INFC20, Avancerad Databassystem - Dokumentation

LUNDS UNIVERSITET

Ekonomihögskolan

Grupp 10

John Hagelin	910511-1638
Erik Bååth	920415-1279
Calle Svensson Rundgren	920506-4539
Emil Wiberg	911107-4879

Lärare

Erdogan Ucan

Källförteckning

Bakgrund
Bakgrunden i projektet grundas i det systemvetenskapliga kandidatprogrammet, terminen fem. Momentet utgör en del av totalt tre för kursen och är ett praktikfall som ska omfatta ett program med koppling till en databas.

Mål
Vår grupps huvudmål är att skapa ett bra system som uppfyller den givna kravspecifikationen. Sedan har vi skapat egna mål för uppgiften som till exempel att alla ska vara delaktiga under utvecklingen och att vi ska vara stolta över vår slutprodukt.

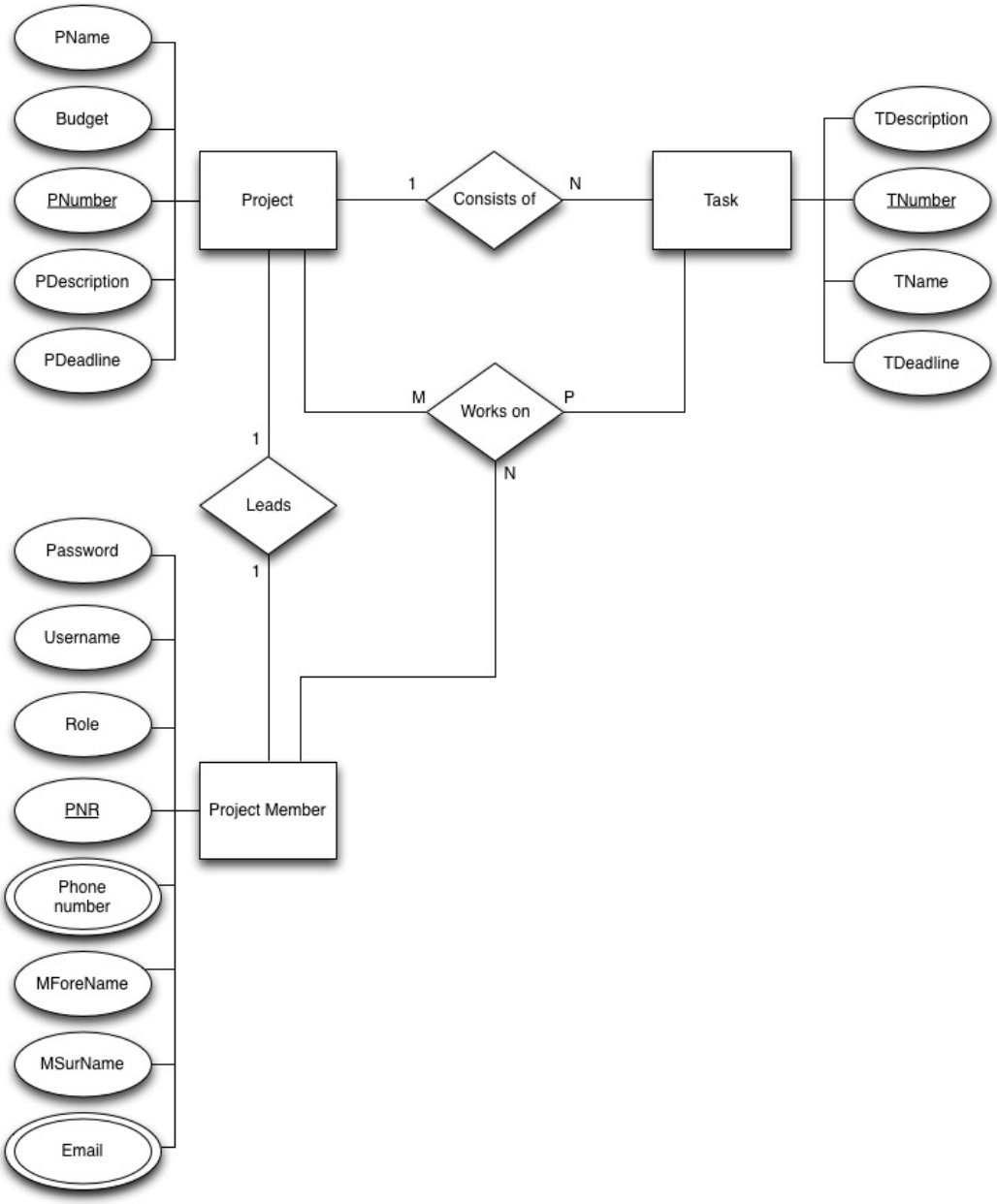
Syfte
Projektets syfte är att vi som projektmedlemmar skall erhålla ökad kunskap inom databas/server-utveckling och klient-utveckling.

