University of British Columbia, Vancouver

Department of Computer Science

---

1. **CPSC 304 Project Cover Page**

Milestone #: _2_

Date: _2024-10-13_

Group Number: _Group 106_

Name	**Student Number**	**CS Alias (Userid)**	**Preferred E-mail Address**
Ankur Bhardwaj	83640458	y8e4o	ankurb75@gmail.com
Divy Patel	82174020	s2x3p	divy07ubc@gmail.com
Oliver Shen	32805475	a3e5k	oliverdsheny@gmail.com

By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above. (In the case of Project Milestone 0, the main purpose of this page is for you to let us know your e-mail address, and then let us assign you to a TA for your project supervisor.)

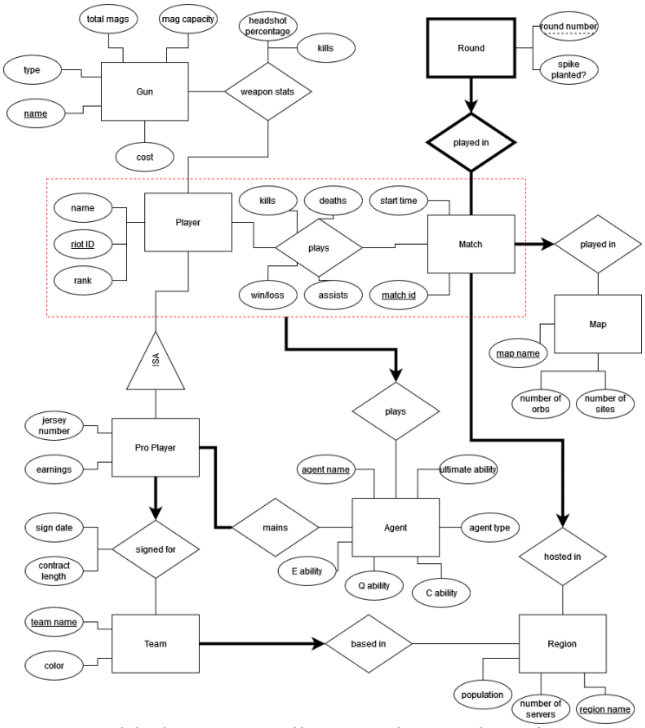
In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia

---

2. A brief (~2-3 sentences) summary of your project. Many of your TAs are managing multiple projects so this will help them remember details about your project.

   This project focuses on modeling the various entities and relationships in Valorant such as players, matches, agents, maps, teams and region. The application could be used to track and manage data related to professional players like the matches they participate in and the statistics surrounding their performances.

3. The ER diagram you are basing your item #3 (below) on. This ER diagram may be the same as your milestone 1 submission or it might be different. If you have made changes from the version submitted in milestone 1, attach a note indicating what changes have been made and why.
   If you have decided not to implement the suggestions given by your project mentor, please be sure to leave a note stating why. This is not to say that you must do everything that your project mentor says. In many instances, there are trade-offs between design choices and your decision may be influenced by different factors. Your TAs will often leave suggestions that are meant to help massage your project into a form that will fit with the requirements in future project milestones. If you choose not to take their advice, it would be helpful for them to know why to



- We added a new attribute to the weak entity “Round” as suggested in the feedback.

---

4. The schema derived from your ER diagram (above). For the translation of the ER diagram to the relational model, follow the same instructions as in your lectures. The process should be reasonably straightforward. For each table:
   a. List the table definition (e.g., Table1(attr1: domain1, attr2: domain2, ...)). Make sure to include the domains for each attribute.
   b. Specify the primary key (PK), candidate key (CK), **foreign keys (FK)**, and other constraints (e.g., not null, unique, etc.) that the table must maintain.

