Databasimplementation

Av Erik Gustafsson (a23erigu)

Contents

Diagrammet

Delen av diagrammet som valdes att användas för databasen var Tomtenisse, Mellanchef, Byggare, leksak och verktyg.

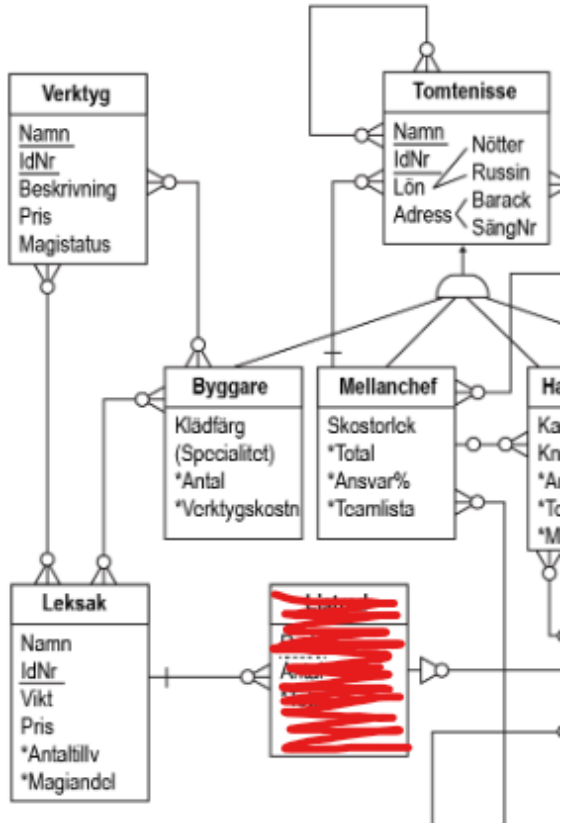

Bild på delen av diagrammet jag valt

Anledningen jag valde att ta dessa delar var då jag gärna ville göra något med nissarna och hur de administrativt hörde samman. Att bara ha det administrativa hade inte riktigt fungerat för databasen dock så jag valde att ta bort handledare och informatör för att i stället få med verktyg och leksak då det gav databasen mer av en fysisk grund att kunna använda men som ändå hängde ihop med nissarna genom byggare.

Antaganden

Några av antaganden som gjordes till databasen var i tomtenisse där jag valde att lägga lön direkt i tomtenisse medan adress fick vara sin egen tabell. Detta gjordes då både nöter och russin kan klara sig direkt i tomtenisse tabellen utan några riktiga problem medan adress hade problemet av att två tomtenissar inte kan ha samma adress så den fick vara sin egen tabell för att kunna uppnå detta krav. Till tomtenisse så skapade också en tabell för lagen då det bara ska få vara två nissar i ett lag så att ha det som sin egen tabell blev det enklades sättet att uppfylla detta. Ett annat antagande som gjordes var att inte ta med specialister i bygger då det kändes extremt komplicerad att lyckas få med korrekt och den uppfyllde inte riktigt någon funktion så den togs inte med.

Datatyper

Det användes några olika datatyper i denna databas. Det vanligaste är varchar, int och datetime. Varchar är nog den mest använda och är primärt för att kunna få med strängar i databasen då det är det enklaste sättet att kunna göra detta. Int användes om någon rad i en tabel skulle innehålla nummer då det är det enklast att spara nummer som inte och det tillåter en att göra matematiska ekvationer på det (int används dock INTE för id då id är ofta bestående av både nummer och bokstäver). Datetime är till för att kunna spara tid i en databas och använde för exakt den anledningen då att försöka spara tid annars kan vara väldigt jobbigt.

Constraints

Constrains har använts i näst intill alla tabeller av databasen. Not null är nästan på varje rad i alla tabeller förutom några speciella som exempel skostorlek i tomtenisse. Not null behöver nog inte vara på alla rader men det används mest som en säkerhets grej så en massa null värden inte råka hamna i systemet. Unique används inte så mycket men finns ändå på ställen där jag såg det som viktigt att det endast får finnas en variant av den inputen som i tillexempel i ID för tomtenissar. Check används på vissa ställen i databasen för att kunna uppfylla vissa krav från uppgift beskrivningen som tillexempel till byggare där deras kläder inte får ha färgerna röd eller burgendy så en ”not rlike” fick implementeras.

check (klädfärg not rlike "röd|burgundy")

hur checken för klädfärgen ser ut i koden

Denormalisering

För databasen så användes en av varje typ av denormaliserings tekniker.

