YRKESHÖGSKOLAN
NOVIA

# Överföring av data från SQLite till PostgreSQL

Mattias Långgård

Examensarbete för ingenjör (YH)-examen

El- och automationsteknik

Vasa 2023

**EXAMENSARBETE**

Författare: Mattias Långgård
Utbildning och ort: El- och automationsteknik, Vasa
Inriktning: Informationsteknik
Handledare: Susanne Österholm

Titel: Överföring av data från SQLite till PostgreSQL

---

Datum: 23.5.2023 Sidantal: 16

---

**Abstrakt**

Detta arbete gjordes på uppdrag av Edupower Oy Ab. Uppgiften gick ut på att överföra data från en databas med databashanteringssystemet SQLite till en databas med befintliga data som använder databashanteringssystemet PostgreSQL genom att tillämpa ETL (Extract, Transform, Load) -metodik och effektiv verktygshantering. Målet var att säkerställa en smidig överföring av data från en databas till en annan, samtidigt som man bevarade dataintegriteten och hanterade referenser mellan tabeller.

Genom en noggrann kartläggning och analys av tabellstrukturer, identifierades de relevanta datamängderna för överföringen. Genom att tillämpa ETL-metoden extraherades data från källsystemet och genomgick sedan en omfattande transformationsprocess för att anpassa den till det nya databasformatet. Genom att undersöka olika metoder och verktyg för att lösa detta problem valdes Pgloader som huvudsakligt verktyg för överföringen.

Under överföringsprocessen identifierades utmaningar relaterade till referenser mellan tabeller, vilket ledde till problem vid driftsättningen av applikationen som använde databasen. För att hitta en mer tillförlitlig lösning undersöktes olika möjligheter och diskuterades strategier för att hantera referensproblemet och säkerställa en korrekt driftsättning av systemet. Valet gjordes att söka efter en enklare och mer automatiserad metod för att lösa referensproblemen.

---

Språk: svenska
Nyckelord: SQLite, PostgreSQL, överföring, data

**BACHELOR'S THESIS**

Author: Mattias Långgård
Degree Programme: Electrical and Automation Engineering
Specialisation: Information Technology
Supervisor(s): Susanne Österholm

Title: Migration of Data from SQLite to PostgreSQL

---

Date: 23.5.2023 Number of pages: 16

---

**Abstract**

This work is assigned by Edupower Oy Ab. The task involved transferring data from a database using the SQLite database management system to a database with existing data using the PostgreSQL database management system, applying Extract, Transform, Load (ETL) methodology, and efficient tool management. The goal was to ensure a smooth transfer of data from one database to another while preserving data integrity and managing references between tables.

Through careful mapping and analysis of table structures, the relevant datasets for the transfer were identified. By applying the ETL methodology, data was extracted from the source system and then underwent an extensive transformation process to adapt it to the new database format. After examining various methods and tools to address this problem, Pgloader was chosen as the primary tool for the transfer.

During the transfer process, challenges related to table references were identified, resulting in issues during the deployment of the application using the database. To find a more reliable solution, different possibilities were explored, and strategies were discussed to address the reference problem and ensure the proper deployment of the system. The decision was made to search for a simpler and more automated method to solve the reference issues.

---

Language: Swedish
Key words: SQLite, PostgreSQL, data migration, data

# Innehållsförteckning

# 1 Introduktion

Detta arbete utfördes på uppdrag av Edupower Oy Ab. Syftet var att överföra data från en databas i en testmiljö av mjukvaran Phonelog som använder ett existerande SQLite-databashanteringssystem, till en ny version av Phonelog som har ett PostgreSQL-databashanteringssystem som körs på DigitalOceans molntjänst.

Phonelog är en app som hämtar samtalsdata från en telefonkatalog så att man kan sortera samtalsdata och lägga till vissa specifika data som hjälper Edupowers försäljare vid kontakt med potentiella kunder.

## 1.1 Edupower Oy Ab

Edupower är ett företag med huvudkontoret på Vasa Universitets campusområde. Det grundades år 2012 och fokus ligger på konsultation, utbildning och träning inom Lean-ledarskap. Kim Westerlund är grundaren för företaget och har lång erfarenhet i energibranschen och internationell affärsverksamhet. Edupower coachar och utbildar kunder i Lean-ledarskapets metoder och verktyg med målet att inrätta bestående produktivitetsframsteg hos företag och organisationer. (Westerlund, u.d.).

