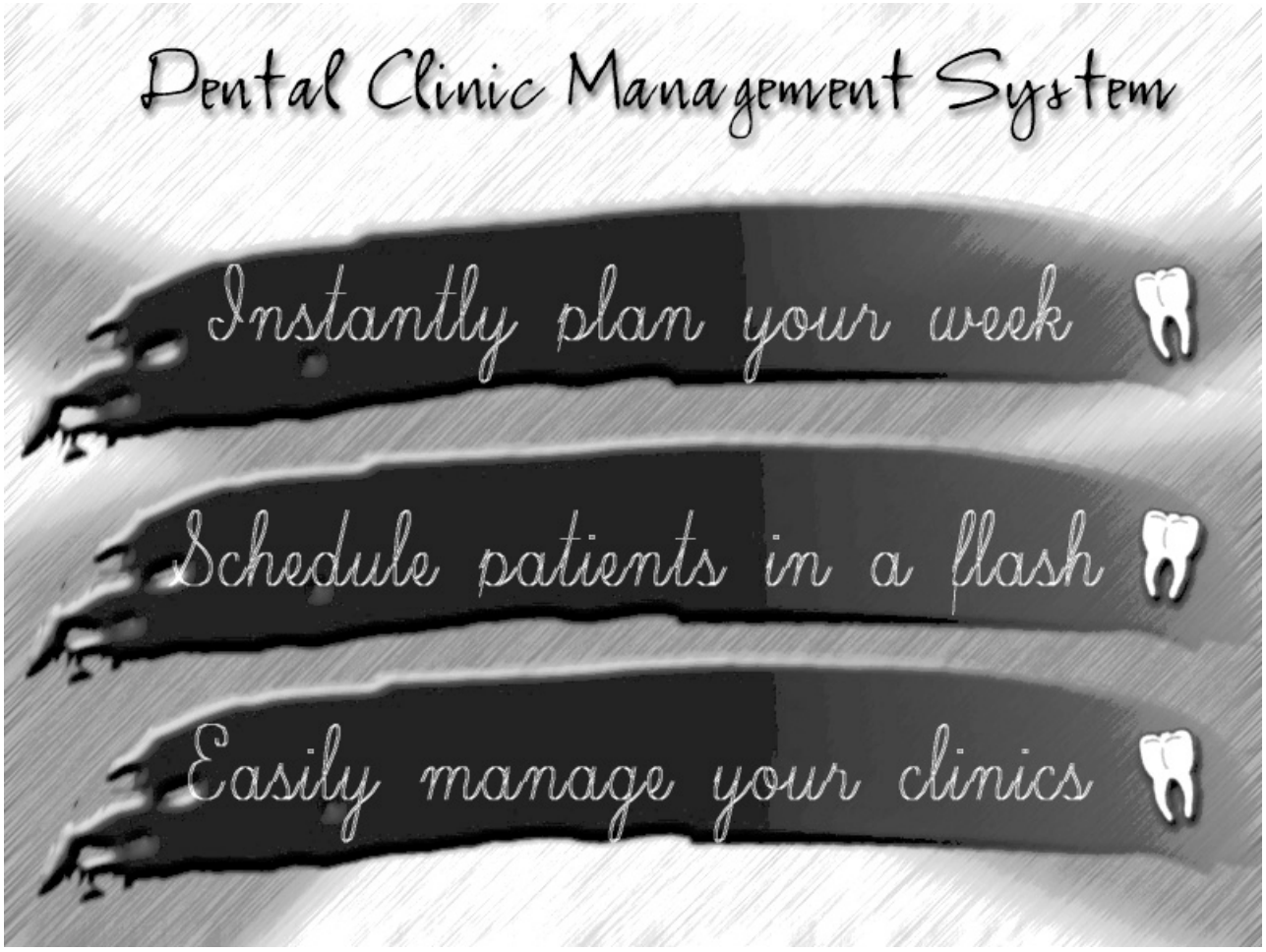


DENTIST PROJECT

Final Report of Development
By Team 12

Concordia
UNIVERSITY

Final Report on Database Implementation and GUI Design For the Dentist Project

Handed in by Team 12:
Yann McCready
Gaston D. Vaillancourt
François-Michel Brière
Frédéric Rioux
Marc-André Laverdière

Submitted to:
Z. Zhao

As required for COMP353, section AB

August 14th 2001

Department of Computer Science
Faculty of Engineering and Computer Science

Table of Contents for the Final Report

Table of Contents for the Appendixes

Table of Figures

Introduction

A fictitious dental company has requested that we develop a software package implemented on a database allowing various scheduling functions. The database project is centered on the dentist and includes various features for office assistants. This team took in consideration the specifications provided and rebuilt them using a system developer/end user approach. This method allowed us to efficiently analyze the various needs of the office assistants, managers and dentists in our product, then Dental Clinic Management System (DCMS).

This report reviews the process used by the team to efficiently model the constraints and requirements of the business, as well as the design decisions made by the team.

The team used Oracle Reports and Oracle Forms under SunOS UNIX to implement the GUI. These tools have been ported on multiple platforms, which allows polyvalence on future hardware and software levels for client companies wishing to use DCMS.

Development Specifications & Database Implementation

Final Design

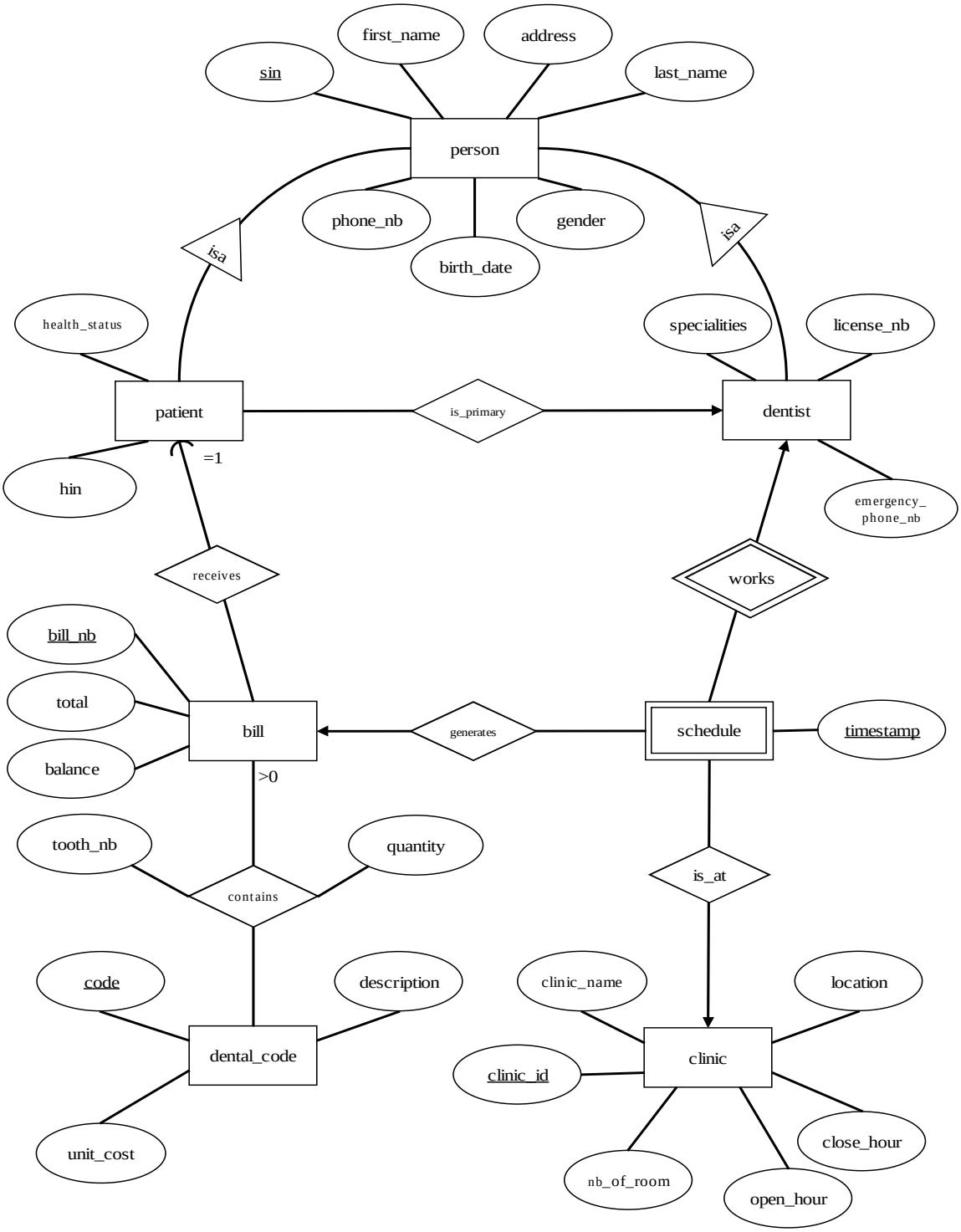


Figure 1: Final E/R Design

TB_Dentist

TB_dentist
(dentist_sin, first_name, last_name, address, phone_nb, birth_date, gender, specialties, licence_nb, emergency_phone_nb)

Server-Based Constraints

1. None of the fields can be left to NULL.

Assumptions

1. Each dentist is assumed to have provided a phone number as emergency contact information
2. The SIN is assumed to be correct, and no validation is performed.
3. A dentist can also be a patient.
4. The specialties field is to be represented as a character string.
5. We assume that the dentists' professional information (license number, address, etc.) is always correct and that the license is always valid.
6. All dentists are considered as active.

Non-Key Indexes

1. last_name, first_name

Oracle implementation

```
CREATE TABLE TB_dentist (
---------- -- Attributes -- ----------
dentist_sin NUMBER (9),
first_name VARCHAR2 (30),
last_name VARCHAR2 (30),
address VARCHAR2 (255),
phone_nb VARCHAR2 (20),
birth_date DATE,
gender CHAR (1),
specialities VARCHAR2 (255),
licence_nb NUMBER (12),
emergency_phone_nb VARCHAR2 (20),
---------------- -- Keys and Unicity -- ----------------
PRIMARY KEY (dentist_sin),
UNIQUE (licence_nb),
------ -- Checks -- ------
CHECK (dentist_sin >= 0),
CHECK (gender IN ('F', 'M')),
CHECK (dentist_sin IS NOT NULL), CHECK (licence_nb IS NOT NULL),
CHECK (first_name IS NOT NULL), CHECK (last_name IS NOT NULL),
CHECK (address IS NOT NULL), CHECK (phone_nb IS NOT NULL), CHECK
(birth_date IS NOT NULL), CHECK (gender IS NOT NULL), CHECK
(specialities IS NOT NULL), CHECK (emergency_phone_nb IS NOT NULL)
 );
```

Figure 2: The Oracle SQL Implementation of TB_dentist

TB_Patient

TB_patient
(patient_sin, first_name, last_name, address, phone_nb, birth_date, gender, health_status, hin, primary_d_sin)

Server-Based Constraints

1. None of the fields can be left to NULL.
2. No patients' primary dentist can have the same SIN that the patient's SIN (i.e., a dentist being a patient cannot be her/his own dentist).

Assumptions

1. The patient's online record doesn't include any non-textual data, such as X-rays.
2. The SIN and HIN are assumed to be correct, and no validation is performed.
3. All patients and dentists are considered as active.
4. A patient can also be a dentist. Such patients, however, are not offered any discounts or privileges compared to any other patients

Non-Key Indexes

1. last_name, first_name
2. HIN

Oracle implementation

```
CREATE TABLE TB_patient (
---------- -- Attributes -- ----------
patient_sin NUMBER (9),
first_name VARCHAR2 (30),
last_name VARCHAR2 (30),
address VARCHAR2 (255),
phone_nb VARCHAR2 (20),
birth_date DATE,
gender CHAR (1),
health_status VARCHAR2 (1023),
hin CHAR (12),
primary_d_sin NUMBER (9),
---------- -- Keys and Unicity -- ----------
PRIMARY KEY (patient_sin),
FOREIGN KEY (primary_d_sin) REFERENCES TB_dentist (dentist_sin),
UNIQUE (hin),
------ -- Checks -- ------
CHECK (patient_sin > = 0),
CHECK (gender IN ('F', 'M')),
CHECK (patient_sin <> primary_d_sin),
CHECK (patient_sin IS NOT NULL),
CHECK (first_name IS NOT NULL), CHECK (last_name IS NOT NULL),
CHECK (address IS NOT NULL), CHECK (phone_nb IS NOT NULL),
CHECK (gender IS NOT NULL),
CHECK (birth_date IS NOT NULL), CHECK (hin IS NOT NULL)
);
```

Figure 3: The Oracle SQL Implementation of TB_patient

TB_Clinic

TB_clinic
(clinic_id, clinic_name, location, close_hour, open_hour, nb_of_room)

Server-Based Constraints

1. None of the clinic information can be set to NULL.
2. The clinic_ID number is determined sequentially and incrementally and should be automatically generated (i.e. no insertion in the table should bother about the clinic_ID).

Assumptions

1. No room is directly assigned to any given dentist. The rooms are for use by all dentists.
2. Each clinic opens and closes at the same set of hours, from Monday to Friday.
3. A clinic can be temporarily closed. As such, the number of rooms can be set to zero.

