YRKESHÖGSKOLAN
NOVIA

# Visualisering av realtidsdata i Grafana

Casper Mattsson

Examensarbete för ingenjör (YH)-examen

El- och automationsteknik

Vasa 2024

**EXAMENSARBETE**

Författare: Casper Mattsson
Utbildning och ort: El- och automationsteknik, Vasa
Inriktning: Informationsteknik
Handledare: Jan Berglund - Novia, Adam Mattsson - Optinova

Titel: Visualisering av realtidsdata i Grafana

---

Datum: 23.5.2024 Sidantal: 25

---

**Abstrakt**

Detta examensarbete har utförts på uppdrag av och i samarbete med Optinova. Syftet har varit att visualisera realtidsdata från extruderingsprocessen i Grafana samt utforska möjligheter och begränsningar med Grafana för detta ändamål.

Arbetet består av en teoretisk och en praktisk del. Den teoretiska delen ger en inblick i hur databaser, fram för allt relationsdatabaser, och databashanteringssystemen som används i detta arbete fungerar. Vidare behandlas visualiseringsverktyg, huvudsakligen Grafana, samt affärssystemet Monitor.

Genom diskussioner och möten med målgrupperna kunde förslagen på visualisering av realtidsdata konstateras och möjligheterna med Grafana kunde utforskas därefter.

Under processen att visualisera realtidsdata genom att utveckla databasfrågor för att hämta data från databaserna upptäcktes vissa begränsningar med Grafana för detta ändåmål. Grafana stöder inte integration mot databashanteringssystemet Sybase och därav kunde inte data hämtas från en av databaserna. Det är heller inte möjligt att kombinera data från flera än två databaser om de har olika nycklar. Detta innebär att möjligheterna med att använda Grafana för att visualisera realtidsdata från extruderingsprocessen inte kunde utforskas fullt ut.

Resultatet är flera instrumentpaneler som innehåller visualiseringar av processdata och utnyttjandegrad i realtid. Det ger en god grund till eventuell fortsatt utveckling.

---

Språk: svenska
Nyckelord: Grafana, IoT, PostgreSQL, MSSQL, visualisering

**BACHELOR'S THESIS**

Author: Casper Mattsson
Degree Programme: Electrical and Automation Engineering
Specialisation: Information Technology
Supervisor(s): Jan Berglund - Novia, Adam Mattsson - Optinova

Title: Real-time data visualization in Grafana

---

Date: 23.5.2024 Number of pages: 25

---

**Abstract**

This thesis was conducted at the request of and in collaboration with Optinova. The purpose of this thesis was to visualize real-time data from the extrusion process in Grafana and explore possibilities and limitations in using Grafana for this purpose.

The work consists of a theoretical part and a practical part. The theoretical part provides an insight into databases, particularly relational databases, and their functionality and the database management systems used in this work. Furthermore, visualization tools are covered, primarily Grafana, and the ERP system Monitor.

Through discussions and meetings with the target groups the suggestions for visualization of real-time data could be determined and the possibilities in Grafana could be explored thereafter.

During the process of visualizing real-time data through the development of database queries for gathering data from the databases, limitations with Grafana were found. Grafana does not support Sybase as a data source and therefore data could not be fetched from one database. It also is not possible to combine data from more than two databases if they have different keys. This meant that the possibilities in using Grafana to visualize real-time data from the extrusion process could not be fully explored.

The result of this work are several dashboards with visualizations of process data and utilization rate in real-time. There is a good foundation for possible continued development.

---

Language: Swedish
Key words: Grafana, IoT, PostgreSQL, MSSQL, visualization

# Innehållsförteckning

# 1 Inledning

Detta examensarbete är utfört på uppdrag av och i samarbete med Optinova. Arbetet handlar om att presentera realtidsdata som samlas in från extruderingsprocessen och presentera den i ett lämpligt visualiseringsverktyg. Detta för att ge de olika målgrupperna en bättre översikt över de olika parametrarna i realtid och om de håller sig inom lämpliga intervall. För detta arbete valdes Grafana av uppdragsgivaren.

## 1.1 Uppdragsgivare

Optinova grundades 1971 och är ett världsledande företag inom extrudering av avancerade medicinska och industriella slangar. Med försäljningskontor runt om i världen och fyra extruderingsanläggningar i Finland, Thailand och USA, betjänar de partners från över 50 länder över olika branscher. (About Optinova). I tillverkningen av plastslangar är kvalitet avgörande, därför har Optinova under de senaste åren börjat implementera Industri 4.0-teknik som ett sätt att analysera tillverkningsprocessen genom datainsamling.

## 1.2 Syfte

Syftet med detta examensarbete var att visualisera realtidsata från extruderingsprocessen i ett lämpligt visualiseringsverktyg, i detta fall Grafana, och att undersöka möjligheterna och begränsningarna att kombinera data från flera olika databaser i Grafana. Datakällorna är:

- Monitor, Sybase
- Monitor MI, PostgreSQL
- Elektroniskt körprotokoll, MSSQL
- Monitor MI OPC, PostgreSQL

För att genomföra detta behövde först de olika användarnas krav undersökas för att förstå behoven kring visualisering av realtidsdata, detta diskuterades i möten med bland andra CTO och produktionsledare.

## 1.3 Mål

Användarna ska ha en översikt över de olika parametrarna baserat på deras förslag. För linjerna behöver utnyttjandegraden och processdata presenteras. För fabriken behöver utnyttjandegraden presenteras. Eftersom det handlar om att åskådliggöra data är det viktigt att visualiseringen är intuitiv och man ska enkelt kunna tolka de olika värdena, med hjälp av den grafiska designen. Detta arbete ska även göra det möjligt att presentera ovan på en delad skärm eller TV utan krav på inloggning eller licens.

# 2 Teori

I detta kapitel behandlas teorin bakom databaser, framför allt relationsdatabaser. Även SQL som frågespråk, olika databashanteringssystem och visualiseringsverktyg samt affärssystemet Monitor behandlas i detta kapitel.

## 2.1 Databaser

Med data avser man information som samlas in om till exempel en person, husdjur, dator och dess attribut. En databas är en organiserad samling av denna information. Ett exempel på detta i vardagslivet kan vara att du har skrivit ner olika produkter som du vill köpa, då kommer produkternas namn, pris och så vidare vara en analog databas. Den digitala motsvarigheten blir att produkterna är en tabell där namn och pris är olika kolumner och produkternas namn är varje rad, detta lagras sedan i ett datorsystem. (What are databases?).

Det finns flera olika typer av databaser, till exempel: relationsdatabaser, objektorienterade databaser, tidsseriedatabaser, NoSQL-databaser och molndatabaser. Den vanligaste typen av databaser är relationsdatabaser. (What is a Database?).

