Pokémon™ Database

By Charles Schmitz

May 2017

Table of contents

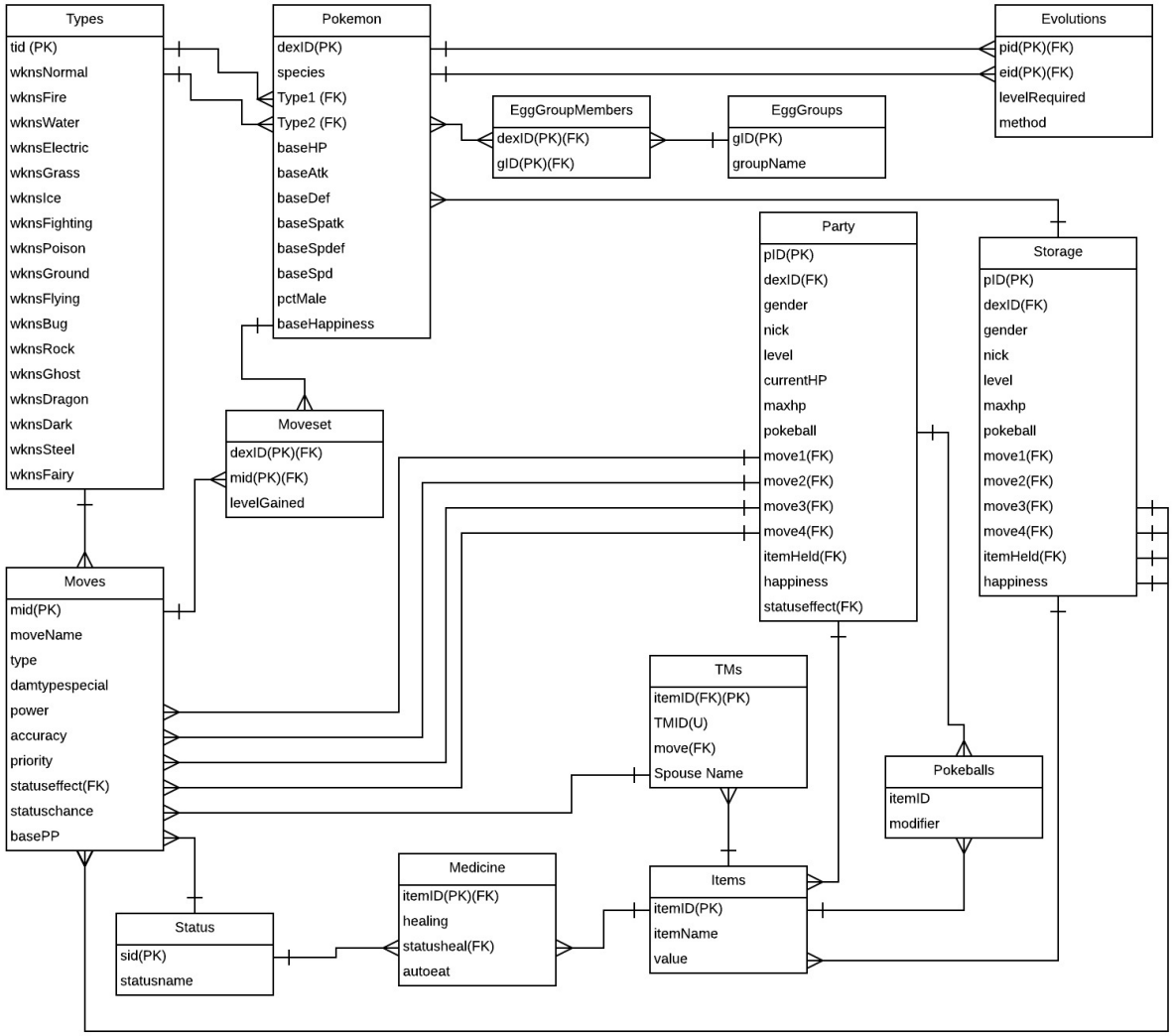
Executive summary

Pokemon is a long-standing franchise containing many creatures, moves, items, typings, and evolution cases, which have been added over the course of its 20-year development span.

Newer players may find memorizing information regarding over 800 unique Pokemon daunting, and would likely appreciate a way to quickly find relevant information regarding a new Pokemon they've just encountered for the first time, or perhaps look for a good addition to their team.

