# DBS – Projekt Protokoll

|  |  |  |
| --- | --- | --- |
| **Titel:** | Fußballverein | * FBV |
|  |  |  |
| **Leitung:** | **Carab Lucian-Valentin** | |
| **Projektteilnehmer:** | Pchelnikava Aliaksandra |  |
|  | Seidel Hans |  |
|  | Höller Nicholas |  |
|  |  |  |

## Dokumentbeschreibung:

Dieses Dokument beinhaltet sowohl die Planung als auch die Dokumentation der einzelnen Arbeitsschritte. Außerdem wurde es für das Projektteam als *Mini-Kanban* verwendet.

Das Projekt ist grundlegend in drei Teile aufgegliedert (Planung, Datenbank, Interface), wobei der Datenbanken-Bereich den größten Arbeitsaufwand beinhaltet.

Es wurden mehrere Teammeetings abgehalten und die notwendigen Planungsdokumente erstellt. Da der Fokus allerdings auf Datenbanken-Entwicklung gesetzt war und nicht Projektmanagement wurde auf einen großen Teil der Feinplanung verzichtet.

Inhalt

[DBS – Projekt Protokoll 1](#_Toc58240333)

[Dokumentbeschreibung: 1](#_Toc58240334)

[Planung 3](#_Toc58240335)

[ER-Modell 4](#_Toc58240336)

[Relationen-Modell 5](#_Toc58240337)

[Select-Statements 9](#_Toc58240338)

[DB-Users 11](#_Toc58240339)

[VIEWS 12](#_Toc58240340)

[Procedures & Functions 14](#_Toc58240341)

[Package 20](#_Toc58240342)

[Triggers 21](#_Toc58240343)

# Planung

Grundsätzlich haben wir für dieses Projekt in einzelnen Etappen die Planung gestaltet. D.h. begonnen wurde mit der Planung der Datenbank -> dann folgte die Umsetzung der Datenbank. In einem weiteren Schritt wurde die Planung der SELECTS, TRIGGERS, PACKAGES usw. vorgenommen und in Folge wieder die Umsetzung durchgeführt.

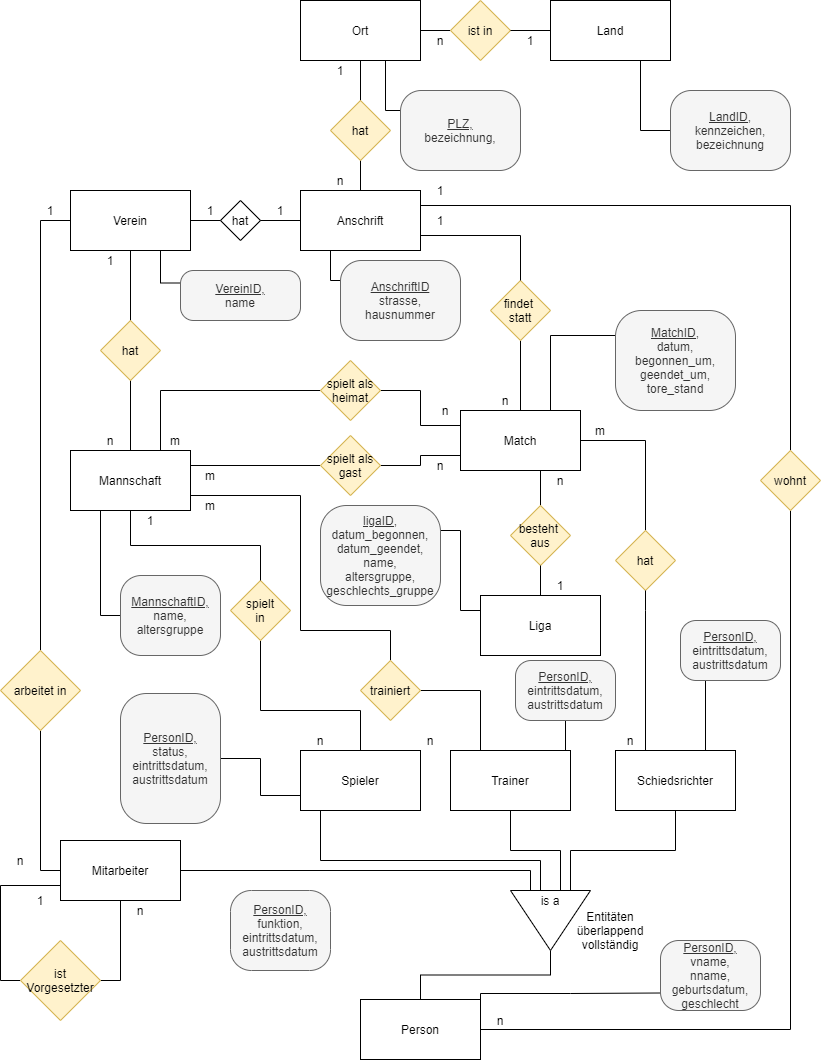
Für die Planung wurde ein ER-Modell und ein Relationen Modell erstellt. Das ER-Modell wurde gemeinsam entworfen. Nach dem Entwurf wurden die Create- und Insert-Statements im Team aufgeteilt und jeder hat einen Teil der Arbeit übernommen. Da alle nach dem entworfenen ER-Modell gearbeitet haben, war das Zusammenfügen der einzelnen Teilbereiche mit ziemlich wenig Arbeitsaufwand verbunden.

