Examinationsuppgift Databasteknik

Henrik Jansson

DevOps Engineer (DevOps23), Lernia, Alvik

Handledare Lennart Johansson

Datum 2024-03-21

Inlämningsuppgift har genomförts inom ramen för yrkeshögskolestudier vid Lernia.

1. Sammanfattning

Företaget NordicTech Handels är ett snabbt växande företag som säljer nordiska livsstilsprodukter. I och med detta vill företaget utveckla och modernisera sin databas och dess infrastruktur. I takt med ökad omsättning av produkter och att kunderna blir fler vill även företaget föra statistik. Kundupplevelsen är viktig för NordicTech och därför anser ägaren att recensioner och feedback är ett av kraven i utvecklandet av sin databas.

Med de ambitioner och krav som NordicTech ställer är det viktigt för företaget att klara av och hantera den mängd data som uppstår.

Arbetet med att skapa en databas involverar många steg, från planering, kravinsamling, implementering och testning. Genom att följa en bra praxis och använda lämpliga verktyg och tekniker görs för att få en bra effektivitet i arbetet med att skapa en väl utformad och stabil databas. Detta är viktigt för att skapa det som företaget är i behov av och matcha de krav som ställs.

2.Innehållsförteckning

Innehåll

3. Inledning

MySQL och databaser. Häng med på en resa i skapandet och utvecklandet av en databas och varför det är så viktigt med en stabil databas i företagets infrastruktur. Att ha en stabil, pålitlig och bra infrastruktur påverkar utvecklingen i positiv bemärkelse i form av prestanda, effektivitet men även vidareutveckling i framtiden.

Syftet med denna rapport är att vi ska gå igenom och utforska de olika områden och vad som krävs i skapandet och utvecklandet av en databas. Varför är områden som design, säkerhet och skalning viktiga i en databas?

Syftet med detta arbete är att utveckla NordicTech Handels databas. Företaget expanderar snabbt och är i stort behov en modern, robust och pålitlig databas. Det finns några typer av databaser men relationsdatabaser (MySQL) är den typen vi går igenom i denna rapport. Som det låter, databasen har relationer och det är dessa vi kommer skapa i vår design med hänsyn till kravbilden som finns.

Ju fler data som lagras och hanteras desto mer resurser krävs för att nå en hög prestanda. I MySQL-databaser pratar vi om att minska redundans, detta menas kort och gott att vi inte ska ha samma information på fler ställen då detta försämrar prestanda då mer data behöver bearbetas. Ta ett exempel, det är onödigt att ha två pennor i en hand när du bara kan använda en? Med en bra design och minskat redundans skapar vi oss förutsättningarna för en bättre prestanda.

När vi pratar om effektivitet och prestanda inom databaser är en viktig del i designen att vi tar hänsyn till normaliseringsformerna. ”Första normaliseringsformen” och ”Andra normaliseringsformen”, även kallat 1NF och 2NF. Vi delar upp data i flera olika ”tabeller” för att sedan skapa relationer mellan tabellerna. Hur gör vi detta?

Efter vi skapat vår design i form av ett ER-diagram ska vi omsätta detta i tabeller. Väl i tabellerna måste vi skapa nycklar och relationer som Primary Key och Foreign Key. PK och FK är relationerna om vilka tabeller som hämtar data från vilken.

Mer om detta om hur vi går till väga med skapandes genom att använda oss av språket SQL.

4. Projektplan

4.1 Krav och Analys:

Vad ska databasen innehålla? Finns det krav? Vilka är behoven? Hur ser framtiden ut? Denna information är grunden för designen.
Analysen har en stor betydelse då vi måste vara väldigt noggranna i vår bedömning där vi säkerställer att krav och behov är relevanta.

Några viktiga krav att ta hänsyn till:

- Kundhantering
- Produkthantering
- Orderhantering
- Skalbarhet
- Säkerhet
- Språk
- Geografiska servrar

4.2 Design:

Innan vi bygger vår databas behövs en ritning tas fram i form av ett ER-diagram med en struktur där vi kan se en röd tråd om hur tabeller, entiteter och attribut håller samman. Vi gör detta dels för att säkerställa att relevant information om kraven finns med och att det finns en klarhet i vad som ska ske i skapandet av databasen.