Non-Key Indexes

1. clinic_name

Oracle implementation

```
CREATE TABLE TB_clinic (
---------- -- Attributes -- ----------
clinic_id NUMBER (6),
clinic_name VARCHAR2 (255),
location VARCHAR2 (255),
close_hour DATE,
open_hour DATE,
nb_of_room NUMBER (2),
---------------- -- Keys and Unicity -- ----------------
PRIMARY KEY (clinic_id),
------ -- Checks -- ------
CHECK (clinic_id >= 0),
CHECK (nb_of_room >= 0),
CHECK (clinic_id IS NOT NULL),
CHECK (nb_of_room IS NOT NULL),
CHECK (clinic_name IS NOT NULL),
CHECK (location IS NOT NULL),
CHECK (close_hour IS NOT NULL),
CHECK (open_hour IS NOT NULL)
);
```

Figure 4: The Oracle SQL Implementation of TB_clinic

TB_Dental_Code

TB_dental_code
(code, description, unit_cost)

Server-Based Constraints

1. Every dental code is an integer greater than zero.
2. The cost of every dental code is superior or equal to zero.

Assumptions

1. All the billing codes are included in this table. As such it also includes administrative codes, such as penalties and various charges that are not dental-related.
2. The various treatments are not listed by specialty.

Oracle implementation

```
CREATE TABLE TB_dental_code (
---------- -- Attributes -- ----------
code NUMBER (4),
description VARCHAR2 (255),
unit_cost NUMBER (7,2),
---------- -- Keys and Unicity -- ----------
PRIMARY KEY (code),
------ -- Checks -- ------
CHECK (code > 0),
CHECK (unit_cost >= 0),
CHECK (code IS NOT NULL),
CHECK (unit_cost IS NOT NULL),
CHECK (description IS NOT NULL)
 );
```

Figure 5: The Oracle SQL Implementation of TB_dental_code

TB_Bill

TB_bill
(bill_nb, total, balance, patient_sin)

Server-Based Constraints

1. The bill number is determined sequentially and incrementally and should be automatically generated (i.e. no insertion on the table should bother about the bill number).
2. An appointment for a patient implies a bill, whether or not the patient was present to receive treatment.
3. The balance and total can only be greater or equal than zero.
4. The balance must be lower or equal to zero.

Client-Based Constraints

1. The billing department can query by the bill number but not determine the bill number.

Assumptions

1. The patient is billed for the work of a single dentist.
2. The dentist who performed the operation is qualified to do so.

Non-Key indexes

1. patient_sin

Oracle implementation

```
CREATE TABLE TB_bill (
---------- -- Attributes -- ----------
bill_nb NUMBER (9), total NUMBER (7,2),
balance NUMBER (7,2), patient_sin NUMBER (9),
---------- -- Keys and Unicity -- ----------
PRIMARY KEY (bill_nb),
FOREIGN KEY (patient_sin) REFERENCES TB_patient (patient_sin),
------ -- Checks -- ------
CHECK (bill_nb >=0),
CHECK (total >= 0),
CHECK (balance >= 0),
CHECK (patient_sin >= 0),
CHECK (balance <= total),
CHECK (bill_nb IS NOT NULL),
CHECK (total IS NOT NULL),
CHECK (balance IS NOT NULL)
);
```

Figure 6: The Oracle SQL Implementation of TB_bill

TB_Bill_Items

TB_bill_items
(code, bill_nb, tooth_nb, quantity)

Server-Based Constraints

1. The quantity has to be greater than zero.
2. The tooth number is in the interval of 0 to 49. Normally, the teeth are in the range of 11 to 48, but we keep ourselves margins to maneuver. Other teeth number could be used for administrative purposes.

Oracle implementation

```
CREATE TABLE TB_bill_items (
        -- ----------
        -- Attributes
        -- ----------
        code NUMBER (4),
        tooth_nb NUMBER (2),
        quantity NUMBER (2),
        bill_nb NUMBER (9),
        -- ----------------
        -- Keys and Unicity
        -- ----------------
        PRIMARY KEY (code, bill_nb, tooth_nb),
        FOREIGN KEY (bill_nb) REFERENCES TB_bill (bill_nb),
        FOREIGN KEY (code) REFERENCES TB_dental_code (code),
        -- ------
        -- Checks
        -- ------
        CHECK (code > 0),
        CHECK (bill_nb >= 0),
        CHECK (tooth_nb >= 0),
        CHECK (tooth_nb < 50),
        CHECK (quantity >= 0),,
        CHECK (tooth_nb IS NOT NULL),
        CHECK (quantity IS NOT NULL)
);
```

Figure 7: The Oracle SQL Implementation of TB_bill_items

TB_schedule

TB_schedule
(timestamp, dentist_sin, clinic_id, bill_nb)

Server-Based Constraints

1. The time of the beginning of the appointment is a multiple of halves of hours.

Client-based Constraints

1. The insertion of the dentist's schedule should set the bill_nb field to NULL. Only the dentist scheduling form can perform insert/delete operations and only the patient scheduling form should update the bill_nb field.
2. A dentist's scheduling inside a clinic is limited by the rooms available and the clinic's opening hours. That means that no dentist can give his or her schedule for a clinic if the sum of the dentists in that clinic at that period of time is equal to the number of rooms in the clinic. This constraint was intended to be a server-based constraint implemented by trigger, but we decided to opt for an application-based constraint in order to provide better feedback to the user.
3. The validation of the scheduling in concordance with the clinic's opening hours is done by the client in order to provide error feedback to the user.

Assumptions

1. The scheduling is done manually by office assistants based on the dentist's wished schedule. All the conflict resolution is handled manually according to some corporate policy and should not be taken care of by the system.
2. The dentists can work in any given number of clinics in the same day, and no policy regarding travel and eating times are to be handled by the system.
3. No overbooking is to be done and the dentist handles only one appointment in a single time slot.
4. All the clients must have an appointment in order to see a dentist.
5. The appointment doesn't include the motive of the visit or the treatments to be performed.
6. The system doesn't consider if the day is an holiday or a weekend.

Non-Key Indexes

1. bill_number

Oracle implementation

```
 CREATE TABLE TB_schedule (
---------- -- Attributes -- ----------
timestamp DATE,
dentist_sin NUMBER (9),
clinic_id NUMBER (6),
bill_nb NUMBER (9),
----------------
Keys and Unicity -- ----------------
PRIMARY KEY (timestamp, dentist_sin),
FOREIGN KEY (dentist_sin) REFERENCES TB_dentist (dentist_sin),
FOREIGN KEY (clinic_id) REFERENCES TB_clinic (clinic_id),
FOREIGN KEY (bill_nb) REFERENCES TB_bill (bill_nb),
------ -- Checks -- ------
CHECK (timestamp IS NOT NULL)
 );
```

Figure 8: The Oracle SQL Implementation of TB_schedule

General Constraints

Referential Integrity Constraints

1. No dentist can be deleted from the database if there was any billed appointments.
   - Automatically implemented in the referential integrity constraints
2. No clinic can be delete if there are any scheduled avalabilities.
   - Automatically implemented in the referential integrity constraints
3. No dental code can be removed if it has been used in any bill.
   - Automatically implemented in the referential integrity constraints
4. No patient can be deleted unless it has no bill, paid or unpaid.
   - Automatically implemented in the referential integrity constraints
5. No element of the schedule can be deleted if the bill_number field is not NULL.
   - Implemented using a trigger:

```
CREATE OR REPLACE TRIGGER KeepSchedule
BEFORE DELETE ON TB_schedule
FOR EACH ROW
BEGIN

IF :old.bill_nb IS NULL
  THEN
        NULL; -- Its okay to delete if field is null
  ELSE
        RAISE_APPLICATION_ERROR(-20004, 'Error: Apointment cannot be deleted');
  END IF;
END;
```

Figure 9: The Oracle SQL Implementation of Constraint #5

6. A bill cannot be deleted, unless it is empty.
   - Automatically implemented in the referential integrity constraints
7. No dentist can work at two clinics at the same time.
   - Automatically as the primary key of tb_schedule
8. No patient can have two appointments at the same time.
   - Implemented using a trigger:
9. The treating dentist cannot be the treated patient.
   - Implemented using a trigger:
10. No dentist can add scheduling in a clinic that is already full.
    - Implemented using a trigger:

```
CREATE OR REPLACE TRIGGER ScheduleConsistency
BEFORE INSERT ON TB_schedule
FOR EACH ROW
DECLARE
  Patient NUMBER;
  Counter NUMBER;
BEGIN

  -- This trigger disallows the insertion of conflicting apointments for the patient

  -- Finds the patient Id
  SELECT Patient_sin INTO Patient
  from tb_bill B
  where B.bill_nb = :NEW.bill_nb;

  -- This trigger disallows the insertion of an apointment where the dentist would also be the
patient.
  -- if The patient's sin is different and the dentist's sin, then the insertion is allowed
  IF Patient = :NEW.dentist_sin THEN
          RAISE_APPLICATION_ERROR(-20006,'Error: The patient cannot treat self');
  END IF;

  --Finds all other apointments at the same timeslot
  SELECT COUNT(timestamp) INTO Counter
  FROM tb_bill B, tb_schedule S
  WHERE B.patient_sin = Patient
  AND B.bill_nb = S.bill_nb
  AND :NEW.timestamp = timestamp;

-- if more than one at the same period of time, then we generate an error, as it would be
overbooking
  IF Counter < 1 THEN
          NULL;
  ELSE
          RAISE_APPLICATION_ERROR(-20005,'Error: The patient already has an
apointment at that moment');
  END IF;
END;

-- Only the bill number is to be updated
CREATE OR REPLACE TRIGGER ScheduleBillUpdates
BEFORE UPDATE ON TB_schedule
FOR EACH ROW
DECLARE
  Patient NUMBER;
  Counter NUMBER;
  ClinicSize NUMBER;
BEGIN
IF :NEW.bill_nb <> :OLD.bill_nb THEN
IF :NEW.bill_nb IS NOT NULL THEN
  -- This trigger disallows the insertion of conflicting appointments for the patient
```

Figure 10: The Oracle SQL Implementation of Constraint #8,9,10

```
 -- Finds the patient Id
 SELECT Patient_sin INTO Patient
 from tb_bill B
 where B.bill_nb = :NEW.bill_nb;

 -- This trigger disallows the insertion of an apointment where the dentist = patient.
 -- if The patient's sin is different and the dentist's sin, then the insertion is allowed
 IF Patient = :NEW.dentist_sin THEN
        RAISE_APPLICATION_ERROR(-20006,'Error: The patient cannot treat self');
 END IF;

 --Finds all other apointments at the same timeslot
 SELECT COUNT(timestamp) INTO Counter
 FROM tb_bill B, tb_schedule S
 WHERE B.patient_sin = Patient
 AND B.bill_nb = S.bill_nb
 AND :NEW.timestamp = timestamp;

 -- if more than one at the same period of time, then we generate an error
 IF Counter < 1 THEN
        NULL;
 ELSE
        RAISE_APPLICATION_ERROR(-20005,'Error: The patient already has an
apointment at that moment');
 END IF;
END IF;
END IF;
IF :NEW.bill_nb IS NOT NULL THEN
-- This ensures that a dentist in an appointment cannot be a
patient.
   SELECT Patient_sin INTO Patient
   FROM tb_bill
   WHERE bill_nb = :NEW.bill_nb;

   IF Patient = :NEW.dentist_sin THEN
      RAISE_APPLICATION_ERROR(-2008,'Error: The patient cannot
treat self');
   END IF;
END IF;
END;
.
RUN
END;
```

Figure 11: The Oracle SQL Implementation of Constraint #8,9,10 (continued)

Views

In order to facilitate some queries, a few views have been created instead of using joints.

VW_Dentist_Schedule

Dentist Schedule allows a simple querying, matching the dentist, patient and billing. This view is used for the online dentist scheduling form.

```
CREATE VIEW VW_DENTIST_SCHEDULE AS(
        SELECT S.TIMESTAMP,
             S.DENTIST_SIN,
             S.CLINIC_ID,
             S.BILL_NB,
             B.PATIENT_SIN,
             P.FIRST_NAME,
             P.LAST_NAME,
             P.PHONE_NB
        FROM   TB_SCHEDULE S,
             TB_BILL     B,
             TB_PATIENT  P
        WHERE  S.BILL_NB = B.BILL_NB (+)
          AND  B.PATIENT_SIN = P.PATIENT_SIN (+)
);
```

Figure 12: The Oracle SQL Implementation of VW_dentist_schedule

VW_Patient_Appointment

Patient Appointment allows simple querying, matching the dentist, clinic and billing. This view is used for the online patient scheduling form.

```
CREATE VIEW VW_PATIENT_APPOINTMENT AS (
        SELECT S.TIMESTAMP,
             S.DENTIST_SIN,
             S.CLINIC_ID,
             S.BILL_NB,
             B.PATIENT_SIN,
             D.LAST_NAME,
             D.FIRST_NAME,
             C.CLINIC_NAME
        FROM   TB_SCHEDULE S,
             TB_BILL     B,
             TB_DENTIST  D,
             TB_CLINIC   C
        WHERE  S.BILL_NB = B.BILL_NB (+)
      AND  S.DENTIST_SIN = D.DENTIST_SIN
      AND  S.CLINIC_ID = C.CLINIC_ID);
```

Figure 13: The Oracle SQL Implementation of VW_patient_appointment

VW_BILL_ITEMS

Bill Items allows simple querying, matching the various billing elements. This view is used for the online billing form.

```
CREATE VIEW VW_BILL_ITEMS AS(
      SELECT BC.bill_nb AS Bill_nb,
            BC.code AS Code,
            DC.description AS Description,
            BC.tooth_nb AS Tooth_nb,
            BC.quantity AS Quantity,
            DC.unit_cost AS Unit_cost,
            DC.unit_cost * BC.quantity AS Total
      FROM   TB_bill_items BC,
            TB_dental_code DC
      WHERE  BC.code = DC.code
      ) ;
```

Figure 14: The Oracle SQL Implementation of VW_bill_items

GUI Implementation

Common Features

The Search Button

The Search Button allows the user to select and search trough a list of values to get a specific information. When the ok button is pressed, then the related fields will be filled automatically.

