Databasimplementation

Av Erik Gustafsson (a23erigu)

Contents

[Diagrammet 2](#_Toc177742141)

[Antaganden 2](#_Toc177742142)

[Datatyper 3](#_Toc177742143)

[Constraints 3](#_Toc177742144)

[Denormalisering 3](#_Toc177742145)

[Merging 3](#_Toc177742146)

[Codes 4](#_Toc177742147)

[Vertikal split 5](#_Toc177742148)

[Horizontal split 5](#_Toc177742149)

[Indexering 6](#_Toc177742150)

[Vyer 6](#_Toc177742151)

# Diagrammet

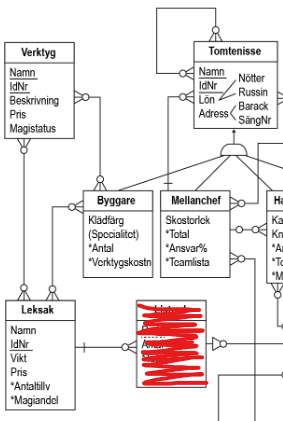
Delen av diagrammet som valdes att användas för databasen var Tomtenisse, Mellanchef, Byggare, leksak och verktyg.

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][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

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 not null,

primary key(Namn, IdNr)

)engine=innodb;

Koden för MagiskaVerktyg

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

## 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

# 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