Självständigt arbete på grundnivå

Independent degree project - first cycle

Datateknik
Computer Engineering

Scrambling av databaser
Validering och implementering av scrambling av databas

Fredrik Öberg

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

Kraven hur personlig data hanteras har på senare tid blivit mycket mer strikta med nya förordningar som GDPR. Vilket betyder att företag måste se över hur dom spara och hanterar data. Vidare så finns det en hel bransch som jobbar med att analysera och anonymisera databaser för att skapa testdata för företag att använda för tester. Hur kan dessa företag garantera att de kan lämna över deras databas för just detta. Easit AB vill att ett system ska byggas för att scrambla databaser så att struktur och data i databasen är oigenkännligt som sedan kan lämnas över till Easit för analysering. Med Huvudmålet att med hjälp av befintlig funktionalitet i Easit Test Engine ETE kunna scrambla kunders databaser och data till oigenkännlighet så att överlämning av databasen kan ske utan risk för tester. Men även att validera de scramblingmetoder som lösningen innehåller.

Nyckelord: Testdata, Databas, Dcrambling, Java, db2

Abstract

The demands on how personal data is handled have recently become much more strict with new regulations such as GDPR. Which means companies need to review how they save and manage data. Furthermore, there is a whole industry that works with analyzing and anonymizing databases to create testdata for companies to use for tests. How can these companies guarantee that they can hand over their database for this particular purpose. Easit AB wants a system to be built for scrambling databases so that the structure and data in the database are unrecognizable, which can then be submitted to Easit for analysis.With the main objective, using existing functionality in the Easit Test Engine ETE, see if you can scramble customers databases and data to unrecognizable so that the handover of the database can be done without risk. But also to validate the scrambling methods that the solution contains.

Keywords: Testdata, Databases, Scrambling, Java, db2

Förord

Jag vill tacka studentföreningen Mytec och Reza Moossavi som gav mig möjligheten att få kontakt med Easit AB för att göra examensarbete. Och ett större tack till Robert och hans teamet som utvecklar Easit Test Engine för att vara där när hjälp när det behövdes.

Innehållsförteckning

Terminologi

Förkortningar

JDCB	Java Database Connectivity
API	Application Programming Interface
JAVA	SE Java Standard Edition
ETE	Easit Test Engine
SQL	Structured Query Language
IBM	International Business Machines Corporation

1 Inledning

Vi lever i ett samhälle där människan cirkulerar runt digitala enheter såsom smartphones, plattor, smartklockor och mycket mera. Alla enheter är på något sätt ihopkopplade med varandra. Tack vare detta har många nya applikationer skapats där majoriteten är beroende av data från användaren för att fungera. Hur förlita sig individen på företag att deras data hanteras på rätt sätt? Lagar och regler på hur företag ska sköta detta finns såsom dataskyddsförordningen (GDPR) (Datainspektionen, 2019) [1] som verkställdes 2018 av EU som skapar riktlinjer för hur företag hanterar persondata.

Vidare så har företag många anledningar till att hålla ordning på sin data och hur den datan är sparat. Det kan vara lagar som gör att man måste spara datan på ett visst sätt som GDPR. Man kan ha en speciell struktur på databaserna. Den data som databasen innehåller kan vara känslig eller helt enkelt inte vill att folk utanför företaget ska se datan i tabellerna och deras struktur.

1.1 Bakgrund och problemmotivering

Utveckling av Data Management-funktionalitet förekommer för att kunna analysera och presentera data utifrån en stor mängd data. För att göra detta kräver man tillgång till databaser och deras datamodeller av olika form såsom storlek och innehåll. Det finns i dagsläget publika databaser som går att ladda ner för att använda för detta ändamål men antalet är begränsat.

För att göra det enklare att analysera databaser så skulle det underlätta om man som företag skulle kunna arbeta på flera databaser än de publika som finns att använda. Detta skulle underlätta om en programvara som kan ta databaser och scrambla dem till oigenkännlighet så att databasen utan risk kan lämnas över för analys.

Easit AB vill då utvärdera vilka möjligheter det finns för att utveckla funktionalitet med hjälp av de befintliga metoderna för att hantera testdata och avidentifiering med deras utvecklade produkt ETE för att scrambla deras kunders databaser och deras data till oigenkännlighet så att företaget utan risk kan lämna över den data till Easit för tester.

1.2 Om Easit AB

Easit AB grundades 1999 av två entreprenörer Ola Holm och Åke Lindblom. Med en vision om att skapa lättanvända och flexibla verktyg. Easit fokuserar på tre huvudområden Ärendehantering, Testdata och Konsult. Easit utvecklar Webbaserade lösningar med billig drift och införanden där kunderna är Myndigheter, Landsting och Kommuner. Företaget har två kontor ett i Sundsvall och ett i Stockholm.

1.3 Övergripande syfte

Huvudmålet och Syftet med projektet är att med hjälp av befintlig funktionalitet i Easit Test Engine ETE kunna scrambla kunders databaser och data till oigenkännlighet så att överlämning av databasen kan ske utan risk för tester. Men även att validera de scramblingmetoder som lösningen innehåller.

1.4 Avgränsningar

Projektet är avgränsad till att bygga ett verktyg skapad med java och med hjälp av ETE befintliga sätt att arbeta på databaser. Finna ett sätt att scrambla en DB2 databas baserat på lite data ifrån användaren. Beroende på hur känslig databasen är ska man scrambla databasmodellen eller enbart data i databasen. Men även kunna applicera ett filter där man kan specificera vilka fält som inte ska scramblas. Möjligheten att tanka ner denna scramblade databas och installera den hos Easit på ett lätt och smidigt sätt.

Verktyget kommer även att begränsas genom att använda sig av spring boot som skapar lätta ensamstående applikationer som körs på ett lätt och smidigt sätt.

1.5 Konkreta och verifierbara mål

Målet med projektet är att med hjälp av befintliga sättet att hantera databaserna i ETE så ska en spring boot applikation skapas för att scrambla en databas så att dess innehåll inte går att identifiera efter scramblingen men även datamodellen den är uppbyggd på. Projektet har fyra huvudmål.

- *Scrambla den data som databasen innehåller.*
- *Scrambla strukturen på databasmodelen.*
- *Skapa ett enkelt installationspaket.*
- *Applicera ett filter på kolumner som ej ska scramblas.*

Dessa fyra huvudmål beskrivs utförligt i kravspecifikation (se Kapitel 4 Metod).

Utifrån dessa målen ska en validering på de scramblingmetoder som har blivit vald genomföras. Genom att göra en jämförelse med alla metoderna ska frågeställningen när vet jag att scramblingen är tillräcklig? besvaras efter färdig konstruktion.

1.6 Översikt

Kapitel 2 tar upp teorin som projektet är baserat på för att kunna förstå rapporten. Där efter kommer kapitel 3 där vilka metoder har använts för att lyckats åstadkomma detta projekt. Kapitel 4 tar upp konstruktionen av den lösning som har skapats och hur den fungerar. Efter detta kapitel kommer kapitel 5 som presenterar alla resultat som framställts av projektet. Till sist kommer kapitel 6 där utvärdering av resultatet framställs. Efter detta kommer även en källförteckning där rapporten hänvisar läsaren för mer information om olika ämnen.

1.7 Författarens bidrag

Alla koder som tillhör utformningen hur scramblingen på databasen är skapad av författaren själv men även hur installationspaketet har blivit framtagen. Den kod för att kunna prata med databaserna på de viss som görs är taget ifrån Easit egenutvecklade vara ETE. Sedan har Java SE8 funktionaliteter används.

2 Teori

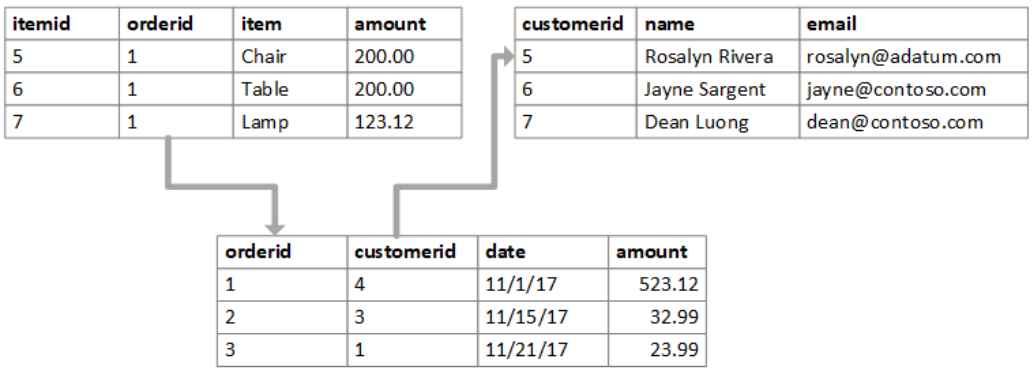
Detta kapitel beskriver de teoretiska elementen som arbetet grundar sig i för att ge läsaren den bakgrund för att kunna förstå hur och vad de kommande kapitlen grundar sig i. Här kommer programvaran ETE, teori hur databashantering hanteras i java, och övrig nödvändig tekniker förklaras

2.1 Databas

Databaser är ett sätt att samla och strukturera stora mängder data. För att på ett så effektivt sätt som möjligt kunna hämta och bevara data. Det finns många olika typer av databaser men detta projekt kommer jobba med relationsdatabaser som kommer att förklaras i kapitel 2.1.1.

