Veranstaltung: Datenbanken (Tlc) S15 | Übungsblatt: 2 | Übungsgruppe: 2

Dokumentation zu dem Übungsblatt 2

Bearbeitet durch: Mesut Koc, Anton Kirakozov

Inhaltsverzeichnis

[Einführung 1](#_Toc419830244)

[Erforderliche software 1](#_Toc419830245)

[Aufgabe 3 2](#_Toc419830246)

[Buchungsort : 2](#_Toc419830247)

[Vogelart : 2](#_Toc419830248)

[Benutzer : 2](#_Toc419830249)

[Rolle : 2](#_Toc419830250)

[Hat\_Rolle : 2](#_Toc419830251)

[Checkliste : 2](#_Toc419830252)

[Beobachtung : 2](#_Toc419830253)

[Aufgabe 4 2](#_Toc419830254)

[ Skript zum Erstellen von Merlin: 2](#_Toc419830255)

[ Skript zum Löschen von Tabellen: 5](#_Toc419830256)

[Aufgabe 5 5](#_Toc419830257)

[ Umgang mit Nullwerten: 5](#_Toc419830258)

[ Unser Skript, um Datensätze zu Importieren: 5](#_Toc419830259)

[ Unser Skript, um Die **Checkliste** zu Importieren: 6](#_Toc419830260)

[Annahme: 6](#_Toc419830261)

[ Überprüfung, ob die korrekte ID für die Region verwendet wurde: 7](#_Toc419830262)

[Aufgabe 6 7](#_Toc419830263)

# Einführung

In dieser Dokumentation beschreiben wir unsere Lösungen zu dem Übungsblatt 2. Dabei geht es darum, den vorherigen logischen Entwurf ins Relationsmodell zu überführen. Zusätzlich implementieren wir ein SQL-Skript, welches unser gesamtes Merlin-Schema vollständig anlegt.

# Erforderliche software

Für die Umsetzung unserer Aufgaben *(explizit: Aufgabe 4,5 & 6)* benutzen wir die Software ,,SQL Developer“. Die Software befindet sich ebenfalls im Rechner der AI-Labore.

# Aufgabe 3

|  |
| --- |
| Buchungsort : { [BeobachtungsID : *int*, Level 1 : *string*, Level 2 : *string*, Level 3 : *string*] }. |
| Vogelart :{ [VID : int, NameEgl : *string*, NameDeu : *string*, NameLat : *string*, Art : *string*, Unterart : *string*] } |
| Benutzer :{ [BenutzerID : int, Vorname : *string*, Nachname : *string*, Registrierungsdatum : *date*, E-Mail : *string*, Password(MD5) : *string*] } |
| Rolle :{ [Rid : *int*, Rolle : *string*] } |
| Hat\_Rolle :{ [RID : *int*, BenutzerID : int] } |
| Checkliste : { [BeobachtungsID VID : *int*] } |
| Beobachtung : { [BeobachtungsID VID : *int*, *BenutzerID* : int, Zeitanfang : *date*, Zeitende : date, Kommentar : *string*] } |

# Aufgabe 4

## Skript zum Erstellen von Merlin:

Beim Erstellen der Tabellen muss man drauf achten, dass man zuerst (!) die Tabellen erstellt und erst dann die Relations-Tabellen.

**CREATE TABLE** *Beobachtungsort* (

*BeobachtungsortID* NUMBER GENERATED ALWAYS AS IDENTITY NOT NULL,

*Region* VARCHAR(250) NOT NULL,

*Land* VARCHAR(250),

*Stadt* VARCHAR(250),

*CONSTRAINT* Beobachtungsort\_pk

*PRIMARY* KEY(BeobachtungsortID));

**CREATE TABLE** *Vogelart* (

*VID* NUMBER GENERATED ALWAYS AS IDENTITY NOT NULL,

*NAME\_ENG* VARCHAR(250),

*NAME\_DEU* VARCHAR(250),

*NAME\_LAT* VARCHAR(250),

*Art* VARCHAR(250),

*Unterart* VARCHAR(250),

CONSTRAINT *Vogelart\_pk*

PRIMARY KEY(VID));

**CREATE TABLE** *Benutzer* (

*BenutzerID* NUMBER GENERATED ALWAYS AS IDENTITY NOT NULL,

*Vorname* VARCHAR(45) NOT NULL,

*Nachname* VARCHAR(45) NOT NULL,

*RegisterDate* DATE NOT NULL,

*Email* VARCHAR(45) NOT NULL,

*Password*\_ VARCHAR(45) NOT NULL,

CONSTRAINT *Benutzer\_pk*

PRIMARY KEY(BenutzerID));