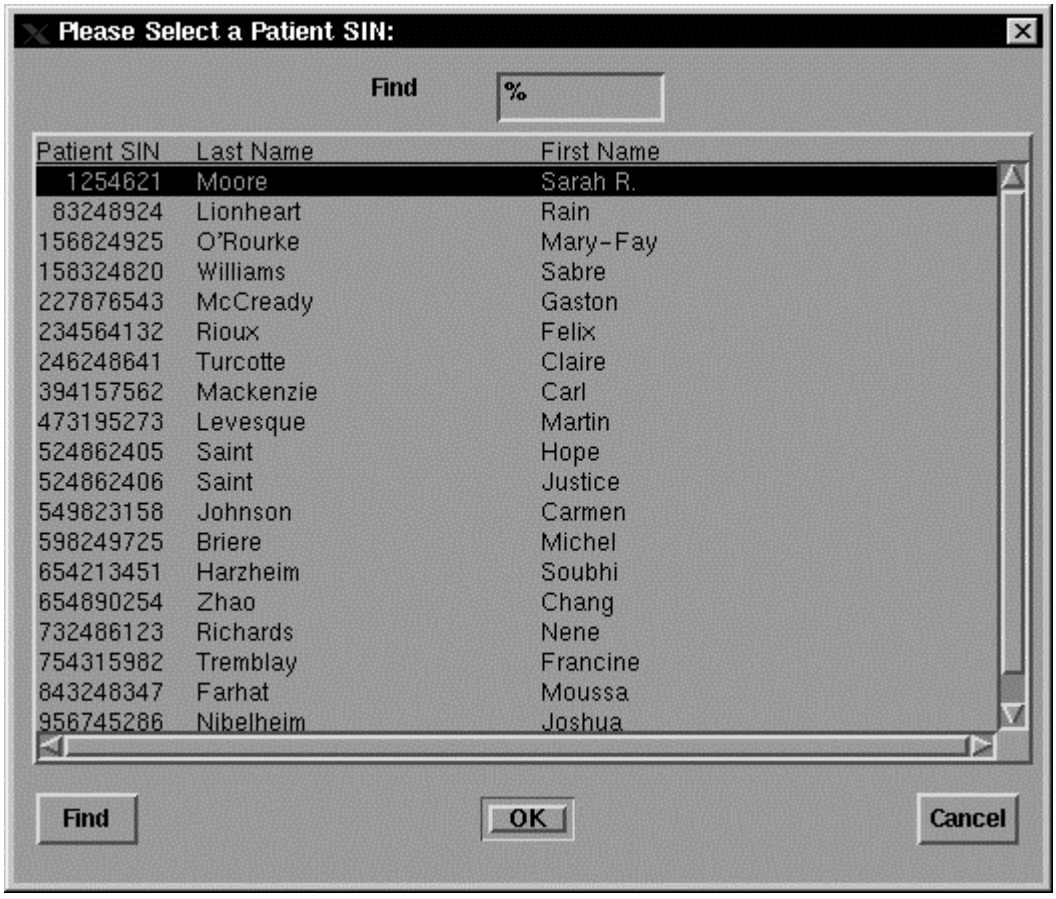


Figure 15: An Example of a Search Menu

The Modified Indicator

In most forms, the ‘SAVE’ button will change to ‘*SAVE*’ in order to show the user the need to commit the changes that have been made to the fields.

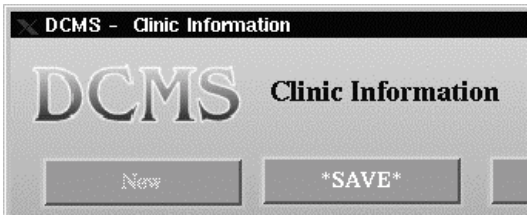


Figure 16: An Example of the Modified Indicator

The Back Button

All forms have a ‘Back’ button that allows the user to return to the Main Menu.

Figure 17: An Example of the Back Button

The Menu Bar

All the forms have a common menu bar that allows easy navigation from one window to the other.

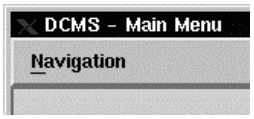


Figure 18 An Example of the Menu Bar

Overview of Screens and Features

The Main Screen

The Main Screen allows the user to access all the other screens. Each button will call the appropriate form and hitting on the 'Exit System' button will close DCMS.

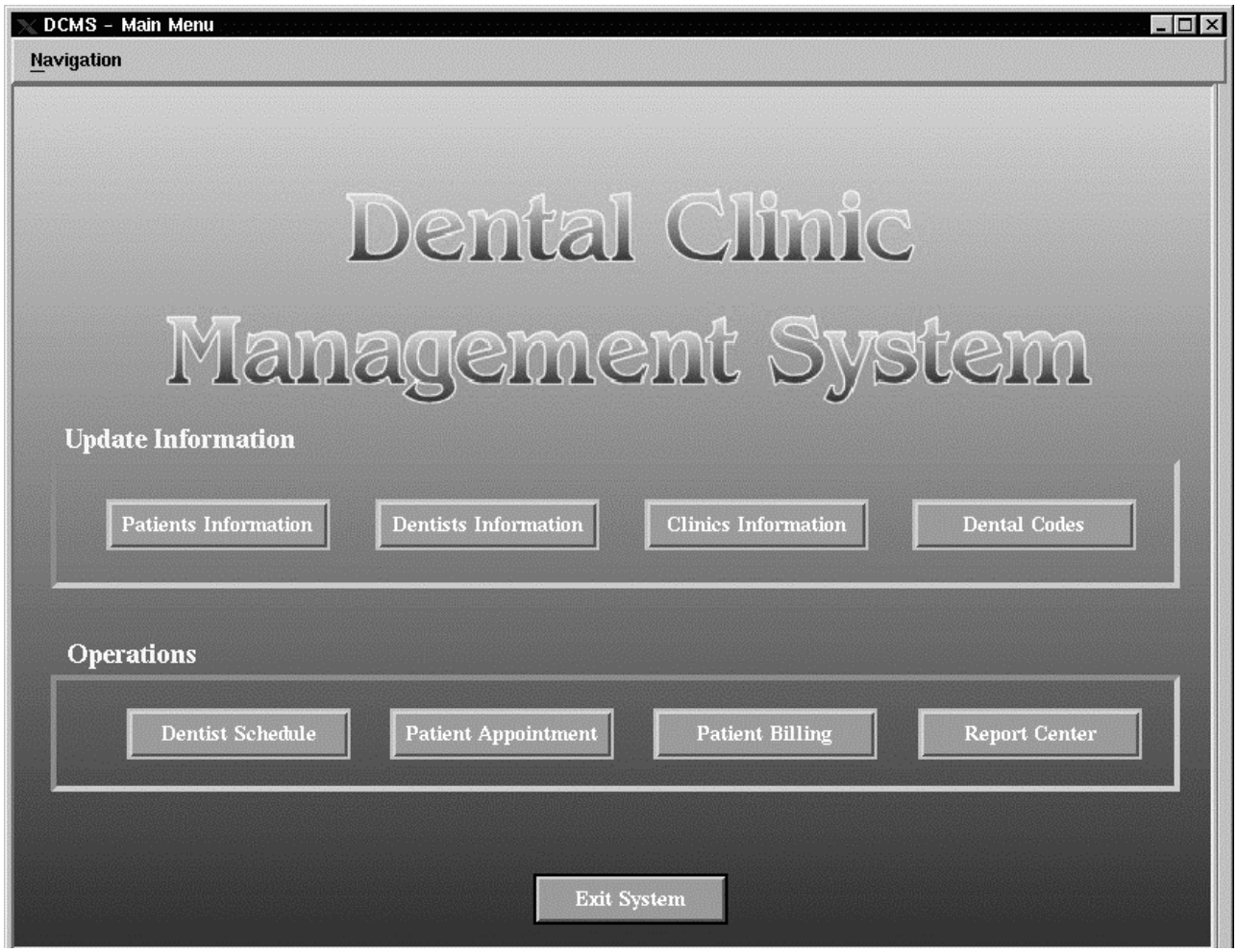


Figure 19: The Main Form

The Clinic Information Form

The Clinics Information Form allows the user to create or update dental clinics. The buttons allows the user to select a clinic from the list of already existing clinics, create a new one, save changes, etc.

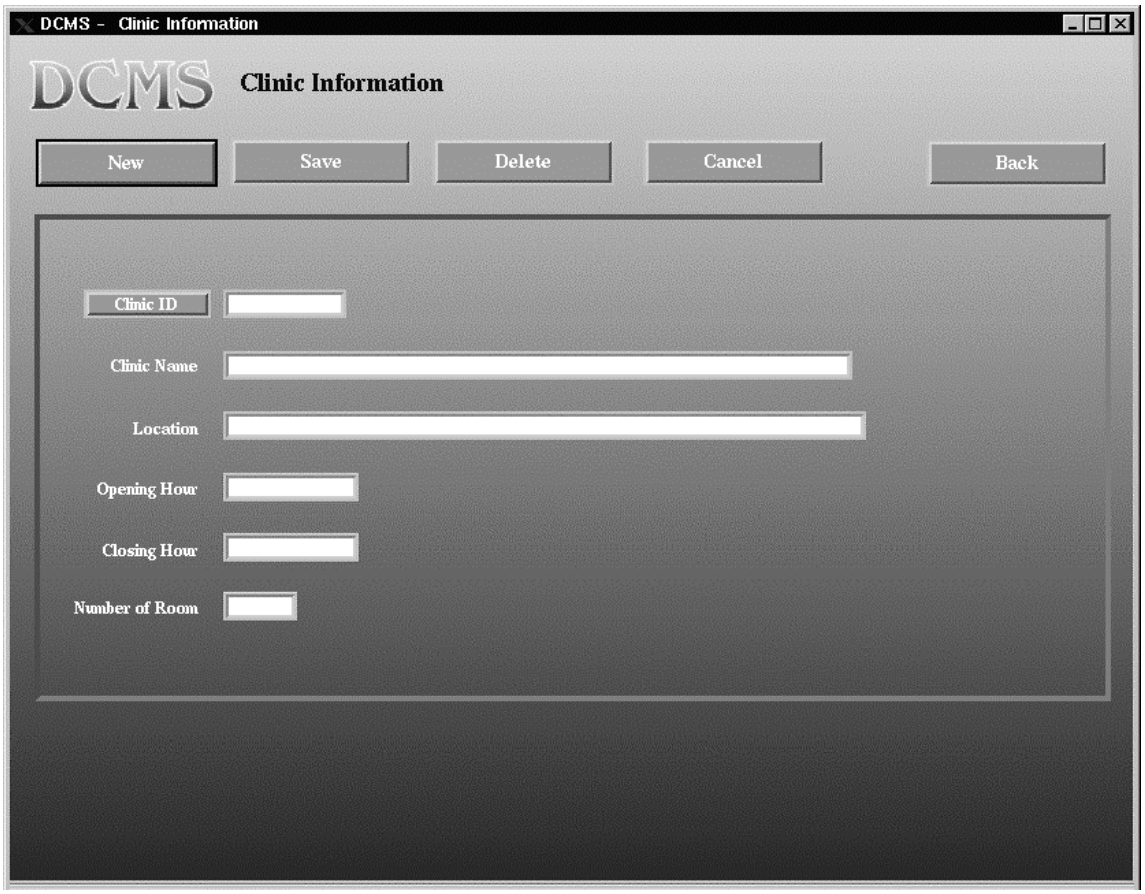


Figure 20: Clinic Information Window

The Dentist Information

The Dentists Information Form allows the user to add dentists in the system and update their information. The buttons allow the user to select a dentist from a list, create a new one, save changes, etc.

