Final Database

Daniel Gisolfi

Section 112

Table of Contents

Database Description

I have chosen to create a relational database of the department I work in, Marist college IT. The IT department at Marist is a perfect example of a database as there are many groups of people who interact with each other and have different access, managers, and pay. The department itself has many sub-departments that work as small pieces in the main IT department. It will be useful to represent these small sub-departments as well as their relationships with the college, technology, and others. This Database is a good representation of how the Information department currently is set up and operates, this will help in developing a working database that corresponds to this diagram.

Business Rules

- Within the IT Department there are sub-departments, each sub-department has a building location, room number, and name
- There are jobs in each department and a job can only belong to one department, each job is described with a name and description as well as a corresponding department
- All jobs have amends and are paid either a salary or an hourly wage depending on the job, a job can only have one pay option
- Jobs are worked by either student employees or staff, a student employee is a student of the college works part-time at the college. A staff member is a full-time worker at the college.
- Student employees have a status of either commuter or resident and are managed by one staff member
- Student employees are assigned shifts and can only work a set amount of hours usually 20 and request a number of hours they would like to work
- Student employees are assigned projects that have set names and descriptions
- Staff may be deployed a device which has a serial and is either a desktop or laptop
- Everyone whether student employee or staff have a user account, the username is set as there first name and they set there own password
- All user accounts can be given access to drives, drives have a name and a maximum storage that cannot be exceeded(it can be increased, however)

ER Diagram

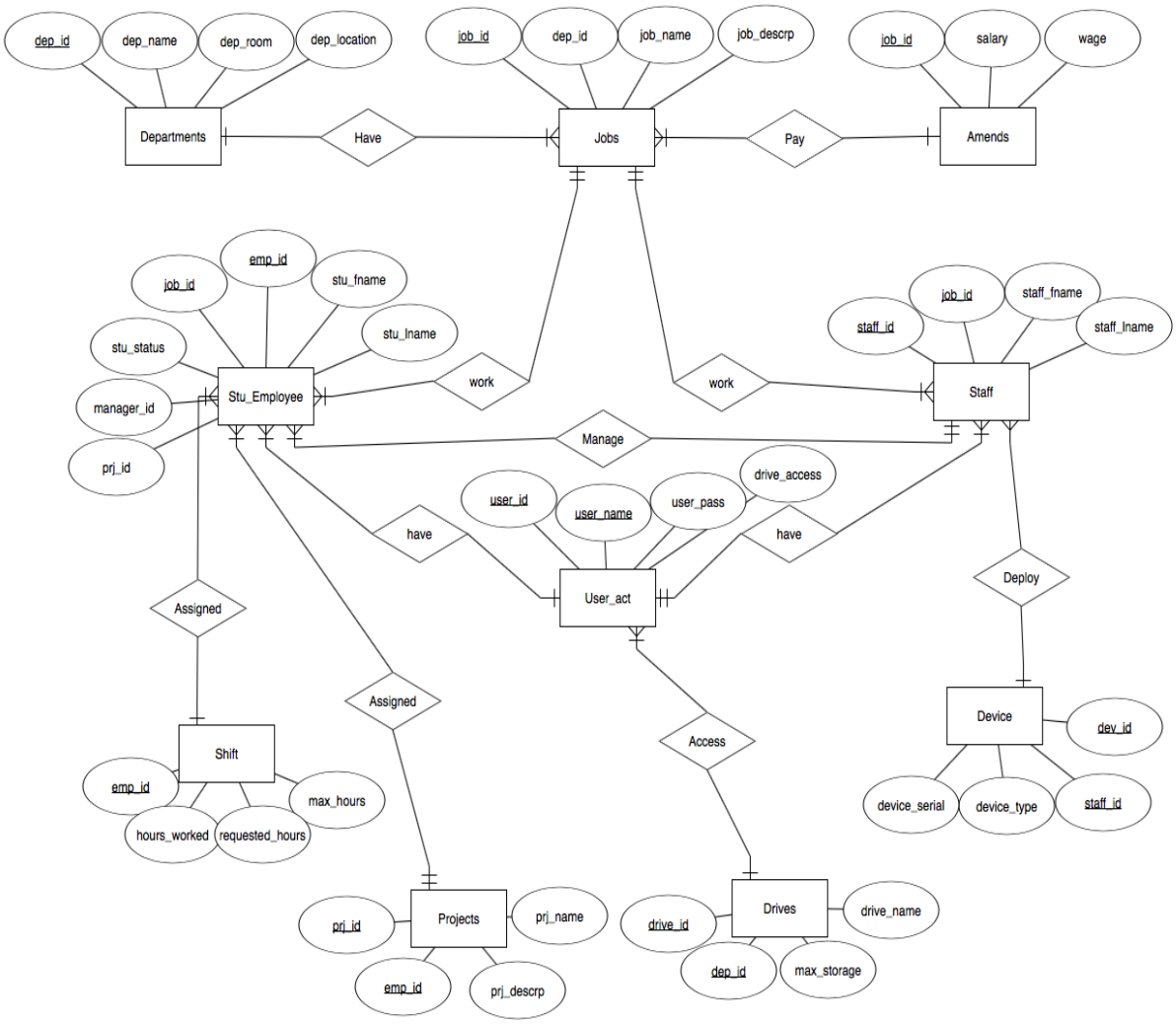

Table Definitions

Amends

3NF Justification - This table is in 3rd normal form because there is a defined key: job_id which references Jobs: job_id and does not have any transitive dependencies, job_id is out of the table. Additionally no partial dependencies exist.

