PD2 Weapons Datastore Implementation

Overview of Datastore

Background Information

Payday 2 is a first-person shooter consisting of a variety of jobs that the player can either do by themselves or in a team of up to four. The player is offered a huge range of dynamic contracts, from small convenience store hits, kidnappings and looting major bank vaults.

Purpose of Database

One of the largest components of Payday 2 is a range of weapons. The Payday 2 arsenal covers sniper rifles and assault rifles to grenade launchers. Complimentary to the number of guns is the number of weapon modifications. However, there were complications regarding the data collection of weapon modification. So weapon modifications will be substituted with weapons skins.

To understand how this database is going to be used, we must understand some of the basics regarding weapons in Payday 2. Firstly every player has four weapon slots:

1. Primary
2. Secondary
3. Melee
4. Throwable

These weapons do not have the same stats. For example, primary and secondary weapons have a stat called "threat", a higher "threat" value that will cause enemies to target this player. While throwable weapons don't have this stat, instead they have a "cooldown" stat, which is the amount of time they must wait before utilising the weapon again.

Secondly, these weapons can have their stats changed by using weapon modification as previously mentioned, but also with weapon skins.

Lastly, not all weapons, modifications and skins are obtained the same way. Some of the ways they can obtained are by spending in-game currency, completing specific heists and achievements and buying downloadable content (DLCs).

Examples of Uses

Some of the examples that a player might use want this database would be:

- To comparing different weapons and their stats.
- Obtaining information on how to obtain a specific weapon or skin.
- To see what skins are available for a specific weapon.

Database Design

Data Collection

Before starting to design or implementing anything, it was essential to know what type of data I would be dealing with (structured or unstructured) and where I was going to get the data.

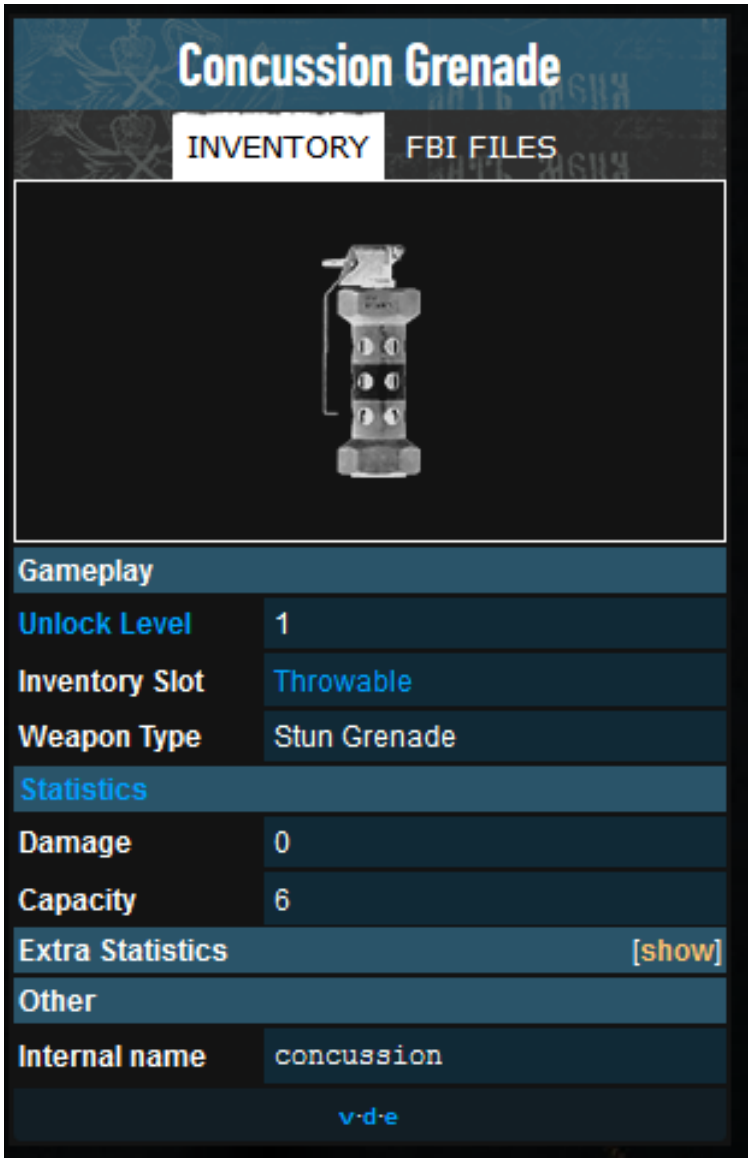
All the data gathered for the database comes from a Payday 2 fan page (https://payday.fandom.com/wiki/PAYDAY_2). The original table for weapons was formatted as you can see in the table below.

