Demographic Surveillance System Implementation

- **Low-level structures of the PDA**
- **PC/PDA communication**

LUNDS TEKNISKA HÖGSKOLA
Lunds universitet

Bachelor thesis:
Anders Hovén
Johan Augustsson

LTH School of Engineering at Campus Helsingborg
Lund University
Box 882
SE-251 08 Helsingborg
Sweden

LTH Ingenjörshögskolan vid Campus Helsingborg
Lunds Universitet
Box 882
251 08 Helsingborg

Printed in Sweden
Media-Tryck
Biblioteksdirektionen
Lund University
Lund 2004

Abstract

Demographic Surveillance System Implementation

- Low-level structures of the PDA
- PC/PDA communication

This thesis report describes the continuing work to develop a tool for demographic research at CIDS (Centro de Investigación en Demografía y Salud Universidad Nacional Autónoma de Nicaragua), UNAN, Leon/Nicaragua.
The report is divided into four parts:

- Revision of the existing database as a result of newly found requierments
- Implementation of the low-level design on the PDA
- Communication between the PC and the PDA
- A mid-storage database on the client-PC

Key-words: ER-diagram, PDA, .Net

Sammanfattning

Demographic Surveillance System Implementation

- Low-level structures of the PDA
- PC/PDA communication

Denna examensrapport redogör för det fortsatta arbetet med att utveckla ett verktyg för demografisk forskning på CIDS (Centro de Investigación en Demografía y Salud Universidad Nacional Autónoma de Nicaragua), UNAN, Leon/Nicaragua.
Rapporten är uppdelad i fyra delar:

- Revision av den existerande databasen p.g.a. nytillkomna krav
- Implementering av lågnivå-design på PDA'n
- Kommunikation mellan PC och PDA
- En mellanlagrings-databas på klienten

Nyckelord: ER-diagram, PDA, .Net

Foreword

In October 2004 we got the question if we were interested in making our degree thesis in Nicaragua. The work should be a continuing on Nils Assarsson, Tommy Ljunggrens degree thesis, Demographic surveillance system database, done in mars to may in 2003 and Mattias Pedersen, Axel Mårtensson degree thesis Demographic Surveillance System Database - A conceptual description of the database, done in mars to may 2004.
Our work would be to implement the low-level software on the PDA and the communications between the PDA and the PC (the conduit). Furthermore our thesis will include a revision of the existing ER-diagram that was established by Mattias Pedersen and Axel Mårtensson.

List of contents:

1 Introduction

The Centre for Demographic and Health Research (CIDS) in Leon/Nicaragua is managing an epidemiological database, which enables studies of reproductive events and child mortality. The database has been used in inter and multidisciplinary studies on fertility and monitoring trends and social determinants of infant and under-five mortality. The database was also utilized in various sub-studies on maternal mortality, teenage sexuality and reproduction, sexual behaviour, domestic violence and impact of women's access and control over resources on child survival. A follow-up and continuation of these studies are planned. The database has also been used as a sampling frame for studies of other research projects at the Medical Faculty in León.

At present time, the information-collection to maintain and extend the data in the database is done only on paper forms. This is a time-consuming system that needs great manpower and has a high error risk. The idea is to computerize the process and use handheld computers (PDA) instead of the paper forms. The PDA will be able to validate the input "on-spot" when the fieldworkers are conducting interviews. This function will prevent from getting obviously false data in the database.

This bachelor thesis was made at CIDS, UNAN, León, Nicaragua.
Centro de Investigación en Demografía y Salud Universidad Nacional Autónoma de Nicaragua.

1.1 Extent

This thesis is divided into four parts:

- A revision of the existing database (DSSdb) to determine if more/less attributes are needed.
- Implementation of the low-level structures in the PDA.
- Implementation of the communication/data transfer between the PDA and Client-PC.
- Specification of a mid-storage database on the client-PC.

2 Limitations and deviations

The conduit

-If a hotsync is made two times instead of only one when data is to be transferred to the PDA problems will occur. This is because the system thinks that the field-worker already have done the interviews and is checking in the data. This is a limitation that has to be corrected in further versions.

-We haven’t spent much time on making the system return understandable fault-messages. The messages that are returned from the system are meant for a programmer.

The PDA

-The memory in the PDA will only be sufficient for a limited number of households/persons. This is due to the limited amount of memory in the PDA and the fact that we haven’t spent much time in optimizing the memory usage.

3 Revision of the Database-description (DSSDb)

3.1 Introduction

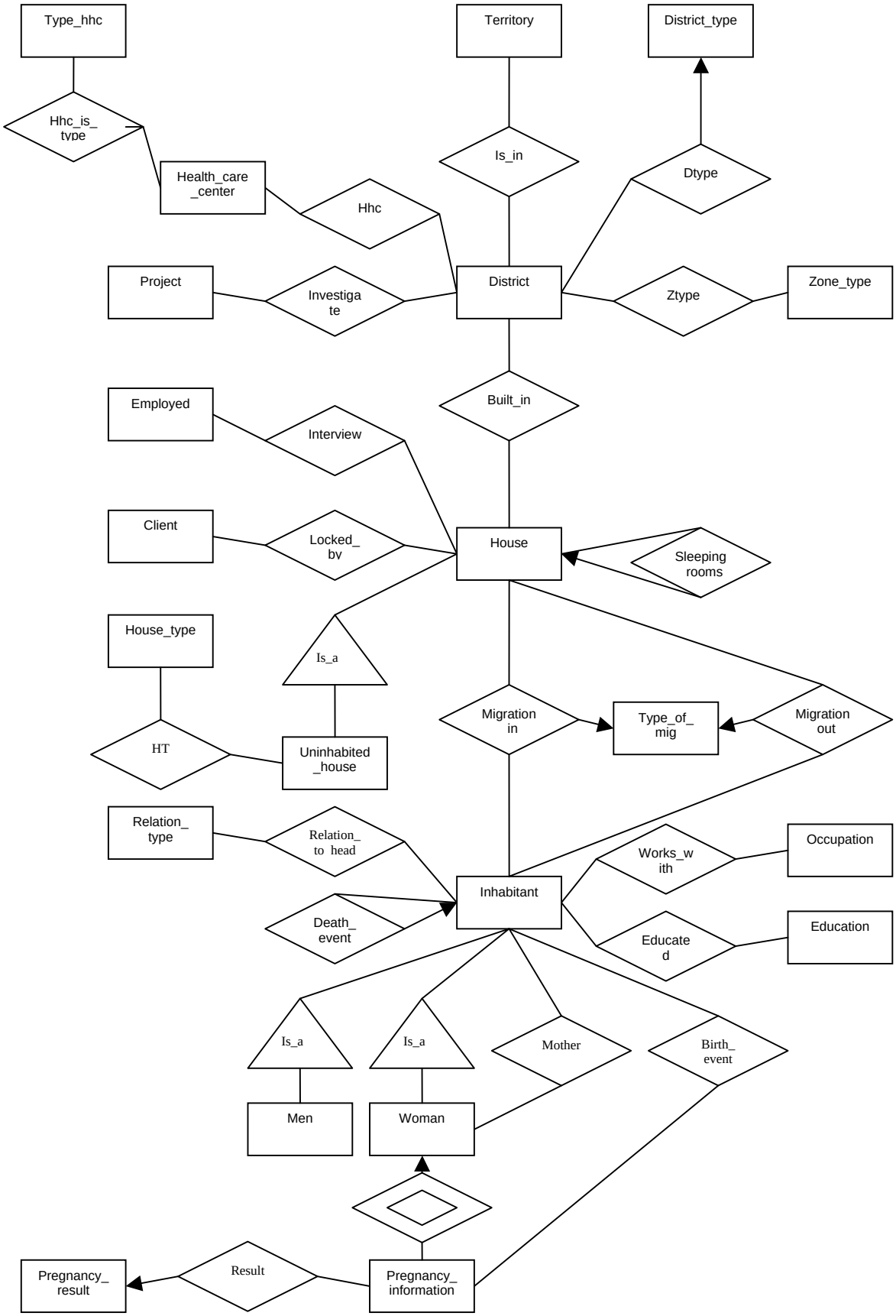
Because of small changes/updates in the requirement specification and to incorporate the database with the rest of the system we had to make a revision of the existing database. In this chapter we are going to describe what changes we have made and present the current database specification.

3.2 ER-Diagram

This chapter contains the new entity-relationship diagram for the database.

3.2.1 Overview

Figure 3.1 The ER-diagram, overview


Type_hhc
Territory
District_type
Hhc_is_
type
Is_in
Dtype
Health_care
_center
Hhc
Project
Investiga
te
District
Ztype
Zone_type
Built_in
Employed
Interview
Client
Locked_
by
House
Sleeping
rooms
House_type
Is_a
Migration
in
Type_of_
mig
Migration
out
HT
Uninhabited
_house
Relation_
type
Relation_
to_head
Works_w
ith
Occupation
Inhabitant
Death_
event
Educate
d
Education
Is_a
Is_a
Mother
Birth_
event
Men
Woman
Pregnancy_
result
Result
Pregnancy_
information

3.2.2 Detailed ER-diagram

Figure 3.2 Part of ER-diagram, detailed

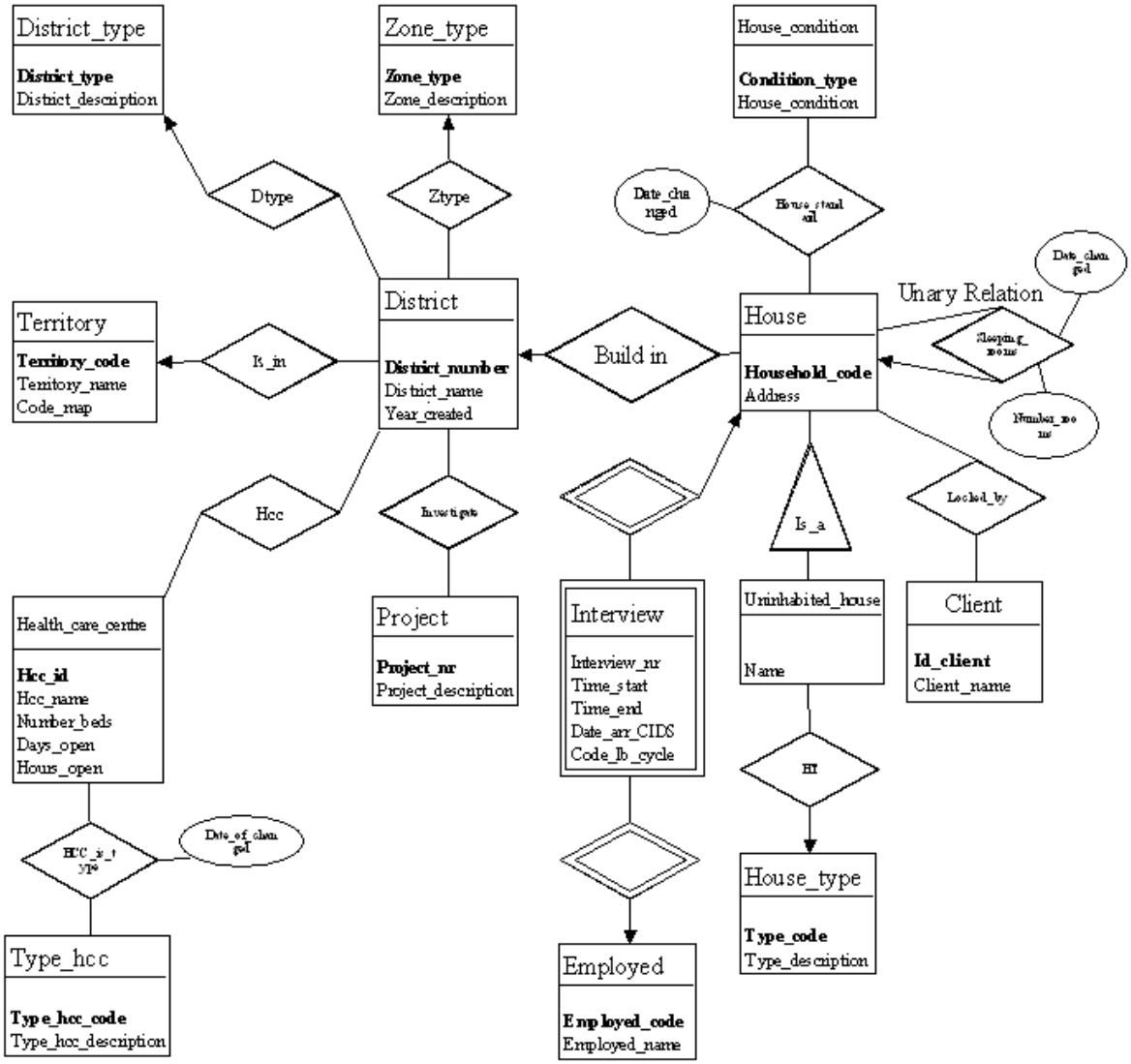
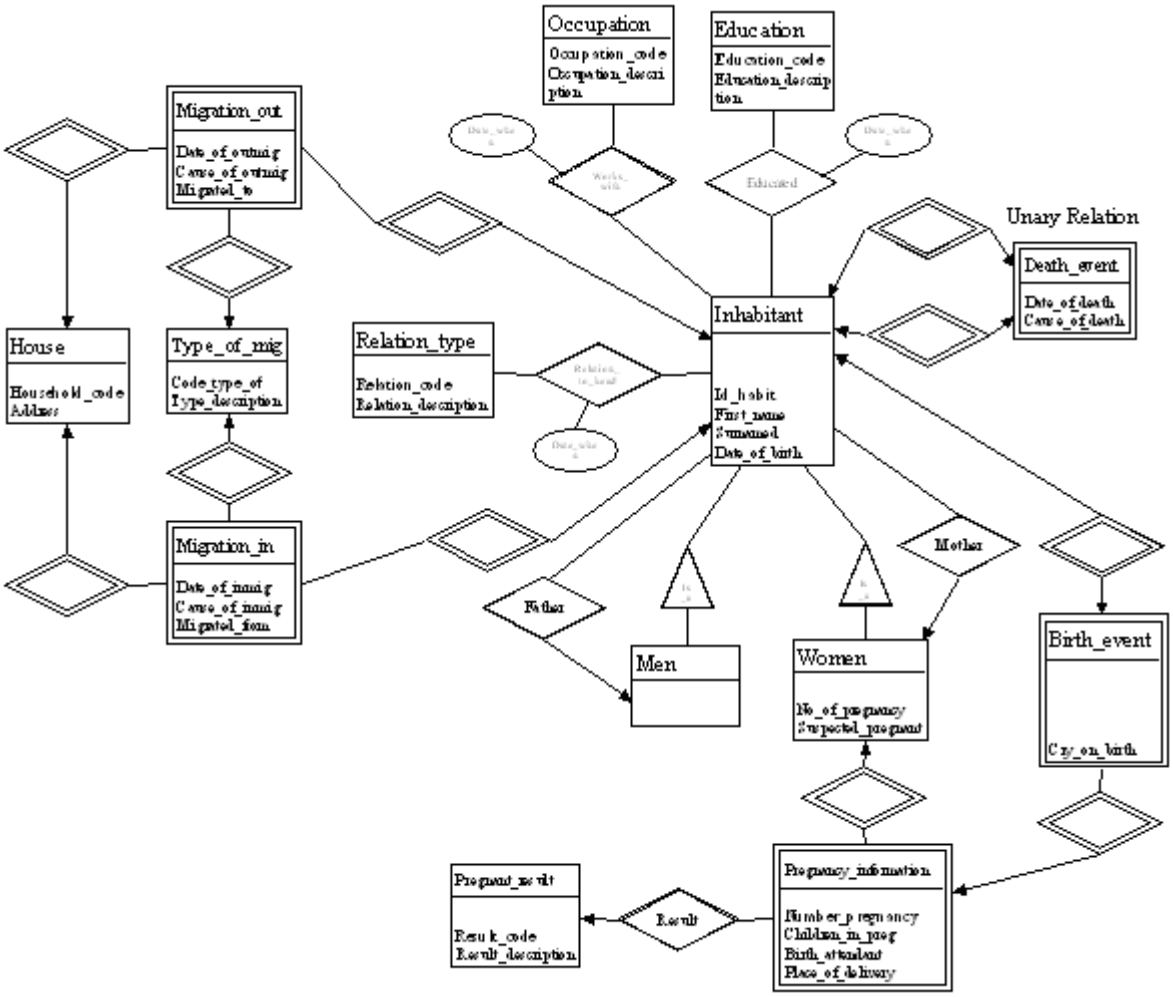


Figure 3.3 Part of ER-diagram, detailed



3.3 Entity specification

This chapter describes what the strong and weak entities are. What they stands fore in reality.

