A Database Design Proposal for Marist College ResNet

By Gabrielle Romano

December 2nd 2013

Table of Contents

The Goal of this database proposal is to support Marist College Residential Networking computer services operations. Currently ResNet is supporting 6,000 combined students and faculty; this database documents the employee's actions and customer's devices that have had technical difficulties and the solutions used to fix the problems.

The first section of this proposal displays the layout of the database design and the relationships between each of the sections within the networking department.

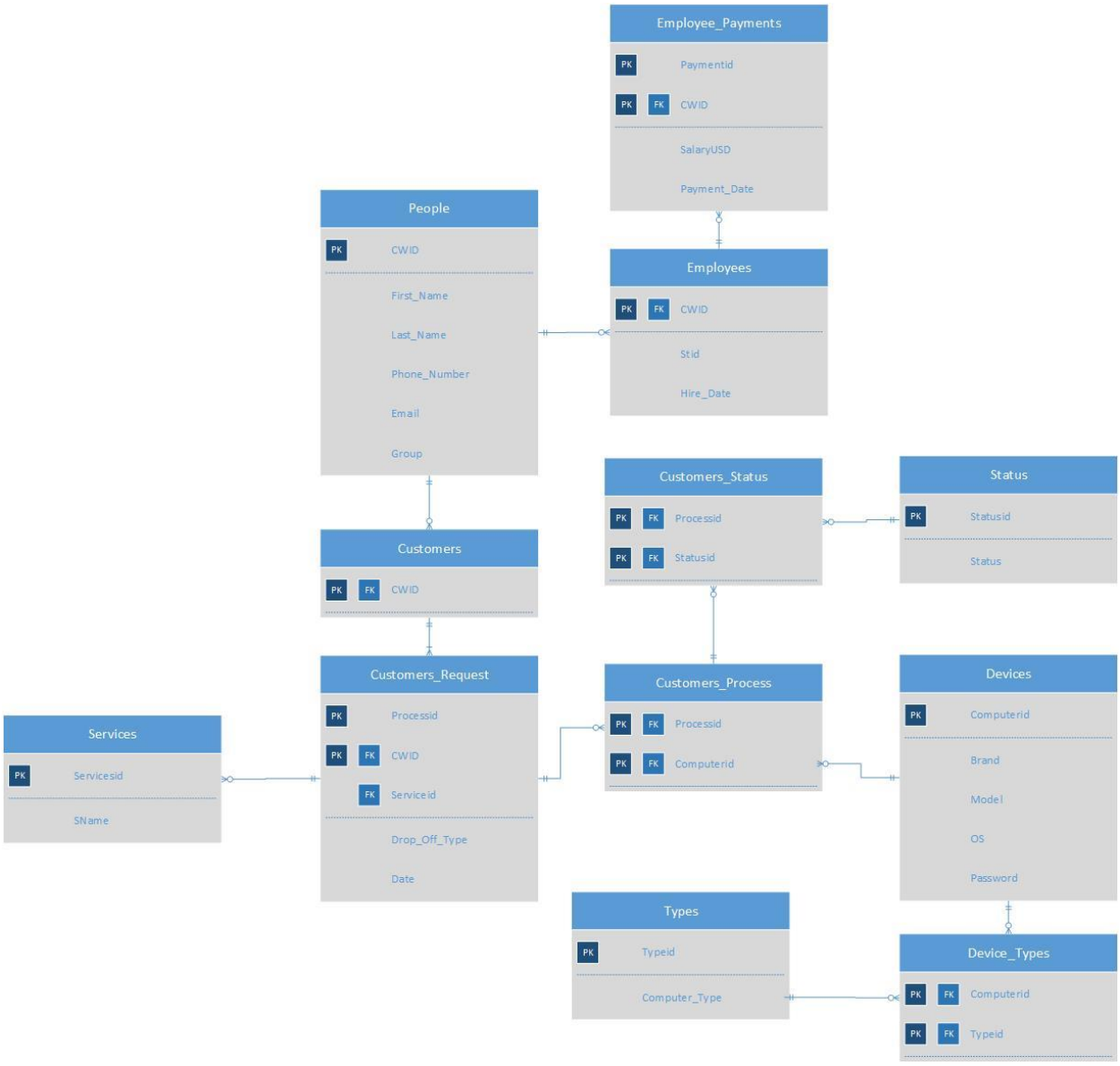
The following section of the proposal displays and describes an in-depth description of each table within the layout of the database. This includes a description of each table, its functional dependencies, create statements and sample data.

