DATENBANKEN

Praktikum 2,5

DATENBANKEN

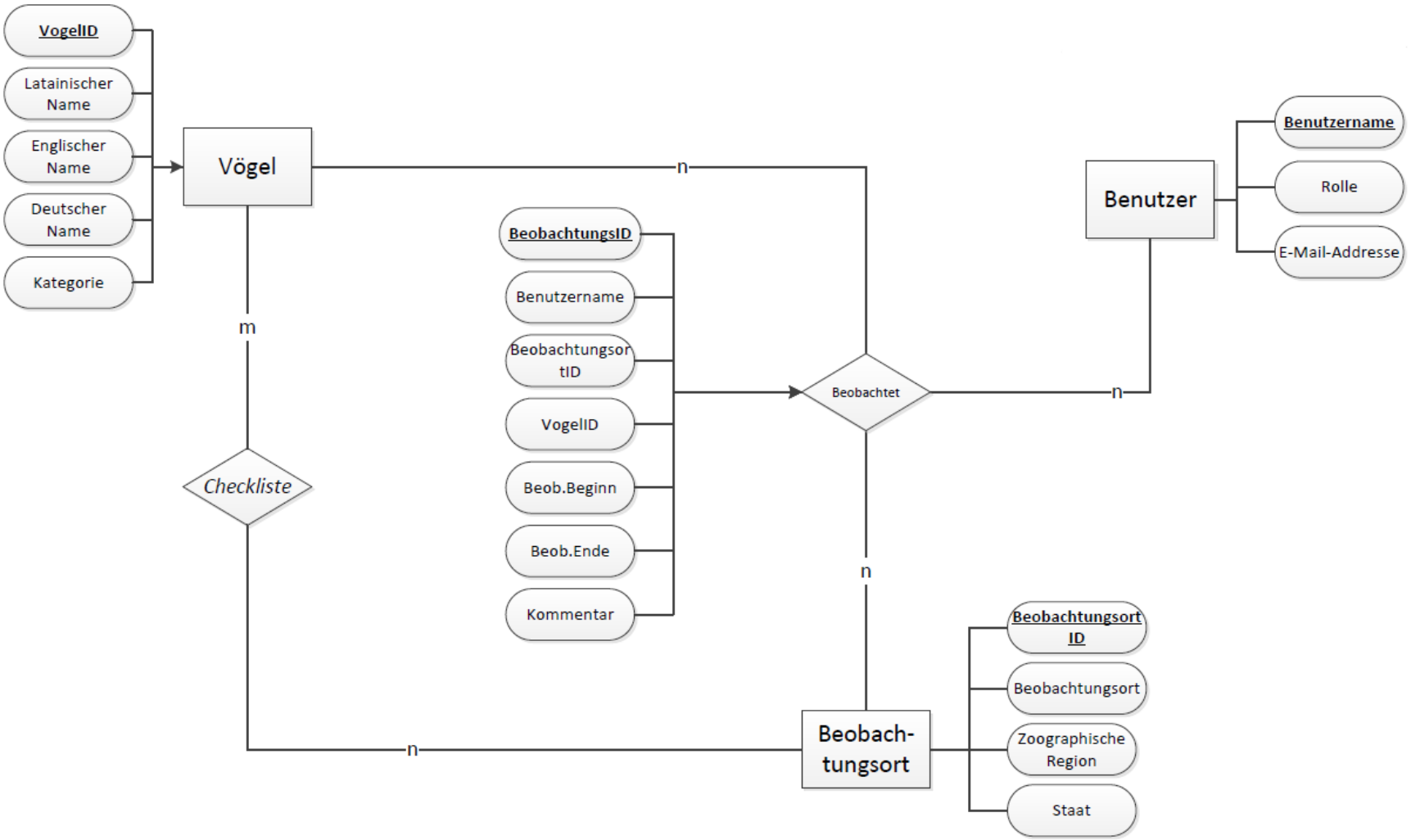
Praktikum 2,5

PRAKTIKUM 2,5

Dies ist die Dokumentation zum Praktikum zwei. Es umfasst eine schriftliche Ausarbeitung zu Aufgabe 3,4 und 6. Ebenfalls eine kurze Erklärung zu den Änderungen aus dem vorherigen Praktikum, sowie das erneuerte ER-Diagramm. Zu Aufgabe 5 liegt der SQL Code vor der eigens kommentiert ist.