Designen av diagrammet ska ta hänsyn till normaliseringsformerna för att minska redundansen. Med redundans menas att vi organiserar data med tabeller och minskar dubbla data.
Första (1NF) och andra (2NF) normaliseringsformen ökar även dataintegriteten.

Designen och dess skalbarheten är viktig för NordicTech då expansionen är ett faktum. Eftersom databasen ställer höga krav på prestanda behöver vi se över hur vi skalbarheten på ett säkert och ett kostnadseffektivt sätt.

Någon form av dokumentation om hur skapandet har skett bör också finnas med. Detta för att skapa förståelse för andra men även för utveckling och underhåll.

Ta hänsyn till säkerheten. Vem/vilka ska ha tillgång. Detta gör vi genom att sätta olika roller som skapar behörighet.

4.3 Implementering och utveckling:

Det är i detta skede databasen ska ta form. Att skapa tabeller med attribut, nycklar och relationer

- Databasstruktur
- Datatyper
- Tabeller
- Entiteter
- Attribut
- Relationer

En viktig del i designen är att etablera nycklar så som Primary Key (PK) och Foregin Key (FK). Unika identifierare i varje tabell krävs för att klargöra och säkerställa dataintegriteten. *Lite mer om PK/FK och relationer finns i avsnittet Resultat*.

Rätt datatyper ökar prestandan i vår databas och säkerställer även att rätt datatyper används för att få korrekt data i fälten. När data hämtas via en query behöver databasen sortera och få fram rätt data, med hjälp av de olika datatyperna förenklar vi och effektiviserar bearbetningstiden.

4.4. Testning av CRUD

För att säkerställa att det vi har skapat fungerar bör vi göra någon form av test. Är tabellnamn och rader rätt implementerat?

Genom några enkla Queries kommer vi att få fram de resultat vi vill ha.

5. Metoder

5.1 Krav & Analys

Genom möten och en hel del frågeställningar tar vi fram en kravbild som täcker helheten. Vi måste se över och analysera företagets processer. Vad är de viktigaste kraven? Genom gemensamma diskussioner och satta prioriteringar så det blir enklare för oss att skapa en bild över hur designen i ett ER-diagram ska se ut.

5.2 Verktyg & tekniker

För skapandet av databasen använder jag mig av verktyget Mysql Workbench. Genom att skriva SQL har jag skapat mina tabeller, attribut, datatyper och relationer.

Mer om detta i resultat.

5.3 Design – Modellering - Säkerhet

Mitt ert diagram har jag skapat i verktyget Mysql Workbench. Det är i designstadiet vi definierar och utformar databasen med noga hänsyn till normaliseringsformerna. I mysql workbench är det enkelt att skapa tabeller och sätta relationer via ett relationssystem som heter ***Craws foot.***

Mer om detta i resultat.

När det kommer till att designa säkerhet finns det många aspekter att ta hänsyn till. Några av de vanliga säkerhetsåtgärderna som vidtas är saker som behörigheter, säkra lösenord, säkerhetskopiering och uppdateringar. Beroende på kravbilden och vad organisationen efterfrågar anser jag att den ska noggrant bedömas och planeras just för de kraven som ställs.

Vi kommer inte i denna rapport göra någon steg-för-steg för skapandet av säkerheten men jag vill lägga stor vikt på att nämna detta: ***Att designa säkerhet med hänsyn till kraven som ställs bör alltså projektplanen ha en separat del just för säkerheten. Dels för att uppvisa extra tydlighet och att fokus här måste vara av högsta kaliber. Dokumentation är även extra noga här.***

5.4 Implementering

Med hjälp av mysql workbench ska vi nu omsätta den design vi har skapat tidigare moment. Genom att använda oss av SQL kommandon som vi har skapat från designen är det nu enkelt att skapa tabeller, attribut, datatyper, nycklar och relationerna.