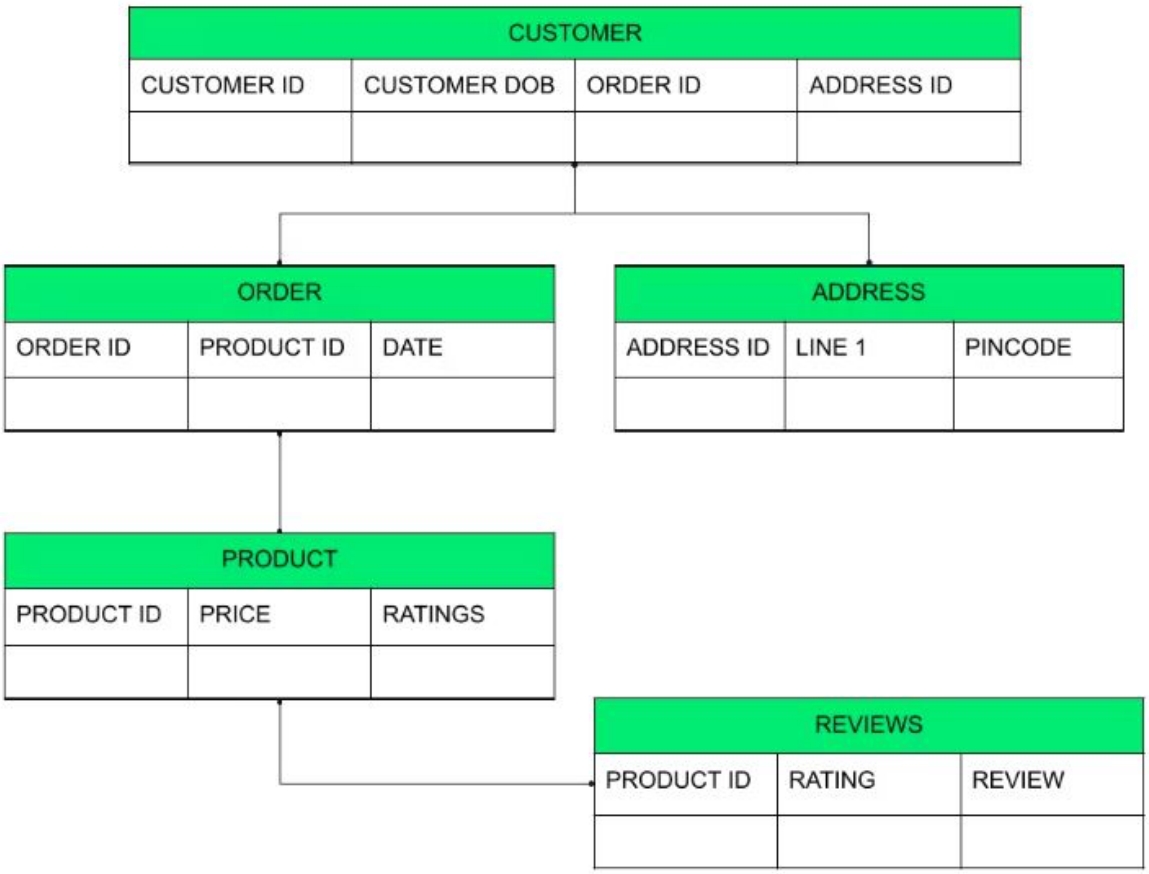
### 2.1.1 Relationsdatabaser

Relationsdatabaser härstammar från relationsmodellen, som Edgar Frank Codd utvecklade i sin uppsats A Relational Model of Data for Large Shared Data Banks 1970. (Relational database, 2024). Det Codd föreslog var att i stället för att organisera data enligt en hierarkisk struktur kunde man i stället byta till en struktur där data organiseras i tabeller, bestående av rader och kolumner. Dessa rader har ett unikt ID kallat primärnyckel och det är via denna primärnyckel som man kan kombinera data från två olika tabeller. Om man hänvisar till denna primärnyckel från en annan tabell kallas det för främmande nyckel. (What is a relational database?).

Om man tar det tidigare exemplet om produkter kan man föreställa sig att man har en tabell för kunder med kolumnrubrikerna: kund-ID, namn och adress och en tabell för produkter med kolumnrubrikerna: produkt-ID, produktnamn, pris och kund-ID. I tabellen för kunder är kund-ID primärnyckeln som är unik för varje kund och i tabellen för produkter

är produkt-ID primärnyckeln som är unik för varje produkt. Genom att använda kund-ID som en främmande nyckel i produkttabellen kan man koppla samman tabellerna och ange vilken produkt som har beställts av vilken kund. Detta är ett förenklat exempel men med utökade tabeller kan man snabbt samla in stora mängder information, till exempel hur många kunder som gjort en beställning enligt datum och om de har fått beställningen i tid. (What is a Relational Database (RDBMS)?).

Här är ett annat exempel på en koppling mellan flera tabeller:




*Figur 1. Koppling mellan flera tabeller.* (Relational vs. Non-Relational Databases)**.**

## 2.2 SQL

Som en följd av Edgar Frank Codds teorier började Donald D. Chamberlin och Raymond F. Boyce under tidigt 1970-tal utveckla ett programmeringsspråk för att modifiera och hämta data från IBM:s databashanteringssystem System R. Chamberlin och Boyce ansåg att det borde vara möjligt att skapa ett programmeringsspråk baserat på Codds matematiska teorier. Genom att ersätta den matematiska notationen med nyckelord kunde de göra det

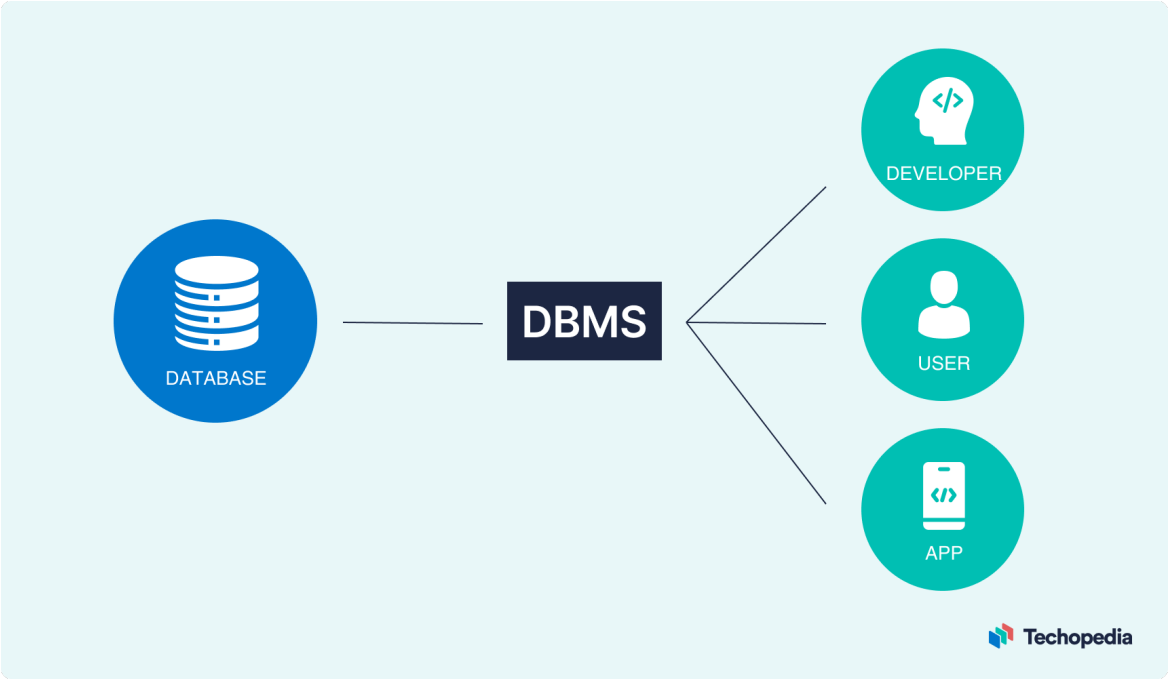
dels enklare att skriva, dels enklare att förstå för allmänheten. Initialt kallades det SEQUEL men bytte senare namn till SQL som står för Structured Query Language. (Chamberlin, 2012).

