Ninja Database

By Graham Harrison

4/24/14

CMPT 308

Table of Contents

Executive Summary

Overview

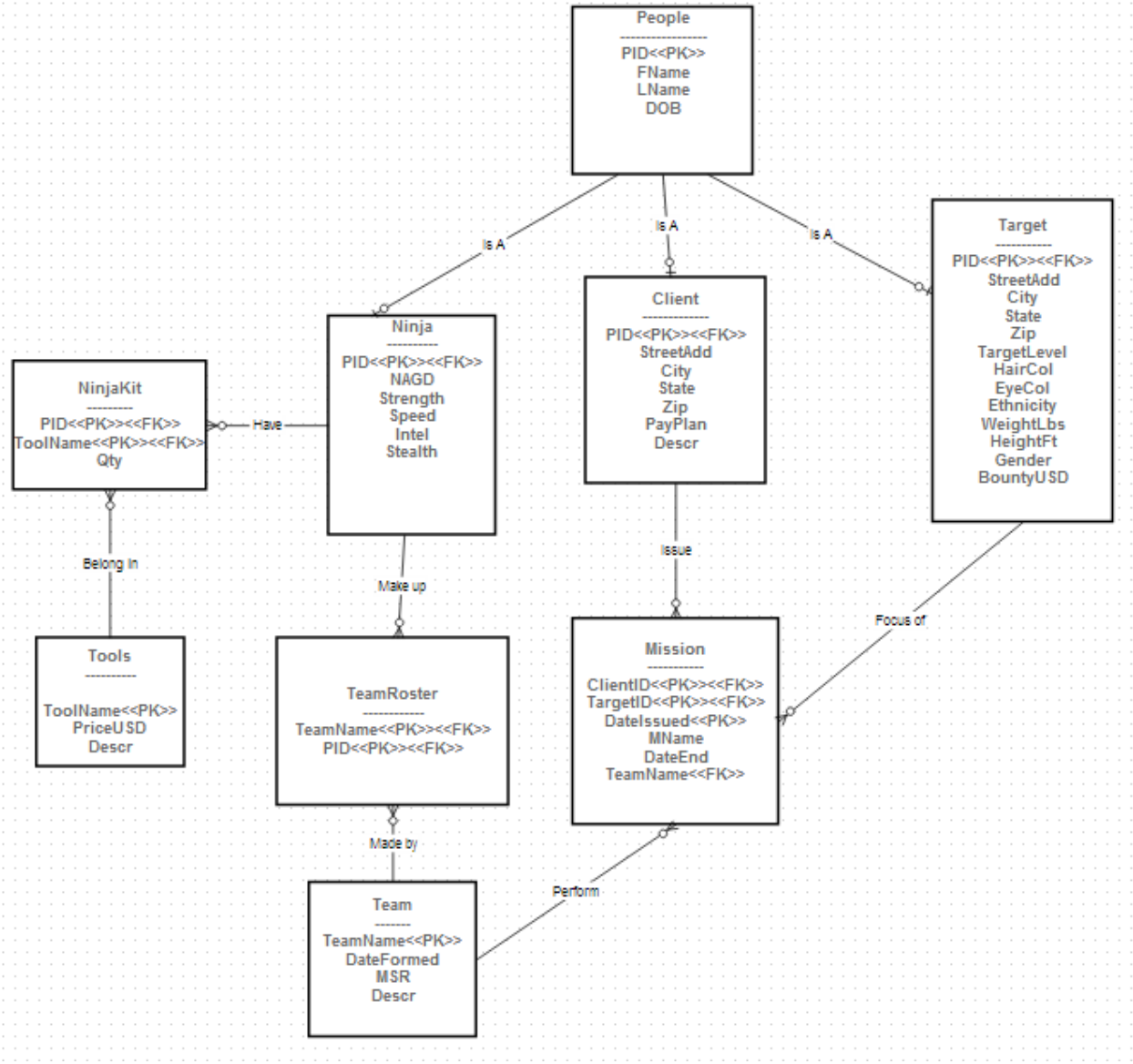
In the life of a ninja, every decision that is made could be the deciding factor between life and death. The information used to make these decisions are by far the ninja's greatest tool at his or her disposal. It is essential to record all relevant data concerning many aspects of the ninja world such as ninja teams, ninja equipment, and previous and ongoing missions in order to ensure that the ninja clan's recourses are being optimally used. In order to pick the right ninja for a mission or team, Ninja Clan Leaders must take the ninja's skills, equipment, and previous experience into account.