The following pages demonstrate the database's design as a whole, first with the E/R diagram, detailing all relationships, then the tables that contain the data, then the views, stored procedures, and report queries that will make working with this data meaningful. The roles shown will keep players from modifying the core information in the database. Implementation notes and possibilities for future expansion will be covered near the end.

E/R Diagram


Types
tid (PK)
wknsNormal
wknsFire
wknsWater
wknsElectric
wknsGrass
wknsIce
wknsFighting
wknsPoison
wknsGround
wknsFlying
wknsBug
wknsRock
wknsGhost
wknsDragon
wknsDark
wknsSteel
wknsFairy
Pokemon
dexID(PK)
species
Type1 (FK)
Type2 (FK)
baseHP
baseAtk
baseDef
baseSpatk
baseSpdef
baseSpd
pctMale
baseHappiness
Evolutions
pid(PK)(FK)
eid(PK)(FK)
levelRequired
method
EggGroupMembers
dexID(PK)(FK)
gID(PK)(FK)
EggGroups
gID(PK)
groupName
Party
pID(PK)
dexID(FK)
gender
nick
level
currentHP
maxhp
pokeball
move1(FK)
move2(FK)
move3(FK)
move4(FK)
itemHeld(FK)
happiness
statuseffect(FK)
Storage
pID(PK)
dexID(FK)
gender
nick
level
maxhp
pokeball
move1(FK)
move2(FK)
move3(FK)
move4(FK)
itemHeld(FK)
happiness
Moveset
dexID(PK)(FK)
mid(PK)(FK)
levelGained
Moves
mid(PK)
moveName
type
damtypespecial
power
accuracy
priority
statuseffect(FK)
statuschance
basePP
TMs
itemID(FK)(PK)
TMID(U)
move(FK)
Spouse Name
Pokeballs
itemID
modifier
Medicine
itemID(PK)(FK)
healing
statusheal(FK)
autoeat
Items
itemID(PK)
itemName
value
Status
sid(PK)
statusname

Types Table

This table maintains the various types used in the game both offensively and defensively.

```
CREATE TABLE Types (
        tid             text not null,
        wknsNormal numeric(2,1),
        wknsFire numeric(2,1),
        wknsWater numeric(2,1),
        wknsElectric numeric(2,1),
        wknsGrass numeric(2,1),
        wknsIce numeric(2,1),
        wknsFighting numeric(2,1),
        wknsPoison numeric(2,1),
        wknsGround numeric(2,1),
        wknsFlying numeric(2,1),
        wknsPsychic numeric(2,1),
        wknsBug numeric(2,1),
        wknsRock numeric(2,1),
        wknsGhost numeric(2,1),
        wknsDragon numeric(2,1),
        wknsDark numeric(2,1),
        wknsSteel numeric(2,1),
        wknsFairy numeric(2,1),
        primary key(tid)
);
```

Output pane

Data Output | Explain | Messages | History

	tid text	wknsnormal numeric(2,1)	wknsfire numeric(2,1)	wknswater numeric(2,1)	wknselectric numeric(2,1)
1	Normal	1.0	1.0	1.0	1.0
2	Fire	1.0	0.5	2.0	1.0
3	Water	1.0	0.5	0.5	2.0
4	Electric	1.0	1.0	1.0	0.5
5	Grass	1.0	2.0	0.5	0.5
6	Ice	1.0	2.0	1.0	1.0
7	Fighting	1.0	1.0	1.0	1.0
8	Poison	1.0	1.0	1.0	1.0

Dependencies: Type -> all weakness modifiers

Pokemon Table

This table holds all the specific information regarding individual Pokemon. This table is used as a reference for many other tables to connect various information.

```
CREATE TABLE Pokemon (
  dexID      numeric(4,0) not null,
  species    text,
  Type1      text not null references Types(tid),
  Type2      text references Types(tid),
  baseHP     integer not null,
  baseAtk    integer not null,
  baseDef    integer not null,
  baseSpAtk integer not null,
  baseSpDef integer not null,
  baseSpd    integer not null,
  pctMale numeric(4,3),
  baseHappiness integer not null,
  primary key(dexID)
);
```

Output pane

Data Output | Explain | Messages | History

	dexid numeric(4,0)	species text	type1 text	type2 text	basehp integer	baseatk integer
1	4	Charmander	Fire		39	52
2	5	Charmeleon	Fire		58	64
3	6	Charizard	Fire	Flying	78	84
4	27	Sandshrew	Ground		50	75
5	28	Sandslash	Ground		75	100
6	37	Vulpix	Fire		38	41
7	38	Ninetales	Fire		73	76
8	133	Eevee	Normal		55	55
9	134	Vaporeon	Water		130	65

OK.