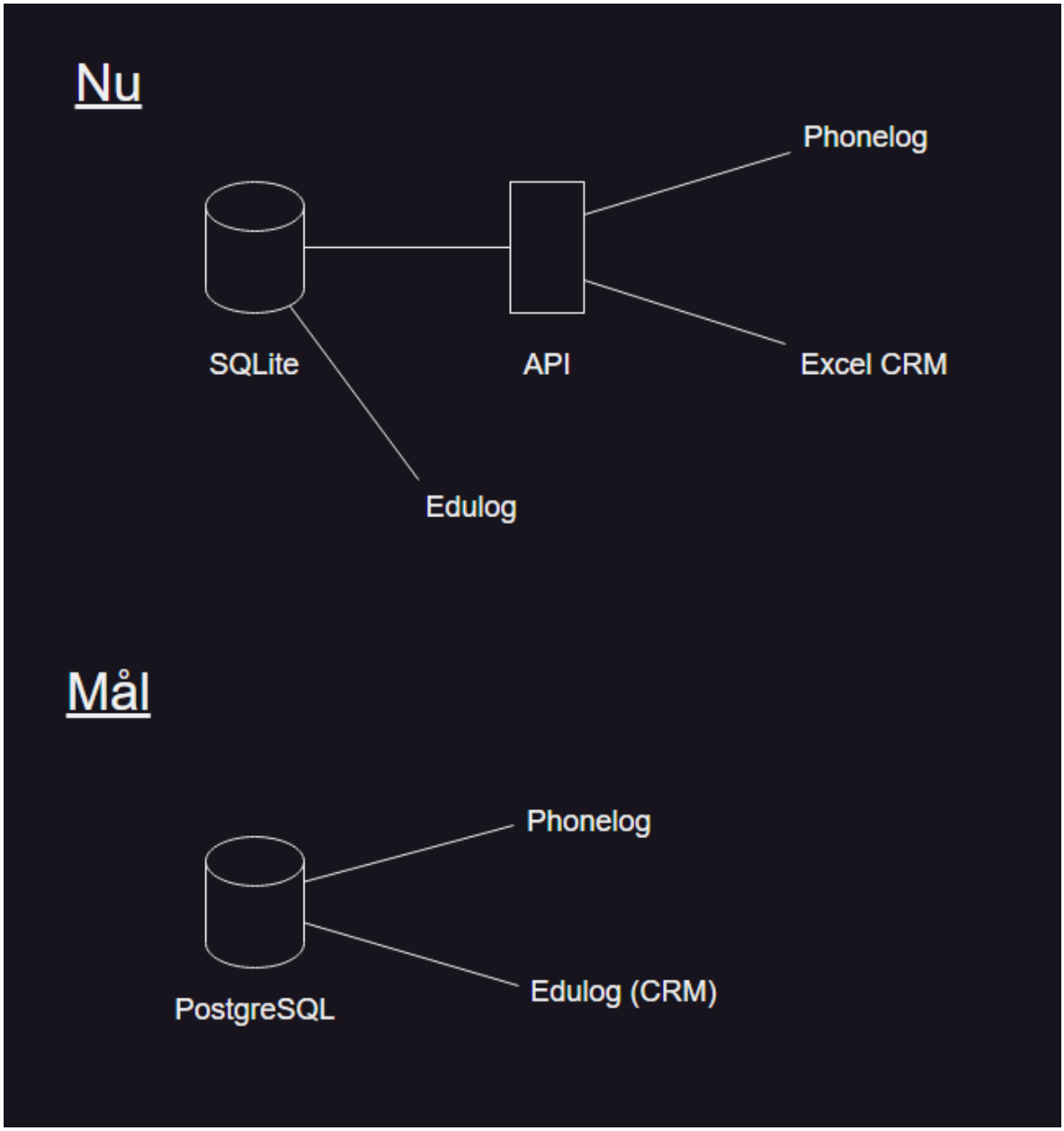
## 1.2 Bakgrund och motiv för överföringen

Teamet skapade inledningsvis ett konceptbevis för synkronisering av data mellan Phonelog och Excel CRM-system i en testmiljö. Båda systemen använder API-anrop för att ansluta till databasen, men databasen i testmiljön är SQLite, vilket kräver att man bygger ett anpassat API som kan ta emot HTTP-förfrågningar, utföra databasoperationer och returnera HTTP-svar.

Till en början fungerade den här lösningen bra för små tillämpningar, testning och snabb prototypframställning, men i takt med att projektets omfattning ökade, ökade den tekniska skulden och underhållsbördan för API-kodbasen. Dessutom ville Edupower övergå från Excel CRM till ett webbaserat CRM som de planerar att bygga i Edulog, som är Edupowers webbserver.

För att lösa dessa problem beslutade teamet att byta från SQLite till Postgres med en direkt databasanslutning, vilket borde lösa de flesta av dessa problem. Eftersom alla data lagras i

SQLite finns det dock ett behov av att överföra dem till Postgres. Denna process är inte okomplicerad eftersom delar av den är komplexa på grund av övervakade kopplingar och referenser mellan tabeller.




Figur 1: Diagram som beskriver funktionen nu och den önskvärda funktionen efter överföring.

# 2 Databashanterare

I detta kapitel presenteras två populära databashanteringssystem, SQLite och PostgreSQL. Båda databashanteringssystemen är baserade på öppen källkod och används för att hantera data i en mängd olika områden, från mobila applikationer till stora företagsapplikationer.

## 2.1 Översikt över SQLite- och PostgreSQL-databashanteringssystem

SQLite är en populär relationell databas som används för att lagra och hantera strukturerade data. Den skapades av Richard Hipp år 2000 och har sedan dess blivit ett av de mest använda inbäddade databashanteringssystemen. (SQLite, u.d.).

Historien om SQLite börjar med Hipp's arbete som entreprenör för amerikanska flottan. Han var ansvarig för att utveckla mjukvara för att hantera data på trånga budgetar och begränsade resurser. Under denna tid insåg Hipp behovet av en lättanvänd och snabb databas för att möta kraven på inbäddade system.

Som svar på dessa utmaningar började Hipp att utveckla SQLite som en minimalistisk databasimplementering. Han fokuserade på att skapa en självständig databasmotor som kunde köras utan externa beroenden eller en separat serverprocess. Detta gjorde det möjligt att inbädda SQLite i olika applikationer utan att kräva omfattande konfiguration eller installation. (Cassel, 2021).

