Dragon Hunter DB

By

Christina “D”. Craig

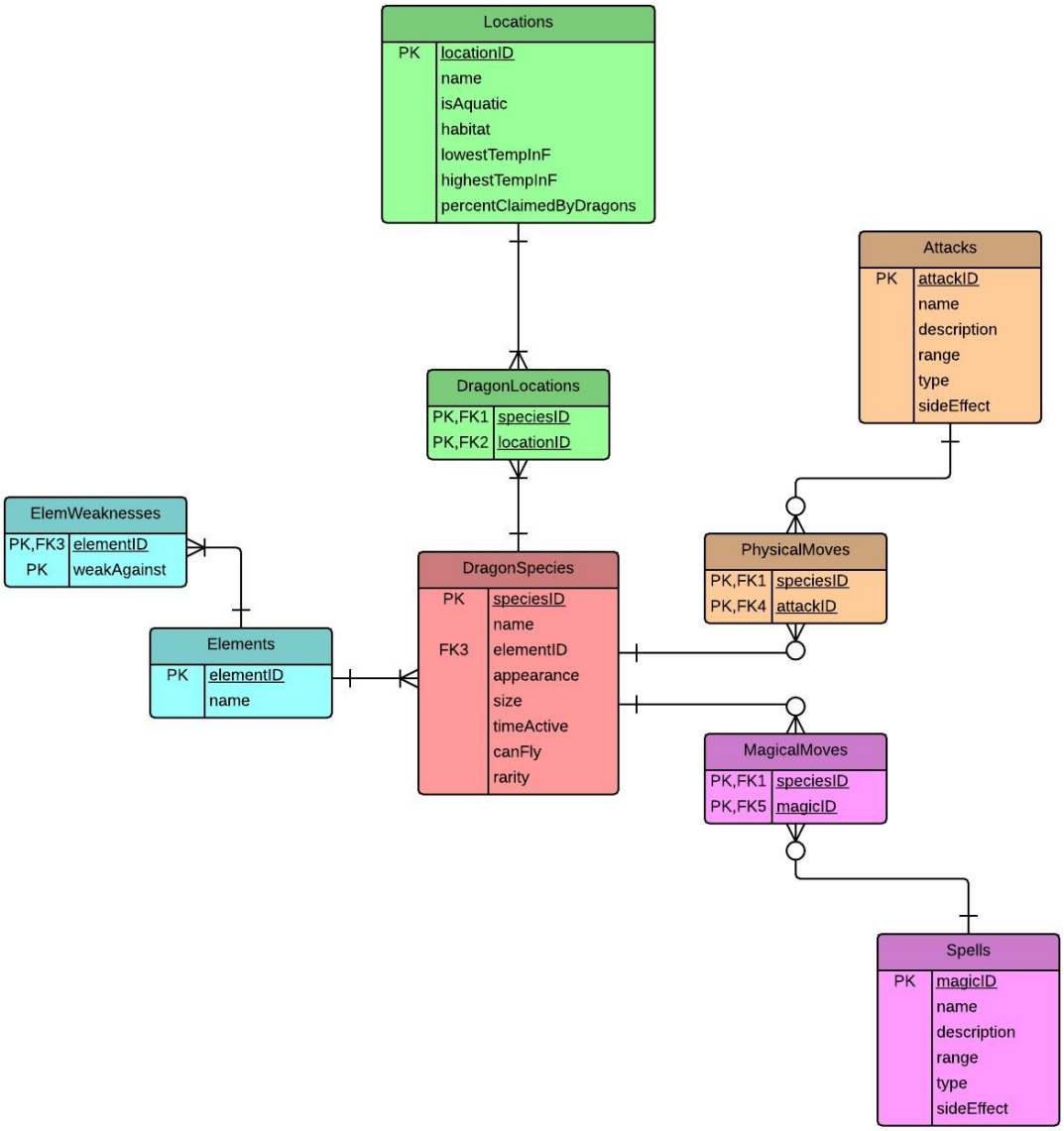
Executive Summary

Attention all adventurers! The Great Wizard Magnus has devised a new spell that will allow any common adventurer to view an extensive collection of information on our most cunning and deadly enemy—the Dragon! Magnus knows that not all of you adventurers are skilled in the magical arts, so he has graciously created several hundred magical stone tablets and scattered them around our land in both cities and wildlands for your convenience! Simply approach a stone tablet and write upon it with the tip of your finger a "query." The tablet will then shift and show you the answer you desire! Ask it anything about dragons, and it will know the answer!

