Personal Health Management

Project Report

Team 6

Abinav Pothuganti - apothug@ncsu.edu

Sai Sri Harsha Kunapareddy - skunapa@ncsu.edu

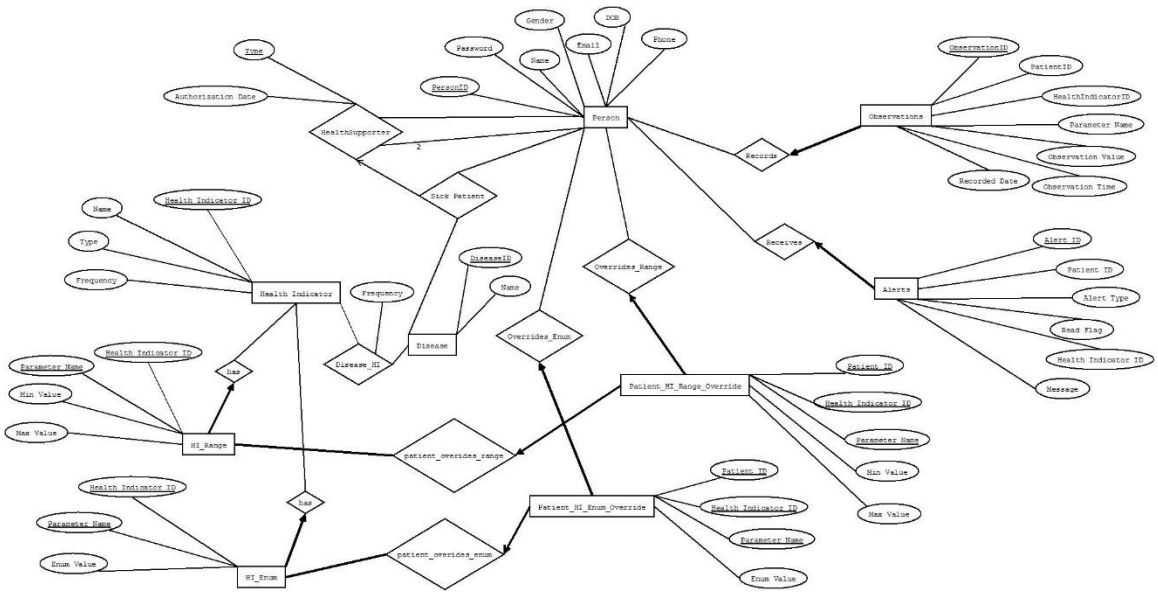
Shiv Shankar Barai - sbarai@ncsu.edu

Venkata Sai Prashanth Rallapalli - vrallap@ncsu.edu

1. Problem Statement

The purpose of this application is to organize the data to assist people in tracking health information. The application lets two types of users – Patients and Health supporters to track observations regarding the patient's health indicators and generate alerts when something is not usual. Several functionalities are provided for registered patients and health supporters such as adding new diseases for patients, adding new health supporters, adding health indicators to be tracked for each disease, manually over riding the range of health indicators, generating alerts when the observations are found to be in violation etc. Maintaining all this information manually is a tedious task and difficult to track for patients and health supporters. Hence this application is useful to automatically notify patients and health supporters of the current health indicators of the patient and take action appropriately. While patients have access only to their information and can modify anything that is related to them, health supporters can view multiple patient information related to them and can only modify some of the patient's details.

