Kandidatuppsats

-

Databaser
Webbutveckling

Databaser
Datateknik

Albin Rönnkvist

Mittuniversitetet
MID SWEDEN UNIVERSITY

Campus Härnösand Universitetsbacken 1, SE-871 88. **Campus Sundsvall** Holmgatan 10, SE-851 70 Sundsvall.
Campus Östersund Kunskapens väg 8, SE-831 25 Östersund.
Phone: +46 (0)771 97 50 00, Fax: +46 (0)771 97 50 01.

Sammanfattning

I detta projekt ska jag sätta samman, designa, bygga, implementera och dokumentera ett fungerande databassystem med hjälp av en databashanterare. Databassystemet ska stödja en delmängd av verksamheten i en verklig organisation, t.ex. en avdelning inom ett stort företag. Tanken är att databasen jag skapar skulle kunna användas av någon. Den färdiga databasen kommer bestå av minst 6 tabeller.

Innehållsförteckning

Terminologi

Akronymer/Förkortningar

PK	Primary Key
FK	Foreign Key

1 Introduktion

Jag ska skapa ett databassystem som ska stödja en delmängd av en förening. Jag har valt att skapa en databas som beskriver en förenings lag med dess spelare och ledare.

Projektet kommer delas in i fyra faser. Dessa faser är: Konceptuell datamodellering(ER/EER), logisk databasdesign(Bas-relationer), fysisk databasdesign och implementering av databasen.

1.1 Bakgrund och problemmotivering

Att använda sig av databasteknik har många fördelar om man jämför med alternativen. Det är en teknik som underlättar hanteringen av data inom företag och organisationer. Databashanterare är relativt enkla att hantera med ett textbaserat gränssnitt. Det är även simpelt att göra komplicerade saker vilket gör systemet kraftfullt. En till fördel är att det är lätt att göra ändringar vilket gör systemet flexibelt. Det finns väldigt många fler fördelar med databaser och databashanterare men i sin helhet så är det ett system som är kraftfullt, flexibelt och enkelt att hantera.

1.2 Högnivåproblemformulering

Syftet med detta projekt är att kunna skapa ett databassystem som senare kan användas av ett riktigt företag eller organisation. Jag som databasutvecklare ska kunna använda mig av de kunskaper jag fått i denna kurs för att sätta samman, designa, bygga, implementera och dokumentera ett fungerande databassystem.

1.3 Avgränsningar

Denna rapport kommer endast gå in på databasteknik.

1.4 Detaljerad problemformulering

Först måste jag skapa en konceptuell datamodellering. Det gör jag genom att först beskriva föreningen som ska använda min databas och vad de vill lagra i databasen. Vilken specifik information de vill lagra om olika saker som finns i föreningen och hur de sakerna hör ihop. Det kan vara t.ex. information om personer och lag samt hur de hör ihop. När jag skrivit klart beskrivningen så ska jag göra ett ER-diagram utav den. I diagrammet definieras entiteterna med dess attribut och samband.

När den konceptuella datamodelleringen är färdig så kan jag börja skapa en logisk databasdesign. Först måste jag översätta mitt ER-diagram till bas-relationer enligt reglerna för relationsdatabaser. Detta gör jag med hjälp av ett annat dia-

gram. Här identifierar jag även primär- och främmande nycklar för varje relation.

När det är klart så kan jag börja med den fysiska databasdesignen. Jag börjar med att skapa regler eller SK. integritetsvillkor för alla kolumner i databastabellerna. På så sätt begränsar jag vilka data som kan lagras i databasen. Efter det måste jag beskriva referensintegritetvillkoren. Det innebär att om jag lägger in data i en tabell som är beroende av en annan tabell så måste den datan även finnas i andra tabellen. T.ex. om en person i tabellen **Person** bor på en adress med nummer **19** så måste även en adress med nummer **19** finnas i tabellen **Address**. Om en person bor på en adress så måste även adressen finnas. Efter det ska jag identifiera och beskriva eventuella index som behövs för icke-nyckelattribut. Sedan måste jag även fundera över om jag vill de-normalisera några relationer eller inte.

När alla förberedelser är klara så kan jag börja implementera databasen i en databashanterare. Då börjar jag med att skapa tabellerna. Sedan definierar jag datatyper, primärnycklar, index, integritetsvillkor och referensintegritetsvillkor. När allting är sammanställt så ska jag först mata in testdata för att testa databasens integritet och referensintegritetsbegränsningar(de jag har bestämt). Testdatan ska innehålla felaktig data som strider mot integriteten och begränsningarna. Misslyckas lagringen så fungerar databasen korrekt och om lagringen lyckas så måste jag ändra något. På så sätt kan jag kontrollera att mina begränsningar fungerar. Efter det lägger jag till exempeldata till mina tabeller som inte strider mot mina begränsningar. På så sätt kan jag visa att databasen fungerar.