Magnus has called this new magical breakthrough the "Dragon Hunter Datatbase." The objectives of Dragon Hunter DB are to provide adventurers with the information they need to travel safely and successfully defeat a dragon should they come across one. The database contains information on all of the dragon species within our country's borders. Anything from the dragon's location, attacks, and spells to its elemental weaknesses is available for viewing in the database.

Magnus has been very generous in releasing the secrets of creating this amazing magical database! Below are all of the magical enchantments and incantations that went into creating Dragon Hunter DB. You all know that Magnus is a huge supporter of open-source magic, so those of you who are skilled with magic can go ahead and improve the database even further! Magnus has even admitted the shortcomings of the database and proposed future enhancements that may increase its effectiveness in the future. Without further ado, please enjoy the magnificent Dragon Hunter DB!

Magical ER Diagram



Tables

DragonSpecies

This table lists and describes all of the dragon species in our country's borders. Included are the size, time of day that the beast is most active, ability to fly, and rarity of the dragon.

First, we need to create the enumerated types that size and rarity use:

```
CREATE TYPE rarity AS enum ('very common', 'common', 'uncommon', 'rare', 'very
    rare', 'extremely rare', 'one of a kind');

CREATE TYPE size AS enum ('extremely small','very small', 'small', 'medium', 'large',
    'very large', 'extremely large'); Now for the actual table:

CREATE TABLE DragonSpecies (
        speciesID       integer         NOT NULL PRIMARY KEY,
        speciesName text                NOT NULL UNIQUE,
        elementID       integer         NOT NULL REFERENCES Elements(elementID),
        appearance      text            NOT NULL,
        size            size            NOT NULL,
        timeActive      text            NOT NULL,
        canFly          boolean         NOT NULL,
        rarity          rarity          NOT NULL
);
```

Functional dependencies: speciesID → speciesName, elementID, appearance, size, timeActive, canFly, rarity

speciesid integer	speciesname text	elem integ	appearance text	size size	timeactive text	canf bool	rarity rarity
1	Black Drake	4	Primarily	medium	Night	t	extremely rare
2	Southern Sea Terror	2	Primarily	extremely large	Day	f	rare
3	Sun Serpent	5	Entirely a	medium	Day	t	common
4	Eastern Lava-Eater	1	Primarily	very large	Day	t	common
5	Earthshatterer	6	Primarily	extremely large	Day	f	uncommon
6	Wind Wyrm	7	Primarily	small	Night	t	common
7	Bolt Lizard	3	Bright yel	very small	Day	f	very rare
8	Red Drake	1	Primarily	medium	Day	t	common
9	Altreus	5	The dragon	large	Day	t	one of a kind

Locations

This table lists the locations in which our dragon enemies live. The table includes the name of the location, whether it is aquatic or terrestrial, the habitat type, the lowest and highest possible temperatures in degrees Fahrenheit, and the percentage of the land actively occupied and claimed by dragons.

```
CREATE TABLE Locations (
    locationID                integer     NOT NULL PRIMARY KEY,
    name                      text        NOT NULL,
    isAquatic                 boolean     NOT NULL,
    habitat                   text        NOT NULL,
    lowestTempInF             integer     NOT NULL,
    highestTempInF            integer     NOT NULL,
    percentClaimedByDragons   integer     NOT NULL
);
```

Functional dependencies:

locationID → name, isAquatic, habitat, lowestTempInF, highestTempInF, percentClaimedByDragons

locationid integer	name text	isaquati boolean	habitat text	lowesttempinf integer	highesttempinf integer	percentclaimedbydragons integer
1	Everdark Forest	f	Conife	-40	50	85
2	Dead Gulf	t	Saltwa	-45	25	100
3	Mulgore Grasslands	f	Grassl	40	100	50
4	Mount Doom	f	Volcan	200	1000	40
5	Eastern Sea	t	Saltwa	30	85	60
6	Giri Desert	f	Desert	75	150	20

