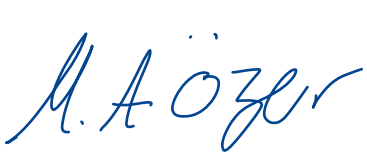
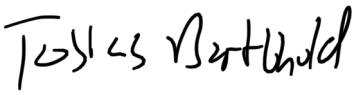
Eigenständigkeitserklärung

§20 (6) der Rahmenprüfungsordnung:
Referate und schriftliche Hausarbeiten müssen mit einer schriftlichen Versicherung der Studierenden in Form einer Eigenständigkeitserklärung versehen sein.

- „Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig angefertigt und mich keiner
  fremden Hilfe bedient sowie keine anderen als die angegebenen Quellen und Hilfsmittel benutzt
  habe. Alle Stellen, die wörtlich oder sinngemäß veröffentlichten oder nicht veröffentlichten
  Schriften und anderen Quellen entnommen sind, habe ich als solche kenntlich gemacht. Diese
  Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen."

Achtung!
Ohne unterschriebene Eigenständigkeitserklärung kann die schriftliche Hausarbeit nicht bewertet
werden!





Mehmet Ali Özer	Yusuf Bas	Tobias Barthold
7206358	7209349	7209370

Onlinemarktplatz

Mehmet Özer - 7206358

Yusuf Bas - 7209349

Tobias Barthold - 7209370

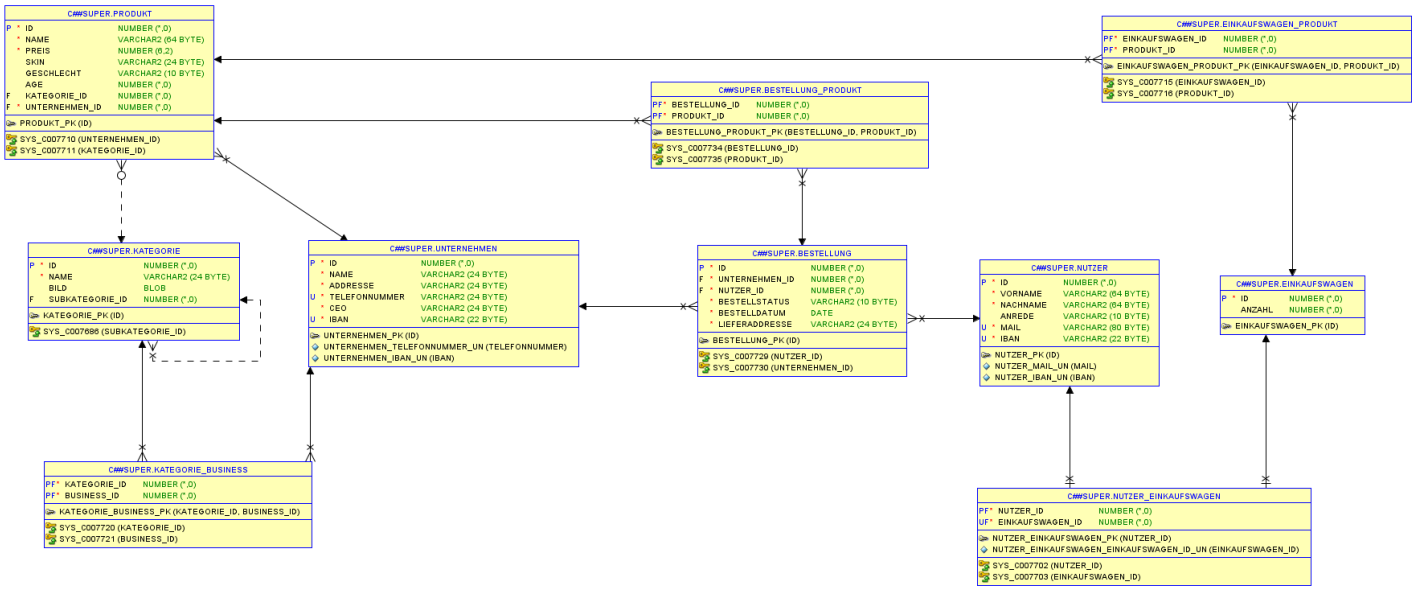
Unser Anwendungsszenario behandelt einen Online-Marktplatz auf dem man alle möglichen Tiere und Insekten kaufen kann. Nutzer können sich einen Account machen und sich entsprechend in den Webshop einloggen und dann sofort Tiere kaufen oder an Versteigerungen teilnehmen. Wir nutzen insgesamt 6 Integritätsbedingungen in diesem Kontext, zum einen NOT NULL, weil gewisse Felder wie z. B. Primary keys (in folgenden Dokumenten genannt: PKs) oder auch der Name bei einem Kunden oder bspw. Die IBAN immer angegeben werden müssen. Wir nutzen auch UNIQUE was dafür sorgt, dass Felder wie z.B. die IBAN oder die Telefonnummer immer eindeutige Werte haben. Bei dem Geschlecht der Tiere haben wir einen CHECK eingebaut der überprüft ob die Eingabe Männlich oder Weiblich ist, wenn nicht wird sie nicht angenommen. Beim löschen von Einträgen verwenden wir auch Foreign Key Constraints z. B. ON DELETE CASCADE oder ON DELETE SET NULL, sodass u.a. bei unserem Produkt, also dem Tier, dieses Tier gelöscht wird wenn das Unternehmen gelöscht wird was selbiges verkauft, oder aber die kategorie_id auf ihren DEFAULT Wert zurückgesetzt wird (in dem Fall NULL) wenn eine Kategorie gelöscht wird in der das Tier zu finden war. Für unser wiederholt ausführbares Installationsskript verwenden wir zum löschen der Tabellen auch CASCADE CONSTRAINTS, da sonst beim Löschen von Tabellen die auf Primary Keys von anderen Tabellen referenzieren um einen Foreign Key zu befüllen ein fehler geschmissen wird. Diese Integritätsbedingung sorgt also dafür dass die CASCADE CONSTRAINTS die wir angelegt haben ebenfalls gelöscht werden dürfen.