Merging

Merging var faktiskt en av de svårare för mig att implementera då jag personligen föredra att ha mitt program väldigt splittrat men till slut så hittades det att Chefnisse kunde sättas ihop med tomtenisse. Chefnisse var ett arv av tomtenisse, hade bara relationer med tomtenisse och dess enda icke främmande nyckel var skostorlek så att sätta ihop den med tomtenisse var inte något problem. En annan sak var att både tomtenisse och chefnisse skulle ha främmande nycklar från varandra vilket skapa ett problem när tabellerna skulle implementeras men det problemet kunde i stället undvikas genom att sätta ihop tabellerna. Att sätta ihop tabellerna gjorde dock att en ny tabell för vem som är chef över vem behövdes skapas då tomtenisse annars skulle blivit en väldigt stor tabell.

```
create table Tomtenisse(
        Namn varchar(20) not null,
        IdNr char(23) not null unique,
        Nötter int not null,
        Russin int not null,
        Skostorlek varchar(20),
        check (Skostorlek rlike 'mini|medium|maxi|ultra|mega'),
        check (IdNr rlike '[0-9][0-9][0-9][0-9][0-9][0-9]-[0-9][0-9][0-9][0-9]-[0-9]-[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'),
        primary key(Namn, IdNr)
)engine=innodb;
```

Tomtenisse tabellen efter att chefnisse sates in i den.

Codes

Codes var den enklaste av normaliserings teknikerna att implementera. Jag valde att implementera codes på leksaker med deras namn då leksakers namn inte är copywriter så det kan finas flera med samma. För att implementera codes så skapades en ny tabell vid namn av LeksaksNamn som då innehåller namnen och deras koder.

```
create table LeksakNamn(
        Namn varchar(20) not null,
        NamnKod char(8) not null unique,
        primary key(NamnKod)
)engine=innodb;
```

Koden för LeksaksNamn tabellen

Jag valde också att görat så leksak kunde ha både koder och texter som namn för att koppla till leksaksnamn men också så om det finns en leksak med ett unikt namn behöver de inte göras om till kod.

Vertikal split

Vertikal split var något jag också hade några problem med att hitta vart jag skulle kunna görat men till slut ås valdes verktyg tabellen. Verktyg tabellen fick bli splittrad till två vid namn av MagiskaVerktyg och IkeMagiskaVerktyg då det kan vara så att en nisse bara vill/kan använda icke magiska verktyg och då sliper de titta igenom alla verktyg. För att skapa dessa tabeller så duplicerades verktyg. Den nya verktyg förlorade magistatus och byta namn medan den gamla bara bytte namn. Detta gjorde att tabellerna är väldigt lika så information kan flytas mellan de utan några riktiga problem.

```
create table MagiskaVerktyg(
        Namn varchar(20) not null,
        IdNr char(8) not null unique,
        Pris int not null,
        Magistatus int,
        primary key(Namn, IdNr)
)engine=innodb;
```

Koden för MagiskaVerktyg tabellen

```
create table IkeMagiskaVerktyg(
        Namn varchar(20) not null,
        IdNr char(8) not null unique,
        Pris int not null,
        primary key(Namn, IdNr)
)engine=innodb;
```

Koden för IkeMagiskaVerktyg tabellen

Horizontal split

Den horisontella spliten som skulle göra i databasen var inte jättesvår att göra då det fans en perfekt kandidat att görat på, nämligen verktygs beskrivning. Anledningen verktygbeskrivningen fungerade så väl är då beskrivningar ofta är långa och tar mycket resurser att spara och hämta. För att göra spliten så togs Beskrivning ut ur MagiskaVerktyg och IkeMagiskaVerktyg för a skapa en ny tabell vid namn VerktygBeskrivning som länkade tillbaka till MagiskaVerktyg och IkeMagiskaVerktyg.

```
create table VerktygBeskrivning(
        Namn varchar(20) not null,
        IdNr char(8) not null unique,
        Beskrivning varchar(60) not null,
        primary key(Namn, IdNr)
)engine=innodb;
```

