METAL GEAR SOLID
DATABASE DESIGN PROPOSAL

Caleb Rogers

THE PHANTOM COMPANY

TACTICAL ESPIONAGE OPERATIONS

METAL GEAR SOLID

Table of Contents

THE PHANTOM PAIN

METAL GEAR SOLID V

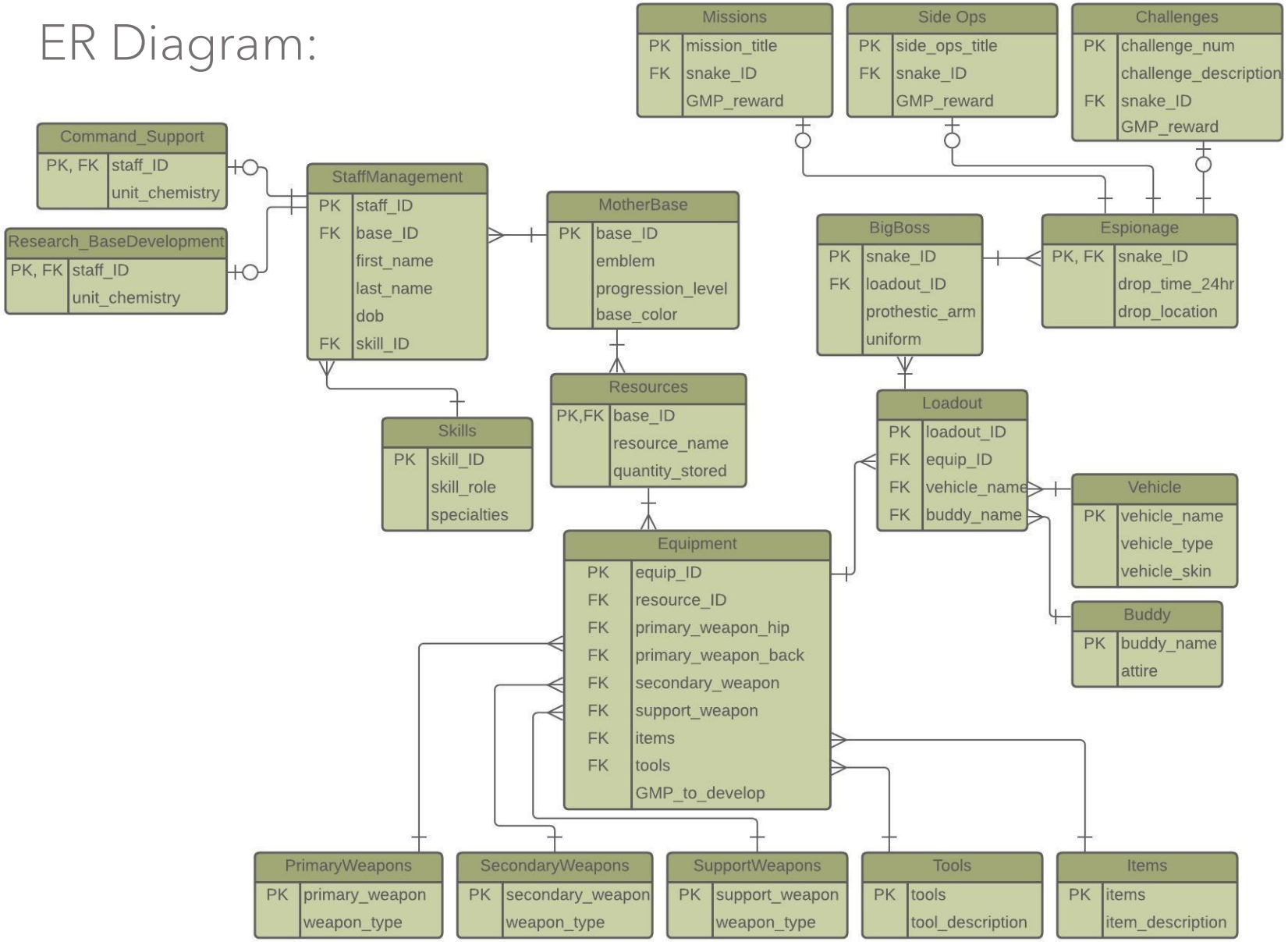
TACTICAL ESPIONAGE OPERATIONS

Executive Summary

Metal Gear Solid V was a video game that was released in 2015. It was the fifth installment of a game series that revolutionized the stealth genre and introduced video game culture to many growing up in 2000s. MGSV revolves around Venom Snake (Big Boss) who awakes from a nine-year coma after his old mother base was raided and destroyed. The fifth installment of this series has Venom Snake rebuild his Mother Base and build a private militia capable of taking down the organization that caused his demise.

