ORPHAN BLACK

Database Proposal

For Clone Club

Tuesday, April 26, 2016

Table of Contents

I. Executive Summary

II. Entity Relationship Diagram
  - A. ZIP
  - B. People
  - C. Family
  - D. CloneClub
  - E. Clones
  - F. LEDA
  - G. Monitoring
  - H. CASTOR
  - I. Occupations
  - J. Jobs
  - K. CombatSkills
  - L. Skills
  - M. MurderAttempts
  - N. MurderMethod

III. Views
  - A. CloneClub
  - B. Dyad

IV. Reports
  - A. FightPrep
  - B. Monitors
  - C. Families

V. Stored procedures
  - A. increaseLevel(PID, Skill ID)
  - B. assasinationsAttempted(VictimID)
  - C. assasinationAttempts(VictimID)

VI. Triggers
  - A. triggerLevelCheck

VII. Security - grant and revoke for users and groups

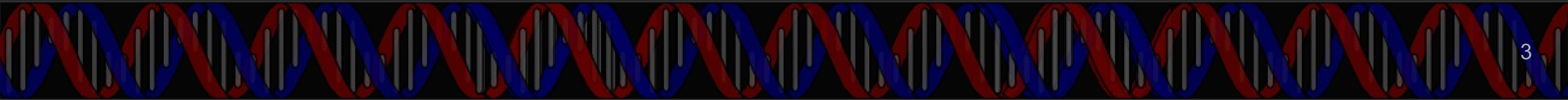
VIII. Implementation Notes

IX. Known Problems

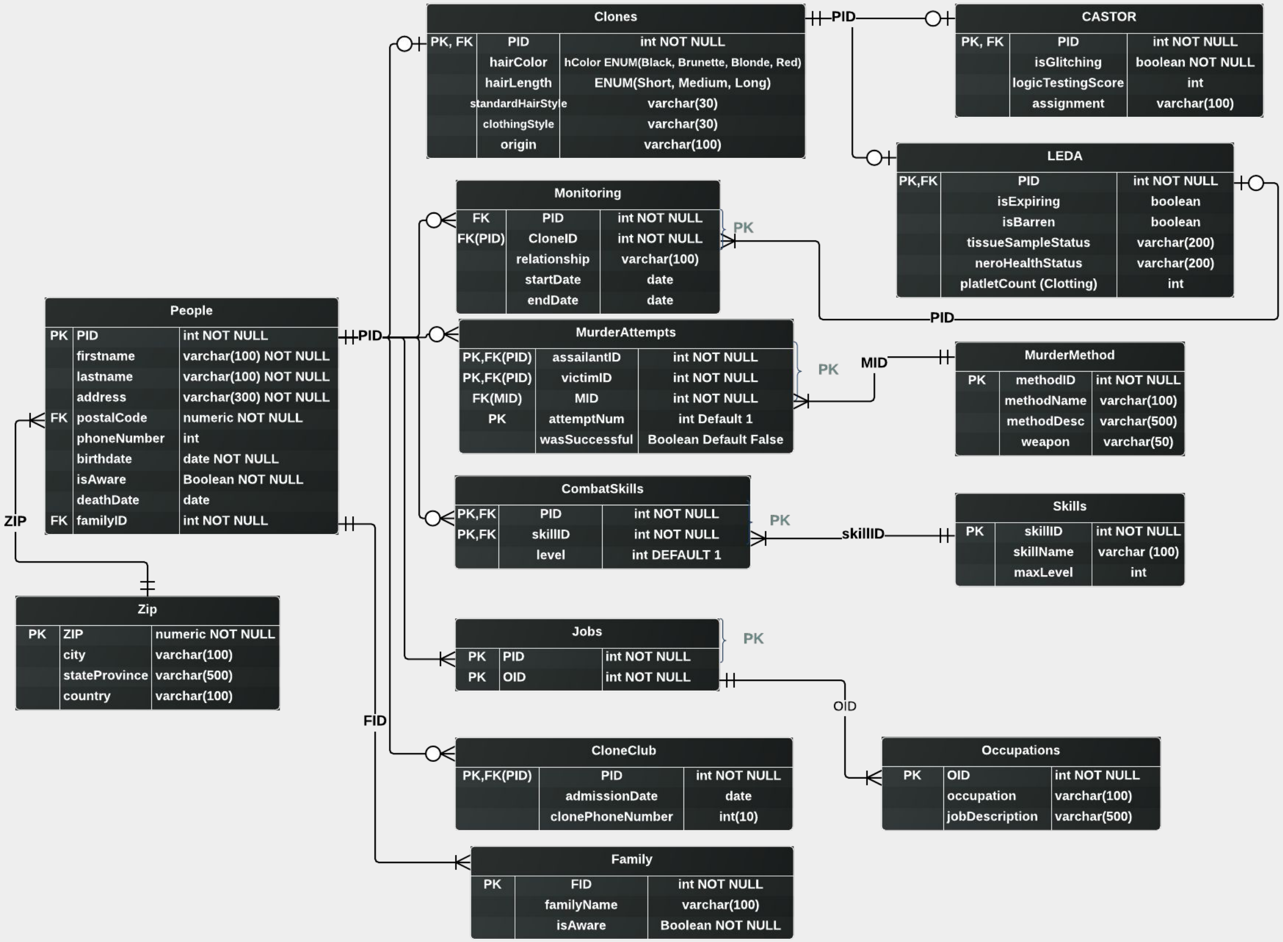
X. Future Enhancements

Executive Summary

The clone club (Sarah Manning and her sister clones) is expanding and constantly learning new information about themselves and their brother clones in the CASTOR Project. As their numbers, allies, and enemies increase they believe a system needs to be put in place to help them keep track of themselves, their brothers their nuclear families and everything that has gone down from Killer-Clones to Pro-Clones. Dyad knowing more than expected as usual expects to have access to this database. Sarah and her sisters agree that that access must be restricted. Additionally they all have different tricks up their sleeve when it comes to fighting. So they also hope to keep track of combat skills and for loyalty purposes who has tried to kill who for example. This is a database for Clone Club.



Entity Relationship Diagram



Clones

PK, FK	PID	int NOT NULL
	hairColor	hColor ENUM(Black, Brunette, Blonde, Red)
	hairLength	ENUM(Short, Medium, Long)
	standardHairStyle	varchar(30)
	clothingStyle	varchar(30)
	origin	varchar(100)

CASTOR

PK, FK	PID	int NOT NULL
	isGlitching	boolean NOT NULL
	logicTestingScore	int
	assignment	varchar(100)

LEDA

PK,FK	PID	int NOT NULL
	isExpiring	boolean
	isBarren	boolean
	tissueSampleStatus	varchar(200)
	neroHealthStatus	varchar(200)
	platletCount (Clotting)	int

Monitoring

FK	PID	int NOT NULL
FK(PID)	CloneID	int NOT NULL
	relationship	varchar(100)
	startDate	date
	endDate	date

People

PK	PID	int NOT NULL
	firstname	varchar(100) NOT NULL
	lastname	varchar(100) NOT NULL
	address	varchar(300) NOT NULL
FK	postalCode	numeric NOT NULL
	phoneNumber	int
	birthdate	date NOT NULL
	isAware	Boolean NOT NULL
	deathDate	date
FK	familyID	int NOT NULL

MurderAttempts

PK,FK(PID)	assailantID	int NOT NULL
PK,FK(PID)	victimID	int NOT NULL
FK(MID)	MID	int NOT NULL
PK	attemptNum	int Default 1
	wasSuccessful	Boolean Default False

MurderMethod

PK	methodID	int NOT NULL
	methodName	varchar(100)
	methodDesc	varchar(500)
	weapon	varchar(50)

CombatSkills

PK,FK	PID	int NOT NULL
PK,FK	skillID	int NOT NULL
	level	int DEFAULT 1

Skills

PK	skillID	int NOT NULL
	skillName	varchar (100)
	maxLevel	int

Zip

PK	ZIP	numeric NOT NULL
	city	varchar(100)
	stateProvince	varchar(500)
	country	varchar(100)

Jobs

PK	PID	int NOT NULL
PK	OID	int NOT NULL

Occupations

PK	OID	int NOT NULL
	occupation	varchar(100)
	jobDescription	varchar(500)

CloneClub

PK,FK(PID)	PID	int NOT NULL
	admissionDate	date
	clonePhoneNumber	int(10)