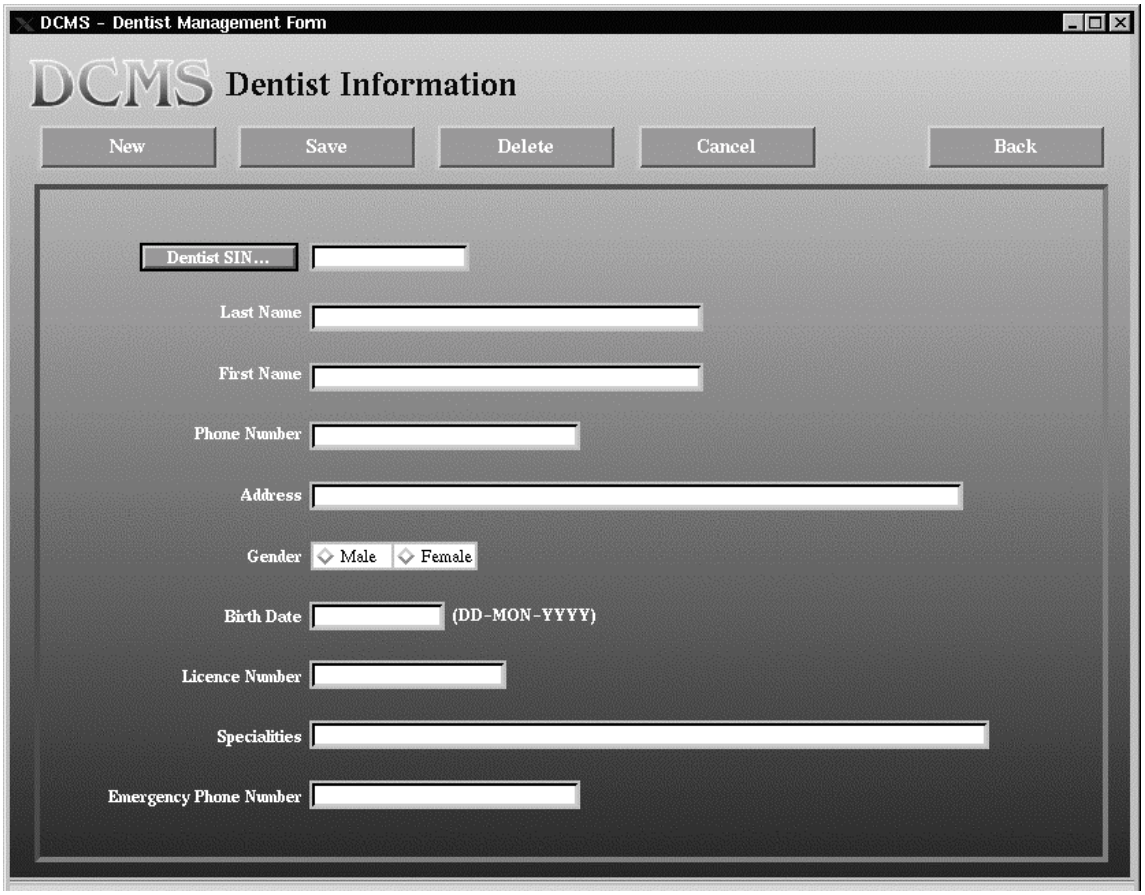


Figure 21: Dentist Management Window

The Patient Information

The Patients Information Form allows the user to add patients in the system and update their information. The buttons allows the user to select a patient from a list, create a new one, save changes, automatically generate the HIN (Health Insurance Number), etc.

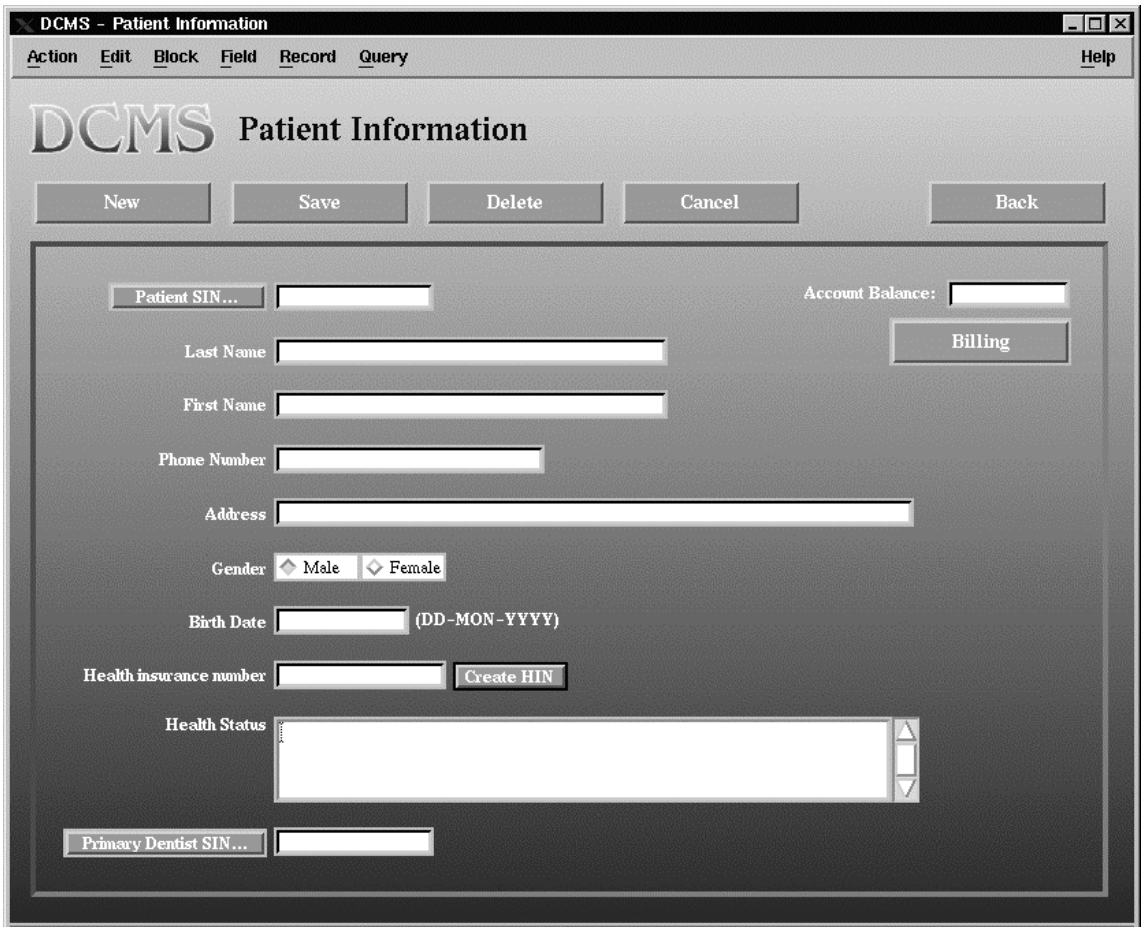


Figure 22 Patient Information Window

The Dental Codes

The Dental Codes form allows the user to keep track of the dental and administrative codes used in billing. The buttons allow the user to select a code from a list, add a new treatment, save changes, etc.

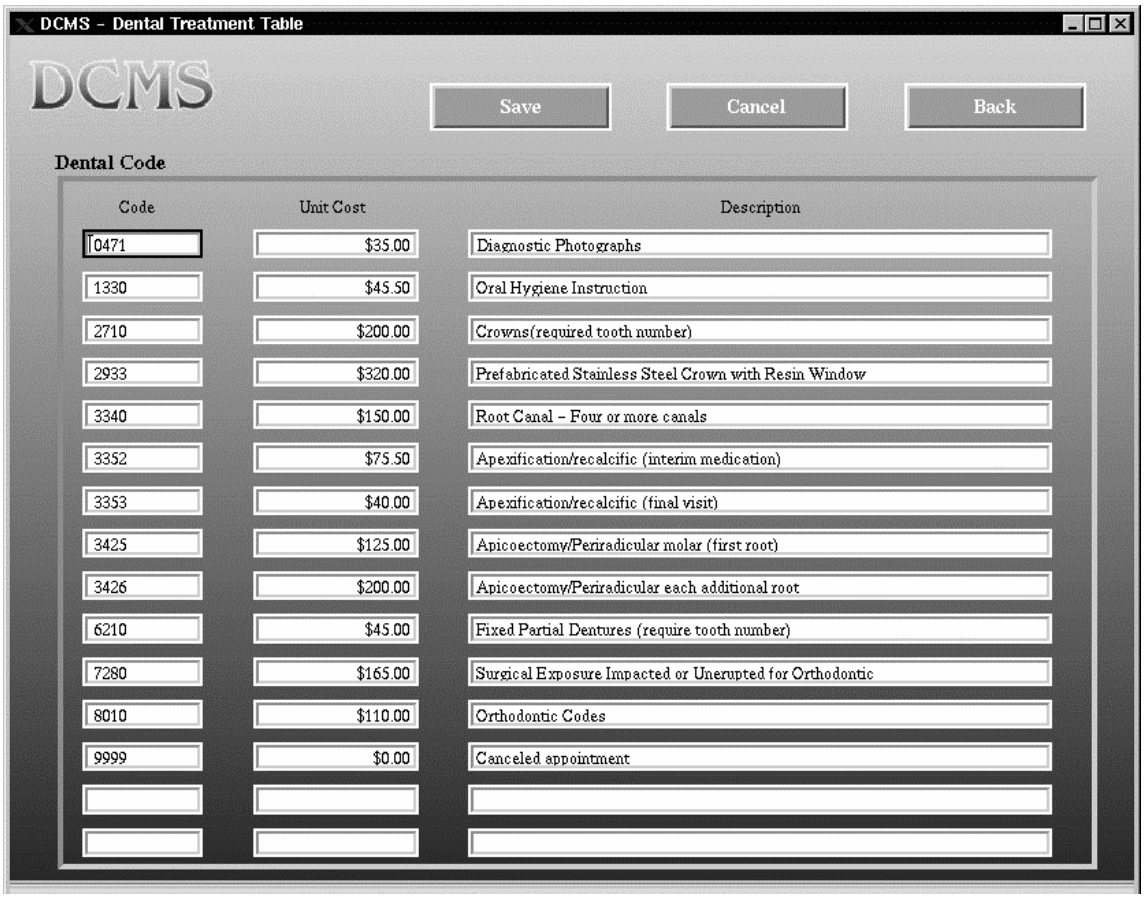


Figure 23: Dental Code Window

The Dentist Schedule

The Dentist Schedule form allows the user to update the schedule of a dentist, selecting from a range of dates and hours to work. A special option allows insertion whenever possible according to the clinic's usage. Appointments are displayed and the dentist can also remove previously assigned timeslots.

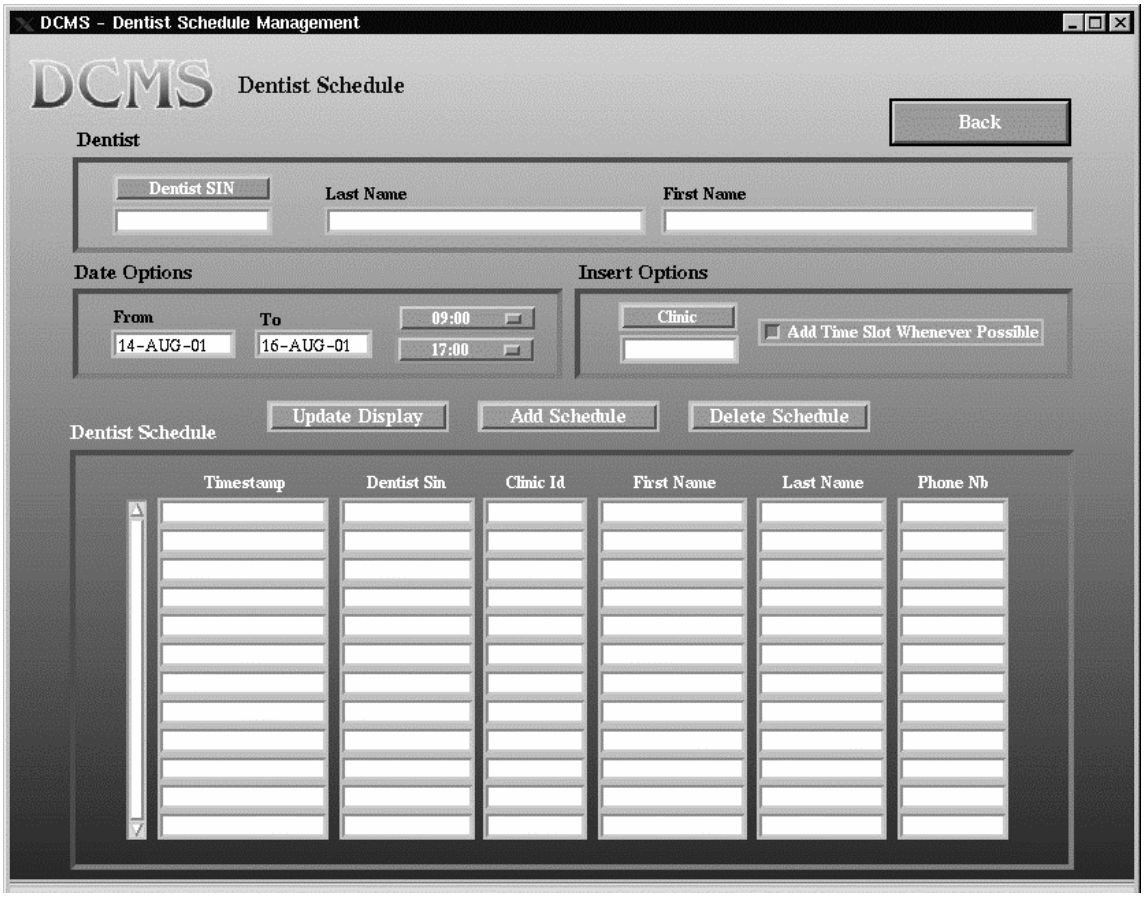


Figure 24: Dentist Schedule Management Window

The Patient Appointment

The Patient Appointment form allows the user to add an appointment with an available dentist. The user can select a patient from the database, select which a dentist and/or clinic if required, and the appropriate amount of time required for the patient's appointment.