DragonLocations

This table connects the DragonSpecies and Locations tables because a single dragon species can live in multiple locations.

```
CREATE TABLE DragonLocations (
    speciesID     integer  NOT NULL REFERENCES DragonSpecies(speciesID),
    locationID    integer  NOT NULL REFERENCES Locations(locationID),
    PRIMARY KEY(speciesID,locationID)
);
```

Functional dependencies: speciesID,

locationID → no dependencies

speciesid integer	locationid integer
1	1
2	5
2	2
3	3
3	6
4	4
5	3
5	6
6	3
7	4
7	6
8	4
9	2
9	5
9	4

Elements

This table lists the names of all of the possible elements that a dragon can be. Each dragon can only have a single element, but there can be multiple species of dragons that share an element. The number of fire dragons is far too high!

```
CREATE TABLE Elements (
    elementID    integer NOT NULL PRIMARY KEY,
    name         text    NOT NULL
);
```

Functional dependencies: elementID

→ name

elementid integer	name text
1	fire
2	water
3	thunder
4	darkness
5	light
6	earth
7	air

ElemWeaknesses

Every element has one or more weaknesses! How does one put out a fire? Well, with water of course! This table lists elements by ID and their weaknesses by name. Don't ask why that's so inconsistent, you might anger the great Magnus, and we really don't need another month long drought around here.

```
CREATE TABLE ElemWeaknesses (
    elementID       integer NOT NULL REFERENCES Elements(elementID),
    weakAgainst     text    NOT NULL,
    PRIMARY KEY(elementID,weakAgainst)
);
```

Functional dependencies:

elementID,weakAgainst → no dependencies

elementid integer	weakagainst text
1	water
1	earth
1	light
2	thunder
3	earth
4	light
5	air
6	air
6	water
7	thunder
7	darkness
2	darkness

Attacks

This table lists and describes all of the different types of attacks that dragons have tried (and failed) to use against the great Magnus. He has carefully observed these attacks first hand and has compiled information including the name, description, range, type, and side effect of each attack.

```
CREATE TABLE Attacks (
        attackID        integer         NOT NULL PRIMARY KEY,
        name            text            NOT NULL,
        description     text            NOT NULL,
        range           text            NOT NULL,
        type            text            NOT NULL,
        sideEffect      text
);
```

Functional dependencies: attackID → name,

description, range, type, sideEffect

attack integer	name text	description text	range text	type text	sideeffect text
1	Bite	The dragon uses its jaws to	melee	offensive	Bleeding. Lost limbs.
2	Claw	The dragon uses its claws t	melee	offensive	Bleeding.
3	Horn Thrust	The dragon uses the horns o	melee	offensive	Bleeding. Dismemberment.
4	Tail Swipe	The dragon uses its tail to	medium	both	
5	Wing Shield	The dragon raises its wings	melee	defensive	

PhysicalMoves

This table connects the DragonSpecies and Attacks tables because each dragon can employ multiple physical attacks! Some dragons like to use more parts of their bodies for combat than others, so don't be surprised if a dragon with claws just won't use the Claw attack!

```
CREATE TABLE PhysicalMoves (
        speciesID       integer NOT NULL REFERENCES DragonSpecies(speciesID),
        attackID        integer NOT NULL REFERENCES Attacks(attackID),
        PRIMARY KEY(speciesID,attackID)
    );
```

Functional dependencies: speciesID,

attackID → no dependencies

speciesid integer	attack intege
1	1
1	3
2	1
2	4
3	1
3	3
3	4
4	1
4	2
4	3
4	5
5	1
5	2
5	3
5	4
6	1
6	4
7	1
7	2

Spells

This table lists and describes all magical spells that a dragon may use to sizzle, burn, or otherwise maim the unprepared adventurer. Never fear, Magnus has taken it upon himself to carefully observe each spell down to the most minuscule detail in order to provide a detailed report!

```
CREATE TABLE Spells (
        magicID             integer NOT NULL PRIMARY KEY,
        name        text    NOT NULL,
        description text    NOT NULL,
        range       text    NOT NULL,
        type        text    NOT NULL,
        sideEffect  text
);
```