En viktig del att ha i åtanke när vi skapar vår databas är att tabeller är det första som bör skapas. När vi sätter relationer mellan tabeller behöver båda tabellerna finnas och ha fullständigt innehåll. Återigen är vikten av ett grundläggande och noggrant designat ER-diagram viktigt.

Beroende på kraven runt säkerheten sätter kan vi även sätta dessa när vi implementerar databasen.

Sammanfattning av metoder uppvisas i kommande avsnitt Resultat. Där kan vi följa de steg för steg i en del av implementeringen.

5.5 Testning av CRUD

INSERT INTO:

När vi väl är i gång med databasen kan behöriga användare till exempel lägga till data genom att använda oss av en INSERT INTO

```
1    INSERT INTO
2    `nordictech`.`user`
3    (`userid`, `username`, `password`, `fullname`,
4     `address`, `city`,  `county`, `zip`, `country`, `phone`,  `email`)
5    VALUES
6    ('2', 'Martin', 'hej123', 'henrik jansson',
7    'kolonigatan 2', 'sodertalje', 'sthlm', '15145', 'sweden', '0', '0');
8
```

userid	username	password	fullname	address	city	county	zip	country	phone	email
1	henrik	hej123	henrik jansson	kolonigatan 2	sodertalje	sthlm	15145	sweden	0	0
2	Martin	hej123	henrik jansson	kolonigatan 2	sodertalje	sthlm	15145	sweden	0	0
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

SELECT:

För att läsa data. Exempel

	userid	username	password	fullname	address	city	county	zip	country	phone	email
▶	1	henrik	hej123	henrik jansson	kolonigatan 2	sodertalje	sthlm	15145	sweden	0	0
	2	Martin	hej123	henrik jansson	kolonigatan 2	sodertalje	sthlm	15145	sweden	0	0

I vår databas kör vi denna operation.

```
1 • use nordictech;
2 • select username from user where userid =1;
```

Resultat: username visas

	username
▶	henrik

Update:

För att göra ändringar. Exempel:

Här ser vi att username henrik har email = 0.

	userid	username	password	fullname	address	city	county	zip	country	phone	email
▶	1	henrik	hej123	henrik jansson	kolonigatan 2	sodertalje	sthlm	15145	sweden	0	0
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

I vår databas kör vi denna operation.

```
use nordictech;
update user SET email = 'henrik.jansson@hejsan.se' where username = 'henrik';
```

Resultat: email har uppdaterats.

	userid	username	password	fullname	address	city	county	zip	country	phone	email
▶	1	henrik	hej123	henrik jansson	kolonigatan 2	sodertalje	sthlm	15145	sweden	0	henrik.jansson@hejsan.se
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Delete:

För att radera data från vår databas. Exempel:

Vi har här två users.

	userid	username	password	fullname	address	city	county	zip	country	phone	email
▶	1	henrik	hej123	henrik jansson	kolonigatan 2	sodertalje	sthlm	15145	sweden	0	0
	2	Martin	hej123	henrik jansson	kolonigatan 2	sodertalje	sthlm	15145	sweden	0	0
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

I vår databas kör vi denna operation.

```
2 •  use nordictech;
3 •  delete from user where userid = 2
```

Resultatet: userid 2 har raderats.

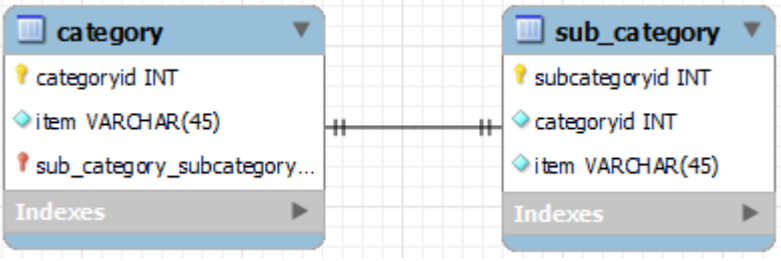
	userid	username	password	fullname	address	city	county	zip	country	phone	email
▶	1	henrik	hej123	henrik jansson	kolonigatan 2	sodertalje	sthlm	15145	sweden	0	0
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