Depedencies: dexID-> species, types, all base stats, pctMale, basehappiness

Evolutions Table

This is where we keep all the data relevent to how and what Pokemon evolve.

```
CREATE TABLE Evolutions(
  pID numeric(4,0) not null references Pokemon(dexID),
  eID numeric(4,0) not null references Pokemon(dexID),
  levelRequired integer,
  method text,
primary key(pID, eID)
);
```

Dependencies: pid + eid -> levelRequired, method

Output pane

Data Output | Explain | Messages | History

	pid numeric(4,0)	eid numeric(4,0)	levelrequired integer	method text
1	4	5	16	level up
2	5	6	36	level up
3	447	448	1	level up with maximum happiness during the day
4	653	654	16	level up

EggGroups and EggGroups Members Tables

These two tables handle the egg groups and how breeding functions in the game. EggGroupMembers references Pokemon and EggGroups tables, relating them.

```
CREATE TABLE EggGroups(
  gID integer not null,
  groupName text not null,
  primary key(gID)
);
```

```
CREATE TABLE EggGroupMembers(
  dexID numeric(4,0) not null references
Pokemon(dexID),
  gID integer not null references EggGroups(gID),
  primary key (dexID, gID)
```

Output pane

Data Output | Explain | Messa

	gid integer	groupname text
1	0	Monster
2	1	Dragon
3	2	Field
4	3	Human-Like
5	4	Flying

Output pane

Data Output | Explain | Messages

	dexid numeric(4,0)	gid integer
1	4	0
2	4	1
3	5	0
4	5	1
5	6	0
6	6	1
7	447	2
8	447	3

Status Table

This table holds all the status effects in the game used in battle, caused by moves

```
CREATE TABLE status(
  sID integer,
  statusname text,
  primary key(sID)
);
```

Output pane

Data Output | Explain | Messages

	sid integer	statusname text
1	0	Sleep
2	1	Burn
3	2	Paralysis
4	3	Freeze
5	4	Poison
6	5	Badly Poisoned
7	10	Flinch
8	11	Confused
9	12	Bind

Moves Table

This table contains all the moves that are learnable by various Pokemon. This table references the status table and the types table, which are important in combat.

```
CREATE TABLE Moves(
  mID integer not null,
  MoveName text not null,
  type text not null references Types(tid),
  damtypespecial boolean,
  power integer,
  accuracy numeric(4,3),
  priority integer,
  statuseffect integer references status(sid),
  statuschance numeric(4,3),
  basePP integer not null,
  primary key(mID)
);
```

Output pane

Data Output | Explain | Messages | History

	mid integer	movename text	type text	damtypespecial boolean	power integer
1	0	Tackle	Normal	f	50
2	1	Ember	Fire	t	40
3	2	Water Gun	Water	t	40
4	3	Vine Whip	Grass	f	45
5	4	Rock Smash	Normal	f	40
6	5	Gust	Flying	f	40
7	6	Aura Sphere	Fighting	t	80
8	7	Flash Cannon	Steel	t	80
9	8	Vacuum Wave	Fighting	t	40

This table relates pokemon to the moves they can learn, and what level they can learn it at. NULL level fields indicate egg moves or moves learnable by TM only.

```
CREATE TABLE Moveset(
  dexID numeric(4,0) not null references Pokemon(dexID),
  mID integer not null references Moves(mID),
  levelGained integer,
  primary key (dexID, mID)
);
```

Output pane

Data Output | Explain | Messages | Histo

	dexid numeric(4,0)	mid integer	levelgained integer
1	4	10	0
2	5	10	0
3	6	10	0
4	4	1	7
5	5	1	7
6	6	1	7
7	447	11	0
8	448	6	0
9	448	7	

Items Table

This table acts as a base for all the items in the game and their pokedollar values, which are split into subsets of items, including medicine for recovery, TMs which teach moves, and Pokeballs for catching pokemon.

```
CREATE TABLE Items(
  itemID integer not null,
  itemName text not null,
  value integer,
  primary key(itemID)
);
```