Birth_event
Weak entity, relation between Pregnancy_information and Inhabitant.
Contains data about a specific birth.
Because the old database contains information about 70000 inhabitants that not have the information about their birth, exist this relation to contain that information.
A Birth_event is when a new inhabitant is born.

Clients
Strong entity, Contains data about the clients that connects to the handheld.
Clients is a list with all clients.

Death_event
Weak entity, unary relation with Inhabitant. Contains data about the death of the inhabitant.
A Death_event is when an inhabitant has died.

District
Strong entity. Contains data about a district.
District is a small land area inside a territory. Houses exist in a district.

District_type
Strong entity. Contains information about which type a district can be.
District_type is a list with different type a district can be.

Educated
Weak entity, relation between Education and Inhabitant. Contains information when a person had a certain kind of education.

Education
Strong entity. Contains data about different educations.
Education is a list with different levels of educations.

Employed
Strong entity. Contains data about employed on CIDS. Used for identification of interviewers and computer operators.
An employed is a person that works at CIDS and handle interviews in the field or at the office.

Hcc_is_type
Weak entity, relation between Health_care_centre and Type_hcc. Contains information about when health care centre changed type.

Health_care_centre
Strong entity. Contains data about Health Care Centres.
A health care centre is where inhabitants can get medical help.

House
Strong entity. Contains data about houses.
House is a building in a district. All buildings in the district should be included.

House_condition
Strong entity. Contains information about different conditions.
House_condition is a list with the different condition types on a house.

House_standard
Weak entity, relation between House and House_condition. Contains information when a house changed condition.

House_type
Strong entity. Contains information about which types uninhabited house can be.
House_type is a list with the different types of uninhabited houses.

Inhabitant
Strong entity. Contains primary data about inhabitants.
Contains common information about inhabitants.
An Inhabitant is a person in the study area.

Interview
Weak entity, relation between House and Employed. Contains data about interviews.
An Interview is a document where the field worker fills in information about who lives in a house and what status they have.

Locked_by
Weak entity, Contains data about which clients that have locked which houses.
Locked by is a list of all houses currently locked.

Men
Sub entity to Inhabitant.
A man is a male inhabitant in the study area.

Migration_in
Weak entity, relation between House and Inhabitant. Contains data about migration in.
Migration_in is when an inhabitant moves in to a house in the study area.

Migration_out
Weak entity, relation between House and Inhabitant. Contains data about migration out.
Migration_out is when an inhabitant moves out from a house in the study area.

Occupation
Strong entity. Contains information about which categories of occupations a habitant can have.
Occupation is a list with the different categories of occupations.

Pregnancy_information
Weak entity, existence dependent, needs Women for identification. Contains data about pregnancies.
Pregnancy_information is information about a woman when she is pregnant and what is happening with the pregnancy.

Pregnant_result
Strong entity. Contains which results a pregnancy can have.
Pregnant_result is a list with different results of a pregnancy.

Project
Strong entity. Contains information about projects that are affecting districts.

Relation_to_head
Weak entity, relation between Inhabitant and Relation_type. Contains data about when an inhabitant had which relation to the family head.

Relation_type
Strong entity. Contains information about which relation an inhabitant can have to the family head.
Relation_type is a list with the different relations to the family head.

Sleeping_rooms
Weak entity, unary relation with House. Contains information about how many sleeping rooms a house had a special date.

Territory
Strong entity. Contains data about territories.
A Territory is a land area that has many districts inside it.

Type_hcc
Strong entity. Contains which type a health care centre can bee.
Type_hcc is a list with the different types.

Type_of_mig
Strong entity. Contains which types of migration that can occur.
Type_of_mig is a list with different types of migration.

Uninhabited_house
Sub entity to House. Contains information about uninhabited houses.
An Uninhabited_house is a house where no inhabitants live.

Women
Sub entity to Habitant. Contains data that are specific to women.
A woman is a female inhabitant in the study area.

Works_with
Weak entity, relation between Occupation and Inhabitant. Contains information when a person had a certain kind of occupation.

Zone_type
Strong entity. Contains which zone types a district can be.
Zone_type is a list with different zone types.

3.4 Attribute specification

This chapter specifies all the attributes in the entities. What the information in the attribute stands fore. A * after the name of the entity means that the entity did not exists in the original design (*Demographic surveillance system database* by Mattias Pedersen, Axel Mårtensson, 2003).

Name	**Attribute description**
Birth_event	
Cry_on_birth	If the inhabitant cried when he/she was born.
*Clients **	
Client_name	The name of the client.
Death_event	
Cause_of_death	The cause of the death.
Date_of_death	The date when the inhabitant died.
District	
District_number	The combined figures for the district and the territory. E.g. SE8 for Avangasca Sur in Sutiava.
District_name	The name of the district. E.g. Avangasca Sur.
Year_created	Which year the district was founded.
District_type	
District_type	Identification number for the district type.
District_description	Description of the district type. E.g. Quarter or Suburb.
Educated	
Date_when	When an inhabitant had a certain kind of education.
Education	
Education_code	Identification number for the education.
Education_description	The description of the education-level.

Employed

Employed_code	Identification number of the employed.
Employed_name	The name of the employed.

Hcc_is_type

Date_of_changed	When the health care centre changed type.

Health_care_centre

Hcc_id	Identification number for the health care centre.
Hcc_name	Name of the health care centre.
Number_beds	Numbers of beds in the health care centre.
Days_open	Numbers of days the health care centre is open in a week.
Hours_open	How many hours the health care centre is open per day.

House

Household_code	The identification number for the hose. E.g. SE8A101.
Address	The address to the house.

House_condition

Condition_type	Identification number for the house condition.
House_condition	A description of the condition type. E.g. dirt floor or cement floor.

House_standard

Date_changed	When the house changed condition.

House_type

Type_code	Identification number for the house type.
Type_description	A description of the type. E.g. what type of school.

Inhabitant	
Id_habit	Identification number for the inhabitant.
First_name	The first name of the inhabitant.
Surname	The surname of the inhabitant.
Date_of_birth	When the inhabitant is born.
Interview	
Interview_nr	The number of the interview.
Date_arr_CIDS	When the interview arrived to CIDS.
Code_lb_cycle	Which cycle the interview is made in.
Time_start	When the interview starts at the household.
Time_end	When the interview ends at the household.
Locked_by *	(does not contain attributes)
Men	(does not contain attributes)
Migration_in	
Date_of_inmig	When the migration happened.
Cause_of_inmig	Why the inhabitant migrated.
Migration_out	
Date_of_outmig	When the migration happened.
Cause_of_outmig	Why the inhabitant migrated.
Occupation	
Occupation_code	Identification number for the occupation.
Occupation_description	Description of the occupation.
Pregnancy_information	
Number_pregnancy	How many pregnancy has the woman had with this pregnancy.
Children_in_preg	Number of children in the pregnancy. E.g. 1 for single child, 2 for twins etc.
Birth_attendant	Attendant at the birth.
Place_of_delivery	Where the child was born.

Pregnant_result

Result_code	Identification number for the result.
Result_description	Description of result of a pregnancy.

Project

Project_description	Description of the project.

Relation_to_head

Date_when	When the inhabitant had the relation to the head.

Relation_type

Relation_code	Identification number for the relation to the family head.
Relation_description	Description of the relation to the family head.

Sleeping_rooms

Number_rooms	Number of sleeping rooms in the house.
Date_changed	The date when the number of rooms changed

Territory

Territory_code	The code of the territory. E.g. M, P and S
Territory_name	The name of the territory. E.g. Mantica, Perla Maria and Sutiava.
Code_map	The code to the map. E.g. L for Leon.

Type_hcc

Type_hcc_code	Identification number for type of health care centre.
Type_hcc_description	Description of the type of health care centre.

Type_of_mig

Code_type_of	Identification number for type of migration.
Type_description	Description of the type of migration.

Uninhabited_house

Name	Name of the uninhabited house.

Women

No_of_pregnancy	How many times the woman has been pregnant.
Suspected_pregnant	Yes/No if she suspects she is pregnant or not.

Works_with

Date_when	When an inhabitant had a certain kind of occupation.

Zone_Type

Zone_type	Identification number for the zone type.
Zone_description	The name Rural, Urban

3.5 Detailed description of relations

This chapter describes which attributes are keys and foreign keys in relations. Keys are market with 🗝 and foreign keys are market with 🔗. The key that stands first are chosen as primary key.

Birth_event(Cry_on_birth, Id_habit, Id_woman, Number_pregnant)
🗝 {Id_habit}
🔗 Id_habit in Inhabitant
🔗 Id_woman in Pregnancy_information
🔗 Number_pregnancy in Pregnancy_information

Clients(Id_client, Client_name)
🗝 {Id_client}

Death_event(Cause_of_death, Date_of_death, Id_habit)
🗝 {Id_habit}
🔗 Id_habit in Inhabitant

District(District_number, District_name, Year_created, Territory_code, Zone_type)
🗝 {District_number}
🔗 Territory_code in Territory
🔗 Zone_type in Zone_type

District_type(District_type, District_description)
🗝 {District_type}

Dtype(District_type, District_number)
🗝 {District_number}
🔗 District_type in District_type
🔗 District_number in District

Educated(Date_when, Education_code, Id_habit)
🗝 {Education_code, Id_habit}
🔗 Education_code in Education
🔗 Id_habit in Inhabitant

Education(Education_code, Education_description)
🗝 {Education_code}

Employed(Employed_code, Employed_name)
🗝 {Employed_code}

Hcc(Hcc_id, District_number)
🗝 {Hcc_id, District_number}
🔗 Hcc_id in Health_care_centre
🔗 District_number in District

Hcc_is_type(Date_of_changed, Hcc_id, Type_hcc_code)
🗝 {Hcc_id, Type_hcc_code}
🔗 Hcc_id in Health_care_centre
🔗 Type_hcc_code in Type_hcc

Health_care_centre(Tab_hcc_id, Hcc_id, Hcc_name, Number_beds, Days_open, Hours_open)
🗝 {Tab_cc_id}

House(Tab_house_id, Household_code, Address, District_number)
🗝 {Tab_house_id}
🔗 District_number in District

House_condition(Condition_type, House_condition)
🗝 {Condition_type}

House_standard(Date_changed, Condition_type, Household_code)
🗝 {Condition_type, Household_code}
🔗 Condition_type in House_condition
🔗 Household_code in House

House_type(Type_code, Type_description)
🗝 {Type_code}

Inhabitant(Tab_id_habit, Id_habit, First_name, Surname, Date_of_birth, Father, Mother)
🗝 {Tab_id_habit}
🔗 Father in Inhabitant
🔗 Mother in Women

Interview(Interview_nr, Date_arr_CIDS, Code_lb_cycle, Time_start, Time_end, Household_code, Employed_code)
🗝 {Interview_nr}
🔗 Household_code in House
🔗 Employed_code in Employed

Investigate(District_number, Project_nr)
🗝 {District_number, Project_nr}
🔗 District_number in District
🔗 Project_nr in Project

Locked_by(Tab_house_id, Id_client)
🗝 {Tab_house_id, Id_client}
🔗 Tab_house_id in House
🔗 Id_client in Clients

Men(Id_habit)
🗝 {Id_habit}
🔗 Id_habit in Inhabitant

Migration_in(Date_of_inmig, Cause_of_inmig, Type_of_migration, Id_habit, Household_code)
🗝 {Date_of_inmig, Id_habit}
🔗 Household_code in House
🔗 Id_habit in Inhabitant
🔗 Type_of_migration in Type_of_mig

Migration_out(Date_of_outmig, Cause_of_outmig, Type_of_migration, Id_habit, Household_code)
🗝 {Date_of_outmig, Id_habit}
🔗 Household_code in House
🔗 Id_habit in Inhabitant
🔗 Type_of_migration in Type_of_mig

Occupation(Occupation_code, Occupation_description)
🗝 {Occupation_code}

Pregnancy_information(Number_pregnancy, Children_in_preg, Birth_attendant, Place_of_delivery, Id_woman, Result_code)
🗝 {Id_woman, Number_pregnancy}
🔗 Id_woman in Women
🔗 Result_code in Pregnant_result

Pregnant_result(Result_code, Result_description)
🗝 {Result_code}

Project(Project_nr, Project_description)
🗝 {Project_nr}

Relation_to_head(Date_when, Id_habit, Relation_code)
🗝 {Id_habit, Date_when}
🔗 Id_habit in Inhabitant
🔗 Relation_code in Relation_type

Relation_type(Relation_code, Relation_description)
🗝 {Relation_code}

Sleeping_rooms(Number_rooms, Date_changed, Household_code)
🗝 {Household_code, Date_changed}
🔗 Household_code in House

Territory(Territory_code, Territory_name, Code_map)
🗝 {Territory_code}

Type_hcc(Type_hcc_code, Type_hcc_description)
🗝 {Type_hcc_code}

Type_of_mig(Code_type_of, Type_description)
🗝 {Code_type_of}

Uninhabited_house(Name, Household_code, Type_code)
🗝 {Household_code}
🔗 Type_code in House_type
🔗 Household_code in House

Women(Id_habit, No_of_pregnancy, Suspected_pregnant)
🗝 {Id_habit}
🔗 Id_habit in Inhabitant

Works_with(Date_when, Occupation_code, Id_habit)
🗝 {Occupation_code, Id_habit}
🔗 Occupation_code in Occupation
🔗 Id_habit in Inhabitant

Zone_type(Zone_type, Zone_description)
🗝 {Zone_type}

3.6 Normalisation

This chapter shows that all tables are in the normal form BCNF, Boyce-Codd Normal Form.