The third section of this proposal shows examples of reports and views aptitudes of the database.

The fourth section documents the possible triggers and stored procedures that can be implemented within the database.

The Last segment will provide a final analysis of the project as well as some ideas to consider as we look towards the future.

Entity Relationship Diagram



Create Statements

People *table:*

Employees and customers can be people and will inherit a person's traits from the People table.

Functional Dependencies:

CWID → First_Name, Last_Name, Phone_Number, Email, Group

Create Statement:

```
CREATE TABLE People (
        CWID            INT             NOT NULL    UNIQUE      PRIMARY KEY,
        First_Name      VARCHAR(25)     NOT NULL,
        Last_Name       VARCHAR (25)    NOT NULL,
        Phone_Number    INT             NOT NULL,
        Email           VARCHAR (50),
        Group           VARCHAR (50)
);
```

Sample Data:

CWID	First_Name	Last_Name	Phone_Number	Email	Group
20084734	Gabrielle	Romano	845-747-9874	Gabrielle.Romano@marist.edu	Student
20034852	Travis	Beatty	845-834-1273	Travis.Beatty@marist.edu	Student
20074623	Christopher	Phrisce	845-832-9264	Christopher.Phrisce@marist.edu	Student
10198273	Jimmy	Chan	845-982-7463	Jimmy.Chan@marist.edu	Student
20093847	TK	Black	845-726-9182	TK.Black@marist.edu	Student
10192836	Ricky	Adams	845-736-9283	Ricky.Adams@marist.edu	Faculty
20043192	John	Smith	845-234-5433	John.Smith@marist.edu	Student
10124234	Amy	Baker	845-856-9483	Amy.Baker@marist.edu	Student
10194739	James	Jordan	201-453-8754	James.Jordan@marist.edu	Faculty
10192837	Jack	Grand	845-847-7362	Jack.Grand@marist.edu	Student
10198726	Jessica	Taylor	201-532-6251	Jessica.Taylor@marist.edu	Faculty
20098726	Tylor	Laoufter	201-918-8373	Tylor.Laoufter@marist.edu	Student

Create Statements

Employees *table:*

The Employees table is a sub-type of the People table in order to keep track of all basic employee information. The added information within the Employees table is exclusive to each individual Employee.

Functional Dependencies:

CWID → Stid, Hire_Date

Create Statement:

```
CREATE TABLE Employees (
        CWID        INT          NOT NULL     UNIQUE      REFERENCES People(CWID),
        Stid        VARCHAR(50)  NOT NULL,
        Hire_Date   DATE,
        PRIMARY KEY ( CWID )
);
```

Sample Data:

CWID	Stid	Hire_Date
20084734	strg1	2012-05-10
20034852	stbe1	2011-05-20
20074623	stpp3	2011-11-18
10198273	stjc1	2010-08-10
20093847	sttk2	2013-09-19
10192836	stra2	2011-12-15

Create Statement

Employee_Payments *table:*

The Employee_Payments table displays employees' salaries and the days in which they are paid based off of the CWID and Paymentid.

Functional Dependencies:

Paymentid, CWID → SalaryUSD, Payment_Date

Create Statement:

```
CREATE TABLE Employee_Payments (
    Paymentid       INT     NOT NULL,
    CWID            INT     NOT NULL    REFERENCES People(CWID),
    SalaryUSD       INT     NOT NULL,
    Payment_Date    DATE    NOT NULL,
    PRIMARY KEY ( Paymentid, CWID )
    );
```

Sample Data:

Paymentid	CWID	SalaryUSD	Payment_Date
1	**20084734**	8.00	2013-11-30
2	**20034852**	9.00	2013-10-30
3	**20074623**	8.00	2013-11-15
4	**10198273**	10.00	2013-10-15
5	**20093847**	9.00	2013-08-15
6	**10192836**	40.00	2013-11-30
7	**20084734**	8.00	2013-12-15
8	**20034852**	9.00	2013-11-15
9	**20074623**	8.00	2013-11-30
10	**10198273**	9.00	2013-10-30
11	**20093847**	10.00	2013-09-15
12	**10192836**	40.00	2013-12-15

Create Statements

Customers *table:*

Functional Dependencies:

CWID →

Create Statement:

```
CREATE TABLE Customers (
        CWID        INT     NOT NULL    UNIQUE      REFERENCES People(CWID),
        PRIMARY KEY ( CWID )
);
```

Sample Data:

CWID
20043192
10124234
10194739
10192837
10198726
20098726

Create Statements

Customers_Request *table:*

Functional Dependencies:

Processid, CWID→Date, Servicesid

Create Statement:

```
CREATE TABLE Customers_Request (
        Processid       INT     NOT NULL,
        CWID            INT     NOT NULL        REFERENCES People(CWID),
        Servicesid      INT     NOT NULL        REFERENCES Services(Servicesid),
        CDate   DATE    NOT NULL,
        PRIMARY KEY ( Processid, CWID )
);
```

Sample Data:

Processid	CWID	ComputerID	Drop_Off_Type
1	**20043192**	1	2013-10-12
2	**20043192**	2	"2013-10-30"
3	**20043192**	6	"2013-11-27"
4	**20043192**	2	"2013-12-03"
5	**10124234**	9	"2012-08-12"
6	**10124234**	8	"2013-10-30"
7	**10194739**	10	"2011-11-27"
8	**10194739**	6	"2013-12-03"
9	**10194739**	3	"2011-11-27"
10	**10194739**	1	"2013-12-03"
11	**10192837**	1	"2010-04-02"
12	**10198726**	2	"2011-11-27"
13	**10198726**	4	"2013-12-03"
14	**10198726**	1	"2010-04-02"
15	**20098726**	8	"2010-06-02"

Create Statements

Services *table:*

The Services table illustrates a list of the possible services technicians can perform on the various devices that customers bring in to ResNet.

Functional Dependencies:

Servicesid → SName

Create Statement:

```
CREATE TABLE Services (
    Servicesid      INT             NOT NULL        PRIMARY KEY,
    SName           VARCHAR(25)     NOT NULL
);
```

Sample Data:

Servicesid	SName
1	Virus Scan
2	Network Setup
3	Hardware Repair
4	OS Updates
5	Mail Setup
6	Network Exception
7	Reimage
8	Secure Data
9	Port Activation
10	Reinstall OS

Create Statements

Customer_Process *table:*

Functional Dependencies:

Processid, Computerid →

Create Statement:

```
CREATE TABLE Customers_Process (
        Processid       INT             NOT NULL        REFERENCES Customers_Requests(Processid),
        Computerid      VARCHAR(50)     NOT NULL        REFERENCES Devices(Computerid),
        PRIMARY KEY ( Processid, Computerid )
);
```

Sample Data:

Processid	ComputerID
1	00:13:00:E1:11:11
2	00:13:00:E1:11:11
3	00:12:11:E2:12:12
4	00:13:33:E3:33:44
5	00:0C:29:4D:71:EB
6	00:0B:30:5D:72:EA
7	00:00:5A:99:62:50
8	00:00:5A:99:62:50
9	00:00:5B:77:63:40
10	00:99:5C:44:62:41
11	00:99:12:A2:22:3F
12	00:87:F3:A2:98:BA
13	00:87:F3:A2:98:BA
14	00:89:F4:A3:99:AB
15	00:34:A5:7D:F3:60

Create Statements

Customers_Status *table:*

The Customers_Status table illustrates a list of processes and corresponding Statusids to determine what status a customer's device is in.

