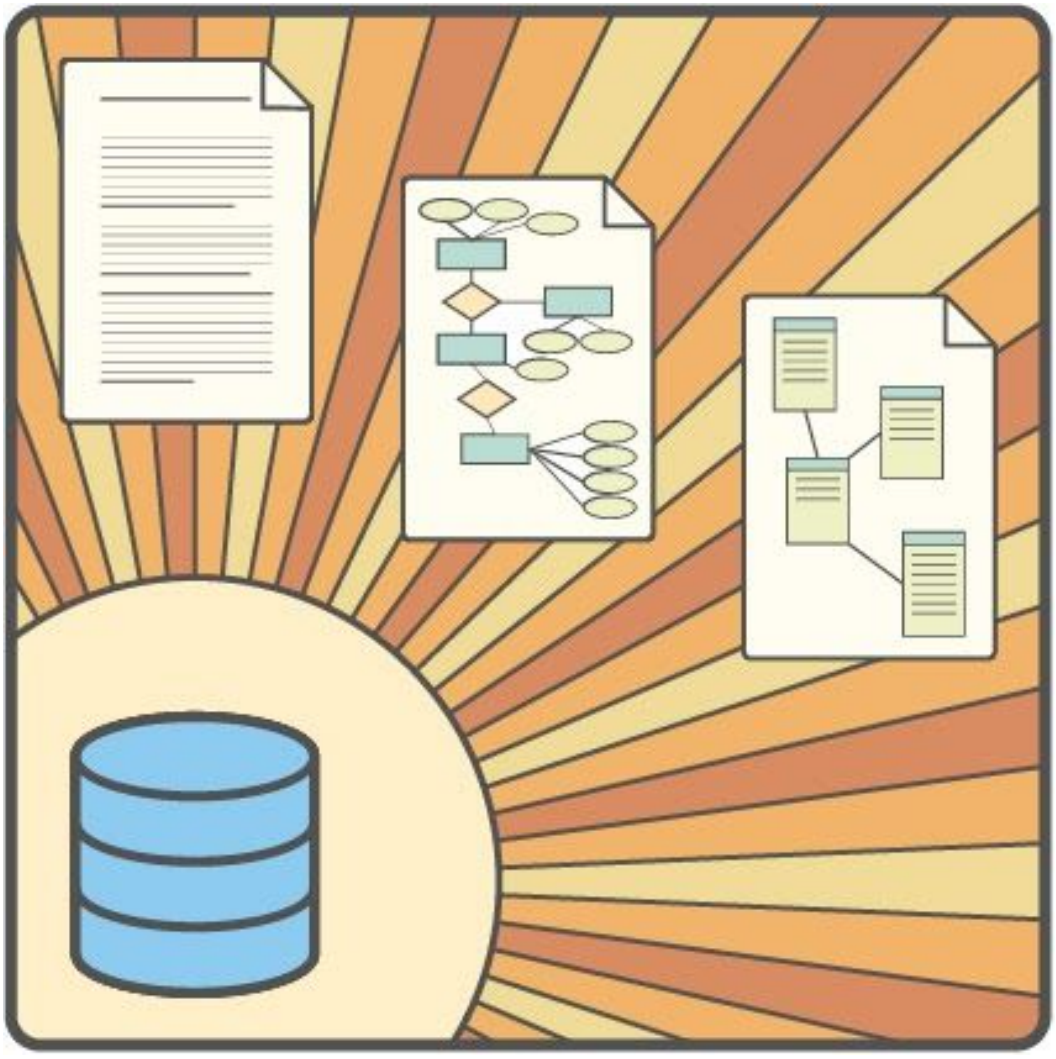
Databasmodellering

Konceptuell, Logisk & Fysisk modellering av databas



Daniel Andersson

daap19

Innehåll

Konceptuell modellering **3**

Beskriving av databasen 3

Entiteter 4

Relationer 4

ER-diagram 5

Kardinalitet 6

Attribut 6

Logisk modellering **8**

Relationsmodellen 8

Tabeller och attribut 9

Primära & främmande nycklar 9

Modifieringar / Rättelser 10

Anpassningar / Översättning till relationsmodellen 11

Fysisk modellering **13**

Skapandet av databasen eshop 13

Funktionsstöd som databasen ska innehålla 14

Appendix (SQL) **15**

Skapa databasen och användare (setup.sql) 15

Skapa tabeller och vyer för databasen (ddl.sql) 15

Steg 1 - Ta bort alla tabeller ifall de existerar, även vyer om de existerar. 15

Steg 2 - Skapa tabeller med attribut och nycklar. 16

Steg 3 - Skapa vyer 20

Konceptuell modellering

Beskriving av databasen

Databasen har syftet att vara grunden till en webbplats med målet att verksamhetens produkter nu kan säljas via ett e-handelssystem (e-shop) på en webbplats. Databasen innehåller följande information, ett **kundregister**, en **produktkatalog**, ett **lager**, en lista för **ordrar**, **plocklistor** för ordrarna, **fakturor** baserat på ordrarna som görs av kunden och en aktivitetslogg för att se över alla händelser som sker i databasen så om problem uppstår blir det lättare att lösa.