Primary Edit

Assault Rifles Edit

Weapon	rof	mun	mag	dmg	acc	stb	con	thr	rep	Source
Golden AK.762	560.7	90	30	97	60	44	11	22	17	Community
AMCAR	545.5	220	20	36	36	76	21	14	0	Base Game
AK	652.2	150	30	40	48	60	16	14	1	Base Game
CAR-4	600	150	30	38	44	60	20	14	4	Base Game
UAR	750	150	30	38	64	40	20	14	8	Base Game
AK.762	560.7	90	30	80	60	44	13	22	16	Base Game
JP36	705.9	240	30	36	40	64	19	13	16	Base Game
M308	705.9	70	10	160	84	36	8	31	26	Base Game
AK5	705.9	150	30	38	60	60	18	14	33	Base Game
AMR-16	857.1	90	30	80	56	32	17	14	39	Base Game
Commando 553	714.3	240	30	35	36	56	22	12	0	Armored Transport
Eagle Heavy	612.2	100	20	81	72	44	8	22	0	Gage Weapon Pack #01

These tables were available for primary, secondary and melee weapons but not throwable weapons. The throwable weapons stats table (t_stats) was inputted manually. The layout of these weapons is similar to the figure below.

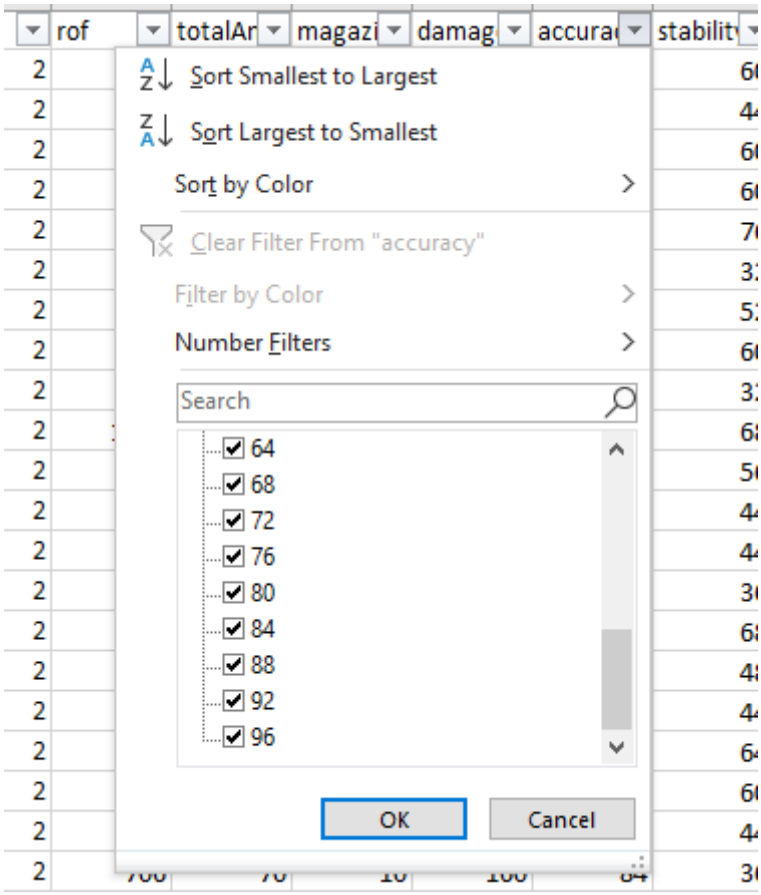


Data Cleaning and Transformation

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	weaponID	name	slotID	source	type	rof	totalAr	magazi	damag	accura	stabilit	concea	threat	reputa	skinID
2	1	AK	1	5	2	652	150	30	40	48	60	16	14	1	8
3	2	AK.762	1	5	2	561	90	30	80	60	44	13	22	16	28
4	3	AK17	1	30	2	652	105	35	99	68	60	16	14	16	77
5	4	AK5	1	5	2	706	150	30	38	60	60	18	14	33	52
6	5	AMCAR	1	5	2	546	220	20	36	36	76	21	14	0	1
7	6	AMR-16	1	5	2	857	90	30	80	56	32	17	14	39	57
8	7	Bootleg	1	52	2	667	200	100	38	24	52	20	14	21	80
9	8	CAR-4	1	5	2	600	150	30	38	44	60	20	14	4	15
10	9	Cavity 9mm	1	61	2	706	99	33	65	80	32	25	31	15	76
11	10	Clarion	1	26	2	1000	240	30	35	36	68	24	14	27	84
12	11	Commando 553	1	4	2	714	240	30	35	36	56	22	12	0	66
13	12	Eagle Heavy	1	34	2	612	100	20	81	72	44	8	22	0	69
14	13	Falcon	1	55	2	698	100	20	80	68	44	10	22	42	103
15	14	Galant	1	5	2	600	72	8	160	84	36	20	31	25	39
16	15	Gecko 7.62	1	26	2	845	150	30	42	48	68	15	24	34	96
17	16	Gewehr 3	1	26	2	652	100	20	80	68	48	12	26	52	106