**CREATE TABLE** *Rolle* (

*RID* NUMBER GENERATED ALWAYS AS IDENTITY NOT NULL,

*Rolle* VARCHAR(45) NOT NULL,

CONSTRAINT *Rolle\_pk* PRIMARY KEY(RID));

**CREATE TABLE** *Hat\_Rolle*(

*BenutzerID* int not null,

*RID* int not null,

PRIMARY KEY(BenutzerID, RID),

FOREIGN KEY(BenutzerID) REFERENCES Benutzer(BenutzerID),

FOREIGN KEY(RID) REFERENCES Rolle(RID));

**CREATE TABLE** *Checkliste* (

*BeobachtungsortID* INT NOT NULL,

*VID* INT NOT NULL,

PRIMARY KEY(BeobachtungsortID,VID),

FOREIGN KEY(BeobachtungsortID) REFERENCES Beobachtungsort(BeobachtungsortID),

FOREIGN KEY(VID) REFERENCES Vogelart(VID));

**CREATE TABLE** *Beobachtung* (

*BeobachtungsortID* INT NOT NUll,

*VID* INT NOT NULL,

*BenutzerID* INT NOT NULL,

*Zeitanfang* TIMESTAMP NOT NULL,

*Zeitende* TIMESTAMP NOT NULL,

*Kommentar* VARCHAR(250),

PRIMARY KEY(BeobachtungsortID,VID),

FOREIGN KEY(BeobachtungsortID) REFERENCES Beobachtungsort(BeobachtungsortID),

FOREIGN KEY(VID) REFERENCES Vogelart(VID),

FOREIGN KEY(BenutzerID) REFERENCES Benutzer(BenutzerID));

**Anmerkung:** Wir haben für die Primärschlüssel die Autoinkrement-Funktion von *ORACLE* verwendet *(NUMBER GENERATED ALWAYS AS IDENTITY)*.

**DROP TABLE** *Checkliste*;

**DROP TABLE** *Beobachtung*;

**DROP TABLE** *Beobachtungsort*;

**DROP TABLE** *Vogelart*;

**DROP TABLE** Hat\_*Rolle*;

DROP TABLE Rolle;

**DROP TABLE** *Benutzer*;

## Skript zum Löschen von Tabellen:

Beim Löschen muss drauf geachtet werden, dass man vorher die Relationstabellen löscht, bevor man die einheitlichen Tabellen löscht.

# Aufgabe 5

## Umgang mit Nullwerten:

Bei dieser Aufgabe haben wir alle Datensätze aus der Merlin-Beispieldatenbank importiert. Dabei gab es natürlich auch Komplikationen, wie erwartet. Wir hatten einige Spalten als „not null“ definiert, sodass diese Tupel befüllt werden müssen. Das Problem war jedoch, dass aus der Beispieldatenbank einige Datensätze allerdings NULL-Werte hatten. Wir haben uns dabei gedacht, dass z.B. zu bestimmten englischen Namen keine deutschen Namen vorhanden sind, deswegen haben wir NULL-Werte mit importiert.

## Unser Skript, um Datensätze zu Importieren:

Wir haben hier alle Datensätze *(ca. 31.000)* importiert. Da unsere Tabelle etwas anders aufgebaut ist als die vorgegebene MERLIN-Datenbank, mussten wir einige Datensätze anpassen. Die deutschen Namen waren zum Beispiel separat gespeichert. Also mussten wir erst alle Datensätze importieren und dann dementsprechend zu den gleichen englischen Namen *(aus zwei Tabellen)* die deutschen Namen hinzufügen. Das ging mit dem UPDATE-Befehl. Des Weiteren gab es bei uns zwei Lösungsansätze für die Art und Unterart.

Wir konnten nicht eindeutig herausfiltern, ob die Kategorie aus der MERLIN-Tabelle BIRDS Art und Unterart die beiden Attribute bilden, oder ob FAMILY und ORDER aus der gleichnamigen Tabelle die beiden Art und Unterart Attribute beschreiben. Hinzukommend könnte man zusätzlich noch die deutschen Namen mit den lateinischen Namen vergleichen, wir haben uns jedoch für die englische Variante entschieden.

