# Databasmodellering

Olov Wimark 2020

# Innehållsförteckning

# Konceptuell modell

Jag försöker att skapa en generisk databas som med rätt små förändringar skulle kunna användas i ett annat företag.

I det här arbetet följer jag kokboken[1]. Det som blir tydligt för mig är att arbetet behöver flöda lite framåt och bakåt mellan de fastlagda stegen. Jag kanske inser att något saknas, eller att något är överflödigt. I den här rapporten väljer jag att visa det, jag redigerar inte bort ändringarna.

## Företaget Endgame Keyboard

Jag skall skapa ett försäljningssystem för ett bolag som säljer tangentbord. I huvudsak vänder man sig till kännare och människor som jobbar mycket med just tangentbord. Det skall finnas färdiga varianter av tangentbord, custom-set som levereras färdigmonterade och kit för den som vill montera själv.

Det skall också finnas lösa delar. Brytare, tangenter, kretskort, fötter, stabiliserare och chassi, avsett för den som verkligen vill modda sitt tangentbord.

Till skillnad från liknande försäljare på annat håll så riktar man in sig mot den nordiska marknaden.

## Steg 1: Beskriv databasen i ett textstycket

Jag passar på att stryka under nyckelord som kommer att vara användbara i databasen – entiteter som ofta blir databasens tabeller

Vår databas skall hantera ett kundregister och ett produktregister. För att underlätta för kunden vore det bra med produktkategorier, och produktbilder.

Databasen skall också innehålla ett lager där man kan se vilket antal av produkter som finns tillgängliga och en, eller flera hyllnoteringar.

Kunden skall kunna skapa en order, som förutom önskade produkter innehåller kundens detaljer.

Utifrån ordern skall en plocklista skapas. Den innehåller samma information som ordern, plus uppgifter om var produkten finns i lagret.

---

1 https://dbwebb.se/kunskap/kokbok-for-databasmodellering

När leveransen skickas skall en faktura skapas.

Systemet skall ha en logg.

## Steg 2: Skriv ner alla entiteter

Jag skriver ner de understrukna nyckelord som jag fann i steg 1. Redan nu inser jag att det kommer att behövas korstabeller och jag hoppar tillbaka till steg ett för att lägga till produktkategorier

- Kundregister
- Produktregister
- Produktkategorier
- Produktbilder
- Lagerstatus
- Hyllplaceringar
- Order
- Plocklista
- Faktura
- Logg