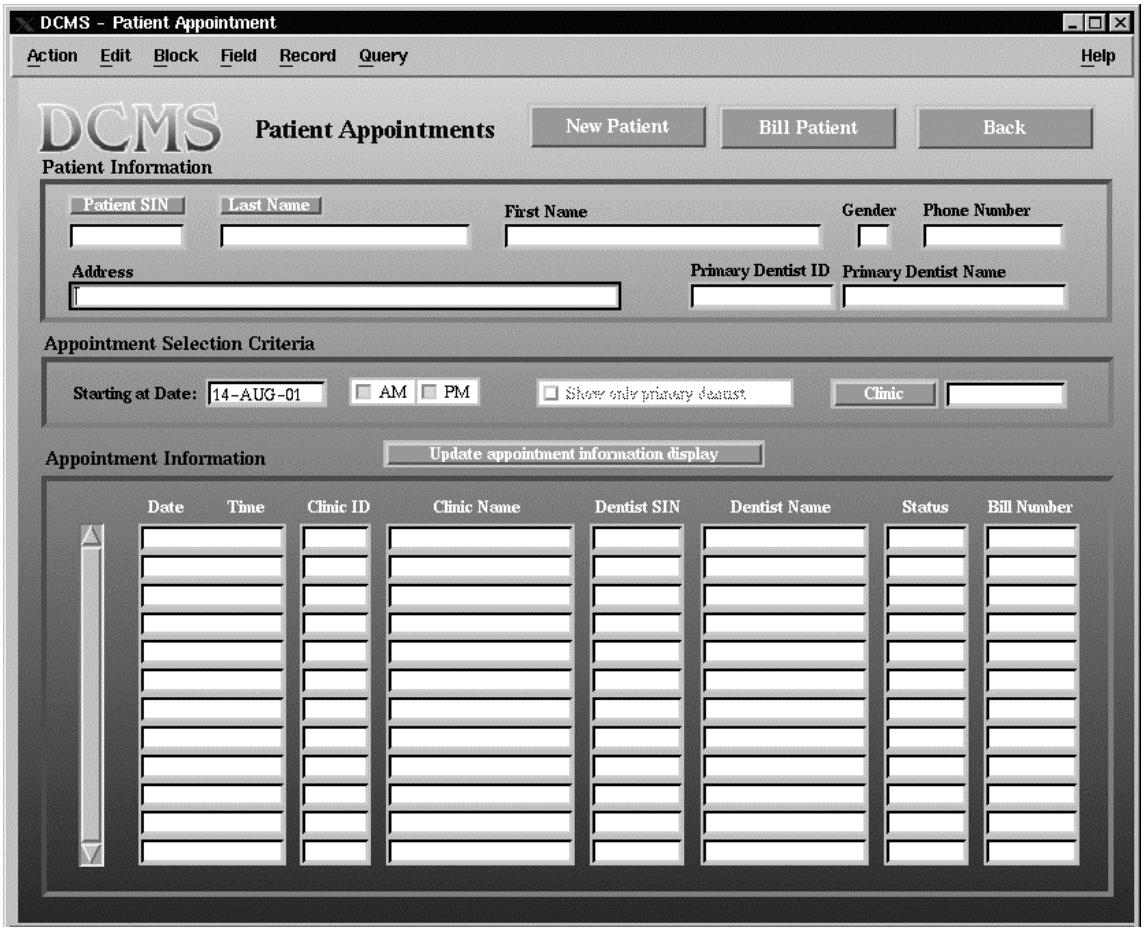


Figure 25: Patient Appointment Window

The Patient Billing

The Patient Billing form allows the user to add and remove items on the bill. The clerk can select from the appropriate dental codes to populate the bill, setting correctly the amount of treatments, the teeth that were treated, update the balance and print a bill for the patient.

Figure 26: The Patient Billing Management Window

The Report Center

The Report Center form allows the administrative staff to preview and print out reports according to the parameters they choose. The first report displays all the appointments, ordered by clinic, for any given period of time. The second report allows the accounting staff to monitor bills that were not paid by the patients, and which dentists are owed money. Please consult the next section (Reports) for more information about the layout of these reports.

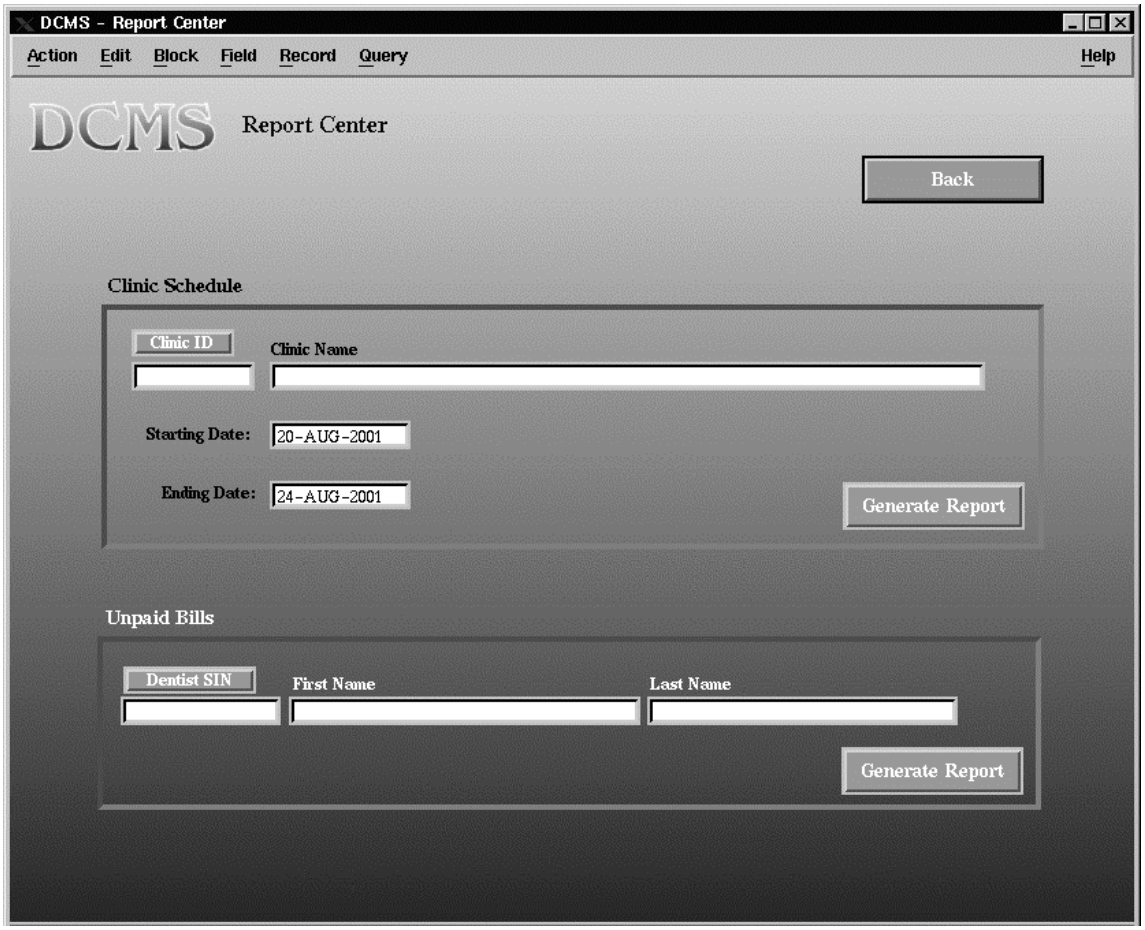


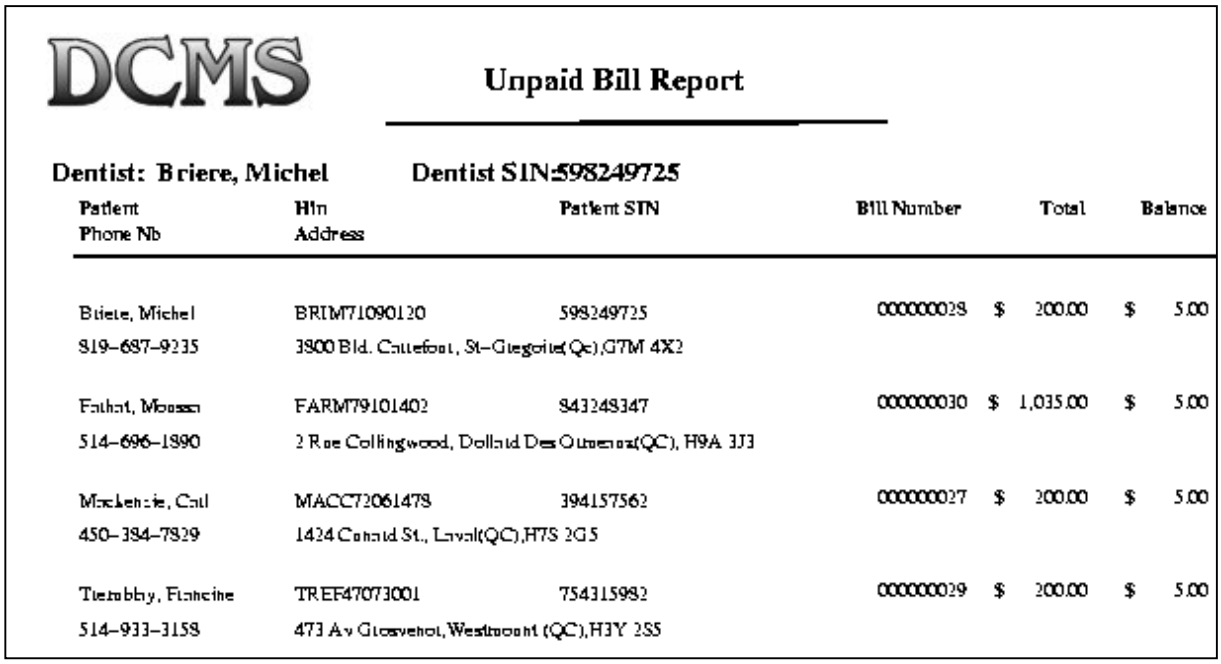
Figure 27 The Report Center Window

Reports

The reports generated by DCMS are of administrative nature, allowing paper output of the clinic's most important data: the scheduling and the billing information. As such, reports designed to print bills or display the list of unpaid bills were developed for the company's internal financial bookkeeping. Furthermore, a schedule report allows the administrative assistants and dentists to know the upcoming and previous appointments.

Unpaid Bill Report

The unpaid bill report allows the user to query the database and print a list of all unpaid bills, sorted by dentist. The user can also query the database for bills unpaid to a specific dentist.



DCMS

Unpaid Bill Report

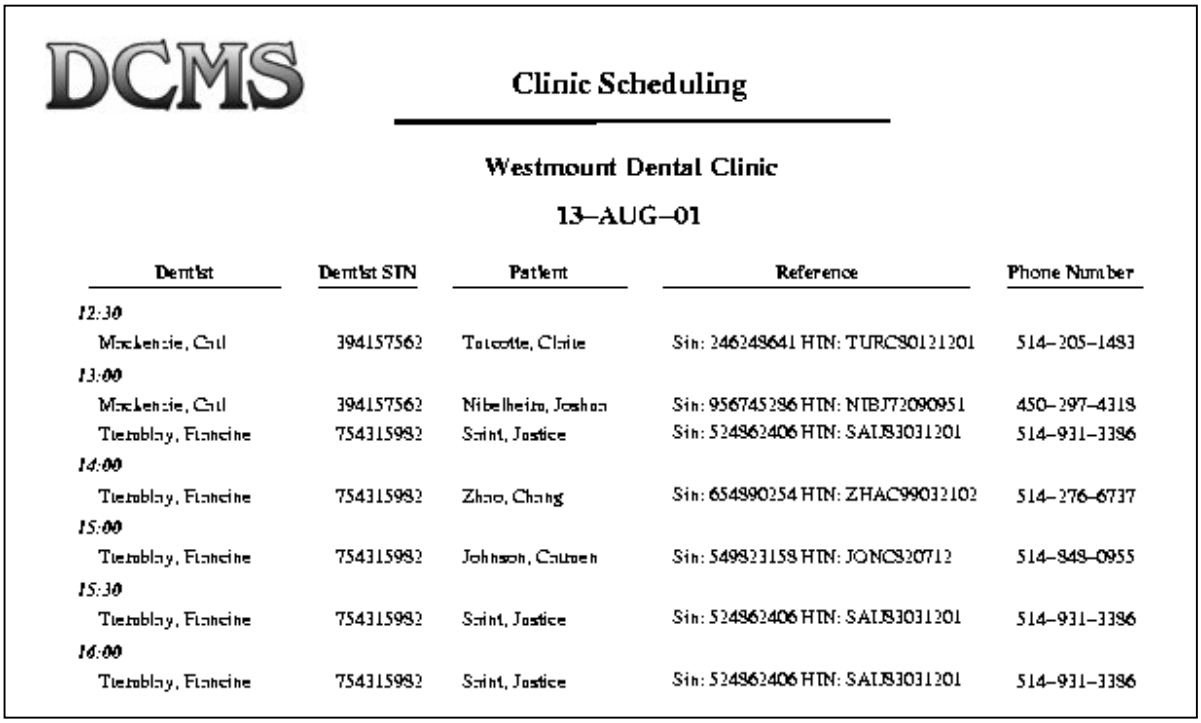
Dentist: Briere, Michel **Dentist SIN:598249725**

Patient Phone Nb	Hin Address	Patient SIN	Bill Number	Total	Balance
Briere, Michel	BRIM71090120	598249725	00000028	$ 200.00	$ 5.00
819-687-9235	3800 Bld. Cartefeut, St-Gregoire(Qc),G7M 4X2				
Farhat, Moussa	FARM79101402	843248347	00000030	$ 1,035.00	$ 5.00
514-696-1890	2 Rue Collingwood, Dollard Des Ormeaux(QC), H9A 3J3				
Mackenzie, Carl	MACC72061478	394157562	00000027	$ 200.00	$ 5.00
450-384-7829	1424 Canard St., Laval(QC),H7S 2G5				
Tremblay, Francine	TREF47073001	754315982	00000029	$ 200.00	$ 5.00
514-933-3158	473 Av Grosvenor,Westmount (QC),H3Y 2S5				

Figure 28: Example of Unpaid Bill Report

Clinic Scheduling Report

The Clinic Scheduling Report allows the user to print the list of appointments for all clinics or a given clinic on any period of time.



