TABLE OF CONTENTS

ABSTRACT...........................................................................................................2
BACKGROUND AND RATIONALE.................................................................3
NARRATIVE--------------------------------------------------------------------------5
ENVIRONMENT-----------------------------------------------------------------------8
SYSTEM ANALYSIS, DESIGN AND IMPLEMENTATION........................................9
APPLICATION TESTING AND PERFORMANCE EVALUATION..........................26
CONCLUSIONS-----------------------------------------------------------------------27
REFERENCES AND BIBLIOGRAPHY---------------------------------------------------28
APPENDIX--------------------------------------------------------------------------29

A. DATABASE THE TABLE STRUCTURES.
B. THE DATABASE TRIGGERS.
C. THE DATA FOR THE DATABASE.
D. ENTITY-RELATIONSHIP DIAGRAM.
E. SAMPLE WINDOWS.
F. REPORTS.
ABSTRACT

The purpose of the project has been to develop a system to automate the Dietary Department at St. Mary’s Hospital in Port Arthur, Texas. This newly developed system allows for coordination and efficient management of patients’ dietary requirements among the dietary, the nursing, and the food service facilities within the hospital. The system utilizes a relational database to track each inpatient’s information, particularly his or her diagnosis, dietary restrictions, food allergies, and food and drug interactions. The front end of the system facilitates to the patient’s diet based on updated laboratory results and doctors and dieticians recommendations, thereby ensuring that the patient receives an appropriate diet.
BACKGROUND AND RATIONALE

St. Mary’s Hospital is a 300-bed, multi-specialty hospital in Port Arthur, Texas. It is a member of the Sisters of Charity hospitals located throughout the state of Texas.

At the present time, when a patient enters the hospital, all of the patient information and history are recorded (handwritten), and organized into the patient’s chart by the Admission Department. The patient’s chart is then transferred to the nurses’ station, on the floor to which the patient is assigned, and remains there until the patient is discharged. All of the other departmental personnel including dietician, doctor, and the Food Service staff can access the patient’s information only from the nurses station. This process is time-consuming and requires additional personnel. Data redundancy is common.

In order to make all the patient’s information available to all concerned departments and to avoid functional overlap and data redundancy, a central database system was created for the Dietary Department. Employees from the Dietary, Food Service, and Nurses departments were interviewed, patient’s charts were reviewed and the for the information needed to design the central database system. This collection of data took over a period of six weeks.
The department of Information Systems and the Dietary Department at the hospital also wanted to automate the Dietary Department to enhance accuracy, speed, and quality with which the patients can be served by preparing specific diets. In addition, all the departments are able to quickly access the required data from the central database system.
NARRATIVE

The Automated Dietary System for St. Mary’s Hospital runs on a multi-user system. The Admissions Department enters the initial information about the patient such as name, insurance information, medical history. All other departments are then able to access but not edit this information.

The automated system combines the power of a relational database management system with that of a user-friendly, graphical user interface (PowerBuilder 5.0). The user can access, modify (update) or retrieve data as well as perform desired functions by simply selecting a button on the screen, moving the mouse, making menu selections, or by entering the requested data. The system guides the user and prompts him or her for necessary input. Since the PowerBuilder software accesses the database and allows the users to update the information at any time, the system eliminates the need to duplicate data and ensures data is up-to-date, hence promoting data consistency.

The users of the database such as doctors, nurses, pharmacists, dieticians and food service personnel can navigate through the database system and also produce reports by using the PowerBuilder graphic user interface software.
All the major and minor components of the developed system provide the following basic features:

1. Enter new information
2. Update information
3. Delete obsolete information
4. Generate appropriate reports
5. Display information

Enter New Information:

This option allows the user to add new information to the database. For example, the user is able to add a new patient, new diagnosis and new drugs.

Update the Information:

The user upon entering the appropriate code, is able to change the existing information in the database. The system allows the user to browse through the list of patients and select the patient’s record to update. After selecting the patient, the system the user is guided through prompts to reach the information he or she wishes to change. By selecting the Save button after the change is made, the user updates the record.
Delete Obsolete Information:

Entering the patient’s number as in the update procedure accesses a patient’s record. The selected information from the patient record is deleted by choosing the Delete button.

