MARIST
Residential Networking

A Database System Proposal for the Residential Networking (ResNet) Department at Marist College in Poughkeepsie, New York

Benjamin C. DelGiorno • CMPT 308L • April 25, 2014

TABLE OF CONTENTS

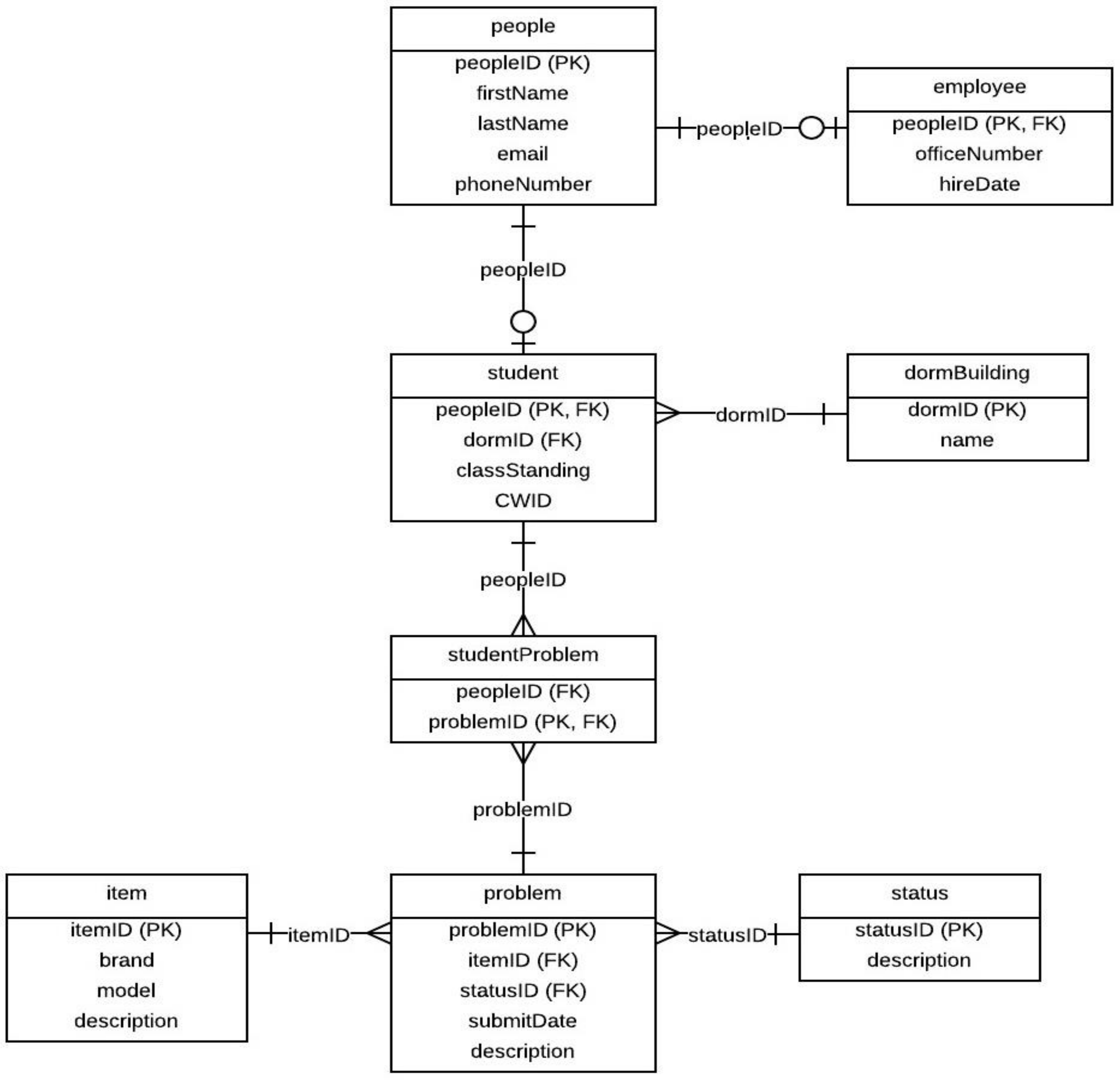
EXECUTIVE SUMMARY

The Residential Networking (ResNet) Department at Marist College serves a vital role in assisting all members of the student body across campus in successfully connecting to the Marist network. The Department also assists students with malware removal as well as general PC support. This document provides an overview of a comprehensive database management system designed specifically for the ResNet department. The database takes into account all aspects of the department as well as the students that it caters to.

The database is comprised of several unique tables including employees, students, and problems, all of which are addressed and thoroughly illustrated in this document. A detailed entity relationship diagram is included in this document along with examples of views, stored procedures, and triggers. Other major topics of importance addressed in this document include security features, reports, implementation notes, and potential future enhancements to the database. Through the implementation of this database, the Residential Networking Department at Marist College will be able to conduct its work on campus in a much more effective and more efficient manner.

The database will allow employees to track problems that students have and enable them to see the status of those problems as they move through the repair process. It will also allow employees to view past problems that students have experienced and also prioritize student problems based on the date they were submitted. This database system will allow students to report their problems to ResNet in a convenient and expedient way, allowing technical issues to be remedied soon after they are encountered. The database will allow for easier communication between students and employees at ResNet, drastically improve efficiency, and revolutionize the Residential Networking student network support experience. This database is sophisticated, yet simple to comprehend and interpret and it could most certainly support the nearly 4,500 undergraduate students on the Marist College campus in Poughkeepsie, New York. This database was designed and developed using PostgreSQL Version 1.12.