Family

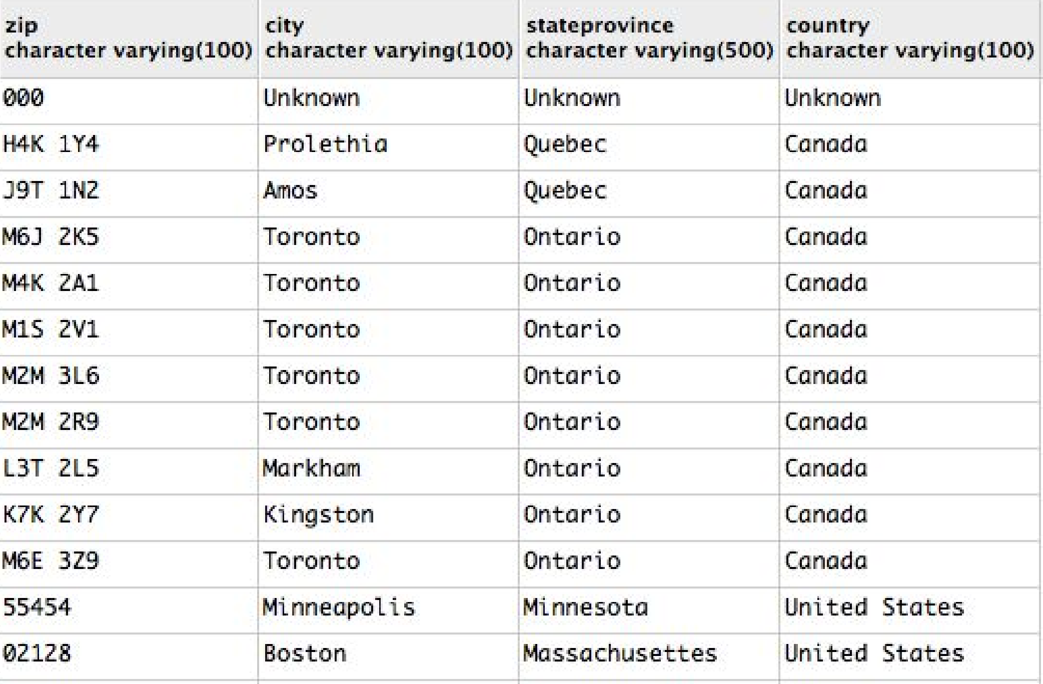
PK	FID	int NOT NULL
	familyName	varchar(100)
	isAware	Boolean NOT NULL

ZIP

```
CREATE TABLE IF NOT EXISTS Zip (
  ZIP varchar(100) NOT NULL,
  city varchar(100),
  stateProvince varchar(500),
  country varchar(100),
  PRIMARY KEY (ZIP)
);
```

Functional Dependencies

ZIP → (city, stateProvince, country)



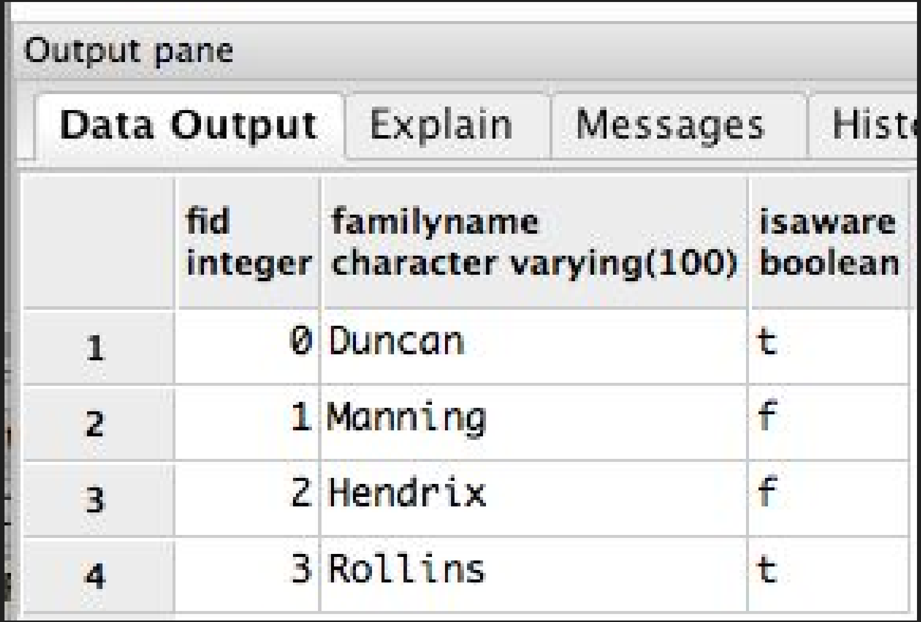
zip character varying(100)	city character varying(100)	stateprovince character varying(500)	country character varying(100)
000	Unknown	Unknown	Unknown
H4K 1Y4	Prolethia	Quebec	Canada
J9T 1N2	Amos	Quebec	Canada
M6J 2K5	Toronto	Ontario	Canada
M4K 2A1	Toronto	Ontario	Canada
M1S 2V1	Toronto	Ontario	Canada
M2M 3L6	Toronto	Ontario	Canada
M2M 2R9	Toronto	Ontario	Canada
L3T 2L5	Markham	Ontario	Canada
K7K 2Y7	Kingston	Ontario	Canada
M6E 3Z9	Toronto	Ontario	Canada
55454	Minneapolis	Minnesota	United States
02128	Boston	Massachusettes	United States

Family

```
CREATE TABLE IF NOT EXISTS Families (
  FID int NOT NULL,
  familyName varchar(100),
  isAware Boolean NOT NULL,
  PRIMARY KEY (FID)
);
```

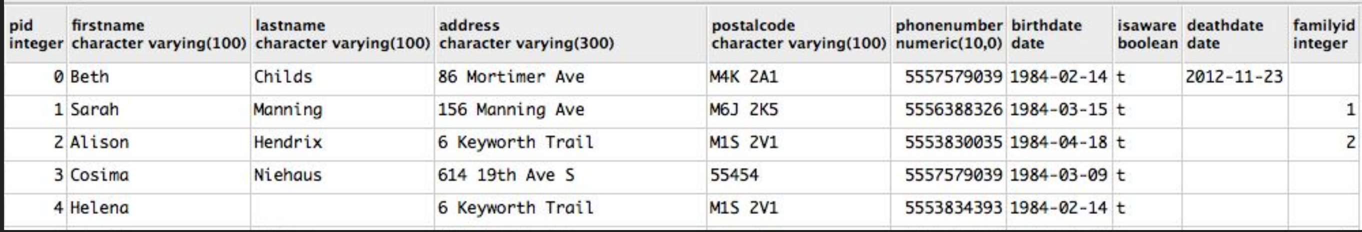
Functional Dependencies

FID → (familyName, isAware)



	fid integer	familyname character varying(100)	isaware boolean
1	0	Duncan	t
2	1	Manning	f
3	2	Hendrix	f
4	3	Rollins	t

People



pid integer	firstname character varying(100)	lastname character varying(100)	address character varying(300)	postalcode character varying(100)	phonenumber numeric(10,0)	birthdate date	isaware boolean	deathdate date	familyid integer
0	Beth	Childs	86 Mortimer Ave	M4K 2A1	5557579039	1984-02-14	t	2012-11-23	
1	Sarah	Manning	156 Manning Ave	M6J 2K5	5556388326	1984-03-15	t		1
2	Alison	Hendrix	6 Keyworth Trail	M1S 2V1	5553830035	1984-04-18	t		2
3	Cosima	Niehaus	614 19th Ave S	55454	5557579039	1984-03-09	t		
4	Helena		6 Keyworth Trail	M1S 2V1	5553834393	1984-02-14	t		

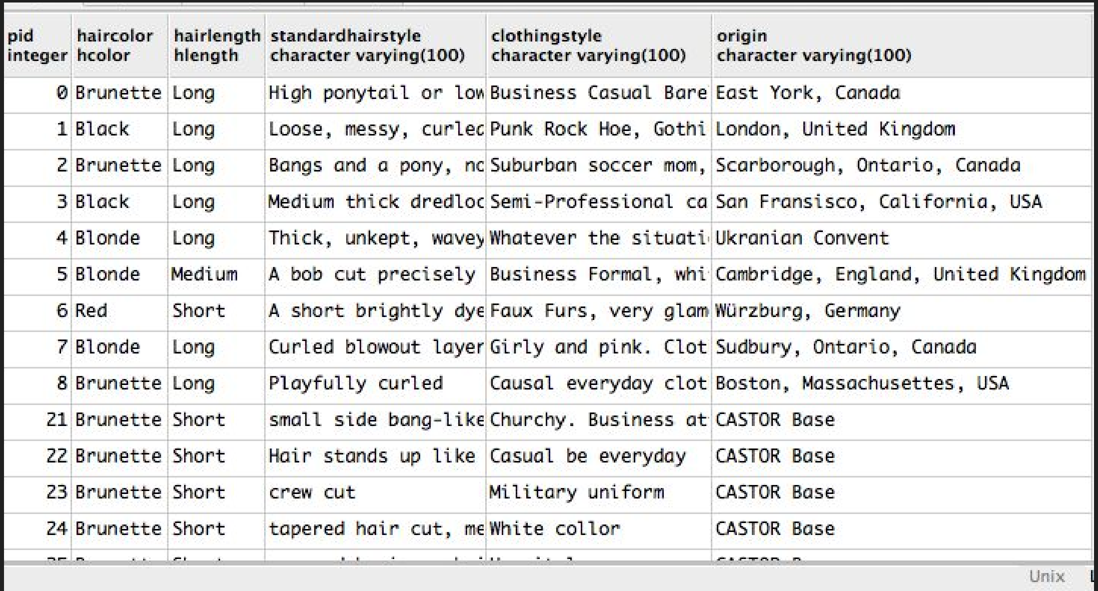
```
CREATE TABLE IF NOT EXISTS People (
  PID int NOT NULL UNIQUE,
  firstname varchar(100) NOT NULL,
  lastname varchar(100) NOT NULL,
  address varchar(300) NOT NULL,
  postalCode varchar(100) NOT NULL,
  phoneNumber numeric(10) NOT NULL,
  birthdate date NOT NULL,
  isAware Boolean NOT NULL,--of Clones
  deathDate date,
  familyID int,
  PRIMARY KEY (PID),
  FOREIGN KEY (postalCode) REFERENCES Zip(ZIP),
  FOREIGN KEY (familyID) REFERENCES Families(FID));
```