Projektbeskrivning
Då gruppen fick i uppgift för kursen att utveckla ett valfritt system skrivet i valfritt programmeringsspråk, så valde vi som respons att utveckla ett projekthanteringssystem.
Med hänsyn till kriterierna som berör serversidan (felhantering, dataintegritet och konstistens) så har vi utformat databasen med inbyggda sk. stored procedures och triggers, som håller felhantering i form av try- och catchmetoder. Vidare valde gruppen att utveckla klienten i .NET-miljö, detta då det förenklar databas-integrering avsevärt jämfört med t.ex. utveckling i Oracle-miljö. Tänkt var att programmet skulle leva upp till uppgiftkriterier som; hålla en MVC-arkitektur enl. proper systemutveckling och bestå av gränssnitt, dialog, funktionalitet samt felhantering.

Systembeskrivning

Vi har valt att utveckla ett generellt projekthanterings-system. Systemets huvudanvändare är Projektledare och projektdeltagare, där respektive huvudanvändare har olika rättigheter i systemet. Projektledaren har rättigheter till att administrera projekten, användare, uppgifter och vem som jobbar med vad, medan projektdeltagaren har rättighet att se vilka projekt den har tilldelade, vilka uppgifter som finns i respektive projekt, samt vilka som tillhör varje projekt.
Beroende på om personen som loggar in är projektledare eller projektdeltagare visas olika interface för hur projektdata visualiseras.

Databas-modell



Kravspecifikation

- Systemet skall:
- Visa projekt för en viss projektmedlem
- Visa medlemmar för ett visst projekt
- Visa Task(s)
- Visa Avklarade Projekt för en viss projektmedlem
- Skapa/ändra/ta bort ett projekt
- Skapa/ändra/ta bort en task
- Skapa/ändra/ta bort projektmedlem
- MVC-mönster
- Naming Convention

Bugglista/Systembrister

- I de datagrid som laddar in data då man klickar på en row i en annan datagrid kan ibland bugga och inte alltid uppdateras direkt och man kan behöva klicka flera gånger på den row man vill åt.
- Vid login fungerar ibland inte krypteringen och man får ett fel, kan bero på att man försöker logga in för fort eller dyl.

Nästa version

Nedan följer en lista på funktioner som skulle kunna implementeras i nästa version:

- Visualisera tid kvar tills deadline
- Webbapplikation
- Smartphone-applikation

Moduler

- LINQ to SQL

Vi har använt oss av LINQ to SQL för att generera modellentiteter, stored procedures och triggers från databasen till vårat klientprojekt.

- SimpleCrypt

Vi har använt oss av SimpleCrypt för att enkelt kunna kryptera en användares lösenord med PBKDF2 som är en krypteringsmetod som saltar och hashar lösenordet med den unika salten som genereras till varje enskild användare.

- Transaction Scope

Vi har använt oss av transaction scope för att hantera transactions till databasen.

Mjukvara och hårdvara

Vi har arbetat i Windowsmiljö med utvecklingsverktyget Visual Studio 2012 för att skapa en desktop- applikation i programmeringsspråket C# och Microsoft SQL Management Studio för att skapa vår databas.

Responstider

På grund av att vi varken har speciellt komplicerade Stored Procedures eller stor testdata, så varierar responstiden mellan 1-3 millisekunder

Triggers

```
create trigger OnDeleteProjectTrigger
on Project
instead of delete
as
delete worksOn
where pno in
(select pno from deleted)
delete tasks
where pno in
(select pno from deleted)
delete project
where pno in
(select pno from deleted)
go
```

```
create trigger OnDeleteProjectMemberTrigger
on ProjectMember
instead of delete
as
delete PMemEmailList
where pnr in
(select pnr from deleted)
delete PMemPhoneList
where pnr in
(select pnr from deleted)
delete WorksOn
where pnr in
(select pnr from deleted)
delete ProjectMember
where pnr in
(select pnr from deleted)
go
```