DCMS

Clinic Scheduling

Westmount Dental Clinic

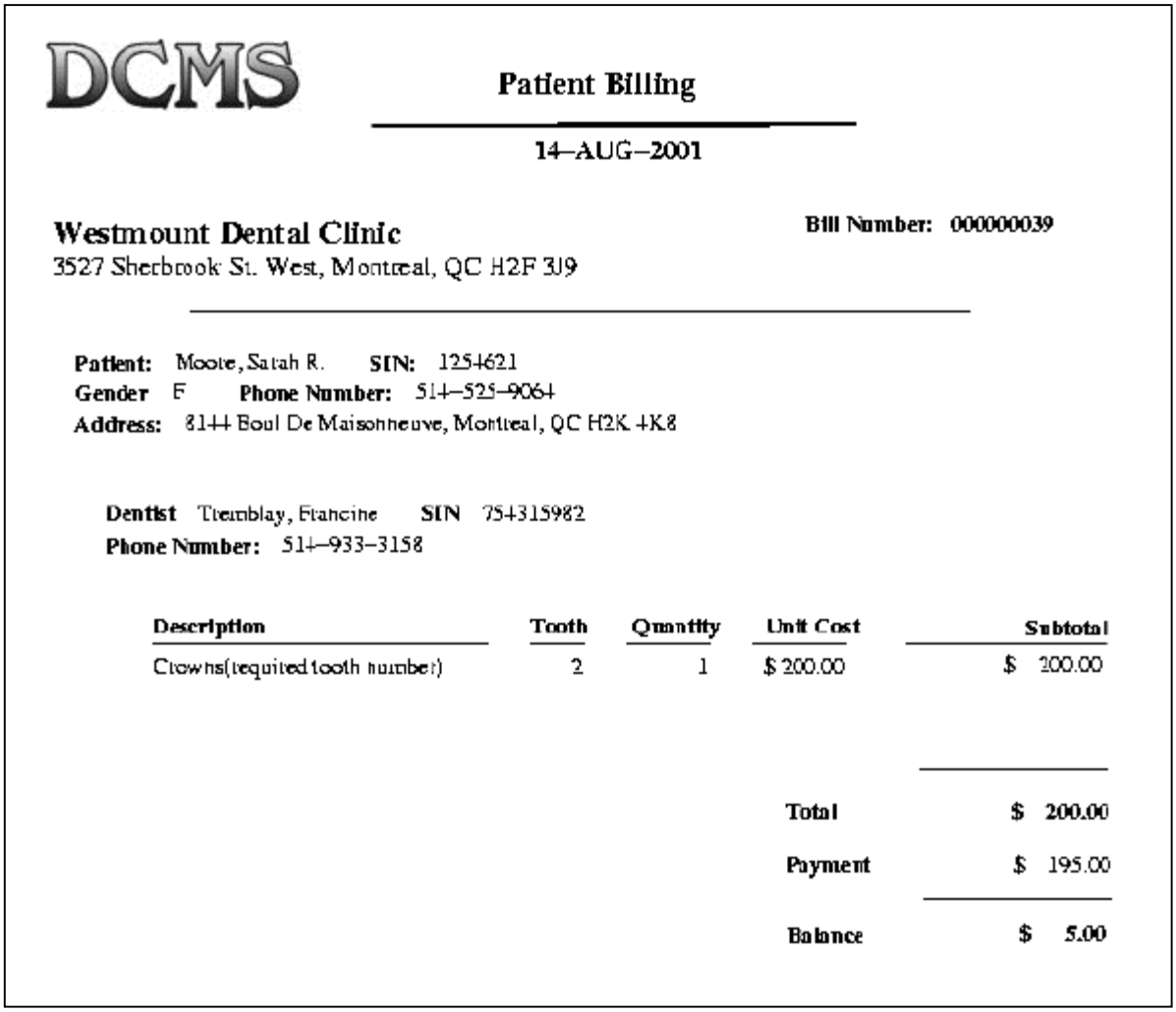
13–AUG–01

Dentist	Dentist SIN	Patient	Reference	Phone Number
12:30				
Mackenzie, Cnil	394157562	Tatcotte, Clnite	Sin: 246248641 HIN: TURCS0121201	514–205–1483
13:00				
Mackenzie, Cnil	394157562	Nibelheito, Joshon	Sin: 956745286 HIN: NIBJ72090951	450–297–4318
Tierablay, Fiancine	754315982	Saint, Justice	Sin: 524862406 HIN: SAIJS3031201	514–931–3386
14:00				
Tierablay, Fiancine	754315982	Zhao, Chang	Sin: 654890254 HIN: ZHAC99032102	514–276–6737
15:00				
Tierablay, Fiancine	754315982	Johnson, Caitoen	Sin: 549823158 HIN: JONCS20712	514–848–0955
15:30				
Tierablay, Fiancine	754315982	Saint, Justice	Sin: 524862406 HIN: SAIJS3031201	514–931–3386
16:00				
Tierablay, Fiancine	754315982	Saint, Justice	Sin: 524862406 HIN: SAIJS3031201	514–931–3386

Figure 29: Example of Clinic Scheduling Report

The Billing Report

The Billing Report allows the user to print a hardcopy version of the billing information for a specific bill number. It will display the header of the clinic, the patient and the treating dentist, as well as the details and charges of the treatment performed.



DCMS

Patient Billing

14–AUG–2001

Westmount Dental Clinic
3527 Sherbrook St. West, Montreal, QC H2F 3J9

Bill Number: 000000039

Patient: Moore, Sarah R. **SIN:** 1254621
Gender F **Phone Number:** 514–525–9064
Address: 8144 Boul De Maisonneuve, Montreal, QC H2K 4K8

Dentist Tremblay, Francine **SIN** 754315982
Phone Number: 514–933–3158

Description	Tooth	Quantity	Unit Cost	Subtotal
Crowns(requited tooth number)	2	1	$ 200.00	$ 200.00

Total	**$ 200.00**
Payment	**$ 195.00**
Balance	**$ 5.00**

Figure 30 Example of Patient Billing Report

Division of Labour

Task Assignments

Yann McCready: Team Leader, Project Coordinator, GUI Artist, Controller

Gaston D. Vaillancourt: Head Technical Manager, Test Manager, GUI Artist

François-Michel Brière: Dental Advisor, GUI Artist

Frederic Rioux: Database Programmer, Head Database Designer

Marc-André Laverdière: Technical Writer, Controller, Test Data Developer, Report Designer

All members: Database Designer, GUI Programmer & PL/SQL Programmer, Database & Application Tester

Task Descriptions

- **Team Leader:**
  The team leader is the official representative of the team to the instructor.
- **Project Coordinator:**
  The project coordinator manages the project and handles team meetings and schedules.
- **Technical Manager:**
  The technical Manager validates each step of the development phase, provides supplemental analysis on design and offers training references to the other team members.
- **Dental Advisor:**
  The dental advisor models the real-life situations regarding the dental clinic environment.
- **Database Programmer:**
  The database programmer implements the relationship schemas and restrictions.
- **Technical Writer:**
  The technical writer produces documentation and reports.
- **Test Manager**
  The test manager coordinates the final testing phase and the preliminary testing of the database.
- **Test Data Developer:**
  The test data developer programs scripts to populate the tables with test data.
- **Report Designer:**
  The report designer programs the reports and the forms to invoque them.
- **Controller:**
  The Controller validates that the project respects the specifications.
- **GUI Artist:**
  The GUI artist creates the major building blocks of the graphical user interface, evaluates user-friendliness and determines a standardized ‘look-and-fell’ for the application based on focus groups.
- **Database Designer:**
  The database designer participates in the overall design phase.
- **GUI Programmer & PL/SQL Programmer:**
  The GUI programmer and PL/SQL programmer implements the GUI in Oracle Forms and implements PL/SQL functions for Oracle Forms triggers.
- **Database & Application Tester:**
  Populates tables and tests the database with sample queries. Also ensure the correct functioning of the GUI program.

APPENDIX: Business and Requirements Specifications

The specification list has been provided from the user point of view, and therefore needed to be rewritten using a database designer/developer approach.

Database Requirements

The database must be able to handle large amount of data based on multiple years of services. As such, an optimal storage of the information is to be considered. The DCMS must be able to handle the following information:

Dentists' records

- A first and last name
- A license number
- Birth Date
- Specialty
- Address and contact information
- SIN
- Emergency contact information

Patients' records

- A first and last name
- SIN
- Gender
- Birth Date
- Address
- Phone number
- The patient's primary dentist
- Account information
- Health's notes
- Health Insurance Number

Billing information

- All the Dental Codes of the treatments received
- The Total
- The Balance to be paid
- The patient who received treatment and the treating dentist

Clinic information

- The name of the clinic
- The address of the clinic
- The schedule of the clinic
- The number of examination rooms in the clinic

Scheduling information

- All the time slots to be allocated are of 30 minutes
- All the time slots make a bridge between a dentist and a clinic to allow an appointment
- The schedule is dependent on the clinic’s opening hours
- The schedule relates to a patient needing to receive treatment

Feature Requirements

A basic set of user-friendly features is necessary in order to allow the good usage of the DCMS by the office assistants and dentists in order to maximize productivity.

Form Organization

The forms should be implemented in a way to reduce typing as much as possible. Furthermore, the following forms are to be implemented in the DCMS, presented in hierarchical order:

- Main Screen
  - Patient Information Form
    - Allows creation of new patients
    - Allows updating the information of existing patients
    - Displays the patient's account status
  - Clinic Information Form
    - Allows creation of new clinics
    - Allows updating the information of clinics
  - Dentist Information Form
    - Allows creation of new dentists
    - Allows updating the information of existing dentists
  - Dentist's Scheduling Form
    - Allows the dentist to insert appointment availabilities
  - Patient's Scheduling Form
    - Uses the dentists' availabilities to schedule a patient
    - Allows the creation of a new bill
  - Dental Code Management Form
    - Allows creation of new dental codes
    - Allows updating the information of current codes
  - Billing Form
    - Allows the insertion of dental codes in the patient's bill
    - Allows to print the bill via a report
  - Report Centre
    - Allows to print a clinic's schedule for the upcoming week via a report
    - Allows to print the bills unpaid to the treating dentist via a report

Report Generation

The reports are non-interactive components of the DCMS used to display static information. The following reports should be generated by the DCMS:

- Bill Generation
  - Will display the contents of a specific bill, as well as its total and balance
  - Requires the bill information key in order to select the appropriate bill
- Schedule Generation
  - Will display the schedule of a clinic on a certain interval of time
  - Requires the clinic information key in order to select the appropriate clinic
  - Requires the interval of time upon which to display the scheduling
  - The default interval of time is from the next Monday to the Friday after.
- Unpaid Bill Generation
  - Will display the contents of all bills which balance is not to zero based on the dentist who performed the treatment billed.
  - Requires the dentists' information key in order to select the bills related to the good dentist.

APPENDIX: Development Process

Brainstorming

The brainstorming step of the process involved each member of the team using the requirements and developing an E/R model on their own.

Concept Definition and Preliminary ER Diagram Synthesis

As, from last step, each member came up with a different view of the schema, as well as different approach for the appointment system. After discussion, it was determined that the dentist's schedule would be used as the key point of the whole scheduling and billing system. We also determined the weak relationships and entities in the diagram and reviewed the specifications, insuring that all the user's requirements were covered.

Validation of E/R Diagram and Conversion into Relational Schemes

The E/R diagram previously reached a near-complete level and the relational schemas were translated from the E/R model. The team refined the E/R diagram to its final form, modifying the relational schemes at the same time.

In order to optimize the data size usage, the relations have been evaluated and merged together, greatly lowering the redundancy of foreign keys in tables of little use. Once that step has been completed, we evaluated the tables and determined that they were respecting the BCNF standards and that, therefore, would not need to be divided. The final result is available in the Normalized Relational Design section.

Development Standards, Overview of Preliminary Report and Database Design

At this meeting, the final report has been presented for verification and the first evaluation of the development standards was being laid out. A summary of all the constraints and assumptions was also built in order to clarify what was before decentralized pieces of information. Also, the team discussed the required additions to the database for views and indexes.
The first application-related concerns were expressed at that meeting.