Generate Reports:

The report option is the final option within the automated Dietary System. The user is able to select a particular report from the menu option. The Print option lets the user print the selected report.
ENVIRONMENT

This system is a multi-user system. The client can use either Windows 3.11, Windows 95 or Windows NT. The client can be a low-end machine, but the server requires at least a Pentium 133 MHZ with 32MB memory. All the client machines are connected to the server by a 10Base-2 coaxial cable and need a network adapter card, one for each machine.

The software consists of PowerBuilder 5.0, which provides a graphical user interface on the client machines and a Relational DataBase which uses Oracle 7.0.
SYSTEM ANALYSIS, DESIGN AND IMPLEMENTATION

Preliminary studies at St. Mary’s Hospital indicated that the hospital uses older information and lags behind in adapting to the rapid development of technologies.

By adopting client/server technology, work at the hospital can be less stressful, more productive and more enjoyable. Using a relational database such as Oracle and GUI-based user friendly front-end technologies like PowerBuilder, many operations at St. Mary’s Hospital were automated. As the hospital system is very large and complex, only the Dietary department was studied completely.

After the code was developed and tested, an executable version was installed at the hospital and evaluated against the present system.

The project was divided into various stages:

1. System Analysis
2. System Design: back-end and front-end design
3. Application Development
4. Application Testing
5. Performance Evaluation
SYSTEM ANALYSIS

The personnel from the nursing, pharmacy, dietary, laboratory, kitchen and medical departments, who are responsible for the operation of the existing dietary system, were interviewed, the sequence of operations was analyzed and various entities and their relationships were identified.

The total hospital system is very large and complex, hence only systems closely related to the dietary department were considered. The sequence of operations is listed below in a simplified way:

1. When a patient arrives at the hospital, the registration nurse notes the patient history, insurance information, etc., and a room is allotted. Also, a nurse notes the patient’s physical conditions and various symptoms into the patient’s record.

2. A doctor examines the patient and usually orders some tests that are recorded in the patient’s chart.

3. If indicated, the patient is taken to various labs (x-ray, CAT-scan, etc.) and the samples of blood, urine, etc. are sent for analysis.

4. When the test results arrive, the doctor completes the diagnosis, sends any prescriptions needed to the pharmacy department, and makes a note of the diagnosis in the patient’s chart.
5. The pharmacy department prepares the medicines and sends them to the nurse’s station.

6. The nurse in charge of the patient makes a note of all the patient’s drugs, food allergies and vital signs (periodically) into the patient’s chart.

7. A dietician works out a diet for each patient from the available menu and within the guidelines.

8. A nurse places an order with the food service department.

9. The doctor changes the prescription of drugs as needed as the patient progresses.

10. The food service department works out all the other details like, the patient room number and time the food should be sent.

The tasks for the food department are listed below:

- After all the orders are received the food service department decides what items are to be prepared and in what quantity, and whether the existing inventory of groceries meets the requirements. They also determine what groceries to order, from whom and what time they should be delivered. In addition, they create work sheets for the cook, cook’s assistants, store people, grocers, food distributing people so that there is no confusion anywhere.
• Store people work out all details to supply material in time. They also have to see that no supplied food has an expired date.

• Cooks plan the menus for the day and assign work to their assistants.

• After the food is prepared, distributors come into play. Following their work sheets and menus, they distribute the food.

The system is automated very well in a client-server environment since most of the tedious work, such as preparing directions for serving food within dietician guidelines and communicating information from one section to the other, are done by the computer system.

The Automated Dietary System uses a RDBMS, together with a GUI, event-driven, object-oriented application developed in PowerBuilder on several client machines. The new sequence of operations using the new system is as follows:

1. When a patient arrives at the hospital, a registration nurse enters the patient information directly into the system while asking questions and entering the answers on a GUI form. A room is then allotted based on the patient’s and doctor’s preferences. An admittance number is allotted
automatically which binds the patient with the attending doctor, the
assigned room, and the approximate days of stay.

2. After the patient has been moved to the assigned room, a staff nurse enters
all physical conditions and symptoms information directly into the
computer system after opening a proper form for the patient.

3. Similarly, the doctor enters his or her recommendations, diagnosis and
tests required on a computer form.

4. When the tests are completed, the laboratory staff enter the results directly
into the system.

5. Once all the information pertaining to the particular patient is completed,
the doctor can look very quickly through the patient’s computerized
record and make the final diagnosis.

6. When the doctor enters the diagnosis through a computer form, the
system informs the dietician to plan an appropriate diet, the pharmacist to
prepare medicines, and the nurse at the nurse’s station to take an
appropriate course of action.

