

# Bericht Datenbankprojekt

DBS, FS24

Football Talent Scout

Team 23

Fabian Stettler ([fabian.stettler@stud.hslu.ch](mailto:fabian.stettler@stud.hslu.ch))

Dominik Arnet ([dominik.arnet@stud.hslu.ch](mailto:dominik.arnet@stud.hslu.ch))

05.04.2024

<b>1 EINFÜHRUNG &amp; AUSGANGSLAGE .....</b>	<b>4</b>
<b>2 PROJEKTIDEE &amp; USE CASE.....</b>	<b>4</b>
2.1 MOTIVATION .....	4
2.2 PERSONAS.....	4
2.3 ENTSCHEIDUNGSUNTERSTÜTZUNG FÜR WERTGENERIERUNG .....	4
2.4 ERFORDERLICHE DATENQUELLEN FÜR DIE ANALYSE.....	5
2.5 DATENBANKTECHNOLOGIE MIT BEGRÜNDUNG.....	5
<b>3 DATENMODEL &amp; DATENBANKSCHEMA .....</b>	<b>6</b>
3.1 KONZEPTIONELLES SCHEMA .....	7
3.2 ERSTELLUNG DER TABLES.....	8
3.2.1 <i>In SQL (DDL)</i> .....	8
3.2.1.1     Constraints .....	13
3.2.2 <i>Datenbankschema</i> .....	15
3.3     NORMALISIERUNG SQL .....	16
3.3.1 <i>club_has_stadium_name</i> .....	16
3.3.2 <i>club_has_seats</i> .....	16
3.3.3 <i>club_has_domestic_competition</i> .....	16
3.3.4 <i>game_event_has_date</i> .....	17
3.4     IN MONGO DB (JSON-PROTOTYPEN).....	17
3.4.1.1     Players Document .....	17
3.4.1.2     Games Document .....	18
3.4.1.3     Clubs Document .....	20
3.5     DENORMALISIERUNG JSON .....	22
<b>4 DATEN LADEN &amp; TRANSFORMIEREN .....</b>	<b>23</b>
4.1     VISUALISIERUNG BENUTZTER SYSTEME .....	23
4.2     LOGINDATEN .....	23
4.3     VORGEHEN LADEN DER DATEN .....	24
4.3.1 <i>Load SQL</i> .....	24
4.3.2 <i>Transform SQL</i> .....	29
4.3.3 <i>Load MongoDB</i> .....	31
4.3.4 <i>Transform MongoDB</i> .....	32
4.3.4.1     Datentyp Anpassungen .....	32
4.3.4.2     Erstellen der JSON-Dokumente .....	34
4.3.4.3     Weitere Dokumente Games und Players Documents .....	37
<b>5 DATEN ANALYSIEREN &amp; AUSWERTEN.....</b>	<b>38</b>
5.1     DATENABFRAGEN ERKLÄRUNG .....	38
5.2     SQL ABFRAGEN .....	39
5.2.1 <i>Abfrage scouting Spieler SQL</i> .....	39
5.2.2 <i>Abfrage analysis Teams mit späten Treffern SQL</i> .....	40
5.2.3 <i>Abfrage analysis Aufstellung Gegner</i> .....	41
5.3     MONGO DB ABFRAGEN.....	41
5.3.1 <i>Abfrage scouting Spieler MongoDB</i> .....	41
5.3.2 <i>Abfrage analysis Teams mit späten Treffern MongoDB</i> .....	44
5.3.3 <i>Abfrage analysis Aufstellung des Gegners in MongoDB</i> .....	46
5.3.4 <i>Diskussion der Resultate</i> .....	48
<b>6 EFFIZIENZ &amp; PERFORMANCE .....</b>	<b>49</b>
6.1     OPTIMIERUNG MONGODB.....	49

6.2	OPTIMIERUNG SQL.....	55
<b>7</b>	<b>DATENSCHUTZ &amp; DATENBANKSICHERHEIT .....</b>	<b>60</b>
7.1	BESCHREIBUNG DER SICHERHEITSZIELE .....	60
7.2	MASSNAHMEN.....	61
7.2.1	<i>Definition Massnahmen Verfügbarkeit SQL .....</i>	61
7.2.2	<i>Umsetzung Massnahmen Verfügbarkeit SQL .....</i>	61
7.2.2.1	Backup mit mysqldump .....	61
7.2.3	<i>Umsetzung Massnahmen Verfügbarkeit MongoDB.....</i>	63
7.2.4	<i>Definition Massnahmen Vertraulichkeit.....</i>	64
7.2.4.1	Umsetzung Massnahmen Vertraulichkeit SQL .....	65
7.2.4.2	Passwörter.....	65
7.2.4.3	User Privilegien.....	65
7.2.5	<i>Umsetzung Massnahmen Vertraulichkeit MongoDB .....</i>	70
7.2.5.1	User Erstellung .....	70
7.2.6	<i>Definition Massnahmen Integrität SQL .....</i>	71
7.2.7	<i>Umsetzung Massnahmen Integrität SQL.....</i>	72
7.2.8	<i>Umsetzung Massnahmen Integrität MongoDB.....</i>	74
<b>8</b>	<b>VISUALISIERUNG &amp; ENTScheidungsunterstützung .....</b>	<b>74</b>
8.1	ZUGRIFF AUF VM UND METABASE STARTEN .....	74
8.2	ANBINDUNGEN DER DATENBANKEN .....	74
8.2.1	<i>SQL Anbindung .....</i>	74
8.2.2	<i>NoSQL Anbindung.....</i>	76
8.3	ÜBERSICHT DER VISUALISIERUNGEN.....	76
8.4	FORMATION WINS.....	77
8.4.1	<i>SQL.....</i>	77
8.4.1.1	Filter .....	78
8.4.1.2	Visualisierung .....	78
8.4.2	<i>NoSQL .....</i>	79
8.4.2.1	Visualisierung .....	80
8.4.3	<i>Vergleich der Resultate.....</i>	80
8.5	LATE GOALS .....	81
8.5.1	<i>SQL.....</i>	81
8.5.1.1	Visualisierung .....	82
8.5.2	<i>NoSQL .....</i>	82
8.5.2.1	Visualisierung .....	83
8.5.3	<i>Vergleich der Visualisierungen .....</i>	83
8.6	PLAYER SCOUTING .....	83
8.6.1	<i>SQL.....</i>	84
8.6.1.1	Filter .....	84
8.6.1.2	Visualisierung .....	85
8.6.2	<i>NoSQL .....</i>	85
8.6.2.1	Filter .....	86
8.6.2.2	Visualisierung .....	86
8.6.3	<i>Vergleich der Visualisierungen .....</i>	86
8.7	ENTSCHEIDUNGSUNTERSTÜTZUNG .....	86
<b>9</b>	<b>VERWENDUNG VON CHATGPT .....</b>	<b>88</b>
<b>10</b>	<b>FAZIT &amp; LESSONS LEARNED .....</b>	<b>88</b>

## **1 Einführung & Ausgangslage**

Im Rahmen des Moduls DBS soll je eine Datenbank mit MySQL und einem NoSQL Ansatz (MongoDB) erstellt werden. Die Daten dieser zwei DBs sollen mittels der Visualisierungssoftware Metabase anschaulich dargestellt werden, um einem oder mehreren Akteuren eine Entscheidungsunterstützung zu geben. Durch diese soll ein Mehrwert generiert werden.

Trainer oder Coach einer Fussballmannschaft zu sein ist sicherlich ein stressiger Job. Neue talentierte Spieler zu finden und feststellen welche Topscorer für die eigene Mannschaft am besten geeignet sind, ist kein Leichtes.

Football Talent Scout will hier Abhilfe schaffen. Ziel unseres Projektes ist es den Trainern und Coaches von Fussballmannschaften aktuelle Spielerdaten anschaulich zu präsentieren. Somit können sich diese ein besseres Bild über mögliche Kandidaten machen. Zusätzlich soll es den Coaches ermöglicht werden das gegnerische Team vor einem Duell zu analysieren.

## **2 Projektidee & Use Case**

### **2.1 Motivation**

Da wir beide sehr fussballbegeistert sind, lag es schnell auf der Hand, dass wir etwas rund um den weltbekannten Ballsport umsetzen möchten. Da uns aber nicht nur Fouls und Goals interessieren, sondern auch der Spielertransfermarkt und Spielstatistiken, setzten wir es uns zum Ziel einen Service zu erstellen welcher Coaches und Managers von Fussballmannschaften gross bis klein in ihren Entscheidungen unterstützt.

### **2.2 Personas**

Jose ist Trainer eines Vereins der zweitstärksten Schweizer Liga und möchte interessante Spieler für seinen Verein verpflichten. Er hat jedoch keine grossen Ressourcen im Scouting Bereich und ist deshalb im Nachteil gegenüber den grossen Clubs der ersten Schweizer Liga. Weil er nur einen einzigen Scout hat, möchte er den Kreis interessanter Spieler zuerst mit anderen Methoden einschränken. Mit dem scoutingAndAnalysis Tool kann er dies gut machen und so nur noch die sehr interessanten Spieler live beobachten.

Als die Saison von Jose dann gestartet ist, analysiert er dann den ersten Gegner. Da seine Ressourcen zur Gegnerbeobachtung auch knapp sind, hilft ihm auch in diesem Bereich das ScoutingAndAnalysis Tool. Er kann mit diesem zum Beispiel die Formation, die der Gegner oft gespielt hat, direkt aus dem Tool auslesen.

Unser Projekt zielt darauf ab, Personen wie Jose in seinen Entscheidungen zu unterstützen.

### **2.3 Entscheidungsunterstützung für Wertgenerierung**

Die Benutzer vom scoutingAndAnalysis Tool sollen Spieler anhand von einigen Merkmalen suchen, filtern und sortieren können.

Folgende nicht abschliessende Merkmale haben wir ausgesucht:

- Nachname
- Jetziger Club
- Alter
- Geburtsland
- Position
- Subposition
- Spielfuss
- Grösse

- Vertragsablaufdatum
- Name des Agenten
- Marktwert (in Euro)
- FIFA-Ratings

In einem zweiten Schritt wollen wir auch einem Verein die Möglichkeit geben, mithilfe unserer Entscheidungsunterstützung taktische Vorbereitungen vor oder während des Spiels vorzunehmen. Es soll dabei möglich sein, die wahrscheinliche Aufstellung des Gegners zu antizipieren und taktische Einwechslungen aufgrund der Spielweise des Gegners vorzunehmen. Ein Trainer kann dann analysieren, welche Teams, immer späte Tore erzielen und so eine Einwechselung eines defensiven Spielers gegen diese Teams vorbereiten.

## 2.4 Erforderliche Datenquellen für die Analyse

- **Football Data from Transfermarkt:** Diese Datenquelle enthält über 60'000 Spiele aus mehreren Saisons, der grössten Wettbewerbe in Europa.  
Jedes game besteht aus game\_id, season, date (Spieldatum), home\_club\_id, away\_club\_id, home\_club\_goals, away\_club\_goals, stadium, dem referee und vielen Weiteren.  
Des Weiteren beinhaltet die Datenquelle ebenfalls Informationen zu Spielern und anderen Vereinen. Der Nutzer kann somit zu einem spezifischen Spieler herausfinden, wie viele Tore dieser erzielt, oder wie viele Minuten gespielt hat. Das sind alles wichtige Informationen für den Nutzer.
- Quelle: [https://www.kaggle.com/datasets/davidcariboo/player-scores?select=game\\_events.csv](https://www.kaggle.com/datasets/davidcariboo/player-scores?select=game_events.csv)  
Format: CSV (494.87MB)
- **FIFA Player Ratings:** Diese Datenquelle enthält diverse Spalten in welchen verschiedene Spielaspekte der Spieler von 0 bis 100 bewertet werden. Diese Bewertungen wurden im Rahmen des Video-Spiels Fifa 23 erhoben. Diese Daten werden ergänzend zu den Spielerdaten aus dem anderen Datenset in die gleiche Table geladen. Das Matching muss dabei über den vollen Namen der Spieler laufen, da dieses Datenset keine Primary Keys enthält. Insgesamt befinden sich 17530 Ratings in dem CSV.

Quelle: [https://www.kaggle.com/datasets/babatundezennith/fifa-archive?select=Fifa\\_23\\_Players\\_Data.csv](https://www.kaggle.com/datasets/babatundezennith/fifa-archive?select=Fifa_23_Players_Data.csv)  
Format: CSV (1041KB)

- **Limitationen beider Datenquellen**

Es gibt bei der ersten Datenquelle "Football Data from Transfermarkt" einige Daten, die nicht gebraucht werden und noch gelöscht werden müssen. Bei der Datenquelle "Fifa Player Ratings" ist die Aktualität der Daten sicherlich eine Limitation, da diese Bewertung stark schwanken können. Vor allem bei jungen Spielern kann es grosse Veränderungen in kurzen Zeitspannen geben. Diese Daten sind von dem Video Game Fifa 23, also doch bereits ein bisschen älter.

## 2.5 Datenbanktechnologie mit Begründung

Für unser Projekt verwenden wir MySQL und MongoDB. Die Datenbanksprache SQL ist schon sehr lange auf dem Markt und hat sich als robuste Lösung hervorgetan. MySQL wird oft gegenüber PostgreSQL bevorzugt, aufgrund seiner einfacheren Einrichtung und Verwendung.

MongoDB ist vor allem bei grossen Datensätzen oder auf mehreren Hardware-Systemen sehr interessant,

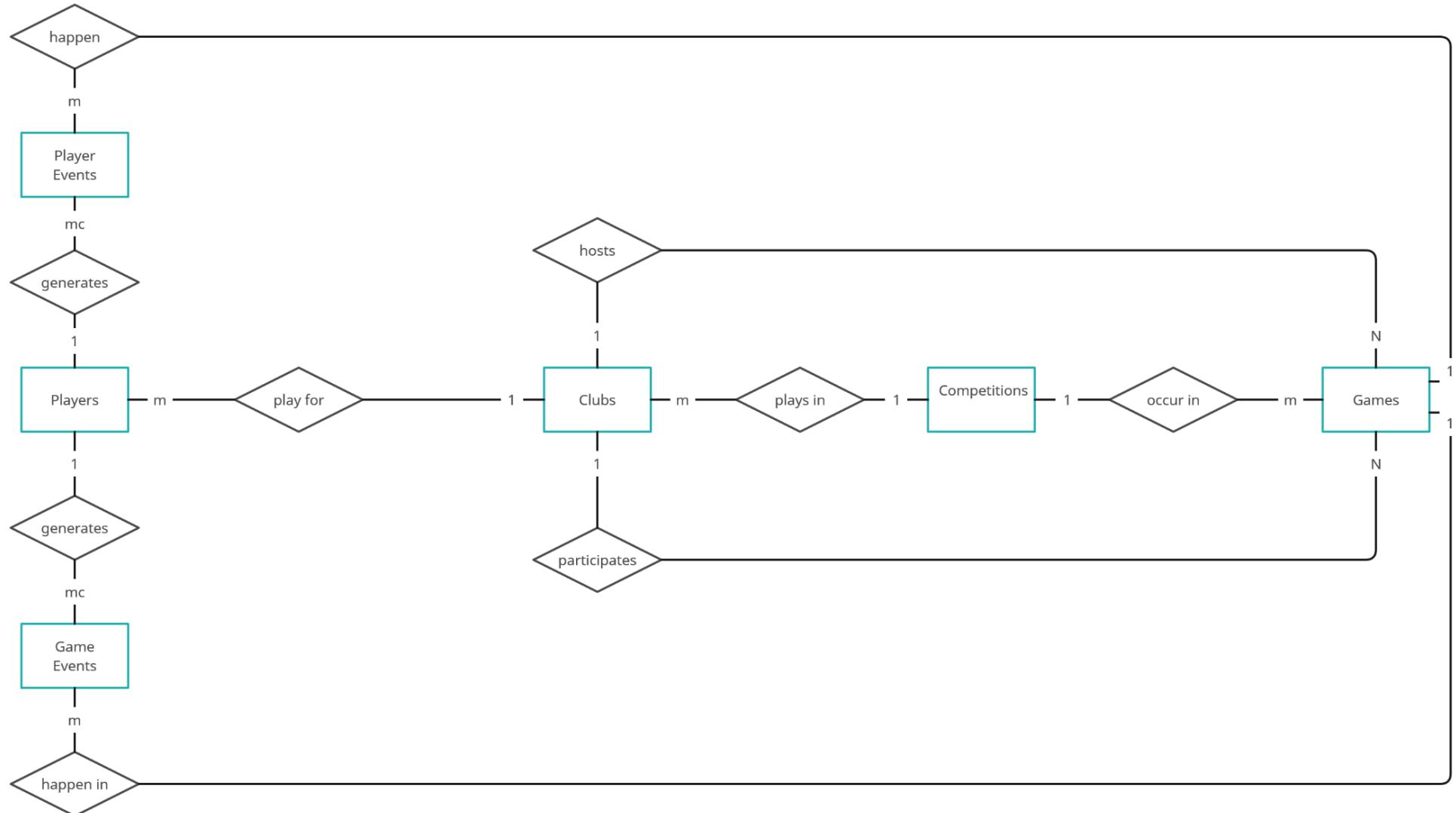
da es schnellere Resultate erzielen kann, aufgrund der Struktur wie Daten gespeichert werden. Im Gegensatz zu MySQL werden in MongoDB die Daten in Dokumenten gespeichert und nicht in Tabellen. Die Wahl der Datenbanktechnologien war aber in diesem Projekt vorgegeben, was unsere Wahl somit begründet. MySQL überzeugt im Gegensatz zu PostgreSQL vor allem durch seine Einfachheit in der Handhabung. MySQL hat dafür ein bisschen weniger advanced Möglichkeiten wie zum Beispiel, das Verwenden von Geo-Daten und erweiterte Datentypen.

Der Datenbank-Server von MySQL wird dabei auf einer virtuellen Maschine gehostet. Um die Datenbank zu bearbeiten, wird MySQL-Workbench genutzt, mit welcher auf den MySQL-Server zugegriffen wird. Die MongoDB Instanz läuft ebenfalls auf der virtuellen Maschine. Zuerst wurde diese im MongoDB Cluster in der Cloud gehostet, doch aufgrund von Memory-Engpässen haben wir diese dann neu lokal aufgesetzt. Mittels der Applikation Metabase werden die Daten der beiden Datenbanken visualisiert dargestellt für die Buntzer.

### **3 Datenmodel & Datenbankschema**

Basierend auf den beiden Datenquellen wurde das nachfolgende ER-Diagramm erstellt. Hierbei wurden bewusst die Attribute der Entitäten weggelassen, um für eine bessere Übersicht zu sorgen. Die Attribute werden nachfolgend genauer erklärt.

### 3.1 Konzeptionelles Schema



## 3.2 Erstellung der Tables

Aus dem oben dargestellten ERD wurden zehn SQL Tables für MySQL und drei Documents für MongoDB definiert. In den nachfolgenden Unterkapiteln wird erklärt, wie diese erstellt wurden.

### 3.2.1 In SQL (DDL)

Aufgrund der Normalisierung, welche im Kapitel 3.6 noch genauer erläutert wird, erstellten wir insgesamt zehn Tables in unser MySQL DB.

- players
- player\_events
- games
- game\_events
- game\_event\_has\_date
- clubs
- club\_has\_stadium\_name
- club\_has\_domestic\_competition
- club\_has\_seats
- Competitions

Die nachfolgenden Create Table Statements werden verwendet, um Tabellen in der MySQL Umgebung zu erstellen. Nach dem CREATE TABLE wird der Name des neuen tables definiert, es kann auch noch die betreffende Datenbank vor dem table spezifiziert werden. Danach werden die verschiedenen Namen der Attribute definiert mit dem dem jeweiligen Datentyp. Zudem spezifiziert man hier auch die Primary keys.

#### players

Der players Table enthält Informationen zu Spielern aus beiden Datenquellen. Aus dem CSV des Transfermarkts erhalten wir ID, Name, Vorname, die Letzte gespielte Saison, Geburtsland und Datum, Staatsangehörigkeit, Club, Position, Unterposition, Spielfuss, Grösse in cm, Vertrags Ablaufdatum, Name des Agenten, Liga und momentaner Marktwert in Euro. Aus dem Datenset von FIFA erhalten wir 12 Bewertungen auf einer Skala von 0-100 zu verschiedenen Aspekten des Spielers.

Die Spalte tmp\_fullname stammt aus den Transfermarkt Daten, ist aber bei den FIFA-Ratings auch vorhanden. Über dieses Feld kann daher später beim Laden der Daten das korrekte Matching garantiert werden.

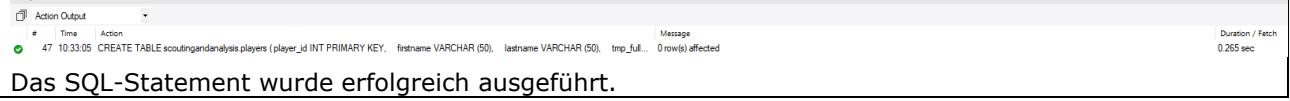
```
--Erstellen des Tables players

CREATE TABLE scoutingandanalysis.players (
    player_id INT PRIMARY KEY,
    firstname VARCHAR (50),
    lastname VARCHAR (50),
    tmp_fullname VARCHAR(255),
    last_season YEAR,
    current_club_id INT,
    country_of_birth VARCHAR (50),
    city_of_birth VARCHAR (50),
    country_of_citizenship VARCHAR (30),
    date_of_birth DATE,
    sub_position VARCHAR (20),
    position VARCHAR (20),
    foot VARCHAR (6),
    height_in_cm INT,
```

```

contract_exp_date DATE,
agent_name VARCHAR (30),
current_club Domestic_competition_id VARCHAR (6),
marketvalue_in_euro INT,
fifa_overall TINYINT,
fifa_potential TINYINT,
fifa_pace_total TINYINT,
fifa_shooting_total TINYINT,
fifa_passing_total TINYINT,
fifa_dribbling_total TINYINT,
fifa_defending_total TINYINT,
fifa_physicality_total TINYINT,
fifa_crossing TINYINT,
fifa_finishing TINYINT,
fifa_heading_accuracy TINYINT,
fifa_short_passing TINYINT
);

```

Das SQL-Statement wurde erfolgreich ausgeführt.

### **tmp\_fifa\_ratings**

Um im players Table die Daten der zweiten Datenquelle, die FIFA-Ratings, hinzuzufügen, erstellten wir eine temporäre Tabelle, in der sich die vollen Namen und die 12 Bewertungen zu dem jeweiligen Spieler befinden. Über den vollen Namen kann später die players Tabelle geupdatet werden und so die entsprechenden Ratings den Spielern hinzugefügt werden. In dem Daten von FIFA ist auch ein Marktwert enthalten, dieser wird aber nicht verwendet, da die Transfermarkt Datenquelle diesen Wert bereits zur Verfügung stellt.

```
--Erstellen des Tables tmp_fifa_ratings
```

```

CREATE TABLE scoutingandanalysis.tmp_fifa_ratings (
    full_name VARCHAR(255),
    overall TINYINT,
    potential TINYINT,
    value_in_euro INT,
    pace_total TINYINT,
    shooting_total TINYINT,
    passing_total TINYINT,
    dribbling_total TINYINT,
    defending_total TINYINT,
    physicality_total TINYINT,
    crossing TINYINT,
    finishing TINYINT,
    heading_accuracy TINYINT,
    short_passing TINYINT
);

```

Output			
#	Time	Action	Message
24	10:29:58	create table scoutingandanalysis.tmp_ffa_ratings ( full_name VARCHAR(255), overall TINYINT, potential TINYINT, value_in_euro INT, pa... )	0 row(s) affected

Das SQL-Statement wurde erfolgreich ausgeführt.

### player\_events

In der player\_events Tabelle werden Tore, Assists und Rote bzw. Gelbe Karten abgespeichert, welche ein Spieler (Referenz auf player\_id) in einem gewissen Spiel (Referenz auf game\_id) erzielt. Zusätzlich wird das Datum des Spiels und wie viele Minuten der Spieler aktive gespielt hat abgespeichert.

```
--Erstellen des Tables player_events
```

```
CREATE TABLE scoutingandanalysis.player_events (
    player_event_id VARCHAR(255),
    game_id INT,
    player_id INT,
    game_date DATE,
    yellow_cards INT,
    red_cards INT,
    goals INT,
    assists INT,
    minutes_played INT
);
```

Output			
#	Time	Action	Message
45	14:46:28	use scoutingandanalysis	0 row(s) affected
46	14:46:28	CREATE TABLE player_events ( player_event_id VARCHAR(255), game... )	0 row(s) affected

Das SQL-Statement wurde erfolgreich ausgeführt.

### games

in der games Tabelle sind Daten zu einzelnen Spielen enthalten. Neben der notwendigen ID auch noch in welcher Liga und welcher Saison das Spiel stattfand, die Runde (Matchday), das Datum, Referenz auf den Host und den Away Club, der Endstand der Tore (separat gespeichert pro Mannschaft), die Manager der beiden Mannschaften, in welchem Stadion das Spiel ausgetragen wurde, wie viele Leute zuschauten, wer der Schiedsrichter war und die Spielformation der beiden Clubs.

```
--Erstellen des Tables games
```

```
CREATE TABLE games (
    game_id INT PRIMARY KEY,
    competition_id VARCHAR(5),
    season YEAR,
    round VARCHAR(50),
    game_date DATE,
    home_club_id INT,
    away_club_id INT,
    home_club_goals INT,
    away_club_goals INT,
    home_club_manager_name VARCHAR(100),
```

```

        away_club_manager_name VARCHAR(100),
        stadium VARCHAR(100),
        attendance INT,
        referee VARCHAR(100),
        home_club_formation VARCHAR(40),
        away_club_formation VARCHAR(40)
    );

```

Output			
#	Time	Action	Message
42	11:00:04	use scoutingandanalysis	0 row(s) affected
43	11:00:04	CREATE TABLE games ( game_id INT PRIMARY KEY, competition_id VARCHAR...	0 row(s) affected

Das SQL-Statement wurde erfolgreich ausgeführt.

### game\_events

Die game\_events unterscheiden sich zu den player\_events insofern, dass sie Spiel Events mit einer genaueren Beschreibung festhalten und in welcher Spielminute diese geschehen sind. Zusätzlich wird festgehalten welcher Spieler, von welchem Club dieses Ereignis erzeugt hat. Bei Toren mit einem Assist und bei Spielerwechseln wird die Spieler ID des anderen Spielers auch gespeichert.

```
--Erstellen des Tables game_events

CREATE TABLE game_events (
    game_event_id VARCHAR(50) PRIMARY KEY,
    game_event_date date,
    game_id INT,
    `minute` INT,
    game_event_type VARCHAR(50),
    club_id INT,
    player_id INT,
    description VARCHAR(255),
    player_in_id INT,
    player_assist_id INT
);
```

3 17:05:42 CREATE TABLE game\_events ( game\_event\_id VARCHAR(50) PRIMARY KEY, game\_event\_date date, game\_id INT, `minute` INT, 0 row(s) affected

0.344 sec

Das SQL-Statement wurde erfolgreich ausgeführt.

### game\_event\_has\_date

Da das Datum der game\_events nicht abhängig ist von der game\_event\_id, sondern von der game\_id wurde das game\_event\_date in eine separate Tabelle ausgelagert.

```
--Erstellen des Tables game_events_has_date

CREATE TABLE game_events_has_date (
    game_event_id varchar(50),
    game_event_date date,
    primary key (game_event_id, game_event_date)
);
```

```
1 17:27:20 CREATE TABLE game_events_has_date ( game_event_id varchar(50), game_event_date date, primary key (game_event_id, game_event_date) ) 0 row(s) affected
```

Das SQL-Statement wurde erfolgreich ausgeführt.

### competitions

Die competitions Tabelle enthält lediglich die Kürzel und den ausgeschriebenen Titel von verschiedenen Ligen.

```
--Erstellen des Tables competitions
```

```
CREATE TABLE competitions(
    competition_id VARCHAR(10) PRIMARY KEY,
    competition_code VARCHAR(50)
);
```

Output			
#	Time	Action	Message
10	09:35:51	use scoutingandanalysis	0 row(s) affected
11	09:35:51	create table competitions(competition_id varchar(10) primary key, competition_code VARCHAR(50))	0 row(s) affected

Das SQL-Statement wurde erfolgreich ausgeführt.

### clubs

Neben ID, Code und vollem Namen des Clubs enthält der clubs Table eine Referenz zur Liga, die Anzahl Spieler total, das Durchschnittsalter, wie viele Spieler in der Nationalmannschaft spielen, Name des Heimstadions und Anzahl Plätze darin und in welchem Jahr der Club zuletzt aktiv war.

```
--Erstellen des Tables clubs
```

```
CREATE TABLE clubs (
    club_id INT PRIMARY KEY,
    club_code VARCHAR(50),
    club_name VARCHAR(100),
    domestic_competition_id VARCHAR(6),
    squad_size FLOAT,
    average_age FLOAT,
    national_team_players INTEGER,
    stadium_name VARCHAR(100),
    stadium_seats INTEGER,
    last_season year
);
```

```
1 16:01:44 CREATE TABLE clubs ( club_id INT PRIMARY KEY, club_code ... 0 row(s) affected 0.110 sec
```

Das SQL-Statement wurde erfolgreich ausgeführt.

## clubs Hilfstabellen

Für den clubs Table wurden 3 Hilfstabellen erstellt.

--Erstellen des Tables club_has_seats	--Erstellen des Tables club_has_stadium_name	--Erstellen des Tables club_has_seats
<pre>CREATE TABLE club_has_seats (     club_id INT,     stadium_seats int,     primary key (club_id, stadium_seats) );</pre>	<pre>CREATE TABLE club_has_stadium_name (     club_id INT,     stadium_name VARCHAR(50),     PRIMARY KEY (club_id, stadium_name) );</pre>	<pre>CREATE TABLE club_has_seats (     club_id INT,     stadium_seats INT,     PRIMARY KEY (club_id, stadium_seats) );</pre>

21 15:36:42 CREATE TABLE club_has_seats ( club_id INT, stadium_seats i... 0 row(s) affected	0.140 sec
18 15:32:15 CREATE TABLE club_has_stadium_name ( club_id INT, stadiu... 0 row(s) affected	0.109 sec
21 15:36:42 CREATE TABLE club_has_seats ( club_id INT, stadium_seats i... 0 row(s) affected	0.140 sec

Die SQL-Statements wurden erfolgreich ausgeführt.