Lagret innehåller samtliga **produkter** som verksamheten säljer från **produktkatalogen**. Lagret presenterar **produkterna** genom *produktnummer*, *produktnamn*, *lagerplats*, *saldo*, *m.fl.*. *Produktnumret* är unikt och är refererens till produktkatalogen. *Lagerplatsen* är unik och är sammansatt av hyllan som platsen finns i och positionen i hyllan och höjdnivån på platsen. *Saldot* beskriver hur många av produkten som finns på lagerplatsen.

Produkter identifieras av ett unikt *produktnummer*. Varje **produkt** tillhör en/flera *kategorier* av produkter (exempelvis. tröjor, byxor, simkläder, träningskläder m.fl.), varje produkt finns placerade på en/flera *lagerplatser* och varje produkt har en/flera *produktbilder*. Alla **produkter** har även produktspecifik information lagrad i form av de vanliga attributen, *produktnummer*, *produktnamn*, *produktinformation, pris*.

Kunden skapar en order genom att beställa ett urval **produkter** som finns på **lagret**. **Produkterna** läggs till en **plocklista** som tillhör en **order** som ägs av **kunden** . En **kund** har ett unikt *kundnummer*, ett *namn* som är sammansatt av *förnamn* och *efternamn*, en *adress* som är sammansatt av *gatunamn, husnummer, postnummer, stad* och *land*, en *emailadress*, ett/flera *telefonnummer*.

Orderlistan ska innehålla alla **ordrar** som är skapade av **kunder**. Varje **order** har ett unikt *ordernummer*, ett *kundnummer*, en *leveransadress*, ett *pris* som är härlett från alla produkter som kunden beställt, en *orderstatus*. Orderstatus beskriver hur långt ordern är behandlad av verksamheten, exempelvis ny order, plockad order, levererad order, fakturerad order, makulerad order m.fl.

Plocklista skapas av **kunden** och innehåller de **produkter** som **kunden** väljer. Varje **plocklista** tillhör en **order**. **Plocklistan** har ett *indexnummer,* ett *produktnummer*, ett *produktnamn*, en *lagerplats* och *antal* av produkten som beställts.

En **faktura** baseras på en **order** som **kunden** har gjort och **plocklistan** inkluderar det urval **produkter kunden** valt. Urvalet **produkter** härleder till det totala pris som **fakturan** presenterar. **Fakturan** har ett unikt *fakturanummer,* ett *ordernummer,* ett *kundnummer*, en *leveransadress*, *fakturastatus*, *pris*.

Aktivitetsloggen sparar aktiviteter för ordrar i databasen för att kunna överse, korrigera eventuella fel som kan uppstå och underlätta vidareutveckling av databasen som ligger till grund för verksamhetens system. Aktivitetsloggen innehåller exempelvis en förflyttning av antal produkter i saldo från lagerplats till ett ordernummer.

