

UNIVERSITY OF HERTFORDSHIRE

ADVANCED DATABASES

ASSINGMENT TITLE: WINTERBOURNE NURSING HOME

MODULE CODE: 7COM1022

TUTOR: DR BERNADETTE-MARIE BYRNE

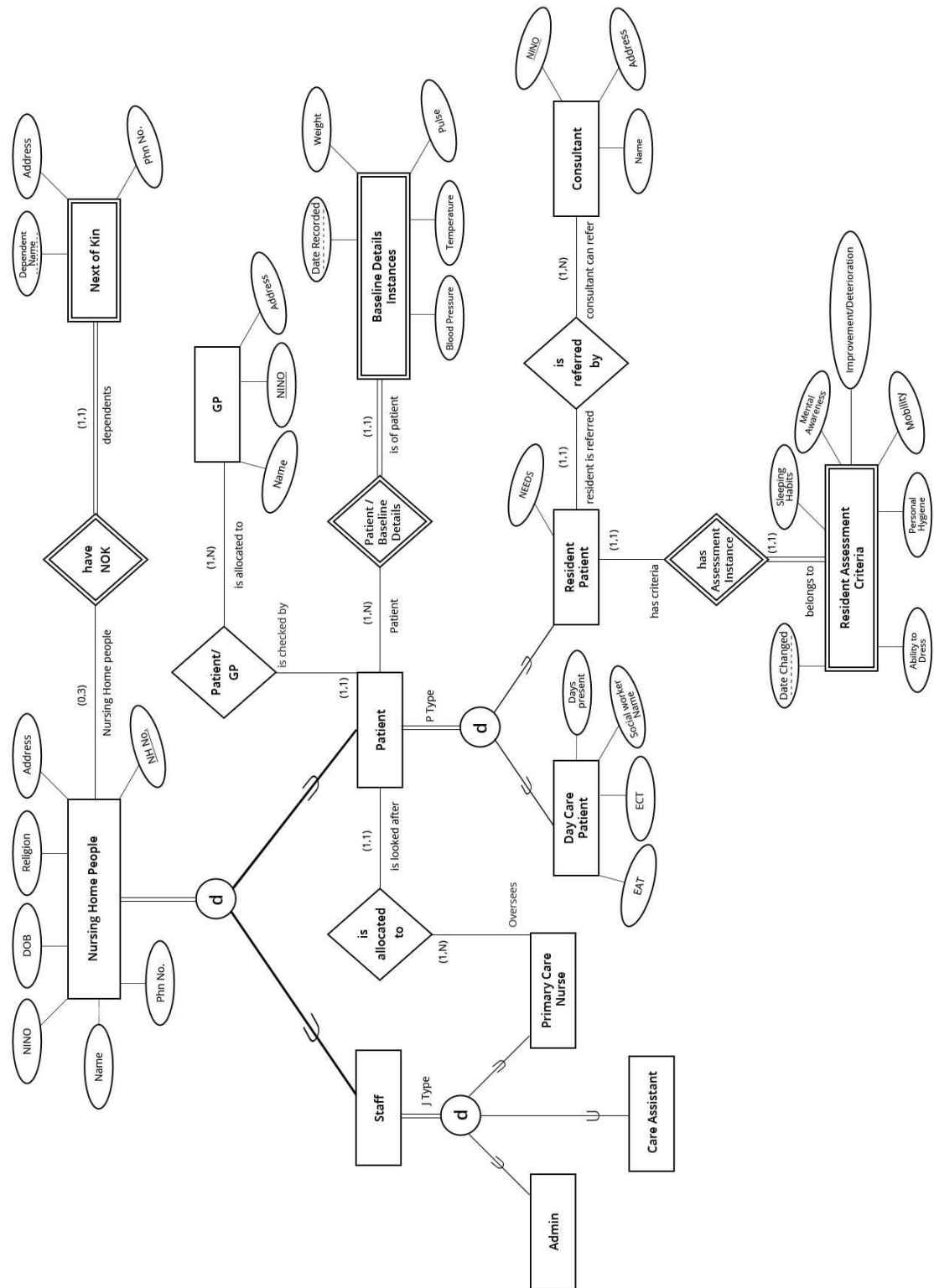
Connection Name: Prabesh Connection

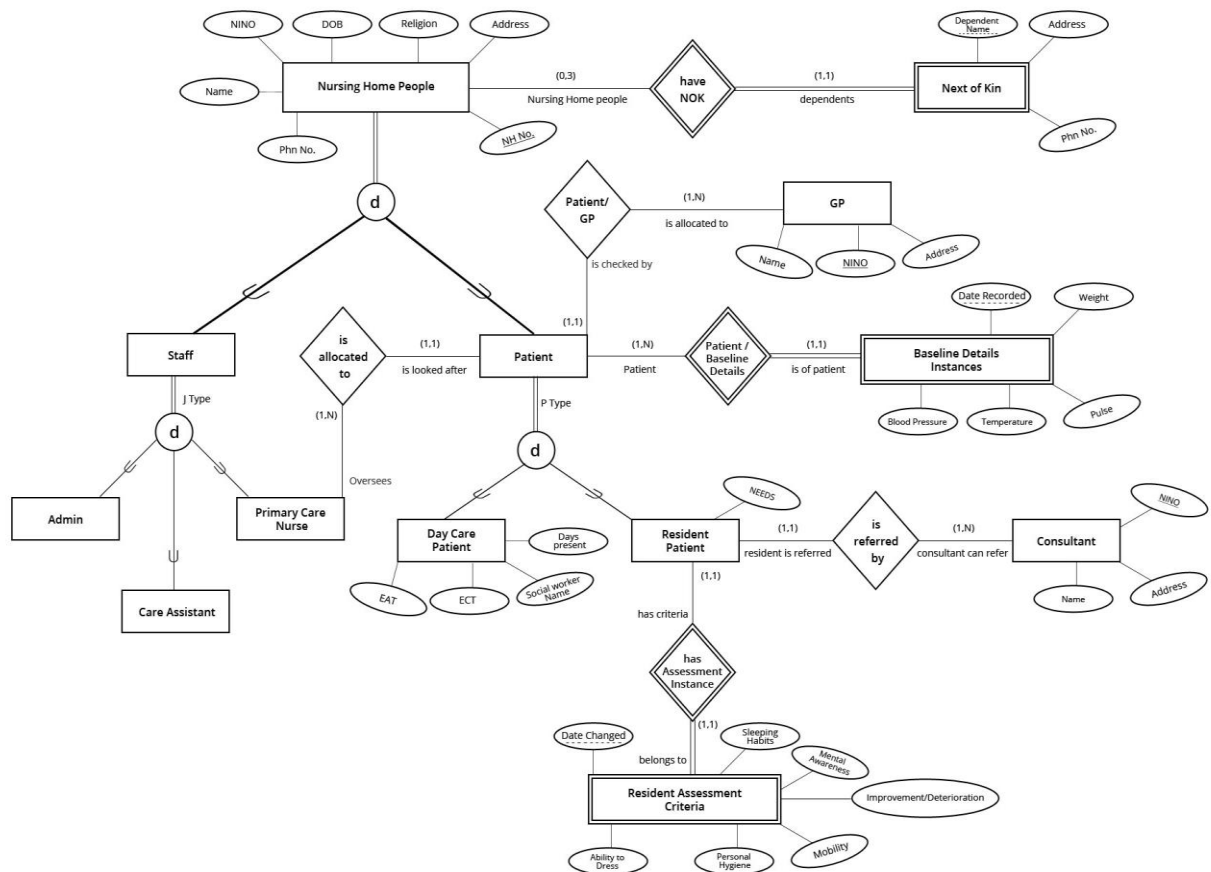
OMKAR KANDEL PRABESH 14185367

PREETI KANDWAL 14192459

SUSAN MOOL 14188699

Winterbourne Nursing Home EER Diagram:





Part One: Description/ Assumptions of Winterbourne Nursing Home Enhanced Entity Relationship Diagram:

In order to systematically represent the entities, specialization is used where the class Nursing Home People is divided into subclasses namely: Staff and Patient. The staff sub class is further divided into three subclasses: Administrative Staff (Admin), Care Assistant and Primary Care Nurse. Here J Type is the defining attribute of the specialization whereas P Type is the defining attribute in Patient which is divided into Resident Patient and Day Care Patient. Every member of the Nursing Home has a unique identifier namely, Nursing Home Number. Each specialization in the diagram represents total disjoint participation as no member of the subclasses fall in more than one subclass and the division into the sub classes is also exhaustive i.e. each member of the class has to belong to one of the subclasses.