## Steg 3: Skriv ner alla relationer och visa i en matris

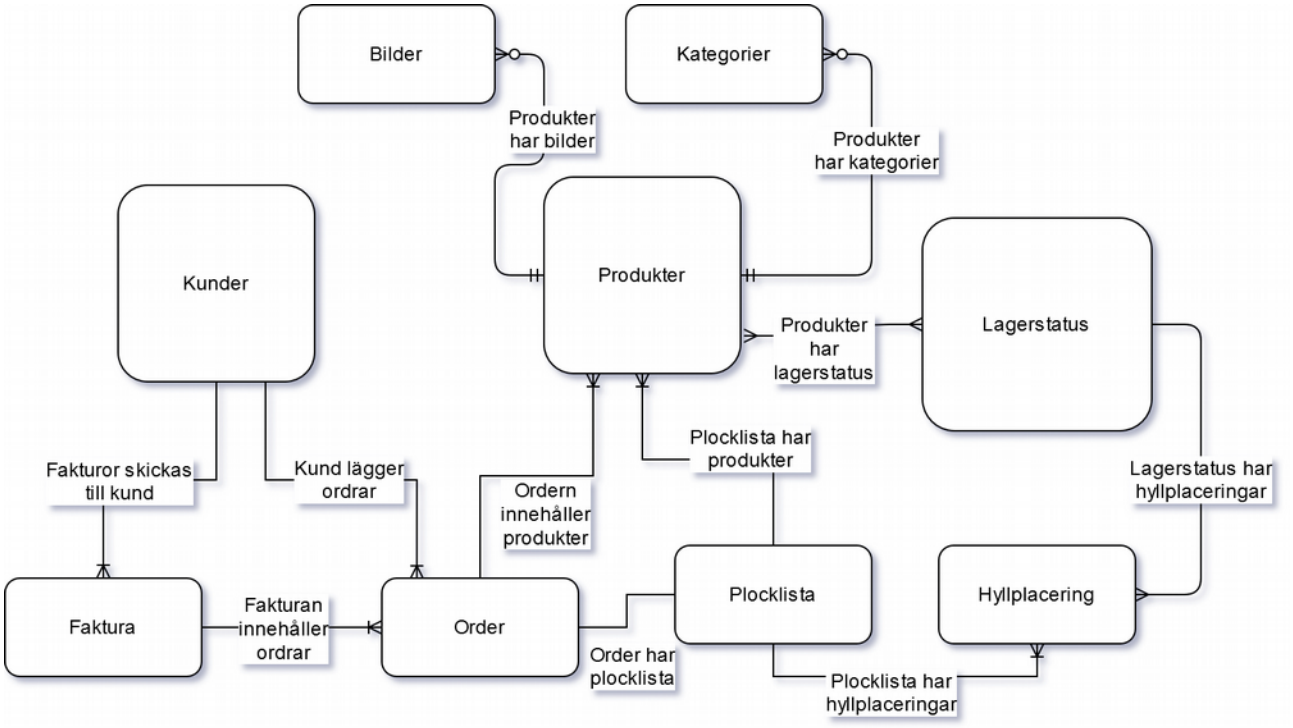
Även här finner jag en entitet som saknas: produktbilder. Det kommer behövas flera bilder per produkt så jag lägger till ett bildregister. Då bilderna kommer att associeras med en enskild produkt så tror jag inte att det kommer behövas någon korstabell där. Kanske blir det tydligare efterhand?

| **Entiteter** | **Har** | |
|---|---|---|
| Produkter | Kategorier | Bilder |
| Lager | Lagerstatus | Hyllplacering |
| Kunder | Ordrar | Fakturor |
| Order | Plocklista | Faktura |
| Logg | | |

Jag behöver bryta ut loggen en smula här. Den kommer att registrera när en order läggs, när en plocklista skapas, när en leverans skickas och när en faktura betalas. Många transaktioner förs in i loggen så den behöver skugga flera av tabellerna; kunder, produkter, order, faktura. Å andra sidan behöver den inte ge data tillbaka till övriga tabeller. Den får stå för sig själv tills vidare.

## Steg 4: Rita ER-diagram med entiteter och relationer

Jag sätter ihop ett diagram med en enkel skiss över relationerna i databasen.