Output pane

Data Output | Explain | Messages | Histo

	itemid integer	itemname text	value integer
1	0	Potion	200
2	1	Super Potion	500
3	2	Oran Berry	
4	3	Pokeball	100
5	4	Great Ball	400
6	5	TM91	
7	6	TM97	
8	7	Everstone	200
9	8	Water Stone	5000

Item Subtables

```
CREATE TABLE Medicine(
  itemID integer references Items(itemID),
  healing integer,
  statusheal integer references status(sid),
  autoeat boolean not null,
  primary key(itemId)
);
```

Data Output | Explain | Messages | History

	itemid integer	healing integer	statusheal integer	autoeat boolean
1	0	20		f
2	1	50		f
3	2	10		t

```
CREATE TABLE Pokeballs(
  itemID integer references Items(itemID),
  modifier numeric(4,3),
  primary key(itemID)
);
```

Output pane

Data Output | Explain | Messag

	itemid integer	modifier numeric(4,3)
1	3	1.000
2	4	1.200

```
CREATE TABLE TMs(
  itemID integer references Items(itemID),
  TMID integer unique,
  move integer references Moves(mid),
  primary key(itemID)
);
```

Output pane

Data Output | Explain | Messages

	itemid integer	tmid integer	move integer
1	5	91	7
2	6	97	9

Inventory Table

This is where we maintain the player's inventory

```
CREATE TABLE Inventory(
  itemID integer references Items(itemID),
  qty integer,
  primary key(itemID)
);
```

Output pane

Data Output | Explain

	itemid integer	qty integer
1	3	15
2	0	10
3	2	5

Party Table

This is where we maintain the records of the player's current party pokemon and all of their statuses, moves, and HP total.

```
CREATE TABLE Party(
  pid integer not null,
  dexID integer not null references Pokemon(dexID),
  gender text,
  nick text,
  level integer not null,
  currentHP integer not null,
  maxHP integer not null,
  pokeball integer references Pokeballs(itemID),
  move1 integer references Moves(mID),
  move1PP integer,
  move2 integer references Moves(mID),
  move2PP integer,
  move3 integer references Moves(mID),
  move3PP integer,
  move4 integer references Moves(mID),
  move4PP integer,
  itemHeld integer references Items(itemID),
  happiness integer not null,
  statuseffect integer references status(sID),
  primary key(pid)
```

Output pane

Data Output | Explain | Messages | History

	pid integer	dexid integer	gender text	nick text	level integer	currenthp integer	maxhp integer	pokeball integer	move1 integer	move1pp integer	move integ
1	0	448	Female	Mya	100	100	100	3	6	20	
2	1	654	Female	Selkie	100	100	100	3	1	25	

Storage Table

Similar to the party table, this is where the player's Pokemon are stored when they are not in the party. Less information is needed here.

```
CREATE TABLE STORAGE(
  bid integer not null,
  dexID integer not null references Pokemon(dexID),
  gender text,
  nick text,
  level integer not null,
  currentHP integer not null,
  maxHP integer not null,
  pokeball integer references Pokeballs(itemID),
  move1 integer references Moves(mID),
  move2 integer references Moves(mID),
  move3 integer references Moves(mID),
  move4 integer references Moves(mID),
  itemHeld integer references Items(itemID),
  happiness integer not null,
  primary key(bid)
);
```