2. E-R Diagram



3. Entities and Relationships

3.1 Entities

1. **Person:** Persons have a unique id and password, using which they login into the Health system. Both Patients and Health Supporters are together categorized as Persons. Although most of their common details are stored in Persons, Patients and Health Supporters have different privileges and hence are divided into different relations for persons.
2. **Disease:** Diseases are uniquely identified by a disease id. This entity is related to persons and health indicators and can be used to get the disease name to show patients or health supporters.
3. **Health Indicators:** Health indicators are uniquely identified by id. Health indicators are common attributes for all patients including well patients. But they are also specific to the type of disease a patient has. Some health indicators have range values while others have enumeration values.
4. **Health Indicators - Range:** Health indicators which have range values. These are uniquely identified by health indicator id. They have minimum and maximum recommended values.
5. **Health Indicators - Enumeration:** Health indicators which have enumeration values. These are uniquely identified by health indicator id.
6. **Health Indicators Range Patient Override:** If a patient overrides a health indicator with range values the corresponding changes are done in this table. Otherwise this table contains default recommended values.
7. **Health Indicators Enumeration Patient Override:** If a patient overrides a health indicator with enumeration values the corresponding changes are done in this table. Otherwise this table contains default recommended values.
8. **Observations:** Patients are required or recommended to take checkup and note down the observation values depending on the disease they have. For a well patient, the observation values are general recommendations. Each observation is uniquely identified by an id and also references person id and health indicator id for which the observation is done.
9. **Alerts:** Alerts are generated if the observations are not done by the recommend frequency. Each alert has an alert id by which it is identified and the patient id for which the alert is generated. Alerts can be cleared by patients by taking the observations or by health supporters after viewing them.

3.2 Relationships

Diseases for patient: Each sick patient has at least one disease. The patient id and disease id mapping is stored in the patient table.

Health Supporter: Health Supporters can have more than one patients related to them. And they can be authorized to view a patient information for a future date as well. They are identified by their id in the person table which acts as a foreign key.

Patient: In addition to the attributes present in Person, each Patient has at least one disease associated with him/her. So, this entity has a mapping of person ids and the id of the disease associated.

Health Supports for patient: Each sick patient has at least one and at most two health supporters and each health supporter can have multiple patients associated with him / her.

Disease health indicators (Range): Each disease is associated with prominent health indicators that need to be in specific range for the disease to be kept in check.

Disease health indicators (Enum): Some health indicators have enum values and these are also associated with diseases.

HI to Patient Override (Range & Enum): The patient override tables contain default HI values if no overriding is done.

Person to Patient Override (Range & Enum): A person can change health indicator recommended values.

Observations for patients: Each patient needs to have observations taken regularly and the frequency and the health indicators for which the observations need to be taken depend on the disease that he has.

Alerts for patients: Alerts are generated when an observation is missing for a patient or the observation does not lie in the desired range.

