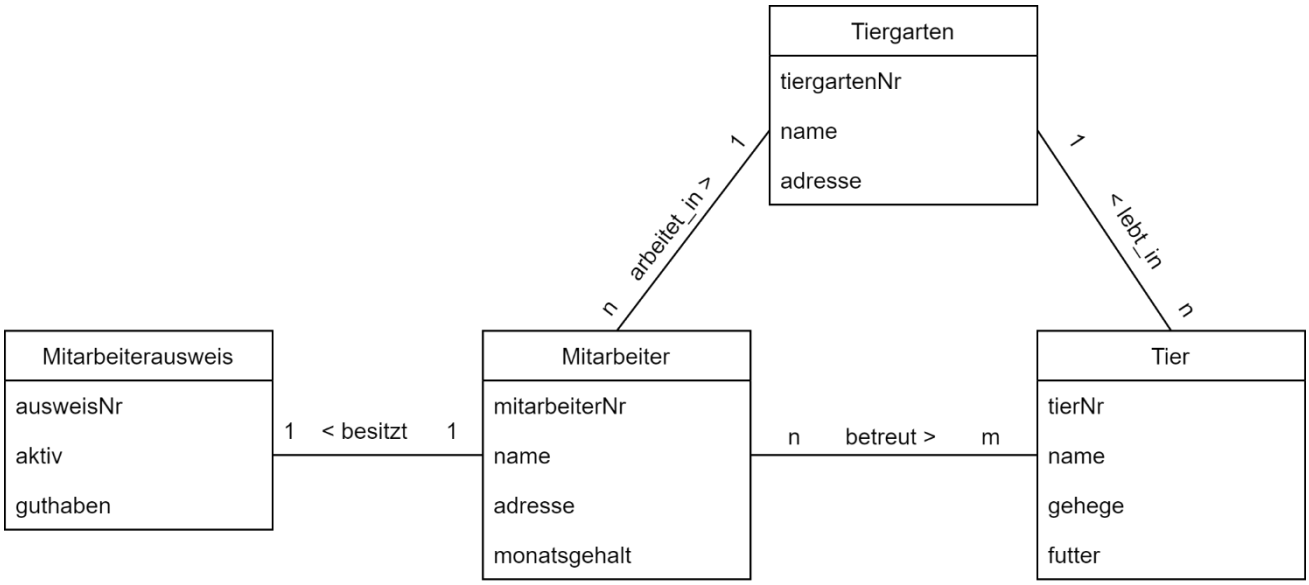
Inf 10 Vom Klassendiagramm zur Datenbank Datum:

Um Redundanz so gut wie möglich zu vermeiden, werden in Datenbank mehrere kleinere Tabellen verwendet. Modelliert man eine realistische Situation, so kann zunächst ein **Klassendiagramm** erstellt werden.

Erinnerung: Ein Klassendiagramm beschreibt die einzelnen Klassen und deren Beziehungen untereinander.

Beispiel:



Da die einzelnen „Klassen“ letztendlich Tabellen darstellen müssen keine Methoden, sondern nur Attribute angegeben werden – diese werden dann die entsprechenden Spalten einer Tabelle angeben.

Zusätzlich wird bei den Beziehungskanten angegeben, wie viele „Partner“ die Objekte haben können (Fachbegriff: **Kardinalität**). Man unterscheidet drei verschiedene Beziehungsarten:

1: **1 −Beziehung**: Die Zuordnung zweier Objekte ist immer eindeutig.

Im Beispiel: Ein Mitarbeiter besitzt genau einen Ausweis und jeder Ausweis wird auch nur von einem Mitarbeiter verwendet.

1: $\boldsymbol{n}$ **− Beziehung**: Die Zuordnung ist nur in eine Richtung eindeutig.

Im Beispiel: Jeder Mitarbeiter arbeitet in genau einem Tiergarten (Festlegung, die aus dem Klassendiagramm hervorgeht!), aber in einem Tiergarten arbeiten ggf. viele Mitarbeiter.

Analog: Ein Tier lebt in genau einem Tiergarten, aber in einem Tiergarten leben viele Tiere.

$\boldsymbol{n}$: $\boldsymbol{m}$ **− Beziehung**: Die Zuordnung ist in keine Richtung eindeutig.

Im Beispiel: Ein Mitarbeiter kann verschiedene Tiere betreuen, allerdings kann jedes Tier auch mehrere Betreuer haben.

Möchte man das Klassendiagramm in eine Datenbank überführen erstellt man ein **relationales Datenbankmodell** (es werden Relationen also Beziehungen zwischen den Klassen betrachtet!).

Dabei wird wie folgt vorgegangen (zur besseren Übersicht: Datentypen werden weggelassen!):

Schritt 1: Anlegen aller Klassen in einer Tabelle

Tiergarten[tiergartenNr, name, adresse]

Tier[tierNr, name, gehege, futter]

Mitarbeiter[mitarbeiterNr, name, adresse, monatsgehalt]

Mitarbeiterausweis[ausweisNr, aktiv, guthaben]

Schritt 2: $1:n$ und $1:1-$ Beziehungen „einbauen“

Um tabellenübergreifende Abfragen machen zu können und keine Informationen zu verlieren, muss eine „Verbindung“ der einzelnen Tabellen hergestellt werden. Dies geschieht über **Fremdschlüssel**, hierbei handelt es sich um den Primärschlüssel einer anderen Tabelle, der als Referenz eingetragen wird. Ein Fremdschlüssel wird im Schema „überstrichen“.

Bei $1:1-$ Beziehungen spielt es keine Rolle, in welcher der beiden Tabellen der Fremdschlüssel angelegt wird. Bei $1:n-$ Beziehungen dagegen wird der Fremdschlüssel auf der „mehrdeutigen“ Seite eingebaut:

Tier[tierNr, name, gehege, futter, $\overline{\text{tiergartenNr}}$]

Mitarbeiter[mitarbeiterNr, name, adresse, monatsgehalt, $\overline{\text{tiergartenNr}}$, $\overline{\text{ausweisNr}}$]

Schritt 3: $n:m-$ Beziehungen einbauen

Diese Art von Beziehung lässt sich nicht in den bereits vorhandenen darstellen, da sie auf beiden Seiten mehrdeutig ist. Es wird deswegen eine weitere Tabelle eingeführt, die die Primärschlüssel der beiden beteiligten Tabellen als Fremdschlüssel enthält.

Betreut[$\overline{\text{tierNr}}$, $\overline{\text{mitarbeiterNr}}$]

In phpmyadmin angelegt ergeben sich damit die folgenden Strukturen:

1 tiergartenNr	int(11)	
2 name	varchar(255)	
3 adresse	varchar(255)	

1 mitarbeiterNr	
2 name	
3 adresse	
4 monatsgehalt	
5 tiergartenNr	
6 ausweisNr	

1 tierNr	
2 name	
3 gehege	
4 futter	
5 tiergartenNr	

1 ausweisNr	
2 aktiv	
3 guthaben	

1 tierNr	
2 mitarbeiterNr	

Durch das Erstellen der Fremdschlüssel prüft das DBMS die **referentielle Integrität**. Das bedeutet, dass als Fremdschlüssel nur Einträge vorkommen können, die bereits in den ursprünglichen Tabellen existieren!