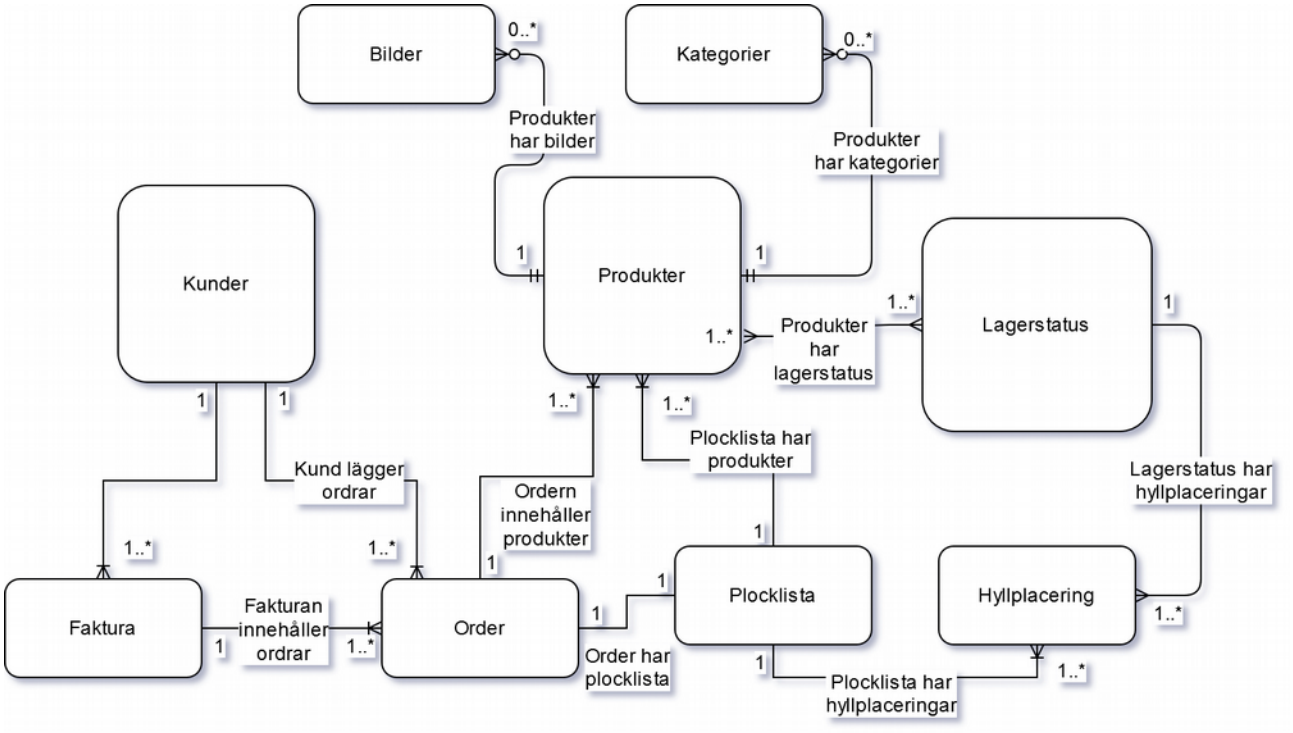



Ett enkelt ER-diagram. Jag kan ana en struktur för lagringen, och hur den kommer att relatera till en trolig arbetsgång.

## Steg 5: Komplettera med Kardinalitet

Nedan försöker jag få en uppfattning om hur de olika entiteterna hör ihop. Till exempel kan jag uttyda följande:

- En kund kan ha flera fakturor utställda till sig.
- En kund kan också ha flera lagda ordrar.
- En order innehåller en eller flera produkter
- En order genererar en plocklisa.