### 3.2.1.1 Constraints

Nachdem alle Tables mir den korrekten Spalten erstellt wurden, wurden noch die Foreign Key Constraints erstellt.

```
ALTER TABLE scoutingandanalysis.club_has_domestic_competition
ADD CONSTRAINT fk_club_id_has_dc FOREIGN KEY (club_id) REFERENCES clubs(club_id);

ALTER TABLE scoutingandanalysis.club_has_domestic_competition
ADD CONSTRAINT fk_dc_id_has_dc FOREIGN KEY (domestic_competition_id) REFERENCES
competitions(competition_id);

ALTER TABLE scoutingandanalysis.club_has_seats
ADD CONSTRAINT fk_club_id_has_seats FOREIGN KEY (club_id) REFERENCES
clubs(club_id);

ALTER TABLE scoutingandanalysis.club_has_stadium_name
ADD CONSTRAINT fk_club_id_has_stadium_name FOREIGN KEY (club_id) REFERENCES
clubs(club_id);

ALTER TABLE scoutingandanalysis.game_events
ADD CONSTRAINT fk_game_events_game_id FOREIGN KEY (game_id) REFERENCES
games(game_id);

ALTER TABLE scoutingandanalysis.game_events
ADD CONSTRAINT fk_club_id_events_game_id FOREIGN KEY (club_id) REFERENCES
clubs(club_id);
```

```

ALTER TABLE scoutingandanalysis.game_events
ADD CONSTRAINT fk_player_game_events_id FOREIGN KEY (player_id) REFERENCES
players(player_id);

ALTER TABLE scoutingandanalysis.game_events
ADD CONSTRAINT fk_player_in_events_game_id FOREIGN KEY (player_id) REFERENCES
players(player_id);

ALTER TABLE scoutingandanalysis.game_events
ADD CONSTRAINT fk_player_assist_events_game_id FOREIGN KEY (player_id) REFERENCES
players(player_id);

ALTER TABLE scoutingandanalysis.game_events_has_date
ADD CONSTRAINT fk_game_event_id_game_event_has_date FOREIGN KEY (game_event_id)
REFERENCES game_events(game_event_id);

ALTER TABLE scoutingandanalysis.games
ADD CONSTRAINT fk_game_competition_id FOREIGN KEY (competition_id) REFERENCES
competitions(competition_id);

ALTER TABLE scoutingandanalysis.games
ADD CONSTRAINT fk_away_club_id FOREIGN KEY (away_club_id) REFERENCES
clubs(club_id);

ALTER TABLE scoutingandanalysis.games
ADD CONSTRAINT fk_host_club_id FOREIGN KEY (home_club_id) REFERENCES
clubs(club_id);

ALTER TABLE scoutingandanalysis.player_events
ADD CONSTRAINT fk_player_events_player_id FOREIGN KEY (player_id) REFERENCES
players(player_id);

ALTER TABLE scoutingandanalysis.player_events
ADD CONSTRAINT fk_player_events_game_id FOREIGN KEY (game_id) REFERENCES
games(game_id);

ALTER TABLE scoutingandanalysis.players
ADD CONSTRAINT fk_player_club_id FOREIGN KEY (fk_current_club_id) REFERENCES
clubs(club_id);

```

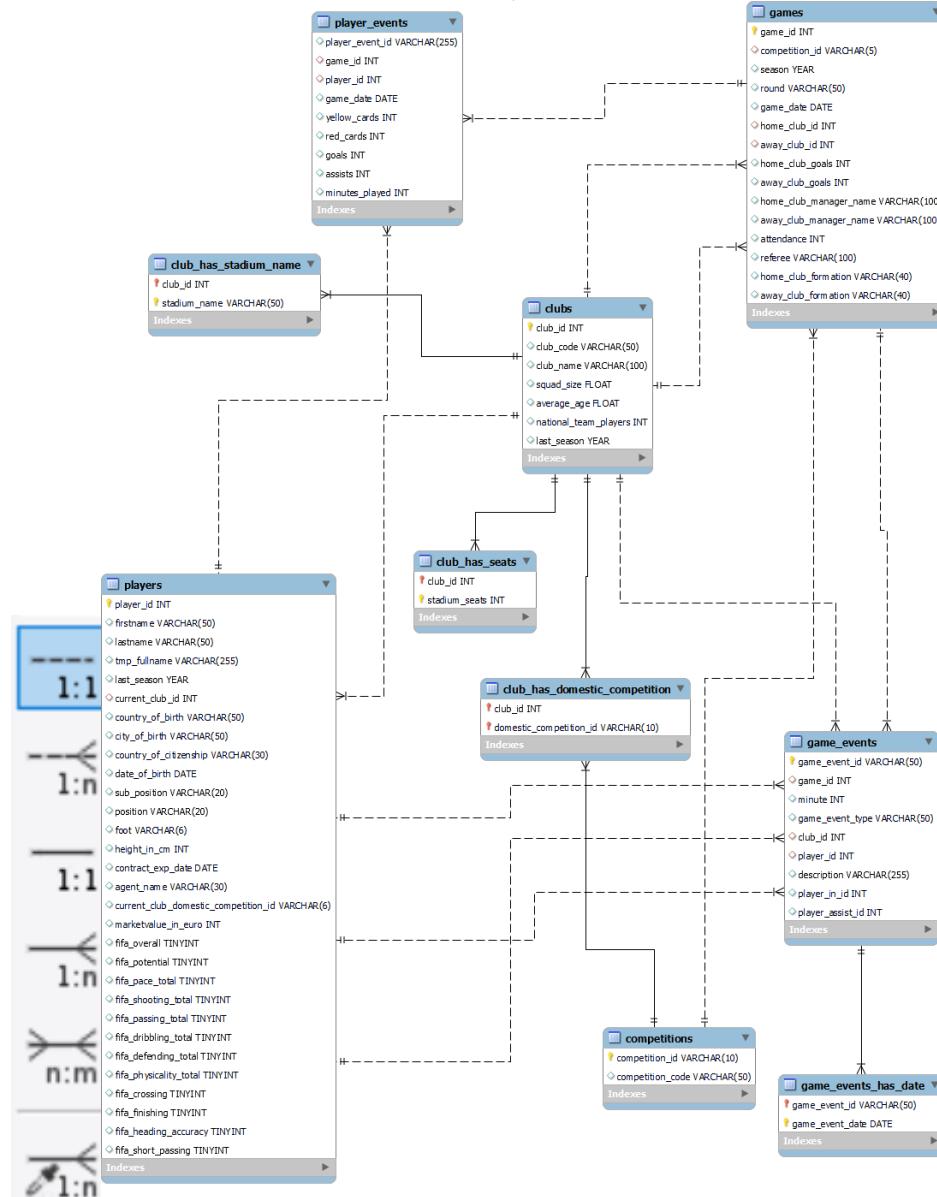
#	Time	Action	Message	Duration / Fetch
128	15:10:49	ALTER TABLE scoutinganalysis.club_has_domestic_competition ADD CONSTRAINT fk_club_id_has_dc FOREIGN KEY (club_id) REFERENCES club(id)	426 row(s) affected Records: 426 Duplicates: 0 Warnings: 0	1.485 sec
129	15:10:50	ALTER TABLE scoutinganalysis.club_has_domestic_competition ADD CONSTRAINT fk_dc_id_has_dc FOREIGN KEY (domestic_competition_id) REFERENCES domestic_competition(id)	426 row(s) affected Records: 426 Duplicates: 0 Warnings: 0	0.922 sec
130	15:10:51	ALTER TABLE scoutinganalysis.club_has_seats ADD CONSTRAINT fk_club_id_has_seats FOREIGN KEY (club_id) REFERENCES club(id)	426 row(s) affected Records: 426 Duplicates: 0 Warnings: 0	0.812 sec
131	15:10:52	ALTER TABLE scoutinganalysis.club_has_stadium_name ADD CONSTRAINT fk_club_id_has_stadium_name FOREIGN KEY (club_id) REFERENCES club(id)	426 row(s) affected Records: 426 Duplicates: 0 Warnings: 0	0.719 sec
132	15:10:53	ALTER TABLE scoutinganalysis.game_events ADD CONSTRAINT fk_game_events_id FOREIGN KEY (game_id) REFERENCES game(id)	192981 row(s) affected Records: 192981 Duplicates: 0 Warnings: 0	51.922 sec
133	15:11:45	ALTER TABLE scoutinganalysis.game_events ADD CONSTRAINT fk_club_id_events_game_id FOREIGN KEY (club_id) REFERENCES clubs(id)	192981 row(s) affected Records: 192981 Duplicates: 0 Warnings: 0	31.031 sec
134	15:12:16	ALTER TABLE scoutinganalysis.game_events ADD CONSTRAINT fk_player_game_events_id FOREIGN KEY (player_id) REFERENCES player(id)	192981 row(s) affected Records: 192981 Duplicates: 0 Warnings: 0	30.437 sec
135	15:12:46	ALTER TABLE scoutinganalysis.game_events ADD CONSTRAINT fk_player_in_events_game_id FOREIGN KEY (player_id) REFERENCES player(id)	192981 row(s) affected Records: 192981 Duplicates: 0 Warnings: 0	36.813 sec
136	15:13:23	ALTER TABLE scoutinganalysis.game_events ADD CONSTRAINT fk_player_assist_events_game_id FOREIGN KEY (player_id) REFERENCES player(id)	192981 row(s) affected Records: 192981 Duplicates: 0 Warnings: 0	37.812 sec
137	15:14:01	ALTER TABLE scoutinganalysis.game_events_has_date ADD CONSTRAINT fk_game_event_has_date FOREIGN KEY (game_event_id) REFERENCES game_events(id)	192981 row(s) affected Records: 192981 Duplicates: 0 Warnings: 0	12.141 sec
138	15:14:13	ALTER TABLE scoutinganalysis.games ADD CONSTRAINT fk_game_competition_id FOREIGN KEY (competition_id) REFERENCES competitions(id)	17458 row(s) affected Records: 17458 Duplicates: 0 Warnings: 0	8.625 sec
139	15:14:21	ALTER TABLE scoutinganalysis.games ADD CONSTRAINT fk_away_club_id FOREIGN KEY (away_club_id) REFERENCES clubs(club_id)	17458 row(s) affected Records: 17458 Duplicates: 0 Warnings: 0	7.610 sec
140	15:14:29	ALTER TABLE scoutinganalysis.games ADD CONSTRAINT fk_host_club_id FOREIGN KEY (home_club_id) REFERENCES clubs(club_id)	17458 row(s) affected Records: 17458 Duplicates: 0 Warnings: 0	3.484 sec
141	15:14:32	ALTER TABLE scoutinganalysis.player_events ADD CONSTRAINT fk_player_events_player_id FOREIGN KEY (player_id) REFERENCES player(id)	474725 row(s) affected Records: 474725 Duplicates: 0 Warnings: 0	60.344 sec
142	15:15:33	ALTER TABLE scoutinganalysis.player_events ADD CONSTRAINT fk_player_events_game_id FOREIGN KEY (game_id) REFERENCES game(id)	474725 row(s) affected Records: 474725 Duplicates: 0 Warnings: 0	35.187 sec
143	15:16:08	ALTER TABLE scoutinganalysis.players ADD CONSTRAINT fk_player_club_id FOREIGN KEY (current_club_id) REFERENCES clubs(club_id)	15348 row(s) affected Records: 15348 Duplicates: 0 Warnings: 0	5.625 sec

Die Constraints wurden erfolgreich hinzugefügt.

### 3.2.2 Datenbankschema

Die nachfolgende Grafik zeigt nun das spezifisch in unserer Datenbank verwendete Schema. Es lässt sich direkt in MySQL generieren und zeigt die relations zwischen den Entitätsmengen an. Man kann das Datenbankschema in der workbench über den Reiter Database->Reverse Engineer selbst erstellen. Zudem werden hier auch die genauen Attribute mit Datentypen beschrieben. In mysql selbst lassen sich dann auch die Indizes auf den spezifischen tables anzeigen. Eine Legende zu den Verbindungen lässt sich ebenfalls finden. Diese korrespondieren zu den Kardinalitäten in unserem konzeptionellen Schema. Das

Schema ist also noch etwas ausführlicher, aber ansonsten identisch zu unserem konzeptionellen Schema.



### 3.3 Normalisierung SQL

Durch die Normalisierung der SQL-Umgebung entstanden vier neue Tables. In den ursprünglichen Daten sind keine Many zu Many Beziehung vorhanden, weshalb wir diese nicht normalisieren mussten.

#### 3.3.1 club\_has\_stadium\_name

In der clubs Tabelle war ursprünglich das Heimstadion abgespeichert. Da dies aber für zwei Clubs das gleiche sein könnte wurde der Stadium Name mit Referenz auf die Club ID Ausgelagert.

#### 3.3.2 club\_has\_seats

Die Platzanzahl der Heimstadien wurde aus dem gleichen Grund wie Namen ausgelagert.

#### 3.3.3 club\_has\_domestic\_competition

Für die Referenz von Clubs auf eine Domestic Competition wurde eine Hilfstabelle erstellt. Rückblickend stellte sich heraus, dass diese Normalisierung gar nicht nötig war. Die ursprüngliche Überlegung war hier,

dass ein Club in einer Landesliga und in einer Internationalen Liga spielen könnte. Dies ist aber gar nicht der Fall in den Quelldaten.

### 3.3.4 game\_event\_has\_date

Da eine Game Entität bereits ein Datum hat ist es logisch, dass alle Game Events, welche in diesem Game stattfanden, dieses Datum teilen. Die Spalte hätte daher auch gelöscht werden können, aber wir entschieden uns die Daten einfach auszulagern, da sie gewisse Selektionen vereinfachen könnten.

## 3.4 In Mongo DB (JSON-Prototypen)

Ähnlich wie beim SQL-Ansatz werden für die Daten geeignete Behälter benötigt. Im NoSQL Ansatz sind dies drei sogenannte Documents welche die Daten enthalten. Nachfolgend werden die drei JSON-Prototypen (quasi Vorlagen) dieser drei Documents aufgezeigt. Wir haben drei Dokumente gewählt, da wir uns für mögliche, konkrete Querys nicht bereits im Voraus beschränken wollten. Die drei Ausgangspunkte für Abfragen, die uns interessieren könnten, gehen aber immer von einem Spieler, einem Club oder einem Game aus. Deswegen wurden die Dokumente auch auf dem root Level mit diesen Informationen bestückt. Also beispielsweise Informationen zum Spieler im Players Document. GamesDocument und ClubsDocument beinhalten dann die Informationen zu den Games respektive Clubs auf dem root-Level. Weitere Informationen, die im Bezug zum root-Level auch interessant sind, wurden dann ebenfalls im Schema aufgenommen, sodass dann bei Abfragen der direkte Zugriff auf diese auch möglich ist. Mit diesem Ansatz können ressourcenintensive lookups vermieden werden. Hier wird auch klar der Unterschied zwischen MongoDB und SQL erkennbar. In MongoDB packt man die relevanten Daten besser in ein einziges Dokument, während in SQL diese Informationen eher in zwei verschiedenen tables stehen. Dies wird aber im Denormalisierung JSON-Kapitel genauer beschrieben.

### 3.4.1.1 Players Document

```
{  
    "_id": "ObjectId",  
    "player_id": "Number",  
    "firstname": "String",  
    "lastname": "String",  
    "tmp_fullname": "String",  
    "last_season": "Number",  
    "current_club_id": "Number",  
    "country_of_birth": "String",  
    "city_of_birth": "String",  
    "country_of_citizenship": "String",  
    "date_of_birth": "ISODate",  
    "sub_position": "String",  
    "position": "String",  
    "foot": "String",  
    "height_in_cm": "Number",  
    "contract_exp_date": "ISODate or null",  
    "agent_name": "String",  
    "current_club_domestic_competition_id": "String",  
    "marketvalue_in_euro": "Number",  
    "fifa_overall": "Number",  
    "fifa_potential": "Number",  
    "fifa_pace_total": "Number",  
    "fifa_shooting_total": "Number",
```

```

"fifa_passing_total": "Number",
"fifa_dribbling_total": "Number",
"fifa_defending_total": "Number",
"fifa_physicality_total": "Number",
"fifa_crossing": "Number",
"fifa_finishing": "Number",
"fifa_heading_accuracy": "Number",
"fifa_short_passing": "Number",
"Player_Events": [
  {
    "_id": "ObjectId",
    "player_event_id": "Number",
    "game_id": "Number",
    "player_id": "Number",
    "game_date": "ISODate",
    "yellow_cards": "Number",
    "red_cards": "Number",
    "goals": "Number",
    "assists": "Number",
    "minutes_played": "Number"
  }
],
"Clubs": [
  {
    "_id": "ObjectId",
    "club_id": "Number",
    "club_code": "String",
    "domestic_competition_id": "String",
    "squad_size": "Number",
    "average_age": "Number",
    "national_team_players": "Number",
    "stadium_name": "String",
    "stadium_seats": "Number",
    "last_season": "Number",
    "club_name": "String"
  }
]
}

```

### 3.4.1.2 Games Document

```
{
  "_id": "ObjectId",
  "game_id": "Number",
  "competition_id": "String",
  "season": "Number",
  "round": "String",
}
```

```

"game_date": "ISODate",
"home_club_id": "Number",
"away_club_id": "Number",
"home_club_goals": "Number",
"away_club_goals": "Number",
"home_club_manager_name": "String",
"away_club_manager_name": "String",
"attendance": "Number or null",
"referee": "String",
"home_club_formation": "String",
"away_club_formation": "String",
"Game_Events": [
  {
    "_id": "ObjectId",
    "game_event_id": "String",
    "game_id": "Number",
    "minute": "Number",
    "game_event_type": "String",
    "club_id": "Number",
    "player_id": "Number",
    "description": "String",
    "player_in_id": "Number",
    "player_assist_id": "Number or null",
    "game_event_date": "ISODate"
  }
],
"Competition": [
  {
    "_id": "ObjectId",
    "competition_id": "String",
    "competition_code": "String"
  }
],
"Home_Club": [
  {
    "_id": "ObjectId",
    "club_id": "Number",
    "club_code": "String",
    "domestic_competition_id": "String",
    "squad_size": "Number",
    "average_age": "Number",
    "national_team_players": "Number",
    "stadium_name": "String",
    "stadium_seats": "Number",
    "last_season": "Number",
    "club_name": "String"
  }
]

```

```

],
"Away_Club": [
{
  "_id": "ObjectId",
  "club_id": "Number",
  "club_code": "String",
  "domestic_competition_id": "String",
  "squad_size": "Number",
  "average_age": "Number",
  "national_team_players": "Number",
  "stadium_name": "String",
  "stadium_seats": "Number",
  "last_season": "Number",
  "club_name": "String"
}
]
}

```

#### 3.4.1.3 Clubs Document

```

{
  "club_id": "Number",
  "club_code": "String",
  "domestic_competition_id": "String",
  "squad_size": "Number",
  "average_age": "Number",
  "national_team_players": "Number",
  "stadium_name": "String",
  "stadium_seats": "Number",
  "last_season": "Number",
  "club_name": "String",
  "Players": [
    {
      "_id": "ObjectId",
      "player_id": "Number",
      "firstname": "String",
      "lastname": "String",
      "tmp_fullname": "String",
      "last_season": "Number",
      "current_club_id": "Number",
      "country_of_birth": "String",
      "city_of_birth": "String",
      "country_of_citizenship": "String",
      "date_of_birth": "Date",
      "sub_position": "String",
      "position": "String",
      "foot": "String",
    }
  ]
}

```

```

    "height_in_cm": "Number",
    "contract_exp_date": "Date",
    "agent_name": "String",
    "current_club_domestic_competition_id": "String",
    "marketvalue_in_euro": "Number",
    "fifa_overall": "Number",
    "fifa_potential": "Number",
    "fifa_pace_total": "Number",
    "fifa_shooting_total": "Number",
    "fifa_passing_total": "Number",
    "fifa_dribbling_total": "Number",
    "fifa_defending_total": "Number",
    "fifa_physicality_total": "Number",
    "fifa_crossing": "Number",
    "fifa_finishing": "Number",
    "fifa_heading_accuracy": "Number",
    "fifa_short_passing": "Number"
  }
],
"Games": [
{
  "_id": "ObjectId",
  "game_id": "Number",
  "competition_id": "String",
  "season": "Number",
  "round": "String",
  "game_date": "Date",
  "home_club_id": "Number",
  "away_club_id": "Number",
  "home_club_goals": "Number",
  "away_club_goals": "Number",
  "home_club_manager_name": "String",
  "away_club_manager_name": "String",
  "attendance": "Number",
  "referee": "String",
  "home_club_formation": "String",
  "away_club_formation": "String"
}
],
"Competition": [
{
  "_id": "Object ID",
  "competition_id": "String",
  "competition_code": "String"
}
]

```

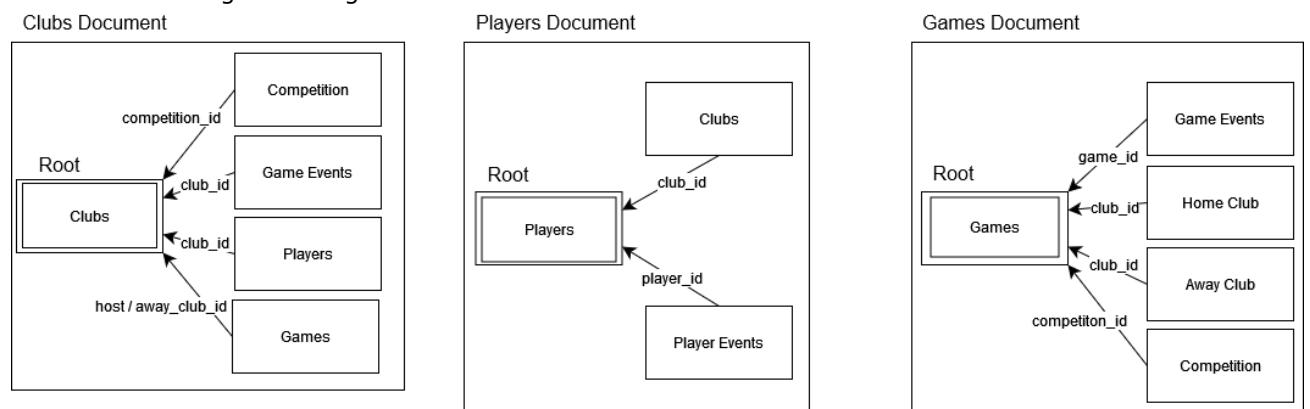
```

],
"Game_Events": [
{
  "_id": "ObjectId",
  "game_event_id": "String",
  "game_id": "Number",
  "minute": "Number",
  "game_event_type": "String",
  "club_id": "Number",
  "player_id": "Number",
  "description": "String",
  "player_in_id": "Number",
  "player_assist_id": "Number",
  "game_event_date": "Date"
}
]
}

```

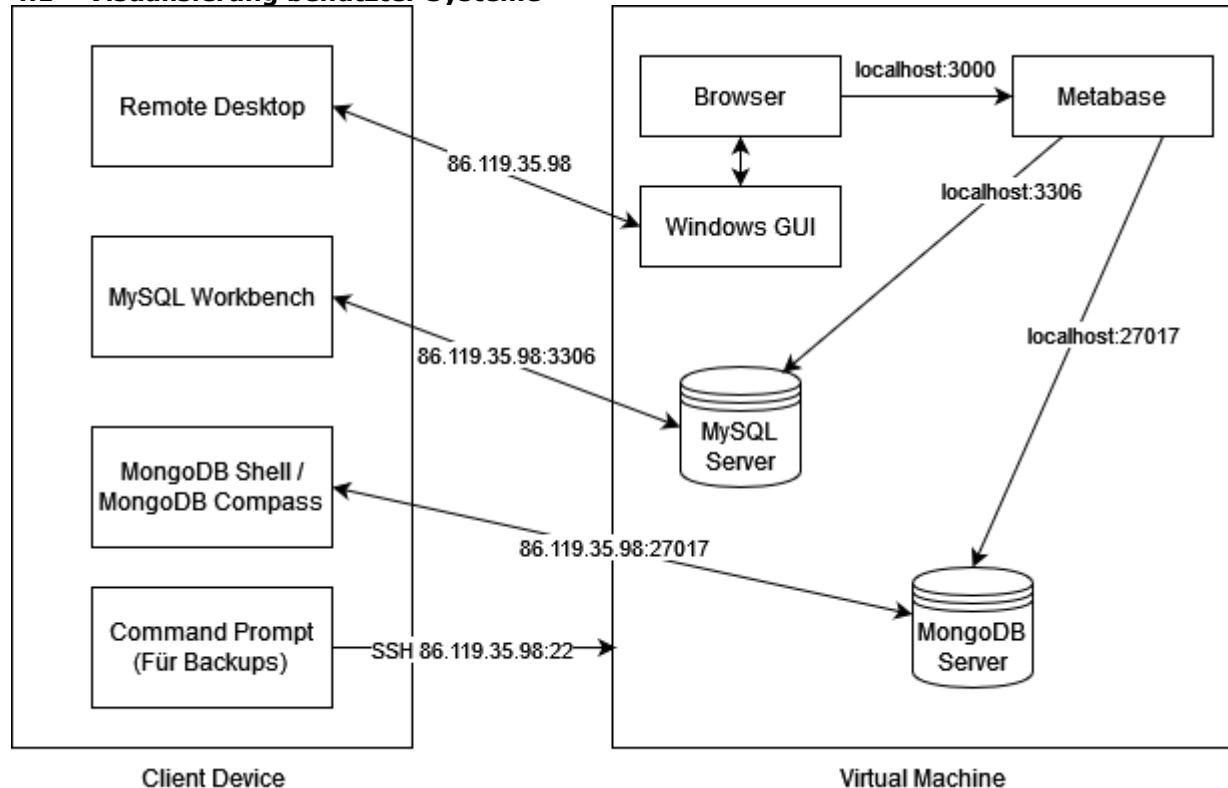
### 3.5 Denormalisierung JSON

Im Gegensatz zu den normalisierten Beziehungen in SQL sind die Beziehungen in MongoDB aggregiert worden. Die End-Dokumente beinhalten die Daten, mithilfe verschiedener Objekte. Ein Objekt ist dabei ähnlich wie eine SQL-Zeile. Der Unterschied zu SQL liegt aber darin, dass in einem Objekt, mehrere andere "tables" verpackt werden können. Ein Beispiel ist ein Objekt im Dokument PlayersDocument, welches die "tables" PlayerEvents und Clubs, bereits in sich enthält. Die Informationen sind also sprichwörtlich NICHT ausgelagert, sondern eingelagert. Nachfolgend werden die Abhängigkeiten auch noch als ERD-Diagramm abgebildet.



## 4 Daten laden & transformieren

### 4.1 Visualisierung benutzter Systeme



Vom Client Device aus kann per Remote Desktop, MySQL Workbench, MongoDB Shell oder MongoDB Compass auf die Virtuelle Maschine zugegriffen werden. Mittels MySQL Workbench können Änderungen an der MySQL Datenbank vorgenommen und Daten selektiert werden. Gleichtes gilt für die MongoDB Shell / Compass und die MongoDB. Zudem kann auch vom backup Client mit SSH auf die VM zugegriffen werden, um das Backup auf dem Client zu transferieren.

Um auf das Metabase, welches auf der Virtuellen Maschine läuft, zuzugreifen, muss per Remote Desktop auf der VM ein Browser gestartet werden. In diesem können dann über die Adresse localhost:3000 die Daten aus den beiden Datenbanken angezeigt werden.

### 4.2 Logindaten

Was	Benutzername	Passwort
VM (dbs-21, 86.119.35.98)	dbsstudent	5Jv6AEwXqNtVih
MySQL Datenbank	admin	HPgr_9\$5
	gui	f13SF1k_\$
	developer	Hb7890:,
MongoDB Datenbank	myAdmin	w-V}*yZ.kc&dSUW98Q,L/#
	developer	X8z3\$'KU>Q
	guiUser	h<sGL3ZF>:#z2VN%
Metabase	dominik.arnet@stud.hslu.ch	si*8N#EsU~*7kv{}

ConnectionString MongoDB admin	mongosh "mongodb://myAdmin:w-V%7D*yZ.kc%26dSUW98Q%2CL%2F%23@86.119.35.98:27017/admin"
--------------------------------	---

<b>ConnectionString MongoDB developer</b>	mongosh "mongodb://developer:X8z3%24%27KU%3EQ@86.119.35.98:27017/admin"
<b>ConnectionString MongoDB gui</b>	Mongosh "mongodb://guiUser:h%3CsGL3ZF%3E%3A%23z2VN%25@86.119.35.98:27017/scoutingAndAnalysis"

### 4.3 Vorgehen Laden der Daten

Um das Laden der nicht unnötig rechenintensiv zu gestalten, wurden vor dem Laden der Daten in die SQL-Umgebung alle Foreign Key Constraints gelöscht. Dies verhindert unnötige Checks zwischen den Tables während dem Laden der Daten.

Disclaimer: Um nicht alle Spaltennamen immer mühselig auszuschreiben wurde beim Erstellen der Load Scripts ChatGPT zur Hilfe gezogen.

#### 4.3.1 Load SQL

##### players

Mittels dem SET GLOBAL local\_infile=TRUE; Statement wird MySQL so konfiguriert, dass es erlaubt Daten aus CSV-Dateien aus einem Client Gerät auf den Server in der VM zu laden. Mit dem LOAD DATA LOCAL INFILE wird der Pfad definiert, in den sich das CSV befindet. Mithilfe dieses Befehls werden die Daten aus dem CSV in die MySQL Umgebung geladen. INTO TABLE definiert die Ziel Tabelle. Weitere Befehle wie FIELDS TERMINATED BY und OPTIONALLY ENCLOSED BY befassen sich mit der Struktur des CSVs.