This Metal Gear Solid database design proposal maps the functionality of Big Boss's Mother Base, how it has integrated a staff system, displays the development of technology that is usable by Big Boss, and the espionage tasks Big Boss undertakes. This proposal has a normalized ER diagram that visualizes the system's relationships; tables that allow data to be inserted to represent the correlating information; views, triggers, and stored procedures that analyze and query the data inserted in the database; security that grants and revokes roles, and the corresponding allowed involvement with the database; and finally, implementation notes, known problems, and future enhancements which discuss the issues that exist with the proposal, and how added time and attention could better the database result.

ER Diagram:


Missions
PK mission_title
FK snake_ID
GMP_reward
Side Ops
PK side_ops_title
FK snake_ID
GMP_reward
Challenges
PK challenge_num
challenge_description
FK snake_ID
GMP_reward
Command_Support
PK, FK staff_ID
unit_chemistry
Research_BaseDevelopment
PK, FK staff_ID
unit_chemistry
StaffManagement
PK staff_ID
FK base_ID
first_name
last_name
dob
FK skill_ID
MotherBase
PK base_ID
emblem
progression_level
base_color
BigBoss
PK snake_ID
FK loadout_ID
prothestic_arm
uniform
Espionage
PK, FK snake_ID
drop_time_24hr
drop_location
Resources
PK,FK base_ID
resource_name
quantity_stored
Skills
PK skill_ID
skill_role
specialties
Loadout
PK loadout_ID
FK equip_ID
FK vehicle_name
FK buddy_name
Vehicle
PK vehicle_name
vehicle_type
vehicle_skin
Buddy
PK buddy_name
attire
Equipment
PK equip_ID
FK resource_ID
FK primary_weapon_hip
FK primary_weapon_back
FK secondary_weapon
FK support_weapon
FK items
FK tools
GMP_to_develop
PrimaryWeapons
PK primary_weapon
weapon_type
SecondaryWeapons
PK secondary_weapon
weapon_type
SupportWeapons
PK support_weapon
weapon_type
Tools
PK tools
tool_description
Items
PK items
item_description


METAL GEAR SOLID V
THE PHANTOM PAIN
TACTICAL ESPIONAGE OPERATIONS

-- Tables --

MotherBase:

This table contains information on the different options of Mother Bases, the focused base of this database being the Dimond Dogs Mother Base. This database design stems from the relationships and functions of the Mother Base.

```
-- MotherBase
CREATE TABLE MotherBase (
    base_ID           char(3)     not null,
    emblem            text        not null,
    progression_level varchar(2) not null,
    base_color        text,
    primary key (base_ID)
);
```

	base_id [PK] character (3)	emblem text	progression_level character varying (2)	base_color text
1	B01	diamond_do...	A	yellow
2	B02	XOF	B-	purple
3	B03	rogue_coyote	C+	red

Functional Dependencies: base_ID --> emblem, progression_level, base_color

StaffManagement:

This table contains all the staff member's information and references the skills they possess. The staff in this table are placed in units that operate the Mother Base.

```
-- StaffManagement
CREATE TABLE StaffManagement (
    staff_ID        char(3) not null,
    base_ID         char(3) not null references MotherBase(base_ID),
    first_name      text,
    last_name       text    not null,
    dob             date    not null,
    skill_ID        varchar not null references Skills(skill_ID),
    primary key(staff_ID)
);
```

	staff_id [PK] character (3)	base_id character (3)	first_name text	last_name text	dob date	skill_id character varying
1	M11	B03	Alan	Labouseur	1970-01...	S3
2	M22	B01	Venom	Snake	1932-04...	S1
3	M33	B02	Huey	Emmerich	1945-10...	S4
4	M44	B01	sniper	Quiet	1951-06...	S5
5	M55	B03	Hideo	Kojima	1963-08...	S2
6	M66	B01	Benedict	Miller	1937-06...	S6
7	M77	B01	Revolver	Ocelot	1930-02...	S7
8	M88	B02	Skull	Face	1900-12...	S4

Functional Dependencies: staff_ID --> base_ID, first_name, last_name, dob, skill_ID

THE PHANTOM PAIN
METAL GEAR SOLID V
TACTICAL ESPIONAGE OPERATIONS

Skills:

This table contains the different skills available to the staff members that operate the Mother Base. There are 4 branches of skills (Sneaking/Deployment, Medical, Technical, Recon) and each skill has possible special abilities that greater improves the Mother Bases' progression and operation.

```
-- Skills
CREATE TABLE Skills (
    skill_ID    char(2)     not null,
    skill_role  text        not null,
    specialties text,
    primary key(skill_ID)
);
```

	skill_id [PK] character (2)	skill_role text	specialties text
1	S1	sneaking/dep...	rescuer
2	S2	medical	chemist
3	S3	technical	elite_engineer
4	S4	recon	arms_dealer
5	S5	sneaking/dep...	scout
6	S6	recon	surveyor
7	S7	sneaking/dep...	gambler

Functional Dependencies: skill_ID --> skill_role, specialties

THE PHANTOM PAIN

METAL GEAR SOLID V

TACTICAL ESPIONAGE OPERATIONS

Command_Support & Research_BaseDevelopment units:

These two tables show which staff make up the main units that operate the Mother Base. Staff members can be in either unit, or maybe both.