Table description - This table holds all the data for what job gets paid a staff salary or an employee wage as well as what that value is. It has 2 columns for each and either can be null depending on what job it is, determined by the job_id which is taken from the jobs table.

```
--Amends.sql
--Create and populate Amends Table
--Author: Daniel Gisolfi
--DB Management Final Project

CREATE TABLE Amends(
```

```
    job_id              int                 NOT NULL,
    salary             int,
    wage              DECIMAL(5,2));

ADD CONSTRAINT fk_job_id_Amends FOREIGN KEY (job_id) REFERENCES
jobs(job_id);

--Input all staff pay
INSERT INTO Amends (job_id, salary)
    VALUES (1, 60000);
INSERT INTO Amends (job_id, salary)
    VALUES (3, 60000);
INSERT INTO Amends (job_id, salary)
    VALUES (5, 60000);
INSERT INTO Amends (job_id, salary)
    VALUES (7, 60000);
INSERT INTO Amends (job_id, salary)
    VALUES (9, 60000);
INSERT INTO Amends (job_id, salary)
    VALUES (11, 60000);
INSERT INTO Amends (job_id, salary)
    VALUES (13, 60000);
INSERT INTO Amends (job_id, salary)
    VALUES (15, 60000);
INSERT INTO Amends (job_id, salary)
    VALUES (17, 60000);

--Insert all employees pay
INSERT INTO Amends (job_id, wage)
    VALUES (2, 10.40);
INSERT INTO Amends (job_id, wage)
    VALUES (4, 10.40);
INSERT INTO Amends (job_id, wage)
    VALUES (6, 10.40);
INSERT INTO Amends (job_id, wage)
    VALUES (8, 10.40);
INSERT INTO Amends (job_id, wage)
    VALUES (10, 10.40);
INSERT INTO Amends (job_id, wage)
    VALUES (12, 10.40);
INSERT INTO Amends (job_id, wage)
    VALUES (14, 10.40);
INSERT INTO Amends (job_id, wage)
    VALUES (16, 10.40);
INSERT INTO Amends (job_id, wage)
    VALUES (18, 10.40);
```

Departments

3NF Justification - This table is in 3rd normal form because there is a defined primary key: dep_id and does not have any transitive dependencies. Additionally no partial dependencies exist.

Table description - This table defines the specifics of each sub-department within IT, including the name, location and room number of each department. This is necessary to refer to when finding what staff or employees are located where.

```
--Departments.sql
--Create and populate Departments Table
--Author: Daniel Gisolfi
--DB Management Final Project

CREATE TABLE Departments(
    dep_id              int                 NOT NULL,
    dep_name          VARCHAR2(20)     NOT NULL,
    dep_location     VARCHAR2(20)     NOT NULL,
    dep_room         int              NOT NULL);



ADD CONSTRAINT pk_dep_id PRIMARY KEY (dep_id);

INSERT INTO Customers (dep_id, dep_name, dep_location, dep_room)
    VALUES ( 1, 'Desktop', 'Donnelly', 101);
INSERT INTO Customers (dep_id, dep_name, dep_location, dep_room)
    VALUES ( 2, 'Resnet', 'Donnelly', 101);
INSERT INTO Customers (dep_id, dep_name, dep_location, dep_room)
    VALUES ( 3, 'Help Desk', 'Donnelly', 258);
INSERT INTO Customers (dep_id, dep_name, dep_location, dep_room)
    VALUES ( 4, 'Web Services', 'Library', 310);
INSERT INTO Customers (dep_id, dep_name, dep_location, dep_room)
    VALUES ( 5, 'Applications', 'Donnelly', 260);
INSERT INTO Customers (dep_id, dep_name, dep_location, dep_room)
    VALUES ( 6, 'Digital Education', 'Donnelly', 260);
INSERT INTO Customers (dep_id, dep_name, dep_location, dep_room)
    VALUES ( 7, 'Networking', 'Donnelly', 110);
INSERT INTO Customers (dep_id, dep_name, dep_location, dep_room)
    VALUES ( 8, 'Telecom', 'Donnelly', 110);
INSERT INTO Customers (dep_id, dep_name, dep_location, dep_room)
    VALUES ( 9, 'Card Services', 'Donnelly', 110);
```

Devices

3NF Justification - This table exists in 3rd normal form, there is a primary key and the transitive dependency was taken out of the table and exists as a foreign key, additionally there are no partial dependencies