7. The doctor prescribes a course of treatment and the nurses use the system
to track whether the prescribed medicines are given to the patient on time
and if other procedures are followed correctly.

8. When a dietician prescribes a diet, the nurse lets the patient choose from a
menu for lunch, dinner and breakfast.
9. A menu card is presented to the patient to assist him or her in selecting from the menu by presenting him or her with a list of valid foods under each group. Validation is done by the Dietitian to ensure that a food is within the guidelines and that it does not interact with known allergies or medicines the patient is taking at the present time.

10. When the patient completes choosing from the menu, the dietician checks for correctness and passes it on to the Food Service Department.

11. The menus are in turn sent to the kitchen staff.

12. The cooks and assistant cooks along with the Food Service supervisors prepare grocery lists and work schedules for the kitchen staff.

13. The cooks and assistant cooks prepare meals according to the menus.

14. Food servicee staff then follow the Patient Diet Prescription. Reports generated by the automated system to distribute meals to the patients in their rooms.

15. Food Servicee department is also responsible for supplying dietary supplements to the patients according to the report generated by the new system.

The automated model reduces the workload on the hospital staff, presents a less stressful life, and improves productivity of employees by a considerable factor. Users like it because it is user friendly.
FRONT-END DESIGN

This is the part in which the user interacts with the application. The requirements for the system are:

- It should be easy for the user to understand how to interact with the application.
- It should be esthetically pleasing.
- It should be pleasant for the user to work with.
- It should remove as much load as possible from the user.
- It should make use of advanced features like event-driven programming, object-oriented and polymorphism.
- It should have good security features since the application is exposed to the public and contains very private information.
- It should provide a smooth flow of the application with good navigational features.

Keeping the requirements in mind an overview of the front-end is presented.

Five major modules are presented. These modules exist with several other modules that are outside the scope of this project. Each module is presented with its functionality, a list of event-driven features, a list of major validation procedures, and a list of forms and reports that are present in the module. The modules are:
1. System Administration
2. Admittance of patients
3. Treatment of patients
4. Diet of the patients
5. Reports
SYSTEM ADMINISTRATION

Functionality:

This module deals with all the objects that directly deal with patients. The objects are nurses, doctors, dietitians, drugs, diets and various test categories. The module maintains all the objects. When a new drug is available in the hospital, it is added to the database. All the other objects are maintained similarly.

The main purpose is to keep up-to-date records of all these objects in the system so that all the transactions involved will have first-hand information of these objects. The user can search a specific item in all the tables, or insert, update, and delete the item in the tables they operate on. These tasks are done before any actual transaction begins. Each object has an object id in the system. So, all the transaction can use their “id” for reference. For example, each doctor will be assigned a doctor id.

List of Forms:

1. Doctor’s information display
2. Doctor’s information modification
3. Nurse’s information display
4. Nurse information modification
5. Dietician’s information display
6. Dietician’s information modification
7. Drug’s information display
8. Drug’s information modification
9. Diet’s information display
10. Diet’s information modification
11. Test category’s information display
12. Test category’s information modification

In the above forms, the user displays the required information as needed. Also, he or she is able to add, update, and delete the corresponding items of all the objects when the necessity arises. For example, if a doctor is newly employed by the hospital, the user is able to add the doctor to the database. If some information relating to a doctor has been changed or the operator inputs some incorrect information into the system, the system provides the way to let user modify and update it accordingly. If a doctor leaves the hospital, the information pertaining to this doctor can be deleted from the system. The above transactions can be performed for all the objects in this module, such as patients, nurses, diets and drugs.
ADMITTANCE OF PATIENTS

Functionality:

This module handles functions that deal with patient’s admittance.

All the system’s actual transactions begin after the patient visits the hospital. When a patient comes to the hospital, the nurse first checks if the patient has visited the hospital before. If so, he or she gets the patient’s general information from the patient’s master table and only adds the new information relating the patient’s current visit. Otherwise, the nurse inputs the patient’s general information, assigns the patient a patient id for later use, and then continues to log all the current visit information for this patient. If the nurse needs any information related to the patient’s previous visit, the system provides a very convenient way to let the nurse search the information he or she needs. The nurse can also modify the patient’s visit information in case of data entry errors or if some information the patient provided has changed since he or she was admitted to the hospital. When a patient dies the nurse can delete the obsolete patient visit information from the database with permission from his or her supervisor. All the recorded information is the basis for the following processes.

