1 Introduktion

Denna rapport beskriver arbetet under det första av kursens tre delmoment. I detta steg har en databas konstruerats för den fiktiva organisationen ”Polismyndigheten för Undanhållande av Centrala Kunskaper Om utomjordingar” som går under förkortningen ”PUCKO”.

I den första fasen i arbetet studerades kravspecifikationen som tillhandahållits som underlag. Denna specifikation innehöll en stor mängd krav som översteg det omfång som studenten skulle modellera. Således extraherades de mest intressanta entiteterna – 7st av totalt 26st och antaganden gjordes för att skapa roller med rättigheter och möjligheter som ledde till en rimlig förmåga att operera i det begynnande informationssystemet (IS:et).

I det nästa steget togs begränsningar fram s.k. ”constraints”. Dessa lades in delvis med simpla check-villkor men även med ett antal triggers, vilket är funktioner som aktiveras vid en särskild händelse eller interaktion med databasen.

Avslutningsvis koncentrerades arbetet kring att optimera databasen och skapa rättigheter. Optimeringen uppnåddes genom att tilldela så små datatyper som nödvändigt samt genom att skapa index, merga och denormalisera relevanta tabeller. Rättigheterna togs fram genom att skapa användare ”users” och ge dessa åtkomst till tabeller och/eller vyer baserade på underliggande tabeller.

2 Modellering

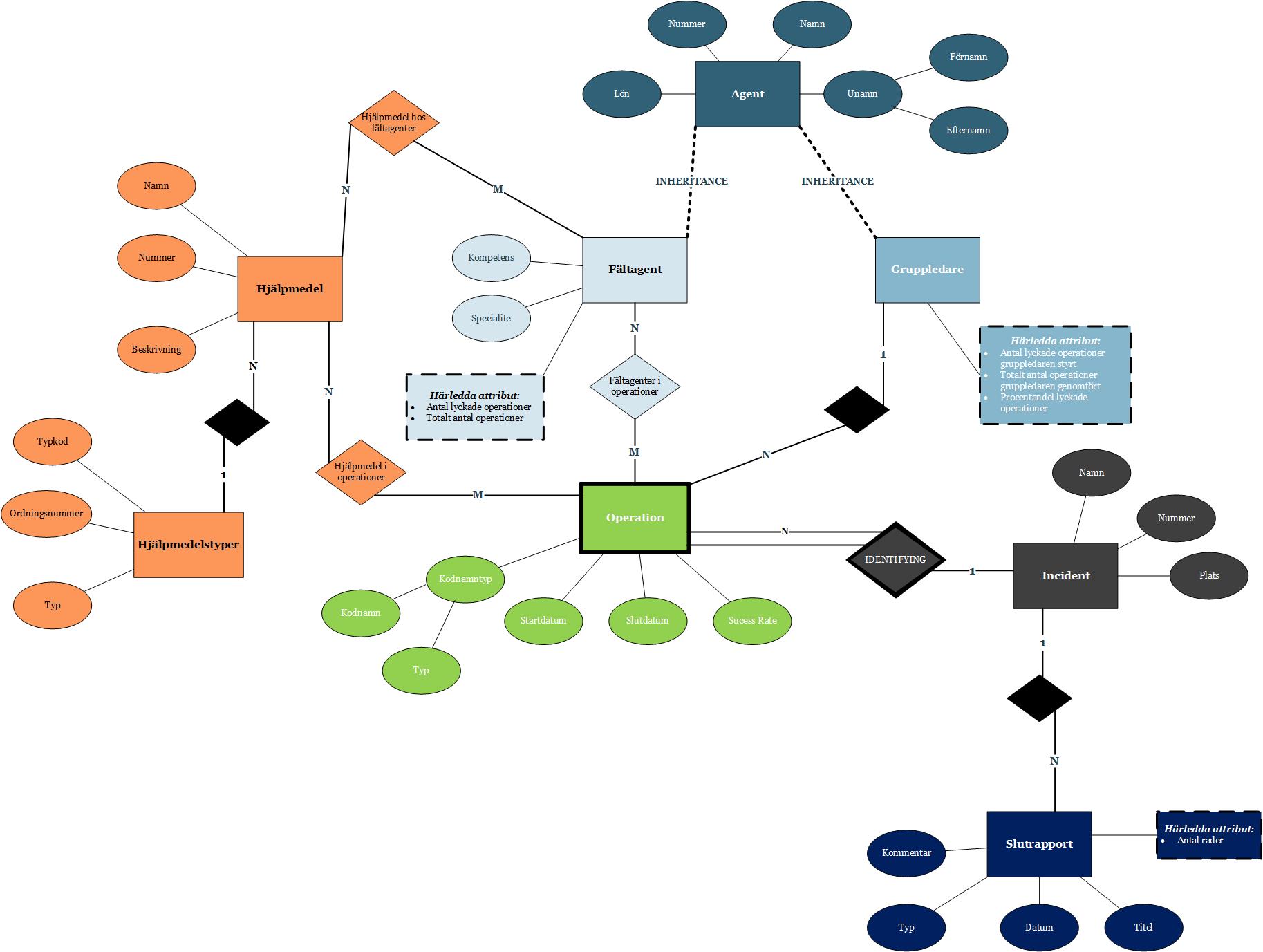
Den framväxande systemprototyp som skulle skapas modellerades med ER-, Use Case- och flödesdiagram. Detta skapade ett stöd för att inkludera alla rimliga krav och göra välövervägda antaganden kring förändringar i kravspecifikationen.