Functional dependencies: magicID → name,

description, range, type, sideEffect

magicid integer	name text	description text	range text	type text	sideeffect text
1	Fire Breath	The dragon breathes de	long	offensive	May cause severe burns.
2	Thunder Breath	The dragon breathes de	medium	offensive	May cause paralysis.
3	Ice Breath	The dragon inhales and	long	offensive	May cause freezing or pneumonia.
4	Shadow Breath	The dragon inhales and	short	offensive	May cause insanity or illusions.
5	Earthquake	The dragon causes the	long	offensive	May cause the earth to swallow y
6	Air Breath	The dragon inhales and	long	offensive	May cause a tornado to form.
7	Elemental Shield	The dragon forms a sh	short	defensive	Touching the shield may cause an
8	Elemental Storm	The dragon causes its	long	offensive	Side effects depend on element.
9	Shapeshift	The dragon can assume	melee	defensive	May cause not knowing who the he
10	Stealth	Allows the dragon to	melee	defensive	Attacks from

MagicalMoves

Similar to the PhysicalMoves table, this table seeks to connect the DragonSpecies and Spells tables. Each dragon can use multiple spells, and it isn't completely uncommon for a dragon of a certain element to use a spell from another element, so prepare for that!

```
CREATE TABLE MagicalMoves (
        speciesID       integer   NOT NULL REFERENCES DragonSpecies(speciesID),
        magicID         integer   NOT NULL REFERENCES Spells(magicID),
        PRIMARY KEY(speciesID,magicID)
);
```

Functional dependencies:

speciesID,magicID → no dependencies

speciesid integer	magicid integer
1	4
1	7
1	8
1	10
1	9
2	3
2	4
2	8
2	10
3	1
3	2
3	8
4	1
4	5
4	7
4	8
5	2
5	5
5	7
6	2

Views

Dragon Locations View

This view will allow you to easily see all of the dragon species and all of their possible locations.

```
CREATE VIEW DragonsAndLocations
AS
SELECT d.name AS "speciesName", l.name AS "locationName"
FROM DragonSpecies d, Locations l, DragonLocations dl
WHERE d.speciesID = dl.speciesID AND l.locationID = dl.locationID
ORDER BY d.name
```

speciesName text	locationName text
Altreus	Dead Gulf
Altreus	Mount Doom
Altreus	Eastern Sea
Black Drake	Everdark Forest
Bolt Lizard	Giri Desert
Bolt Lizard	Mount Doom
Earthshatterer	Mulgore Grasslands
Earthshatterer	Giri Desert
Eastern Lava-Eater	Mount Doom
Red Drake	Mount Doom
Southern Sea Terror	Eastern Sea
Southern Sea Terror	Dead Gulf
Sun Serpent	Mulgore Grasslands
Sun Serpent	Giri Desert
Wind Wyrm	Mulgore Grasslands

Dragon Spells View

This view allows you to view all of the spells of every dragon species.

```
CREATE VIEW DragonSpells
AS
SELECT d.name AS "speciesName", s.name AS "spellName"
FROM DragonSpecies d, Spells s, MagicalMoves m
WHERE d.speciesID = m.speciesID  AND s.magicID = m.magicID
ORDER BY d.name
```

speciesName text	spellName text
Black Drake	Shadow Breath
Black Drake	Elemental Shield
Black Drake	Elemental Storm
Black Drake	Stealth
Black Drake	Shapeshift
Bolt Lizard	Thunder Breath
Earthshatterer	Thunder Breath
Earthshatterer	Earthquake
Earthshatterer	Elemental Shield
Eastern Lava-Eater	Fire Breath
Eastern Lava-Eater	Earthquake
Eastern Lava-Eater	Elemental Shield
Eastern Lava-Eater	Elemental Storm
Red Drake	Fire Breath
Red Drake	Elemental Storm
Southern Sea Terror	Elemental Storm
Southern Sea Terror	Shadow Breath
Southern Sea Terror	Stealth
Southern Sea Terror	Ice Breath

