# Intro

Under dagens presentation så vill vi visa vad vid har gjort, men också ge en liten bild av vad affärsnyttan för företaget var.

# Företaget och BeX Online

BeX lanserades 2010. Skaparna hade många års erfarenhet av att arbeta med Microsoft Navision men tröttnade på att det inte fanns någon standardlösning i en bransch där affärssystem mycket väl kan vara det. Affärssystemsimplementationer kostar generellt sett flera miljoner att genomföra och tar flera månader. BeX blev omnämnda i höstas för att de lyckades implementera sin plattform hos AlexandAlexa, en av världens största e-handlar för barnkläder, på två veckor.

## Vad är BeX Online?

BeX Online är ett molnbaserat omni-kanals affärssystem nischat mot retail och e-commerce.  
Det går att koppla på retail, e-commerce och b2b-system och systemet koms åt genom webbläsaren.

I examensarbetet så fokuserades det på dessa områden och då framförallt på sql-databasen. Databassystemet är SQL Server 2014 Web Edition om det är någon i publiken som är insatt.

# Vad är deras problem?

När ett sådant system används så bildas det väldigt mycket data över tid. Denna data vill man kunna hämta fram och ta beslut ifrån, d.v.s. skapa beslutsunderlag. Detta görs idag via affärsrapporter. Dessa rapporter gräver igenom hela systemets databas och sammanställer viktigt information som t ex: lagerrapporter, inköpsrapporter, försäljningsstatistik etc. Dessa rapporter är ofta väldigt viktiga för ledning så att de får en överblick över hela verksamheten. Just försäljningsstatistik är något som de högre cheferna och ledning ofta tittar på och det är därför väldigt viktigt att dessa fungerar bra för att bibehålla en bra kundrelation.

## Mål

* Identifiera och kartlägg hur processen bakom dessa rapporter fungerar. Förstå vilka delar av systemet som påverkas och hur framställningen av beslutsunderlagen fungerar.
* Hitta flaskhalsar och andra suboptimala saker.
* Analysera och jämför olika lösningar på dessa problem som finns.
* Ge förslag på förbättringar som företaget kan göra på sin arkitektur för att förbättra dessa processer.

# DEMO

Och nu för att ni ska få se vad det är vi faktiskt har försökt snabba upp så ska vi demonstrera det i systemet.

-Visa liveuppdateringen.

-Visa en rapport

# Hur arbetade vi?

Kolla på sliden

# Research

## Indexfragmentering

Det finns intern och extern fragmentering. Intern fragmentering är när många av sql-databasens sidor är halvfullar, vilket ofta händer när man bygger om index. Extern fragmentering är när sidorna ligger i fel ordning, vilket gör att databasen måste utföra random-searches vilket är väldigt lågnsamt, framförallt på traditionella mekaniska hårddiskar. Fun fact: En mekanisk hårddisk presterar ungefär 1% av sin totalhastighet när den utför slumpläsningar jämfört med sekventiella läsningar eftersom att den måste flytta runt läshuvudet.

## Hekaton

Hekaton är Microsoft’s lösning till behovet av att ha tabeller i ram-minnet. Att skapa sådana tabeller är inte mycket svårare än att skapa en vanlig tabell. Det är dock ganska annorlunda då indexering t ex sker på ett helt annat sätt. Hekaton har visat på stor prestandaförbättringar hos andra företag.

## Underhåll

När en databas används under tid så förändras den. Den växer och/eller krymper och mycket data ändras. Det som händer då är att indexeringen bryts ner och blir fragmenterad, innehåller håligheter osv. Statistik på tabeller och index som används av querys blir också utdaterad och behöver underhållas. Därför bör en databas med jämna rum underhållas och uppdatera saker som t ex indexering och statistik.

# Tidiga findings

Saker som vi upptäckte väldigt tidigt i arbetet var att företaget udnerhöll sin databas dåligt. Många index var väldigt fragmenterade och när man tittade på execution plansen för queries så varnade många för saknade index.

# Nya index

* Visa originaltider
* Visa utan indexering
* Nya index skapades genom att titta på stored procedures och vad de sökte efter.

# Maintenance

## Data and log file management

Behöver nödvändigtvis inte vara med i en maintenance plan, förutom att hålla kolla på log-filernas tillväxt. Men det man bör se över:

* Data- och log-filerna ska vara skilda från varandra och helst finnas på olika hårddiskar.
* Auto-growth, dvs hur dess filer växer skall konfigueras korrekt. Ofta är detta satt till en procentsats, 10%, Men detta kan ge upphov till eskaleringsproblem. Det är bättre att anpassa detta till en fix storlek.
* Shrinking bör aldrig användas då det är en väldigt resurskrävande process som även kan ge upphov till korrupta filer.

# Defragmentering av index & statistik

Det är väldigt vanligt att en underhållsplan defragmenterar alla index oavsett om de behöver fragmenteras eller inte. Detta är väldigt resurskrävande och man bör därför fixa endast de index som behövs. Samma sak med statistiken.

# Resultat av maintenance

Orange är med är med när endast tillräckligt stora index korrigerade och blått är när alla index defragmenterades oavsett storlek.

# Rekommendation

## Maintenance

Säger sig självt

## In-Memory

Kunde tyvärr inte testas då det kräver SQL Server Enterprise editions som kostar drygt 100 000 per licens men det rekommenderas starkt att företaget experimenterar med det om de köper det i framtiden. En annan lösning som undersöktes men som tyvärr inte kom med i exjobbet var Speedment. Det kom inte med för att företaget inte har släppt en version till SQL Server än. Speedment är en JAVA-lösning som speglar databasen. Tidiga tester med tekniken har gett väldigt lovande resultat och företaget påstår speedment kan göra alla datasökningar med O(1)-komplexitet.