Database Proposal

Prepared for: Alan Labouseur, Professor
Prepared by: Grace McCue, Student
April 24, 2016

Table of Contents

EXECUTIVE SUMMARY

Objective

To design and implement a database for The Wizarding World of Harry Potter.

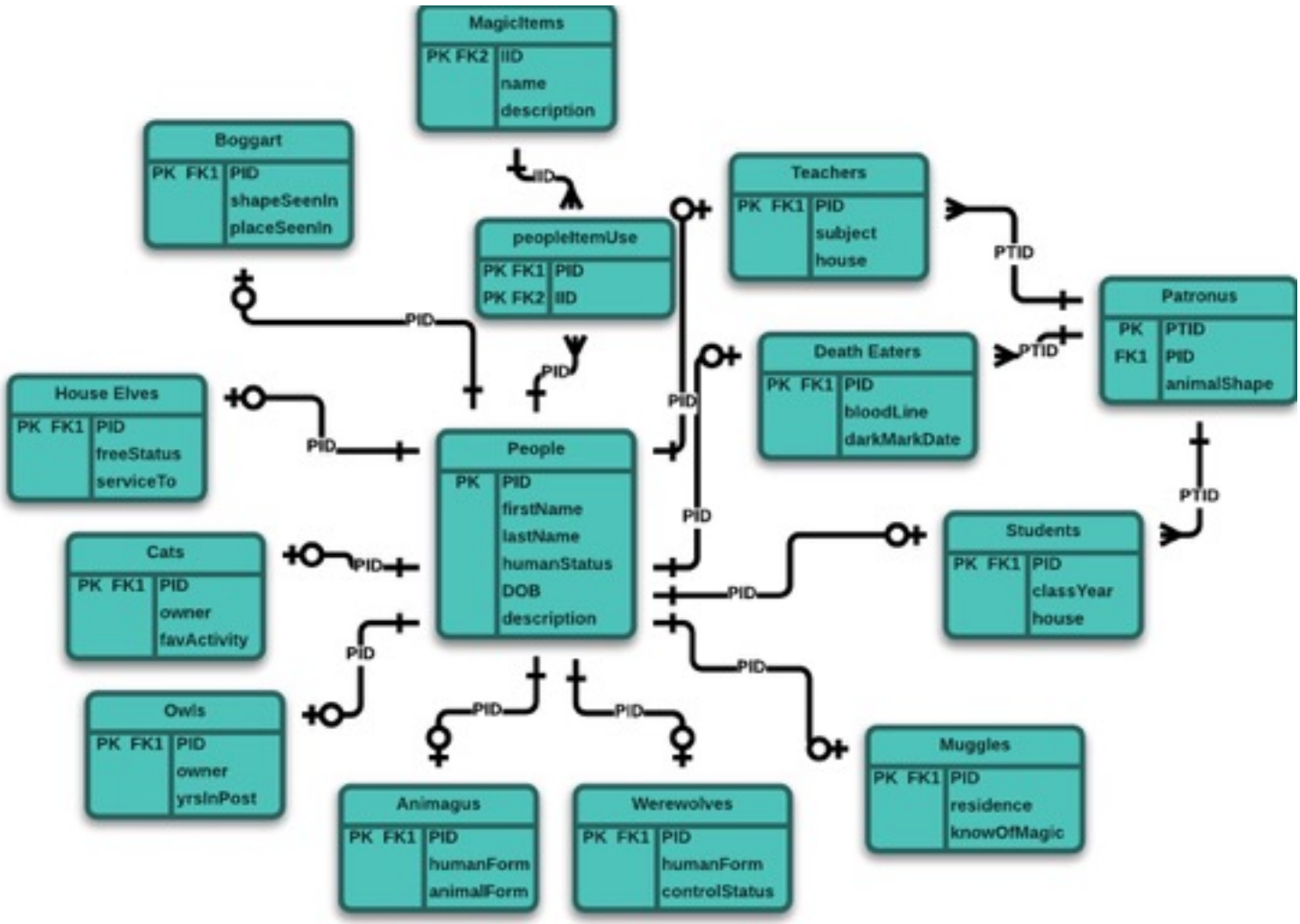
Goals

The goal of this database is to clearly layout the types of humans and nonhumans (and those in-between) that are found throughout the realm of Harry Potter. Users of this database will include anyone interested in or curious about a magical, wonderful world outside of this one.

Project Outline

First the reader will find the Entity Relationship Diagram, this outlines a few pieces of data in this magical world of Harry Potter. The reader then will be able to see all the tables created in the database along with their contents. Next the reader will be able to see a few examples of some queries. Stored procedures and their triggers may also be viewed in this document along with security for the database and any issues and how they may be fixed.