När alla kontroller är klara så ska jag designa och exekvera ett antal SQL-frågor för att lista ut data från min databas.

2 Teori

En **databas** är en organiserad samling av information som är strukturerad på ett sätt där det ska vara så lätt som möjligt att söka och hämta specifik information samt kunna ändra den. I detta projekt kommer jag jobba med en **relationsdatabas** vilket är en databas som är organiserad i relationer. Relationerna kallas även tabeller och de består av kolumner(fält) och rader(tupler). En tabell beskriver ofta en specifik typ av information, t.ex. personer. Kolumnerna kan man se som olika värden som beskriver tabellen mer specifikt. T.ex. så kanske man vill veta personens namn, personnummer och ort. De värdena beskrivs i kolumnerna. Det är i raderna som sedan data om varje enskild person lagras där man utgår från kolumnernas värden. All data i en specifik kolumn måste vara av samma typ. I en rad kan det då ha lagrats: Albin, 9712257864, Stockholm. Alltså namn, personnummer och ort. Lägger jag till en ny person så lagras den på en ny rad. Det går att skapa flera tabeller och koppla ihop dessa med relationer. För att hantera databasen använder man sig ofta av språket SQL.[2]

Structured query language(SQL) är ett standardiserat programspråk för att lagra, hämta och ändra data i en relationsdatabas. Man använder alltså språket för att kommunicera med en databas. Några vanliga databasmotorer som stödjer SQL är: Oracle, Microsoft SQL Server, Microsoft Access, MariaDB, DB2 etc. [1]

3 Metod

Konceptuell datamodellering
Jag kommer skriva ner beskrivningen av föreningen och det de vill lagra i ett textdokument. När beskrivningen är klar så kommer jag skapa ett ER-diagram på en hemsida som heter ”draw.io”.

Logisk databasdesign
Relationerna kommer jag visa i ett diagram där även primärnycklar och främmandenycklar kommer beskrivas tydligt för varje relation.

Fysisk databasdesign
Denna design kommer beskrivas i ett nytt diagram.

Databas implementering
Jag kommer utgå från ovanstående delar och skapa kod i MySQL Workbench som stämmer överens med det jag bestämt tidigare. Testdatan, exempeldatan och SQL-frågorna kommer även skapas i MySQL Workbench.

4 Konstruktion

Konceptuell datamodellering

Beskrivning:
Jag börjar med beskrivningen av föreningen och vad de vill lagra. Beskrivningen lyder:

Förening är en fotbollsförening och de vill lagra sina lag, spelare och ledare.