4. Relational Model:

```
Person(
 pid integer PRIMARY KEY,
 p_password varchar2(30) NOT NULL,
 p_name varchar2(50) NOT NULL,
 dob date NOT NULL,
 gender varchar2(1) NOT NULL,
```

```
  address varchar2(100),
  email varchar2(30) NOT NULL,
  phone number(10) NOT NULL,
  CHECK (gender IN ('M', 'F'))
);

Disease(
        did integer PRIMARY KEY,
        dname varchar2(20) NOT NULL,
        UNIQUE(dname)
);

Patient(
  pid integer,
  did integer,
  PRIMARY KEY(pid,did),
  FOREIGN KEY(did) REFERENCES Disease(did) ON DELETE CASCADE,
  FOREIGN KEY(pid) REFERENCES Person(pid) ON DELETE CASCADE
);

HealthSupporters(
  pid integer,
  hs_pid integer,
  hs_type varchar(10),
  auth_date date NOT NULL,
  PRIMARY KEY(pid,hs_type),
  UNIQUE(pid, hs_pid),
  FOREIGN KEY (pid) REFERENCES Person(pid) ON DELETE CASCADE,
  FOREIGN KEY (hs_pid) REFERENCES Person(pid) ON DELETE CASCADE,
  CHECK (hs_type IN ('PRIMARY', 'SECONDARY'))
);

HealthIndicators(
  hid integer PRIMARY KEY,
  hname varchar2(20) NOT NULL,
  hi_type varchar2(10) NOT NULL,
  frequency integer,
  CHECK (hi_type IN ('RANGE', 'ENUM'))
);

Disease_HI(
  did integer,
  hid integer,
  frequency integer,
```

```
  PRIMARY KEY(did,hid),
  FOREIGN KEY(did) REFERENCES Disease(did) ON DELETE CASCADE,
  FOREIGN KEY(hid) REFERENCES HealthIndicators(hid)
);

HI_RangeValues(
  hid integer,
  param_name varchar2(20) NOT NULL,
  min_value integer,
  max_value integer,
  PRIMARY KEY (hid,param_name),
  FOREIGN KEY(hid) REFERENCES HealthIndicators(hid),
  CHECK(max_value >= min_value)
);

HI_EnumValues(
  hid integer,
  param_name varchar2(20) NOT NULL,
  enum varchar2(20),
  PRIMARY KEY (hid, param_name),
  FOREIGN KEY(hid) REFERENCES HealthIndicators(hid)
  CHECK (enum IN ('Happy','Sad','1','2','3','4','5','6','7','8','9','10'))
);

Patient_HIRangeValues_Override(
  pid integer,
  hid integer,
  param_name varchar2(20),
  min_value integer,
  max_value integer,
  PRIMARY KEY (pid,hid,param_name),
  FOREIGN KEY(hid, param_name) REFERENCES HI_RangeValues(hid, param_name),
  FOREIGN KEY(pid) REFERENCES Person(pid) ON DELETE CASCADE,
  CHECK(max_value >= min_value)
);

Patient_HIEnumValues_Override(
  pid integer,
  hid integer,
  param_name varchar2(20),
  enum varchar2(20),
  PRIMARY KEY (pid,hid),
  FOREIGN KEY(pid) REFERENCES Person(pid) ON DELETE CASCADE
  FOREIGN KEY(hid, param_name) REFERENCES HI_EnumValues(hid, param_name),
```

```
  CHECK (enum IN ('Happy','Sad','1','2','3','4','5','6','7','8','9','10'))
);

Observations(
  observation_id integer PRIMARY KEY,
  pid integer,
  hid integer,
  param_name varchar2(20),
  observation_value varchar2(20) NOT NULL,
  observation_time date NOT NULL,
  recorded_date date  DEFAULT sysdate NOT NULL,
  FOREIGN KEY (pid) REFERENCES Person(pid) ON DELETE CASCADE,
);

Alerts(
  aid integer PRIMARY KEY,
  pid integer,
  alert_type varchar2(20) NOT NULL,
  read_flag varchar2(1) NOT NULL,
  FOREIGN KEY (pid) REFERENCES Person(pid) ON DELETE CASCADE,
  CHECK(alert_type IN ('OUTSIDE_LIMIT', 'LOW_ACTIVITY'))
);
```

5. Constraints

- Gender of a person should be Male (M) or Female (F).
- A health supporter should be one of the two types: Primary or Secondary.
- A health indicator can have range values or enum values depending on which their respective values will be stored in the range table or enum table.
- For a health indicator whose values are of range type the maximum value of the indicator must be greater than or equal to the minimum value.
- Alerts that are generated can be of two types: OUTSIDE_LIMIT when the observed values for the patient are outside the range that is recommended and LOW_ACTIVITY when the patient does not take observations with in the recommended frequency.
- For a patient to add a health indicator, he should already be present in the system as a person.
- All attributes for a person are mandatory except the address.
- Disease name is mandatory for a disease.
- Authorization date is mandatory for a health indicator as it indicates the date from which the patient's data can viewed.
- For a sick patient to be registered he needs to be in the system and also the disease he has should be present in the system.

- Health indicator name and type is mandatory.
- In the Patient override tables for health indicators, each patient will initially have the range and enum values of the health indicator reference tables depending on the disease.

