MARIST

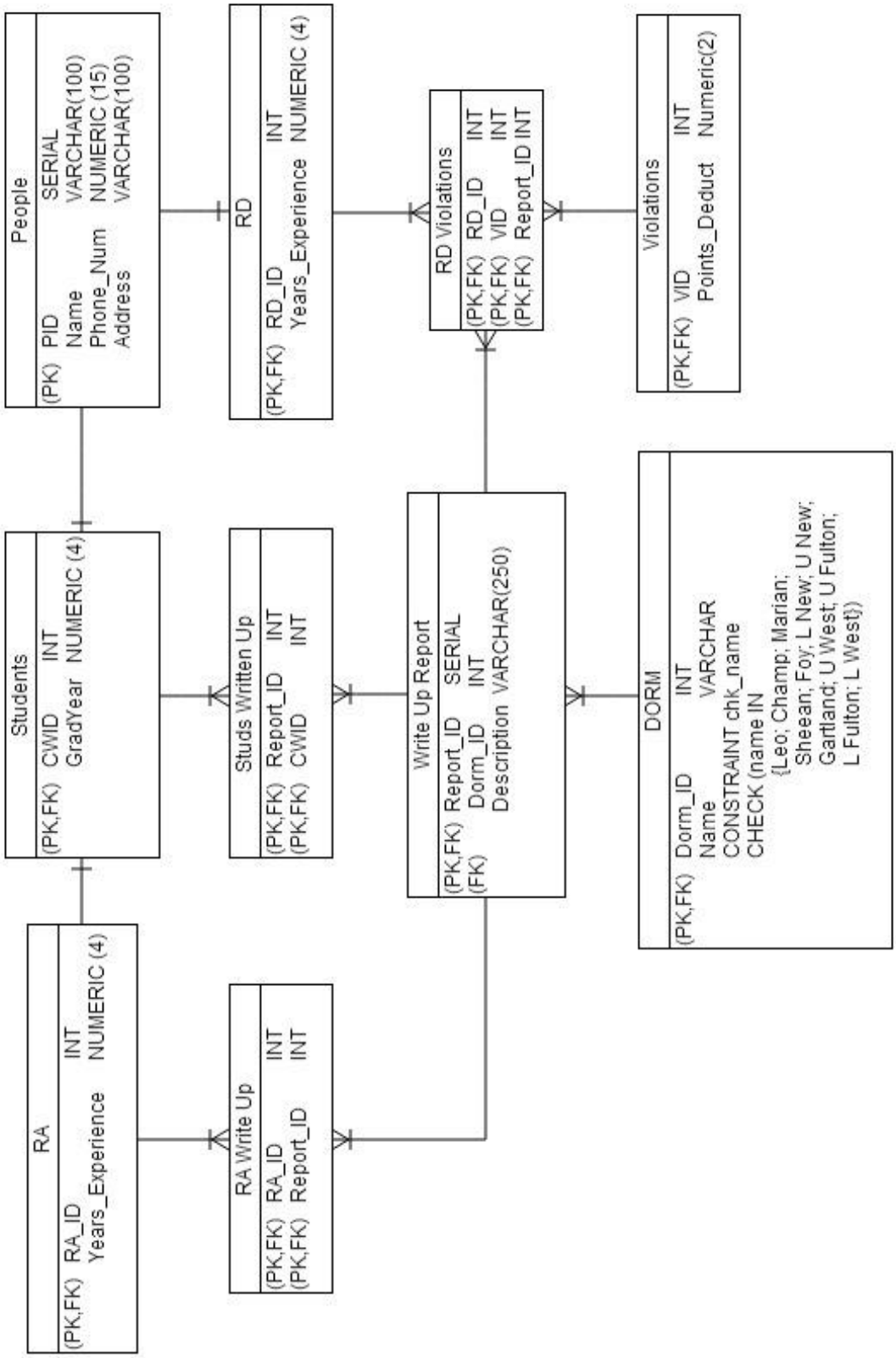
Write Up Log

(FOR RA AND RD USE ONLY)