- Gun(gunName: str, gunType: str, totalMags: int, magCapacity: int, gunCost: int)
  - CK: {totalMags, magCapacity}
  - NOT NULL: gunType, totalMags, magCapacity, gunCost
- WeaponStats(**gunName**: str, **riotID**: str, headshotPercentage: int, kills, int)
- Map(mapName: str, numOrbs: int, numSites: int)
  - NOT NULL: numSites
- Agent (agentName: str, agentType: str, qAbility: str, eAbility: str, cAbility: str, ultimateAbility: str)
  - NOT NULL: agentType, qAbility, eAbility, cAbility, ultimateAbility
- Player (playerName: str, riotID: str, rank: int)
- PlayedInMatch(**riotID**: str, **matchID**: int, **agentName**: str, kills: int, deaths: int, won: bool, assists: int)
  - CK: {matchId, agentName, won}
  - NOT NULL: agentName
- Match(matchID: int, startTime: datetime, **regionName**: str, **mapName**: str)
  - NOT NULL: startTime, mapName
- Round (roundNumber: int, **matchID**: int, spikePlanted: bool)
- Region (population : string, numServers : int, regionName : string)
- Team (teamName : string, color : string, **regionName**: string)
- ProPlayer (riotID: string, jerseyNumber : int, earnings : int, signDate : date, contractLength : int, **teamName** : string)
  - CK: {jerseyNumber, teamName}, {signDate, contractLength, teamName}
- Mains (**riotID**: string, **agentName**: string)

---

5. Functional Dependencies (FDs)
   a. Identify the functional dependencies in your relations, including the ones involving all candidate keys (including the primary key).
   PKs and CKs are considered functional dependencies and should be included in the list of FDs. You do not need to include trivial FDs such as A →A.
   Note: In your list of FDs, there must be some kind of valid FD other than those identified by a PK or CK. If you observe that no relations have FDs other than the PK and CK(s), then you will have to intentionally add some (meaningful) attributes to show valid FDs. We want you to get a good normalization exercise. Your design must go through a normalization process. You do not need to have a non-PK/CK FD for each relation but be reasonable. If your TA feels that some non-PK/CK FDs have been omitted, your grade will be adjusted accordingly.

- Gun(gunName: str, gunType: str, totalMags: int, magCapacity: int, gunCost: int)
  - gunName -> gunType, totalMags, magCapacity, gunCost
  - gunCost, totalMags -> gunType
  - totalMags, magCapacity -> gunName, gunType, gunCost
- WeaponStats(**gunName**: str, **riotID**: str, headshotPercentage: int, kills, int)
  - gunName, riotID -> headshotPercentage, kills
- Map(mapName: str, numOrbs: int, numSites: int)
  - mapName -> numOrbs, numSites
  - numSites -> numOrbs
- Agent (agentName: str, agentType: str, qAbility: str, eAbility: str, cAbility: str, ultimateAbility: str)
  - agentName -> agentType, qAbility, eAbility, cAbility, ultimateAbility
- Player (playerName: str, riotID: str, rank: int)
  - riotID -> playerName, rank
- Match(matchID: int, startTime: datetime, **regionName**: str, **mapName**: str)
  - matchID -> startTime, regionName, mapName
- PlayedInMatch(**riotID**: str, **matchID**: int, **agentName**: str, kills: int, deaths: int, won: bool, assists: int)
  - matchID, riotID -> kills, deaths, won, assists
  - agentName, matchID, won -> riotID
- Region (population: int, numServers : int, regionName : string)
  - regionName -> numServers, population
  - numServers -> population
- Team (teamName : string, color : string, **regionName**: string)

- teamName -> color, regionName

- ProPlayer (riotID : string, jerseyNumber : int, earnings : int, signDate : date, contractLength : int, **teamName** : string)
  - riotID -> jerseyNumber, earnings, signDate, contractLength, teamName
  - jerseyNumber, teamName -> riotID
  - signDate, contractLength, teamName -> jerseyNumber
- Mains (**riotID** : string, **agentName** : string)
  - riotID -> agentName
- Round (roundNumber: int, **matchID**: int, spikePlanted: bool)
  - roundNumber, matchID -> spikePlanted

6. Normalization

a. Normalize each of your tables to be in 3NF or BCNF. Give the list of tables, their primary keys, their candidate keys, and their foreign keys after normalization. You should show the steps taken for the decomposition in a manner similar to that done in class. Should there be errors, and no work is shown, no partial credit can be awarded without steps shown.
The format should be the same as Step 3, with tables listed similar to Table1(attr1:domain1, attr2:domain2, …). ALL Tables must be listed, not only the ones post normalization.