Wir haben für dieses Projekt ein GitHub – Repository erstellt, um distribuiert arbeiten zu können und die Software in verschiedene Version gesichert zu haben.

Um die Testdaten zu generieren haben wir [www.generatedata.com](http://www.generatedata.com) verwendet.

## ER-Modell

Das ER-Modell wurde einmalig verändert und die Veränderungen wurden in der Datenbank aktualisiert. Dies ist das aktuelle ER-Modell.



## Relationen-Modell

Das Relationen-Modell wurde entgegengesetzt der eigentlichen Vorgehensweiße nach dem Erstellen der Datenbank modelliert und nicht aus dem ER-Modell heraus modelliert.

|  |  |  |
| --- | --- | --- |
| **Tabellen-Name:** | Land |  |
|  |  | ***Datentyp*** |
| **Primär-Schlüssel:** | **LandID** | Number |
| **Fremd-Schlüssel:** | ***-*** | - |
| **Attribute:** | Kennzeichen | Varchar (20) |
|  | Bezeichnung | Varchar (255) |
|  |  |  |
| **Tabellen-Name:** | ORT |  |
|  |  | ***Datentyp*** |
| **Primär-Schlüssel:** | **PLZ** | Number |
| **Fremd-Schlüssel:** | ***LandID*** | Number |
| **Attribute:** | Bezeichnung | Varchar(255) |
|  |  |  |
| **Tabellen-Name:** | Anschrift |  |
|  |  | ***Datentyp*** |
| **Primär-Schlüssel:** | **AnschriftID** | Number |
| **Fremd-Schlüssel:** | ***PLZ*** | Number |
| **Attribute:** | Strasse | Varchar(255) |
|  | Hausnummer | Varchar(20) |
|  |  |  |
| **Tabellen-Name:** | Verein |  |
|  |  | ***Datentyp*** |
| **Primär-Schlüssel:** | **VereinID** | Number |
| **Fremd-Schlüssel:** | ***AnschriftID*** | Number |
| **Attribute:** | Name | Varchar(255) |
|  |  |  |
| **Tabellen-Name:** | Liga |  |
|  |  | ***Datentyp*** |
| **Primär-Schlüssel:** | **LigaID** | Number |
| **Fremd-Schlüssel:** | ***-*** | - |
| **Attribute:** | Datum\_Begonnen | Date |
|  | Datum\_Geendet | Date |
|  | Name | Varchar(255) |
|  | Altersgruppe | Number |
|  | Geschlechts\_Gruppe | Char (1) |

|  |  |  |
| --- | --- | --- |
| **Tabellen-Name:** | Mannschaft\_Match |  |
|  |  | ***Datentyp*** |
| **Primär-Schlüssel:** | **HeimmannschaftID**  **GastmannschaftID**  **MatchID** | Number  Number  Number |
| **Fremd-Schlüssel:** | ***-*** | - |
| **Attribute:** | - | - |
|  |  |  |
| **Tabellen-Name:** | Mannschaft\_Trainer |  |
|  |  | ***Datentyp*** |
| **Primär-Schlüssel:** | **MannschaftID**  **TrainerID** | Number  Number |
| **Fremd-Schlüssel:** | ***-*** | - |
| **Attribute:** | - | - |
|  |  |  |
| **Tabellen-Name:** | Match\_Schiedsrichter |  |
|  |  | ***Datentyp*** |
| **Primär-Schlüssel:** | **MatchID**  **SchiedsrichterID** | Number  Number |
| **Fremd-Schlüssel:** | ***-*** | - |
| **Attribute:** | - | - |
|  |  |  |
| **Tabellen-Name:** | Match |  |
|  |  | ***Datentyp*** |
| **Primär-Schlüssel:** | **MatchID** | Number |
| **Fremd-Schlüssel:** | ***AnschriftID***  ***LigaID*** | Number  Number |
| **Attribute:** | Datum | Date |
|  | Begonnen\_um | Timestamp |
|  | Geendet\_um | Timestamp |
|  | Tore\_Stand | Varchar (7) |
|  |  |  |
| **Tabellen-Name:** | Mannschaft |  |
|  |  | ***Datentyp*** |
| **Primär-Schlüssel:** | **MannschaftID** | Number |
| **Fremd-Schlüssel:** | ***VereinID*** | Number |
| **Attribute:** | Name | Varchar (255) |
|  | Altersgruppe | Number |