The Command_Support unit use staff members that either are higher in command that direct base functions or are specialized soldiers that deploy to battlefields.

The Research_BaseDevelopment unit use staff members to research new technology, gain intel, and develop the progress of the Mother Base.

```
-- Command&Support_Unit
CREATE TABLE Command_Support (
    staff_ID        char(3) not null references StaffManagement(staff_ID),
    unit_chemistry  int,
    primary key (staff_ID)
);

-- Research&BaseDevelopment_Unit
CREATE TABLE Research_BaseDevelopment (
    staff_ID        char(3) not null references StaffManagement(staff_ID),
    unit_chemistry  int,
    primary key (staff_ID)
);
```

Command_Support:

	staff_id [PK] character (3)	unit_chemistry integer
1	M22	99
2	M44	37
3	M66	89
4	M77	90
5	M88	95

Research_BaseDevelopment:

	staff_id [PK] character (3)	unit_chemistry integer
1	M11	86
2	M33	68
3	M55	81
4	M77	90

Functional Dependencies:
staff_ID --> unit_chemistry

Resources:

This table stores the resource, GMP, that Mother Bases uses to develop technology. Each Mother Base has its own quantity of the resource stored.

```
-- Resources
CREATE TABLE Resources (
    base_ID             char(3) not null references MotherBase(base_ID),
    resource_name       text    not null,
    quantity_stored     int     not null,
    primary key (base_ID)
);
```

	base_id [PK] character (3)	resource_name text	quantity_stored integer
1	B01	GMP	1475000
2	B02	GMP	2100000
3	B03	GMP	1015000

Functional Dependencies: base_ID --> resource_name, quantity_stored

Equipment:

This table contains all the developed equipment sets for each Mother Base. Each set developed is based off the available resources provided by the Mother Base. A different variety of primary weapons, secondary weapons, support weapons, items, and tools can be chosen from their strong entity tables that store those options. The resulting develop equipment sets are then used in Big Boss's loadout that he uses in the battlefield.

```
-- Equipment
CREATE TABLE Equipment (
    equip_ID            char(3) not null,
    resource_ID         char(3) not null references Resources(base_ID),
    primary_weapon_hip  text    not null references PrimaryWeapons(primary_weapon),
    primary_weapon_back text    not null references PrimaryWeapons(primary_weapon),
    secondary_weapon    text    not null references SecondaryWeapons(secondary_weapon),
    support_weapon      text    not null references SupportWeapons(support_weapon),
    items               text    not null references Items(items),
    tools               text    not null references Tools(tools),
    GMP_to_develop      int     not null,
    primary key (equip_ID)
);
```

Functional Dependencies:
equip_ID --> resource_ID, primary_weapon_hip, primary_weapon_back, secondary_weapon, support_weapon, items, tools, GMP_to_develop

	equip_id [PK] character (3)	resource_id character (3)	primary_weapon_hip text	primary_weapon_back text	secondary_weapon text	support_weapon text	items text	tools text	gmp_to_develop integer
1	E11	B01	AM_MRS-4	BRENNAN_LRS-46	WU_S.PISTOL	decoy	phantom_...	iDROID	750000
2	E22	B01	sinful_butterfly	S100	MACHT_37	M21_D-MINE	C.BOX	INT-SCO...	1250000
3	E33	B01	BRENNAN_LRS-46	SVG-76	WU_S333	flare_grenade	NVG	fulton_d...	900000

PrimaryWeapons:
This table contains primary weapons that are used by the weak entity, Equipment. It stores the weapon's name and type.

```
-- PrimaryWeapons
CREATE TABLE PrimaryWeapons (
    primary_weapon  text not null,
    weapon_type     text not null,
    primary key (primary_weapon)
);
```

	primary_weapon [PK] text	weapon_type text
1	AM_MRS-4	assult_rifle
2	SVG-76	assult_rifle
3	S100	shotgun
4	ISANDO_RGL-220	grenade_launcher
5	BRENNAN_LRS-46	sniper_rifle
6	sinful_butterfly	sniper_rifle
7	LPG-61	light_machine_gun

Functional Dependencies: primary_weapon --> weapon_type

SecondaryWeapons:
This table contains secondary weapons that are used by the weak entity, Equipment. It stores the weapon's name and type.

```
-- SecondaryWeapons
CREATE TABLE SecondaryWeapons (
    secondary_weapon text not null,
    weapon_type      text not null,
    primary key (secondary_weapon)
);
```

	secondary_weapon [PK] text	weapon_type text
1	WU_S.PISTOL	handgun
2	WU_S333	handgun
3	MACHT_37	submachine_gun

Functional Dependencies: secondary_weapon --> weapon_type

SupportWeapons:

This table contains support weapons that are used by the weak entity, Equipment. It stores the weapon's name and type.

```
-- SupportWeapons
CREATE TABLE SupportWeapons (
    support_weapon  text not null,
    weapon_type     text not null,
    primary key (support_weapon)
);
```

	support_weapon [PK] text	weapon_type text
1	decoy	throwable_weapon
2	hand_grenade	throwable_weapon
3	flare_grenade	throwable_weapon
4	C-4	placed_weapon
5	M21_D-MINE	placed_weapon

Functional Dependencies: support_weapon --> weapon_type

Items:
This table contains items that are used by the weak entity, Equipment. These items are used in the battlefield along with Big Boss's weapons. This table stores the items' name and a description of what it does.

```
-- Items
CREATE TABLE Items (
    items            text not null,
    item_description text,
    primary key (items)
);
```

	items [PK] text	item_description text
1	phantom_cig...	speeds perception of ti...
2	NVG	night vision goggles
3	C.BOX	cardboard box to hide

Functional Dependencies: items --> item_description

Tools:

This table contains tools that are used by the weak entity, Equipment. These tools are used in the battlefield along with Big Boss's weapons. This table stores the tool's name and a description of what it does.

```
-- Tools
CREATE TABLE Tools (
    tools            text not null,
    tool_description text,
    primary key (tools)
);
```

	tools [PK] text	tool_description text
1	INT-SCOPE	variable-magnification binoculars
2	iDROID	portable data device
3	fulton_device	battlefield extraction

Functional Dependencies: tools --> tool_description

THE PHANTOM PAIN

TACTICAL ESPIONAGE OPERATIONS

METAL GEAR SOLID V

Loadout:

This table contains all the available loadouts Big Boss can use in his deployments in the battlefields. It is composed of an equipment set that was developed by Mother Base, a vehicle that is chosen from the options of vehicles in its own strong entity table, and a buddy that too is chosen from its own strong entity table.

```
-- Loadout
CREATE TABLE Loadout (
    loadout_ID      char(3) not null,
    equip_ID        char(3) not null references Equipment(equip_ID),
    vehicle_name    text    not null references Vehicle(vehicle_name),
    buddy_name      text    not null references Buddy(buddy_name),
    primary key (loadout_ID)
);
```

	loadout_id [PK] character (3)	equip_id character (3)	vehicle_name text	buddy_name text
1	L1	E11	APE_T-41LV	D-Dog
2	L2	E22	ZHUK_RS-Z0	Quiet
3	L3	E33	UTH-66_BLACKFOOT	D-Horse
4	L4	E11	STOUT_IFV-FS	D-Walker

Functional Dependencies: loadout_ID --> equip_ID, vehicle_name, buddy_name

Vehicle:

This table contains vehicle options that Big Boss can chose from to add to a loadout. These vehicles are stored at Mother Base and transported to Big Boss's designated drop zones when deployed for combat or infiltration.

```
-- Vehicle
CREATE TABLE Vehicle (
    vehicle_name    text not null,
    vehicle_type    text not null,
    vehicle_skin    text not null,
primary key (vehicle_name)
);
```

	vehicle_name [PK] text	vehicle_type text	vehicle_skin text
1	APE_T-41LV	four_wheel_drive	standard_camo
2	BOAR-53CT	truck	desert_camo
3	ZHUK_RS-Z0	wheeled_AFV	red_variant
4	STOUT_IFV-FS	wheeled_AFV	black_variant
5	UTH-66_BLACKFOOT	helicopter	standard_camo

Functional Dependencies: vehicle_name --> vehicle_type, Vehicle_skin

THE PHANTOM PAIN

METAL GEAR SOLID V

TACTICAL ESPIONAGE OPERATIONS

Buddy:

This table contains buddy options that Big Boss can chose from to add to a loadout. These buddies are transported with Big Boss to their designated drop zones when deploying for combat or infiltration.

```
CREATE TABLE Buddy (
    buddy_name  text not null,
    attire      text,
    primary key (buddy_name)
);
```

Click to add text

	buddy_name [PK] text	attire text
1	D-Horse	battle_dr...
2	D-Dog	eyepatch
3	Quiet	naked
4	D-Walker	CCCP-W...