Objectives

The following document describes the Ninja database, whose main purpose is to record data about Ninjas, Clients, Targets and Tools. By keeping track of different ninja's skills and equipment, better ninja teams may be formed. Also, by keeping track of a ninja team's mission success rate, Ninja Clan Leaders can see which teams are working well and which are not. Maintaining detailed records about a target's location and physical description will also aid ninjas during their missions. Finally, by keeping track of missions past and ongoing, ninja clan leaders are able to easily see what is going on within their clan.

This document will describe the different database objects, views, reports, procedures, triggers, and security features that make up the Ninja database. It will also describe how the Ninja database can be implemented, any known problems, and will examine what features can be added in the future.

Entity Relationship Diagram



Tables

People

Purpose

The People table is a parent table for the Ninja, Client and Target tables. It stores first and last name information as well as the person's date of birth and a PID(person ID)

Create Statement

Create Table People (PID integer not null primary key,

FName text,

LName text,

DOB date);

Functional Dependencies

PID →FName, LName, DOB

Sample Data

Data Output | Explain | Messages | History

	pid integer	fname text	lname text	dob date
1	1	Graham	Harrisc	1994-04-13
2	2	Jon	Harrisc	1993-02-24
3	3	Bill	Guy	1990-12-25
4	4	Gus	Golly	1991-09-12
5	5	Carl	Green	1991-10-11
6	6	Ben	Blue	1991-11-12
7	7	Scott	Tan	1992-01-02
8	8	Stacy	Yellow	1992-03-03
9	9	Henry	Hall	1993-05-05
10	10	Phillip	Cane	1993-05-06
11	11	Greg	Ghastly	1989-02-14
12	12	Megan	Ken	1995-12-12
13	13	Billy	Bob	1995-09-13
14	14	Frank	Hall	1995-07-07
15	15	Lucy	Lu	1996-05-02

Ninja

Purpose

The Ninja table contains all the members of the People table that are ninjas. It records the ninja's graduation date from the ninja academy (NAGD) as well as their strength, speed, intelligence (Intel), and stealth skills which are rated numerically from 0-10.

Create Statement

```
Create Table Ninja(PID integer not null primary key references People(PID),
                   NAGD date,
                   Strength integer not null check(Strength>=0 or Strength<=10),
                   Speed integer not null  check(Speed>=0 or Speed<=10),
                   Intel integer not null check(Intel>=0 or Intel<=10),
                   Stealth integer not null check(Stealth>=0 or Strength<=10));
```

Functional Dependencies

PID → NAGD, Strength, Speed, Intel, Stealth

Sample Data

	pid integer	nagd date	strength integer	speed integer	intel integer	stealth integer
1	1	2012	4	7	10	8
2	2	2011	10	5	4	5
3	3	2011	5	5	5	5
4	4	2012	3	5	4	8
5	5	2009	7	6	4	8
6	6	2009	4	9	5	6
7	7	2009	4	8	0	2
8	8	2010	5	8	2	5
9	9	2010	10	10	10	10

Client

Purpose

The Client table contains all the People that are Clients for the ninja clan. The table contains information about the client's address, their payment plan(PayPlan) which is either Standard or Premium, and has a description (Descr).