**2.1 Entity Relationship (ER) Modell**

En ER modell togs fram utifrån det IE schema samt den kravspecifikation som en tidigare konsultfirma levererat. Nedärvningsrelationen mellan Agent-entiteten samt Fältagent & Gruppledare är disjunkt och tvingad. Således valdes det som Martin Fowler (n.d.) beskriver som ”Concrete Table Inheritance”. I denna lösning tas helt enkelt Agent-entiteten bort och dess attribut placeras i vardera av de tabeller som befinner sig nedåt i nedärvnings-relationen.

En annan förändring som gjordes från det levererade IE-diagrammet och som baserades på kravspecifikationen var att dela upp Organisations-entitetens attribut ”Kodnamntyp” till ”Kodnamn” och ”Typ”. Detta eftersom det då blir möjlig att söka på typer av operationer individuellt. Vidare skapades en enda helt ny tabell: ”Hjälpmedelstyper”. Detta gjordes genom att använda koder och ledde till två förbättringar. Dels minskades redundansen i hjälpmedelstabellen, och dels blev det möjligt att ange ”Ordningsnummer” från 1-15 för var och ett av attributet ”Typkod”.

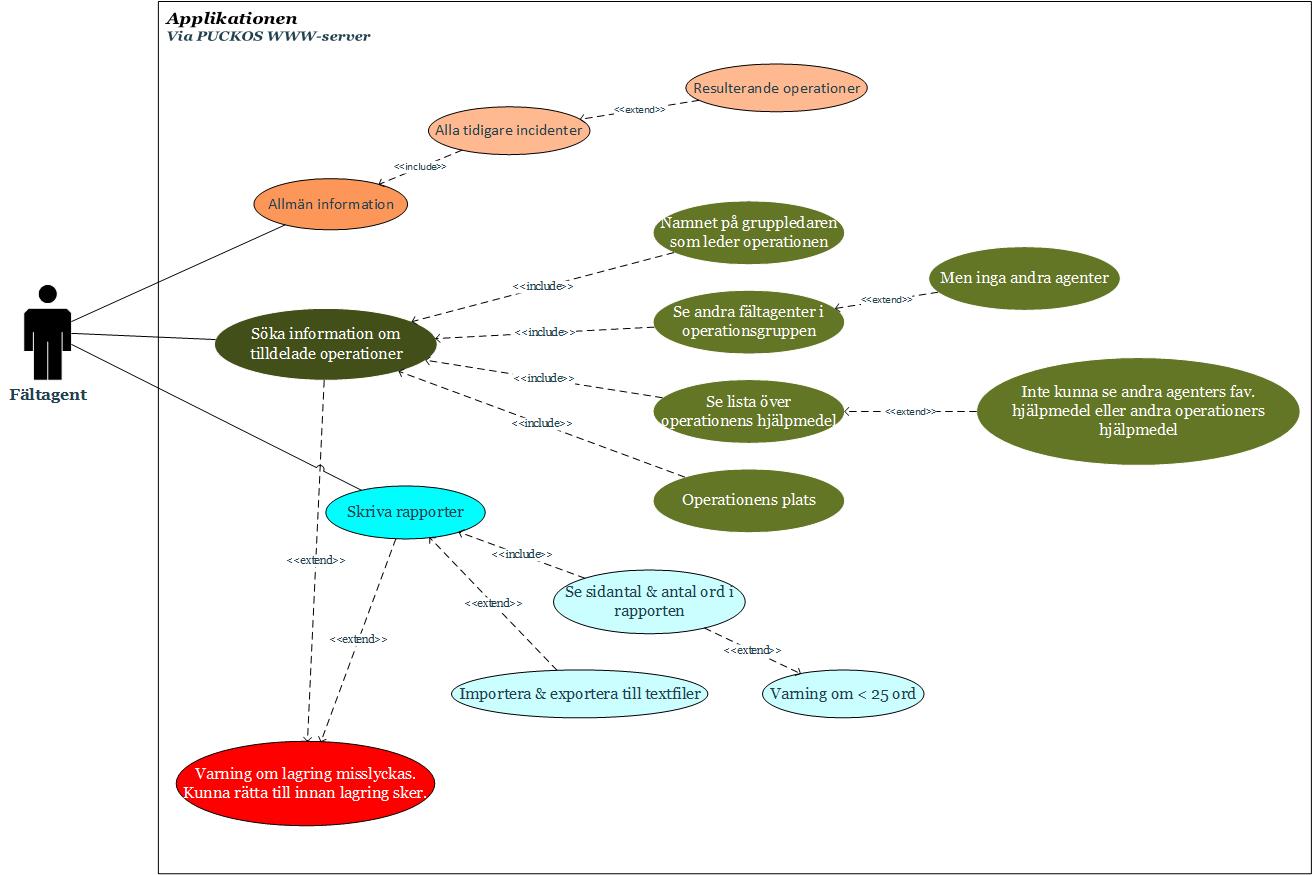
Alla många till många-relationer i diagrammet modellerades som sambandstyper som blev egna tabeller i det logiska schemat och entiteten ”Rapport” gjordes om till att endast omfatta ”Slutrapport” genom att slå samman entiteterna ”Rapport” och ”Slutrapport” i det MYSQL-baserade logiska schemat.