PostgreSQL är en kraftfull och populär objektrelationell databas (ORDBMS) som har funnits sedan 1986. Den har sitt ursprung vid University of California, Berkeley och har sedan dess utvecklats till ett av de mest robusta och avancerade öppna källkods-databashanteringssystemen. (Kuhn, 2018).

PostgreSQL-projektet började som en forskningssatsning för att skapa en bättre version av Ingres-databashanteringssystemet. Professor Michael Stonebraker och hans team vid Berkeley ansåg att Ingres kunde förbättras genom att inkludera objektorienterade funktioner och bättre stöd för relationella egenskaper. Detta arbete ledde till utvecklingen av en prototyp för Postgres (Post Ingres), som senare utvecklades till PostgreSQL. (W3schools, History of PostgreSQL, u.d.).

Under tidiga utvecklingssteg fokuserade PostgreSQL på att implementera kärnfunktionaliteten för att hantera komplexa datastrukturer och förbättra skalbarheten. Projektet betonade också på ACID-kompatibilitet (Atomicity, Consistency, Isolation, Durability) och transaktionssäkerhet, vilket resulterade i en databas som passar för både små och stora företagsapplikationer. (PostgreSQL, u.d.).

Ett viktigt steg i PostgreSQLs historia var frisläppandet av den första stabila versionen, PostgreSQL 6.0, år 1996. Denna version erbjöd flera innovativa funktioner, inklusive transaktioner, deklarativ referentiell integritet och flerspråkigt stöd. Den blev snabbt populär inom forskarsamhället och bland utvecklare. (PostgreSQL, A brief history of PostgreSQL, u.d.).

## **2.2** Jämförelse av de två databashanteringssystemen

SQLite är ett litet och fristående system för hantering av relationsdatabaser som lämpar sig väl för lokal lagring och småskaliga tillämpningar. Det är utformat för att kunna bäddas in i ett program och kräver ingen nätverksanslutning eller separat serverprocess, endast en liten DLL-fil på några megabyte. SQLite är känt för sin höga prestanda, tillförlitlighet och stöd för ett stort antal SQL-funktioner. Den är särskilt lämplig för mobilapplikationer och webbapplikationer. Dess enkelhet och lilla storlek gör det till ett populärt val för utvecklare som behöver ett snabbt och lättanvänt databashanteringssystem.

SQLite har en begränsning i storlek på 281 terabytes. Det lagrar hela databasen på en enda fil, vilket kan leda till problem med filsystem som begränsar storleken på filer. För väldigt stora datamängder eller tillämpningar som kräver så frekventa läs- och skrivoperationer att det krävs flera servrar rekommenderas det att man använder sig av ett databashanteringssystem med klient-serverstruktur. (SQLite, 2022).

SQLites transaktionshantering stöder endast läsåtgärder, vilket betyder att två användare kan inte samtidigt utföra querys som ändrar på data inne i en SQLite-hanterad databas. Det går dock utmärkt att utföra querys som läser data.

PostgreSQL är ett relationsdatabashanteringssystem med öppen källkod som är känt för sin stabilitet, tillförlitlighet och skalbarhet. Det erbjuder en rad funktioner, bland annat stöd för avancerad indexering, transaktionshantering och användardefinierade funktioner. Det

har stöd för flera programmeringsspråk, vilket gör det mångsidigt för utvecklare. PostgreSQLs höga nivå av dataintegritet och konsistens gör den lämplig för tillämpningar som kräver strikt datastyrning. Den är skalbar och kan hantera stora datamängder och applikationer med hög trafik. Dess karaktär av öppen källkod och aktiv utvecklargemenskap gör det till ett populärt val för organisationer av alla storlekar.

Både PostgreSQL och SQLite har vidsträckt stöd för olika datatyper, till exempel int, char, string, boolean och geometriska datatyper. Dock så ger PostgreSQL en mera extensiv lista av inbyggda sådana, som kan vara särskilt användbara för specialiserade applikationer. Exempelvis ger PostgreSQL stöd för arrays, JSON och XML datatyper, nätverksadresser och mera. (PostgreSQL, Chapter 8. Data Types, u.d.).

### **2.2.1** Funktionalitet

Gällande funktionalitet har PostgreSQL fler avancerade funktioner än SQLite. Det stöder till exempel transaktionshantering fullt ut, vilket innebär att flera databasåtgärder kan grupperas tillsammans i en enda transaktion som garanterar att antingen alla åtgärder genomförs eller inga alls. Detta är särskilt användbart i företagsapplikationer där dataintegritet är av största vikt. PostgreSQL erbjuder också stöd för flera användare, vilket möjliggör samtidig åtkomst till databaser. (Samuel, 2021).

PostgreSQL är strängare än SQLite när det kommer till upprätthållningen av datatyper. Till skillnad från SQLite, kräver PostgreSQL att alla värden som läggs in i en kolumn överenskommer med datatypen för den kolumnen. Om ett värde inte kan omvandlas till den motsvarande datatypen kommer PostgreSQL att ge ett felmeddelande. Utöver inbyggda datatyper stöder PostgreSQL också användar-definierade datatyper. Detta kan vara användbart för att skapa anpassade datatyper som är skräddarsydda för din applikation och domän. En annan viktig skillnad mellan de två databashanteringssystemen är sättet de hanterar dataintegritetsbegränsningar. Även om SQLite stöder några grundläggande dataintegritetsbegränsningar, exempelvis primärnycklar och främmande nycklar, erbjuder PostgreSQL ett mer avancerat stöd för dataintegritet. Bland annat CHECK-begränsningar och UNIQUE- begränsningar. (Tableplus, 2018).

