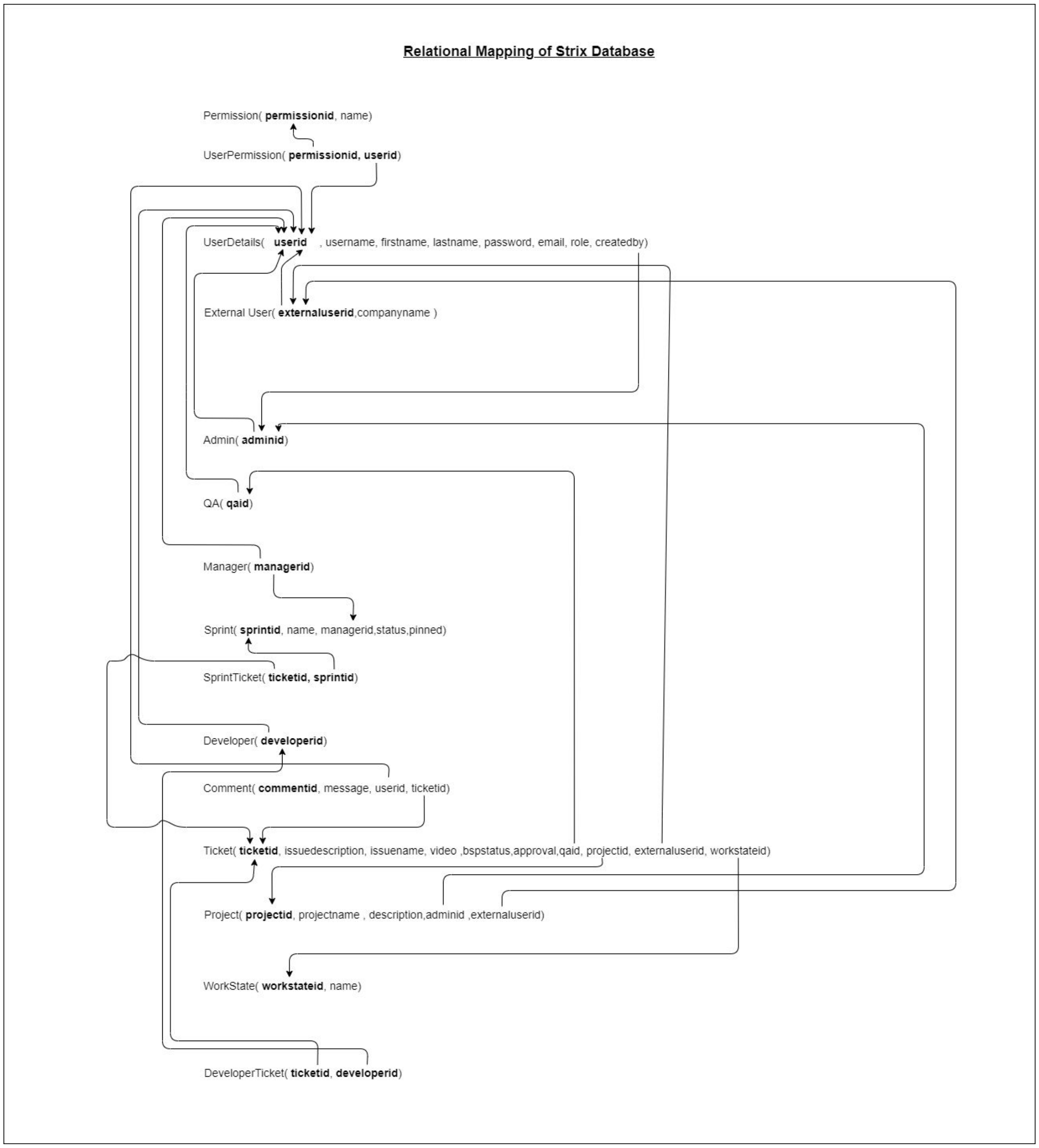
- Permission has a unique permission ID and a name and also the database keeps data about the comments (which message is delivered using a comment and their unique ID).