2.1.1 Relationsdatabas

Denna typ av databas samlar stor mängd data som redan har existerande relationer mellan varandra. Denna stora mängden data blir organiserad med hjälp av en mängd av olika tabeller där kolumner och rader kommer att representera olika typer av objekt i databasen. Varje kolumn innehåller en typ av data och ett fält där det faktiska värdet sparas. Dessa kolumner tillsammans formar tabellen. Raderna i tabellen visar en gemensam samling av värden som tillhör ett objekt eller insättning i tabellen. Detta kan man se i figur 1. Där till exempel tabellen med itemid, orderid, item och amount där är item och amount fält där data är sparad. Varje rad kan också ha en unik identifierare som kallas nycklar. Och ifrån tabellen som nämndes innan så är itemid, orderid just detta. Vad dessa är här för att göra är att genom att skapa dessa nycklar så kommer man kunna skapa relationer till andra tabeller detta är de gråa pilarna som illustreras i figur 1. Vidare så finns även främmande nycklar [2] och andra nycklar att ta hänsyn till. [3]



Figur1: Illustration av hur en enkel Relationsdatabas kan se ut.

2.1.2 SQL Structured Query Language

Structured Query Language (SQL) är ett standardiserat sätt att kommunicera med databaser. Här skapas SQL statement som används för att utföra operationer på databasen. Ett par exempel på operationer som går att gör är att uppdatera data i den, hämta data. Figur 2 visar ett lätt statements. Några vanliga databashanteringsverktyg som använder SQL är IBM db2, Oracle, Sybase, Microsoft SQL Server, Access. Även fast alla dessa databashanteringsverktyg stöder SQL så har dom sina egna unika sätt också Men SQL standard kommandon består utav är Select, Insert, Update, Delete, Create, and Drop med dessa kan man utföra majoriteten av alla uppgifter man kan komma till at göra med en databas. [4]

```
SELECT country_id, country_name
FROM countries
WHERE region_id=1
ORDER BY country_name;
```

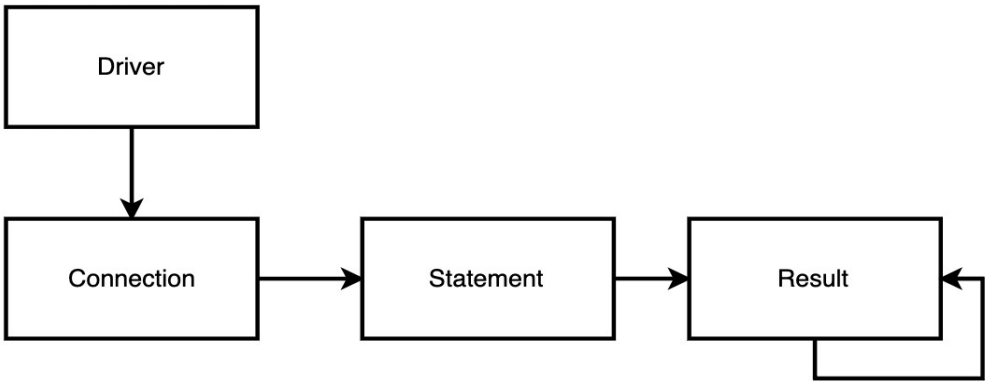
Figur:2 Ett enkelt Sql-statement.

2.1.3 DB2

International Business Machines Corporation (IBM) är skaparen av db2 som är ett databashanteringsverktyg designat för att hjälpa att hantera strukturerade och ostrukturerade data [5]. DB2 är baserat på SQL man har sedan byggt ut DB2 så att det finns stöd för Objektorienterade funktioner och icke-relationella strukturer med XML. Ursprungligen tillverkades db2 för specifika maskiner men sedan 1990 bestämde IBM att dom skulle tillverka ett universellt sätt att använda db2 på. [5]

2.1.4 JDCB

JDBC är ett API för att hantera förfrågningar till databaser genom java där JDBC är en del java Standard Edition (java SE). Detta system är en uppsättning av några interfaces och klasar skrivna i java. Användning av detta system gör att du kan koppla upp dig på databaser, skicka förfrågningar skrivna i SQL, och hantera det resultat som du får tillbaka. Och eftersom JDBC ingår i java SE kan ett javaprogram alltid koppla upp sig på ett databashanterings-verktyg så länge drivrutinerna för det databashanteringsverktyg finns tillgängligt. Vidare så visar figur 3 hur processen JDBC utför för att hämta data ifrån en databas. [6]



Figur4: Processen som java JDBC utför.

2.2 Java Spring Boot

Java Spring Boot är ett java-baserat ramverk som används för att skapa Micro Service applikationer. Systemet är skapad av Pivotal Team och gör det lätt att skapa ensamstående och produktions-redo applikationer. [7] Detta ramverk är baserat på java och ett antal tredjeparts-ramverk som gör det lätt att komma igång med ditt projekt. Anledningen till att ramverket är populärt är för den lilla konfigurationen som krävs och väl när konfiguration ska utföras gör man detta med hjälp av konfigurationsfiler.[8]

2.3 Easit Test Engine

För att kunna utföra test med testdata kräver det att man på ett säkert sätt vet att den data man använder i testerna är tillåtna att användas enligt lagar och regler såsom GDPR och andra förordningar som finns. Detta är en komplicerad process som inte är helt enkelt att lyckas med. Följder av att inte använda riktiga testdata i sina tester utan istället använder produktionsdata kan leda till stora kostnader och att man riskerar att vara oförenligt med existerande persondatalagstiftning. Easit Test Engine (ETE) är ett databashanteringsvektyg som är utvecklat för att jobbar med avidentifiering av testdata för att kunna på ett säkert och smidigt sätt göra testdata förenlig med lagar och regler så att den går att använda i tester.[9]

2.3.1 Testdata

För att testa att ett system fungerar som det ska så behöver man i många fall som användare skriva in data i systemet och all form av inskrivna data i systemet är klassificerat som testdata. Denna testdata kan skrivas in manuellt när exekvering av programmet sker men även i form av automatiserade tester som skriver ifrån XML, databaser, textfiler av automatiserade verktyg. Två olika fall av resultat kan man använda testdata till för analysera Man kan förvänta sig testet att lyckas eller inte lyckas. Man kan generera denna testdata själv eller använda sig av befintliga verktyg.[10]

2.3.2 Avidentifiering

Avidentifiering är en teknik för att göra personlig information anonym. Vad som klassificerar att det är personlig data är alla uppgifter som på något viss kan knytas till en viss person.[11]

2.4 Generering av nummer

Att generera en sekvens av nummer och skapa en Random Number Generator (RNG) där numren inte går att förutse med bättre förutsättningar än med slumpmässig sannolikhet. Dessa RNG kan vara hardware random-number generators (HRNG) eller Pseudo Random Number Generator (PRNG) men även cryptographically secure pseudo-random number generator (CSPRNG) dessa förklaras i 2.3.1–2.3.3

2.4.1 Generering av nummer med hårdvara

HRNG är en metod för att generera verkliga slumpmässiga tal från en fysiks process istället för att använda sig av matematiska algoritmer. Denna enhet kopplas in i datorn. Huvudsakliga användningsområden dessa enheter används i är inom kryptografin för generering av nycklar och lotteri-maskiner. [12]

2.4.2 Pseudorandom nummer

För att åstadkomma ett sätt att generera slumpmässiga nummer så finns det Pseudo Random Number Generator (PRNG) som grundar sig i Moduler aritmetik. En sekvens av slumpmässiga tall producera med hjälp av algoritmer som approximativt kommer att ha egenskaperna hos slumpmässiga tal. Detta är möjligt tack vare att talen baseras på ett frö som är startpunkten för de genererade talen. Detta är en snabb metod som genererar många tall på kort tid men men denna sekvens kan bli om genererad ifall man skulle veta startpunkten. [13]

2.4.3 Starka slumpmässiga nummer generatorer

cryptographically secure pseudo-random number generator (CSPRNG) är en PRNG som följer de egenskaper som krävs för att göra den användbar inom kryptografi. Inom kryptografin används detta för att generera nycklar. Det ska inte gå att generera om efter att man ha genererat den en gång. Den måste upplevas som slumpmässiga. Och man kan inte på förväg veta vad den kommer att generera.[14]

2.5 Java generering av slumpmässiga strängar

I kapitel 2.5.1 till 2.5.5 kommer att presentera olika metoder som har hittas För att kunna generera slumpmässiga strängar. Alla metoder går att använda i java eller så är det ett tredjeparts-bibliotek bygd i java. Javas grundläggande plattform Java SE (Standard Edition). Är kärnan till all java kod och innehåller den grundläggande bibliotek och API som javaprogrammerare ska kunna handera. Några exempel är java.lang, java.io, java.math, java.net, java.util.