LIST OF FORMS:

1. Display patient’s general information
2. Record patient’s general information
3. Display patient’s visit information
4. Log patient’s visit information

TREATMENT PROCESS OF THE PATIENT

Functionality:

This module handles the functions and objects that deal with patient treatment. When the nurse finishes the patient visit information record, the patient goes into treatment process by the selected doctor. The doctor orders some tests if required and analyzes all the vital signs of the patient and the results of the tests ordered. The doctor also decides on the course of treatment and the drugs to use.

List of Forms:

1. Find vital signs information
2. Log vital signs information
3. Tests conducted
4. Log tests information for patients
5. Conduct patient’s lab test
6. Search lab test results for patient
7. Look for a diagnosis prescription for a patient
8. Doctor prepares diagnosis prescription for a patient
9. Drug prescription information query
10. Doctor gives a drug prescription

PATIENT'S DIET

Functionality:

This module handles the functions that deal with the diet of the patients. When the doctor finishes his or her diagnosis for the patient and gives the patient a corresponding drug prescription, the dietician analyzes all the information, which includes patient’s illness situation, patient’s physical condition, foods to avoid, patient’s food dislikes and patient’s food likes and gives his or her diet schedule for the patient. The schedule will be adjusted from time to time depending upon the patient’s information.

After the patient has been admitted into the room the treatment process begins. One important process of the treatment is to prepare the proper diet for the patient according to his or her situation. The dieticians carefully selects the diet components for the patient so that it will be beneficial to the patient’s recovery. In this process, the
dieticians get all the information of the patient, such as food likes, and food dislikes, from the newly developed central database and give the patient’s final diet prescriptions.

List of Forms:

1. Find foods avoided
2. Log avoided foods for patient
3. Display foods liked
4. Update foods liked for patient
5. Search for foods the patient dislikes
6. Log the foods the patient dislikes

REPORTS

Functionality:

This module is responsible for the reports for the whole system. The reports include patient’s visit information, foods liked, foods disliked, foods avoided, doctor in the hospital, nurse in the hospital, dietician in the hospital, vital signs reports and the dietary prescription reports.
List of Reports

1. Patient information
2. Doctor information
3. Nurse information
4. Dietician information
5. Patient visit information
6. Vital signs information
7. Dietary prescription

BACK-END DESIGN

The system uses Personal Oracle 7.3 for Windows 95 as its back end relational database system, which maintains the data for the whole system. For each transaction, the client interacts with the Oracle database server. The client requests needed information from the Oracle database and asks the server to update or modify some information. When needed, the server adds new information for the clients. The clients only need to provide the data they want to add to the Oracle RDBMS and then the Oracle server can provide the services according to the client’s request. In our system, the clients provide
requests in the PowerBuilder GUI environment. After the Oracle database server finishes
the service, the client can view the changes in the PowerBuilder GUI environment.

Entities

The system consists of seven major entities: doctor, nurse, dietician, patient, diet, drug
and test category. Correspondingly, master tables have been created in the system. They
are “doctor”, “patient”, “nurse”, “dietician”, “diet”, “drug”, “testcategory” “patient visit”,
“vital signs”, “test”, “lab_test”, “diagnosis”, “diet_px”, and “drug_px” The tables are
attached in the APPENDIX.

Relationship among entities

The relationships among the above entities is detailed in the E-R diagram in the
APPENDIX.

Triggers and Procedures

According to the system requirement, three Oracle triggers and one PL/SQL
procedure were created and used for the system. The three triggers are used for the
system data consistency, and the PL/SQL procedure was used for data loading purpose.
Constraints in the system

In order to maintain data consistency in the system, more than 20 Oracle constraints have been created. They include Primary key, Foreign key and Not Null property. All of the constraint definitions are included in the APPENDIX.
APPLICATION TESTING AND PERFORMANCE

EVALUATION

The completed software was installed on a Pentium at the clinical Dietitian’s office and tested with real data. The automated System performed all the required functions satisfactorily.

The new system eliminates the duplication of forms thereby reducing paperwork, ensuring data consistency, and providing a less error prone if not error free environment.

With the new system the reports are generated by the computer rather than being handwritten thereby reducing the workload on the hospital staff by a considerable factor.