IGNORE 1 LINES sorgt dafür, dass der Header des CSVs nicht in der DB landen. Diese Befehlsstruktur ist für alle LOAD-Befehle dieselbe. Die Spalten des CSVs werden anschliessend mit führendem @ definiert.

Das Laden der Daten in die players Tabelle musste in zwei Schritten erfolgen, da die Daten darin aus zwei verschiedenen Quellen stammten. In einem ersten Schritt wurden die Daten aus dem CSV des Transfermarktes in die Tabelle geladen. Mittels dem SET-Keyword konnten die Spalten aus dem CSV auf die Spalten der Tabelle gelinkt werden, auch wenn die Namen nicht identisch sind. Im gleichen Schritt können leere Felder mit NULL ersetzt werden und Datumswerte richtig gecastet werden.

```
-- Laden der Daten in den Table players

SET GLOBAL local_infile = TRUE;

LOAD DATA LOCAL INFILE 'C:/Informatik Studium/Sem 2. FS24/01 DBS/data/players.csv'
INTO TABLE players
FIELDS TERMINATED BY ','
optionally enclosed by ''
IGNORE 1 LINES
(@player_id,@first_name,@last_name,@`name`,@last_season,@current_club_id,@player_code,@country_of_birth,@city_of_birth,@country_of_citizenship,@date_of_birth,@sub_position,@position,@foot,@height_in_cm,@contract_expiration_date,@agent_name,@image_url,@url,@current_club Domestic_competition_id,@current_club_name,@market_value_in_euro,@highest_market_value_in_euro)

SET player_id = NULLIF(@player_id, ''),
firstname = NULLIF(@first_name, ''),
lastname = NULLIF(@last_name, ''),
tmp_fullname = NULLIF(@`name`, ''),
```

```

last_season = NULLIF(@last_season, ''),
current_club_id = NULLIF(@current_club_id, ''),
country_of_birth = NULLIF(@country_of_birth, ''),
city_of_birth = NULLIF(@city_of_birth, ''),
country_of_citizenship = NULLIF(@country_of_citizenship, ''),
date_of_birth = NULLIF(@date_of_birth, ''),
sub_position = NULLIF(@sub_position, ''),
position = NULLIF(@position, ''),
foot = NULLIF(@foot, ''),
height_in_cm = NULLIF(@height_in_cm, ''),
contract_exp_date = CAST(NULLIF(@contract_exp_date, '') AS DATE),
agent_name = NULLIF(@agent_name, ''),
current_club_domestic_competition_id =
NULLIF(@current_club_domestic_competition_id, ''),
marketvalue_in_euro = NULLIF(@market_value_in_euro, ''));
```

48 10:33:05 SET GLOBAL local\_infile = TRUE  
 49 10:33:05 LOAD DATA LOCAL INFILE 'C:/Informatik Studium/Sem 2. FS24/01 DBS/data/players.csv' INTO TABLE players FIELDS TERMINATED BY ',' o... 30505 row(s) affected Records: 30505 Deleted: 0 Skipped: 0 Warnings: 0

0.000 sec  
 2.844 sec

30505 Einträge wurden in 2.844s erfolgreich geladen. Wie sich später herausstellte wurde das contract\_exp\_date nicht in die SQL umgebung geladen, obwohl keine Warnings oder Errors angezeigt wurden. Wir beschloss dieses Feld so zu belassen, da es keine tragende Rolle für die Use Cases darstellt.

### **tmp\_fifa\_ratings**

Das CSV der FIFA-Ratings, welche auch in die players Table sollen, wurde zuerst in eine temporäre Tabelle geladen. Dies ermöglichte uns mittels einem UPDATE die Daten korrekt in die players Tabelle zu übertragen. Dies wird im Nächsten Kapitel genauer erklärt. Wieder wurde das SET-Keyword verwendet, um die Spalten des CSV und der Tabellen zu matchen und allfällige NULL-Werte zu setzen. Da der Marktwert (value\_in\_euro) schon im ersten Datenset enthalten ist, wird dieser gar nicht erst in den temporären Table geladen.

```
-- Laden der Daten in den Table tmp_fifa_ratings

SET GLOBAL local_infile = TRUE;

LOAD DATA LOCAL INFILE 'C:/Informatik Studium/Sem 2. FS24/01
DBS/data/Fifa_23_Players_Data.csv'
INTO TABLE tmp_fifa_ratings
FIELDS TERMINATED BY ','
IGNORE 1 LINES
(@full_name,@overall,@potential,@value_in_euro,@pace_total,@shooting_total,@passing_total,@dribbling_total,@defending_total,@physicality_total,@crossing,@finishing,
@heading_accuracy,@short_passing)

SET full_name = NULLIF(@full_name, ''),
overall = NULLIF(@overall, ''),
potential = NULLIF(@potential, ''),
pace_total = NULLIF(@pace_total, ''),
shooting_total = NULLIF(@shooting_total, ''),
passing_total = NULLIF(@passing_total, ''),
dribbling_total = NULLIF(@dribbling_total, ''),
```

```

defending_total = NULLIF(@defending_total, ''),
physicality_total = NULLIF(@physicality_total, ''),
crossing = NULLIF(@crossing, ''),
finishing = NULLIF(@finishing, ''),
heading_accuracy = NULLIF(@heading_accuracy, ''),
short_passing = NULLIF(@short_passing, '');

```

25 10:29:58 SET GLOBAL local\_infile = TRUE 0 row(s) affected 0.078 sec  
26 10:29:58 LOAD DATA LOCAL INFILE 'C:/Informatik Studium/Sem 2. FS24/01 DBS/data/Fifa\_23\_Players\_Data.csv' INTO TABLE tmp\_fifa\_ratings FIELDS TER... 17529 row(s) affected Records: 17529 Deleted: 0 Skipped: 0 Warnings: 0 2.172 sec

Die 17529 Einträge wurden erfolgreich in 2.172s in die SQL Umgebung geladen.

### player\_events

Für die player\_events wurde wieder SET verwendet, um Column-matching und Casting zu machen beim Load.

```
-- Laden der Daten in den Table player_events
```

```

SET GLOBAL local_infile = true;

load data local infile 'C:/Informatik Studium/Sem 2. FS24/01
DBS/data/appearances.csv'
INTO TABLE player_events
FIELDS TERMINATED BY ','
ENCLOSED BY ""
LINES TERMINATED BY '\n'
IGNORE 1 LINES
(@appearance_id, game_id, player_id, @player_club_id, @player_current_club_id,
@`date`, @player_name, @competition_id, yellow_cards, red_cards, goals, assists,
minutes_played)
SET player_event_id = NULLIF(@appearance_id, ''),
game_id = NULLIF(game_id, ''),
player_id = NULLIF(player_id, ''),
game_date = STR_TO_DATE(@`date`, '%Y-%m-%d'),
yellow_cards = yellow_cards,
red_cards = red_cards,
goals = goals,
assists = assists,
minutes_played = minutes_played;

```

47 14:46:29 SET GLOBAL local\_infile = true 0 row(s) affected 0.000 sec  
48 14:46:29 load data local infile 'C:/Informatik Studium/Sem 2. FS24/01 DBS/data/appear... 1563583 row(s) affected Records: 1563583 Deleted: 0 Skipped: 0 Warnings: 0 44.750 sec

Dies Load dauerte ganze 44.75s da es sich um 1'563'583 Einträge handelt.

### games

```
-- Laden der Daten in den Table games
```

```
SET GLOBAL local_infile = true;
```

```

load data local infile 'C:/Informatik Studium/Sem 2. FS24/01 DBS/data/games.csv'
into table scoutingandanalysis.games
fields terminated by ','

```

```

enclosed by """
ignore 1 lines
(@game_id, @competition_id, @season, @round, @game_date, @home_club_id,
@away_club_id, @home_club_goals, @away_club_goals, @home_club_position,
@away_club_position, @home_club_manager_name, @away_club_manager_name, @stadium,
@attendance, @referee, @url, @home_club_formation, @away_club_formation,
@home_club_name, @away_club_name, @aggregate, @competition_type)
SET game_id = NULLIF(@game_id, ''),
competition_id = NULLIF(@competition_id, ''),
season = NULLIF(@season, ''),
round = NULLIF(@round, ''),
game_date = NULLIF(@game_date, ''),
home_club_id = NULLIF(@home_club_id, ''),
away_club_id = NULLIF(@away_club_id, ''),
home_club_goals = NULLIF(@home_club_goals, ''),
away_club_goals = NULLIF(@away_club_goals, ''),
home_club_manager_name = NULLIF(@home_club_manager_name, ''),
away_club_manager_name = NULLIF(@away_club_manager_name, ''),
stadium = NULLIF(@stadium, ''),
attendance = NULLIF(@attendance, ''),
referee = NULLIF(@referee, ''),
home_club_formation = NULLIF(@home_club_formation, ''),
away_club_formation = NULLIF(@away_club_formation, ');

```

44	11:00:04	SET GLOBAL local_infile = true	0 row(s) affected	0.015 sec
45	11:00:04	load data local infile 'C:/Informatik/Studium/Sem 2. FS24/01 DBS/data/games.csv' in...	67648 row(s) affected Records: 67648 Deleted: 0 Skipped: 0 Warnings: 0	6.016 sec

Die 67648 Einträge wurden in 6.016s erfolgreich in die games Tabelle geladen.

### game\_events

```

-- Laden der Daten in den Table game_events

SET GLOBAL local_infile = true;

LOAD DATA LOCAL INFILE
'C:/Users/fabia/Documents/HSLU/Semester2/DB_Material/Football_Data_from_Transfermarkt/kaggle_data/game_events.csv'
INTO TABLE scoutingandanalysis.game_events
FIELDS TERMINATED BY ';'
OPTIONALLY ENCLOSED BY ''
IGNORE 1 LINES
(@game_event_id, @game_event_date, @game_id, @`minute`, @game_event_type,
@club_id, @player_id, @player_in_id, @player_assist_id, @`description`)

SET game_event_id = NULLIF(@game_event_id, ''),
game_event_date = NULLIF(@game_event_date, ''),
game_id = NULLIF(@game_id, ''),
`minute` = NULLIF(@`minute`, ''),
game_event_type = NULLIF(@game_event_type, ''),

```

```

club_id = NULLIF(@club_id, ''),
player_id = NULLIF(@player_id, ''),
`description` = NULLIF(@`description`, ''),
player_in_id = NULLIF(@player_in_id, ''),
player_assist_id = NULLIF(@player_assist_id, '');

```

⚠ 4 17:06:05 LOAD DATA LOCAL INFILE 'C:/Users/fabia/Documents/HSLU/Semester2/DB\_Material/Football\_Data\_from\_Transfermarkt/kaggle\_dat...' 114.250 sec

Dieser Load Befehl schloss mit Warnings ab nach 114.25s. Dies lag an der Formatierung des game\_date in den Quelldaten. Unten sind die Warnings welche zurückgegeben wurden ersichtlich.

```

4095 Delimiter ',' in position 2 in datetime value '18.08.2012' at row 93 is deprecated. Prefer the standard '-'.
1265 Data truncated for column 'game_event_date' at row 93
4095 Delimiter ',' in position 2 in datetime value '18.08.2012' at row 94 is deprecated. Prefer the standard '-'.
1265 Data truncated for column 'game_event_date' at row 94
4095 Delimiter ',' in position 2 in datetime value '18.08.2012' at row 95 is deprecated. Prefer the standard '-'.
1265 Data truncated for column 'game_event_date' at row 95
4095 Delimiter ',' in position 2 in datetime value '18.08.2012' at row 96 is deprecated. Prefer the standard '-'.
1265 Data truncated for column 'game_event_date' at row 96
4095 Delimiter ',' in position 2 in datetime value '18.08.2012' at row 97 is deprecated. Prefer the standard '-'.
1265 Data truncated for column 'game_event_date' at row 97
4095 Delimiter ',' in position 2 in datetime value '18.08.2012' at row 98 is deprecated. Prefer the standard '-'.
1265 Data truncated for column 'game_event_date' at row 98
4095 Delimiter ',' in position 2 in datetime value '18.08.2012' at row 99 is deprecated. Prefer the standard '-'.
1265 Data truncated for column 'game_event_date' at row 99
4095 Delimiter ',' in position 2 in datetime value '18.08.2012' at row 100 is deprecated. Prefer the standard '-'.
1265 Data truncated for column 'game_event_date' at row 100
4095 Delimiter ',' in position 2 in datetime value '18.08.2012' at row 101 is deprecated. Prefer the standard '-'.
1265 Data truncated for column 'game_event_date' at row 101
4095 Delimiter ',' in position 2 in datetime value '18.08.2012' at row 102 is deprecated. Prefer the standard '-'.
1265 Data truncated for column 'game_event_date' at row 102
4095 Delimiter ',' in position 2 in datetime value '18.08.2012' at row 103 is deprecated. Prefer the standard '-'.

```

Die Daten wurden aber trotzdem korrekt geladen, und daher waren keine weiteren Massnahmen nötig.

## competitions

Für den competitions Table war die SET-Syntax nicht nötig, da die Spalten im CSV und im SQL übereinstimmten.

```

-- Laden der Daten in den Table competitions

SET GLOBAL local_infile = true;

load data local infile 'C:/Informatik Studium/Sem 2. FS24/01 DBS/Football Data
from Transfermarkt/competitions.csv'
into table competitions
fields terminated by ','
ignore 1 lines;

```

⌚ 12 09:35:51 SET GLOBAL local\_infile = true 0 row(s) affected 0.015 sec  
⚠ 13 09:35:51 load data local infile 'C:/Informatik Studium/Sem 2. FS24/01 DBS/Football D... 43 row(s) affected, 43 warning(s): 1262 Row 1 was truncated; it contained m... 0.016 sec

Es handelte sich bei den competitions nur um 43 Einträge. Sie wurden mit Warnings geladen, da im CSV mehr Columns als in der Tabelle vorhanden waren. Diese Warnings können daher ignoriert werden.

## clubs

Für den clubs Table wurde wieder auf das SET-Keyword zurückgegriffen, aus den gleichen Gründen wie bei den Vorherigen Tabellen.

```

-- Laden der Daten in den Table clubs

SET GLOBAL local_infile = true;

```

```

LOAD DATA LOCAL INFILE
'C:/Users/fabia/Documents/HSLU/Semester2/DB_Material/Football_Data_from_Transfermarkt/kaggle_data/clubs.csv'
INTO TABLE scoutingandanalysis.clubs
FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY "'"
IGNORE 1 LINES
(@club_id, @club_code, @`name`, @domestic_competition_id, @total_market_value,
@squad_size, @average_age, @foreigners_number, @foreigners_percentage,
@national_team_players, @stadium_name, @stadium_seats, @net_transfer_record,
@coach_name, @last_season, @filename, @url)

SET club_id = NULLIF(@club_id, ''),
club_code = NULLIF(@club_code, ''),
club_name = NULLIF(@`name`, ''),
domestic_competition_id = NULLIF(@domestic_competition_id, ''),
squad_size = NULLIF(@squad_size, ''),
average_age = NULLIF(@average_age, ''),
national_team_players = NULLIF(@national_team_players, ''),
stadium_name = NULLIF(@stadium_name, ''),
stadium_seats = NULLIF(@stadium_seats, ''),
last_season = NULLIF(@last_season, '');

```

1 16:32:22 LOAD DATA LOCAL INFILE 'C:/Users/fabia/Documents/HSLU/Sem... 426 row(s) affected Records: 426 Deleted: 0 Skipped: 0 Warnings: 0 0.078 sec

Die 426 Einträge wurden ohne Warnings in die SQL-Umgebung geladen.

#### 4.3.2 Transform SQL

##### **players & tmp\_fifa\_ratings**

Um die FIFA-Ratings den players hinzuzufügen, wurden zuerst nicht aktuelle Spieler gelöscht. Diese Löschung betrifft alle Spieler, welche ihre letzte Saison vor 2020 hatten. Insgesamt wurden so 15157 Einträge gelöscht und es blieben 15348 aktuelle Einträge übrig. Danach wurde die players table geupdated mit den FIFA-Ratings über einen Join mit dem vollen Namen. Hier ist kein Insert oder Load Anweisung mehr nötig, da schon beide Datensätze in jeweils einer Tabelle in MySQL sind. Von den 15348 noch vorhandenen Spielern wurden nun bei 5996 die Fifa Ratings hinzugefügt. Dies entspricht ca. 40% aller Spieler und ist somit eine vertretbare Grösse.

```

-- FIFA Rating zu players hinzufügen

delete from players where last_season < 2020;

update players p
join tmp_fifa_ratings t on p.tmp_fullname = t.full_name
SET p.fifa_overall = t.overall, p.fifa_potential = t.potential, p.fifa_pace_total =
t.pace_total, p.fifa_shooting_total = t.shooting_total, p.fifa_passing_total =
t.passing_total, p.fifa_dribbling_total = t.dribbling_total,
p.fifa_defending_total = t.defending_total, p.fifa_physicality_total =
t.physicality_total, p.fifa_crossing = t.crossing, p.fifa_finishing = t.finishing,
```

```
p.fifa_heading_accuracy = t.heading_accuracy, p.fifa_short_passing =  
t.short_passing
```

```
50 10:33:08 delete from players where last_season < 2020 15157 row(s) affected 0.625 sec  
51 10:33:09 update players p join tmp_fifa_ratings t on p.tmp_fulname = t.full_name SET p.fifa_overall = t.overall, p.fifa_potential = t.potential, p.fifa_pace_total ... 5996 row(s) affected Rows matched: 5996 Changed: 5996 Warnings: 0 458.843 sec
```

Die beiden SQL-Statements wurden ohne Warnungen ausgeführt. Das Update dauerte jedoch ca. 459s (7.65 min) da für jeden der 15348 Spieler alle 17529 Einträge der tmp\_fifa\_ratings auf ein Matching geprüft werden mussten.

### player\_events

Da wir nur Spieler, die in 2020 oder später spielten, betrachten, macht es Sinn die player\_events gleichermassen einzuschränken.

```
-- player_events vor 2020 löschen
```

```
delete from player_events where game_date < '2020-01-01';
```

```
62 15:12:28 delete from player_events where game_date < '2020-01-01' 981259 row(s) affected 36.031 sec
```

Von den 1'563'583 Einträgen wurden 981'259 gelöscht da sie vor dem 01.01.2020 geschahen.

### games

Die games Tabelle wurde in gleicher Manier transformiert.

```
-- games vor 2020 löschen
```

```
Delete from games where season < 2020;
```

```
64 11:17:49 Delete from games where season < 2020 45439 row(s) affected 1.031 sec
```

Alle Spiele vor 2020 (45439 Einträge) wurden erfolgreich gelöscht.

Zusätzlich wurde die Spalte stadium von der Tabelle games gelöscht, da wir später merkten, dass wir für diese keine Verwendung haben.

```
-- Spalte stadium von games löschen
```

```
alter table games  
drop column stadium;
```

```
26 18:40:34 alter table games drop column stadium
```

```
0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0 0.266 sec
```

Die Spalte wurde erfolgreich gelöscht.

### game\_events

Die game\_events wurden gleich transformiert wie die Games Tabelle.

```
-- game_events vor 2020 löschen
```

```
Delete from game_events where game_event_date < '2020-01-01';
```

```
1 17:38:07 DELETE FROM game_events WHERE game_event_date < '2020-01-01' 450226 row(s) affected 71.016 sec
```

450226 Einträge wurden in ca. 71s erfolgreich gelöscht.

### 4.3.3 Load MongoDB

Die Daten wurden in MongoDB geladen mithilfe des Befehls `mongoimport`. Die Daten befinden sich alle auf einer privaten Remote-Maschine in Form von .csv-Dateien. Alle .csv-Dateien wurden aus der MySQL-Datenbank exportiert, um eine Datenkonsistenz zwischen den beiden Datenbanken zu erreichen. Es wurden alle .csv-Dateien zuerst in ein reines Datendokument importiert und erst dann in einem zweiten Schritt in ein adäquateres MongoDB-Dokument transformiert.

Das MongoDB Query auf dem Client, also unserem persönlichen Gerät ausgeführt. Beim `mongoimport` Query gibt dabei gewisse Parameter, die immer gleichbleiben. Mit dem `--host` Parameter wird die IP unserer VM angegeben und mit `--port` der Port, auf welchem der MongoDB-Server Daten empfängt. Zudem werden mit `--username` und `--password` die Credentials des Users `adminUser` angegeben (User existiert mittlerweile nicht mehr). Des Weiteren werden die spezifische Datenbank und die Collection angegeben, in welche das .csv-Dokument gespeichert wird. Schlussendlich wird auch definiert, dass 4 Threads gleichzeitig arbeiten, ein Batch eine Größe von 1000 Elementen beinhaltet und der Ort der Datei auf der lokalen Maschine. Die import Querys unterscheiden sich nur in den Ziel Collections auf dem MongoDB Server und dem Dateipfad zum lokalen file auf dem Client.

#### Players\_raw

```
PS C:\Users\fabia> mongoimport --host 86.119.35.98 --port 27017 --username adminUser --password adminPassword --authenticationDatabase admin --db scoutingAndAnalysis --collection Players_raw --type csv --headerline --numInsertionWorkers 4 --batchSize 1000 --file C:\Users\fabia\Documents\HSLU\Semester2\DB_Material\MongoDB\uploadedFilesToMongoDB\ExportPlayers.csv
2024-05-10T15:42:21.456+0200      connected to: mongodb://86.119.35.98:27017/
2024-05-10T15:42:23.660+0200      15348 document(s) imported successfully. 0 document(s) failed to import.
PS C:\Users\fabia>
```

#### PlayerEvents\_raw

```
2024-05-02T13:38:28.355+0200      Failed: open C:\Users\fabia\Documents\HSLU\Semestermongoimport --host 86.119.35.98 --port 27017 --use
rname adminUser --password adminPassword --authenticationDatabase admin --db scoutingAndAnalysis --collection PlayerEvents_raw --type
(csv --headerline --numInsertionWorkers 4 --batchSize 1000 --file C:\Users\fabia\Documents\HSLU\Semester2\DB_Material\MongoDB\uploade
dFilesToMongoDB\ExportPlayerEvents.csv
2024-05-02T13:39:36.964+0200      connected to: mongodb://86.119.35.98:27017/
2024-05-02T13:39:39.978+0200      [#####.....] scoutingAndAnalysis.PlayerEvents_raw 4.46MB/23.8MB (18.7%)
2024-05-02T13:39:42.966+0200      [##########.....] scoutingAndAnalysis.PlayerEvents_raw 8.62MB/23.8MB (36.2%)
2024-05-02T13:39:45.972+0200      [###############.....] scoutingAndAnalysis.PlayerEvents_raw 13.3MB/23.8MB (55.8%)
2024-05-02T13:39:48.964+0200      [####################.....] scoutingAndAnalysis.PlayerEvents_raw 18.2MB/23.8MB (76.4%)
2024-05-02T13:39:51.964+0200      [####################.....] scoutingAndAnalysis.PlayerEvents_raw 23.0MB/23.8MB (96.6%)
2024-05-02T13:39:52.805+0200      [####################.....] scoutingAndAnalysis.PlayerEvents_raw 23.8MB/23.8MB (100.0%)
2024-05-02T13:39:52.806+0200      474725 document(s) imported successfully. 0 document(s) failed to import.
PS C:\Users\fabia>
```

#### Games\_raw

```
PS C:\Users\fabia> mongoimport --host 86.119.35.98 --port 27017
--username adminUser --password adminPassword --authenticationDa
tabase admin --db scoutingAndAnalysis --collection Games_raw --t
ype csv --headerline --numInsertionWorkers 4 --batchSize 1000 --
file C:\Users\fabia\Documents\HSLU\Semester2\DB_Material\MongoDB
\uploadedFilesToMongoDB\ExportGames.csv
2024-05-02T13:22:33.725+0200      connected to: mongodb://86.119.3
5.98:27017/
2024-05-02T13:22:35.747+0200      17458 document(s) imported succe
ssfully. 0 document(s) failed to import.
PS C:\Users\fabia> |
```

#### GameEvents\_raw

```
PS C:\Users\fabia> mongoimport --host 86.119.35.98 --port 27017 --username adminUser --password adminPassword --authenticationDatabase
admin --db scoutingAndAnalysis --collection GameEvents_raw --type csv --headerline --numInsertionWorkers 4 --batchSize 1000 --file
C:\Users\fabia\Documents\HSLU\Semester2\DB_Material\MongoDB\uploadedFilesToMongoDB\ExportGameEventsReformatted2.csv
2024-05-10T11:14:15.449+0200      connected to: mongodb://86.119.35.98:27017/
2024-05-10T11:14:18.460+0200      [#####.....] scoutingAndAnalysis.GameEvents_raw 4.30MB/18.8MB (22.9%)
2024-05-10T11:14:21.450+0200      [##########.....] scoutingAndAnalysis.GameEvents_raw 8.88MB/18.8MB (47.2%)
2024-05-10T11:14:24.449+0200      [###############.....] scoutingAndAnalysis.GameEvents_raw 14.3MB/18.8MB (76.2%)
2024-05-10T11:14:27.457+0200      [####################.....] scoutingAndAnalysis.GameEvents_raw 18.2MB/18.8MB (96.9%)
2024-05-10T11:14:27.957+0200      [####################.....] scoutingAndAnalysis.GameEvents_raw 18.8MB/18.8MB (100.0%)
2024-05-10T11:14:27.957+0200      192981 document(s) imported successfully. 0 document(s) failed to import.
PS C:\Users\fabia>
```

#### Clubs\_raw

```
PS C:\Users\fabia> mongoimport --host 86.119.35.98 --port 27017 --username adminUser --password adminPassword --authenticationDatabase admin --db scoutingAndAnalysis --collection Clubs_raw --type csv --headerline --numInsertionWorkers 4 --batchSize 1000 --file C:\Users\fabia\Documents\HSLU\Semester2\DB_Material\MongoDB\uploadedFilesToMongoDB\clubs.csv
2024-04-29T14:53:10.308+0200      connected to: mongodb://86.119.35.98:27017/
2024-04-29T14:53:11.226+0200      426 document(s) imported successfully. 0 document(s) failed to import.
PS C:\Users\fabia> |
```

### Competitions\_raw

```
PS C:\Users\fabia> mongoimport --host 86.119.35.98 --port 27017 --username adminUser --password adminPassword --authenticationDatabase admin --db scoutingAndAnalysis --collection Competitions_raw --type csv --headerline --numInsertionWorkers 4 --batchSize 1000 --file C:\Users\fabia\Documents\HSLU\Semester2\DB_Material\MongoDB\uploadedFilesToMongoDB\ExportCompetitions.csv
2024-05-10T11:38:57.849+0200      connected to: mongodb://86.119.35.98:27017/
2024-05-10T11:38:58.251+0200      43 document(s) imported successfully. 0 document(s) failed to import.
PS C:\Users\fabia>
```

## 4.3.4 Transform MongoDB

Die reinen Daten Dokumente werden in einem zweiten Schritt in logische Dokumente transformiert. Die bereits im Kapitel zu den JSON-Prototypen erwähnte Strukturierung der Daten in diese drei Dokumente werden nun vollzogen. Zudem werden zuerst auch noch weitere Anpassungen gemacht wie beispielsweise das Anpassen von Datentypen in den raw\_Documents.

### 4.3.4.1 Datentyp Anpassungen

#### String zu ISO-Date

Im Dokument PlayerEvents\_raw wird das Attribut game\_date von einem string in ein date umgewandelt sodass in Abfragen dann auch als date möglich und vergleichbar sind.

Im query wird dann auf das PlayerEvents\_raw Dokument die updateMany Funktion aufgerufen, welche mehrere Objekte gleichzeitig verändern kann. Dann werden nur Objekte berücksichtigt, auf denen game\_date existiert und ein string ist. Mit set wird das field game\_date mit der Funktion toDate in einen ISO-Date datentyp umgewandelt.

```
scoutingAndAnalysis> db.PlayerEvents_raw.updateMany(
...   { game_date: { $exists: true, $type: "string" } }, // Stellt sicher, dass das Feld existiert und ein String ist
...   [
...     {
...       $set: {
...         game_date: {
...           $toDate: "$game_date" // Konvertiert den String zu einem Datum
...         }
...       }
...     }
...   ]
... );
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 474725,
  modifiedCount: 474725,
  upsertedCount: 0
}
```

Eine identische Funktion wird auch auf folgenden Dokumenten und fields ausgeführt, um echte ISO-Date Datentypen zu erhalten.

Dokument	Players_raw	Games_raw	Game_Events_raw
Attribute	date_of_birth	game_date	Game_event_date
Matched / Modified Count	14116	17458	192981

In einem zweiten Schritt werden beim Import falsch gelesene Null Werte in korrekte Nullwerte umgewandelt. Diese Transformation wird auch an einem bestimmten Beispiel gezeigt. In diesem query werden alle Einträge in Players\_raw neugesetzt. Das query ist ein bisschen zu lang, um es ganz abzubilden, weshalb hier nur der Anfang und das Resultat des querys zu finden sind.

Das query sucht für jedes Objekt im Dokument Players\_raw, ob der Wert für das Attribut \N entspricht. Wenn ja wird es mit dem korrekten Null value ersetzt und wenn ein korrekter Wert gesetzt ist, wird das Feld ignoriert. Zudem können auch mehrere conditions wahr sein, also auch mehrere Attribute auf einem Objekt verändert werden. Dies wird mit dem {multi:true} am Ende signalisiert.

```
db.Players_raw.updateMany(
  {},
  [
    { $set: {
      contract_exp_date: {
        $cond: {
          if: { $eq: ["$contract_exp_date", "\N"] },
          then: null,
          else: "$contract_exp_date"
        }
      },
      ...
    ...
  ...
  ...
}
```

```
...
  },
  fifa_heading_accuracy: {
    $cond: {
      if: { $eq: ["$fifa_heading_accuracy", "\N"] },
      then: null,
      else: "$fifa_heading_accuracy"
    }
  },
  fifa_short_passing: {
    $cond: {
      if: { $eq: ["$fifa_short_passing", "\N"] },
      then: null,
      else: "$fifa_short_passing"
    }
  }
},
  ],
  { multi: true }
);
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 15348,
  modifiedCount: 15346,
  upsertedCount: 0
}
```

Es werden auch hier auf mehreren anderen Dokumenten auch weitere Null Transformationen vorgenommen

<b>Dokument</b>	Games_raw	GameEvents_r
		aw

<b>Attribut</b>	attendance	Player_in_id, player_assist_i d
<b>Matched / Modified Count</b>	5028	192981

#### 4.3.4.2 Erstellen der JSON-Dokumente

Nachfolgend wird für jedes Document immer das root Level als Ausgang geladen. Die Operationen ähneln sich stark bei allen Dokumenten, weshalb die Transformation nur am Clubs\_Dokument exemplarisch gezeigt wird.

