**Genshin – SQL DB Report**

**Table of Content**

1. **Description**
2. **Tablets**
   1. **Elements**
   2. **Weapon\_types**
   3. **Artefact\_types**
   4. **Weapons**
   5. **Artefacts**
   6. **Forge**
   7. **Domains**
   8. **Characters**
   9. **Players**
3. **Entity Relationship Diagram**
4. **Mapping to Functional Schema**
5. **Functional Dependency Diagram**
6. **Semantic Constrain**
   1. **Key Constrains**
   2. **Attributes Constrains**
   3. **Cascades**

**7.4 Triggers**

1. **View & Security**
   1. **Views**
   2. **Roles**
2. **Appendix**

**Description**

I have chosen to implement a database based on “Genshin Impact”. The database could be used to keep entries of objects, provide a quick overview to each players’ state and suggestions for character building.

**Tables**

This database contains 9 tables.

**Elements** contains entries of unique elements.   
Most entity in this game is assigned an element.  
The table documents elements’ unique id, in-game name and elemental effects.

**Weapon\_types** contains entries of different weapon types.

Different characters have restriction for what weapons they could equip based on their respective weapon type.

The table documents their unique id and their in-game name.

**Artefact\_types** contains entries of different artefacts types.

Different characters have restriction for what artefacts they could equip based on their respective weapon type.

The table documents their unique id and their in-game name.

**Weapons** contains entries of different weapons.

Characters can equip weapons to enhance their stats.

All weapons in entries must be of type documented in **Weapon\_types** (Foreign Key).

The damage provided by the weapon(s) must also be positive.

The table documents weapons’ unique id, their in-game name, their damage and their weapon type.

**Artefacts** contains entries of different artefacts.

Characters can equip artefacts to enhance their stats.

All artefacts in entries must be of type documented in **Artefact\_types** (Foreign Key).

The damage provided by the artefact(s) must also be positive.

The table documents artefacts’ unique id, their in-game name, their damage and their artefact type.

**Forge** contains entries of different crafting recipes.

Player can craft weapons in the forge using in-game currency (mora).

Any weapons that could be crafted must be documented in **Weapons** (Foreign Key).

The success chance for a craft must also be within 0 and 100 inclusively.

Craft cost is defaulted to 0 but allows negative cost for additional complexity.

The table documents crafting recipes’ unique id, weapons to craft, mora cost and crafting success chance.

**Domains** contains entries of different domains.

Domains can drop in-game currency (mora) and artefacts.

Any artefacts that could be dropped must be documented in **Artefacts** (Foreign Key).

Mora drop and artefact drop chance is defaulted to 0.

Negative mora drop is allowed to stimulate entry cost (when applicable) and drop chance must be within 0 and 100 inclusively.

The table documents domains’ unique id, name, mora drop, artefact drop and artefact drop chance.

**Characters** contains entries of different playable characters.

Player can select a character and equip items on said characters.

Characters’ element must be documented in **Elements** (Foreign Key), a character can only equip weapons and artefacts of their respective types.

A character’s damage must also be positive.

The table documents characters’ unique id, name, damage, weapon type, artefact type and element.

**Players** contains entries of different players with their selected character, weapon and artefact.

The table documents players’ unique id, name, mora balance, selected character, equipped weapon and artefact.

**Entity Relationship Diagram**

Diagram

Description automatically generated

**Mapping to Relational Schema**

**Diagram

Description automatically generated**

**Functional Dependency Diagram**

Timeline

Description automatically generated

**Semantic Constraint**

**Key Constrains**

Every table has a Primary Key that is unique, and not null.

This is enforced with **AUTO\_INCREMENT** keyword, which auto increments every entry without us specifying a value.

**Foreign key** was also declared.

Depending on the situation, a **NOT NULL** keyword pair is also added.

One such example is weapon\_type in **Weapons**, the system dictates that no weapons of undocumented type in **Weapon\_types** is allowed.

A counterexample is player\_artefact in **Players**, a player can have a character without equipping artefact, and such NULL value is allowed.

**Attributes Constraints**

Most attributes were constrained to be not null with **NOT NULL** keyword pair.

Some were allowed a default value using the **DEFAULT** keyword.