|  |  |  |
| --- | --- | --- |
| **Tabellen-Name:** | Spieler |  |
|  |  | ***Datentyp*** |
| **Primär-Schlüssel:** | **PersonID** | Number |
| **Fremd-Schlüssel:** | ***MannschaftID*** | Number |
| **Attribute:** | Status | Char(5) |
|  | Eintrittsdatum | Date |
|  | Austrittsdatum | Date |
|  |  |  |
| **Tabellen-Name:** | Trainer |  |
|  |  | ***Datentyp*** |
| **Primär-Schlüssel:** | **PersonID** | Number |
| **Fremd-Schlüssel:** | ***-*** | - |
| **Attribute:** | Eintrittsdatum | Date |
|  | Austrittsdatum | Date |
|  |  |  |
| **Tabellen-Name:** | Schiedsrichter |  |
|  |  | ***Datentyp*** |
| **Primär-Schlüssel:** | **PersonID** | Number |
| **Fremd-Schlüssel:** | ***-*** | - |
| **Attribute:** | Eintrittsdatum | Date |
|  | Austrittsdatum | Date |
|  |  |  |
| **Tabellen-Name:** | Mitarbeiter\_Vorgesetzter |  |
|  |  | ***Datentyp*** |
| **Primär-Schlüssel:** | **PersonID** | Number |
| **Fremd-Schlüssel:** | ***VorgesetzterID*** | Number |
| **Attribute:** | - | - |
|  |  |  |
| **Tabellen-Name:** | Mitarbeiter |  |
|  |  | ***Datentyp*** |
| **Primär-Schlüssel:** | **PersonID** | Number |
| **Fremd-Schlüssel:** | ***VereinID*** | Number |
| **Attribute:** | Funktion | Varchar (50) |
|  | Eintrittsdatum | Date |
|  | Austrittsdatum | Date |

|  |  |  |
| --- | --- | --- |
| **Tabellen-Name:** | Person |  |
|  |  | ***Datentyp*** |
| **Primär-Schlüssel:** | **PersonID** | Number |
| **Fremd-Schlüssel:** | ***AnschriftID*** | Number |
| **Attribute:** | Vorname | Varchar (50) |
|  | Nachname | Varchar (50) |
|  | Geburtsdatum | Date |
|  | Geschlecht | Char (1) |

# Select-Statements

Die Folgenden Select-Statements wurden von den bestehenden Teammitgliedern entwickelt. Dieses Dokument wurde auch als *Mini-Kanban* verwendet, weshalb die untenstehenden Tabellen einen Status Wert mit ABGESCHLOSSEN haben. Die Werte, die verwendet wurden, sind:

|  |  |  |  |
| --- | --- | --- | --- |
| BEREIT | IN ARBEIT | ZUM TESTEN | ABGESCHLOSSEN |
|  |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Status: | BEREIT | *Zugewiesen an:* | HÖL | Optional: | N |
| Art: | SELECT | *Name:* | Liga-Matches | | |
| Aufgabe: | Es ist ein Select-Statement geschrieben, welches angibt, wie viele Matches einer Liga zugewiesen sind. | | | | |
|  |
| Code: |  | | | | |
|  |
| Ausführung: |  | | | | |
|  |
| Resultat (Z.B. Screenshot): |  | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Status: | BEREIT | *Zugewiesen an:* | PCH | Optional: | N |
| Art: | ABGESCHLOSSEN | *Name:* | Trainer-Spieler | | |
| Aufgabe: | Es ist ein Select-Statement geschrieben, dass alle Trainer auflistet und die Spieler, die er trainiert. Es ist nach den Trainern gruppiert. | | | | |
|  |
| Code: | SELECT CONCAT(CONCAT(Person.vorname,' '),Person.nachname) AS Trainer, Mannschaft.name AS Mannschaft, tmp\_Spieler.Spieler as Spieler FROM Person  JOIN Trainer ON (Person.personid = Trainer.personid)  JOIN MANNSCHAFT\_TRAINER ON MANNSCHAFT\_TRAINER.TrainerID = Trainer.Personid  JOIN MANNSCHAFT ON MANNSCHAFT\_TRAINER.Mannschaftid = Mannschaft.mannschaftid  JOIN (SELECT Spieler.mannschaftid as ID, CONCAT(CONCAT(Person.vorname,' '),Person.nachname) AS Spieler FROM Person  JOIN Spieler ON (Person.personid = Spieler.personid)) tmp\_Spieler  ON tmp\_Spieler.ID = Mannschaft.mannschaftid  ORDER BY Trainer,Spieler; | | | | |
|  |
| Ausführung: |  | | | | |
|  |
| Resultat (Z.B. Screenshot): |  | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Status: | ABGESCHLOSSEN | *Zugewiesen an:* | SEI | Optional: | N |
| Art: | SELECT | *Name:* | Tore-Durchschnitt | | |
| Aufgabe: | Es ist ein Select-Statement geschrieben, welches den durchschnittlichen Tore-Stand aller Mannschaften ausgibt. Es soll nach den besten Mannschaften absteigend geordnet werden. | | | | |
|  |
| Code: | SELECT Mannschaft.Name AS Mannschaft, ((HEIM.tore\_stand+GAST.tore\_stand)/(HEIM.anz+GAST.anz)) AS TORE\_DURCHSCHNITT FROM MANNSCHAFT  JOIN  (SELECT MANNSCHAFT\_MATCH.Heimmannschaftid AS mID, SUM(regexp\_substr(tore\_stand,'[^/]+',1)) AS tore\_stand, COUNT(tore\_stand) as anz  FROM MANNSCHAFT\_MATCH JOIN MATCH USING (MatchID)GROUP BY MANNSCHAFT\_MATCH.Heimmannschaftid) HEIM  ON Mannschaft.MannschaftID = HEIM.mID  JOIN  (SELECT MANNSCHAFT\_MATCH.GastmannschaftID AS mID, SUM(regexp\_substr(tore\_stand,'[^/]+',2)) AS tore\_stand, COUNT(tore\_stand) as anz  FROM MANNSCHAFT\_MATCH JOIN MATCH USING (MatchID)GROUP BY MANNSCHAFT\_MATCH.GastmannschaftID) GAST  ON Mannschaft.MannschaftID = GAST.mID  ORDER BY TORE\_DURCHSCHNITT DESC, Mannschaft; | | | | |
|  |
| Ausführung: | - | | | | |
|  |
| Resultat (Z.B. Screenshot): |  | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Status: | ABGESCHLOSSEN | *Zugewiesen an:* | CAR | Optional: | N |
| Art: | SELECT | *Name:* | Verein-Daten | | |
| Aufgabe: | Es werden alle Vereine mit ihrer Anschrift, der Anzahl der Mannschaften ausgegeben werden. | | | | |
|  |
| Code: | A picture containing chart  Description automatically generated | | | | |
|  |
| Ausführung: | - | | | | |
|  |
| Resultat (Z.B. Screenshot): | Graphical user interface, application, table  Description automatically generated | | | | |