Das ClubsDocument ist also zuerst eine reine Kopie des Clubs\_raw Dokument. Danach werden die interessanten Informationen aus anderen raw Dokumenten mit einem lookup in das ClubsDocument gemerged. Dies wird anhand des "foreign key" gemacht. In MongoDB gibt es zwar keinen im System definierten foreign key, trotzdem kann anhand dieser attribute der lookup vorgenommen werden, da sie die gleichen attribute sind.

## ClubsDocument

### Players\_raw

Hier wird ein lookup gemacht mit einem foreign field (Feld im Players\_raw) und dem local field (Feld im Clubs\_Document). Hier ist das gemeinsame Feld die club\_id. In der merge stage der aggregation pipeline werden dann die neuen Daten, die beim lookup angehängt werden, wieder in das ClubsDocument eingefügt. Dies wird anhand der id gemacht, sodass jedes Objekt die korrekten Daten bekommt. Wenn also das passende Objekt gefunden wurde (noch ohne zusätzliche Informationen), wird dieses ersetzt.

Danach sind die Objekte mit den neuen Informationen ergänzt worden.

```
scoutingAndAnalysis> db.ClubsDocument.aggregate([
...   {
...     $lookup: {
...       from: "Players_raw",
...       localField: "club_id",
...       foreignField: "current_club_id",
...       as: "Players"
...     }
...   },
...   {
...     $merge: {
...       into: "ClubsDocument", // Merging back into the same collection
...       on: "_id",           // Use the document ID to match documents
...       whenMatched: "merge", // Merge the updated 'Clubs' into the existing documents
...       whenNotMatched: "discard" // Ignore if no matching document is found (should not happen in this context)
...     }
...   }
... ])
```

### GameEvents\_raw

Hier wird ebenfalls auf das attribut club\_id einen lookup gemacht und somit GameEvents\_raw an das Clubs\_Document angefügt.

```
scoutingAndAnalysis> db.ClubsDocument.aggregate([
...   {
...     $lookup: {
...       from: "GameEvents_raw",
...       localField: "club_id",
...       foreignField: "club_id",
...       as: "Game_Events"
...     }
...   },
...   {
...     $merge: {
...       into: "ClubsDocument", // Merging back into the same collection
...       on: "_id",           // Use the document ID to match documents
...       whenMatched: "merge", // Merge the updated 'Clubs' into the existing documents
...       whenNotMatched: "discard" // Ignore if no matching document is found (should not happen in this context)
...     }
...   }
... ])
```

### *Competitions\_raw*

Hier werden die Inhalte des Competitions\_raw an das ClubsDocument angefügt, und zwar mithilfe der competition\_id.

```
scoutingAndAnalysis> db.ClubsDocument.aggregate([
...   {
...     $lookup: {
...       from: "Competitions_raw",
...       localField: "domestic_competition_id",
...       foreignField: "competition_id",
...       as: "Competition"
...     }
...   },
...   {
...     $merge: {
...       into: "ClubsDocument", // Merging back into the same collection
...       on: "_id",           // Use the document ID to match documents
...       whenMatched: "merge", // Merge the updated 'Clubs' into the existing documents
...       whenNotMatched: "discard" // Ignore if no matching document is found (should not happen in this context)
...     }
...   }
... ])
```

### *Games\_raw*

Hier werden die Daten aus Games\_raw für jeden Club in das ClubsDocument angefügt. Etwas spezieller ist hier, dass ein Club Auswärtsspiele und Heimspiele haben kann. Deshalb werden in diesem query beide benutzt. Das pipeline-Dokument im \$lookup-Abschnitt filtert die Games\_raw-Dokumente so, dass nur diejenigen Spiele zurückgegeben werden, bei denen entweder home\_club\_id oder away\_club\_id mit club\_id aus ClubsDocument übereinstimmt. Diese gefilterten Spiele werden dann dem Games-Feld im jeweiligen ClubsDocument hinzugefügt.

```
scoutingAndAnalysis> db.ClubsDocument.aggregate([
...   {
...     $lookup: {
...       from: "Games_raw",
...       let: { clubId: "$club_id" },
...       pipeline: [
...         {
...           $match: {
...             $expr: {
...               $or: [
...                 { $eq: ["$home_club_id", "$$clubId"] },
...                 { $eq: ["$away_club_id", "$$clubId"] }
...               ]
...             }
...           }
...         ],
...         as: "Games"
...       }
...     },
...     {
...       $merge: {
...         into: "ClubsDocument",
...         on: "_id",
...         whenMatched: "merge",
...         whenNotMatched: "discard"
...       }
...     }
...   ]
... ])
```

#### 4.3.4.3 Weitere Dokumente Games und Players Documents

Die weiteren End-Dokumente wurden ebenfalls ähnlich erstellt wie das GamesDocument. Es wird deshalb nicht genau auf den mongoDB Syntax eingegangen, sondern nur noch auf spezielle Abweichungen oder Besonderheiten. Es werden aber alle Screenshots zu den Querys gezeigt.

##### Games Dokument

###### Away\_Club

Hier werden die Informationen des away\_club bezüglich des spezifischen game geladen.

Home\_Club ist dabei identisch zu diesem query.

```
scoutingAndAnalysis> db.GamesDocument.aggregate([
...   {
...     $lookup: {
...       from: "Clubs_raw",
...       localField: "away_club_id",
...       foreignField: "club_id",
...       as: "Away_Club"
...     }
...   },
...   {
...     $merge: {
...       into: "GamesDocument", // Merging back into the same collection
...       on: "_id",           // Use the document ID to match documents
...       whenMatched: "merge", // Merge the updated 'Clubs' into the existing documents
...       whenNotMatched: "discard" // Ignore if no matching document is found (should not happen in this context)
...     }
...   }
... ])
```

###### Competition

Hier wird die passende competition\_id den Games Objekten hinzugefügt.

```
scoutingAndAnalysis> db.GamesDocument.aggregate([
...   {
...     $lookup: {
...       from: "Competitions_raw",
...       localField: "competition_id",
...       foreignField: "competition_id",
...       as: "Competition"
...     }
...   },
...   {
...     $merge: {
...       into: "GamesDocument", // Merging back into the same collection
...       on: "_id",           // Use the document ID to match documents
...       whenMatched: "merge", // Merge the updated 'Clubs' into the existing documents
...       whenNotMatched: "discard" // Ignore if no matching document is found (should not happen in this context)
...     }
...   }
... ])
```

###### Game Events

Hier werden die Game Events zu den passenden Games Objekten geladen. Speziell ist hier, dass ein leerer Game\_Events array erstellt wird, wenn es keine gameEvents gibt.

```
scoutingAndAnalysis> db.GamesDocument.aggregate([
...   {
...     $lookup: {
...       from: "GameEvents_raw",
...       localField: "game_id",
...       foreignField: "game_id",
...       as: "Game_Events"
...     }
...   },
...   {
...     $addFields: {
...       Game_Events: {
...         $ifNull: ["$Game_Events", []] // Ensures that 'Clubs' is an empty array if no clubs are found
...       }
...     }
...   },
...   {
...     $merge: {
...       into: "GamesDocument", // Merging back into the same collection
...       on: "_id",           // Use the document ID to match documents
...       whenMatched: "merge", // Merge the updated 'Clubs' into the existing documents
...       whenNotMatched: "discard" // Ignore if no matching document is found (should not happen in this context)
...     }
...   }
... ])
```

##### Players Document

## Clubs

Hier werden Clubs an das Players\_Document hinzugefügt.

```
scoutingAndAnalysis> db.PlayersDocument.aggregate([
...   {
...     $lookup: {
...       from: "Clubs_raw",
...       localField: "current_club_id",
...       foreignField: "club_id",
...       as: "Clubs"
...     }
...   },
...   {
...     $addFields: {
...       Clubs: {
...         $ifNull: ["$Clubs", []] // Ensures that 'Clubs' is an empty array if no clubs are found
...       }
...     }
...   },
...   {
...     $merge: {
...       into: "PlayersDocument", // Merging back into the same collection
...       on: "_id", // Use the document ID to match documents
...       whenMatched: "merge", // Merge the updated 'Clubs' into the existing documents
...       whenNotMatched: "discard" // Ignore if no matching document is found (should not happen in this context)
...     }
...   }
... ]);

scoutingAndAnalysis>
```

## Player Events

Hier werden die Player Events an das Players Document angehängt.

```
scoutingAndAnalysis> db.Players_raw.aggregate([
...   {
...     $lookup: {
...       from: "PlayerEvents_raw",
...       localField: "player_id",
...       foreignField: "player_id",
...       as: "Player_Events"
...     }
...   },
...   {
...     $merge: {
...       into: "PlayersDocument", // Merging back into the same collection
...       on: "_id", // Use the document ID to match documents
...       whenMatched: "replace", // Replace the existing document
...       whenNotMatched: "insert" // Do nothing if no matching document is found
...     }
...   }
... ]);

scoutingAndAnalysis>
```

## 5 Daten analysieren & auswerten

### 5.1 Datenabfragen Erklärung

Bei den Abfragen wurden zwei verschiedene Arten von Nutzen mit den Querys geschaffen. Zum einen möchten wir Informationen über Spieler und potenzielle Spieler erhalten (scouting). Im scouting query wurden dabei Spieler gesucht die optimal zu unserem Verein passen. Eingeschränkt wurden also nach country\_of\_birth, der sub\_position des Spielers und dem market\_value\_in\_euro.

Zum anderen möchten wir Informationen zu unseren Gegnern erhalten (analysis). Es ist zum Beispiel interessant welche der Gegner in unserer Liga immer spät im Spiel Tore erzielen. Wenn wir diese Information kennen, kann der Trainer unserer Mannschaft sich auf den Gegner einstellen und entsprechende taktische Massnahmen treffen, um ein spätes gegenerisches Tor möglichst zu verhindern. Ein weiterer use case im analysis Bereich ist auch, die mögliche Aufstellung eines Gegners zu antizipieren, indem die vergangenen Aufstellungen analysiert werden. So kann der Trainer unserer Mannschaft die optimalen taktischen Gegenmassnahmen bereits vor dem Anpfiff vornehmen

## 5.2 SQL Abfragen

### 5.2.1 Abfrage scouting Spieler SQL

The screenshot shows the Oracle SQL Developer interface. At the top, there is a toolbar with various icons. Below the toolbar, the SQL editor window displays the following SQL code:

```
1 -- view which finds the best young players with the provided age, at a certain position and market_value o
2 • CREATE OR REPLACE VIEW bestYoungPlayers AS
3
4 select firstname, lastname, height_in_cm
5   from players
6  where players.country_of_birth = 'Switzerland' and
7    players.sub_position = 'Centre-Back' and
8    players.marketvalue_in_euro > 1000000
9
10 order by height_in_cm DESC;
11
12 • select * from bestYoungPlayers
```

To the right of the SQL editor, a message box states: "Automatic context help disabled. Use the toolbar manually get help for the current caret position or toggle automatic help".

Below the SQL editor is the Result Grid window, which displays the following data:

firstname	lastname	height_in_cm
Cédric	Zesiger	194
Berat	Djimsti	190
Nico	Elvedi	189
Milos	Veljkovic	188
Manuel	Akanji	188
Edimilson	Fernandes	187
Becir	Omeragic	187
Edimilson	Costa	186

At the bottom of the interface, the Output window shows the execution log:

Action Output
# Time Action Message Duration / Fetch
6 16:31:07 CREATE OR REPLACE VIEW bestYoungPlayers AS select firstname, lastname, ... 0 row(s) affected 0.266 sec

1. In einem ersten Schritt wird hier eine view erstellt, welche dann ganz am Ende mit einem select statement abgefragt wird, wo uns alle Spalten der view ausgegeben werden.
2. Das Hauptstatement besteht aus einem select auf dem table players.
3. Die Abfrage wird weiter eingeschränkt mit einer where clause, welche nach dem country\_of\_birth, der sub\_position und dem marketvalue\_in\_euro eingeschränkt wird. Wir sind nur an Spielern aus der Schweiz, mit Position Centre-Back (Innenverteidigung) und einem Marktwert der grösser als eine Million ist. Diese Anforderungen wurden mit einem "and" keyword verknüpft und müssen alle erfüllt werden.
4. Zusätzlich sortieren wir die Auswahl noch nach der Grösse des Spielers. In der Innenverteidigung ist die Grösse auch interessant, weshalb wir den grössten Innenverteidiger zuoberst anzeigen lassen. Dies wird mit dem keyword DESC erreicht.

## 5.2.2 Abfrage analysis Teams mit späten Treffern SQL

```

3 • CREATE OR REPLACE VIEW LateGoalsByClubsAfter80 AS
4   SELECT
5     ge.club_id,
6     c.club_name,
7     cdc.domestic_competition_id,
8     COUNT(*) AS goalsAfter80
9
10  FROM
11    game_events ge
12  JOIN
13    clubs c ON ge.club_id = c.club_id
14  JOIN
15    club_has_domestic_competition cdc ON ge.club_id = cdc.club_id
16  JOIN
17    game_events_has_date ged ON ge.game_event_id = ged.game_event_id
18 WHERE
19   ge.game_event_type = 'Goals' AND
20   ge.minute > 79 AND
21   -- ge.club_id = findClubId('FC Bayern München') and
22   cdc.domestic_competition_id = 'GB1' AND
23   ged.game_event_date BETWEEN '2021-08-14' AND '2022-03-24'
24
25 GROUP BY
26   ge.club_id
27 ORDER BY
28   goalsAfter80 DESC
29 LIMIT 5;
30
31 • SELECT * FROM LateGoalsByClubsAfter80;

```

club_id	club_name	domestic_competition_id	goalsAfter80
281	Manchester City Football Club	GB1	16
985	Manchester United Football Club	GB1	15
379	West Ham United Football Club	GB1	15
631	Chelsea Football Club	GB1	14
31	Liverpool Football Club	GB1	13

Output

#	Time	Action	Message
1	16:44:58	CREATE OR REPLACE VIEW LateGoalsByClubsAfter80 AS SELECT ge.club_id, c.club_name, cdc.domestic_competition_id, COUNT(*) AS ...	0 row(s) affected
2	16:44:58	SELECT * FROM LateGoalsByClubsAfter80 LIMIT 0, 50000	5 row(s) returned

1. Hier werden im ersten SELECT Statement die club\_id, den club\_name, die domestic\_competition\_id und die Anzahl Tore nach der 80 Minute selektiert.
2. Um die benötigten Daten zu erhalten, werden die ausgelagerten tables club\_has\_domestic\_competition und game\_events\_has\_date, sowie der table clubs zum table game\_events gejoined. Dies passiert bei den drei Keywords JOIN. Nachdem keyword ON wird, dann immer angegeben auf welches Attribut der Join erfolgen soll.
3. In der where clause wird dann eingeschränkt.  
Der game\_event\_type vom table game\_events muss vom type "Goals" sein. Die Minute muss strikt grösser sein als 79. Die Liga wird mit der competition\_id eingeschränkt und der game\_event\_date muss im Zeitraum der Saison 2021/2022 liegen.  
Die Zeile wird im obersten SELECT also nur berücksichtigt, falls alle diese Bedingungen zustimmen.
4. Danach wird nach der club\_id gruppiert. Alle Einträge mit derselben club\_id gruppiert und dann mit dem COUNT(\*) im obersten SELECT gezählt.
5. Zum Schluss werden die gruppierten Resultate dann noch mit dem keyword DESC nach der Grösse absteigend sortiert und mit 5 auszugebenden Zeilen limitiert mit dem LIMIT keyword.
6. Auch hier wird mit einer zusätzlichen view gearbeitet.

### 5.2.3 Abfrage analysis Aufstellung Gegner

```
7
8 -- EXPLAIN
9 • 9 select formation, count(*) as count from (
10   select home_clubFormation as formation
11     from games
12    where (home_club_id = 235)
13    and games.game_date BETWEEN '2020-08-14' AND '2022-02-01'
14  UNION ALL
15  select away_clubFormation as formation from games
16  where (away_club_id = 235)
17  and games.game_date BETWEEN '2020-08-14' AND '2022-02-01'
18  as homeAndAwayGames
19 group by formation
20 order by count(*) DESC;
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
formation	count			
▶ 4-3-3 Attacking	24			
4-3-3 Defending	17			
3-4-3	4			
5-3-2	3			
3-5-2	3			
3-4-2-1	2			
5-4-1	2			
4-4-1-1	1			
3-4-1-2	1			

1. In diesem query wird zuerst ein select statement auf die formation und ein count(\*) angewendet. Es wird also gezählt, wie oft eine spezifische Formation beim selektierten Verein gespielt wurde.
  2. In einem zweiten Schritt wird das subquery definiert. In dieser Untermenge werden zwei Mengen mit einem UNION ALL Operator zusammengefügt. Die erste Menge sind alle Heimspiele und die zweite Menge sind alle away\_games desselben Teams. Es wird dabei der Zeitraum dieser Spiele eingeschränkt, wobei game\_date mit dem BETWEEN keyword gefiltert. Es werden also alle Einträge behalten, die einen Wert haben, welcher zwischen den beiden date-Werten liegt. Danach wurden die Resultat nach der formation gruppiert.
  3. Schlussendlich werden die Daten nach der Grösse, respektive der Anzahl der Formationen gruppiert. Die Formation, die am meisten gespielt wurde, erscheint dann ganz oben.

## 5.3 Mongo DB Abfragen

### **5.3.1 Abfrage scouting Spieler MongoDB**

```
scoutingAndAnalysis> db.PlayersDocument.aggregate([
...   {
...     $match: {
...       country_of_birth: "Switzerland", // Filters players born in Switzerland
...       sub_position: "Centre-Back", // Filters players who play as Centre-Back
...       marketvalue_in_euro: { $gt: 1000000 } // Filters players with a market value over 1,000,000 Euros
...     }
...   },
...   {
...     $sort: {
...       height_in_cm: -1 // Sorts the documents by height in descending order
...     }
...   },
...   {
...     $project: {
...       firstname: 1, // Include the firstname in the output
...       lastname: 1, // Include the lastname in the output
...       height_in_cm: 1 // Include the height in cm in the output
...     }
...   }
... ]);
[
```

1. Hier wird eine Abfrage mit aggregate auf das Dokument "PlayersDocument" gemacht. Die aggregation-Pipeline besteht dabei aus drei Teilen.
2. Der match-Operator filtert dabei die Objekte des PlayersDocument nach den drei gleichen Kriterien wie in der SQL-Abfrage. Dass ein Objekt nach der match stage also noch berücksichtigt werden soll, muss es die entsprechenden attribute besitzen.
3. Sort sortiert danach die übrigen Objekte nach der Grösse absteigend.
4. Project gibt dann nur die drei Attribute aller übrig gebliebenen Objekte auf der Konsole aus.

**Resultat:**

```
{
  "_id": ObjectId('663e37adela85805ea0f0c78'),
  "firstname": "Cédric",
  "lastname": "Zesiger",
  "height_in_cm": 194
},
{
  "_id": ObjectId('663e37acela85805ea0efcd9'),
  "firstname": "Berat",
  "lastname": "Djimsiti",
  "height_in_cm": 190
},
{
  "_id": ObjectId('663e37acela85805ea0efe0e'),
  "firstname": "Nico",
  "lastname": "Elvedi",
  "height_in_cm": 189
},
{
  "_id": ObjectId('663e37acela85805ea0efb78'),
  "firstname": "Milos",
  "lastname": "Veljkovic",
  "height_in_cm": 188
},
{
  "_id": ObjectId('663e37adela85805ea0f04c8'),
  "firstname": "Manuel",
  "lastname": "Akanji",
  "height_in_cm": 188
},
{
  "_id": ObjectId('663e37adela85805ea0f01fb'),
  "firstname": "Edimilson",
  "lastname": "Fernandes",
  "height_in_cm": 187
},
{
  "_id": ObjectId('663e37aeela85805ea0f151f'),
  "firstname": "Becir",
  "lastname": "Omeragic",
  "height_in_cm": 187
},
{
  "_id": ObjectId('663e37acela85805ea0efa5f'),
  "firstname": "Fabian",
  "lastname": "Schär",
  "height_in_cm": 186
},
{
  "_id": ObjectId('663e37adela85805ea0f0639'),
  "firstname": "Eray",
  "lastname": "Cömert",
  "height_in_cm": 183
},
{
  "_id": ObjectId('663e37acela85805ea0ef6d5'),
  "firstname": "Ricardo",
  "lastname": "Rodríguez",
  "height_in_cm": 182
},
{
  "_id": ObjectId('663e37aeela85805ea0f1520'),
  "firstname": "Leonidas",
  "lastname": "Stergiou",
  "height_in_cm": 181
}
```

	firstname	lastname	height_in_cm
▶	Cédric	Zesiger	194
	Berat	Djimsiti	190
	Nico	Elvedi	189
	Milos	Veljkovic	188
	Manuel	Akanji	188
	Edimilson	Fernandes	187
	Becir	Omeragic	187
	Fabian	Schär	186
	Eray	Cömert	183
	Ricardo	Rodríguez	182
	Leonidas	Stergiou	181

Beim Vergleich mit der Abfrage im SQL-Teil, stellen wir fest, dass der Inhalt der Ausgabe identisch ist.

### 5.3.2 Abfrage analysis Teams mit späten Treffern MongoDB

```
scoutingAndAnalysis> db.ClubsDocument.aggregate([
...   {
...     $match: {
...       "Competition.competition_id": "GB1" // Filtert Clubs in der Premier League
...     }
...   },
...   {
...     $unwind: "$Game_Events" // Entfaltet das Array "Game_Events" für jedes Event
...   },
...   {
...     $match: {
...       "Game_Events.game_event_type": "Goals", // Filtert auf Tor-Ereignisse
...       "Game_Events.minute": { $gt: 79 }, // Filtert auf späte Tore nach der 79. Minute
...       "Game_Events.game_event_date": {
...         $gte: ISODate("2021-08-14T00:00:00.000Z"),
...         $lte: ISODate("2022-03-24T23:59:59.999Z")
...       } // Zeitfenster für die Spiele
...     }
...   },
...   {
...     $group: {
...       _id: "$club_id", // Gruppiert nach club_id
...       club_name: { $first: "$club_name" }, // Speichert den Clubnamen
...       goalsAfter80: { $sum: 1 } // Zählt die Tore nach der 79. Minute
...     }
...   },
...   {
...     $sort: { goalsAfter80: -1 } // Sortiert die Ergebnisse nach der Anzahl der Tore absteigend
...   },
...   {
...     $limit: 5 // Begrenzt die Ergebnisse auf die Top 5
...   }
... ]);
```

1. Match behaltet alle Objekte die der competition\_id "GB1" entsprechen.
2. Mit unwind wird dann pro Objekt im GameEvents array ein neues Objekt erstellt.
3. Mit match werden, dann diese neu generierten Objekte wieder eingeschränkt. Zum einen auf die Minute in der das Tor geschossen wurde und zum anderen auf den Zeitraum
4. Dann werden mit group die übrig gebliebenen Objekte gruppiert und gezählt.
5. Mit sort werden die Ergebnisse dann sortiert und mit limit auf 5 beschränkt.

#### Resultat:

```

{
  {
    _id: 281,
    club_name: 'Manchester City Football Club',
    goalsAfter80: 16
  },
  {
    _id: 379,
    club_name: 'West Ham United Football Club',
    goalsAfter80: 15
  },
  {
    _id: 985,
    club_name: 'Manchester United Football Club',
    goalsAfter80: 15
  },
  { _id: 631, club_name: 'Chelsea Football Club', goalsAfter80: 14 },
  { _id: 31, club_name: 'Liverpool Football Club', goalsAfter80: 13 }
]
scoutingAndAnalysis>

```

Result Grid				
	dub_id	dub_name	domestic_competition_id	goalsAfter80
▶	281	Manchester City Football Club	GB1	16
	985	Manchester United Football Club	GB1	15
	379	West Ham United Football Club	GB1	15
	631	Chelsea Football Club	GB1	14
	31	Liverpool Football Club	GB1	13

Auch hier lässt sich feststellen, dass die Resultate übereinstimmen.

### 5.3.3 Abfrage analysis Aufstellung des Gegners in MongoDB

```

scoutingAndAnalysis> db.GamesDocument.aggregate([
...   {
...     $match: {
...       $or: [
...         {
...           home_club_id: 235,
...           game_date: {
...             $gte: ISODate("2020-08-14T00:00:00.000Z"),
...             $lte: ISODate("2022-02-01T00:00:00.000Z")
...           }
...         },
...         {
...           away_club_id: 235,
...           game_date: {
...             $gte: ISODate("2020-08-14T00:00:00.000Z"),
...             $lte: ISODate("2022-02-01T00:00:00.000Z")
...           }
...         }
...       ]
...     }
...   },
...   {
...     $lookup: {
...       from: "ClubsDocument",
...       localField: "home_club_id",
...       foreignField: "club_id",
...       as: "home_club_info"
...     }
...   },
...   {
...     $lookup: {
...       from: "ClubsDocument",
...       localField: "away_club_id",
...       foreignField: "club_id",
...       as: "away_club_info"
...     }
...   },
...   {
...     $project: {
...       formation: {
...         $cond: {
...           if: { $eq: ["$home_club_id", 235] },
...           then: "$home_clubFormation",
...           else: "$away_clubFormation"
...         }
...       },
...       club_name: {
...         $cond: {
...           if: { $eq: ["$home_club_id", 235] },
...           then: { $arrayElemAt: ["$home_club_info.club_name", 0] },
...           else: { $arrayElemAt: ["$away_club_info.club_name", 0] }
...         }
...       }
...     }
...   },
...   {
...     $group: {
...       _id: { formation: "$formation", club_name: "$club_name" },
...       count: { $sum: 1 }
...     }
...   },
...   {
...     $sort: { count: -1 }
...   }
... ]);

```

1. In einem ersten Schritt werden die korrekten Objekte mit der gewünschte club\_id mit der home\_club\_id oder der away\_club\_id gefiltert. Zudem wird auch der Zeitraum eingeschränkt. Die club\_id kann beliebig gewählt werden. In Beispiel wurde die club\_id 235 gewählt, welche dem club Rooms Katholieke Combinatie Waalwijk entspricht.
2. Nach dem Filtern wird dann der Lookup ins Clubs\_Document gemacht und die interessanten Informationen aus diesem Document herangezogen. Erstens werden alle home\_club\_id's und zweitens alle away\_club\_id's gejoined.
3. In der dritten Stufe werden dann die formation und der club\_name ausgegeben. Es wird jeweils noch geprüft, ob beim momentanen Objekt nun die home\_club oder away\_clubFormation berücksichtigt wird. Diese Funktionalität geschieht mit der if else Anweisung. Es wird dann mit der Funktion arrayElemAt entweder auf den away\_club oder den home\_club namen zugegriffen.

4. In der letzten Aggregation Stufe werden die einzelnen Ergebnisse dann gruppiert und gezählt und dann als count ausgegeben
5. Schlussendlich wird dann wiederum absteigen sortiert.

Resultat:

```
[
  {
    _id: {
      formation: '4-3-3 Attacking',
      club_name: 'Rooms Katholieke Combinatie Waalwijk'
    },
    count: 24
  },
  {
    _id: {
      formation: '4-3-3 Defending',
      club_name: 'Rooms Katholieke Combinatie Waalwijk'
    },
    count: 17
  },
  {
    _id: {
      formation: '3-4-3',
      club_name: 'Rooms Katholieke Combinatie Waalwijk'
    },
    count: 4
  },
  {
    _id: {
      formation: '5-3-2',
      club_name: 'Rooms Katholieke Combinatie Waalwijk'
    },
    count: 3
  },
  {
    _id: {
      formation: '3-5-2',
      club_name: 'Rooms Katholieke Combinatie Waalwijk'
    },
    count: 3
  },
  {
    _id: {
      formation: '5-4-1',
      club_name: 'Rooms Katholieke Combinatie Waalwijk'
    },
    count: 2
  },
  {
    _id: {
      formation: '3-4-2-1',
      club_name: 'Rooms Katholieke Combinatie Waalwijk'
    },
    count: 2
  },
  {
    _id: {
      formation: '3-4-1-2',
      club_name: 'Rooms Katholieke Combinatie Waalwijk'
    },
    count: 1
  },
  {
    _id: {
      formation: '4-4-1-1',
      club_name: 'Rooms Katholieke Combinatie Waalwijk'
    },
    count: 1
  }
]
```

	formation	count
▶	4-3-3 Attacking	24
	4-3-3 Defending	17
	3-4-3	4
	5-3-2	3
	3-5-2	3
	3-4-2-1	2
	5-4-1	2
	4-4-1-1	1
	3-4-1-2	1

Auch hier lässt sich feststellen, dass die Resultate übereinstimmen.

#### **5.3.4 Diskussion der Resultate**

Es wurden jetzt also drei konkrete use cases unserer Datenbankanwendung herausgearbeitet.

1. Es kann nach beliebigen Spielern mit vordefinierten Charakteristika gesucht werden. Zu diesen Spielern, kann eine grosse Bandbreite an Zusatzinformationen abgefragt werden. Dies hilft einem Verein den optimalen, neuen Spieler für sich zu finden. In unserem Resultat wollte der Coach unbedingt einen Schweizer Spieler der Innenverteidiger ist und mindestens einen Marktwert von einer Million besitzt. Dem Coach war auch die Grösse des Abwehrspielers wichtig, da er unbedingt eine bessere Kopfballabwehr benötigt. Trotzdem wollte er kleinere Spieler nicht Apriori ausschliessen, weshalb er die resultierenden Spieler lediglich der Grösse nach sortiert hat.
2. Es kann eine Liga nach Teams analysiert werden, welche immer spät im Spiel treffen. Mit diesem zusätzlichen Wissen kann der Coach eines Teams eine bessere Entscheidung in den entscheidenden letzten Spielminuten treffen. Er kann beispielsweise einen weiteren defensiven Spieler für einen offensiven Spieler einwechseln, um das Tor zu verhindern, oder der Mannschaft taktische Anweisungen geben, das Tor noch mehr zu verteidigen. In unserem Resultat sehen wir die aufgelisteten Teams, die immer spät treffen. Wenn ich also jetzt ein Coach eines anderen Teams bin und ich gegen Manchester City oder West Ham United spiele, kann ich dementsprechend handeln.
3. Beim dritten use case, werden die vergangenen Aufstellungen des Gegners analysiert und aggregiert. Mit diesem zusätzlichen Wissen kann der Coach eine bessere Entscheidung treffen, mit welcher Aufstellung er spielen möchte. Wenn also der Gegner wahrscheinlich mit nur 2 Mittelfeldspielern spielt, kann er die taktische Entscheidung treffen mit 3 Mittelfeldspielern zu spielen, um eine Überzahl herzustellen. In unserem Resultat spielt der Gegner also höchstwahrscheinlich in einer 4-3-3 Formation, da er bis jetzt in 71% der Fälle in einer solchen Formation gespielt hat. Der Trainer kann also jetzt eine bewusste Entscheidung auf Basis dieser Grundlage treffen.

## 6 Effizienz & Performance

### 6.1 Optimierung MongoDB

Um die Laufzeitoptimierung vorzunehmen, wird das Query zur Analyse der Aufstellung der gegnerischen Mannschaft herangezogen und dann optimiert. Es wird aber nur dieses eine Query exemplarisch beschleunigt.

Bei der Analyse des Query fällt auf, dass wir zwei lookups im ClubsDocument vollziehen. Lookups sind in MongoDB immer sehr ressourcenintensiv, da MongoDB eigentlich nicht darauf ausgelegt ist. Die Laufzeit der Querys sind optimaler, wenn alle notwendigen Daten in demselben Dokument zu finden sind.

Zudem fällt ein lookup besonders ins Gewicht, wenn viele Operationen durchgeführt werden müssen. In unserem Fall müssen  $17'458 \times 426$  (Länge der Dokumente GamesDocument und ClubsDocument) also knapp  $7'500'000$  Operationen durchgeführt werden. Dies ist schon eine anständige Menge. Die Laufzeit für dieses erste Query mit lookup beträgt knapp eine Sekunde (**862ms**) und wurde mit zwei Variablen start und end gemessen, wobei die erste zu Beginn des Dokuments definiert worden ist und die zweite am Ende des Dokuments. Den Variablen wurde jeweils mit der Funktion Date die jetzige Zeit zugewiesen.