Onlinemarktplatz

Mehmet Özer - 7206358

Yusuf Bas - 7209349

Tobias Barthold - 7209370



Wir haben zum einen ein Unternehmen, ein Unternehmen kann natürlich mehrere Produkte also in dem Fall Tiere haben. In unserem Fall haben wir so entschieden, dass ein Produkt einzigartig ist, weil es ein Lebewesen ist und man ein Tier in der Regel ja nicht einfach nach Rasse und Fell auswählt, sodnern auch nach Namen und/oder Persönlichkeit. Entsprechend eine 1:N Beziehung von Unternehmen zu Produkt/Tier. Somit ist bei Produkt ein Fremdschlüssel der auf den Primary-Key von einem Business referenziert. Ein Unternehmen kann mehrere Kategorien von Tieren/Produkten bewirtschaften. Aber eine Kategorie wie z.B. Raubtiere kann natürlich auch von mehreren Unternehmen befüllt werden. Dementsprechend ist dies eine N:M (Many-to-Many) Beziehung und wir brauchen einen Junction-Table in dem 2 Fremdschlüssel sind die jeweils auf den PK der betroffenen Tabelleneinträge referenzieren. Bestellungen können natürlich viele verschiedene Produkte enthalten. Trotzdem gehört eine Bestellung immer genau zu einem Unternehmen, da wir beim Kauf die Produkte aus dem Warenkorb nehmen, über das Produkt dann herausfinden zu welchem Unternehmen es gehört und dies dann in eine Bestellung packen, die zu dem Nutzer und dem Unternehmen gehört. Entsprechend ist dies eine 1:N Beziehung von Unternehmen zu Bestellung und eine Bestellung hat einen Fremdschlüssel der auf die ID des Business referenziert. Produkte also Tiere müssen nicht unbedingt eine Kategorie haben, das macht es auch einfacher Kategorien zu bearbeiten/verwalten. Im Frontend würde dann einfach für kategory_id == NULL z.B. "other" angezeigt werden. Eine Kategorie kann natürlich mehrere Produkte haben, ansonsten wäre Kategoriesierung nutzlos. Somit exisitert ein Fremdschlüssel im Produkt, der auf die Kategorie verweist und auch NULLABLE ist. Ein Produkt kann in mehreren Einkaufswagen vorkommen aber es dürfen keine Duplikate vorkommen, weil man nicht 10x denselben Hund kaufen kann, unsere Produkte sind mehr oder weniger einzigartig. Daher wird hier abgesehen von den Fremdschlüsseln im Junction-Table der Many-to-Many Beziehung auch ein Composite (Primary) Key aus den beiden Referenzen gebaut. Ein Nutzer hat immer einen Einkaufswagen der Befüllt und beim Kauf geleert wird. Dies ist eine 1:1 Beziehung. In dem Fall haben wir dies auch mit einem Junction Table und einem Primary-Key aus einem Foreign-Key sowie einem Foreign-Key mit separatem UNIQUE Constraint realisiert. Nutzer können auch mehrere Bestellungen aufgeben, allerdings kann eine Bestellung nur von jeweils einem einzelnen Nutzer aufgegeben werden. Daher hat die Bestellung einen Fremdschlüssel, der auf den Nutzer verweist, der sie aufgab. Eine Kategorie kann eine rekursive Beziehung zu einer anderen Kategorie haben, da eine Kategorie auch beliebig viele Subkategorien haben kann. Z.B. können Hunde Subkategorien wie Dalmatiner oder Deutscher Schäferhund haben. Somit kann eine Kategorie einen Fremdschlüssel haben der auf eine andere Kategorie verweist.