Dragon Attacks View

This view allows you to view all of the physical attacks of every dragon species!

```
CREATE VIEW DragonAttacks
AS
SELECT d.name AS "speciesName", a.name AS "attackName"
FROM DragonSpecies d, Attacks a, PhysicalMoves p
WHERE d.speciesID = p.speciesID  AND a.attackID = p.attackID
ORDER BY d.name
```

speciesName text	attackName text
Altreus	Claw
Altreus	Bite
Altreus	Tail Swipe
Altreus	Wing Shield
Altreus	Horn Thrust
Black Drake	Bite
Black Drake	Horn Thrust
Bolt Lizard	Claw
Bolt Lizard	Bite
Earthshatterer	Horn Thrust
Earthshatterer	Tail Swipe
Earthshatterer	Bite
Earthshatterer	Claw
Eastern Lava-Eater	Bite
Eastern Lava-Eater	Claw
Eastern Lava-Eater	Horn Thrust
Eastern Lava-Eater	Wing Shield
Red Drake	Claw
Red Drake	Horn Thrust
Red Drake	Wing Shield

Dragon Elements View

This view shows all dragon species, their elements, and their element's weaknesses!

```
CREATE VIEW DragonElements
AS
SELECT d.name AS "speciesName",e.name AS "elementName", w.weakAgainst AS
        "weaknessName"
FROM DragonSpecies d, Elements e, ElemWeaknesses w
WHERE d.elementID = e.elementID AND w.elementID = d.elementID
ORDER BY d.name
```

speciesName text	elementNa text	weakness text
Altreus	light	air
Black Drake	darkness	light
Bolt Lizard	thunder	earth
Earthshatterer	earth	air
Earthshatterer	earth	water
Eastern Lava-Eater	fire	water
Eastern Lava-Eater	fire	light
Eastern Lava-Eater	fire	earth
Red Drake	fire	water
Red Drake	fire	earth
Red Drake	fire	light
Southern Sea Terror	water	thunder
Southern Sea Terror	water	darkness
Sun Serpent	light	air
Wind Wyrm	air	darkness
Wind Wyrm	air	thunder

Reports and Their Queries

Dragons that have a weakness to water

Selects all dragon species that have a weakness to water and displays their name, element, and location so that you can go splash them with a bucket or something.

```
SELECT d.name AS "speciesName", e.name AS "elementName", l.name as "locationName"

FROM DragonSpecies d, Elements e, Locations l, ElemWeaknesses w, DragonLocations dl

WHERE d.elementID = w.elementID AND w.elementID = e.elementID AND w.weakAgainst =
        'water' AND dl.speciesID = d.speciesID AND l.locationID = dl.locationID

ORDER BY d.name
```

speciesName text	eleme text	locationName text
Earthshatterer	earth	Mulgore Grasslands
Earthshatterer	earth	Giri Desert
Eastern Lava-Eater	fire	Mount Doom
Red Drake	fire	Mount Doom

Dragons that live in locations that are at least 80% claimed by their kind

Selects all dragons and locations that have at least 80% of their land claimed by dragonkind. This is useful for when you're planning a nice picnic and you want your food to be heated up (or drenched in saltwater in some cases) upon arrival.

```
SELECT d.name AS "speciesName", l.name AS "locationName", l.percentClaimedByDragons
    AS "picnicQuality"
FROM DragonSpecies d, Locations l, DragonLocations dl
WHERE d.speciesID = dl.speciesID AND l.locationID = dl.locationID AND
    l.percentClaimedByDragons > 80
ORDER BY l.percentClaimedByDragons
```

speciesName text	locationName text	picnicQuality integer
Black Drake	Everdark Forest	85
Southern Sea Terror	Dead Gulf	100
Altreus	Dead Gulf	100