Functional Dependencies: buddy_name --> attire

BigBoss:

This table displays the loadout and appearance of Big Boss when deploying to the battlefield. Big Boss can use a variety of loadouts from its loadout table, use different loadouts with different prosthetic arms that adds different functionalities from its own strong entity table, and different uniforms that add different appearances that also has its own strong entity table.

```
-- BigBoss
CREATE TABLE BigBoss (
    snake_ID        char(2) not null,
    loadout_ID      char(2) not null references Loadout(loadout_ID),
    prosthetic_arm  text,
    uniform         text,
    primary key (snake_ID)
);
```

	snake_id [PK] character (2)	loadout_id character (2)	prosthetic_arm text	uniform text
1	S1	L1	bionic_arm	desert_fox
2	S2	L2	stun _arm	woodland
3	S3	L2	blast_arm	olive_drab
4	S4	L4	rocket_arm	tiger_stripe
5	S5	L3	hand_of_jehuty	wetwork

Functional Dependencies: snake_ID --> loadout_ID, prosthetic_arm, uniform

THE PHANTOM PAIN
METAL GEAR SOLID V
TACTICAL ESPIONAGE OPERATIONS

Espionage:

This table contains the different drop times and drop locations Big Boss can use when deploying to perform either missions, side ops, or challenges. It carries the primary key of snake_ID, which carries Big Boss's choices of loadout, vehicle, and buddy.

```
-- Espionage
CREATE TABLE Espionage (
    snake_ID        char(3) not null references BigBoss(snake_ID),
    drop_time_24hr  int     not null,
    drop_location   text    not null,
    primary key (snake_ID)
);
```

	snake_id [PK] character (3)	drop_time_24hr integer	drop_location text
1	S1	600	Da Shago Kallai
2	S2	1800	Da Ghwandai Khar
3	S3	1200	Qarya Askhra Ee
4	S4	0	Aabe Shifap Ruins

Functional Dependencies: snake_ID --> drop_time_24hr, drop_location

THE PHANTOM PAIN
METAL GEAR SOLID V
TACTICAL ESPIONAGE OPERATIONS

Missions, SideOps, Challenges:

These tables contains the task details of the objectives Big Boss completed when deploying for combat or infiltration. The main missions are the large events that snake did to progress the story. Side ops are tasks that Big Boss did that can earn GMP but has potential of new technology or personnel. Challenges often require tedious tasks that help complete the overall story and adds a little extra resources to the Mother Base. All three tasks result in possible GMP that is used to develop the Mother Base.

```
CREATE TABLE Missions (
    mission_title   text    not null,
    snake_ID        char(3) not null references Espionage(snake_ID),
    GMP_reward      int,
    primary key (mission_title)
);

CREATE TABLE SideOps (
    side_ops_title  text    not null,
    snake_ID        char(3) not null references Espionage(snake_ID),
    GMP_reward      int,
    primary key (side_ops_title)
);

CREATE TABLE Challenges (
    challenge_num           int     not null,
    challenge_description   text    not null,
    snake_ID                char(3) not null references Espionage(snake_ID),
    GMP_reward              int,
    primary key (challenge_num)
);
```

Missions, SideOps, Challenges (cont.)

	mission_title [PK] text	snake_id character (3)	gmp_reward integer
1	metallic_archaea	S2	720000
2	the_white_mamba	S3	650000
3	footprints_of_the_...	S2	570000
4	hellbound	S4	600000

Functional Dependencies: mission_title --> snake_ID, GMP_reward

	side_ops_title [PK] text	snake_id character (3)	gmp_reward integer
1	extract_highly_skill...	S1	300000
2	eliminate_heavy_inf...	S1	250000
3	extract_legendary_...	S4	300000
4	secure_UA-Drone_b...	S3	20000

Functional Dependencies: side_ops_title --> snake_ID, GMP_reward

	challenge_num [PK] integer	challenge_description text	snake_id character (3)	gmp_reward integer
1	1	performed_400_total_Fulton_...	S3	150000
2	2	Obtained the codename "FOX"	S2	170000
3	3	Achieved a single successful ...	S2	200000

Functional Dependencies: challenge_num --> challenge_description, snake_ID, GMP_reward

-- Views, Triggers, Stored Procedures, Security --

Views: fullBaseStaff

This query results with all the information related to Mother Base's staff members. It displays the staff members with their correlating skills and specialties and the unit's information they are a part of.

```sql
-- fullBaseStaff
CREATE OR REPLACE VIEW fullBaseStaff AS
SELECT Staff.first_name, Staff.last_name, Staff.dob, Skills.skill_role, Skills.specialties,
       Staff.skill_ID, MB.emblem as MotherBase, MB.base_ID, coalesce(CS.staff_ID, 'none') as Command_Support_Unit,
       coalesce(CS.unit_chemistry, 0) as CS_Unit_Chemistry, coalesce(RBD.staff_ID, 'none') as Research_BaseDevelopment_Unit,
       coalesce(RBD.unit_chemistry, 0) as RBD_Unit_Chemistry
FROM MotherBase MB
INNER JOIN StaffManagement Staff on MB.base_ID = Staff.base_ID
INNER JOIN Skills on Skills.skill_ID = Staff.skill_ID
LEFT OUTER JOIN Command_Support CS on CS.staff_ID = Staff.staff_ID
LEFT OUTER JOIN Research_BaseDevelopment RBD on RBD.staff_ID = Staff.staff_ID;
```