Functional Dependencies

PID → (Firstname, lastname, address, postalCode, phoneNumber, birthdate, isAware, deathDate date, familyID)

Clones

```
DROP TYPE IF EXISTS hColor;
CREATE TYPE hColor AS ENUM('Black',
'Brunette', 'Blonde', 'Red', 'Dyed');
DROP TYPE IF EXISTS hLength;
CREATE TYPE hLength AS ENUM('Short',
'Medium', 'Long');

CREATE TABLE IF NOT EXISTS Clones (
  PID int NOT NULL,
  hairColor hColor,
  hairLength hLength,
  standardHairStyle varchar(100),
  clothingStyle varchar(100),
  origin varchar(100),
  PRIMARY KEY (PID),
  FOREIGN KEY (PID) REFERENCES
People(PID));
```



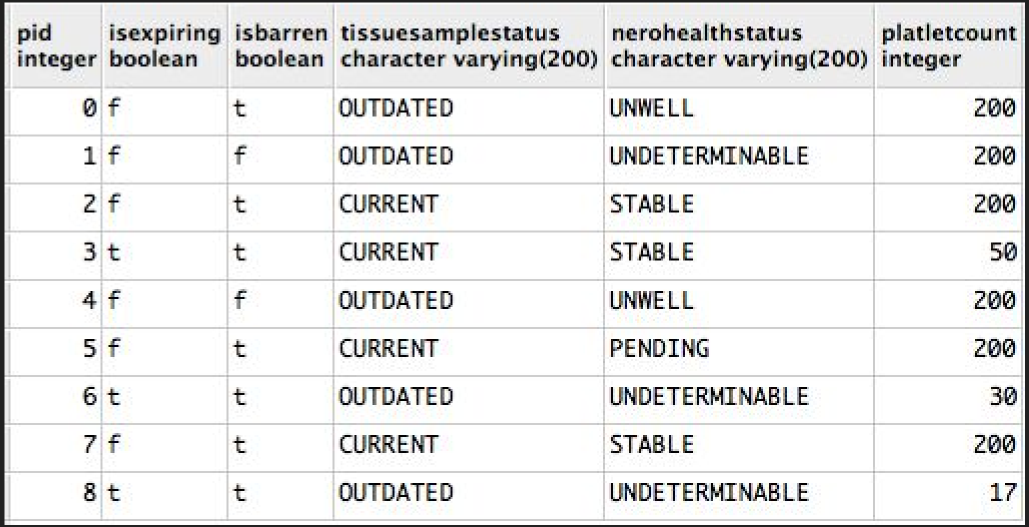
pid	haircolor	hairlength	standardhairstyle	clothingstyle	origin
integer	hcolor	hlength	character varying(100)	character varying(100)	character varying(100)
0	Brunette	Long	High ponytail or low	Business Casual Bare	East York, Canada
1	Black	Long	Loose, messy, curlec	Punk Rock Hoe, Gothi	London, United Kingdom
2	Brunette	Long	Bangs and a pony, nc	Suburban soccer mom,	Scarborough, Ontario, Canada
3	Black	Long	Medium thick dredloc	Semi-Professional ca	San Fransisco, California, USA
4	Blonde	Long	Thick, unkept, wavey	Whatever the situati	Ukranian Convent
5	Blonde	Medium	A bob cut precisely	Business Formal, whi	Cambridge, England, United Kingdom
6	Red	Short	A short brightly dye	Faux Furs, very glam	Würzburg, Germany
7	Blonde	Long	Curled blowout layer	Girly and pink. Clot	Sudbury, Ontario, Canada
8	Brunette	Long	Playfully curled	Causal everyday clot	Boston, Massachusettes, USA
21	Brunette	Short	small side bang-like	Churchy. Business at	CASTOR Base
22	Brunette	Short	Hair stands up like	Casual be everyday	CASTOR Base
23	Brunette	Short	crew cut	Military uniform	CASTOR Base
24	Brunette	Short	tapered hair cut, me	White collor	CASTOR Base

Functional Dependencies

PID → (hairColor, hairLength, standardHairStyle, clothingStyle, origin)

LEDA

```
CREATE TABLE IF NOT EXISTS LEDA (
  PID int NOT NULL,
  isExpiring boolean,
  isBarren boolean,
  tissueSampleStatus varchar(200),
  neuroHealthStatus varchar(200),
  platletCount int, --Clotting: Describes ability to clot.
  PRIMARY KEY (PID),
  FOREIGN KEY (PID) REFERENCES Clones(PID));
```



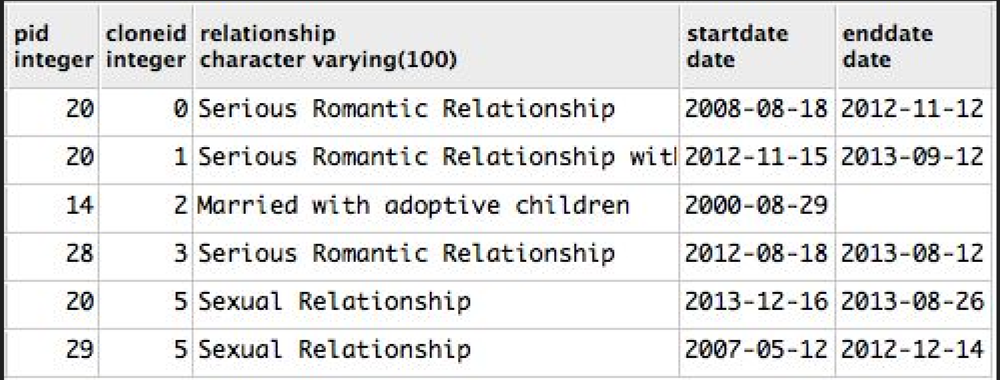
pid	isexpiring	isbarren	tissuesamplestatus	nerohealthstatus	platletcount
integer	boolean	boolean	character varying(200)	character varying(200)	integer
0	f	t	OUTDATED	UNWELL	200
1	f	f	OUTDATED	UNDETERMINABLE	200
2	f	t	CURRENT	STABLE	200
3	t	t	CURRENT	STABLE	50
4	f	f	OUTDATED	UNWELL	200
5	f	t	CURRENT	PENDING	200
6	t	t	OUTDATED	UNDETERMINABLE	30
7	f	t	CURRENT	STABLE	200
8	t	t	OUTDATED	UNDETERMINABLE	17

Functional Dependencies

PID → (isExpiring, isBarren, tissueSampleStatus, neuro)

Monitoring

```
CREATE TABLE IF NOT EXISTS Monitoring (
  PID int NOT NULL,
  CloneID int NOT NULL,
  relationship varchar(100) NOT NULL,
  startDate date NOT NULL,
  endDate date,
  PRIMARY KEY (PID, CloneID),
  FOREIGN KEY (PID) REFERENCES People(PID),
  FOREIGN KEY (CloneID) REFERENCES LEDA(PID));
```



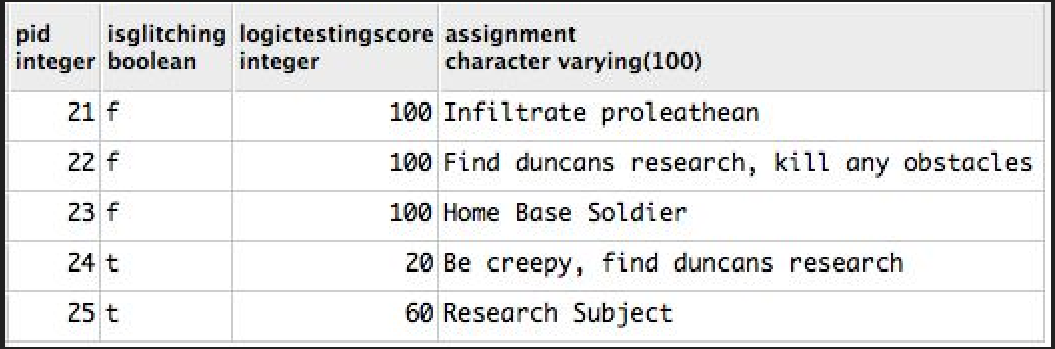
pid integer	cloneid integer	relationship character varying(100)	startdate date	enddate date
20	0	Serious Romantic Relationship	2008-08-18	2012-11-12
20	1	Serious Romantic Relationship witl	2012-11-15	2013-09-12
14	2	Married with adoptive children	2000-08-29	
28	3	Serious Romantic Relationship	2012-08-18	2013-08-12
20	5	Sexual Relationship	2013-12-16	2013-08-26
29	5	Sexual Relationship	2007-05-12	2012-12-14

Functional Dependencies

(PID,CloneID) → (relationship, startDate, endDate)