Approval

The team members all voted for the acceptance of the task distribution and the design. Unanimity was required before going further. The preliminary report was also validated in this process done by e-mail.

Introductory GUI Design

The team met again to discuss GUI issues, mainly regarding the ‘look and feel’ and user-friendliness of the DCMS Client Interface. At this moment, the key interface element, the scheduler, has been designed and various features were agreed upon in order to facilitate end-user-friendliness.

Basic Database Implementation

A first database construction script was built, implementing the basic server-based constraints. Some testing data was added and the basic forms (Dentists, Clinics and Patients) were programmed. At this moment, nothing but text fields were present, but it allowed a first overview of the general user environment. Some other assumptions and constraints considerations showed up and were addressed.

General Development

The work has been further subdivided among team members and each team member entered a learning phase at the same time. The team members had to familiarize with the PL/SQL language as well as the specifics of Oracle Form and Oracle Report.

The forms and reports were designed in that phase using a main programmer/secondary developer method. In this development scheme, a main developer was assigned to the development of a form in order to lie out the form's functionality and interface, then allowing another team member to develop a part of the features. As such, a better understanding of the other's development process was achieved, as well as allowing team member to specialize themselves in the development of a certain category of features.

Data Population and Testing

Standard data population scripts were designed and validated in order to ensure efficient and reliable testing environment, therefore building the DCMS Test Database.

Besides the usual data entered during the other stages of development, a more extensive testing has been done after all the pieces were developed. The behavior of all forms and reports has been validated to ensure that specifications were respected.

Integration and Quality Insurance

All the separate pieces of code, forms and reports, were put together by the running and invoking splash screen program and the specifications and constraints were double-checked.

Demo

In the presence of the team members and of the instructors, the final beta version of the DCMS Test Database and DCMS Client Interface were presented for evaluation.

APPENDIX: Preliminary Report

Introduction

The Dentist Project constitutes our end-of-term database project, a mandatory requirement for the successful completion of the COMP 353 course. This project consists of the implementation of an Oracle database to support various scheduling and billing functions, as well as a Graphical User Interface (GUI) to facilitate its functionalities by the users, using Oracle Form (PL/SQL) and Oracle Report.

Final Design of the Database

The database system that will be presented in the following section is the fruit of multiple hours of teamwork and individual design. For more information regarding the process that led to the final versions of the E/R diagram and the normalized relational designs, please consult Appendix A. Also present in Appendix B is the list of the team members as well as their individual responsibilities.

Final E/R design

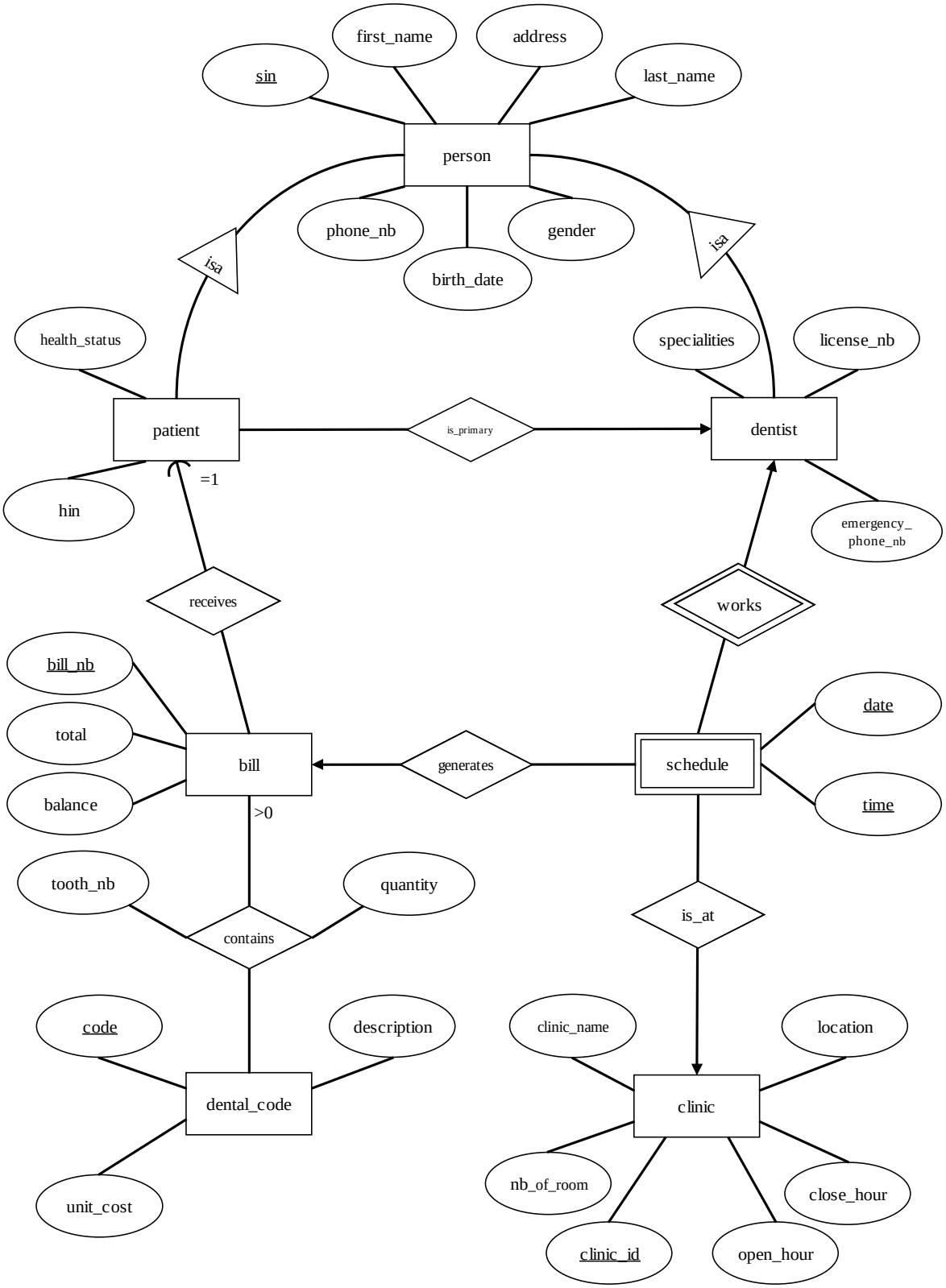


Figure 31 : Final E/R Diagram

The Normalized Relational Design & Functional Dependencies

TB_patient (patient_sin, first_name, last_name, address, phone_nb, birth_date, gender, health_status, hin, primary_dentist_sin)
patient_sin → first_name, last_name, address, phone_nb, birth_date, gender, health_status, hin, primary_dentist_sin

TB_dentist (dentist_sin, first_name, last_name, address, phone_nb, birth_date, gender, specialties, licence_nb, emergency_phone_nb)
dentist_sin → first_name, last_name, address, phone_nb, birth_date, gender, specialities, licence_nb, emergency_phone_nb

TB_clinic (clinic_id, clinic_name, location, close_hour, open_hour, nb_of_room)
clinic_id → clinic_name, location, close_hour, open_hour, nb_of_room

TB_dental_code (code, unit_cost, description)
code → unit_cost, description

TB_bill (bill_nb, total, balance, patient_sin)
bill_nb → total, balance, patient_sin

TB_bill_contains
(code, bill_nb, quantity, tooth_nb)
code, bill_nb, tooth_nb → quantity

TB_schedule (time, date, dentist_sin, clinic_id, bill_number)
time, date, dentist_sin, clinic_id → bill_number

All these relations are compliant to the BCNF standard.

Application-Based Constraints

1. The insertion of the dentist's schedule should set the bill_number field to NULL. Only the dentist scheduling form can perform insert/delete operations and only the patient scheduling form should update the bill_number field.
2. The day of the week (i.e. Monday, Tuesday, etc.) is to be determined by algorithm.

Referential Integrity Constraints

1. No dentist can be deleted from the database if there were any billed appointments.
2. No patient can take an appointment without a dentist having allowed that particular time slot and clinic in his schedule.
3. No dental code can be removed if it has been used in any bill.
4. No patient can be deleted unless it has no bill, paid or unpaid.
5. No element of the schedule can be deleted if the bill_number field is not NULL.

Server-Based Constraints

1. A dentist's scheduling inside a clinic is limited by the rooms available and the clinic's opening hours. That means that no dentist can give his or her schedule for a clinic if the sum of the dentists in that clinic at that period of time is equal to the number of rooms in the clinic.
2. The balance and total can only be greater or equal than zero.
3. The SIN and HIN is of fixed size.
4. The bill number is determined sequentially and incrementally and should be automatically generated (i.e. no insertion on the table should bother about the bill number).
5. The time of the beginning of the appointment is a multiple of halves of hours.
6. The cost of every dental code is superior or equal to zero.
7. Every dental code is an integer greater than zero.
8. The quantity has to be greater than zero.
9. None of the clinic information can be set to NULL.
10. The clinic_ID number is determined sequentially and incrementally and should be automatically generated (i.e. no insertion in the table should bother about the clinic_ID).
11. Date must be a valid date and must be after 2000.
12. None of the fields from TB_Patient and TB_Dentist can be left to NULL.
13. No patients' primary dentist can have the same SIN that the patient's SIN (i.e., a dentist being a patient in its own clinic cannot be its own dentist).

Assumptions

1. The scheduling is done manually by office assistants based on the dentist's wished schedule. All the conflict resolution is handled manually according to some corporate policy and should not be taken care of by the system.
2. The patient's online record doesn't include any non-textual data, such as X-rays.
3. The dentists can work in any given number of clinics in the same day, and no policy regarding travel and eating times are to be handled by the system.
4. No room is directly assigned to any given dentist. The rooms are for use by all dentists.
5. Each dentist is assumed to have provided a phone number as emergency contact information
6. No overbooking is to be done and the dentist handles only one appointment in a single time slot.
7. All the clients must have an appointment in order to see a dentist.
8. The appointment doesn't include the motive of the visit or the treatments to be performed.
9. We assume that a dentist with the required specialty performed specialty-related treatments.
10. No information for any employee other than the dentists of the dental company is held in the database.
11. The SIN and HIN are assumed to be correct, and no validation is performed.
12. Each clinic opens and closes at the same set of hours, from Monday to Friday.
13. A dentist can also be a patient.
14. The specialties field of the dentist is to be represented as a character string for the sake of simplicity.
15. We assume a new database with no data previous from the year 2000.
16. We assume that the dentists' professional information (license number, address, etc.) is always correct and that the license is always valid.
17. All patients and dentists are considered as active.
18. An appointment for a patient implies a bill, whether or not the patient was present to receive treatment.
19. There is no support for holidays in the system.

Keys

TB_patient
Patient has the patient_sin as a primary key.

TB_dentist
Dentist has dentist_sin as a primary key.

TB_clinic
Clinic has its own clinic_ID as primary key.

TB_dental_code
Dental_Code has its own internal primary key, being the code attribute.

TB_bill
Bill has its own internal primary key, being the bill_nb. Also, it includes patient_sin as a foreign key from patient_sin in the TB_patient table.

TB_contains
Contains has the tooth_nb internal key, but does not act as a primary key. The code key comes from the code attribute in TB_dental_code.

TB_schedule
Schedule uses the date and time internal attributes to contribute to the key. The attributes dentist_sin (from TB_dentist), clinic_id (from TB_clinic) and bill_number(from TB_Bill) are foreign keys, even tough bill_number is not part of the superkey in this relation.

Appendix A: Methodology and Evolution

Analysis

The first step of the process for the team has been a careful evaluation of the requirements document, this step being done separately by each team member. A formalized specification list has then been issued by the combination of all points of view. Also, most assumptions have been laid out at that point in time.

Preliminary Assumptions

The preliminary assumptions were drawn before the formalization of the specifications by noticing gaps in the provided requirements. Such gaps meant some room of maneuvering from our part and hence were the source of our first assumptions. We also used the help of a dentist to provide a better understanding of the inner workings of the dental business in order to facilitate analysis and modeling of the requirements.

Assumptions:

- The scheduling is done manually by office assistants based on the dentist's wished schedule. All the conflict resolution is handled manually according to some corporate policy and should not be taken care of by the system.
- The patient's online record is not multimedia, meaning that it is stored as text data.
- The dentists can work in any given number of clinics in the same day, and no policy regarding travel and eating times are to be handled by the system.
- No room is directly assigned to any given dentist. The rooms are for use by all dentists.
- Each dentist is assumed to have provided a phone number as emergency contact information
- No overbooking is to be done and the dentist handles only one appointment in a single time slot.
- All the clients must have an appointment in order to see a dentist.
- The appointment doesn’t include the motive of the visit or the treatments to be performed.
- We assume that a dentist with the right specialty performed specialty-related treatments.
- No information for any other employee of the dental company is held in the database.

Rewritten Specifications

The specification list has been provided from the user point of view, and therefore needed to be rewritten using a database approach.

Database Requirements:

- The database holds Dentists' records, including
  - A first and last name
  - A license number
  - Specialty
  - Contact information
  - SIN
  - Address.

