**Базы данных**
**Проект**
*Лим В.Е. БПИ225*
*Бийбосунов И.А. БПИ225*

## Аннотация

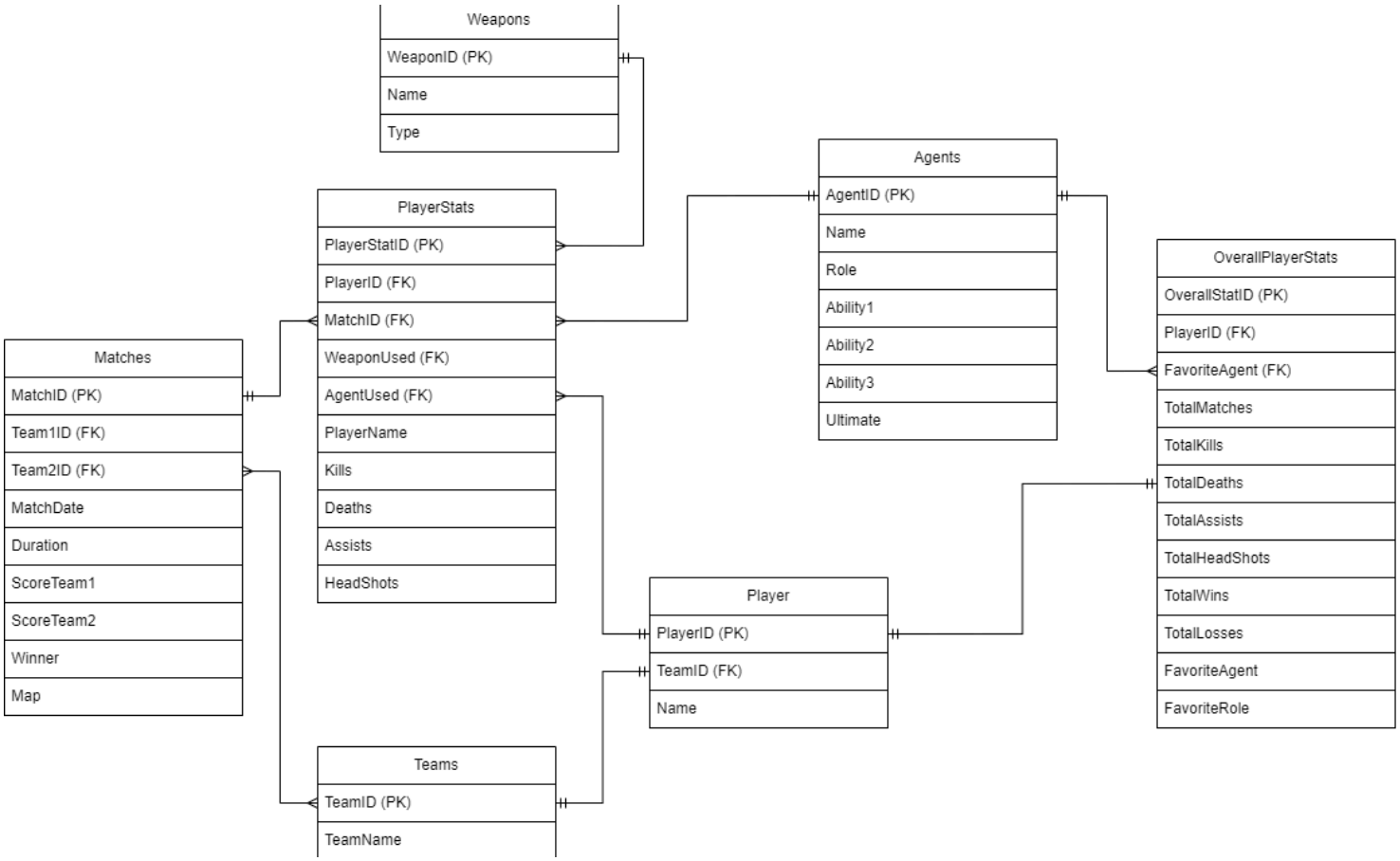
Разрабатываемый проект “Valorant Analyzer” представляет из себя систему аналитики и статистики по киберспортивной дисциплине Valorant. Valorant - это тактический онлайн шутер, матчи по которому проходят на различных картах. В игре есть несколько видов оружия, список персонажей (агентов), каждый из которых имеет свою роль (дуэлянт, контроллер, защитник, инициатор), у каждого из которых есть уникальные три способности и ультимативная способность. Игры ведутся до 13 побед в раундах или до первой разницы в 2 очка по раундам в овертаймах.