Create Statement

Create Table Client(PID integer not null primary key references People(PID),

StreetAdd varchar(25),

City varchar(25),

State varchar(2),

Zip varchar(5),

PayPlan varchar(9) check(PayPlan='Standard' or PayPlan='Premium'),

Descr text);

Functional Dependencies

PID → StreetAdd, City, State, Zip, PayPlan, Descr

Sample Data

Data Output | Explain | Messages | History

	pid integer	streetadd character varying(25)	city character varying(25)	state character varying(2)	zip character varying(5)	payplan character varying(9)	descr text
1	10	139 Ellis Rd	Havertown	PA	19083	Standard	Phillip likes long walks on
2	11	3399 North Rd	Poughkeepsie	NY	12601	Standard	Greg likes Fishing
3	12	222 Dark Ln	SomeTown	CA	19583	Premium	Megan like to bowl

Target

Purpose

The Target table contains all past and present targets of the ninja clan, and is a child of the People table. The table contains the target's address, target level (which is either Low Medium or High), information about their physical description, a general description, and their bounty (BountyUSD).

Create Statement

```
Create Table Target(PID integer not null primary key references People(PID),
                    StreetAdd varchar(25),
                    City varchar(25),
                    State varchar(2),
                    Zip varchar(5),
                   TargetLevel text check(TargetLevel='Low' or TargetLevel='Medium'
                                          or TargetLevel='High'),
                    HairCol varchar(15),
                    EyeCol varchar(15),
                    Ethnicity varchar(15),
                    WeightLBS float(53),
                    HeightFt float (53),
                    Gender char check(Gender='M' or Gender='F'),
                    Descr text,
                    BountyUSD decimal(19,2));
```

Sample Data

Data Output | Explain | Messages | History

	pid integer	streetadd character varyi	city character va	state character	zip charact	targetlevel text	haircol charact	eyecol charact	ethnici charac	weightlb double pi	heightft double pr	gende charac	descr text	bountyusd numeric(19
1	13	345 Astreet	Acity	WA	19283	Low	Grey	Green	White	140.5	5.5	M	Billy works	500.00
2	14	3658 Fox Blv	Hopeville	PA	19039	Medium	Red	Blue	Asian	120	5	M	Frank runs	1000.00
3	15	77 West Rd	Easton	TX	93929	High	Black	Brown	White	120.5	5	F	Lucy has a	5000.00

Tools

Purpose

The Tools table contains data concerning the different types of tools used by ninjas, including the tool name, price, and a description of the tool.

Create Statement

Create Table Tools(ToolName varchar(25) not null primary key,

PriceUSD decimal(19,2),

Descr text);

Functional Dependencies

ToolName → PriceUSD, Descr

Sample Data

	toolname character varyi	priceusd numeric(1	descr text
1	Ninja Star	1.50	Throwable Sharp Projectile Medium Range
2	Sword	300.00	Long Ninja Blade
3	Knife	250.00	Short Ninja Blade
4	Nunchuck	50.99	Two pieces of wood connected by a chain
5	Smoke Bomb	3.00	Creates a cloud of smoke
6	Flash Bomb	3.00	Creates blinding flash of light
7	Bow	220.99	Device that shoots arrows long distances
8	Arrow	4.00	Sharp Projectile
9	Poison	100.00	Deadly Ninja Poison

NinjaKit

Purpose

The NinjaKit table associates ninjas with the equipment they possess. It contains data about which ninja has what equipment, as well as how much of it they have.

Create Statement

Create Table NinjaKit(PID integer not null references Ninja(PID),

ToolName varchar(25) not null references Tools(ToolName),

Qty integer check(Qty>=0),

Primary Key(PID, ToolName));

Functional Dependencies

PID, ToolName → QTY

Sample Data

	pid integer	toolname character varyi	qty integer
1	1	Ninja Star	20
2	1	Sword	1
3	2	Knife	1
4	2	Bow	1
5	2	Arrow	20
6	2	Smoke Bomb	5
7	3	Nunchuck	1
8	3	Poison	2
9	4	Sword	2
10	4	Flash Bomb	4
11	5	Knife	1
12	5	Ninja Star	15
13	6	Bow	1
14	6	Arrow	50
15	7	Sword	1
16	8	Nunchuck	1
17	9	Sword	1
18	9	Nunchuck	1
19	9	Ninja Star	10

Team

Purpose

The Team table contains information on all the ninja teams within the ninja clan, such as when the team was formed, the team's mission success rate (MSR), and a description (Descr) of the team.