The new system also allows the hospital staff to reference the patient information quickly.
CONCLUSIONS

The Automated Dietary System allows for coordination and efficient monitoring of a patient's dietary requirements among the dietary, nursing, and food service facilities within St. Mary's Hospital. With the newly developed system the doctors, nurses, pharmacists, laboratory personnel and food service personnel are able to review the patient's information, particularly his or her diagnosis, dietary restrictions, food allergies, and food and drug interaction. The user was able to print forms regarding patient's requirements and progress.
REFERENCE AND BIBLIOGRAPHY


APPENDIX
THE DATABASE TABLE STRUCTURES:
A.

----first
create table diet
( Diet_No integer not null,
  FoodsToAvoid varchar2(1000),
  FoodsToTake varchar2(1000),
  constraint p_pk8
  primary key (Diet_No)
);

----second
create table dietician
( Dietician_No integer,
  LastName varchar2(30),
  FirstName varchar2(30),
  Gender varchar2(1),
  SSN integer,
  DLN varchar2(20),
  DOB date,
  Address1 varchar2(50),
  Address2 varchar2(50),
  City varchar2(30),
  State varchar2(2),
  Zip varchar2(10),
  DayPhone varchar2(10),
  EvePhone varchar2(10),
  WeeklySchedules varchar2(1000),
  constraints p_pk4
  primary key (Dietician_No)
);

----third;
create table doctor
( Doctor_No integer,
  LastName varchar2(30),
  FirstName varchar2(30),
  Gender varchar2(1),
  SSN integer,
  DLN varchar2(20),
  DOB date,
  Address1 varchar2(50),
  Address2 varchar2(50),
  City varchar2(30),
  State varchar2(2),
  Zip varchar2(10),
  DayPhone varchar2(10),
  EvePhone varchar2(10),
  WeeklySchedules varchar2(1000),
  constraints p_pk2
  primary key (Doctor_No)
);

----4 th
create table drug
( Drug_No integer not null,
DrugPrescription varchar2(1000),
SideEffects varchar2(1000),
FoodInteractons varchar2(1000),
constraint p_pk10
primary key (Drug_No)
);

---5 th
create table nurse
( Nurse_No integer,
  LastName varchar2(30),
  FirstName varchar2(30),
  Gender varchar2(1),
  SSN integer,
  DLN varchar2(20),
  DOB date,
  Address1 varchar2(50),
  Address2 varchar2(50),
  City varchar2(30),
  State varchar2(2),
  Zip varchar2(10),
  DayPhone varchar2(10),
  EvePhone varchar2(10),
  WeeklySchedules varchar2(1000),
  constraints p_pk3
primary key (Nurse_No)
);

---6 th
create table patient
( Patient_No integer not null,
  LastName varchar2(30),
  FirstName varchar2(30),
  Gender varchar2(1),
  SSN integer,
  DLN varchar2(20),
  DOB date,
  Address1 varchar2(50),
  Address2 varchar2(50),
  City varchar2(30),
  State varchar2(2),
  Zip varchar2(10),
  DayPhone varchar2(10),
  EvePhone varchar2(10),
  MedicalHist varchar2(1000),
  constraints p_pk1
primary key (Patient_No)
);

---7 th
create table test_category
(TestCategory_No integer not null,
  CategoryDesc varchar2(100),
  constraint p_pk13
primary key (TestCategory_No)
);
CREATE TABLE patient_visit
(
    PatientVisit_No integer not null,
    DateAdmitted date,
    InsuranceCompanyInfo varchar2(1000),
    RoomAllotted varchar2(100),
    ApproxDaysOfStay integer,
    Symptoms varchar2(1000),
    PatientPhysicalCondition varchar2(2000),
    CourseOfTreatment varchar2(1000),
    PatientDietaryLikes varchar2(1000),
    PatientDietaryDislikes varchar2(1000),
    FoodAllergies varchar2(1000),
    Doctor_No integer,
    Nurse_No integer,
    Patient_No integer,
    Dietician_No integer,
    constraints p_pk200
    primary key (PatientVisit_No),
    constraints f_fk1
    foreign key (Patient_No) references patient,
    constraints f_fk2
    foreign key (Doctor_No) references doctor,
    constraints f_fk3
    foreign key (Nurse_No) references nurse,
    constraints f_fk4
    foreign key (Dietician_No) references dietician
);

CREATE TABLE vital_signs
(
    PatientVisit_No integer,
    VitalSigns_No integer,
    VitalSigns_Notes varchar2(1000),
    constraints p_pk5
    primary key (PatientVisit_No, VitalSigns_No),
    constraints f_fk5
    foreign key (PatientVisit_No) references patient_visit
);