SQL kan delas in i olika dialekter eller varianter vilket relationella databashanteringssystem använder sig av. Till exempel kallas dialekten för PostgreSQL PL/pgSQL och varianten för Microsoft SQL Server Transact-SQL. Dessa dialekter har unika fördelar såsom PostgreSQL:s tillägg PostGIS som gör det möjligt att hantera och analysera stora mängder geografiska data vid stadsplanering. (Sardar, 2023).

## 2.3 Databashanteringssystem

För att kunna komma åt, hantera och modifiera data i databaser behövs databashanteringssystem, DMBS, som fungerar som ett gränssnitt mellan databasen och användaren, utvecklaren, administratören eller applikationen. Genom att hantera data, databasmotorn och databasschemat kan dessa databashanteringssystem tillhandahålla dataintegritet, dataskydd och förhindra dataförlust. (What is Database Management Systems (DBMS)?).

Databashanteringssystem började utvecklas redan då datorer började användas för databehandling under 1960-talet. Charles W. Bachman och hans team på General Electric utvecklade det första kommersiella databashanteringssystemet i slutet av 1960-talet och det kallades IDS, Integrated Data Store. Precis som med databaser finns det en rad olika databashanteringssystem. Man delar ofta upp dem i relationella och icke-relationella databashanteringssystem och därtill finns det ytterligare indelningar beroende på fördelarna med dem. (Rouse, 2024).




*Figur 2. DBMS som mellanprogram.* (Rouse, 2024).

### 2.3.1 PostgreSQL

PostgreSQL är ett objektrelationellt databashanteringssystem, ORDBMS, som började utvecklas 1986 på University of California, Berkeley. Likt relationella databashanteringssystem bygger det på relationer mellan tabeller men också funktionaliteten hos objektorienterade databashanteringssystem, vilket tillåter en större variation i datatyper som kan hanteras. PostgreSQL är idag den mest avancerade databashanteraren som är baserad på öppen källkod. Fram tills 1994 användes QUEL som programmeringsspråk men byttes sedan till SQL 1995 då Postgres95 släpptes. 1996 byttes namnet till PostgreSQL som det även kallas idag. (Kuhn, 2018).

PostgreSQL började som ett forskningsprojekt som kallades POSTGRES (Post-Ingres). Projektet leddes av professorn Michael Stonebraker som ansåg att det fanns förbättringsmöjligheter för databashanteringssystemet Ingres men att dessa funktionaliteter skulle vara för svåra att implementera i det existerande systemet och började därför utveckla ett nytt databashanteringssystem. (A Brief History of PostgreSQL).

### 2.3.2 Microsoft SQL Server

Microsoft SQL Server, MSSQL, är ett relationellt databashanteringssystem, RDBMS, och det har utvecklats av Microsoft. MSSQL använder SQL som frågespråk och varianten av SQL som används kallas för Transact-SQL. Första utgåvan lanserades 1989 och idag finns det åtta versioner som stöds av Microsoft. (Microsoft SQL Server, 2024).

En fördel med MSSQL är dess datalager och analytiska förmåga. Genom att kombinera det med SQL Server Analysis Services och Power BI kan man genomföra komplexa dataanalyser som används till exempel till medicinsk forskning genom att analysera patientdata. (Sardar, 2023).

## 2.4 Visualiseringsverktyg

Idag används stora mängder data och det är viktigt att kunna analysera och visualisera den för att lättare se mönster och trender och baserat på det fatta beslut. I stället för att titta på stora tabeller med data kan man enklare identifiera mönster och förstå data med hjälp av visualiseringsverktyg. Visualiseringsverktyg kan använda data från databaser och presentera den grafiskt, detta kan göras med till exempel olika diagram såsom paj- eller stapeldiagram som sedan kan ändras efter behov. (Data visualisation beginner's guide: a definition, examples and learning resources).

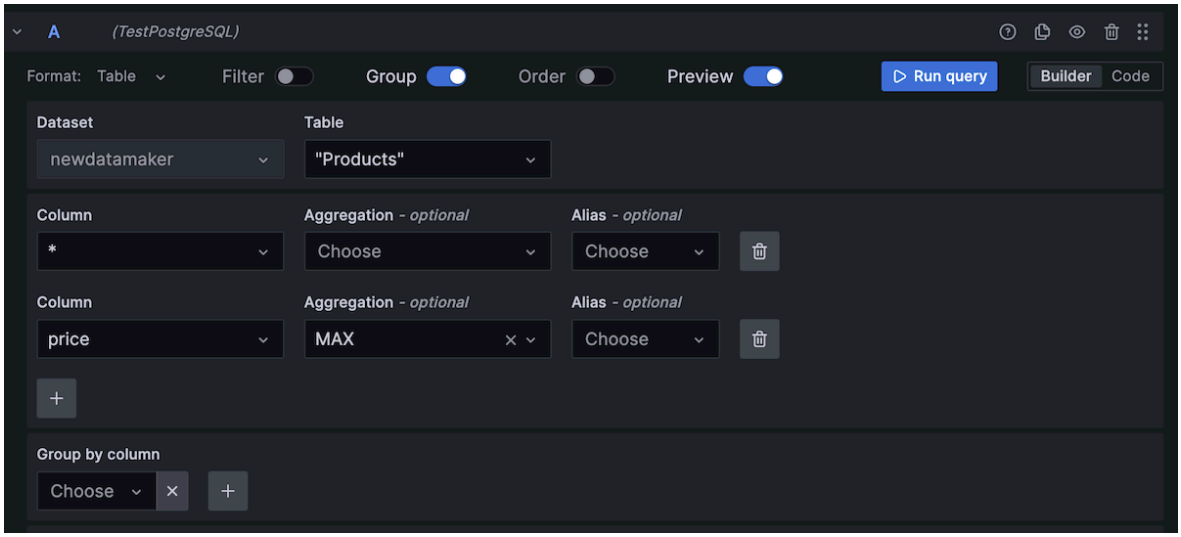
Genom att använda olika färger och rätt typer av grafer kan man lyfta fram relevant information vilket gör att åskådaren snabbt kan särskilja och tolka olika data.

### 2.4.1 Grafana

Grafana är en webbapplikation för analys och interaktiv visualisering som lanserades första gången 2014 av Torkel Ödegaard och den är baserad på öppen källkod. Grafana möjliggör skapandet av diagram, grafer och larm när den är ansluten till kompatibla datakällor. I början låg fokuset på tidsseriedatabaser såsom InfluxDb och Prometheus men utvecklades sedan till att stödja relationsdatabaser såsom Microsoft SQL Server och PostgreSQL. (Grafana, 2024).