Create Statement

Create Table Team(TeamName varchar(25) not null primary key,

Dateformed date,

MSR decimal(5,2) check(MSR>=0 or MSR<=1),

Descr text);

Functional Dependencies

TeamName → DateFormed, MSR, Descr

Sample Data

	teamname character varying(25)	dateformed date	msr numeric	descr text
1	Team One	2013-11-11	0.89	This is Team One
2	Team Two	2012-12-29	0.50	This is Team Two
3	Team Three	2013-09-09	0.85	This is team three
4	Super Team	2014-04-24	1.00	The Super Team

TeamRoster

Purpose

The TeamRoster table associates ninjas with the team or teams that they belong to.

Create Statement

Create Table TeamRoster(TeamName varchar(25) not null references Team(TeamName),

PID integer not null references Ninja(PID),

primary key (TeamName,PID));

Functional Dependencies

TeamName, PID →

Sample Data

Data Output | Explain | Messages | His

	teamname character varying(25)	pid integer
1	Team One	1
2	Team One	2
3	Team One	3
4	Team Two	4
5	Team Two	5
6	Team Two	6
7	Team Three	7
8	Team Three	8
9	Team Three	9
10	Super Team	9
11	Super Team	1
12	Super Team	2

Mission

Purpose

The Mission table records data concerning past and present missions of the ninja clan. The table stores the client who requests the mission, the target of the mission, a mission name (MName), the team assigned the mission and the start and end dates of the mission.

Create Statement

Create Table Mission(ClientID integer not null references Client(PID),

TargetID integer not null references Target(PID),

DateIssued date not null,

MName varchar(25),

DateEnd date,

TeamName varchar(25) references Team(TeamName),

primary key (ClientID, TargetID, DateIssued));

Functional Dependencies

ClientID, TargetID, DateIssued →MName, DateEnd, TeamName

Sample Data

	clientid integer	targetid integer	dateissued date	mname character varying(25)	dateend date	teamname character varying(25)
1	10	13	2013-05-0	Operation Blue Daisy	2013-05-10	Team One
2	11	14	2013-06-0	Operation Ivy	2013-06-07	Team Two
3	12	15	2013-07-0	Operation Endgame	2013-07-07	Team Three
4	12	15	2014-04-2	Operation Endgame part 2		Super Team

Views

NinjaDetails

Purpose

While a select * on the Ninja table outputs a result that only displays a ninja's PID, the NinjaDetails view grabs the Ninja's first and last names from the people table, concatenates the two names into a single field called NinjaName and displays the NinjaName instead of the PID. In addition, field names that have been abbreviated in order to ensure easier administration coding have been restored to their original form. For example, the NAGD field is renamed NinjaAcademyGraduationDate. The view also uses the KitCost stored procedure to sum up the cost of the ninja's tool kit.

Create Statement

```
Create View NinjaDetails as
  Select Concat(FName,' ',LName) as NinjaName,
           NAGD as NinjaAcademyGraduationDate,
           Strength,
           Speed,
           Intel as Intelligence,
          Stealth,
          Concat('$',KitCost(Ninja.pid)) as KitCost
  from People,
        Ninja
  Where People.PID=Ninja.PID
  order by NinjaName asc
```

Sample Output

Data Output | Explain | Messages | History

	ninjaname text	ninjaacademygraduationdate date	strength integer	speed integer	intelligence integer	stealth integer	kitcost text
1	Ben Blue	2009-10-10	4	9	5	6	$420.9
2	Bill Guy	2011-02-02	5	5	5	5	$250.9
3	Carl Green	2009-10-10	7	6	4	8	$272.5
4	Graham Harrison	2012-05-25	4	7	10	8	$330.0
5	Gus Golly	2012-05-25	3	5	4	8	$612.0
6	Henry Hall	2010-03-11	10	10	10	10	$365.9
7	Jon Harrison	2011-02-02	10	5	4	5	$565.9
8	Scott Tan	2009-10-10	4	8	0	2	$300.0
9	Stacy Yellow	2010-03-11	5	8	2	5	$50.99

ClientDetails

Purpose

Like the NinjaDetails view the ClientDetails view takes a client's first and last names and concatenates them into a single field called ClientName. In addition, the view concatenates address information and expands all abbreviations.