ENTITY RELATIONSHIP DIAGRAM



TABLES

PEOPLE TABLE

The People table holds all the people that can be found in and around the magical world of Harry Potter and their basic attributes.

```
CREATE TABLE people (
        PID             char(4)         NOT NULL,
        firstName       varchar(100)    NOT NULL,
        lastName        varchar(100)    NOT NULL,
        humanStatus     varchar(100)    NOT NULL,
        DOB             date            NOT NULL,
        description     varchar(250)    NOT NULL,
        CONSTRAINT people_pkey PRIMARY KEY (PID)
);
```

functional dependencies

PID —> firstName, lastName, humanStatus, DOB, description

sample data on next page…

	pid [PK] character(4)	firstname character varying(100)	lastname character varying(100)	humanstatus character varying(100)	dob date	description character varying(
1	p001	Harry	Potter	human	1980-07-31	dark hair, glas
2	p002	Ron	Weasley	human	1980-03-01	red hair, hamm
3	p003	Hermione	Granger	human	1979-09-19	crazy brown hai
4	p004	Draco	Malfoy	human	1980-06-05	blond hair, thi
5	p005	Neville	Longbottom	human	1980-07-30	brown hair, alv
6	p006	Albus	Dumbledore	human	1881-06-30	long grey hair
7	p007	Minerva	McGonagall	human and non-human	1935-10-04	brown/grey hai
8	p008	Severus	Snape	human	1960-01-09	shoulder lengt
9	p009	Rubeus	Hagrid	human and non-human	1928-12-06	half-giant wiz
10	p010	Remus	Lupin	human and non-human	1960-03-10	great professor
11	p011	Filius	Flitwick	human and non-human	1963-10-07	part goblin, s
12	p012	Gilderoy	Lockhart	human	1964-01-26	good looking, c
13	p013	Ginevra	Weasley	human	1981-08-11	red hair, hamm
14	p014	Luna	Lovegood	human	1981-02-13	long blond hai
15	p015	Pomona	Sprout	human	1931-05-15	short and fat,
16	p016	Alan	Labouseur	human	1985-01-12	obviously awso
17	p017	Bellatrix	Lestrange	human	1951-05-02	long curly bla
18	p018	Lucius	Malfoy	human	1954-02-17	long bold hair,
19	p019	Peter	Pettigrew	human and non-human	1960-05-17	crazy hair, ug
20	p020	Tom	Riddel	human	1926-12-31	aka Lord Volde
21	p021	Petunia	Dursley	human	1959-05-12	smug, thinks s
22	p022	Vernon	Dursley	human	1955-05-12	fat, uggly, tri
23	p023	Dudley	Dursley	human	1980-06-23	fat, ugly, mak
24	p024	Grace	McCue	human	1995-02-12	beutiful and a

STUDENTS TABLE

The students table is a subtype of the people table, it holds all the students that can be found at Hogwarts, the best school of witchcraft and wizardry in the world, and their attributes.

```
CREATE TABLE students (
        PID             char(4)         NOT NULL,
        classYear       varchar(100)    NOT NULL,
        house           varchar(100)    NOT NULL,
        CONSTRAINT students_pkey PRIMARY KEY (PID),
        FOREIGN KEY (PID) REFERENCES people (PID)
);
```

	pid [PK] character(4)	classyear character varying(100)	house character varying(100)
1	p001	graduated 2000	gryffindor
2	p002	graduated 2000	gryffindor
3	p003	graduated 2000	slytherin
4	p005	graduated 2000	gryffindor
5	p007	graduated 1955	gryffindor
6	p008	graduated 1980	slytherin
7	p009	graduated 1948	gryffindor
8	p010	graduated 1980	gryffindor
9	p011	graduated 1983	ravenclaw
10	p012	graduated 1984	ravenclaw
11	p013	graduated 2001	gryffindor
12	p014	graduated 2001	ravenclaw
13	p015	graduated 1951	hufflepuff

functional dependencies

PID —> classYear, house

TEACHERS TABLE

The Teachers table is a subtype of the people table, it holds all the teachers that can be found at Hogwarts and their attributes.

```
CREATE TABLE teachers (
        PID             char(4)         NOT NULL,
        subject         varchar(100)    NOT NULL,
        house           varchar(100)    NOT NULL,
        CONSTRAINT teachers_pkey PRIMARY KEY (PID),
        FOREIGN KEY (PID) REFERENCES people (PID)
);
```

	pid [PK] character(4)	subject character varying(100)	house character varying(100)
1	p007	Transfiguration	gryffindor
2	p008	Potions	slytherin
3	p009	Care of Magical Creatures	gryffindor
4	p010	Defense Against the Dark Arts	gryffindor
5	p011	Charms	ravenclaw
6	p012	Defense Against the Dark Arts	ravenclaw
7	p015	Herbology	hufflepuff
8	p016	Magical Databases	gryffindor

functional dependencies

PID —> subject, house

DEATH EATERS

The Death Eaters table is a subtype of the people table, it holds all the death eaters that are known about in the world of Harry Potter and their attributes.