CASTOR

```
CREATE TABLE IF NOT EXISTS CASTOR (
  PID int NOT NULL,
  isGlitching boolean NOT NULL,
  logicTestingScore int NOT NULL,
  assignment varchar(100),
  PRIMARY KEY (PID),
  FOREIGN KEY (PID) REFERENCES Clones(PID));
```



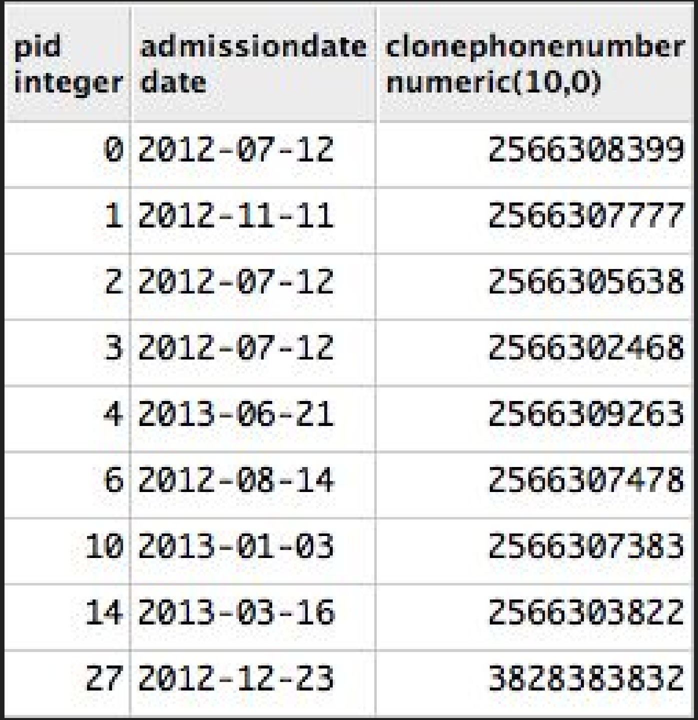
pid integer	isglitching boolean	logictestingscore integer	assignment character varying(100)
21	f	100	Infiltrate proleathean
22	f	100	Find duncans research, kill any obstacles
23	f	100	Home Base Soldier
24	t	20	Be creepy, find duncans research
25	t	60	Research Subject

Functional Dependencies

PID → (isGlitching, logicTestingScore, assignment)

CloneClub

```
CREATE TABLE IF NOT EXISTS CloneClub (
  PID int NOT NULL,
  admissionDate date,
  clonePhoneNumber numeric(10),
  PRIMARY KEY (PID),
  FOREIGN KEY (PID) REFERENCES People(PID));
```



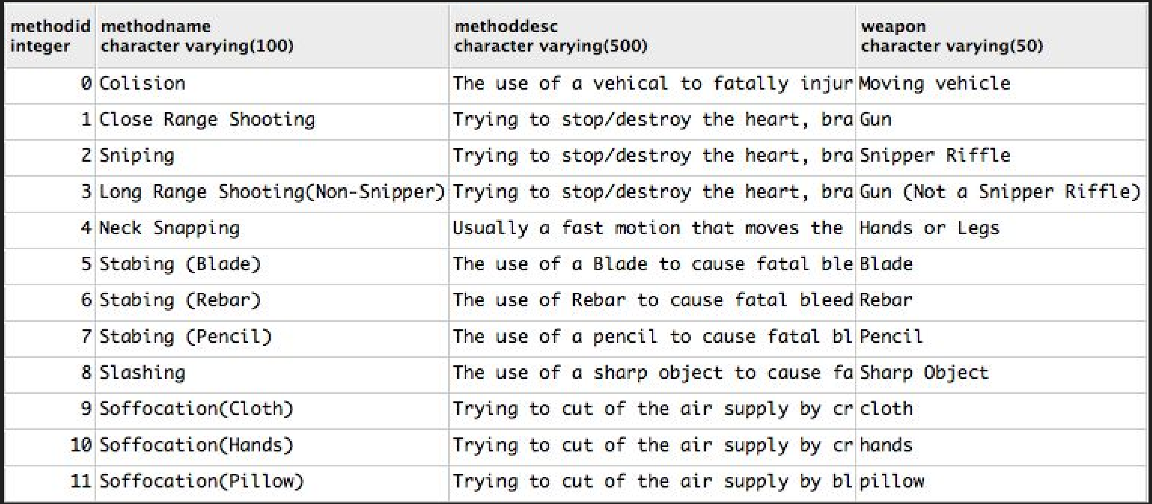
pid integer	admissiondate date	clonephonenumber numeric(10,0)
0	2012-07-12	2566308399
1	2012-11-11	2566307777
2	2012-07-12	2566305638
3	2012-07-12	2566302468
4	2013-06-21	2566309263
6	2012-08-14	2566307478
10	2013-01-03	2566307383
14	2013-03-16	2566303822
27	2012-12-23	3828383832

Functional Dependencies

PID → (admissionDate, clonePhoneNumber)

MurderMethod

```
CREATE TABLE IF NOT EXISTS
MurderMethod (
  methodID int NOT NULL,
  methodName varchar(100),
  methodDesc varchar(500),
  weapon varchar(50),
  PRIMARY KEY (methodID));
```



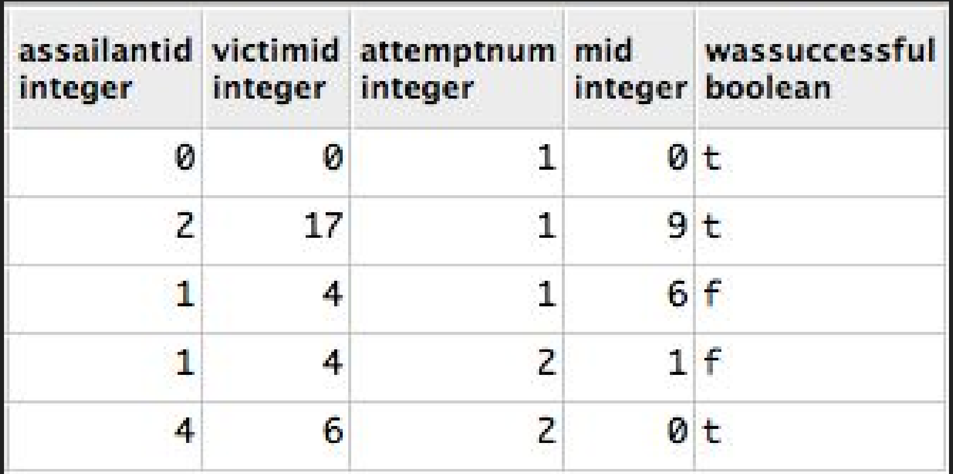
methodid integer	methodname character varying(100)	methoddesc character varying(500)	weapon character varying(50)
0	Colision	The use of a vehical to fatally injur	Moving vehicle
1	Close Range Shooting	Trying to stop/destroy the heart, bra	Gun
2	Sniping	Trying to stop/destroy the heart, bra	Snipper Riffle
3	Long Range Shooting(Non-Snipper)	Trying to stop/destroy the heart, bra	Gun (Not a Snipper Riffle)
4	Neck Snapping	Usually a fast motion that moves the	Hands or Legs
5	Stabing (Blade)	The use of a Blade to cause fatal ble	Blade
6	Stabing (Rebar)	The use of Rebar to cause fatal bleed	Rebar
7	Stabing (Pencil)	The use of a pencil to cause fatal bl	Pencil
8	Slashing	The use of a sharp object to cause fa	Sharp Object
9	Soffocation(Cloth)	Trying to cut of the air supply by cr	cloth
10	Soffocation(Hands)	Trying to cut of the air supply by cr	hands
11	Soffocation(Pillow)	Trying to cut of the air supply by bl	pillow

Functional Dependencies

MethodID → (methodName, methodDesc, weapon)

MurderAttempts

```
CREATE TABLE IF NOT EXISTS
MurderAttempts (
  assailantID int NOT NULL,
  victimID int NOT NULL,
  attemptNum int DEFAULT 1 NOT NULL,
  MID int NOT NULL,
  wasSuccessful Boolean DEFAULT False,
  PRIMARY KEY (assailantID, victimID,
attemptNum),
  FOREIGN KEY (assailantID)
REFERENCES People(PID),
  FOREIGN KEY (victimID) REFERENCES
People(PID));
```



assailantid integer	victimid integer	attemptnum integer	mid integer	wassuccessful boolean
0	0	1	0	t
2	17	1	9	t
1	4	1	6	f
1	4	2	1	f
4	6	2	0	t