The reason for specializing is because the subclasses have specific attributes and although it might complicate queries, it is quite likely that null values in many tuples will be generated if specialization in relations is not done. Avoiding this constitutes a proper database design.

Each Primary Care Nurse is assumed to oversee many patients while each patient is allocated only one Primary Care Nurse.

Each member of the Nursing Home has Next of Kin. Here we have assumed that at the least they don't have any and at maximum they will have 3.

Resident Patients have one specific attribute named Needs which Day Care Patients don't have. Day Care Patients also have Expected Arrival Time (EAT) and Expected Collection Time (ECT) and Days present signifies the days on which they are present at the Nursing Home.

Every patient also has his/her baseline details which have to be recorded every month. A weak entity is established for this because without patient, this entity does not hold any independent meaning. Similarly we also have Resident Assessment Criteria for each patient.

Both consultants and GPs are uniquely identified by their National Insurance Number (NINo). A consultant can refer many resident patients (only) while each resident patient is only consulted by one Consultant. Also, a single GP can supervise many patients but each patient is registered only to one GP.

Part Two: Set of Relations:

- Staff (NHNo, NINo, Name, Phone Number, Religion, DOB, J Type, Address)

This relation lists the attributes of the staff sub class of the Nursing Home People class. NHNo (Nursing Home Number) is the primary key which uniquely identifies each and every member of this relation. Attributes that are required are included. One step of mapping specialization is to include the defining attribute as an attribute which J Type does here. It specifies what the job type is for the staff members.

- Patient (P_NHNo, NINo, Name, Phone Number, Religion, DOB, PType, *PCNNHNo**, *GP NINo**)

Here the list of all common attributes are included for both patient types. The PType attribute specifies the type of patient in the Nursing Home.

- Resident_Patient (RP_NHNo, *CONINo**, Needs)

The above relation is for the Resident Patients who have their own fields. In this case we have the Nursing Home Number of Resident Patient (RP_NHNo) as the unique identifier (the primary key which is also included in the Patient Table), and National Insurance Number of Consultants (CO Nino) as foreign key which match the resident patients to the respective consultant. RP_NHNo has been used instead of NHNo to make it more meaningful and as is CO NINo. The needs attribute is a field where we can store the special needs that are evaluated/ recognised by the nurses.

- Day_Care_Patient(DCP_NHNo, Expected Arrival Time, Expected Collection Time, Social Worker Name)

Similar is the case with Day Care Patient who have their own fields relevant to their characteristics. Needs are not included (as only residents are privileged for that). For day care patients, three additional attributes Expected Arrival Time (EAT), Expected Collection Time (ECT) and Social Worker Name assigned to them are included.

- Consultant (CO NINo, Name, Address)

- GP (GP NINo, Name, Address)

Consultants and GPs have their respective NINos as primary keys.

- Next Of Kin (NHNo*, Name, Address, Phone Number)

Next Of Kin is a weak entity which takes NHNo and Name (of the kin) together as the primary key. All individuals of the Nursing Home have this.

- Baseline_Details_Instances (NHNo*, Date Recorded, Weight, Pulse, Temperature, Blood Pressure)

This is also a weak entity. Here the date on which the baseline details are recorded and the NHNo act as the primary key. The NHNo can only be of either of the two patients. Each date and NHNo uniquely identifies the baseline details which are taken every month. All the instances are recorded.

- Resident_Assessment_Criteria (RP NHNo*, Date Changed, Improvement/Deterioration, Personal Hygiene, Mobility, Ability to Dress, Mental Awareness, Sleeping Habits)

We are also required to update the assessment details of patients depending on whether their condition has improved or worsened. The attributes (characteristics of these very details) along with the date these records were updated and why; either because of improvement or deterioration in conditions is also recorded.

- DCP_days (DCP NHNo*, Days Present)

To avoid multi valued attributes which is prohibited by 1NF DCP_Days relation is created for day care patients where the record of days on which they are at the Nursing Home is kept. A day care patient can be at the nursing home more than one day a week.