	first_name text	last_name text	dob date	skill_role text	specialties text	skill_id character varying	motherbase text	base_id character (3)	command_support_unit character	cs_unit_chemistry integer	research_basedevelopment_unit character	rbd_unit_chemistry integer
1	Venom	Snake	1932-04...	sneaking/dep...	rescuer	S1	diamond_dogs	B01	M22	99	none	0
2	sniper	Quiet	1951-06...	sneaking/dep...	scout	S5	diamond_dogs	B01	M44	37	none	0
3	Benedict	Miller	1937-06...	recon	surveyor	S6	diamond_dogs	B01	M66	89	none	0
4	Revolver	Ocelot	1930-02...	sneaking/dep...	gambler	S7	diamond_dogs	B01	M77	90	M77	90
5	Skull	Face	1900-12...	recon	arms_dealer	S4	XOF	B02	M88	95	none	0
6	Hideo	Kojima	1963-08...	medical	chemist	S2	rogue_coyote	B03	none	0	M55	81
7	Alan	Labouseur	1970-01...	technical	elite_engineer	S3	rogue_coyote	B03	none	0	M11	86
8	Huey	Emmerich	1945-10...	recon	arms_dealer	S4	XOF	B02	none	0	M33	68

THE PHANTOM PAIN
METAL GEAR SOLID V
TACTICAL ESPIONAGE OPERATIONS

Views: aestheticCustomization

This query returns all the current customizations available to Big Boss to modify his and his companion's appearances.

```
-- aestheticCustomization
CREATE OR REPLACE VIEW aestheticCustomization AS
SELECT BB.snake_ID, BB.uniform, BB.prosthetic_arm, V.vehicle_name, V.vehicle_skin, Bud.buddy_name, Bud.attire
FROM BigBoss BB
INNER JOIN Loadout LO on LO.loadout_ID = BB.loadout_ID
INNER JOIN Vehicle V on V.vehicle_name = LO.vehicle_name
INNER JOIN Buddy Bud on Bud.buddy_name = LO.buddy_name;
```

	snake_id character (2)	uniform text	prosthetic_arm text	vehicle_name text	vehicle_skin text	buddy_name text	attire text
1	S1	desert_fox	bionic_arm	APE_T-41LV	standard_camo	D-Dog	eyepatch
2	S2	woodland	stun _arm	ZHUK_RS-Z0	red_variant	Quiet	naked
3	S3	olive_drab	blast_arm	ZHUK_RS-Z0	red_variant	Quiet	naked
4	S4	tiger_stripe	rocket_arm	STOUT_IFV-FS	black_variant	D-Walker	CCCP-W...
5	S5	wetwork	hand_of_jehuty	UTH-66_BLACKFOOT	standard_camo	D-Horse	battle_dr...

Reports: calculateStaffChemistry

This query calculates the total average between both the Command_Support and Research_BaseDevelopment units. The resulting total chemistry is useful to determining the the Mother Base's progression.

```
-- calculateStaffChemistry
SELECT avg(n)::numeric(10,2)
FROM (SELECT AVG(unit_chemistry) FROM Command_Support
        UNION ALL
        SELECT AVG(unit_chemistry) FROM Research_BaseDevelopment) t(n);
```

	avg numeric (10,2)
1	81.63

THE PHANTOM PAIN
METAL GEAR SOLID V
TACTICAL ESPIONAGE OPERATIONS

Reports: missionsDuringDay

This query returns the espionage tasks that Big Boss deployed and performed during the daytime, along with the tasks corresponding information.

```
-- missionsDuringDay
SELECT Esp.snake_ID, Esp.drop_time_24hr, Miss.mission_title, Miss.GMP_reward, SO.side_ops_title, SO.GMP_reward,
       C.challenge_description, C.GMP_reward, Esp.drop_location
FROM Espionage Esp
INNER JOIN Missions Miss on Miss.snake_ID = Esp.snake_ID
INNER JOIN SideOps SO on SO.snake_ID = Esp.snake_ID
INNER JOIN Challenges C on C.snake_ID = Esp.snake_ID
where 0600 < Esp.drop_time_24hr and Esp.drop_time_24hr <= 1800;
```

	snake_id character (3)	drop_time_24hr integer	mission_title text	gmp_reward integer	side_ops_title text	gmp_reward integer	challenge_description text	gmp_reward integer	drop_location text
1	S3	1200	the_white_mamba	650000	secure_UA-Drone_b...	20000	performed_400_total_Fulton_...	150000	Qarya Askhra Ee