Stored Procedures

```
create procedure updateWorksOn(
@Pnr int,
@Tno int,
@Pno int,
@tnoIn int,
@pnoIn int,
@startDate date)
as begin
	begin try
		update WorksOn
		set PNR = @Pnr, TNo = @tnoIn, PNo = @pnoIn, StartDate =
@startDate
		where PNR = @Pnr
		and TNo = @Tno
		and PNo = @Pno
	end try
	begin catch
		execute usp_GetErrorMessage
	end catch
end

create procedure updateTask(
@TNo int,
@PNo int,
@TName varchar(100),
@TDesc varchar(100),
@TDeadline date
)
as
begin
	begin try
		update Tasks
		set PNo = @PNo, TName = @TName, TDesc = @TDesc, TDeadline =
@TDeadline
		where TNo = @TNo
	end try
	begin catch
		execute usp_GetErrorMessage
	end catch
end
```

```
create procedure updateProjectMember(
@Pnr int,
@Forename varchar(100),
@Surname varchar(100),
@MUsername varchar(100),
@MPassword varchar(100),
@MRole varchar(100)
)
as
begin
	begin try
		update ProjectMember
		set Forename = @Forename, Surname = @Surname,
MUsername = @MUsername, MPassword = @MPassword, MRole =
@MRole
		where PNR = @Pnr
	end try
	begin catch
		execute usp_GetErrorMessage
	end catch
end

create procedure updateProject(
@Pno int,
@PName varchar(100),
@PBudget int,
@PDesc varchar(100),
@PDeadline date,
@ProjectLeaderPnr int
)
as
begin
	begin try
		update Project
		set PName = @PName, PBudget = @PBudget, PDesc =
@PDesc, PDeadline = @PDeadline, ProjectLeaderPNR =
@ProjectLeaderPnr
		where PNo = @Pno
	end try
	begin catch
		execute usp_GetErrorMessage
	end catch
end
```

```
create procedure updatePMemPhoneList(
@OldPnr int,
@OldPhoneNr int,
@Pnr int,
@PhoneNr int
)
as
begin
	begin try
		update PMemPhoneList
		set PNR = @Pnr, PhoneNr = @PhoneNr
		where PNR = @OldPnr
		and PhoneNr = @OldPhoneNr
	end try
	begin catch
		execute usp_GetErrorMessage
	end catch
end

create procedure updatePMemEmailList(
@OldPnr int,
@OldEmail varchar(100),
@Pnr int,
@Email varchar(100)
)
as
begin
	begin try
		update PMemEmailList
		set PNR = @Pnr, Email = @Email
		where PNR = @OldPnr
		and Email = @OldEmail
	end try
	begin catch
		execute usp_GetErrorMessage
	end catch
end
```

```
create procedure setWorksOn(
@Pnr int,
@TNo int,
@PNo int,
@StartDate date
)
as
begin
	begin try
		insert into WorksOn values(@Pnr, @TNo, @PNo, @StartDate)
	end try
	begin catch
		execute usp_GetErrorMessage
	end catch
end

create procedure setTask(
@TNo int,
@PNo int,
@TName varchar(100),
@TDesc varchar(100),
@TDeadline date
)
as
begin
	begin try
		insert into Tasks values(@TNo, @PNo, @TName, @TDesc,
@TDeadline)
	end try
	begin catch
		execute usp_GetErrorMessage
	end catch
end
```

```
create procedure setProjectMember(
@Pnr int,
@Forename varchar(100),
@Surname varchar(100),
@MUsername varchar(100),
@MPassword varchar(100),
@MRole varchar(100),
@MPasswordSalt varchar(100),
@UserType varchar(100)
)
as
begin
	begin try
		insert into ProjectMember values(@Pnr, @Forename, @Surname,
@MUsername, @MPassword, @MRole, @MPasswordSalt, @UserType)
	end try
	begin catch
		execute usp_GetErrorMessage
	end catch
end

create procedure setProject(
@Pno int,
@PName varchar(100),
@PBudget int,
@PDesc varchar(100),
@PDeadline date,
@ProjectLeader int
)
as
begin
	begin try
		insert into Project values(@Pno, @PName, @PBudget, @PDesc,
@PDeadline, @ProjectLeader)
	end try
	begin catch
		execute usp_GetErrorMessage
	end catch
end
```

```
create procedure setPMemPhoneList(
@Pnr int,
@PhoneNr int
)
as
begin
	begin try
		insert into PMemPhoneList values(@Pnr, @PhoneNr)
	end try
	begin catch
		execute usp_GetErrorMessage
	end catch
end

create procedure setPMemEmailList(
@Pnr int,
@Email varchar(100)
)
as
begin
	begin try
		insert into PMemEmailList values(@Pnr, @Email)
	end try
	begin catch
		execute usp_GetErrorMessage
	end catch
end

create procedure getTaskByProjectUser(
  @usn varchar(100),
  @pno int)


as begin
  begin try
    select * from Tasks where TNo in (select Tno from WorksOn where PNo
in
                                (select Pno from ProjectMember where
MUsername = @usn))
                                and PNo = @pno