Birth_event(Cry_on_birth, Id_habit, Id_woman, Number_pregnant)
🗝 {Id_habit}
Id_habit ➔ Cry_on_birth, Id_woman, Number_pregnant
All left lines are superkeys ⇨ in BCNF

Clients(Id_client, Client_name)
🗝 {Id_client}
Id_client ➔ Client_name
All left lines are superkeys ⇨ in BCNF

Death_event(Cause_of_death, Date_of_death, Id_habit)
🗝 {Id_habit}
Id_habit ➔ Cause_of_death, Date_of_death
All left lines are superkeys ⇨ in BCNF

District(District_number, District_name, Year_created, Territory_code, Zone_type)
🗝 {District_number}
District_number ➔ District_name, Year_created, Territory_code, Zone_type
All left lines are superkeys ⇨ in BCNF

District_type(District_type, District_description)
🗝 {District_type}
District_type ➔ District_description
All left lines are superkeys ⇨ in BCNF

Dtype(District_type, District_number)
🗝 {District_number}
District_number ➔ District_type
All left lines are superkeys ⇨ in BCNF

Educated(Date_when, Education_code, Id_habit)
🗝 {Education_code, Id_habit}
Education_code, Id_habit ➔ Date_when
All left lines are superkeys ⇨ in BCNF

Education(Education_code, Education_description)
🗝 {Education_code}
Education_code ➔ Education_description
All left lines are superkeys ⇨ in BCNF

Employed(Employed_code, Employed_name)
🗝 {Employed_code}
Employed_code ➔ Employed_name
All left lines are superkeys ⇨ in BCNF

Hcc(Hcc_id, District_number)
🗝 {Hcc_id, District_number}
All left lines are superkeys ⇨ in BCNF

Hcc_is_type(Date_of_changed, Hcc_id, Type_hcc_code)
🗝 {Hcc_id, Type_hcc_code}
Hcc_id, Type_hcc_code ➔ Date
All left lines are superkeys ⇨ in BCNF

Health_care_centre(Tab_hcc_id, Hcc_id, Hcc_name, Number_beds, Days_open, Hours_open)
🗝 {Tab_hcc_id}
Tab_hcc_id ➔ Hcc_id, Hcc_name, Number_beds, Days_open, Hours_open
All left lines are superkeys ⇨ in BCNF

House(Tab_house_id, Household_code, Address, District_number)
🗝 {Tab_house_id}
Tab_house_id ➔ Address, District_number, Household_code
All left lines are superkeys ⇨ in BCNF

House_condition(Condition_type, House_condition)
🗝 {Condition_type}
Condition_type ➔ House_condition
All left lines are superkeys ⇨ in BCNF

House_standard(Date_changed, Condition_type, Household_code)
🗝 {Condition_type, Household_code}
Condition_type, Household_code ➔ Date_changed
All left lines are superkeys ⇨ in BCNF

House_type(Type_code, Type_description)
🗝 {Type_code}
Type_code ➔ Type_description
All left lines are superkeys ⇨ in BCNF

Inhabitant(Tab_id_habit, Id_habit, First_name, Surname, Date_of_birth, Father, Mother)
🗝 {Tab_id_habit}
Tab_id_habit ➔ Id_habit, First_name, Surname, Date_of_birth, Father, Mother
All left lines are superkeys ⇨ in BCNF

Interview(Interview_nr, Date_arr_CIDS, Code_lb_cycle, Time_start, Time_end, Household_code, Employed_code)
🗝 {Interview_nr}
Interview_nr ➔ Date_arr_CIDS, Code_lb_cycle, Time_start, Time_end, Household_code, Employed_code
All left lines are superkeys ⇨ in BCNF

Investigate(District_number, Project_nr)
🗝 {District_number, Project_nr}
All left lines are superkeys ⇨ in BCNF

Locked_by(Tab_house_id, Id_client)
🗝 {Tab_house_id, Id_client}
All left lines are superkeys ⇨ in BCNF

Men(Id_habit)
🗝 {Id_habit}
All left lines are superkeys ⇨ in BCNF

Migration_in(Date_of_inmig, Cause_of_inmig, Type_of_migration, Id_habit, Household_code)
🗝 {Date_of_inmig, Id_habit}
Date_of_inmig, Id_habit ➔ Cause_of_inmig, Type_of_migration, Household_code
All left lines are superkeys ⇨ in BCNF

Migration_out(Date_of_outmig, Cause_of_outmig, Type_of_migration, Id_habit, Household_code)
🗝 {Date_of_outmig, Id_habit}
Date_of_outmig, Id_habit ➔ Cause_of_outmig, Type_of_migration, Household_code
All left lines are superkeys ⇨ in BCNF

Occupation(Occupation_code, Occupation_description)
🗝 {Occupation_code}
Occupation_code ➔ Occupation_description
All left lines are superkeys ⇨ in BCNF

Pregnancy_information(Number_pregnancy, Children_in_preg,
Birth_attendant, Place_of_delivery, Id_woman, Result_code)
🗝 {Id_woman, Number_pregnancy}
Id_woman, Number_pregnancy ➔ Result_code, Children_in_preg,
Birth_attendant, Place_of_delivery
All left lines are superkeys ⇨ in BCNF

Pregnant_result(Result_code, Result_description)
🗝 {Result_code}
Result_code ➔ Result_description
All left lines are superkeys ⇨ in BCNF

Project(Project_nr, Project_description)
🗝 {Project_nr}
Project_nr ➔ Project_description
All left lines are superkeys ⇨ in BCNF

Relation_to_head(Date_when, Id_habit, Relation_code)
🗝 {Id_habit, Date_when}
Id_habit, Date_when ➔ Relation_code
All left lines are superkeys ⇨ in BCNF

Relation_type(Relation_code, Relation_description)
🗝 {Relation_code}
Relation_code ➔ Relation_description
All left lines are superkeys ⇨ in BCNF

Sleeping_rooms(Number_rooms, Date_changed, Household_code)
🗝 {Household_code, Date_changed}
Household_code, Date_changed ➔ Number_rooms
All left lines are superkeys ⇨ in BCNF

Territory(Territory_code, Territory_name, Code_map)
🗝 {Territory_code}
Territory_code ➔ Territory_name, Code_map
All left lines are superkeys ⇨ in BCNF

Type_hcc(Type_hcc_code, Type_hcc_description)
🗝 {Type_hcc_code}
Type_hcc_code ➔ Type_hcc_description
All left lines are superkeys ⇨ in BCNF

Type_of_mig(Code_type_of, Type_description)
🗝 {Code_type_of}
Code_type_of ➔ Type_description
All left lines are superkeys ⇨ in BCNF

Uninhabited_house(Name, Household_code, Type_code)
🗝 {Household_code}
Household_code ➔ Name, Type_code
All left lines are superkeys ⇨ in BCNF

Women(Id_habit, No_of_pregnancy, Suspected_pregnant)
🗝 {Id_habit}
Id_habit ➔ No_of_pregnancy, Suspected_pregnant
All left lines are superkeys ⇨ in BCNF

Works_with(Date_when, Occupation_code, Id_habit)
🗝 {Occupation_code, Id_habit}
Occupation_code, Id_habit ➔ Date_when
All left lines are superkeys ⇨ in BCNF

Zone_type(Zone_type, Zone_description)
🗝 {Zone_type}
Zone_type ➔ Zone_description
All left lines are superkeys ⇨ in BCNF

3.7 Detailed description of table implementation

This chapter describes the table implementation of the database.
Attributes market with 🗝 in key type are primary keys, and attribute market with 🔗 in key type are foreign keys.

Key type	Attribute name	Data type
Birth_event		
	Cry_on_birth	Boolean
🗝 🔗	Id_habit	Integer
🔗	Id_woman	Integer
🔗	Number_pregnant	Smallint
Clients		
🗝	Id_client	Integer
	Client_name	Varchar(50)
Death_event		
	Date_of_death	Date
	Cause_of_death	Varchar(50)
🗝 🔗	Id_habit	Integer
District		
	Year_created	Smallint
	District_name	Varchar(50)
🗝	District_number	Char(3)
🔗	Territory_code	Char(1)
🔗	Zone_type	Integer
District_type		
	District_description	Varchar(25)
🗝	District_type	Integer
Dtype		
🗝 🔗	District_number	Char(3)
🔗	District_type	Integer
Educated		
	Date_when	Date
🗝 🔗	Education_code	Integer
🗝 🔗	Id_habit	Integer

Education

	Education_description	Varchar(50)
🗝	Education_code	Integer

Employed

	Employed_name	Varchar(50)
🗝	Employed_code	Integer

Hcc

🗝 🔓	Hcc_id	Integer
🗝 🔓	District_number	Char(3)

Hcc_is_type

	Date_of_changed	Date
🗝 🔓	Hcc_id	Integer
🗝 🔓	Type_hcc_code	Integer

Health_care_centre

	Hcc_name	Varchar(50)
	Number_beds	Smallint
	Days_open	Smallint
	Hours_open	Smallint
	Hcc_id	Char(4)
🗝	Tab_hcc_id	Integer

House

	Household_code	Char(7)
	Address	Varchar(200)
🗝	Tab_house_id	Integer
🔓	District_number	Char(3)

House_condition

	House_condition	Varchar(50)
🗝	Condition_type	Char(4)

House_standard

	Date_changed	Date
🗝 🔓	Condition_type	Char(4)
🗝 🔓	Household_code	Integer

House_type

	Type_description	Varchar(50)
🗝	Type_code	Integer

Inhabitant

	Id_habit	Char(9)
	First_name	Varchar(50)
	Surname	Varchar(50)
	Date_of_birth	Date
🔑	Tab_id_habit	Integer
🔓	Father	Integer
🔓	Mother	Integer

Interview

	Date_arr_CIDS	Date
	Code_lb_cycle	Varchar(5)
	Time_start	Timestamp
	Time_end	Timestamp
🔑	Interview_nr	Integer
🔓	Household_code	Integer
🔓	Employed_code	Integer

Investigate

🔑 🔓	District_number	Char(3)
🔑 🔓	Project_nr	Integer

Locked_by

🔑 🔓	Tab_house_id	Integer
🔑 🔓	Id_client	Integer

Men

🔑 🔓	Id_habit	Integer

Migration_in

	Cause_of_inmig	Varchar(50)
🔑	Date_of_inmig	Date
🔑 🔓	Id_habit	Integer
🔓	Household_code	Integer
🔓	Type_of_migration	Integer

Migration_out

	Cause_of_outmig	Varchar(50)
🔑	Date_of_outmig	Date
🔑 🔗	Id_habit	Integer
🔗	Household_code	Integer
🔗	Type_of_migration	Integer

Occupation

	Occupation_description	Varchar(50)
🔑	Occupation_code	Integer

Pregnancy_information

	Children_in_preg	Smallint
	Birth_attendant	Varchar(50)
	Place_of_delivery	Varchar(50)
🔑 🔗	Id_woman	Integer
🔑	Number_pregnancy	Smallint
🔗	Result_code	Integer

Pregnant_result

	Result_description	Varchar(30)
🔑	Result_code	Integer

Project

	Project_description	Varchar(50)
🔑	Project_nr	Integer

Relation_to_head

🔑	Date_when	Date
🔑 🔗	Id_habit	Integer
🔗	Relation_code	Integer

Relation_type

	Relation_description	Varchar(50)
🔑	Relation_code	Integer

Sleeping_rooms

	Number_rooms	Smallint
🗝 🔒	Household_code	Integer
🗝	Date_changed	Date

Territory

	Territory_name	Varchar(50)
	Code_map	Varchar(30)
🗝	Territory_code	Char(1)

Type_hcc

	Type_hcc_description	Varchar(30)
🗝	Type_hcc_code	Integer

Type_of_mig

	Type_description	Varchar(50)
🗝	Code_type_of	Integer

Uninhabited_house

	Name	Varchar(50)
🗝 🔒	Household_code	Integer
🔒	Type_code	Integer

Women

	Suspected_pregnant	Boolean
	No_of_pregnancy	Smallint
🗝 🔒	Id_habit	Integer

Works_with

	Date_when	Date
🗝 🔒	Occupation_code	Integer
🗝 🔒	Id_habit	Integer

Zone_type

	Zone_description	Varchar(25)
🗝	Zone_type	Integer

4 Low-level implementation of the PDA

Even though new PDAs are very fast and has big memories you still can't compare one to a stationary computer. Therefore we decided to choose the fast and memory efficient program language C that also is standard in PDA programming.

4.1 Memory organization

The storage memory in the Palm PDA is organized in databases. Every database is automatically stored in the PDA memory without any demands of knowing anything about the complex memory handling. Every program can create several databases. A database in the PDA is not like a database in a normal computer. It's more like an ordinary file. The database is furthermore divided into records. This makes it possible for the PDA memory organizer to divide the database into non-contiguous memory chunks. To make it easier for the programmer every record can be assigned to a unique serial number, type-code (or easier explained: category). We have been using this possibility to assign each record with a type-code shown in section 4.2.

4.1.1 Opening and closing the database

The database can be opened in a couple of different ways. We chose a method were you open the database by including the creator id of the application. This wouldn't have worked if there were several databases that the application needed to read and write from, but we have only implemented one database. The first time this application is run on a PDA no database will exist and therefore a small routine checks for this state and creates a new database if necessary.

When you open a database the program will return an active link to the database. This link is later used whenever the program needs to access data from the records. To close the database just send close database event to the system with the active link.

4.1.2 Finding records

Always when you work with records you need an index of the record. When you create a new record you always get this index number, but normally you need to get this index for already existing records that need to be changed, moved or deleted.

When we want to find a special record we iterate thru all the objects in a special category, and compare every records data with the desired information. This will give you the index number as well as a handle to the object. A handle is an object with witch it's possible to communicate with the record.

4.1.3 Deleting records

When you want to delete a record of the database the PalmOS leaves you with three choices. The biggest difference between the methods is if the storage manager is supposed to delete the record immediately after it's told to do so, or if it only should mark the record for deletion. The reason for this is because the stationary computer needs to know when a record has been deleted. If your using one of the two methods that only mark the record for deletion the storage manager will delete the record first after a successful synchronization with a stationary computer.

We never need to tell the stationary computer when a record needs to be deleted, because we synchronize everything on every synchronization, and due to this we use the more simple command DmRemoveRecord that only needs the database object and index as reference.

4.2 Description of the record-types in the PDA

Name	Type-code	Explanation
catOccupation	1	List of all occupation-types available
catEducation	2	List of all education-types available
catRelationType	3	List of all relation-types available
catHouse	4	List of all houses in the study area
catInhabitant	5	List of all inhabitants in the study area
catPregnancy	6	List of all pregnancy-types available
catPregnancyResult	7	List of all pregnancies for inhabitants
catHouseCondition	8	List of all house condition-types available

4.3 Description of the records in the PDA

Name: Occupation
Type-code: 1

Attribute name	Bytes	Type
Occupation_description	51	char[]
Occupation_code	2	Int16

Name: Education
Type-code: 2

Attribute name	Bytes	Type
Education_description	51	char[]
Education_code	2	Int16

Name: Relation_type
Type-code: 3

Attribute name	Bytes	Type
Relation_description	51	char[]
Relation_code	2	Int16

Name: House
Type-code: 4

Attribute name	Bytes	Type
Tab_house_id	4	Int32
Household_code	8	char[]
Address	201	char[]
Water	5	char[]
Wall	5	char[]
Floor	5	char[]
Toilet	5	char[]
Sleeping_rooms	2	Int16

Name: Inhabitant
Type-code: 5

Attribute name	Bytes	Type
Tab_id_habit	4	Int32
Inhb_tab_house_id	4	Int32
Id_habit	10	Char[]
First_name	51	Char[]
Surname	51	Char[]
Date_of_birth	14	DateTimeType
Tab_id_Mother	4	Int32
Occupation_code	2	Int16
Education_code	2	Int16
Relation_to_head_code	2	Int16
IsAMan	1	Boolean
Birth_Cry_on_birth	1	Boolean
Birth_ID_Pregnancy	2	Int16
Death_Date_of_death	14	DateTimeType
Death_description	256	Char[]
Death_Certificate_description	201	Char[]
MigIn_Date_of_inmigration	14	DateTimeType
MigIn_Cause_of_inmig	51	Char[]
MigIn_Inmig_description	201	Char[]
MigOut_Date_of_outmigration	14	DateTimeType
MigOut_Cause_of_outmig	51	Char[]
MigOut_Outmig_description	201	Char[]
IsNewPerson	1	Boolean

Name: Pregnancy_information
Type-code: 6

Attribute name	Bytes	Type
Tab_id_habit	4	Int32
Children_in_preg	2	Int16
Birth_attendant	51	Char[]
Place_of_delivery	51	Char[]
Number_pregnancy	2	Int16
Result_code	2	Int16

Name: Pregnant_result
Type-code: 7

Attribute name	Bytes	Type
Result_description	31	Char[]
Result_code	2	Int16

Name: House_condition
Type-code: 8

Attribute name	Bytes	Type
House_condition	51	Char[]
Condition_type	5	Char[]

5 PDA/PC data transfer

Our conduit software (”DSS_Conduit”) has the purpose that at every HotSync operation synchronize the data on the palm with the mid-storage database on the client (”Client_DB”). DSS_Conduit is a standard executable win32 software with a very small user interface (Fig. 5.1) where the user (fieldworker) can log on. After that a user has logged on to the system DSS_conduit will syncronize the palm and reload it with specific data for the current user (e.g witch houses the fieldworker is going to interview). DSS_Conduit is written in Visual Basic.NET witch is an interpretating language and therefor requires an interpretation software (.Net Framework).