Functional Dependencies

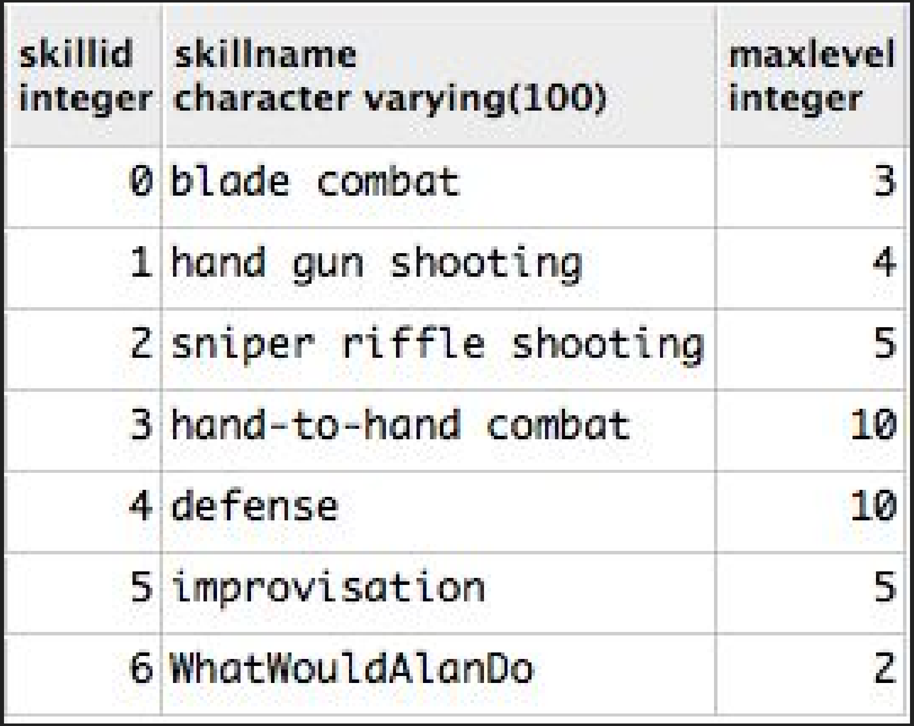
(assailantID, victimID, attemptNum) → (wasSuccessful)

Skills

```
CREATE TABLE IF NOT EXISTS Skills (
  skillID int NOT NULL,
  skillName varchar (100) NOT NULL,
  maxLevel int,
  PRIMARY KEY (skillID));
```

Functional Dependencies

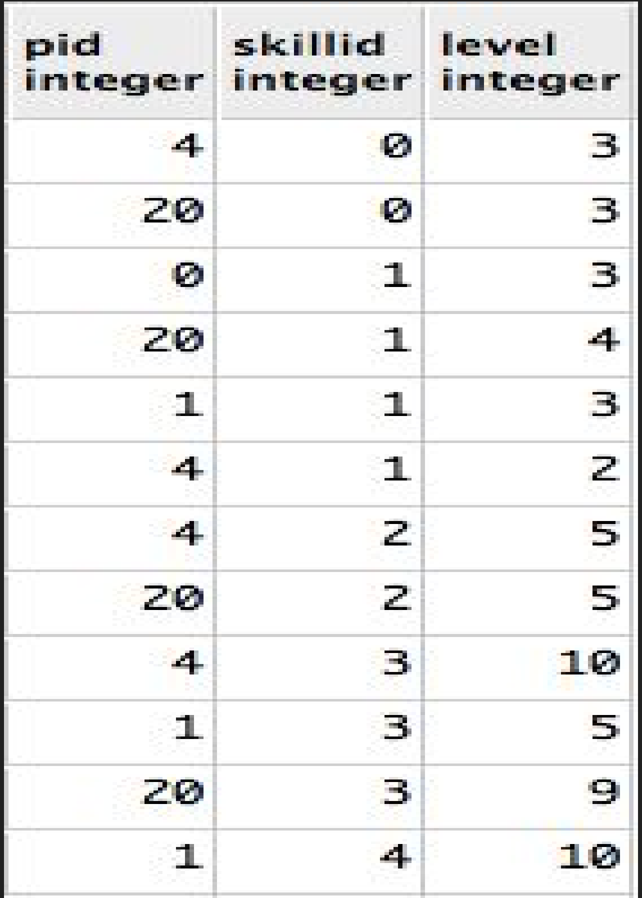
skillID → (skilllName, maxLevel)



skillid	skillname	maxlevel
integer	character varying(100)	integer
0	blade combat	3
1	hand gun shooting	4
2	sniper riffle shooting	5
3	hand-to-hand combat	10
4	defense	10
5	improvisation	5
6	WhatWouldAlanDo	2

CombatSkills

```
CREATE TABLE IF NOT EXISTS
CombatSkills (
  PID int NOT NULL,
  skillID int NOT NULL,
  level int DEFAULT 1,
  PRIMARY KEY (PID, skillID),
  FOREIGN KEY (PID) REFERENCES
People(PID),
  FOREIGN KEY (skillID) REFERENCES
Skills(skillID));
```



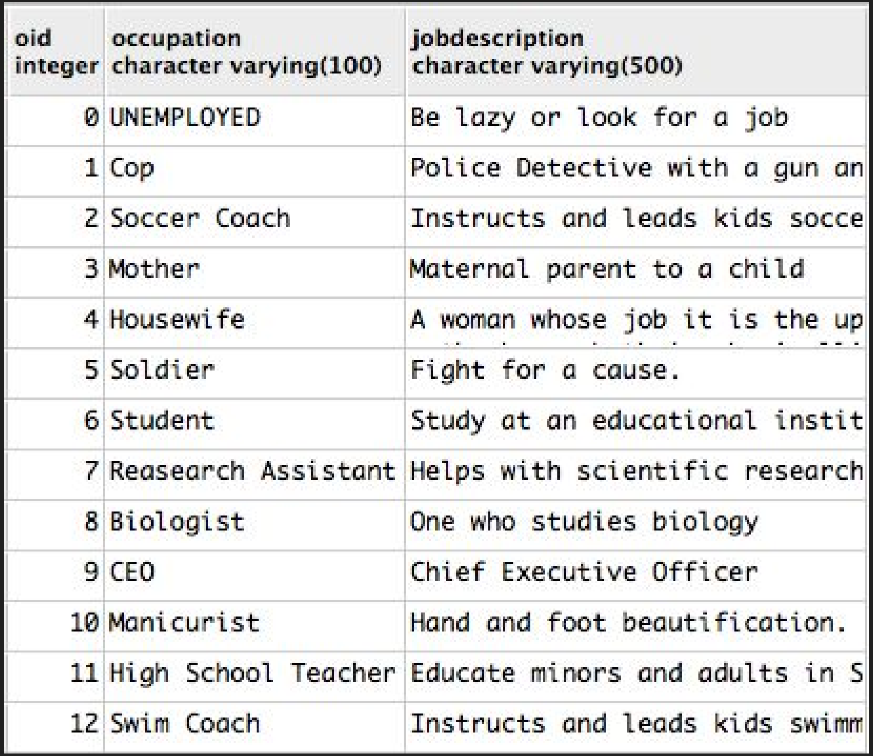
pid integer	skillid integer	level integer
4	0	3
20	0	3
0	1	3
20	1	4
1	1	3
4	1	2
4	2	5
20	2	5
4	3	10
1	3	5
20	3	9
1	4	10

Functional Dependencies

(PID, SkillID) → (level)

Occupations

CREATE TABLE IF NOT EXISTS
Occupations (
  OID int NOT NULL,
  occupation varchar(100),
  jobDescription varchar(500),
  PRIMARY KEY (OID));



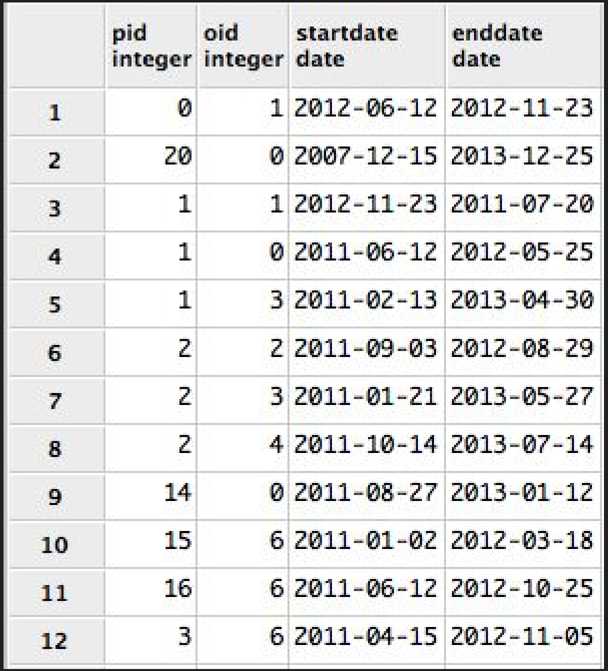
oid integer	occupation character varying(100)	jobdescription character varying(500)
0	UNEMPLOYED	Be lazy or look for a job
1	Cop	Police Detective with a gun an
2	Soccer Coach	Instructs and leads kids socce
3	Mother	Maternal parent to a child
4	Housewife	A woman whose job it is the up
5	Soldier	Fight for a cause.
6	Student	Study at an educational instit
7	Reasearch Assistant	Helps with scientific research
8	Biologist	One who studies biology
9	CEO	Chief Executive Officer
10	Manicurist	Hand and foot beautification.
11	High School Teacher	Educate minors and adults in S
12	Swim Coach	Instructs and leads kids swimm

Functional Dependencies

OID → (occupation, jobDescription)