CREATE TABLE test
(
    Test_No integer not null,
    TestDesc varchar2(100),
    TestCategory_No integer,
    constraint p_pk14
    primary key (Test_No),
    constraint f_fk16
    foreign key (TestCategory_No) references test_category
);

CREATE TABLE lab_test
(
    PatientVisit_No integer not null,
    LabTest_No integer not null,
    ResultOfTest varchar2(300),
    constraint p_pk15

primary key (PatientVisit_No, LabTest_No),
    constraint f_fk17
    foreign key (PatientVisit_No) references patient_visit
);

create table test_ltx
 ( PatientVisit_No integer not null,
   LabTest_No integer not null,
   Test_No integer not null,
   constraint p_pk16
   primary key (PatientVisit_No, LabTest_No, Test_No),
   constraint f_fk18
   foreign key (PatientVisit_No) references patient_visit,
   constraint f_fk19
   foreign key (PatientVisit_No, LabTest_No) references lab_test,
   constraint f_fk20
   foreign key (Test_No) references test
);

create table diagnosis
 ( PatientVisit_No integer not null,
   Diagnosis_No integer not null,
   InitialDiagnosisNotes varchar2(1000),
   DetailedDiagnosisNotes varchar2(1000),
   OtherObservations varchar2(1000),
   constraint p_pk6
   primary key (PatientVisit_No, Diagnosis_No),
   constraint f_fk6
   foreign key (PatientVisit_No) references patient_visit
);

create table diet_pres
 ( PatientVisit_No integer not null,
   DietPres_No integer not null,
   DateDietPrescribed date,
   TimeOfDayToGiveDiet char(1),
   DietarySupplements varchar2(1000),
   constraint p_pk7
   primary key (PatientVisit_No, DietPres_No),
   constraint f_fk7
   foreign key (PatientVisit_No) references patient_visit
);

create table diet_px
 ( PatientVisit_No integer not null,
   DietPres_No integer not null,
   Diet_No integer not null,
   constraint p_pk9
   primary key (PatientVisit_No, DietPres_No, Diet_No),
   constraint f_fk9
   foreign key (PatientVisit_No) references patient_visit,
   constraint f_fk11
   foreign key (Diet_No) references diet,
constraint f_fk10
foreign key (PatientVisit_No, DietPres_No) references diet_pres
;

------ 16 th
create table drug_pres
( PatientVisit_No integer not null,
  DrugPres_No integer not null,
  DateDrugPrescribed date,
  Dosage varchar2(100),
  TimeOfDayToGiveDrug varchar2(1),
  FoodDrugInteractions varchar2(1000),
  constraint p_pk11
  primary key (PatientVisit_No, DrugPres_No),
  constraint f_fk12
  foreign key (PatientVisit_No) references patient_visit
);

------ 17 th
create table drug_px
( PatientVisit_No integer not null,
  DrugPres_No integer not null,
  Drug_No integer not null,
  constraint p_pk12
  primary key (PatientVisit_No, DrugPres_No, Drug_No),
  --- constraint f_fk13
  --- foreign key (PatientVisit_No) references patient_visit,
  constraint f_fk14
  foreign key (PatientVisit_No, DrugPres_No) references drug_pres,
  constraint f_fk15
  foreign key (Drug_No) references drug
);

------ 18 th
create table diagnosis_diet_px
( PatientVisit_No integer not null,
  DietPres_No integer not null,
  Diagnosis_No integer not null,
  constraint p_pk17
  primary key (PatientVisit_No, DietPres_No, Diagnosis_No),
  --- constraint f_fk21
  --- foreign key (PatientVisit_No) references patient_visit,
  constraint f_fk22
  foreign key (PatientVisit_No, DietPres_No) references diet_pres,
  constraint f_fk23
  foreign key (PatientVisit_No, Diagnosis_No) references diagnosis
);

------ 19 th
create table diagnosis_drug_px
( PatientVisit_No integer not null,
  DrugPres_No integer not null,
  Diagnosis_No integer not null,
  constraint p_pk18
  primary key (PatientVisit_No, DrugPres_No, Diagnosis_No),
  --- constraint f_fk24
  --- foreign key (PatientVisit_No) references patient_visit,
constraint f_fk25
    foreign key (PatientVisit_No, DrugPres_No) references drug_pres,
constraint f_fk26
    foreign key (PatientVisit_No, Diagnosis_No) references diagnosis
);