ENTITY RELATIONSHIP DIAGRAM



TABLES

People Table

The people table holds all basic information about the people in this database which include either employees or students. The employee and student tables both contain more specific information about the people that are in them.

```
create table if not exists people (
    peopleID    int  not null,
    firstName   text not null,
    lastName    text not null,
    email       text not null,
    phoneNumber text not null,
  primary key(peopleID)
);
```

Functional Dependencies

peopleID —› firstName, lastName, email, phoneNumber

Sample Data

peopleID	firstName	lastName	email	phoneNumber
1	Benjamin	DelGiorno	Benjamin.delgiorno1@marist.edu	570-428-3901
2	Bill	Gates	William.Gates@marist.edu	901-230-6785
3	Ted	Codd	Edgar.Codd@marist.edu	618-458-1984
4	Alan	Labouseur	Alan.Labouseur@marist.edu	845-898-7345
5	Robert	Smith	Robert.Smith1@marist.edu	714-398-1765
6	Matt	Jones	Matthew.Jones1@marist.edu	391-470-1776
7	Albert	Einstein	Albert.Einstein@marist.edu	856-597-1841
8	Jack	Walsh	John.Walsh1@marist.edu	609-298-1493
9	James	Bond	James.Bond1@marist.edu	215-968-3407
10	Ellen	Hancock	Ellen.Hancock@marist.edu	501-637-2146
11	Barack	Obama	Barack.Obama@marist.edu	845-570-3270
12	Dennis	Murray	Dennis.Murray@marist.edu	914-771-3663
13	Jane	Doe	Jane.Doe1@marist.edu	641-708-9461
14	Nelly	Goletti	Nelly.Goletti1@marist.edu	717-640-3197

TABLES

Employee Table

This table contains information pertinent to employees including the office number of each employee and the date they were hired to work in the ResNet Department.

```
create table if not exists employee (
    peopleID     int  not null,
    officeNumber text not null,
    hireDate     date not null default current_timestamp,
  primary key(peopleID),
  foreign key(peopleID) references people(peopleID)
);
```