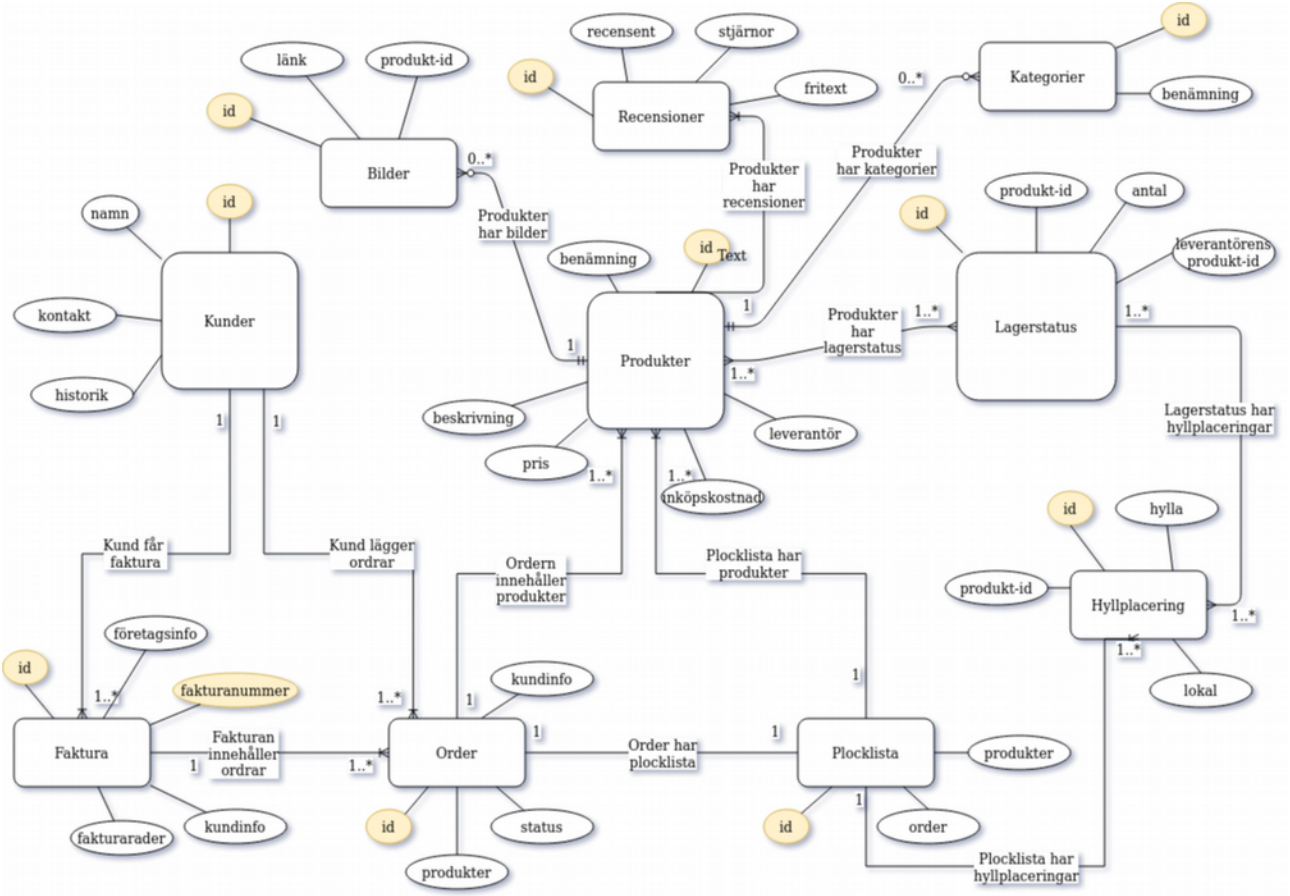



## Steg 6: Attribut och kandidatnycklar

Jag avslutar den konceptuella fasen med att ta fram förväntade attribut och försöker att identifiera kandidatnycklar. Jag använder beskrivningen vid steg 1 för att finna attribut. Jag börjar med att rada upp dem i textformat. Jag lägger till tabellen recensioner och funderar på att ta bort tabellen hyllplacering. Varje produkt kan behöva ha flera hyllplaceringar, så jag behåller den tabellen. Jag inser att jag förmodligen behöver en del korstabeller; fakturaposter, produktrecensioner, kundfaktura. Förmodligen flera, men det kommer jag att reda ut i den logiska fasen. Det är svårt att hålla isär stegen.

- Kund (id, namn, adress, postadress, e-post, telefon, historik)
- Fakturor (id, fakturanummer, fakturarader, kundinfo, företagsinfo)
- Order (id, kund-id, produkter, status)

- Produkter (id, benämning, beskrivning, bild,pris, inköpskostnad, leverantör)
- Recensioner (id, recensent, betyg, fritext)
- Lagerstatus (id, produkt-id, antal, leverantörens produkt-id)
- Hyllplacering (id, produkt, lokal, hylla)
- Plocklista (id, order-id, produkter, hyllplaceringar)
- Bilder (id, länk till bilden, produkt-id)
- Kategorier (id, benämning)



# Logisk modell

Nu börjar jag titta på hur databasen skall sitta ihop. Jag börjar med att rita om diagrammet i grunden. Jag flyttar in attributen i tabellerna, och kikar vidare på relationerna.