Разрабатываемая система будет полезна для тренеров и менеджеров киберспортивных команд, рядовых и профессиональных игроков, а также для киберспортивных аналитиков. С помощью собранной статистики менеджеры смогут успешно совершать трансферы в межсезонье, тренеры смогут вносить эффективные коррективы в составы и тактику, игроки смогут отслеживать свои сильные и слабые стороны, а также анализировать свою игру.

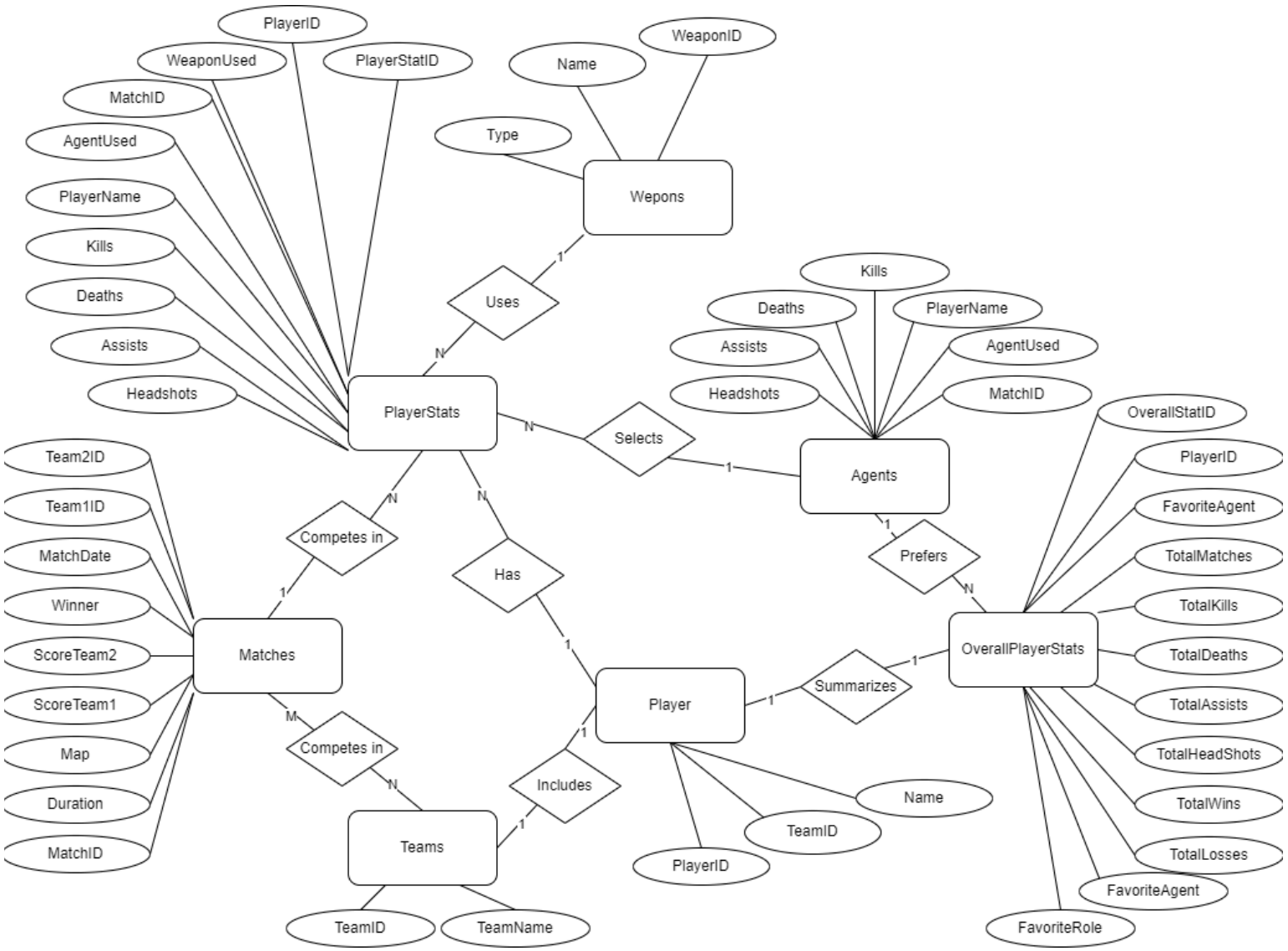
## Функциональные требования

1. Хранение данных о матчах, включая карты, режимы, счет матча, продолжительность и дату проведения.
2. Формирование общей статистики за все матчи каждого игрока: количество убийств, смертей, ассистов, хедшотов, использование различных агентов и оружия.
3. Формирование детализированной статистики по каждому матчу для каждого игрока: количество убийств, смертей, ассистов, хедшотов, использование различных агентов и оружия.