Det är möjligt i Grafana att använda deras *builder* för att skapa databasfrågor genom att hämta olika tabeller och kolumner via deras användargränssnitt men man kan även skriva

egna databasfrågor i *code*. Att använda *builder* fungerar bra då man gör enklare databasfrågor men vid mera komplicerade databasfrågor med olika villkor och så vidare är det bäst att skriva egna databasfrågor. Nedan visas ett exempel på hur *builder* ser ut för PostgreSQL.




*Figur 3. Grafanas builder för enklare databasfrågor.* (PostgreSQL data source).

I Grafana kan man skapa instrumentpaneler som fungerar som en översikt, i instrumentpanelen kan man sedan lägga till mindre paneler där data från datakällorna visualiseras. Till dessa visualiseringar kan man även skapa larm som notifierar användaren, exempelvis via e-post, om värdena hamnar utanför ett visst intervall som man kan definiera i regler. (MetricFire, 2023).

Det är möjligt att kombinera flera olika datakällor och eftersom det är en webbapplikation behövs det bara en webblänk för att titta på visualiseringen, utan krav på inloggning eller licens, vilket gör det passligt att ha på en delad skärm eller TV. Detta är förutsatt att Grafana är installerat på en server och att man är ansluten till domännätverket.

### 2.4.2 Power BI

Power BI är ett analysverktyg som kan användas för visualisering och det har utvecklats av Microsoft. Det konceptualiserades av Thierry D'Hers och Amir Netz och designades sedan av Ron George 2010, då med namnet Project Crescent. Microsoft presenterade detta som Power BI i september 2013, som en del av Office 365, och det släpptes för allmänheten 2015. (Microsoft Power BI, 2024).

Det finns flera versioner av Power BI såsom Power BI Desktop, Pro, Mobile, Premium. Power BI finns i olika prisklasser och som kostnadsfri version, den kostnadsfria har dock begränsad funktionalitet. Power BI stöder olika datakällor såsom Excel, PDF, JSON, Microsoft SQL Server och PostgreSQL. (Biswal, 2023).

För att titta på rapporter som är skapade med Power BI behöver man en licens. Alternativt kan man skriva ut dem vartefter eller ladda upp dem på nätet, dock kan vem som helst titta på dem då. (Publish to web from Power BI).

## 2.5 Monitor ERP System

Monitor är ett affärssystem som grundades 1974 i Hudiksvall och det används dagligen av över 5500 företag och är översatt till 14 språk. (Om Monitor ERP).

ERP står för Enterprise Resource Planning och det är ett affärssystem som hjälper företag att sköta och hantera olika affärsprocesser såsom tillverkning, försäljning och bokföring. ERP-system började utvecklas redan under 1960-talet men då som enkla program, Inventory Control-program, för att underlätta företagsledning. (Vad är ERP?).

Tidigare var ERP-system ofta uppbyggda som separat programvara utan koppling till andra system, idag är de dock integrerade med flera områden och fungerar som ett enda lättarbetat system.

Monitor består av sex moduler:

- Tillverkningsmodul
- Försäljningsmodul
- Inköpsmodul
- Lagermodul
- Tidsrapporteringsmodul
- Bokföringsmodul

I dessa moduler kan man göra diverse arbetsuppgifter, exempelvis registrera hur länge det har tagit att tillverka en order, behandla fakturor och lägga upp artiklar.

### 2.5.1 Monitor MI

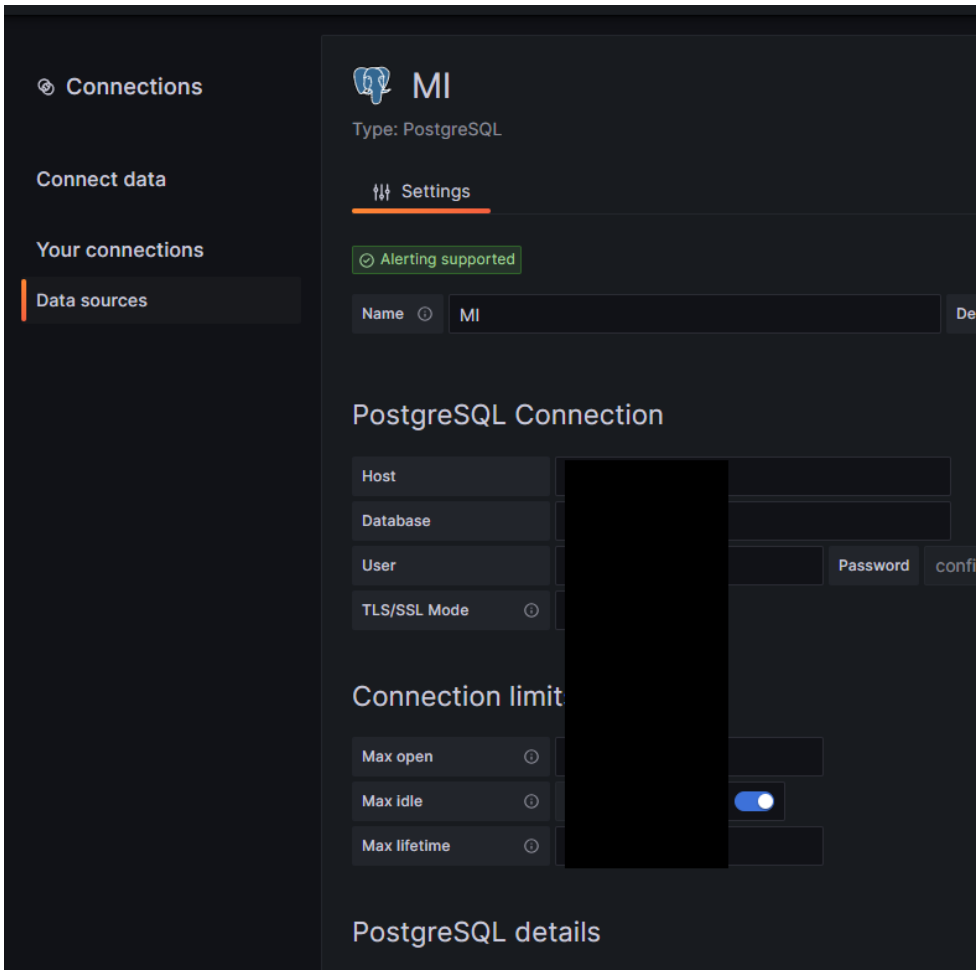
MI står för maskinintegration och det innebär att man integrerar maskiner med Monitor ERP System. På varje maskin kopplas en PLC eller OPC som tolkar signalen från maskinen och rapporterar därefter till Monitor. OPC möjliggör insamlingen av parametrar såsom tryck, ovalitet och yttre diameter. Maskinintegration ger fördelar såsom möjligheten att samla in data i realtid, mindre manuell rapportering för operatörerna och förbättrad maskinbevakning. (Maskinintegration).