Functional Dependencies

peopleID —› officeNumber, yearsWorked

Sample Data

peopleID	officeNumber	hireDate
2	DN101	1999-05-22
3	DN107	2003-02-15
4	DN103	2007-08-10
7	DN105	1993-09-21
10	DN111	2009-03-12
11	DN106	2012-01-04
12	DN109	2013-08-26

TABLES

Student Table

This table contains more specific information about students including their current class standing as well as the dorm in which they currently live.

```
create table if not exists student (
    peopleID        int  not null,
    dormID          int  not null,
    classStanding   text not null,
    CWID            text not null,
  primary key(peopleID) references people(peopleID),
  foreign key(dormID) references dormBuilding(dormID)
);
```

Functional Dependencies

peopleID ⟶ dormID,classStanding, CWID

Sample Data

peopleID	dormID	classStanding	CWID
1	05	Sophomore	200-34-624
5	01	Freshman	200-18-478
6	06	Sophomore	200-14-735
8	02	Junior	200-22-124
9	04	Freshman	200-71-740
13	06	Sophomore	200-28-641
14	05	Sophomore	200-36-426

TABLES

DormBuilding Table

This table contains the dorm identification numbers for each residence hall as well as the name of the residence hall.

```
create table if not exists dormBuilding (
    dormID   int  not null,
    name     text not null,
  primary key(dormID);
);
```

Functional Dependencies

dormID ⟶ name

Sample Data

dormID	name
01	Leo Hall
02	Shehan Hall
03	Marian Hall
04	Champagnat Hall
05	Midrise Hall
06	Gartland Commons

TABLES

StudentProblem Table

This table conveniently links students with the problems they have encountered. It should also be noted that this table contains no functional dependencies.

```
create table if not exists studentProblem (
    peopleID    int not null,
    problemID   int not null,
    primary key(problemID),
    foreign key(problemID) references problem(problemID),
    foreign key(peopleID)  references people(peopleID)
);
```

Functional Dependencies

None

Sample Data

peopleID	problemID
1	1
5	2
6	3
8	4
9	5
13	6
14	7

TABLES

Problem Table

```
create table if not exists problem (
    problemID     int not null,
    itemID        int not null,
    statusID      int not null,
    submitDate    date not null default current_timestamp,
    description   text not null,
  primary key(problemID),
  foreign key(itemID)    references item(itemID),
  foreign key (statusID) references status(statusID)
);
```

Functional Dependencies

problemID ⟶ itemID, statusID, submitDate, description

Sample Data

problemID	itemID	statusID	submitDate	description
1	12699	2	2014-04-16	Cannot connect to the Network
2	67898	1	2014-04-20	Virus on computer
3	56347	2	2014-04-22	Ethernet connectivity difficulties
4	13452	3	2014-04-12	Cannot connect to the Network via Wi-Fi
5	78654	1	2014-04-18	Cisco software issues
6	25761	1	2014-04-24	Spyware on computer
7	13858	2	2014-04-19	Cannot connect to the Network

TABLES

Item Table

This table contains all of the data pertaining to student's technological devices that are experiencing problems. This includes an item identification number, the brand, model, and a description of each item.

```
create table if not exists item (
    itemID      int  not null,
    brand       text not null,
    model       text not null,
    description text not null,
  primary key(itemID)
);
```

Functional Dependencies

itemID —› brand, model, description

Sample Data

itemID	brand	model	description
12699	Apple	iMac	Desktop Computer
67898	Toshiba	Satellite C55D	Laptop Computer
56347	Sony	VAIO T Series	Laptop Computer
13452	Lenovo	ThinkPad	Laptop Computer
78654	Asus	X551CA	Laptop Computer
25761	HP	Pavilion	Laptop Computer
13858	Apple	iPad 2	Tablet Computer

TABLES

Status Table

This table contains information regarding the current status of an item represented by a status identification number and a corresponding status description.

```
create table if not exists status (
    statusID    int not null,
    description text not null
  primary key(statusID)
);
```