5.5 Kort om relationer

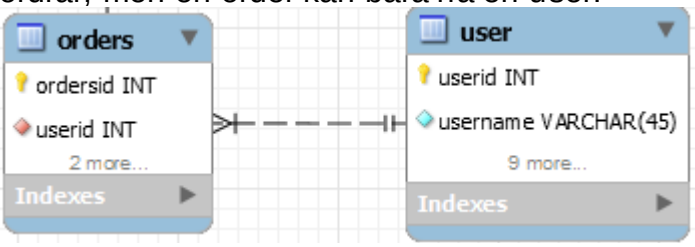
Att sätta relationer mellan tabeller kan vara klurigt. I relationsdatabasen använder vi som**: one-to-many, one-to-one, many-to-many** för att definiera sambandet. Detta förklarar hur tabellerna har möjlighet att prata med varandra. Till exempel: ***En user kan ha flera ordrar – en order kan inte ha flera users.***

Databasen måste ha organiserade data och det är precis vad relationerna är till för. När vi ska lagra eller hämta information behöver vi som administratörer på ett effektivt sätt hantera den data och information vi vill få ut.

- **One-to-one relation**
  I denna typ av relation skapar vi en relation mellan rader i två olika tabeller. Till exempel kan bara en category ha ett unikt sub_cagegoryid

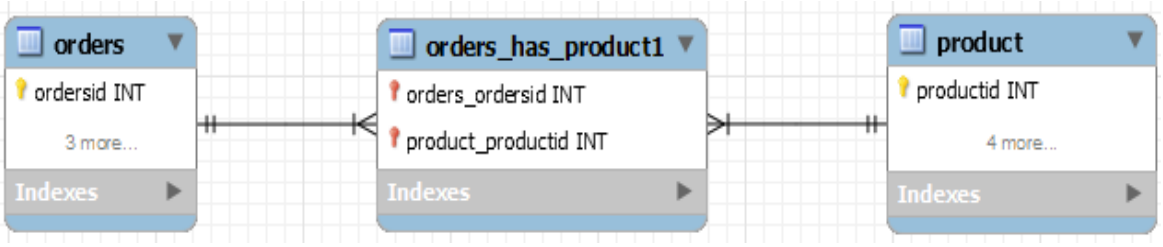


- **Many-to-one**
  Denna typ av relation kommer till användning när vi ska ha någon slags hierarkis ordning mellan tabeller. Som i tidigare exempel. En user kan ha flera ordrar, men en order kan bara ha en user.