2.5.1 Apache Commons Lang

Java har ett stort utbud av paket som bygger vidare på java SE för att man ska kunna utnyttja javas grunda klasser. En av dessa paket är Apache Commons Lang so förser java med lite extra funktioner som kan vara till stor hjälp. Vad paketet innehåller allt ifrån sträng, Array och nummer manipulation av sina grundklasser till numeriska metoder, samtidighet och förbättringar när det kommer till java.util.Date och många hjälpfunktioner som hashCode, toString and equals och några datastrukturer som pairs och triples. [15]

2.5.2 Math.random klassen

Hur vanlig java kan användas för att generera strängar kan göras med hjälp av javas Math.random klass denna klass tillhör grundläggande bibliotek i Java SE som är en funktion som returnerar ett pseudo genererat flyttal mellan $0 \leq x < 1$ där x är det genererade talet där fördelningen är ungefär normalfördelat. Med detta kan man man skala det till den mängd som man skulle vilja begära.[16]

2.5.3 Charset

Charset använder också vanlig java för att generera strängar genom Teckenkodning vilket är ett sätt presentera en mängd karaktärer. Denna mängd kan vara alfabetet. Vidare så kan detta kodas på olika sätt det kan till exempel vara elektriska stötar, bit-mönster som oktäter eller naturliga tal.[17]

2.5.4 Reguljära uttryck

Reguljära uttryck är att begrepp inom datavetenskapen som beskriver hur man kan visa olika mängder av strängar. Dessa utryck är uppbyggda av strängar som följer en viss syntaxregel. Olika användningsområden för Reguljära uttryck är exteditorer och programspråk för sökning och textmanipulation. Programmerare använder denna metod för att söka och filtrera ut onödiga data så som att finna alla mailadresser i en text.[18]

2.6 Java generering av slumpmässiga tal

Tidigare hittades tre olika sätt att generera slumpmässiga tal. Java kan producra två av dessa sätt är pseudorandom och Starka slumpmässiga nummergereratorer den tredje hardware metoden vilket är en fysisk enhet vilket ej går att applicera så bra. Nedan följer några metoder som finns i java för att genera tal.

2.6.1 Java math class

Javas math klass innehåller de grundläggande matematiska operationerna så som logaritmer, kvadratroten, trigonometriska funktionerna med mera. I jämfört med andra klasser som StrictMath så är uppbyggnaden av Math annorlunda på det viss att den inte är definierad att returnera bit för bit vilket medför bättre prestanda. [10]

2.6.2 Java Random Class

Java random klassen används för att generera en ström av slumpmässiga tal är pseudorandom. Klassen använder sig av en 48-bit frö och är modifierad med en linear congruential formula och kan leverera upp till en 32 bitars tal som är pseudorandom. I java 8 presenterades en ny metod ints() som kan ge en oändlig ström av pseudorandom tal där de går att välja vilka mellan vilka tal som den ska generera mellan .[19]

2.6.3 Java SecureRandom

Denna klass är skapad för att nå de krav en CSPRNG för att ett slumpmässigt tal ska vara tillräckligt för kryptering. Denna klass klara testerna som måste klaras för att få vara tillåten och den måste producera en sekvens av nummer som inte är förutsägbara. [20]

2.7 Relaterad arbete

Det finns mycket arbete inom kryptera eller dölja personlig data på något viss. I detta kapitel kommer det andra företag som också har en scramblingmetod men även lite forskning och patentsökningar.

2.7.1 DATPROF

Under den studie om vad som redan finns på marknaden som att scramblar databaser så undersöktes vissa konkurrenter till Easit ab en av dem var DATPROF. Som är ett företag som jobbar med databashanterings-verktyg. Deras implementering av scambling av databaser grundar sig i en väldigt simpel metod genom att byta ut alla karaktärer till ”x” och alla nummer till ”1” [21]

2.7.2 Patent

Arbetsmarknaden för att arbeta med datamaskning är hård och det innehåller mycket pengar. De flesta företagen vill hålla deras metoder hemliga eller att ingen annan ska kunna använda dem så detta är nog den stora anledningen till att det sker så mkt patentering inom detta område för om man skulle ha patent på en metod som skulle vara bra för något inom maskning av data så är pengar en stor del av varför man skulle vilja ha patent men även för att se till att just denna metod är begränsad så att kanske bara företaget som har patenten använder den.

Så när man söker information om just scrambling och avmaskning av databaser så kommer mång patentsökningar upp till exempel [22] som beskriver ett system för att utföra datamaskning på databaser. Men även [23] som är ett patent på en metod för att kryptera kolumner i en databas.

3 Metod

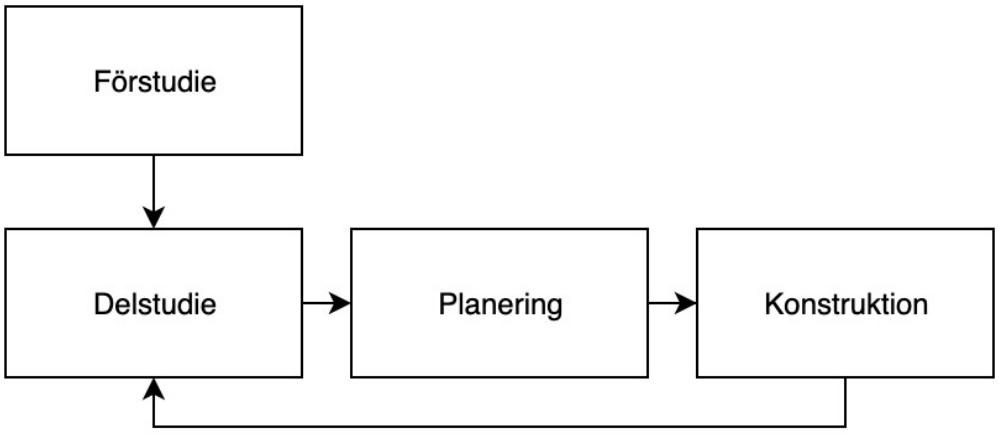
Detta kapitel beskriver de metoder som används för att uppfylla de Konkreta och verifierbara mål som har beskrivits i kapitel 1.5. Genom att först förklara hur arbetsgången kommer att utföras under den period projektet har blivit genomförd. Sedan kommer vilka verktyg som ska används för att nå målen. Sist beskrivs hur man ska göra för att välja vilken av de scramblingmetoder som ska lösa problemet ska bli vald men även hur validering av dessa metoder kommer att utföras.

3.1 Förstudie

Denna studie kommer att fokusera på att få den kunskap som krävs för att kunna lösa de formulerade problemen. Där en stor del är att bli bekväm med ETE och hur detta databashanterings-verktyg används men även den kod som har har erhållits ifrån Easit för att sedan få en bra överblick över vad problemet grundar sig i. Denna studie innehåller även att få en förståelse på hur man kan scrambla olika typer av data en databas kan innehålla.

3.2 Arbetsgång

Projektets arbetsgång baseras på att den förstudien som har ägt rum. Sedan för att finna en lösning till de fyra konkreta verifierbara målen Och kommer under projektets gång genomgå tre olika faser vardera. Figur5 visar helheten hur arbetet kommer att utföras och vilka tre faser som målen kommer att genomgå.



Figur5: Illustration av den iterativa arbetsgång för de mål som ska uppfyllas.

3.2.1 Delstudie

För varje verifierbart mål så kommer en förstudie att utföras. I denna kommer information om vad som ska göras och hur det ska göras undersökas. Genom detta så ska man åstadkomma en så bra överblick över problemet som möjligt innan man börjar planera eller implementera sin lösning.

3.2.2 Planering

Efter att ha skaffat en bra överblick över problemet så planeras lösningen. Detta gör man att åstadkomma ett så bra resultat som möjligt på en gång. Detta utförs genom att ta de målet som ska lösas och bryta ner det problemet till mindre bitar som sedan kommer konstrueras med hjälp av en iterativ process.

3.2.3 Konstruktion

Efter att ha åstadkommit en bra förståelse och man har ett problem som är nedbrutet i mindre problem återstår konstruktionen. Dessa mindre problem som har blivit konstruerade kommer nu bli till verklighet och programmeras. Efter denna konstruktion kommer det finnas kod som är i ett stadie så att det fungerar. Detta händer då iterativt mellan dessa fyra konkreta verifierbara målen.

3.3 Verktyg

I kapitel 3.3.1 till 3.3.3 presenteras det olika verktygen som kommer att användas under projektet för att konstruera systemet.

3.3.1 Programmeringsspråk

De verktyg som kommer att byggas kommer vara utvecklad med java och deras implementering av hantering av databaser JDBC. Vidare så kommer databasspråket SQL användas då den databas som kommer användas är en IBM DB2 databas.

3.3.2 Utvecklingsmiljöer

Den kod som kommer att skapas kommer vara skriven i IntelliJ IDEA som då är utvecklarnas standard på företaget. Vilket hjälper detta projekt ifall några frågor och problem skulle uppstå under projektets gång. För att testa det system som har utvecklads så jobbar koden emot ETE som är Easit databashanterings-verktyg detta för att se hur källan och måldatabasen ändras efter körning

3.3.3 Illustrationer

Det verktyg som har används förr att illustrera relationer mellan de skapade klasserna och hur dom arbetar tillsammans är draw.io. Detta verktyget har även används för att förklara processer som händer under körningen av programmet och andra relationer.