Figur 1. Initialt ER-diagram för databasprototypen, innan de-normalisering.

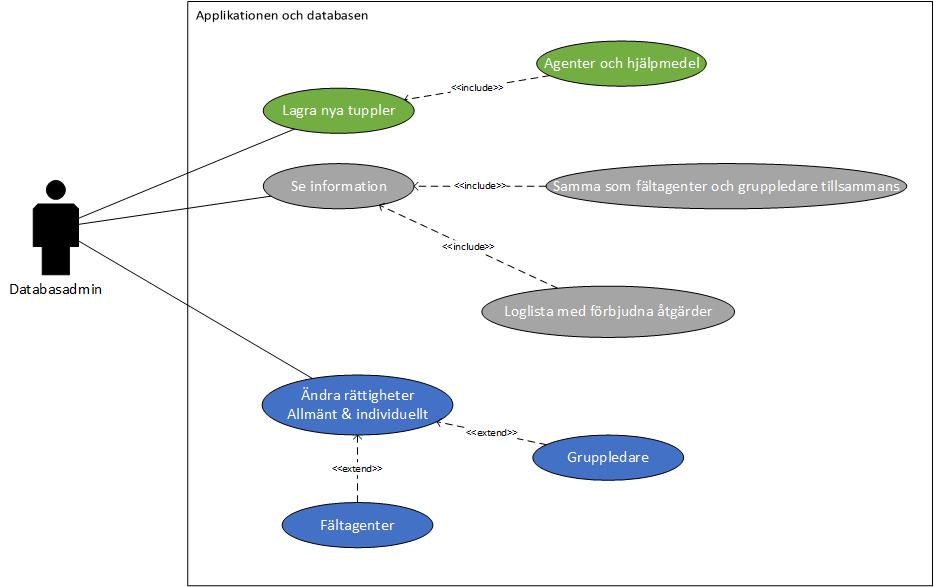
**2.2 Use Cases**

Två modeller av typen ”UML Use Case” skapades. Syftet med dessa är att modellera de handlingar som aktörer kan utföra i systemet och skapades därmed för Fältagent och Databasadministratör. Inget Use Case gjordes för den tredje rollen – Gruppledare – eftersom deras möjligheter var allt för omfattande och därför valdes i stället ett flödesdiagram för deras aktiviteter.



Figur 2. Use Case diagram för aktören "Fältagent" och dess aktiviteter i systemet.

Avslutningsvis gjordes även ett Use Case för databasadministratör, en aktör med mindre variation sina möjligheter att interagera.

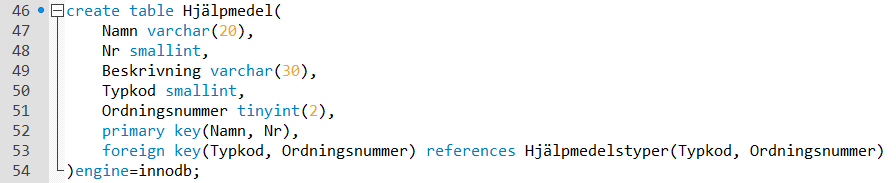


Figur 3. Use Case för "Databasadmin".

Flödesdiagrammet som gjordes för rollen “Gruppledare” var allt för omfattande för att inkluderas i detta dokument och bifogas därför som en separat fil.

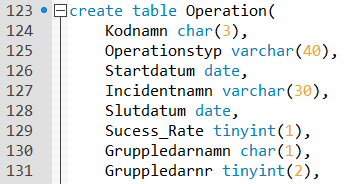
3 Tabeller och datatyper

I arbetet har ett delfokus varit att använda så små datatyper som möjligt. Detta för att optimera prestandan i databasen. I de fall där längden på en string väljs ”varchar”, eftersom det är en dynamisk data typ som kan anpassa sin längd efter det värde som stoppas in. I figur 4 finns attributet ”Namn” som är ”varchar(20)” d.v.s. det finns plats för som mest 20 tecken men om färre tecken stoppas in minskas behållaren. I samma tabell kan ses ”Nr” och ”Typkod” som är av typen ”smallint”. Denna integerbehållare (behållare som lagrar integerdata) är avsevärt större än den minsta behållaren – ”tinyint” – som bara kan ta tal mellan 0 – 255. Detta ansågs för begränsat i detta fall och därför valdes smallint som kan ta värden mellan +/- 32, 768. Attributet ”Ordningsnummer” däremot kommer alltid att vara mellan 1-15 och därför innebär det en bättre anpassning att välja integertypen ”tinyint” (Byham et al, 2017).



Figur 4. Exempel på optimering av datatyper i tabellen "Hjälpmedel".