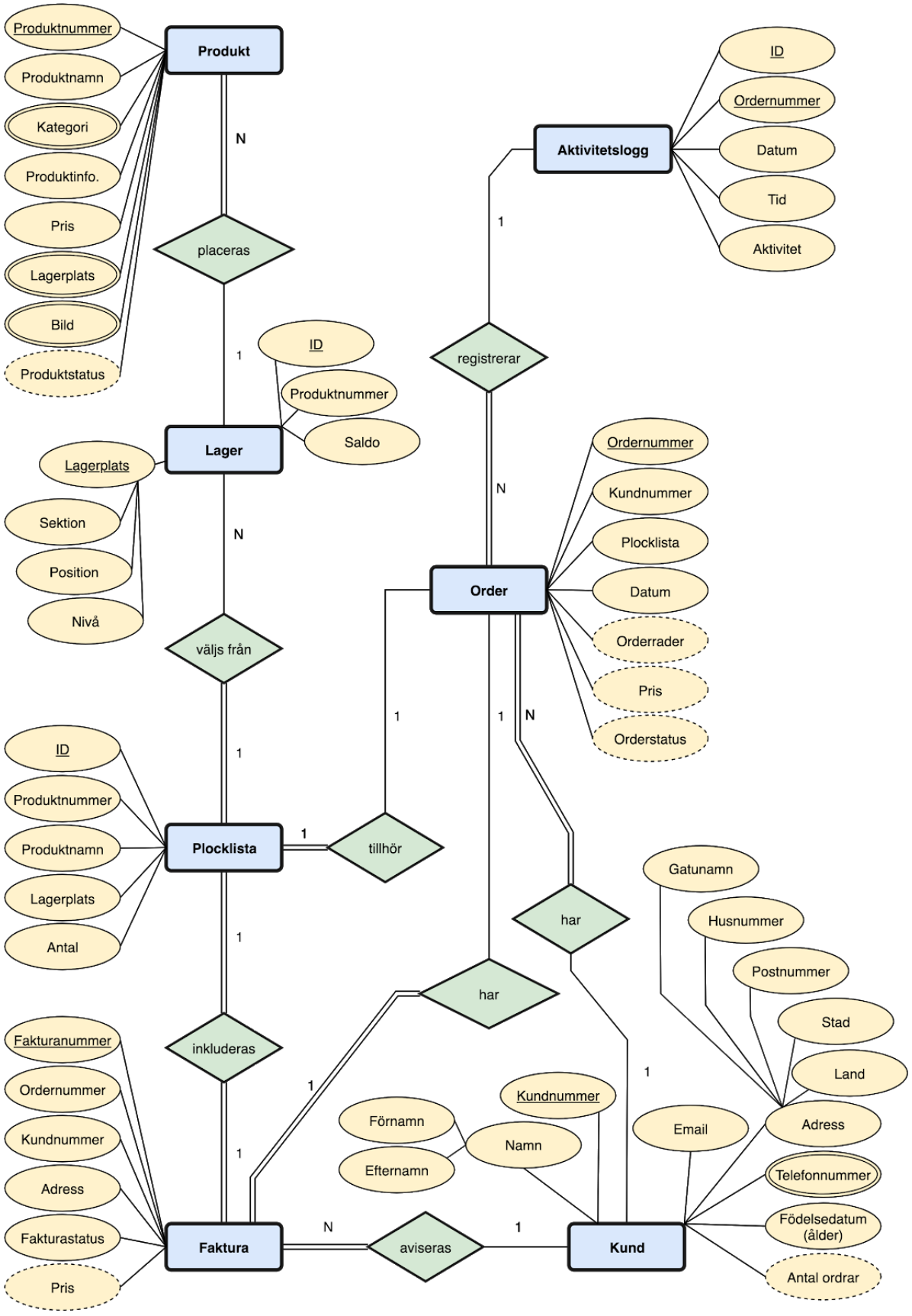
Entiteter

1. Produkt
2. Lager
3. Order
4. Plocklista
5. Kund
6. Faktura
7. Aktivitetslogg

Relationer

ENTITETER	**Lager**	**Produkt**	**Order**	**Plocklista**	**Kund**	**Faktura**
Lager (L)		innehåller		tar från L saldo		
Produkt (P)	finns i		läggs till av kund i	läggs till av kund i		
Order (O)				har	görs av	har
Plocklista (PL)	Tar från L	innehåller en/flera	tillhör en			inkluderas i
Kund (K)	väljer P från L-saldo	beställer	gör en	gör ett urval P som blir		får en F baserat på O
Faktura (F)			baseras på	inkluderar PL	skickas till	

ER-diagram


Produktnummer
Produktnamn
Kategori
Produktinfo.
Pris
Lagerplats
Bild
Produktstatus
Produkt
N
placeras
1
ID
Produktnummer
Saldo
Lager
Lagerplats
Sektion
Position
Nivå
N
väljs från
1
ID
Produktnummer
Produktnamn
Lagerplats
Antal
Plocklista
1
tillhör
1
1
inkluderas
1
Fakturanummer
Ordernummer
Kundnummer
Adress
Fakturastatus
Pris
Faktura
N
aviseras
1
Kund
Förnamn
Efternamn
Namn
Kundnummer
Email
Adress
Telefonnummer
Födelsedatum (ålder)
Antal ordrar
Gatunamn
Husnummer
Postnummer
Stad
Land
har
1
har
1
registrerar
N
Order
1
1
N
Ordernummer
Kundnummer
Plocklista
Datum
Orderrader
Pris
Orderstatus
Aktivitetslogg
1
ID
Ordernummer
Datum
Tid
Aktivitet

Kardinalitet

Entitet 1	Kardinalitet	Entitet 2
Lager	1 : N	Produkt
Produkt	N : 1	Plocklista
Plocklista	1 : 1	Order
Plocklista	1 : 1	Faktura
Kund	1 : N	Faktura
Kund	1 : N	Order
Order	1 : 1	Faktura
Order	N : 1	Aktivitetslogg

Lager innehåller en eller flera produkter. Produkter har fullständigt deltagande i lager, dvs om produkt finns måste den finnas i lagret.
Från lagret tas en eller flera produkter och läggs till en plocklista. Plocklistan har ett fullständigt deltagande eftersom den måste innehålla produkter för att existera.
En plocklista tillhör en order med fullständigt deltagande eftersom utan produkter kan inte ordern existera.
En plocklista har fullständigt deltagande med en faktura och en faktura måste inkludera en plocklista för att kunna existera.
En order har en faktura och en faktura har fullständigt deltagande i en order.
En eller fler fakturor aviseras till en kund och fakturan har fullständigt deltagande dvs att en faktura måste vara till en kund.
En eller flera ordrar är från en kund och ordern har fullständigt deltagande dvs att varje order måste vara från en kund.
En eller flera ordrar registreras i aktivitetsloggen och ordern har fullständigt deltagande.