Create Statement

```
Create View ClientDetails as
  Select Concat(FName,' ',LName) as ClientName,
         Concat(StreetAdd,' ',City,' ',State,' ',Zip) as Address,
         PayPlan as PaymentPlan,
         Descr as Description
  from People,
      Client
  Where People.PID=Client.PID
  order by ClientName asc;
```

Sample Output

	clientname text	address text	paymentplar character va	description text
1	Greg Ghastly	3399 North Rd Poughkeepsie NY 12601	Standard	Greg likes Fishing
2	Megan Ken	222 Dark Ln SomeTown CA 19583	Premium	Megan like to bowl
3	Phillip Cane	139 Ellis Rd Havertown PA 19083	Standard	Phillip likes long walks

TargetDetails

Purpose

Like the previous two views, the TargetDetails view concatenates the target's first and last names. It also concatenates the address information into a field called address and adds a dollar sign to the bounty field.

Create Statement

```
Create View TargetDetails as
  Select Concat(FName,' ',LName) as TargetName,
         Concat(StreetAdd,' ',City,' ',State,' ',Zip) as Address,
         TargetLevel,
         HairCol as HairColor,
         EyeCol as EyeColor,
         Ethnicity,
         WeightLbs,
         HeightFt,
         Gender,
         Concat('$',BountyUSD) as Bounty
  from People,
      Target
  Where People.PID=Target.PID
  order by TargetName;
```

Sample Output

	targetname text	address text	targetle text	haircol charact	eyecol charac	ethnicit charact	weightl double	height double	gen cha	bounty text
1	Billy Bob	345 Astreet Acity WA 19283	Low	Grey	Green	White	140.5	5.5	M	$500.00
2	Frank Hall	3658 Fox Blv Hopeville PA 19039	Medium	Red	Blue	Asian	120	5	M	$1000.00
3	Lucy Lu	77 West Rd Easton TX 93929	High	Black	Brown	White	120.5	5	F	$5000.00

Equipment

Purpose

The Equipment view concatenates the first and last names of a ninja and displays the equipment they have.

Create Statement

```
Create View Equipment as
  Select Concat(FName,' ',LName) as NinjaName,
         ToolName,
         Qty
  from People,
       NinjaKit
  where People.pid=NinjaKit.PID
  order by NinjaName asc
```

Sample Output

	ninjaname text	toolname character varying(25)	qty integer
1	Ben Blue	Arrow	50
2	Ben Blue	Bow	1
3	Bill Guy	Nunchuck	1
4	Bill Guy	Poison	2
5	Carl Green	Ninja Star	15
6	Carl Green	Knife	1
7	Graham Harrison	Sword	1
8	Graham Harrison	Ninja Star	20
9	Gus Golly	Flash Bomb	4
10	Gus Golly	Sword	2
11	Henry Hall	Ninja Star	10
12	Henry Hall	Nunchuck	1
13	Henry Hall	Sword	1
14	Jon Harrison	Knife	1
15	Jon Harrison	Smoke Bomb	5
16	Jon Harrison	Arrow	20
17	Jon Harrison	Bow	1
18	Scott Tan	Sword	1
19	Stacy Yellow	Nunchuck	1

TeamMembers

Purpose

The TeamMembers view replaces the PID with a concatenation of the ninja's first and last names.

Create Statement

```
Create View TeamMambers as
    Select TeamName,
            Concat(FName,' ',LName) as NinjaName
    from People,
         TeamRoster
    where People.PID=TeamRoster.PID
    order by Teamname asc
```

Sample Output

	teamname character varying(25)	ninjaname text
1	Super Team	Jon Harrison
2	Super Team	Graham Harrison
3	Super Team	Henry Hall
4	Team One	Graham Harrison
5	Team One	Jon Harrison
6	Team One	Bill Guy
7	Team Three	Scott Tan
8	Team Three	Stacy Yellow
9	Team Three	Henry Hall
10	Team Two	Ben Blue
11	Team Two	Carl Green
12	Team Two	Gus Golly

MissionDetails

Purpose

The MissionDetails view replaces both the ClientID and the TargetID with their associated first and last names as concatenated strings. It also extends field name abbreviations.