Ett annat exempel på attribut som lagrats de och data typer som valts som behållare kan beskådas i tabellen ”Operation” (figur 5). Här har attributet ”Kodnamn” komplimenterats med ”char(3)”. Char skiljer sig från varchar genom att storleken är förutbestämd – om t.ex. längden 3 väljs kan den inte minska eller öka i storlek beroende på värdet som stoppas in. I detta fall kommer alltså behållaren alltid att lagra tre tecken. I det fall då man med säkerhet vet att just tre tecken kommer att lagras innebär det dock en prestandavinst att använda char(3) i stället för varchar(3). Eftersom det i detta fall finns kunskap om att ”Kodnamn” alltid kommer att vara tre tecken väljs således char(3).



Figur 5. Urval av "Operation"-tabellen för att visa de attribut som lagrats.

En annan intressant data typ som använts på flera håll i databasen är ”Date”. Det finns en rad alternativ för att lagra datum i SQL: ”Date”, ”Datetime”, ”Timestamp”, ”Time” och ”Year”. I denna databas valdes typen ”Date” eftersom den lagrar datum i formatet ”ååå-mm-dd”. Detta är den detaljnivå som databasen kräver för att hantera inlagring av rapporter.

Skillnaden mellan Date och Datetime är att den senare även lagrar specifikt klockslag dvs ”tt-mm-ss”; timme-minut-sekund. Detta är en större detaljnivå och begränsar därmed prestandan något mer än date.

Vidare har typen ”Timestamp” förmåga att lagra tidsintervall vilket inte varit nödvändigt i denna databas. Avslutningsvis kan nämnas ”Time” som lagrar en tidslängd upp till maximalt 838 timmar, 59 minuter och 59 sekunder (Oracle Corp., 2017).

En funktion som är relaterad till ”Date” som används för att skriva logiska villkor i stället för att lagra tid är ”Year”. I figur 6 kan ses ett exempel på hur funktionen använts för att räkna ut antalet dagar i det slutdatum som lagrats, och ”checkar” så att det är större än antalet dagar i startdatumet. På detta sätt skapa sen kontroll på att inlämnade rapporter har ett senare slutdatum än startdatum. Detta specifika villkor komplimenteras även med att året skall vara samma – annars blir det svårt att jämföra dagar om t.ex. startdatumet är 1 januari 2017 och slutdatumet är 31 december 2016. För att hantera detta fortsätter villkoret med ”OR (YEAR(Slutdatum) > YEAR(Startdatum)).



Figur 6. Checkvillkor som använder varianter av data typen "Year".

Andelen lyckade operationer i Gruppledare-tabellen lagrades med datatypen ”decimal(3, 2)”. Decimal lagrar decimaltal och har två parametrar. Talet 3 beskriver det totala antal siffror som behållaren kan lagra dvs tre siffror. Det andra talet – 2 – beskriver hur många siffror som kan lagras till höger om decimaltecknet. I detta fall kan behållaren alltså ta 9, 99 som mest och 0,01 som minst.



Figur 7. Användning av data typen decimal.