CREATE TABLE deathEaters (

PID char(4) NOT NULL,

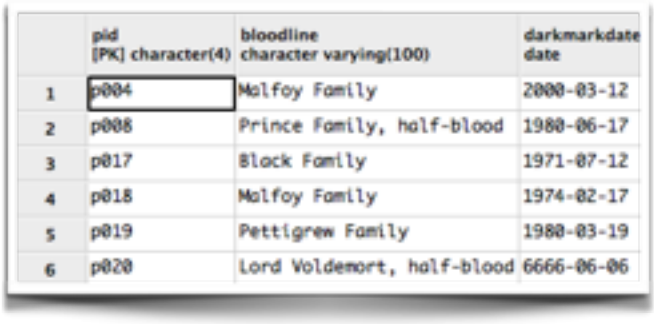
bloodLine varchar(100) NOT NULL,

darkMarkDate date NOT NULL,

CONSTRAINT deathEaters_pkey PRIMARY KEY (PID),

FOREIGN KEY (PID) REFERENCES people (PID)

);



	pid [PK] character(4)	bloodline character varying(100)	darkmarkdate date
1	p004	Malfoy Family	2000-03-12
2	p008	Prince Family, half-blood	1980-06-17
3	p017	Black Family	1971-07-12
4	p018	Malfoy Family	1974-02-17
5	p019	Pettigrew Family	1980-03-19
6	p020	Lord Voldemort, half-blood	6666-06-06

functional dependencies

PID —> bloodLine, darkMarkDate

MUGGLES

The Muggles tables is a subtype of the people table, it holds all the muggle that may or may not know about the magical world of Harry Potter, and the attributes they poses.

CREATE TABLE muggles (

PID char(4) NOT NULL,

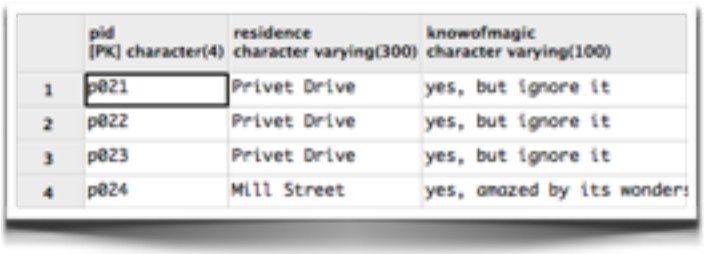
residence varchar(300) NOT NULL,

knowOfMagic varchar(100) NOT NULL,

CONSTRAINT muggles_pkey PRIMARY KEY (PID),

FOREIGN KEY (PID) REFERENCES people(PID)

);



	pid [PK] character(4)	residence character varying(300)	knowofmagic character varying(100)
1	p021	Privet Drive	yes, but ignore it
2	p022	Privet Drive	yes, but ignore it
3	p023	Privet Drive	yes, but ignore it
4	p024	Mill Street	yes, amazed by its wonder

functional dependencies

PID —> residence, knowOfMagic

HOUSE ELVES TABLE

The House Elves Table is a subtype of the people table, it holds all the house elves in the world of Harry Potter and their attributes.

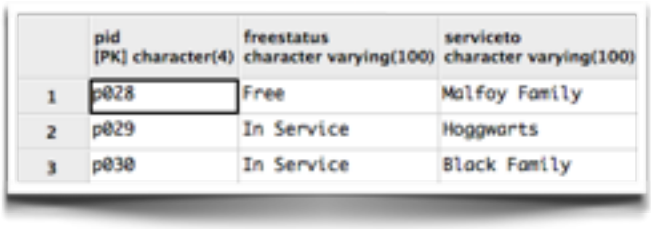
CREATE TABLE houseElves (

PID	char(4)	NOT NULL,
freeStatus	varchar(100)	NOT NULL,
serviceTo	varchar(100)	NOT NULL,

CONSTRAINT houseElves_pkey PRIMARY KEY (PID),

FOREIGN KEY (PID) REFERENCES people(PID)

);



	pid [PK] character(4)	freestatus character varying(100)	serviceto character varying(100)
1	p028	Free	Malfoy Family
2	p029	In Service	Hoggwarts
3	p030	In Service	Black Family

functional dependencies

PID —> freeStatus, serviceTo

CATS TABLE

The Cats table is a subtype of the people table, it holds all of the cats that are found in the magical world of Harry Potter and their attributes.

CREATE TABLE cats (

PID	char(4)	NOT NULL,
owner	varchar(100)	NOT NULL,
favActivity	varchar(300)	NOT NULL,

CONSTRAINT cats_pkey PRIMARY KEY (PID),

FOREIGN KEY (PID) REFERENCES people(PID)

);

	pid [PK] character(4)	owner character varying(100)	favactivity character varying(300)
1	p031	Ron Weasley	annoying ron
2	p032	Argus Filch	watching harry and his friends

functional dependencies

PID —> owner, favActivity

OWLS TABLE

The owls table is a subtype of the people table, it holds all of the owls that can be found in the magical world of Harry Potter and their attributes.