Fig. 5.1: The DSS_Conduit user interface.

5.1 Description of the conduit function

5.1.1 Reading

Because of the fact that the Palm PDA does not have what most people would call a database (chapter 4.1), but more of a large table with records of different size, is reading done in a bit different way. All the reading from the palm's memory is enclosed in the sub-routine readFromPDA() where the following happens:
All records, without considering witch type, are looped and a bit-array with the record-data is read. For each record the type is checked with a case-statement. Depending on the type of record, different code is executed that translates the "ones and zeros" in different ways to the appropriate .NET datatypes that the conduit-program understands (list of the record-types in capter 4.3).
If the record is a register-record (a record that is only used in the PDA for displaying but that is not changed, e.g Occupations, Educations) are they just ignored. For every different type are the attributes read at fixed positions in the bit-array. This means that if anything is changed in the structure of a record in the PDA the same changes must be made in the conduite. To read the attributes in the bit-array a function in the Palm CDK (Palm Conduit Development Kit) is used. All attributes that are read from the array are then to be inserted in the Client_DB. This is done with an OleDb database-connection to the database. The data are then inserted into the appropriate tables in the database using the built-in ADO connection I .Net and SQL expressions.

5.1.2 Erasing

After the reading has been completed is the subroutine erasePalmDB() called on to erase al records in the PDA. This subroutine uses functions in the Palm CDK to do this (see Appendix B:erasePalmDB).

5.1.3 Writing

In this step two types of data are written to the PDA: Register-records and interview-specific records (see definitions in chapter 7). For each type of data that are to be written to the PDA the data is read from the appropriate table in the Client_DB. For the register-records alla information in the tables are transferred. For the interview-specific datatypes are the rows that are marked with the current fieldworker's id and with the toPDA flag set as true tramnsfered. After the transfer is complete they are removed from the client database.

6 Client Database

This database that is located on the client computer is meant s a mid-storage database. The system could have been build without the database but that would require a constant stable connection to the server and this is a bit of a problem in Nicaragua today.
This database is NOT a relation database but more of a mirror of whats is stored in the PDA. The databse is not designed to be the most effective but we don't think that it is necessary because of the small amount of data that will be stored on it.

6.1 Description of the tables in the database

Name: Occupation

Attribute name	Type	Description
ocuDescr	String	The name of an occupation in clear text (e.g carpenter).
ocuCode	Integer	The occupation code (a integer number)

Name: Education

Attribute name	Type	Description
eduDescr	String	The name of an education in clear text (e.g 3 years university).
eduCode	Integer	The education code (a integer number)

Name: Relations

Attribute name	Type	Description
relDescr	String	The name of an relation in clear text (e.g brother).
relCode	Integer	The relation code (a integer number)

Name: House

Attribute name	Type	Description
Tab_house_id	Integer	A unique identifier for a house (only used in the databse)
Household_code	String	The code for the house (CIDS-standard)
Address	String	The direction to the house.
Water	String	The code for the type of water supply the house is having.
Wall	String	The code for the type of walls In the house.
Floor	String	The code for the type of floor In the house.
Toilet	String	The code for the sanitary equipment that is uses in the house.
Sleeping_rooms	Integer	The number of sleeping rooms in the house.
toPDA	Boolean	Decides if the reow is to be transferd to the PDA.
PDAid	Integer	Id-number for the fieldworker to whom this house has been assigned.

Name: Inhabitant

Attribute name	Type	Description
Id	Integer	A record counter/identifier
Tab_id_habit	Integer	The id of the inhabitant
Inhb_tab_house_id	Integer	The id of the house in witch the person lives.
Id_habit	String	The code for the inhabitant (CIDS-standard)
First_name	String	The first name.
Surname	String	The surname name
Date_of_birth	Date	The date of birth.
Tab_id_Mother	Integer	The id of the mother to this inhabitant.
Occupation_code	Integer	The id of the inhabitants eccupation.
Education_code	Integer	The id of the inhabitants education.
Relation_to_head_code	Integer	The id of the inhabitants education.
IsAMan	Boolean	A flag that determines the sex of the inhabitant. (true=man, false=woman)
Birth_Cry_on_birth	Boolean	A flag that indicates if the inhabitant cried when born.
Birth_ID_Pregnancy	Integer	The id of the pregnancy that resulted in this inhabitant.
Death_Date_of_death	Date	Date when the inhabitant died (if dead).
Death_description	String	Description of what caused the death.
Death_Certificate_ description	String	Information from the doctors death certificate.
MigIn_Date_of_ inmigration	Date	Date when inhabitant migrated to the current house.
MigIn_Cause_of_inmig	String	Description of the cause of the migration.
MigIn_Inmig_description	String	Other information such as from where the inhabitant migrated.

MigOut_Date_of_ outmigration	Date	Date when the inhabitant migrated out from the previous house.
MigOut_Cause_of_outmig	String	Description of the cause of the migration.
MigOut_Outmig_ description	String	Other information such as from where the inhabitant migrated.
toPDA	Boolean	Indicates if the house is to be transferred to the PDA.
PDAid	Integer	The id of the user (fieldworker) that is assign this inhabitant.

Name: Pregnancy

Attribute name	**Type**	**Description**
Tab_id_habit	Integer	Id for the pregnancy.
Children_in_preg	Integer	Number of children this pregnancy resulted in.
Birth_attendant	String	The name of the birth attendant.
Place_of_delivery	String	The place of delivery.
Number_pregnancy	Integer	Witch number of pregncy this is for the mother.
Result_code	Integer	The id of the pregnancy result.
toPDA	Boolean	Indicates if the house is to be transferred to the PDA.
PDAid	Integer	The id of the user (fieldworker) that is assign this inhabitant.

Name: PregnancyResults

Attribute name	**Type**	**Description**
preDesc	String	The name of this pregnancy result (a brief description of what wa the result of the pregnancy, e.g stillbirth, healthy)
preCode	Integer	The id number for this pregnancy result.

Name: House_condition

Attribute name	Type	Description
hcDescr	String	The description of the house condition (eg Light, Paper walls).
hcType	String	The code for the house condition combined of ome letter for the type of condition (e.g L=light, P=walls) and a id number.

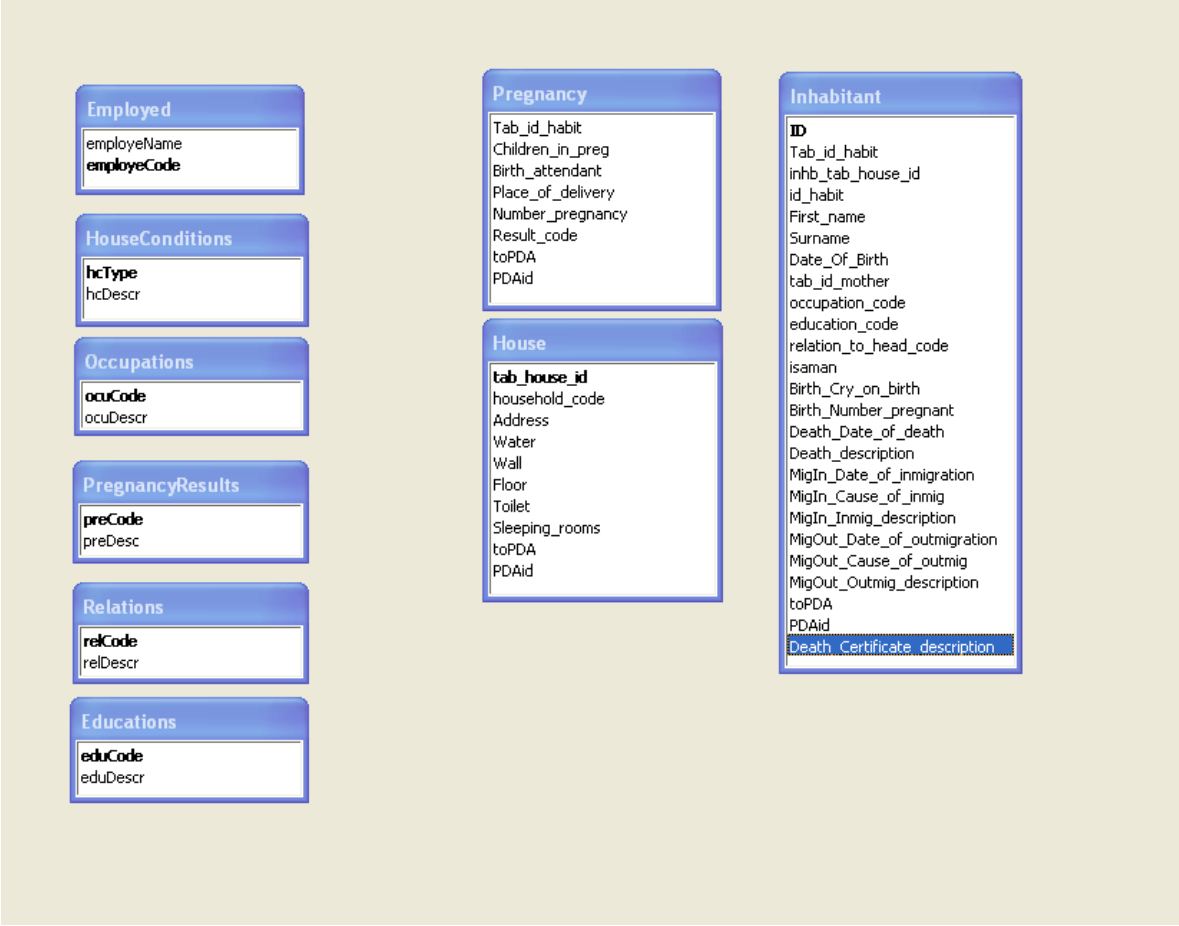


Fig 5.1: A screenshot of the access database tables.

7 References

Litterature

“Programming Visual Basic for the Palm OS”
Matthew Holmes, Patrick Burton, Roger Knoell
Publisher: O'Reilly Media, Inc.; 1 edition (April 16, 2002)
ISBN: 0596002009

“Palm OS Programming Bible”
Lonnon R. Foster
Publisher: Wiley Publishing Inc.; 2 edition (2002)
ISBN: 0764549618

“Developing windows based application with Microsoft Visual Basic.NET and Visual C#.NET”
Matthew A. Stoecker
Publisher: Microsoft Press (2003)
ISBN: 0735619253

Electronic Source

PalmSource’s “Palm OS® Programmer's API Reference”
http://www.palmos.com/dev/support/docs/palmos/PalmOSReference/

8 Appendix A – Low-level design on the PDA

DbMethods.h

```
#ifndef DBMETHODS_H_
#define DBMETHODS_H_

#include "dbStructures.h"

// ******         Open database           ****** //
extern Err OpenDatabase(void);

// ******         Close database          ****** //
extern void CloseDatabase(void);

// ******      Show memory error          ****** //
extern Err showMemError(void);

// ******       Delete pregnancy X        ****** //
extern Err deletePregnancy(int tab_id_mother, int
pregnancy_nbr);

// ** Load all pregnancies for inhabitant(x) *** //
extern Err loadPregnaciesForInhb(int tabIdHabit);

// ******      Delete inhabitant X        ****** //
extern Err deleteInhabitant(int tab_id);

// ******      Load all inhabitants       ****** //
extern Err loadInhabitant(void);

// ******      Load all occupations       ****** //
extern Err loadOccupations(void);

// ******       Load all educations       ****** //
extern Err loadEducations(void);

// ******       Load all relations        ****** //
extern Err loadRelationType(void);

// ******    Load all pregnancy types     ****** //
extern Err loadPregnancyResults(void);
```

```
// ******   Load all house conditions     ****** //
extern Err loadHouseConditionsFromDB(void);

// ****  Load all data in a specific house  **** //
extern Err loadDataInHouse(void);

// ******     Search and load house       ****** //
extern Err findHouse(char Household[8]);

// ****  House comparion callback function  **** //
extern Int16 HouseCompareFunc(occupation* h1,
occupation* h2, Int16 other, SortRecordInfoPtr
rec1SortInfo, SortRecordInfoPtr rec2SortInfo,
MemHandle appInfoH);

// ******    Update house information     ****** //
extern Err UpdateHouse(void);

// ** Inhabitant comparion callback function  ** //
extern Int16 InhabitantCompareFunc(inhabitant* h1,
inhabitant* h2, Int16 other, SortRecordInfoPtr
rec1SortInfo, SortRecordInfoPtr rec2SortInfo,
MemHandle appInfoH);

// **  Update / store inhabitant information  ** //
extern Err SaveInhabitant(void);

// **  Pregnancy comparion callback function  ** //
extern Int16
PregnancyCompareFunc(pregnancy_information* h1,
pregnancy_information* h2, Int16 other,
SortRecordInfoPtr rec1SortInfo, SortRecordInfoPtr
rec2SortInfo, MemHandle appInfoH);

// **  Update / store pregnancy information   ** //
extern Err SavePregnancyForInhb(void);


#endif
```

DbStructures.h

```
#ifndef DBSTRUCTURES_H_
#define DBSTRUCTURES_H_

/* --------------------- */
/*   Object Structures   */
/* --------------------- */
typedef struct
{
        char Education_description[51];
        int Education_code;
}education;

typedef struct
{
        char Occupation_description[51];
        int Occupation_code;
}occupation;

typedef struct
{
        char Relation_description[51];
        int Relation_code;
}relation_type;

typedef struct
{
        char House_condition[51];
        char Condition_type[5];
}house_condition;

typedef struct
{
        char Result_description[31];
        int Result_code;
}pregnant_result;

typedef struct
{
        Int32 Tab_house_id;
        char Household_code[8];
        char Address[201];
        char Water[5];
        char Wall[5];
```

```
        char Floor[5];
        char Toilet[5];
        Int16 Sleeping_rooms;
}house;

typedef struct
{
        Int32 Tab_id_habit;
        int Children_in_preg;
        char Birth_attendant[51];
        char Place_of_delivery[51];
        int Number_pregnancy;
        int Result_code;
}pregnancy_information;

typedef struct
{
        Int32 Tab_id_habit;
        Int32 Inhb_tab_house_id;
        char Id_habit[10];
        char First_name[51];
        char Surname[51];
        DateTimeType Date_of_birth;
        Int32 Tab_id_Mother;
        int Occupation_code;
        int Education_code;
        int Relation_to_head_code;
        Boolean IsAMan;
        Boolean Birth_Cry_on_birth;
        int Birth_ID_Pregnancy;
        DateTimeType Death_Date_of_death;
        char Death_description[256];
        char Death_Certificate_description[201];
        DateTimeType MigIn_Date_of_inmigration;
        char MigIn_Cause_of_inmig[51];
        char MigIn_Inmig_description[201];
        DateTimeType MigOut_Date_of_outmigration;
        char MigOut_Cause_of_outmig[51];
        char MigOut_Outmig_description[201];
        Boolean IsNewPerson;
}inhabitant;
```

```
typedef struct
{
        int inhabitant;
        int occupation;
        int education;
        int relationType;
        int houseCondition;
        int pregnantResults;
        int nbrOfPregnancyForX;
} nbrOfElements;


/* --------------------- */
/* External declarations */
/* --------------------- */

extern DmOpenRef gLibDB;
extern occupation  *occupationP;
extern education *educationP;
extern relation_type *relation_typeP;
extern house_condition *house_conditionP;
extern house *houseP;
extern inhabitant *inhabitantP;
extern pregnant_result *pregnant_resultP;
extern pregnancy_information
        *pregnancy_informationP;
extern nbrOfElements NbrOfElements;


#endif
```