View: Pokedex

This view shows the player all relevent type weaknesses and resistences per Pokemon in conjunction with the rest of the stats and information.

```
CREATE OR REPLACE VIEW Pokedex AS
SELECT pokemon.dexid, species, type1, type2, basehp, baseatk, basedef, basespdef, basespd, pctmale, basehappiness,
        SUM(t1.wknsnormal*coalesce(t2.wknsnormal, 1)) as Normal_Dmg,
        SUM(t1.wknsfire*coalesce(t2.wknsnormal, 1)) as Fire_Dmg,
        SUM(t1.wknswater*coalesce(t2.wknswater, 1)) as Water_Dmg,
        SUM(t1.wknselectric*coalesce(t2.wknselectric, 1)) as Electric_Dmg,
        SUM(t1.wknsgrass*coalesce(t2.wknsgrass, 1)) as Grass_Dmg,
        SUM(t1.wknsice*coalesce(t2.wknsice, 1)) as Ice_Dmg,
        SUM(t1.wknsfighting*coalesce(t2.wknsfighting, 1)) as Fighting_Dmg,
        SUM(t1.wknspoison*coalesce(t2.wknspoison, 1)) as Poison_Dmg,
        SUM(t1.wknsground*coalesce(t2.wknsground, 1)) as Ground_Dmg,
        SUM(t1.wknsflying*coalesce(t2.wknsflying, 1)) as Flying_Dmg,
        SUM(t1.wknspsychic*coalesce(t2.wknspsychic, 1)) as Psychic_Dmg,
        SUM(t1.wknsbug*coalesce(t2.wknsbug, 1)) as Bug_Dmg,
        SUM(t1.wknsrock*coalesce(t2.wknsrock, 1)) as Rock_Dmg,
        SUM(t1.wknsghost*coalesce(t2.wknsghost, 1)) as Ghost_Dmg,
        SUM(t1.wknsdragon*coalesce(t2.wknsdragon, 1)) as Dragon_Dmg,
        SUM(t1.wknsdark*coalesce(t2.wknsdark, 1)) as Dark_Dmg,
        SUM(t1.wknssteel*coalesce(t2.wknssteel, 1)) as Steel_Dmg,
        SUM(t1.wknsfairy*coalesce(t2.wknsfairy, 1)) as Fairy_Dmg
        FROM pokemon
        JOIN Types as t1 on Pokemon.type1 = T1.tid LEFT OUTER JOIN Types as T2 ON Pokemon.type2 = T2.tid
        group by pokemon.dexID
        order by pokemon.dexID;
```

View: Pokedex (continued)

Output pane

Data Output | Explain | Messages | History

	dexid numeric(4,0)	species text	type1 text	type2 text	basehp integer	baseatk integer	basedef integer	basespdef integer	basespd integer	pctmale numeric(4,3)	basehappiness integer	normal_dmg numeric	fire_dmg numeric	water_dmg numeric
1	4	Charmander	Fire		39	52	43	50	65	0.500	70	1.0	0.5	2.0
2	5	Charmeleon	Fire		58	64	58	65	80	0.500	70	1.0	0.5	2.0
3	6	Charizard	Fire	Flying	78	84	78	85	100	0.500	70	1.00	0.50	2.00
4	27	Sandshrew	Ground		50	75	85	30	40	0.500	70	1.0	1.0	2.0
5	28	Sandslash	Ground		75	100	110	55	65	0.500	70	1.0	1.0	2.0
6	37	Vulpix	Fire		38	41	40	65	65	0.250	70	1.0	0.5	2.0
7	38	Ninetales	Fire		73	76	75	100	100	0.250	70	1.0	0.5	2.0
8	133	Eevee	Normal		55	55	50	65	55	0.875	70	1.0	1.0	1.0

View: Evolve

Shows all evolution trends by species name, level, requirement, and resulting Pokemon species name

```
CREATE OR REPLACE VIEW Evolve AS
SELECT pokemon.species, evolutions.levelrequired,
    evolutions.method, x.species as evolves_to FROM Pokemon
    join evolutions on pokemon.dexid = evolutions.pid
    join pokemon as x on evolutions.eid = x.dexid;
```

Data Output | Explain | Messages | History

	species text	levelrequired integer	method text	evolves_to text
1	Charmander	16	level up	Charmeleon
2	Charmeleon	36	level up	Charizard
3	Riolu	1	level up with maximum happiness during the day	Lucario
4	Fennekin	16	level up	Braixen
5	Braixen	36	level up	Delphox
6	Sandshrew	22	level up	Sandslash
7	Eevee	1	give a water stone	Vaporeon
8	Eevee	1	give a thunder stone	Jolteon
9	Eevee	1	give a fire stone	Flareon

Stored procedure: potentialMates

If the player wishes to breed Pokemon to make stronger ones for competitive battling, they can use this function, with a species name specified, to return all pokemon which share an egg group with it.

```
CREATE OR REPLACE FUNCTION potentialMates(pokename TEXT)
RETURNS TABLE(species2 TEXT) AS
$$
DECLARE
	pokename TEXT := $1;
BEGIN
return query
SELECT DISTINCT z.species
		FROM pokemon as y
		join EggGroupMembers as a on y.dexid = a.dexid
		JOIN egggroupmembers as b on a.gid = b.gid
		join pokemon as z on b.dexid = z.dexid
		WHERE y.species = pokename;
return;
END;
$$ LANGUAGE plpgsql;
```

Data Output | Explain | Messa

	potentialmates text
1	Vaporeon
2	Fennekin
3	Jolteon
4	Sandslash
5	Braixen
6	Lucario
7	Vulpix
8	Riolu
9	Flareon