Steffen Giersch &
Maria Lüdemann

INHALTSVERZEICHNIS


VogelID
Latainischer Name
Englischer Name
Deutscher Name
Kategorie
Vögel
n
Benutzer
Benutzername
Rolle
E-Mail-Addresse
BeobachtungsID
Benutzername
Beobachtungsort ID
VogelID
Beob.Beginn
Beob.Ende
Kommentar
Beobachtet
n
m
Checkliste
n
n
Beobach-tungsort
Beobachtungsort ID
Beobachtungsort
Zoographische Region
Staat

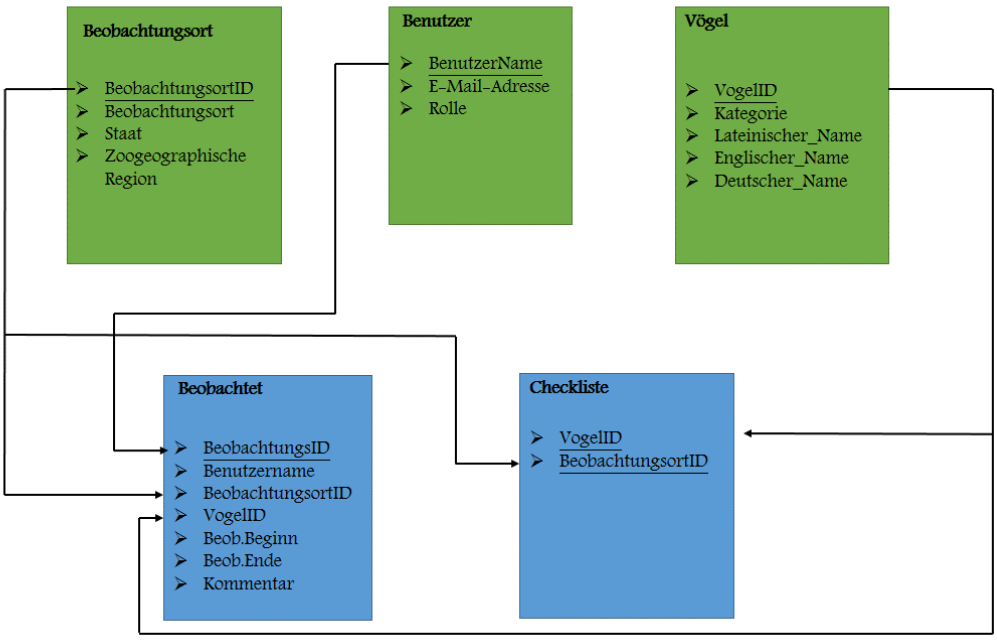
Korrekturen aus Aufgabe 1

Zu Nummer 1 und 3 aus der Korrektur von Aufgabe 1:
Das ER-Modell wurde angepasst um Unterarten handeln zu können.
Zudem wurde auch die Handhabung von Beobachtungen und Checklisten gleichgestellt.
Hieraus ergab sich ein wesentlich kleineres ER Modell, was als erste Seite eingefügt ist.

Zu Nummer 2 aus der Korrektur von Aufgabe 1:
Benutzer erhalten „Ränge“ von 1-3 die ihnen, mit aufsteigendem Rang, mehr Rechte zuordnen.
Dieses Rechtemanagement sollte in übergeordneter Software abgehandelt werden, da wir in Oracle als Administrator fungieren.

AUFGABE 3

Alle Entitäten werden zu Relationstypen, wobei die Attribute und Primärschlüssel übernommen werden. Die Checkliste wird zu einem Relationstyp, wobei eine m:n Beziehung vorliegt, sodass die VogelID und BeobachtungsortID beide zusammen zum Primärschlüssel werden. Ansonsten liegen keine Attribute vor. Beobachtet wird ebenso zu einem Relationstyp. Auch hier liegt eine m:n Beziehung vor, jedoch nutzen wir eine BeobachtungsID als Primärschlüssel um so auch doppelte Sichtungen an einem Ort vom selben Nutzer mit dem selben Vogel zu ermöglichen. So werden der Benutzername, die BeobachtungsortID und die VogelID zu Fremdschlüsseln für die Entitäteten Vögel, Benutzer und Beobachtungsort.