### **2.2.2** Skalbarhet

När det kommer till skalbarhet är PostgreSQL oftast det bättre alternativet för större applikationer som kräver hög tillgänglighet och prestanda. PostgreSQL är känd för sin skalbarhet och kan enkelt skalas upp genom att lägga till flera noder i en klusterkonfiguration. SQLite, å andra sidan, är bäst lämpad för mindre applikationer som inte kräver hög prestanda eller skalbarhet.

Slutligen, beträffande prestanda, gör PostgreSQL generellt sätt bättre ifrån sig än SQLite för större databaser och mer komplicerade frågor, som kallas queries. Detta är dels på grund av PostgreSQL mer avancerade stöd för indexering och query-optimeringstekniker.

Generellt sett erbjuder PostgreSQL stöd för fler datatyper, strängare kontroll av dem, och mera avancerat stöd för dataintegritetsbegränsning och query-optimering. Emellertid kan SQLite vara enklare att konfigurera för mindre databaser eller applikationer med mindre ambitiösa prestandakrav. (Tableplus, 2018).

# 3 Dataöverföring

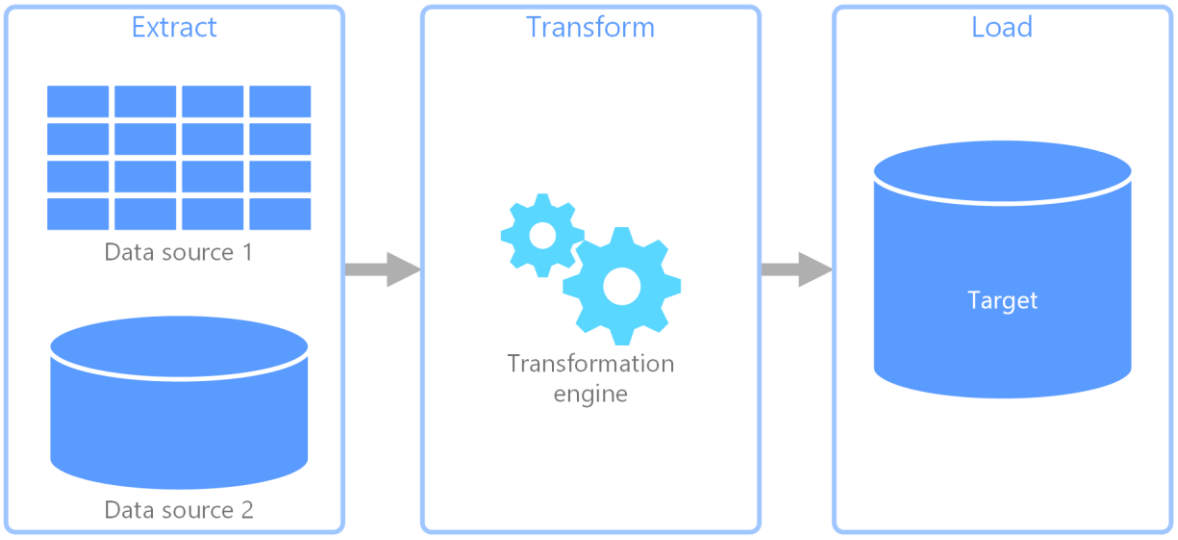
I detta kapitel förklaras dataöverföring, specifikt ETL-processen och dess olika steg. Verktyg som kan användas för att överföra data presenteras också, samt tekniken som användes i det här fallet.

## 3.1 Beskrivning av överföringsprocessen

ETL, vilket står för Extract, Transform, Load, utgör en omfattande process som involverar flera steg för att säkerställa en framgångsrik överföring av data från en befintlig databas till en annan. Den första fasen, extraction (extrahering), innebär att data extraheras från källsystemet eller databasen. Detta inkluderar att identifiera och hämta de relevanta datamängderna som behövs för överföringen. Genom en noggrann analys och planering säkerställs att endast relevanta och nödvändiga data extraheras för att optimera överföringsprocessen.

När data har extraherats går man vidare till transform-fasen. Här genomgår datat en rad omvandlingar och anpassningar för att säkerställa att det blir kompatibelt med det nya formatet och databasstrukturen. Denna omvandling kan innefatta ändringar i datatyper, normalisering av data, rengöring av datafel och eventuella nödvändiga justeringar för att uppfylla kraven för den nya databasen. Genom att tillämpa olika transformationsregler och logik skapas en enhetlig och konsekvent datamodell för överföringen.

Slutligen kommer man till load-fasen, där det transformerade och anpassade data laddas in i den nya databasen. Detta innebär att skapa eller anpassa den nödvändiga databasstrukturen och tabellerna för att rymma det överförda data. Genom att använda ett databashanteringssystem som möter de behov man har för den nya databasen säkerställs att dataintegrationen sker smidigt och effektivt. Här kan det vara viktigt att skapa eller uppdatera referenser, indexering och eventuella nödvändiga dataintegritetskontroller för att garantera att den nya databasen fungerar korrekt och uppfyller behoven för datalagring och informationshantering. (IBM, ETL (Extract, Transform, Load), u.d.).