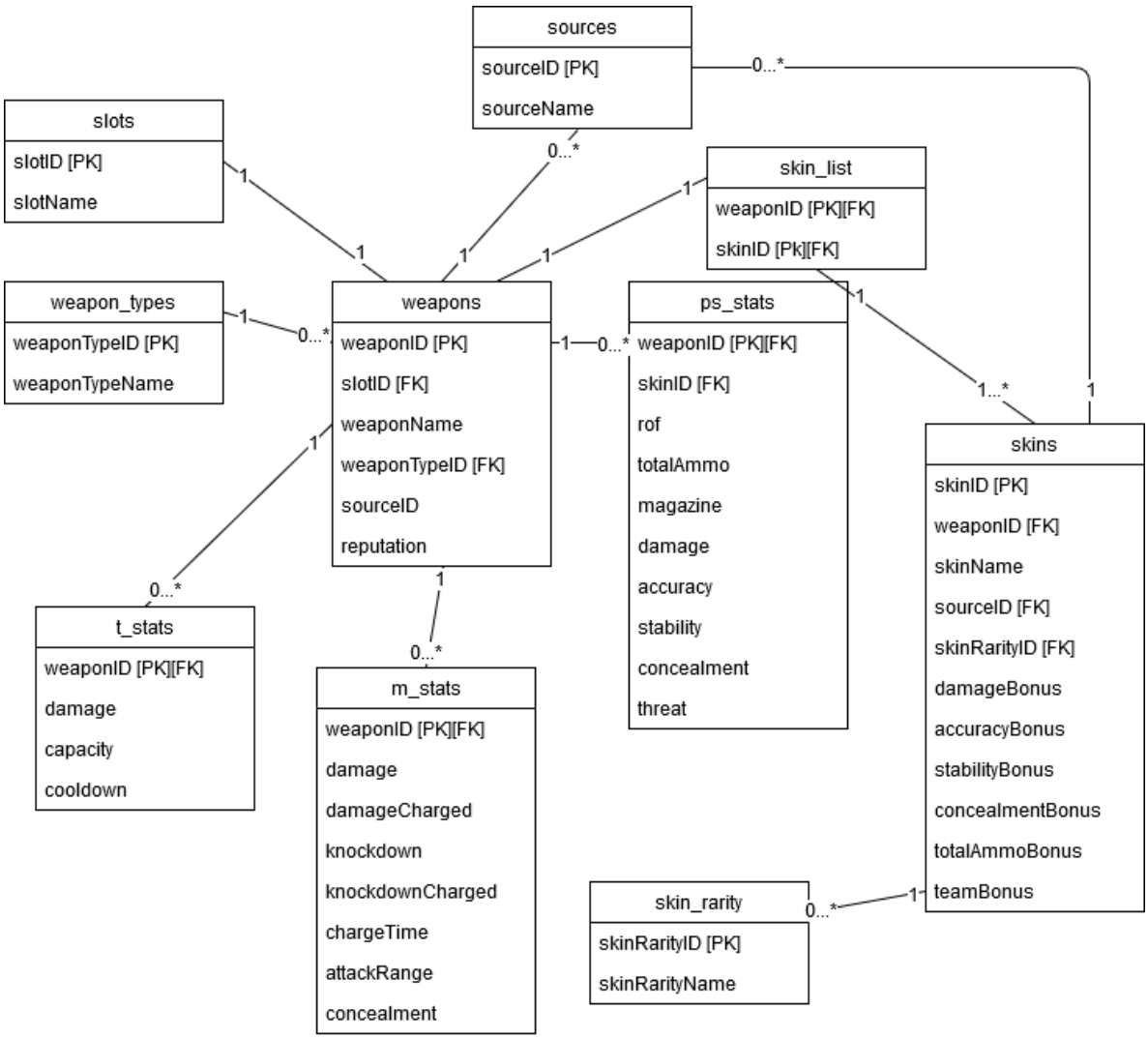
As you can see in the figure above, I introduced new variables that were not in the original table. The weaponID was made to uniquely identify the weapons. The skinID attribute was made to connect the skin list and skins. The skin list is required because a weapon can have more than one skin.

I used the FILTER function to make sure each column had the appropriate values. For example, the accuracy stat of a weapon should be an integer ranging between 0-100.



The VLOOKUP function was used to quickly identify any missing or odd values.

Initial ER Diagram



Choosing The Right database

The database management system that was selected for implementing this datastore was MySQL. This was mainly because the data itself was fully structured. Not to mention the game has been out for a while, so there wouldn't be any much growth. With very little growth meant that I didn't have to focus too much on scalability.

Choosing a NoSQL database management system would be more difficult to implement and use. NoSQL database management systems aim to minimise relational data. If my dataset had been unstructured, semi-structured or had expected exponential growth soon, then NoSQL might have been the way to go.

Attributes Data Types

```
Rerun the table creation queries and import dataset
```

Slots Table

Attribute Name	Data Type	Justification
slotID	TINYINT	There are only for slots in the game. Using the INT data type would just be a waste of storage. TINYINT will allow up to 255.
slotName	VARCHAR(20)	The length of slot names is not similar enough to use the CHAR data type.