Create Statement

```
Create View MissionDetails as
  Select Concat(c.FName,' ',c.LName) as ClientName,
         Concat(t.FName,' ',t.LName) as TargetName,
         DateIssued,
         MName as MissionName,
         DateEnd,
        TeamName
  from People c,
       People t,
      Mission
  Where c.PID=Mission.ClientID
    and t.PID=Mission.TargetID
  order by DateIssued asc
```

Sample Data

	clientname text	targetname text	dateissued date	missionname character varying(25)	dateend date	teamname character varying(25)
1	Phillip Cane	Billy Bob	2013-05-0	Operation Blue Daisy	2013-05	Team One
2	Greg Ghastly	Frank Hall	2013-06-0	Operation Ivy	2013-06	Team Two
3	Megan Ken	Lucy Lu	2013-07-0	Operation Endgame	2013-07	Team Three
4	Megan Ken	Lucy Lu	2014-04-2	Operation Endgame part 2		Super Team

Reports

Ninja Rank

Purpose

A Ninja's skills are broken up into four separate categories: strength, speed, intelligence and stealth. Because of this, it is hard to determine the overall ability of a ninja when compared to other ninjas at first glance. The NinjaRank report eliminates this problem by averaging a ninja's four skill points into an AverageScore field and ranks the ninjas by that score in descending order.

Query

```
Select FName, LName, (Strength+Speed+Intel+Stealth)/4 as AverageScore
from People,
        Ninja
Where People.PID=Ninja.PID
order by AverageScore desc
```

Sample Output

	fname text	lname text	averagescore integer
1	Henry	Hall	10
2	Grahar	Harrison	7
3	Ben	Blue	6
4	Carl	Green	6
5	Jon	Harrison	6
6	Bill	Guy	5
7	Gus	Golly	5
8	Stacy	Yellow	5
9	Scott	Tan	3

Ranged Assassins

Purpose

A Ninja's target might not always be within arm's reach. Knowing which ninjas possess ranged weapons during pre-mission planning is essential to forming a ninja team that will successfully eliminate the target. The following view displays all ninjas that possess ranged weapons such as a bow or ninja stars.

Query

```
Select FName, LName, ToolName
from People,
        Ninja,
        NinjaKit
Where People.PID=Ninja.PID
    and Ninja.PID=NinjaKit.PID
    and (NinjaKit.ToolName='Bow'
            or NinjaKit.ToolName='Ninja Star')
order by ToolName asc
```

Sample Output

	fname text	lname text	averagescore integer
1	Henry	Hall	10
2	Graham	Harrison	7
3	Ben	Blue	6
4	Carl	Green	6
5	Jon	Harrison	6
6	Bill	Guy	5
7	Gus	Golly	5
8	Stacy	Yellow	5
9	Scott	Tan	3

Ninjas currently on a mission

Purpose

A ninja clan leader must always know which ninja are currently out in the field on a mission and which ninja are ready to be given a new assignment. Knowing who is currently available and who is not is essential to managing ninja. The following query is designed to return all ninja that are currently out on a mission and unavailable for any additional work.

Query

```
Select FName, LName, MName,Mission.TeamName, DateIssued

from People,

    Ninja,

    TeamRoster,

    Mission

Where People.PID=Ninja.PID

  and TeamRoster.PID=Ninja.PID

  and TeamRoster.TeamName=Mission.TeamName

  and Mission.DateEnd is Null

order by DateIssued Desc
```

Sample Output

Data Output | Explain | Messages | History

	fname text	lname text	mname character varying(25)	teamname character varying(25)	dateissued date
1	Henry	Hall	Operation Endgame part 2	Super Team	2014-04-2
2	Graham	Harrison	Operation Endgame part 2	Super Team	2014-04-2
3	Jon	Harrison	Operation Endgame part 2	Super Team	2014-04-2

Ninjas with the Most Experience

Purpose

When choosing ninja for missions of great difficulty, it helps for ninja clan leaders to prioritize their ninja based on two characteristics: the first being the amount of time a ninja has been a ninja, and the second being previous experience pursuing high class targets. The following query returns all ninja that have previously hunted high value targets and orders them by time since their Ninja Academy Graduation Date in ascending order.