CREATE TABLE owls (

PID char(4) NOT NULL,

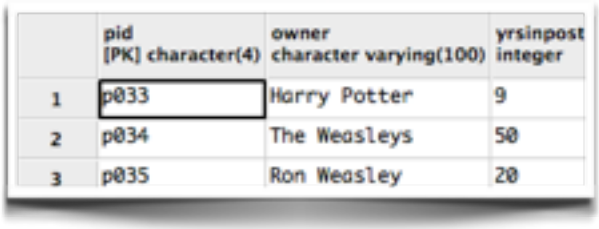
owner varchar(100) NOT NULL,

yrsInPost integer NOT NULL,

CONSTRAINT owls_pkey PRIMARY KEY (PID),

FOREIGN KEY (PID) REFERENCES people(PID)

);



	pid [PK] character(4)	owner character varying(100)	yrsinpost integer
1	p033	Harry Potter	9
2	p034	The Weasleys	50
3	p035	Ron Weasley	20

functional dependencies

PID —> owner, yrsInPost

WEREWOLVES TABLE

The werewolves table is a subtype of the people table, it holds all the known werewolves in the world of Harry Potter and their attributes.

CREATE TABLE werewolves (

PID char(4) NOT NULL,

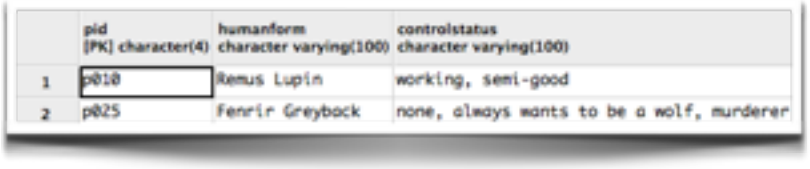
humanForm varchar(100) NOT NULL,

controlStatus varchar(100) NOT NULL,

CONSTRAINT werewolves_pkey PRIMARY KEY (PID),

FOREIGN KEY (PID) REFERENCES people(PID)

);



	pid [PK] character(4)	humanform character varying(100)	controlstatus character varying(100)
1	p010	Remus Lupin	working, semi-good
2	p025	Fenrir Greyback	none, always wants to be a wolf, murderer

functional dependencies

PID —> humanForm, controlStatuS

ANIMAGUS TABLE

The Animagus table is a subtype of the people table. it holds all the known animagus people in the world of Harry Potter and their attributes.

CREATE TABLE animagus (

PID char(4) NOT NULL,

humanForm varchar(100) NOT NULL ,

animalForm varchar(100) NOT NULL,

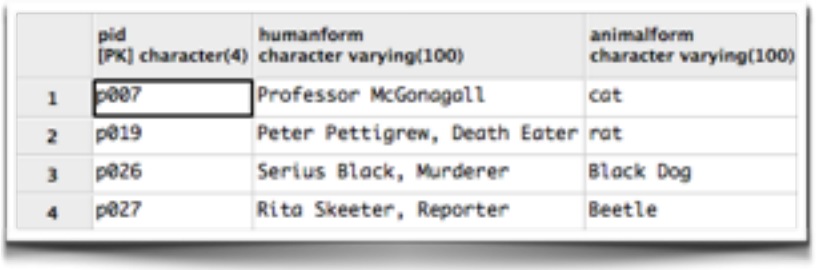
CONSTRAINT animagus_pkey PRIMARY KEY (PID),

FOREIGN KEY (PID) REFERENCES people(PID)

);

functional dependencies

PID —> humanForm, animalForm



	pid [PK] character(4)	humanform character varying(100)	animalform character varying(100)
1	p007	Professor McGonagall	cat
2	p019	Peter Pettigrew, Death Eater	rat
3	p026	Serius Black, Murderer	Black Dog
4	p027	Rita Skeeter, Reporter	Beetle

BOGGART TABLE

The Boggart table is a subtype of the people table, it holds all the known Boggarts in the wizarding world and their known attributes.

CREATE TABLE boggart (

PID char(4) NOT NULL,

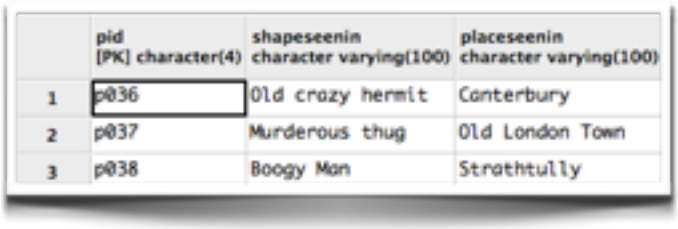
shapeSeenIn varchar(100) NOT NULL,

placeSeenIn varchar(100) NOT NULL,

CONSTRAINT boggart_pkey PRIMARY KEY (PID),

FOREIGN KEY (PID) REFERENCES people(PID)

);



	pid [PK] character(4)	shapeseenin character varying(100)	placeseenin character varying(100)
1	p036	Old crazy hermit	Canterbury
2	p037	Murderous thug	Old London Town
3	p038	Boogy Man	Strathtully

functional dependencies

PID —> shapeSeenIn, placeSeenIn

PATRONUS TABLE

The Patronus table holds peoples patronus, the shape, and who it belongs to.