Table description - This table defines what devices are deployed to who. This is only an option for staff and is a value of either Laptop or Desktop for dev_type. Additionally, there is a dev_id and a serial number to identify the device.

```
--Devices.sql
--Create and populate Devices Table
--Author: Daniel Gisolfi
--DB Management Final Project

CREATE TABLE Devices(
    dev_id        int                NOT NULL,
    staff_id    int                NOT NULL,
    dev_type    VARCHAR2(20)    NOT NUll,
    dev_serial    VARCHAR2(20)    NOT NUll);

ADD CONSTRAINT pk_dev_id PRIMARY KEY (dev_id);
ADD CONSTRAINT fk_staff_id_dev FOREIGN KEY (staff_id) REFERENCES
Staff(staff_id);

INSERT INTO shifts (dev_id, staff_id, dev_type, dev_serial)
    VALUES (1, 1, 'Desktop', 'F3K8D7B5');
INSERT INTO shifts (dev_id, staff_id, dev_type, dev_serial)
    VALUES (2, 2, 'Desktop', 'V6K8D7B5');
INSERT INTO shifts (dev_id, staff_id, dev_type, dev_serial)
    VALUES (3, 3, 'Laptop', 'J7K8D7B5');
INSERT INTO shifts (dev_id, staff_id, dev_type, dev_serial)
    VALUES (4, 4, 'Desktop', 'Q8K8D7B5');
INSERT INTO shifts (dev_id, staff_id, dev_type, dev_serial)
    VALUES (5, 5, 'Laptop', 'P3K8D7B5');
INSERT INTO shifts (dev_id, staff_id, dev_type, dev_serial)
    VALUES (6, 6, 'Laptop', 'S3K8D7B5');
INSERT INTO shifts (dev_id, staff_id, dev_type, dev_serial)
    VALUES (7, 7, 'Laptop', 'L3K8D7B5');
INSERT INTO shifts (dev_id, staff_id, dev_type, dev_serial)
    VALUES (8, 8, 'Desktop', 'D3K8D7B5');
INSERT INTO shifts (dev_id, staff_id, dev_type, dev_serial)
    VALUES (9, 9, 'Desktop', 'T3K8D7B5');
INSERT INTO shifts (dev_id, staff_id, dev_type, dev_serial)
    VALUES (10, 10, 'Laptop', 'R3K8D7B5');
```

Drives

3NF Justification - This table is in 3rd normal form because there is a primary key: drive_id and does not have any transitive dependencies, departments is out of the table. Additionally, no partial dependencies exist

Table description - This table holds all the data for all shared drives across the department. There are departments with drives and some without, it is even possible for a department to have more than one. Each Drive has an ID, a corresponding department ID, a name and the maximum storage of the drive, which can be altered to allow more room

```
--Drives.sql
--Create and populate Drives Table
--Author: Daniel Gisolfi
--DB Management Final Project

CREATE TABLE Drives(
    drive_id        int                NOT NULL,
    dep_id            int                NOT NULL,
    drive_name        VARCHAR2(20)    NOT NUll,
    max_storage        VARCHAR2(20)    NOT NUll);

ADD CONSTRAINT pk_drive_id PRIMARY KEY (drive_id);
ADD CONSTRAINT fk_dep_id_drive FOREIGN KEY (dep_id) REFERENCES
Departments(dep_id);


INSERT INTO Drives (drive_id, dep_id, drive_name, max_storage)
    VALUES (1, 1, 'DesktopSharedDrive', '3TB');
INSERT INTO Drives (drive_id, dep_id, drive_name, max_storage)
    VALUES (2, 2, 'ResnetSharedDrive', '1TB');
INSERT INTO Drives (drive_id, dep_id, drive_name, max_storage)
    VALUES (3, 3, 'HelpDeskSharedDrive', '5TB');
INSERT INTO Drives (drive_id, dep_id, drive_name, max_storage)
    VALUES (4, 4, 'WebServicesSharedDrive', '2TB');
INSERT INTO Drives (drive_id, dep_id, drive_name, max_storage)
    VALUES (5, 5, 'ApplicationsSharedDrive', '1TB');
INSERT INTO Drives (drive_id, dep_id, drive_name, max_storage)
    VALUES (6, 6, 'DigitalEducationSharedDrive', '500GB');
INSERT INTO Drives (drive_id, dep_id, drive_name, max_storage)
    VALUES (7, 7, 'NetworkingSharedDrive', '2TB');
INSERT INTO Drives (drive_id, dep_id, drive_name, max_storage)
    VALUES (8, 8, 'TelecomSharedDrive', '2TB');
INSERT INTO Drives (drive_id, dep_id, drive_name, max_storage)
    VALUES (9, 9, 'CardServicesSharedDrive', '4TB');
INSERT INTO Drives (drive_id, dep_id, drive_name, max_storage)
    VALUES (10, 1, 'DesktopDevicesSharedDrive', '10TB');
```

Jobs

3NF Justification - The jobs table has a primary key of job_id defined to refer to jobs further down the tables, it also has a foreign key referring to the department in which a job belongs. There are no partial dependencies within the table