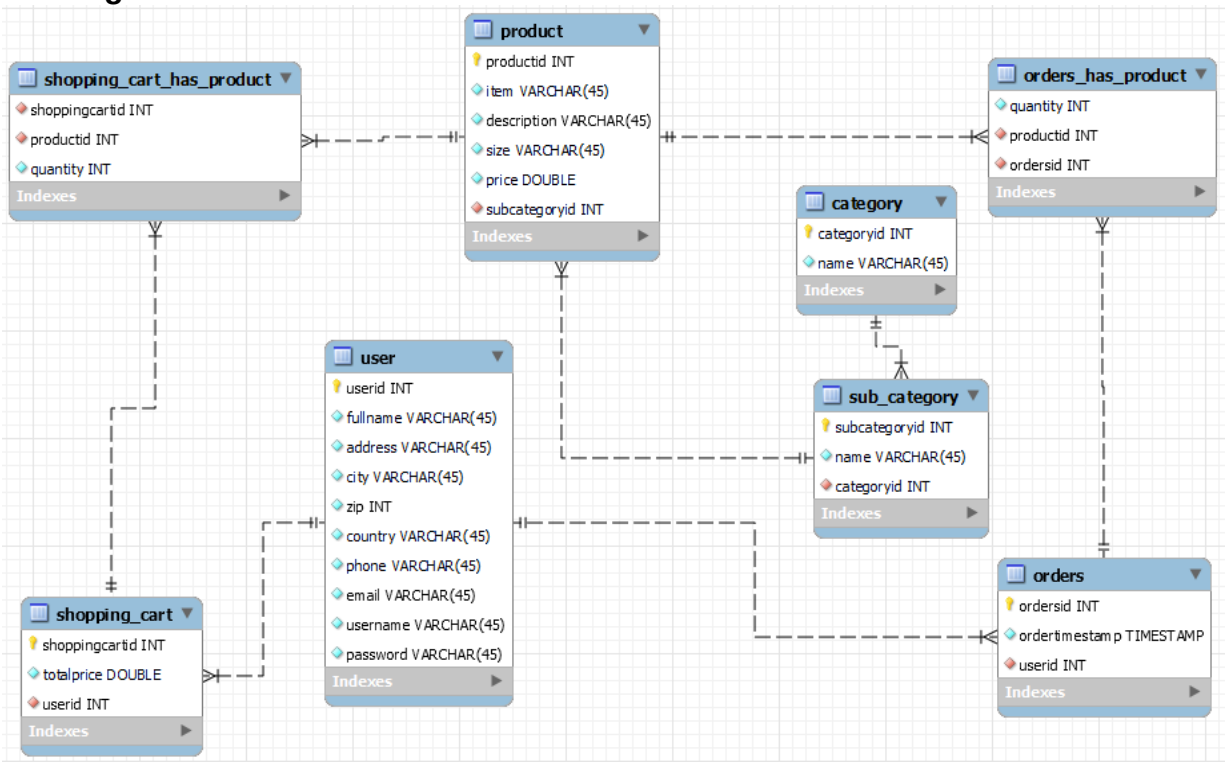


- **Many-to-many**

  I denna relation är det möjligt att skapa flera relationer mellan tabellerna. Vanligtvis när vi använder denna typ av relationer krävs ett ”junction-table”. Denna tabell kan vi se som en mellanhand, en tabell som länkar ihop tabeller.



ER-diagram med relationer mellan tabeller



5.6 Dokumentation

Dokumentationen som förts som uppvisar de steg-för-steg som gjorts

Hjälpmedel: Visual Studio Code

(Exempel på hur detta kan göras)

```
Step#1: Creates table
step#2: add primary key
step#3: adding relations between tables
-------------------------------------
#CATEGORY
-------------------------------------
#1
create table category(
    categoryid INT, Not null;
    item varchar(45));

#2
alter table category
add primary key (categoryid);
-------------------------------------
SUB_CATEGORY
-------------------------------------
#1
create table sub_category(
    subcategoryid INT, Not null;
    categoryid INT, not null;
    item varchar(45)); Not null;

#2
alter table sub_category
add primary key (subcategoryid)

#3
alter table sub_category
add foreign key (categoryid) references category (categoryid);
```

6.Resultat

För att ge dig som läsare en mer förståelse för hur skapandet och administrerande av databasen ser ut ska vi gå igenom några exempel. Vi kommer även gå igenom CRUD-operationer som jag tidigare nämnt.

Steg1 -CREATE TABLE:

Här skapar vi tabellen ”Produkt” som har några attribut som ska fyllas med data. Vi behöver även sätta datatyper så som int, double och VarChar. Detta för att endast tillåtna typer av data lagras på varje rad. En ”item” i detta läge får inte vara namnlös och måste ha ett värde.

```
---------------------------------------
PRODUCTS
---------------------------------------
#1
create table product(
    productid INT, Not null;
    sub_categoryid int, Not null;
    item varchar(45), Not null;
    description varchar(45),Not null;
    price double);Not null;
```

Steg2 – ADD PRIMARY KEY:

Jag nämnde tidigare Primary Key och Foreign Key. Nycklar för att skapa relationer mellan tabeller som möjliggör kommunikationen.

```
---------------------------------------
PRODUCTS
---------------------------------------
#1
create table product(
    productid INT, Not null;
    sub_categoryid int, Not null;
    item varchar(45), Not null;
    description varchar(45),Not null;
    price double);Not null;

#2
    alter table product
    add primary key (productid)
```

	productid	sub_categoryid	item	description	price
*	NULL	NULL	NULL	NULL	NULL

Nu har vi alltså skapat ett ”Table” där vi har satt productid som Primary key.