Figur 2: Bilden beskriver ETL-processen. (Microsoft, Extract, transform, and load (ETL), u.d.).

I detta fall innebär överföringen att cirka 10 000 rader data ska laddas in i den nya databasen. Även om datamängden kanske inte når de enorma volymer som vanligtvis överförs med ETL-processen, kommer de grundläggande principerna och metoderna fortfarande att tillämpas för att säkerställa en korrekt överföring och korrekt anpassning av data till den nya miljön. Med noggrann planering, genomförande och övervakning av ETL-processen kan en lyckad överföring och integration av data uppnås, vilket resulterar i en robust och fungerande databaslösning för det nya systemet.

### **3.2** Verktyg som används för överföringen

Pgloader är ett överföringsverktyg mellan databashanterare som erbjuder flera fördelar jämfört med andra liknande verktyg. En av dess huvudfördelar är dess flexibilitet - det stödjer ett brett utbud av databasformat inklusive CSV, MySQL, SQLite, Oracle och PostgreSQL, vilket gör det enkelt att migrera data från ett format till ett annat.

Pgloader är skapat av Dimitri Fontaine som är en välkänd person i öppen källkod-gemenskapen med över tjugo års erfarenhet av mjukvaruutveckling, arkitektur och administration. Han har bidragit till olika projekt inom öppen källkod som utvecklare, underhållare och mjukvaruarkitekt. (Fontaine, Pgloader, about us, u.d.a).

Utöver dess flexibilitet är Pgloader också utformat för prestanda, med parallell bearbetning och bulkinläsnings-tekniker som minskar överföringstider. Det är ett utvecklat och stabilt

verktyg som har testats omfattande i produktionsmiljöer och aktivt underhålls och uppdateras.

Pgloader har också ett enkelt och intuitivt kommandoradsgränssnitt som gör det lätt att konfigurera och köra dataöverföringar. Det erbjuder detaljerad dokumentation och exempel för att hjälpa användare att komma i gång snabbt, även för användare som kanske inte är bekanta med kommandoradsverktyg.

Slutligen är Pgloader ett öppet källkodsverktyg, vilket innebär att det är fritt tillgängligt att använda, modifiera och distribuera. Detta gör det till ett flexibelt och anpassningsbart alternativ för databasöverföring. (Fontaine, Pgloader, introduction, u.d.b).

Skriptning är ett annat sätt som kan användas för att automatisera ETL-processen. Genom att använda skript kan flera steg inom ETL-processen utföras snabbt och effektivt. Detta är fördelaktigt eftersom processen är väldigt flexibel om man väljer att använda sig av skript.

Det blir enkelt att anpassa sig till förändringar i datakällor eftersom du kan skräddarsy ditt skript efter specifika behov och krav. Användningen av skript ger också en form av reproducerbarhet som är tidigare har varit svårt att uppnå med andra metoder. Samma skript kan köras flera gånger för att extrahera, transformera och ladda in data på ett konsekvent sätt, vilket underlättar felsökning och spårning av eventuella problem. (Giardina, 2021) (Dhopade, 2021).

I dagens läge har dock ETL-processen blivit enklare på grund av företag som erbjuder tjänster för att göra ETL med grafiska gränssnitt, vilket bidrar till en nivå av enkelhet så att man kan skala ned tidskravet av en ETL process med upp till flera dagar. (Dhopade, 2021).

### **3.3** Strategi och teknik för dataöverföring

Överföringen sker via WSL, Windows Subsystem for Linux, eftersom Pgloader har mycket bredare stöd på Unix-plattformar vilket gör det enklare att följa dokumentation och hitta information i den miljön. WSL är utvecklat av Microsoft och det låter programmerare köra en Linux-miljö direkt i Windows utan krånglet med virtuella maskiner. (Microsoft, 2022).

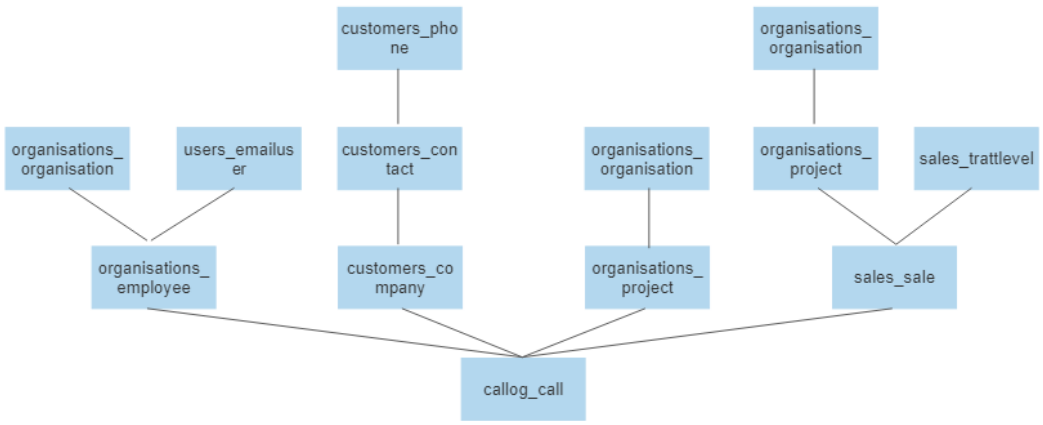
Innan själva överföringen genomfördes, genomfördes en kartläggning av tabellerna som hade främmande nycklar. En omfattande analys utfördes för att skapa ett detaljerat

"tabellträd", vilket visade referenserna mellan tabellerna. Detta tabellträd blev en ovärderlig visuell representation som underlättade förståelsen av det befintliga databasschemat och referenserna mellan olika tabeller.