Some checks were also done to ensure that an entry is within valid range, e.g. craft\_success\_chance within 0 to 100 inclusively with  
CHECK (craft\_success\_chance BETWEEN 0 AND 100)

Some attributes constrain:

Weapon damage is positive

CHECK (weapon\_damage > 0)

Artefact damage boost is positive

CHECK (art\_damage > 0)

Domain artefact drop chance within 0 to 100 inclusively

CHECK (domain\_drop\_chance BETWEEN 0 AND 100)

Character damage is positive

CHECK (char\_damage > 0)

Player in-game balance (mora) is positive

CHECK (player\_mora >= 0)

**Cascades**

**Cascade**(s) where also added to maintain the integrity of the database.

In **Forge**

FOREIGN KEY (craft\_item)

REFERENCES Weapons (weapon\_id) ON DELETE CASCADE

For example, deletes the tuple in case where the weapon\_id is removed, this made logical sense as there is no need for a recipe if the item to be forged is removed.

In **Players**

FOREIGN KEY (player\_artefact)

REFERENCES Artefacts (art\_id) ON DELETE SET NULL

However, this sets player\_artefact to **NULL** in case art\_id is removed, this made logical sense as we allow players to have characters without artefacts.

**Triggers**

There is one trigger defined. This trigger adds a suffix to duplicated player\_name.

E.g adding a duplicated ‘Name’ as player\_name would turn the duplicated to ‘Name1’.

CREATE TRIGGER change\_player\_name

BEFORE INSERT ON Players

FOR EACH ROW

BEGIN

DECLARE original\_player\_name VARCHAR(25);

DECLARE suffix\_counter INT;

SET original\_player\_name = NEW.player\_name;

SET suffix\_counter = 1;

WHILE EXISTS (SELECT TRUE FROM Players WHERE player\_name = NEW.player\_name) DO

SET NEW.player\_name = concat(original\_player\_name, suffix\_counter);

SET suffix\_counter = suffix\_counter + 1;

END WHILE;

END //

**View & Security**

**Views**

There are two views defined for this project, **Quickview** and **Build\_recommendations**.

The views are created using CREATE VIEW viewname AS SELECT …

**Quickview** provides a quick overview into the state of each player(s) whereas **Build\_recommendations** provides recommendations based on players’ selected characters.

**Roles**

There are four roles in this implementation, **Dbmaster**, **Developer**, **Gamemaster** and **Player**.

Dbmaster has complete control of the database.

A Dbmaster can also grant privileges to other roles.

CREATE ROLE Dbmaster

GRANT ALL PRIVILEGES ON \* TO Dbmaster WITH GRANT OPTION;

Developer has access to every tables.

CREATE ROLE Developer

GRANT ALL PRIVILEGES ON \* TO Developer;

Gamemaster has full access to both views and SELECT privileges on all other tables.

CREATE ROLE Gamemaster

GRANT SELECT ON \* TO Gamemaster

GRANT ALL PRIVILEGES ON Quickview TO Gamemaster;

GRANT ALL PRIVILEGES ON Build\_recommendations TO Gamemaster;

Player has SELECT access to all tables.