# DB-Users

Password für alle erstellten User soll sein: ganz\_geheim!

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Status: | ABGESCHLOSSEN | *Zugewiesen an:* | PCH | Optional: | N |
| Art: | User | *Name:* | User\_op | | |
| Aufgabe: | Es ist ein User erstellt, der alle Rechte in der FBV-DB hat, allerdings keine Rechte in anderen Schemata. Er darf keine User / Rollen erstellen oder vergeben. | | | | |
|  |
| Code: | CREATE USER admin  IDENTIFIED BY ganz\_geheim  DEFAULT TABLESPACE data\_ts  QUOTA 100M ON test\_ts  QUOTA 500K ON data\_ts  TEMPORARY TABLESPACE temp\_ts  PROFILE admin;    GRANT CONNECT TO admin;  GRANT CREATE SESSION TO admin;  GRANT CREATE TABLE TO admin;  GRANT CREATE PROCEDURE TO admin;  GRANT CREATE TRIGGER TO admin;  GRANT CREATE VIEW TO admin;  GRANT CREATE SEQUENCE TO admin;  GRANT ALTER ANY TABLE TO admin;  GRANT ALTER ANY PROCEDURE TO admin;  GRANT ALTER ANY TRIGGER TO admin;  GRANT DELETE ANY TABLE TO admin;  GRANT DROP ANY TABLE TO admin;  GRANT DROP ANY PROCEDURE TO admin;  GRANT DROP ANY TRIGGER TO admin;  GRANT DROP ANY VIEW TO admin; | | | | |
|  |
| Ausführung: |  | | | | |
|  |
| Resultat (Z.B. Screenshot): |  | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Status: | BEREIT | *Zugewiesen an:* | HÖL | Optional: | J |
| Art: | User | *Name:* | User\_r | | |
| Aufgabe: | Es ist ein User erstellt, der keine Create Rechte hat und keine Daten hinzufügen darf (Unabhängig der Methode). Er darf lediglich Daten über vordefinierte Views oder SELECTS auslesen. In beiden Fällen ist der User auf das FBV-Schema limitiert. | | | | |
|  |
| Code: |  | | | | |
|  |
| Ausführung: |  | | | | |
|  |
| Resultat (Z.B. Screenshot): |  | | | | |