# 3 Genomförande

På uppdrag av Optinova valdes Grafana som visualiseringsverktyg för att undersöka möjligheterna med att använda Grafana till att visualisera realtidsdata från extruderingsprocessen. Före själva arbetet i Grafana kunde påbörjas behövde först vissa konfigurationer vara gjorda:

- Monitor MI anslutet och konfigurerat mot en extruder för insamling av processdata.
- Grafana installerat på en server.
- Systemkonton skapade för att läsa data från databaserna.
- Nätverkskonfigurering mellan Grafana och databaserna.

Under tiden detta gjordes hölls möten och diskussioner för att reda ut vad som är relevant att presentera gällande processdata och utnyttjandegrad.




*Figur 4. Konfigurering av databaskoppling i Grafana.*

## 3.1 Användningsfall

Efter diskussioner och möten skrevs användningsfall för att förtydliga vad målgrupperna tänker sig. Man kan dela upp målgrupperna i operatörer och produktionsledning. Användningsfallen skrevs på detta sätt:

- Målgrupp:
    - Operatör
- Krav:
    - Som operatör vill jag se att extruderingsprocessen följer kraven på parametrarna.
    - Som operatör vill jag bli notifierad om parametrarna hamnar utanför kraven.
- Genomförande:
    - Se antal producerande enheter.
    - Se aktuell hastighet jämfört med planerad hastighet.
    - Se yttre diameter och ovalitet under en tidsperiod jämfört med min- och maxgränser.
    - Notis om hastigheten underskrider 75% planerad hastighet.
    - Planerad tid < förbrukad tid + tid kvar på ordern = gult.
    - Förbrukad tid + tid kvar > 3 standardavvikelser mot planerat = rött.
    - Om en parameter > 3 standardavvikelser utanför medel, varna ”Det här är mycket större/mindre än normalt. Kontrollera att allt är rätt.”.

- Målgrupp:
    - Produktionsledning

- Krav:
    - Som produktionsledare vill jag se utnyttjandegraden för linjerna.
- Genomförande:
    - Medelvärde och konfidensintervall för alla uppmätta dimensioner.
    - Utnyttjandegrad för linjerna, kombinerat och var för sig.
    - Utnyttjandegrad som pajdiagram.

## 3.2 Händelseförlopp

I figur 5 kan man se hur det går till för att kunna visualisera realtidsdata från extruderingsprocessen i Grafana. PLC och OPC är kopplat till en extruder, Monitor MI är anslutet och konfigurerat mot en extruder för att samla in processdata och i Grafana hämtar man sedan data från olika databaser och visualiserar den.


*Figur 5. Händelseförlopp.*

## 3.3 Tillvägagångssätt

Som det nämndes tidigare i kapitlet behövde först kraven undersökas, när det fanns en tydlig uppfattning av det som behöver göras kunde arbetet påbörjas i Grafana. Excel-filer som innehöll databasernas tabeller med exempeldata i kolumnerna försågs av handledaren. Genom att studera filerna kunde man förstå vilka tabeller och kolumner som kommer behövas till de olika visualiseringarna och eventuella kopplingar mellan databaserna. Det var också viktigt att kontrollerna maskinernas ID och deras namn för att säkerställa att rätt maskiner visas upp då utnyttjandegrad ska presenteras.

Först skapades en instrumentpanel för testning, där testades det att hämta data samt användningen av olika visualiseringar. Grafanas dokumentation lästes genom för att ge en inblick i hur visualiseringarna fungerar och vad det finns för alternativ då man ska presentera data. Efter att ha testat att hämta data som är relevant för att presentera utnyttjandegraden skapades instrumentpanelen där visualiseringarna för utnyttjandegraden ska placeras.

Enligt användningsfallen i 3.1 vill målgrupperna se utnyttjandegraden i pajdiagram och därmed lästes Grafanas dokumentation för pajdiagram samt testades hur man kan presentera den på bästa sätt. (Pie chart).

# 4 Resultat

Grafana stöder inte integration mot Sybase, därav kan man inte hämta data från en av databaserna. Följden av det är att det inte går att få med till exempel avrapporterat antal jämfört med planerat antal. Det är inte heller möjligt att kombinera data från flera än två databaser med olika nycklar och därför kunde inte min- och maxvärden kombineras med realtidsdata för en aktiv order i samma graf. Det innebär också att det inte gick att få in larmregler om parametrarna hamnar utanför de min- och maxgränserna.

## 4.1 Utnyttjandegrad

Utnyttjandegraden kunde dock presenteras enligt de krav som förmedlades. Genom att dela upp stopporsakernas koder i tre olika kategorier, stopp, ställ, ingen produktion och annars produktion, kan man med koden i kodexempel 1 få fram hur många timmar maskinerna har haft på vardera kategorin under dagens datum. Samma kod används för att få fram dagarna bakåt i tiden med undantaget att man lägger till *current_date - integer '1'* upp till - *integer '6'* vilket är specifikt för PostgreSQL. Man kan också se att varaktigheten behöver delas med 36 000 000 000 för att få fram tiden i timmar, alltså lagras tiden i databasen i tiondels mikrosekunder.

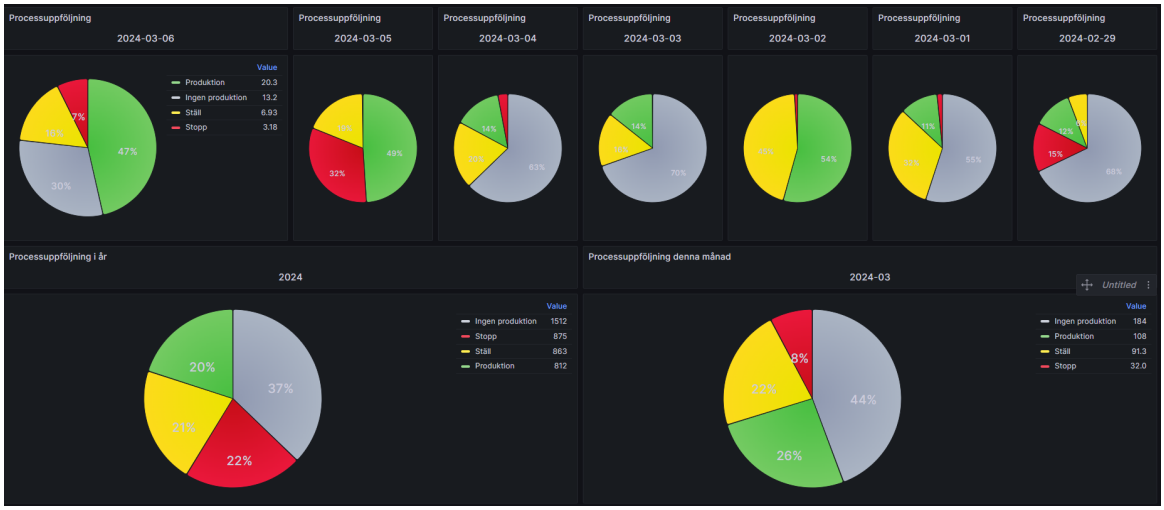
*Kodexempel 1. Utnyttjandegrad för dagens datum.*

```
select  status_text, duration/36000000000 as hours
from (select  cast(created_at as date) as report_date,case indirect_code
when        then 'Stopp'
when        then 'Stopp'
when        then 'Stopp'
when        then 'Stopp'
when        then 'Stopp'
when        then 'Stopp'
when        then 'Stopp'
when        then 'Stopp'
when        then 'Stopp'
when        then 'Stopp'
when        then 'Stopp'
when        then 'Stopp'

when        then 'Ställ'
when        then 'Ställ'
when        then 'Ställ'
when        then 'Ställ'
when        then 'Ställ'
when        then 'Ställ'

when        then 'Ingen produktion'
when        then 'Ingen produktion'
else 'Produktion' end as status_text, sum(calculated_time) as duration
from report_item group by report_date, status_text) q where report_date=current_date
```

Detta visualiseras sedan med pajdiagram där olika färger representerar kategorierna och antalet timmar för varje kategori visas i en tabell bredvid diagrammet för dagens, månadens och årets utnyttjandegrad. Varje kategori visar även hur många procent av totalen som den representerar för utvalt tidsintervall. Det finns ett diagram för varje dag upp till en vecka bakåt i tiden och även för nuvarande månad och hela året. Det visas i figur 6.