1. Daten einfügen in unsere Vogelart, erstmal alle Daten die eine Art bilden:

**INSERT INTO** Vogelart (Name\_Eng, NAME\_LAT, Art)

**SELECT** B\_ENGLISH\_NAME, B\_SCIENTIFIC\_NAME, B\_Category

**FROM** Merlin.Birds MB

**WHERE** MB.B\_CATEGORY='species';

Kontrolle, ob die Arten korrekt in unsere Datenbank eingefügt worden sind:

**SELECT** \* **FROM** vogelart;

1. Daten einfügen in unsere Vogelart, die eine Unterart bilden:

**INSERT INTO** Vogelart (NAme\_Eng, NAME\_LAT, Unterart)

**SELECT** B\_ENGLISH\_NAME, B\_SCIENTIFIC\_NAME, B\_Category

**FROM** Merlin.Birds MB

**WHERE** MB.B\_CATEGORY='subspecies';

1. Nachdem die Englischen und die lateinischen Namen eingefügt worden sind, aktualisieren wir die Tabelle Vogelart und fügen die deutschen Namen der Vögel hinzu. Dafür vergleichen wir die englischen Namen der Tabelle BIRDS\_DE und unserer Vogeltabelle:

**UPDATE** vogelart

**SET** NAME\_DEU = (**SELECT** DE\_DEUTSCH

**FROM** MERLIN.BIRDS\_DE mb

**WHERE** mb.DE\_ENGLISCH = Vogelart.NAME\_ENG);

1. Das gleiche Update führen wir mit der Tabelle BIRDS\_IOC durch, da es dort auch zusätzlich deutsche Namen gibt:

**UPDATE** vogelart

**SET** NAME\_DEU = (**SELECT** IOC\_German\_name

**FROM** MERLIN.BIRDS\_IOC mbc

**WHERE** mbc.IOC\_ENGLISH\_Name = Vogelart.NAME\_ENG);

## Unser Skript, um Die **Checkliste** zu Importieren:

## Annahme:

* Unserer Interpretation nach gibt es alle Vögel mit deutschen Namen auch in Deutschland bzw. wir definieren das. Demnach setzt sich die Checkliste aus dem

Vögel mit deutschen Namen und Region Westpaläarktis und dem Land Deutschland.

Ein Testdatensatz für unsere Tabelle Ort. Wir fügen dort die Region und das Land Deutschland hinzu:

**INSERET INTO** Beobachtungsort(Region, Land)

values('Westpaläarktis (WP)','Deutschland');

Ein weiterer Datensatz für die Region , jedoch diesmal mit Land und Stadt als NULL VALUE:

**INSERT INTO** Beobachtungsort(Region)

values('Westpaläarktis (WP)');

Wir fügen alle uns zur Verfügung stehenden Regionen hinzu aus der Tabelle Birds. Wir nehmen an, dass die B\_Range unsere Region ist:

**INSERT INTO** BEOBACHTUNGSORT(Region)

**SELECT DISTINCT** merlin.birds.b\_range

**FROM** merlin.birds where merlin.birds.b\_range is not null;

Nun befüllen wir unsere Checkliste nach der angegebenen Bedingung:

**INSERT INTO** Checkliste (BeobachtungsortID, VID)

**SELECT** BeobachtungsortId, VID

**FROM** Beobachtungsort B, Vogelart V

**WHERE** v.Name\_DEU **IS NOT NULL**

**AND** b.Land ='Deutschland';

## Überprüfung, ob die korrekte ID für die Region verwendet wurde:

**SELECT** \* **FROM** Beobachtungsort

**WHERE** BEOBACHTUNGSORTID = '22701';

# Aufgabe 6

Wir tragen in die Tabelle Beobachtung Datensätze ein, diese werden mit der Checkliste verglichen. Vögel die nicht in Deutschland gesichtet werden können dürfen nicht eingetragen werden:

**INSERT INTO** Beobachtung (BeobachtungsortID, VID, BenutzerID, Zeitanfang, Zeitende)

**SELECT** 26181,6106,1,'20.05.2015 17:00', '20.05.2015 18:00' FROM DUAL

**WHERE EXISTS** (**SELECT** Beobachtungsortid **FROM** Beobachtungsort **WHERE** Beobachtungsort.Beobachtungsortid = 26181)

**AND EXISTS** (**SELECT** VID **FROM** Checkliste c **WHERE** c.VID = 6106);

Hier können wir nochmal sichergehen, dass unsere Funktion klappt. Denn dieser Vogel ist in der Checkliste überhaupt nicht enthalten:

**INSERT INTO** Beobachtung(BeobachtungsortID, VID, BenutzerID, Zeitanfang, Zeitende)

**SELECT** 26181,4004,1,'20.05.2015 17:00', '20.05.2015 18:00' FROM DUAL

**WHERE EXISTS** (**SELECT** Beobachtungsortid **FROM** Beobachtungsort **WHERE** Beobachtungsort.Beobachtungsortid = 26181)

**AND EXISTS** (**SELECT** VID **FROM** Checkliste c **WHERE** c.VID = 4004);