  end try
  begin catch
    execute usp_GetErrorMessage
  end catch
end
```

```
create procedure getTask
as
begin
	begin try
		select * from Tasks
	end try
	begin catch
		execute usp_GetErrorMessage
	end catch
end

create procedure getProjectsForCurrentUser(
	@userName varchar (100))
as
begin
  begin try
    select * from project where PNo in(select pno from WorksOn where pnr
in(select pnr from ProjectMember where MUsername = @userName))
    end try
    begin catch
      execute usp_geterrormessage
    end catch
end

create procedure getProjectMemebersForCurrentProject(
	@Pno int)
as
begin
  begin try
      select * from  ProjectMember where PNR in(select pnr from WorksOn
where pno = @Pno )
  end try
  begin catch
    execute usp_geterrormessage
  end catch
end

create procedure getProjectMember(
@PNR int)
as
begin
	begin try
		select * from ProjectMember
		where PNR = @PNR
	end try
	begin catch
		execute usp_GetErrorMessage
	end catch
end
```

```
create procedure getProject
as
begin
	begin try
		select * from Project
	end try
	begin catch
		execute usp_GetErrorMessage
	end catch
end

create procedure getPMemPhoneList(
@PNR int,
@PhoneNr int)
as
begin
	begin try
		select * from PMemPhoneList
		where PNR = @PNR
		and PhoneNr = @PhoneNr
	end try
	begin catch
		execute usp_GetErrorMessage
	end catch
end

create procedure getCurrentUser(
    @username varchar(max))
as
begin
	begin try
		select * from ProjectMember where MUsername = @username
	end try
	begin catch
		execute usp_GetErrorMessage
	end catch
end

create procedure getAllTask
as
begin
	begin try
		select * from Tasks
	end try
	begin catch
		execute usp_GetErrorMessage
	end catch
end
```

```
create procedure getAllProject
as
begin
  begin try
    select * from Project
  end try
  begin catch
    execute usp_geterrormessage
  end catch
end

create procedure getAllPmem
as
begin
  begin try
    select * from ProjectMember
  end try
  begin catch
    execute usp_geterrormessage
  end catch
end

create procedure deleteWorksOn(
@PNR int,
@TNo int,
@PNo int)
as
begin
        begin try
                delete WorksOn
                where PNR = @PNR
                and TNo = @TNo
                and PNo = @PNo
        end try
        begin catch
                execute usp_GetErrorMessage
        end catch
end
```

```
create procedure deleteTask(
@TNo int)
as
begin
	begin try
		delete Tasks
		where TNo = @TNo
	end try
	begin catch
		execute usp_GetErrorMessage
	end catch
end

create procedure deleteProjectMember(
@PNR int)
as
begin
	begin try
		delete ProjectMember
		where PNR = @PNR
	end try
	begin catch
		execute usp_GetErrorMessage
	end catch
end

create procedure deleteProject(
@PNo int)
as
begin
	begin try
		delete Project
		where PNo = @PNo
	end try
	begin catch
		execute usp_GetErrorMessage
	end catch
end
```

```
create procedure deletePMemPhoneList(
@PNR int,
@PhoneNr int)
as
begin
	begin try
		delete PMemPhoneList
		where PNR = @PNR
		and PhoneNr = @PhoneNr
	end try
	begin catch
		execute usp_GetErrorMessage
	end catch
end

create procedure deletePMemEmailList(
@PNR int,
@Email varchar)
as
begin
	begin try
		delete PMemEmailList
		where PNR = @PNR
		and Email = @Email
	end try
	begin catch
		execute usp_GetErrorMessage
	end catch
end

create procedure getHighestPno
as
begin
		begin try
			select max(Pno) from project
		end try
		begin catch
			execute usp_errorgetmessage
		end catch
end
```

```
create procedure checkUserPassword(
@username varchar(100),
@password varchar(100))
as
	begin
		begin try
			if exists(select pnr from ProjectMember where MUsername
= @username and MPassword = @password)
				return 1
			else
			return 0
		end try
		begin catch
			execute usp_getErrorMessage
		end catch
end

create procedure usp_GetErrorMessage
as
begin
select
	ERROR_NUMBER() AS ErrorNumber
  ,ERROR_SEVERITY() AS ErrorSeverity
  ,ERROR_STATE() AS ErrorState
  ,ERROR_PROCEDURE() AS ErrorProcedure
  ,ERROR_LINE() AS ErrorLine
  ,ERROR_MESSAGE() AS ErrorMessage
end
```