LUND
UNIVERSITY

Centre for Demographic and Health Research

Demographic Surveillance System Database

Final Report

Author

Jon Lennryd

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

Contents

Sammanfattning

Detta examensarbete presenterar ett förslag på en konceptuell design av en demografisk databas för CIDS i Nicaragua. Vidare beskrivs en implementation av denna databas. Slutligen beskrivs en metod för att överföra data från den nuvarande databasen till den nya, tillsammans med en mindre implementation.

CIDS – Center of Investigation of Demography and Health är placerat i Nicaragua, Leon, och arbetar med att samla in statistik om befolkningen. Deras huvudområden är graviditetsstatistik, barnadödlighet, folkhälsa, utbildningsnivå, uppskatta storleken på fattigdom, hur folk flyttar inom landet samt hur de flyttar in och ut ur landet.

Allt detta lagras och bearbetas i ett databassystem, som är uppbyggt i Microsoft Access. Systemet har utvecklats under lång tid, och har anpassats efterhand som nya krav har tillkommit. Idag är det en komplex databas med väldigt många olika typer av information, varur forskare och andra kan hämta historiska data från 1993 fram till idag. CIDS är unikt i Nicaragua, det finns ingen statlig eller kommunal organisation som samlar in denna typ av statistik, och det innebär att CIDS arbete är mycket viktigt för att kunna få en korrekt bild av Nicaraguas befolkning.

Den nuvarande databasdesignen lider av ’ihoplappningssyndromet’, det vill säga att de under tidens gång har lagt till ny funktionalitet och tabeller, och ständigt lappat ihop det nya med det gamla. Resultatet efter mer än tio års lappning och påbyggnad är en tämligen svåröverskådlig och ömtålig design, med många undantagsregler och givetvis en mängd fel som smugit sig in. Organisatoriska misstag har också gjort att en del data har försvunnit. Missförstånd har orsakat att en del data har matats in felaktigt, så att denna i slutändan har blivit oanvändbar. Avsaknaden av dokumentation har inneburit att när folk med kunskap har slutat, så har senare försök att ändra i databasens design försvårats, och misstag har begåtts.

Mitt examensarbete har bestått i att presentera en ny databasdesign som kan ersätta den nuvarande. Den nya designen skulle lösa många av de nuvarande problemen och dessutom underlätta framtida förändringar.

Det var av yttersta vikt att också kunna presentera en metod för att överföra den befintliga datan från den nuvarande databasen till den nya på ett fullständigt säkert sätt, där man kan visa att ingen data och inga relationer går förlorade i processen.

Det skulle också finnas ett användargränssnitt för att mata in ny data, och redigera befintlig, samt ett gränssnitt för att kunna samla ihop och presentera statistiken för de forskare som sedan ska använda sig av den.

Mitt examensarbete bygger vidare på ett arbete som Nils Assarsson och Tommy Ljunggren gjorde 2003 [5], där de identifierade de brister och problem som finns i CIDS databassystem, och även i deras sätt att samla in statistiken.

Nyckelord

Longitudinell Databasdesign, Demographic, Konceptuell Databasdesign, Center of Investigation of Demography and Health

Abstract

This examination work presents a proposal of a conceptual design of a demographic database for CIDS in Nicaragua. Furthermore, an implementation of this database is described. Finally a description of a method to transfer data from the current database to the new is given, together with a smaller implementation.

CIDS – Center of Investigation of Demography and Health is located in Nicaragua, Leon, and works with collecting statistics about the population. Their main areas of work are statistics about pregnancies, infant mortality, public health, levels of education, poverty index, internal migration, immigration and emigration.

All this information is stored and processed in a database system, which is created in Microsoft Access. The system has been developed during a long time, and gradually adapted when new requirements has been added. Today it is a complex database with very much and diverse types of information, from where researchers and others can get historical data spanning from 1993 until today. CIDS is unique in Nicaragua, there is no governmental or communal organization which collects this type of statistics, which means that CIDS work is very important in order to get a correct picture of the population of Nicaragua.

The current database suffers from the 'patch and paste' syndrome, which means that new functionality and tables has been added during its long existence, and fixed and patched so it will work with the current design. The result of more than ten years of fixing and adding is a quite fragile design, which is hard to understand and maintain. Lots of errors have been inserted as well, due to a bad user interface. Organizational mistakes have made some data disappear. Misinterpretations have caused some data to get entered wrong, which in the end have rendered the data unusable. The lack of updated documentation has meant problems when a person with knowledge about the system has disappeared from CIDS, so later changes and updates of the database's design has been difficult, and mistakes has been done.

The main point of this examination work is to present a new database design which can replace the current. This new design should solve most of the current problems and also simplify future changes of the design.

It was of the greatest importance to also present a method to transfer the existing data from the current database to the new one in a completely safe way, where it is possible to show that no data and no relations are lost in the process.

There should also be a user interface, where new data could be entered and changed, and an interface for collecting and present the statistics to the researchers which will use it in their work.

My examination work is a continuation of a work that Nils Assarsson and Tommy Ljunggren did in 2003 [5], where they identified the shortcomings and problems existing at CIDS in their database system and their way of collecting the statistical information.

Keywords

Longitudal Database Design, Demographic, Conceptual Database Design, Center of Investigation of Demography and Health

1 Introduction

Me and Robert Tublén first heard about CIDS in 2005, when the previous group of students made a presentation at LTH in Helsingborg. We soon got in contact with Nils Assarsson, which together with Tommy Ljunggren in 2003 had done their bachelor thesis in Nicaragua. Their work describes a complete analysis as well as a solution to the problem CIDS has at the moment.

This examination job is a sub-set of Nils and Tommy's work; to implement a new database, fulfilling the requirements they had defined. The aim has been to implement the user interface and the process of transferring the data from the current to the new database.

During the work this project turned out to be several distinct and possibly separate projects. First, to create a conceptual design of the database which fulfils the needs defined in Nils and Tommy's work. [5]

Secondly, implement the database. This included a lot of decisions, and was tightly connected with the development of the conceptual design.

Third, import and verify the data from the database they currently use. This became the most important and time consuming task.

Fourth, create an access system, with which researchers and students can use to access the data in a sensible way. It should also be used by the persons responsible for maintaining and entering the data.

Since the project quickly turned out to be so large, Robert and I decided to design and implement the database, and then work with different parts of the project. Robert concentrated on the access system for the users and researchers, while I concentrated on the import and verification of the data from the existing database.

Much time has been devoted in investigating the organisation of CIDS, the data collection process and the reasons for collecting the data. After reading these documents, you will have a good understanding of the current database design they use at CIDS as well as my suggestion to a new design.

1.1 Purpose

The main purpose of this examination work is to create a conceptual design of a demographic surveillance database.

The sub-goals are to make an implementation of this conceptual design and to import and verify the data from the current database into the new database.

1.2 Intended Audience

Intended audience is people who will implement the database, and people working with the database when it is implemented. Also the people working at CIDS might have good use of reading this work.

2 Problem description

The current database at CIDS, named Linea de Base [7], is a result of ten years of re-design, adding new functionality, with different people working with the design. There is a lack of overall design, long term planning and lack of sufficient training of personnel. There have been loss of data during re-design phases, and no restoration has been done. Also, there is no person who is responsible for the design and everyday maintenance.

The documentation of the database design and actual implementation is sparse and outdated, which makes the organisation depending very much on just a few persons with most of the knowledge. This makes it very time consuming and expensive to modify and support the database.

The complicated and redundant implementation of the database creates a lot of exceptions in the normal workflow, which requires the technicians to manually calculate and collect data from the database. This steal much time from their normal work, and it is easy to make mistakes. Ideally such a process should be fully automated by designated scripts created by database experts in cooperation with the professor of the area of interest. As an example the last public presentation of the statistics of the population [2] took one and a half year to assemble.