CREATE ROLE Player

GRANT SELECT ON \* TO Player

**Appendix**

**CREATE DATABASE IF NOT EXISTS Genshin;**

**USE Genshin;**

**# Create tables**

**CREATE TABLE Elements (**

**element\_id INT NOT NULL UNIQUE AUTO\_INCREMENT,**

**element\_name VARCHAR(25) NOT NULL UNIQUE,**

**element\_effect VARCHAR(50) NOT NULL,**

**PRIMARY KEY (element\_id)**

**);**

**CREATE TABLE Weapon\_types (**

**wt\_id INT NOT NULL UNIQUE AUTO\_INCREMENT,**

**wt\_name VARCHAR(25) NOT NULL UNIQUE,**

**PRIMARY KEY (wt\_id)**

**);**

**CREATE TABLE Weapons (**

**weapon\_id INT NOT NULL UNIQUE AUTO\_INCREMENT,**

**weapon\_name VARCHAR(25) NOT NULL UNIQUE,**

**weapon\_type INT NOT NULL,**

**weapon\_damage INT NOT NULL,**

**PRIMARY KEY (weapon\_id),**

**FOREIGN KEY (weapon\_type)**

**REFERENCES Weapon\_types (wt\_id) ON DELETE CASCADE,**

**CHECK (weapon\_damage > 0)**

**);**

**CREATE TABLE Forge (**

**craft\_id INT NOT NULL UNIQUE AUTO\_INCREMENT,**

**craft\_item INT NOT NULL UNIQUE,**

**craft\_mora\_cost INT DEFAULT 0,**

**craft\_success\_chance INT DEFAULT 0,**

**PRIMARY KEY (craft\_id),**

**FOREIGN KEY (craft\_item)**

**REFERENCES Weapons (weapon\_id) ON DELETE CASCADE,**

**CHECK (craft\_success\_chance BETWEEN 0 AND 100)**

**);**

**CREATE TABLE Artefact\_types (**

**arty\_id INT NOT NULL UNIQUE AUTO\_INCREMENT,**

**arty\_name VARCHAR(25) NOT NULL UNIQUE,**

**PRIMARY KEY (arty\_id)**

**);**

**CREATE TABLE Artefacts (**

**art\_id INT NOT NULL UNIQUE AUTO\_INCREMENT,**

**art\_name VARCHAR(25) NOT NULL UNIQUE,**

**art\_type INT NOT NULL,**

**art\_damage INT NOT NULL,**

**PRIMARY KEY (art\_id),**

**FOREIGN KEY (art\_type)**

**REFERENCES Artefact\_types (arty\_id) ON DELETE CASCADE,**

**CHECK (art\_damage > 0)**

**);**

**CREATE TABLE Domains (**

**domain\_id INT NOT NULL UNIQUE AUTO\_INCREMENT,**

**domain\_name VARCHAR(25) NOT NULL UNIQUE,**

**domain\_item\_drop INT,**

**domain\_mora\_drop INT DEFAULT 0,**

**domain\_drop\_chance INT DEFAULT 0,**

**PRIMARY KEY (domain\_id),**

**FOREIGN KEY (domain\_item\_drop)**

**REFERENCES Artefacts (art\_id) ON DELETE CASCADE,**

**CHECK (domain\_drop\_chance BETWEEN 0 AND 100)**

**);**

**CREATE TABLE Characters (**

**char\_id INT NOT NULL UNIQUE AUTO\_INCREMENT,**

**char\_name VARCHAR(25) NOT NULL UNIQUE,**

**char\_damage INT NOT NULL,**

**char\_element INT,**

**char\_weapon\_type INT NOT NULL,**

**char\_art\_type INT NOT NULL,**

**PRIMARY KEY (char\_id),**

**FOREIGN KEY (char\_element)**

**REFERENCES Elements (element\_id) ON DELETE SET NULL,**

**FOREIGN KEY (char\_weapon\_type)**

**REFERENCES Weapon\_types (wt\_id) ON DELETE CASCADE,**

**FOREIGN KEY (char\_art\_type)**

**REFERENCES Artefact\_types (arty\_id) ON DELETE CASCADE,**

**CHECK (char\_damage > 0)**

**);**

**CREATE TABLE Players (**

**player\_id INT NOT NULL UNIQUE AUTO\_INCREMENT,**

**player\_name VARCHAR(25) UNIQUE NOT NULL,**

**player\_mora INT DEFAULT 0,**

**player\_character INT,**

**player\_weapon INT,**

**player\_artefact INT,**

**PRIMARY KEY (player\_id),**

**FOREIGN KEY (player\_character)**

**REFERENCES Characters (char\_id) ON DELETE SET NULL,**

**FOREIGN KEY (player\_weapon)**

**REFERENCES Weapons (weapon\_id) ON DELETE SET NULL,**

**FOREIGN KEY (player\_artefact)**

**REFERENCES Artefacts (art\_id) ON DELETE SET NULL,**

**CHECK (player\_mora >= 0)**

**);**

**# Create trigger**

**# Add numerical suffix to duplicated player\_name**

**DELIMITER //**

**CREATE TRIGGER change\_player\_name**

**BEFORE INSERT ON Players**

**FOR EACH ROW**

**BEGIN**

**DECLARE original\_player\_name VARCHAR(25);**

**DECLARE suffix\_counter INT;**

**SET original\_player\_name = NEW.player\_name;**

**SET suffix\_counter = 1;**

**WHILE EXISTS (SELECT TRUE FROM Players WHERE player\_name = NEW.player\_name) DO**

**SET NEW.player\_name = concat(original\_player\_name, suffix\_counter);**

**SET suffix\_counter = suffix\_counter + 1;**

**END WHILE;**

**END**

**//**

**# Insert tuples**

**INSERT INTO Elements**

**VALUES (DEFAULT,'Pyro','Applies Burning'),**

**(DEFAULT,'Hyro','Slows enemies'),**

**(DEFAULT,'Dendro','Applies flamable'),**

**(DEFAULT,'Anemo','Spreads elemental effects'),**

**(DEFAULT,'Elctro','Paralyses enemies'),**

**(DEFAULT,'Geo','Creates shields');**

**INSERT INTO Weapon\_types**

**VALUES (DEFAULT,'Claymore'),(DEFAULT,'Bow'),(DEFAULT,'Sword'),(DEFAULT,'Catalyst'),(DEFAULT,'Polearm');**

**INSERT INTO Weapons**

**VALUES**

**(DEFAULT,'Skyrider Claymore',1,30),(DEFAULT,'Sacrificial Claymore',1,40),(DEFAULT,'Claymore Prime',1,50),**

**(DEFAULT,'Skyrider Bow',2,30),(DEFAULT,'Sacrificial Bow',2,40),(DEFAULT,'Bow Prime',2,50),**

**(DEFAULT,'Skyrider Sword',3,30),(DEFAULT,'Sacrificial Sword',3,40),(DEFAULT,'Sword Prime',3,50),**

**(DEFAULT,'Skyrider Catalyst',4,30),(DEFAULT,'Sacrificial Catalyst',4,40),(DEFAULT,'Catalyst Prime',4,50),**

**(DEFAULT,'Skyrider Polearm',5,30),(DEFAULT,'Sacrificial Polearm',5,40),(DEFAULT,'Polearm Prime',5,60);**

**INSERT INTO Forge**

**VALUES**

**(DEFAULT,1,530,50),(DEFAULT,2,1200,30),(DEFAULT,3,1700,20),**

**(DEFAULT,4,510,42),(DEFAULT,5,1030,31),(DEFAULT,6,1800,25),**

**(DEFAULT,7,550,40),(DEFAULT,8,1070,37),(DEFAULT,9,1850,20),**

**(DEFAULT,10,570,52),(DEFAULT,11,1300,32),(DEFAULT,12,1500,24),**

**(DEFAULT,13,530,53),(DEFAULT,14,1020,33),(DEFAULT,15,1900,23);**

**INSERT INTO Artefact\_types**

**VALUES (DEFAULT,'Blessing'),(DEFAULT,'Charm'),(DEFAULT,'Obession');**

**INSERT INTO Artefacts**

**VALUES**

**(DEFAULT,'Hunter\'s Blessing',1,30),(DEFAULT,'Warden\'s Blessing',1,40),(DEFAULT,'Cursed Blessing',1,50),**

**(DEFAULT,'Hunter\'s Charm',2,30),(DEFAULT,'Warden\'s Charm',2,40),(DEFAULT,'Cursed Charm',2,50),**

**(DEFAULT,'Hunter\'s Obession',3,30),(DEFAULT,'Warden\'s Obession',3,40),(DEFAULT,'Cursed Obession',3,50);**

**INSERT INTO Domains**

**VALUES**

**(DEFAULT,'Dragon\'s Den',null,1000,0),**

**(DEFAULT,'Hunter\'s Altar',1,0,50),(DEFAULT,'Warden\'s Altar',2,-100,30),(DEFAULT,'Cursed Altar',3,-1000,30),**

**(DEFAULT,'Hunter\'s Treasury',4,0,40),(DEFAULT,'Warden\'s Treasury',5,-150,35),(DEFAULT,'Cursed Treasury',6,-1200,35),**

