# PROJECT PHASE 1

## 1. Introduction to the mini-world
We have made a mini world of the popular anime series – ‘Naruto’. With the reference to the series, we have made a database that keeps track of missions, client, and ninja data and keep track of mission logistics.

## 2. Purpose of the database
To keep track of the missions given by the client, the ninja who is doing missions, the organization, etc.

## 3. Users of the database
- Village leader (Kage)
- Team leader
- Team member
- Client

## 4. Applications of the database
- Village leader can view and track of all the missions, clients, ninjas, etc. In short, village leaders have access to all the information.
- Team leader has access to the details about his/her team
- Clients can view and keep a track of their missions.

## 5. Database Requirements

a. Assumptions
- Ninjas have unique code names.
- Each ninja has been provided an ID that is unique from all other ninjas in the village.
- We have assumed VILLAGE MEMBER as a class and then NINJA can be its subclass. {we haven’t mentioned it in the doc for now}
- Clients have unique SSN.

## b. Strong entity types

| NINJA | MISSION | TEAM | CLIENT |
|---|---|---|---|
| • Name | • mission_no | • team no. | • Name |
| • id no. | • mission_info | • team_name | • ssn no. |
| • DOB | • mission_name | • leader_id | • address |
| • age | • team assigned | • no. of members | |
| • team no. | • status | | |
| • rank | • client | | |
| • start date | • cost | | |
| • weapons | • mission rank | | |
| • summons | | | |
| • code name | | | |
| • leader_id | | | |

In NINJA: id_no and code_name are candidate keys, team_no is foreign key referencing TEAM. Age is derived attribute, Name is composite attribute and summmons and weapons are multi-valued attributes.
In MISSION: Mission_no is primary key, team_assigned is foreign referencing TEAM and client is also a foreign key referencing CLIENT.
In TEAM: Team_no is primary key, leader_id is foreign key referencing NINJA.
In CLIENT : ssn_no is primary key and Name and Address are composite attributes.

## c. Weak entity types

| WEAPONS | SUMMONS |
|---|---|
| • name | • name |
| • mfg date | • ***owner_id*** |
| • type | • species |
| • ***owner_id*** | • residence |

In WEAPONS and SUMMONS: owner_id is foreign key referencing NINJA.

## d. Relationship types

1. **<u>Ninja-Team</u>**: Ninja **BELONGS TO** Team
2. **<u>Ninja- Ninja(leader)</u>**: Ninja **HAS A** leader

3. **Ninja-Weapons-Summons**: Ninja **OWNS** Weapons and Summons
4. **Client-Mission**: Client **REQUESTS** Mission
5. **Team-Mission**: Team **IS ASSIGNED** Mission

   (For subclass: Ninja **IS A** Village Member) [Will be expanded upon later when a subclass is covered]

i. **Max Degree:** 3

ii. **Participating entity types:**

   Ninja-Team [Degree=2]
   Ninja- Ninja(leader) [Degree=2] (Recursive Relationship).
   Ninja-Weapons-Summons [Degree=3]
   Client-Mission [Degree=2]
   Team-Mission [Degree=2]

e. Cardinality ratio/ Participation constraint/ (min, max) constraint

1. **Ninja-Village Member** [Subclass]
2. **Ninja-Team** [N:1 relation]
3. **Ninja-Ninja(leader)**[N:1 relation]
4. **Ninja-Weapons-Summons** [1: N relation]
5. **Client-Mission** [1: N relation]
6. **Team-Mission** [1: N relation]

f. Degree = 3 relationship type

Ninja **OWNS** Weapons and Summons

## 6. Functional Requirements

**MODIFICATIONS:**

### 1. INSERT

- Insert_ninja: inserts ninja with corresponding team_number and leader_id. Check to make sure that each team has exactly one leader.
- Insert_mission: inserts new mission. Check that the client ssn for the mission exists in client table.
- Insert_team: inserts a new team. Check that leader rank is not genin.
- Insert_weapon/ Insert_summons : Insert weapon/ summon. Check that owner exists in ninja table.

### 2. UPDATE:

- Update_mission_status: updates the status of the mission starting from its assignment till its completion(successful/failure).
- Update_ninja_rank: Update the rank of a ninja if he/she gets promoted.

**3. DELETE:**

- Delete_weapons: deletes broken weapons that no longer can be used.

***RETRIEVALS:***

1. SELECTION
    - Retrieve ninja entries where the rank is Genin (or Chunin/ Jounin).
    - Retrieve missions where mission status is ‘Ongoing’.

2. PROJECTION:
    - Retrieve names of teams with more than 5 members.
    - Retrieve names and team leader of teams with no Jounin.

3. AGGREGATE
    - Min_mission_cost : displays least mission_cost.
    - Max_mission_cost : displays maximum mission_cost.
    - Avg_mission_cost : Displays avg.

mission cost

4. SEARCH
    - Search_Team_name_xyz: searches and lists all the team names whose names contain the string ‘xyz’.
    - Search_Code_Name_n: searches and lists out all the code names of ninjas whose names start with the letter ‘n’.

5. ANALYSIS

- We can obtain the list of ninjas and their rankings after successfully completing each mission. The top 10 ninjas who have completed their mission successfully will be listed.
- Display client’s names and details with total mission costs (of all missions requested by that client) greater than the average mission cost.

**Constraints**

- Each team must have exactly one leader.
- Each mission can be assigned to only one team.
- Team leader rank cannot be Genin.

## 7. Summary

The given database will be of use in keeping track of the mission logistics and ninja performance.