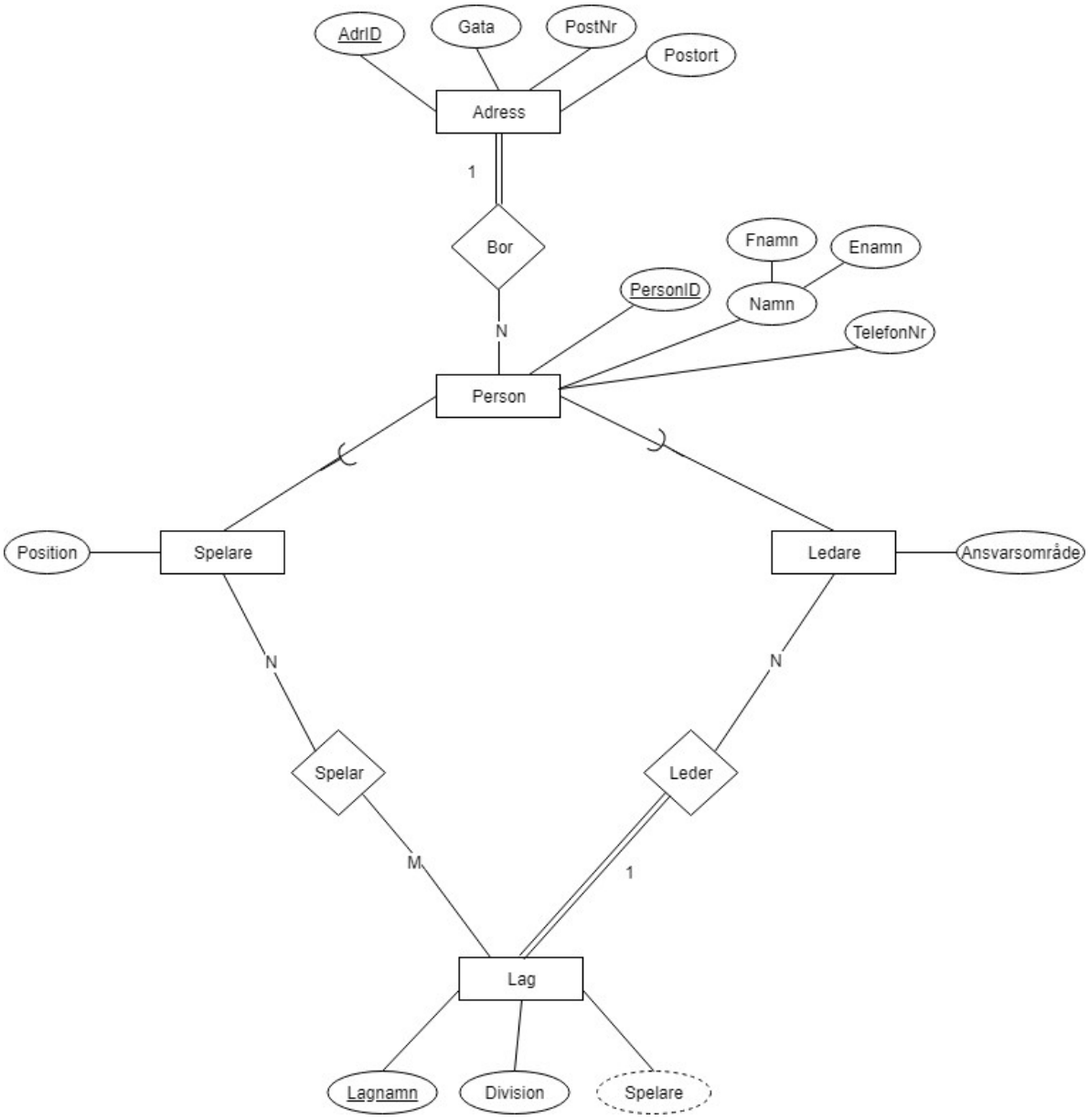
Varje lag har ett unikt lagnamn, en serie/division de spelar i och ett antal spelare. Lagen har även en eller fler ledare.

En spelare kan spela i flera lag samtidigt och varje spelare har ett unikt id, förnamn, efternamn, adress, telefonnummer samt en position.

En ledare måste leda ett lag och den kan endast leda ett lag åt gången. En ledare kan ha olika ansvarsområden. Varje ledare har likt spelarna ett unikt id, förnamn, efternamn, adress och telefonnummer.

Varje adress har ett unikt id, en gata, ett postnummer och en postort.

ER-diagram:



Logisk databasdesign
Jag översätter ovanstående ER-diagram till bas-relationer för att på så sätt skapa en uppsättning av relationer för databasen. Jag översätter alltså ER-diagrammet till tabeller som sedan ska gå att stoppa in i en relationsdatabashanterare.

Jag börjar med att göra en tabell för varje entitetstyp eftersom jag vet att varje vanlig entitetstyp blir en tabell enligt reglerna för relationsdatabaser. Entiteterna som finns i ER-diagrammet ovan är: Person, Adress, Spelare, Ledare och Lag. Jag vet även att vanliga attribut blir kolumner i tabellerna så jag kan lägga till de direkt. Attributet spelare i entiteten Lag läggs inte till i tabellen eftersom det

är ett härlett attribut. Nu kan jag även ta reda på primärnycklarna för varje tabell. Varje unikt attribut, alltså de understreckade attributen i ER-diagrammet blir en primärnyckel i den tabellen. Person har PersonID som PK, Adress har AdrID som PK och Lag har Lagnamn som PK.