Attribut

Entitet	Attribut	Kandidatnycklar

Lager	• ID
	• Lagerplats
	  ○ Hylla
	  ○ Position
	  ○ Nivå
	• Produktnummer
	• Produktnamn
	• Saldo
		• ID
		• Lagerplats

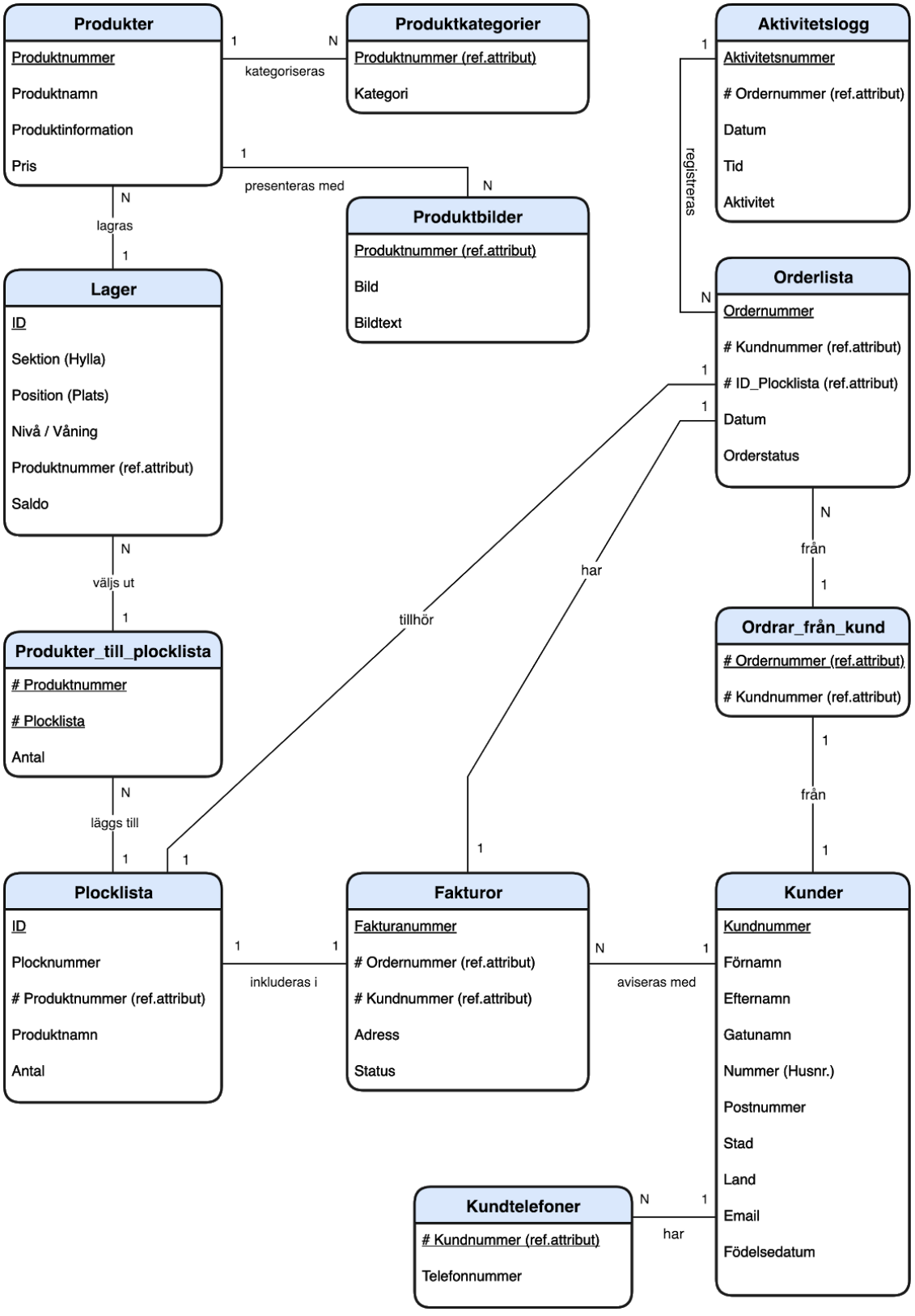
Produkt	• Produktnummer • Produktnamn • Kategori (er) • Produktinformation • Bild (er) • Lagerplats • Pris • Status	• Produktnummer
Order	• <u>Ordernummer</u> • <u>Kundnummer</u> • Plocklista • Datum • Pris • Orderstatus	• Ordernummer

Kund	• Kundnummer
	• Namn
	  ○ Förnamn
	  ○ Efternamn
	• Adress
	  ○ Bostadsadress + husnummer
	  ○ Postnummer
	  ○ Stad
	  ○ Land
	• Email
	• Telefonnummer
	• Födelsedatum (Ålder)
	• *Orderantal* (Härlett attri.)
		• Kundnummer

Faktura	• Fakturanummer • Ordernummer • Kundnummer • Adress • Status	• Fakturanummer
Plocklista	• Produktnummer • Produktnamn • Lagerplats • Antal	• ID
Aktivitetslogg	• Aktivitetsnummer (ID) • Ordernummer • Datum • Aktivitet	• Aktivitetsnummer • Ordernummer