Sources Table

Attribute Name	Data Type	Justification
sourceID	SMALLINT	Uniquely identifies each source for weapons and skins. There are currently roughly 60 possible sources. Using SMALLINT will allow up to 32,767.
sourceName	VARCHAR(40)	The length of source names is not similar enough to use the CHAR data type.

Weapon Types Table

Attribute Name	Data Type	Justification
weaponTypeID	SMALLINT	There are at least 20 weapon types across the slots.
weaponTypeName	VARCHAR(40)	The length of weapon type names is not similar enough to use the CHAR data type.

Weapon

Attribute Name	Data Type	Justification
weaponID	SMALLINT	There are over 200 weapons across the slots. The number is not expected to increase too much since it is an old game with few updates.
slotID	TINYINT	There are only for slots in the game. Using the INT data type would just be a waste of storage. TINYINT will allow up to 255.
weaponName	VARCHAR(40)	The length of weapon names is not similar enough to use the CHAR data type.
weaponTypeID	SMALLINT	There are at least 20 weapon types across the slots.
sourceID	SMALLINT	Uniquely identifies each source for weapons and skins. There are currently roughly 60 possible sources. Using SMALLINT will allow up to 32,767.
reputation	TINYINT	The level a must reach before the weapon becomes available. · The maximum level is 100.

Skin Rarity Table

Attribute Name	Data Type	Justification
skinRarityID	TINYINT	The skins are divided into 5 tiers. Even if new tiers are introduced it won't exceed 255.
skinRarityName	VARCHAR(20)	The length of skin rarity names is not similar enough to use the CHAR data type.

Skins Table

Attribute Name	Data Type	Justification
skinID	SMALLINT	Uniquely identifies each skin of primary and secondary weapons. There are currently approximately 500 skins. That number will not reach 32,767.
weaponID	SMALLINT	There are over 200 weapons across the slots. The number is not expected to increase too much since it is an old game with few updates.
skinName	VARCHAR(40)	The length of skin names is not similar enough to use the CHAR data type.
sourceID	SMALLINT	Uniquely identifies each source for weapons and skins. There are currently roughly 60 possible sources. Using SMALLINT will allow up to 32,767.
skinRarityID	TINYINT	The skins are divided into 5 tiers. Even if new tiers are introduced it won't exceed 255.
damageBonus	INT	Bonuses can be positive or negative.
accuracyBonus	INT	Bonuses can be positive or negative.
stabilityBonus	INT	Bonuses can be positive or negative.
concealmentBonus	INT	Bonuses can be positive or negative.
totalAmmoBonus	INT	Bonuses can be positive or negative.
teamBonus	DECIMAL(5,2)	This bonus was measured in percentage which has been converted to decimal in the database.

Skin List Table (weak entity)

Attribute Name	Data Type	Justification
weaponID	SMALLINT	There are over 200 weapons across the slots. The number is not expected to increase too much since it is an old game with few updates.
skinID	SMALLINT	Uniquely identifies each skin of primary and secondary weapons. There are currently approximately 500 skins. That number will not reach 32,767.

ps_stats Table

Attribute Name	Data Type	Justification
weaponID	SMALLINT	There are over 200 weapons across the slots. The number is not expected to increase too much since it is an old game with few updates.
skinID	SMALLINT	Uniquely identifies each skin of primary and secondary weapons. · There are currently approximately 500 skins. That number will not reach 32,767.
rof	INT	Some weapons have an absurd number for this stat so TINYINT and SMALLINT are unsuitable.
totalAmmo	INT	Some weapons have an absurd number for this stat so TINYINT and SMALLINT are unsuitable.
magazine	INT	Some weapons have an absurd number for this stat so TINYINT and SMALLINT are unsuitable.
damage	INT	Some weapons have an absurd number for this stat so TINYINT and SMALLINT are unsuitable.
accuracy	INT	Caps at 100.
stability	INT	Caps at 100.
concealment	INT	Caps at 100.
stability	INT	Caps at 100.
threat	INT	Caps at 100.

m_stats Table

Attribute Name	Data Type	Justification
weaponID	SMALLINT	There are over 200 weapons across the slots. The number is not expected to increase too much since it is an old game with few updates.
damage	INT	Some weapons have an absurd number for this stat so TINYINT and SMALLINT are unsuitable.
damageCharged	INT	Some weapons have an absurd number for this stat so TINYINT and SMALLINT are unsuitable.
knockdown	INT	Some weapons have an absurd number for this stat so TINYINT and SMALLINT are unsuitable.
knockdownCharged	INT	Some weapons have an absurd number for this stat so TINYINT and SMALLINT are unsuitable.
chargeTime	DECIMAL(5,2)	It is measured in seconds.
attackRange	INT	Some weapons have an absurd number for this stat so TINYINT and SMALLINT are unsuitable.
concealment	INT	Caps at 100.

t_stats Table

Attribute Name	Data Type	Justification
weaponID	SMALLINT	There are over 200 weapons across the slots. The number is not expected to increase too much since it is an old game with few updates.
damage	INT	Some weapons have an absurd number for this stat so TINYINT and SMALLINT are unsuitable.
capacity	INT	The number of times they can use that specific weapon. -1 represents weapons that have infinite uses. Infinite use items could be represented by leaving the value blank but that is ambiguous.
cooldown	INT	Most throwable weapons have 0 cooldowns. Cooldowns are measured in seconds.