Table description - This table defines a job with a name and description for each regardless if a person is a staff or a student employee there job is derived from the jobs table along with what department it belongs to.

```
--jobs.sql
--Create and populate Jobs Table
--Author: Daniel Gisolfi
--DB Management Final Project

CREATE TABLE Jobs(
    job_id              int                 NOT NULL,
    dep_id             int                 NOT NUll,
    job_name          VARCHAR2(20)    NOT NULL,
    job_descrp         VARCHAR2(40)    NOT NULL);

ADD CONSTRAINT pk_job_id PRIMARY KEY (job_id);
ADD CONSTRAINT fk_dep_id_jobs FOREIGN KEY (dep_id) REFERENCES
Departments(dep_id);


--Insert all entries for all departments
INSERT INTO Jobs (job_id, dep_id, job_name, job_descrp)
    VALUES ( 1, 1, 'Desktop Admin', 'Manage employees and runs the Desktop
department');
INSERT INTO Jobs (job_id, dep_id, job_name, job_descrp)
    VALUES ( 2, 1,'Desktop Employee', 'Works in the Desktop department');

INSERT INTO Jobs (job_id, dep_id, job_name, job_descrp)
    VALUES ( 3, 2,'Resnet Admin', 'Manage employees and runs the Resnet
department');
INSERT INTO Jobs (job_id, dep_id, job_name, job_descrp)
    VALUES ( 4, 2,'Resnet Employee', 'Works in the Resnet department');

INSERT INTO Jobs (job_id, dep_id, job_name, job_descrp)
    VALUES ( 5, 3,'Help Desk Admin', 'Manages employees and runs the Help
Desk department');
INSERT INTO Jobs (job_id, dep_id, job_name, job_descrp)
    VALUES ( 6, 3,'Help Desk Employee', 'Works in the Help Desk
department');

```

```
INSERT INTO Jobs (job_id, dep_id, job_name, job_descrp)
    VALUES ( 7, 4,'Web Services Admin', 'Manage employees and runs the Web Services department');
INSERT INTO Jobs (job_id, dep_id, job_name, job_descrp)
    VALUES ( 8, 4,'Web Services Employee', 'Works in the Web Services department');

INSERT INTO Jobs (job_id, dep_id, job_name, job_descrp)
    VALUES ( 9, 5,'Applications Admin', 'Manage employees and runs the Applications department');
INSERT INTO Jobs (job_id, dep_id, job_name, job_descrp)
    VALUES ( 10, 5,'Applications Employee', 'Works in the Applications department');

INSERT INTO Jobs (job_id, dep_id, job_name, job_descrp)
    VALUES ( 11, 6,'Digital Education Admin', 'Manage employees and runs the Digital Education department');
INSERT INTO Jobs (job_id, dep_id, job_name, job_descrp)
    VALUES ( 12, 6,'Digital Education Employee', 'Works in the Digital Education department');

INSERT INTO Jobs (job_id, dep_id, job_name, job_descrp)
    VALUES ( 13, 7,'Networking Admin', 'Manage employees and runs the Networking department');
INSERT INTO Jobs (job_id, dep_id, job_name, job_descrp)
    VALUES ( 14, 7,'Networking Employee', 'Works in the Networking department');

INSERT INTO Jobs (job_id, dep_id, job_name, job_descrp)
    VALUES ( 15, 8,'Telecom Admin', 'Manage employees and runs the Telecom department');
INSERT INTO Jobs (job_id, dep_id, job_name, job_descrp)
    VALUES ( 16, 8,'Telecom Employee', 'Works in the Telecom department');

INSERT INTO Jobs (job_id, dep_id, job_name, job_descrp)
    VALUES ( 17, 9,'Card Services Admin', 'Manage employees and runs the Card Services department');
INSERT INTO Jobs (job_id, dep_id, job_name, job_descrp)
    VALUES ( 18, 9,'Card Services Employee', 'Works in the Card Services department');
```

Projects

3NF Justification - This table has a primary key: prj_id without partial dependencies and its transitive dependency was removed in making it a foreign key

Table description - The projects table defines projects that are assigned to stu_employees only, staff assign these projects and oversee them but that relationship exists in one table higher Stu_Employee. Each table has a designated id, name description and a corresponding emp_id referring to Stu_Employee.

```
--Projects.sql
--Create and populate Projects Table
--Author: Daniel Gisolfi
--DB Management Final Project

CREATE TABLE Projects(
    prj_id            int                NOT NULL,
    emp_id            int                NOT NULL,
    prj_name        VARCHAR2(30)    NOT NUll,
    prj_descrp        VARCHAR2(80)    NOT NUll);

ADD CONSTRAINT pk_prj_id PRIMARY KEY (prj_id);
ADD CONSTRAINT fk_emp_id_prj FOREIGN KEY (emp_id) REFERENCES
Stu_Employee(emp_id);

INSERT INTO Projects (prj_id, emp_id, prj_name, prj_descrp)
    VALUES (1, 3, 'Request Organazation', 'Organizing all requests for data
access');
INSERT INTO Projects (prj_id, emp_id, prj_name, prj_descrp)
    VALUES (2, 1, 'Device Deployemnt', 'Organizing all device
deployemnts');
INSERT INTO Projects (prj_id, emp_id, prj_name, prj_descrp)
    VALUES (3, 2, 'Device Repairs', 'Organizing all personal device
repairs');
```

