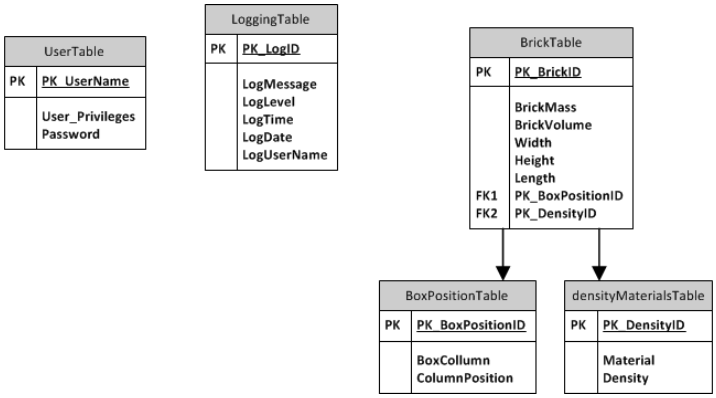
0.1 Data model

En kritisk del af dette system er data storage og data retrieval. Dette er blevet implementeret i form af to relationelle databaser; en distribueret og en lokal.
Til den distribuerede database og til administrationshjemmesiden er et domænenavn blevet købt hos n distribuerede database er lavet som en MySQL databas

0.1.1 Design af database

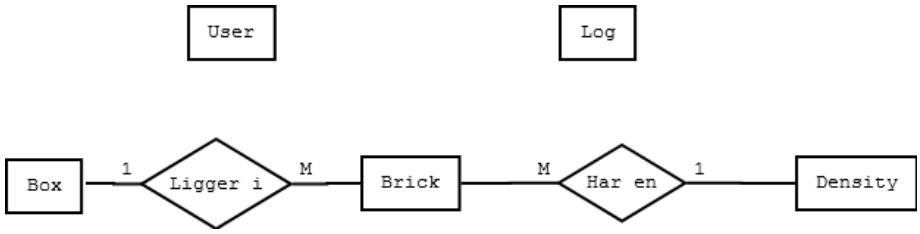
Hoved komponenten i datalagringen i dette projekt består i, at gemme data om en klods, og linke den til en position i en boks samt en densitet/materiale. Hertil består persisteringen også af brugernavne og loggingindlæg. På figur 1 ses et UML OO diagram over databasen



Figur 1: UML OO diagram over databasen

Som man kan se, har et indlæg i BrickTable (Hvor klodsernes beskrivelse ligger) en foreign key til BoxPositionTable (Hvor klodserne ligger) samt DensityMaterialTable(Hvor klodsernes densitet er beskrevet). UserTable(Hvor information om brugere af systemet ligger) og LoggingTable(Hvor alle system events gemmes) havde en relation i de første sprints, hvor LoggingTable havde en foreign key til UserTable, men dette blev kasseret da vi indså, at man i så fald ikke kunne fjerne en bruger fra systemet uden at slette alle brugerens logging indlæg. På figur 2 ses et ER diagram over databasen

Vi har udelukkende en-til-mange forhold i systemet, da flere klodser kan ligge i en kasse, men en klods kan ikke ligge i flere kasser. En Klods har desuden kun 1 densitet, men flere klodser kan godt have samme densitet.
Databasen er på anden normalform da vi ikke har nogen composite keys. Der eksisterer



Figur 2: ER Diagram over databasen

dog transitive afhængigheder så den ikke er på tredje normalform. I BrickTable er Volume transitivt afhængig af Height, Length og Width. Det er dog blevet valgt, at databasen kun skal være på anden normalform, og databasen overholder derfor kravende.

0.1.2 Triggers og stored procedures

Til implementeringen blev der lavet flere triggers og en funktion.

Triggers:

Flere tabeller er blevet lavet ved hjælp af en identity column som primary key. Dette skaber visse problemer i tabellen, navnligt når man prøver at slette en række fra databasen. En delete statement vil lave et hul i primary key kolonnen som ikke er synderligt kønt, og det er ikke muligt at lave en update statement på kolonnen, da det ikke kan gøres på identity kolonne. Der må insertes hvis man midlertidigt slår identity insert til, så ved hjælp af en on delete trigger blev det gjort muligt at lave en update on delete trigger. På kodeudsnit 1 set et kodeudsnit for en update on delete trigger på LoggingTable