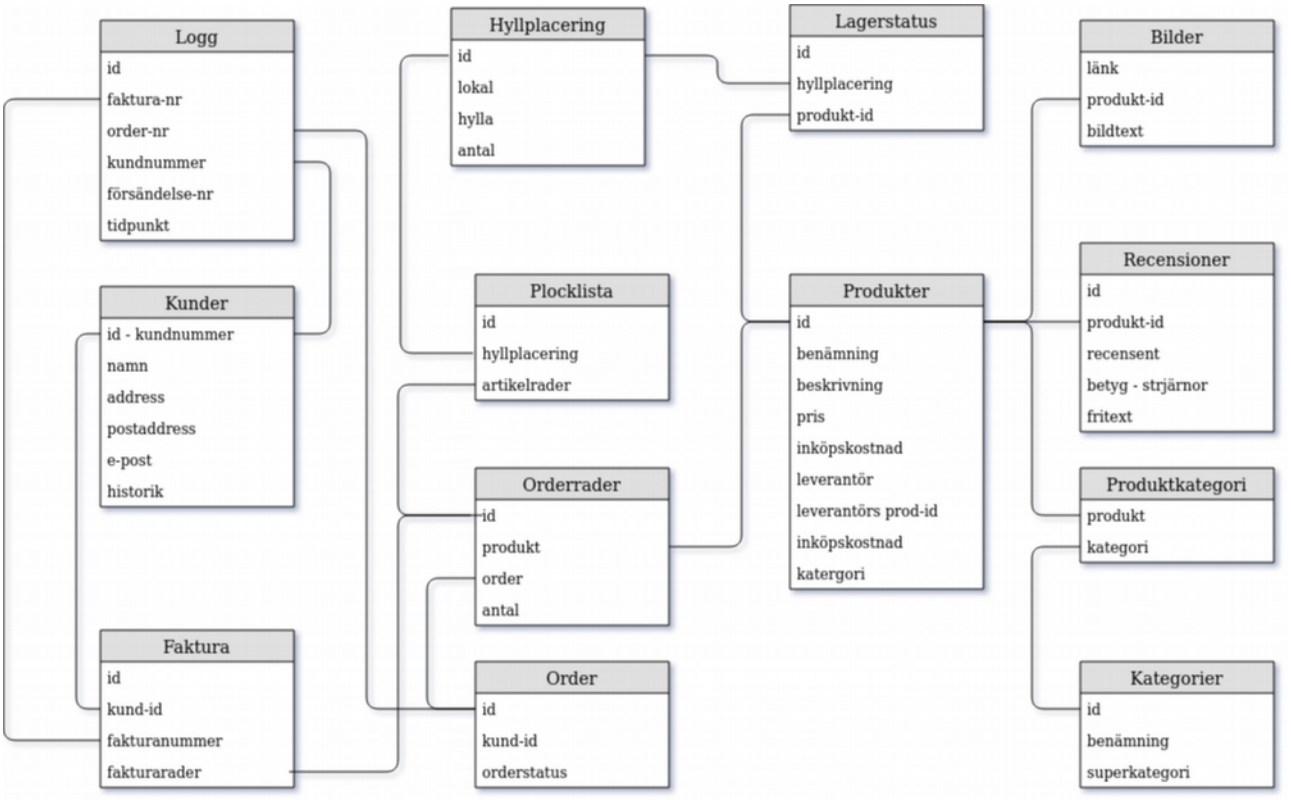
## Steg 7: Relationsmodellen

Jag formulerar om tabellen ordentligt med nya korstabeller för att bryta upp alla en till många-relationer.

Jag ändrar om i attributen en del. Jag vill ha helt unika attribut, det är dumt att mata in samma data på flera platser.

Jag har också varit ganska noga med att ge varje tabell en id-kolumn. I vissa fall kanske det hade räckt med till exempel ett fakturanummer, men vad händer om man måste göra justeringar? Kanske kan samma faktura förekomma i flera versioner?