```
scoutingAndAnalysis> db.GamesDocument.aggregate([
...   {
...     $match: {
...       $or: [
...         {
...           home_club_id: 235,
...           game_date: {
...             $gte: ISODate("2020-08-14T00:00:00.000Z"),
...             $lte: ISODate("2022-02-01T00:00:00.000Z")
...           }
...         },
...         {
...           away_club_id: 235,
...           game_date: {
...             $gte: ISODate("2020-08-14T00:00:00.000Z"),
...             $lte: ISODate("2022-02-01T00:00:00.000Z")
...           }
...         }
...       ]
...     }
...   },
...   {
...     $lookup: {
...       from: "ClubsDocument",
...       localField: "home_club_id",
...       foreignField: "club_id",
...       as: "home_club_info"
...     }
...   },
...   {
...     $lookup: {
...       from: "ClubsDocument",
...       localField: "away_club_id",
...       foreignField: "club_id",
...       as: "away_club_info"
...     }
...   },
...   {
...     $project: {
...       formation: {
...         $cond: {
...           if: { $eq: ["$home_club_id", 235] },
...           then: "$home_club_formation",
...           else: "$away_club_formation"
...         }
...       },
...       club_name: {
...         $cond: {
...           if: { $eq: ["$home_club_id", 235] },
...           then: { $arrayElemAt: ["$home_club_info.club_name", 0] },
...           else: { $arrayElemAt: ["$away_club_info.club_name", 0] }
...         }
...       }
...     }
...   },
...   {
...     $group: {
...       _id: { formation: "$formation", club_name: "$club_name" },
...       count: { $sum: 1 }
...     }
...   },
...   {
...     $sort: { count: -1 }
...   }
... ]);
```

```
scoutingAndAnalysis> print("Laufzeit: " + (end - start) + "ms");
Laufzeit: 862ms
```

Der Query Plan des Querys wurde mit der explain Methode und dem Parameter "queryPlanner" extrahiert. Sie zeigt deutlich wie der Lookup eingeplant ist und das er auf das ClubsDocument ausgeführt wird.

Danach wird das Resultat mit der PROJECTION\_SIMPLE ausgegeben und zuletzt noch der COLLSACAN, welcher die Ergebnisse, innerhalb des *project*, anhand der home\_club\_id oder der away\_club\_id filtert. Zudem zeigt die COLLSCAN stage eindeutig, dass noch kein Index gebraucht wird und dadurch das ganze GamesDocument durchgegangen werden muss. Das group und sort bleiben dabei gleich wie oben.

```
winningPlan: {
  queryPlan: {
    stage: 'EQ_LOOKUP',
    planNodeId: 4,
    foreignCollection: 'scoutingAndAnalysis.ClubsDocument',
    localField: 'away_club_id',
    foreignField: 'club_id',
    asField: 'away_club_info',
    strategy: 'HashJoin',
    inputStage: {
      stage: 'EQ_LOOKUP',
      planNodeId: 3,
      foreignCollection: 'scoutingAndAnalysis.ClubsDocument',
      localField: 'home_club_id',
      foreignField: 'club_id',
      asField: 'home_club_info',
      strategy: 'HashJoin',
      inputStage: {
        stage: 'PROJECTION_SIMPLE',
        planNodeId: 2,
        transformBy: {
          _id: true,
          away_club_formation: true,
          away_club_id: true,
          home_club_formation: true,
          home_club_id: true
        },
        inputStage: {
          stage: 'COLLSCAN',
          planNodeId: 1,
          filter: {
```

Mit der optimierten Variante des Query's wo kein lookup gemacht wird, sondern die bereits in GamesDocument enthaltenen Informationen zu den Clubs genutzt werden, ist das query beträchtlich schneller. Es dauert jetzt nur noch **417ms**. Die runtime wurde also halbiert.

```
scoutingAndAnalysis> db.GamesDocument.aggregate([
...   {
...     $match: {
...       $or: [
...         { home_club_id: 235 },
...         { away_club_id: 235 }
...       ],
...       game_date: {
...         $gte: ISODate("2020-08-14T00:00:00.000Z"),
...         $lte: ISODate("2022-02-01T00:00:00.000Z")
...       }
...     }
...   },
...   {
...     $project: {
...       formation: {
...         $cond: {
...           if: { $eq: ["$home_club_id", 235] },
...           then: "$home_club_formation",
...           else: "$away_club_formation"
...         }
...       },
...       club_name: {
...         $cond: {
...           if: { $eq: ["$home_club_id", 235] },
...           then: { $arrayElemAt: ["$Home_Club.club_name", 0] },
...           else: { $arrayElemAt: ["$Away_Club.club_name", 0] }
...         }
...       }
...     }
...   },
...   {
...     $group: {
...       _id: { formation: "$formation", club_name: "$club_name" },
...       count: { $sum: 1 }
...     }
...   },
...   {
...     $sort: { count: -1 }
...   }
... ]);

scoutingAndAnalysis> print("Laufzeit: " + (end - start) + "ms");
Laufzeit: 417ms
```

Auch beim query Plan fällt auf, dass der zu Beginn eingeplante lookup von vorher wegfällt und direkt die PROJECTION angewendet wird.

```
winningPlan: {
  stage: 'PROJECTION_DEFAULT',
  transformBy: {
    _id: true,
    formation: {
      '$cond': [
        { '$eq': [ '$home_club_id', { '$const': 235 } ] },
        '$home_club_formation',
        '$away_club_formation'
      ]
    },
    club_name: {
      '$cond': [
        { '$eq': [ '$home_club_id', { '$const': 235 } ] },
        {
          '$arrayElemAt': [ '$Home_Club.club_name', { '$const': 0 } ]
        },
        {
          '$arrayElemAt': [ '$Away_Club.club_name', { '$const': 0 } ]
        }
      ]
    }
  },
  inputStage: {
```

Zusätzlich wird ein Index im GamesDocument auf die Felder home\_club\_id, away\_club\_id gesetzt. Mit diesem Index sollte die Ausführung noch einmal schneller von Statten gehen, da die entsprechenden id's schneller gefunden werden können.

```
scoutingAndAnalysis> db.GamesDocument.createIndex({ home_club_id: 1 });
home_club_id_1
scoutingAndAnalysis> db.GamesDocument.createIndex({ away_club_id: 1 });
away_club_id_1
scoutingAndAnalysis> db.getKeys()
TypeError: db.getKeys is not a function
scoutingAndAnalysis> db.GamesDocument.getIndexes();
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { home_club_id: 1 }, name: 'home_club_id_1' },
  { v: 2, key: { away_club_id: 1 }, name: 'away_club_id_1' }
]
scoutingAndAnalysis>
```

Mit den neu gesetzten Indizes ist das query noch einmal ein bisschen schneller. Es dauert jetzt nur noch **332ms**. Auch im queryPlanner sehen wir, dass nicht mehr ein COLLSCAN, sondern ein IXSCAN ausgeführt wird.

```
scoutingAndAnalysis> print("Laufzeit: " + (end - start) + "ms");
Laufzeit: 332ms
```

Auch im query Planer sieht man, dass die Indizes erstellt wurden und jetzt ein IXSCAN, also ein IndexScan und nicht mehr ein COLLSCAN auf die beiden Felder home\_club\_id und away\_club\_id

angewendet wird. Dies ist auch gleich der Beweis, dass die Querys sargable sind, da der gesetzte Index verwendet wird. Man hätte auch direkt im Query gesehen, dass dieses sargable, weil keine speziellen Funktionen auf den Attributen verwendet werden.

```
inputStage: {
  stage: 'OR',
  inputStages: [
    {
      stage: 'IXSCAN',
      keyPattern: { home_club_id: 1 },
      indexName: 'home_club_id_1',
      isMultiKey: false,
      multiKeyPaths: { home_club_id: [] },
      isUnique: false,
      isSparse: false,
      isPartial: false,
      indexVersion: 2,
      direction: 'forward',
      indexBounds: { home_club_id: [ '[235, 235]' ] }
    },
    {
      stage: 'IXSCAN',
      keyPattern: { away_club_id: 1 },
      indexName: 'away_club_id_1',
      isMultiKey: false,
      multiKeyPaths: { away_club_id: [] },
      isUnique: false,
      isSparse: false,
      isPartial: false,
      indexVersion: 2,
      direction: 'forward',
      indexBounds: { away_club_id: [ '[235, 235]' ] }
    }
  ]
},
```

Schlussendlich kann eine materialized view erstellt werden, welche das query nochmals beschleunigt. Das nachfolgende MongoDB statement erstellt diese materialized\_view. Diese ist nichts anderes als eine Vor-Selektion der Daten. Im zweiten query werden dann nur noch die die Resultate gruppiert, gezählt und sortiert. Die query runtime kann damit nochmals knapp halbiert werden und auf **173ms**. Wichtig zu erwähnen ist auch noch, dass die Resultate einzelne Werte sind und schwanken können. Der Trend im

Allgemeinen über die vier verschiedenen querys ist allerdings klar und konsistent.

```
scoutingAndAnalysis> db.GamesDocument.aggregate([
...   {
...     $match: {
...       $or: [
...         { home_club_id: 235 },
...         { away_club_id: 235 }
...       ],
...       game_date: {
...         $gte: ISODate("2020-08-14T00:00:00.000Z"),
...         $lte: ISODate("2022-02-01T00:00:00.000Z")
...       }
...     }
...   },
...   {
...     $project: {
...       game_id: 1, // Hinzufügen des game_id für spätere Verknüpfung
...       formation: {
...         $cond: {
...           if: { $eq: ["$home_club_id", 235] },
...           then: "$home_club_formation",
...           else: "$away_club_formation"
...         }
...       },
...       club_name: {
...         $cond: {
...           if: { $eq: ["$home_club_id", 235] },
...           then: { $arrayElemAt: ["$Home_Club.club_name", 0] },
...           else: { $arrayElemAt: ["$Away_Club.club_name", 0] }
...         }
...       }
...     }
...   },
...   {
...     $out: "MaterializedViewGames"
...   }
... ]);

scoutingAndAnalysis>
```

Die Abfrage des Query wird folgendermassen abgehandelt. Zuerst wird die Start Variable gesetzt, dann der Zugriff auf die view gemacht, das Resultat ausgegeben, die endVariable gesetzt und dann die Zeit

ausgegeben.

```
scoutingAndAnalysis> var start = new Date();

scoutingAndAnalysis> db.MaterializedViewGames.aggregate([
...   {
...     $group: {
...       _id: { formation: "$formation", club_name: "$club_name" },
...       count: { $sum: 1 }
...     }
...   },
...   {
...     $sort: { count: -1 }
...   }
... ]);
```

```
scoutingAndAnalysis> var end = new Date();

scoutingAndAnalysis> var duration = end - start;

scoutingAndAnalysis> print("Abfragezeit: " + duration + " ms");
Abfragezeit: 173 ms
```

## 6.2 Optimierung SQL

Auch hier wird nur das Query zur Analyse der gegnerischen Aufstellung beschleunigt.

Um die Laufzeitoptimierung in SQL vorzunehmen, können Indizes auf die relevanten columns gesetzt werden. In unserem Beispiel ist es so, dass die relevanten Indizes bereits gesetzt worden sind. Die Indizes auf home\_club\_id und away\_club\_id wurden nicht explizit gesetzt, sondern wurden automatisch mit dem Einfügen des foreign key constraint hinzugefügt. Da wir in unserem Schema eine eigene Relation für das away\_team und dem home\_team erstellt haben, erübrigt sich das Setzen dieser Indizes. Die Indizes können auch nicht gelöscht werden, ohne den foreign key constraint zu löschen, weshalb keine Zeiten ohne Index vorliegen. Die Abfragen ohne Index wären aber eindeutig langsamer als mit Index. Wir überprüfen zusätzlich noch, ob das query schneller ist mit einem Index auf der game\_date column.

Zudem machen wir auch eine materialized view um das query noch mehr zu beschleunigen.

Um die runtime zu messen, wird das performance\_schema aktiviert, die Historie dessen gelöscht und die interessanten columns aus der Tabelle events\_statements\_history im performance\_schema ausgelesen. Vor dem Auslesen der runtime wird das query logischerweise noch ausgeführt.

Das unoptimierte query dauert ziemlich unkonstant lange, weshalb ein Durchschnitt aus 5 Werten genommen wird (4.9ms, 2.8ms, 9.2ms, 9.3ms, 5.4ms). Der Durchschnitt dieser Werte liegt also bei **6.32ms**

```

1   -- 1. Enable query monitoring
2 • UPDATE performance_schema.setup_consumers SET ENABLED = 'YES' WHERE NAME = 'events_statements_history';
3   -- 2. Clear the history
4 • TRUNCATE TABLE performance_schema.events_statements_history;
5   -- 3. Execute the query
6 • SELECT formation, COUNT(*) as count
7   FROM (
8     SELECT home_clubFormation as formation
9       FROM games
10      WHERE (home_club_id = 235)
11        AND game_date BETWEEN '2020-08-14' AND '2022-02-01'
12    UNION ALL
13    SELECT away_clubFormation as formation
14      FROM games
15      WHERE (away_club_id = 235)
16        AND game_date BETWEEN '2020-08-14' AND '2022-02-01'
17  ) as homeAndAwayGames
18 GROUP BY formation
19 ORDER BY count(*) DESC;
20   -- 4. Check the query time
21 • SELECT event_name,
22       timer_start,
23       timer_end,
24       timer_wait / 100000000000 AS duration_seconds,
25       sql_text
26     FROM performance_schema.events_statements_history
27   WHERE sql_text LIKE 'SELECT formation, COUNT(*) as count%';

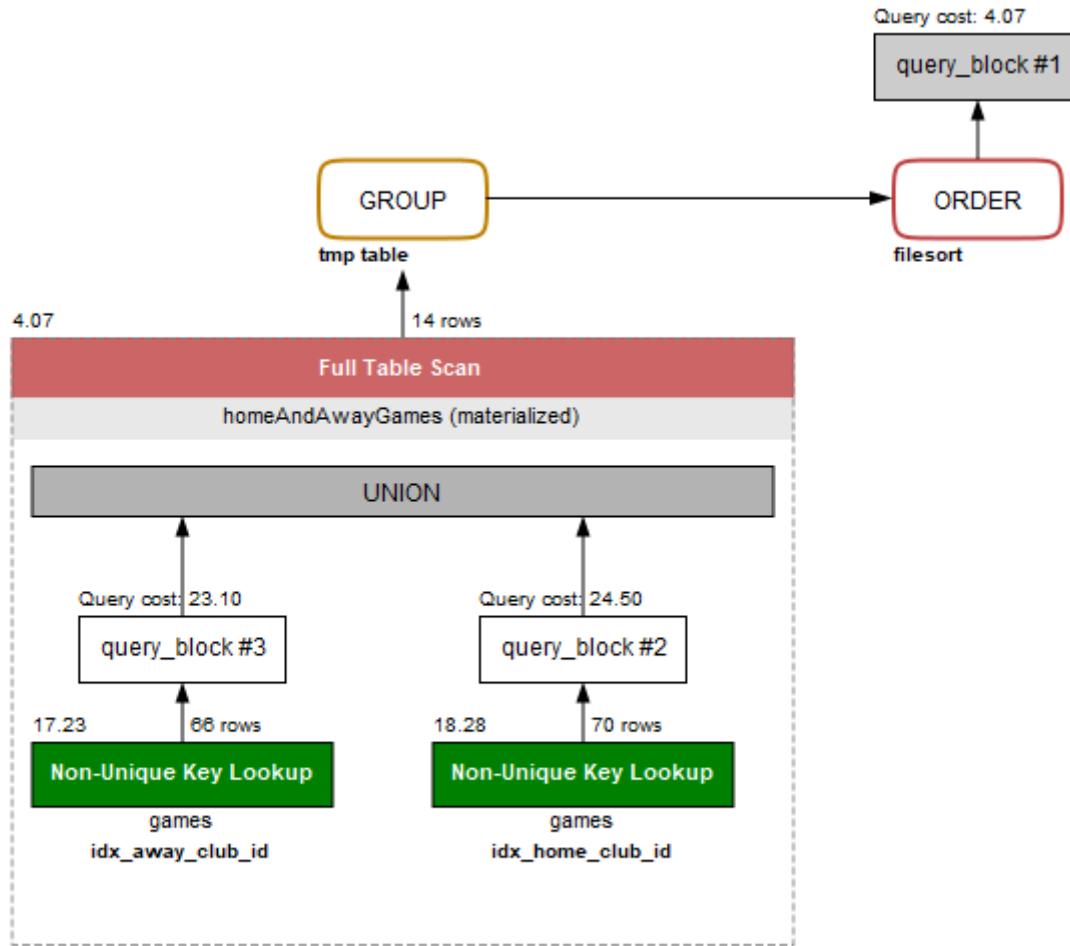
```

Result Grid				
	event_name	timer_start	timer_end	duration_seconds
statement/sql/select	873874938330480	873881983077685	0.0070	SELECT formation, COUNT(*) as count FROM ( ...

Output		
#	Action	Message
1	12:15:48 UPDATE performance_schema.setup_consumers SET ENABLED = 'YES' WHERE NAME = 'events_statements_history'	0 row(s) affected Rows matched: 1 Changed: 0 Warnings: 0
2	12:15:48 TRUNCATE TABLE performance_schema.events_statements_history	0 row(s) affected
3	12:15:48 SELECT formation, COUNT(*) as count FROM ( SELECT home_clubFormation as formation FROM games WHERE (home_club_id = 235) AND game_date BETWEEN '2020-08-14' AND '2022-02-01' ) as homeAndAwayGames GROUP BY formation ORDER BY count(*) DESC;	9 row(s) returned
4	12:15:48 SELECT event_name, timer_start, timer_end, timer_wait / 100000000000 AS duration_seconds, sql_text FROM performance_schema.events_statements_history WHERE sql_text LIKE 'SELECT formation, COUNT(*) as count%';	1 row(s) returned

Der execution Plan zum ersten Query sieht folgendermassen aus. Es werden zwei Abfragen auf den games table gemacht und die beiden Indizes dazu genutzt. Auch die UNION sieht man eindeutig, genauso wie

das GROUP und ORDER Keyword. Es ist auch ersichtlich, dass der idx\_home bzw. away\_club\_id genutzt wird. Er wird unterhalb der games Beschriftung angegeben.



In einem nächsten Schritt werden die Indizes auf der betroffenen column gesetzt. Die Indizes auf away\_club\_id und home\_club\_id wurden bereits gesetzt. Als diese Indizes gesetzt wurden, war uns noch nicht klar, dass bei einem foreign key constraint, automatisch ein Index gesetzt wird. Der andere Index auf game\_date wurde allerdings gesetzt.

```

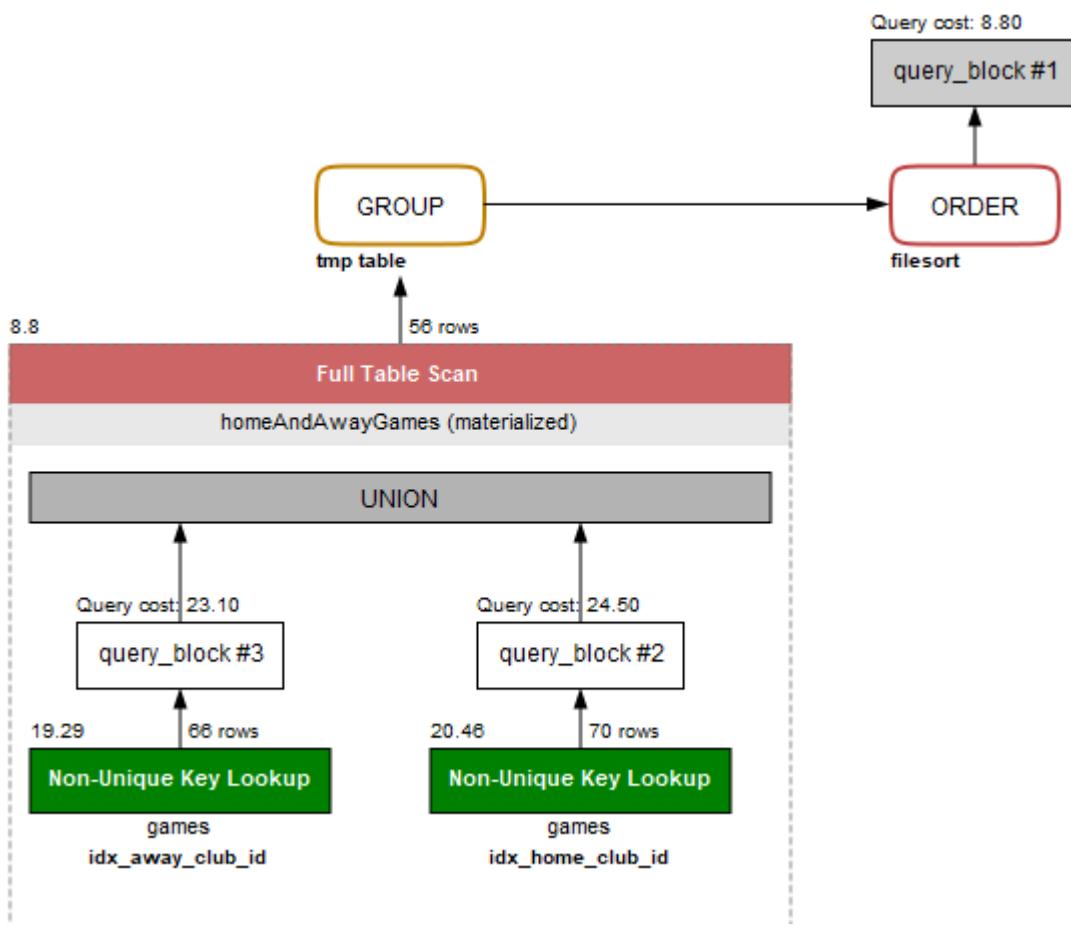
11 • CREATE INDEX idx_game_date ON scoutingandanalysis.games(game_date);
12
  
```

Output		
Action Output		Message
#	Time	Action
1	12:36:50	CREATE INDEX idx_game_date ON scoutingandanalysis.games(game_date)

0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0

Execution Plan nach dem Setzen des game\_date Indizes. Beim Betrachten wird klar, dass die Query cost beim query\_block #3 und #4 genau gleichbleibt. Es werden aber mehr rows beim table scan geschätzt,

als vor dem Setzen des Index game\_date.



Hier ist zusätzlich ersichtlich, dass der Index game\_date ebenfalls genutzt wird. Dies ist eine alternative Darstellung des query Execution Plans.

1	PRIMARY	<derived2>	NULL	ALL	NULL	NULL	NULL	NULL	56	100.00	Using temporary; Using filesort
2	DERIVED	games	NULL	ref	idx_home_club_id, idx_game_date	idx_home_club_id	5	const	70	42.30	Using where
3	UNION	games	NULL	ref	idx_away_club_id, idx_game_date	idx_away_club_id	5	const	66	42.30	Using where

Die runtime wurde wiederum gemessen (6.5ms, 8.4ms, 7.6ms, 3.8ms, 6.4ms) und ist im Durchschnitt **6.54ms**. Dies zeigt deutlich, dass das query sogar ein bisschen langsamer ist als ohne Index auf game\_date. Dies kann unter anderem daran liegen, dass der Optimizer die Vorteile des neuen Indizes falsch bewertet, oder dass game\_date viele Duplikate oder ähnliche Werte besitzt, was vermutlich der Fall ist, weil überwiegend an Wochenenden gespielt wird. Demzufolge gibt es viele Einträge mit denselben Werten. Zusätzlich wirkt sich der Overhead eines zusätzlichen Index eventuell auch noch negativ auf die Laufzeit aus.

event_name	timer_start	timer_end	duration_seconds	sql_text
statement/sql/select	12870277533003450	12870281946691085	0.0044	SELECT formation, COUNT(*) as count FROM ...

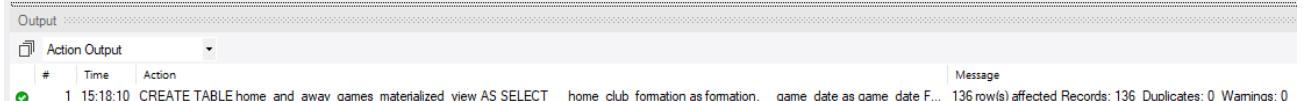
Result 105			Result 106					
Output								
Action Output								
#	Time	Action						
1	15:22:04	UPDATE performance_schema.setup_consumers SET ENABLED = 'YES' WHERE NAME = 'events_statements_history'						
2	15:22:04	TRUNCATE TABLE performance_schema.events_statements_history						
3	15:22:04	SELECT formation, COUNT(*) as count FROM home_and_away_games_materialized_view WHERE game_date BETWEEN '2020-08-14' AND ...						
4	15:22:04	SELECT event_name, timer_start, timer_end, timer_wait / 100000000000 AS duration_seconds, sql_text FROM performa...						
Message								
0 row(s) affected Rows matched: 1 Changed: 0 Warnings: 0								
0 row(s) affected								
9 row(s) returned								
1 row(s) returned								

Schlussendlich wird die materialized view erstellt und auf diese view dann das query ausgeführt. Hier wird also ein zusätzlicher table erstellt mit der Vorauswahl der für uns interessanten Daten. In diesen table werden also die Daten zu einem spezifischen club mit club\_id 235 geladen.

```

1 • CREATE TABLE home_and_away_games_materialized_view AS
2   SELECT
3     home_clubFormation AS formation,
4     game_date AS game_date
5   FROM
6     games
7   WHERE
8     home_club_id = 235
9 UNION ALL
10  SELECT
11    away_clubFormation AS formation,
12    game_date AS game_date
13  FROM
14    games
15  WHERE
16    away_club_id = 235;

```

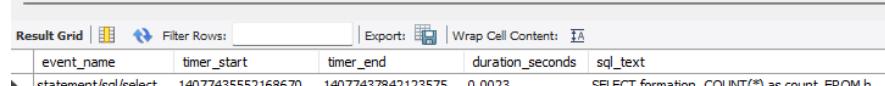
Output  
Action Output  
# Time Action  
1 15:18:10 CREATE TABLE home\_and\_away\_games\_materialized\_view AS SELECT home\_clubFormation as formation, game\_date as game\_date F... 136 row(s) affected Records: 0 Warnings: 0

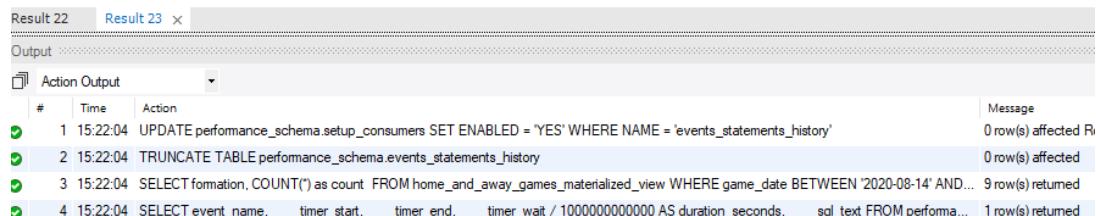
Die runtime des querys die direkt auf die View ausgeführt wird, ist im Durchschnitt tiefer. Die durchschnittliche Zeit des query liegt bei **2.5ms**. Dies entspricht wiederum einer Reduktion um ca. 60% im Vergleich zur ersten Messung. Hier wird also eine eindeutige Verbesserung festgestellt.