3 About CIDS

CIDS is short hand for Centro de Investigación en Demografía y Salud, or Center of Investigation of Demography and Health. CIDS operates a continuous monitoring system of vital statistics of the population in selected rural and urban areas in the municipal of León. Also generates statistics over basic health data, register trends and transitions in disease and mortality patterns. Moreover, serves as sampling frame for research in various specific studies and function as a planning instrument for the health services.

Figure 1. CIDS logotype

Figure 2 shows the selected areas of the municipal of León, and figure 3 shows the selected areas of the city of León. These are carefully selected to get a good average of the rural and urban populations.

For a deeper presentation of CIDS, see the document Demographic and Health Research. [3]

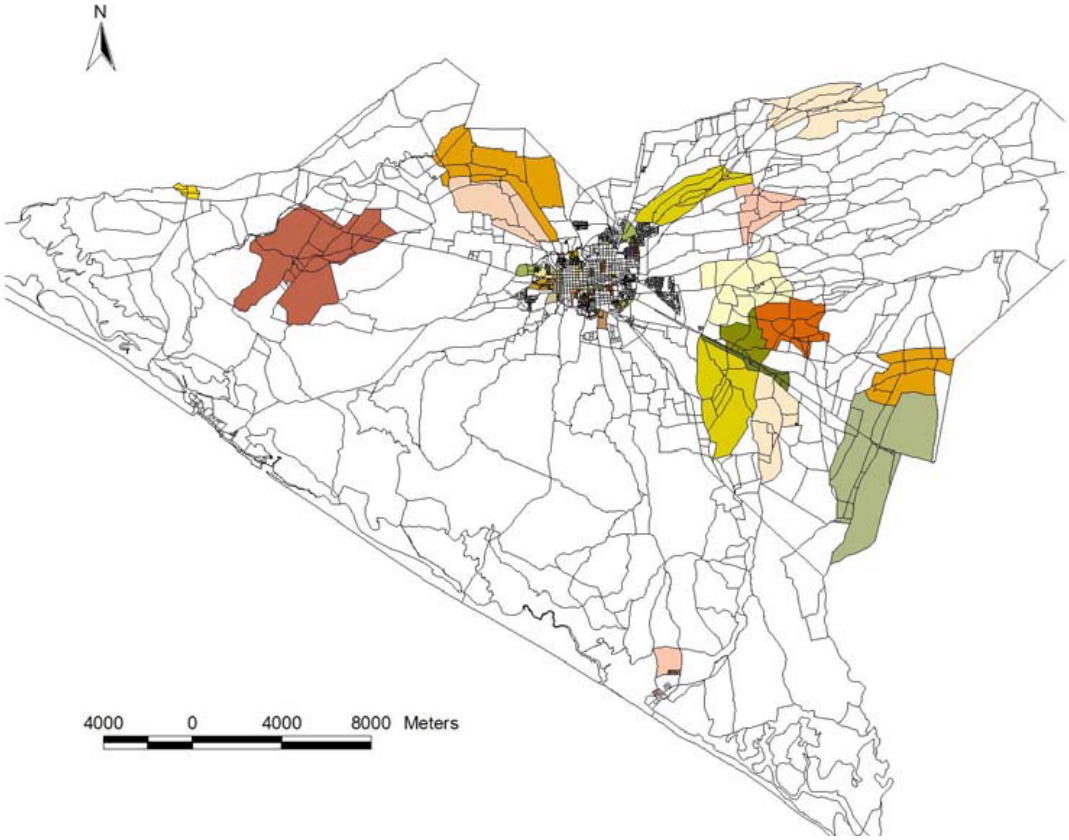


Figure 2. The municipal of León

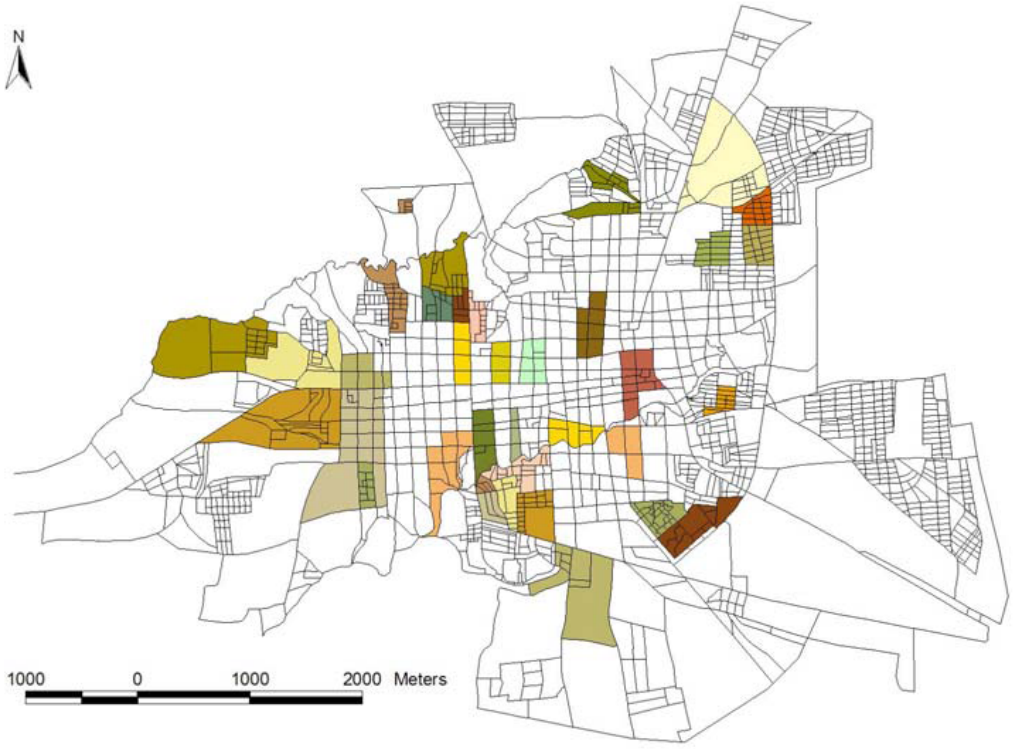


Figure 3. The city of León

3.1 What is a demographic surveillance system?

Simply put, it is a system for storing collected data about a population. It is used to produce useful statistics of various types, which is used by researchers, government etcetera. Different countries and regions collect somewhat different types of data, and apply different algorithms to create the statistics. There exist an international standard for how the statistics should be assembled, in order to make it possible to compare different countries. CIDS work with different factors such as; the standard of the houses, the education levels and job types of the people living in the houses. These factors are weighed together to create a comparable number which give a good picture of the wealth of the house, sub-population or the whole population.

Another large part of the surveillance system is the pregnancy information system which store information about women's pregnancies, and the mortality of infants. This statistics is very important since it gives very much information about the general standards of the hospitals and the health care of the country. The statistics can also be used in the work against the abuse of women and children.

The statistics about how and why people move inside of the areas give hints about general trends, for example an upswing in the number of students in an area can be connected with the fact that a new school has started in the same area, and a new factory is connected with much people moving in to the area. This statistics tell much about an area's development, but also how people move to different houses through the different parts of their life.

For a good example of how the data they collect is used, see the document Sistema de vigilancia en demografía y salud [2]. Another good example is Home Alone – Sibling caretakers in León, Nicaragua [11].

3.2 The process of collecting the data

Most people at CIDS is almost never there. Instead, they are out on the field, in the rural areas, or in the city, visiting the families in the surveillance area. They often know the families very well, and interview all the members of the family and the house.