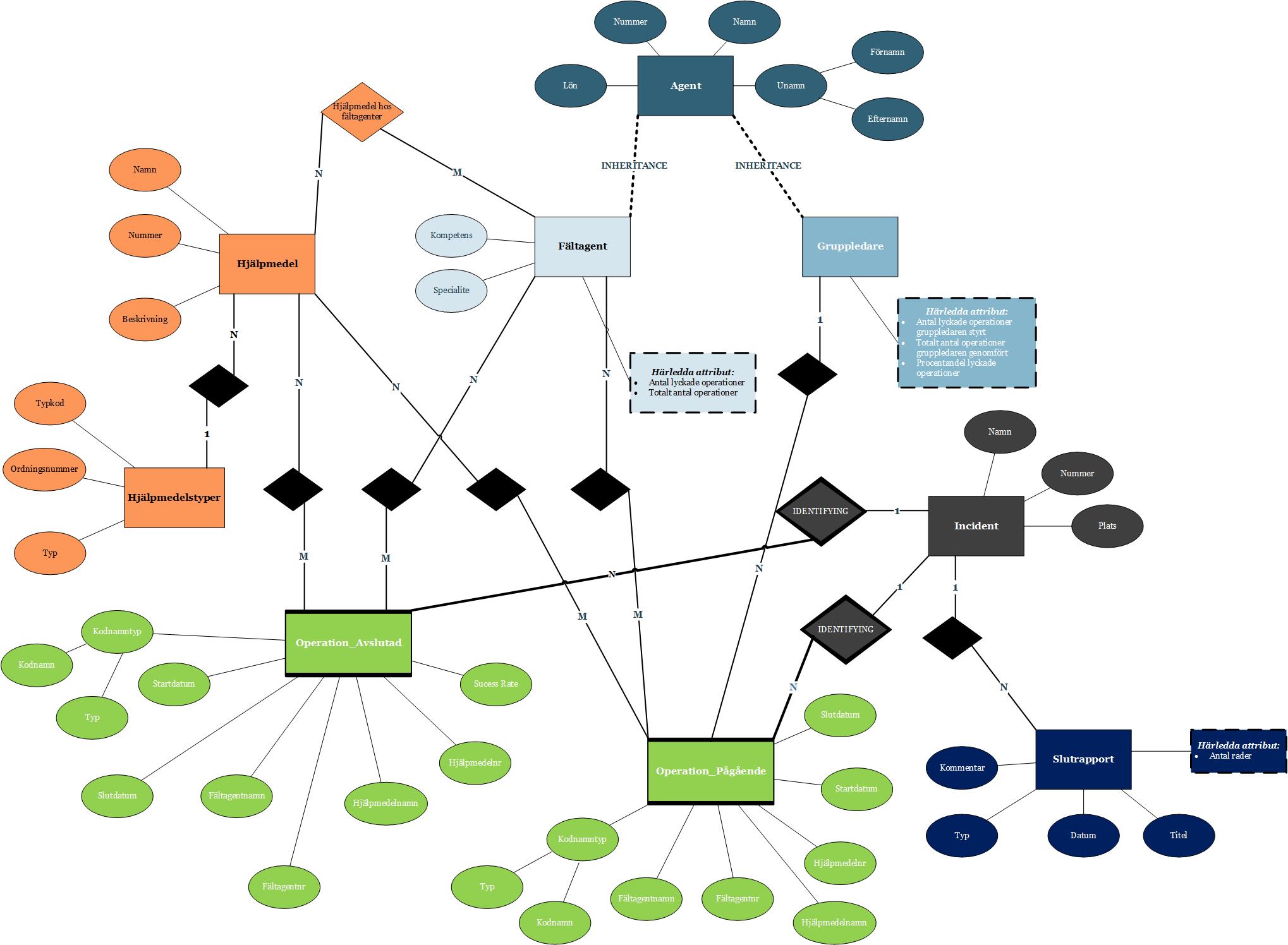
4 Produktkoder

Produktkoder användes i ett fall för att bryta upp en tabell i två. Tabellen ”Hjälpmedel” i figur 4 delades upp till att lagra vissa av sina värden i tabellen ”Hjälpmedelstyper” (figur 7). Hjälpmedelstyper-tabellen använder ”Typkod” som primärnyckel tillsammans med ”Ordningsnummer” och det är även dessa attribut som är referensnycklar i ursprungstabellen Hjälpmedel. Detta skapade en lösning för att attributet ”Ordningsnummer” skulle kunna ha värden mellan 1-15 för varje värde för primärnyckeln ”Typkod”. Det vill säga, en typkod för t.ex. ”pistol” kan finnas i upp till 15 exemplar; Typkod: 123 med Ordningsnummer: 1, 2, 3, n etc. I extratabellen Hjälpmedelstyper placerades även attributet ”Typ” vilket är ett värde som beskriver typ av hjälpmedel och detta leder till att prestandan i Hjälpmedel-tabellen ökar eftersom det finns mindre data som lagras där. Detta ansågs som en förbättring – även om den totala mängden data i databasen som helhet ökade – eftersom fältagenter behöver kunna få information kring hjälpmedel snabbt.

5 De-normaliseringar och mergar

De-normalisering gjordes genom skapandet av Hjälpmedelstyper-tabellen eftersom b.la. attributet ”Typ” flyttades dit ifrån Hjälpmedel-tabellen genom vertikal de-normalisering.

Tabellen ”Operation” omarbetades avsevärt. Dels användes en vertikal denormalisering för att skapa två varianter, dels ”Operationer\_Pågående” och dels ”Operationer\_Avslutade”. Detta eftersom det är pågående operationer som behöver snabbast åtkomst och det är den tabellen som kommer att användas mest. Detta skapade två tabeller med väldigt många kopplingar till andra N: N-tabeller. Dessa sambandstabeller skulle innehålla 2 attribut från den ena tabeller och ca 8 attribut från operations-tabellerna. Av denna anledning mergades sambandstabellerna ”Fältagenter i operationer” och ”Hjälpmedel i operationer” in i de två operationstabellerna.



Figur 8. ER-modellen efter att de-normaliseringen utförts.

5 Index

Ett index skapades för sökningen ”select \* from Operation\_Avslutad order by Slutdatum”. Detta är en fråga som förväntas göras relativt ofta då personal vill få en överblick över de operationer som avslutats, från de senast avslutade och neråt. För att uppnå detta specificeras (Slutdatum ASC), för ”ascending”.



Figur 9. Index för snabba sökningar bland de avslutade operationerna.

Ett andra index som togs fram för att snabba upp databassökningarna var indexet ”Fältagentinfo”. Med hjälp av detta kommer det att gå snabbare att ta fram information för alla fältagenter rangordnade baserat på hur stor andel av deras operationer som är lyckade.



Figur 10. Index för sökningar i fältagent-tabellen där man önskar rangordna resultatet baserat på lyckade\_operationer.

Det tredje och sista indexet som skapades var för incident-tabellens attribut ”Nr”. Detta för att möjliggöra snabba sökningar på såväl nummer som namn (som är primärnyckel).

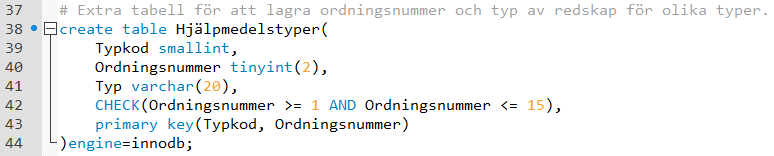


Figur 11. Incidentnummer gjordes även till index eftersom incidentnamn är primärnyckel.