DbStructures.c

```
#include "dbStructures.h"

DmOpenRef gLibDB;

occupation  *occupationP;
education *educationP;
relation_type *relation_typeP;
house_condition *house_conditionP;
house *houseP;
pregnant_result *pregnant_resultP;
inhabitant *inhabitantP;
pregnancy_information *pregnancy_informationP;
nbrOfElements NbrOfElements;
```

DbMethods.c

```
/* ------------------ */
/*  Database Methods  */
/* ------------------ */
#include "dbStructures.h"
#include "dbMethods.h"

#define DBName                  "DSS_DB-DSSD"
#define DBType                  'DATA'
#define AppCreatorID            'DSSD'

#define catOccupation           1
#define catEducation            2
#define catRelationType         3
#define catHouse                4
#define catPregnancyResult      7
#define catInhabitant           5
#define catPregnancy            6
#define catHouseCondition       8



// ********************************************** //
// ******          Open database           ****** //
// ********************************************** //
Err OpenDatabase(void)
{
    Err error = 0;

    if(error)
        return error;

    gLibDB = DmOpenDatabaseByTypeCreator(DBType,
            AppCreatorID, dmModeReadWrite);

    if(!gLibDB)
    {
        error = DmCreateDatabase(0,DBName,
                AppCreatorID, DBType, false);
        if(error)
            return error;

        gLibDB =
                DmOpenDatabaseByTypeCreator
```

```
                (DBType, AppCreatorID,
                dmModeReadWrite);
        if(!gLibDB)
            return error;
    }
    return error;
}


// ********************************************** //
// ******          Close database          ****** //
// ********************************************** //
void CloseDatabase(void)
{
    DmCloseDatabase(gLibDB);
}


// ********************************************** //
// ******      Show memory error           ****** //
// ********************************************** //
Err showMemError(void)
{
    ErrAlertCustom(ErrOKAlert, " Not Enough
            Memory ","","");
    return -1;
}


// ********************************************** //
// ******       Delete pregnancy X         ****** //
// ********************************************** //
Err deletePregnancy(int tab_id_mother, int
        pregnancy_nbr)
{
    MemHandle newRecordH;
    UInt16 index=0;
    pregnancy_information * myPregnancy;
    int nbrOfPregnancy, counter=0;

    nbrOfPregnancy=DmNumRecordsInCategory(gLibDB
            , catPregnancyResult);
    if(!nbrOfPregnancy)
        return 1;
```

```
    while( (myPregnancy->Tab_id_habit !=
            pregnancy_informationP
            [counter].Tab_id_habit)
            || (myPregnancy->Number_pregnancy !=
            pregnancy_informationP
            [counter].Number_pregnancy) )
    {
        newRecordH=DmQueryNextInCategory
                (gLibDB, &index,
                catPregnancyResult);

        if (newRecordH) {
            myPregnancy =
                    ((pregnancy_information*)
                    MemHandleLock(newRecordH));
            MemHandleUnlock(newRecordH);
        } else
            break;
        index++;
        if( (myPregnancy->Tab_id_habit !=
                pregnancy_informationP
                [counter].Tab_id_habit)
                || (myPregnancy->Number_pregnancy !=
                pregnancy_informationP
                [counter].Number_pregnancy) )
            counter++;
    }

    if(!newRecordH || (counter==nbrOfPregnancy))
        return 1;

    DmRemoveRecord(gLibDB, index);
    return 0;
}




// ********************************************** //
// ** Load all pregnancies for inhabitant(x) *** //
```

```
// ********************************************* //
Err loadPregnaciesForInhb(int tabIdHabit)
{
    MemHandle newRecordH;
    UInt16 index=0;
    pregnancy_information * myPregnancy_information;
    int nbrOfPregnancies, counter=0;

    nbrOfPregnancies=DmNumRecordsInCategory(gLibDB,
            catPregnancy);
    if(!nbrOfPregnancies)
        return 1;

    //Oversized variable
    if(!(pregnancy_informationP =
            (pregnancy_information*) MemPtrNew
            (30*sizeof(pregnancy_information))))
        return showMemError();

    NbrOfElements.nbrOfPregnancyForX=0;
    while(counter!=nbrOfPregnancies)
    {
        newRecordH=DmQueryNextInCategory
                (gLibDB, &index, catPregnancy);
        if (newRecordH) {
            myPregnancy_information =
                    ((pregnancy_information*)
                    MemHandleLock(newRecordH));

            pregnancy_informationP[NbrOfElements.
                    nbrOfPregnancyForX]=
                    *myPregnancy_information;

            NbrOfElements.nbrOfPregnancyForX++;

            MemHandleUnlock(newRecordH);
        } else
            break;

        counter++;
        index++;
    }

    if(!newRecordH)
```

```
        return 1;
    return 0;
}



// ********************************************** //
// ******      Delete inhabitant X         ****** //
// ********************************************** //
Err deleteInhabitant(int tab_id)
{
    MemHandle newRecordH;
    UInt16 index=0;
    inhabitant * myInhabitant;
    int nbrOfInhabitants, counter=0;

    nbrOfInhabitants=DmNumRecordsInCategory(gLib
            DB, catInhabitant);
    if(!nbrOfInhabitants)
        return 1;

    while( (myInhabitant->Tab_id_habit !=
            inhabitantP[counter].Tab_id_habit) )
    {
        newRecordH=DmQueryNextInCategory(gLibDB,
                &index, catInhabitant);

        if (newRecordH) {
            myInhabitant = ((inhabitant*)
                    MemHandleLock(newRecordH));
            MemHandleUnlock(newRecordH);
        } else
            break;
        index++;
        if( (myInhabitant->Tab_id_habit !=
                inhabitantP[counter].Tab_id_habit) )
            counter++;
    }

    if(!newRecordH || (counter==nbrOfInhabitants))
        return 1;

    DmRemoveRecord(gLibDB, index);
    return 0;
```

```
}



// ********************************************** //
// ******      Load all inhabitants       ****** //
// ********************************************** //
Err loadInhabitant(void)
{
    MemHandle newRecordH;
    UInt16 index=0;
    inhabitant * myInhabitant;
    int nbrOfInhabitants;

    nbrOfInhabitants=DmNumRecordsInCategory(gLibDB,
            catInhabitant);

    if(!nbrOfInhabitants)
        return 1;

    //Oversized variable
    if(!(inhabitantP = (inhabitant*)
            MemPtrNew(50*sizeof(inhabitant))))
        return showMemError();

    NbrOfElements.inhabitant=0;
    while(nbrOfInhabitants)
    {
        newRecordH=DmQueryNextInCategory
                (gLibDB, &index, catInhabitant);

        if (newRecordH) {
            myInhabitant = ((inhabitant*)
                    MemHandleLock(newRecordH));

            if(myInhabitant>Inhb_tab_house_id
                    ==houseP[0].Tab_house_id)
            {
            inhabitantP[NbrOfElements.
                    inhabitant]=*myInhabitant;
                NbrOfElements.inhabitant++;
            }
            MemHandleUnlock(newRecordH);
        } else
```

```
            break;
        index++;
        nbrOfInhabitants--;
    }
    return 0;
}



// ********************************************** //
// ******      Load all occupations       ****** //
// ********************************************** //
Err loadOccupations(void)
{
    MemHandle newRecordH;
    UInt16 index=0;
    occupation * myOccupation;
    int nbrOfOccupations;

    nbrOfOccupations=DmNumRecordsInCategory(gLibDB,
            catOccupation);

    if(!nbrOfOccupations)
        return 1;

    if(!(occupationP = (occupation*)
            MemPtrNew(nbrOfOccupations
            *sizeof(occupation))))
        return showMemError();

    NbrOfElements.occupation=0;
    while(nbrOfOccupations)
    {
        newRecordH=DmQueryNextInCategory
                (gLibDB, &index, catOccupation);


        if (newRecordH) {
            myOccupation = ((occupation*)
                    MemHandleLock(newRecordH));
            occupationP[NbrOfElements.
                    occupation]=*myOccupation;
            NbrOfElements.occupation++;
            MemHandleUnlock(newRecordH);
```

```
        }
        index++;
        nbrOfOccupations--;
    }
    return 0;
}



// ********************************************** //
// ******       Load all educations       ****** //
// ********************************************** //
Err loadEducations(void)
{
    MemHandle newRecordH;
    UInt16 index=0;
    education * myEducation;
    int nbrOfEducations;

    nbrOfEducations=DmNumRecordsInCategory(gLibDB,
            catEducation);

    if(!nbrOfEducations)
        return 1;
    if(!(educationP = (education*)
            MemPtrNew(nbrOfEducations
            *sizeof(education))))
        return showMemError();

    NbrOfElements.education=0;
    while(nbrOfEducations)
    {
        newRecordH=DmQueryNextInCategory(gLibDB,
                &index, catEducation);
        if (newRecordH) {
            myEducation = ((education*)
                    MemHandleLock(newRecordH));
            educationP[NbrOfElements.education]=
                    *myEducation;
            NbrOfElements.education++;
            MemHandleUnlock(newRecordH);
        }
        index++;
        nbrOfEducations--;
```

```
    }
    return 0;
}



// ********************************************** //
// ******       Load all relations        ****** //
// ********************************************** //
Err loadRelationType(void)
{
    MemHandle newRecordH;
    UInt16 index=0;
    relation_type * myRelationType;
    int nbrOfRelationTypes;

    nbrOfRelationTypes=DmNumRecordsInCategory(
            gLibDB, catRelationType);

    if(!nbrOfRelationTypes)
        return 1;

    if(!(relation_typeP = (relation_type*)
            MemPtrNew(nbrOfRelationTypes
            *sizeof(relation_type))))
        return showMemError();

    NbrOfElements.relationType=0;
    while(nbrOfRelationTypes)
    {
        newRecordH=DmQueryNextInCategory(gLibDB,
                &index, catRelationType);
        if (newRecordH) {
            myRelationType = ((relation_type*)
                    MemHandleLock(newRecordH));
            relation_typeP[NbrOfElements.
                    relationType]=*myRelationType;
            NbrOfElements.relationType++;
            MemHandleUnlock(newRecordH);
        }


        index++;
        nbrOfRelationTypes--;
```

```
    }
    return 0;
}



// ********************************************** //
// ******    Load all pregnancy types     ****** //
// ********************************************** //
Err loadPregnancyResults(void)
{
    MemHandle newRecordH;
    UInt16 index=0;
    pregnant_result * myPregnantResult;
    int nbrOfPregnancyResults;

    nbrOfPregnancyResults=DmNumRecordsInCategory(
            gLibDB, catPregnancyResult);

    if(!nbrOfPregnancyResults)
        return 1;

    if(!(pregnant_resultP = (pregnant_result*)
            MemPtrNew(nbrOfPregnancyResults
            *sizeof(pregnant_result))))
        return showMemError();

    NbrOfElements.pregnantResults=0;
    while(nbrOfPregnancyResults)
    {
        newRecordH=DmQueryNextInCategory(gLibDB,
            &index, catPregnancyResult);
        if (newRecordH) {
            myPregnantResult = ((pregnant_result*)
                    MemHandleLock(newRecordH));
            pregnant_resultP[NbrOfElements
                    .pregnantResults]=
                    *myPregnantResult;
            NbrOfElements.pregnantResults++;
            MemHandleUnlock(newRecordH);
        }

        index++;
        nbrOfPregnancyResults--;
```

```
    }
    return 0;
}



// ********************************************** //
// ******   Load all house conditions     ****** //
// ********************************************** //
Err loadHouseConditionsFromDB(void)
{
    MemHandle newRecordH;
    UInt16 index=0;
    house_condition * myHouse_condition;
    int nbrOfHouseConditions;

    nbrOfHouseConditions=DmNumRecordsInCategory(
            gLibDB, catHouseCondition);

    if(!nbrOfHouseConditions)
        return 1;

    if(!(house_conditionP = (house_condition*)
            MemPtrNew(nbrOfHouseConditions*
            sizeof(house_condition))))
        return showMemError();

    NbrOfElements.houseCondition=0;
    while(nbrOfHouseConditions)
    {
        newRecordH=DmQueryNextInCategory(gLibDB,
                &index, catHouseCondition);
        if (newRecordH) {
            myHouse_condition = ((house_condition*)
                    MemHandleLock(newRecordH));
            house_conditionP[NbrOfElements.
                    houseCondition]=
                    *myHouse_condition;
            NbrOfElements.houseCondition++;
            MemHandleUnlock(newRecordH);
        }

        index++;
        nbrOfHouseConditions--;
```

```
    }
    return 0;
}



// ********************************************** //
// ****  Load all data in a specific house  **** //
// ********************************************** //
Err loadDataInHouse(void)
{
    Err error=0;

    error += loadInhabitant();
    error += loadOccupations();
    error += loadEducations();
    error += loadRelationType();
    error += loadPregnancyResults();
    error += loadHouseConditionsFromDB();

    return error;
}



// ********************************************** //
// ******     Search and load house       ****** //
// ********************************************** //
Err findHouse(char Household[8])
{
    MemHandle newRecordH;
    UInt16 index=0;
    house * myHouse;
    int nbrOfHouses, counter=0;

    nbrOfHouses=DmNumRecordsInCategory(gLibDB,
            catHouse);
    if(!nbrOfHouses || (Household==NULL))
        return 1;              // No houses in DB

    if(!(houseP = (house*)
            MemPtrNew(1*sizeof(house))))
        return showMemError();
    houseP[0].Household_code[0]=NULL;
```

```
    while( (StrCompare(Household,
            houseP[0].Household_code)!=0) &&
            (counter!=nbrOfHouses) )
    {
        newRecordH=DmQueryNextInCategory(gLibDB,
                &index, catHouse);
        if (newRecordH) {
            myHouse = ((house*)
                    MemHandleLock(newRecordH));
            houseP=myHouse;
            MemHandleUnlock(newRecordH);
        } else
            break;
        index++;

        if((StrCompare(Household,
                houseP[0].Household_code)!=0))
            counter++;
    }

    if(!newRecordH || (counter==nbrOfHouses))
        return 1;

    return loadDataInHouse();
}



// ********************************************** //
// ****  House comparion callback function  **** //
// ********************************************** //
Int16 HouseCompareFunc(occupation* h1, occupation*
h2, Int16 other, SortRecordInfoPtr rec1SortInfo,
SortRecordInfoPtr rec2SortInfo, MemHandle appInfoH)
{
    return StrCompare(h1->Occupation_description,
            h2->Occupation_description);
}




// ********************************************** //
// ******    Update house information     ****** //
```

```
// ********************************************** //
Err UpdateHouse(void)
{
    MemHandle changedRecordH, oldH;
    house *oldRecord;
    int error;
    UInt16 index=0;

    //Search for old ID
    oldH=DmQueryNextInCategory(gLibDB, &index,
            catHouse);
    if (oldH) {
        oldRecord = ((house*) MemHandleLock(oldH));
        MemHandleUnlock(oldH);
        while( (oldRecord->Tab_house_id !=
                houseP[0].Tab_house_id) ) {
            index++;
            oldH=DmQueryNextInCategory(gLibDB,
                    &index, catHouse);
            if (!oldH) {
                index=DmFindSortPosition(gLibDB,
                        &houseP[0], NULL, (DmComparF
                        *) HouseCompareFunc, NULL);
                break;
            }
            oldRecord = ((house*)
                    MemHandleLock(oldH));
            MemHandleUnlock(oldH);
        }
    } else
        index=DmFindSortPosition(gLibDB, &houseP[0],
                NULL, (DmComparF *) HouseCompareFunc,
                NULL);
    changedRecordH = DmNewHandle(gLibDB,
            sizeof(houseP[0]));
    if(!changedRecordH)
        error = DmGetLastErr();
    else
    {
        DmWrite(MemHandleLock(changedRecordH),
                0, &houseP[0],
                sizeof(houseP[0]));
```

```
        error = DmAttachRecord(gLibDB, &index,
                changedRecordH,
                (oldH==NULL)?NULL:&oldH);
        MemHandleUnlock(changedRecordH);
    }
    return error;
}



// ********************************************** //
// ** Inhabitant comparion callback function  ** //
// ********************************************** //
Int16 InhabitantCompareFunc(inhabitant* h1,
inhabitant* h2, Int16 other, SortRecordInfoPtr
rec1SortInfo, SortRecordInfoPtr rec2SortInfo,
MemHandle appInfoH)
{
    return StrCompare(h1->Surname, h2->Surname);
}



// ********************************************** //
// **  Update / store inhabitant information  ** //
// ********************************************** //
Err SaveInhabitant(void)
{
    MemHandle newRecordH, oldH;
    inhabitant *oldRecord;
    UInt16 index=0;
    Err error=0;
    UInt16 attributes;
    int counter=0;

    while( (counter!=NbrOfElements.inhabitant) &&
            !error ) {
        //Search for old ID
        index=0;
        oldH=DmQueryNextInCategory(gLibDB, &index,
                catInhabitant);
        if (oldH) {
```

```
            oldRecord = ((inhabitant*)
                    MemHandleLock(oldH));
            MemHandleUnlock(oldH);
            while( (oldRecord->Tab_id_habit !=
                    inhabitantP[counter].
                    Tab_id_habit) ) {
                index++;
                oldH=DmQueryNextInCategory(gLibDB,
                        &index, catInhabitant);
                if (!oldH) {
                    index=DmFindSortPosition(gLibDB,
                            &inhabitantP[counter],
                            NULL,(DmComparF *)
                            InhabitantCompareFunc,
                            NULL);
                    break;
                }
                oldRecord = ((inhabitant*)
                        MemHandleLock(oldH));
                MemHandleUnlock(oldH);
            }
        } else
            index=DmFindSortPosition(gLibDB,
                    &inhabitantP[counter], NULL,
                    (DmComparF *)
                    InhabitantCompareFunc, NULL);
        newRecordH = DmNewHandle(gLibDB,
                    sizeof(inhabitantP[counter]));
        if(!newRecordH)
            error = DmGetLastErr();
        else
        {
            DmWrite(MemHandleLock(newRecordH), 0,
                    &inhabitantP[counter],
                    sizeof(inhabitantP[counter]));
            error = DmAttachRecord(gLibDB,
                    &index, newRecordH,
                    (oldH==NULL)?NULL:&oldH);
            DmRecordInfo(gLibDB, index,
                    &attributes, NULL, NULL);
            attributes &= ~dmRecAttrCategoryMask;
            attributes |= catInhabitant;
            attributes |= dmRecAttrDirty;
```

```
            DmSetRecordInfo(gLibDB, index,
                    &attributes, NULL);
            MemHandleUnlock(newRecordH);
        }
        counter++;
    }
    return error;
}



// ********************************************** //
// **  Pregnancy comparion callback function  ** //
// ********************************************** //
Int16 PregnancyCompareFunc(pregnancy_information*
h1, pregnancy_information* h2, Int16 other,
SortRecordInfoPtr rec1SortInfo, SortRecordInfoPtr
rec2SortInfo, MemHandle appInfoH)
{
    if(h1->Number_pregnancy>h2->Number_pregnancy)
        return 1;
    if(h1->Number_pregnancy<h2->Number_pregnancy)
        return -1;
    return 0;
}



// ********************************************** //
// **  Update / store pregnancy information   ** //
// ********************************************** //
Err SavePregnancyForInhb(void)
{
    MemHandle newRecordH, oldH;
    pregnancy_information *oldRecord;
    UInt16 index=0;
    Err error=0;
    UInt16 attributes;
    int counter=0;

    while( (counter!=NbrOfElements.
            nbrOfPregnancyForX) && !error ) {
        Search for old ID
        index=0;
```

```
        oldH=DmQueryNextInCategory(gLibDB, &index,
                catPregnancyResult);
        if (oldH) {
            oldRecord = ((pregnancy_information*)
                    MemHandleLock(oldH));
            MemHandleUnlock(oldH);
            while( (oldRecord->Tab_id_habit !=
                    pregnancy_informationP[counter]
                    .Tab_id_habit) ) {
                index++;
                oldH=DmQueryNextInCategory(gLibDB,
                        &index, catPregnancyResult);
                if (!oldH) {
                    index=DmFindSortPosition(gLibDB,
                            &pregnancy_informationP
                            [counter], NULL,
                            (DmComparF *)
                            PregnancyCompareFunc,
                            NULL);
                    break;
                }
                oldRecord = ((pregnancy_information*)
                        MemHandleLock(oldH));
                MemHandleUnlock(oldH);
            }
        } else
            index=DmFindSortPosition(gLibDB,
                    &pregnancy_informationP[counter],
                    NULL, (DmComparF *)
                    PregnancyCompareFunc, NULL);
        newRecordH = DmNewHandle(gLibDB,
                sizeof(pregnancy_informationP
                [counter]));
        if(!newRecordH)
            error = DmGetLastErr();
        else
        {
            DmWrite(MemHandleLock(newRecordH), 0,
                    &pregnancy_informationP[counter],
                    sizeof(pregnancy_informationP
                    [counter]));
            error = DmAttachRecord(gLibDB, &index,
                    newRecordH,
                    (oldH==NULL)?NULL:&oldH);
```

```
            DmRecordInfo(gLibDB, index, &attributes,
                    NULL, NULL);
            attributes &= ~dmRecAttrCategoryMask;
            attributes |= catPregnancyResult;
            attributes |= dmRecAttrDirty;
            DmSetRecordInfo(gLibDB, index,
                    &attributes, NULL);
            MemHandleUnlock(newRecordH);
        }
        counter++;
    }
    return error;
}
```