```

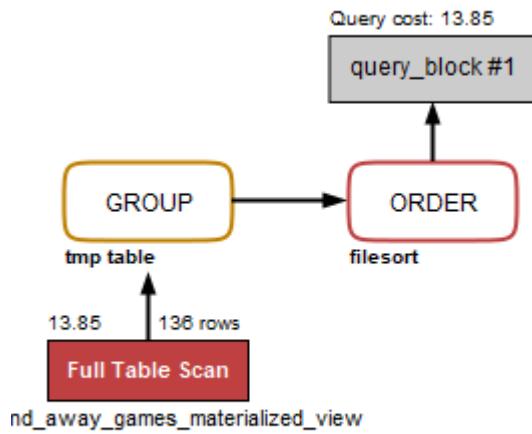
1 -- 1. Aktivieren der Überwachung von Abfragen
2 • UPDATE performance_schema.setup_consumers SET ENABLED = 'YES' WHERE NAME = 'events_statements_history';
3
4 -- 2. Leere die Historie
5 • TRUNCATE TABLE performance_schema.events_statements_history;
6
7 • SELECT formation, COUNT(*) AS count
8   FROM home_and_away_games_materialized_view
9  WHERE game_date BETWEEN '2020-08-14' AND '2022-02-01'
10 GROUP BY formation
11 ORDER BY count(*) DESC;
12
13 -- 4. Überprüfe die Abfragezeit
14 • SELECT event_name,
15   timer_start,
16   timer_end,
17   timer_wait / 1000000000000 AS duration_seconds,
18   sql_text
19   FROM performance_schema.events_statements_history
20  WHERE sql_text LIKE 'SELECT formation, COUNT(*) AS count%';

```

Result Grid | Filter Rows: Export: Wrap Cell Content:  
event\_name timer\_start timer\_end duration\_seconds sql\_text  
statement/sql/select 14077435552168670 14077437842123575 0.0023 SELECT formation, COUNT(\*) AS count FROM h...

Result 22 Result 23 x  
Output  
Action Output  
# Time Action Message  
1 15:22:04 UPDATE performance\_schema.setup\_consumers SET ENABLED = 'YES' WHERE NAME = 'events\_statements\_history' 0 row(s) affected R  
2 15:22:04 TRUNCATE TABLE performance\_schema.events\_statements\_history 0 row(s) affected  
3 15:22:04 SELECT formation, COUNT(\*) AS count FROM home\_and\_away\_games\_materialized\_view WHERE game\_date BETWEEN '2020-08-14' AND... 9 row(s) returned  
4 15:22:04 SELECT event\_name, timer\_start, timer\_end, timer\_wait / 1000000000000 AS duration\_seconds, sql\_text FROM performa... 1 row(s) returned

Bei der Analyse des execution Plan sieht man auch klar, dass keine Union mehr gibt und die Daten direkt aus dem view table extrahiert werden. Ansonsten bleibt das query gleich. Es wird auch gruppiert und dann sortiert.



Zusätzlich gilt auch zu erwähnen, dass die ersten beiden querys sargable sind und die gesetzten Indizes nutzen können. Dies sieht man auch an den Screenshots der beiden ersten querys von oben. Die fetten schwarzen Indizes unterhalb der games table bestätigen, dass. Wir nutzen auch keine Funktion mehr, wie STR\_TO\_DATE welche das Verwenden eines Index ebenfalls eingeschränkt hätte.

## 7 Datenschutz & Datenbanksicherheit

### 7.1 Beschreibung der Sicherheitsziele

In diesem Abschnitt werden die drei Sicherheitsziele Vertraulichkeit, Integrität und Verfügbarkeit beschrieben und konkrete Sicherheitsrisiken bei unserem Projekt dargelegt.

#### Vertraulichkeit

Dieses Sicherheitsziel setzt sich mit dem Zugriff auf Daten auseinander. Es soll dabei nicht möglich sein, für einen User auf Informationen zuzugreifen, welche er nicht sehen darf, oder nicht sehen muss. Man spricht dabei auch oft von Need-to-Know Prinzip.

Konkrete Sicherheitsschwächen in unserem Projekt bezüglich der Vertraulichkeit sind:

1. Keine oder schlechte Berechtigungsvergaben innerhalb der Datenbank. Es gibt keine Abstufung an User-Profilen mit verschiedenen Zugriffsrechten. Bei der Vertraulichkeit geht es dabei eher um die Einsicht in Informationen, also um Leserechte.
2. Schwache oder keine Authentifizierungsmethoden. Wenn keine oder nur schlechte Passwörter beim Zugriff auf die Datenbank abgefragt werden, kann ein Angreifer leichter auf die Datenbank zugreifen und damit die Vertraulichkeit verletzen.

## **Verfügbarkeit**

Dieses Sicherheitsziel beschäftigt sich damit, wie zeitlich konstant der Datenbankservice angeboten werden kann und wie schnell, er bei einem Angriff oder bei einer defekten Hardware wieder angeboten werden kann.

Konkrete Sicherheitsschwächen in unserem Projekt, welche die Verfügbarkeit verletzen könnten, wären:

1. Es werden keine Backups gemacht, was in einem Katastrophenfall zu einer Verletzung der Verfügbarkeit führen kann.

## **Integrität**

Dieses Sicherheitsziel beschäftigt sich damit, wie das die Daten integer sind. Es stellt somit sicher, dass die Daten korrekt und unverändert bleiben.

Konkrete Sicherheitsschwächen in unserem Projekt wären:

1. Eine falsche Berechtigungsvergabe, welche Schreibrechte an User verteilt, welche die Daten nicht verändern sollten, können.
2. Backups die nicht ausreichend geschützt sind, können von einem Angreifer verändert werden.

## **7.2 Massnahmen**

### **7.2.1 Definition Massnahmen Verfügbarkeit SQL**

Es wird ein Backup der ganzen Datenbank benötigt, um diese wiederherstellen zu können. Dies wird in unserem Projekt auf der lokalen Maschine mit mysqldump erstellt und danach auf eine andere Maschine kopiert. Dies geschieht alles manuell, kann aber im Falle eines regelmässigen Backups auch automatisiert werden.

### **7.2.2 Umsetzung Massnahmen Verfügbarkeit SQL**

#### **7.2.2.1 Backup mit mysqldump**

Um ein Backup der Datenbank scoutingandanalysis zu erstellen, wird folgendermassen vorgegangen.

1. Log-in via ssh auf die VM:  

```
Copyright (C) Microsoft Corporation. Alle Rechte vorbehalten.

Installieren Sie die neueste PowerShell für neue Funktionen und Verbesserungen! https://aka.ms/PSWindows

PS C:\Users\fabia> ssh dbsstudent@86.119.35.98
dbsstudent@86.119.35.98's password:
Microsoft Windows [Version 10.0.20348.2113]
(c) Microsoft Corporation. All rights reserved.

dbsstudent@DBS-21 C:\Users\dbsstudent>
```
2. Erstellen von zwei Dateien mithilfe eines bat backupscript's. Die erste Datei ist für das Erstellen der Struktur mit CREATE commands verantwortlich und die zweite beinhaltet die reinen Daten der Datenbank. Die Option -t im ersten Befehl sagt aus, dass nur die Daten gespeichert werden soll und die Option -d im zweiten Befehl erstellt das Skript zur Kreierung der tables. Bei beiden Befehlen wird mit dem admin user auf die Mysql Datenbank eingeloggt und auf die Datenbank zugegriffen. Der Output-Pfad, der sich auf der VM befindet, wird dann auch angegeben.

#### **backupscript.bat**



```
backup.bat - Notepad
File Edit Format View Help
@echo off

"C:\Program Files\MySQL\MySQL Server 8.0\bin\mysqldump.exe" -t -u admin -p"HPgr_9$5" scoutingandanalysis > "C:\Users\dbsstudent\Desktop\insert.sql"
"C:\Program Files\MySQL\MySQL Server 8.0\bin\mysqldump.exe" -d -u admin -p"HPgr_9$5" scoutingandanalysis > "C:\Users\dbsstudent\Desktop\create.sql"
```

Manuelles Ausführen des backupscripts.bat auf der VM

```
dbsstudent@DBS-21 C:\Users\dbsstudent\Desktop>backup.bat
mysqldump: [Warning] Using a password on the command line interface can be insecure.
mysqldump: [Warning] Using a password on the command line interface can be insecure.

dbsstudent@DBS-21 C:\Users\dbsstudent\Desktop>
```

3. Log-out von der VM
4. Kopieren der Daten von der VM auf das lokale Gerät mit scp. Mit scp werden beide files auf das lokale Gerät kopiert.

Diese Befehle werden auf dem Gerät ausgeführt, wo das Backup gespeichert werden soll.

```
PS C:\Users\fabia> scp dbsstudent@86.119.35.98:"C:/Users/dbsstudent/Desktop/insert.sql" "C:/Users/fabia/Desktop/Screenshots und Querys MySQL/SQL Security/insert.sql"
dbsstudent@86.119.35.98's password:
insert.sql                                                 100%   60MB  10.9MB/s  00:05
PS C:\Users\fabia> scp dbsstudent@86.119.35.98:"C:/Users/dbsstudent/Desktop/create.sql" "C:/Users/fabia/Desktop/Screenshots und Querys MySQL/SQL Security/create.sql"
dbsstudent@86.119.35.98's password:
create.sql                                              100%   16KB  1.0MB/s  00:00
PS C:\Users\fabia>
```

Zusätzlich könnten bei regelmässigen Backups das erste .bat file auf der VM periodisch und automatisch ausgeführt werden. Auch die scp Befehle könnte automatisiert passieren, sodass die Backups komplett automatisch ablaufen.

Schlussendlich wurden die beiden erstellen Dateien auch dazu benutzt ein echtes Backup wieder hochzuladen. Hier wurde der mysql Befehl genutzt, um die Backup Datenbank zu erstellen mit dem create.sql file. Danach wurden die Daten mit dem insertfile in die Datenbank geladen. Zweimal musste das Passwort angegeben werden. Das Backup ist auch in mysql ersichtlich. Es hätte logischerweise auch auf einem anderen MySQL Server wiederhergestellt werden können.

```
C:\windows\system32>mysql -u admin -p scoutingandanalysisBackupRestore < "C:\Users\dbsstudent\Desktop\create.sql"
Enter password: *****

C:\windows\system32>mysql -u admin -p scoutingandanalysisBackupRestore < "C:\Users\dbsstudent\Desktop\insert.sql"
Enter password: *****

C:\windows\system32>
```

Danach wurde eine Stichprobe der Anzahl Einträge im table club\_has\_domestic\_competition abgefragt und mit der Originaldatenbank verglichen und festgestellt, dass die Anzahl Einträge übereinstimmen. Die Daten wurden also alle geladen.

```
1 •  select count(*) from scoutingandanalysisbackuprestore.club_has_domestic_competition;
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	count(*)			
▶	426			

```
1 • select count(*) from scoutingandanalysis.club_has_domestic_competition;
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	count(*)			
▶	426			

### 7.2.3 Umsetzung Massnahmen Verfügbarkeit MongoDB

Um in MongoDB die Verfügbarkeit sicherzustellen, muss ein Backup der Daten gemacht werden. Dieses Backup wird auf einer Backup Maschine gemacht, derselben auf welcher bereits das SQL-Backup gemacht wurde. Im Gegensatz zu SQL wird das Backup direkt auf diese Maschine exportiert, und zwar mit mongodump. Es existiert also zu keiner Zeit ein lokales Backup auf der VM. Um den mongodump Befehl auszuführen, muss die BackupMaschine also im HSLU-Netzwerk sein, die IP der VM, die betroffene Datenbank und mit der out option der Ort auf der Backup-Maschine angegeben werden. Den Fortschritt des Backups sieht man direkt in der geöffneten shell.

```
C:\Users\fabia>mongodump --host 86.119.35.98 --port 27017 --db scoutingAndAnalysis --out C:/Users/fabia/Desktop/MongoDB_Screenshots/Backups/BackupProject  
2024-05-23T13:12:00.918+0200 [#####] scoutingAndAnalysis.Players_raw 15348/15348 (100.0%)  
2024-05-23T13:12:01.378+0200 done dumping scoutingAndAnalysis.Players_raw (15348 documents)  
2024-05-23T13:12:01.434+0200 writing scoutingAndAnalysis.Clubs_raw to C:/Users/fabia/Desktop/MongoDB_Screenshots/Backups/BackupProject/scoutingAndAnalysis\Clubs  
2024-05-23T13:12:01.845+0200 done dumping scoutingAndAnalysis.Clubs_raw (426 documents)  
2024-05-23T13:12:01.912+0200 writing scoutingAndAnalysis.MaterializedViewFormation to C:/Users/fabia/Desktop/MongoDB_Screenshots/Backups/BackupProject\scoutingAndAnalysis\MaterializedViewFormation  
2024-05-23T13:12:02.117+0200 done dumping scoutingAndAnalysis.MaterializedViewFormation (57 documents)  
2024-05-23T13:12:02.181+0200 writing scoutingAndAnalysis.Competitions_raw to C:/Users/fabia/Desktop/MongoDB_Screenshots/Backups/BackupProject\scoutingAndAnalysis\Competitions  
2024-05-23T13:12:02.346+0200 done dumping scoutingAndAnalysis.Competitions_raw (43 documents)  
2024-05-23T13:12:03.905+0200 [#####] scoutingAndAnalysis.PlayerEvents_raw 377365/474725 (79.5%)  
2024-05-23T13:12:03.965+0200 [#####] scoutingAndAnalysis.PlayersDocument 5706/15348 (37.2%)  
2024-05-23T13:12:03.965+0200 [.....] scoutingAndAnalysis.ClubsDocument 0/426 (0.0%)  
2024-05-23T13:12:03.966+0200 [#####] scoutingAndAnalysis.PlayerEvents_raw 474725/474725 (100.0%)  
2024-05-23T13:12:06.146+0200 done dumping scoutingAndAnalysis.PlayerEvents_raw (474725 documents)  
2024-05-23T13:12:06.987+0200 [#####] scoutingAndAnalysis.PlayersDocument 7809/15348 (50.9%)  
2024-05-23T13:12:06.987+0200 [.....] scoutingAndAnalysis.ClubsDocument 0/426 (0.0%)  
2024-05-23T13:12:09.909+0200 [#####] scoutingAndAnalysis.PlayersDocument 7809/15348 (50.9%)  
2024-05-23T13:12:09.909+0200 [.....] scoutingAndAnalysis.ClubsDocument 0/426 (0.0%)  
2024-05-23T13:12:09.909+0200 [#####] scoutingAndAnalysis.PlayersDocument 7809/15348 (50.9%)  
2024-05-23T13:12:12.911+0200 [#####] scoutingAndAnalysis.ClubsDocument 101/426 (23.7%)  
2024-05-23T13:12:12.911+0200 [.....] scoutingAndAnalysis.ClubsDocument 101/426 (23.7%)  
2024-05-23T13:12:21.918+0200 [#####] scoutingAndAnalysis.PlayersDocument 7809/15348 (50.9%)  
2024-05-23T13:12:21.918+0200 [.....] scoutingAndAnalysis.ClubsDocument 101/426 (23.7%)  
2024-05-23T13:12:21.918+0200 [#####] scoutingAndAnalysis.PlayersDocument 11306/15348 (73.7%)  
2024-05-23T13:12:21.918+0200 [.....] scoutingAndAnalysis.ClubsDocument 183/426 (43.0%)  
2024-05-23T13:12:24.907+0200 [#####] scoutingAndAnalysis.PlayersDocument 11306/15348 (73.7%)  
2024-05-23T13:12:24.907+0200 [.....] scoutingAndAnalysis.ClubsDocument 183/426 (43.0%)  
2024-05-23T13:12:21.918+0200 [#####] scoutingAndAnalysis.PlayersDocument 11306/15348 (73.7%)  
2024-05-23T13:12:21.918+0200 [.....] scoutingAndAnalysis.ClubsDocument 183/426 (43.0%)  
2024-05-23T13:12:24.918+0200 [#####] scoutingAndAnalysis.PlayersDocument 11306/15348 (73.7%)  
2024-05-23T13:12:24.918+0200 [.....] scoutingAndAnalysis.ClubsDocument 183/426 (43.0%)  
2024-05-23T13:12:27.966+0200 [#####] scoutingAndAnalysis.PlayersDocument 11306/15348 (73.7%)  
2024-05-23T13:12:27.966+0200 [.....] scoutingAndAnalysis.ClubsDocument 183/426 (43.0%)  
2024-05-23T13:12:27.966+0200 [#####] scoutingAndAnalysis.PlayersDocument 11306/15348 (73.7%)  
2024-05-23T13:12:27.966+0200 [.....] scoutingAndAnalysis.ClubsDocument 183/426 (43.0%)  
2024-05-23T13:12:30.919+0200 [#####] scoutingAndAnalysis.PlayersDocument 11306/15348 (73.7%)  
2024-05-23T13:12:30.919+0200 [.....] scoutingAndAnalysis.ClubsDocument 183/426 (43.0%)  
2024-05-23T13:12:31.319+0200 [#####] scoutingAndAnalysis.ClubsDocument 426/426 (100.0%)  
2024-05-23T13:12:31.319+0200 done dumping scoutingAndAnalysis.ClubsDocument (426 documents)  
2024-05-23T13:12:31.319+0200 [#####] scoutingAndAnalysis.PlayersDocument 15348/15348 (100.0%)  
2024-05-23T13:12:32.081+0200
```

Um sicherzustellen das wir das Backup auch wieder auf der DB herstellen können nutzen wir mongorestore. Beim restore werden neben den parameter von vorher auch ein username und das Password des usernames angegeben. Aus dem zweiten Screenshot kann festgestellt werden, dass der restore funktioniert hat.

```
C:\Users\fabia>mongorestore --host 86.119.35.98 --port 27017 --username adminUser --password adminPassword --authenticationDatabase admin --db scoutingAndAnalysisRestore C:/Users/fabia/Desktop/MongoDB_Screenshots/Backups/BackupProject/scoutingAndAnalysis
2024-05-23T13:36:35.730+0200      734270 document(s) restored successfully. 0 document(s) failed to restore.
```

```
C:\Users\fabia>
```

Auch beim Befehl `db.stats()`, welcher die wichtigsten Kenngrößen der Datenbank ausgibt, lässt sich feststellen, dass der Parameter `dataSize` in beiden Datenbanken genau gleich gross ist. Es wird also in beiden Datenbanken genau gleich viel Speicher für die Daten verwendet.

```
scoutingAndAnalysis> db.stats()
{
  db: 'scoutingAndAnalysis',
  collections: Long('10'),
  views: Long('0'),
  objects: Long('734276'),
  avgObjSize: 524.3840980775622,
  dataSize: 385042658,
  storageSize: 788455424,
  indexes: Long('11'),
  indexSize: 7983104,
  totalSize: 796438528,
  scaleFactor: Long('1'),
  fsUsedSize: 25209233408,
  fsTotalSize: 53684989952,
  ok: 1
}
scoutingAndAnalysis> use scoutingAndAnalysisRestore
switched to db scoutingAndAnalysisRestore
scoutingAndAnalysisRestore> db.stats()
{
  db: 'scoutingAndAnalysisRestore',
  collections: Long('10'),
  views: Long('0'),
  objects: Long('734270'),
  avgObjSize: 524.3880806787694,
  dataSize: 385042436,
  storageSize: 106459136,
  indexes: Long('11'),
  indexSize: 8318976,
  totalSize: 114778112,
  scaleFactor: Long('1'),
  fsUsedSize: 25209233408,
  fsTotalSize: 53684989952,
  ok: 1
}
```

#### 7.2.4 Definition Massnahmen Vertraulichkeit

Die erste Sicherheitsschwachstelle wäre die fehlende oder unzureichende Abstufung von Rechten auf unserer Datenbank.

Um diesem Umstand entgegenzuwirken, definieren wir auf unserer Datenbank drei verschiedene User-Profile. Erstens der admin-Zugriff, welcher alle Aufgaben ausführen kann, die auf dem Datenbank-Server

ausgeführt werden können. Er soll Lesezugriff auf alle Datenbanken haben und auch von allen externen IP-Adressen erreichbar sein.

Zweitens der developer-Zugriff, welcher alle Operationen auf einer Datenbank ausführen können sollte. Der developer soll genau wie der admin Lesezugriff auf alle Datenbanken haben und auch von allen remote Hosts auf die Datenbank zugreifen können.

Drittens der Zugriff eines GuiUser über Metabase, welcher nur Lesezugriff auf die DB mit der extra dafür erstellten view erhalten soll.

Zusätzlich gilt zu erwähnen, dass der Zugriff nur im HSLU-Netzwerk möglich ist, da wir unseren Datenbank-Dienst nicht im Internet anbieten.

Die zweite Sicherheitsschwachstelle wäre die Wahl des Passwortes respektive das Definieren einer Passwortrichtlinie. Für Daten, die korrekt und zuverlässig geschützt werden sollen, muss ein Passwort mindestens eine Länge von 12 Zeichen besitzen, Gross – und Kleinbuchstaben, mindestens eine Zahl und ein Sonderzeichen besitzen. So wird der Raum an möglichen Passwörtern genügend gross, dass man sich gegen brute-force Attacken geschützt fühlen kann.

Für unser akademisches Projekt haben wir uns aber auf mindestens 8 Zeichen mit Sonderzeichen, Zahl, Gross – und Kleinbuchstaben beschränkt.

#### 7.2.4.1 Umsetzung Massnahmen Vertraulichkeit SQL

#### 7.2.4.2 Passwörter

Um das Setzen von Passwörtern auch einzufordern, wird ein Passwort-Policy-Plugin installiert und auch direkt aktiviert. Zudem wird mit der letzten Zeile die momentane Passwort-Policy ebenfalls angezeigt.

```
1 •  INSTALL COMPONENT 'file:///component_validate_password';
2 •  SET GLOBAL validate_password.policy = 1;
3 •  SHOW VARIABLES LIKE 'validate_password.%';
```

Die Ausgabe zeigt deutlich, dass unsere Passwort-Policy von oben für unser Projekt bereits erfüllt ist. Wenn also in Zukunft ein neuer User mit neuem Passwort erstellt wird, welcher dieser Policy nicht entspricht, wird eine Warnung angezeigt und das Erstellen des Users verhindert.

Variable_name	Value
validate_password.changed_characters_perce...	0
validate_password.check_user_name	ON
validate_password.dictionary_file	
validate_password.length	8
validate_password.mixed_case_count	1
validate_password.number_count	1
validate_password.policy	MEDIUM
validate_password.special_char_count	1

#### Ausgabe

2 16:57:47	INSTALL COMPONENT file:///component_validate_password'	0 row(s) affected	0.390 sec
3 16:57:47	SET GLOBAL validate_password.policy = 1	0 row(s) affected	0.047 sec
4 16:57:47	SHOW VARIABLES LIKE 'validate_password.%'	8 row(s) returned	0.015 sec / 0.000 sec

#### 7.2.4.3 User Privilegien

Nach dem Setzen der Policy können nun die User erstellt werden. Als erstes wird der admin User erstellt. Unter Server->Users and Privileges können einzelne User direkt erstellt werden. Es muss dabei der login

Name, der authentication type, die Daten der möglichen hosts von welchen dieser User zugreifen kann und das Passwort angegeben werden. Hier ein Screenshot zur Erstellung des admin Users.

Login Name: admin You may create multiple accounts with the same name to connect from different hosts.

Authentication Type: caching\_sha2\_password For the standard password and/or host based authentication, select 'Standard'.

Limit to Hosts Matching: % % and \_ wildcards may be used

Password: \*\*\*\*\* Type a password to reset it. Consider using a password with 8 or more characters with mixed case letters, numbers and punctuation marks.

Confirm Password: \*\*\*\*\* Enter password again to confirm.

Expire Password

Authentication String: \$A\$005\$&@□=□h]?□□Qi&Rn` C Authentication plugin specific parameters.

See the plugin documentation for valid values and details.

Zudem kann man im Reiter Administrative Roles ebenfalls die konkreten Rollen oder Privilegien begutachten und anpassen, welcher der admin User besitzen soll. Er hat alle Rollen, die zur Verfügung stehen und kann somit alles auf allen Datenbanken lesen

**Details for account admin@%**

Role Description

- DBA grants the rights to perform all tasks
- MaintenanceAdmin grants rights needed to maintain server
- ProcessAdmin rights needed to assess, monitor, and kill any user proce...
- UserAdmin grants rights to create users logins and reset passwords
- SecurityAdmin rights to manage logins and grant and revoke server an...
- MonitorAdmin minimum set of rights needed to monitor server
- DBManager grants full rights on all databases
- DBDesigner rights to create and reverse engineer any database sche...
- ReplicationAdmin rights needed to setup and manage replication
- BackupAdmin minimal rights needed to backup any database

Global Privileges

- ALTER
- ALTER ROUTINE
- CREATE
- CREATE ROUTINE
- CREATE TABLESPACE
- CREATE TEMPORARY TABLES
- CREATE USER
- CREATE VIEW
- DELETE
- DROP
- EVENT
- EXECUTE
- FILE
- GRANT OPTION
- INDEX
- INSERT
- LOCK TABLES
- PROCESS
- REFERENCES
- RELOAD
- REPLICATION CLIENT
- REPLICATION SLAVE
- SELECT
- SHOW DATABASES
- SHOW VIEW
- SHUTDOWN
- SUPER
- TRIGGER
- UPDATE

Als zweites wird der developer User mit dem GUI von MySQL erstellt. Der developer hat Zugriffe auf alle Datenbanken, kann auch das Schema verändern und hat Rechte, um Backups zu machen. Für die Vertraulichkeit sind jedoch die Leserechte wichtig. Der developer hat Leserechte auf allen Datenbanken durch die Rolle DBManager. Weitere Privilegien wie das Erstellen von User Logins sind im Untersagt, da er die Rolle UserAdmin nicht besitzt.

The screenshot shows the 'Users and Privileges' section of MySQL Workbench. On the left, a list of existing users is shown:

User	From Host
admin	%
admin	localhost
gui	localhost
mysql.infoschema	localhost
mysql.session	localhost
mysql.sys	localhost
newuser	%
root	localhost

The main area is titled 'Details for account newuser@%' and contains tabs for 'Login', 'Account Limits', 'Administrative Roles', and 'Schema Privileges'. The 'Login' tab is selected, showing fields for 'Login Name' (developer), 'Authentication Type' (caching\_sha2\_password), 'Limit to Hosts Matching' (%), 'Password' (\*\*\*\*\*), 'Confirm Password' (\*\*\*\*\*), and 'Expire Password'.

The 'Administrative Roles' tab lists various MySQL roles with their descriptions:

Role	Description
<input type="checkbox"/> DBA	grants the rights to perform all tasks
<input type="checkbox"/> MaintenanceAdmin	grants rights needed to maintain server
<input type="checkbox"/> ProcessAdmin	rights needed to assess, monitor, and kill any user proce...
<input type="checkbox"/> UserAdmin	grants rights to create users logins and reset passwords
<input type="checkbox"/> SecurityAdmin	rights to manage logins and grant and revoke server an...
<input type="checkbox"/> MonitorAdmin	minimum set of rights needed to monitor server
<input checked="" type="checkbox"/> DBManager	grants full rights on all databases
<input checked="" type="checkbox"/> DBDesigner	rights to create and reverse engineer any database sche...
<input type="checkbox"/> ReplicationAdmin	rights needed to setup and manage replication
<input checked="" type="checkbox"/> BackupAdmin	minimal rights needed to backup any database
<input type="checkbox"/> Custom	custom role

The 'Schema Privileges' tab lists global privileges, many of which are checked:

- ALTER
- ALTER ROUTINE
- CREATE
- CREATE ROUTINE
- CREATE TABLESPACE
- CREATE TEMPORARY TABLES
- CREATE USER
- CREATE VIEW
- DELETE
- DROP
- EVENT
- EXECUTE
- FILE
- GRANT OPTION
- INDEX
- INSERT
- LOCK TABLES
- PROCESS
- REFERENCES
- RELOAD
- REPLICATION CLIENT
- REPLICATION SLAVE
- SELECT
- SHOW DATABASES
- SHOW VIEW
- SHUTDOWN
- SUPER
- TRIGGER
- UPDATE

Zuletzt wird der gui User erstellt, welcher nur vom localhost, also der VM, auf welcher die Datenbank läuft auf die Datenbank zugreifen kann. Dieser User wurde mit commands und nicht mit dem GUI erstellt.

```
1 •  CREATE USER gui@localhost
2      IDENTIFIED BY 'f13SF1k_$';
```

### Ausgabe

```
| 2 10:58:22 CREATE USER gui@localhost IDENTIFIED BY f13SF1k_$' 0 row(s) affected 0.062 sec
```

Zudem wurden dem User gui User auch direkt nur ein select Zugriff auf die Datenbank gewährt, welche nur aus views der «original» Datenbank besteht. Es muss also zuerst diese Datenbank erstellt und mit views gefüllt werden. Die Erstellung und Befüllung der Datenbank wird im folgenden Query ausgeführt. Die oberste Zeile erstellt die Datenbank und alle weiteren Zeilen kreeieren eine view auf der neuen Datenbank mit einem select statement auf der originalen Datenbank für einen spezifischen table. Schlussendlich werden die priviliges caches geflushed sodass die Änderung auch eintreten und direkt die Berechtigungen des gui User angezeigt. Der Vorteil dieser zweiten Datenbank ist, dass der gui User gar keinen Zugriff auf die Original Datenbank hat, und somit bei einem Angriff eingeschränkter ist.