AUFGABE 4

In der PDF im Anhang sind das Relationsmodell in der 3. Normalform und die funktionalen Abhängigkeiten in formaler Schreibweise zu finden.

Die Abhängigkeiten haben keine Auswirkungen auf unsere Normalformen, da diese zwar rein formal eine große Auswirkung gehabt hätten (z.B. Lateinischer Name -> Englischer Name) aber technisch gesehen keinen Vorteil gebracht hätten. Dies ging aus einem Gespräch im Praktikum mit ihnen hervor.

AUFGABE 5

Eine SQL-Datei mit Tabellen Erstellung, Befüllung und anschließender Löschung liegt vor für die bessere Lesbarkeit nochmal bei.

```
/*#################################################
BEGINN TABELLEN ERSTELLEN
###############################################################*/
create table Voegel
  (VogelID integer not null PRIMARY KEY,
  Kategorie         varchar2(20) not null,
  Latainischer_Name varchar2(80) unique not null,
  Englischer_Name   varchar2(70),
  Deutscher_Name    varchar2(50),
/*Ist nicht not null, weil es einen sehr viel größeren Aufwand
erzeugt hätte, die Tabellen zu befüllen*/
  check(Kategorie in ('species', 'subspecies', 'group (monotypic)',
  'group (polytypic)'))
  );

create table Beobachtungsort
  (BeobachtungsortID integer not null PRIMARY KEY,
  Beobachtungsort       varchar2(500) not null,
  Staat                 varchar2(40),
  Zoographische_Region  varchar2(50)
  );

create table Benutzer
  (BenutzerName         varchar2(60) not null PRIMARY KEY,
  EMail_Addresse        varchar2(100) unique not null,
  Rolle                 integer not null
  );
```

```
create table Checkliste
  (VogelID              integer not null,
  BeobachtungsID        integer not null,
  PRIMARY KEY(VogelID,BeobachtungsID),
  FOREIGN KEY(VogelID)        references Voegel,
  FOREIGN KEY(BeobachtungsID) references Beobachtungsort
  );

create table Beobachtet
  (BeobachtungsID     integer not null PRIMARY KEY,
  BenutzerName        varchar2(60) not null,
  BeobachtungsortID   integer not null,
  VogelID             integer not null,
  BeobBeginn          date,
  BeobEnde            date,
  Kommentar           clob,
  FOREIGN KEY(BenutzerName)       references Benutzer,
  FOREIGN KEY(BeobachtungsortID)  references Beobachtungsort,
  FOREIGN KEY(VogelID)            references Voegel
  );

/*################################################
 BEGINN TABELLEN LÖSCHEN
##############################################################*/

drop table beobachtet cascade constraints PURGE;
drop table checkliste cascade constraints PURGE;
drop table Beobachtungsort cascade constraints PURGE;
drop table Voegel cascade constraints PURGE;
drop table benutzer cascade constraints PURGE;
```

AUFGABE 6

Die Daten in den Tabellen Birds, Birds_de und Birds_IOC haben teils ergänzende, teils widersprüchliche und teils komplett fehlende Daten.
Beispielsweise gab es viele deutsche Namen in Birds_de, die in Birds_IOC nicht vorhanden waren, andererseits aber auch widersprüchliche Namen.
Weil wir die deutschen Namen in die Tabelle Voegel überführen mussten, war es notwendig eine Tabelle als „aussagekräftiger“ als die andere zu bewerten
Wir haben uns dafür entschieden, dass die Tabelle Birds_IOC als aussagekräftiger zu behandeln ist, da diese wesentlich mehr Einträge als Birds_de hat.
Deutsche Namen, die nicht in Birds_IOC enthalten waren, wurden jedoch auch in Voegel ergänzt und wie im Lastenheft gefordert (A01), wurden Arten, zu denen keinerlei deutscher Name zuzuordnen war,

komplett gelöscht. Zudem wurden die Unterarten, zu deren Arten kein deutscher Name vorhanden war mit gelöscht.