Tabellträdet var en grafisk representation där varje nod representerade en tabell och linjerna mellan noderna indikerade förekomsten av referenser. Genom att iaktta detta träd kunde man lätt avgöra vilka tabeller som var beroende av varandra.

Denna analys gav insikt i databasens struktur och hjälpte till att identifiera de tabeller som skulle ingå i överföringen. Tabellerna i trädet hade övervakade referenser till andra tabeller, de inkluderades i överföringsprocessen för att säkerställa att integriteten i data bevarades.

Den systematiska undersökningen av tabellträdet gav en klar uppfattning om hur dataflödet och referenserna skulle hanteras under överföringen. Det möjliggjorde en strukturerad planering och säkerställde att alla relevanta tabeller och deras kopplingar togs med i överföringsprocessen.




Figur 1: Tabellträdet som kartlagts för att förenkla processen.

# 4 Genomförande och resultat

PostgreSQL-systemet körs på DigitalOceans molntjänst. Datat överförs från en .sqlite-fil till en PostgreSQL-server för testningssyfte på en lokal maskin och om all data är i ordning så laddas det upp till servern på DigitalOcean.

## 4.1 Genomgång av överföringsprocessen

Överföringsprocessen inleddes genom att schemalägga en tidpunkt när servern skulle tas ner för testning. Den första överföringen utfördes med en konfiguration av verktyget Pgloader som resulterade i att befintliga rader i Postgres-databasen raderades och ersattes med data från SQLite-databasen. Detta var inte önskvärt eftersom det skulle innebära att till exempel användarnas lösenord och användarinformation skulle överföras från en äldre version av databasen.

```
load database
        from 'db.sqlite3'
        into postgresql://doadmin:**********@db-**********-user-**********.ondigitalocean.com>

    WITH include drop, create tables, create indexes, reset sequences

    CAST type num to timestamp

    including only table names like 'calllog_call', 'organisations_employee', 'customers_company', 'organisations_project', 'organisations_organisation', 'users_emailuser', 'customers_contact', 'sales_trattlevel', 'customers_phone'

    set work_mem to '16MB', maintenance_work_mem to '512MB';
```

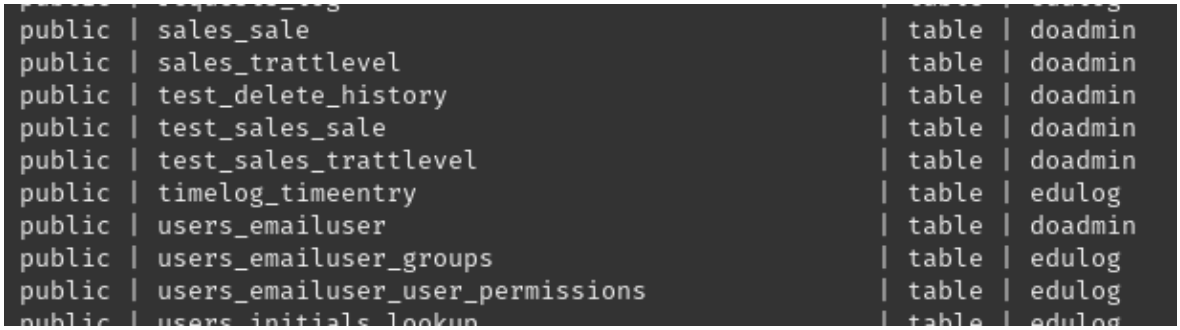
Figur 2: Den första iterationen av Pgloader-konfigurationsfilen.

I figur 3 visas den första versionen av konfigurationsfilen som användes med Pgloader. De första fyra raderna laddar databasen och berättar för verktyget varifrån det ska ta data, i detta fall från filen db.sqlite3 och köra det till adressen på DigitalOcean. Adressen har delvis gömts på grund av att detta är konfidentiell information.

På rad sex och åtta berättar konfigurationsfilen åt Pgloader hur data som importeras ska modifieras. Exempelvis ändrar man SQLites datatyp num till PostgreSQLs timestamp som i detta fall motsvarar varandra. På rad nio till tolv berättar man åt verktyget vilka tabeller som ska tas med i överföringen. Den sista raden är oviktig i detta fall eftersom den endast begränsar hur mycket minne Pgloader ska få använda i överföringen.

Det noterades också att om Pgloader-konfigurationen ändras så att verktyget inte tar bort befintliga tabeller och skapar om dem kan det leda till problem med referenser mellan tabellerna, vilket gör databasen oanvändbar.

Dessutom upptäcktes det att tabellerna i Postgres hade tilldelats fel ägare. Ägaren var doadmin i stället för edulog, vilket är den korrekta ägaren i detta fall. Detta fel uppstod på grund av ett konfigurationsfel i Pgloader. Även om det var en mindre fråga var det viktigt att rätta till det för att säkerställa noggrannheten och integriteten i data i den nya databasen.




Figur 3: Skärmbild som visar tabellernas ägare efter överföringen.

Efter den första överföringen togs beslutet att genomföra en backupåterställning och försöka utföra överföringen på nytt. Den andra överföringen genomfördes efter att överföringsverktyget hade konfigurerats. Resultatet var något bättre då all data laddades in i PostgreSQL-databasen. Dock uppstod ett problem när man försökte starta den webbserver som använder databasen.