PART THREE: SQLCreate Table Statements:

--SQL for staff table creation

create table staff

(STAFF_nh# varchar2(12) not null,

staff_nin varchar2(12) not null,

staff_name varchar2(32) not null,

staff_jobtype varchar2(32) not null,

Staff_DATEOFBIRTH DATE not null,

Staff_RELIGION VARCHAR2(32) not null,

staff_address varchar2(64) not null,

STAFF_ph# number(12) not null,

constraint PK primary key(STAFF_nh#)

)

/

--SQL for patient table creation

create table patient

(p_nh# varchar2(12) not null,

p_name varchar2(32) not null,

p_nin varchar2(12) not null,

p_ph# number(12) not null,

p_dob DATE not null,

p_RELIGION VARCHAR2(32) not null,

Ptype VARCHAR2 (12),

```
p_gp_nin varchar2(12),
```

```
constraint p_pk primary key(p_nh#)
```

```
)
```

```
/
```

```
--SQL for daycare patient table creation
```

```
create table daycare_patient
```

```
(dcp_nh# varchar2(12) not null,
```

```
DCP_eat      varchar2 (8) not null,
```

```
DCP_ect varchar2 (8) not null,
```

```
DCP_SOCIAL_WORKER_NAME VARCHAR2(20),
```

```
constraint dcp_pk primary key(dcp_nh#),
```

```
constraint dcp_fk6 foreign key(dcp_nh#) references patient(p_nh#)
```

```
)
```

```
/
```

```
--SQL for daycare patient days table creation
```

```
create table dcp_days
```

```
(dcpw_nh# varchar2(12) not null,
```

```
dcpw_dayspresent varchar2(32) not null,
```

```
constraint dcpw_pk primary key (dcpw_nh#,dcpw_dayspresent),
```

```
constraint dcpw_fk7 foreign key ( dcpw_nh#) references daycare_patient(dcp_nh#)
```

```
)
```

```
/
```


--SQL for resident patient table creation

create table resident_patient

(rp_nh# varchar2(12) not null,

rp_CONSULTANT_NINovarchar2(12),

rp_needsvarchar2(64),

constraint rp_pk primary key(rp_nh#),

constraint rp_fk1 foreign key(rp_consultant_nino) references

consultant(c_CONSULTANT_NINo),

constraint rp_fk2 foreign key(p_gp_nino) references gp(gp_nino),

constraint rp_fk3 foreign key(p_pcn_nh#) references staff(staff_nh#),

constraint rp_fk4 foreign key (rp_nh#) references patient (p_nh#)

)

/

--SQL for Next of Kin (for both patient and staff) table creation

create table next_of_kin

(nok_nh# varchar2(12) not null,

nok_namevarchar2(32) not null,

nok_addressvarchar2(64),

nok_ph# number(12),

constraint nok_pk primary key(nok_nh#,nok_name),

constraint nok_fk2 foreign key (nok_nh#) references patient(p_nh#)

)

/

--SQL for baseline details Instances (all Patients) table creation

create table baseline_details_instances

(BdI_NH# varchar2(12) not null,

BdI_DATErecorded DATE not null,

BdI_PATIENT_WEIGHT VARCHAR2(16) not null,

BdI_PATIENT_PULSE VARCHAR2(16) not null,

BdI_PATIENT_TEMPERATURE VARCHAR2(16) not null,

BdI_PATIENT_BLOODPRESSURE VARCHAR2(16) not null,

constraintbdi_pk primary key(bdi_nh#,bdi_daterecorded),

constraint bdi_fk1 foreign key (bdi_nh#) references patient(p_nh#)

)

/

--SQL for resident assessment criteria table creation

create table resident_assessment_criteria

(RAC_NH# varchar2(12) not null,

RAC_DATECHANGED DATE not null,

RAC_SLEEPING_HABITS VARCHAR2(32) not null,

RAC_MENTAL_AWARENESS NUMBER(1) not null,

RAC_ABILITY_TO_DRESS VARCHAR2(32) not null,

RAC_MOBILITY NUMBER(1) not null,

RAC_PERSONAL_HYGIENE NUMBER(1) not null,

RAC_IMPROVED_or_deteriorated varchar(16),

constraintRAC_pk primary key(RAC_nh#, RAC_datechanged)

constraint rac_fk1 foreign key (rac_nh#) references resident_patient (rp_nh#)

)

/

--SQL for consultant table creation

create table consultant

(c_CONSULTANT_NINo varchar2(12) not null,

c_CONSULTANT_name VARCHAR2(32) not null,

constraintc_pk primary key(c_CONSULTANT_NINo)

)

/

--SQL for GP table creation

create table gp

(GP_Ninovarchar2(12) not null,

gp_nameVARCHAR2(20) not null,

GP_Address VARCHAR2(20) not null,

constraintgp_pk primary key(GP_Nino)

)

/

- a) Produce a list of the current residential patients at the hospital.