Database Implementation

Constructing The Database And Tables

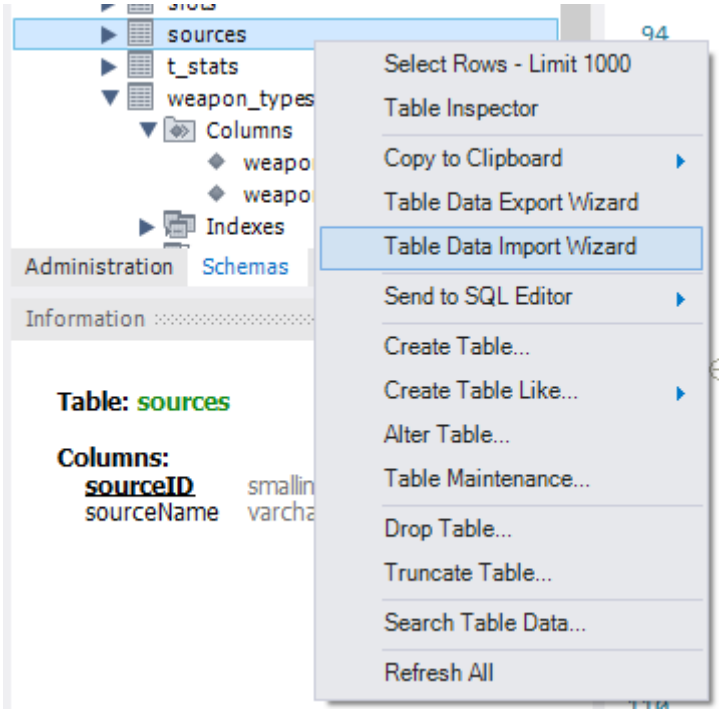
```
CREATE DATABASE pd2;
USE pd2;
```

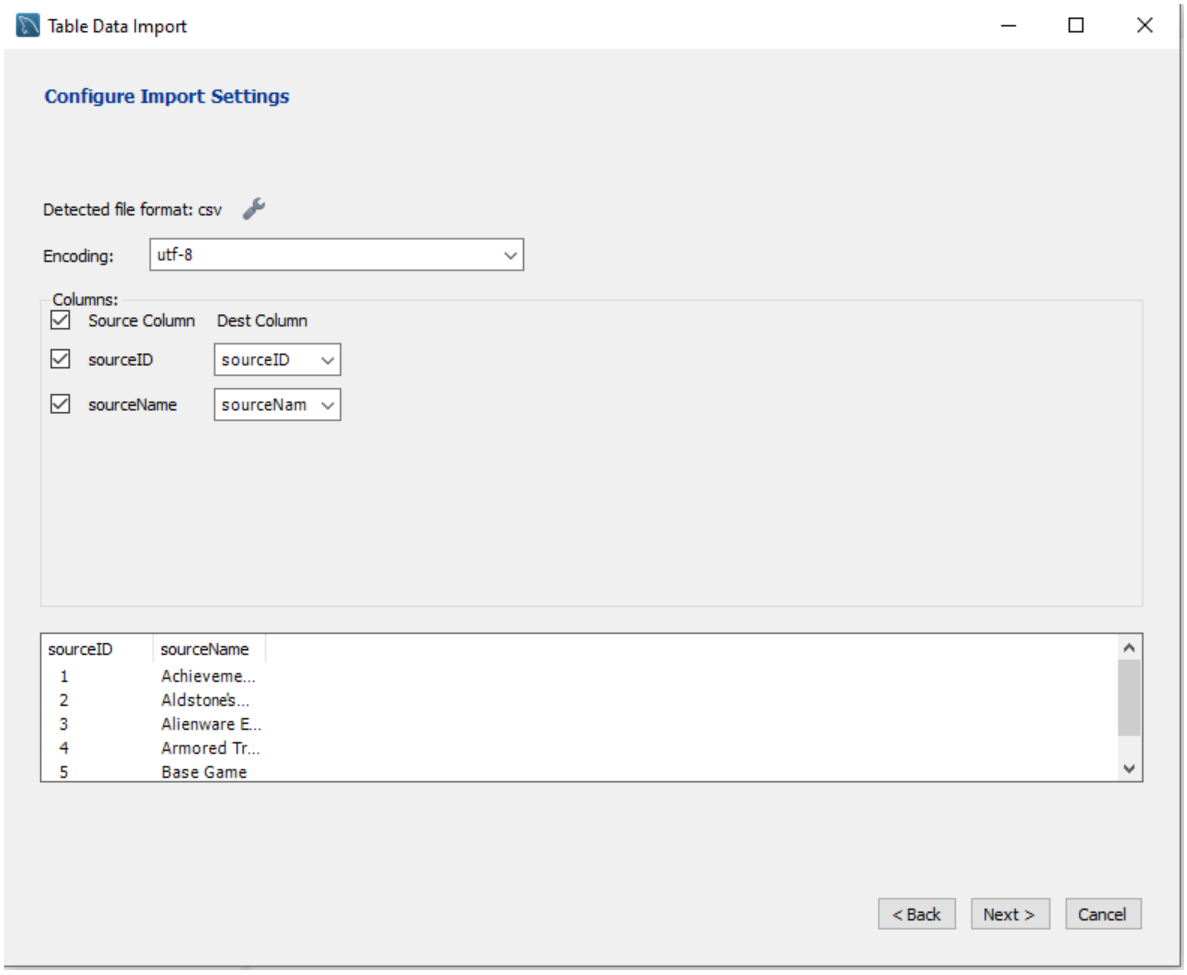
```sql
CREATE TABLE slots
(
    slotID TINYINT UNSIGNED NOT NULL,
    slotName VARCHAR(20) NOT NULL,
    PRIMARY KEY(slotID)
);

CREATE TABLE sources
(
    sourceID SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
    sourceName VARCHAR(40) NOT NULL,
    PRIMARY KEY(sourceID)
);

CREATE TABLE weapon_types
(
    weaponTypeID SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
    weaponTypeName VARCHAR(40) NOT NULL,
    PRIMARY KEY(weaponTypeID)
);

CREATE TABLE weapons
(
    weaponID SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
    slotID TINYINT UNSIGNED NOT NULL,
    weaponName VARCHAR(40) NOT NULL,
    weaponTypeID SMALLINT UNSIGNED NOT NULL,
    sourceID SMALLINT UNSIGNED NOT NULL,
    reputation TINYINT UNSIGNED NOT NULL,
    PRIMARY KEY(weaponID),
    FOREIGN KEY(slotID) REFERENCES slots(slotID),
    FOREIGN KEY(weaponTypeID) REFERENCES weapon_types(weaponTypeID),
    FOREIGN KEY(sourceID) REFERENCES sources(sourceID)
);

CREATE TABLE skin_rarity
(
    skinRarityID TINYINT UNSIGNED NOT NULL AUTO_INCREMENT,
    skinRarityName VARCHAR(20) NOT NULL,
    PRIMARY KEY(skinRarityID)
);

CREATE TABLE skins
(
    skinID SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
    skinName VARCHAR(40) NOT NULL,
    sourceID SMALLINT UNSIGNED NOT NULL,
    skinRarityID TINYINT UNSIGNED NOT NULL,
    damageBonus INT NOT NULL,
    accuracyBonus INT NOT NULL,
    stabilityBonus INT NOT NULL,
    concealmentBonus INT NOT NULL,
    totalAmmoBonus INT NOT NULL,
    teamBonus DECIMAL(5,2) NOT NULL,