Once or twice a week they return to CIDS to deliver their reports to the persons working with the quality control, three iron ladies which knows everything worth to know about the families and house holds. They check the collected data against previously collected data, and find and correct the errors.

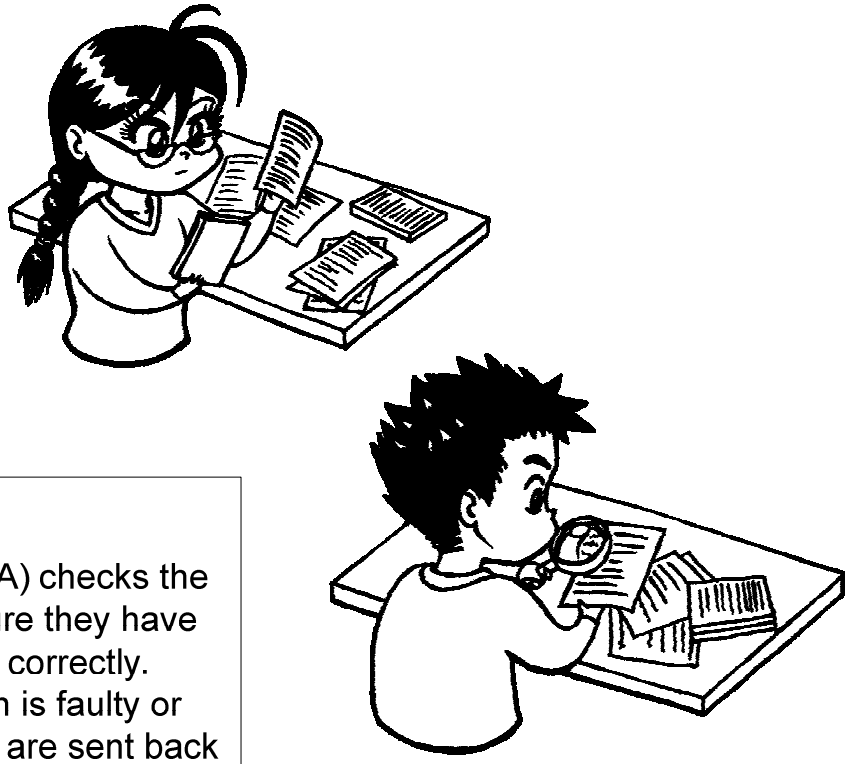
When the quality control has done their work, a small group of women enter the data into the database. They will find any remaining errors and correct them.

The group of database experts fix any problems and support the data enter group.

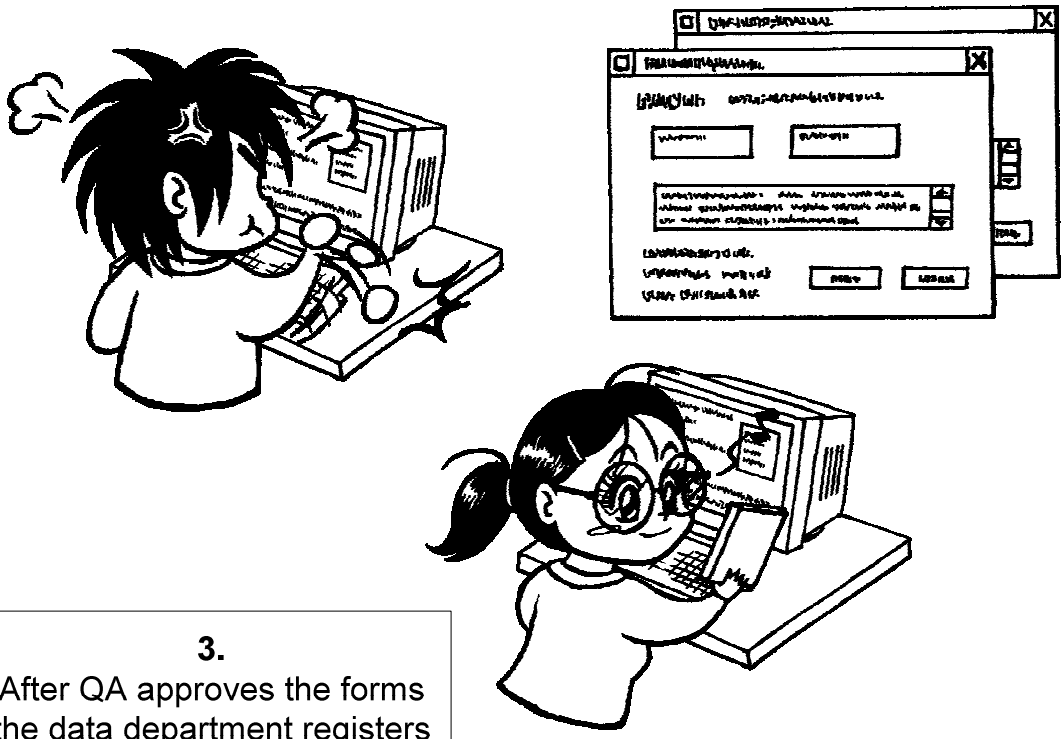
Around all this, experts in different areas work with guest professors and companies buying the statistics. It is common that Swedish professors and researchers come to visit CIDS to assemble the statistics needed for their work.

The pictures below give an overview of the data collection process.

1.
The fieldworkers collect information about the house and its inhabitants. Name, birth date, level of education and so forth, is collected for each member of the household.



2.
Quality control (QA) checks the forms to make sure they have been filled out correctly.
If the information is faulty or missing, the forms are sent back to the fieldworkers who revisit the house to obtain the information.



3.
After QA approves the forms the data department registers the new information.
This is done using the input forms supplied by the system.

4.
Researchers works with people at CIDS to extract the data needed for their research.

3.3 The current database system

The current database is implemented in Microsoft Access, and is used in the everyday work at CIDS. The user interface is created in Access as well.

Nils Assarssons and Tommy Ljunggrens work [5] give a very good overview of the database. A pretty complete overview can be found together with a transfer analysis in the document Database Emigration Specification, [DES].

4 Overview of documents

These chapters give a brief overview of the documents of this work.

4.1 The new database conceptual design

The conceptual design is unquestionably the largest part of the work, and every effort has been made in order to make it clear and easy to understand.