EER Diagram



Link : **EER Diagram (1).png**

Relational Schema



Relational Mapping of Strix Database

Permission(**permissionid**, name)

UserPermission(**permissionid, userid**)

UserDetails(**userid** , username, firstname, lastname, password, email, role, createdby)

External User(**externaluserid**,companyname)

Admin(**adminid**)

QA(**qaid**)

Manager(**managerid**)

Sprint(**sprintid**, name, managerid,status,pinned)

SprintTicket(**ticketid, sprintid**)

Developer(**developerid**)

Comment(**commentid**, message, userid, ticketid)

Ticket(**ticketid**, issuedescription, issuename, video ,bspstatus,approval,qaid, projectid, externaluserid, workstateid)

Project(**projectid**, projectname , description,adminid ,externaluserid)

WorkState(**workstateid**, name)

DeveloperTicket(**ticketid**, **developerid**)

Link : RelationalSchema.drawio

SQL Queries

```
select * from Admins

select * from UserDetails

select * from Permission

select * from UserPermission

select * from External_User

select * from QA

select * from Manager

select * from Developer

select * from DeveloperTicket

select * from Comment

select * from WorkState

select * from Sprint

select * from SprintTicket

select * from Project

select * from Ticket
```

```
-- Retrieving total number of Tickets per a project
select projectid,count(ticketid) as totalnumberoftickets
from Ticket
group by projectid

--Retrieving data where ticket is not approved yet
select *
from Ticket
where approval =0

--Getting project records where each external user has involved with
select p.projectid, ex.externaluserid, p.projectname, p.description , ex.companyname
from Project p , External_User ex
where p.externaluserid = ex.externaluserid


--Retrieving data where each manager's sprint with status 1
select ud.userid, ud.firstname , ud.email, s.sprintid , s.name, s.status
from UserDetails ud ,  sprint s
where  ud.userid= s.managerid and status=1
```

Stored Procedure

1. Count no of tickets handled by developers for given 2 dates

```
create proc CountDeveloper(
   @day1 date,@day2 date
)
as
   begin
      select L.developerid,count(L.ticketid) as noOfTicketsHandeled
      from
         (select *
         from DeveloperTicket
         where time between @day1 and @day2) L
      group by developerid
   end
```

2. Create permission

```
create proc CreatePermission(
   @permissionid varchar(4),
   @name varchar(20)
)
as
   begin
      if not exists(select * from Permission where permissionid=@permissionid)
      begin
         insert into Permission values(@permissionid,@name)
      end
  end
```

3. Create workstate

```
create proc CreateWorkState(
  @workstateid varchar(4),
  @workstatename varchar(20)
)
as
  begin
    if not exists(select * from WorkState where workstateid=@workstateid)
    begin
      insert into WorkState values(@workstateid,@workstatename)
    end
  end
```

4. Add Comment

```
create proc AddComment(
  @commentid varchar(5),
  @commentdescription varchar(100),
  @userid varchar(5),
  @ticketid varchar(5)
)
as
  begin
    if not  exists(select * from Comment where commentid=@commentid) and
          exists(select * from Ticket where ticketid=@ticketid) and
          exists(select * from UserDetails where userid=@userid)
    begin
      insert into Comment values(@commentid,@commentdescription,@userid,@ticketid)
    end
  end
```

5. Create project

```
create proc CreateProject(
   @projectid varchar(5),
   @projectname varchar(20),
   @description varchar(100),
   @adminid varchar(5),
   @externaluserid varchar(5)
)
as
   begin
      if not  exists(select * from Project where projectid=@projectid) and
            exists(select * from Admins where adminid=@adminid) and
            exists(select * from External_User where externaluserid=@externaluserid)
      begin
         insert into Project values(@projectid, @projectname, @description, @adminid,
@externaluserid)
      end
   end
```

6. Create ticket

```
create proc CreateTicket(
   @ticketid varchar(5),
   @issuesescription varchar(150),
   @issuename varchar(20),
   @video  varchar(50),
   @bspstatus bit,
   @approval bit,
   @projectid varchar(5),
   @externaluserid varchar(5),
   @qaid varchar(5),
   @workstateid varchar(5)
)
as
   begin
      if not  exists(select * from Ticket where ticketid=@ticketid) and
            exists(select * from Project where projectid=@projectid) and
            exists(select * from External_User where externaluserid=@externaluserid) and
            exists(select * from QA where qaid=@qaid) and
            exists(select * from WorkState where workstateid=@workstateid)
      begin
         insert into Ticket values(@ticketid, @issuesescription, @issuename, @video,
@bspstatus,@approval,@projectid,@externaluserid,@qaid,@workstateid)
      end
   end
```