    PRIMARY KEY(skinID),
    FOREIGN KEY(sourceID) REFERENCES sources(sourceID),
    FOREIGN KEY(skinRarityID) REFERENCES skin_rarity(skinRarityID)
```

```
);

CREATE TABLE skin_list
(
    weaponID SMALLINT UNSIGNED NOT NULL,
    skinID SMALLINT UNSIGNED NOT NULL,
    PRIMARY KEY(weaponID,skinID),
    FOREIGN KEY(weaponID) REFERENCES weapons(weaponID),
    FOREIGN KEY(skinID) REFERENCES skins(skinID)
);

CREATE TABLE ps_stats
(
    weaponID SMALLINT UNSIGNED NOT NULL,
    rof INT UNSIGNED NOT NULL,
    totalAmmo INT UNSIGNED NOT NULL,
    magazine INT UNSIGNED NOT NULL,
    damage INT UNSIGNED NOT NULL,
    accuracy INT UNSIGNED NOT NULL,
    stability INT UNSIGNED NOT NULL,
    concealment INT UNSIGNED NOT NULL,
    threat INT UNSIGNED NOT NULL,
    PRIMARY KEY(weaponID),
    FOREIGN KEY(weaponID) REFERENCES weapons(weaponID)
);

CREATE TABLE m_stats
(
    weaponID SMALLINT UNSIGNED NOT NULL,
    damage INT UNSIGNED NOT NULL,
    damageCharged INT UNSIGNED NOT NULL,
    knockdown INT UNSIGNED NOT NULL,
    knockdownCharged INT UNSIGNED NOT NULL,
    chargeTime DECIMAL(5,2) NOT NULL,
    attackRange INT UNSIGNED NOT NULL,
    concealment INT UNSIGNED NOT NULL,
    PRIMARY KEY(weaponID),
    FOREIGN KEY(weaponID) REFERENCES weapons(weaponID)
);

CREATE TABLE t_stats
(
    weaponID SMALLINT UNSIGNED NOT NULL,
    damage INT UNSIGNED NOT NULL,
    capacity INT NOT NULL,
    cooldown INT UNSIGNED NOT NULL,
    PRIMARY KEY(weaponID),
    FOREIGN KEY(weaponID) REFERENCES weapons(weaponID)
);

SET AUTOCOMMIT = false;
```

Importing the data into the database

Each table was imported into the database using the Table Data Import Wizard available on MySQL.