Jobs

```
CREATE TABLE Jobs (
  PID int NOT NULL,
  OID int NOT NULL,
  startDate date NOT NULL,
  endDate date,
  PRIMARY KEY (PID, OID),
  FOREIGN KEY (PID) REFERENCES
People(PID),
  FOREIGN KEY (OID) REFERENCES
Occupations(OID));
```



	pid integer	oid integer	startdate date	enddate date
1	0	1	2012-06-12	2012-11-23
2	20	0	2007-12-15	2013-12-25
3	1	1	2012-11-23	2011-07-20
4	1	0	2011-06-12	2012-05-25
5	1	3	2011-02-13	2013-04-30
6	2	2	2011-09-03	2012-08-29
7	2	3	2011-01-21	2013-05-27
8	2	4	2011-10-14	2013-07-14
9	14	0	2011-08-27	2013-01-12
10	15	6	2011-01-02	2012-03-18
11	16	6	2011-06-12	2012-10-25
12	3	6	2011-04-15	2012-11-05

Functional Dependencies

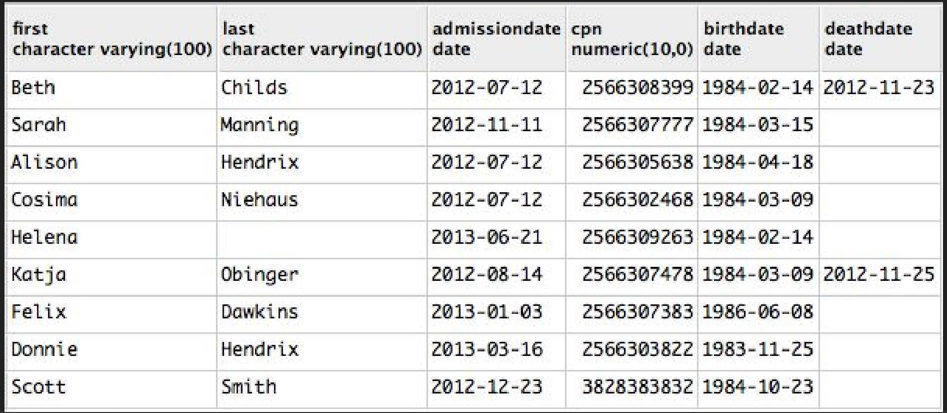
PID → (isGlitching, logicTestingScore, assignment)

Views

View: CloneClubMembers

Clone Club is both self Aware LEDA clones and any clone-aware family members with clone phones. Anyone with a cloneClubID should be able to view everyone else in clone clubs info. Clone phone numbers, addresses etc.

```
--Clone Club View--
CREATE VIEW CloneClubDirectory AS
	SELECT p.firstname AS FIRST, p.lastname AS Last, cc.admissionDate, cc.
clonePhoneNumber AS CPN,
	p.birthdate, p.deathdate
	FROM People p, CloneClub cc
	WHERE p.pid=cc.pid;
```

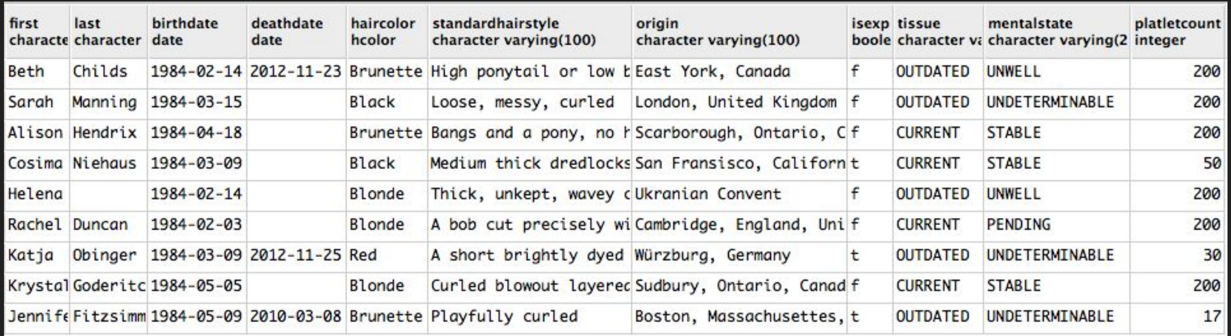


first character varying(100)	last character varying(100)	admissiondate date	cpn numeric(10,0)	birthdate date	deathdate date
Beth	Childs	2012-07-12	2566308399	1984-02-14	2012-11-23
Sarah	Manning	2012-11-11	2566307777	1984-03-15	
Alison	Hendrix	2012-07-12	2566305638	1984-04-18	
Cosima	Niehaus	2012-07-12	2566302468	1984-03-09	
Helena		2013-06-21	2566309263	1984-02-14	
Katja	Obinger	2012-08-14	2566307478	1984-03-09	2012-11-25
Felix	Dawkins	2013-01-03	2566307383	1986-06-08	
Donnie	Hendrix	2013-03-16	2566303822	1983-11-25	
Scott	Smith	2012-12-23	3828383832	1984-10-23	

View: Dyad

The clones want to share their findings to seem peaceful with Dyad but do not want to divulge all of their findings. To prevent Dyad from seeing more than the data on themselves they have made a LEDA Clone only view for Dyad.

```
--Dyad View--
CREATE VIEW Dyad AS
        SELECT p.firstname AS FIRST, p.lastname AS Last, p.birthdate, p.deathdate, c.hairColor,
        c.standardHairStyle, c.origin, l.isExpiring AS isExp, l.tissuesamplestatus AS tissue,
        l.neurohealthstatus AS mentalState, l.platletCount
        FROM People p, Clones c, LEDA l
        WHERE p.pid=c.pid
        AND   c.pid=l.pid;
```



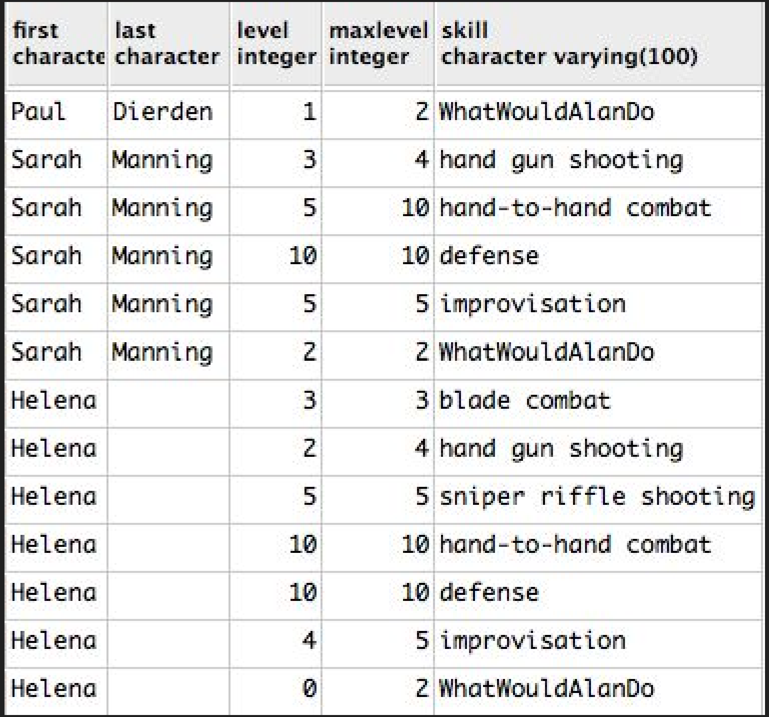
first	last	birthdate	deathdate	haircolor	standardhairstyle	origin	isexp	tissue	mentalstate	platletcount
characte	character	date	date	hcolor	character varying(100)	character varying(100)	boole	character vɛ	character varying(2	integer
Beth	Childs	1984-02-14	2012-11-23	Brunette	High ponytail or low b	East York, Canada	f	OUTDATED	UNWELL	200
Sarah	Manning	1984-03-15		Black	Loose, messy, curled	London, United Kingdom	f	OUTDATED	UNDETERMINABLE	200
Alison	Hendrix	1984-04-18		Brunette	Bangs and a pony, no h	Scarborough, Ontario, C	f	CURRENT	STABLE	200
Cosima	Niehaus	1984-03-09		Black	Medium thick dredlocks	San Fransisco, Californ	t	CURRENT	STABLE	50
Helena		1984-02-14		Blonde	Thick, unkept, wavey c	Ukranian Convent	f	OUTDATED	UNWELL	200
Rachel	Duncan	1984-02-03		Blonde	A bob cut precisely wi	Cambridge, England, Uni	f	CURRENT	PENDING	200
Katja	Obinger	1984-03-09	2012-11-25	Red	A short brightly dyed	Würzburg, Germany	t	OUTDATED	UNDETERMINABLE	30
Krystal	Goderitc	1984-05-05		Blonde	Curled blowout layerec	Sudbury, Ontario, Canad	f	CURRENT	STABLE	200
Jennife	Fitzsimm	1984-05-09	2010-03-08	Brunette	Playfully curled	Boston, Massachusettes,	t	OUTDATED	UNDETERMINABLE	17

Reports

Report: FightPrep

Allows clones to view combatSkills of a person to help prepare them to fight and hopefully win against that person or to choose the best clone to fight against a certain person.

```
--FightPrep Report--
SELECT p.firstname AS FIRST, p.lastname
AS Last, s.skillname AS skill, cs.level, s.
maxLevel
FROM People p, CombatSkills cs, Skills s
WHERE p.pid= cs.pid
AND   s.skillID=cs.skillID;
```