7. Create sprint

```
create proc CreateSprint(
   @sprintid varchar(5),
   @name varchar(20),
   @status bit,
   @pinned bit,
   @managerid varchar(5)
)
as
   begin
      if not  exists(select * from Sprint where sprintid=@sprintid) and
            exists(select * from Manager where managerid=@managerid)
      begin
         insert into Sprint values(@sprintid, @name, @status, @pinned, @managerid)
      end
   end
```

8. Create QA

```
create proc CreateQA(
   @qaid varchar(5)
)
as
   begin
      if not exists(select * from QA where qaid=@qaid)
      begin
         insert into QA values(@qaid)
      end
   end
```

9. Create Developer

```
create proc CreateDeveloper(
   @developerid varchar(5)
)
as
   begin
      if not exists(select * from Developer where developerid=@developerid)
      begin
         insert into Developer values(@developerid)
      end
   end
```

10. Create Manager

```
create proc CreateManager(
   @managerid varchar(5)
)
as
   begin
      if not exists(select * from Manager where managerid=@managerid)
      begin
         insert into Manager values(@managerid)
      end
   end
```

11. Create ExternalUser

```
create proc CreateExternalUser(
   @externaluserid varchar(5)
)
as
   begin
      if not exists(select * from External_User where externaluserid=@externaluserid)
      begin
         insert into External_User values(@externaluserid)
      end
   end
```

12. Create Admin

```
create proc CreateAdmin(
   @adminid varchar(5)
)
as
   begin
      if not exists(select * from Admins where adminid=@adminid)
      begin
         insert into Admins values(@adminid)
      end
   end
```

```
--13.Create UserDetails

create proc CreateUserDetails(
   @userid varchar(5),
   @username varchar(50),
   @firstname varchar(50),
   @lastname varchar(50),
   @password varchar(10),
   @email nvarchar(50),
   @role varchar(20),
   @createdby varchar(5)
)
as
   begin
      if not exists(select * from External_User)
      begin
         if @role='Admin'
         begin
            insert into UserDetails
values(@userid,@username,@firstname,@lastname,@password,@email,@role,NULL)
         end
         else
         begin
            print 'You need to create a admin first'
         end
      end
      else
      begin
         if not exists(select * from Admins where adminid=@createdby)
         begin
```

```
            insert into UserDetails
values(@userid,@username,@firstname,@lastname,@password,@email,@role,@createdby)
          end
      end
  end
```

Views

1. View user role

```
CREATE VIEW UserRoleView
AS
  SELECT *
  FROM UserDetails
  WHERE role = 'Customer'
```

2. View ticket with it’s project and current workstate

```
CREATE VIEW ticketstatus
AS
  SELECT ticketid,issuename,projectid,workstatename
  FROM Ticket t,WorkState w
  WHERE t.workstateid=w.workstateid;
```

3. View projects with it’s customer

```
CREATE VIEW cutomerproject
AS
  SELECT projectname,companyname,firstname
  FROM External_User e,Project p,UserDetails u
```

```
   WHERE e.externaluserid= p.externaluserid and e.externaluserid=u.userid
```

4. View total number of tickets in a sprint

```
CREATE VIEW numberofticket

AS

   SELECT sprintid,count(ticketid) tickets

   FROM SprintTicket s group by sprintid
```

5. View active issues regarding a particular project

```
CREATE VIEW activebugs

AS

   SELECT ticketid,issuename,projectname,workstateid

   FROM Ticket t,Project p where t.projectid=p.projectid and not workstateid ='ws004'
```