6 Begränsningar

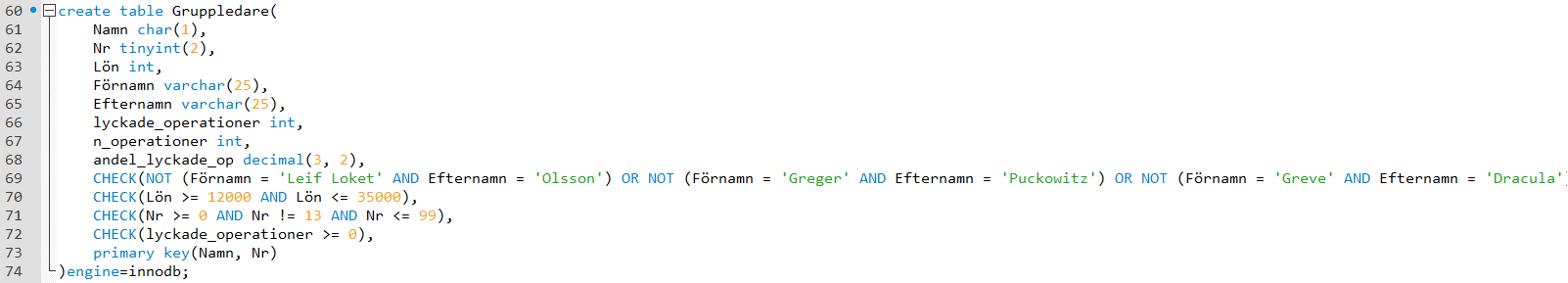
En serie begränsningar i vilken typ av data som kan lagras i databasen infördes. Den ena gruppen begräsningar programmerades med check-villkor och den andra dynamiska gruppen skapades med s.k. ”triggers”.

**6.1 Urval av checkvillkoren**



Figur 12. Checkvillkor i hjälpmedelstyper som gör att endast ordningsnummer mellan 1-15 kan lagras.

I gruppledare infördes flera checkvillkor bland annat med avseende på för- och efternamn. Vidare skapades villkor för minimi- och maximilön, vilka nummer som kunde användas samt att andelen lyckade operationer måste vara minst 0. Liknande begränsningar infördes i tabellen ”Fältagent”.



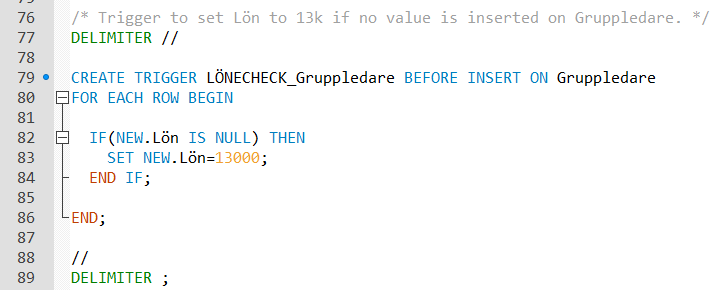
Figur 13. Checkvillkor i tabellen "Gruppledare".

7 Triggers

**6.1 Urval av triggers**

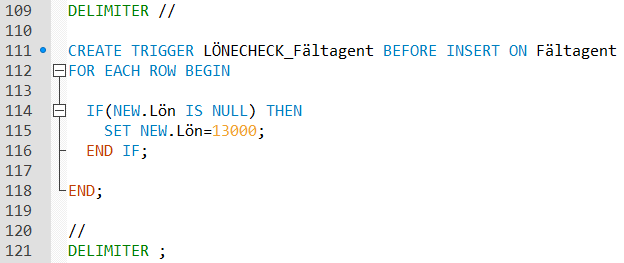
En serie triggers skrevs för att skapa möjlighet att ändra databasen baserat på interaktion med användarna. I de flesta fall baseras villkoret på en insert i en tupel i någon av tabellerna.

Den första trigger som skevs ”LÖNECHECK\_Gruppledare” aktiveras vid en insättning i tabellen Gruppledare. I det fall att cellen ”Lön” för en viss tupel lämnas som NULL skall i stället Lön = 13 000 sättas in.



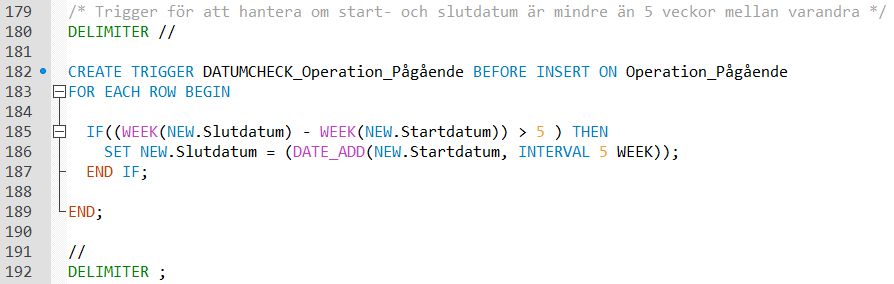
Figur 14. Trigger för att hantera kolumnen "Lön".