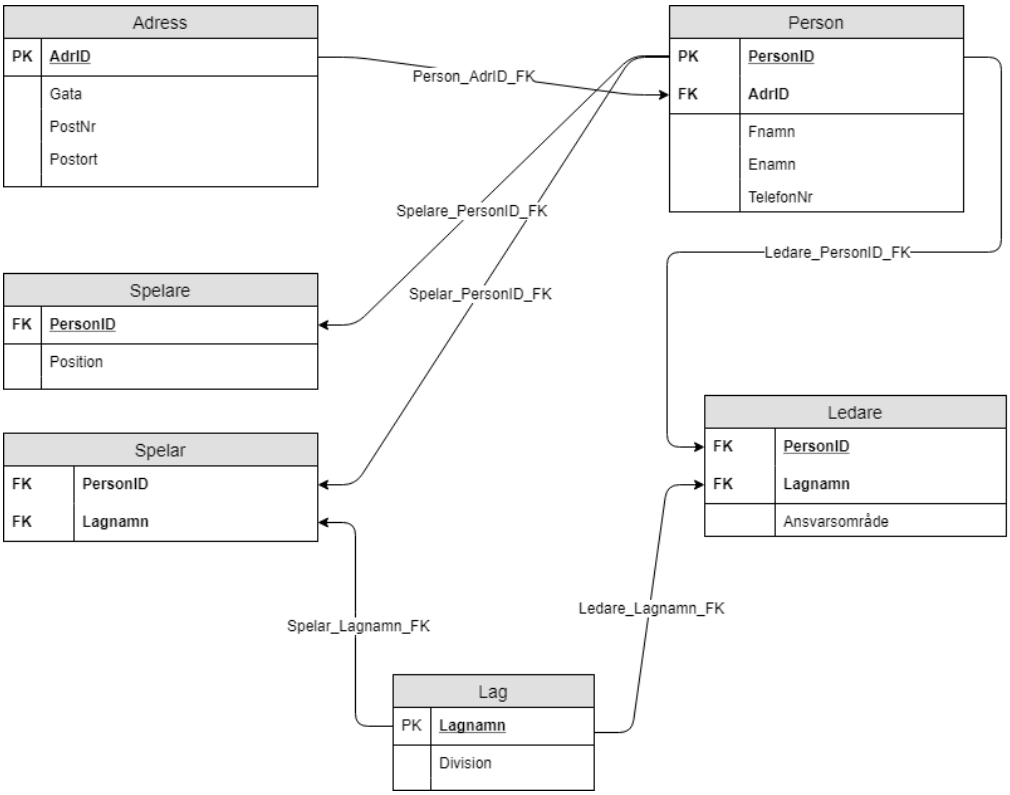
Jag måste även räkna in alla sambandstyper från ER-diagrammet.

Spelare och Ledare är subklasser av superklassen Person. Dessa entiteter får då en egen tabell som nämnt ovan men dessa är beroende av Person. Om det inte finns någon Person så finns det inte heller några Spelare eller Ledare. I båda subklasserna blir då PersonID referensattribut.

Varje 1:N-sambandstyp blir ett referensattribut i ”många”-entitetstypens tabell. Adress och Person har ett sådant samband och därför ska Adress-entitetens primärnyckel lagras som ett främmande nyckel-attribut i Person-entiteten. Alltså AdrID blir ett referensattribut i Person. Även Lag och ledare har ett sådant samband. Där blir Lagnamn ett referensattribut i Ledare. Det går alltså inte att lagra en adress i tabellen Person utan att den adressen finns i tabellen Adress och det går inte heller att vara ledare för ett lag som inte finns i tabellen Lag.

Varje N:M-sambandstyp blir en egen tabell. Därför blir sambandet mellan Spelare och Lag en egen tabell vid namn Spelar. Spelar-tabellen får då två referensattribut, PersonID från Spelare och Lagnamn från Lag.

I diagrammet nedan visas allting tydligt:



Fysisk databasdesign

I denna del beskriver jag dataintegritet- och referensintegritetsvillkoren.

Detta beskrivs i diagrammet nedan:

Person	
PK	PersonID SMALLINT NOT NULL
FK	AdrID INT(11) NOT NULL
	Fnamn VARCHAR(20)
	Enamn VARCHAR(40)
	TelefonNr VARCHAR(20)

Spelare	
FK	PersonID SMALLINT NOT NULL UNIQUE
	Position VARCHAR(2)

Adress	
PK	AdrID INT(11) NOT NULL
	Gata VARCHAR(50)
	PostNr VARCHAR(6)
	Postort VARCHAR(40)

Ledare	
FK	PersonID SMALLINT NOT NULL UNIQUE
FK	Lagnamn VARCHAR(20) NOT NULL
	Ansvarsområde VARCHAR(20)

Lag	
PK	Lagnamn VARCHAR(20) NOT NULL
	Division VARCHAR(20)

Spelar	
FK	PersonID SMALLINT NOT NULL
FK	Lagnamn VARCHAR(20) NOT NULL

Dataintegritetsvillkor:
VARCHAR, INT och SMALLINT bestämmer vilken typ av data som får fyllas i(Textsträng eller siffror) och ett max antal tecken som får anges. NOT NULL innebär att dessa värden måste vara ifyllda. UNIQUE innebär att endast ett specifikt värde får finnas i samma kolumn i en tabell. T.ex. så kan en person med ett PersonID inte finnas på två eller fler rader i tabellen Spelare.