3.4 Val av scramblingmetoder

Hur dessa scramblingmetoder kommer att väljas kommer att baseras på två delar den första som kommer vara en studie för att kunna besvara vad som är bra och dåligt med algoritmen i syfte med detta projekt

Den andra delen är en serie av mätningar på scramblingmetoderna för att testa deras hastighet. Här kommer varje algoritm utsättas för 4 körningar där de algoritmer som ska generera strängar kommer att generera strängar på längd 10 och där ska detta hända 5 gånger vardera förr n 100 000, 1000000, 10000000, 100000000 där sedan en standardavvikelse kommer att beräknas. samma förutsättningar kommer att appliceras på generering av tal men där kommer också den generera slumpmässiga tal mellan 0 till 1000. Sedan sammanställs detta i tabeller och sedan utvärderas.

3.5 Validering av metod

Efter att dessa frågeställningarna har blivit besvarade kommer en bedömning ske utifrån ett teknisk perspektiv av implementationer av de olika metoderna . Bedömningen kommer visa på vilken metod som är mest lämpad för att nå de mål som har blivit satta på bästa sätt. Tillsammans med dessa mätningar på snabbheten hos de algoritmer Dessa mätningar kommer att generaliseras emot hur länge de skulle ta att scramla en hel databas för att se om metoderna har stor påverkan på resultatet.

4 Konstruktion

I detta kapitel får läsaren en förståelse av hur den tekniska delen i projektet är utvecklad. Efter inledningen följer det 4.1 där en kravspecifikation beskrivs, 4.2 som förklara och beskrivning hur systemet ska fungera följt av 4.2 där kodstruktur och hur programmet är uppbyggnad förklaras, Sedan 4.4 till 4.6 som ger en förklaring på hur de mål som ska uppfyllas är bygd har uppfyllts.

4.1 Kravspecifikation

Kapitel 4.1.1 till 4.1.4 kommer att beskriva vad de olika delarna i projektet är och vad de har för krav utifrån den information som har erhållits av företaget

4.1.1 Scrambla datan

Scrambla själva datan i databasen kommer bli scramblad. Baserat på vad för datatyp och dess storlek som kolumnen är kommer en viss scrambling hända. De datatyper som kommer att stödjas är INTEGER, DATE, DECIMAL. FLOAT. Baserat på att dessa är de vanligaste typerna i en databas. Även ett filter ska konstrueras så att man kan välja vilka kolumner som inte ska bli scramblade.

4.1.2 Scrambla Datamodel

Att scrambla datamodellen innebär att kolumnnamn och nyckelvärdena i tabellen ska bli genererad till slumpade värden. Här ska stöd för ett antal DDL-kommandon av typen ALTER, CREATE, DELETE TABLE på grund av primärnycklar måste tabellen droppas och skapas om med det nya namnet. Men detta ska ej vara obligatoriskt att köra denna typen av scrambling om kundens datamodell inte betraktas som känslig. Ett filter ska gå kunna appliceras för att undanta vissa kolumner ej ska scramblas.

Användning

Under hela projektet ska en stäva efter lite data ifrån användaren ska ges vid körning och baserat på målen så har en lista på möjliga parametrar tagits fram.

- *Ska modellen vara scrambland?*
- *Finns det ett filter att applicera?*
- *Information att komma åt databasen?*

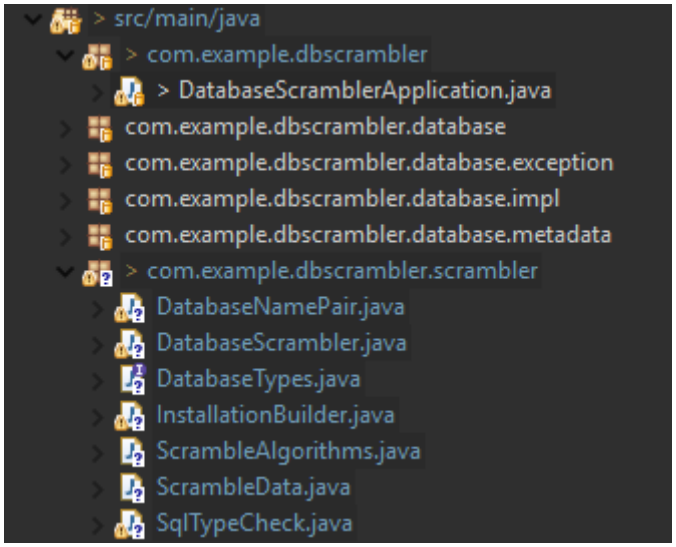
Detta system kommer fungera som ett steg för steg metod där användaren blir tillfrågastäld vad för information som man vill ha.

4.1.3 Installation

Efter konstruktion ska scramblingen ha ett enkelt sätt att installera paketet. Efter genomkörning så kommer företaget leverera ett enkelt installationspack av den nya databasen genom ETE och denna är så scramblad och helt okej att lämna över till företaget för analys.

4.2 Scramblingsystem

Följande underkapitlen beskriver designen av varje delsystem i scramblingen av databaser där illustrationer av klasser och relationer hur dessa klasser arbetar med varandra beskrivs. Applikationen är baserat på ett spring boot ramverk och grunden i den konstruerade applikationen är DatabaseScramblerApplication som implementerar ApplicationRunner enligt figur 6 som visar hur Javaprojektets filstruktur och uppbyggd är strukturerad.



Figur6: Filsystem för projektet

huvuduppgiften i DatabaseScramblerApplication är att man ska skapa en uppkoppling till en databaskälla och en koppling till måldatabas. Sedan ska man överföra data ifrån källan till måldatabasen och medans denna process utförs så tillämpas det scramblingmetoder för att scambla den data som kommer att föras in i måldatabasen.

4.2.1 Scrambler

För att kunna scrambla den data som överförs så har ett javapaket skapats. Detta paket innehåller all väsentlig funktionalitet för att scrambla data och bygga upp SQL-filer för att skapa ett installationspaket mer om detta i kapitel 4.2.3 och 4.5. Scramblingen är uppbyggt på ett javainterface DatabaseTypes som kommer stå som en grund för vilka typer som kommer att gå scrambla. Figur 7 två visar vilka typer som detta system kommer att stödja

```
public String varcharScramble(int size);
public String filterVarcharScramble(int size, ArrayList<String> filters);
public String intergerScramble(int min, int max);
public String dateScramble(String date);
public String decimalScramble(String size);
```

Figur7: vilka metoder som DatabaseTypes interface kräver

För att vissa hur hela javapaketet samarbeta so har figur3 skapats för att illustrera relationer av alla klasserna.

4.2.2 Data

Hur data blir scramblade baserar sig som sagt på klasen DataScrambler som bygger på interfacet som nämns tidigare. Denna klass implementerar de scramblingmetoder som kommer att scrambla de olika typerna av data som kan finnas i databasen. Vidare så har klassen DatabaseScrambler skapats som får information om vilken tabell, kolumn och värdet på raderna på ett iterativt sett. Vidare så väljer klassen då vilken metod som ska appliceras baserat på vilken typ av kolumn som ska scamblas.

4.2.3 Modell

Modellen scamblas när uppkomling till källan har skett. Då kommer man generera SQL-filer baserat på informationen i från källan och återskapa databasen fast i SQL-satser. Under denna process kommer genererade tabellnamn ersätta den riktiga namnen och bibehålla den ursprungliga databasstukturen. Denna processen att bugga upp SQL filer hanteras av installationen klassen som implementerar metoder för att bygga upp och överföra SQL-satserna till textfiler

4.2.4 Filter

För att kunna bestäma om kolumner inte ska scramblas så kommer en textfil fyllas med information. Denna information kommer sedan att bli jämförd med de kolumner som finns i databasen medans denna överföring till måldatabasen händer. Genom att jämföra textfilens innehåll med vad de kolumnamn som dyker upp under överföringen så kommer det gå att bestämma om kolumnen ska scramblas eller inte.

4.3 Scramblingsmetoder

En studie om vad java har för olika bibliotek och funktioner som kan generera slumpmässiga nummer och strängar gjordes. Vidare så sammanställdes en samling av metoder som senare kommer att valideras med hjälp av prestandamätningar och information om fördelar och nackdelar med dessa metoderna.

Heltal

Efter att ha gjort en studie av vad för olika metoder för generering av heltal så har tre olika sätt se kapitel 2.4.1 till 2.4.3. Dessa tre studerades och jämfördes med varandra för att kunna hitta vilken som var bäst för detta projektets syfte. vidare så undersöktes java för att få en förståelse på vad för verktyg som finns tillgängligt för att generera dessa tal. I resultatet presenteras val av metod och varför den metoden användes.

Varchar

Efter att ha gjort en studie av vad för olika metoder i java för att generera Strängar så har fyra olika metoder hittas se kapitel2.5.1 till 2.5.4. Dessa metoder bygger på helt vanlig java, bibliotek som går att infoga i dit projekt men även charsets och genom att låta slumpmässiga tal genereras som stämmer överens med det valda charsetet kommer man kunna på ett slumpmässigt sätt generera en sträng. Även kombinationer av dessa metoder går att konstruera. Med hjälp av det prestandatest och analys av javakoden så kommer en jämförelse genomföras och presenteras i resultatet.