Queries

Creating a view to querying primary and secondary weapons

```
CREATE VIEW ps_details AS
SELECT weaponID AS id, slotID AS slot, sourceID AS source, weaponTypeID AS type,
weaponName as name,
rof, totalAmmo, magazine, damage, accuracy,
stability, concealment, threat
FROM weapons
NATURAL JOIN ps_stats
NATURAL JOIN slots
NATURAL JOIN sources
NATURAL JOIN weapon_types;
```

Accessing information about a primary/secondary weapon

Without restrictions:

```
SELECT name, totalAmmo, magazine, damage, accuracy, stability, concealment,
threat
FROM ps_details
ORDER by name;
```

Output:

name	totalAmmo	magazine	damage	accuracy	stability	concealment	threat
Airbow	30	6	700	84	84	5	10
AK	150	30	40	48	60	16	14
AK.762	90	30	80	60	44	13	22
AK17	105	35	99	68	60	16	14
AK5	150	30	38	60	60	18	14
Akimbo Bernetti 9	168	28	37	52	60	28	6
Akimbo Castigo .44	72	12	180	76	4	28	24
Akimbo Chimano 88	170	34	37	52	60	30	8
Akimbo Chimano Compact	160	20	37	52	60	30	9
Akimbo Chimano Custom	96	32	65	68	52	29	9
Akimbo Compact-5	240	60	36	28	76	23	6
Akimbo Contractor	90	30	66	68	60	27	9
Akimbo Crosskill	100	20	65	68	52	27	10
Akimbo Crosskill Guard	170	34	37	64	60	30	8
Akimbo Deagle	60	20	145	76	28	26	24
Akimbo Heather Submac...	160	64	42	52	44	25	9
Akimbo Interceptor 45	104	26	65	68	52	29	8
Akimbo Krinkov	120	60	80	60	44	21	22
AMCAR	220	20	36	36	76	21	14
AMR-16	90	30	80	56	32	17	14
Arbiter	15	5	700	96	96	18	37
Baby Deagle	60	12	145	68	32	29	9
Bernetti 9	154	14	37	56	60	30	6
Blaster 9mm	220	20	36	28	76	27	12
Bootleg	200	100	38	24	52	20	14
Breaker 12G	21	7	80	48	28	20	28

With restrictions:

```
SELECT name,totalAmmo,magazine,damage,accuracy,stability,concealment,threat
FROM ps_details
WHERE weaponID = 45;
```

Output:

	name	totalAmmo	magazine	damage	accuracy	stability	concealment	threat
▶	Brothers Grimm 12G	96	16	18	12	24	21	28

Updating a weapon stats

```
SELECT * FROM ps_details WHERE weaponID = 1;
```

Output:

	id	slot	source	type	name	rof	totalAmmo	magazine	damage	accuracy	stability	concealment	threat
▶	1	1	5	2	AK	652	150	30	40	48	60	16	14

```
UPDATE ps_details
SET damage = damage + 10
WHERE id = 1;
```

```
SELECT * FROM ps_details WHERE weaponID = 1;
```

Output:

	id	slot	source	type	name	rof	totalAmmo	magazine	damage	accuracy	stability	concealment	threat
▶	1	1	5	2	AK	652	150	30	50	48	60	16	14

As you can see the damage stat has been updated. Patches may affect stats, so it is important to update it on the weapons.

Altering tables

The original weapons table

```
SELECT * FROM weapons;
```

Output:

weaponID	slotID	weaponName	weaponTypeID	sourceID	reputation
1	1	AK	2	5	1
2	1	AK.762	2	5	16
3	1	AK17	2	30	16
4	1	AK5	2	5	33
5	1	AMCAR	2	5	0
6	1	AMR-16	2	5	39
7	1	Bootleg	2	52	21
8	1	CAR-4	2	5	4
9	1	Cavity 9mm	2	61	15
10	1	Clarion	2	26	27
11	1	Commando 553	2	4	0
12	1	Eagle Heavy	2	34	0
13	1	Falcon	2	55	42
14	1	Galant	2	5	25
15	1	Gecko 7.62	2	26	34
16	1	Gewehr 3	2	26	52
18	1	JP36	2	5	16
19	1	Lion's Roar	2	23	28
20	1	Little Friend ...	2	46	41
21	1	M308	2	5	26
22	1	Queen's Wrath	2	11	24
23	1	UAR	2	5	8
24	1	Union 5.56	2	5	0

Adding a new column "cost" has been added weapons table

```
ALTER TABLE weapons
ADD cost INT UNSIGNED NOT NULL;
```

Changing the cost of one of the weapons

```
UPDATE weapons
SET cost = 100
WHERE weaponID = 1;
```

```
SELECT * FROM weapons;
```

