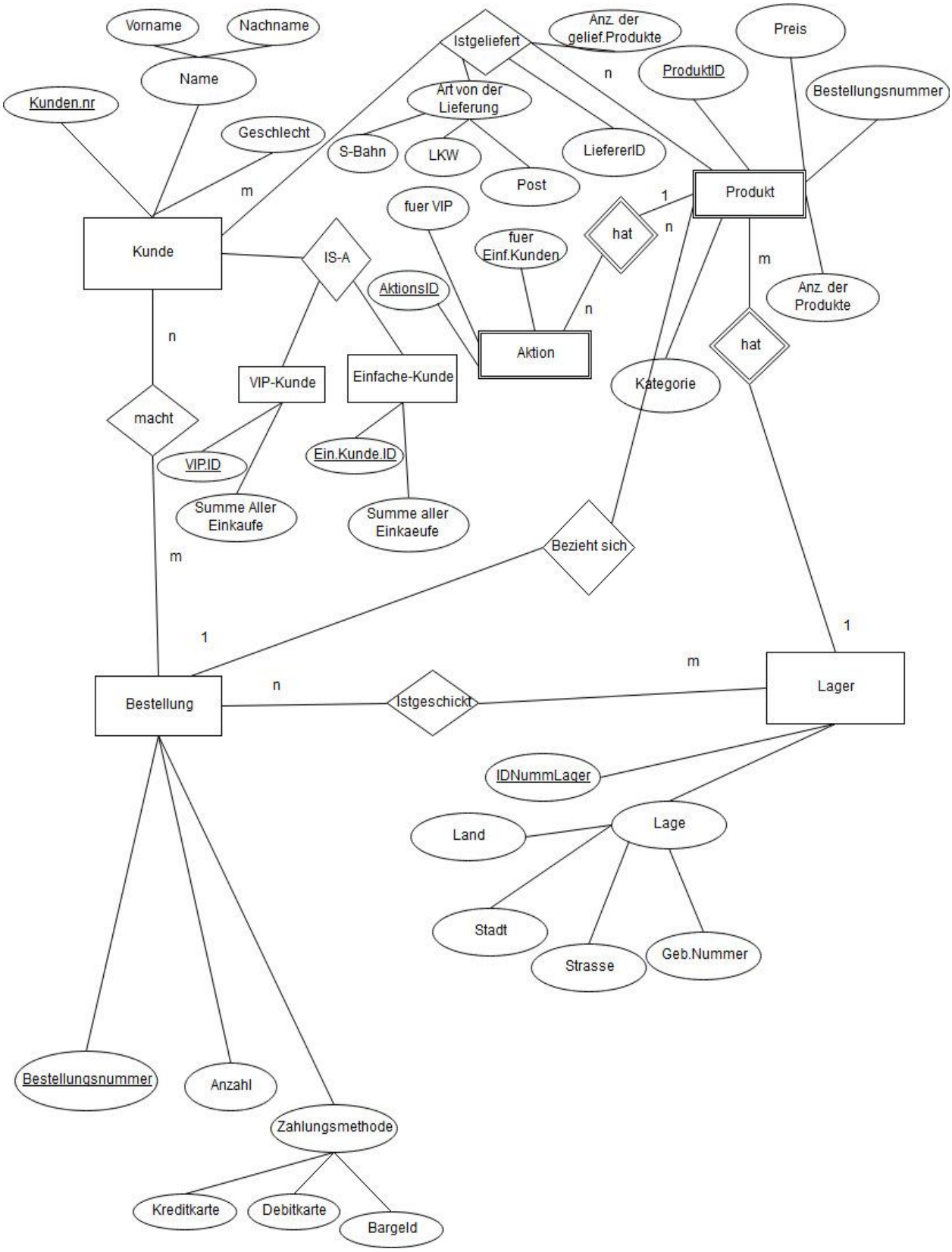
Online-Shop.

Meilenstein 1: Anforderungsanalyse & Konzeptioneller Entwurf.