Referensintegritetsvillkor:
En person måste ha en adress och den adressen måste även existera i tabellen Adress. Annars ska det inte gå att lägga till personen.

En spelare måste ha ett PersonID som måste existera i tabellen Person. En spelare måste även ha ett unikt sådant PersonID, finns en person med samma PersonID som användaren försöker lägga till så ska det inte gå.

Samma sak gäller för ledare som för spelare. En Ledare ska också leda ett lag och det lagets Lagnamn måste existera i tabellen Lag.

Samma sak som för spelare och ledare gäller även i Spelar-tabellen. Alltså att ett PersonID måste fyllas i och det måste existera i tabellen Person. Här behöver inte PersonID vara unikt eftersom en spelare kan återkomma flera gånger i ta-

bellen i och med att en spelare kan spela i flera lag samtidigt. En spelar måste även spela i ett lag och det Lagnamnet måste existera i tabellen lag.

Index
Jag lägger även till unika index för PersonID i både spelare och ledare så att man inte ska kunna lägga till samma person flera gånger i de tabellerna.

De-normalisering
Jag har valt att inte de-normalisera något. Det som skulle kunna de-normaliseras är adress och person, att jag lägger till fulla adressen i person-tabellen. Istället för ett Adress ID i person-tabellen som kan vara lite svårt att ta reda på. Här använder jag istället en fråga med JOIN som sätter ihop tabellerna. I spelar-tabellen skulle jag även kunnat haft med namn på spelare istället för bara id. Jag använder istället en procedur för att få den utskriften.

Databas implementering
Jag börjar med att skapa en databas där jag sedan lägger in tabellerna och dess villkor med hjälp av SQL-kod som jag skriver i MySQL Workbench. Jag sparar sedan de i en fil med namnet forening_create, Alltså de tabeller och villkor som skapas(create). Sedan gör jag en ny SQL-fil för alla värden jag vill lägga till i tabellerna, den sparas som forening_insert.

När allt är skapat så lägger jag till några exempelvärden som stämmer överens med de uppsatta villkoren.

Efter det så testar jag att alla villkor stämmer genom att försöka lagra en uppsättning av felaktig testdata i databasen. All felaktig testdata jag försöker lagra går ej och rätt felmeddelande skrivs ut. Då vet jag även att min databas fungerar som tänkt. Det jag försökt lagra är bland annat NULL-värden i primärnycklar och främmande nycklar där det inte får vara NULL. Jag har även försökt lagra flera rader med samma värden där främmande nyckeln är unik.

När jag testat allt så skriver jag olika SQL-frågor som är relevanta för föreningen jag gör databasen åt. En viktig ändring jag gjorde med en fråga var att lägga till auto_increment till PersonID så att användare inte behöver fylla i det. En ny person får då automatiskt ett unikt PersonID. Frågan jag ställde ser ut såhär: ”ALTER TABLE PERSON MODIFY COLUMN PersonID SMALLINT NOT NULL auto_increment;”. Övriga frågor och implementering finns i ett dokument med beskrivning som ligger i samma mapp som denna rapport. Mappen heter ”SQL”.

Användning av databas
Databasen heter ”forening” och är implementerad i XAMPP som använder sig av MySQL.

5 Resultat

Jag har nu skapat en simpel databas för en delmängd i en idrottsförening som vill lagra spelare och ledare, dess adress samt vilka lag de tillhör. Denna databas lagrar spelare och ledare som personer med person- och kontaktinformation. Det går att ta reda på vilket/vilka lag en spelare spelar i och vilket lag en ledare tillhör. Det går även att ta reda på information om specifika lag.

6 Slutsatser

Databasen skulle kunna användas av en förening även fast den är ganska simpel. Det går att bygga vidare på denna databas för att kunna lagra mer information och inkludera flera delar i föreningen som ekonomi. Tanken med denna databas var att den bara skulle sköta om en delmängd av föreningen och kan därför implementeras i andra delar för att skapa en komplett databas till en professionell förening.

Källförteckning

[1] SQLCourse, ”What is SQL?” http://www.sqlcourse.com/intro.html
Hämtad 2018-03-10

[2] Tore Risch, *Databasteknik*. 1:16 uppl. Lund: Studentlitteratur AB., 2017

Bilaga A: Dokumentation av egenutvecklad programkod

Exempel på underrubrik