6. Functional Dependencies and Normal Forms:

1. PERSON
   pid → pid, p_password, p_name, dob, gender, address, email, phone

2. DISEASE
   did → did, dname

3. HEALTH SUPPORTERS
   pid → pid, hs_pid, hs_type, auth_date

4. PATIENT
   pid, did → pid, did

5. ALERTS
   aid → aid, pid, alert_type, hid, read_flag, message

6. OBSERVATIONS
   observation_id → observation_id, pid, hid, param_name, observation_value, observation_time, recoded_date

7. HEALTH INDICATORS
   hid → hid, hname,frequency,hi_type

8. HI_RANGEVALUES
   hid → hid, param_name, min_value, max_value

9. HI_ENUMVALUES
   hid → hid, param_name, enum

10. PATIENT_HIENUMVALUES_OVERRIDE
    pid, hid → pid, hid, param_name, enum

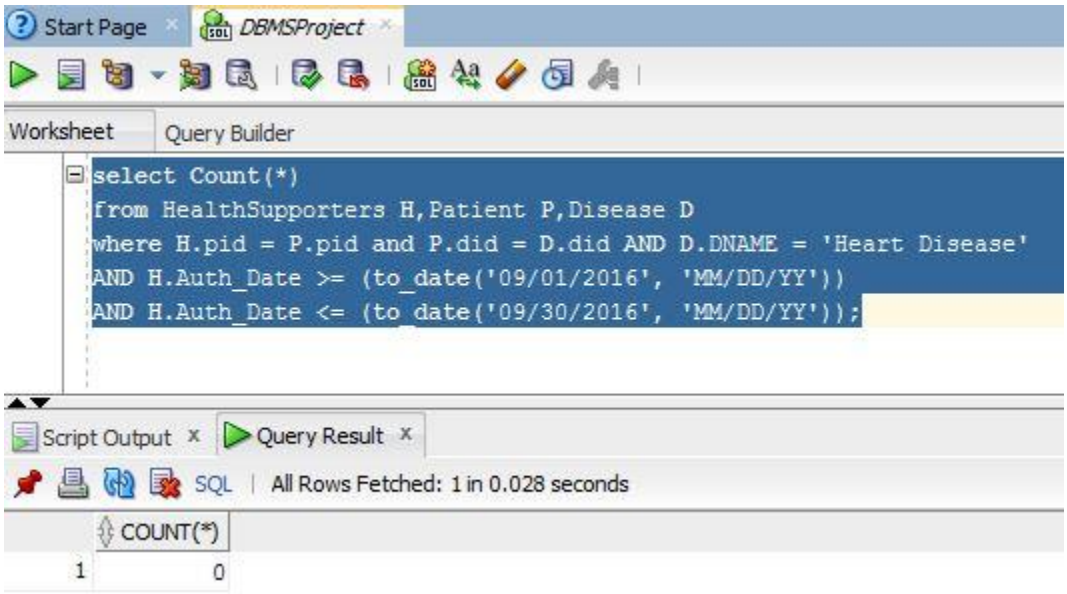
11. PATIENT_HIRANGEVALUES_OVERRIDE
    pid, hid, param_name → pid, hid, param_name, min_value, max_value