SQL Query 1

```
select * from patient  
  
where Ptype= 'Resident';
```

- b) A patient with the name James Gump has gone missing during his daily walk.
We need a list of his next-of-kin with telephone numbers.

SQL Query 2

```
select p_name,NOK_NAME, nok_ph#  
  
From Next_of_Kin, patient  
  
where nok_nh#= p_nh# AND P_Name= 'James Gump';
```

- c) We now need a report showing Mr. Gump's Base Line details from January
this year to the present day.

SQL Query 3

```
select p_name,bdi_nh#, bdi_daterecorded,  
BDI_PATIENT_BLOODPRESSURE,BDI_PATIENT_PULSE,BDI_PATIENT_TEMPERATURE,BDI_PATIENT_WEIGHT  
  
From Baseline_details_instances, PATIENT  
  
where  
  
bdi_nh# = p_nh# AND  
  
P_Name= 'James Gump' AND  
  
BDI_daterecorded between '01-Jan-15' and '08-Dec-15';
```

- d) Produce a report showing day care patients, the primary care nurse they have
been allocated and the patient's GP

SQL Query 4

```
select p_name, staff_name, gp_name, staff_jobtype, PTYPE
from PATIENT, GP,STAFF
where
p_pcn_nh#= staff_nh# AND p_gp_Nino= gp_nino AND PType= 'Day Care' AND
Staff_JobType= 'Primary Care Nurse';
```

- e) How many patients had mental awareness scores of 1 or 2 at yesterday's date and are allocated to a GP called Dr J Shipman.

SQL Query 5

```
select COUNT (*)
from PATIENT, resident_assessment_criteria
where p_nh#=rac_nh# AND
rac_mental_awareness= '1' or rac_mental_awareness='2' AND
RAC_DATECHANGED= '07-Dec-15' and p_NH# IN
      (SELECT P_NH#
       FROM PATIENT,GP
       WHERE P_gp_NINO=gp_NINO AND
            GP_Name= 'Dr J Shipman');
```

Screenshots from the SQL Query with Sample Data:

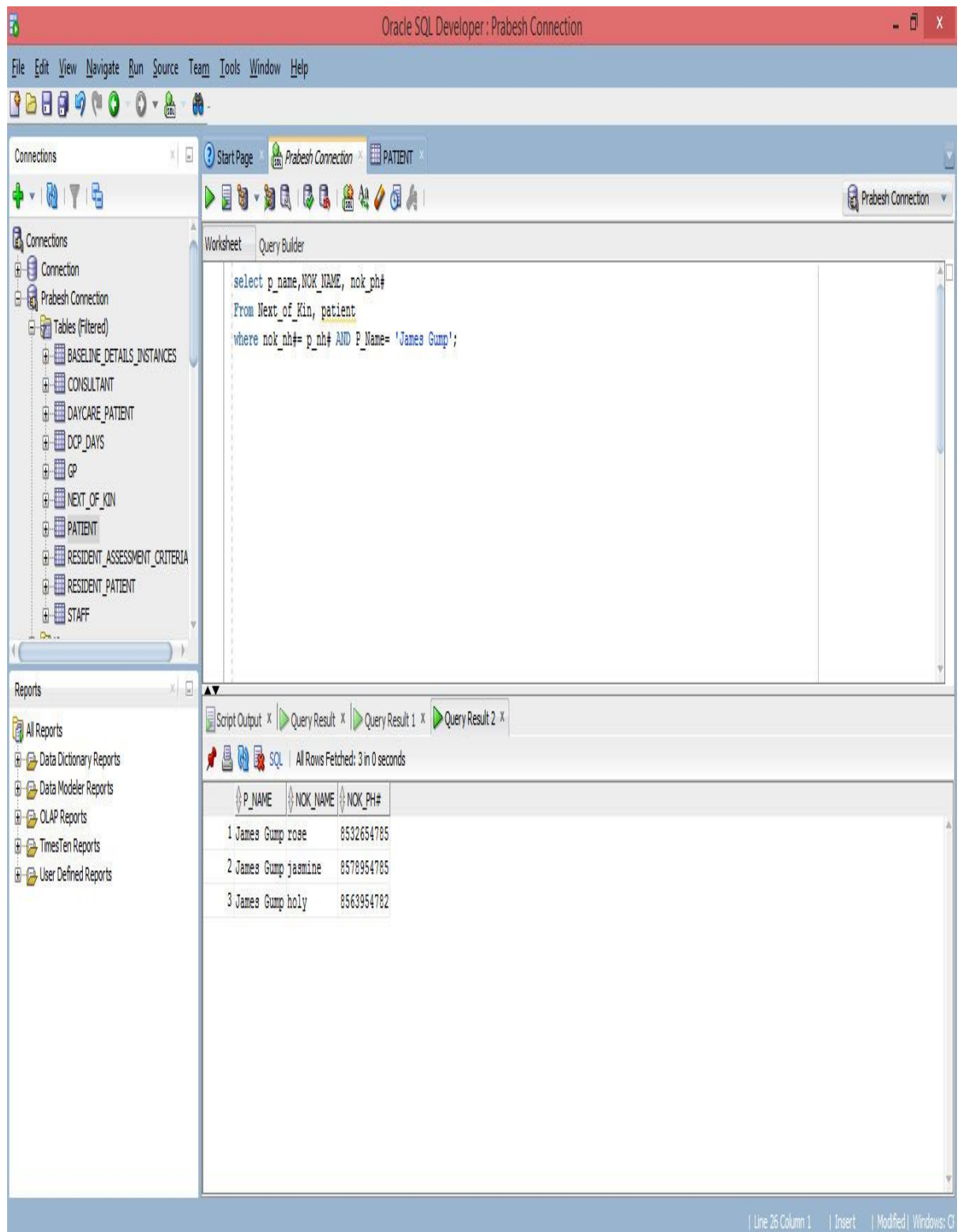
The screenshot displays the Oracle SQL Developer interface with the 'Prabesh Connection' selected. The 'Connections' pane on the left shows a tree of tables, including 'BASELINE_DETAILS_INSTANCES', 'CONSULTANT', 'DAYCARE_PATIENT', 'DCP_DAYS', 'GP', 'NEXT_OF_KIN', 'PATIENT', 'RESIDENT_ASSESSMENT_CRITERIA', 'RESIDENT_PATIENT', and 'STAFF'. The 'Query Builder' pane shows the following SQL query:

```
select * from patient
where Ptype= 'Resident';
```

The 'Query Result' pane at the bottom shows the results of the query, with 2 rows fetched in 0.062 seconds. The results are displayed in a table with the following columns: P_NH#, P_NAME, P_NDNO, P_PH#, P_DOB, P_RELIGION, P_GP_NDNO, P_PCN_NH#, and PTYPE.

	P_NH#	P_NAME	P_NDNO	P_PH#	P_DOB	P_RELIGION	P_GP_NDNO	P_PCN_NH#	PTYPE
1	NHN20R	James Gump	NIN90P	7485435364	23-JAN-70	Christian	NTNS1g	NHN11A	Resident
2	NHN21R	JamMY 1	NIN91P	7485656895	02-FEB-72	Muslim	NTNS1g	NHN12A	Resident

The status bar at the bottom indicates 'Line 2 Column 25 | Insert | Modified | Windows: C'.



Oracle SQL Developer: Prabesh Connection

File Edit View Navigate Run Source Team Tools Window Help

Connections

- Connections
- Connection
- Prabesh Connection
 - Tables (Filtered)
 - BASILINE_DETAILS_INSTANCES
 - CONSULTANT
 - DAYCARE_PATIENT
 - DCP_DAYS
 - GP
 - NEXT_OF_KIN
 - PATIENT
 - RESIDENT_ASSESSMENT_CRITERIA
 - RESIDENT_PATIENT
 - STAFF

Worksheet Query Builder

```
select p_name,NOK_NAME, nok_ph#  
From Next_of_Kin, patient  
where nok_nh#= p_nh# AND P_Name= 'James Gump';
```

Script Output x Query Result x Query Result 1 x Query Result 2 x

SQL All Rows Fetched: 3 in 0 seconds

	P_NAME	NOK_NAME	NOK_PH#
1	James Gump	rose	8532654785
2	James Gump	jasmine	8578954785
3	James Gump	holy	8563954782

| Line 26 Column 1 | Insert | Modified | Windows: C

Oracle SQL Developer: Prabesh Connection

File Edit View Navigate Run Source Team Tools Window Help

Connections

Prabesh Connection

Tables (Filtered)