Kodeudsnit 1: SQL trigger for update af en identity kolonne

```
1 Create trigger [dbo].[LoggingTrigger]
2 ON [dbo].[LoggingTable]
3 FOR Delete
4 as
5
6 Declare @DeletedID bigint
7 Declare @TempTable Table (ID bigint, msg varchar(500), lvl nchar↩
    (50), ltime nchar(50), ldate nchar(50), luser nchar(50))
8 Declare @Reseed int
9 Set @DeletedID = (Select MIN(PK_LogID) From deleted)
```

```
10
11 Insert into @TempTable select * from LoggingTable where PK_LogID ↩
     > @DeletedID
12 update @TempTable set ID = ID - 1
13 delete from LoggingTable where PK_LogID > @DeletedID
14
15 SET IDENTITY_INSERT LoggingTable ON
16 insert into LoggingTable (PK_LogID,LogMessage,LogLevel,LogTime,↩
     LogDate,LogUserName) select * from @TempTable
17
18 SET IDENTITY_INSERT LoggingTable OFF
19 set @Reseed = (Select MAX(PK_LogID) from LoggingTable)
20 if @Reseed >= 0
21   dbcc checkident(LoggingTable, reseed, @Reseed)
22 else
23   dbcc checkident(LoggingTable, reseed, 0)
24 GO
```

Det første der sker efter erklæringen af variable er at DeletedID bliver sat til den mindste PKLogID af deleted. Deleted er en tabel der indeholder det slettede data. Herefter bliver alle rækker der har et større PKLogID end DeleteID, dvs alle rækker efter de slettede, sat ind i en midlertidig tabel. Så inkrementeres PKLogID i den midlertidige tabel med en, hvorefter alle rækker med større indlæg end de slettede, slettes fra tabellen. Så enables indsætning på en identity kolonne og den midlertidige tabel indsættes i LoggingTable. Identity reseedes til sidst til den største værdi af PKLogID i tabellen. Hvis tabellen er tom reseedes der til nul. Med dette er der blevet opnået en update statement på en identity kolonne. Dette sker ved samtlige tabeller der har en identity kolonne.
Triggeren for BrickTable har dog et par tilføjelser. Når man sletter en klods så er det ikke nok bare at slette klodsen og opdatere primary key'en. Den række i BoxPositionTable som BrickTable refererer til skal også slettes og opdateres. Dette gøres ved at gemme foreign keyen over til BoxPositionTable fra det slettede, indsætte et dummy indlæg i BoxPositionTable og derefter slette den række som den slettede BrickTable række refererede til. Herefter slettes den række der har en primary key der stemmer overens med den slettede foreign key fra BrickTable, hvilket vil køre BoxPositionTable update on delete trigger. BrickTables update on delete fortsætter, og til sidst trækkes der en fra BoxPositionTables foreign key i BrickTable og dummy rækken i BoxPositionTable slettes.

Functions:

Den måde databasentilgangen er lavet på gør, at generel tilgang kan laves med funktioner på C# siden, men der er et enkelt sted hvor det er nødvendigt at bruge en lang join funktion. Denne funktion er specifik for denne join, dvs. tabeller og kolonner er hardcodet ind i join funktionen på sql-siden. På kodeudsnit 2 ses koden bag funktionen. Den skal bruges fordi data om klodsen (ie. Densitet og placering i boksen) gemmes i andre tabeller end klodstabellen.

Kodeudsnit 2: SQL funktion for inner join for brick data

```
1 create function [dbo].[GetTotalBrickData]()
2 returns @ReturnTable table
3 (
4 BoxColumn bigint, ColumnPosition bigint, Length float,
5 Height float, Width float, BrickVolume float,
6 BrickMass float, Material nchar(50), Density float
7 )
8 as
9 begin
10 insert @ReturnTable
11 SELECT
12 BoxPositionTable.BoxCollumn, BoxPositionTable.ColumnPosition,
13 BrickTable.Length, BrickTable.Height, BrickTable.Width,
14 BrickTable.BrickVolume, BrickTable.BrickMass,
15 DensityMaterialsTable.Material, DensityMaterialsTable.Density
16
17 FROM BoxPositionTable
18 INNER JOIN BrickTable ON
19 BoxPositionTable.PK_BoxPositionID = BrickTable.FK_BoxPositionID
20 INNER JOIN DensityMaterialsTable ON
21 BrickTable.FK_DensityID = DensityMaterialsTable.PK_DensityID
22
23 Return
24 end
25 GO
```

Denne funktion skal bruges ved visning af data om en klods på systemets GUI. Der skal altså bruges data fra flere tabeller end en, og frem for at lave flere select statements i træk, blev det valgt at lave en specifik funktion til at håndtere denne datahentning.