CREATE TABLE patronus (

PTID	char(5)	NOT NULL,
PID	char(4)	NOT NULL,
animalShape	varchar(100)	NOT NULL,

CONSTRAINT patronus_pkey PRIMARY KEY (PTID),

FOREIGN KEY (PID) REFERENCES people(PID)

);

functional dependencies

PTID —> PID, animalShape

	ptid [PK] character(5)	pid character varying(100)	animalshape character varying(100)
1	pt001	p001	stag
2	pt002	p002	terrier
3	pt003	p003	otter
4	pt004	p006	pheonix
5	pt005	p007	cat
6	pt006	p008	doe
7	pt007	p010	wolf
8	pt008	p012	butterfly
9	pt009	p013	horse
10	pt010	p014	hare
11	pt011	p016	horse

MAGICITEMS TABLE

The MagicItems table holds item ID, the name of the item, and its description.

CREATE TABLE magicItems (

IID	char(4)	NOT NULL,
name	varchar(100)	NOT NULL,
description	varchar(200)	NOT NULL,

CONSTRAINT magicItems_pkey PRIMARY KEY (IID)

);

functional dependencies

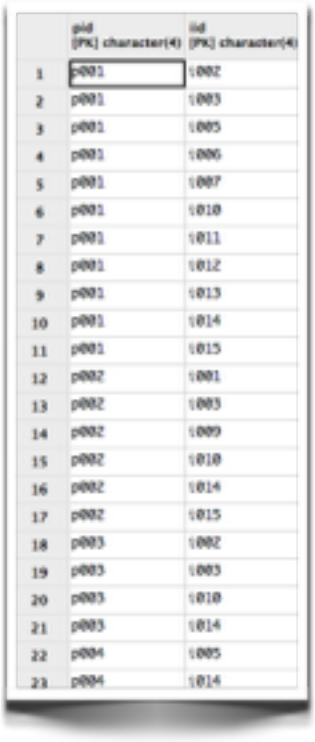
IID —> name, description

	iid [PK] character(4)	name character varying(100)	description character varying(200)
1	i001	Howler	to convey an extremly angry message very loudly and publicl
2	i002	Time-Turner	time travel piece, resembles an hourglass on a necklace
3	i003	invisibility cloak	renders whatever it covers unseeable
4	i004	deluminator	used to remove or absorb the light from any light source t
5	i005	invisibility cloak	renders whatever it covers unseeable
6	i006	elder wand	extremely powerful wand made of elder wood with a core of T
7	i007	resurrection stone	allows the holder to bring back deceased loved ones, in a s
8	i008	the marauders map	magical map of hogwarts made by "Mony", "Wormtail", "Padfo
9	i009	remembrall	small, clear orb, about the size of a tennis ball, containi
10	i010	weasley family clock	a special clock in their home, the Burrow, with nine hands,
11	i011	Wizards Chess	The pieces are magically animated, and they violently attac
12	i012	invisibility cloak	renders whatever it covers unseeable
13	i013	Tom Riddles Diary	tom riddles second horcrux due to the killing of mooning my
14	i014	Goblet of Fire	used solely to choose the participating school champions, s
15	i015	Philosophers Stone	it changes all metals to gold, and can be used to brew a po
16	i016	Sorting Hat	able to read minds and determine which of the four school h
17	i017	Arthur Weasleys Ford Anglia	the vehicle can fly, become invisible, and carry the entire

PEOPLEITEMUSE TABLE

The peopleItemUse table is basically an inventory of when a person uses a magical item. It contains person ID and item ID.

```
CREATE TABLE peopleItemUse (
        PID     char(4)         NOT NULL,
        IID     char(4)         NOT NULL,
        PRIMARY KEY (PID, IID),
        FOREIGN KEY (PID) REFERENCES people (PID),
        FOREIGN KEY (IID) REFERENCES magicItems (IID)
);
```



	pid [PK] character(4)	iid [PK] character(4)
1	p001	i002
2	p001	i003
3	p001	i005
4	p001	i006
5	p001	i007
6	p001	i010
7	p001	i011
8	p001	i012
9	p001	i013
10	p001	i014
11	p001	i015
12	p002	i001
13	p002	i003
14	p002	i009
15	p002	i010
16	p002	i014
17	p002	i015
18	p003	i002
19	p003	i003
20	p003	i010
21	p003	i014
22	p004	i005
23	p004	i014

functional dependencies

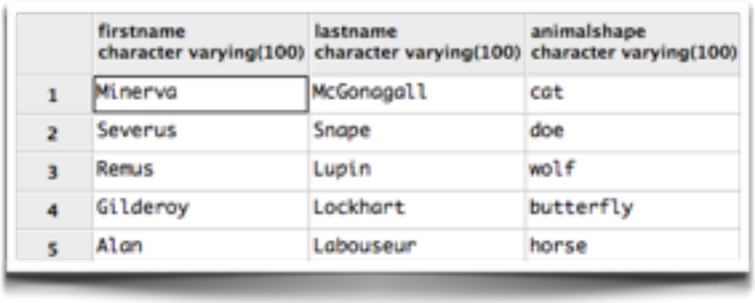
(PID, IID) —> NA

VIEWS

TEACHERS PATRONUS VIEW

The TeachersPatronus View shows all the first names, last names, and animal shapes of all the teachers with a known patronus.

```
CREATE VIEW TeachersPatronus AS

SELECT pe.firstname, pe.lastname, pt.animalshape

FROM people pe, patronus pt, teachers t

WHERE pe.pid = pt.pid AND pt.pid = t.pid;
```