Functional Dependencies:

Processid, Statusid →

Create Statement:

```
CREATE TABLE Customers_Status (
        Processid       INT     NOT NULL    REFERENCES Customers_Process(Processid),
        Statusid        INT     NOT NULL    REFERENCES Status(Statusid),
        PRIMARY KEY ( Processid, Statusid )
);
```

Sample Data:

Processid	Statusid
1	1
2	2
3	1
4	4
5	1
6	3
7	3
8	1
9	1
10	1
11	1
12	1
13	3
14	1
15	4

Create Statements

Status *table:*

The Status table illustrates a list of the possible status conditions of a computer or other device that was or is in the process of being repaired.

Functional Dependencies:

Statusid → Status

Create Statement:

```
CREATE TABLE Status (
        Statusid   INT      NOT NULL      PRIMARY KEY,
        Status     VARCHAR(50)
);
```

Sample Data:

Statusid	Status
1	Fixed
2	Hardware Repair
3	Not Fixed
4	In Progress
5	Picked Up

Create Statements

***Devices** table:*

The Devices table illustrates a list of computers and mobile devices that were brought into ResNet for repair as well as their corresponding information.

Functional Dependencies:

Computerid → Brand, Model, OS, Password

Create Statement:

```
CREATE TABLE Devices (
        Computerid      VARCHAR(50)   NOT NULL      PRIMARY KEY,
        Brand           VARCHAR(50),
        Model           VARCHAR(50),
        OS              VARCHAR(50),
        Password        VARCHAR(50)
);
```

Sample Data:

Computerid	Brand	Model	OS	Password
00:13:00:E1:11:11	Apple	MacBookPro	10.9	Baseball1
00:12:11:E2:12:12	Apple	IPhone	6.8	9483
00:13:33:E3:33:44	Microsoft	Null	Null	Null
00:0C:29:4D:71:EB	Lenovo	T510	Windows 7	Iluvmykitty
00:0B:30:5D:72:EA	Andriod	Galexy SIII	Null	Null
00:00:5A:99:62:50	Apple	IMac	8.3	Snapcracklepop
00:00:5B:77:63:40	Sony	Null	Null	Null
00:99:5C:44:62:41	Apple	IPad 2	7.9	9876
00:99:12:A2:22:3F	Lenovo	T530	Windows 8	Grass22
00:87:F3:A2:98:BA	Toshiba	Null	Windows 8	BlackFriday
00:89:F4:A3:99:AB	Nintendo	Null	Null	Null
00:34:A5:7D:F3:60	Microsoft	Surface	Windows 8	847394Gs

Create Statements

Device_Types *table:*

The Device_Types table illustrates the correlation of a list of devices to its corresponding device type.

Functional Dependencies:

Computerid, Typid →

Create Statement:

```
CREATE TABLE Device_Types (
        Computerid      VARCHAR(50)   NOT NULL      REFERENCES Devices(Computerid),
        Typeid          INT           NOT NULL      REFERENCES Types(Typeid),
        PRIMARY KEY ( Computerid, Typeid )
);
```

Sample Data:

Computerid	Typeid
00:13:00:E1:11:11	1
00:12:11:E2:12:12	3
00:13:33:E3:33:44	8
00:0C:29:4D:71:EB	1
00:0B:30:5D:72:EA	3
00:00:5A:99:62:50	2
00:00:5B:77:63:40	9
00:99:5C:44:62:41	4
00:99:12:A2:22:3F	1
00:87:F3:A2:98:BA	1
00:89:F4:A3:99:AB	10
00:34:A5:7D:F3:60	4

Create Statements

Types table:

The Types table illustrates a list of the possible types a device can be considered.

Functional Dependencies:

Typeid → Computer_Type

Create Statements:

```
CREATE TABLE Types (
        Typeid          INT     NOT NULL    UNIQUE      REFERENCES Types(Typeid),
        Computer_Type   VARCHAR(50)  NOT NULL,
        PRIMARY KEY ( Typeid )
);
```

Sample Data:

Typeid	Computer_Type
1	Laptop
2	Desktop
3	Phone
4	Tablet
5	Server
6	Printer
7	Switch
8	Xbox
9	PlayStation
10	Wii

Views

Views: Customers_Number

This view is used to calculate how many times a customer has dropped off their computer. This could be useful for interpreting if a device has had the same problem multiple times.