Output:

	weaponID	slotID	weaponName	weaponTypeID	sourceID	reputation	cost
▶	1	1	AK	2	5	1	100
	2	1	AK.762	2	5	16	0
	3	1	AK17	2	30	16	0
	4	1	AK5	2	5	33	0
	5	1	AMCAR	2	5	0	0
	6	1	AMR-16	2	5	39	0

Removing the "cost" column from the weapons table

```
ALTER TABLE weapons
DROP COLUMN cost;
```

```
SELECT * FROM weapons;
```

weaponID	slotID	weaponName	weaponTypeID	sourceID	reputation
1	1	AK	2	5	1
2	1	AK.762	2	5	16
3	1	AK17	2	30	16
4	1	AK5	2	5	33
5	1	AMCAR	2	5	0
6	1	AMR-16	2	5	39
7	1	Bootleg	2	52	21
8	1	CAR-4	2	5	4
9	1	Cavity 9mm	2	61	15
10	1	Clarion	2	26	27
11	1	Commando 553	2	4	0
12	1	Eagle Heavy	2	34	0
13	1	Falcon	2	55	42
14	1	Galant	2	5	25

The game developers may decide to add a new stat or remove a stat to weapons.

Querying for weapons without skins

```
SELECT weaponID, slotName, weaponName
FROM weapons w
NATURAL JOIN slots
WHERE NOT EXISTS (
    SELECT * FROM skin_list sl
    WHERE w.weaponID = sl.weaponID
    );
```

weaponID	slotName	weaponName
176	Melee	Trench Knife
177	Melee	Clover's Shillelagh
178	Melee	Arkansas Toothpick
179	Melee	Jackpot
180	Melee	Croupier's Rake
181	Melee	Switchblade
182	Melee	Buzzer
183	Melee	Okinawan Style Sai
184	Melee	Kunai Knife
185	Melee	Diving Knife
186	Melee	Great Sword
187	Melee	Weapon Butt
188	Melee	Nagant Bayonet
189	Throwables	Concussion Grenade
190	Throwables	Smoke Bomb
191	Throwables	Stoic's Hip Flask
192	Throwables	Pocket ECM
193	Throwables	HEF Grenade
194	Throwables	Ace of Spades
195	Throwables	Incendiary Grenade
196	Throwables	Matryoshka Grenade
197	Throwables	Frag Grenade
198	Throwables	Molotov Cocktail
199	Throwables	Dynamite
200	Throwables	Shuriken
201	Throwables	Javelin

Querying with aggregation functions

The following query is used find the highest damage weapon

```
SELECT * FROM ps_details
WHERE damage = (
    SELECT MAX(damage) FROM ps_details
    );
```

id	slot	source	type	name	rof	totalAmmo	magazine	damage	accuracy	stability	concealment	threat
116	Secondary	The OVERKILL Pack	Special	HRL-7	30	4	1	10000	96	96	5	37

More aggregation functions

The following query shows the number of each type of weapon available. As you can see there are a lot of axes available.

```
SELECT weaponTypeName as weapon_types, slotName as slot, COUNT(*) as count FROM
weapons
NATURAL JOIN slots
NATURAL JOIN weapon_types
GROUP BY weaponTypeID
ORDER BY count DESC;
```

	weapon_types	slot	count
▶	Axe	Melee	37
	Assault Rifle	Primary	24
	Knife	Melee	22
	Submachine Guns	Secondary	17
	Pistol	Secondary	17
	Special	Primary	16
	Shotgun	Primary	15
	Akimbo	Primary	14
	Sniper Rifle	Primary	10
	Light Machine Guns	Primary	5
	Fists	Melee	5
	Projectile	Throwables	5
	Fragmentation Gr...	Throwables	3
	Weapon	Melee	2
	Flag	Melee	1
	Smoke Grenade	Throwables	1
	Improvised Incend...	Throwables	1

Limitations

The data is quite accurate but it is not up to date with the current patch. This means that the data used in the database does not include a lot of the newer weapons and skins. It also doesn't reflect any stat changes to weapons. The other limitation is that because of the different stats of each weapon, the database was designed so run queries limited to the slot type. An example of this would be querying for the highest "damage" stat weapon. There is no way to query for the highest "damage" stat across all weapon slots. It would have to be done individually. The "damage" stat could be moved be moved over to the weapons table but it would not make much sense to have all the other stats in different tables. A way to resolve might be to recreate this database in NoSQL instead.

Conclusion

The PayDay 2 weapons database was created to enable players to quickly compare the statistics between the weapons and skins as there was no easy way to do so. Ultimately the PayDay 2 weapons database was successful but it has limitations such as not being up to date with the current patch or being able to run certain stat queries across the weapons from different weapon slots. To reduce the limitations and improve performance, exploring data migration from relational database to NoSQL database might be worth looking into.

References

PAYDAY THE WIKI 2019, *Weapons (Payday 2)*, viewed on 11 October 2019, <https://payday.fandom.com/wiki/Weapons_(Payday_2)>.

Steam 2019, PAYDAY 2, viewed 12 October 2019, <https://store.steampowered.com/app/218620/PAYDAY_2/>.