På samma sätt som LÖNECHECK\_Gruppledare skrevs en trigger för Fältagent-tabellen. Både Fältagent och Gruppledare får alltså en lön på 13 000 kr om inget värde väljs för attributet.



Figur 15. Även Fältagent bevakas av en lönetrigger.

En trigger skrevs också för att fullfölja kravet att intervallet mellan start- och slutdatum skall vara 5 veckor om intervallet sätts till mer än 5 veckor. Detta gjordes för såväl pågående operationer (figur 15) som avslutade operationer.

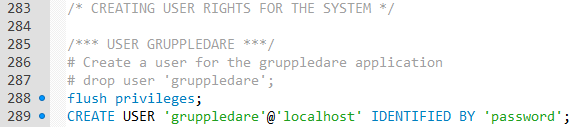


Figur 16. trigger som bevakar att ett korrekt intervall av start- och slutdatum sätts för en operation.

8 Vyer & rättigheter

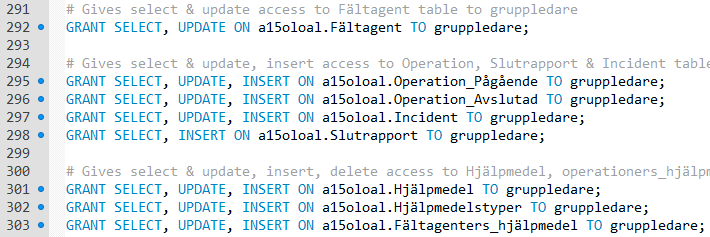
I databassystemet är det viktigt att det finns specifika regler för vad de två användarna – fältagent och gruppledare kan se och modifiera. Genom att kontrollera rättigheter och skapa vyer kan databasadministratören värna om databassystemets konfidentialitet och integritet, två av de viktigaste egenskaperna för det som utgör informationssäkerhet.

Den första användare som skapas är den för gruppledarna.

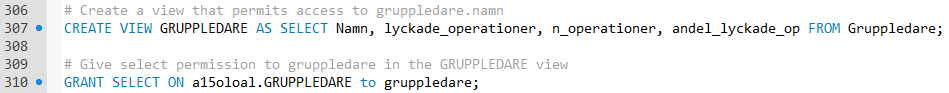


Figur 17. Den första användaren som skapades, "gruppledare".

Ett urval av de rättigheter som gavs till ”gruppledare”-användaren som skapades i figur 17.



För gruppledare-rollen skapades avslutningsvis en vy som innehåller andra gruppledares namn samt härledda attribut. Genom att skapa en vy med informationen finns det ingen risk att gruppledar-användarna kan hacka systemet och komma åt illegitim information.



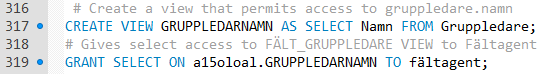
Figur 18. En vy skapades för att lagra delar av tabellens information och på så sätt skyddar den hemliga infon.

Den andra rollen som skapades var för fältagenterna.



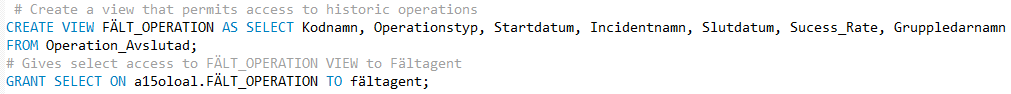
Figur 19. Skapande av rollen "fältagent".

För Fältagent-rollen skapades mer utförliga restriktioner eftersom de har betydligt färre rättigheter än gruppledarna. På samma sätt som för gruppledare skapades en vy som innehåller gruppledarnamn. Detta stärker som tidigare nämnt konfidentialiteten i systemet.



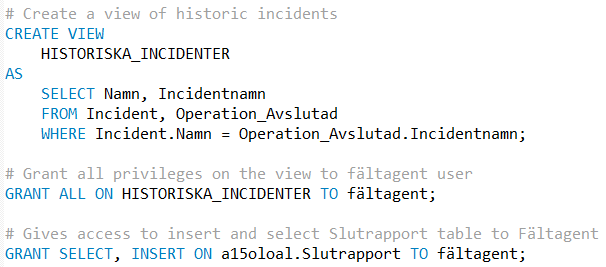
Figur 20. Fältagenterna kan se gruppledarnas namn genom en vy.

Vidare kan nämnas att fältagenterna skall kunna se (dvs SELECT) information om historiska operationer som har avslutats.



Figur 21. Vy som ger fältagenterna tillgång till avslutade operationer.

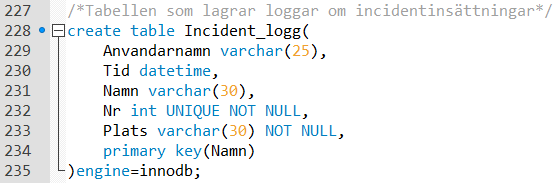
Avslutningsvis kan fältagent-användaren se information om incidenter – men endast avslutade incidenter. För att uppnå detta skapades ett villkor som säger att incidentnamnet ska finnas både i ”Incident”-tabellen samt ”Operationer\_Avslutad”-tabellen. På detta sätt väljs endast historiska incidenter.