Shifts

3NF Justification - This table has a designated key of emp_id which is a foreign key to Stu_Employee. There are no transitive or partial dependencies making this table 3rd normal form

Table description - The Shifts table designates how many hours an employee has, will and can work. There is a set max amount of hours and is subject to change if it is during the semester or not. The table is connected to the Stu_Employee through the foreign key emp_id

```sql
--Shifts.sql
--Create and populate Shifts Table
--Author: Daniel Gisolfi
--DB Management Final Project

CREATE TABLE Shifts(
    emp_id            int        NOT NULL,
    requested_hours    int        NOT NULL,
    hours_worked    int        NOT NUll,
    max_hours        int        NOT NUll);

ADD CONSTRAINT fk_emp_id_shift FOREIGN KEY (emp_id) REFERENCES
shifts(emp_id);


INSERT INTO shifts (emp_id, requested_hours, hours_worked, max_hours)
    VALUES (1, 17, 15, 20);
INSERT INTO shifts (emp_id, requested_hours, hours_worked, max_hours)
    VALUES (2, 10, 12, 20);
INSERT INTO shifts (emp_id, requested_hours, hours_worked, max_hours)
    VALUES (3, 12, 11, 20);
INSERT INTO shifts (emp_id, requested_hours, hours_worked, max_hours)
    VALUES (4, 18, 10, 20);
INSERT INTO shifts (emp_id, requested_hours, hours_worked, max_hours)
    VALUES (5, 20, 15, 20);
INSERT INTO shifts (emp_id, requested_hours, hours_worked, max_hours)
    VALUES (6, 17, 13, 20);
INSERT INTO shifts (emp_id, requested_hours, hours_worked, max_hours)
    VALUES (7, 18, 16, 20);
INSERT INTO shifts (emp_id, requested_hours, hours_worked, max_hours)
    VALUES (8, 14, 11, 20);
INSERT INTO shifts (emp_id, requested_hours, hours_worked, max_hours)
    VALUES (9, 10, 4, 20);
INSERT INTO shifts (emp_id, requested_hours, hours_worked, max_hours)
    VALUES (10, 13, 9, 20);
```

Staff

3NF Justification - This table includes a primary and foreign key, no data is repeated and is void of any partial or transitive dependencies making the table is in at least 3rd normal form.

Table description - If a person is full time, and not a student they are classified as staff, and have there owned ID as well as first and last name. This table keeps track of who is staff in order to derive who gets a device, what pay they receive etc.

```sql
--Staff.sql
--Create and populate Staff Table
--Author: Daniel Gisolfi
--DB Management Final Project

CREATE TABLE Staff(
    staff_id        int                NOT NULL,
    job_id             int                NOT NULL,
    staff_fname        VARCHAR2(20)    NOT NUll,
    staff_lname        VARCHAR2(20)    NOT NUll);

ADD CONSTRAINT pk_staff_id PRIMARY KEY (staff_id);
ADD CONSTRAINT fk_job_id_staff FOREIGN KEY (job_id) REFERENCES
jobs(job_id);

INSERT INTO Staff (staff_id, job_id, stu_fname, stu_lname)
    VALUES (1, 1, 'Nick', 'Smith');
INSERT INTO Staff (staff_id, job_id, stu_fname, stu_lname)
    VALUES (2, 3, 'Marty', 'Philips');
INSERT INTO Staff (staff_id, job_id, stu_fname, stu_lname)
    VALUES (3, 5, 'Katherine', 'Jacobs');
INSERT INTO Staff (staff_id, job_id, stu_fname, stu_lname)
    VALUES (4, 7, 'Harry', 'Potter');
INSERT INTO Staff (staff_id, job_id, stu_fname, stu_lname)
    VALUES (5, 9, 'Christopher', 'Depalma');
INSERT INTO Staff (staff_id, job_id, stu_fname, stu_lname)
    VALUES (6, 11, 'Chuck', 'Bass');
INSERT INTO Staff (staff_id, job_id, stu_fname, stu_lname)
    VALUES (7, 13, 'Elizabeth', 'Brown');
INSERT INTO Staff (staff_id, job_id, stu_fname, stu_lname)
    VALUES (8, 15, 'Greg', 'Demassi');
INSERT INTO Staff (staff_id, job_id, stu_fname, stu_lname)
    VALUES (9, 17, 'Parker', 'Cannon');
INSERT INTO Staff (staff_id, job_id, stu_fname, stu_lname)
    VALUES (10, 19, 'Kaitlin', 'Defranco');
```

Stu_Employee

3NF Justification - This table includes a primary and two foreign keys, no data is repeated although the job_id and emp_id are similar each is necessary. The table doesn't include any partial or transitive dependencies making the table is in at least 3rd normal form.