Thomas McArdle

Database Systems

Table Of Contents


People
(PK) PID SERIAL
Name VARCHAR(100)
Phone_Num NUMERIC (15)
Address VARCHAR(100)
RD
(PK,FK) RD_ID INT
Years_Experience NUMERIC (4)
RD Violations
(PK,FK) RD_ID INT
(PK,FK) VID INT
(PK,FK) Report_ID INT
Violations
(PK,FK) VID INT
Points_Deduct Numeric(2)
Students
(PK,FK) CWID INT
GradYear NUMERIC (4)
Studs Written Up
(PK,FK) Report_ID INT
(PK,FK) CWID INT
Write Up Report
(PK,FK) Report_ID SERIAL
(FK) Dorm_ID INT
Description VARCHAR(250)
DORM
(PK,FK) Dorm_ID INT
Name VARCHAR
CONSTRAINT chk_name
CHECK (name IN
{Leo; Champ; Marian;
Sheean; Foy; L New; U New;
Gartland; U West; U Fulton;
L Fulton; L West})
RA
(PK,FK) RA_ID INT
Years_Experience NUMERIC (4)
RA Write Up
(PK,FK) RA_ID INT
(PK,FK) Report_ID INT

Executive Summary

This Document Provides an in depth outline of a database system designed to help Residential Assistants and Residential Directors keep track of students write up reports. The database can keep track of students that got written up, where they were written up, who wrote them up, and the RD's final decision on the amount of priority points deducted. The ER diagram shows the relationship between all of the tables in the database. It also shows what data types are excepted for each column in each table and the restrictions that apply to some columns. There is provided sample data to demonstrate some of the functions of the database.

Overview

The database system can be broken down into multiple sections. Students, RA's, RD's, Write Up Reports, and Violations. As the students do something wrong, an RA will file a report on a student. A Report will consist of the student's name, description of the event, and where the event occurred.

The RD's will have the final say on the amount of priorities points deducted depending on the situations. This will also be stored in the database and can be looked up per student.

People

The **people** table will contain all of the people entered into the database. A person can either be a **Student** or a **RD**. A student can then further be extend to an **RA** because you have to be a **student** in order to be an **RA**. All values have to be filled in, none can be left null or empty.

```
-- People --
CREATE TABLE people_persons (
        pid         SERIAL NOT NULL PRIMARY KEY,
        name        VARCHAR(50),
         phone_num NUMERIC(10),
        address VARCHAR (100)
);
```

	pid integer	name character varying(50)	phone_num numeric(10,0)	address character varying(100)
1	2	Tom Mc	6313327857	Florida
2	3	Plunkett	1564315342	yonkers
3	4	Chris Smith	456456456	Queens
4	5	Jones	898998989	Poughkeepsie
5	6	Farrell	323232111	Hawaii
6	7	Dillon	1235455698	Colorado
7	8	Mac Attack	1365659895	Cuba
8	9	The X Factor	1289654766	Ireland
9	1	Adam Scott	1234567890	17 Walabee Lane

Student

The **student** table holds the information for a student. However most of the student's personal information is stored in the **people table.** The students table references the people table whenever it needs access to the other information. The **students** table also contains the graduation year of the student. This could be used to sort the students by their class.

```
-- Students --
CREATE TABLE students (
        CWID INT NOT NULL REFERENCES people_persons(pid)
                unique ,
        GradYear NUMERIC(4)
 );
```

	cwid integer	gradyear numeric(4,0)
1	1	2014
2	2	2015
3	3	2014
4	6	2014
5	7	2015
6	8	2014

RA

The **RA** table contains all the residential assistants that are currently employed. Because RA's have to be a student the **RA** table references the **students** table. Further information would then reference the **peoples** table. The **RA** table also contains a column for the amount of experienced years, labeled **years_experience.**

```
-- RA --
CREATE TABLE RA (
        RA_ID INT NOT NULL REFERENCES students(cwid)
        UNIQUE,
        years_experience NUMERIC(4)
);
```

	ra_id integer	years_experience numeric(4,0)
1	7	2
2	8	2
3	6	0

RD

The **RD** table is the table where all RD's are stored. This table has a column in which it references the people table in order to retrieve more basic information about the person. The table also contains a years of experienced column names **years_experienced** similar to the RA's table.

```
-- RD --
CREATE TABLE RD (
        RD_ID INT NOT NULL REFERENCES
       people_persons(pid) UNIQUE ,
        years_experience NUMERIC(4)
);
```

	rd_id integer	years_experience numeric(4,0)
1	4	4
2	5	3

DORMS

The **dorms** table contains all of the dorms on campus. It has a restrain on the input so you can only enter specific string that match the restraint. This was done to avoid redundancy in the table. This table is referenced by the **write_up_report** table when the location of a write up is needed.

```
-- DORM --
CREATE TABLE Dorm (
  Dorm_ID SERIAL NOT NULL PRIMARY KEY,
  Name VARCHAR(50) UNIQUE
      CONSTRAINT chk_name CHECK (name IN ('Leo',   'Champ', 'Marian', 'Sheean', 'Foy', 'L New', 'U New',    'Gartland','U West', 'U Fulton', 'L Fulton','L West'))
);
```

	dorm_id integer	name character varying(50)
1	1	Leo
2	2	Champ
3	3	Marian
4	4	Sheean
5	5	Foy
6	6	L New
7	7	U New
8	8	Gartland
9	9	U West
10	10	U Fulton
11	11	L Fulton
12	12	L West