# VIEWS

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Status: | ABGESCHLOSSEN | *Zugewiesen an:* | PCH | Optional: | N |
| Art: | VIEW | *Name:* | list\_matches | | |
| Aufgabe: | Es werden alle Matches ausgegeben, die in einer Liga geplant oder gespielt sind / wurden. | | | | |
|  |
| Code: | CREATE OR REPLACE VIEW show\_matches  AS SELECT match.matchid as "MatchID", match.datum as "Date", liga.name as "Liga"  FROM match JOIN liga ON match.ligaID = liga.ligaID ORDER BY liga.name,match.matchid; | | | | |
|  |
| Ausführung: | select \* from show\_matches; | | | | |
|  |
| Resultat (Z.B. Screenshot): |  | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Status: | BEREIT | *Zugewiesen an:* | HÖL | Optional: | N |
| Art: | VIEW | *Name:* | List\_person | | |
| Aufgabe: | Es werden alle Personen ausgegeben, mit der aktiven Rolle, als auch mit den früheren Rollen. | | | | |
|  |
| Code: |  | | | | |
|  |
| Ausführung: |  | | | | |
|  |
| Resultat (Z.B. Screenshot): |  | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Status: | ABGESCHLOSSEN | *Zugewiesen an:* | CAB | Optional: | N |
| Art: | VIEW | *Name:* | Show\_ranking | | |
| Aufgabe: | Es werden alle Mannschaften eines Vereins angezeigt. Die Mannschaften werden alphabethisch nach dem Namen des Vereins und der Mannschaft geordnet. (Die Mannschaften sind nicht nach dem Verein gruppiert, der Verein wird einfach mehrfach ausgegeben). | | | | |
|  |
| Code: | Text  Description automatically generated | | | | |
|  |
| Ausführung: | SELECT \* FROM show\_ranking; | | | | |
|  |
| Resultat (Z.B. Screenshot): | Table  Description automatically generated | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Status: | ABGESCHLOSSEN | *Zugewiesen an:* | SEI | Optional: | N |
| Art: | VIEW | *Name:* | Show\_current\_ranking | | |
| Aufgabe: | Es wird ein Ranking einer Liga ausgegeben. Es zeigt an, welche Mannschaft im Moment am besten ist und die meisten Punkte hat. Die Punkte werden in dieser View als zusätzliche Spalte ausgegeben und nach den gewonnenen Spielen berechnet:   * Gewonnen: 3 Punkte * Unentschieden: 1 Punkt * Verloren: 0 Punkte | | | | |
|  |
| Code: | CREATE OR REPLACE VIEW show\_current\_ranking AS  SELECT tmp.liganame AS "Liganame", tmp.Mannschaft AS "Mannschaft",  COUNT(tmp.ergebniss) AS "Gespielt",  COUNT(CASE WHEN tmp.ergebniss = 'W' THEN 1 END) AS "Gewonnen",  COUNT(CASE WHEN tmp.ergebniss = 'D' THEN 1 END) AS "Unnentschieden",  COUNT(CASE WHEN tmp.ergebniss = 'L' THEN 1 END) AS "Verloren",  SUM(CASE WHEN tmp.ergebniss = 'W' THEN 3 WHEN tmp.ergebniss = 'D' THEN 1 WHEN tmp.ergebniss = 'L' THEN 0 ELSE 0 END) AS "Punkte" FROM (    SELECT LIGA.name AS Liganame, (VEREIN.Name||' - '||Mannschaft.Name) AS Mannschaft,  CASE  WHEN Mannschaft\_Match.Heimmannschaftid = Mannschaft.MannschaftID AND regexp\_substr(MATCH.Tore\_Stand,'[^/]+',1) > regexp\_substr(MATCH.Tore\_Stand,'[^/]+',2) THEN 'W'  WHEN Mannschaft\_Match.Heimmannschaftid = Mannschaft.MannschaftID AND regexp\_substr(MATCH.Tore\_Stand,'[^/]+',1) = regexp\_substr(MATCH.Tore\_Stand,'[^/]+',2) THEN 'D'  WHEN Mannschaft\_Match.Heimmannschaftid = Mannschaft.MannschaftID AND regexp\_substr(MATCH.Tore\_Stand,'[^/]+',1) < regexp\_substr(MATCH.Tore\_Stand,'[^/]+',2) THEN 'L'  END AS Ergebniss FROM LIGA    JOIN MATCH USING (LigaID)  JOIN Mannschaft\_Match USING (MatchID)  JOIN Mannschaft ON Mannschaft\_Match.Heimmannschaftid = Mannschaft.MannschaftID  JOIN VEREIN USING (VereinID)  UNION  SELECT LIGA.name AS Liganame, (VEREIN.Name||' - '||Mannschaft.Name) AS Mannschaftname,  CASE  WHEN Mannschaft\_Match.GastmannschaftID = Mannschaft.MannschaftID AND regexp\_substr(MATCH.Tore\_Stand,'[^/]+',2) > regexp\_substr(MATCH.Tore\_Stand,'[^/]+',1) THEN 'W'  WHEN Mannschaft\_Match.GastmannschaftID = Mannschaft.MannschaftID AND regexp\_substr(MATCH.Tore\_Stand,'[^/]+',2) = regexp\_substr(MATCH.Tore\_Stand,'[^/]+',1) THEN 'D'  WHEN Mannschaft\_Match.GastmannschaftID = Mannschaft.MannschaftID AND regexp\_substr(MATCH.Tore\_Stand,'[^/]+',2) < regexp\_substr(MATCH.Tore\_Stand,'[^/]+',1) THEN 'L'  ELSE 'U'  END AS Ergebniss FROM LIGA  JOIN MATCH USING (LigaID)  JOIN Mannschaft\_Match USING (MatchID)  JOIN Mannschaft ON Mannschaft\_Match.GastmannschaftID = MannschaftID  JOIN VEREIN USING (VereinID)  ) tmp  GROUP BY tmp.liganame,tmp.Mannschaft  ORDER BY 1,7 DESC; | | | | |
|  |
| Ausführung: | SELECT \* FROM show\_current\_ranking; | | | | |
|  |
| Resultat (Z.B. Screenshot): |  | | | | |