Functional Dependencies

statusID ⟶ description

Sample Data

statusID	description
1	Received
2	In Progress
3	Complete

VIEW DEFINITIONS

ProblemsReceived

This view conveniently displays all of the problems that have been received by the ResNet Department, but have not yet been addressed by an employee. This view can assist the department in being aware of what problems need to be remedied and can help with planning and efficient scheduling to solve each problem in an expedient manner.

```
create view ReceivedProblems as
select p.submitDate  as "Date",
        p.problemID   as "Problem ID",
        p.itemID      as "Item ID",
        p.description as "Problem Description"
from problem p, item i, status s
where p.itemID   = i.itemID
  and p.statusID = s.statusID
  and p.statusID = 1;
```

Sample Output

Date	Problem ID	Item ID	Problem Description
2014-04-18	5	78654	Cisco software issues
2014-04-20	2	67898	Virus on computer
2014-04-24	6	25761	Spyware on computer

VIEW DEFINITIONS

ProblemsInProgress

This view conveniently displays all of the problems that are currently in the process of being fixed by an employee in the ResNet Department. This view can assist the department in being aware of which problems are currently being addressed and can help employees to keep track of each problem they are responsible for.

```
create view InProgressProblems as
select p.submitDate  as "Date",
        p.problemID   as "Problem ID",
        p.itemID      as "Item ID",
        p.description as "Problem Description"
from problem p, item i, status s
where p.itemID   = i.itemID
  and p.statusID = s.statusID
  and p.statusID = 2;
```

Sample Output

Date	Problem ID	Item ID	Problem Description
2014-04-16	1	12699	Cannot connect to the Network
2014-04-19	7	13858	Cannot connect to the Network
2014-04-22	3	56347	Ethernet connectivity difficulties

REPORTS

Show all Employees

This query will illustrate all current employees that work in the ResNet department.

```
select p.firstName, p.lastName, p.email, p.phoneNumber
from people p
where p.peopleID in (
    select peopleID
    from employee
);
```

Sample Output

firstName	lastName	email	phoneNumber
Bill	Gates	William.Gates@marist.edu	901-230-6785
Ted	Codd	Edgar.Codd@marist.edu	618-458-1984
Alan	Labouseur	Alan.Labouseur@marist.edu	845-898-7345
Albert	Einstein	Albert.Einstein@marist.edu	856-597-1841
Ellen	Hancock	Ellen.Hancock@marist.edu	501-637-2146
Barack	Obama	Barack.Obama@marist.edu	845-570-3270
Dennis	Murray	Dennis.Murray@marist.edu	914-771-3663

REPORTS

Show names of students with problems living in Gartland Commons

This query will show the names of students currently living in Gartland Commons that are experiencing problems.

```
select firstName, lastName
from people
where problem.problemID = studentProblem.problemID
  and studentProblem.peopleID = student.peopleID
  and student.dormID = dormBuilding.dormID
  and dormBuilding.dormID = 06;
```

Sample Output

firstName	lastName
Matt	Jones
Jane	Doe

STORED PROCEDURES

Add_People Stored Procedure

This stored procedures allows additional individuals to be added to the database. This stored procedure would enable the database administrator to input new data into the people table. This would be extremely useful in a case when new employees are hired in the department and also when a new class of incoming freshman students is enrolled at Marist.

```
create or replace function add_people("firstName" text,
"lastNname" text,
"email" text, "phoneNumber" text)
 RETURNS void AS
$BODY$ begin
  insert into people values (firstName, lastName, email,
phoneNumber);
end$BODY$
 language plpgsql;
```

Sample Input

```
insert into people (firstName, lastName, email, phoneNumber)
values
('Nick', 'Howard', 'Nick.Howard1@marist.edu', '215-340-1788')
```

TRIGGER

Delete Resolved Problem Trigger

This trigger will delete problems from the database when they have been effectively resolved by the ResNet department.

```
create function delResolvedProblem()
returns trigger AS $$
begin
   delete from problems
   where problem.problemID not exists(
       select problem.problemID
       from problem
   )
return null;
end;
$$ language plpgsql;

create trigger delResolvedProblem
    after delete on problem
    for each row
    execute procedure delResolvedProblem();
```

Result

When a problem is fully resolved, this trigger will delete the problem from the database when it is executed.