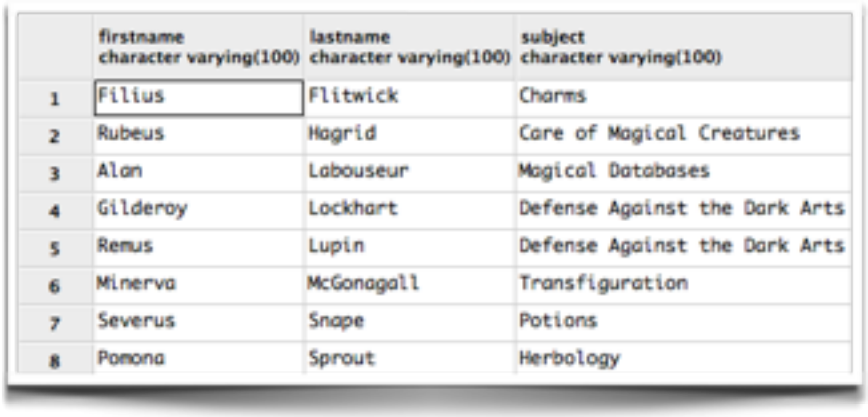


	firstname character varying(100)	lastname character varying(100)	animalshape character varying(100)
1	Minerva	McGonagall	cat
2	Severus	Snape	doe
3	Remus	Lupin	wolf
4	Gilderoy	Lockhart	butterfly
5	Alan	Labouseur	horse

TEACHER SUBJECTS VIEW

The teacher subjects view shows all first name, last name, and subject of all teachers.

```
CREATE VIEW teacherSubjects AS

SELECT firstName, lastName, subject

FROM people

INNER JOIN teachers

ON people.pid = teachers.pid

ORDER BY lastName;
```

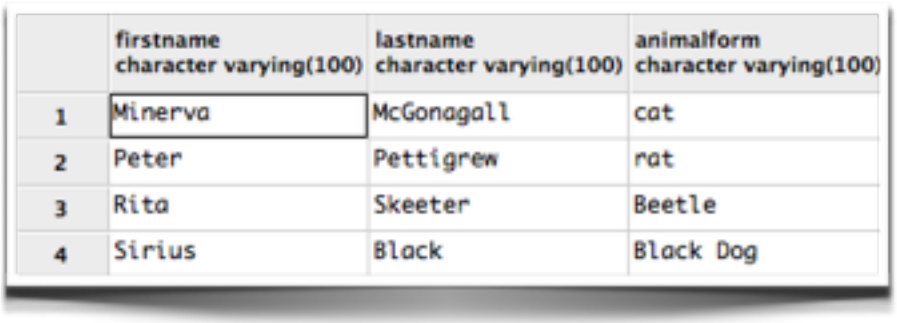


	firstname character varying(100)	lastname character varying(100)	subject character varying(100)
1	Filius	Flitwick	Charms
2	Rubeus	Hagrid	Care of Magical Creatures
3	Alan	Labouseur	Magical Databases
4	Gilderoy	Lockhart	Defense Against the Dark Arts
5	Remus	Lupin	Defense Against the Dark Arts
6	Minerva	McGonagall	Transfiguration
7	Severus	Snape	Potions
8	Pomona	Sprout	Herbology

HUMAN AND NON-HUMAN ANIMAGUS FORM VIEW

Th human and non-human animagus form view shows the human and non-human people's first name and last name and their animagus form.

```
CREATE VIEW peopleAnimagus AS

SELECT firstName, lastName, animalForm

FROM people p

INNER JOIN animagus a

ON p.pid = a.pid

ORDER BY firstName;
```



	firstname character varying(100)	lastname character varying(100)	animalform character varying(100)
1	Minerva	McGonagall	cat
2	Peter	Pettigrew	rat
3	Rita	Skeeter	Beetle
4	Sirius	Black	Black Dog

REPORTS

Interesting Queries - these queries will demonstrate the potential of this database and the information it holds. Each query will exemplify a way to extract data from this database and be able to use it for useful information.