I korstabellerna blir kombinationerna unika. Ett produkt-id möter bara ett kategori-id en gång. Produktkategori skall bilda unika rader.

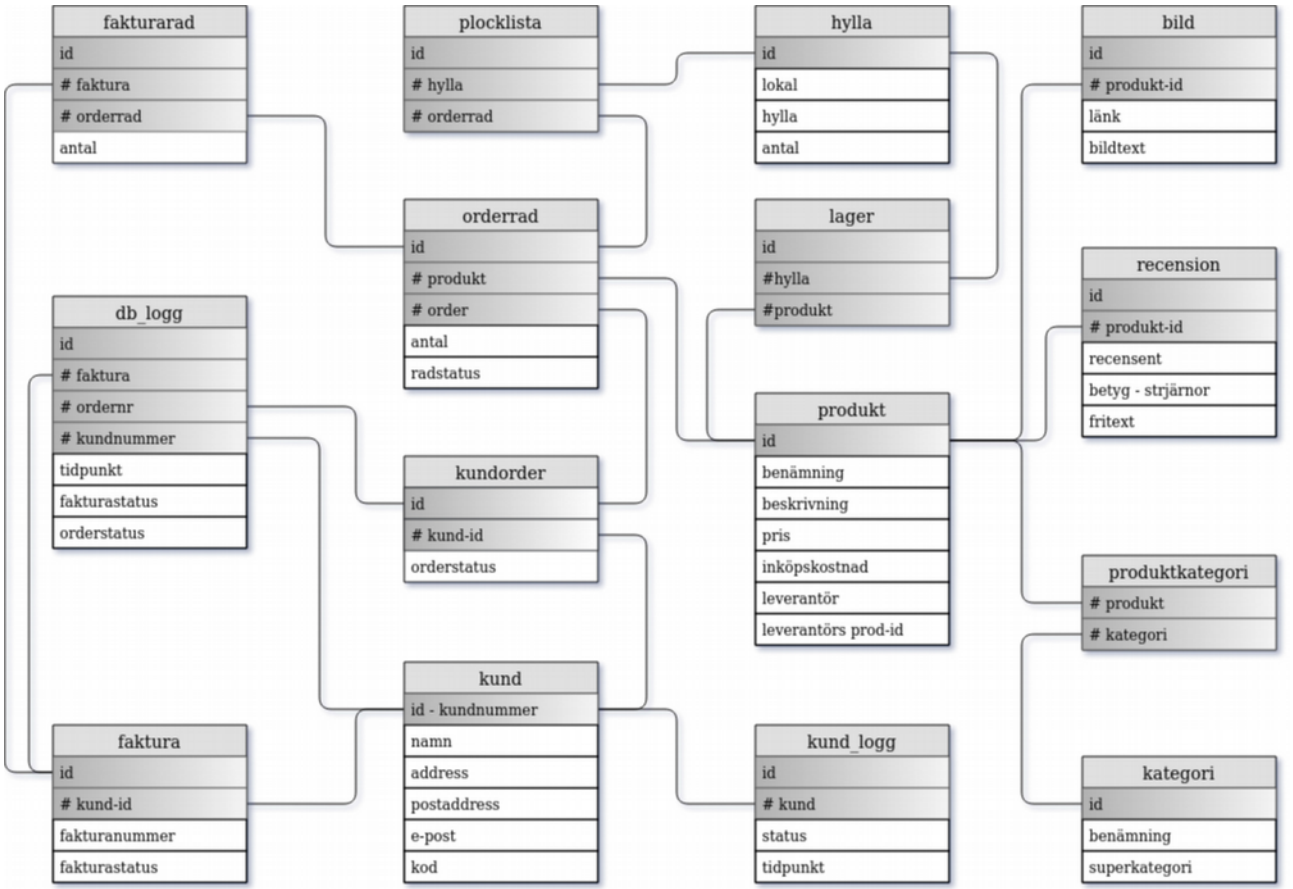


Jag inser att jag missat möjlighet för en användare att logga in. Jag lägger till fältet *kod* i kund-tabellen. In- och utloggningar skulle behöva sparas i loggen, eller i alla fall i en logg, det är säkrare än att förlita sig helt på session-data och cookies.