Table description - This a large table containing a employees full name and status of whether or not they commute or reside on campus. Additionally it is connected to the Staff table to derive who manages each employee.

```sql
--Stu_Employee.sql
--Create and populate Student Employee Table
--Author: Daniel Gisolfi
--DB Management Final Project

CREATE TABLE Stu_Employee(
    emp_id            int            NOT NULL,
    job_id            int            NOT NULL,
    manager_id        int            NOT NULL,
    stu_fname        VARCHAR2(20)    NOT NUll,
    stu_lname        VARCHAR2(20)    NOT NUll,
    stu_status        SET('resident','commuter') NOT NULL,);

ADD CONSTRAINT pk_emp_id PRIMARY KEY (emp_id);
ADD CONSTRAINT fk_job_id_emp FOREIGN KEY (job_id) REFERENCES jobs(job_id);
ADD CONSTRAINT fk_manager_id_emp FOREIGN KEY (manager_id) REFERENCES
Staff(staff_id);

INSERT INTO Stu_Employee (emp_id, job_id, manager_id, stu_fname, stu_lname,
stu_status)
    VALUES (1, 2, 1, 'Ian', 'Smith', 'resident');
INSERT INTO Stu_Employee (emp_id, job_id, manager_id, stu_fname, stu_lname,
stu_status)
    VALUES (2, 4, 2, 'Marco', 'James', 'commuter');
INSERT INTO Stu_Employee (emp_id, job_id, manager_id, stu_fname, stu_lname,
stu_status)
    VALUES (3, 6, 3, 'Erin', 'mills', 'resident');
INSERT INTO Stu_Employee (emp_id, job_id, manager_id, stu_fname, stu_lname,
stu_status)
    VALUES (4, 8, 4, 'Frankie', 'shayman', 'resident');
INSERT INTO Stu_Employee (emp_id, job_id, manager_id, stu_fname, stu_lname,
stu_status)
    VALUES (5, 6, 5, 'Daniel', 'Gisolfi', 'commuter');
INSERT INTO Stu_Employee (emp_id, job_id, manager_id, stu_fname, stu_lname,
stu_status)
    VALUES (6, 10, 6, 'Anthony', 'Diamco', 'resident');
INSERT INTO Stu_Employee (emp_id, job_id, manager_id, stu_fname, stu_lname,
stu_status)
    VALUES (7, 12, 7, 'Brendan', 'Kelly', 'resident');
INSERT INTO Stu_Employee (emp_id, job_id, manager_id, stu_fname, stu_lname,
stu_status)
    VALUES (8, 14, 8, 'Maya', 'James', 'resident');
INSERT INTO Stu_Employee (emp_id, job_id, manager_id, stu_fname, stu_lname,
stu_status)
    VALUES (9, 16, 9, 'Nicole', 'Ferone', 'resident');
```

```
INSERT INTO Stu_Employee (emp_id, job_id, manager_id, stu_fname, stu_lname, stu_status)
    VALUES (10, 18, 10, 'James', 'Corcoran', 'commuter');
INSERT INTO Stu_Employee (emp_id, job_id, manager_id, stu_fname, stu_lname, stu_status)
    VALUES (11, 4, 2, 'Gerald', 'Hawthorne', 'commuter');
```

User_Act

3NF Justification - This table has 2 possible keys either emp_id or staff_id which depends on their job in the jobs table. Additionally, the transitive and partial dependencies have been removed making it a table in 3rd normal form.