Logisk modellering

Relationsmodellen



Tabeller och attribut

- Produkter (*produktnummer, produktnamn, produktinformation, pris*)
- Produktbilder (*produktnummer, bild, bildtext*)
- Produktkategorier (*produktnummer, kategori*)
- Lager (*ID, sektion, position, nivå, produktnummer, saldo*)
- Orderlista (*ordernummer, kundnummer, plocklista, datum, orderstatus*)
- Ordrar_från_kund (*ordernummer, kundnummer*)
- Kunder (*kundnummer, förnamn, efternamn, gatunamn, nummer, postnummer, stad, land, email, födelsedatum*)
- Aktivitetslogg (*aktivitetsnummer, ordernummer, datum, tid, aktivitet*)
- Produkter_till_plocklista (*produktnummer, plocklista, antal*)
- Plocklista (*ID, produktnummer, produktnamn, antal*)
- Fakturor (*fakturanummer, ordernummer, kundnummer, adress, status*)

Primära & främmande nycklar

Tabell	Primärnyckel	Främmande nycklar
Produkter	Produktnummer	
Produktkategorier	Produktnummer	
Kunder	Kundnummer	
Lager	id	
Produkt_på_lagerplater	Produktnummer	Lager (id)
Produkter_till_plocklista	Plocklista (id)	Produktnummer
Produktbilder	Produktnummer	
Plocklista	plocklista (id)	Produktnummer
Orderlista	Ordernummer	Kundnummer Plocklista Fakturanummer
Ordrar_från_kund	Ordernummer	Kundnummer
Kundtelefoner	Kundnummer	
Fakturor	Fakturanummer	Kundnummer Ordernummer
Aktivitetslogg	Aktivitetsnummer	Ordernummer

Modifieringar / Rättelser

Jag hittade några saker i min *konceptuella modellering* som jag inte tror fungerar i den *fysiska modellen* för att jag möjligtvis angivet/tänkt på fel sätt tidigare så nu har jag försökt kompenserat för det. Det största felet var att jag hade fel ordning på **produkter** och **lager** i flödet. För att kunna lösa mina initiering av databasen genom filerna *setup.sql* och *ddl.sql* behöver jag ändra ordningen till det omvända.
Ett annat fel som skulle uppstå var att en *produkt* kunde inte ligga på fler *lagerplatser* eftersom *lagerplats* var placerat på *produkter*. Eftersom jag vill uppnå möjligheten till en *produkt* på en/flera *lagerplatser* och inte en *lagerplats* med en/flera *produkter* på ändras nu ordningen.
Ett annat fel jag tidigare gjort var *sambandet* mellan **produkter** och **ordrar** vilket i verkligheten kan stämma att ordrar innehåller produkter men här tänker jag att kunden skapar en **plocklista** som i sin tur blir den order som beställs. Ordern innehåller dock endast information som knyter alla delarna, **plocklista**, **faktura** och **kund** tillsammans.
För att skapa en hållbar bild av min databas ändra jag därför min *konceptuella modell* även om det har under föreläsning sagts att man kan kompensera för fel i övergången till den logiska modellen utan att gå tillbaka och ändra, för min del handlar det även om förståelse så jag lägger lite extra arbete på det.
Andra fel jag märkte var, i entitetstypen **Lager** var attributet *lagerplats* ett sk *flervärt attribut* angivet med dubbelring runt men det går inte eftersom varje *lagerplats* är unik.
Jag tänkte fel från början med hur jag får en **produkt** att kunna vara på fler *lagerplatser* men det löser jag enklare med samma *produktnummer* på fler rader i **lager**tabellen istället.
Liknande problem finns i entitetstypen **produkter** där attributet *produktinformation* även ska vara ett *vanligt attribut* istället för ett *flervärt attribut* som tidigare angivet.
Möjligheten finns att man kanske vill dela upp informationen i fler värden i framtiden beroende på hur seriöst informationen ska skrivas om produkten.
För att kunna lösa ett problem med hur många produkter som kan tas från **lagret** och läggas till **plocklistor** behövde även *sambandstypen* mellan **lager** och **plocklista** ändras till ett N:M (*många-till-många*) -samband.
I några av *enhetstyperna* som är angivna har jag tänkt att man behöver ett *totalpris* som *attribut* men när jag tänker vidare på det så blir det sk *härledda attribut* från det som beställts där man multiplicerar styckpriset per produkt med antalet och sen adderar samman totalen. I *relationsmodellen* ska jag inte ange den typen av *attribut* så de har tagits bort i översättningen. Det blev väldigt tydligt var jag tidigare har tänkt fel när jag läste kapitel 6 i boken *databasteknik* som beskriver hur man översätter den *konceptuella modellen* till en *logisk modell* på ett bra sätt. Det är inte helt säkert att min modell nu fungerar men jag tror att den håller bättre än tidigare.