# Схема базы данных



# ER-диаграмма



## Сущности и связи

На всякий случай следует пояснить, что

- Матчи проводятся между двумя командами
- В каждой команде есть игроки
- Для проведения матча каждый игрок выбирает одного персонажа и одно оружие
- Игроки могут находиться только в одной команде в конкретный момент времени
- Команды могут играть сколько угодно матчей
- Для каждого игрока общая статистика одна и с каждым матчем она просто изменяется
- В каждом матче формируется множество статистик (по одной для каждого игрока)
- Каждый персонаж может присутствовать во многих статистиках по матчу
- В каждой общей статистике игрока может быть только один любимый персонаж
- Каждый персонаж может быть любимым у разных игроков

Таким образом связи между сущностями “Матчи”, “Игроки”, “Оружия”, “Агенты”, “Общая статистика”, “Статистика по матчу”,”Команды” понятны. Опираясь на описанные выше отношения, можно определить для сущностей их атрибуты и типы данных, которыми они будут представлены:

### Matches

- **MatchID** *(PK, SERIAL)* — уникальный идентификатор матча.
- **Team1ID** *(INT, FK)* — идентификатор первой команды.
- **Team2ID** *(INT, FK)* — идентификатор второй команды.
- **MatchDate** *(DATE)* — дата проведения матча.
- **Duration** *(INT)* — длительность матча в минутах.
- **ScoreTeam1** *(INT)* — количество выигранных раундов первой команды.
- **ScoreTeam2** *(INT)* — количество выигранных раундов второй команды.
- **Winner** *(INT)* — идентификатор команды-победителя.
- **Map** *(VARCHAR)* — название карты, на которой проводился матч.

**PlayerStats (Статистика игрока за матч)**

- **PlayerStatID** *(PK, SERIAL)* — уникальный идентификатор статистики игрока.
- **PlayerID** *(INT, FK)* — идентификатор игрока.
- **MatchID** *(INT, FK)* — идентификатор матча, к которому относятся данные.
- **WeaponUsed** *(INT, FK)* — идентификатор оружия, которое использовал игрок.
- **AgentUsed** *(INT, FK)* — идентификатор агента, выбранного игроком.
- **PlayerName** *(VARCHAR)* — имя игрока.
- **Kills** *(INT)* — количество убийств, совершенных игроком.
- **Deaths** *(INT)* — количество смертей игрока.
- **Assists** *(INT)* — количество ассистов.
- **HeadShots** *(INT)* — количество попаданий в голову.

**Weapons**

- **WeaponID** *(PK, SERIAL)* — уникальный идентификатор оружия.
- **Name** *(VARCHAR)* — название оружия.
- **Type** *(VARCHAR)* — тип оружия (например, винтовка, пистолет).

**Agents (Персонаж)**

- **AgentID** *(PK, SERIAL)* — уникальный идентификатор агента.
- **Name** *(VARCHAR)* — имя агента.
- **Role** *(VARCHAR)* — роль агента (например, дуэлянт, инициатор).
- **Ability1** *(VARCHAR)* — описание первой способности агента.
- **Ability2** *(VARCHAR)* — описание второй способности агента.
- **Ability3** *(VARCHAR)* — описание третьей способности агента.
- **Ultimate** *(VARCHAR)* — описание ультимативной способности.

**Teams**

- **TeamID** *(PK, SERIAL)* — уникальный идентификатор команды.
- **TeamName** *(VARCHAR)* — название команды.

**OverallPlayerStats (Общая статистика игрока за все матчи)**

- **OverallStatID** *(PK, SERIAL)* — уникальный идентификатор общей статистики игрока.
- **PlayerID** *(INT, FK)* — идентификатор игрока.
- **FavoriteAgent** *(INT, FK)* — идентификатор любимого агента.
- **TotalMatches** *(INT)* — общее количество матчей игрока.
- **TotalKills** *(INT)* — общее количество убийств игрока.
- **TotalDeaths** *(INT)* — общее количество смертей игрока.
- **TotalAssists** *(INT)* — общее количество ассистов игрока.
- **TotalHeadShots** *(INT)* — общее количество попаданий в голову.
- **TotalWins** *(INT)* — общее количество побед.
- **TotalLosses** *(INT)* — общее количество поражений.
- **FavoriteRole** *(VARCHAR)* — любимая роль игрока.

**Player**

- **PlayerID** *(PK, SERIAL)* — уникальный идентификатор игрока.
- **TeamID** *(INT, FK)* — идентификатор команды игрока.
- **Name** *(VARCHAR)* — имя игрока.