Problemet härledde sig till felaktiga referenser mellan data i tabellerna. På grund av att felsökningsläget på webbservern var avstängd kunde man inte fastställa vad orsaken var exakt. Detta skulle eventuellt kunna lösas med komplexa skript och korrigeringar. Men eftersom Edupower sökte efter en enklare lösning som skulle vara genomförbar flera gånger, valde man att inte fortsätta med denna möjlighet.

### **4.2** Analys av överföringens prestanda och skalbarhet

Överföringens effektivitet kan ändå ses som väldigt positiv, med de knappa 10 000 raderna data som fördes över tog det Pgloader en dryg tiondels sekund att överföra alla rader. Detta visar på att lösningen skulle kunna användas i stor skala så länge att man korrigerar referenser mellan tabeller efter att man har importerat data.

| | | | | |
|---|---|---|---|---|
| users_emailuser | 1 | 0 | | 0.010s |
| organisations_employee | 1 | 0 | | 0.010s |
| customers_phone | 0 | 4 | 0.1 kB | 0.000s |
| customers_contact | 0 | 4 | 0.1 kB | 0.010s |
| organisations_organisation | 1 | 0 | | 0.000s |
| sales_trattlevel | 0 | 25 | 0.5 kB | 0.010s |
| customers_company | 0 | 3 | 0.1 kB | 0.010s |
| organisations_project | 1 | 0 | | 0.010s |
| calllog_call | 0 | 9416 | 2.0 MB | 0.100s |
| sales_sale | 1 | 0 | | 0.010s |
| COPY Threads Completion | 0 | 4 | | 0.110s |
| Reset Sequences | 0 | 9 | | 0.010s |
| Create Foreign Keys | 3 | 8 | | 0.010s |
| Install Comments | 0 | 0 | | 0.000s |
| Total import time | 5 | 9452 | 2.0 MB | 0.130s |

Figur 4: Skärmbild som visar prestandat av överföringsverktyget.

### **4.3** Jämförelse av data före och efter överföringen

Det är viktigt att notera att överföringen inte var problemfri. Efter att ha undersökt datat manuellt upptäckte man att det fanns felaktiga referenser mellan tabellerna. Detta betyder att kopplingarna mellan olika datavärden i databasen inte var korrekta. När referensema inte är korrekta kan det leda till problem vid överföring och integration av datat i webbservern. Komplexa skript och korrigeringar skulle ha krävts för att lösa problemet med felaktiga referenser mellan data och tabeller som påverkade starten av webbservern.

# 5 Slutsats

Detta kapitel innehåller en sammanfattning på överföringen och en diskussion kring utmaningarna i detta projekt. Slutsatsen bidrar till att ge en överblick över resultatet av arbetet.

## 5.1 Sammanfattning av överföringsprocessen

Sammanfattningsvis visar denna studie att överföringsprocessen från SQLite till PostgreSQL innebar utmaningar som att hantera befintliga rader, korrekta ägarinställningar och felaktiga referenser. För att framgångsrikt överföra data mellan dessa databassystem krävs noggrann planering och implementering av lämpliga åtgärder för att upprätthålla datakvalitet och integritet i den nya databasen.

## 5.2 Diskussion

En av de främsta utmaningarna som identifierades under överföringen var hanteringen av befintliga rader i Postgres-databasen. Vid den första överföringen, genomförd med hjälp av Pgloader-verktyget, raderades de befintliga raderna och ersattes med data från SQLite-databasen. Detta resulterade i att äldre versioner av data, inklusive användarnas lösenord och användarinformation, överfördes. Detta var inte önskvärt och ledde till problem med datakvalitet och säkerhet i den nya databasen.

En annan utmaning som uppkom under överföringsprocessen var hanteringen av referenser mellan tabellerna. Ändringar i Pgloader-konfigurationen kunde leda till problem med referenser, vilket gjorde den nya databasen oanvändbar om tabellerna inte skapades om korrekt. Detta visar vikten av att noggrant analysera och förstå datamodellen och referenserna mellan tabellerna innan överföringen genomförs.

Vidare identifierades en viktig aspekt av överföringsprocessen i form av felaktiga ägarinställningar för tabellerna i Postgres-databasen. I stället för att ha rätt ägare edulog var ägaren inställd som doadmin på grund av en konfigurationsmiss i Pgloader. Även om det kan tyckas vara en mindre fråga, har ägarinställningar en betydande roll för att upprätthålla integriteten och hanteringen av databasen. Korrigeringen av ägarinställningarna var nödvändig för att säkerställa att rättigheter och behörigheter hanterades korrekt.

Efter att ha genomfört försöken med överföring av data insåg man att felsökningsläget på webbservern hade varit avstängt under dessa överföringar. Detta innebar att man inte kunde undersöka vad som orsakade problemet med att webbservern inte startade med det nya datat i databasen. Denna avstängning av felsökningsläget försvårade processen att fastställa den exakta orsaken till problemet. Det var delvis på grund av det här som försöken stoppades och man valde att utforska andra alternativ för överföringen.