*Figur 6. Utnyttjandegrad enligt olika tidsintervall.*

För att få fram månadens och årets utnyttjandegrad behöver *date_part-funktionen* användas vilket kan ses i kodexempel 2. (PostgreSQL DATE_PART() Function).

*Kodexempel 2. Utnyttjandegrad för året.*

```
select  status_text,
 duration/36000000000 as hours
from (select  date_part('year',created_at) as report_year,case indirect_code
when        then 'Stopp'
when        then 'Stopp'
when        then 'Stopp'
when        then 'Stopp'
when        then 'Stopp'
when        then 'Stopp'
when        then 'Stopp'
when        then 'Stopp'
when        then 'Stopp'
when        then 'Stopp'
when        then 'Stopp'
when        then 'Stopp'

when        then 'Ställ'
when        then 'Ställ'
when        then 'Ställ'
when        then 'Ställ'
when        then 'Ställ'
when        then 'Ställ'

when        then 'Ingen produktion'
when        then 'Ingen produktion'
else 'Produktion' end as status_text ,sum(calculated_time) as duration
from report_item group by report_year,status_text) q where report_year=date_part('year',current_date)
```

Enligt användningsfall i kapitel 3.1 bör det även finnas utnyttjandegrad för varje linje var för sig. Under dataanalysen upptäcktes det att andra maskiner fanns med i tabellen som inte ska användas och vi behövde utesluta dem i koden. För att få fram utnyttjandegraden för varje linje behövde två tabeller kombineras, som beskrevs i kapitel 2.1.1, genom en *left join* för att få fram namnet på maskinen och även utesluta de som inte producerar.

Likt tidigare exempel delas stopporsakerna in i tre kategorier, annars produktion, och varje maskin kommer visa sin andel av de fyra kategorierna och de kommer sorteras enligt namnet på maskinen. Kodexemplet nedan visar summan av en vecka bakåt i tiden men också varje maskins andel. Då användes *report_date >= current_date - integer '6'* som förklarades tidigare i kapitlet.

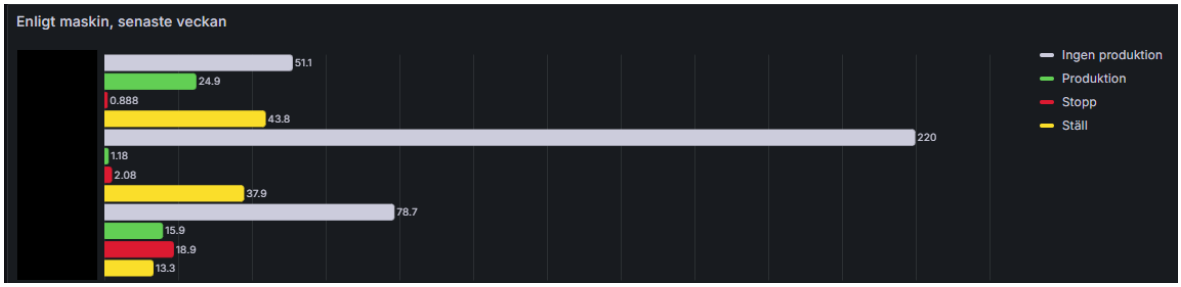
*Kodexempel 3. Utnyttjandegrad varje linje var för sig, den senaste veckan.*

```
select  status_text, terminal_name,
sum(duration/36000000000) as hours
from (select  date(created_at) as report_date,case indirect_code
when       then 'Stopp'
when       then 'Stopp'
when       then 'Stopp'
when       then 'Stopp'
when       then 'Stopp'
when       then 'Stopp'
when       then 'Stopp'
when       then 'Stopp'
when       then 'Stopp'
when       then 'Stopp'
when       then 'Stopp'
when       then 'Stopp'

when       then 'Ställ'
when       then 'Ställ'
when       then 'Ställ'
when       then 'Ställ'
when       then 'Ställ'
when       then 'Ställ'

when       then 'Ingen produktion'
when       then 'Ingen produktion'
else 'Produktion' end as status_text ,sum(calculated_time) as duration, machine_id
from report_item group by machine_id,report_date,status_text) q
left join (select distinct machine_id,terminal_name from  manual_work_log_item
where (machine_id!=      and terminal_name not in (                          ) or (machine_id=     and terminal_name=          )) m on m.machine_id = q.machine_id
where report_date>=current_date- integer'6'
group by status_text, terminal_name
ORDER BY terminal_name
```

Det ger grafen i figur 7. Ett stapeldiagram valdes för detta där y-axeln är namnet på maskinen och x-axeln visar antalet timmar. Egentligen hade man velat ha en stapel för varje maskin som är uppdelad i fyra färger, på så sätt skulle man i figuren nedan ha endast tre staplar i stället för 12. Det gick tyvärr inte att ordna en sådan visualisering och i stället fick det bli en stapel för varje maskins kategori i ordningen grå, grön, röd och gul, vilket man ser i figuren nedan. Det gjordes en för dagens datum och en som summerar senaste veckan.




*Figur 7. Utnyttjandegrad för varje linje var för sig, den senaste veckan.*

Efter att ha skapat instrumentpanelen för utnyttjandegrad för alla maskiner gjordes två instrumentpaneler enligt figur 6 för två av de maskinerna skilt.

## 4.2 Processdata

Som det nämndes tidigare i kapitlet gick det tyvärr inte att hämta data från en av databaserna samt att det inte gick att kombinera data från flera än två databaser med olika

nycklar. Det gjorde att några av förslagen som togs upp i användningsfallen i kapitel 3.1 inte kunde uppfyllas.

I figur 8 kan man se instrumentpanelen som skapades för processdata.


*Figur 8. Processdata för en maskin*

Längst upp till vänster på instrumentpanelen i figur 8 kan man se när data senast har hämtats. Nedanför det visas den nuvarande orderns ordernummer, artikelnummer samt när ordern har startats.