Sneaky Dragons

Selects all dragons and locations of dragons that can either use stealth or shapeshifting magic, and shows which one they're using. Sometimes they have both. Spooky tricky dragons.

```
SELECT d.name AS "speciesName", l.name AS "locationName", s.name AS "spellName"
FROM DragonSpecies d, Locations l, DragonLocations dl, Spells s, MagicalMoves m
WHERE d.speciesID = dl.speciesID AND l.locationID = dl.locationID AND
      s.magicID = m.magicID AND d.speciesID = m.speciesID AND s.name = 'Stealth'
OR d.speciesID = dl.speciesID AND l.locationID = dl.locationID AND
      s.magicID = m.magicID AND d.speciesID = m.speciesID AND s.name = 'Shapeshift'
ORDER BY d.name
```

speciesName text	locationName text	spellName text
Altreus	Mount Doom	Shapeshift
Altreus	Eastern Sea	Shapeshift
Altreus	Dead Gulf	Shapeshift
Black Drake	Everdark Forest	Shapeshift
Black Drake	Everdark Forest	Stealth
Southern Sea Terror	Eastern Sea	Stealth
Southern Sea Terror	Dead Gulf	Stealth

Put Wings on Anything and it Instantly Becomes More Terrifying

This query lists all of the dragons, along with their locations, rarity, and size that can fly. Just when you thought it couldn't get any worse, some of these can swoop right out of the air and gobble you right up. Good thing Magnus can fly too. You know, wizards and shit.

```
SELECT d.name AS "speciesName", d.size AS "speciesSize", d.rarity as "speciesRarity", l.name
        AS "locationName"
FROM DragonSpecies d, Locations l, DragonLocations dl
WHERE d.speciesID = dl.speciesID AND l.locationID = dl.locationID AND d.canFly = true
ORDER BY d.size, d.rarity
```

speciesName text	speciesSize size	speciesRarity rarity	locationName text
Wind Wyrm	small	common	Mulgore Grasslands
Sun Serpent	medium	common	Mulgore Grasslands
Sun Serpent	medium	common	Giri Desert
Red Drake	medium	common	Mount Doom
Black Drake	medium	extremely rare	Everdark Forest
Altreus	large	one of a kind	Dead Gulf
Altreus	large	one of a kind	Eastern Sea
Altreus	large	one of a kind	Mount Doom
Eastern Lava-Eater	very large	common	Mount Doom

Stored Procedures

HasTheColor(colorName text)

Returns all of the dragons that are the given color. That way you'll know exactly which one's head will match your mantelpiece best! In fact, this function can be used to search for any string in a description, so search for bulky dragons, slim dragons, or snake-like dragons too!

```
CREATE OR REPLACE FUNCTION HasTheColor(colorName text)
RETURNS TABLE("speciesName" text, "speciesDescription" text) AS $$
BEGIN
RETURN QUERY SELECT d.name AS "speciesName", d.appearance AS "speciesDescription"
FROM DragonSpecies d
WHERE d.appearance LIKE ('%' || colorName || '%')
ORDER BY d.name;
END;
$$ LANGUAGE plpgsql;
```

Example HasTheColor('black'):

hasthecolor record
("Black Drake","Primarily a blueish black with a burnt orange underside. Inside of
("Eastern Lava-Eater","Primarily a deep red with a jet black underside. Inside of

LocationTemp(low integer, high integer)

This function will return all locations that have temperatures within the given range. You can just give it a low temp and it'll return everything higher. I…I mean the Great and Marvelous Magnus wanted this function to allow the user to just give a high temp if so desired, but default parameters are weird and Magnus is tired because it is 5:00 AM and she…I mean he should have started this project earlier but didn't and…Oh never mind here it is:

```
CREATE OR REPLACE FUNCTION LocationTemp(low integer DEFAULT -9001, high
    integer DEFAULT 9001)
RETURNS TABLE("locationName" text, "lowTemp" integer, "highTemp" integer) AS $$
BEGIN
RETURN QUERY SELECT l.name AS "locationName", l.lowestTempInF AS "lowTemp",
    l.highestTempInF AS "highTemp"
FROM Locations l
WHERE l.lowestTempInF >= low AND l.highestTempInF <= high
ORDER BY l.name;
END;
$$ LANGUAGE plpgsql;
```

Example LocationTemp(30)

locationtemp record
("Eastern Sea",30,85)
("Giri Desert",75,150)
("Mount Doom",200,1000)
("Mulgore Grasslands",40,10