# Procedures & Functions

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Status: | ABGESCHLOSSEN | *Zugewiesen an:* | ~~HÖL~~ CAB | Optional: | N |
| Art: | Function | *Name:* | getPersonID | | |
| Aufgabe: | Es wird die PersonID einer Person zurückgegeben.  Input Parameter sind Vorname und Nachname.  Folgende Return Codes sind definiert:  ret = -3 … Falsche Eingabe (Z.B. falsche Anzahl an Parameter)  ret = -2 … Mehrere Personen gefunden  ret = -1 ... Person nicht gefunden  ret >= 0 … PersonID | | | | |
|  |
| Code: | create or replace function getPersonID(i\_vorname in varchar2, i\_nachname in varchar2)  return Number  is  v\_count\_pers Number;  v\_persid Number;  begin  select count(\*) into v\_count\_pers from person where vorname = i\_vorname and nachname = i\_nachname;  if v\_count\_pers > 1 then  return -2;  elsif v\_count\_pers < 1 then  return -1;  else  select personid into v\_persid from person where vorname = i\_vorname and nachname = i\_nachname;  return v\_persid;  end if;  end;  / | | | | |
|  |
| Ausführung: | select getPersonID('Tanek', 'Lucas') from dual;  select getPersonID('Valentin', 'Caraba') from dual; | | | | |
|  |
| Resultat (Z.B. Screenshot): |  | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Status: | BEREIT | *Zugewiesen an:* | PCH | Optional: | N |
| Art: | Function | *Name:* | getVereinID | | |
| Aufgabe: | Es wird die VereinID einer Person zurückgegeben.  Input Parameter ist die PersonenID oder Vorname und Nachname (TIPP: Function-Overloading).  Wenn Vorname und Nachname eingegeben sind wird die PersonenID mit getPersonID() abgefragt.  Folgende Return Codes sind definiert:  ret = -4X … Mehrere Vereine gefunden. Person ist aktuell bei VereinID = X.  ret = -3 … Falsche Eingabe (Z.B. falsche Anzahl an Parameter)  ret = -2 … Mehrere Personen gefunden  ret = -1 ... Person nicht gefunden  ret >= 0 … VereinID. | | | | |
|  |
| Code: |  | | | | |
|  |
| Ausführung: |  | | | | |
|  |
| Resultat (Z.B. Screenshot): |  | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Status: | ABGESCHLOSSEN | *Zugewiesen an:* | CAB | Optional: | N |
| Art: | Function | *Name:* | personIsA | | |
| Aufgabe: | Es wird zurückgegeben, als was eine Person gespeichert ist.  Input Parameter ist die PersonenID oder Vorname und Nachname.  (TIPP: Function-Overloading und defaultParameter).  Wenn Vorname und Nachname eingegeben sind wird die PersonenID mit getPersonID() abgefragt.  Folgende Return Codes sind definiert:  ret = -2 … Mehrere Personen gefunden  ret = -1 ... Person nicht gefunden  ret = 0 … Die Person wird in keiner Untergruppe gefunden.  Ret = X … Die Person ist ein (X=1; Mitarbeiter / X=2; Spieler /  X=3; Trainer / X=4; Schiedsrichter) | | | | |
|  |
| Code: | Graphical user interface, text  Description automatically generated with medium confidence | | | | |
|  |
| Ausführung: | SELECT personIsA(631) FROM dual; | | | | |
|  |
| Resultat (Z.B. Screenshot): | Graphical user interface, text, application, email  Description automatically generated | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Status: | IN ARBEIT | *Zugewiesen an:* | SEI | Optional: | N |
| Art: | Function | *Name:* | personHistory | | |
| Aufgabe: | Es wird zurückgegeben, ob die Person die Art gewechselt hat. Also ob die Person als Spieler in einem Verein war und jetzt als Trainer arbeitet.  Input Parameter ist die PersonenID / Vorname und Nachname / VereinID und PersonID / VereinID und Vorname und Nachname (TIPP: Function-Overloading).  Wenn Vorname und Nachname eingegeben sind wird die PersonenID mit getPersonID () abgefragt. Die VereinID kann mit der PersonenID angegeben werden  Folgende Return Codes sind definiert:  ret = -5 … Die Person war bei mehr als 2 Vereinen.  Ret = -4X00Z00Y …  X = Aktueller Verein  Z = Vorheriger Verein  Y = Code gleicht dem Ergebniss ret >= 0  ret = -3 … Falsche Eingabe (Z.B. falsche Anzahl an Parameter)  ret = -2 … Mehrere Personen gefunden  ret = -1 ... Die Person wird in keiner Untergruppe gefunden.  ret = 0 … Die Person ist keiner Untergruppe zugeordnet.  ret = A[B][C][..] … Die Person ist A, war zuvor B, war vor B C, … ABC usw. sind nach dem gleichen Prinzip aufgebaut wie (ret => 1) aus der Funktion personIsA. | | | | |
|  |
| Code: |  | | | | |
|  |
| Ausführung: |  | | | | |
|  |
| Resultat (Z.B. Screenshot): |  | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Status: | ABGESCHLOSSEN | *Zugewiesen an:* | CAB | Optional: | N |
| Art: | Procedure | *Name:* | addPerson | | |