## Нормализация

Наша схема нормализована, так как она атомарна, у нас нет частичных зависимостей, а также транзитивных зависимостей.

Рассмотрим возможные последствия пренебрежения нормализацией нашей схемы. Пусть у нас в проекте была бы таблица matches_data, в которой мы бы хранили ***MatchID, Map, TeamID, PlayerID, Kills, Deaths, ScoreTeam1, ScoreTeam2, Assists, Duration, Date***. Тогда для одного матча в такой таблице хранилось бы 10 записей (так как в каждой команде по 5 игроков). При этом в столбцах ***MatchID, Map, Duration, Date*** в 10 строках и в столбцах ***TeamID, ScoreTeam1, ScoreTeam2*** в двух группах по 5 строках были бы одинаковые данные (то есть для записи каждого игрока из команды). То есть у нас дублировались бы данные, соответственно могли бы возникнуть следующие аномалии:

**Аномалия удаления:**

Если мы захотим удалить всех игроков какого-либо матча, то мы потеряем информацию о всем матче, которая в идеале не должна зависеть от данных об игроках. То есть целостность данных будет нарушена.

**Аномалия вставки:**

Допустим мы захотим добавить новую команду, но без сыгранного ими матча мы не сможем это сделать. Иначе придется задавать фиктивные значения для других полей, но такой подход нарушит целостность данных.

**Вывод:**

Для того, чтобы таких ситуаций не возникало, мы и поделили описываемую таблицу matches_data на те, которые описали вот здесь.

## Создание таблиц (SQL DDL)

```
-- Таблица команд
CREATE TABLE Teams (
    TeamID SERIAL PRIMARY KEY,
    TeamName VARCHAR(100) NOT NULL
);

-- Таблица игроков
CREATE TABLE Players (
    PlayerID SERIAL PRIMARY KEY,
    TeamID INT REFERENCES Teams(TeamID) ON DELETE SET
NULL,
    Name VARCHAR(100) NOT NULL
);

-- Таблица матчей
CREATE TABLE Matches (
    MatchID SERIAL PRIMARY KEY,
    Team1ID INT REFERENCES Teams(TeamID) ON DELETE
CASCADE,
```

```
    Team2ID INT REFERENCES Teams(TeamID) ON DELETE
CASCADE,
    MatchDate DATE NOT NULL,
    Duration INTERVAL NOT NULL,
    ScoreTeam1 INT NOT NULL,
    ScoreTeam2 INT NOT NULL,
    Winner INT REFERENCES Teams(TeamID),
    Map VARCHAR(100) NOT NULL
);

-- Таблица оружия
CREATE TABLE Weapons (
    WeaponID SERIAL PRIMARY KEY,
    Name VARCHAR(100) NOT NULL,
    Type VARCHAR(50) NOT NULL
);

-- Таблица агентов
CREATE TABLE Agents (
    AgentID SERIAL PRIMARY KEY,
    Name VARCHAR(100) NOT NULL,
    Role VARCHAR(50) NOT NULL,
    Ability1 VARCHAR(100),
    Ability2 VARCHAR(100),
    Ability3 VARCHAR(100),
    Ultimate VARCHAR(100)
);

-- Таблица статистики игроков за матч
CREATE TABLE PlayerStats (
    PlayerStatID SERIAL PRIMARY KEY,
    PlayerID INT REFERENCES Players(PlayerID) ON DELETE
CASCADE,
    MatchID INT REFERENCES Matches(MatchID) ON DELETE
CASCADE,
    WeaponUsed INT REFERENCES Weapons(WeaponID),
    AgentUsed INT REFERENCES Agents(AgentID),
    PlayerName VARCHAR(100) NOT NULL,
    Kills INT NOT NULL DEFAULT 0,
    Deaths INT NOT NULL DEFAULT 0,
    Assists INT NOT NULL DEFAULT 0,
```

```
    HeadShots INT NOT NULL DEFAULT 0
);

-- Таблица общей статистики игроков
CREATE TABLE OverallPlayerStats (
    OverallStatID SERIAL PRIMARY KEY,
    PlayerID INT REFERENCES Players(PlayerID) ON DELETE
CASCADE,
    FavoriteAgent INT REFERENCES Agents(AgentID),
    TotalMatches INT NOT NULL DEFAULT 0,
    TotalKills INT NOT NULL DEFAULT 0,
    TotalDeaths INT NOT NULL DEFAULT 0,
    TotalAssists INT NOT NULL DEFAULT 0,
    TotalHeadShots INT NOT NULL DEFAULT 0,
    TotalWins INT NOT NULL DEFAULT 0,
    TotalLosses INT NOT NULL DEFAULT 0,
    FavoriteRole VARCHAR(50)
);
```