Stored Procedures: use_weapon

This stored procedure stores a function that accepts an input of a primary weapon name, and results with the available equipment that uses that entered weapon, whether the holstered weapon is assigned as a hip option or a back option.

```
CREATE OR REPLACE FUNCTION use_weapon(text) RETURNS
    TABLE (equip_ID char(3)) AS
$$
DECLARE
    primaryWeapon ALIAS FOR $1;
BEGIN
    RETURN QUERY
        SELECT Equip.equip_ID
        FROM Equipment Equip
        INNER JOIN PrimaryWeapons PWH on PWH.primary_weapon = Equip.primary_weapon_hip
        INNER JOIN PrimaryWeapons PWB on PWB.primary_weapon = Equip.primary_weapon_back
        WHERE
            PWH.primary_weapon = primaryWeapon OR
            PWB.primary_weapon = primaryWeapon;
END;
$$
LANGUAGE plpgsql;

SELECT *
FROM use_weapon('sinful_butterfly');
```

	equip_id character
1	E22

THE PHANTOM PAIN
METAL GEAR SOLID V
TACTICAL ESPIONAGE OPERATIONS

Stored Procedures: find_mission

This stored procedure accepts an input of the selected "snake_ID" which represents Big Boss's choices in loadout, vehicle, and buddy. The resulting output displays the missions Big Boss completed with the selected preferences.

```
CREATE OR REPLACE FUNCTION find_mission(char(2)) RETURNS
    TABLE (mission_title text) AS
$$
DECLARE
    snakeID ALIAS FOR $1;
BEGIN
    RETURN QUERY
        SELECT Miss.mission_title
        FROM Missions Miss
        INNER JOIN Espionage Esp on Esp.snake_ID = Miss.snake_ID
        INNER JOIN BigBoss BB on BB.snake_ID = Esp.snake_ID
        WHERE BB.snake_ID = snakeID;
END;
$$
LANGUAGE plpgsql;

SELECT *
FROM find_mission('S2');
```

	mission_title text
1	metallic_archaea
2	footprints_of_the_phantom

Trigger: check_primary_weapon

This trigger uses the 'check_primary_type()' stored procedure which uses a function to determine if a new inserted primary weapon meant to be holstered on Big Boss's hip or back. Trying to insert a weapon in the wrong location will result in the trigger which display's an error message that prevents the mixed insert.

```
-- check_primary_type
CREATE OR REPLACE FUNCTION check_primary_type()
    RETURNS TRIGGER AS $$
BEGIN
    IF exists (SELECT Equip.primary_weapon_hip
                FROM Equipment Equip
                where primary_weapon_hip in (select primary_weapon
                                             from PrimaryWeapons
                                             where weapon_type = 'grenade_launcher'
                                             or weapon_type = 'sniper_rifle'
                                             or weapon_type = 'light_machine_gun'))
            THEN
                RAISE EXCEPTION 'Cannot holster a large weapon on the hip. Insert the weapon so that it is holstered on the back.';
                RETURN NULL;
    ELSE
        RETURN NEW;
    END IF;
END;
$$
LANGUAGE plpgsql;

CREATE TRIGGER check_primary_type AFTER INSERT ON Equipment
FOR EACH ROW EXECUTE PROCEDURE check_primary_type();

INSERT INTO Equipment
(equip_ID, resource_ID, primary_weapon_hip, primary_weapon_back, secondary_weapon, support_weapon, items, tools, GMP_to_develop)
VALUES
('E44','B01','LPG-61','ISANDO_RGL-220','WU_S333','hand_grenade','NVG','INT-SCOPE',800000);
```

```
ERROR:  Cannot holster a large weapon on the hip. Insert the weapon so that it is holstered on the back.
CONTEXT:  PL/pgSQL function check_primary_type() line 11 at RAISE
SQL state: P0001
```

Trigger: cantKillBoss

This trigger uses the 'cantKillBoss()' stored procedure which checks if the person trying to be deleted is Big Boss. If it is, the trigger will catch the delete attempt and send an error to protect the Mother Base's leader.

```
-- cantKillBoss
CREATE OR REPLACE FUNCTION cantKillBoss()
RETURNS TRIGGER AS
$$
BEGIN
    IF OLD.first_name = 'Venom' and OLD.last_name = 'Snake'
    THEN RAISE EXCEPTION 'Cannot remove Big Boss, he is our leader!';
    END IF;
END;
$$
LANGUAGE plpgsql;

CREATE TRIGGER cantKillBoss
BEFORE DELETE ON StaffManagement
FOR EACH ROW EXECUTE PROCEDURE cantKillBoss();

DELETE FROM StaffManagement Staff
WHERE Staff.last_name = 'Snake';
```

```
ERROR:  Cannot remove Big Boss, he is our leader!
CONTEXT:  PL/pgSQL function cantkillboss() line 4 at RAISE
SQL state: P0001
```

THE PHANTOM PAIN
METAL GEAR SOLID V
TACTICAL ESPIONAGE OPERATIONS

Security

```
CREATE ROLE big_boss;
CREATE ROLE ranked_member;
CREATE ROLE staff_member;

REVOKE ALL ON ALL TABLES IN SCHEMA public from big_boss;
REVOKE ALL ON ALL TABLES IN SCHEMA public from ranked_member;
REVOKE ALL ON ALL TABLES IN SCHEMA public from staff_member;

GRANT ALL ON ALL TABLES IN SCHEMA public to big_boss;
```