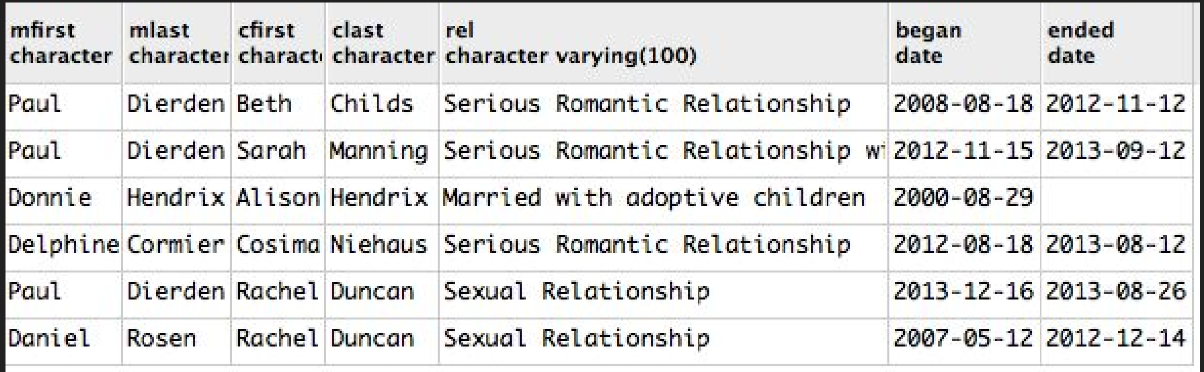


first	last	level	maxlevel	skill
characte	character	integer	integer	character varying(100)
Paul	Dierden	1	2	WhatWouldAlanDo
Sarah	Manning	3	4	hand gun shooting
Sarah	Manning	5	10	hand-to-hand combat
Sarah	Manning	10	10	defense
Sarah	Manning	5	5	improvisation
Sarah	Manning	2	2	WhatWouldAlanDo
Helena		3	3	blade combat
Helena		2	4	hand gun shooting
Helena		5	5	sniper riffle shooting
Helena		10	10	hand-to-hand combat
Helena		10	10	defense
Helena		4	5	improvisation
Helena		0	2	WhatWouldAlanDo

Report: Monitors

Allows clones to observe the monitors that each clone has had currently and in the past.

```
 --Monitor Report--
SELECT m.firstname AS MFIRST, m.lastname AS MLAST,
c.firstname AS CFIRST, c.lastname AS CLAST,
mon.relationship AS rel,
mon.startdate AS BEGAN,
mon.enddate AS ENDED
FROM People m, People c,
Monitoring mon
WHERE mon.pid= m.pid
AND   mon.cloneid = c.pid;
```

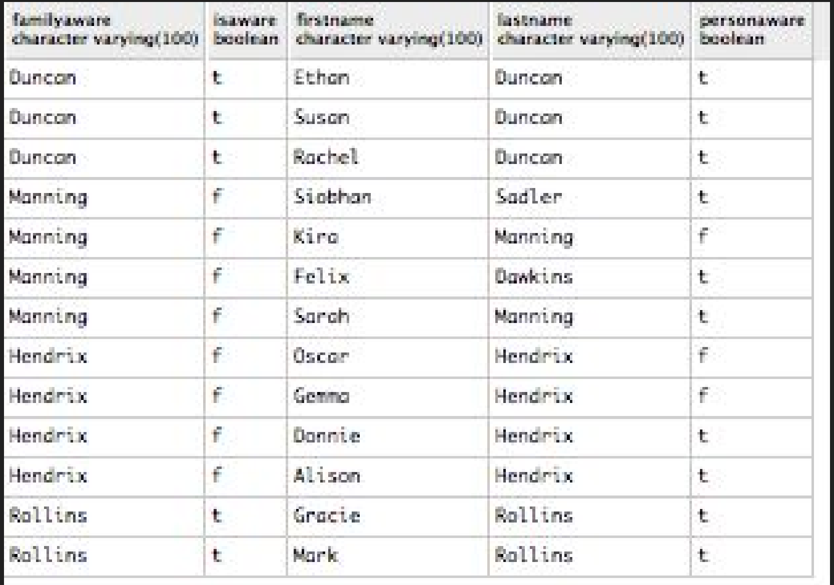


mfirst character	mlast character	cfirst characte	clast character	rel character varying(100)	began date	ended date
Paul	Dierden	Beth	Childs	Serious Romantic Relationship	2008-08-18	2012-11-12
Paul	Dierden	Sarah	Manning	Serious Romantic Relationship wi	2012-11-15	2013-09-12
Donnie	Hendrix	Alison	Hendrix	Married with adoptive children	2000-08-29	
Delphine	Cormier	Cosima	Niehaus	Serious Romantic Relationship	2012-08-18	2013-08-12
Paul	Dierden	Rachel	Duncan	Sexual Relationship	2013-12-16	2013-08-26
Daniel	Rosen	Rachel	Duncan	Sexual Relationship	2007-05-12	2012-12-14

Report: Families

Displays people in families by FID. As in the Family table, family refers to legal family. No cousins, genetic identicals, or non-spousal significant others to keep the table from becoming too complex.

```
--Familes Report--
SELECT f.familyName AS FamilyAware,
isAware AS PersonAware
FROM People p, Families f
WHERE f.fid= p.familyid
ORDER BY f.FID;
```



familyaware character varying(100)	isaware boolean	firstname character varying(100)	lastname character varying(100)	personaware boolean
Duncan	t	Ethan	Duncan	t
Duncan	t	Susan	Duncan	t
Duncan	t	Rachel	Duncan	t
Manning	f	Siobhan	Sadler	t
Manning	f	Kira	Manning	f
Manning	f	Felix	Dawkins	t
Manning	f	Sarah	Manning	t
Hendrix	f	Oscar	Hendrix	f
Hendrix	f	Gemma	Hendrix	f
Hendrix	f	Donnie	Hendrix	t
Hendrix	f	Alison	Hendrix	t
Rollins	t	Gracie	Rollins	t
Rollins	t	Mark	Rollins	t

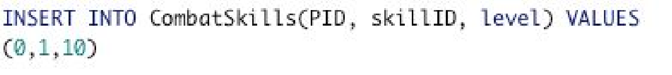
Trigger

Trigger: triggerLevelCheck

This checks whether or not a new combatSkills exceed the maximum skills.skillLevel and throws an error if it does.

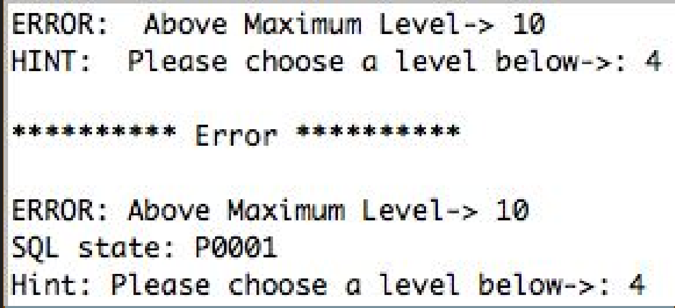
```
CREATE OR REPLACE FUNCTION levelCheck() RETURNS TRIGGER AS $CombatSkills$
        BEGIN
                IF      NEW.level > (SELECT maxLevel
                                     FROM   skills
                                     WHERE skillID=NEW.skillID)
                THEN
                        RAISE EXCEPTION 'Above Maximum Level-> %',      NEW.level
                        USING HINT = 'Please choose a level below->: '|| (SELECT maxLevel
                                                                          FROM   skills
                                                                          WHERE skillID=NEW.skillID);
                END IF;
                return NEW;
        END;
        $CombatSkills$ LANGUAGE plpgsql;

CREATE TRIGGER triggerLevelCheck AFTER INSERT OR UPDATE
ON CombatSkills
FOR EACH ROW EXECUTE PROCEDURE levelCheck();
```

So if we execute…



```
INSERT INTO CombatSkills(PID, skillID, level) VALUES
(0,1,10)
```

Then we will get the error...



```
ERROR:  Above Maximum Level-> 10
HINT:  Please choose a level below->: 4

********** Error **********

ERROR: Above Maximum Level-> 10
SQL state: P0001
Hint: Please choose a level below->: 4
```

Security

Permissions

- Those who are aware only have permissions to see and alter some tables
- Dyad will have the most restricted permissions besides those who are newly aware who have read-only permissions.
- Those who have been in clone club for at least 1 year have full access except for dropping tables
- The database administrator has all permissions including but not limited to create and drop permission.