- BASLINE_DETAILS_INSTANCES
- CONSULTANT
- DAYCARE_PATIENT
- DCP_DAYS
- GP
- NEXT_OF_KIN
- PATIENT
- RESIDENT_ASSESSMENT_CRITERIA
- RESIDENT_PATIENT
- STAFF

Worksheet

```

select p_name,bdi_nh#, bdi_daterecorded, BDI_PATIENT_BLOODPRESSURE,BDI_PATIENT_PULSE,BDI_PATIENT_TEMPERATURE,BDI_PATIENT_WEIGHT
From Baseline_details_instances, PATIENT
where
bdi_nh# = p_nh# AND
P_Name= 'James Gump' AND
BDI_daterecorded between '01-Jan-15' and '08-Dec-15';

```

Script Output x Query Result x Query Result 1 x Query Result 2 x

SQL All Rows Fetched: 12 in 0.031 seconds

	P_NAME	BDI_NH#	BDI_DATERECORDED	BDI_PATIENT_BLOODPRESSURE	BDI_PATIENT_PULSE	BDI_PATIENT_TEMPERATURE	BDI_PATIENT_WEIGHT
1	James Gump	NHN20R	08-JAN-15	Normal	90 bpm	98 Celsius	84 kg
2	James Gump	NHN20R	08-FEB-15	Low	85 bpm	98 Celsius	85 kg
3	James Gump	NHN20R	08-MAR-15	Normal	90 bpm	98 Celsius	89 kg
4	James Gump	NHN20R	08-APR-15	High	85 bpm	98 Celsius	88 kg
5	James Gump	NHN20R	08-MAY-15	Normal	90 bpm	98 Celsius	87 kg
6	James Gump	NHN20R	08-JUN-15	Low	85 bpm	100 Celsius	83 kg
7	James Gump	NHN20R	08-JUL-15	Normal	90 bpm	98 Celsius	82 kg
8	James Gump	NHN20R	08-AUG-15	Low	85 bpm	98 Celsius	81 kg
9	James Gump	NHN20R	08-SEP-15	Normal	90 bpm	98 Celsius	80 kg
10	James Gump	NHN20R	08-OCT-15	Low	85 bpm	98 Celsius	85 kg
11	James Gump	NHN20R	08-NOV-15	Normal	90 bpm	98 Celsius	84 kg
12	James Gump	NHN20R	08-DEC-15	Low	85 bpm	98 Celsius	85 kg

Reports

- All Reports
- Data Dictionary Reports
- Data Modeler Reports
- OLAP Reports
- TimesTen Reports
- User Defined Reports

Line 13 Column 1 | Insert | Modified | Windows: C

The screenshot shows the Oracle SQL Developer interface with the title bar 'Oracle SQL Developer: Prabesh Connection'. The menu bar includes File, Edit, View, Navigate, Run, Source, Team, Tools, Window, and Help. The toolbar contains icons for file operations and database actions. The left pane shows the 'Connections' tree with 'Prabesh Connection' selected, and a 'Tables (Filtered)' list including BASELINE_DETAILS_INSTANCES, CONSULTANT, DAYCARE_PATIENT, DCP_DAYS, GP, NEXT_OF_KIN, PATIENT, RESIDENT_ASSESSMENT_CRITERIA, RESIDENT_PATIENT, and STAFF. The main workspace is in 'Query Builder' mode, displaying the following SQL query:

```
select p_name, staff_name, gp_name, staff_jobtype, PTYPE
from PATIENT, GP, STAFF
where p_con_nh# = staff_nh# AND p_gp_nino = gp_nino AND PType = 'Day Care' AND Staff_JobType = 'Primary Care Nurse';
```

Below the query editor, the 'Script Output' tab shows 'All Rows Fetched: 2 in 0.031 seconds'. The 'Query Result' tab displays the following data:

	P_NAME	STAFF_NAME	GP_NAME	STAFF_JOBTYPE	PTYPE
1	SHAIUN p	Mariana golden	Dr Ashton Derek	Primary Care Nurse	Day Care
2	RITCHELL Mary	lark	Dr Fredo Smith	Primary Care Nurse	Day Care

The bottom status bar indicates 'Line 15 Column 1 | Insert | Modified | Windows: C'.