4.3.1 Algoritmer heltal

- Algoritm 1 och algoritm 2 är baserat på java math där färsta returnera ett flyttal och den andra ett heltal där länden av vilka tal som ska genereras går att bestämma med parametrar.

- Algoritm 3 använder sig av random() klassen i java och retunerar den nästkommande talet i sekvensen av de genererade talen.

- Algoritm 4 denna grundar sig också i random klassen men använder sig av metoden ints() som returnerar en ström av pseudogenerarade tal.

- Algoritm 5 denna är baserad på den kryptosäkra klassen securerandom som uppfyller de krav som förväntas för att vara godkänd för att använda inom kryptografi för generering av nycklar.

4.3.2 Algoritmer varchar

- Algoritm 1 är helt vanlig java där det numeriska representation av bokstäver mellan a till z. Med hjälp av random-klassen slumpas bokstäver mellan a till z.

- Algoritm 2 allokerar an array av den storlek som ska genereras och med hjälp av charsetet ”UTF-8” slumpas det fram en karaktär i hela charsetet med random.next.byte() .

- Algoritm 3 är baserat på att man skapar sim egen mängd genom att göra en string med alla karaktärer som man önskar att ha med i genereringen sedan använder den math.random() för att slumpa fram vilken av bokstäverna.

- Algoritm 4 väljer utefrån ”UTF-8” men tar bara alfabetet som man kan specificera själ.

- Algoritm 5 använder sig av Regular expression fär att specificera den mängd som man ska välja mellan.Algoritm

- 6,7,8 är baserat på apache commons som bygger vidare på java och ser mycket mer elegant ur när man skriver och använder funktionerna. Genom att ha parametrar som ger en val att ha numeriska eller alfabetiska eller båda när man genererar

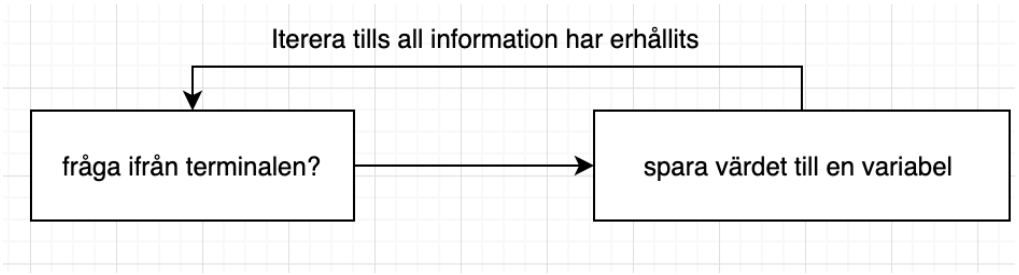
4.3.3 Decimaltal och datum

Studien gav även metoder för hur decimaltal genereras och samma princip som för generering av heltal som nämndes i kapitel 4.3.1 kommer att använda för att besluta om hur decimaltal ska genereras.

Vidare har vi datum och Eftersom ett datum är ursprungligen går att dela upp i tre olika tal så kommer datum separat bygga vidare på hur typen heltal genereras. För eftersom att genom att dela upp datumet går det att applicera den metod som man använde för att generera heltal på dessa tre delar och sedan bygga upp dem igen.

4.4 Användarvänlighet

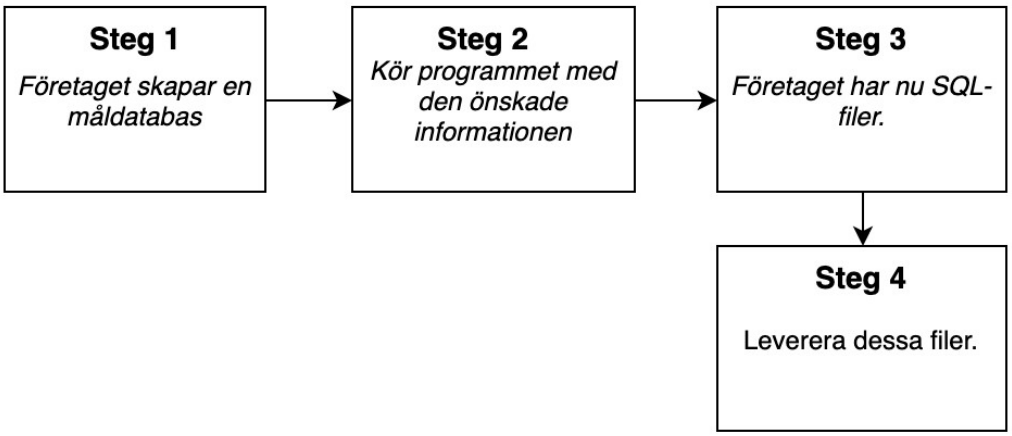
Projektets stora fokus ligger på själva scramblingen så ett enkelt kommandobaserat grönsnitt kommer att utvecklas där användaren kommer få fylla in den nödvändiga informationen för att scramblingprocessen ska gå att utföra. Denna process kommer gå till som figur8 visar. Frågor kommer bli ställda till användaren som besvaras med ett konkret svar och där efter sparas in i satta variabler.



Figur8: illustration av Användarvänlighet

4.5 Installationspaket

Körningen av scramblingen kommer att generera SQL-satser som kommer enkelt gå att förvara i tex-format och bli skickad till de företag som vill ha den scramblade databasen i figur 9 följer de fyra steg som denna process kommer att bestå av.



Figur9: Illustration av de steg installationen består av.

Sedan när företaget som ska ha den scramblade databasen har fått dessa filer kan de enkelt återskapa databasen fast med scamblad innehåll genom att ta dessa SQL-filer och med valfri SQL-kompatibla databasprogram som stödjer DB2 exekvera dessa filer och får tillgång till den scamblade databasen. Vidare så kan man med hjälp av någon textredigerare redigera och titta på dessa SQL-filer.

4.6 Validering

För att kunna validera dessa algoritmer konstruerades kod som finns i bilaga F för att kunna köra dessa mätningar som kommer att utföras och beräkna alla tagningar vad deras medelvärde och standardavvikelse är och utifrån detta så har man analyserat resultatet och även analyserat hur koden av de metoderna ser ut och vad som är bra och dåligt med den.

5 Resultat

Målet med detta projekt var (se kapitel 1.3) var att med hjälp av Easit befintliga kod utveckla ett system för att kunna scrambla databaser så att dess innehåll och struktur så att det var oigenkännligt.

Problemet delades in i fem olika fem olika moment första momentet där datan i databaserna skulle scamblas sedan skulle själva modellen av databasen. Sedan för att kunna använda systemet så har det utvecklats ett kommandobaserat system. Det sista praktiska uppgifterna var att hitta ett sätt att skapa ett installationspaket som skulle vara lätt att installera och använda.

Vidare för att kunna avgöra när scramblingen av datat var färdig och vilka metoder som passade bäst för projektets syfte har mätningar på scamblingsalgoritmer utförts. I 5.1 – 5.3 kommer resultatet av det system som kommer att kunna scrambla databaser indelade i de konkreta och verifierbara mål som hade satts och sist kommer valideringen på de metod som har valts att använda. Men även hur användning och instalationspaketets resultat är. Scrambling

Det slutliga systemet för att scrambla databasen basera sig i att man har en databaskälla som är den databas som man ska scambla. Vidare så har man en måldatabas som är där för att kunna överföra data ifrån skällan till målet. Medans denna överföring så appliceras scrambling av data och modell detta gör genom att konstruera SQL-filer är uppbyggd på den scamblande datan. Här nedan följer hur data-modellen och själva datan har scamblas.

5.1.1 Data

Resultatet av att scambla datan i databasen kan visas med hjälp ett databashanteringsprogram så kan man se hur måldatabasens innehåll har förändrats. Detta gör så att användaren kan bekräfta lätt om datan har blivit scramblad eller inte.

5.1.2 Modell

Själva modellen av databasen kan man ej se scamblad med hjälp av måldatabasen utan den blir genererad när insättningarna och skapande satserna i SQL skapas som sedan blir insatta i SQL-filer. Figur 10 och Figur 11 visar hur dessa filer se ut innan och efter scramblingen.

```
INSERT INTO "mediatype" ("MediaTypeId", "Name") VALUES  (1,'MPEG audio file'),
                                                        (2,'Protected AAC audio file'),
                                                        (3,'Protected MPEG-4 video file'),
                                                        (4,'Purchased AAC audio file'),
                                                        (5,'AAC audio file');
```

Figur10 SQL-filen innan scrambling.

```
INSERT INTO "IblHUMTLe" ("xbRDNsNobVt","143J") VALUES   ("1","tSQnoLekdLHWOSx"),
                                                        ("2","hyOkTsU0sQkVIpq3CUYhB1d "),
                                                        ("3","iVVLTkxJ3RqqzrsGOXCRzaz4ufq"),
                                                        ("4","d7LAPtCCgaoCi8rI 3vfKWc4"),
                                                        ("5","2TKkRZmfuU KN1");
```

Figur11 SQL-filen efter scrambling.

5.1.3 Filter

Resultatet av det filter som har blivit skapad är i form av en textfil där namn på de kolumner som man inte vill att man ska scramblad formuleras på så sätt att man urskiljer kolumnerna med ett ”,” och skriver på detta sätt. ”columnamn1, columnamn2, columnamn3……columnamn*n”. sedan* kommer denna ladas in i en Array som kommer användas som filter i programkörningen.