Clone Aware Role

The clone aware role is defined for those who are newly admitted to clone club. Before giving them full access to our database they are allowed to access this role as they may be CASTOR or DYAD spies and should not have complete access to Clone Club’s information.

```
CREATE ROLE CloneAware;
REVOKE ALL PRIVILEGES ON Jobs FROM CloneAware;
REVOKE ALL PRIVILEGES ON Occupations FROM CloneAware;
REVOKE ALL PRIVILEGES ON CombatSkills FROM CloneAware;
REVOKE ALL PRIVILEGES ON Skills FROM CloneAware;
REVOKE ALL PRIVILEGES ON MurderMethod FROM CloneAware;
REVOKE ALL PRIVILEGES ON MurderAttempts FROM
CloneAware;
REVOKE ALL PRIVILEGES ON CloneClub FROM CloneAware;
REVOKE ALL PRIVILEGES ON CASTOR FROM CloneAware;
REVOKE ALL PRIVILEGES ON Monitoring FROM CloneAware;
REVOKE ALL PRIVILEGES ON LEDA FROM CloneAware;
REVOKE ALL PRIVILEGES ON Clones FROM CloneAware;
REVOKE ALL PRIVILEGES ON People FROM CloneAware;
REVOKE ALL PRIVILEGES ON Families FROM CloneAware;
REVOKE ALL PRIVILEGES ON Zip FROM CloneAware;

GRANT SELECT ON Jobs TO CloneAware;
GRANT SELECT ON Occupations TO CloneAware;
GRANT SELECT ON CombatSkills TO CloneAware;
GRANT SELECT ON Skills TO CloneAware;
GRANT SELECT ON CloneClub TO CloneAware;
GRANT SELECT ON LEDA TO CloneAware;
GRANT SELECT ON Clones TO CloneAware;
GRANT SELECT ON People TO CloneAware;
GRANT SELECT ON Zip TO CloneAware;
```

Dyad Role

The Dyad role restricts Dyad personnel from being able to access information that Clone Club wishes to hide from them. This indicates information found by Clone Club and unknown to Dyad. Additionally, it includes information Clone Club is aware unbenounced to Dyad.

```
CREATE ROLE Dyad;
REVOKE ALL PRIVILEGES ON Jobs FROM Dyad;
REVOKE ALL PRIVILEGES ON Occupations FROM Dyad;
REVOKE ALL PRIVILEGES ON CombatSkills FROM Dyad;
REVOKE ALL PRIVILEGES ON Skills FROM Dyad;
REVOKE ALL PRIVILEGES ON MurderMethod FROM Dyad;
REVOKE ALL PRIVILEGES ON MurderAttempts FROM Dyad;
REVOKE ALL PRIVILEGES ON CloneClub FROM Dyad;
REVOKE ALL PRIVILEGES ON CASTOR FROM Dyad;
REVOKE ALL PRIVILEGES ON Monitoring FROM Dyad;
REVOKE ALL PRIVILEGES ON LEDA FROM Dyad;
REVOKE ALL PRIVILEGES ON Clones FROM Dyad;
REVOKE ALL PRIVILEGES ON People FROM Dyad;
REVOKE ALL PRIVILEGES ON Families FROM Dyad;
REVOKE ALL PRIVILEGES ON Zip FROM Dyad;

GRANT SELECT ON Occupations TO Dyad;
GRANT SELECT, UPDATE, INSERT ON Monitoring TO Dyad;
GRANT SELECT, INSERT ON LEDA TO Dyad;
```

Clone Club Role

The clone club role is defined those who are in clone club. These people are in the “circle of trust.” Only those who have proved their loyalty or are pioneers of Clone Club have access through this role.

```
CREATE ROLE CloneClub;
REVOKE ALL PRIVILEGES ON Jobs FROM CloneClub;
REVOKE ALL PRIVILEGES ON Occupations FROM CloneClub;
REVOKE ALL PRIVILEGES ON CombatSkills FROM CloneClub;
REVOKE ALL PRIVILEGES ON Skills FROM CloneClub;
REVOKE ALL PRIVILEGES ON MurderMethod FROM CloneClub;
REVOKE ALL PRIVILEGES ON MurderAttempts FROM CloneClub;
REVOKE ALL PRIVILEGES ON CloneClub FROM CloneClub;
REVOKE ALL PRIVILEGES ON CASTOR FROM CloneClub;
REVOKE ALL PRIVILEGES ON Monitoring FROM CloneClub;
REVOKE ALL PRIVILEGES ON LEDA FROM CloneClub;
REVOKE ALL PRIVILEGES ON Clones FROM CloneClub;
REVOKE ALL PRIVILEGES ON People FROM CloneClub;
REVOKE ALL PRIVILEGES ON Families FROM CloneClub;
REVOKE ALL PRIVILEGES ON Zip FROM CloneClub;
```

```
GRANT SELECT, UPDATE, INSERT ON Jobs TO CloneClub;
GRANT SELECT, UPDATE, INSERT ON Occupations TO CloneClub;
GRANT SELECT, UPDATE, INSERT ON CombatSkills TO CloneClub;
GRANT SELECT, UPDATE, INSERT ON Skills TO CloneClub;
GRANT SELECT, INSERT ON MurderMethod TO CloneClub;
GRANT SELECT, INSERT ON MurderAttempts TO CloneClub;
GRANT SELECT, INSERT ON CloneClub TO CloneClub;
GRANT SELECT, INSERT ON CASTOR TO CloneClub;
GRANT SELECT, UPDATE, INSERT ON Monitoring TO CloneClub;
GRANT SELECT, UPDATE, INSERT ON LEDA TO CloneClub;
GRANT SELECT, UPDATE, INSERT ON Clones TO CloneClub;
GRANT SELECT, UPDATE, INSERT ON People TO CloneClub;
GRANT SELECT, UPDATE, INSERT ON Families TO CloneClub;
GRANT SELECT, INSERT ON Zip TO CloneClub;
```

DBAdmin Role

The DBAdmin role is for the Database Administrator(s). Most likely this job will be awarded to Cosima, Sarah or Alison in the event that the hired DBAdmin is absent. All permissions are granted to this role.

```
CREATE ROLE DBAdmin;
REVOKE ALL PRIVILEGES ON Jobs FROM DBAdmin;
REVOKE ALL PRIVILEGES ON Occupations FROM DBAdmin;
REVOKE ALL PRIVILEGES ON CombatSkills FROM DBAdmin;
REVOKE ALL PRIVILEGES ON Skills FROM DBAdmin;
REVOKE ALL PRIVILEGES ON MurderMethod FROM DBAdmin;
REVOKE ALL PRIVILEGES ON MurderAttempts FROM DBAdmin;
REVOKE ALL PRIVILEGES ON CloneClub FROM DBAdmin;
REVOKE ALL PRIVILEGES ON CASTOR FROM DBAdmin;
REVOKE ALL PRIVILEGES ON Monitoring FROM DBAdmin;
REVOKE ALL PRIVILEGES ON LEDA FROM DBAdmin;
REVOKE ALL PRIVILEGES ON Clones FROM DBAdmin;
REVOKE ALL PRIVILEGES ON People FROM DBAdmin;
REVOKE ALL PRIVILEGES ON Families FROM DBAdmin;
REVOKE ALL PRIVILEGES ON Zip FROM DBAdmin;
```

```
GRANT ALL PRIVILEGES ON Jobs TO DBAdmin;
GRANT ALL PRIVILEGES ON Occupations TO DBAdmin;
GRANT ALL PRIVILEGES ON CombatSkills TO DBAdmin;
GRANT ALL PRIVILEGES ON Skills TO DBAdmin;
GRANT ALL PRIVILEGES ON MurderMethod TO DBAdmin;
GRANT ALL PRIVILEGES ON MurderAttempts TO DBAdmin;
GRANT ALL PRIVILEGES ON CloneClub TO DBAdmin;
GRANT ALL PRIVILEGES ON CASTOR TO DBAdmin;
GRANT ALL PRIVILEGES ON Monitoring TO DBAdmin;
GRANT ALL PRIVILEGES ON LEDA TO DBAdmin;
GRANT ALL PRIVILEGES ON Clones TO DBAdmin;
GRANT ALL PRIVILEGES ON People TO DBAdmin;
GRANT ALL PRIVILEGES ON Families TO DBAdmin;
GRANT ALL PRIVILEGES ON Zip TO DBAdmin;
```

Implementation Notes

- Dyad and personnel should never be allowed to look at the database itself.
- Certain Select Queries such as one that brings up clone club members information would be easy to execute with a button and print out to a textbox in a mobile platform.
- (Optional) With cooperation from Dyad other LEDA clones and monitors can be found.
- Family IDs refer to legally related people. (i.e. Alison, Donnie, Gemma and Oscar) Biological relationships get too complicated for implementation to start off with.
- Be careful of who is given DBAdmin access as it is all permissions. Anyone with this access can completely DESTROY or PERVERT the data such that the database is unusable.

Know Problems

- CASTOR(logicTestingScore) is an int which assumes that the current score will always be out of the same NUMBER or that the testing system will not change.
- The Clones table has data that can become non-atomic if one were to use a CSV to express a clothing style for example.
- Family table is not fully representative of a person’s entire family
- CloneClub(clonePhoneNumber) assumes that clone phones will always have US numbers.
- Clones(Orgin) is a vague it is supposed to suggest cultural background and/or ethnicity. There is probably a better more informational way to represent this. Ethnicity wasnt used because ethnicity is usually more general than a specific place or region. Like hispanic is an ethnicity but Upper East Side New Yorker is suggestive of a specific culture.
- Jobs are not well specified as definced. 2 people can start on the same day in occupation soldier but Location.

Future Enhancements

- Make more entities to standardize the differences between each clone. Restructure the families entity so that it can give biological relationships instead of just legal family relationships.
- More entities to separate allies from enemies and be able to report that.
- Find a way to represent each clones synthetic sequence.
- Better represent each clones nurture element because that is what separates them as different people.