Table description - All student employees and staff are users, each with their own account, the username is derived from there first name and the password is created by the user they would be hashed and salted but for now I left them as astricts as there is no need to retrieve them. Additionally, most users have access to at least one drive if not more so the access is stored in this table as well.

```
--User_Act.sql
--Create and populate Users Table
--Author: Daniel Gisolfi
--DB Management Final Project

CREATE TABLE User_Act(
    staff_user_id    int,
    emp_user_id        int,
    drive_access    int,
    user_name        VARCHAR2(20)    NOT NUll,
    user_pass        VARCHAR2(20)    NOT NUll);

CONSTRAINT fk_staff_user_id FOREIGN KEY (staff_user_id) REFERENCES Staff(staff_id);
CONSTRAINT fk_emp_user_id FOREIGN KEY (emp_user_id) REFERENCES Stu_Employee(emp_id);

--input staff
INSERT INTO User_Act (staff_user_id, drive_access, user_name, user_pass)
    VALUES (1, 3, 'Nick', '******');
INSERT INTO User_Act (staff_user_id, drive_access, user_name, user_pass)
    VALUES (2, 6, 'Marty', '******');
INSERT INTO User_Act (staff_user_id, drive_access, user_name, user_pass)
    VALUES (3, 5, 'Katherine', '******');
INSERT INTO User_Act (staff_user_id, drive_access, user_name, user_pass)
    VALUES (4, 7, 'Harry', '******');
```

```
INSERT INTO User_Act (staff_user_id, drive_access, user_name, user_pass)
    VALUES (5, 8, 'Christopher', '******');
INSERT INTO User_Act (staff_user_id, drive_access, user_name, user_pass)
    VALUES (6, 3, 'Chuck', '******');
INSERT INTO User_Act (staff_user_id, drive_access, user_name, user_pass)
    VALUES (7, 9, 'Elizabeth', '******');
INSERT INTO User_Act (staff_user_id, drive_access, user_name, user_pass)
    VALUES (8, 10, 'Greg', '******');
INSERT INTO User_Act (staff_user_id, drive_access, user_name, user_pass)
    VALUES (9, 1, 'Parker', '******');
INSERT INTO User_Act (staff_user_id, drive_access, user_name, user_pass)
    VALUES (10, 5, 'Kaitlin', '******');

--input Employees
INSERT INTO User_Act (emp_user_id, drive_access, user_name, user_pass)
    VALUES (1, 4,  'Ian', '******');
INSERT INTO User_Act (emp_user_id, drive_access, user_name, user_pass)
    VALUES (2, 3, 'Marco', '******');
INSERT INTO User_Act (emp_user_id, drive_access, user_name, user_pass)
    VALUES (3, 6, 'Erin', '******');
INSERT INTO User_Act (emp_user_id, drive_access, user_name, user_pass)
    VALUES (4, 2, 'Frankie', '******');
INSERT INTO User_Act (emp_user_id, drive_access, user_name, user_pass)
    VALUES (5, 8, 'Daniel', '******');
INSERT INTO User_Act (emp_user_id, drive_access, user_name, user_pass)
    VALUES (6, 7, 'Anthony', '******');
INSERT INTO User_Act (emp_user_id, drive_access, user_name, user_pass)
    VALUES (7, 9, 'Brendan', '******');
INSERT INTO User_Act (emp_user_id, drive_access, user_name, user_pass)
    VALUES (8, 3, 'Maya', '******');
INSERT INTO User_Act (emp_user_id, drive_access, user_name, user_pass)
    VALUES (9, 5, 'Nicole', '******');
INSERT INTO User_Act (emp_user_id, drive_access, user_name, user_pass)
    VALUES (10, 7, 'James', '******');
INSERT INTO User_Act (emp_user_id, , user_name, user_pass)
    VALUES (11, 'Gerald', '******');
```

Queries

Query 1

Name every job that has an employee who works more than 10 hours

```sql
SELECT jobs.job_name
FROM jobs
WHERE NOT EXISTS
    (SELECT *
    FROM Stu_Employee
    WHERE Jobs.job_id = Stu_Employee.job_id
    AND NOT EXISTS
        (SELECT *
        FROM Shifts
        WHERE  Stu_Employee.emp_id = Shifts.emp_id
        AND   Shifts.Requested_worked > 10));
```

Result

	JOB_NAME
1	Desktop Admin
2	Desktop Employee
3	Resnet Admin
4	Help Desk Admin
5	Help Desk Employee
6	Web Services Admin
7	Applications Admin
8	Applications Employee
9	Digital Education Admin
10	Digital Education Employee
11	Networking Admin
12	Networking Employee
13	Telecom Admin
14	Card Services Admin

Cardinality = 14

Query 2

Get the last name of students who only work 17 hours

```sql
SELECT Stu_Employee.stu_lname
FROM Stu_Employee
WHERE EXISTS
    (SELECT *
    FROM Shifts
    WHERE Shifts.requested_hours = 17
    AND Shifts.emp_id = Stu_Employee.emp_id);
```

Result

	STU_LNAME
1	Smith
2	Diamco
3	Hawthorne

Cardinality = 3

Query 3

Name the students who do not have access to a drive

```
SELECT Stu_Employee.stu_lname
FROM Stu_Employee
WHERE EXISTS
    (SELECT *
    FROM User_act
    WHERE drive_access IS NULL
    AND Stu_Employee.emp_id = User_act.emp_user_id);
```

Result

	STU_LNAME
1	Hawthorne

Cardinality = 1

Query 4

Get the names of student employees and the projects they work on if any

```
SELECT DISTINCT Stu_Employee.stu_lname , Projects.prj_name
FROM Stu_Employee LEFT JOIN Projects
ON Stu_Employee.emp_id = Projects.emp_id;
```

Result

	STU_LNAME	PRJ_NAME
1	mills	Request Organazation
2	Diamco	(null)
3	Smith	Device Deployemnt
4	Gisolfi	(null)
5	Hawthorne	(null)
6	shayman	(null)
7	Kelly	(null)
8	James	Device Repairs
9	James	(null)
10	Ferone	(null)
11	Corcoran	(null)

Cardinality = 11

Query 5

Get all Drive Access numbers and the user's that have access to them if any

```
SELECT DISTINCT Stu_Employee.stu_lname, User_act.drive_access
FROM Stu_Employee RIGHT JOIN User_act
ON Stu_Employee.emp_id = User_act.emp_user_id;
```

Result

	STU_LNAME	DRIVE_ACCESS
1	shayman	2
2	Gisolfi	8
3	Diamco	7
4	(null)	7
5	(null)	6
6	James	3
7	Kelly	9
8	(null)	5
9	(null)	8
10	Hawthorne	(null)
11	(null)	1
12	(null)	3
13	Smith	4
14	mills	6
15	Ferone	5
16	Corcoran	7
17	(null)	10
18	(null)	9