Koden för VerktygBeskrivning tabellen

Indexering

De skapades två index i programmet (utöver de autogenererade). Indexen skapades för att göra det enklare att söka på specifika rader i programmet. Det två indexen är ett index på skostorlek från Tomtenisse och ett index på klädfärg från Byggare.

```
create index TomtenisseSkostorlek on Tomtenisse(Skostorlek ASC) using BTREE;
create index ByggareKlädfärg on Byggare(Klädfärg ASC) using BTREE;
```

koden för båda index

Anledningen det sattes index på dessa två rader är då båda kan vara saker som man ofta vill söka på men inte är nycklar. Skostorlek har med att göra om en tomtenisse är en chef eller inte (också vilken grad av chef de är) och efter som chefnissar inte har någon egen tabell så behövs skostorlek för att kunna hitta vilka som är chefer. Klädfärg för byggare har att göra med vad byggarnissar specialiserar sig på att bygga vilket kan vara något viktigt att söka på för både tomten själv men även chefnissar så de vet vem som är bra på att bygga vad. Dessa tabeller har också att de båda kommer mest sökas på då det endast ändras/läggs till om nya tomtenissar skapar/anlitas så tomten behöver inte vara räd att indexen kommer slöa för mycket.

Vyer

Tre vyer används i denna databas. Av dessa tre så är två simplificationer och en specialisering. Det två simplificationerna är allaVerktyg och Mellannisse. AllaVerktyg fungerar som det låter och visar alla verktyg magiska eller inte. Den uppnår detta genom att göra en union mellan MagiskaVerktg och IkeMagiskaVerktyg där magistatus för IkeMagiskaVerktyg sätts som null.

```
create view allaVerktyg as

select Namn, IdNr, Pris, Magistatus from MagiskaVerktyg union

select Namn, IdNr, Pris, Null as "Magistatus" from IkeMagiskaVerktyg;
```

koden för allaVerktyg vyn

Mellannisse är en vy som gör det möjligt att få ut alla chefer. Anledningen denna skapades var då chef tabellen sattes ihop med Tomtenisse så kunde man inte se vilka som var chefer på något enkelt sätt. Vyn fungerar ganska simpelt då det är en select på allt där skostorlek inte är null.

```
create view Mellanisse as

select * from Tomtenisse where Skostorlek is not NULL;
```

koden för Mellannisse vyn

Sista vyn är en specialist vy vid namn av KräverMagi. KräverMagi är till för att kunna få vilka leksaker som behöver magi för att kunna skapas. Denna vy var tänkt att vara för de byggarnissar som specialiserar sig inom magi så det vet vilka leksaker de ska jobba med. Denna vy var lite mer komplicerad att göra då den krävde en select mellan Leksak Behöver och allaVerktyg för att få vilka leksaker som inte behöver magi för att skapas.

```
create view KräverMagi as

select  Leksak.NamnKod, Leksak.IdNr from Leksak, Behöver, allaVerktyg

where Leksak.IdNr = Behöver.LIdNr and allaVerktyg.Namn = Behöver.VNamn and
allaVerktyg.IdNr = Behöver.VIdNr and allaVerktyg.Magistatus is not null;
```

koden för KräverMagi vyn

Stored procedures

Till databasen så skapades fyra procedur för att hjälpa till. Den första av proceduren är getNissar och fungerar genom att ge ut alla tomtenissar som finns. Denna procedur skapades då det kan finnas tomtenissar som inte ska ha tillgång till tomtenisse tabellen men ska ändå kunna se alla tomtenissar.

```
create procedure getNissar()

begin

        select * from Tomtenisse;

end//
```

koden för getNissar proceduren

Procedur två är getLeksakerPåPris vilket hämtar leksaker med mindre eller lika med pris än vad som skickas med. Anledningen denna skapades var då det kan vara viktig för byggarnissar att veta vilka leksaker som är billiga så de nya/sämre byggarnissarna vet vad de kan jobba på. Procedur två fungerar genom att de tar med ett nummer när proceduren kallas vilket sen används för att få ut alla leksaker som har ett pris under eller lika med det numret.

```
create procedure getLeksakerPåPris(in checkPris int)

begin

        select * from Leksak where Leksak.Pris <= checkPris;

end//
```

koden för getLeksakerPåPris

Tredje proceduren är väldigt lik till den andra proceduren och göt mer eller mindre samma sak förutom att den stoppar en från att skriva in ett tal som är under noll. Denna procedur ska användas på samma sätt som procedur två bara att denna så kan inte tomtenissarna skriva in något negativt tal då det inte ska finnas någon leksak med negativt pris. Proceduren var inte svårare att göra då det användes procedur två som grund men sen la till en if sats som kollar om det medskickade talet är under noll och ger då en signal.

```
create procedure getLeksakerPåPris(in checkPris int)

begin

        if (checkPris < 0) then

                signal sqlstate '45000' set message_text = "priset måste vara 0
                eller mer";

        else

                select * from Leksak where Leksak.Pris <= checkPris;

        end if;

end//
```