Violations

The violation table contains a **VID** and points deducted column. This table is referenced by the **RD Violations**. It is used to keep track of the amount of points that can be deducted. This is to avoid redundancy in another table.

```
-- Violations --
CREATE TABLE violations (
  VID serial not null primary key,
  points_deduct  int Unique
);
\
```

	vid integer	points_deduct integer
1	1	1
2	2	2

Write Up Report

The **Write_Up_Report** is probably the most important table if this database system. This table is what combines **student**, **RA**, **Dorm**, **RD** and **violations**. This table references all of the tables mentioned above using ID numbers to keep track of all of the filed reports. This table also contains the description of the event that is being filed.

```
-- Write Up Report --
CREATE TABLE write_up_report (
        report_ID SERIAL NOT NULL PRIMARY KEY,
         LID  INT NOT NULL REFERENCES DORM(DORM_ID),
         Descrip VARCHAR(250)
);
```

	report_id integer	lid integer	descrip character varying(250)
1	1	6	To Many Lines of code
2	2	3	Music too loud

RA Write Up

The **RA_Write_Up** table is another associative table. This is used to connect the **RA** table to **Write_Up_Reports.** This table technically doesn't get any data inputed into it through a user. Instead it references columns in the RA table and the **Write_Up_Reports** table. The reason this connecting table is used is so that when filing a report, you can have multiple RA's to one report and multiple reports to a single RA.

```
-- RA WRITE UP --
CREATE TABLE RA_write_up(
        RA_ID INT NOT NULL REFERENCES ra(ra_id) ,
        report_ID  INT NOT NULL REFERENCES
            write_up_report(report_ID)
);
```

	ra_id integer	report_id integer
1	7	1
2	8	2

RD Violations

The **RD_Violations** table is another associative table. It is used to connect the points deducted by the RA to the actual report. It also connects the RD that deducted the points to the report. This table doesn't take in input through a human but references three surrounding tables for its data. This helps to avoid duplications throughout the database.

```
-- RD Violations --
CREATE TABLE RD_violations (
        RD_ID INT NOT NULL REFERENCES RD(RD_ID),
         VID  INT NOT NULL REFERENCES violations(VID),
         Report_ID INT NOT NULL REFERENCES
              write_up_report(Report_ID)
);
```

	rd_id integer	vid integer	report_id integer
1	4	1	1
2	5	2	2

Students Written Up

The **Studs_Written_Up** table is an associative table used to connect the **students** table to the **write_up_report** table. This table references both the student table and write up report table in oreder to allow for many students to be written up, and allow many write ups to contain many students. Even though the table doesn't look like much it is very powerful.

```
-- Students Written Up --
CREATE TABLE studs_written_up(
  CWID        INT NOT NULL REFERENCES students(CWID),
  Report_ID   INT NOT NULL REFERENCES
write_up_report(report_ID)
);
```

	cwid integer	report_id integer
1	1	1
2	2	2

FUNCTIONAL DEPENDCIES

People
PID → Name, Phone_Num, Address

Students
CWID → GradYear

RA
RA_ID → Years_Experience

RD
RD_ID →

DORM
Dorm_ID → Name

Violations
VID → Points_Deducted

RD_Violations
{RD_ID, VID, Report_ID} →

RA_Write_Up
{RA_ID, Report_ID} →

Studs_Written_Up
{Report_ID, CWID} →

Write_Up_Report
Report_ID → Descrip

Reports

Students Written Up By Class

```
SELECT DISTINCT Name, gradyear
FROM people_persons p,
        students s,
        studs_written_up swu,
        write_up_report wur
WHERE   S.CWID = P.PID
     AND S.CWID = swu.cwid
     AND WUR.Report_ID = SWU.Report_ID
ORDER BY s.GradYear
```

What RA's Wrote Which Students

```
SELECT p1.name, p2.name

FROM   people_persons p1,
        people_persons p2,
        ra r,
        students s,
       ra_write_up rwu,
       studs_written_up swu,
       write_up_report wur

WHERE      p1.pid = r.ra_id
       AND p2.pid = s.cwid
       AND s.cwid = swu.cwid
       AND r.ra_id = rwu.ra_id
       AND rwu.report_id = wur.report_id
       AND swu.report_id = wur.report_id
```