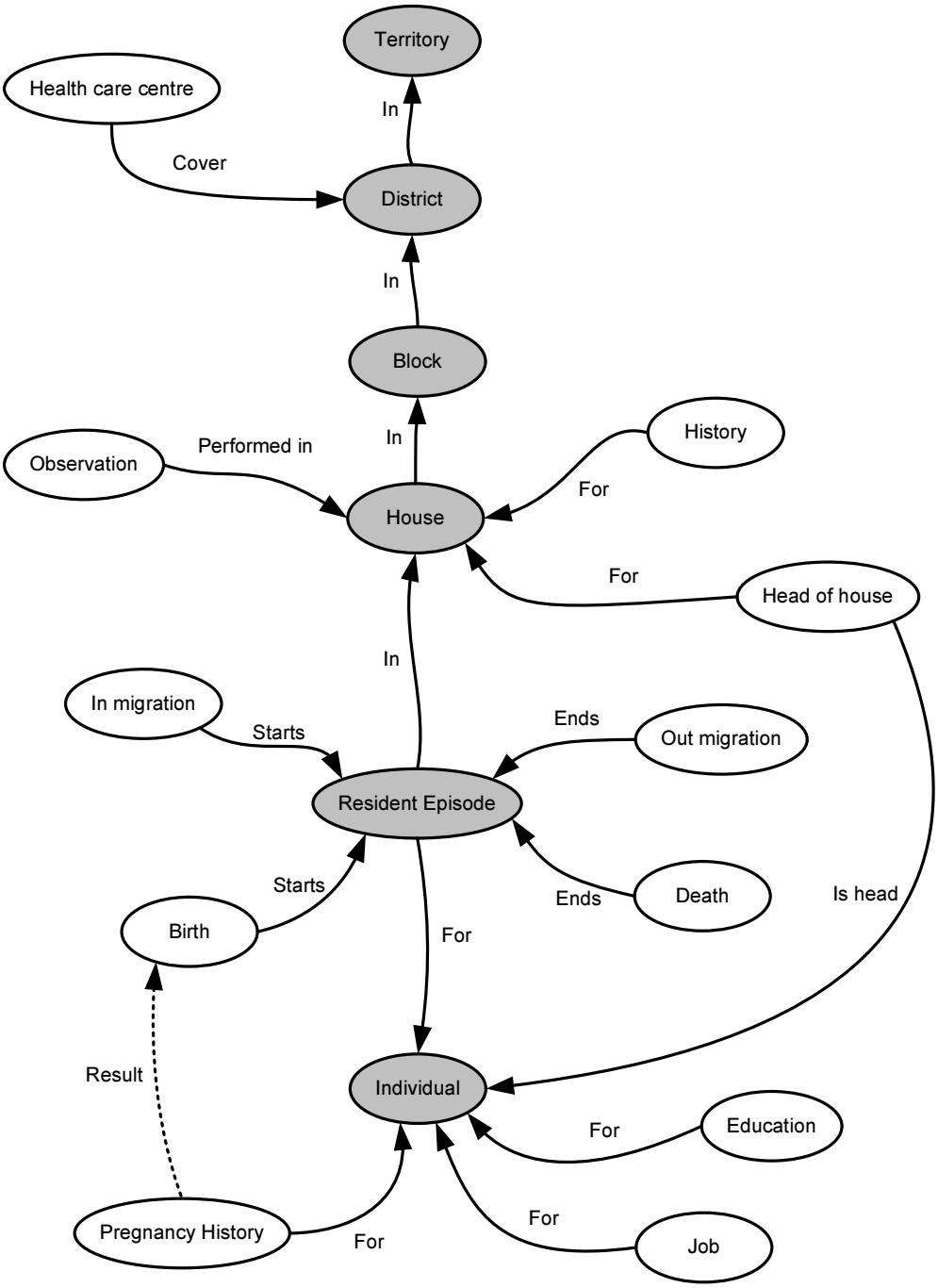


Figure 4.

The conceptual design of the database has gone through several iterations of design and a lot of changes. The core of the design is the episode-concept, which is a common

conceptual design used in many real systems. For a good overview of this concept, please read the document Longitudinal Database Design [1]. We decided to build our design based on these design principals, since we think it is a superior design approach which fits perfectly for databases containing data spanning over time.

The idea is to represent the data in the form of events and episodes. An episode is made from two events, one starting and one ending event. An event is a point in time together with some data about the event. The design is fully documented in the document Database Conceptual Design [DSSDBCD].

4.2 Implementation of the database

It started with a test implementation in MySQL, which was continuously developed during the design work.

After some discussions with the workers at CIDS, we developed a version of the system in Microsoft Access, since they have everyday knowledge of this database system. This way they do not have to learn a new database system, which lower the amount of work when they start to use the new database.

The implementation is not finished because the conceptual design still has some unsolved issues. See chapter ‘Future improvements’. However, most parts are functional and quite well tested.

The actual implementation of the new database is described in detail in Database Implementation Description [DID].

4.3 The transfer process

During the development of the database conceptual design and the actual implementation, I developed a program for transferring the existing data to the database. This helped me to increase my understanding of their design and its problems. In the end this task turned out to be quite large since it requires a full understanding of both databases.

The program is functional yet missing much functionality, but the document describing the transfer process is much more complete.

The transfer process is described and discussed in the document Database Emigration Specification [DES].