Bilden visar hur databasen ser ut efter Steg1 och Steg2.

Column Name	Datatype	PK	NN
productid	INT	☑	☑
sub_categoryid	INT	☐	☑
item	VARCHAR(45)	☐	☑
description	VARCHAR(45)	☐	☑
price	DOUBLE	☐	☑

Steg3 – ADD FOREIGN KEY:

Låt säga att vi i tidigare skede har skapat våra table ORDERS och USER.

För att skapa relationen mellan dessa tar vi hjälp av #3.
(bortse från #1 och #2 i svata rutan)

Här vill jag ändra något i tabellen orders. I tabellen orders vill jag ha en foreign key som är userid som kommer från table user.

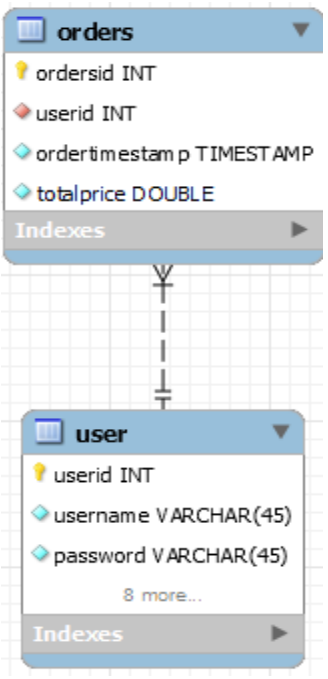
```
3 •    alter table orders
4      add foreign key (userid) references user (userid);
```

```
ORDERS
-------------------------------------

#1
create table orders(
    ordersid int, Not null;
    userid int, not null;
    ordertimestamp timestamp, Not null;
    totalprice double); Not null;

#2
    alter table order
    add primary key (ordersid)

#3
    alter table orders
    add foreign key (userid) references user (userid);
```



Bilden ovan visar ett exempel på en many-to-one relation.

---

6 Diskussion

Databaser är komplexa och kräver enormt med hängivenhet. Att vara noggrann, insatt och mycket dedikerad till området kommer resultera i bästa resultat. I skapandet av en effektiv och stabil databas vill jag säga att det är som krävs.

Resultatet jag fått fram i form av de steg jag gjort ser bra ut med tanke på den begränsade kunskap jag besitter på. Jag tror att arbetet hade sett annorlunda ut om kraven var andra och om min kunskap var bättre.

Har jag varit tillräckligt kritisk i mitt tankesätt? Har jag tillräckligt med kunskap för att förstå alla steg och områden? Hade jag haft detta hade förmodligen velat gjort en mer noggrann och mer utvecklad analys av: Utvecklingsmöjligheter, säkerhet och prestanda/effektivitet.

I mitt arbete saknas utvecklingsmöjligheter så som tar hänsyn till språk och geografiska lägen. Hur skulle detta ha granskats, designats och implementerats? Nu i efterhand tänker jag att ett större fokus vid krav och analys borde ha gjorts för att ta hänsyn till flera områden än det som gjorts i arbetet.

7 Slutsats

I detta arbete har de mest grundläggande saker tagits upp om hur vi skapar en databas med tabeller, relationer med mera. Jag har gett dig en grundläggande förståelse över hur arbetet sker, och vilka möjligheter det finns till att göra databasen mycket mer komplex.

I det stora hela är skapandet av databaser komplext där vi behöver ta hänsyn till många områden och aspekter. Noggrannhet och kritiskt tänkande är avgörande i slutändan. Att noggrant utvärdera de krav som organisationen ställer skapar vi oss möjligheter till bättre utveckling och resultat för databasen.

För att skapa den sista ”finishen” för NordicTech behöver jag implementera och ta hänsyn till flera saker än det som hittills skett. Detta kan innebära saker som hur säkerheten ser ut, global expansion med språk och länder.

Sist men inte minst vår plats för databasen. Ska vi ha fysiska servrar eller virtuella? Detta är något som jag definitivt borde haft med i planeringen i krav och analys. Att tilldela resurser för databasen med hänsyn till expanderingen är en sak vi absolut inte får bortse från.