## Получение общей статистики и статистики за матч (SQL DML)

### 1. Добавление матча

```
INSERT INTO Matches (MatchID, Team1ID, Team2ID,
MatchDate, Duration, ScoreTeam1, ScoreTeam2, Winner, Map)
VALUES
    (1, 1, 2, '2024-12-01', INTERVAL '45 minutes', 13, 9,
1, 'Ascent'),
    (2, 3, 4, '2024-12-02', INTERVAL '50 minutes', 13, 7,
3, 'Haven');
```

### 2. Получение общей статистики игрока

```
SELECT
    P.Name AS PlayerName,
    OPS.TotalMatches,
    OPS.TotalKills,
    OPS.TotalDeaths,
    OPS.TotalAssists,
    OPS.TotalHeadShots,
    OPS.TotalWins,
    OPS.TotalLosses,
    A.Name AS FavoriteAgent,
    OPS.FavoriteRole
FROM
    OverallPlayerStats OPS
JOIN
    Player P ON OPS.PlayerID = P.PlayerID
JOIN
    Agents A ON OPS.FavoriteAgent = A.AgentID;
```

### 3. Получение детализированной статистики по матчам

```
SELECT
    P.Name AS PlayerName,
    M.MatchID,
    M.MatchDate,
    PS.Kills,
    PS.Deaths,
```

```
    PS.Assists,
    PS.HeadShots,
    A.Name AS AgentUsed,
    W.Name AS WeaponUsed
FROM
    PlayerStats PS
JOIN
    Player P ON PS.PlayerID = P.PlayerID
JOIN
    Matches M ON PS.MatchID = M.MatchID
JOIN
    Agents A ON PS.AgentUsed = A.AgentID
JOIN
    Weapons W ON PS.WeaponUsed = W.WeaponID
ORDER BY
    M.MatchDate DESC;
```

### Пример транзакции

Пусть мы провели матч между двумя командами, нам нужно обновить данные в таблицах.
Можем объединить запросы в транзакцию таким образом:

```
BEGIN;

-- Добавляем новый матч
INSERT INTO Matches (Team1ID, Team2ID, MatchDate,
Duration, ScoreTeam1, ScoreTeam2, Winner, Map)
VALUES (1, 2, '2024-12-15', '01:30:00', 10, 8, 'Team1',
'Map1')
RETURNING MatchID INTO new_match_id;

-- Добавляем статистику для всех 10 игроков
INSERT INTO PlayerStats (PlayerID, MatchID, WeaponUsed,
AgentUsed, Kills, Deaths, Assists, HeadShots)
VALUES
(1, new_match_id, 1, 1, 5, 2, 3, 1),
(2, new_match_id, 2, 2, 3, 1, 2, 0),
(3, new_match_id, 3, 3, 4, 3, 1, 2),
(4, new_match_id, 4, 1, 2, 4, 0, 1),
```

```
(5, new_match_id, 5, 2, 1, 1, 1, 0),
(6, new_match_id, 6, 1, 6, 3, 2, 2),
(7, new_match_id, 7, 2, 2, 2, 1, 0),
(8, new_match_id, 8, 3, 3, 4, 3, 3),
(9, new_match_id, 9, 1, 0, 5, 0, 0),
(10, new_match_id, 10, 2, 1, 1, 1, 0);

-- Обновляем общую статистику игроков
WITH updated_stats AS (
    SELECT
        PlayerID,
        SUM(Kills) AS NewKills,
        SUM(Deaths) AS NewDeaths,
        SUM(Assists) AS NewAssists,
        SUM(HeadShots) AS NewHeadShots
    FROM PlayerStats
    WHERE MatchID = new_match_id
    GROUP BY PlayerID
)
UPDATE OverallPlayerStats
SET
    TotalMatches = TotalMatches + 1,
    TotalKills = TotalKills + us.NewKills,
    TotalDeaths = TotalDeaths + us.NewDeaths,
    TotalAssists = TotalAssists + us.NewAssists,
    TotalHeadShots = TotalHeadShots + us.NewHeadShots,
    TotalWins = TotalWins + CASE WHEN Winner =
OverallPlayerStats.PlayerID THEN 1 ELSE 0 END,
    TotalLosses = TotalLosses + CASE WHEN Winner !=
OverallPlayerStats.PlayerID THEN 1 ELSE 0 END
FROM updated_stats us
WHERE OverallPlayerStats.PlayerID = us.PlayerID;

COMMIT;
```