Anpassningar / Översättning till relationsmodellen

Från konceptuell modell till logisk behöver jag göra några anpassningar för att resultatet ska följa relationsmodellen. För tabellen **produkter** har jag i den konceptuella modellen åtta attribut av tre olika typer, vanliga attribut, flervärt attribut och härledda attribut. Först så tar jag bort de härledda attributen eftersom de värden skapas med hjälp av värden i andra kolumner och i mitt fall är det *produktstatus* som ska indikera exempelvis om produkten finns, behöver beställas eller kanske är utgående ur sortimentet. Det baseras på saldot för produkten i tabellen lager och kan bättre placeras i en anpassad vy med en join. Kandidatnycklar för tabellen dvs kolumner med unika värden är produktnummer och väljs därför som primärnyckel för **produkter**. De flervärda attributen *kategori, lagerplats* och *bild* som ska kunna hålla flera olika värden för samma produkt blir egna tabeller. Undantaget är *lagerplats* som tas bort eftersom vi placerar *produktnummret* i tabellen **lager** men på fler olika *lagerplatser*. Tabellerna skapas är sk *attributtabeller* och kommer i sin tur kunna presentera produkten och de värden som det flervärda attributet har. Vi undviker ett problem med normalisering där tabellen för **produkter** annars kan bli svår att tyda. De två tabeller som skapas är **produktkategorier** och **produktbilder**. **Produktkategorier** har attributen *produktnummer* och *kategori. Produktnummer* blir primärnyckel för tabellen **produktkategorier**. **Produktbilder** har attributen *produktnummer, bild* och *bildtext* och primärnyckeln är även där *produktnummer.*

I tabellen **lager** har jag sex kolumner där de första tre är *id, produktnummer* och *saldo. Id* är ett tilltaget attribut för att skapa en unik numerisk nyckel att använda för varje lagerplats och blir därav primärnyckeln för tabellen. Jag väljer att använda det även för att försöka uppnå en högre normaliseringsgrad och göra det enklare att referera till lagerplatserna från andra tabeller. Om mitt val är korrekt har jag kanske även undvikit några framtida problem genom att uppnå en högre normalisering. *Produktnumret* är referensattribut, angivet med foreign key, till produkten som finns placerad på lagerplatsen. Eftersom jag vill att lagerplatsen ska vara ett sammansatt attribut som återspeglar *hyllan* produkten står i, *positionen/platsen* i hyllan, och *höjdnivån/våningen* på den platsen behöver jag enligt relationsmodellen göra en kolumn för vardera attribut som det sammansatta attributet består av. Jag antar att jag genom en vy för plocklistan kan placera lagerplatsen sammansatt som önskat men i min tabell blir det tre kolumner *sektion, position* och *nivå. Saldot* visar antalet av produkten som finns på varje lagerplats.

För att översätta sambandet mellan **lager** och **plocklista** som är ett många-till-många-samband (N:M) där många produkter från lagret ska kunna läggas till många olika plocklistor behöver jag skapa en *sambandstabell* och den kallar jag **produkter_till_plocklista**. I sambandstabellen behöver jag två unika attribut som refererar till **lager** och **plocklista** för att skapa kopplingen. Attributen i listan blir därför totalt tre, *produktnummer, plocklista* och *antal*. Primärnyckeln för tabellen blir *plocklista.* Främmande nyckel blir *produktnummer.*

Nästa tabell blir **plocklista** och består av produkter som kunden väljer i verksamhetens eshop. Även om kunder väljer produkter så väljer hen dem från vad som finns på lager och

utifrån det saldo som finns där. **Plocklistan** har fem attribut, *id, produktnummer, produktnamn, lagerplats* och *antal*. Plocklistan ligger till grund för att en order av produkter ska vara möjlig och den inkluderas i fakturan med exempelvis en anpassad vy som kan kallas produktspec. eller annat passande namn för att visa kunden vad som har skickats och fakturerats för. Den används även till att summera orderns totala pris som specas i fakturan och ordern via härledda attribut. Primärnyckeln för plocklista är id. Referensattribut till andra tabeller är *produktnummer* och *lagerplats*.

Precis som nämnt finns en tabell för ordrar, **orderlista** och den har i den konceptuella modellen sju attribut som i den logiska blir fyra då tre är härledda tas bort. Attributen är *ordernummer, kundnummer, plocklista* och *datum*. De tre härledda attribut som inte blir kolumner är *orderrader, pris* och *orderstatus.* Primärnyckeln för tabellen blir *ordernummer* och främmande nycklar är *kundnummer* och *plocklista*.

Det är klart att varje order behöver en kund för att kunna ta betalt för de produkter som beställts så tabellen **kunder** skapas för det. **Kunder** har tio attribut och är nuvarande eshopens största tabell i mängd olika attribut. Här finns tre olika typer av attribut likt tabellen *produkter* och de behandlas lika här för att övergå till en logisk modell. De härledda attributen tas inte med, det flervärda attributet blir en egen tabell och de sammansatta tabellerna delas upp och samtliga attribut blir istället en kolumn per attribut. Attributen som tabellen slutligen består av är *kundnummer, förnamn, efternamn, gatunamn, nummer, postnummer, stad, land, email* och *födelsedatum.* Eftersom vi vill kunna lagra fler olika telefonnummer till en kund blir det en egen tabell, **kundtelefoner** som har attributen *kundnummer* och *telefonnummer*. Primärnyckeln för kunder *kundnummer* i både **kunder** och **kundtelefoner**. Referensattribut/främmande nycklar är *ordernummer, kundnummer* och *packsedel* (refererar till plocklistans primärnyckel). Referens attributen finns endast på tabellen **kunder**.