12. DIESEASE_HI
    pid, hid → pid, hid, frequency

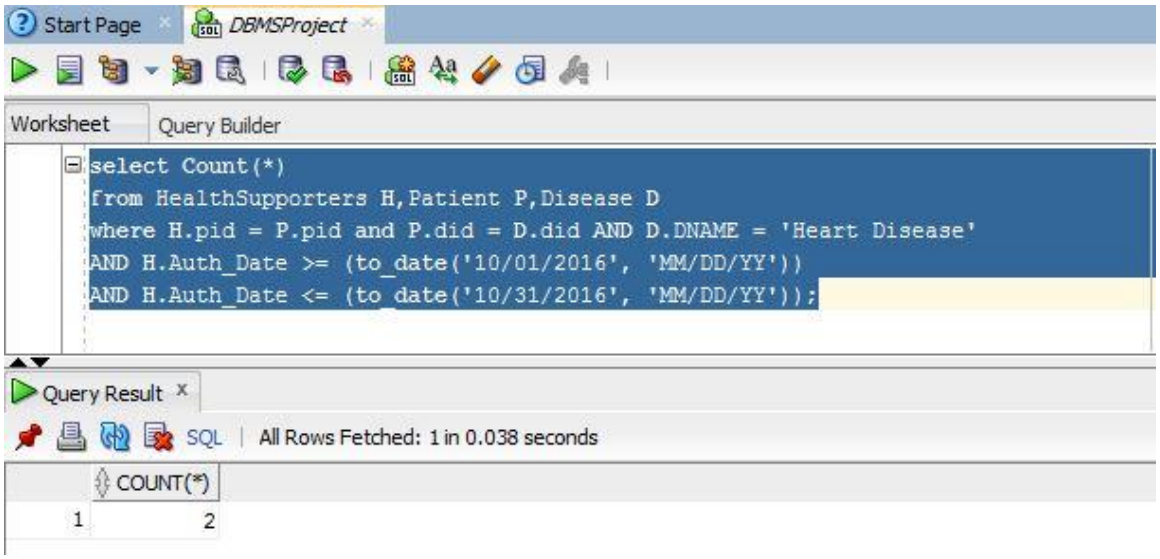
7. SQL queries and the result Snapshots

1. List the number of health supporters that were authorized in the month of September 2016 by patients suffering from heart disease.

```
select Count(*)
from HealthSupporters H,Patient P,Disease D
where H.pid = P.pid and P.did = D.did AND D.DNAME = 'Heart Disease' AND
H.Auth_Date >= (to_date('09/01/2016', 'MM/DD/YY')) AND H.Auth_Date <=
(to_date('09/30/2016', 'MM/DD/YY'));
```

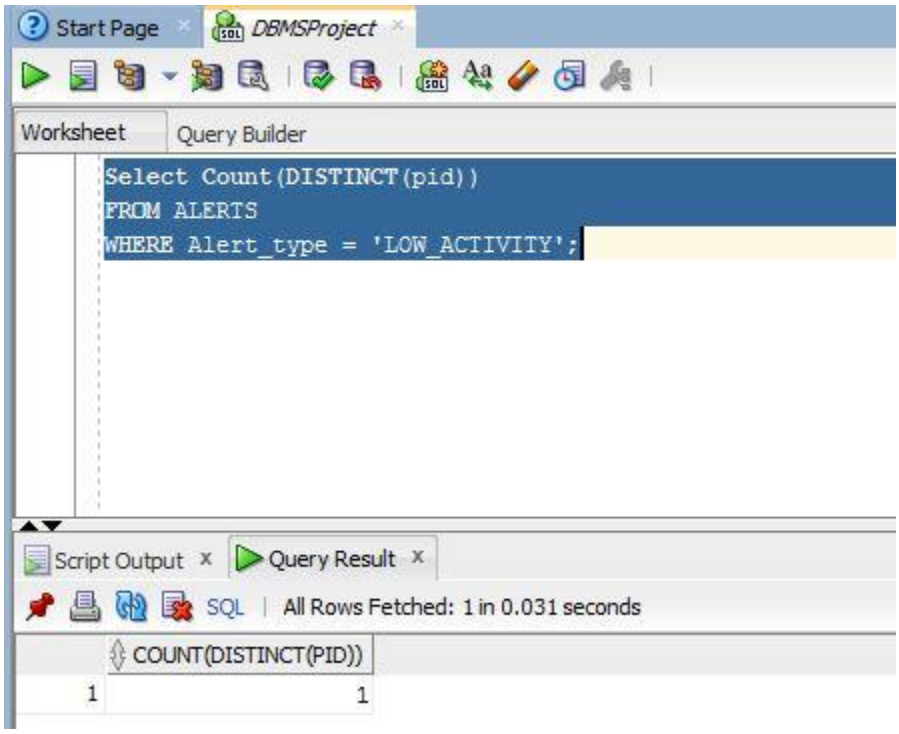


If the query is for October we get 2 health supporters as the result. Both of the health supporters are of Patient 1. Following is the screenshot for October.