The tables that's not in BCNF:

- Gun(gunName: str, gunType: str, totalMags: int, magCapacity: int, gunCost: int)

gunName -> gunType, totalMags, magCapacity, gunCost

gunCost, totalMags -> gunType

totalMags, magCapacity -> gunName, gunType, gunCost

$gunName^+ = gunName, gunType, totalMags, magCapacity, gunCost$ (key)

$\text{gunCost, totalMags}^+ = \text{gunCost, totalMags, gunType}$

$totalMags, magCapacity^+ =$

$gunName, gunType, totalMags, magCapacity, gunCost$ (key)

$\text{gunCost, totalMags}^+$ is not in BCNF, decompose:

magCapacity, gunName | gunCost,totalMags | gunType

So the new tables are:

Gun_1(gunName: str, magCapacity:int, gunCost:int)

Gun_2(gunCost: int, totalMags: int, gunType: str)

- Map(mapName: str, numOrbs: int, numSites: int)
  mapName -> numOrbs, numSites; numSites -> numOrbs
  $\text{mapName}^+ = \text{mapName}, \text{numOrbs}, \text{numSites}$ (key)
  $\text{numSites}^+ = \text{numSites}, \text{numOrbs}$
  numSites^+ is not in BCNF, decompose:
  mapName numSites numOrbs
  So $Map_1(\text{numSites}, \text{numOrbs})$, $Map_2(\text{mapName}, \text{numSites})$
  The new tables are:
  Map_1(numSites: int, numOrbs: int)
  Map_2(mapName: str, numSites: int)

- Region (population: int, numServers : int, regionName : string)
  regionName -> numServers, population; numServers -> population
  $\text{regionName}^+ = \text{regionName}, \text{numServers}, \text{population}$ (key)
  $\text{numServers}^+ = \text{numServers}, \text{population}$ (key)
  numServers^+ is not in BCNF, decompose:
  regionName numServers population
  So $\text{Region}_1(\text{regionName}, \text{numServers})$ $\text{Region}_2(\text{numServers}, \text{population})$
  The new tables are:
  Region_1(regionName: string, numServers: int)
  Region_2(numServers: int, population: int)

- PlayedInMatch(**riotID**: str, **matchID**: int, **agentName**: str, kills: int, deaths: int, won: bool, assists: int)
  matchID, riotID -> kills, deaths, won, assists; agentName, matchID, won -> riotID
  $\text{matchID}, \text{riotID}^+ = \text{matchID}, \text{riotID}, \text{kills}, \text{deaths}, \text{won}, \text{assists}$ (key)
  $\text{agentName}, \text{matchID}, \text{won}^+ =$
  $\text{matchID}, \text{riotID}, \text{matchID}, \text{riotID}, \text{kills}, \text{deaths}, \text{won}, \text{assists}$ (key)
  Every closure is a key, this is already in BCNF, no need to decompose

- ProPlayer (riotID : string, jerseyNumber : int, earnings : int, signDate : date, contractLength : int, **teamName** : string)
  riotID -> jerseyNumber, earnings, signDate, contractLength, teamName
  jerseyNumber, teamName -> riotID
  signDate, contractLength, teamName -> jerseyNumber
  $\text{riotID}^+ = \text{riotID}$, $\text{jerseyNumber}, \text{earnings}, \text{signDate}, \text{contractLength}, \text{teamName}$

(key)

$\text{jerseyNumber, teamName}^{+} =$

riotID , jerseyNumber, earnings, signDate, contractLength, teamName (key)

$\text{signDate, contractLength, teamName}^{+} =$

riotID , jerseyNumber, earnings, signDate, contractLength, teamName (key)

Every closure is a key, this is already in BCNF, no need to decompose

The decomposed list of tables are:

- Gun_1(gunName: str, **magCapacity**:int, **gunCost**:int)
- Gun_2(gunCost: int, totalMags: int, gunType: str)
- WeaponStats(**gunName**: str, **riotId**: str, headshotPercentage: int, kills, int)
- Map_1(numSites: int, numOrbs: int)
- Map_2(mapName: str, **numSites**: int)
- Agent (agentName: str, agentType: str, qAbility: str, eAbility: str, cAbility: str, ultimateAbility: str)
    - NOT NULL: agentType, qAbility, eAbility, cAbility, ultimateAbility
- Player (playerName: str, riotID: str, rank: int)
- PlayedInMatch(**riotId**: str, **matchId**: int, **agentName**: str, kills: int, deaths: int, won: bool, assists: int)
    - CK: {matchId, agentName, won}
    - NOT NULL: agentName
- Match(matchId: int, startTime: datetime, **regionName**: str, **mapName**: str)
    - NOT NULL: startTime, mapName
- Round (roundNumber: int, **matchId**: int, spikePlanted: bool)
- Region_1(regionName: string, **numServers**: int)
- Region_2(numServers: int, population: int)
- Team (teamName : string, color : string, **regionName**: string)
- ProPlayer (riotID: string, jerseyNumber : int, earnings : int, signDate : date, contractLength : int, **teamName** : string)
    - CK: {jerseyNumber, teamName}, {signDate, contractLength, teamName}
- Mains (**riotID**: string, **agentName**: string)