I figur 8 kan man se att det finns en graf som visar tryck och varv enligt tid och en visualisering kallad *gauge* som visar värdena numeriskt samt fylls i förhållande till värdena. (Gauge). Alla dessa värden hämtas från en aktiv order, därav behöver data kombineras från tre databaser för att få fram nuvarande och nominella värden i samma graf. Man behöver kombinera data från Monitor MI och Monitor MI OPC på nyckeln *machine_id* och data från Monitor MI och körprotokoll på nyckeln *order_number*. På så sätt skulle man kunna hämta ut aktuella och nominella värden för en aktiv order.

Eftersom det inte gick placerades de nominella värdena i separata numeriska visualiseringar kallad *stat*. Bristerna med detta är att man inte tydligt kan använda färger i visualiseringarna för aktuella eftersom varje order har olika värden som de behöver uppfylla. Detta gör att den grafiska designen inte blir intuitiv då man inte tydligt kan se om värdena håller sig inom lämpliga intervall. För gauge-visualiseringen skulle man även kunna ange min- och maxvärden vilket skulle representera båda ändorna.

För utnyttjandegraden behövde endast data från Monitor MI hämtas och kombineras från tabeller. I Monitor OPC finns varje maskin som separat tabell och således behöver inte data kombineras från flera tabeller för att exempelvis visa aktuellt tryck. Däremot hade man velat kombinera det med nominellt tryck för att kunna visualisera min- och maxvärden för aktuell order och utforska implementeringen av larm baserat på det. Istället hämtades aktuella värden enligt kodexemplet nedan.

*Kodexempel 4. Databasfråga för att hämta aktuellt tryck och varv per minut.*

```
SELECT
distinct
time,
extruder_pressure_1 as "Pressure",
extruder_r_p_m as "RPM"


FROM                                    order by time desc LIMIT 20000
```

Som det nämndes i kapitel 3.3 behövde Excel-filer studeras för att som i detta fall, ta reda på vad de olika OPC-parametrarna kallas i körprotokoll. *Codetext* är i detta fall vad OPC-parametern kallas i körprotokoll och för *extruder_pressure_1* kallas den ”TRYCK”. I körprotokoll finns de nominella värdena och för att kunna få fram dem för en aktiv order behöver data från MSSQL-databasen elektroniskt körprotokoll kombineras med Monitor MI. Data hämtas inte längre än en vecka tillbaka i tiden för att begränsa mängden data som hämtas. De nominella värdena visas sedan i en stat-visualisering. (Stat).

*Kodexempel 5. SQL-fråga för att hämta nominellt tryck från körprotokoll.*

```
SELECT
m.ordernr as order_number
,m.partid
,nom.codetext
,nom.value as nomval


  FROM
  [Korprotokoll].MainData m
  inner join
  (select p.*,d.*, t.protocoldescriptionid,t.Revision  from Processcard.[Data] p
  left join protocol.template t  on p.templateid = t.ID
  left join protocol.description d on d.id = t.ProtocolDescriptionID
  where t.columnindex=1) nom on nom.partid = m.partid and nom.Revision = m.ProtocolTemplateRevision


  where nom.codetext = 'TRYCK'
    and date_start > getdate()-7
```

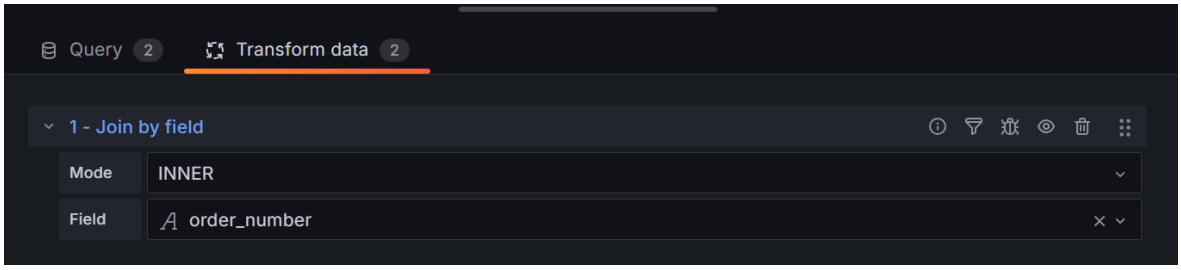
I Monitor MI finns artikelnummer, ordernummer och starttid i tabellen *current_work* vilka hämtas och med villkoren att ordern är aktiv samt att maskinens ID motsvarar den specifika maskinen. Koden ses nedanför.

*Kodexempel 6. Hämta data från en aktiv order för en specifik maskin.*

```
1   SELECT order_number, part_number, cast(machine_id as text),start_time
2   FROM current_work WHERE (is_active = true AND machine_id = [redacted])
```

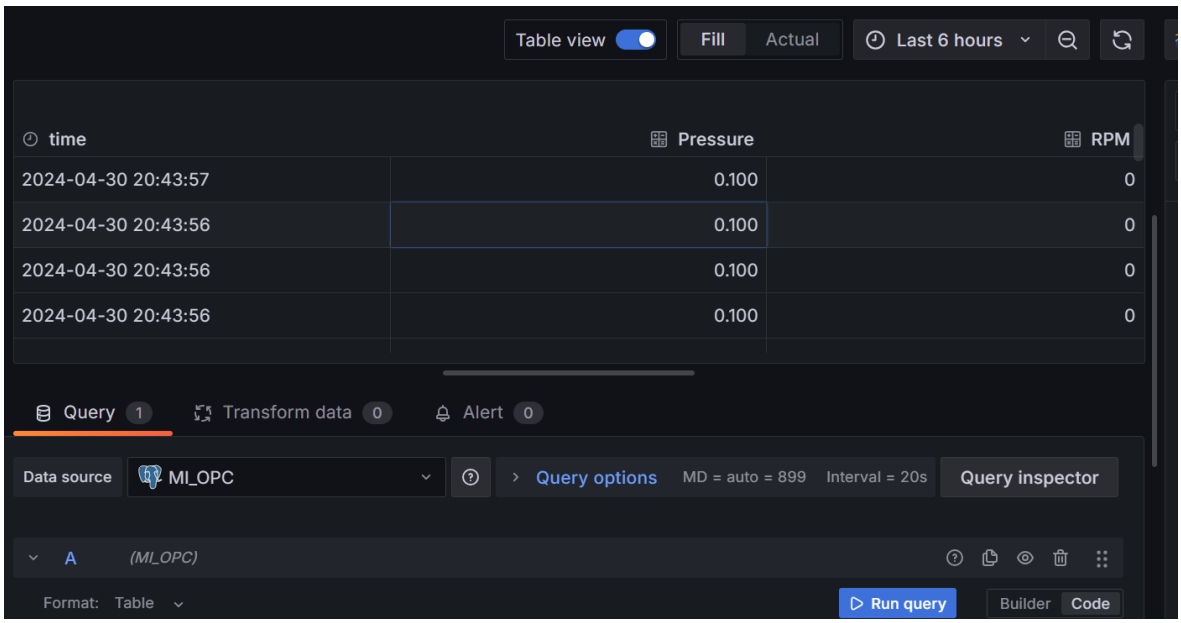
En *inner join* görs på *order_number* för att kombinera data från elektroniskt körprotokoll och Monitor MI, i detta fall med Grafanas funktion ”Transform Data”. (Transform data).