9 Appendix B – Implementation of the conduit

```
Imports PDDirectLib
Imports PDStandardLib
Imports System.Data.OleDb
Imports System.Data.Odbc

Public Class Form1
    Inherits System.Windows.Forms.Form

#Region " Windows Form Designer generated code "

    Public Sub New()
        MyBase.New()

        'This call is required by the Windows Form Designer.
        InitializeComponent()

        'Add any initialization after the InitializeComponent() call

    End Sub

    'Form overrides dispose to clean up the component list.
    Protected Overloads Overrides Sub Dispose(ByVal disposing As Boolean)
        If disposing Then
            If Not (components Is Nothing) Then
                components.Dispose()
            End If
        End If
        MyBase.Dispose(disposing)
    End Sub

    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer
```

```
    Friend WithEvents cmdAbort As System.Windows.Forms.Button
    Friend WithEvents cmdLogIn As System.Windows.Forms.Button
    Friend WithEvents cmbUsers As System.Windows.Forms.ComboBox
    Friend WithEvents Button1 As System.Windows.Forms.Button
    Friend WithEvents Panel1 As System.Windows.Forms.Panel
    <System.Diagnostics.DebuggerStepThrough()> Private Sub
InitializeComponent()
        Me.cmdAbort = New System.Windows.Forms.Button
        Me.cmdLogIn = New System.Windows.Forms.Button
        Me.cmbUsers = New System.Windows.Forms.ComboBox
        Me.Button1 = New System.Windows.Forms.Button
        Me.Panel1 = New System.Windows.Forms.Panel
        Me.SuspendLayout()
        '
        'cmdAbort
        '
        Me.cmdAbort.Location = New System.Drawing.Point(104, 48)
        Me.cmdAbort.Name = "cmdAbort"
        Me.cmdAbort.TabIndex = 0
        Me.cmdAbort.Text = "Abort"
        '
        'cmdLogIn
        '
        Me.cmdLogIn.Location = New System.Drawing.Point(184, 48)
        Me.cmdLogIn.Name = "cmdLogIn"
        Me.cmdLogIn.TabIndex = 1
        Me.cmdLogIn.Text = "Login"
        '
        'cmbUsers
        '
        Me.cmbUsers.DisplayMember = "Name"
        Me.cmbUsers.Location = New System.Drawing.Point(8, 16)
        Me.cmbUsers.Name = "cmbUsers"
        Me.cmbUsers.Size = New System.Drawing.Size(256, 21)
        Me.cmbUsers.TabIndex = 2
        Me.cmbUsers.ValueMember = "Id"
```

```
        '
        'Button1
        '
        Me.Button1.Location = New System.Drawing.Point(16, 56)
        Me.Button1.Name = "Button1"
        Me.Button1.TabIndex = 3
        Me.Button1.Text = "Button1"
        '
        'Panel1
        '
        Me.Panel1.Location = New System.Drawing.Point(0, 0)
        Me.Panel1.Name = "Panel1"
        Me.Panel1.Size = New System.Drawing.Size(16, 16)
        Me.Panel1.TabIndex = 4
        '
        'Form1
        '
        Me.AutoScaleBaseSize = New System.Drawing.Size(5, 13)
        Me.ClientSize = New System.Drawing.Size(272, 86)
        Me.Controls.Add(Me.Panel1)
        Me.Controls.Add(Me.Button1)
        Me.Controls.Add(Me.cmbUsers)
        Me.Controls.Add(Me.cmdLogIn)
        Me.Controls.Add(Me.cmdAbort)
        Me.Name = "Form1"
        Me.Text = "Login"
        Me.ResumeLayout(False)

    End Sub

#End Region
```

```
Private Const PDA_DB_NAME As String = "DSS_DB-DSSD"
Private CLIENT_DB_CSTRING As String =
        "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & _
        Application.StartupPath & "/db/db.mdb;"
Private SERVER_DB_CSTRING As String =
        "DRIVER={MaxDB};SERVER=165.98.144.9;DATABASE=DSS
        DB;UID=CIDS;PWD=CIDS;"
Private _user As User

'The sub-routine that gets triggerd when the program starts up by the Form's
load event.
Private Sub Form1_Load(ByVal sender As Object, ByVal e As
        System.EventArgs) Handles MyBase.Load

    Dim con As New OleDbConnection(CLIENT_DB_CSTRING)
    Dim sql As New OleDbCommand("SELECT * FROM Employed", con)
    Dim reader As OleDbDataReader

    Try

        con.Open()
        reader = sql.ExecuteReader()
        cmbUsers.Items.Clear()

        While reader.Read

            Dim temp As New User
            temp.Id = reader("employeCode")
            temp.Name = reader("employeName")
            cmbUsers.Items.Add(temp)

        End While

    Catch ex As Exception

        If (MsgBox(ex.Message, MsgBoxStyle.OKOnly) =
            MsgBoxResult.OK) Then

            Me.Close()

        End If

    End Try
End Sub
```

```
Private Sub doTheConduitWork()
    'LOAD FROM PDA
    readFromPDA()

    'ERASE THE PDA DATABASE
    erasePalmDB()

    'WRITE TO PDA
    'Open connection to the database
    Dim con As New OleDbConnection(CLIENT_DB_CSTRING)
    Dim reader As OleDbDataReader
    con.Open()

    'Educations
    Try

        Dim sql_edu As New OleDbCommand("SELECT * FROM
            Educations", con)

        reader = sql_edu.ExecuteReader
        While reader.Read
            writeEducationToPDA(reader("eduCode"), reader("eduDescr"))
        End While

    Catch ex As Exception

        MsgBox(ex.Message)

    Finally

        reader.Close()

    End Try
```

```
'Occupations
Try
    Dim sql_ocu As New OleDbCommand("SELECT * FROM
        Occupations", con)

    reader = sql_ocu.ExecuteReader
    While reader.Read
        writeOccupationToPDA(reader("ocuCode"), reader("ocuDescr"))
    End While

Catch ex As Exception

    MsgBox(ex.Message)

Finally

    reader.Close()

End Try

'Relations
Try
    Dim sql_rel As New OleDbCommand("SELECT * FROM Relations",
        con)

    reader = sql_rel.ExecuteReader
    While reader.Read
        writeRelationToPDA(reader("relCode"), reader("relDescr"))
    End While

Catch ex As Exception

    MsgBox(ex.Message)

Finally

    reader.Close()

End Try
```

```
'House_conditions
Try
   Dim sql_hc As New OleDbCommand("SELECT * FROM
         HouseConditions", con)

   reader = sql_hc.ExecuteReader
   While reader.Read
      writeHouseConditionsToPDA(reader("hcType"), reader("hcDescr"))
   End While

Catch ex As Exception

   MsgBox(ex.Message)

Finally

   reader.Close()

End Try

'PREGNANCY_RESULT
Try
   Dim sql_pre As New OleDbCommand("SELECT * FROM
         PregnancyResults", con)

   reader = sql_pre.ExecuteReader
   While reader.Read
      writeResultToPDA(reader("preCode"), reader("preDesc"))
   End While

Catch ex As Exception

   MsgBox(ex.Message)

Finally

   reader.Close()

End Try
```

```
'Houses
'Retrive the awaiting houses from the client db
Dim sql_houses As New OleDbCommand("SELECT * FROM House
        WHERE toPDA=true AND PDAid=" & _user.Id, con)


Try
   Dim houseidcoll As New Collection

   reader = sql_houses.ExecuteReader
   While reader.Read

      Dim houseid As Integer = reader("tab_house_id")
      writeHouseToPDA(houseid, reader("household_code"),
        reader("Address"), reader("Water"), reader("Wall"),
        reader("Floor"), reader("Toilet"), reader("Sleeping_rooms"))

      'Adds the housid to the "to-delete-house-id-collection"
      houseidcoll.Add(houseid)

   End While
   reader.Close()

   'Delete the house from client db
   For n As Integer = 1 To houseidcoll.Count
      Dim sql_delete_house As New OleDbCommand("DELETE *
        FROM House WHERE tab_house_id= " & houseidcoll(n), con)
      sql_delete_house.ExecuteNonQuery()
   Next

Catch ex As Exception

   MsgBox(ex.Message)

Finally
   'If the program generated exeption and the reader is not closed,
   ' close the reader.
   If Not reader.IsClosed Then
      reader.Close()
   End If
End Try
```

```
'Inhabitants

    'Retrive the awaiting inhabitants from the client db
    Dim sql_inhabit As New OleDbCommand("SELECT * FROM
            Inhabitant WHERE toPDA=true AND PDAid=" & _user.Id, con)

    Try
       reader = sql_inhabit.ExecuteReader
       Dim inhidcoll As New Collection
       While reader.Read
          Dim inhid As Integer = reader("tab_id_habit")
          writeInhabitantToPDA(inhid, reader("Tab_id_habit"),
            reader("id_habit"), reader("First_name"), reader("Surname"),
            reader("Date_Of_Birth"), reader("tab_id_mother"),
            reader("occupation_code"), reader("education_code"),
            reader("relation_to_head_code"), reader("isaman"),
            reader("Birth_Cry_on_birth"), reader("Birth_Number_pregnant"),
            reader("Death_Date_of_death"), reader("Death_description"),
            reader("Death_Certificate_description"),
            reader("MigIn_Date_of_inmigration"),
            reader("MigIn_Cause_of_inmig"),
            reader("MigIn_Inmig_description"),
            reader("MigOut_Date_of_outmigration"),
            reader("MigOut_Cause_of_outmig"),
            reader("MigOut_Outmig_description"))

          'Adds the housid to the "to-delete-house-id-collection"
          inhidcoll.Add(inhid)
       End While
       reader.Close()

       'Delete the inhabitant from client db
       For n As Integer = 1 To inhidcoll.Count

            Dim sql_delete_inh As New OleDbCommand("DELETE * FROM
            Inhabitant WHERE Tab_id_habit= " & inhidcoll(n), con)
          sql_delete_inh.ExecuteNonQuery()

       Next

    Catch ex As Exception

       MsgBox(ex.Message)
```

```
    Finally
        'If the program generated exeption and the reader is not closed,
        ‘close the reader.
        If Not reader.IsClosed Then
            reader.Close()
        End If

    End Try


    'Pregnancies
    Try
        Dim sql_pre As New OleDbCommand("SELECT * FROM
              Pregnancy", con)

        reader = sql_pre.ExecuteReader
        While reader.Read
            writePregnancyToPDA(reader("Tab_id_habit"),
              reader("Children_in_preg"), reader("Birth_attendant"),
              reader("Place_of_delivery"), reader("Number_pregnancy"),
              reader("Result_code"))
        End While

    Catch ex As Exception

        MsgBox(ex.Message)

    Finally

        reader.Close()

    End Try

    con.Close()
End Sub
```

```
#Region "Button handling"

    '"Log In"-button clicked
    Private Sub cmdLogIn_Click(ByVal sender As System.Object, ByVal e As
                System.EventArgs) Handles cmdLogIn.Click

        If Not IsNothing(cmbUsers.SelectedItem) Then

            _user = cmbUsers.SelectedItem
            Dim t As New Threading.Thread(AddressOf doTheConduitWork)
            t.Start()

        Else

            MsgBox("Choose a user!", MsgBoxStyle.OKOnly)

        End If

    End Sub

    '"Abort"-button clicked
    Private Sub cmdAbort_Click(ByVal sender As System.Object, ByVal e As
                System.EventArgs) Handles cmdAbort.Click
        Me.Close()
    End Sub

#End Region
```

```
#Region "Read From PDA"

    Private Sub readFromPDA()

        'Declare the database connection
        Dim con As New OleDbConnection(CLIENT_DB_CSTRING)
        '   Declare the PDDatabaseQuery object
        Dim pDbQuery As New PDDatabaseQuery
        '   Declare the PDRecordAdapter object
        Dim pMemo As PDRecordAdapter
        Dim PHotSync As New PDHotSyncUtility

        If Not PHotSync.IsSyncInProgress Then
            MsgBox("COM Sync Suite objects are only active during the HotSync
                process. Refer to the COMSyncCompanion.pdf manual for more
                information on how to create a debug environment.",
                vbInformation, "Information")
            Exit Sub
        End If

        '   Open the Memo database
        pMemo = pDbQuery.OpenRecordDatabase(PDA_DB_NAME,
                "PDDirect.PDRecordAdapter",
                PDDirectLib.EAccessModes.eRead Or
                PDDirectLib.EAccessModes.eWrite Or
                PDDirectLib.EAccessModes.eShowSecret)

        '   Declare the record header and data
        Dim nIndex As Long
        Dim vUniqueId As Object
        Dim nCategory As Long
        Dim eAttributes As ERecordAttributes
        Dim vData As Object

        '   Declare the PDUtility object
        Dim pUtility As New PDUtility

        '   Declare the Memo string
        Dim sMemo As String

        '   Declare the count of records containing the string
        Dim nCount As Long
        Dim nTest As Object
```

```
'   Read the first record
pMemo.IterationIndex = 0

If (pMemo.RecordCount > 0) Then
    vData = pMemo.ReadNext(nIndex, vUniqueId, nCategory,