```
-- make Visualization of ScoutingAndAnalysis View
CREATE DATABASE visualization_ScoutingAndAnalysis;

CREATE OR REPLACE VIEW visualization_ScoutingAndAnalysis.club_has_domestic_competition AS
SELECT * FROM scoutingandanalysis.club_has_domestic_competition;

CREATE OR REPLACE VIEW visualization_ScoutingAndAnalysis.club_has_seats AS
SELECT * FROM scoutingandanalysis.club_has_seats;

CREATE OR REPLACE VIEW visualization_ScoutingAndAnalysis.club_has_stadium_name AS
SELECT * FROM scoutingandanalysis.club_has_stadium_name;

CREATE OR REPLACE VIEW visualization_ScoutingAndAnalysis.clubs AS
SELECT * FROM scoutingandanalysis.clubs;

CREATE OR REPLACE VIEW visualization_ScoutingAndAnalysis.competitions AS
SELECT * FROM scoutingandanalysis.competitions;

CREATE OR REPLACE VIEW visualization_ScoutingAndAnalysis.game_events AS
SELECT * FROM scoutingandanalysis.game_events;

CREATE OR REPLACE VIEW visualization_ScoutingAndAnalysis.game_events_has_date AS
SELECT * FROM scoutingandanalysis.game_events_has_date;

CREATE OR REPLACE VIEW visualization_ScoutingAndAnalysis.games AS
SELECT * FROM scoutingandanalysis.games;

CREATE OR REPLACE VIEW visualization_ScoutingAndAnalysis.player_events AS
SELECT * FROM scoutingandanalysis.player_events;

CREATE OR REPLACE VIEW visualization_ScoutingAndAnalysis.players AS
SELECT * FROM scoutingandanalysis.players;
```

## Ausgabe

14	17:35:19	CREATE DATABASE visualization_ScoutingAndAnalysis	1 row(s) affected	0.063 sec
15	17:35:19	CREATE OR REPLACE VIEW visualization_ScoutingAndAnalysis.club_has_domestic_competition ...	0 row(s) affected	0.125 sec
16	17:35:19	CREATE OR REPLACE VIEW visualization_ScoutingAndAnalysis.club_has_seats AS SELECT * F...	0 row(s) affected	0.046 sec
17	17:35:19	CREATE OR REPLACE VIEW visualization_ScoutingAndAnalysis.club_has_stadium_name AS SEL...	0 row(s) affected	0.032 sec
18	17:35:19	CREATE OR REPLACE VIEW visualization_ScoutingAndAnalysis.clubs AS SELECT * FROM scou...	0 row(s) affected	0.078 sec
19	17:35:19	CREATE OR REPLACE VIEW visualization_ScoutingAndAnalysis.competitions AS SELECT * FRO...	0 row(s) affected	0.047 sec
20	17:35:19	CREATE OR REPLACE VIEW visualization_ScoutingAndAnalysis.game_events AS SELECT * FRO...	0 row(s) affected	0.047 sec
21	17:35:19	CREATE OR REPLACE VIEW visualization_ScoutingAndAnalysis.game_events_has_date AS SEL...	0 row(s) affected	0.047 sec
22	17:35:19	CREATE OR REPLACE VIEW visualization_ScoutingAndAnalysis.games AS SELECT * FROM sco...	0 row(s) affected	0.047 sec
23	17:35:19	CREATE OR REPLACE VIEW visualization_ScoutingAndAnalysis.player_events AS SELECT * FR...	0 row(s) affected	0.109 sec
24	17:35:20	CREATE OR REPLACE VIEW visualization_ScoutingAndAnalysis.players AS SELECT * FROM sco...	0 row(s) affected	0.110 sec

Danach wird dem gui User die Berechtigung auf die visualization\_scoutingAndAnalysis Datenbank gewährt. Diese Rechtevergabe kann dann auch beispielsweise im gui überprüft werden. Er hat aber ansonsten gar keine Leserechte oder Rollen.

```
GRANT SELECT ON visualization_ScoutingAndAnalysis.*  
TO gui@localhost;
```

1	22:13:44	GRANT SELECT ON visualization_scoutingandanalysis.* TO 'gui'@'localhost'	0 row(s) affected	0.063 sec
---	----------	--	-------------------	-----------

The screenshot shows the MySQL Workbench User Accounts interface. On the left, a table lists users and their hosts. In the center, the 'Details for account gui@localhost' pane is open, showing the schema 'visualization\_scoutingandanalysis' with the privilege 'SELECT'. Below this, a list of roles and their descriptions is shown. On the right, a large list of global privileges is available for selection.

User	From Host
admin	%
developer	%
gui	localhost
mysql.infoschema	localhost
mysql.session	localhost
mysql.sys	localhost
root	localhost

**Details for account gui@localhost**

Schema: visualization\_scoutingandanalysis Privileges: SELECT

Schema and Host fields may use % and \_ wildcards.  
The server will match specific entries before wildcarded ones.

Role Description

- DBA grants the rights to perform all tasks
- MaintenanceAdmin grants rights needed to maintain server
- ProcessAdmin rights needed to assess, monitor, and kill any user process
- UserAdmin grants rights to create users/login and reset passwords
- SecurityAdmin rights to manage logins and grant and revoke server accounts
- MonitorAdmin minimum set of rights needed to monitor server
- DBManager grants full rights on all databases
- DBDesigner rights to create and reverse engineer any database schema
- ReplicationAdmin rights needed to setup and manage replication
- BackupAdmin minimal rights needed to backup any database

Global Privileges

- ALTER
- ALTER ROUTINE
- CREATE
- CREATE ROUTINE
- CREATE TABLESPACE
- CREATE TEMPORARY TABLES
- CREATE USER
- CREATE VIEW
- DELETE
- DROP
- EVENT
- EXECUTE
- FILE
- GRANT OPTION
- INDEX
- INSERT
- LOCK TABLES
- PROCESS
- REFERENCES
- RELOAD
- REPLICATION CLIENT
- REPLICATION SLAVE
- SELECT
- SHOW DATABASES
- SHOW VIEW
- SHUTDOWN
- SUPER
- TRIGGER
- UPDATE

Zuletzt können wir uns alle von uns definierten User in mySql anzeigen lassen.

1 • **SELECT user, host FROM mysql.user;**

Ausgabe:

The screenshot shows the MySQL Workbench Result Grid. It displays the results of the query 'SELECT user, host FROM mysql.user;'. The results are as follows:

user	host
admin	%
developer	%
gui	localhost
mysql.infoschema	localhost
mysql.session	localhost
mysql.sys	localhost
root	localhost

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	22:48:49	SELECT user, host FROM mysql.user LIMIT 0, 50000	7 row(s) returned	0.015 sec / 0.000 sec

In der Ausgabe sehen wir, dass die User admin und developer von allen hosts erreichbar sind und der gui host nur über den localhost. Also genau wie wir es in unserer Definition der Massnahmen definiert haben.

## 7.2.5 Umsetzung Massnahmen Vertraulichkeit MongoDB

### 7.2.5.1 User Erstellung

Zuerst werden die notwendigen User erstellt. Zuerst wird der developer User erstellt. Mit dem nachfolgenden Befehl wird der developer erstellt. Dazu muss der Username, das Passwort und die Rollen des Users definiert. Der developer hat also Lese – und Schreibrechte auf allen Datenbanken, aber kann keine neuen DB's erstellen und hat auch keine Administrator-Rechte. Er wird zudem in der admin Datenbank angelegt.

```
admin> db.createUser({  
...   user: "developer",  
...   pwd: "X8z3$'KU>Q",  
...   roles: [  
...     { role: "readWriteAnyDatabase", db: "admin" }  
...   ]  
... })  
{ ok: 1 }  
admin>
```

Desweiteren wird auch der gui User erstellt. Er hat nur die Rolle read auf der scoutingAndAnalysis DB. Er hat also nur Leserechte auf dieser spezifischen DB und kann beispielsweise keine neuen Collections erstellen. Im zweiten Screenshot sehen wir dann auch dass der guiUser keine Daten in die Datenbank einfügen kann. Weil der guiUser auf die scoutingAndAnalysis Datenbank eingeschränkt ist, wird er auch dort erstellt.

```
scoutingAndAnalysis> db.createUser({  
...   user: "guiUser",  
...   pwd: "h<sGL3ZF>:#z2VN%",  
...   roles: [{ role: "read", db: "scoutingAndAnalysis" }]  
... })  
{ ok: 1 }  
scoutingAndAnalysis> db.yourCollection.insertOne({ test: "data" })  
MongoServerError[Unauthorized]: not authorized on scoutingAndAnalysis to execute command { insert: "yourCollection", documents: [ { test: "data", _id: ObjectId('664fa8c64164463d12eccea0') } ], ordered: true, lsid: { id: UUID("ed2d2731-a817-4691-b7bc-60cf7c79431e") }, $db: "scoutingAndAnalysis" }  
scoutingAndAnalysis> |
```

Schlussendlich wird auch noch der admin User myAdmin erstellt. Der myAdmin User hat also Admin-Rechte auf alle Datenbanken in dieser MongoDB Instanz. Zudem hat er mit root auch die Möglichkeit Datenbanken zu erstellen oder löschen und weitere Admin-Rechte.

```
admin> db.createUser({ user: "myAdmin", pwd: "w=V)xyZ hc&SUW98Q,L/#", roles: [ { role: "dbAdmin", db: "admin" }, { role: "userAdminAnyDatabase", db: "admin" }, { role: "root", db: "admin" } ] })
```

Zuletzt kann mit dem command db.system.users.find().pretty() alle definierten User auf allen Datenbanken aufgelistet werden. Die system.users collection beinhaltet dabei die Informationen über aller

User der MongoDB. Pretty macht die Ausgabe lediglich etwas leserlicher.

```
admin> db.system.users.find().pretty()
[ {
  _id: 'scoutingAndAnalysis.guilUser',
  userId: UUID('df7c504d-a0e8-43fe-b49d-1c4f51c36a49'),
  user: 'guilUser',
  db: 'scoutingAndAnalysis',
  credentials: {
    'SCRAM-SHA-1': {
      iterationCount: 10000,
      salt: 'QBbkqHN02j4y80+zq7bYlg==',
      storedKey: 'wPHUVA90vdst8sVd67ps/J1aTQ=',
      serverKey: 'kg+ru0bD0k9Ko4ofd10uiKXHiI='
    },
    'SCRAM-SHA-256': {
      iterationCount: 15000,
      salt: 'o/TV7R+jjv5yIWpK8vSK0K/XiuSA2+TDBMsFDA==',
      storedKey: 'Kn/LYbzQeR5b0vjGz40YeetJ1uRif9rR8Biuw0JLA=',
      serverKey: 'bI7hsUIZUJf3bm6/zoB4vmk+zL+OosbHK34AZxUdq2g='
    }
  },
  roles: [ { role: 'read', db: 'scoutingAndAnalysis' } ]
},
{
  _id: 'admin.myAdmin',
  userId: UUID('c8284c5a-6d36-4e0a-afb1-0f7cf1844c66'),
  user: 'myAdmin',
  db: 'admin',
  credentials: {
    'SCRAM-SHA-1': {
      iterationCount: 10000,
      salt: 'CVjqPzkqlfUhC+vBATkZNQ==',
      storedKey: 'n4s1/1V9ibHGolhTsjobTH8gXB0=',
      serverKey: 'c8wf83XTxFc2MMEhaQvb+TddmW0='
    },
    'SCRAM-SHA-256': {
      iterationCount: 15000,
      salt: '+fstdw405tIV39mn2KwQXKIn3c31xh2w8mThdA==',
      storedKey: '5vI7zaW01sCx3zeKX6+nuGAy/R9UAZgoVonenh3XHyQ=',
      serverKey: 'HWc81CdYEq3hJRvIlFgHUqy9erJU+EsZleWEXCPg/4='
    }
  },
  roles: [
    { role: 'dbAdmin', db: 'admin' },
    { role: 'userAdminAnyDatabase', db: 'admin' },
    { role: 'root', db: 'admin' }
  ]
},
{
  _id: 'admin.developer',
  userId: UUID('b5612a29-f969-4b8d-a4a1-47f434a5eb89'),
  user: 'developer',
  db: 'admin',
  credentials: {
    'SCRAM-SHA-1': {
      iterationCount: 10000,
      salt: 'mdqgYQ1JSATBKErfGtP/ow==',
      storedKey: 'FTS3tmpQOpnf6HmVf/Vfc3munn8=',
      serverKey: 'bXWEp19C7nT6b50l0/DXrhzHwqs='
    },
    'SCRAM-SHA-256': {
      iterationCount: 15000,
      salt: 'qNUMIAssCgLn/rk5W9llF04Y8UJeN1jJaAspQ0==',
      storedKey: 'W/nKXvByvdKzooPImmJhoXZ+9wDzdJShAyg9I7370ag=',
      serverKey: 'IOXJFUgcvSQDR/LDGJqqdGI5xPvd0yB88RlCLHDYrhc='
    }
  },
  roles: [ { role: 'readWriteAnyDatabase', db: 'admin' } ]
]
admin> |
```

### 7.2.6 Definition Massnahmen Integrität SQL

Bei der Integrität stehen vor allem die Schreibrechte der verschiedenen User im Vordergrund. User, die keine Schreibrechte brauchen, sollen diese auch nicht bekommen. Mit den eingeschränkten Rechten können die User die Daten nicht verändern, wenn sie dies nicht dürfen. Die Integrität der Daten ist also geschützt. Der admin User hat Schreibrechte auf Alles. Der developer hat Schreibrechte auf alle Datenbanken. Der Metabase User hat überhaupt keine Schreibrechte.

Im Weiteren wurden die Backups, die wie unter dem Punkt Verfügbarkeit beschrieben worden sind, unbedingt noch verschlüsseln. Ohne diese zusätzliche Verschlüsselung des Backups, kann ein Angreifer die Daten einsehen (Verstoss gegen Vertraulichkeit), aber auch die Daten ändern (Verletzung gegen die Integrität). Die Verschlüsselung wird mit openssl auf der Maschine, wo das Backup gespeichert wird,

verschlüsselt.

### 7.2.7 Umsetzung Massnahmen Integrität SQL

Um die Rechte der User zu begutachten, öffnet man die GUI-Ansicht unter Server -> Users and Privileges. Dann kann der Reiter administrative roles geöffnet werden, um die Rollen und Privilegien eines Users anzuschauen. Dieser Vorgang ist identisch zu den bereits bei der Vertraulichkeit erwähnten Schreibrechten.

Die Rechtevergabe wurde bereits im Kapitel zur Verfügbarkeit konfiguriert. Hier wird nur nochmals konkret auf die drei User mit den verschiedenen Rechten eingegangen.

Der Gui User hat lediglich ein SELECT Recht und demnach gar keine Schreibrechte auf die visualization\_scoutingandanalysis Datenbank, welche die Integrität der Daten verletzen könnte.

Details for account **gui@localhost**

Login Account Limits Administrative Roles Schema Privileges

Schema	Privileges
visualization_scoutingandanalysis	SELECT

Der admin und der developer User haben beide alle Schreibrechte auf Datenbanken. Beide User benötigen die Schreibrechte allerdings auch. Sie haben beide die Privilegien DROP, DELETE, ALTER oder INSERT.

Details for account **admin@%**

Login Account Limits Administrative Roles Schema Privileges

Role	Description
DBA	grants the rights to perform all tasks
MaintainanceAdmin	grants rights needed to maintain server
ProcessAdmin	rights needed to assess, monitor, and kill any user process
UserAdmin	grants rights to create user logins and reset passwords
SecurityAdmin	rights to manage logins and grant and revoke server and database level permissions
MonitorAdmin	minimum set of rights needed to monitor server
DBManager	grants full rights on all databases
DBDesigner	rights to create and reverse engineer any database schema
ReplicationAdmin	rights needed to setup and manage replication
BackupAdmin	minimal rights needed to backup any database

Global Privileges

- ALTER
- CREATE ROUTINE
- CREATE
- CREATE ROUTINE
- CREATE TABLESPACE
- CREATE TEMPORARY TABLES
- CREATE USER
- CREATE VIEW
- DELETE
- DROP**
- EVENT
- EXECUTE
- FILE
- GRANT OPTION
- INDEX
- INSERT
- LOCK TABLES
- PROCESS
- REFERENCES
- RELLOAD
- REPLICATION CLIENT
- REPLICATION SLAVE
- SELECT
- SHOW DATABASES
- SHOW VIEW
- SHUTDOWN
- SUPER
- TRIGGER
- UPDATE

Details for account developer@%			
Login	Account Limits	Administrative Roles	Schema Privileges
<input type="checkbox"/> DBA	grants the rights to perform all tasks		
<input type="checkbox"/> MaintenanceAdmin	grants rights needed to maintain server		
<input type="checkbox"/> ProcessAdmin	rights needed to assess, monitor, and kill any user proce...		
<input type="checkbox"/> UserAdmin	grants rights to create users, logins and reset passwords		
<input type="checkbox"/> SecurityAdmin	rights to manage logins and grant and revoke server an...		
<input type="checkbox"/> MonitorAdmin	minimum set of rights needed to monitor server		
<input checked="" type="checkbox"/> DBManager	grants full rights on all databases		
<input type="checkbox"/> DBDesigner	rights to create and reverse engineer any database sche...		
<input type="checkbox"/> ReplicationAdmin	rights needed to setup and manage replication		
<input checked="" type="checkbox"/> BackupAdmin	minimal rights needed to backup any database		
<b>Global Privileges</b>			
<input checked="" type="checkbox"/>	ALTER		
<input checked="" type="checkbox"/>	ALTER ROUTINE		
<input checked="" type="checkbox"/>	CREATE		
<input checked="" type="checkbox"/>	CREATE ROUTINE		
<input checked="" type="checkbox"/>	CREATE TABLESPACE		
<input checked="" type="checkbox"/>	CREATE TEMPORARY TABLES		
<input checked="" type="checkbox"/>	CREATE USER		
<input checked="" type="checkbox"/>	CREATE VIEW		
<input checked="" type="checkbox"/>	DELETE		
<input checked="" type="checkbox"/>	DROP		
<input checked="" type="checkbox"/>	EVENT		
<input type="checkbox"/>	EXECUTE		
<input type="checkbox"/>	FILE		
<input checked="" type="checkbox"/>	GRANT OPTION		
<input checked="" type="checkbox"/>	INDEX		
<input checked="" type="checkbox"/>	INSERT		
<input checked="" type="checkbox"/>	LOCK TABLES		
<input type="checkbox"/>	PROCESS		
<input type="checkbox"/>	REFERENCES		
<input type="checkbox"/>	RELOAD		
<input type="checkbox"/>	REPLICATION CLIENT		
<input type="checkbox"/>	REPLICATION SLAVE		
<input checked="" type="checkbox"/>	SELECT		
<input checked="" type="checkbox"/>	SHOW DATABASES		
<input checked="" type="checkbox"/>	SHOW VIEW		
<input type="checkbox"/>	SHUTDOWN		
<input type="checkbox"/>	SUPER		
<input checked="" type="checkbox"/>	TRIGGER		
<input checked="" type="checkbox"/>	UPDATE		

Die Verschlüsselung des Backups wird mit openssl realisiert. Zuerst wird aber ein archive mit tar erstellt, welches die beiden files vom Backup beinhaltet. Danach wird das archive mithilfe von opessl verschlüsselt, sodass es nicht mehr lesbar ist. Es wird dabei ein Passwort gesetzt.

```
C:\Users\fabia\Desktop>tar -cvf backup.tar C:\Users\fabia\Desktop\Backup
tar: Removing leading drive letter from member names
a Users/fabia/Desktop/Backup
a Users/fabia/Desktop/Backup/create.sql
a Users/fabia/Desktop/Backup/insert.sql

C:\Users\fabia\Desktop>openssl enc -aes-256-cbc -salt -in backup.tar -out backupEnc.tar.enc
enter aes-256-cbc encryption password:

Verifying - enter aes-256-cbc encryption password:

*** WARNING : deprecated key derivation used.
Using pbkdf2 would be better.
```

Beim Öffnen des archives im verschlüsselten Zustand kann dieses nicht gelesen werden. Wenn die files aber wieder entschlüsselt werden sollen, kann dies ebenfalls gemacht werden. Der Befehl erstellt ein decrypted archive, welches dann wieder normal gelesen werden kann. Dieses wird erstellt, nachdem das beim Verschlüsseln gesetzte Passwort wieder eingegeben wurde.

Logischerweise muss das initiale, unverschlüsselte Backup nach dem Verschlüsseln gelöscht werden, sodass ein Angreifer nicht einfach das unverschlüsselte archive lesen kann. Mit dem verschlüsselten Backup kann die Integrität bei einer Attacke auf die Backup-Maschine also erhalten werden.

### 7.2.8 Umsetzung Massnahmen Integrität MongoDB

Die User-Rechte wurden bereits im Kapitel 7.26 erläutert. Diese schützen also nicht nur vor Lesezugriff, sondern verhindern auch das Ändern von sensiblen Daten. Dies wurde auch bereits exemplarisch für den guiUser gezeigt.

Die Integrität wird logischerweise auch durch die decryption, wie im Kapitel 7.2.8 beschrieben geschützt. Ein Angreifer könnte theoretisch auch einfach eine ungeschützte Datei verändern (nicht nur Einsehen), auch wenn sie nur die Backup-Datei ist, und somit eventuell die Integrität der Daten verletzen. Die decryption des MongoDB Backup kann genauso vorgenommen werden wie die decryption des SQL-Backups, welches im vorherigen Kapitel beschrieben wurde.

## 8 Visualisierung & Entscheidungsunterstützung

Um die Daten der beiden Datenbanken zu visualisieren, wurde Metabase verwendet. Die Visualisierung wurden jeweils getrennt für SQL und NoSQL umgesetzt.

### 8.1 Zugriff auf VM und Metabase starten

Um die Visualisierungen einzusehen, muss per Remote Desktop auf die VM zugegriffen werden.

Was	Benutzername	Passwort
VM (dbs-21, 86.119.35.98)	dbsstudent	5Jv6AEwXqNtVih

Anschliessend kann per Command Prompt der Metabase Service gestartet werden.

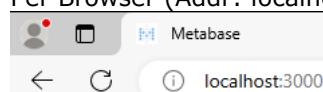
```
C:\Users\dbsstudent>cd Desktop\Metabase
```

```
C:\Users\dbsstudent\Desktop\Metabase>java -jar metabase.jar
```

Das Set-Up ist nach ca. 1.5 min fertig. (Metabase Initialization COMPLETE)

```
2024-06-14 09:16:57,645 INFO database.core :: Metabase Initialization COMPLETE in 1.4 mins  
2024-06-14 09:18:03,955 INFO i18n.impl :: Reading available locales from locales.clj...  
2024-06-14 09:18:04,313 INFO util.fonts :: Reading available fonts from /frontend/client/ap
```

Per Browser (Addr: localhost:3000) kann nun auf den Metabase Dienst zugegriffen werden.



Metabase	dominik.arnet@stud.hslu.ch	si*8N#EsU~*7kv}
----------	----------------------------	-----------------

### 8.2 Anbindungen der Datenbanken

Von Metabase aus können per GUI die Anbindungen an mehrere Datenbanken konfiguriert werden.

#### 8.2.1 SQL Anbindung

	Vom Dropdown konnte der korrekte Datenbanktyp ausgewählt werden.
	Ein Sprechender Anzeigename wurde gewählt.
	Da der MySQL Server und Metabase beide auf der VM laufen kann als host «localhost» angegeben werden.

<p>Database type</p> <p>MySQL</p> <p>Display name</p> <p>scoutingAndAnalysis_SQL</p> <p>Host</p> <p>localhost</p> <p>Port</p> <p>3306</p> <p>Database name</p> <p>visualization_scoutingandanalysis</p> <p>Username</p> <p>gui</p> <p>Password</p> <p>*****</p> <p>Use a secure connection (SSL)</p> <p><input checked="" type="checkbox"/></p> <p>Server SSL certificate chain</p> <p>Paste the contents of the server's SSL certificate chain here</p> <p>Use an SSH-tunnel</p> <p>If a direct connection to your database isn't possible, you may want to use an SSH tunnel.</p> <p><a href="#">Learn more.</a></p> <p><a href="#">Hide advanced options</a></p> <p>Allow unfolding of JSON columns</p> <p>This enables unfolding JSON columns into their component fields. Disable unfolding if performance is slow. If enabled, you can still disable unfolding for individual fields in their settings.</p> <p>Additional JDBC connection string options</p> <p>trustServerCertificate=true</p>	<p>Der Port ist standartmässig auf 3306 festgelegt.</p> <p>Die Datenbank, welche die Daten zur Visualisierung enthält wird angegeben.</p> <p>Da das Metabase nur Leseberechtigungen benötigt, wird der gui User verwendet. (PW: fl3SF1k_ \$)</p> <p>Die SSL-Verbindung wurde auf Empfehlung von ChatGPT eingeschalten und der zusätzliche JDBC-Parameter trustServerCertificate=true gesetzt. Ansonsten erschien teilweise der Fehler: <b>Could not connect to address=(host=86.119.35.98) (port=3306)(type=master) : RSA public key is not available client side (option serverRsaPublicKeyFile)</b></p> <p>Mit dem Aktivieren von SSL und dem Parameter konnte dieser gänzlich behoben werden.</p>
---	--

### 8.2.2 NoSQL Anbindung

<p><b>Database type</b></p> <p>MongoDB</p> <p><b>Display name</b></p> <p>scoutingAndAnalysis_NoSQL</p> <p><b>Fill out individual fields</b></p> <p>Paste your connection string</p> <pre>mongodb://guiUser:h%3CsGL3ZF%3E%3A%23z2VN%25@localhost:27017/scoutin</pre> <p><b>Use an SSH-tunnel</b></p> <p>If a direct connection to your database isn't possible, you may want to use an SSH tunnel.</p> <p><a href="#">Learn more.</a></p> <p><b>Hide advanced options ^</b></p> <p><b>Rerun queries for simple explorations</b></p> <p>We execute the underlying query when you explore data using Summarize or Filter. This is on by default but you can turn it off if performance is slow.</p> <p><b>Choose when syncs and scans happen</b></p> <p>By default, Metabase does a lightweight hourly sync and an intensive daily scan of field values. If you have a large database, turn this on to make changes.</p> <p><b>Periodically refingerprint tables</b></p> <p>This enables Metabase to scan for additional field values during syncs allowing smarter behavior, like improved auto-binning on your bar charts.</p> <p><b>Save changes</b></p>	<p>Auch für die Anbindung der MongoDB konnte direkt der korrekte Datenbanktyp ausgewählt werden.</p> <p>Es wurde ein sprechender Anzeigename gewählt, um SQL und NoSQL unterscheiden zu können.</p> <p>Anders als bei MySQL wurde für die MongoDB Anbindung direkt der ConnectionString aus Kapitel 4.2 verwendet. Alle anderen Einstellungen wurden beim Standard belassen.</p>
--	--

### 8.3 Übersicht der Visualisierungen

<p><b>Our analytics</b></p> <table border="1"> <thead> <tr> <th>Type</th><th>Name</th></tr> </thead> <tbody> <tr> <td><input type="checkbox"/></td><td>Formation Wins Analytics NoSQL</td></tr> <tr> <td><input type="checkbox"/></td><td>Formation Wins Analytics SQL</td></tr> <tr> <td><input type="checkbox"/></td><td>Late Goals Analytics NoSQL</td></tr> <tr> <td><input type="checkbox"/></td><td>Late Goals Analytics SQL</td></tr> <tr> <td><input type="checkbox"/></td><td>Player Infos NoSQL</td></tr> <tr> <td><input type="checkbox"/></td><td>Player Infos SQL</td></tr> <tr> <td><input type="checkbox"/></td><td>Player Scouting NoSQL</td></tr> <tr> <td><input type="checkbox"/></td><td>Player Scouting SQL</td></tr> </tbody> </table>	Type	Name	<input type="checkbox"/>	Formation Wins Analytics NoSQL	<input type="checkbox"/>	Formation Wins Analytics SQL	<input type="checkbox"/>	Late Goals Analytics NoSQL	<input type="checkbox"/>	Late Goals Analytics SQL	<input type="checkbox"/>	Player Infos NoSQL	<input type="checkbox"/>	Player Infos SQL	<input type="checkbox"/>	Player Scouting NoSQL	<input type="checkbox"/>	Player Scouting SQL	<p>Im Tab «Our Analytics» sind folgende Visualisierungen zu finden.</p> <p>Formation Wins SQL/NoSQL ermöglicht es die Aufstellungen, welche für einen bestimmten Gegner zu einem Sieg führten, einzusehen. Zusätzlich kann der Zeitraum in dem gemessen werden soll spezifiziert werden.</p> <p>Bei Late Goals Analytics SQL /NoSQL kann eingesehen werden in wie vielen Spielen eine Mannschaft ein Goal nach der oder während der 80 Minute erzielt hat.</p> <p>Die Player Infos SQL/NoSQL Questions sind nicht zu beachten, da diese auf den respektiven Dashboards (blaues Icon) verwendet werden. Diese Dashboards ermöglichen es einem Benutzer wie Jose Spieler anhand einer grossen Anzahl von Merkmalen zu suchen.</p>
Type	Name																		
<input type="checkbox"/>	Formation Wins Analytics NoSQL																		
<input type="checkbox"/>	Formation Wins Analytics SQL																		
<input type="checkbox"/>	Late Goals Analytics NoSQL																		
<input type="checkbox"/>	Late Goals Analytics SQL																		
<input type="checkbox"/>	Player Infos NoSQL																		
<input type="checkbox"/>	Player Infos SQL																		
<input type="checkbox"/>	Player Scouting NoSQL																		
<input type="checkbox"/>	Player Scouting SQL																		

## 8.4 Formation Wins

Die Formation Wins Analyse wurde einmal in SQL und einmal in NoSQL umgesetzt. Dieses Query basiert auf dem im Kapitel 5.2.3 bzw. 5.3.3 erklärten Use Case Query, mit dem Unterschied, dass nur gewonnene Spiele angezeigt werden.

### 8.4.1 SQL

Das SQL Query wurde als native Question in Metabase umgesetzt. In ein Select werden die Formationen und die Anzahl Spiele selektiert. Diese stammen wiederum aus einem Subquery, welches aus zwei Selects besteht die zu einer Union verbunden werden. Da eine Mannschaft entweder Host oder Guest sein kann wird in den beiden Selects im Subquery die Home bzw. Away Club Formation selektiert, wo das Game Date in einem gewissen Range ist und die Anzahl Tore grösser ist als die des Gegners (Spiel gewonnen). Vorher wird bei beiden aber noch die Clubs Tabelle dazu gejoint, damit Benutzer wie Jose nach Namen des Klubs filtern können und nicht nach einer kryptischen ID.

Die Resultate werden nach Formation gruppiert und nach Anzahl gewonnene Spiele absteigend sortiert. Mittels den Zwei geschweiften Klammern kann in Metabase ein Query parametrisiert werden. Diese Filtermöglichkeiten werden im nächsten Kapitel erklärt.