Create View Query:

```
CREATE VIEW Customer_Number AS

SELECT count(c.CWID),

        c.cwid,

        p.First_Name,

        p.Last_Name

FROM Customers_Requests c

INNER JOIN People p

        ON c.CWID = p.CWID

GROUP BY c.CWID, p.First_Name, p.Last_Name

ORDER BY count(c.CWID) DESC
```

Sample Data:

Count	CWID	First_Name	Last_Name
4	20043192	John	Smith
4	10194739	James	Jordan
3	10194739	Jessica	Taylor
2	10124234	Amy	Baker
1	10192837	Jack	Grand
1	20098726	Tylor	Laoufter

Views

Views: Customers_Services

This view is used to calculate what services have been done on a device. This could be useful for figuring out what Services have been done on a device in the past in order to prepare solutions for future problems.

Create View Query:

```
CREATE VIEW Customers_Services AS

SELECT  s.SName,

        c.CWID,

        p.First_Name

FROM Customers_Requests c

INNER JOIN People p ON c.CWID=p.CWID

INNER JOIN Services s ON c.Servicesid=s.Servicesid

GROUP BY        s.SName, c.CWID, p.First_Name

ORDER BY        c.CWID
```

Sample Data:

SName	CWID	First_Name
Port Activation	10124234	Amy
Secure Data	10124234	Amy
Virus Scan	10192837	Jack
Hardware Repair	10194739	James
Network Exception	10194739	James
Reinstall OS	10194739	James
Virus Scan	10194739	James
Network Setup	10198726	Jessica
OS Updates	10198726	Jessica
Virus Scan	10198726	Jessica
Network Exception	20043192	John
Network Setup	20043192	John
Virus Scan	20043192	John
Secure Data	20098726	Tylor

Reports

REPORTS:

This report shows the students or faculty who have had Virus Scans on their devices. This could also be used on all of the other services if specified in the last sub-query.

Report Query:

```
SELECT First_Name, Last_Name, CWID
FROM People
WHERE CWID IN (
        SELECT CWID
        FROM Customers
        WHERE CWID IN (
                SELECT CWID
                FROM Customers_Requests
                WHERE Servicesid IN (
                        SELECT Servicesid
                        FROM services
                        WHERE Sname = 'Virus Scan'
                                )
                        )
ORDER BY Last_Name ASC
                        );
```

Sample Data:

First_Name	Last_Name	CWID
John	Smith	20043192
James	Jordan	10194739
Jack	Grand	10192837
Jessica	Taylor	10198726

Triggers

Triggers:

This trigger will execute after an Insert statement in the Employees table.

Hire_Employee Trigger:

```
CREATE OR REPLACE TRIGGER Hire_Employee

AFTER INSERT OF CWID

ON People

FOR EACH ROW

BEGIN

INSERT INTO Employees

        VALUES ( CWID, Stid, Hire_Date )

END;
```

Triggers:

This trigger will execute after an Insert statement in the Employees table.

***Fire_Employee* Trigger:**

```
CREATE OR REPLACE TRIGGER Fire_Employee

BEFORE DELETE OF CWID

ON People

FOR EACH ROW

BEGIN

INSERT INTO Employee_Fired (CWID, Stid, Hire_Date )

        VALUES (:OLD.CWID, :OLD.Stid, :OLD.Hire_Date );

END;
```

Stored Proceadures

Security

Security:

Allow customer access to know only the Status and Service that was performed on their computer.

Future Enhancements

Final Notes:

Known Problems:

- More Views should be created to determine statistics of devices that are serviced at ResNet.
- More security enhancements should be implemented on the database to protect customer's information.
- Since Marist College has different roles in regards to employees, branch tables off of the Employees table should be created to greater specify the difference between full time and part time employees.
    - A security revoke feature could also be implemented onto a possible part-time employees table to greater secure customer's information.

Future Enhancements:

- Add support for customers who want to mail in their computer over the summer. This will require implementing more attributes about the user.
- Add more views to calculate statistics for a particular year or month.
- Add more security enhancements to determine who has access to certain tables within the database.
- Add more stored procedures and triggers to the database to increase data integrity and consistency.