| Aufgabe: | In dieser Procedure wird eine Person hinzugefügt. Es wird mit der Procedure sichergestellt, dass die Person auch einer Untergruppe zugewiesen ist (Mitarbeiter, Spieler, Trainer oder Schiedsrichter).  Ist die Person ein Mitarbeiter, wird sie gleich einem Verein zugewiesen.  Ist die Person ein Spieler oder ein Trainer wird sie gleich einer Mannschaft zugewiesen.  Die ID der Person wird automatisch erstellt.  Input Parameter sind: vname, nname, gebDat, geschlecht, eintrittsdatum  Zusätzlich Inputparameter: isA, funktionStatus, VereinMannschaftID  Optionaler Parameter ist (Tipp: Procedure Overloading): anschriftID.  Vname,nname,gebDat und Geschlecht dürfen nicht null sein.  Ist das Eintritts Datum null, wird es automatisch das aktuelle Datum genommen.  isA darf nicht null sein und ist nach dem gleichen Prinzip aufgebaut wie (ret => 1) aus der Funktion personIsA.  funktionStatus darf null sein, wenn isA = 3/4.  VereinMannschaftID darf null sein, wenn isA = 4.  anschriftID darf null sein oder weggelassen werden. | | | | |
|  |
| Code: | Graphical user interface, text, application, email  Description automatically generated | | | | |
|  |
| Ausführung: |  | | | | |
|  |
| Resultat (Z.B. Screenshot): | Table  Description automatically generated | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Status: | ABGESCHLOSSEN | *Zugewiesen an:* | PCH | Optional: | N |
| Art: | Procedure | *Name:* | addLiga | | |
| Aufgabe: | In dieser Procedure wird eine Liga hinzugefügt. Es wird mit der Procedure sichergestellt, dass die Liga auch mindestens aus 5 Matches besteht.  Die ID der Liga wird automatisch erstellt.  Input Parameter sind: name, beginn, ende, altersgruppe, geschlecht, t\_matches  Kein Input Parameter darf null sein.  Die Anschrift kann temporär auf -1.  Dafür muss eine Pseudo-Anschrift, ein Pseudo-Ort und ein Pseudo-Land erstellt werden.  Der input Parameter t\_matches ist vom Type ownArr.  Der Type ownArr kann mehrere Number Datentypen beinhalten und ist loopable. | | | | |
|  |
| Code: | create or replace procedure addLiga(  v\_name Varchar,  v\_begin\_d Date,  v\_end\_d Date,  v\_altersgruppe Number,  v\_geschlecht Varchar,  t\_matches ownArr)  as  v\_Ligaid Number;  begin  if t\_matches.count < 5 then  raise\_application\_error(-20001,'Minimum rows for matches: 5');  else  SELECT max(Liga.ligaid)+1 INTO v\_Ligaid FROM Liga;    INSERT INTO liga(ligaid, datum\_begonnen, datum\_geendet, name, altersgruppe, geschlechts\_gruppe) VALUES (v\_Ligaid, v\_begin\_d,v\_end\_d,v\_name,v\_altersgruppe,v\_geschlecht);    FOR i IN 1..t\_matches.count  LOOP  INSERT INTO Match (Matchid,Anschriftid,Ligaid) VALUES (t\_matches(i),-1,v\_Ligaid);  END LOOP;  end if;  end;  /  INSERT INTO Land (Landid) VALUES (-1);  INSERT INTO ORT (PLZ,Landid) VALUES (-1,-1);  INSERT INTO Anschrift (AnschriftID,PLZ) VALUES (-1,-1); | | | | |
|  |
| Ausführung: | DECLARE  v\_na ownArr;  begin  v\_na := ownArr();  v\_na.EXTEND(5);  v\_na(1) := 105;  v\_na(2) := 106;  v\_na(3) := 107;  v\_na(4) := 108;  v\_na(5) := 109;  addLiga('test',TO\_DATE('05/11/2020','DD-MM-YYYY'),TO\_DATE('22/01/2021','DD-MM-YYYY'),15,'M',v\_na);  end; | | | | |
|  |
| Resultat (Z.B. Screenshot): |  | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Status: | BEREIT | *Zugewiesen an:* | SEI | Optional: | N |
| Art: | Procedure | *Name:* | addMannschaft | | |
| Aufgabe: | In dieser Procedure wird eine Mannschaft hinzugefügt. Es wird mit der Procedure sichergestellt, dass die Mannschaft auch einem Verein zugewiesen ist und mind. Einen Trainer hat.  Ist die Person ein Mitarbeiter, wird sie gleich einem Verein zugewiesen.  Die ID der Mannschaft wird automatisch erstellt.  Input Parameter sind: name, altersgruppe, t\_trainer  Kein Input Parameter darf null sein.  Der input Parameter t\_trainer ist vom Type ownArr.  Der Type ownArr kann mehrere Number Datentypen beinhalten und ist loopable. | | | | |
|  |
| Code: |  | | | | |
|  |
| Ausführung: |  | | | | |
|  |
| Resultat (Z.B. Screenshot): |  | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Status: | BEREIT | *Zugewiesen an:* | HÖL | Optional: | N |
| Art: | Procedure | *Name:* | addMatch | | |
| Aufgabe: | In dieser Procedure wird ein Match hinzugefügt. Es wird mit der Procedure sichergestellt, dass das Match auch mind. 1 Schiedrichter zugewiesen hat und das genau zwei Mannschaften bestimmt sind.  Die ID des Matches wird automatisch erstellt.  Input Parameter sind: datum, begonnen, geendet, heim, gast, t\_schiedsrichter  Kein Input Parameter darf null sein.  Der input Parameter t\_schiedsrichter ist vom Type ownArr.  Der Type ownArr kann mehrere Number Datentypen beinhalten und ist loopable. | | | | |
|  |
| Code: |  | | | | |
|  |
| Ausführung: |  | | | | |
|  |
| Resultat (Z.B. Screenshot): |  | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Status: | ABGESCHLOSSEN | *Zugewiesen an:* | SEI | Optional: | N |
| Art: | TYPE | *Name:* | ownArr | | |
| Aufgabe: | Eine Instanz dieser Type kann mehrere Number-Datensätze speichern und ausgeben. | | | | |
|  |
| Code: | CREATE OR REPLACE TYPE ownArr AS VARRAY(50) OF NUMBER;  / | | | | |
|  |
| Ausführung: | DECLARE  v\_na ownArr;  BEGIN  v\_na := ownArr();  v\_na.EXTEND(3);  v\_na(1) := 64;  v\_na(2) := 65;  v\_na(3) := 66;  -- Other possibility to fill:  -- v\_na := (64,65,66);  FOR i IN 1..v\_na.count  LOOP  DBMS\_OUTPUT.PUT\_LINE('Part of v\_na:'||v\_na(i));  END LOOP;  END; | | | | |
|  |
| Resultat (Z.B. Screenshot): |  | | | | |