Cardinality =18

Query 6

Get the names of all people with jobs at Marist separated by if they are a student employee or staff

```
SELECT DISTINCT Stu_Employee.stu_lname, Staff.staff_lname
FROM Stu_Employee FULL OUTER JOIN Staff
ON Stu_Employee.job_id = Staff.job_id;
```

Result

	STU_LNAME	STAFF_LNAME
1	(null)	Jacobs
2	Diamco	(null)
3	(null)	Bass
4	(null)	Demassi
5	(null)	Cannon
6	(null)	Smith
7	Gisolfi	(null)
8	Hawthorne	(null)
9	(null)	Brown
10	(null)	Defranco
11	Kelly	(null)
12	shayman	(null)
13	(null)	Philips
14	(null)	Depalma
15	Smith	(null)
16	Corcoran	(null)
17	James	(null)
18	Ferone	(null)
19	(null)	Potter
20	mills	(null)

Cardinality = 20

Query 7

Get the Department, Job, salary, user_name and device type deployed to them for all staff with an ID grater than 5

```
SELECT Departments.dep_name, jobs.job_name, Amends.salary,
User_act.user_name, devices.dev_type
FROM Departments, jobs, Amends, staff, User_act, Devices
WHERE Departments.dep_id = jobs.dep_id
AND jobs.job_id = Amends.job_id
AND jobs.job_id = Staff.job_id
AND Staff.staff_id = User_act.staff_user_id
AND Devices.staff_id = staff.staff_id
AND Staff.staff_id > 5;
```

Result

	DEP_NAME	JOB_NAME	SALARY	USER_NAME	DEV_TYPE
1	Digital Education	Digital Education Admin	60000	Chuck	Laptop
2	Networking	Networking Admin	60000	Elizabeth	Laptop
3	Telecom	Telecom Admin	60000	Greg	Desktop
4	Card Services	Card Services Admin	60000	Parker	Desktop

Cardinality = 4

Query 8

Get the number of hours still needed to be worked by an employee to reach requested hours as well as there name

```
SELECT Stu_Employee.stu_lname, SUM(Shifts.requested_hours -
Shifts.hours_worked)
FROM Stu_Employee, Shifts
WHERE Shifts.hours_worked < Shifts.requested_hours
AND Stu_Employee.emp_id = Shifts.emp_id
GROUP BY Stu_Employee.stu_lname;
```

Result

	STU_LNAME	HOURSLEFTTOWORK
1	Gisolfi	5
2	Smith	2
3	mills	1
4	shayman	8
5	Diamco	4
6	Kelly	2
7	James	3
8	Ferone	6
9	Corcoran	4
10	Hawthorne	10

Cardinality = 10

Query 9

Show all staff and employees and what their jobs are

```sql
SELECT Jobs.Job_name, Stu_Employee.Stu_lname
FROM Jobs, Stu_Employee
WHERE Jobs.job_id = Stu_Employee.job_id

UNION

SELECT Jobs.job_name, Staff.Staff_lname
FROM Jobs, Staff
WHERE Jobs.job_id = Staff.job_id;
```

Result

	JOB_NAME	STU_LNAME
1	Applications Admin	Depalma
2	Applications Employee	Diamco
3	Card Services Admin	Cannon
4	Card Services Employee	Corcoran
5	Desktop Admin	Smith
6	Desktop Employee	Smith
7	Digital Education Admin	Bass
8	Digital Education Employee	Kelly
9	Help Desk Admin	Jacobs
10	Help Desk Employee	Gisolfi
11	Help Desk Employee	mills
12	Networking Admin	Brown
13	Networking Employee	James
14	Resnet Admin	Philips
15	Resnet Employee	Hawthorne
16	Resnet Employee	James
17	Telecom Admin	Demassi
18	Telecom Employee	Ferone
19	Web Services Admin	Potter
20	Web Services Employee	shayman

Cardinality = 20

Query 10

Give the last name and room number of where all IT Members work

```sql
SELECT Departments.dep_room, Staff.staff_lname LastName
FROM Departments, Jobs, Staff
WHERE Departments.dep_id = Jobs.dep_id
AND Jobs.job_id = Staff.job_id

UNION


SELECT Departments.dep_room, Stu_Employee.stu_lname LastName
FROM Departments, Jobs, Stu_Employee
WHERE Departments.dep_id = Jobs.dep_id
AND Jobs.job_id = Stu_Employee.job_id
Order By LastName;
```

Result

	DEP_ROOM	LASTNAME
1	260	Bass
2	110	Brown
3	110	Cannon
4	110	Corcoran
5	110	Demassi
6	260	Depalma
7	260	Diamco
8	110	Ferone
9	258	Gisolfi
10	101	Hawthorne
11	258	Jacobs
12	101	James
13	110	James
14	260	Kelly
15	101	Philips
16	310	Potter
17	101	Smith
18	258	mills
19	310	shayman

Cardinality = 19