5.2 Användning

Användningen av applikationen är kommandobaserat och figur 12 visar hur processen går till. Denna fråga som visas i figuren kommer att bytas ut tills alla nödvändiga data har erhållits. Med lite information ifrån användaren så kommer applikationen köras ifrån start till slut och efter körningen kommer användaren stå med några filer som är lätt att installera

```
DatabaseScramblerApplication [Java Application] C:\Program Files\Java\jre1.8.0_201\bin\javaw.exe (27 maj 2019 14:43:20)
Do you want to scramble the databsemodel? (y/yes) or (n/no)n
Do you wont to apply a filter? (y/yes) or (n/no)y
```

Figur12 En del av de kommandobaserde systemet som har skapats.

5.3 Installation

Resultatet av installationen av den scramblade databasen gav två SQL-filer där ena som skapar datastrukturen i databasen såsom tabellerna, vad kolumnerna har för typ, storlek och deras förhållande till andra tabeller. Andra filen består utav den data databasen ska innehålla som är skriven med hjälp av insert statemens.

5.4 Mätningar

Resultatet av de mätningar som har skett är indelade i kapitel 5.4.1 som presenterats de sex algoritmer för generering av decimal och heltal vidare presenterar 5.4.2 resultatet på de nio algoritmerna för att generera strängar.

5.4.1 Tal

Tre olika sätt att generera slumpmässiga tal hittades under den studie som har blivit utförd och en av dem är att generera nummer med en fysiks enhet denna metod uteslutandes på grund av det kostar att skaffa denna enhet och med tanke på att scramblingen måste installeras och att den händer på olika platser.

De sex algoritmer för att generera tal som finns i bilaga B kan sammanställas med hjälp av prestandamätningarna som bilaga C innehåller. Utifrån dessa så kan man säga att algoritm 1 och 2 är nästa helt lika när det kommer till hastighet. Dessa två är implementerad exat lika det som skiljer dem åt är de värde den returen. Algoritm 1 returnera int och Algoritm 2 returnerar float. Om man tittar på de resterande 4 algoritmer ser man att deras prestanda är extremt mycket sämre än algoritm 1 och 2. vidare har resultatet sammanställts i tabell 1 där algoritmerna och deras tider finns sorterad bäst till sämst.

T*abell1 Hastigheterna för de snabbaste algoritmerna.*

Algoritm nr	Medeltid(ms)
1	26 398
2	26 528
3	83 406
4	104 641
5	För lång tid

En grej som man måste ta hänsyn till är algoritm 6 den är brutalt mkt långsammare än den snabbaste. Anledningen är att denna algoritm är krypterings säker vilket betyder att den når de krav som ställs av en slumptals-generator för att gå

att använda i kryptografi. I detta projekt så kommer man förstöra data och det kommer inte finnas någon väg tillbaka så denna metod uteslutandes.

Vidare så använder algoritm 1 och 2 math.random som är implementerad på det viset att den kan dra ner på kraften på trådar för andra beräkningar för att kunna vara så effektiv som möjligt. Medans algoritm 3 bara returnerar nästa tall i den genererade sekvensen. I detta projekt så kommer det inte behövas ta hänsyn till att vara så sparsam som möjligt på kraften på trådar så den slutliga metoden blir både algoritm 2 och 2 eftersom ena hanterar decimalt och andra heltal.

5.4.2 Strängar

I bilaga E ser man mätningarna ifrån de algoritmer man har jämfört utifrån dessa så kan de nio algoritmerna delas in i tre grupper dåliga, medel och bra baserat på deras tid och avvikelse. Bland de dåliga algoritmerna finns algoritm 4 och fem i medelgruppen finns 7,6,8 och gruppen bra finns 1,2,3 vidare så har en sammanställning gjorts som visas i tabell 2 på de bästa algoritmerna vilket baseras på hur deras medeltid har set ut.

Tabell2 Hastigheterna för de snabbaste algoritmerna.

Algoritm nr	Medeltid(ms)	Sdv(ms)
1	19 255	40
2	23 881	104
3	26 421	237
8	29 287	50
7	31 730	328
6	32 907	394
4	49 229	335
5	273 097	168

Detta resultat visar vilken algoritm som är snabbast men det finns fördelar med de långsammare som man måste ta hänsyn till som t.ex. algoritm 2 är baserat på ett charset ”UTF-8” dessa går att variera till vilket sett som helst detta är en stor fördel om man skulle vilja ha med svenska karaktärer som ”ÅÄÖ”. Samma tänk går att applicera på algorithm 3 där skapar man en sträng med vilka karaktärer man vill att den ska skapa strängen med. Så då är frågan om företaget som vill scrambla databasen att karaktärer som ”ÅÄÖ” ska finnas med i scramblingen.

5.5 Slutgiltiga metoden

Baserat på de mätningar som har skett och den information om de algoritmer som visade sig vara bra så blev den slutliga algoritmen för att generera en sträng algoritm 1. dels för att den visade sig vara den snabbaste algoritmen med minst avvikelse men även för att den är baserat på javas grundläggande språk i java.util.random vilket gör att man inte behöver förlita sig på att ha bibliotek.

När det kommer till att generera slumpmässiga tal så kan man på en gång utesluta Secure random för när man talar om databaser så kan storleken vara enorm och en skillnad på några sekunder kan innebära stor skillnad i slutändan och en enligt bilaga D så ser man att skillnaden i tid är stora. Och eftersom skillnaden är så stor så har den snabbaste valts.

6 Slutsatser

I kapitel 5.1 – 5.4 kommer slutsatser av det system som kommer att kunna scrambla databaser som har implementeras indelad i de konkreta och verifierbara mål som hade satts och validering på de metod som har valts att använda. Och även hur man sen kan vidareutveckla systemet

6.1 Val av scramblingmetoder

. De val av scramblingmetoder man valde att använda är baserat på va som var bäst för projektets syfte och genom att ha utfört olika prestandamätningar så kan man i praktiken ha sparat enormt stor tid på grund av att en databas som är i produktion kan innehålla flera miljoner insättningar. För till exempel om en scrambling skulle ta 25 sekunder emot 30 sekunder skulle man generera 100 st strängar så skulle man spara 8 minuter därför var val av scrambling ett mkt viktigt val för att hitta den mest optimala metoderna. Så att de snabbaste metoderna algoritm 1för strängar och algoritm 1 och 2 för tal valdes har stor betydelse. Sen om önskan att ha med karaktärer ifrån olika charset såsom ”UTF-8” så måste man offra lite tid.

6.2 Scrambling

Det som blev det slutliga systemet för att scambla består utav två stora delar. En del som scamblar den data som databasen innehåller. Den andra delen tar han om att scrambla datamodellen hur slutsatsen av dessa två k beskrivs i 6.2.1 och 6.2.2

6.2.1 Data

Att scambla datan i databasen var det första uppgiften som löstes med den kod som Easit AB hade skapat för att demonstrera hur man med hjälp av deras kod kunde överföra data mellan två källor. Efter implementeringen av scramblingen av data va klar och de andra uppgifterna kom in i bilden insågs det att installationspaketet som skulle skapas och scambling av modell kunde göras samtidigt och detta ledde till att måldatabasens modell inte blev scramblad eftersom denna scambling hände på ett annat sätt. Så på ett sätt så är detta dåligt för man har efter scramblingen inget sätt att direkt se hur modellen är scramblad. För att göra detta måste man köra dessa SQL-filer som skapas

6.2.2 Model

Vidare tack vare en annan lösning kom upp så är resultat på hur modellen scamblades kopplad till hur installationspaketet skapades. Medans man skapar dessa SQL-filer så för att dom ska stämma överens så måste ett system för att hålla koll på nycklarna och vad som är scramblad och inte så medans detta paket skapas så ser man till att modellen håller sig intakt. Så för att leverera en scramblad databas så fungerar det bra men för att få se själv hur databasen är lite krångligare.

6.3 Installationspacket

Programmet kommer att köras av de företag som har databasen detta medför en större säkerhet för företaget eftersom dom själva kommer att utföra operationen där företaget kommer efter körning få ett par SQL-filer som kan läsas av alla SQL-kompatibla databasprogram, inklusive FileMaker, Microsoft Access och MySQL. och de kan redigeras med någon textredigerare. och kan då därefter bli analyserad av företaget innan den skickas tillbaka till Easit detta då för om dom känner sig oro att inte alla information är scamblad som dom vill. Vidare är de lika lätt för Easit och installera detta om de har SQL-filer.

6.4 Filter

Filtret som är baserat på en textfil som sedan läses in som en Array. Detta fungerar utmärkt när man inte har så många filter som ska appliceras. Men eftersom databaser kan bli rätt så stora och man lätt kan ha många filter så kan detta leda till att det bli stökigt i filen. Bortsätt ifrån det så är funktionaliteten där och utför vad som är tänkt med filtret.

6.5 Etiska och samhälleliga aspekter

För ett företag som Easit AB som jobbar med testdata och att avidentifiera databaser så att databaser kan användas för tester utan att man använder verkliga data. Då är frågan om hur information rörande personer hanteras en mycket väsentlig fråga att ställa. Och i detta projekt är syftet att förstöra denna information så att det på ett säkert sätt kan garantera att företaget kan lämna ut en databas som ser ut som den ursprungliga men data, namn på kolumner är förstörd och de relationer och kopplingar mellan alla tabeller är intakta.