**(DEFAULT,'Hunter\'s Husk',7,0,45),(DEFAULT,'Warden\'s Husk',8,-250,35),(DEFAULT,'Cursed Husk',9,0,5);**

**INSERT INTO Characters**

**VALUES**

**(DEFAULT,'Diluc',60,1,1,2),**

**(DEFAULT,'Xinyan',40,1,1,1),**

**(DEFAULT,'Razor',40,5,1,2),**

**(DEFAULT,'Sucrose',40,4,4,3),**

**(DEFAULT,'Zhongli',60,6,5,1),**

**(DEFAULT,'Slimy',30,3,3,3),**

**(DEFAULT,'Traveller',60,null,3,1),**

**(DEFAULT,'Childe',60,2,2,3),**

**(DEFAULT,'Fischl',40,4,2,1);**

**INSERT INTO Players**

**VALUES**

**(DEFAULT,'xcalibur2',0,null,null,null),**

**(DEFAULT,'dilucmain',670,1,3,5),**

**(DEFAULT,'electrojojo',400,3,1,5),**

**(DEFAULT,'chu2ni',2820,9,5,3),**

**(DEFAULT,'bigdaddy33',0,5,15,2),**

**(DEFAULT,'slime4life',44,6,9,9),**

**(DEFAULT,'unarmedman666',6600,7,null,9);**

**# Create views**

**# Provides a quick overview to each players' state.**

**CREATE VIEW Quickview AS**

**SELECT**

**p.player\_id AS 'Player ID',**

**p.player\_name AS 'Player Name',**

**p.player\_mora AS 'Mora Remaining',**

**c.char\_name AS 'Character',**

**w.weapon\_name AS 'Equiped Weapon',**

**a.art\_name AS 'Equiped Artefact'**

**FROM Players p**

**LEFT JOIN**

**Artefacts a ON p.player\_artefact = a.art\_id**

**LEFT JOIN**

**Weapons w ON p.player\_weapon = w.weapon\_id**

**LEFT JOIN**

**Characters c ON p.player\_character = c.char\_id;**

**# Provides a recommendations for each player based on their selected character.**

**CREATE VIEW Build\_recommendations AS**

**SELECT**

**p.player\_id AS 'Player ID',**

**p.player\_name AS 'Player Name',**

**c.char\_name AS 'Character',**

**w.weapon\_name AS 'Equiped Weapon',**

**a.art\_name AS 'Equiped Artefact',**

**CASE WHEN c.char\_id is NULL THEN 0**

**ELSE COALESCE(w.weapon\_damage,0) + COALESCE(a.art\_damage,0) + COALESCE(c.char\_damage,0) END AS 'Total Damage',**

**recommended\_weap.weapon\_name 'Recommended Weapon',**

**recommended\_art.art\_name ' Recommended Artefact',**

**CASE WHEN c.char\_id is NULL THEN NULL**

**ELSE COALESCE(recommended\_weap.max\_weapon\_damage,0)+COALESCE(recommended\_art.max\_art\_damage,0)+COALESCE(c.char\_damage,0) END AS 'BiS Damage'**

**FROM**

**Players p**

**LEFT JOIN**

**Artefacts a ON p.player\_artefact = a.art\_id**

**LEFT JOIN**

**Weapons w ON p.player\_weapon = w.weapon\_id**

**LEFT JOIN**

**Characters c ON p.player\_character = c.char\_id**

**LEFT JOIN**

**(SELECT**

**w.weapon\_name,**

**best\_weapon\_by\_type.max\_weapon\_damage,**

**w.weapon\_type**

**FROM**

**Weapons w**

**JOIN (**

**SELECT**

**MAX(weapon\_damage) max\_weapon\_damage, weapon\_type**

**FROM**

**Weapons**

**GROUP BY weapon\_type**

**) best\_weapon\_by\_type**

**ON best\_weapon\_by\_type.max\_weapon\_damage = w.weapon\_damage**

**AND best\_weapon\_by\_type.weapon\_type = w.weapon\_type**

**) recommended\_weap**

**ON c.char\_weapon\_type = recommended\_weap.weapon\_type**

**LEFT JOIN**

**(SELECT**

**art.art\_name,**

**best\_art\_by\_type.max\_art\_damage,**

**art.art\_type**

**FROM**

**Artefacts art**

**JOIN (**

**SELECT**

**MAX(art\_damage) max\_art\_damage, art\_type**

**FROM**

**Artefacts**

**GROUP BY art\_type**

**) best\_art\_by\_type**

**ON best\_art\_by\_type.max\_art\_damage = art.art\_damage**

**AND best\_art\_by\_type.art\_type = art.art\_type**

**) recommended\_art**

**ON c.char\_art\_type = recommended\_art.art\_type;**

**# Create roles**

**# Dbmaster has complete control of the database.**

**CREATE ROLE Dbmaster;**

**GRANT ALL PRIVILEGES ON \* TO Dbmaster WITH GRANT OPTION;**

**# Developer has access to every tables.**

**CREATE ROLE Developer;**

**GRANT ALL PRIVILEGES ON \* TO Developer;**

**# Gamemaster has full access to both views and SELECT privileges on all other tables.**

**CREATE ROLE Gamemaster;**

**GRANT SELECT ON \* TO Gamemaster;**

**GRANT ALL PRIVILEGES ON Quickview TO Gamemaster;**

**GRANT ALL PRIVILEGES ON Build\_recommendations TO Gamemaster;**

**# Player has SELECT access to all tables.**

**CREATE ROLE Player;**

**GRANT SELECT ON \* TO Player;**