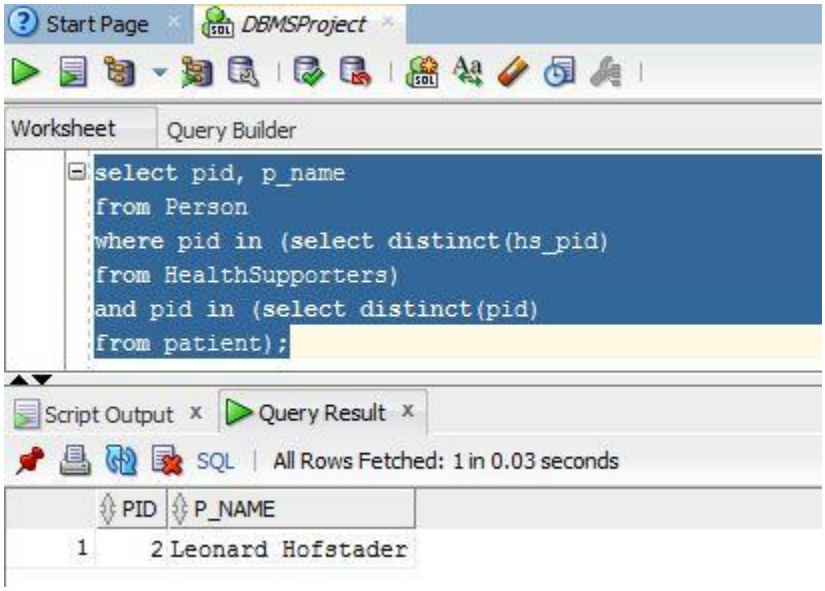
2. Give the number of patients who were not complying with the recommended frequency of recording observations.

```
Select Count(DISTINCT(pid))
FROM ALERTS
WHERE Alert_type = 'LOW_ACTIVITY';
```



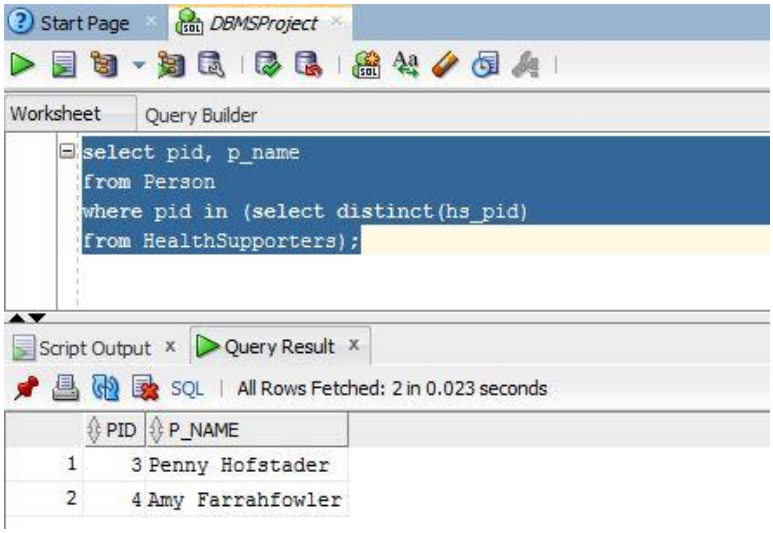
3. List the health supporters who themselves are patients.

```
select pid, p_name
from Person
where pid in (select distinct(hs_pid) from HealthSupporters)
and pid in (select distinct(pid) from patient);
```



4. List the patients who are not 'sick'.

```
Select *
from Person
where pid NOT IN (select pid from Patient);
```



5. How many patients have different observation time and recording time (of the observation).

```
Select count(distinct(pid))
from Observations
where Observation_time != Recorded_Date;
```