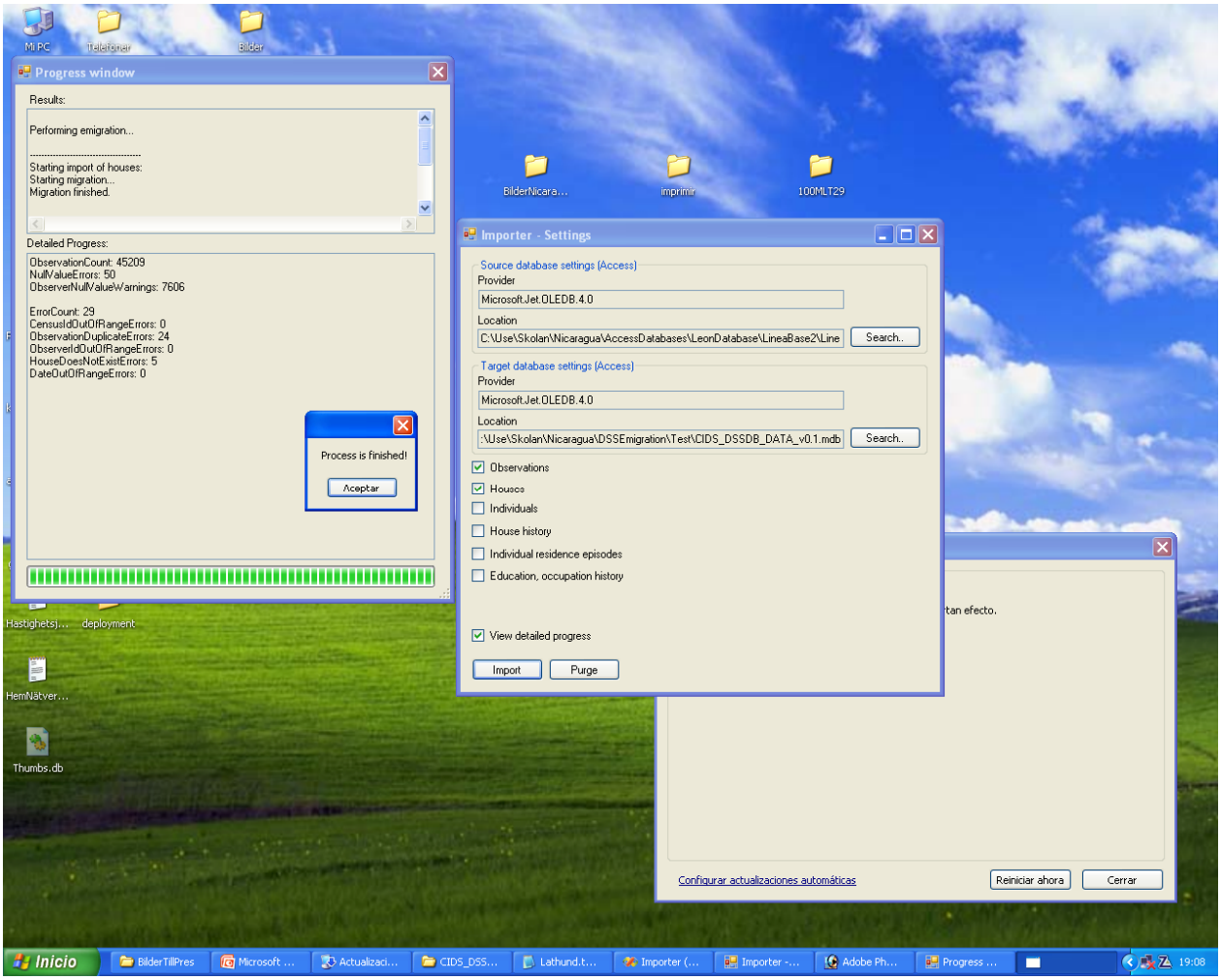


Figure 5. The transfer program in action

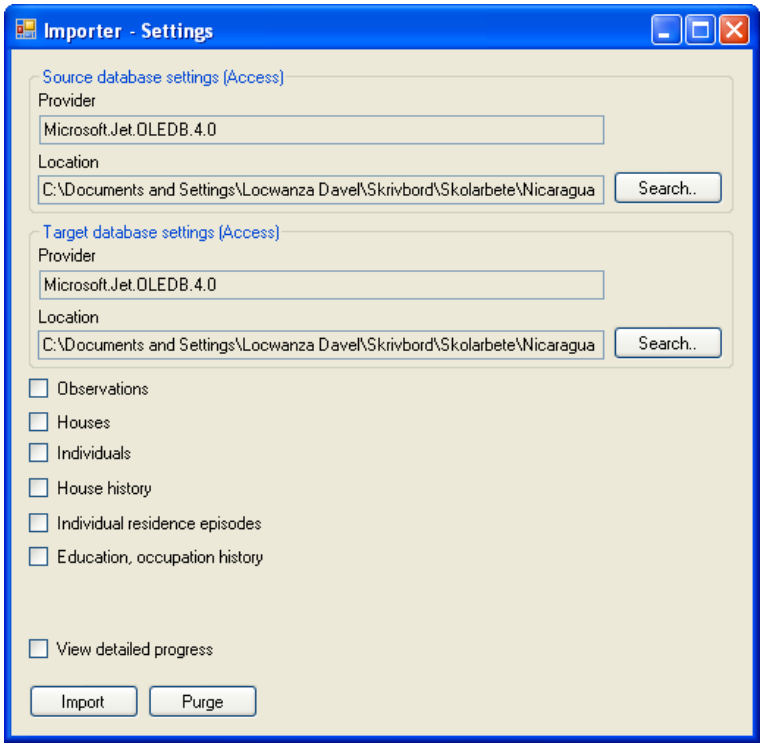


Figure 6. Setup window in the transfer program

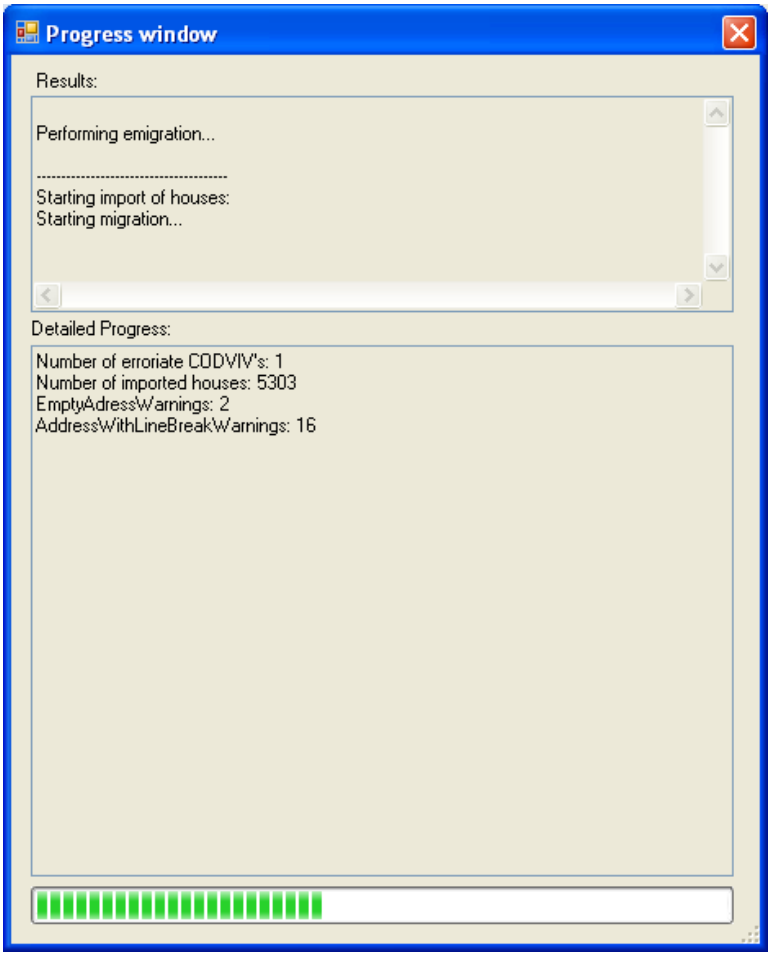


Figure 7. Transfer program in process

4.4 The static values

A lot of values are considered static, i.e. they do not change frequently during the lifetime of the database. These values are all extracted and defined in the document Static Values [SV].

5 Working models

When we started the work, we tried to work according to the waterfall model; first the design, then the implementation and so on. However, this approach probably work best in large companies, and we switched to the spiral model; creating the design, implementing it, and based on the experience of the implementation, modifying the design.

The risk with the spiral model is the risk of not knowing when the work is finished. This also became our greatest challenge, we wanted the design to be 'perfect', so we spent too much time developing and modifying.

We started with a test implementation in MySQL, which we developed during the entire design phase. This approach has proved useful, since a lot of design issues could be solved or avoided by actually see the design in function.

5.1 Obstacles and problems

One of our problems was that we had a hard time to get the facts we needed. There was no document's describing the database; we had to interview the persons working with it everyday, which is a very time consuming yet interesting and funny task.

The language barrier also was a problem since much of the people have limited knowledge of the English language, and our knowledge of the Spanish language was very limited in the beginning. After three months working with the people at CIDS, we felt we had the knowledge and understanding of the problem we needed to create a database covering their needs.

The decision to construct a version of the database in Access delayed the work pretty much, this task was much more than just implement the tables. Several exceptions to the design rules had to be made to make it work. Access lacks much of the functionality of MySql. This switch of database is the reason why some of the documentation might refer to MySql and other parts refer to Access in the database implementation documentation.

5.2 Being Teacher

We had time to hold some classes in programming in the C# language as well. This was a real experience for me, because it were supposed to be held in Spanish and the students had very mixed previous knowledge. It became very 'Spanglish', and instead of the planned one hour classes, it always became two hours. The well-planned classes broke down into walking around and helping, answering questions and solve unexpected problems. Despite all this, it was really funny and I think that everyone including ourselves learned a lot!

6 Current status

During the end of our stay in Nicaragua, we realized we did not have time to finish up the different parts of the project. Instead, an effort was made to document what had been done so far, so if another group of students decides to catch up this work in the future, they will have a better start.

The most important parts of the conceptual design is fairly complete, but there are still areas which can be significantly improved, and some parts are not yet functional.

CIDS can only start using this new database system if the existing data can be transferred in full and without errors. Therefore it is very important to have a detailed description of how to do this. The document Database Emigration Specification [DES] is an extensive description of how to transfer the data, together with the actual emigration application. There is still much work left in this area, but the big part is done.

This examination work is a good piece of work, which could be used as a base for continued work on an actual implementation of a demographic database system for CIDS, and possibly similar projects in Nicaragua or elsewhere.

7 Future improvements

During the work with the database design a number of mistakes has been committed, which was realized late or afterwards. These are things which need a second review before really implementing this database design.

The document Database Emigration Specification [DES] sometimes gives a good discussion about these problematic tables.

7.1 Pregnancy tables design

The design of the pregnancy tables are not finished, and they likely do not fulfil its requirements defined in Nils and Tommy's work [5]. There are several exceptions to the design rules, and sometimes assumptions had to be made about what they needed or how particular functionality is expected to work. All in all this results in a fragile design with some problems attached to that.

7.2 Head of house tables design

The current implementation is unclear and the usefulness discussed. Due to this, little effort has been spent in designing these tables. The importance of this data was realized very late in the project, but there was no time to make a proper design.