En *inner join* gör att endast rader med matchande värden kommer att visas från databaserna, samma ordernummer behöver finnas i databaserna för att en rad ska returneras. (SQL INNER JOIN).




*Figur 9. Inner join med Grafanas ”Transform data”.*

När man väljer att köra en databasfråga kan man ändra visualiseringen till att visa data i en tabell. Detta är hjälpsamt för att säkerställa att man hämtar rätt typ av data och kontrollera att man inte får fram felaktiga data. Det visas i figur 10.




*Figur 10. Tabell vy efter att ha kört en databasfråga i Grafana.*

# 5 Diskussion

Som en följd av att det inte gick att hämta ut alla nödvändiga data samt kombinera data från flera än två datakällor kunde inte möjligheterna i Grafana utforskas fullt ut. I framtiden kunde man göra en förbearbetad databas som synkar data från Monitor, Monitor MI och elektroniskt körprotokoll, på så sätt skulle man ha två databaser att kombinera data från och således kunna skapa larm baserade på min- och maxvärden och ytterligare utforska möjligheterna att implementera larm baserade på standardavvikelser. Man skulle även kunna ha min- och maxvärden för den nuvarande ordern på y-axeln och därmed få en bättre överblick och ha olika färger baserat på värden för nuvarande order. Exempelvis kunde man ha grönt om värdena är inom intervallet och rött om de hamnar utanför. Alternativt kunde man utforska möjligheterna att kombinera all data till en databas i InfluxDB.

Examensarbetet har varit väldigt lärorikt och intressant och det har gett mig en ökad förståelse för databaser och deras användning i arbetslivet. Även hela processen att slutföra ett projekt genom alla dess steg från idé till slutstadiet har gett mig värdefull erfarenhet. Kunskaperna i Grafana är också något jag tar med mig och som kan användas i andra visualiseringsverktyg såsom Power BI.

Avslutningsvis vill jag tacka min handledare på Optinova, Adam Mattsson, och min handledare på Novia, Jan Berglund, för deras handledning under examensarbetet.

# 6 Litteraturförteckning

*A Brief History of PostgreSQL.* (u.d.). Hämtat från PostgreSQL: https://www.postgresql.org/docs/9.3/history.html

*Microsoft SQL Server*. (den 26 Februari 2024). Hämtat från Wikipedia: https://en.wikipedia.org/w/index.php?title=Microsoft_SQL_Server&oldid=1210428592

*Grafana*. (den 28 Februari 2024). Hämtat från Wikipedia: https://en.wikipedia.org/w/index.php?title=Grafana&oldid=1210845302

*What are databases?* (u.d.). Hämtat från Azure: https://azure.microsoft.com/sv-se/resources/cloud-computing-dictionary/what-are-databases

*Microsoft Power BI*. (den 28 Mars 2024). Hämtat från Wikipedia: https://en.wikipedia.org/w/index.php?title=Microsoft_Power_BI&oldid=1215991388

*Relational database.* (den 6 April 2024). Hämtat från Wikipedia: https://en.wikipedia.org/w/index.php?title=Relational_database&oldid=1217554613

*What is a relational database?* (u.d.). Hämtat från Google Cloud: https://cloud.google.com/learn/what-is-a-relational-database

*What is a Relational Database (RDBMS)?* (u.d.). Hämtat från Oracle: https://www.oracle.com/database/what-is-a-relational-database/

*Om Monitor ERP.* (u.d.). Hämtat från Monitor ERP: https://www.monitorerp.com/sv/om-foeretaget-monitor/

*Vad är ERP?* (u.d.). Hämtat från Monitor ERP: https://www.monitorerp.com/sv/vad-aer-erp/

*SQL.* (den 11 April 2024). Hämtat från Wikipedia: https://en.wikipedia.org/w/index.php?title=SQL&oldid=1218363872

*What is a Database?* (u.d.). Hämtat från Oracle: https://www.oracle.com/database/what-is-database/

Rouse, M. (den 12 Januari 2024). *DBMS (Database Management System).* Hämtat från Techopedia: https://www.techopedia.com/definition/24361/database-management-systems-dbms

*What is Database Management Systems (DBMS)?* (u.d.). Hämtat från AppDynamics: https://www.appdynamics.com/topics/database-management-systems

Chamberlin, D. D. (Oktober-December 2012). Early History of SQL. *IEEE Annals of the History of Computing*, ss. 78-82. Hämtat från https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6359709&isnumber=6359700

*Relational vs. Non-Relational Databases.* (u.d.). Hämtat från MongoDB: https://www.mongodb.com/compare/relational-vs-non-relational-databases

Kuhn, W. (den 7 Maj 2018). *A Brief History of PostgreSQL.* Hämtat från Medium: https://medium.com/launch-school/a-brief-history-of-postgresql-36d8d392c611

Sardar, A. S. (den 18 December 2023). *Understanding SQL Dialects: A Deeper Dive into the Linguistic Variations of SQL.* Hämtat från Medium: https://medium.com/@abhapratiti27/understanding-sql-dialects-a-deeper-dive-into-the-linguistic-variations-of-sql-e7e2fdb7509b

Biswal, A. (den 29 Augusti 2023). *What is Power BI?: Architecture, and Features Explained.* Hämtat från Simplilearn: https://www.simplilearn.com/tutorials/power-bi-tutorial/what-is-power-bi

*Maskinintegration.* (u.d.). Hämtat från Monitor ERP: https://help.monitorerp.com/se-monitor_g5/sv-se/Content/Topics/UserGuide/Options/MachineIntegration.htm

*PostgreSQL DATE_PART() Function.* (u.d.). Hämtat från PostgreSQL Tutorial: https://www.postgresqltutorial.com/postgresql-date-functions/postgresql-date_part/

*Data visualisation beginner's guide: a definition, examples and learning resources.* (u.d.). Hämtat från Tableau: https://www.tableau.com/en-gb/learn/articles/data-visualization

MetricFire. (den 28 Augusti 2023). *What is Grafana?* Hämtat från Medium: https://medium.com/@MetricFire/what-is-grafana-8de44d241765

*PostgreSQL data source.* (u.d.). Hämtat från Grafana: https://grafana.com/docs/grafana/latest/datasources/postgres/

*Stat.* (u.d.). Hämtat från Grafana: https://grafana.com/docs/grafana/latest/panels-visualizations/visualizations/stat/

*SQL INNER JOIN.* (u.d.). Hämtat från W3Schools: https://www.w3schools.com/sql/sql_join_inner.asp

*Publish to web from Power BI.* (u.d.). Hämtat från Microsoft: https://learn.microsoft.com/en-us/power-bi/collaborate-share/service-publish-to-web

*Transform data.* (u.d.). Hämtat från Grafana: https://grafana.com/docs/grafana/latest/panels-visualizations/query-transform-data/transform-data/

*Pie chart.* (u.d.). Hämtat från Grafana: https://grafana.com/docs/grafana/latest/panels-visualizations/visualizations/pie-chart/

*Gauge.* (u.d.). Hämtat från Grafana: https://grafana.com/docs/grafana/latest/panels-visualizations/visualizations/gauge/

*About Optinova.* (u.d.). Hämtat från Optinova: https://optinova.com/about/