Den sista tabellen är **aktivitetsloggen** och efter vidare läsning i boken *databasteknik* tror jag inte att den kommer inkluderas i databasen *eshop* eftersom det finns goda skäl till att lägga det i en egen databas separerat från databasen som den ska logga. Boken talar för att placera loggning separat för att göra det säkrare och snabbare. Om databasen behöver göra ändringar och logga de ändringarna kan tar det dubbelt så lång tid att utföra de aktiviteterna. Ett bättre sätt är att två database, hällst separerade från varandra på två olika maskiner, sköter det tillsammans med en som loggar aktiviteterna i den andra och då inte hindrar eller tynger ner flödet på något vis. Jag har dock tagit beslutet att låta min aktivitetslogg ligga kvar mer som en påminnare att jag vill skapa en bra loggning av aktiviteten i min databas. Det blir en tabell som heter **aktivitetslogg** och har fyra attribut, *aktivitetsnummer, ordernummer, datum, tid* och *aktivitet.* Primärnyckeln för tabellen är *aktivitetsnummer.*

Fysisk modellering

Skapandet av databasen eshop

Jag har skapat databasen manuellt genom SQL-kod baserad på ovan presenterade ER-modell. För att skapa databasen har två filer skapats, *setup.sql* och *ddl.sql.* Filen setup skapar *databasen*, databasens *användare* med *lösenord* och tilldelar *rättigheter* till användarna. Filen *ddl.sql* arbetar i två steg, först tar vi bort alla tabeller från databasen om de existerar sen skapar vi dem. Existerar det vyer gör vi lika med dem som för tabeller och vi tar bort allt först innan vi skapar något. Tabeller skapas innan vyer eftersom vyer baseras på tabeller men för nu så existerar inga vyer i databasen så den biten lämnas tom tills vidare behov. När behovet uppstår för en/flera vyer ska de behandlas lika tabeller i filen *ddl.sql*. För att se exakt hur jag gjort, var god se Appendix (SQL) som ligger i slutet av det här dokumentet.
För att även försöka underlätta för mig själv och kanske andra som möjligtvis kommer arbeta med databasen i framtiden har jag även skapat en *reset_db.bash* fil för återställning av databasen. Filen kör automatiskt filerna *setup.sql* och *ddl.sql* för databasen *eshop.* När filerna har har körts utförs kommandot `describe <tabellens namn>;` för varje tabell så att *tabellerna*, deras *nycklar*, tillåtelse för *null-värden* mm presenteras. Sist körs förfrågningen `show tables;` för att visa alla tabeller i databasen.

Funktionsstöd som databasen ska innehålla

1. Lägga till nya kunder.
2. Uppdatera existerande kunder.
3. Radera existerande kund.
4. Visa samtliga registrerade kunder.
5. Söka kunder via information om kunden.
6. Lägga till ny produkt.
7. Uppdatera en existerande produkt.
8. Radera lagerförd produkt.
9. Visa samtliga produkter.
10. Visa produkter enligt kategori.
11. Söka produkter via information om produkten.
12. Placering av produkt i lagret.
13. Uppdatering av produktplacering i lagret.
14. Inleverans av produkter från leverantör/tillverkning.
15. Lägga till nya lagerplatser.
16. Uppdatera lagerplatser.
17. Radera lagerplatser.
18. Visa samtliga lagerplatser och produkterna på plats.
19. Söka på produkt i lagret.
20. Skapa nya plocklistor från produkter som finns lagrade.
21. Uppdatering av plocklista.
22. Lägga till nya ordrar.
23. Uppdatering av order.
24. Makulering av order vilket även innebär återplacering av plockade produkter till lager.
25. Visa samtliga ordrar.
26. Söka via information om ordrar.
27. Skapa ny faktura.
28. Uppdatering av faktura.
29. Registrera aktivitet i databasen till aktivitetsloggen.

Appendix (SQL)

Skapa databasen och användare (setup.sql)

Steg 1 - Skapa databasen “eshop” om den inte redan existerar. Använd sedan databasen “eshop”.

```
create database if not exists eshop;
use eshop;

show databases like "%eshop%";
```

Steg 2 - Ta bort användare om de existerar för att skapa igen. Skapa användare för databasen och ge dem rättigheter till databasen.

```
drop user if exists 'user'@'%';

create user if not exists 'user'@'%'
identified
   with mysql_native_password
   by 'pass'
;

grant all privileges
   on *.*
   to 'user'@'%'
;

show grants for 'user'@%'%';
```

Skapa tabeller och vyer för databasen (ddl.sql)

Steg 1 - Ta bort alla tabeller ifall de existerar, även vyer om de existerar.

```
drop table if exists fakturor;
drop table if exists aktivitetslogg;
drop table if exists orderlista;
drop table if exists order_fran_kund;
drop table if exists kundtelefoner;
drop table if exists kunder;
drop table if exists plocklista;
drop table if exists produktkategorier;
drop table if exists produktbilder;
drop table if exists lager;
drop table if exists produkter;
```

Steg 2 - Skapa tabeller med attribut och nycklar.