7.3 Death cause tables design

These tables have been the hardest to design. There where never an opportunity to talk with any person responsible for handling decisions about the death causes. Unclear information and complicated implementation in the current database made it hard, therefore no finished design exist.

7.4 Summary

Any database design should be based on a detailed requirements specification. Since the design is based on the existing database at CIDS, there might be misinterpretations as well as mistakes.

The proper way might have been to find the documents which describe what the database should and should not contain. There might be some laws, moral valuations, and practical decisions which are not covered in the database design.

When a future database is going to be constructed, this must be taken into consideration. The design need to be compared and updated with the laws and other ethical considerations. Experts at CIDS should be consulted before actually implementing this database design.

8 Conclusion

You can use this research as a background for continued development of a database for collecting, storing and presenting demographic surveillance data. It is especially useful if you intend to transfer the data from the current database to a new database, since much of my efforts have gone into this specific area.

The program for transferring the data from the current database to the new has proved useful for finding errors in the current database. A valuable side-task would be to extend this into a tool for finding errors. This would improve the value of the current database significantly. See the Database Emigration Specification for a full description of the transfer process. [DES]

Since none of this work has come into practical use at CIDS, my hope is that someone else will use this work as a base for their continued work on the demographic surveillance database.

The time in Nicaragua has influenced me very much. During this five months I learned so much about how organizations work, how people in it work, how corporation with other people work. I learned to speak Spanish and I learned about the people of Nicaragua and their culture.

I also learned much about database design, about programming in C#, about statistics and how to read and understand documentation of all kinds.

The lack of documentation taught me how important it is to document a system during it's development, and not after finishing it. Most, if not all software development is a continuous process, with people coming and going. Always require continuous documentation during the development!

I learned tremendously much about how researchers work in the area of health care, schools, hospitals, demography and much more. It is very inspiring to work with people so full of energy and devotion for what they are working with!

I can recommend you to go there and do your examinations work. It is worth all the trouble because it will probably be the experience of your life. If you are open-minded and curious the examination work will be so much more than just a piece of work.

9 People

Lot's of people have been of invaluable help and support during the project, and I try to mention them all here, even though I know I will forget to mention some of them.

Nils Assarsson – He has been my mentor in the beginning, presenting me to the people at CIDS, describing the problem, the work so far, and his own great work on analysis of the use of handheld palm computers for collecting data on the field. He also knows about all good places of León, which he happily introduced to me.

Maria Teresa Orozco – My first contact at CIDS and in Sweden. Already before leaving Sweden she helped me with the understanding of the database system they use today. She has patiently explained the quirky design over and over again, even when she was busy with her work.

Elmer Zelaya – My first contact with CIDS, invaluable source of information, and of course my big thanks for letting me stay at his beautiful hotel, Sueño de Meme.

Francisco Centeno Cardenas is responsible for error checking of the entered data in the database.

Maria Mercedes – My best source of information. She has had much patience with my never ending questions, and without her help this work would not be what it is. Maria Mercedes, Francisca and Aleyda works with the quality control of the collected data before it is entered into the data base. They are also responsible of the training of the interviewers.

Marlon Osmán Melendes and **Darcy** – Working with the GIS system, which keep track of the GPS coordinates for each house which is part of the surveillance. They supplied me with great maps of León and its surroundings.

Wilton José Peréz – Has extensive knowledge of demographic surveillance database systems, is currently studying public health.

Kjerstin Dahlblom – A good friend whom has very much knowledge about CIDS history and the people working there.

Christin Lindholm – My teacher who always have had good points of view and recommendations during this work.

Timothy Lennryd – My brother, for the vivid pictures of the people at CIDS. He can catch the soul of a moment in a picture.

10 Programs used

10.1 MySql

The database used during the most of the development time. It contains all what you can need, and it is free for use.

10.2 MySql Administrator

This program handles the access rights with user accounts, the backup process, and much more.

10.3 MySql Query Browser

Let you create queries easily in an Access like way.

10.4 MySql Workbench

This is the visual tool which gives a good graphical overview of the database design. It also let you design the tables and connections graphically, but it was used mainly for creating the overview pictures.

10.5 Visual Studio

Microsoft Visual Studio for C# is the tool used for developing the application for transferring the data.

10.6 Access

The database system used later in order to make it easier for people at CIDS to start using the database.

10.7 Word

All documentation in this project is made in Microsoft Word.

10.8 Visio

Microsoft Visio is the tool for creating block diagrams and every other type of diagram.

10.9 Powerpoint

Microsoft Powerpoint is used for presentations.

11 Appendix

11.1 This work

This table gives an overview of the documents in this work.

Document	Description
[FR] Final Report	This document.
[DSSDBCD] Database Conceptual Design	The conceptual design of the demographic surveillance system database.
[DID] Database Implementation Specification	An overview of the actual implementation of the database conceptual design.
[DES] Database Emigration Specification	The specification of the transfer process of the data from the CIDS database.
[SV] Static Values	A comprehensive list of all the static values in the database.

11.2 Other documents

This table contain references to documents this work depends on.

Document	Description
[1] Longitudinal Database Design, by Justus Benzler, Sam Clark. 2000	The database design choice depends heavily on this document. It is a short but powerful description of the concept of longitudinal databases.
[2] Sistema de vigilancia en demografía y salud, Leon, Nicaragua – Reporte de línea de base, by Rodolfo Peña, Marlon Meléndez, Wilton Pérez, Carina Källestål. 2005	This is a report assembled from data from the database. It is a very good example of how the data collected is being used.
[3] Demographic and Health Research – a Nicaraguan-Swedish research and training collaboration 2004-2008	This document gives an extensive description of CIDS as an organisation, and the corporation with Swedish Sida.
[4] Demographic & Health Surveillance System, by A.A.M Mobinul Islam. 2002	This is an analysis of the current database design made 2002. There is several change proposals, and some of them have been implemented.
[5] Demographic Surveillance System Database, by Nils Assarsson and Tommy Ljunggren. 2003	Nils and Tommy started this work with identifying the problems CIDS has, presenting what they need, and presenting a solution.
[6] Database Requirement Specification, by Nils Assarsson and Tommy Ljunggren.	The database requirement specification, which part of this work spring from.
[7] Linea de Base	The current database implemented in Microsoft Access, and used in the everyday work at CIDS.
[8] Demographic & Health Surveillance System, by A.A.M Mobinul Islam. 2002	This document is an analysis of the current database, describing several suggestions for improving the database. Some of these are implemented today.

Document	Description
[10] International Classification of Diseases, short ICD.	The format of the code is X00.00, where the X is a letter in the alphabet, the first block of zeroes is the grouping code, and the second block is the specific disease. In Spanish it is called Clasificación Internacional de Enfermedades, short CIE. WHO has the list available at their site, the current address is: http://www.who.int/classifications/icd/en/
[11] Home Alone – Sibling caretakers in León, Nicaragua, by Kjerstin Dahlblom. 2008	This is a Swedish thesis about children working in the home, and the effects of this.

LUND
UNIVERSITY

Centre for Demographic and Health Research

Demographic Surveillance System Database

Conceptual Design

Author

Jon Lennryd

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

Contents

12 Introduction

This document describes the conceptual design of the demographic surveillance system database.

12.1 Purpose

The purpose of this document is to give an understanding of the conceptual design ideas for the system, how it is supposed to work and discussion of question marks in the design that is not yet solved satisfactory.

This document will not delve into every aspect of designing a demographic surveillance system database. Instead it will give a good understanding of the design and way of thinking when planning and implementing an actual database. The document Database Implementation Description, [DID] will give a complete and detailed description of our implementation.

The document Database Emigration Specification [DES] will give a good comparison between the design of CIDS current database and ours, which is also another good way to understand the advantages of our design.