Query

```
Select distinct FName, LName, NAGD

From People,

    Ninja,

    Target,

    Mission,

    TeamRoster

Where People.PID=Ninja.PID

  and Ninja.PID=TeamRoster.PID

  and TeamRoster.TeamName=Mission.TeamName

  and Target.PID=Mission.TargetID

  and Target.TargetLevel='High'

  order by NAGD asc
```

Sample Output

	fname text	lname text	nagd date
1	Scott	Tan	2009-10-10
2	Henry	Hall	2010-03-11
3	Stacy	Yellow	2010-03-11
4	Jon	Harrison	2011-02-02
5	Graham	Harrison	2012-05-25

Stored Procedures

KitCost(int)

Purpose

The KitCost function takes in an integer representing a ninja's PID, and returns the collective cost of his entire Ninja Kit in decimal format(decimal(19,2)). The function takes into account not only the type of items in a ninja's kit, but the quantity of those items as well. Specifically, the function is used in the NinjaDetails view.

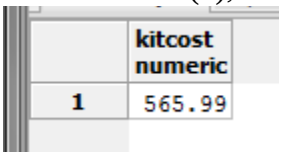
Query

```
Create or replace function kitCost(int) returns Decimal(19,2) as
$$
declare
  NinjaID int :=$1;
  Kitcost decimal(19,2);
begin
  KitCost=Sum(Tools.PriceUSD*NinjaKit.Qty)
  from Tools,
      NinjaKit,
      People,
      Ninja
  where people.pid=NinjaID
    and people.pid=ninja.pid
    and ninja.pid=NinjaKit.pid
    and tools.toolname=ninjakit.toolname;

  return kitcost;
end;
$$
language plpgsql;
```

Sample Output

Select KitCost(2);→



	kitcost numeric
1	565.99

NinjaTeamMates(int)

Purpose

The NinjaTeamMates function takes an integer representing a ninja's PID and returns all of his or her team mates in the form of a table. This function is useful for instantaneously knowing who a ninja is working or has worked with in the past, and could help aid in better team making down the road.

Query

```
Create or replace function NinjaTeamMates(int) returns table(NName text, TName varchar(25))
as
$$
declare
  NinjaID int :=$1;
begin
  return query
    Select distinct concat(p.FName,' ',p.LName) as NName,
                    t2.TeamName
    from people,
         people p,
         TeamRoster t1,
         TeamRoster t2,
         Team
     where T1.Teamname=Team.teamname
       and t2.teamname=team.teamname
       and t2.teamname=team.teamname
       and t1.teamname=t2.teamname
       and t2.pid!=NinjaID
       and t1.pid=NinjaID
       and p.pid=t2.pid
     order by t2.TeamName;

end;
$$
language plpgsql;
```

Sample Output

Select NinjaTeamMates(1);→

	record
1	("Henry Hall","Super Team")
2	("Jon Harrison","Super Team")
3	("Bill Guy","Team One")
4	("Jon Harrison","Team One")

Trigger Functions

PeopleCheck()

Purpose

This function ensures no duplicate data is being inserted into the people table.

Query

```
Create function PeopleCheck() returns trigger as
$PeopleCheck$
begin
  if exists (select pid
          from people
          where Fname=new.FName
             and LName=new.LName
             and DOB=new.DOB)
  then
    raise exception 'Person with same info already exists in the database';
  end if;
  return new;
  end;
  $PeopleCheck$
  language plpgsql;
```

NinjaCheck()

Purpose

This function ensures no duplicate data is inserted into the Ninja table

Query

```
Create Function NinjaCheck() returns trigger as
$NinjaCheck$
begin
  if exists(Select PID
          from Ninja
          Where NAGD=new.NAGD
            and Strength=new.Strength
            and Speed=new.Speed
            and Intel=new.Intel
            and Stealth=new.Stealth)
  Then
    Raise Exception 'Ninja already exists';
  end if;
  return new;
end;
$NinjaCheck$
language plpgsql;
```

ClientCheck()

Purpose

This function ensures no duplicate data is entered into the Client table.