- The database holds Patients' records, including

    - A first and last name
    - SIN
    - Gender
    - Birth Date
    - Address
    - Phone number
    - The patient's primary dentist
    - Account information
    - Health's notes
    - Health Insurance Number

- The database holds billing information. Each bill is linked to a patient and to a serving dentist and also includes:
    - All the Dental Codes of the treatments received
    - The Total
    - The Balance to be paid

- The database holds clinic information, including:
    - The name of the clinic
    - The address of the clinic
    - The schedule of the clinic
    - The number of examination rooms in the clinic

- The database holds scheduling information in the form of a daily schedule for each dentist. These availabilities are in consideration of a clinic and will relate to a patient.
    - All the time slots to be allocated are of 30 minutes.
    - All the time slots make a bridge between a dentist and a clinic to allow an appointment.

GUI Requirements:

- The following forms are required:
    - Patient information form allows adding patients and modifying current patients' data.
    - Dentist information form allows adding dentists and modifying current dentists' data.
    - Clinic information form allows adding clinics and managing the current clinics' information.
    - Dentist scheduling from allows the dentist to enter his working hours into the system.
    - Patient scheduling form uses the working hours provided by the system to give an appointment to the patient.
- The following reports must be presented:
    - Listing of all the upcoming appointments for the following week, sorted by clinic.
    - Listing of all the indebted customers, sorted by dentist.

Server-Based Constraints

- A dentist's scheduling inside a clinic is limited by the rooms available and the clinic's opening hours.
- No patient can take an appointment without a dentist having allowed that particular time slot and clinic.

Application-Based Constraints

- The day of the week (i.e. Monday, Tuesday, etc.) is to be determined by algorithm.

Step-by-Step Design

Brainstorming

The brainstorming step of the process involved each member of the team using the requirements and developing an E/R model on their own. An example of this step is shown in Figure 32.

Concept Definition and Preliminary ER Diagram Synthesis

As, from last step, each member came up with a different view of the schema, as well as different approach for the appointment system. After discussion, it was determined that the dentist's schedule would be used as the key point of the whole scheduling and billing system. We also determined the weak relationships and entities in the diagram and reviewed the specifications, insuring that all the user's requirements were covered.

Validation of ER Diagram and Conversion to Relational Schemes

The E/R diagram previously reached a near-complete level and the relational schemas were translated from the E/R model. The team refined the E/R diagram to its final form, modifying the relational schemes at the same time.

The first relational schema, built using the mechanical process described in the book is shown as Figure 34.

In order to optimize the data size usage, the relations have been evaluated and merged together, greatly lowering the redundancy of foreign keys in tables of little use. Once that step has been completed, we evaluated the tables and determined that they were respecting the BCNF standards and that, therefore, would not need to be divided. The final result is available in the Normalized Relational Design section.

Development Standards, Overview of Final Report and Database Design

At this meeting, the final report has been presented for verification and the first evaluation of the development standards was being laid out. A summary of all the constraints and assumptions was also built in order to clarify what was before decentralized pieces of information. Also, the team discussed the required additions to the database for views and indexes.
The first application-related concerns were expressed at that meeting.

Indexes

- **TB_patient**: index on last_name and HIN
- **TB_dentist**: index on last_name
- **TB_clinic**: index on clinic_name
- **TB_bill**: index on patient_sin
- **TB_schedule**: index on bill_number

Final Approval

The team members all voted for the acceptance of the task distribution and the design. Unanimity was required before going further. The final report was also validated in this process done by e-mail.

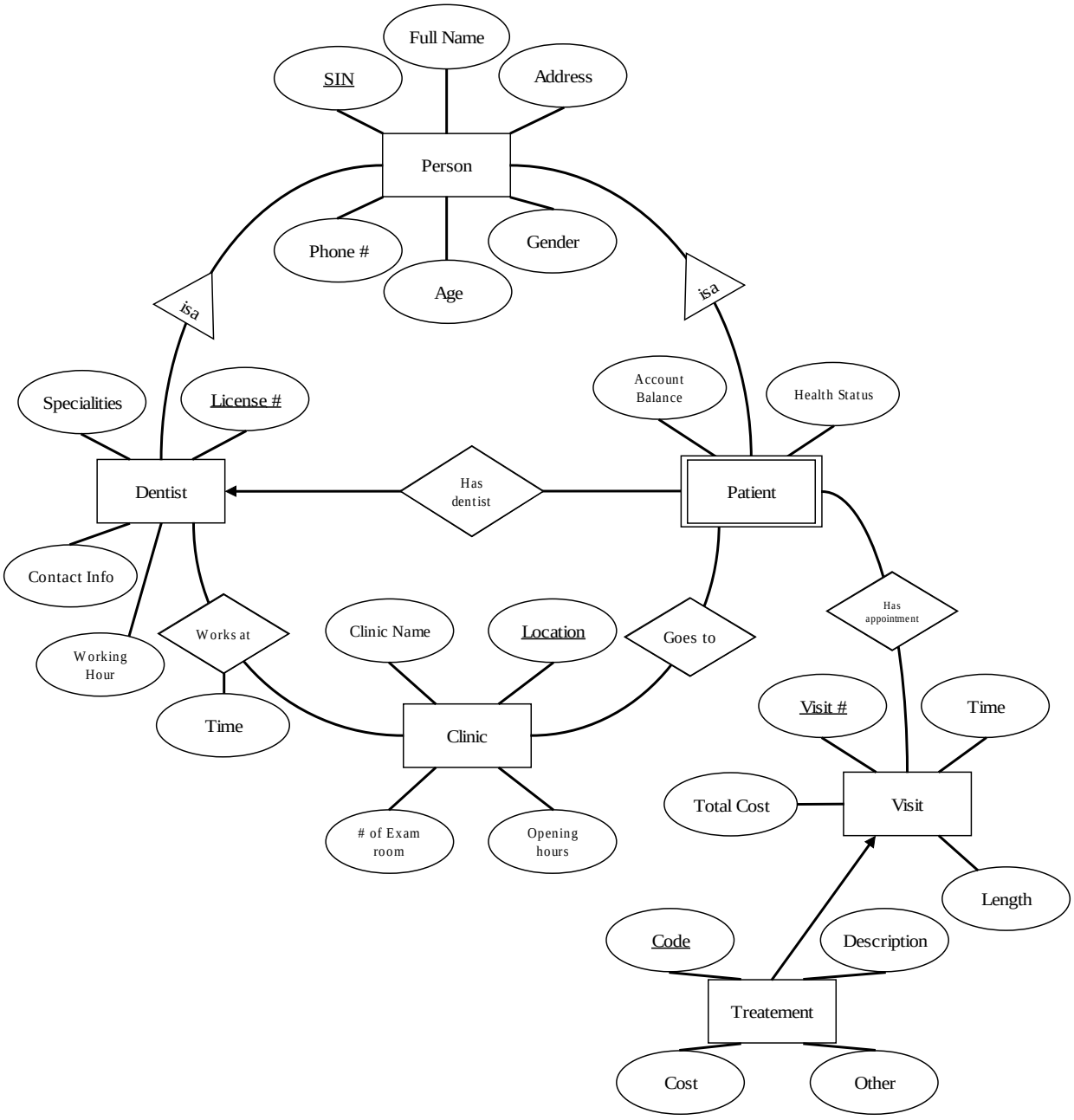


Figure 32: Example of Brainstorming Result from François-Michel Brière

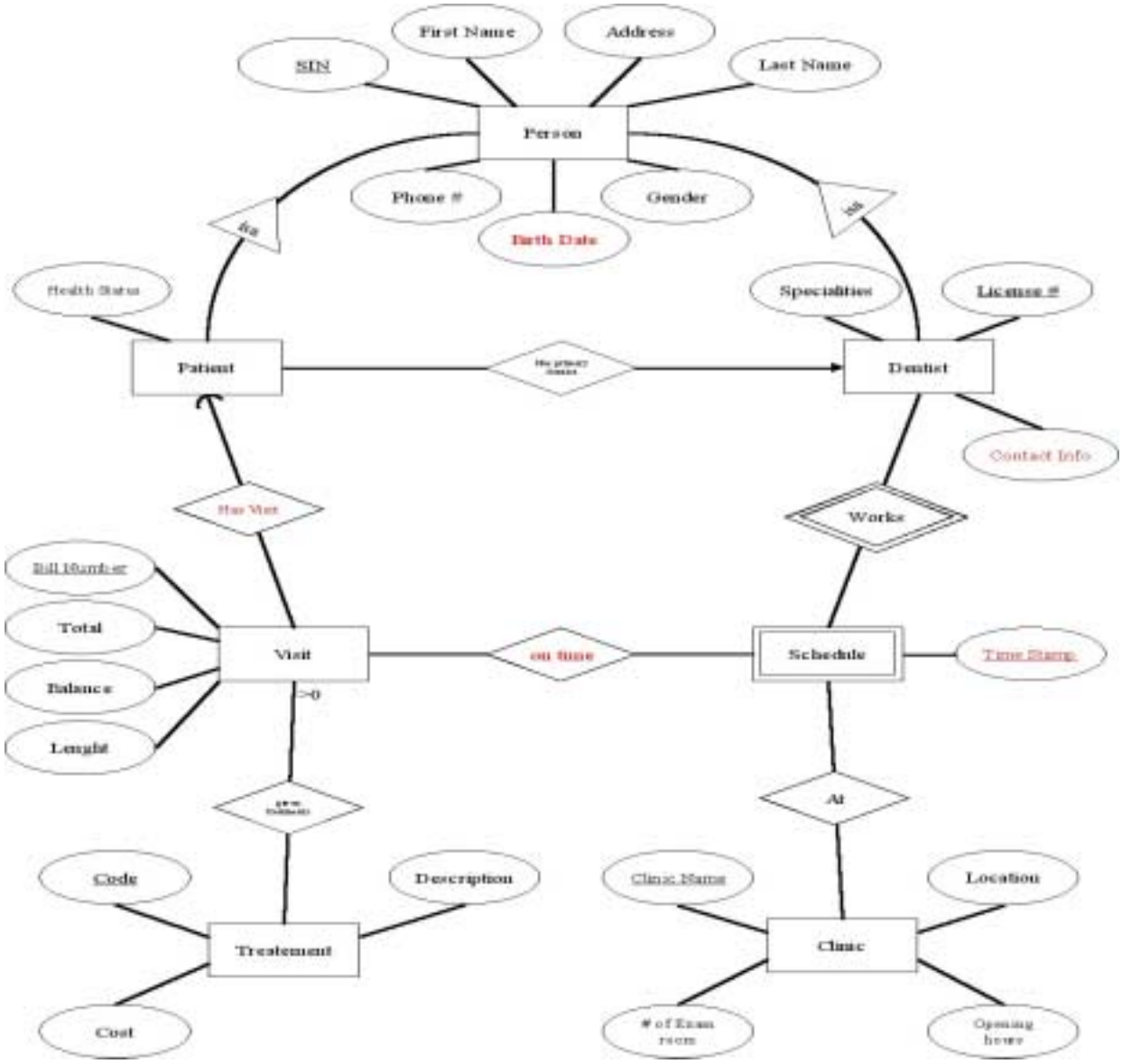


Figure 33 : Intermediate E/R diagram

Person(<u>SIN</u>, LastName, FirstName, Address, Phone#, BirthDate, Gender)
Patient(<u>SIN</u>, HealthStatus)
Dentist(<u>SIN</u>, Licence#, Specialities, ContactMobile#)
IsPrimary(PatientSIN, DentistSIN)
Appointment(<u>BillNumber</u>, Total, Balance, Length)
HasAppointment(<u>SIN, BillNumber</u>)
Treatment(<u>Code</u>, UnitCost, Description)
ReceivedTreatment(<u>Code, BillNumber</u>, Quantity)
Schedule(<u>TimeStamp, SIN, Room#, ClinicName</u>)
TakesPlaceAt(<u>BillNumber, TimeStamp, SIN</u>)
Clinic(<u>ClinicName</u>, Location, #OfRoom, OpenHour, CloseHour)

Figure 34: Preliminary Conversion to Relations

Appendix B: Team Members

Task Assignments

Yann McCready: Team Leader, Project Coordinator, GUI Programmer, Controller

Gaston D. Vaillancourt: Head Technical Manager, Test Coordinator

François-Michel Brière: Dental Advisor, GUI Programmer

Frederic Rioux: Database Programmer, Head Database Designer

Marc-André Laverdière: Technical Writer, Controller, Assistant Coordinator

All members: Database Designer, GUI Artist, PL/SQL Programmer, Database and Application testing

Task Descriptions

- **Database Designer:**
  Draws E/R Diagram and participate in the overall design phase.
- **GUI Artist:**
  Design GUI layout and determine the GUI standards for the programmers.
- **PL/SQL Programmer:**
  Implement functions for the GUI
- **Controller:**
  Validates the work is done according to specifications
- **GUI Programmer:**
  implements the GUI in Oracle Forms
- **Database Programmer:**
  Writes SQL statements to implement the relationship schemas and the restrictions
- **Team Leader:**
  official representative of the team to the teacher
- **Project Coordinator:**
  Manages the project and handles team meetings and schedules.
- **Technical Manager:**
  Validates each step of the development phase, provides supplemental analysis on design and offers training references to the other team members.
- **Database and Application Tester:**
  Populates tables and tests the database with sample queries. Also ensure the correct functioning of the GUI program.
- **Test Manager**
  Coordinates the final testing phase and handles the preliminary test phase of the database.
- **Dental Advisor:**
  Provides a model of the real-life situations regarding the dental clinic environment
- **Technical Writer:**
  Produces reports and internal team documentation.