12.2 Intended Audience and Reading Suggestions

Intended audience is people who will implement the database, as well as people working with the database when it is implemented.

13 Overview of design

In this chapter the conceptual design will be described step by step, starting with a brief overview, which will give a grasp of the different entities and their relation to each other. The subsequent chapters will describe each entity in detail.

13.1 Full overview

Starting with a picture, this descriptive overview will give a good understanding of the conceptual design.

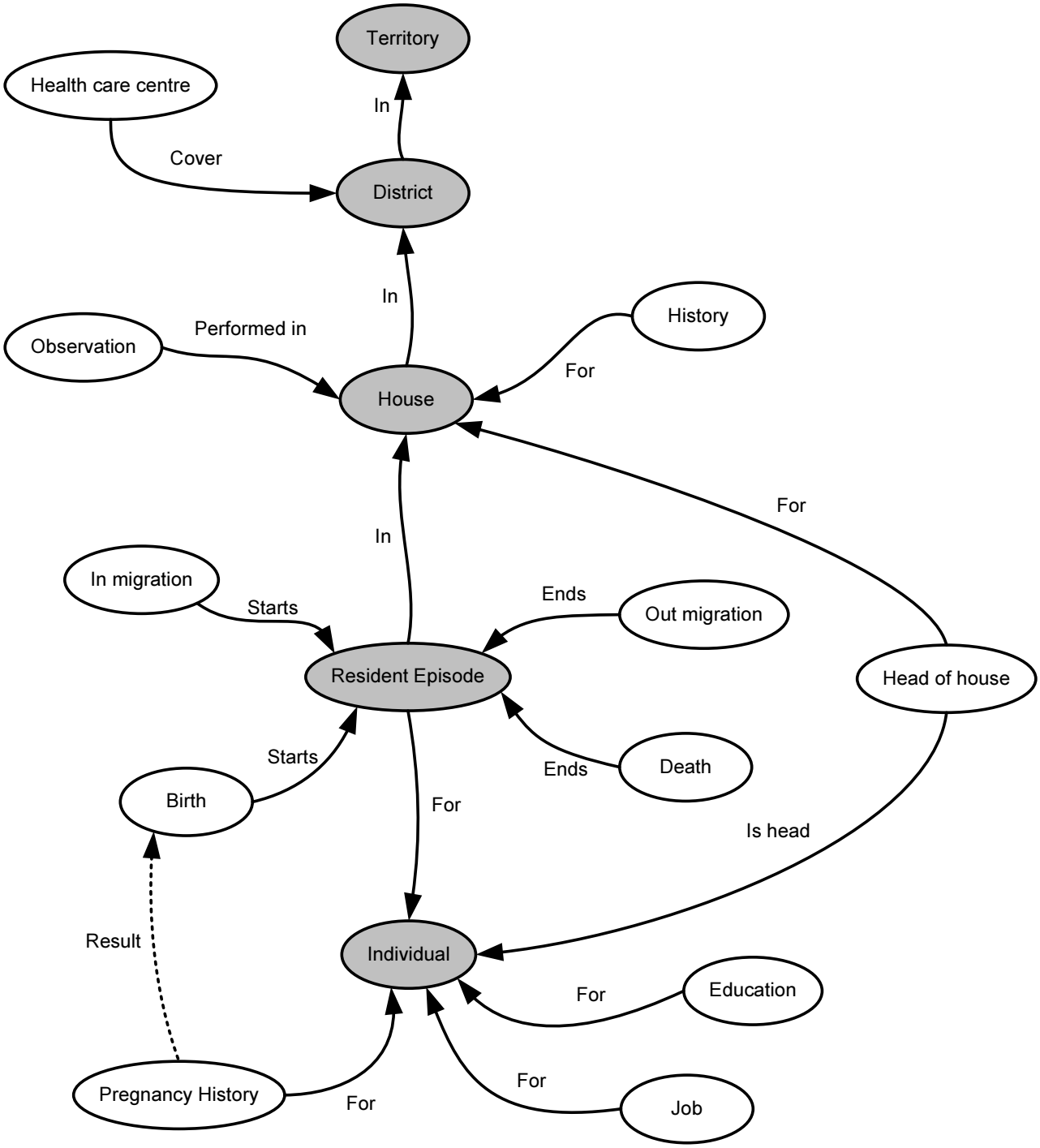


Figure 1

The grey entities in Figure 1 form a line from top to bottom. These are the most important aspects of the design.

Territory and district groups the houses according to their physical location. House describes a single house and its history. Resident episode describes a persons living in a specific house for a certain period. Individual describes a person, her education and job, and if it is a woman, her pregnancy history.

The other (white) entities always points on one of the grey. This indicates that they give extra information to the grey entity. As an example the grey entity Individual has the white entities Education and Job connected to itself. Information about the individual's education is hence stored in the entity Education.

The idea with those entities is to cut out the blocks of information that is repeated, shared between several entities, or maybe is not always available. This design gives a better overview and understanding of the system, as well as unnecessary repetition of data is avoided.

This overview leaves out several details, which is described later in this document, but one thing is worth stating now; The Observation entity contain the date of visit, which is crucial in all history records. Our solution is to have a relation from every history record to the Observation where this was discovered. These relations are left out from the overview in order to concentrate on the basic design principle.

13.2 Area information

A Territory (See figure 2) has a name that is equal to the real name of the area, and a one letter short name. A Territory contains Districts, which has a name, a short name and a zone type. A District contains Blocks, which have a block id.

A District's name is equal to the real name. The short name is exactly two letters, and uniquely identifies the District in a certain Territory. The Zone type describe if the district is a rural or urban area. Note that between two territories the districts short names might be identical, so the short name is not enough to fully identify the District.

A Block only has an id which uniquely identifies it in a District. In the same way as for a district's short name being unique only in its Territory, the Block id is unique only in its District.

Consequently, in order to uniquely identify a certain Block in the entire area, it is necessary to have the following three pieces of data; The id of the Block, The Short name of the District, and the Short name of the Territory.
The typical look for this is formed as a block of characters; MA602, which is M for the Territory named 'Mantica', A6 for the urban District named 'Rpto. Jose B. Escobar', and 02 is the Block id, giving the exact city block.

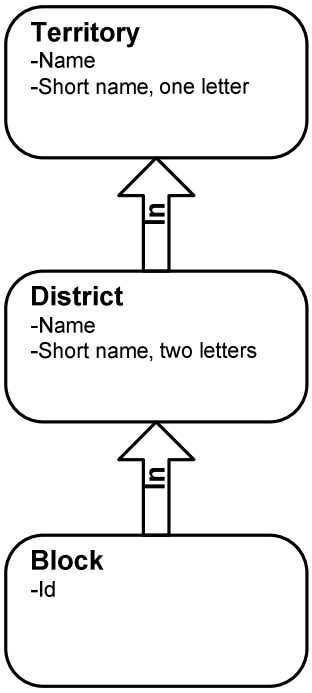


Figure 2

13.3 House

A House resides in a Block, as seen in figure 3, and has an id that uniquely identifies it in the block.

Since a House is referred to directly in so many cases, it has a specific id that uniquely identifies the house in the system. The id is composed of seven characters, the first five is given by the Territory, the District and the Block, and the last two characters is the House id. The typical form of this is; MA602A1, where A1 is the house id.

A House also has a physical address, which aid the interviewers to find it.