Big Boss: has total control of the Mother Base. He is the commander that drives the base's operation, and the gun man that keeps it supplied and running.

```
GRANT SELECT, INSERT, UPDATE, DELETE ON StaffManagement to ranked_member;
GRANT SELECT, INSERT, UPDATE, DELETE ON Skills to ranked_member;
GRANT SELECT, INSERT, UPDATE, DELETE ON Command_Support to ranked_member;
GRANT SELECT, INSERT, UPDATE, DELETE ON Research_BaseDevelopment to ranked_member;
GRANT SELECT, UPDATE ON MotherBase to ranked_member;
GRANT SELECT, UPDATE ON Resources to ranked_member;
GRANT SELECT, UPDATE ON Equipment to ranked_member;
GRANT SELECT, INSERT, UPDATE, DELETE ON PrimaryWeapons to ranked_member;
GRANT SELECT, INSERT, UPDATE, DELETE ON SecondaryWeapons to ranked_member;
GRANT SELECT, INSERT, UPDATE, DELETE ON SupportWeapons to ranked_member;
GRANT SELECT, INSERT, UPDATE, DELETE ON Tools to ranked_member;
GRANT SELECT, INSERT, UPDATE, DELETE ON Items to ranked_member;
GRANT SELECT ON Loadout to ranked_member;
GRANT SELECT ON Vehicle to ranked_member;
GRANT SELECT ON Buddy to ranked_member;
GRANT SELECT ON BigBoss to ranked_member;
GRANT SELECT ON Espionage to ranked_member;
GRANT SELECT ON Missions to ranked_member;
GRANT SELECT ON SideOps to ranked_member;
GRANT SELECT ON Challenges to ranked_member;
```

Ranked Member: Staff members that have higher authority than the rest of the staff have control over modifying the database on the Mother Base and Staff side. This eases the workload of Big Boss so he can focus on his espionages.

THE PHANTOM PAIN
METAL GEAR SOLID V
TACTICAL ESPIONAGE OPERATIONS

Security

```
GRANT SELECT, UPDATE ON StaffManagement to staff_member;
GRANT SELECT ON Skills to staff_member;
GRANT SELECT, UPDATE ON Command_Support to staff_member;
GRANT SELECT, UPDATE ON Research_BaseDevelopment to staff_member;
GRANT SELECT ON MotherBase to staff_member;
GRANT SELECT ON Resources to staff_member;
GRANT SELECT ON Equipment to staff_member;
GRANT SELECT ON PrimaryWeapons to staff_member;
GRANT SELECT ON SecondaryWeapons to staff_member;
GRANT SELECT ON SupportWeapons to staff_member;
GRANT SELECT ON Items to staff_member;
GRANT SELECT ON Tools to staff_member;
GRANT SELECT ON Loadout to staff_member;
GRANT SELECT ON Vehicle to staff_member;
GRANT SELECT ON Buddy to staff_member;
GRANT SELECT ON BigBoss to staff_member;
```

Staff Member: Staff members make up the rest of the population on Mother Base. They can update their staff information and the data related to the units that they are in, but only maintain SELECT privileges for the rest of Mother Base's entities. Staff members do not have authoritative rights to select and view Big Boss's espionage information and the tasks involved. This protects Big Boss's deployment plans from getting in the wrong hands.

-- Implementation Notes, Known Problems, Future Enhancements --

Implementation Notes & Known Problems:

This database proposal is a limited scope to the possibilites that lay within the actual development of MGSV. Within the actual game, Venom Snake can scavenge resources in the battlefield in which takes advantage of the fulton device system. Big Boss was able to extract, not only resources, but equipment, vehicles, and personnel in which he could store or train the new perspective staff to join his Mother Base. Although an intriguing system to play with, its size was unnecessary to my database proposal.

My design summarizes the staff's units into two units. Originally, Mother Base comprises of many different base facilites, all with their own staff and responsibilites. Each platform had different units that had unique responsibilites that resulted with different outcomes. My design, for the sake of conciseness, summarized the many platforms and units to two units and their Mother Base.

Some flaws that could be addressed and improved upon would be the lack of ease for the Mother Base's unit's chemistry to influence and base's progression level. Althought not a true system to the game, that added relationship would have improved the fluidity of the diagram. Another issue that lays is the disconnection of Big Boss's espionage tasks to integrate back into resources.

THE PHANTOM PAIN
METAL GEAR SOLID
TACTICAL ESPIONAGE OPERATIONS

Future Enhancements:

With more time and attention, there would be some improvements that I would like to make that would better this design. First would be to integrate more resources other than the used GMP. Including different resource types would change the relationship of its table, as the current design only allows one resource type per Mother Base.

Another integration would be connecting the earned GMP from missions and tasks back into the Mother Base so that it could be used to create more equipment options.

Metal Gear Solid: Database Design Proposal
(2020)

Developed By:

Caleb Rogers