Die Idee ist ein Online-Shop. Das ist ein klassischer Online-Shop, wo man die Sachen aus verschiedenen Kategorien kaufen kann. Der Kunde macht eine Bestellung, wo jede Bestellung seine eigene Nummer hat. Der Kunde ist auch durch einzigartige Kundennummer bestimmt. Und der Kunde kann eine Anzahl und Zahlungsmethode bestimmen. Jedes Produkt hat auch natürlich seine eigene ID, damit es einfach wäre ein bestimmtes Produkt zu finden. Dann ist eine Bestellung nach dem Lager geschickt. Es wird angezeigt, wie viel von den bestellten Waren übriggeblieben ist. Für jedes Produkt gibt es bestimmte Aktionen, die sich nach der Art von Kunden unterscheiden. Dann wird das Produkt zu einer Kunde geliefert. Das Produkt kann nach unterschiedlicher Art geliefert sein. Und wir sehen auch eine Anzahl der gelieferten Produkte.



Meilenstein 2: Logischer Entwurf.

Kunde (Kunde.nr, Vorname, Nachname, Geschlecht)

PK = {Kunde.nr}

VIP-Kunde (VIP.ID, Kunde.nr, Summe aller Einkäufe)
PK = {VIP.ID, Kunde.nr}
FK1 = {VIP-Kunde.Kunde.nr ◊ Kunde.Kunde.nr}

Einfache-Kunde (Ein.Kunde.ID, Kunde.nr, Summe aller Einkäufe)
PK = {Ein.Kunde.ID, Kunde.nr}
FK1 = {Einfache-Kunde.Kunde.nr ◊ Kunde.Kunde.nr}

Bestellung (Bestellungsnummer, Anzahl, Kreditkarte, DebitKarte, Bargeld)
PK = {Bestellungsnummer}

Macht (Kunde.nr, Bestellungsnummer)
PK = {Kunde.nr, Bestellungsnummer}
FK1 = {Macht.Kunde.nr ◊ Kunde.Kunde.nr}
FK2 = {Macht.Bestellungsnummer ◊ Bestellung.Bestellungsnummer}

Lager (IDNummLager, Land, Stadt, Strasse, Geb.Nummer)
PK = {IDNummLager}

Istgeschickt (Bestellungsnummer, IDNummLager)
PK = {Bestellungsnummer, IDNummLager}
FK1 = {Istgeschickt.Bestellungsnummer ◊ Bestellung.Bestellungsnummer}
FK2 = {Istgeschickt.IDNummLager ◊ Lager.IDNummLager}

Produkt (ProduktID, Anz_Produkte, Preis, Kategorie, IDNummLager, Bestellungsnummer)
PK = {ProduktID, IDNummLager}
FK1 = {Produkt.IDNummLager ◊ Lager.IDNummLager}
FK2 = {Produkt. Bestellungsnummer ◊ Bestellung.Bestellungsnummer}

Aktion (AktionsID, Akt_V, Akt_E, ProduktID,IDNummLager)
PK = {AktionsID, ProduktID,,IDNummLager}
FK1= {Aktion.ProduktID ◊ Produkt.ProduktID}
FK2= {Aktion.IDNummLager ◊ Produkt.IDNummLager}

Istgeliefert (ProduktID, Kunde.nr, IDNummLager, LiefererID, S-Bahn, LKW, Post, anz_gelief_prod)
PK = {ProduktID, Kunde.nr, IDNummLager}
FK1 = {Istgeliefert.IDNummLager ◊ Produkt.IDNummLager }
FK2 = {Istgeliefert.ProduktID ◊ Produkt.ProduktID}
FK3 = {Istgeliefert.Kunde.nr ◊ Kunde.Kunde.nr}

Meilenstein 4: Beginn Implementierung.

Java Implementierung.

Ich habe mit Java die Daten generiert. Für jede Relation generieren wir 1000 verschiedene Daten. Wir machen das am meisten mit if-Anweisungen und for-schleifen. Mein Ziel war einfach zu zeigen, dass meine SQL -Tabellen funktionieren, deshalb habe ich nicht so viele Variationen bei verschiedenen Daten programmiert. Am meistens habe ich 2 oder 3 Variationen für meine Attribute, die dann sich wiederholen.

PHP Implementierung.

Für jede Relation haben wir eine eigene Seite gemacht. Und noch dazu bei mir gibt es eine Hauptseite, wo dann man auf jede Relation übergehen kann. Auf jeder Seite kann man Insert und Delete Anweisungen ausführen. Man kann auch eine ganze Relationstabelle anschauen und auch Suche nach einem bestimmten Merkmal machen. Ich habe auch Kunde Relation mit dem Trigger gemacht. Es heißt, dass ein ID wird automatisch nach Eingabe der Parameter erstellt. Ich habe auch zwei Prozeduren erstellt.