QUERY 1

To return the percent of teachers that are in house Gryffindor

```
SELECT trunc (
        CAST(
            ( SELECT count (t.pid) as gryffindorCount
             FROM teachers t
             WHERE t.house = 'gryffindor'
            ) AS DECIMAL (5,2)
           )
              /(SELECT count (t.pid) AS peopleHouse
               FROM teachers t
              )
    * 100
           ) AS percent_Gryffindor;
```

	percent_gryffindor numeric
1	50

QUERY 2

To return the first name and last name of the teacher for magical databases.

```
SELECT firstName, lastName
FROM people
WHERE pid IN
        (SELECT pid
         FROM teachers
         WHERE subject = 'Magical Databases');
```

	firstname character varying(100)	lastname character varying(100)
1	Alan	Labouseur

QUERY 3

To return the first name, last name, dark mark date, and class year of people who were/are both students and death eaters.

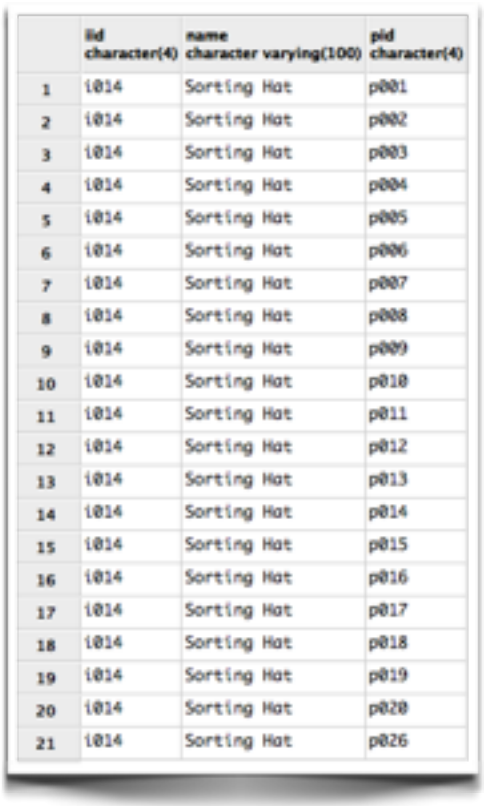
```
SELECT p.firstname, p.lastname, de.darkmarkdate, s.classyear
FROM people p
JOIN deathEaters de
ON p.pid = de.pid
JOIN students s
ON de.pid = s.pid
ORDER BY darkmarkdate desc;
```

	firstname character varying(100)	lastname character varying(100)	darkmarkdate date	classyear character varying(100)
1	Draco	Malfoy	2000-03-12	graduated 2000
2	Severus	Snape	1980-06-17	graduated 1980

QUERY 4

To return all the people that have used a particular item.

```
SELECT i.iid, i.name, pi.pid

FROM magicItems i, peopleItemUse pi

WHERE pi.iid = i.iid AND i.iid = 'i014'

ORDER BY i.name;
```



	iid character(4)	name character varying(100)	pid character(4)
1	i014	Sorting Hat	p001
2	i014	Sorting Hat	p002
3	i014	Sorting Hat	p003
4	i014	Sorting Hat	p004
5	i014	Sorting Hat	p005
6	i014	Sorting Hat	p006
7	i014	Sorting Hat	p007
8	i014	Sorting Hat	p008
9	i014	Sorting Hat	p009
10	i014	Sorting Hat	p010
11	i014	Sorting Hat	p011
12	i014	Sorting Hat	p012
13	i014	Sorting Hat	p013
14	i014	Sorting Hat	p014
15	i014	Sorting Hat	p015
16	i014	Sorting Hat	p016
17	i014	Sorting Hat	p017
18	i014	Sorting Hat	p018
19	i014	Sorting Hat	p019
20	i014	Sorting Hat	p020
21	i014	Sorting Hat	p026

STORED PROCEDURES

These are stored functions that are created for the purpose to be always able to be called on and conduct statements and or calculations automatically so that a query does not need to be used each time.

STORED PROCEDURE 1: PERSONSITEM

This stored procedure shows the items and the people that use them based off of the items themselves.

```
CREATE OR REPLACE FUNCTION personsItem (pid varchar (4))

RETURNS TABLE ( "first name" varchar (100), "last name" varchar (100), "item" varchar (100)) AS

$body$

   BEGIN

      SELECT p.firstname AS "first name", p.lastname AS "Last Name", i.name AS "Item"

      FROM people p, magicItems i, peopleItemUse pi

     WHERE pi.iid = i.iid AND p.pid = pi.pid

      ORDER BY i.name;

   END;

$body$

LANGUAGE plpgsql;
```

	first name character varying(100)	Last Name character varying(100)	Item character varying(100)
1	Harry	Potter	Arthur Weasleys Ford Anglia
2	Ron	Weasley	Arthur Weasleys Ford Anglia
3	Ginevra	Weasley	Arthur Weasleys Ford Anglia
4	Harry	Potter	Goblet of Fire
5	Albus	Dumbledore	Goblet of Fire
6	Ron	Weasley	Howler
7	Tom	Riddel	Philosophers Stone
8	Harry	Potter	Philosophers Stone
9	Pomona	Sprout	Sorting Hat
10	Harry	Potter	Sorting Hat
11	Ron	Weasley	Sorting Hat
12	Hermione	Granger	Sorting Hat
13	Draco	Malfoy	Sorting Hat
14	Neville	Longbottom	Sorting Hat