## Steg 8: Nycklar och attribut

Jag fyller på med unika nycklar för varje entitet, förutom där jag tror att det kommer räcka med sekundärnycklar. Jag raderar någon primärnyckel, och försöker göra diagrammet överskådligt genom att flytta runt tabellerna.

Nu funderar jag också på allvar över vad tabellerna skall heta. ”Order” lär inte fungera jättebra. Vissa namn är onödigt långa, jag skall avstå svenska tecken och bara skriva med gemener.




Kolumnen fakturarader i tabellen kommer att hämtas från färdiga orderrader. Det känns som någon slags kopieringsmanöver, och inte som att de skall hänga ihop dynamiskt. Över huvud taget skall fakturan ha så få länkar som möjligt, det underlättar för oss när jag läser loggen sedan.

# Fysisk modell

Nu får jag sätta oss och koda igen. Alla tankar och idéer skall omsättas i riktiga verktyg.

## Steg 9: Skapa SQL DDL för tabellerna

För att vara noga med teckenkodning skapar jag databasen *Webshop* med fastställd CHARSET och COLLATE. I min DDL-fil fastställer jag vilken databasmotor som skall användas.

Min kod ligger som appendix 1. Jag väljer att koda för hand.

När jag skriver sekundärnycklar behöver jag vara noga och se till att de tabeller som jag hänvisar till redan existerar. Jag skriver stycket med REMOVE IF EXISTS sist, men placerar den överst i filen. Ordningen spelar roll för att jag inte skall få referenskonflikter. De tabeller som har flest länkar raderas först.

Kunder, produkter och hylla har inga sekundärnycklar så jag skapar dem först.

## Steg 10: API

Den här processen går runt i cirklar. Den sista, och längsta cirkeln knyter an till steg 1, beskriv databasen i ett textstycke. Där finns ett par önskemål som jag kan översätta i funktioner som kunden önskar. Jag formulerar dem mer linjärt.

- En kund skall kunna skapa ett konto
- En kund skall kunna logga in och logga ut
- En kund skall kunna lägga en order, med webbetalning
- Ordern skall skickas till lagret
- Ordern skall plockas – procedurer för att scanna varor
- Leveransen skall skickas, faktura eller kontantfaktura skall bifogas
- Lagerinköp skall registreras – procedurer för att scanna varor

Dessutom skall jag kunna skapa diverse rapporter

- Kundreskontra
- Beställningshistorik
- Lagerstatus
- M. fl.