Die Beobachtungsorte waren aus Birds.b_range zu entnehmen. Allerdings sind diese so schlecht dokumentiert, dass es unmöglich ist die Orte einem speziellen Staat oder einer Zoografischen Region zuzuordnen. Daher wurden diese Orte, Dopplungen ausgenommen, komplett in Beobachtungsort.beobachtungsort kopiert.

Die Checkliste wurde gefüllt, indem ein Matching von den Beobachtungsorten über Birds.b_range und die Voegel geführt wurde.

Der folgene SQL Code liegt ebenfalls in der beigefügten SQL Datei:

```
/*################################################
BEGINN TABELLEN BEFÜLLEN
##############################################################*/
/*Fügt alle Einträge aus MERLIN.birds nach VOEGEL*/
INSERT INTO VOEGEL
  (VogelID, Latainischer_Name, Kategorie, englischer_name)
SELECT
  merlin.birds.b_id,
  merlin.birds.b_scientific_name,
  merlin.birds.b_category,
  merlin.birds.b_english_name
FROM MERLIN.birds;

/*Schiebe die deutschen Namen aus BIRDS_DE in die betreffenden Tupel
aus VOEGEL*/
UPDATE VOEGEL
  SET VOEGEL.deutscher_name = ( select MERLIN.BIRDS_DE.de_deutsch
                                from MERLIN.BIRDS_DE
                                where latainischer_name =
                                MERLIN.BIRDS_DE.de_latein);

/*Schiebe die deutschen Namen aus BIRDS_DE in die betreffenden Tupel
aus VOEGEL, wobei nicht kopiert wird, wenn der zu kopierende Wert
(null) ist*/
UPDATE VOEGEL
  SET VOEGEL.deutscher_name = ( select
                                MERLIN.BIRDS_IOC.IOC_GERMAN_NAME
                                from MERLIN.BIRDS_IOC
                                where latainischer_name =
                            MERLIN.BIRDS_IOC.IOC_SCIENTIFIC_NAME)
  WHERE (select MERLIN.BIRDS_IOC.IOC_GERMAN_NAME
  FROM MERLIN.BIRDS_IOC where latainischer_name =
MERLIN.BIRDS_IOC.IOC_SCIENTIFIC_NAME) is not null;

/*Schiebe die englischen Namen aus BIRDS_DE in die betreffenden
Tupel aus VOEGEL, wobei bestehende Einträge nicht überschrieben
werden*/
```

```
UPDATE VOEGEL
  SET VOEGEL.englischer_name = (select MERLIN.BIRDS_DE.de_englisch
                                from MERLIN.BIRDS_DE
                                where latainischer_name =
MERLIN.BIRDS_DE.de_latein)
  WHERE englischer_name is null;

/*Schiebe die englischen Namen aus BIRDS_IOC in die betreffenden
Tupel aus VOEGEL, wobei bestehende Einträge nicht überschrieben
werden*/
UPDATE VOEGEL
  SET VOEGEL.englischer_name = (select
MERLIN.BIRDS_IOC.ioc_english_name
                                from MERLIN.BIRDS_IOC
                                where latainischer_name =
                            MERLIN.BIRDS_IOC.ioc_scientific_name)
WHERE englischer_name is null;

 /*Species und dazugehörige Subspecies ohne deutschen Namen werden
gelöscht*/

DELETE from voegel where
    (trim(substr(latainischer_name, 1, instr(latainischer_name, '
',1, 2))) in (select trim(latainischer_name)

FROM voegel

WHERE deutscher_name is null

and kategorie = 'species')
    OR
    deutscher_name is null and kategorie = 'species');

/*Fülle die ranges aus BIRDS in BEOBACHTUNGSORT, wobei doppelte
nicht mit kopiert werden und überflüssige b_id's für das select
gelöscht werden*/
INSERT INTO BEOBACHTUNGSORT
  (beobachtungsortID, beobachtungsort)
SELECT
  min(merlin.birds.b_id),
  merlin.birds.b_range
FROM merlin.birds where merlin.birds.b_range is not null group by
merlin.birds.b_range;

/*Befülle die Checkliste*/
INSERT INTO CHECKLISTE
  (vogelid, beobachtungsid)
SELECT
  vogelid,
  beobachtungsortid
FROM voegel v, merlin.birds mb, beobachtungsort b where
v.latainischer_name = mb.b_scientific_name
                                                    and
b.beobachtungsort = mb.b_range ;
```