På samma sätt som GDPR fungerar för människor så kommer detta projekt garantera att företaget kan lämna en databas utan att riskera att personlig information kommer till fel personer eller företag som inte ska ha tillgång till denna information. Detta medför även att de personer som är inskriven i denna databas också kan känna sig trygga att informationen inte kommer att hanteras på fel sätt.

Utöver de så är det företaget som har databasen som kommer att hantera och köra scramblingen av databasen så detta stärker förtroendet om att man kan garantera att den ursprungliga datan inte kommer att nå de företaget som ska ha den scramblade databasen.

6.6 Vidareutveckling

Denna scrambing misslyckas med att bibehålla den propagering av data och deras nycklar. I en vidareutveckling så skulle de första va att konstruera en metod som kan behålla denna struktur. För att komma mer närmare riktiga data.

Användarvänligheten för applikationen är baserat på ett enkelt kommandobaserat gränssnitt och är inte helt användarvänligt om man ser det med ett MDI perspektiv för vardagliga människor men för en person som har erfarenhet inom datorer och kommandobaserat system kommer det upplevas lätt. Så att utveckla ett grafiskt gränssnitt där man på ett metodiskt sätt kan scarambla databaser med hjälp av en steg för steg i processen att scrambla databasen.

Vidare så finns det många fler databaser än bara IBM:s DB2 några av dom större och mer använda är Oracles Oracle, Microsofts SQL Server eller MySQL som alla är baserade på SQL som frågespråk men alla dessa är vidareutvecklad har alla har sina egna typer och metoder för att konstruera databaser. Eftersom detta projekt jobbade mot en DB2 databas och inga tester på andra databaser undersökts så att vidareutveckla detta system så att fler databaser stöds. Men även se om att det går att bygga stöd flera typer som är lite mindre vanliga så som BLOB, BIGINT med mera.

De filter som har skapas täcker bara de kolumner som scramblingen ska undanta. Vilket betyder att bara datat i själva kolumnen inte kommer att scramblas. Vidare så hade man kunnat applicera istället för en textfils lösning konstruera ett filter som använder tex Jason för att kunna specificera tabellnamn, Kolumnnamn med mera för att få bättre struktur och ta mer än bara kolumner.

Källförteckning

[1] ” Datakyddsförondingen” https://www.datainspektionen.se/lagar--regler/dataskyddsforordningen/ Hämtad 2019-04 -08

[2] ” What is a relational database?” https://aws.amazon.com/relational-database/ Hämtad 2019-04-0

[3] Thomas Padron-McCarthy, Tore Risch, Databasteknik. 2 uppl, Studentlitteratur AB, 2018

[4] ”What is SQL?” http://www.sqlcourse.com/intro.html Hämtad 2019-06-06

[5] ” DB2 introduction” https://www.tutorialspoint.com/db2/db2_introduction.htm Hämtad 2019-04-08

[6] ” IBM® Informix® JDBC Driver, Version 4.10” https://www.ibm.com/support/knowledgecenter/en/SSGU8G_12.1.0/co m.ibm.jdbc_pg.doc/ids_jdbc_011.htm Hämtad 2019-04-08

[7] ” IBM® Informix® JDBC Driver, Version 4.10” https://www.ibm.com/support/knowledgecenter/en/SSGU8G_12.1.0/co m.ibm.jdbc_pg.doc/ids_jdbc_011.htm Hämtad 2019-04-08

[8] ” Spring Boot” https://www.-technipelago.se/training/show/BOOT-1 Hämtad 2019-04-08

[9] ” Easit” https://easit.se/produkter/ Hämtad 2019-04-08

[10] ” What is test data” https://www.softwaretestingmentor.com/what-is-test-data/ Hämtad 2019-04-0

[11] ” Avidentifiering” https://it-ord.idg.se/ord/pseudonymisering/ Hämtad 2019-06-18

[12] ” Hardware random number generators”
http://robertnz.net/hwrng.htm
Hämtad 2019-04-08

[13] Niels Ferguson Bruce Schneier Tadayoshi Kohno, Cryptography Engineering: Design Principles and Practical Applications. Canada: Wiley Publishing inc , 2010

[14] ”Cryptographically Secure Pseudo-Random Number Generator (CSPRNG)” https://www.veracode.com/blog/research/cryptographically-securepseudo-random-number-generator-csprng Hämtad 2019-06-06

[15] ” Commons Lang” https://commons.apache.org/proper/commons-lang/ Hämtad 2019-04-0

[16] ” Class Math” https://docs.oracle.com/javase/7/docs/api/java/lang/Math.html#random() Hämtad 2019-04-08

[17] ” Terminology” https://docs.oracle.com/javase/tutorial/i18n/text/terminology.html Hämtad 2019-04-0

[18] ” Welcome to Regular-Expressions.info The Premier website about Regular Expressions” https://www.-regular-expressions.info/ Hämtad 2019-04-08

[19] ” Class Random” https://docs.oracle.com/javase/8/docs/api/java/util/Random.html Hämtad 2019-04-08

[20] "Class SecureRandom” https://docs.oracle.com/javase/8/docs/api/java/security/SecureRandom.html
Hämtad 2019-06-06

[21] ”DATPROF. How to mask testdata” https://www.datprof.com/tutorials/how-to-mask-test-data/ Hämtad 2019-04-0

[22] E.B. Boukobza, ”System and method for data masking”, US8,826,370 B2, sep. 2, 2014

[23] C.L Lei, T.K keefe , D.W Wong, ”Method and apparatus for encryptning database Columns”, US8,826,370 B2, sep. 2, 2014

Bilaga A: Scamblings algoritmer för att generera strängar

Algoritm 1

```
public void alg1(int n) {
	int leftLimit = 97; // letter 'a'
	int rightLimit = 122; // letter 'z'
	int targetStringLength = n;
	Random random = new Random();
	StringBuilder buffer = new StringBuilder(targetStringLength);
	for (int i = 0; i < targetStringLength; i++) {
		int randomLimitedInt = leftLimit + (int) (random.nextFloat() * (rightLimit - leftLimit + 1));
		buffer.append((char) randomLimitedInt);
	}
	String generatedString = buffer.toString();
    // System.out.println(generatedString);
}
```

Algoritm 2

```
public void alg2(int n) {
	byte[] array = new byte[n]; // length is bounded by 7
	new Random().nextBytes(array);
	String generatedString = new String(array, Charset.forName("UTF-8"));
	//System.out.println(generatedString);
}
```

Algoritm 3

```
public String alg3(int n)
{
	// chose a Character random from this String
	String AlphaNumericString = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
			+ "0123456789"
			+ "abcdefghijklmnopqrstuvxyz";
	// create StringBuffer size of AlphaNumericString
	StringBuilder sb = new StringBuilder(n);
	for (int i = 0; i < n; i++) {
		// generate a random number between
		// 0 to AlphaNumericString variable length
		int index
			= (int)(AlphaNumericString.length()
			* Math.random());
		// add Character one by one in end of sb
		sb.append(AlphaNumericString
			.charAt(index));
	}
```

```
    //System.out.println(sb.toString());
    return sb.toString();
}
```

Algoritm 4

```
public String alg4(int n)
{
    // length is bounded by 256 Character
    byte[] array = new byte[256];
    new Random().nextBytes(array);
    String randomString
            = new String(array, Charset.forName("UTF-8"));
    // Create a StringBuffer to store the result
    StringBuffer r = new StringBuffer();
    // Append first 20 alphanumeric characters
    // from the generated random String into the result
    for (int k = 0; k < randomString.length(); k++) {
        char ch = randomString.charAt(k);
        if (((ch >= 'a' && ch <= 'z')
                || (ch >= 'A' && ch <= 'Z')
                || (ch >= '0' && ch <= '9'))
                && (n > 0)) {
            r.append(ch);
            n--;
        }
    }
    // return the resultant string
   // System.out.println(r.toString());
    return r.toString();
}
```

Algoritm 5

```
public String alg5(int n)
{
    // length is bounded by 256 Character
    byte[] array = new byte[256];
    new Random().nextBytes(array);
    String randomString
            = new String(array, Charset.forName("UTF-8"));
    // Create a StringBuffer to store the result
    StringBuffer r = new StringBuffer();
    // remove all spacial char
    String  AlphaNumericString
            = randomString
            .replaceAll("[^A-Za-z0-9]", "");
    // Append first 20 alphanumeric characters
    // from the generated random String into the result
    for (int k = 0; k < AlphaNumericString.length(); k++) {
        if (Character.isLetter(AlphaNumericString.charAt(k))
                && (n > 0)
                || Character.isDigit(AlphaNumericString.charAt(k))
```

```
                    && (n > 0)) {
                r.append(AlphaNumericString.charAt(k));
                n--;
            }
        }
        // return the resultant string
        //System.out.println(r.toString());
        return r.toString();}
```

Algoritm 6

```
public void alg7(int n) {
        int length = n;
        boolean useLetters = true;
        boolean useNumbers = false;
            String generatedString = RandomStringUtils.random(length, useLetters, useNumbers);
      // System.out.println(generatedString);
}
```