# Appendix 1: SQL DDL

```
-- Standarder
USE webshop;
SET NAMES `utf8`;
SET default_storage_engine=InnoDB;

-- Töm databasen i omvänd ordning
DROP TABLE IF EXISTS bild;
DROP TABLE IF EXISTS recension;
DROP TABLE IF EXISTS plocklista;
DROP TABLE IF EXISTS fakturarader;
DROP TABLE IF EXISTS faktura;
DROP TABLE IF EXISTS orderrad;
DROP TABLE IF EXISTS kundorder;
DROP TABLE IF EXISTS lager;
DROP TABLE IF EXISTS
     produktkategori;
DROP TABLE IF EXISTS kategori;
DROP TABLE IF EXISTS hylla;
DROP TABLE IF EXISTS produkt;
DROP TABLE IF EXISTS kund;

-- Skapa tabeller utan
-- sekundärnycklar
CREATE TABLE kund
(
    kundnummer INT AUTO_INCREMENT
           NOT NULL PRIMARY KEY,
    namn VARCHAR(20) NOT NULL,
    address VARCHAR(20),
    postaddress VARCHAR(20),
    epost VARCHAR(30),
    telefon CHAR(14)
);

CREATE TABLE produkt
(
    id INT AUTO_INCREMENT NOT NULL
           PRIMARY KEY,
    benamning CHAR(20) NOT NULL,
    beskrivning TEXT,
    pris DOUBLE(6, 2),
    inkopskostnad DOUBLE(6, 2),
    leverantor VARCHAR(20),
    lev_prod_nr VARCHAR(20)
);

CREATE TABLE hylla
(
    id INT AUTO_INCREMENT NOT NULL
           PRIMARY KEY,
    lokal VARCHAR(20),
    hylla VARCHAR(8),
    antal INT
);

CREATE TABLE kategori
(
    id INT AUTO_INCREMENT NOT NULL
           PRIMARY KEY,
    namn VARCHAR(12),
    super VARCHAR(12)
);

-- Skapa tabeller med
-- sekundärnycklar
CREATE TABLE produktkategori
(
    produkt INT,
    kategori INT,
    PRIMARY KEY (produkt,
           kategori),
    FOREIGN KEY (produkt)
           REFERENCES produkt(id),
    FOREIGN KEY (kategori)
           REFERENCES kategori(id)
);



CREATE TABLE lager
(
    id INT AUTO_INCREMENT NOT NULL
           PRIMARY KEY,
    hylla INT,
    produkt INT,
    FOREIGN KEY (hylla)
           REFERENCES hylla(id),
    FOREIGN KEY (produkt)
           REFERENCES produkt(id)
);



-- orderstatus kan vara "inkommen",
-- "plockad","avsänd" el. dyl.
CREATE TABLE kundorder
(
    id INT AUTO_INCREMENT NOT NULL
           PRIMARY KEY,
    kund INT,
    orderstatus VARCHAR(20),
    FOREIGN KEY (kund)
      REFERENCES kund(kundnummer)
);



-- radstatus kan var "klar",
-- "restad", el. dyl.
CREATE TABLE orderrad
(
    id INT AUTO_INCREMENT NOT NULL
           PRIMARY KEY,
    kundorder INT,
    produkt INT,
    antal INT,
    radstatus VARCHAR(12),
    FOREIGN KEY (kundorder)
      REFERENCES kundorder(id),
    FOREIGN KEY (produkt)
      REFERENCES produkt(id)
);
```

```
-- fakturastatus kan vara
-- "skickad","försenad", "inkasso"
-- el. dyl.

CREATE TABLE faktura
(
    id INT AUTO_INCREMENT NOT NULL
     PRIMARY KEY,
    kund INT,
    fakturanummer INT,
    fakturastatus VARCHAR(12),
    FOREIGN KEY (kund) REFERENCES
     kund(kundnummer)
);


CREATE TABLE fakturarad
(
    id INT AUTO_INCREMENT NOT NULL
           PRIMARY KEY,
    faktura INT,
    orderrad INT,
    antal INT,
    FOREIGN KEY (faktura)
           REFERENCES faktura(id),
    FOREIGN KEY (orderrad)
           REFERENCES orderrad(id)
);
```

```
CREATE TABLE plocklista
(
    id INT AUTO_INCREMENT NOT NULL
           PRIMARY KEY,
    orderrad INT,
    hylla INT,
    FOREIGN KEY (orderrad)
           REFERENCES orderrad(id),
    FOREIGN KEY (hylla)
           REFERENCES hylla(id)
);



CREATE TABLE recension
(
    id INT AUTO_INCREMENT NOT NULL
           PRIMARY KEY,
    produkt INT,
    recensent VARCHAR(20),
    betyg TINYINT,
    fritext TEXT,
    FOREIGN KEY (produkt)
           REFERENCES produkt(id)
);


CREATE TABLE bild
(
    id INT AUTO_INCREMENT NOT NULL
           PRIMARY KEY,
    produkt INT,
    lank VARCHAR(25),
    bildtext TINYTEXT,
    FOREIGN KEY (produkt)
           REFERENCES produkt(id)
);
```