Figur 22. Fältagenterna har även rätt att se historiska incidenter samt se och göra insättningar i tabellen "Slutrapporter".

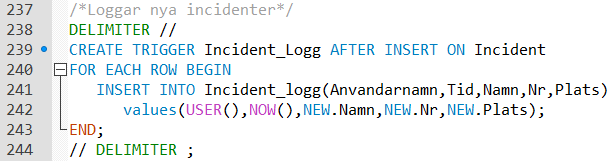
9 Loggtabell

För att främja databasadministratörens arbete skrevs en loggtabell som lagrar information om alla insättningar i Incident-tabellen. Det här ansågs vara viktig och känslig information vilken flera andra tabeller baseras på. Således gjorde tabellen ”Incident\_logg” som lagrar info om de användare och det tidslag då nya incidenter lagrats i databasen.



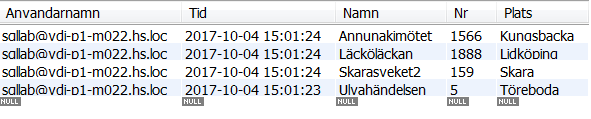
Figur 23. Tabell som lagrar loggar om vem som sätter in ny data i Incident-tabellen.

För att automatiskt ”logga” nya insättningar i Incident-tabellen skrevs en trigger som aktiveras i händelse av att en ny insättning görs i Incident. För att lagra användarnamnet på den person som utförde insättningen används funktionen ”USER()”. Vidare lagras tiden för händelsen med funktionen ”NOW()”.



Figur 24. Trigger som automatiskt stoppar in information i Incident\_logg-tabellen i händelse av att en "insert" görs i Incident.

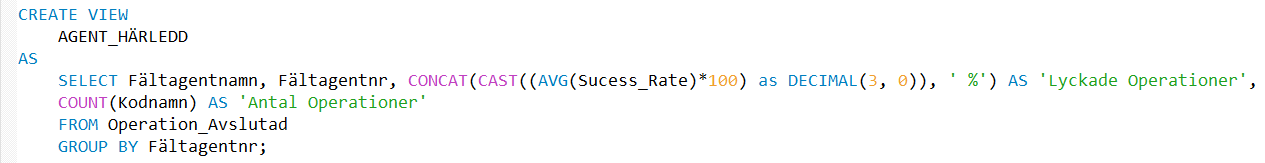
En exempelbild på hur loggtabellen kan se ut efter att ett antal insättningar kan beskådas i figur 25.



Figur 25. Ett exempel på hur en "select" kan se ut i loggtabellen som skapades.

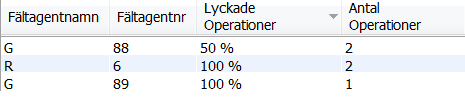
10 Hantering av härledda attribut

De härledda attributen i Fältagent-tabellen hanterades genom att skapa en vy som gjorde dynamiska uträkningar för antal utförda operationer samt andel lyckade operationer.



Figur 26. Vy för hantering av härledda attribut.

Resultatet av den vy som skapades med namnet ”AGENT\_HÄRLEDD” kan beskådas nedan.

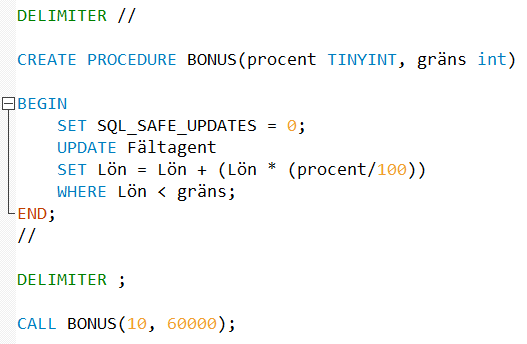


Figur 27. Exempel på hur en select av vyn kan se ut.

11 Procedurer

Procedurer fungerar som en funktion i procedurella- eller som en metod i objektorienterade språk. I SQL är de precis som de triggers som tidigare beskrivs, med undantaget att de inte aktiveras vid en särskild händelse utan när användaren själv kallar på dem. Vidare kan de ta ett antal parametrar i vilka man kan stoppa in argument och få en specialanpassad beräkning.

En procedur vid namn ”BONUS” skapades för att snabbt kunna höja agenternas lön med villkoret att deras nuvarande lön skall vara under det instoppade värdet. I exemplet används procentsatsen 10 % - dvs lönerna skall höjas med detta värde om deras nuvarande lön är under 60 000 kr.



Figur 28. Procedur för att höja alla agenters löner under ett villkor.

Källor

Byham, R., Huss, L., Hamilton, B., Guyer, C. 2017. Int, bigint, smallint, and tinyint (Transact-SQL). Hittad: <https://docs.microsoft.com/en-us/sql/t-sql/data-types/int-bigint-smallint-and-tinyint-transact-sql> [2017-09-26]

Oracle Corporation. 2017. Date and Time Types. Hittad: <https://dev.mysql.com/doc/refman/5.7/en/date-and-time-types.html> [2017-06-26]