          eAttributes)

    '   Loop and search each Memo record for the string
    Do While Not pMemo.EOF
        '   Convert the Memo record to a string
        '   ByteArrayToBSTR returns the next offset
        '   but we don't need it
        pUtility.ByteArrayToBSTR(vData, 0, 32767, sMemo)


        Select Case nCategory
            Case 1
                'Occupation
                'Do nothing, only for transfer to PDA
            Case 2
                'Education
                'Do nothing, only for transfer to PDA
            Case 3
                'RelationType
                'Do nothing, only for transfer to PDA
```

```
Case 4
    'House

     Dim houseid As Integer
    Dim houseCode As String
    Dim houseAdress As String
    Dim water As String
    Dim wall As String
    Dim floor As String
    Dim toilet As String
    Dim sleepRooms As Integer
    pUtility.ByteArrayToDWORD(vData, 0, True, houseid)
    pUtility.ByteArrayToBSTR(vData, 4, 8, houseCode)
    pUtility.ByteArrayToBSTR(vData, 12, 201, houseAdress)
    pUtility.ByteArrayToBSTR(vData, 213, 5, water)
    pUtility.ByteArrayToBSTR(vData, 218, 5, wall)
    pUtility.ByteArrayToBSTR(vData, 223, 5, floor)
    pUtility.ByteArrayToBSTR(vData, 228, 5, toilet)
    pUtility.ByteArrayToWORD(vData, 234, True, sleepRooms)

    Try
       con.Open()
       Dim sql2 As New OleDbCommand("SELECT * FROM
             House WHERE tab_house_id=" & houseid, con)
       If sql2.ExecuteScalar > 0 Then
          Dim sql As New OleDbCommand("UPDATE House "& _
                "SET household_code='" & houseCode & "', " & _
                "Address ='" & houseAdress & "', " & _
                "Water='" & water & "', " & _
                "Wall='" & wall & "', " & _
                "Floor='" & floor & "', " & _
                "Toilet='" & toilet & "', " & _
                "Sleeping_rooms=" & sleepRooms & ", " & _
                "toPDA=false WHERE tab_house_id =" &
                      houseid, con)

          sql.ExecuteNonQuery()
```

```
        Else
            Dim sql As New OleDbCommand("INSERT INTO House
                VALUES(" & houseid & ",'" & houseCode & "','" &
                houseAdress & "','" & water & "','" & wall & "','" &
                floor & "','" & toilet & "'," & sleepRooms & ",false, -
                1)", con)
            sql.ExecuteNonQuery()
        End If

    Catch ex As Exception

        MsgBox(ex.Message)

    Finally

        con.Close()

    End Try
```

```
Case 5
   'Inhabitant

   Dim Tab_id_habit As Int32
   Dim Inhb_tab_house_id As Int32
   Dim Id_habit As String '10
   Dim First_name As String '51
   Dim Surname As String '51
   Dim Date_of_birth As Date
   Dim Tab_id_Mother As Int32
   Dim Occupation_code As Int16
   Dim Education_code As Int16
   Dim Relation_to_head_code As Int16
   Dim IsAMan As Boolean
   Dim Birth_Cry_on_birth As Boolean
   Dim Birth_Number_pregnant As Int16
   Dim Death_Date_of_death As Date
   Dim Death_description As String '301
   Dim death_certificate_description As String
   Dim MigIn_Date_of_inmigration As Date
   Dim MigIn_Cause_of_inmig As String '51
   Dim MigIn_Inmig_description As String '201
   Dim MigOut_Date_of_outmigration As Date
   Dim MigOut_Cause_of_outmig As String '51
   Dim MigOut_Outmig_description As String '201
   Dim isNewPerson As Boolean

   pUtility.ByteArrayToDWORD(vData, 0, True, Tab_id_habit)
   pUtility.ByteArrayToDWORD(vData, 4, True,
            Inhb_tab_house_id)
   pUtility.ByteArrayToBSTR(vData, 8, 10, Id_habit)
   pUtility.ByteArrayToBSTR(vData, 18, 51, First_name)
   pUtility.ByteArrayToBSTR(vData, 69, 51, Surname)

   Dim tempDayBirth, tempMonthBirth, tempYearBirth As
            Integer
   pUtility.ByteArrayToWORD(vData, 126, True, tempDayBirth)
   pUtility.ByteArrayToWORD(vData, 128, True,
            tempMonthBirth)
   pUtility.ByteArrayToWORD(vData, 130, True,
            tempYearBirth)
   Date_of_birth = checkDate(tempYearBirth, tempMonthBirth,
            tempDayBirth)
```

```
pUtility.ByteArrayToDWORD(vData, 134, True,
        Tab_id_Mother)
pUtility.ByteArrayToWORD(vData, 138, True,
        Occupation_code)
pUtility.ByteArrayToWORD(vData, 140, True,
        Education_code)
pUtility.ByteArrayToWORD(vData, 142, True,
        Relation_to_head_code)
IsAMan = vData(144)
Birth_Cry_on_birth = vData(145)
pUtility.ByteArrayToWORD(vData, 146, True,
        Birth_Number_pregnant)

Dim tempDayDeath, tempMonthDeath, tempYearDeath As
        Integer
pUtility.ByteArrayToWORD(vData, 154, True,
        tempDayDeath)
pUtility.ByteArrayToWORD(vData, 156, True,
        tempMonthDeath)
pUtility.ByteArrayToWORD(vData, 158, True,
        tempYearDeath)
Death_Date_of_death = checkDate(tempYearDeath,
        tempMonthDeath, tempDayDeath)
pUtility.ByteArrayToBSTR(vData, 162, 301,
        Death_description)
pUtility.ByteArrayToBSTR(vData, 463, 200,
        death_certificate_description)

Dim tempDayInMig, tempMonthInMig, tempYearInMig As
        Integer
pUtility.ByteArrayToWORD(vData, 624, True,
        tempDayInMig)
pUtility.ByteArrayToWORD(vData, 626, True,
        tempMonthInMig)
pUtility.ByteArrayToWORD(vData, 628, True,
        tempYearInMig)
MigIn_Date_of_inmigration = checkDate(tempYearInMig,
        tempMonthInMig, tempDayInMig)
pUtility.ByteArrayToBSTR(vData, 632, 51,
        MigIn_Cause_of_inmig)
pUtility.ByteArrayToBSTR(vData, 683, 201,
        MigIn_Inmig_description)

Dim tempDayOutMig, tempMonthOutMig, tempYearOutMig
        As Integer
```

```
pUtility.ByteArrayToWORD(vData, 890, True,
        tempDayOutMig)
pUtility.ByteArrayToWORD(vData, 892, True,
        tempMonthOutMig)
pUtility.ByteArrayToWORD(vData, 894, True,
        tempYearOutMig)
MigOut_Date_of_outmigration = checkDate(tempYearOutMig,
        tempMonthOutMig, tempDayOutMig)

pUtility.ByteArrayToBSTR(vData, 898, 51,
        MigOut_Cause_of_outmig)
pUtility.ByteArrayToBSTR(vData, 949, 201,
        MigOut_Outmig_description)
isNewPerson = vData(1150)

Try
   con.Open()
   Dim sql2 As New OleDbCommand("SELECT * FROM
        Inhabitant WHERE Tab_id_habit=" & Tab_id_habit,
        con)
   If sql2.ExecuteScalar > 0 Then
      Dim sql As New OleDbCommand("UPDATE Inhabitant "
                  "SET inhb_tab_house_id='" &
                        Inhb_tab_house_id & "', " & _
                  "id_habit ='" & Id_habit & "', " & _
                  "First_name ='" & First_name & "', " &
                  "Surname ='" & Surname & "', " & _
                  "Date_Of_Birth =#" & Date_of_birth &
                        "#, " & _
                  "tab_id_mother =" & Tab_id_Mother &
                        ", " & _
                  "occupation_code =" &
                        Occupation_code & ", " & _
                  "education_code =" & Education_code
                        & ", " & _
                  "relation_to_head_code =" &
                        Relation_to_head_code & ", "
                        & _
                  "isaman=" & IsAMan & ", " & _
                  "Birth_Cry_on_birth=" &
                        Birth_Cry_on_birth & ", " & _
                  "Birth_Number_pregnant=" &
                        Birth_Number_pregnant & ", "
```

```
                "Death_Date_of_death=#" &
                    Death_Date_of_death & "#, " &
                "Death_description='" &
                    Death_description & "', " & _
                "MigIn_Date_of_inmigration=#" &
                    MigIn_Date_of_inmigration &
                    "#, " & _
                "MigIn_Cause_of_inmig='" &
                    MigIn_Cause_of_inmig & "', "
                    & _
                "MigIn_Inmig_description='" &
                    MigIn_Inmig_description & "',
                    " & _
                "MigOut_Date_of_outmigration=#" &
                    MigOut_Date_of_outmigration
                    & "#, " & _
                "MigOut_Cause_of_outmig='" &
                    MigOut_Cause_of_outmig & "'
            WHERE tab_id_habit =" & Tab_id_habit,
                    con)

    sql.ExecuteNonQuery()

Else

    Dim sql As New OleDbCommand("INSERT INTO
        Inhabitant(Tab_id_habit,inhb_tab_house_id,id_habit,Fi
        rst_name,Surname,Date_Of_Birth,tab_id_mother,occu
        pation_code,education_code,relation_to_head_code,isa
        man,Birth_Cry_on_birth,Birth_Number_pregnant,Deat
        h_Date_of_death,Death_description,MigIn_Date_of_in
        migration,MigIn_Cause_of_inmig,MigIn_Inmig_descr
        iption,MigOut_Date_of_outmigration,MigOut_Cause_
        of_outmig,MigOut_Outmig_description) VALUES("
        & Tab_id_habit & "," & Inhb_tab_house_id & ",'" &
        Id_habit & "','" & First_name & "','" & Surname &
        "',#" & Date_of_birth & "#," & Tab_id_Mother & ","
        & Occupation_code & "," & Education_code & "," &
        Relation_to_head_code & "," & IsAMan & "," &
        Birth_Cry_on_birth & "," & Birth_Number_pregnant
        & ",#" & Death_Date_of_death & "#,'" &
        Death_description & "',#" &
        MigIn_Date_of_inmigration & "#,'" &
        MigIn_Cause_of_inmig & "','" &
        MigIn_Inmig_description & "',#" &
```

```
                MigOut_Date_of_outmigration & "#,'" &
                MigOut_Cause_of_outmig & "','" &
                MigOut_Outmig_description & "')", con)

            sql.ExecuteNonQuery()

        End If

    Catch ex As Exception

        MsgBox(ex.Message)

    Finally

        con.Close()

    End Try
```

```
Case 6
   'Pregnancy
   Dim Tab_id_habit As Int32
   Dim Children_in_preg As Int16
   Dim Birth_attendant As String '51
   Dim Place_of_delivery As String '51
   Dim Number_pregnancy As Int16
   Dim Result_code As Int16

   pUtility.ByteArrayToDWORD(vData, 0, True, Tab_id_habit)
   pUtility.ByteArrayToWORD(vData, 4, True,
            Children_in_preg)
   pUtility.ByteArrayToBSTR(vData, 6, 51, Birth_attendant)
   pUtility.ByteArrayToBSTR(vData, 57, 51, Place_of_delivery)
   pUtility.ByteArrayToWORD(vData, 108, True,
            Number_pregnancy)
   pUtility.ByteArrayToWORD(vData, 110, True, Result_code)

   Try
      con.Open()
      Dim sql2 As New OleDbCommand("SELECT * FROM
            Pregnancy WHERE Tab_id_habit=" & Tab_id_habit &
            " AND Number_pregnancy =" & Number_pregnancy,
            con)
      If sql2.ExecuteScalar > 0 Then
         Dim sql As New OleDbCommand("UPDATE Pragnancy”
            & “SET Children_in_preg=" & Children_in_preg & ",
            " & "Birth_attendant ='" & Birth_attendant & "', " & _
           "Place_of_delivery='" & Place_of_delivery & "', " & _
            "Result_code=" & Result_code & " " & _
            "WHERE Tab_id_habit =" & Tab_id_habit & " AND
            Number_pregnancy =" & Number_pregnancy, con)
            sql.ExecuteNonQuery()

      Else
         Dim sql As New OleDbCommand("INSERT INTO
            Pregnancy(Tab_id_habit,Children_in_preg,
            Birth_attendant,Place_of_delivery,Number_pregnancy,
            Result_code,toPDA,PDAid) VALUES(" &
            Tab_id_habit & "," & Children_in_preg & ",'" &
            Birth_attendant & "','" & Place_of_delivery & "'," &
            Number_pregnancy & "," & Result_code & ",false,-1",
            con)
         sql.ExecuteNonQuery()
      End If
```

```
                Catch ex As Exception

                    MsgBox(ex.Message)

                Finally

                    con.Close()

                End Try

            Case 7
                'PregnancyResult
                'Do nothing, only for transfer to PDA'
            Case 8
                'HouseCondition
                'Do nothing, only for transfer to PDA'

        End Select
        '   Read the next record
        vData = pMemo.ReadNext(nIndex, vUniqueId, nCategory,
          eAttributes)
    Loop

  End If

 End Sub
#End Region
```

```
#Region "Write to PDA"

    Private Sub erasePalmDB()
        '   Declare the PDDatabaseQuery object
        Dim pDbQuery As New PDDatabaseQuery
        '   Declare the PDRecordAdapter object
        Dim pMemo As PDRecordAdapter
        Dim PHotSync As New PDHotSyncUtility

        If Not PHotSync.IsSyncInProgress Then