koden för den nya getLeksakerPåPris proceduren

Den sista proceduren är görVerktygMagisk och gör som det heter om ett icke magiskt verktyg till magiskt. Denna procedur finns för att göra det enklare för tomten och chefnissar att göra om ett icke magiskt verktyg till magiskt om det skulle behövas för nya leksaker. Proceduren tar in tre värden vilket är VerktygNamn, VerktygID och inMagistatus. VerktygNamn och VerktygID används för att kunna hita vilket verktyg som ska ändras medan inMagistatus behövs för att bestämma hur magiskt verktyget ska vara (måste vara ett nummer mellan 0 och 12). Proceduren fungerar genom att den börjar att ta det från IkeMagiskaVerktyg och lägga in det MagiskaVerktyg med magistatus som null. Efter att ha lagt in det nya så uppdaterar de det nya verktyget med den insatta magistatusen. Till sist så tar det bort det gamla verktyget I IkeMagiskaVerktyg. Finns även en if sats för om inMagistaus inte är mellan 0 och 12.

```
create procedure görVerktygMagisk(in verktygNamn varchar(20), in verktygID char(8), in
inMagistatus int)

begin

        if (inMagistatus <= 0 or inMagistatus >= 12) then

                signal sqlstate '45000' set message_text = "Maginivån måste
                vara mellan 1 och 11";

        else

                insert into MagiskaVerktyg(Namn, IdNr, Pris, Magistatus)

                select Namn, IdNr, Pris, Null as "Magistatus" from
                IkeMagiskaVerktyg where Namn = verktygNamn and IdNr =
                verktygID;


                update MagiskaVerktyg set Magistatus = inMagistatus where
                verktygNamn = Namn and verktygID = IdNr;


                delete from IkeMagiskaVerktyg where verktygNamn = Namn
                and verktygID = IdNr;

        end if;

end//
```

koden för görVerktygMagisk proceduren

Triggers

Databasen innehåller fem triggers. De två första triggers har båda att göra med loggen av vilka byggarnissar som använt vilka verktyg. Den första triggern heter BörjaAnvända och loggar när en byggarnisse börjar använda ett verktyg medan den andra triggern vid namn av SlutaAnvända loggar när en byggarnisse slutar använda verktyg. Dessa triggers är till för att byggarnissar ska kunna hålla koll på vem som använt vilket verktyg och när om något skulle hända. Båda triggrarna är sätta på AnvändsAv skillnaden är att BörjaAnvända är efter en insert medan SlutaAnvända är efter en delete annars så göra båda en insert i AnvändsAvLog med den nya/gamla data respektive.

```
create trigger BörjaAnvända after insert on AnvändsAv

for each row begin

        insert into AnvändsAvLog(Händelse, TNamn, TIdNr, VNamn, VIdNr, Tid)
        value("Börja använda", NEW.TNamn, NEW.TIdNr, NEW.VNamn, NEW.VIdNr,
now());

end//
```

koden för BörjaAnvända triggern

```
create trigger SlutaAnvända after delete on AnvändsAv

for each row begin

        insert into AnvändsAvLog(Händelse, TNamn, TIdNr, VNamn, VIdNr, Tid)
        value("Sluta använda", OLD.TNamn, OLD.TIdNr, OLD.VNamn, OLD.VIdNr,
now());

end//
```

koden för SlutaAnvända triggern

Den tredje triggern är LägTillNamnKod. Denna trigger har att göra med att när namn koder läggs till för leksaker så ska alla leksaker i Leksaker tabellen som har namnet som lagts in bytas till namn koden i stället. En log för vilka namn som lagts till i LeksakNamn skapades också så det går att se vilka namn som lagts till och när. Största saken med denna trigger är att man sliper behöva uppdatera Leksaker själv när ett nytt namn kod läggs till utan databasen kan göra det själv man kan också se vilka namn som lagts till när. Koden för LägTillNamnKod är egentligen bara en insert först som lägger in det som hänt i loggen och sen en uppdate på Leksaker tabellen med det nya namn koden.

```
create trigger LägTillNamnKod after insert on LeksakNamn

for each row begin

        insert into LeksakNamnLog(Händelse, Namn, NamnKod, Tid)

        value("Lagt till", new.Namn, new.NamnKod, now());

        update Leksak set NamnKod = new.NamnKod where NamnKod =
        new.Namn;

end//
```