STORED PROCEDURE 2: GRYFFINDORITEMS

This stored procedure shows all the teachers that belong to gryffindor house and the magical items that they have used.

```
CREATE OR REPLACE FUNCTION GryffindorsItems (pid varchar(4))
RETURNS TABLE ("first name" varchar (100), "last name" varchar (100), "item" varchar (100)) AS
$body$
   BEGIN
      SELECT p.firstname AS "first name", p.lastname AS "Last Name", i.name AS "Item"
      FROM people p, magicItems i, peopleItemUse pi, teachers t
      WHERE pi.iid = i.iid AND p.pid = pi.pid AND t.pid = p.pid AND t.house = 'gryffindor'
      ORDER BY i.name;
   END;
$body$
LANGUAGE plpgsql;
```

	first name character varying(100)	Last Name character varying(100)	Item character varying(100)
1	Minerva	McGonagall	Sorting Hat
2	Rubeus	Hagrid	Sorting Hat
3	Remus	Lupin	Sorting Hat
4	Alan	Labouseur	Sorting Hat
5	Remus	Lupin	the marauders map

STORED PROCEDURE 3: TEACHERDOB

This stored procedure shows the date of births of all teachers.

```
CREATE OR REPLACE FUNCTION teacherdob (pid varchar(4))
RETURNS TABLE ( "dateOfBirth" varchar(100)) AS
$body$
   BEGIN
      SELECT p.dob AS "dateOfBirth"
      FROM people p, teachers t
      WHERE p.pid = t.pid
      ORDER BY p.dob;
   END;
$body$
LANGUAGE plpgsql;
```

	dateOfBirth date
1	1928-12-06
2	1931-05-15
3	1935-10-04
4	1960-01-09
5	1960-03-10
6	1963-10-07
7	1964-01-26
8	1985-01-12

TRIGGERS

These are a special kind of stored procedure that automatically execute with events such as insert, update, or delete.

TRIGGER 1: GRYFFINDORITEMS ()

This trigger automatically adds an item to someone in Gryffindor when an insert has been done on the magic items table.

```
CREATE OR REPLACE FUNCTION GryffindorsItems ()

RETURNS TRIGGER AS

$body$

   BEGIN

      SELECT p.firstname AS "first name", p.lastname AS "Last Name", i.name AS "Item"

      FROM people p, magicItems i, peopleItemUse pi, teachers t

      WHERE pi.iid = i.iid AND p.pid = pi.pid AND t.pid = p.pid AND t.house = 'gryffindor'

      ORDER BY i.name;

   END;

$body$

LANGUAGE plpgsql;


CREATE TRIGGER gryffindorsItems

AFTER INSERT ON magicItems

FOR EACH ROW

EXECUTE PROCEDURE gryffindorsItems();
```

TRIGGER 2: PERSONSITEMS

This trigger automatically adds an item to a person at the update of the peopleItemsUse table.

```
CREATE OR REPLACE FUNCTION personsItem ()
RETURNS TRIGGER AS
$body$
   BEGIN
      SELECT p.firstname AS "first name", p.lastname AS "Last Name", i.name AS "Item"
      FROM people p, magicItems i, peopleItemUse pi
      WHERE pi.iid = i.iid AND p.pid = pi.pid
      ORDER BY i.name;
   END;
$body$
LANGUAGE plpgsql;

CREATE TRIGGER personsItem
AFTER INSERT ON peopleItemUse
FOR EACH ROW
EXECUTE PROCEDURE personsItem();
```

SECURITY

This section shows the administrator and the users that are able to have specific permissions in the database.

ADMIN

People who are allowed access of every part and aspect of this database, usually only a select few.

CREATE ROLE admin;

GRANT ALL ON ALL TABLES IN SCHEMA public

TO admin;

WORLD

The rest. Anyone who would like to see and use the database for any reason, excluding the elected administrators.

CREATE ROLE world;

GRANT SELECT ON ALL TABLES IN SCHEMA PUBLIC

TO world;

IMPLICATION NOTES, KNOWN PROBLEMS, AND FUTURE ENHANCEMENTS

This database is meant to be an information storage system for the wizarding world of Harry Potter. Because of the vastness of the magical realms of Hogwarts, witchcraft, and wizardry that ensue in this world, this database just touches the beginnings and is in the earliest, simplest form of what the database is one day going to be.

Implementation of this database had its bumps, but eventually was able to be played out nicely. The beginning of implementing the database came with the issues of how the many different types of people can be anything in the world of harry potter including teachers, students, and death eaters. This was fixed to form what is the second implemented database (the one seen in this proposal). The solution of the type of people became clear that it needed to be addressed in the table itself, hence the humanStatus column in the people table.

All the data throughout this database is based off of the books, the movies, and fan based inputs for information that JK Rowling left out about the world of Harry Potter. One enhancement on this database could be to go into further detail about books vs. movies vs. fan based information.