Functions

1. Project count for given customer

```
create function func1(
   @externaluserid varchar(5)
)
returns int
as
   begin

      declare @projectcount int

      select @projectcount=count(projectid)
      from Project
      where externaluserid = @externaluserid
      group by externaluserid

      if not exists(select * from Project where externaluserid=@externaluserid)
      begin
         set @projectcount = -1
      end
      else if @projectcount = 0
      begin
         set @projectcount = 0
      end
      return @projectcount
   end
```

2. Search word from ticket description

```
create function func2(
   @keyword varchar(10)
)
returns table
as
   return
      select ticketid,issuesescription,issuename
      from Ticket
      where bspstatus = 0 and issuesescription like '%'+@keyword+'%'
```

3. Percentage value for finished ticket for given project

```
create function func3(
   @projectid varchar(5)
)
returns varchar(50)
as
   begin
      declare @totalcount int
      declare @partialcount int
      declare @presentvalue varchar(50)

      select @totalcount=count(projectid)
      from Ticket
      where projectid = @projectid

      select @partialcount=count(projectid)
      from Ticket
      where projectid = @projectid and workstateid='ws004'
```

```
        if @totalcount = 0
        begin
            set @presentvalue = 'This project has no any tickets'
        end
        else
        begin
            set @presentvalue = cast(@partialcount*100/@totalcount as varchar)+ '%'
        end
        return @presentvalue
    end
```

Triggers

1. Generalized relationship trigger

```
create trigger CreationTrigger
on UserDetails
after insert
as
    begin
        declare @role varchar(20)
        declare @userid varchar(5)

        select @role=role,@userid=userid
        from inserted

        if @role='Admin'
        begin
            exec CreateAdmin @userid
        end
        else if @role='QA'
        begin
            exec CreateQA @userid
        end
        else if @role='Customer'
        begin
            exec CreateExternalUser @userid
        end
        else if @role='Developer'
        begin
            exec CreateDeveloper @userid
        end
        else if @role='Manager'
        begin
            exec CreateManager @userid
        end
    end
```

2. Checking work state name

```
create trigger ChechWorkStateName
on WorkState
after insert,update
as
    begin
        declare @name varchar(10)
        declare @total int

        select @name=workstatename
        from inserted

        select @total = L.total
        from(
            select workstatename,count(workstatename) as total
            from WorkState
            group by workstatename) L
        where L.workstatename = @name

        if not @total=1
        begin
            print 'Can not have work state with same name'
            ROLLBACK TRANSACTION
        end
    end
```

3. Check finished tickets

```
create trigger ChechFinishedTickets
on Ticket
after insert,update
as
    begin
        declare @bspstatus bit
        declare @approval bit
        declare @workstateid varchar(5)

        select @bspstatus=bspstatus,@approval=approval,@workstateid=workstateid
        from inserted
        if @workstateid = 'ws004'
        begin
            if not((@bspstatus=0 and @approval=1) or (@bspstatus=1 and @approval=0))
            begin
                print 'Your project is already finished. Hence either bspstaus or approval should be 1'
                ROLLBACK TRANSACTION
            end
        end
    end
```

4. Password Change Restricts

```
create trigger PasswordChange
on UserDetails
after update
as
    begin
        if update(password)
        begin
            print 'You can not change password'
            ROLLBACK TRANSACTION
        end
    end
```

Indexes

```
set statistics io on

create clustered index idx_permissionid on Permission(permissionid)

create nonclustered index idx_proName on Project(projectname)

create nonclustered index idx_issue on Ticket(issuename)

create clustered index idx_SprintTicket on SprintTicket( ticketid, sprintid)

create nonclustered index idx_UserDetails on UserDetails(role)
```

Backup and Recovery Policy

The Multi-tenant Issue Tracking and Reporting System will be developed using PostgreSQL. In order to protect the system data from any system failures, cyber attacks or any server migrations.

PostgreSQL maintains a log called "Write Ahead Log (WAL)" which records every change made to the database's data files. In case of a system crash, the database can be restored by replaying the log entries made since the last checkpoint.

Any inconsistencies of the file system will be corrected by log replaying. Hence, we do not need a consistent file system backup. Moreover, we can stop the replay at any time and have a snapshot of the database. However, this method requires a lot of archival storage and this might incur some traffic in the system as well. Albeit the disadvantages, since the system requires high reliability, this method would be preferred than the other methods.