Tabellen produkter:

```
create table produkter
(
   produktnummer int unique not null,
   produktnamn varchar(30) not null,
   produktinformation varchar(500) not null,
   pris float not null,

   primary key (produktnummer)
)
engine innodb
charset utf8
collate utf8_swedish_ci
;

describe produkter;
-- show create table produkter \G;
```

Tabellen produktkategorier:

```
create table produktkategorier
(
   produktnummer int not null,
   kategori varchar(30) not null,

   primary key (produktnummer)
)
engine innodb
charset utf8
collate utf8_swedish_ci
;

describe produktkategorier;
-- show create table produktkategorier \G;
```

Tabellen produktbilder:

```
create table produktbilder
(
   produktnummer int not null,
   produktbild varchar(30) not null,
   bildtext varchar(100) not null,

   primary key (produktnummer)
)
engine innodb
charset utf8
collate utf8_swedish_ci
;

describe produktbilder;
-- show create table produktbilder \G;
```

Tabellen lager:

```
create table lager
(
   id int unique not null,
   sektion char(2) not null default 'AA',
   position char(2) not null default '01',
   niva char(2) not null default '01',
   produktnummer int not null,
   saldo int not null,

   primary key (id),
   foreign key (produktnummer) references produkter(produktnummer)
)
engine innodb
charset utf8
collate utf8_swedish_ci
;

describe lager;
-- show create table lager \G;
```

Tabellen kunder:

```
create table kunder
(
   kundnummer int unique not null,
   fornamn varchar(20) not null,
   efternamn varchar(20) not null,
   gatunamn varchar(20) not null,
   nummer varchar(6) not null,
   postnummer varchar(10) not null,
   stad varchar(25) not null,
   land varchar(25) not null,
   email varchar(40) not null,
   fodelsedatum date not null,

   primary key (kundnummer)
)
engine innodb
charset utf8
collate utf8_swedish_ci
;

describe kunder;
-- show create table kunder \G;
```

Tabellen plocklista:

```
create table plocklista
(
   id int not null,
   plockindex int unique not null,
   produktnummer int not null,
   produktnamn varchar(30) not null,
   lagerplats int not null,
   antal int not null,

   primary key (id),
   foreign key (produktnummer) references produkter(produktnummer),
   foreign key (lagerplats) references lager(id)
)
engine innodb
charset utf8
collate utf8_swedish_ci
;

describe plocklista;
-- show create table plocklista \G;
```

Tabellen orderlista:

```
create table orderlista
(
   ordernummer int unique not null,
   kundnummer int not null,
   plocklista int not null,
   datum date not null,
   orderstatus varchar(20) not null,

   primary key (ordernummer),
   foreign key (kundnummer) references kunder(kundnummer),
   foreign key (plocklista) references plocklista(id)
)
engine innodb
charset utf8
collate utf8_swedish_ci
;

describe orderlista;
-- show create table orderlista \G;
```

Tabellen order_fran_kund:

```
create table order_fran_kund
(
   ordernummer int unique not null,
   kundnummer int not null,

   primary key (ordernummer),
   foreign key (kundnummer) references kunder(kundnummer)
)
engine innodb
charset utf8
collate utf8_swedish_ci
;

describe order_fran_kund;
-- show create table order_fran_kund \G;
```

Tabellen kundtelefoner:

```
create table kundtelefoner
(
   kundnummer int not null,
   telefonnummer int unique not null,

   primary key (kundnummer)
)
engine innodb
charset utf8
collate utf8_swedish_ci
;

describe kundtelefoner;
-- show create table kundtelefoner \G;
```

Tabellen fakturor:

```
create table fakturor
(
   fakturanummer int unique not null,
   ordernummer int not null,
   kundnummer int not null,
   packsedel int not null,
   adress varchar(30) not null,
   fakturastatus varchar(20) not null,

   primary key (fakturanummer),
   foreign key (ordernummer) references orderlista(ordernummer),
   foreign key (kundnummer) references kunder(kundnummer),
   foreign key (packsedel) references plocklista(id)
)
engine innodb
charset utf8
collate utf8_swedish_ci
;

describe fakturor;
-- show create table fakturor \G;
```

Tabellen aktivitetslogg:

```
create table aktivitetslogg
(
   id int unique not null,
   ordernummer int not null,
   datum date not null,
   tid time not null,
   aktivitet varchar(200) not null,

   primary key (id),
   foreign key (ordernummer) references orderlista(ordernummer)
)
engine innodb
charset utf8
collate utf8_swedish_ci
;
```

Steg 3 - Skapa vyer

Nu finns inga vyer i databasen men de ska vara här när de kommer till behov att skapa dem. Notera även att om vyer existerar ska även de tas bort först om de existerar i filen *ddl.sql* för att sedan skapas efter att alla tabeller har skapats.