---

7. The SQL DDL statements required to create all the tables from item #6. The statements should use the appropriate foreign keys, primary keys, UNIQUE constraints, etc. Unless you know that you will always have exactly x characters for a given character, it is better to use the VARCHAR data type as opposed to a CHAR(Y). For example, UBC courses always use four characters to represent which department offers a course. In that case, you will want to use CHAR(4) for the department attribute in your SQL DDL statement. If you are trying to represent the name of a UBC course, you will want to use VARCHAR as the number of characters in a course name can vary greatly.

```
CREATE TABLE Gun1
(
    gunName         VARCHAR PRIMARY KEY,
    magCapacity     INTEGER NOT NULL,
    gunCost         INTEGER NOT NULL,
    FOREIGN KEY (magCapacity) REFERENCES Gun2(magCapacity)
        ON DELETE SET NULL
        ON UPDATE CASCADE
    FOREIGN KEY (gunCost) REFERENCES Gun2(gunCost)
        ON DELETE SET NULL
        ON UPDATE CASCADE
);

CREATE TABLE Gun2
(
    gunCost INTEGER,
    totalMags INTEGER,
    gunType VARCHAR NOT NULL,
    PRIMARY KEY (gunCost, totalMags)
);

CREATE TABLE WeaponStats
(
    gunName             VARCHAR,
    riotId              VARCHAR,
    headshotPercentage  INTEGER,
    kills               INTEGER,
    PRIMARY KEY (gunName, riotId)
    FOREIGN KEY (gunName) REFERENCES Gun(gunName)
        ON DELETE CASCADE
```

```
        ON UPDATE CASCADE
    FOREIGN KEY (riotId) REFERENCES Player(riotId)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);

CREATE TABLE Map1
(
    numSites INTEGER PRIMARY KEY,
    numOrbs INTEGER
);

CREATE TABLE Map2
(
    mapName     VARCHAR PRIMARY KEY,
    numSites    INTEGER NOT NULL,
    FOREIGN KEY (numSites) REFERENCES Map1(numSites)
        ON DELETE SET NULL
        ON UPDATE CASCADE
)

CREATE TABLE Agent
(
    agentName         VARCHAR PRIMARY KEY,
    agentType         VARCHAR NOT NULL,
    qAbility          VARCHAR NOT NULL,
    eAbility          VARCHAR NOT NULL,
    cAbility          VARCHAR NOT NULL,
    ultimateAbility   VARCHAR NOT NULL
);

CREATE TABLE Player
(
    riotId        VARCHAR PRIMARY KEY,
    playerName    VARCHAR,
    rank          INTEGER
);

CREATE TABLE PlayedInMatch
(
    riotId        VARCHAR,
```

```
    matchId         INTEGER,
    agentName       VARCHAR NOT NULL,
    kills           INTEGER,
    deaths          INTEGER,
    won             BOOLEAN,
    assists         INTEGER,
    PRIMARY KEY (riotId, matchId)
    FOREIGN KEY (riotId) REFERENCES Player(riotId)
        ON DELETE SET NULL
        ON UPDATE CASCADE
    FOREIGN KEY (matchId) REFERENCES Match(matchId)
        ON DELETE CASCADE
        ON UPDATE CASCADE
    FOREIGN KEY (agentName) REFERENCES Agent(agentName)
        ON DELETE SET NULL
        ON UPDATE CASCADE
);

CREATE TABLE Match
(
    matchId         INTEGER PRIMARY KEY,
    startTime       DATETIME NOT NULL,
    regionName      VARCHAR,
    mapName         VARCHAR NOT NULL,
    FOREIGN KEY (regionName) REFERENCES Region(regionName)
        ON DELETE SET NULL
        ON UPDATE CASCADE
    FOREIGN KEY (mapName) REFERENCES Map(mapName)
        ON DELETE SET NULL
        ON UPDATE CASCADE
);

CREATE TABLE Round
(
    roundNumber     INTEGER,
    matchId         INTEGER,
    spikePlanted    BOOLEAN,
    PRIMARY KEY (roundNumber, matchId)
    FOREIGN KEY (matchId) REFERENCES Match(matchId)
        ON DELETE CASCADE
        ON UPDATE CASCADE
```

```
);

CREATE TABLE Region1
(
    regionName   VARCHAR PRIMARY KEY,
    numServers   INTEGER,
    FOREIGN KEY (numServers) REFERENCES Region2(numServers)
        ON DELETE SET NULL
        ON UPDATE CASCADE
);

CREATE TABLE Region2
(
    numServers   INTEGER PRIMARY KEY,
    population   INTEGER
);

CREATE TABLE Team
(
    teamName     VARCHAR PRIMARY KEY,
    color        VARCHAR,
    regionName   VARCHAR,
    FOREIGN KEY (regionName) REFERENCES Region(regionName)
        ON DELETE SET NULL
        ON UPDATE CASCADE
);

CREATE TABLE ProPlayer
(
    riotId           VARCHAR PRIMARY KEY,
    jerseyNumber     INTEGER,
    earnings         INTEGER,
    signDate         DATETIME,
    contractLength   INTEGER,
    teamName         VARCHAR,
    FOREIGN KEY (teamName) REFERENCES Team(teamName)
        ON DELETE SET NULL
        ON UPDATE CASCADE
);

CREATE TABLE Mains
```

```
(
    riotID          VARCHAR,
    agentName       VARCHAR,
    PRIMARY KEY (riotId, agentName)
    FOREIGN KEY (riotId) REFERENCES ProPlayer(riotId)
        ON DELETE CASCADE
        ON UPDATE CASCADE
    FOREIGN KEY (agentName) REFERENCES Agent(agentName)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
```