Query

```
Create function ClientCheck() returns trigger as
$ClientCheck$
begin
  if exists(Select PID
          from client
          where StreetAdd=new.StreetAdd
             and City=new.City
             and State=new.State
             and Zip=new.Zip
             and PayPlan=new.PayPlan
             and Descr=new.Descr)
  then
    Raise exception 'Client already exists';
  end if;
  return new;
end;
$ClientCheck$
language plpgsql;
```

TargetCheck()

Purpose

This function ensures no duplicate data is entered into the Targets table.

Query

```
Create Function TargetCheck() returns trigger as
$TargetCheck$
begin
  if exists(Select PID
          from target
          where StreetAdd=new.StreetAdd
            and City=new.City
            and State=new.State
            and Zip=new.Zip
            and targetlevel=new.targetlevel
            and haircol=new.haircol
            and eyecol=new.eyecol
            and weightlbs=new.weightlbs
            and heightft=new.heightft
            and gender=new.gender
            and bountyusd=new.bountyusd)
  then
```

```
    raise exception 'target already exists';
  end if;
  return new;
end;
$TargetCheck$
language plpgsql;
```

MissionCheck()

Purpose

This function ensures no duplicate data is entered into the Mission table.

Query

```
Create Function MissionCheck() returns trigger as
$MissionCheck$
begin
  if exists(Select ClientID, TargetID, DateIssued
          from mission
          where MName=new.MName
             and DateEnd=new.DateEnd
             and TeamName=new.TeamName)
   then
     Raise exception 'Mission with same information already exists';
   end if;
   return new;
end;
$MissionCheck$
 language plpgsql;
```

Triggers

```
--Insert people
Create Trigger PeopleCheck before insert on People
for each row execute procedure PeopleCheck();

--Insert Ninja
Create Trigger NinjaCheck before insert on Ninja
for each row execute procedure NinjaCheck();

--Insert Client
Create Trigger ClientCheck before insert on Client
for each row execute procedure ClientCheck();
--Insert Target
Create trigger TargetCheck before insert on Target
for each row execute procedure TargetCheck();
--Insert Mission
Create Trigger MissionCheck before insert on Mission
for each row execute procedure MissionCheck();
```

Security

The Ninja database is used by three kinds of users: Administrators, ClanLeaders and ANinjas.

Administrator

The Administrator has complete control over the database.

Grant all Privileges on all tables in schema public to Administrator;

ClanLeader

The ClanLeader has the ability to select, insert into and update all tables.

Grant Select, insert update on all tables in schema public to ClanLeader;

ANinja

ANinja only has the power to select all tables.

Grant select on all tables in schema public to ANinja;

Implementation Notes

- A mission's DateEnd field is filled not when the mission is a success, but when it is over. The mission could either be a success or a failure.
- A ninja's kit must be updated at the end of every mission in order to ensure the accuracy of information, especially if disposable items where used during the mission.
- Ninjas who are KIA must be removed from the system in order to maintain an accurate picture of the ninjas at the Clan Leaders disposal.

Known Problems

- Triggers are missing for a few tables such as the Team table and the Tool table.
- For the security queries, an error pops up indicating that the roles do not exist.
- Triggers do not check for field constrains like the ninja skill level's range of 0-10

Future Enhancements

- Add table that records a missions outcome, whether it was a success or failure.
- Provide age calculation for people
- Provide triggers for the rest of the tables
- Add a field that records how much bounty money a ninja has earned.
- Provide a more accurate Ninja experience field in the NinjaDetails view by finding the difference between the current date and the ninja's Ninja Academy Graduation Date.
- Add more tools to the Tool table
- Modify the database to record a team's ninja leader.