Stored Procedures

This Store Procedure takes in a students CWID number and returns all the times the student has gotten written up by an RA along with which RA filed the report and a description of what occurred.

```
CREATE OR REPLACE FUNCTION chk_student_report(int, refcursor) returns
refcursor
AS $$
DECLARE
  nameparam int := $1;
  resultset refcursor := $2;
BEGIN
OPEN resultset for
 SELECT p1.pid, p1.name as "Student", p2.name as"RA", wur.descrip
 FROM  people_persons p1,
      people_persons p2,
        students s,
        studs_written_up swu,
        write_up_report wur,
        ra,
        ra_write_up rwu
WHERE   p1.pid = s.cwid
        AND s.cwid = swu.cwid
        AND p2.pid = ra.ra_id
        AND ra.ra_id = rwu.ra_id
        AND rwu.report_id = wur.report_id
        AND swu.report_id = wur.report_id
        AND p1.pid = nameparam;
  RETURN resultset;
END;
$$ language plpgsql;

SELECT chk_student_report(2, 'results');
FETCH ALL FROM RESULTS;
```

This stored procedure takes in a students CWID number and returns how many priority points were deducted based on the amount of write ups and the significance of the write up.

```
CREATE OR REPLACE FUNCTION chk_student_penalties(int, refcursor)
RETURNS refcursor
AS $$
DECLARE
  nameparam int := $1;
  resultset refcursor := $2;
BEGIN
OPEN resultset for
 SELECT p.pid, p.name, wur.descrip, v.points_deduct as "Priority Point Lost"
 FROM  people_persons p,
       students s,
       studs_written_up swu,
       RD_violations rv,
       write_up_report wur,
       violations v
WHERE   p.pid = s.cwid
       AND s.cwid = swu.cwid
       AND swu.report_id = wur.report_id
       AND wur.report_id = rv.report_id
       AND rv.vid = v.vid
       AND p.pid = nameparam;
  RETURN resultset;
END;
$$ language plpgsql;

SELECT chk_student_report(2, 'results');
FETCH ALL FROM RESULTS;
```

TRIGGERS

This trigger is designed so that when a report is deleted from the write_up_report table it is also deleted from the violations table. This is to avoid confusion in tables and the data is technically useless at that point.

```
CREATE TRIGGER deleteReport
    AFTER DELETE ON write_up_report
    FOR EACH ROW
    EXECUTE PROCEDURE deleteReport;

CREATE TRIGGER deleteReport()
RETURN tigger AS $$
BEGIN
    DELETE FROM violations
    WHERE v.report_ID NOT EXISTS( SELECT report_ID
                                  FROM write_up_report) ;
```

VIEWS

This view could be used by the RA's in CHAMP to see who has gotten written up. This will help them identified the students room and keep an eye on them so they don't do any more bad things to get written up.

```
CREATE VIEW Students_WRITTEN_UP_IN_MARIAN AS
Select s.cwid, p.name, s.gradyear, p.phone_num, p.address
from    people_persons p,
        students s,
        studs_written_up swu,
        write_up_report wur,
        dorm d
where p.pid = s.cwid
        AND s.cwid = swu.cwid
        AND swu.report_id = wur.report_id
        AND wur.dorm_id = d.dorm_id
        AND d.name =  "MARIAN"
```

	cwid integer	name character varying(50)	gradyear numeric(4,0)	phone_num numeric(10,0)	address character varying(100)
1	2	Tom Mc	2015	6313327857	Florida

IMPLEMENTATION NOTES

- This database was designed so little human error could be made. Due to restraints and the limited fields in each column viewing data will be quick and simple.

SECURITY

- This system will use multiple accounts which will grant access to specific users. This is useful so RD's will have the most access. This is good because they will be the only ones with access to grant priority points and remove them. Students will not have access and RA's will only be able to file reports. RA's will not be able to remove priority points.

KNOWN PROBLEMS

- RA's after they graduate, should be automatically removed from the table. Maybe make the gradyear and grad date and compared it to the actual date. This would remove a human step in having to remove an RA.
- Kept all the tables very generic but many more specific fields could be added.
- Address could technically be Identified by a zip code and then looked up if they are in the United States. Would make sorting students by address more accurate.

FUTURE ENHACEMENTS

- If more time was given, and maybe a decent pay, some things could've been added.
    - For instance every time a student's name is filled in a report they will automatically be sent a notification email notifying them of there actions.
    - Another thing that could be added is an automatic scheduling section. An RD could specify his or her times and the database will fill in the empty slots and that would also be sent along in the email.