# Package

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Status: | BEREIT | *Zugewiesen an:* | CAB | Optional: | N |
| Art: | PACKAGE | *Name:* | fbv | | |
| Aufgabe: | Dieses Package hat folgende public Members:  AddMatch - Procedure  addMannschaft - Procedure  addLiga - Procedure  addPerson - Procedure  personHistory - Function  personIsA - Function  getVereinID - Function  getPersonID - Function  Und folgenden private Members:  ownArr - Type  getNextId – optional Function | | | | |
|  |
| Code: |  | | | | |
|  |
| Ausführung: |  | | | | |
|  |
| Resultat (Z.B. Screenshot): |  | | | | |

# Triggers

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Status: | In Arbeit | *Zugewiesen an:* | PCH | Optional: | N |
| Art: | TRIGGER | *Name:* | - | | |
| Aufgabe: | Nach dem Hinzufügen einer Liga soll sichergestellt werden, dass auch die Daten hinzugefügt wurden (Beginn und Ende). Ansonsten soll der Insert Rückgängig gemacht werden. | | | | |
|  |
| Code: |  | | | | |
|  |
| Ausführung: |  | | | | |
|  |
| Resultat (Z.B. Screenshot): |  | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Status: | BEREIT | *Zugewiesen an:* | SEI | Optional: | N |
| Art: | TRIGGER | *Name:* | - | | |
| Aufgabe: | Wird eine Person hinzugefügt soll kontrolliert werden, ob diese Person bereits in einer anderen Rolle oder in einem anderen Verein tätig ist. Wenn dem so ist soll diese Aktion wieder Rückgängig gemacht werden. (Es soll gecheckt werden, ob das Austrittsdatum bereits eingetragen ist) | | | | |
|  |
| Code: |  | | | | |
|  |
| Ausführung: |  | | | | |
|  |
| Resultat (Z.B. Screenshot): |  | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Status: | BEREIT | *Zugewiesen an:* | HÖL | Optional: | N |
| Art: | TRIGGER | *Name:* | - | | |
| Aufgabe: | Wird eine Person gelöscht soll diese Aktion rückgängig gemacht werden. Es wird nur das Austritts Datum der Person gesetzt.  (Dieser Trigger soll nicht für den User: User\_r gefeuert werden. Dieser Unterpunkt ist optional, da ich nicht weiß, ob er möglich ist) | | | | |
|  |
| Code: |  | | | | |
|  |
| Ausführung: |  | | | | |
|  |
| Resultat (Z.B. Screenshot): |  | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Status: | ABGESCHLOSSEN | *Zugewiesen an:* | CAB | Optional: | N |
| Art: | TRIGGER | *Name:* | - | | |
| Aufgabe: | Wird irgendeine Aktion getätigt sollen folgende Punkte in einer log-Tabelle gespeichert werden:   * User * Aktion * Datum * Uhrzeit   Die Log-Tabelle muss dafür erstellt werden. Fügen sie auch das Drop-Statement hinzu. | | | | |
|  |
| Code: | Text  Description automatically generated | | | | |
|  |
| Ausführung: |  | | | | |
|  |
| Resultat (Z.B. Screenshot): | Graphical user interface, text, application  Description automatically generated | | | | |