Example LocationTemp(40,200)

locationtemp record
("Giri Desert",75,150)
("Mulgore Grasslands",40,100)

DragonAttacks

This function will return the types of attacks a given dragon species likes to use. Simply input the speciesID and you'll be on your way to being murdered and maimed! I mean on your way to conquering a fearsome beast! Yay for reward money and all that, right?

```
CREATE OR REPLACE FUNCTION DragonAttacks(specID integer)
RETURNS TABLE("speciesName" text, "speciesSize" size, "attackName" text, "attackDesc"
    text, "attackRange" text, "attackType" text, "attackSideEffect" text) AS $$
BEGIN
RETURN QUERY SELECT d.name AS "speciesName", d.size AS "speciesSize", a.name AS
    "attackName", a.description AS "attackDesc", a.range AS "attackRange", a.type AS
    "attackType", a.sideEffect AS "attackSideEffect"
FROM Attacks a, PhysicalMoves p, DragonSpecies d
WHERE d.speciesID = p.speciesID AND p.attackID = a.attackID AND d.speciesID = specID
ORDER BY d.name;
END;
$$ LANGUAGE plpgsql;
```

Example select DragonAttacks(6)

dragonattacks record
("Wind Wyrm",small,Bite,"The dragon uses its jaws to clamp down on its victim. Effectiveness depend
("Wind Wyrm",small,"Tail Swipe","The dragon uses its tail to swiftly brush away its victim. This is

Example select DragonAttacks(8)

dragonattacks record
("Red Drake",medium,Claw,"The dragon uses its claws to swipe or rip at its victim. Effectiveness de
("Red Drake",medium,"Horn Thrust","The dragon uses the horns on its head to impale its victim. This
("Red Drake",medium,"Wing Shield","The dragon raises its wings in front of itself to block blows, pr

Triggers

The great and mighty Magnus dislikes triggers, but he is a merciful Archmage and thought they may be useful to you peasant adventurers. A lot goes into making one of these awful things. First, we must create the table that the trigger is going to throw stuff into.

```
CREATE TABLE listener_queue (
        relation            text        NOT NULL,
        relation_pk_value   integer     NOT NULL PRIMARY KEY
);
```

Now that we've created a table for the trigger to throw things into, we've got to create the trigger function that will return the trigger which will call the function that will…just look below.

```
CREATE OR REPLACE FUNCTION dragon_listener() returns trigger as $$
BEGIN
  IF (tg_op = 'DELETE') THEN
    INSERT INTO listener_queue (relation, relation_pk_value) VALUES
        ('DragonSpecies', old.speciesid);
  ELSE
    INSERT INTO listener_queue (relation, relation_pk_value) VALUES
        ('DragonSpecies', new.speciesid);
  END IF;
  RETURN NULL;
END;
$$ LANGUAGE plpgsql;
```

Basically, this stupid func- I mean wonderful, magical function, will wait on an INSERT, UPDATE, or DELETE on the table DragonSpecies. If a DELETE is used, it will put the deleted item's speciesID into the listener_queue. If anything else is used, it will put the new item or updated item into the listener_queue. In other words, it stores changes made to the database so that one can view their heinous mistakes later and weep tears of pain and suffering. Oh, we aren't done yet.

```
CREATE TRIGGER dragon_listener after INSERT or UPDATE or DELETE on DragonSpecies
  for each row execute procedure dragon_listener();
```

Finally, let's test the trigger by inserting a test dragon into the DragonSpecies table:

```
insert into dragonSpecies(speciesID,name,elementID,appearance,size,timeActive,canFly,rarity)
        VALUES (42, 'Test Drake', 7, 'Test description', 'small','Night',true,'one of a kind');
```

The listener_queue looks like this after the insert.

relation text	relation_pk_value integer
DragonSpecies	42

Security

Now, now. I know all of you adventurers are quite excited about all of this new technolo- I mean magic. However, we have to keep things secure around here so we can't just have anyone messing with the database. Yes, yes, I know I said it would be open source, and it truly is! There are just a few…constraints that we must place on exactly how open our source-magic can be. So here are the types of users that will be able to edit the database:

Magnus the Admin

Only Magnus himself can DELETE and ALTER privileges. This may not be convenient for the average admin, but Magnus is an all knowing and all powerful sorcerer. He will have no issue with maintaining the more delicate parts of his database no matter what era or universe he may be in. That being said, Magnus can of course do all of those other silly things that Archmages and below can do. Hell with it, Magnus gets ALL PRIVELEGES. HE IS GOD. HE IS ETERNAL. BOW TO HIM NOW. MUAHAHAHAHA.

```
CREATE ROLE magnus

GRANT ALL PRIVILEGES

ON ALL TABLES IN SCHEMA PUBLIC  TO

magnus
```

Archmage

Only those with the highest skills in the magical arts can alter existing data. Archmages can UPDATE, INSERT, and of course SELECT data. They cannot, however, ALTER priveleges or DELETE existing data. No one fucks with Magnus's original work. NO ONE.

```
CREATE ROLE archmage

GRANT SELECT, INSERT, UPDATE

ON ALL TABLES IN SCHEMA PUBLIC

TO archmage
```

Wizard

Averagely talented magical folk can INSERT their own data and of course SELECT data, but they cannot UPDATE or change any existing data. That would anger the snooty Archmages, and Magnus has had enough of their whining already.

```
CREATE ROLE wizard

GRANT SELECT, INSERT

ON ALL TABLES IN SCHEMA PUBLIC

TO wizard
```

~~Shitty Muggle~~ Adventurer

Those with no magical ability whatsoever get to do nothing. Alright, fine, they can SELECT data and query it and look at it so they can throw themselves at dragons until the sun goes down or their flesh ends up well done. That should keep them quiet for a while.

```
CREATE ROLE adventurer

GRANT SELECT

ON ALL TABLES IN SCHEMA PUBLIC

TO adventurer
```

Implementation Notes

Let's get this straight right now. Magnus is an all-knowing, all-powerful being. I mean, how else would he have faced the dragon god Altreus and lived to tell the tale? However, he has reported that creating this database actually challenged him. Now, now, don't get your mortal, non-magical panties in a twist. That doesn't mean he's any less amazing. He's so amazing that he's decided to share the hardships of developing the database with the general public! How nice is that? Without further ado, here he is with his report!

Magnus here. I'd have to say the hardest part about creating Dragon Hunter DB was deciding what information about the beasts to keep and what to get rid of. I at one point had primaryColor and secondaryColor as separate entries in the DragonSpecies table. I eventually decided to throw a lot of information in the descriptions instead of having separate columns for information that was unnecessary. Another difficult decision was what to make an enumerated type and what to just make text, but I'll get more into that in the next section.

Known Problems

Is this some kind of joke? Everyone knows that there isn't a single possible way that there could be a *problem* with a database created by the all-powerful sorcer- sorry I'm groveling again. Here's Magnus again with more stuff to say.

Yo. Sorry about the apprentice. He's a little kiss-ass because he thinks that'll get him into the advanced magic class faster, but it sure as Merlin's beard won't. Anyway, there are a few problems with DHDB as it is right now. One problem is that certain things are just text fields and should be enumerated types. Other things are enumerated types that aren't giving enough information. For example, a dragon's size is an enumerated type that ranges from 'extremely small' to 'extremely large.' What does that even mean? How small is extremely small? We're talking about dragons, here, so that could be the size of a horse or the size of a house. In case you were wondering, extremely small is about the size of your hand, while extremely large is about the size of a fat, rich king's castle. Still, who knows how large a fat king's castle is? That's why we need numerical measurements for size.

Future Enhancements

Future enhancements would include dragons from all around our planet and from different time periods, past and present. Right now, we only have dragons from within our country's borders, mostly because we kind of like that dragons are murdering people in our enemy countries. However, dragons are becoming an increasing menace that'll spread to our beloved country if we don't help soon. So, we'll be adding way more dragons, attacks, locations, and spells in the future. There may even be new elements! I've been hearing about poison dragons and metal dragons, so I'll have to check that out soon. Until then, good luck, good hunting, and don't use a damn wooden shield against a fire breathing dragon for goodness sakes. Some adventurers never learn.