Stored procedure: transfer

If the player wishes to keep more than SIX Pokemon, it is canon in the Pokemon games that the player cannot have a party of size greater than 6. So, in order to keep their extra Pokemon, they need a storage box to contain those extras, as well as a way to deposit those pokemon.

```
CREATE OR REPLACE FUNCTION transfer(partynum integer)
RETURNS void AS
$$
DECLARE
	partynum integer :=$1;
	a integer ;
	b integer ;
	c text;
	d text;
	e integer;
	f integer;
	g integer;
	h integer;
	i integer;
	j integer;
	k integer;
	l integer;
	m integer;
BEGIN
	SELECT pid, dexID, gender, nick, level, maxHP, pokeball, move1, move2, move3, move4, itemHeld, happiness
	INTO a, b, c, d, e, f, g, h, i, j, k, l, m FROM Party
	WHERE pid = partynum;

	insert into storage (dexID, gender, nick, level, maxHP, pokeball, move1, move2, move3, move4, itemHeld, happiness)
	values (b, c, d, e, f, g, h, i, j, k, l, m);

	DELETE FROM PARTY WHERE pid = partynum;

END;
$$ LANGUAGE plpgsql;
```

Trigger: oversizedParty

In reference to the previously mentioned transfer, it is canon if the player captures another Pokemon with a full party , the new Pokemon is automatically transferred to storage to enforce the 6 party rule, but allow for the convinience of keeping a full party with the player and not having to worry about missing a capture of oportunity.

```
CREATE OR REPLACE FUNCTION oversizedParty() RETURNS TRIGGER AS
$$
DECLARE
	a integer;
	c integer;
BEGIN
	SELECT COUNT(*) INTO c FROM Party;
	IF c > 6 THEN
	SELECT MAX(Party.pid) INTO a FROM Party;
	PERFORM transfer(a);
	END IF;
	RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER osParty
	AFTER INSERT ON PARTY
	FOR EACH STATEMENT
	EXECUTE PROCEDURE oversizedParty();
```

Reports:

We can use the following query to generate a full move list for a given pokemon if we use a “where” clause:

select pokemon.species, moveset.levelgained, moves.movename from pokemon join moveset on pokemon.dexid = moveset.dexid join moves on moveset.mid = moves.mid;

“WHERE pokemon.species = ‘Lucario’; yields the following output

Data Output | Explain | Messages | History

	species text	levelgained integer	movename text
1	Lucario	0	Aura Sphere
2	Lucario		Flash Cannon
3	Lucario		Vacuum Wave
4	Lucario		Dark Pulse

Reports: All pokemon by egg group

This query returns the egg group members table with names rather than ID numbers

```
select pokemon.species, egggroups.groupname
    from pokemon join egggroupmembers
    on pokemon.dexid = egggroupmembers.dexid
    join egggroups on egggroupmembers.gid = egggroups.gid;
```

Data Output | Explain | Messages

	species text	groupname text
1	Charmander	Monster
2	Charmander	Dragon
3	Charmeleon	Monster
4	Charmeleon	Dragon
5	Charizard	Monster
6	Charizard	Dragon
7	Riolu	Field
8	Riolu	Human-Like
9	Lucario	Field

Roles

The admin role is used to add new content to the game

CREATE ROLE admin;

grant all on all tables in schema public to admin;

Roles

The player role is needed for the player to manage their inventory, team, and reserve Pokemon without giving them access to modifying the core game

```
CREATE ROLE player;
REVOKE ALL ON ALL TABLES IN SCHEMA public FROM player;
GRANT SELECT ON ALL TABLES IN SCHEMA public TO player;
GRANT INSERT ON Inventory, Party, Storage TO player;
GRANT UPDATE ON Inventory, Party, Storage TO player;
```

Implementation Notes

- The gender ratio in the pokemon table refers to how often the pokemon appears male, and if it is not male, it is female. NULL indicates the pokemon is genderless.
- The status effect table doesn’t contain information such as the hidden effects inside burn, poison, badly poisoned, paralysis, sleep, etc, as well as normal stat-reducing and boosting abilities.

Known Issues

- The pokedex view doesn’t show egg groups inside of it
- Removing pokemon from storage is currently unimplemented, making the storage box essentially a “black hole”

Future enhancements

- Pokemon abilities
- Withdrawing pokemon from storage
- Simulating Battling/Capturing Pokemon
- Leveling up pokemon