            MsgBox("COM Sync Suite objects are only active during the HotSync
                process. Refer to the COMSyncCompanion.pdf manual for more
                information on how to create a debug environment.",
                vbInformation, "Information")
            Exit Sub
        End If

        '   Open the Memo database
        pMemo = pDbQuery.OpenRecordDatabase(PDA_DB_NAME,
                "PDDirect.PDRecordAdapter",
                PDDirectLib.EAccessModes.eRead Or
                PDDirectLib.EAccessModes.eWrite Or
                PDDirectLib.EAccessModes.eShowSecret)

        If pMemo.RecordCount > 0 Then
            pMemo.RemoveSet(ERemoveSetType.eRemoveAllRecords)
        End If

    End Sub
```

```
Private Sub writeHouseToPDA(ByVal houseid As Integer, ByVal
        housecode As String, ByVal address As String, ByVal water As
        String, ByVal walls As String, ByVal floor As String, ByVal toilet
        As String, ByVal nbrRooms As Integer)

    Dim WPQuery As New PDDatabaseQuery
    Dim UniqueId As Object
    Dim WPRecAdapt As PDRecordAdapter
    '   Declare the PDUtility object
    Dim pUtility As New PDUtility

    WPRecAdapt =
        CType(WPQuery.OpenRecordDatabase(PDA_DB_NAME,
        "PDDirect.PDRecordAdapter",
        PDDirectLib.EAccessModes.eWrite), PDRecordAdapter)

    Dim vData(235) As Byte
    pUtility.DWORDToByteArray(vData, 0, True, houseid)
    pUtility.BSTRToByteArray(vData, 4, housecode)
    pUtility.BSTRToByteArray(vData, 12, address)
    pUtility.BSTRToByteArray(vData, 213, water)
    pUtility.BSTRToByteArray(vData, 218, walls)
    pUtility.BSTRToByteArray(vData, 223, floor)
    pUtility.BSTRToByteArray(vData, 228, toilet)
    pUtility.WORDToByteArray(vData, 232, True, nbrRooms)

    WPRecAdapt.Write(UniqueId, 4,
        PDStandardLib.ERecordAttributes.eDirty, vData)

End Sub
```

```
Private Sub writeEducationToPDA(ByVal eduCode As Integer, ByVal
        eduDesc As String)

    Dim WPQuery As New PDDatabaseQuery
    Dim UniqueId As Object
    Dim WPRecAdapt As PDRecordAdapter
    '   Declare the PDUtility object
    Dim pUtility As New PDUtility

    WPRecAdapt =
        CType(WPQuery.OpenRecordDatabase(PDA_DB_NAME,
        "PDDirect.PDRecordAdapter",
        PDDirectLib.EAccessModes.eWrite), PDRecordAdapter)

    Dim vData(55) As Byte
    pUtility.BSTRToByteArray(vData, 0, eduDesc)
    pUtility.WORDToByteArray(vData, 52, True, eduCode)

    WPRecAdapt.Write(UniqueId, 2,
        PDStandardLib.ERecordAttributes.eDirty, vData)
End Sub
```

```
Private Sub writeOccupationToPDA(ByVal occCode As Integer, ByVal
        occDesc As String)
    Dim WPQuery As New PDDatabaseQuery
    Dim UniqueId As Object
    Dim WPRecAdapt As PDRecordAdapter
    '   Declare the PDUtility object
    Dim pUtility As New PDUtility

    WPRecAdapt =
        CType(WPQuery.OpenRecordDatabase(PDA_DB_NAME,
        "PDDirect.PDRecordAdapter",
        PDDirectLib.EAccessModes.eWrite), PDRecordAdapter)

    Dim vData(55) As Byte
    pUtility.BSTRToByteArray(vData, 0, occDesc)
    pUtility.WORDToByteArray(vData, 52, True, occCode)

    WPRecAdapt.Write(UniqueId, 1,
        PDStandardLib.ERecordAttributes.eDirty, vData)

End Sub
```

```
Private Sub writeRelationToPDA(ByVal relCode As Integer, ByVal
        relDesc As String)
    Dim WPQuery As New PDDatabaseQuery
    Dim UniqueId As Object
    Dim WPRecAdapt As PDRecordAdapter
    '   Declare the PDUtility object
    Dim pUtility As New PDUtility

    WPRecAdapt =
        CType(WPQuery.OpenRecordDatabase(PDA_DB_NAME,
        "PDDirect.PDRecordAdapter",
        PDDirectLib.EAccessModes.eWrite), PDRecordAdapter)

    Dim vData(55) As Byte
    pUtility.BSTRToByteArray(vData, 0, relDesc)
    pUtility.WORDToByteArray(vData, 52, True, relCode)

    WPRecAdapt.Write(UniqueId, 3,
        PDStandardLib.ERecordAttributes.eDirty, vData)

End Sub

Private Sub writeHouseConditionsToPDA(ByVal hcType As String, ByVal
        hcDesc As String)
    Dim WPQuery As New PDDatabaseQuery
    Dim UniqueId As Object
    Dim WPRecAdapt As PDRecordAdapter
    '   Declare the PDUtility object
    Dim pUtility As New PDUtility

    WPRecAdapt =
        CType(WPQuery.OpenRecordDatabase(PDA_DB_NAME,
        "PDDirect.PDRecordAdapter",
        PDDirectLib.EAccessModes.eWrite), PDRecordAdapter)

    Dim vData(55) As Byte
    pUtility.BSTRToByteArray(vData, 0, hcDesc)
    pUtility.BSTRToByteArray(vData, 51, hcType)

    WPRecAdapt.Write(UniqueId, 8,
        PDStandardLib.ERecordAttributes.eDirty, vData)

End Sub
```

```
Private Sub writeResultToPDA(ByVal preCode As String, ByVal preDesc
        As String)
    Dim WPQuery As New PDDatabaseQuery
    Dim UniqueId As Object
    Dim WPRecAdapt As PDRecordAdapter
    '   Declare the PDUtility object
    Dim pUtility As New PDUtility

    WPRecAdapt =
        CType(WPQuery.OpenRecordDatabase(PDA_DB_NAME,
        "PDDirect.PDRecordAdapter",
        PDDirectLib.EAccessModes.eWrite), PDRecordAdapter)

    Dim vData(35) As Byte
    pUtility.BSTRToByteArray(vData, 0, preDesc)
    pUtility.WORDToByteArray(vData, 32, True, preCode)

    WPRecAdapt.Write(UniqueId, 7,
        PDStandardLib.ERecordAttributes.eDirty, vData)

End Sub
```

```
Private Sub writeInhabitantToPDA(ByVal Tab_id_habit As Integer, ByVal
        Inhb_tab_house_id As Integer, ByVal Id_habit As String, ByVal
        First_name As String, ByVal Surname As String, ByVal
        Date_of_birth As Date, ByVal Tab_id_Mother As Integer, ByVal
        Occupation_code As Integer, ByVal Education_code As Integer,
        ByVal Relation_to_head_code As Integer, ByVal IsAMan As
        Boolean, ByVal Birth_Cry_on_birth As Boolean, ByVal
        Birth_Number_pregnant As Integer, ByVal Death_Date_of_death
        As Date, ByVal Death_description As String, ByVal
        Death_Certificate_description As String, ByVal
        MigIn_Date_of_inmigration As Date, ByVal
        MigIn_Cause_of_inmig As String, ByVal
        MigIn_Inmig_description As String, ByVal
        MigOut_Date_of_outmigration As Date, ByVal
        MigOut_Cause_of_outmig As String, ByVal
        MigOut_Outmig_description As String)

    Dim WPQuery As New PDDatabaseQuery
    Dim UniqueId As Object
    Dim WPRecAdapt As PDRecordAdapter
    '   Declare the PDUtility object
    Dim pUtility As New PDUtility

    WPRecAdapt =
        CType(WPQuery.OpenRecordDatabase(PDA_DB_NAME,
        "PDDirect.PDRecordAdapter",
        PDDirectLib.EAccessModes.eWrite), PDRecordAdapter)

    Dim vData(1151) As Byte
    pUtility.DWORDToByteArray(vData, 0, True, Tab_id_habit)
    pUtility.DWORDToByteArray(vData, 4, True, Inhb_tab_house_id)
    pUtility.BSTRToByteArray(vData, 8, Id_habit)
    pUtility.BSTRToByteArray(vData, 18, First_name)
    pUtility.BSTRToByteArray(vData, 69, Surname)
    pUtility.WORDToByteArray(vData, 126, True, Date_of_birth.Day)
    pUtility.WORDToByteArray(vData, 128, True, Date_of_birth.Month)
    pUtility.WORDToByteArray(vData, 130, True, Date_of_birth.Year)
    pUtility.DWORDToByteArray(vData, 134, True, Tab_id_Mother)
    pUtility.WORDToByteArray(vData, 138, True, Occupation_code)
    pUtility.WORDToByteArray(vData, 140, True, Education_code)
    pUtility.WORDToByteArray(vData, 142, True, Relation_to_head_code)
    vData(144) = IsAMan
    vData(145) = Birth_Cry_on_birth
    pUtility.WORDToByteArray(vData, 146, True, Birth_Number_pregnant)
    pUtility.WORDToByteArray(vData, 154, True,
```

```
            Death_Date_of_death.Day)
    pUtility.WORDToByteArray(vData, 156, True,
            Death_Date_of_death.Month)
    pUtility.WORDToByteArray(vData, 158, True,
            Death_Date_of_death.Year)
    pUtility.BSTRToByteArray(vData, 162, Death_description)
    pUtility.BSTRToByteArray(vData, 463, Death_Certificate_description)
    pUtility.WORDToByteArray(vData, 624, True,
            MigIn_Date_of_inmigration.Day)
    pUtility.WORDToByteArray(vData, 626, True,
            MigIn_Date_of_inmigration.Month)
    pUtility.WORDToByteArray(vData, 628, True,
            MigIn_Date_of_inmigration.Year)
    pUtility.BSTRToByteArray(vData, 632, MigIn_Cause_of_inmig)
    pUtility.BSTRToByteArray(vData, 683, MigIn_Inmig_description)
    pUtility.WORDToByteArray(vData, 890, True,
            MigOut_Date_of_outmigration.Day)
    pUtility.WORDToByteArray(vData, 892, True,
            MigOut_Date_of_outmigration.Month)
    pUtility.WORDToByteArray(vData, 894, True,
            MigOut_Date_of_outmigration.Year)
    pUtility.BSTRToByteArray(vData, 898, MigOut_Cause_of_outmig)
    pUtility.BSTRToByteArray(vData, 949, MigOut_Outmig_description)
    vData(1150) = False
    WPRecAdapt.Write(UniqueId, 7,
            PDStandardLib.ERecordAttributes.eDirty, vData)

End Sub
```

```
    Private Sub writePregnancyToPDA(ByVal Tab_id_habit As Int32, ByVal
            Children_in_preg As Int16, ByVal Birth_attendant As String,
            ByVal Place_of_delivery As String, ByVal Number_pregnancy As
            Int16, ByVal Result_code As Int16)

        Dim WPQuery As New PDDatabaseQuery
        Dim UniqueId As Object
        Dim WPRecAdapt As PDRecordAdapter
        '   Declare the PDUtility object
        Dim pUtility As New PDUtility

        WPRecAdapt =
            CType(WPQuery.OpenRecordDatabase(PDA_DB_NAME,
            "PDDirect.PDRecordAdapter",
            PDDirectLib.EAccessModes.eWrite), PDRecordAdapter)

        Dim vData(111) As Byte
        'Pregnancy

        pUtility.DWORDToByteArray(vData, 0, True, Tab_id_habit)
        pUtility.WORDToByteArray(vData, 4, True, Children_in_preg)
        pUtility.BSTRToByteArray(vData, 6, Birth_attendant)
        pUtility.BSTRToByteArray(vData, 57, Place_of_delivery)
        pUtility.WORDToByteArray(vData, 108, True, Number_pregnancy)
        pUtility.WORDToByteArray(vData, 110, True, Result_code)

        WPRecAdapt.Write(UniqueId, 7,
            PDStandardLib.ERecordAttributes.eDirty, vData)
    End Sub

#End Region
```

```
#Region "Subclasses"

    Private Class User
        Private _id As Integer
        Private _name As String

        Public Property Id() As Integer

            Get
                Return _id
            End Get

            Set(ByVal Value As Integer)
                _id = Value
            End Set

        End Property

        Public Property Name() As String

            Get
                Return _name
            End Get

            Set(ByVal Value As String)
                _name = Value
            End Set

        End Property

    End Class

#End Region
```

```
#Region "Functions"

    Private Function PalmLongToDate(ByVal d As Long) As Date

        Dim SecsSince1904 As Double
        Dim DaysSince1904 As Double

        Const SecsPerDay As Long = 86400
        Const UnsignedLngMax As Double = 4294967296.0#

        ' Handle signed/unsigned issues and use a double to prevent overflow.
        If d < 0 Then
            SecsSince1904 = UnsignedLngMax + d
        Else
            SecsSince1904 = d
        End If

        ' Figure out how many days have passed since 1904. Then add to
        ' earliest possible date. Let VB adjust for leap year, etc!
        DaysSince1904 = SecsSince1904 / SecsPerDay
        Return DateSerial(1904, 1, 1).AddDays(CLng(DaysSince1904))
    End Function

    Private Function PalmLongToTime(ByVal T As Long) As Date

        Dim Hours As Integer
        Dim Minutes As Integer
        Dim Seconds As Integer

        ' Strip off any vestigal seconds, handle signed/unsigned issues.
        If T < 0 Then T = T + &H10000
        T = T And &H1FFFF

        ' Calculate hours, minutes and seconds based
        Hours = T \ 3600
        Minutes = (T \ 60) Mod 60
        Seconds = T Mod 60

        Return TimeSerial(Hours, Minutes, Seconds)

    End Function
```

```
    Private Function checkDate(ByRef year As Integer, ByRef month As
            Integer, ByRef day As Integer) As Date

        If ((year < 1) Or (month < 1) Or (day < 1)) Or ((year > 2100) Or (month
                > 12) Or (day > 31)) Then
            Return New Date(1700, 1, 1)
        Else
            Return New Date(year, month, day)
        End If

    End Function
#End Region
```