8. INSERT statements to populate each table with at least 5 tuples. You will likely want to have more than 5 tuples so that you can have meaningful queries later.

```
INSERT INTO Gun1 (gunName, magCapacity, gunCost)
VALUES
        ("Vandal", 25, 2900),
        ("Sheriff", 6, 800),
        ("Outlaw", 2, 2400),
        ("Odin", 100, 3400),
        ("Spectre", 30, 1600)

INSERT INTO Gun2 (gunCost, totalMags, gunType)
VALUES
        (2900, 3, "Rifle"),
        (800, 5, "Pistol"),
        (2400, 6, "Sniper"),
        (3400, 3, "Machine gun"),
        (1600, 4, "SMG")

INSERT INTO WeaponStats (gunName, riotID, headshotPercentage, kills)
VALUES
        ("Vandal", "ABC", 18, 22),
        ("Sheriff", "ABC", 13, 4),
        ("Outlaw", "XYZ", 8, 0),
        ("Odin", "PQR", 4, 14),
```

```
        ("Vandal", "PQR", 23, 34)

INSERT INTO Map1 (numSites, numOrbs)
VALUES
        (2, 3),
        (3, 3),
        (1, 2),
        (4, 4),
        (5, 4)

INSERT INTO Map2 (mapName, numSites)
VALUES
        ("Haven", 3),
        ("Sunset", 2),
        ("Abyss", 1),
        ("Lotus", 4),
        ("Fracture", 5)

INSERT INTO Agent (agentName, agentType, qAbility, eAbility, cAbility, ultimateAbility)
VALUES
        ("Cypher", "Sentinel", "Cyber Cage", "Spycam", "Trapwire", "Neural Theft"),
        ("Gekko", "Initiator", "Wingman", "Dizzy", "Mosh Pit", "Thrash"),
        ("Viper", "Controller", "Poison Cloud", "Toxic Screen", "Snake Bite", "Viper's
Pit"),
        ("Omen", "Controller", "Paranoia", "Dark Cover", "Shrouded Step", "From the
Shadows"),
        ("Jett", "Duelist", "Updraft", "Tailwind", "Cloudburst", "Blade Storm")

INSERT INTO Player (riotID, playerName, rank)
VALUES
        ("ABC", "Mike", "Platinum"),
        ("XYZ", "Zach", "Bronze"),
        ("PQR", "Tyson", "Radiant"),
        ("MNO", "Jordan", "Ascendant"),
        ("IJK", "Trent", "Platinum")

INSERT INTO PlayedInMatch (riotID, matchID, agentName, kills, deaths, won, assists)
VALUES
        ("ABC", 123, "Cypher", 24, 17, TRUE, 11),
```

```
        ("PQR", 123, "Viper", 16, 25, FALSE, 2),
        ("XYZ", 456, "Jett", 8, 25, FALSE, 5),
        ("PQR", 456, "Jett", 31, 9, TRUE, 2),
        ("MNO", 123, "Gekko", 17, 18, TRUE, 9)

INSERT INTO Match (matchID, startTime, regionName, mapName)
VALUES
        (123, 2024-07-22 18:32:12, "EMEA", "Lotus"),
        (456, 2024-09-18 22:11:34, "Pacific", "Haven"),
        (616, 2023-12-25 00:20:48, "North America", "Fracture"),
        (982, 2024-09-01 09:25:22, "EMEA", "Abyss"),
        (438, 2024-09-18 22:12:01, "Pacific", "Sunset")

INSERT INTO Round (roundNumber, matchId, spikePlanted)
VALUES
        (1, 123, TRUE),
        (3, 982, TRUE),
        (6, 456, FALSE,
        (2, 616, FALSE),
        (5, 438, TRUE)


INSERT INTO Region1(regionName, numServers)
VALUES
        ("EMEA", 42),
        ("Pacific", 10),
        ("Asia", 30)
        ("South America", 23)
        ("North America", 28)

INSERT INTO Region2 (numServers, population)
VALUES
        (42, 42000)
        (10, 10000)
        (30, 30000)
        (23, 23000)
        (28, 28000)

INSERT INTO Team (teamName, color, regionName)
VALUES
```

("Team Racket", "WHITE", "Asia")
("Fantastic Five", "BLUE", "North America")
("Bulletproof Girl", "PURPLE" , "Asia")
("We hate Mr. Beast", "YELLOW" , "EMEA")
("Ninjas in pajamas", "PINK", "Pacific")

INSERT INTO ProPlayer (riotID, jerseyNumber, earnings, signDate, contractLength, teamName)
VALUES
("PQR", 10, 430000, 2023-03-10 14:12:10, 2, "Shadow Wizard Money Gang")
("XQC", 42, 114514, 2021-11-12 12:00:00, 8, "Too cool to be Vtubers")
("FINN", 31, 1610421, 2022-04-21 09:08:23, 5, "Turtles in Ninjas pajamas")
("MRB", 66, 100000000, 2023-05-04 07:21:21, 8, "The Jedi Order")
("SPEED", 25, 239999, 2024-02-28 05:12:34, "We LOVE Mr. Beast")

INSERT INTO Mains (riotID, agentName)
VALUES
("PQR", "Cypher")
("XQC", "Jett")
("FINN", "Yoru")
("MRB", "Chamber")
("SPEED", "Neon")