Oracle SQL Developer: Prabesh Connection

File Edit View Navigate Run Source Team Tools Window Help

Connections

- Connections
- Prabesh Connection
 - Tables (Filtered)
 - BASELINE_DETAILS_INSTANCES
 - CONSULTANT
 - DAYCARE_PATIENT
 - DCP_DAYS
 - GP
 - NEXT_OF_KIN
 - PATIENT
 - RESIDENT_ASSESSMENT_CRITERIA
 - RESIDENT_PATIENT
 - STAFF

Reports

- All Reports
- Data Dictionary Reports
- Data Modeler Reports
- OLAP Reports
- TimesTen Reports
- User Defined Reports

Worksheet Query Builder

```
select COUNT (*)
from PATIENT, resident_assessment_criteria
where p_nh=rac_nh# AND
rac_mental_awareness= '1' or rac_mental_awareness='2' AND
RAC_DATECHANGED= '07-Dec-15' and p_nh# IN
(SELECT P_NH#
FROM PATIENT,GP
WHERE P_gp_NINO=gp_NINO AND
GP_Name= 'Dr J Shipman');
```

Script Output x Query Result x

SQL | All Rows Fetched: 1 in 0.015 seconds

COUNT(*)
1

| Line 19 Column 1 | Insert | Modified | Windows: C

Part Four: Suitability of Winterbourne Nursing Home's Implementation of NOSQL in a NOSQL Database:

NoSQL is a good choice for Big Data where we have large volume of data which is variant in nature and is frequent.

Since the Winterbourne Nursing home data can easily be implemented in a structured way and can be accessed simply with the help of SQL, using NoSQL seems to be an undesirable option. Also, the data in Winterbourne Nursing Home is not that huge (in terms of number of users accessing the details and also big number of computers which are present in a distributed system), which is totally opposite to the key part of NoSQL (in NoSQL, the size or volume of the data is considered to have an important role). Since the data used by the nursing home is not that massive, and neither is it required to store large volume of data which requires it to be massively distributed having large number of users, computers, requirements, etc.; it's better to keep it wrapped in SQL.

Similarly, the data in winterbourne nursing home does not have wide variety of data, which means that the data stored is always stored nearly similar to the design and structure of the database design. Since the data does not vary frequently, we can store every information using SQL. Also, we are not performing different read, write, and query over the database, consistency of data is maintained and data can be accessed without any discrimination.

However, some NoSQL systems do support relationships, however maintaining relationship is best achieved in SQL. SQL database provides us with a store of related data in a constructive manner. This basically means that we are being strict but with proper structure. For example, if we have defined any data field as number and we pass any other datatype value like string then SQL will restrict us to do so hence minimising the chances of errors. So keeping that in mind, it's quite easy to form a well-constructed database for nursing home with SQL database.

Also SQL database provides us with the opportunity to update any field later if any modification is required. In case of Winterbourne Nursing Home database, all the requirements are already cited and are clear, so when we have the initial data requirements in our hand it is good to approach work in SQL database rather than using NoSQL. Using NoSQL is effective when initial requirements are difficult to know in the beginning.

In SQL, we can do normalisation of data, which means we can reduce redundancy. In Nursing Home database, we have tried to avoid the data redundancy as much as possible. SQL database provides us the structured query language which is a very powerful tool for manipulating the data. With the use of JOIN clause, we can use and retrieve the data from related tables quickly from the database using a single SQL statement. However, in NOSQL, there is no such function as JOIN. We can do the JOIN command/clause, manually though. Also, NOSQL uses unstructured query language mainly focussing on documents rather than relation or table.

When using the SQL database we enforce certain constraints (e.g referential integrity) into the nursing home data, which the schema will enforce the database to follow every time and make it not possible for users to tamper the data. Hence with SQL we have maintained the data integrity which is desired for securing the data. In contrast, such data integrity option is not present in NOSQL which means one can store any data which they want to store, regardless of the relevant information.

Also looking from the practical perspective in terms of security, SQL is far more reliable than NoSQL. We can say that this is the case because of the lack of knowledge base for NoSQL, as it is still not clear to everyone. And just because NoSQL is fresh, new or can use de-normalised data does not mean we should avoid the designing of schema which actually can lead to a problem at a later stage! As NOSQL database is generally document based; such schema related problems can occur in future.

It is worth mentioning SQL is always helpful and useful when the data requirement is identified upfront and when maintaining data integrity is essential. Because we do not have any unrelated or undetermined data for the nursing home, SQL is definitely a better option compared to NoSQL database.