Algoritm 7

```
public void alg8(int n) {
        String generatedString = RandomStringUtils.randomAlphabetic(n);
        //System.out.println(generatedString);
}
```

Algoritm 8

```
public void alg9(int n) {
        String generatedString = RandomStringUtils.randomAlphanumeric(n);
      // System.out.println(generatedString);
}
```

Bilaga B: scamblings Algoritmer för att generera nummer

Algoritm 1

```
public static double alg1(double min, double max){
	double x = (Math.random()*((max-min)+1))+min;
	return x;
}
```

Algoritm 2

```
public static double alg2(double min, double max){
	double x = (int)(Math.random()*((max-min)+1))+min;
	return x;
}
```

Algoritm 3

```
public static int alg3(int min, int max) {
	Random r = new Random();
	return r.nextInt((max - min) + 1) + min;
}
```

Algoritm 4

```
public static int alg4(int min, int max){
	Random random = new Random();
	return random.ints(min,(max+1)).findFirst().getAsInt();
}
```

Algoritm 5

```
public static void alg5(int num, int min, int max) {
	Random random = new Random();
	random.ints(num,min, max).sorted();
}
```

Algoritm 6

```
public static void alg6(int num, int min, int max) {
	// create instance of SecureRandom class
	SecureRandom rand = new SecureRandom();
	// Generate random integers in range 0 to 999
	int rand_int1 = rand.nextInt(1000);
	//System.out.println("Random Integers: " + rand_int1);
}
```

Bilaga C: scamblings Algoritmer för att generera tal

Algoritm 1

Körning nr	n=250 000 000	n=500 000 000	n=750 000 000	n=1 000 000 000
1	6214	13 692	19 982	26 398
2	6164	13 682	19 599	25 736
3	6236	13 069	19 673	25 587
4	6312	13 634	19 703	26 452
5	6269	13 582	20 397	25 146

Algoritm 2

Körning nr	n=250 000 000	n=500 000 000	n=750 000 000	n=1 000 000 000
1	6454	13 362	20 368	26 528

Algoritm 3

Körning nr	n=250 000 000	n=500 000 000	n=750 000 000	n=1 000 000 000
1	19 578	37 425	57 274	83 406

Algoritm 4

Körning nr	n=250 000 000	n=500 000 000	n=750 000 000	n=1 000 000 000
1	26 893	49 918	74 021	104 641

Algoritm 5

Körning nr	n=250 000 000	n=500 000 000	n=750 000 000	n=1 000 000 000
1	24 566	45 103	65 748	95 796

Algoritm 6

Körning nr	n=100 000	n=1 000 000	n=10 000 000
1	1980	6144	41537

Bilaga D: standardavvikelse och medeltid för Algoritmer

Standardavvikelse 1

Algoritm nr	Sdv för n =100 000	Sdv för n =1 000 000	Sdv i ms för n =10 000 000	Sdv för n =100 000 000
1	6	26	28	40
2	15	49	23	237
3	6	38	117	50
4	16	87	135	168
5	69	95	2208	206
7	4	45	40	394
8	7	33	76	335
9	7	34	70	328

Medeltid 2

Algoritm nr	mean =100 000	mean =1 000 000	mean =10 000 000	mean =100 000 000
1	32	258	1998	19 255
2	68	312	2366	23 881
3	44	330	2761	26 421
4	87	562	4954	49 229
5	358	2777	28 195	273 097
8	56	369	3238	32 907
7	52	398	3320	31 730
6	52	346	2771	29 287

Bilaga E: Mätningar får Algoritmer för att generera strängar

Algoritm 1

Körning nr	n=100 000	n=1 000 000	n=10 000 000	n=100 000000
1	24 ms	249 ms	2049 ms	19 221 ms
2	33 ms	293 ms	2002 ms	19 200 ms
3	31 ms	281 ms	1994 ms	19 254 ms
4	31 ms	255 ms	1983 ms	19 294 ms
5	45 ms	216 ms	1964 ms	19 306 ms

Algoritm 2

Körning nr	n=100 000	n=1 000 000	n=10 000 000	n=100 000 000
1	77 ms	328 ms	2386 ms	24 335 ms
2	73 ms	392 ms	2404 ms	23 741 ms
3	88 ms	298 ms	2346 ms	23 840 ms
4	61 ms	238 ms	2350 ms	23 836 ms
5	43 ms	304 ms	2348 ms	23 653 ms

Algoritm 3

Körning nr	n=100 000	n=1 000 000	n=10 000 000	n=100 000 000
1	45 ms	337 ms	2707 ms	26 506 ms
2	37 ms	359 ms	2715 ms	26 446 ms
3	38 ms	381 ms	2694 ms	26 410 ms
4	47 ms	301 ms	2997 ms	26 362 ms
5	55 ms	275 ms	2695 ms	26 383 ms

Algoritm 4

Körning nr	n=10 000	n=100 000	n=1 000 000	n=10 000 000
1	73 ms	735 ms	4954 ms	49 121 ms
2	78 ms	517 ms	4871 ms	49 114 ms
3	106 ms	511 ms	4862 ms	49 053 ms
4	109 ms	548 ms	4867 ms	49 377 ms

5	73 ms	503 ms	5217 ms	49 482 ms

Algoritm 5

Körning nr	n=10 000	n=100 000	n=1 000 000	n=10 000 000
1	470 ms	2912 ms	27 633 ms	273 032 ms
2	408 ms	2874 ms	31 793 ms	272 851 ms
3	295 ms	2704 ms	29 486 ms	273 267 ms
4	304 ms	2695 ms	25 788 ms	273 403 ms
5	314 ms	2701 ms	26 279 ms	272 934 ms

Algoritm 6

Körning nr	n=100 000	n=1 000 000	n=10 000 000	n=100 000 000
1	54 ms	411 ms	3196 ms	32 867 ms
2	50 ms	422 ms	3316 ms	32 865 ms
3	60 ms	381 ms	3224 ms	32 577 ms
4	62 ms	316 ms	3229 ms	32 577 ms
5	58 ms	317 ms	3229 ms	33 652 ms

Algoritm 7

Körning nr	n=100 000	n=1 000 000	n=10 000 000	n=100 000 000
1	49 ms	403 ms	3318 ms	31 111 ms
2	42ms	462 ms	3332 ms	31 901 ms
3	49 ms	375 ms	3454 ms	31 874 ms
4	56 ms	385 ms	3273 ms	32 087 ms
5	64 ms	368 ms	3226 ms	31 681 ms

Algoritm 8

Körning nr	n=100 000	n=1 000 000	n=10 000 000	n=100 000 000
1	54 ms	386 ms	2909 ms	29 225 ms
2	44 ms	379	2718	29 737 ms
3	50 ms	352	2735	29 035 ms
4	48 ms	296	2733	28 859 ms
5	65 ms	318	2764	29 583 ms

Bilaga F: Kod för hur för de mätningar som har utförts

Mätningar

```
public static void main(String[] args) {
		Algoritms a = new Algoritms();
		long startTime5 = System.currentTimeMillis();
		long endTime5 = System.currentTimeMillis();
		//////////////////////////////////////  run 1
		startTime5 = System.currentTimeMillis();
		for (int x = 0; x <= 100000; x++){
			a.alg9(10);
		}
		endTime5 = System.currentTimeMillis();
		System.out.println( (endTime5-startTime5) + " ms");
		//////////////////////////////////////
		//////////////////////////////////////  run 2
		startTime5 = System.currentTimeMillis();
		for (int x = 0; x <= 100000; x++){
			a.alg9(10);
		}
		endTime5 = System.currentTimeMillis();
		System.out.println( (endTime5-startTime5) + " ms");
		//////////////////////////////////////
		//////////////////////////////////////  run 3
		startTime5 = System.currentTimeMillis();
		for (int x = 0; x <= 100000; x++){
			a.alg9(10);
		}
		endTime5 = System.currentTimeMillis();
		System.out.println( (endTime5-startTime5) + " ms");
		//////////////////////////////////////
		//////////////////////////////////////  run 4
		startTime5 = System.currentTimeMillis();
		for (int x = 0; x <= 100000; x++){
			a.alg9(10);
		}
		endTime5 = System.currentTimeMillis();
		System.out.println( (endTime5-startTime5) + " ms");
		//////////////////////////////////////
		//////////////////////////////////////  run 5
		startTime5 = System.currentTimeMillis();
		for (int x = 0; x <= 100000; x++){
			a.alg9(10);
		}
		endTime5 = System.currentTimeMillis();
		System.out.println( (endTime5-startTime5) + " ms");
		//////////////////////////////////////
	}
}
```

Standardavvikelse och medelvärde

```
public static double calculateSD(double numArray[])
{
	double sum = 0.0, standardDeviation = 0.0;
	int length = numArray.length;
	for(double num : numArray) {
		sum += num;
	}
	double mean = sum/length;
	for(double num: numArray) {
		standardDeviation += Math.pow(num - mean, 2);
	}
	return Math.sqrt(standardDeviation/length);
}

public static double mean(double[] m) {
	double sum = 0;
	for (int i = 0; i < m.length; i++) {
		sum += m[i];
	}
	return sum / m.length;
}
```