```
-- Metabase Query Formation Wins

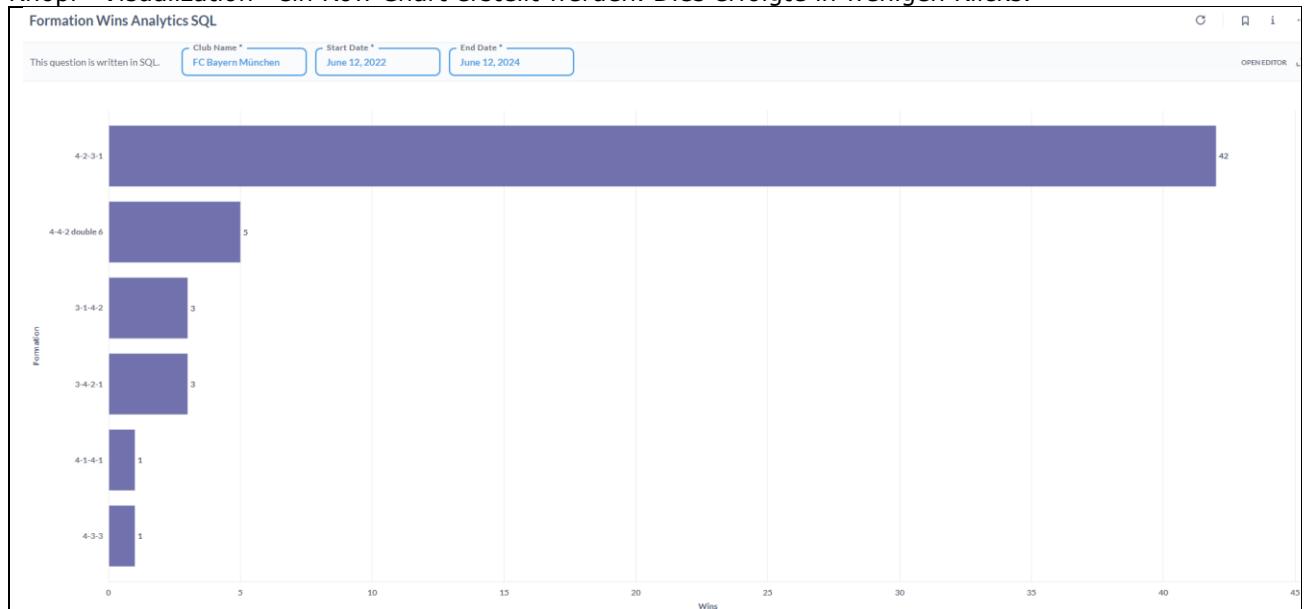
select Formation, count(*) as Wins from (
    select home_club_formation as Formation
    from games join clubs c on games.home_club_id = c.club_id
    where (c.club_name = {{club_name}})
    and games.game_date BETWEEN {{start_date}} AND {{end_date}}
    and home_club_goals > away_club_goals
    UNION ALL
    select away_club_formation as Formation
    from games join clubs c on games.away_club_id = c.club_id
    where (c.club_name = {{club_name}})
    and games.game_date BETWEEN {{start_date}} AND {{end_date}}
    and home_club_goals < away_club_goals)
    as homeAndAwayGames
group by Formation
order by count(*) DESC;
```

### 8.4.1.1 Filter

<p>Variable name <b>club_name</b></p> <p>Variable type Text</p> <p>Filter widget label Club Name</p> <p>How should users filter on this variable?  <input checked="" type="radio"/> Dropdown list      <a href="#">Edit</a>  <input type="radio"/> Search box  <input type="radio"/> Input box         </p> <p>Default filter widget value FC Bayern München <span style="border: 1px solid #ccc; padding: 2px;">x</span></p> <p><input checked="" type="checkbox"/> Always require a value When enabled, people can change the value or reset it, but can't clear it entirely.</p>	<p>Variable name <b>start_date</b></p> <p>Variable type Date</p> <p>Filter widget label Start Date</p> <p>Default filter widget value June 12, 2022 <span style="border: 1px solid #ccc; padding: 2px;">x</span></p> <p><input checked="" type="checkbox"/> Always require a value When enabled, people can change the value or reset it, but can't clear it entirely.</p>	<p>Variable name <b>end_date</b></p> <p>Variable type Date</p> <p>Filter widget label End Date</p> <p>Default filter widget value June 12, 2024 <span style="border: 1px solid #ccc; padding: 2px;">x</span></p> <p><input checked="" type="checkbox"/> Always require a value When enabled, people can change the value or reset it, but can't clear it entirely.</p>
<p>Der Club Name ist standartmäßig FC Bayern München. Die Auswahl von Dropdown List ermöglicht es nach Clubs zu suchen und den korrekten auszuwählen.</p>		<p>Die beiden Datums Filter wurden auf den Datentyp Date festgelegt und als Defaultwerte eine Spanne von zwei Jahren definiert.</p>

### 8.4.1.2 Visualisierung

Das parametrisierte Query ergibt die Folgende Visualisierung. Dazu musste aber zuerst noch über den Knopf «Visualization» ein Row Chart erstellt werden. Dies erfolgte in wenigen Klicks.



### 8.4.2 NoSQL

Das native Query für MongoDB war einiges aufwändiger umzusetzen als das SQL Query. Zuerst wurden im ClubsDocument die Games Einträge geunwinded wo der Club Name und die Date Range matched. Anschliessen wurde ein Feld erstellt, ob die Mannschaft Heimspiel hatte oder nicht. Dies hilft im nachfolgenden Schritt festzustellen welche Mannschaft das Spiel gewonnen hat. Dies ist eher umständlich da die gesuchte Mannschaft entweder Home oder Away sein kann und basierend darauf muss verglichen werden ob Home Goals > Away Goals oder Away Goals > Home Goals um alle gewonnenen Spiele zu finden. Das isHome Feld hilft uns anschliessend wieder, um die korrekte Formation zu selektieren. Zum Schluss wird gleich wie bei SQL nach Formation gruppiert und nach Anzahl gewonnene Spiele sortiert.

```
//Metabase Query Formation Wins

[
  {
    $unwind: "$Games"
  },
  {
    $match: {
      "club_name": "{$club_name}",
      "Games.game_date": {
        $gte: "{$start_date}",
        $lte: "{$end_date}"
      }
    }
  },
  {
    $addFields: {
      isHome: { $eq: ["$Games.home_club_id", "$club_id"] }
    }
  },
  {
    $match: {
      $expr: {
        $cond: {
          if: { $eq: ["$isHome", true] },
          then: { $gt: ["$Games.home_club_goals",
"$Games.away_club_goals"] },
          else: { $gt: ["$Games.away_club_goals",
"$Games.home_club_goals"] }
        }
      }
    }
  },
  {
    $project: {
      Formation: {
        $cond: {
          if: {
```

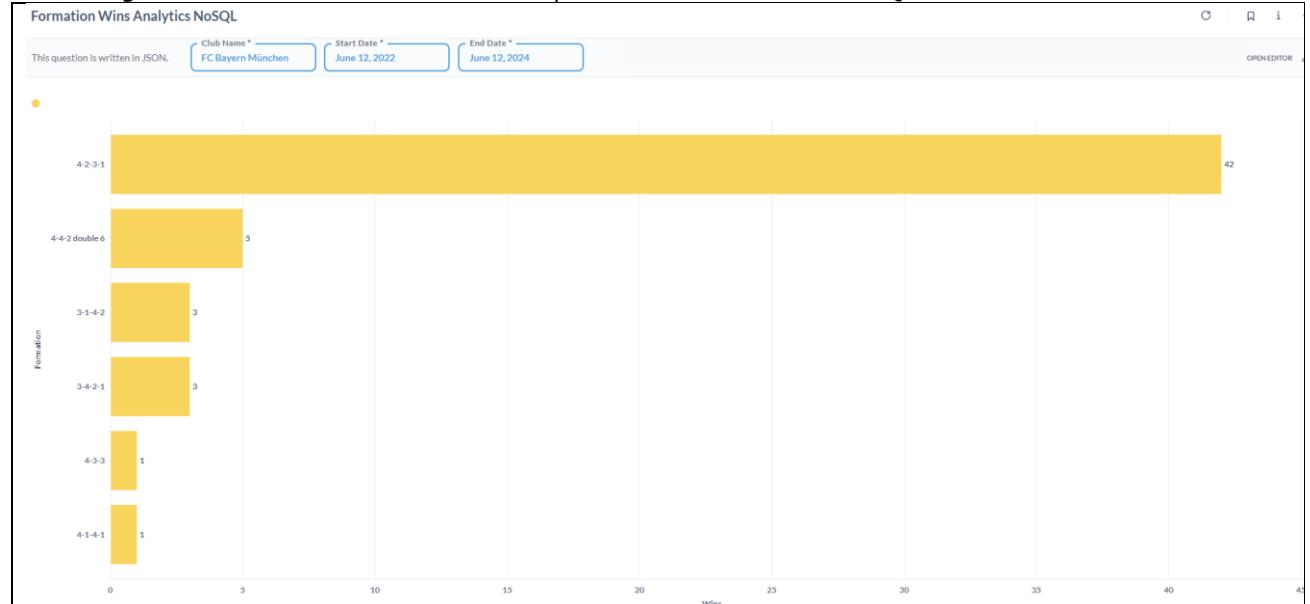
```

        $or: [
            { $and: [{ $eq: ["$isHome", true] }, { $gt:
                ["$Games.home_club_goals", "$Games.away_club_goals"] }] }
        ],
        then: "$Games.home_club_formation",
        else: "$Games.away_club_formation"
    }
}
},
{
$group: {
    _id: "$Formation",
    count: { $sum: 1 }
}
},
{
$sort: { count: -1 }
}
];

```

#### 8.4.2.1 Visualisierung

Die Konfiguration der Filter und des Row Graphen ist identisch wie bei SQL.



#### 8.4.3 Vergleich der Resultate

Aus den Graphen lässt sich schliessen, dass die Resultate identisch sind. Dies wurde auch mit anderen Clubs überprüft und waren auch da identisch.

## 8.5 Late Goals

Das Late Goals Query basiert auf den in Kapitel 5.2.2 / 5.3.2 erklärten Querys. In Metabase wird jedoch nicht nur nach einer Liga gefiltert, sondern nach einer gewissen Mannschaft. Zusätzlich wird angezeigt in wie vielen Spielen wie viele Tore nach oder während der 80 Minute erzielt wurden.

### 8.5.1 SQL

Das native Query in Metabase sieht ziemlich ähnlich aus wie jenes in Kapitel 5.2.2. Es unterscheidet sich durch den distinct count an Game IDs und der Einschränkung auf die Club ID in der Where Klausel. In der Where Klausel befinden sich auf die Metabase typische Parametrisierung mit zwei geschweiften Klammern. Die daraus folgenden Filter wurden wieder gleich konfiguriert wie beim ersten Query.

```
-- Metabase Query Late Goals

SELECT
    ge.club_id,
    c.club_name,
    Count(distinct ge.game_id) AS numberOfGames,
    COUNT(*) AS goalsAfter80
FROM
    game_events ge
    JOIN clubs c ON ge.club_id = c.club_id
    JOIN club_has_domestic_competition cdc ON ge.club_id = cdc.club_id
    JOIN game_events_has_date ged ON ge.game_event_id = ged.game_event_id
WHERE
    ge.game_event_type = 'Goals'
    AND ge.`minute` > 79
    AND c.club_name = {{club_name}}
    AND ged.game_event_date < {{end_date}}
    AND ged.game_event_date > {{start_date}}
GROUP BY
    ge.Club_id
```

### 8.5.1.1 Visualisierung

Die Filteroptionen sehen gleich aus wie im ersten Query. Mit wenigen Klicks konnten die Resultate des Querys als Bar Chart dargestellt werden. Wie unten zu sehen ist schoss der Club FC Bayern München während zwei Jahren in 33 verschiedenen Spielen insgesamt 38 Tore nach öder während der 80 Minute.



### 8.5.2 NoSQL

Das native MongoDB Query basiert wie schon erwähnt auf dem Query in Kapitel 5.3.2. Vom ClubsDocument werden die Game\_events geunwinded. Anschliessend erfolgt das Matching auf Club Name, Event Typ, Spielminute und Datumsrange. Die Resultate werden nach Club Name gruppiert.

```
//Metabase Query Formation Wins

[
  {
    $unwind: "$Game_Events",
  },
  {
    $match: {
      "club_name": "{$club_name}",
      "Game_Events.game_event_type": "Goals",
      "Game_Events.minute": { $gt: 79 },
      "Game_Events.game_event_date": {
        $gte: "{$start_date}",
        $lte: "{$end_date}"
      }
    }
  },
  {
    $group: {
      _id: "$club_name",
      total_goals_after_80: { $sum: 1 },
      distinct_games: { $addToSet: "$Game_Events.game_id" }
    }
  }
]
```

```

        }
    },
{
    $project: {
        total_goals_after_80: 1,
        distinct_games_count: { $size: "$distinct_games" }
    }
}
];

```

### 8.5.2.1 Visualisierung

Die Konfiguration der Filter und des Bar Chart erfolgte gleich wie im SQL Query.



### 8.5.3 Vergleich der Visualisierungen

Aus den beiden Graphen ist ersichtlich, dass die beiden Resultate identisch sind. Dies wurde auch mit anderen Clubs überprüft.

## 8.6 Player Scouting

Verglichen mit den Use Case Querys in Kapitel 5.2.1 und 5.3.1 sind in Metabase viel mehr Filtermöglichkeiten vorhanden und auch kombinierbar.

### 8.6.1 SQL

Für das SQL-Dashboard wurde eine SQL Question in Metabase erstellt. Hier konnten per Klicks in einem ausgesprochen übersichtlichen GUI die gewollten Daten selektiert und weitere dazu gejoined werden. spannend sind auch die sogenannten Custom Cols. Wir erstellten eine solche, welche aus dem Geburtsdatum und dem jetzigen Datum das Alter des Spielers ausrechnet. So müssen Benutzer wie Jose nicht ständig Kopfrechnen um geeignet alte oder junge Spieler zu finden.

The screenshot shows the Metabase SQL editor interface. At the top, it says "Player Infos SQL". Below that, under "Data", there is a dropdown menu set to "Players". Under "Join data", there are two dropdown menus: "Players" and "Clubs", connected by a join condition "on" and a comparison operator "= ". The "Players" table has a column "Current Club ID" and the "Clubs" table has a column "Club ID". Below these sections is a "Custom column" section. It contains a "Age" button with a minus sign and a plus sign. Underneath this, the "EXPRESSION" field contains the code "`= datetimeDiff([Date Of Birth], now, "year")`". Below the expression is a "NAME" field containing the text "Age". At the bottom of the section are "Cancel" and "Update" buttons.

#### 8.6.1.1 Filter

Die Auswahl der Filter auf dem Dashboard ist sehr breit. Die Konfiguration der Filter wurde so vorgenommen, dass sie auf die Spalte übereinstimmen und ggf. der Typ zu einer Dropdown-Liste geändert, damit der Benutzer aus vorhanden Daten auswählen kann.

### 8.6.1.2 Visualisierung

Hier wurde keine Grafik verwendet, um dem Benutzer eine gute tabellarische Übersicht zu präsentieren. Die resultierenden Daten können dann per Klick auf die gewünschte Spalte aufsteigend oder absteigend sortiert werden.

Player Scouting SQL														
Player Infos SQL														
First Name	Last Name	Country Of Birth	City Of Birth	Country Of Citizenship	Date Of Birth	Age	Clubs - Current Club	Current Club ID	Current Club Domestic Competition	Position	Sub Position	Foot	Height In Cm	
édric	Zesiger	2023	Switzerland	Meyrlez FR	Switzerland	June 24, 1998	25	Verein für Leibesübungen Wölfsburg	L1	Defender	Centre-Back	left	194	
erat	Djimulti	2023	Switzerland	Zürich	Albania	February 19, 1993	31	Atalanta Bergamasca Calcio S.p.A.	IT1	Defender	Centre-Back	right	190	
nzo	Adeagbo	2020	Switzerland	Meyrin	Switzerland	January 11, 2002	22	Società Sportiva Lazio S.p.A.	IT1	Defender	Centre-Back	right	189	
lico	Elvedi	2023	Switzerland	Zürich	Switzerland	September 30, 1999	27	Borussia Verein für Leibesübungen 1900 Mönchengladbach	L1	Defender	Centre-Back	right	189	
erkay	Sütlüngöz	2023	Switzerland	Solothurn	Türkiye	March 22, 1996	28	Pendikspor	TR1	Defender	Centre-Back	right	188	
manuel	Akanji	2023	Switzerland	Wiesendangen	Switzerland	July 19, 1995	28	Manchester City Football Club	GB1	Defender	Centre-Back	right	188	
oy	Gelmi	2020	Switzerland	Bülach	Switzerland	March 1, 1993	29	VVV-Venlo	NL1	Defender	Centre-Back	right	188	
san-Pierre	Rhymer	2021	Switzerland	Zürich	Peru	March 16, 1996	28	Neos Podosferikos Sylogos Volou	GR1	Defender	Centre-Back	left	188	
tilos	Veljkovic	2023	Switzerland	Basel	Serbia	September 26, 1995	28	Sportverein Werder Bremen von 1899	L1	Defender	Centre-Back	right	188	
audo	Decarli	2021	Switzerland	Locarno	Switzerland	February 4, 1992	32	Verein für Leibesübungen Bochum 1848 - Fußballgemeinschaft	L1	Defender	Centre-Back	right	188	
rence	Kongolo	2020	Switzerland	Fribourg	Netherlands	February 14, 1994	30	Fulham Football Club	GB1	Defender	Centre-Back	left	188	
artin	Angha	2021	Switzerland	Zürich	Switzerland	January 22, 1994	30	Fortuna Sittardia Combinatie	NL1	Defender	Centre-Back	both	188	
christian	Marques	2021	Switzerland	Uster ZH	Portugal	January 15, 2003	21	Wolverhampton Wanderers Football Club	GB1	Defender	Centre-Back	right	187	
ecir	Omeragic	2023	Switzerland	Genf	Switzerland	January 20, 2002	22	Montpellier Hérault Sport Club	FR1	Defender	Centre-Back	right	187	
tetan	Knezevic	2023	Switzerland	Luzern	Switzerland	October 30, 1996	27	Royal Charleroi Sporting Club	BE1	Defender	Centre-Back	right	187	
dimilson	Fernandes	2023	Switzerland	Sion	Switzerland	April 15, 1996	28	1. Fußball- und Sportverein Mainz 05	L1	Defender	Centre-Back	right	187	
lugo	Monteiro	2022	Switzerland	Genève	Portugal	January 28, 2005	19	Leeds United	GB1	Defender	Centre-Back	right	186	

### 8.6.2 NoSQL

Die Metabase Question für das NoSQL Dashboard wurde gleich wie in SQL konfiguriert, nur logischerweise mit anderen Daten als Grundlage.

The screenshot shows the Metabase NoSQL configuration interface for creating a question. The query is defined as follows:

```

    SELECT
        *
    FROM
        PlayersDocument
    JOIN
        ClubsDocument ON PlayersDocument.Current Club ID = ClubsDocument.Club ID
    WHERE
        Age >= 18
    ORDER BY
        Age DESC
    LIMIT
        10
  
```

The interface includes sections for "Data", "Join data", "Custom column", and "Update" buttons.

### 8.6.2.1 Filter

Leider war es uns nicht möglich alle Filter wie im SQL-Dashboard umzusetzen. Es handelt sich hier um alle FIFA-Ratings. Diese liessen sich in der Filterkonfiguration nicht anwählen. Wir versuchten in den Admin Setting die Table Metadata anzupassen, eine Custom Column zu machen oder gar ein native Query zu schreiben. Alles jedoch ohne Erfolg. Daher wurden diese ausgelassen im NoSQL Dashboard, aber die Wichtigsten Filter sind vorhanden.

### 8.6.2.2 Visualisierung

Die Visualisierung ist bis auf die oben genannten Filter identisch zum SQL-Dashboard. Es lässt sich auch Filtern und sortieren.

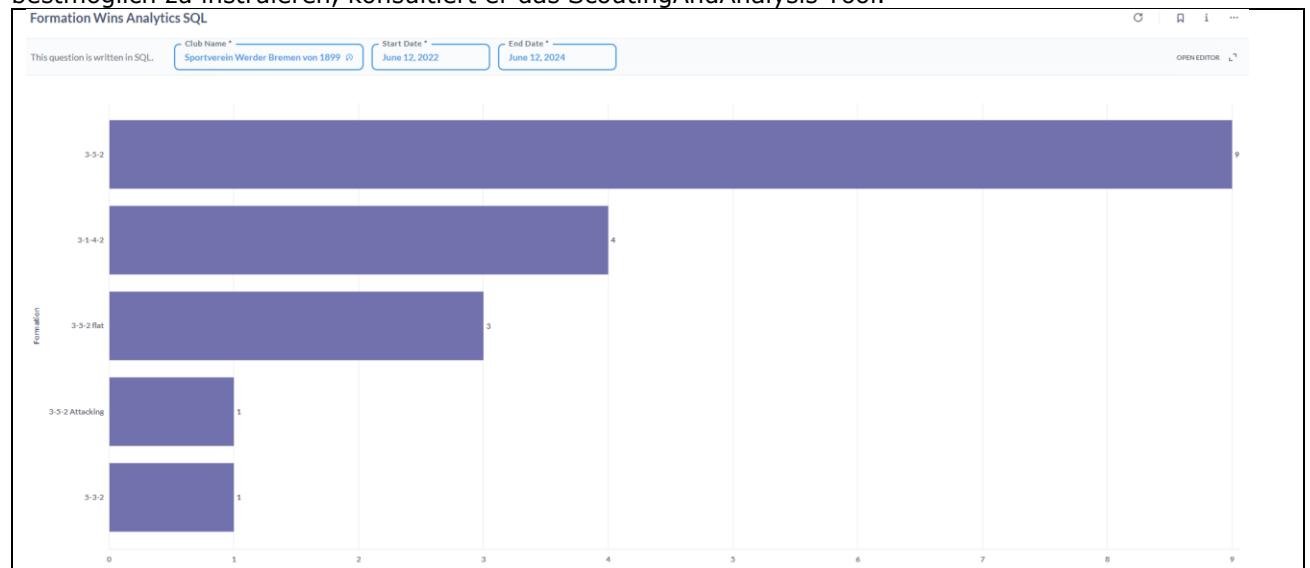
Player Scouting NoSQL														
First Name	Last Name	Country of Birth	Country of Citizenship	Age	Club Name	Club Domestic Competition	Position	Sub Position	Dominant Foot	Height	Contract Expiration Date	Agent Name	Marketvalue in Euro	
<b>Player Infos NoSQL</b>														
Cédric	Zesiger	2023	Switzerland	Meyrlez FR	Switzerland	June 24, 1998	23	Verein für Leibesübungen Wolfsburg	Defender	Centre-Back	left	194	-	IFM
Berat	Djimsiti	2023	Switzerland	Zürich	Albania	February 19, 1993	31	Atalanta Bergamasca Calcio S.p.A.	Defender	Centre-Back	right	190	-	LITCHI AG
Enzo	Adeagbo	2020	Switzerland	Meyrin	Switzerland	January 11, 2002	22	Società Sportiva Lazio S.p.A.	Defender	Centre-Back	right	189	-	
Nico	Elvedi	2023	Switzerland	Zürich	Switzerland	September 30, 1996	27	Borussia Verein für Leibesübungen 1900 Mönchengladbach	Defender	Centre-Back	right	189	-	Gol Internat
Saulo	Decarli	2021	Switzerland	Locarno	Switzerland	February 4, 1992	32	Verein für Leibesübungen Bochum 1848 - Fußballgemeinschaft	Defender	Centre-Back	right	188	-	
Berkay	Söllöngöz	2023	Switzerland	Solothurn	Türkiye	March 22, 1996	28	Pendikspor	Defender	Centre-Back	right	188	-	Hikmet Dag
Manuel	Akanji	2023	Switzerland	Wiesendangen	Switzerland	July 19, 1995	28	Manchester City Football Club	Defender	Centre-Back	right	188	-	IFM
Roy	Gelmi	2020	Switzerland	Bülach	Switzerland	March 1, 1995	29	VVV Venlo	Defender	Centre-Back	right	188	-	IFM
Jean-Pierre	Rhymer	2021	Switzerland	Zürich	Peru	March 16, 1996	28	Neos Podosferikos Syllogos Volou	Defender	Centre-Back	left	188	-	Sportfront G
Terence	Kongolo	2020	Switzerland	Fribourg	Netherlands	February 14, 1994	30	Fulham Football Club	Defender	Centre-Back	left	188	-	
Martin	Angha	2021	Switzerland	Zürich	Switzerland	January 22, 1994	30	Fortuna Sittardia Combinatie	Defender	Centre-Back	both	188	-	ROOF
Milos	Veljkovic	2023	Switzerland	Basel	Serbia	September 26, 1995	28	Sportverein Werder Bremen von 1899	Defender	Centre-Back	right	188	-	LIAN Sports
Christian	Marques	2021	Switzerland	Uster ZH	Portugal	January 13, 2003	21	Wolverhampton Wanderers Football Club	Defender	Centre-Back	right	187	-	Gestifute
Becir	Omeragic	2023	Switzerland	Genf	Switzerland	January 20, 2002	22	Montpellier Hérault Sport Club	Defender	Centre-Back	right	187	-	
Stefan	Knezevic	2023	Switzerland	Luzern	Switzerland	October 30, 1996	27	Royal Charleroi Sporting Club	Defender	Centre-Back	right	187	-	
Edimilson	Fernandes	2023	Switzerland	Sion	Switzerland	April 15, 1996	28	1. Fußball- und Sportverein Mainz 05	Defender	Centre-Back	right	187	-	IFM
Dilogo	Monteiro	2022	Switzerland	Genève	Portugal	January 28, 2003	19	Leeds United	Defender	Centre-Back	right	186	-	MRPPOSITI
Fidan	Alli	2023	Switzerland	Binningen	Kosovo	October 3, 1993	30	Alanyaspor	Defender	Centre-Back	left	186	-	

### 8.6.3 Vergleich der Visualisierungen

Abgesehen von den Diskrepanzen in der Filteranzahl ist das SQL-Dashboard spürbar performanter.

## 8.7 Entscheidungsunterstützung

Unsere Persona Jose hat am nächsten Wochenende ein Spiel gegen Werder Bremen. Um seine Mannschaft bestmöglich zu instruieren, konsultiert er das ScoutingAndAnalysis Tool.



An dem obigen Graph kann Jose erkennen, dass Werder Bremen die Formation 3-5-2 zu den meisten Siegen verholfen hat. Jose kann nun eine gute Konteraufstellung auf diese Formation wählen, wie zum Beispiel 4-3-3 oder 4-4-2 Diamant.

Bei der Late Goal Analyse stellt Jose folgendes fest:



In 17 Spielen erzielte Werder Bremen insgesamt 22 Tore nach oder während der 80 Minute.

Verglichen mit seiner Mannschaft:



Nun weiss Jose, dass das generische Team öfters Goals im späten Spielverlauf erzielt. Er kann nun seine Mannschaft für gewisse Wechsel in den Letzten Minuten bereithalten.

## **9 Verwendung von ChatGPT**

ChatGPT wurde unter anderem in Kapitel 3.2, 4.3.2, genutzt, um die CREATE TABLE und LOAD Querys zu formatieren und ergänzen.

ChatGPT haben wir ebenfalls unterstützenden genutzt, um die Querys in Kapitel 5 und 6 zu erstellen. Auch wurde ChatGPT für die Erstellung der nativen MongoDB Querys in Metabase genutzt.

## **10 Fazit & Lessons Learned**

Im Modul Datenbanken konnten wir unser Wissen und Fähigkeiten in SQL und vor allem in MongoDB ausbauen. Speziell in MongoDB, welches für beide von uns neu war haben wir eine steile Lernkurve durchlaufen. Query's in MongoDB haben wir also beide noch nicht geschrieben. Der Aufbau eines MongoDB Query und die Abfragesprache fanden wir beide interessant und werden in Zukunft, beispielsweise in Projekten rund um AI, von uns wahrscheinlich auch gut zu gebrauchen sein. Ein speziell wichtiges Learning ist auch, dass die Dokumentationen der Aufgaben direkt in die Arbeit eingefügt werden sollten und nicht erst am Schluss. Obwohl wir Screenshots gemacht und die querys gespeichert haben, wurden diese nur abgelegt und nicht direkt in die Arbeit eingepflegt. Es brauchte dann immer einen kurze Aufarbeitungszeit, um den Inhalt wieder präsent zu haben.

Ein weiteres Learning wäre auch, dass wir uns zu Beginn klarer überlegen, welche Daten wir genau benötigen. Somit hätte das Datenbankschema mit den Beschreibungen der Daten schlanker gehalten werden können. Wir waren zu Beginn zwar nicht eingeschränkt bei der konkreten Wahl der Abfragen, haben aber zum Schluss auch einige Daten, die nicht gebraucht werden wie zum Beispiel das Attribut city\_of\_birth auf dem table players.

Ein spannender Lernpunkt war auch das Visualisieren der Daten mittels Metabase. Vor dem Projekt war uns dieses Tool noch nicht vertraut und es war eindrücklich zu sehen, wie mächtig diese simpel-scheinende Tool ist. Mit wenigen Klicks konnten Daten aggregiert und visuell spannend dargestellt werden. Trotzdem hat man die Freiheit für komplexere Abfragen native Queries zu schreiben.