Admin Role

This role is comprised of all upper level managers who have unrestricted access of the database. They are able to execute select, update, delete and insert commands in the database as they may be needed.

```
create role admin;

revoke all privileges on people from admin;
revoke all privileges on employee from admin;
revoke all privileges on student from admin;
revoke all privileges on dormBuilding from admin;
revoke all privileges on studentProblem from admin;
revoke all privileges on problem from admin;
revoke all privileges on item from admin;
revoke all privileges on status from admin;

grant select,insert,update on people to admin;
grant select,insert,update,delete on employee to admin;
grant select,insert,update,delete on student to admin;
grant select,insert,update on dormBuilding to admin;
grant select,insert,update on studentProblem to admin;
grant select,insert,update on problem to admin;
grant select,insert,update on item to admin;
grant select,insert,update on status to admin;
```

SECURITY

Student Role

This role is comprised of all students at Marist College who have restricted access to the database. They are able to do far less in the database compared to the admin role as they should not have a significant amount of access to all of the information.

```
create role student;

revoke all privileges on people from student;
revoke all privileges on employee from student;
revoke all privileges on student from student;
revoke all privileges on dormBuilding from student;
revoke all privileges on studentProblem from student;
revoke all privileges on problem from student;
revoke all privileges on item from student;
revoke all privileges on status from student;

grant select on people to student;
grant select on student to student;
grant select on dormBuilding to student;
grant select on studentProblem to student;
grant select,insert on problem to student;
grant select on item to student;
grant select on status to student;
```

IMPLEMENTATION NOTES

This database would be fairly easy to implement in the ResNet Department at Marist College. Each of the tables in the database are interconnected and easy to understand while maintaining referential integrity. It would take some time to input all of the student and employee data into the system, however, once the system is implemented, the benefits that it will provide far outweigh any costs. More views could be implemented to find unique trends and select information in the database. This database will be incredibly helpful in increasing efficiency in the department and will help to make the problem solving process easier and more convenient for students.

KNOWN PROBLEMS

One of the current main problems with this database is the fact that some of the employees are actually students. This databases strictly separates people into either employees or students which is good for simplicity purposes but it's not technically a completely accurate representation of all the people in the database. Another problem is the lack of a component that updates the status of a problem being handled by ResNet. It would be convenient for the status to be automatically updated as the process is carried out. In addition, there is no way for ResNet to track the previous problems that any given student may have experienced. Previous problems that students experienced may be helpful to employees when resolving problems in the future for a particular student. Another problem with this database is that it does not address which employees are handling which particular student problem. Another potential problem with the database is regarding students that have graduated. There should be a feature built into the database that automatically deletes students after they have graduated from Marist. There should also be a feature that generates unique ID numbers for people, problems, and items to make the database more uniform and ensure that there is no duplicate data.

FUTURE ENHANCEMENTS

If more time was allotted, this database could be more intricate and include an increased number of tables and complexity. One component that would be added to the database would be a pastProblems table which would include an archive of all of the problems that each student has experienced during their time at Marist. Another component that could be added to the database is a table that holds information regarding all of the devices that each student uses on campus which they could register online with the ResNet Department. There could also be a feature that alerts students via email or phone when their computer is ready to be picked up. Future enhancements could also include more information about students and employees as well as more detailed information about the types of problems that students are experiencing. Another potential enhancement would be to link this database with other databases on the Marist College campus for the purpose of sharing similar data amongst the databases. There is significant room for growth and enhancement in this database and several improvements could be made if more time and resources were available.