koden för LägTillNamnKod triggern

Den två sista triggrarna är SäljMagiskaVerktyg och SäljIkeMagiskaVerktyg vilka körs när verktyg tas bort från respektive tabell. Båda triggrarna används för att logga när ett verktygs tas bort och för att stoppa en från att sälja ett verktyg som någon använder. Användningen av triggrarna är för att kunna hålla koll vilka verktyg som tagits bort och när men också för att stoppa verktyg från att tas bort om de används. Båda triggrarnas kod är nästan identisk med några små ändringar som vilken tabell det ligger på och vad magistatus är. Den första delen i triggrarna är en if sats som kollar om någon byggarnisse använder verktygen och ger ett error om det gör det. Om det kommer förbi if satsen så insertar det i VerktygLog och sen tar bort verktygens beskrivning och hur verktyget skulle användas.

Jag är medveten om att båda triggrarna är after delete så teknist sätt så har redan verktyget tagits bort även om if satsen ger error men jag kunde inte list ut hur den skulle kunna göras som before delete så det fick bli på detta sätt.

```
create trigger SäljMagiskaVerktyg after delete on MagiskaVerktyg

for each row begin

        if ((select count(*) from AnvändsAv where VNamn = old.Namn and VIdNr =
        old.IdNr) > 0) then

                signal sqlstate '45000' set message_text = "Deta verktyg
                används av någon";

        else

                insert into VerktygLog(Händelse, Namn, IdNr, Pris, Magistatus,
                Tid)

                value("såldes", old.Namn, old.IdNr, old.pris, old.magistatus,
                now());

                delete from VerktygBeskrivning where Namn = old.Namn and
                IdNr = old.IdNr;

                delete from Behöver where VNamn = old.Namn and VIdNr =
                old.IdNr;

        end if;

end//
```

koden för SäljMagiskaVerktyg

```
create trigger SäljIkeMagiskaVerktyg after delete on IkeMagiskaVerktyg

for each row begin

        if ((select count(*) from AnvändsAv where VNamn = old.Namn and VIdNr =
        old.IdNr) > 0) then

                signal sqlstate '45000' set message_text = "Deta verktyg
                används av någon";

        else

                insert into VerktygLog(Händelse, Namn, IdNr, Pris, Magistatus,
                Tid)

                value("såldes", old.Namn, old.IdNr, old.pris, null, now());

                delete from VerktygBeskrivning where Namn = old.Namn and
                IdNr = old.IdNr;

                delete from Behöver where VNamn = old.Namn and VIdNr =
                old.IdNr;

        end if;

end//
```

koden för SäljIkeMagiskaVerktyg

Rättigheter

En användare skapades till detta program vilket fick bli en byggarnisse.

```
create user "a23eriguByggarNisse"@"%" identified by "ByggaBil";
```

koden för att skapa en användare

Denna användare fick sen rättigheter som skulle kunna vara lämpliga för en byggarnisse. Byggarnissen fick fulla rättigheter för AnvändsAv och Bygger då det ska själva kunna bestämma vad de använder för verktyg och vilken leksak de bygger. Helst så ska byggarnissar ändats kunna byta på vad de själva använder för verktyg och bygger inte andra byggarnissar men jag kunde inte lista ut hur det skulle kunna göras så fick bli så här.

```
grant select, delete, insert on TomteVerkstad.AnvändsAv to
"a23eriguByggarNisse"@"%";

grant select, delete, insert on TomteVerkstad.Bygger to "a23eriguByggarNisse"@"%";
```

koden för att ge fulla rättigheter

Byggarnissen har också fått rättigheter att selecta på ett par tabeller vilket inkluderar båda verktyg tabeller, allaVerktyg vyn, vilka verktyg som behövs, leksaker, leksak namn koder, vyn för vilka leksaker som kräver magi och logarna på verktyg, användes och leksaks namn kod. Anledningen jag gav dessa till byggarnissen är då det är alla saker som byggarnissar skulle kunna behöva se för att arbeta.

```
grant select on TomteVerkstad.allaVerktyg to "a23eriguByggarNisse"@"%";
```

exempel på en av rättigheterna utgiven

Det sista rättigheterna är på procedurers där byggarnissen fick tillåtelse att använda getNissar och getLeksakerPåPris då byggarnissar borde kunna se de andra nissaran (utan att behöva ha någon tillgång till tomtenisse tabellen) och att kunna hitta leksaker beroende på pris kan hjälpa de i deras arbete.