Efter den första överföringen och identifieringen av dessa utmaningar beslutades det att genomföra en backupåterställning och försöka överföringen på nytt. Vid den andra överföringen, efter att ha konfigurerat överföringsverktyget på lämpligt sätt, laddades all data korrekt in i PostgreSQL-databasen. Trots framgången med datalastningen uppstod ett problem vid starten av webbservern som använder sig av databasen. Problemet härledde sig till felaktiga referenser mellan data i tabellerna, vilket krävde komplexa skript och korrigeringar för att lösa. Med tanke på att Edupower sökte efter en enklare lösning som skulle vara genomförbar vid flera tillfällen, valde man att inte fortsätta med dessa åtgärder.

## **5.3** Fortsatt forskning

Det finns definitivt utrymme för fortsatt forskning och utforskning av ämnet. Med tillräckligt med tid och resurser skulle det vara möjligt att hitta en helautomatisk lösning för överföringen genom användning av skript och andra avancerade verktyg. Genom att vidareutveckla och anpassa konfigurationen av Pgloader-verktyget skulle det vara möjligt att säkerställa bevarandet av referenser och korrekt formatering av data.

För att lösa dessa frågor kan det vara fördelaktigt att söka hjälp och stöd från utvecklarna av Pgloader och gemenskapen av användare. Genom att använda resurser som deras GitHub-sida eller till och med genom att kontakta verktygets skapare via e-post skulle man kunna be om råd och vägledning för att förbättra överföringsprocessen och lösa de specifika problemen med referenserna och data.

Genom att involvera experter och utvecklare inom området skulle man kunna dra nytta av deras insikter och erfarenheter för att finna mer avancerade lösningar och möjliga förbättringar. Genom att aktivt delta i diskussioner och dela problemen man stöter på kunde man dra nytta av gemenskapens kunskap och dra fördel av deras expertis för att utforska olika vägar och alternativ för att förbättra överföringsprocessen.

# 6 Litteraturförteckning


Cassel, D. (11.7.2021). *The news stack, Origin story of SQLite*. Hämtat från The news stack-webbsida: https://thenewstack.io/the-origin-story-of-sqlite-the-worlds-most-widely-used-database-software/

Dhopade, K. (24.5.2021). *LinkedIn, Automated ETL vs Manual Coding : Which is better choice in Data Management?* Hämtat från LinkedIn-webbsida: https://www.linkedin.com/pulse/automated-etl-vs-manual-coding-which-better-choice-data-dhopade/

Fontaine, D. (u.d.a). *Pgloader, about us*. Hämtat från Pgloader-webbplats: https://pgloader.io/about/

Fontaine, D. (u.d.b). *Pgloader, introduction*. Hämtat från Pgloader: https://pgloader.readthedocs.io/en/latest/intro.html

Fontaine, D. (u.d.c). *Pgloader, SQLite to PostgreSQL*. Hämtat från Pgloader: https://pgloader.readthedocs.io/en/latest/ref/sqlite.html

Giardina, C. (16.12.2021). *Airbyte, Using an ETL Framework vs Writing Yet Another ETL Script*. Hämtat från Airbyte-webbsida: https://airbyte.com/blog/etl-framework-vs-etl-script

*IBM, ETL (Extract, Transform, Load)*. (u.d.). Hämtat från IBM-webbsida: https://www.ibm.com/topics/etl

Kuhn, W. (7.5.2018). *Medium, A Brief History of PostgreSQL*. Hämtat från Medium: https://medium.com/launch-school/a-brief-history-of-postgresql-36d8d392c611

Microsoft. (8.12.2022). *What is the Windows Subsystem for Linux?* Hämtat från Microsoft-webbplats: https://learn.microsoft.com/en-us/windows/wsl/about

*Microsoft, Extract, transform, and load (ETL)*. (u.d.). Hämtat från Microsoft-webbplats: https://learn.microsoft.com/en-us/azure/architecture/data-guide/relational-data/etl

Nandini. (14.3.2023). *Data Migration Testing Tutorial: A Complete Guide*. Hämtat från Software Testing Help-webbplats: https://www.softwaretestinghelp.com/data-migration-testing/

*PostgreSQL*. (u.d.). Hämtat från Wikipedia-sida: https://en.wikipedia.org/wiki/PostgreSQL

*PostgreSQL, A brief history of PostgreSQL*. (u.d.). Hämtat från PostgreSQL: https://www.postgresql.org/docs/current/history.html

*PostgreSQL, Chapter 8. Data Types*. (u.d.). Hämtat från PostgreSQL: https://www.postgresql.org/docs/current/datatype.html

Samuel, N. (18.5.2021). *Hevo*. Hämtat från Hevo Data-webbplats: https://hevodata.com/learn/sqlite-vs-postgresql/

SQLite. (16.12.2022). *Appropriate Uses For SQLite*. Hämtat från SQLite-webbplats: https://www.sqlite.org/whentouse.html

SQLite. (u.d.). *SQLite, About SQLite*. Hämtat från SQLite: https://sqlite.org/about.html

*SQLite: Transaction*. (2023). Hämtat från SQLite-webbplats: https://www.sqlite.org/lang_transaction.html

Tableplus. (30.8.2018). *Tableplus, SQLite vs PostgreSQL - Which database to use and why?* Hämtat från Tableplus-webbplats: https://tableplus.com/blog/2018/08/sqlite-vs-postgresql-which-database-to-use-and-why.html

*W3schools, History of PostgreSQL*. (u.d.). Hämtat från W3schools: https://www.w3schools.blog/history-postgresql

Westerlund, K. (u.d.). *Edupower*. Hämtat från Edupower: https://edupower.fi/sv/foretaget/