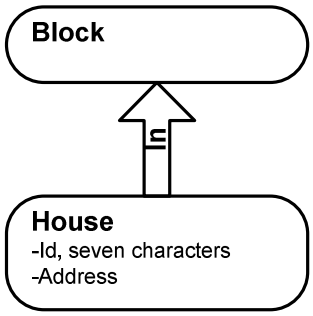


Figure 3

3.13.1 History for a house

There are several types of history for a house, which is recorded every time a change is recognised. (See figure 4)

Along with the date of the visit, the following data is recorded: type of water supply, floor type, wall type, toilet type, number of bedrooms and whether there is electricity or not.

Every type has a set of possible values to choose from, an example from water supply types:

‘Tubería puesto comunal’ – ‘Local communal access’,

‘Río/Quebrada’ – ‘River/Well’,

Etcetera. The full set of accessible values for all types is found in the document Static Values. [SV]

Note that the date of visit is stored in the Observation, which the House History record has a relation to.

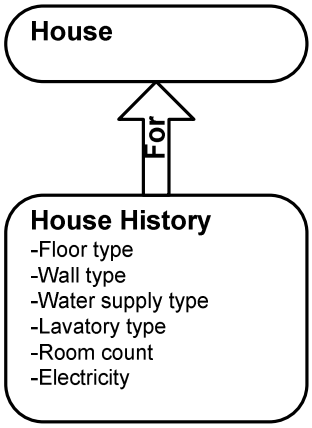


Figure 4

13.4 Individual

As seen in Figure 5, an Individual is a person who lives or has lived inside the surveillance area.

The following data is stored for an individual; Name, Sex, and Date of birth. There is also an Id, constructed of nine characters, which uniquely identifies the individual.

The Id is constructed from the House id where the person is found first, and two characters uniquely identifying the individual in the house. The typical look of this id is MA602A101, where 01 is the individual's id for the house.

If the information is available, and the mother already exists in the database, a reference to the mother is also kept.

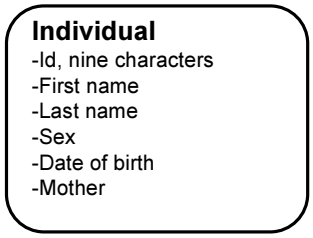


Figure 5

4.13.1 Individual education and occupation history

As seen in figure 6, an individual can have a history for his/hers completed education, and/or current occupation. The history is updated every time a change is recognised.

There are one set of possible values for the education, and one set for the occupation.

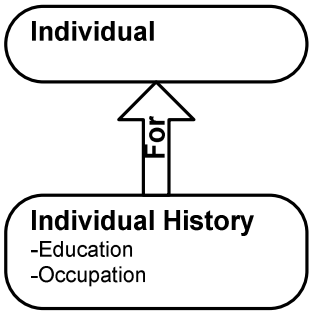


Figure 6

13.5 Residence episode

An episode of residence is a period of time when a certain individual lives in a certain house. (See figure 7)

A residence episode can start in one of three ways; Birth, In-migration or DSS Entry, and end in one of two ways; Out-migration or Death.

Starting date and ending date is also registered.

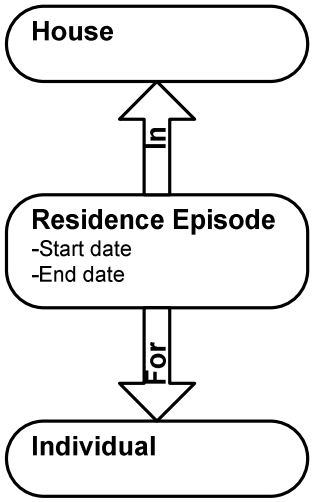


Figure 7

5.13.1 Starting event - Birth

The Birth event starts a residence episode, because a baby is born into the house where the mother lives. (Figure 8)

There is a connection to the birth event.

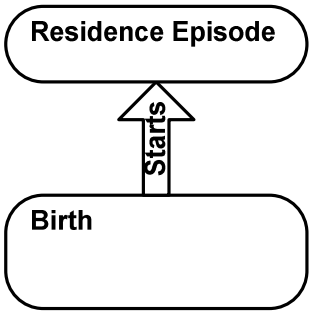


Figure 8

5.13.2 Starting event – In-migration

The In-migration event describes the origin of the person, or where the person comes from, and the reason why he moved. (Figure 9)

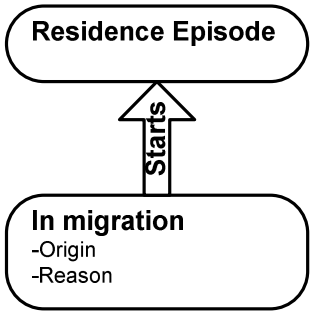


Figure 9

Examples of reasons for moving is;

'Problemas de salud' – 'Health problems',
'Vino a estudiar' – 'Study',
'Unión / Matrimonio' – 'Marriage'

There are a specified set of origins, and a specified set of reasons. If necessary, it is possible to enter a specific textual reason which is not listed in the set of reasons.
The main reason for this is backward comp ability; Thousands of records in the current database are textual, and must be converted first.

5.13.3 Starting event – DSS Entry

The DSS Entry event is only used when the system is started, and when a new area with one or more houses is added to the survey. (Figure 10)

This is a special case which must be considered. If, for example, these people's episode would start with an in-migration event instead of this starting event, the statistics for migration would show a big peak at the date when the new area was added. This would of course be wrong.

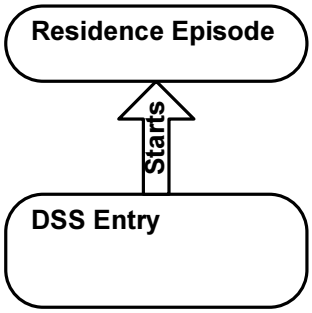


Figure 10

5.13.4 Ending event – Death

As seen in figure 11, the Death event describes the reason, if the person died from injuries and in that case which type of injury, if there is a death certificate and in that case the id to the certificate, if the person died under or shortly after she had a pregnancy.

The death code follows an international standard, called International Classification of Diseases. [10]

The format of the code is X00.00, where the X is a letter in the alphabet, the first block of zeroes is the grouping code, and the second block is the specific disease.

Note: The design work for this is not complete; more information is needed from CIDS in order to fully and correctly implement this.

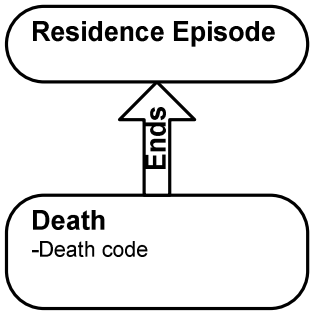


Figure 11

5.13.5 Ending event – Out migration

The Out-migration event describes where the person moves to, and the reason for moving. (Figure 12)

There are a specified set of Out-migration types, a specified set of reasons.
If necessary, there is also the possibility to enter a specific textual reason, not listed in the set of predefined reasons. The main reason for this is backward comp ability; Thousands of records in the current database are textual, and must be converted first.

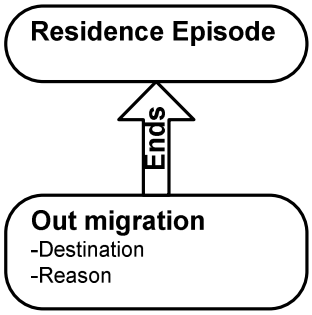


Figure 12

13.6 Pregnancy history episodes

As seen in figure 13, an Individual (Woman) can have a history of pregnancies, which describes each pregnancy. Each pregnancy is an episode with a starting date, and possibly an ending date. A pregnancy can end in one of two ways; Abortion or Birth.

If it ended with an abortion, it can be one of two types; Provoked or Spontaneous.

In the case of a birth, it can have one of two results; Live-born or Still-born. The Place of birth-type is stored, as well as the type of Birth attendant. These are sets of predefined types.
If necessary, both Place of birth and Birth attendant can have a textual description instead of a predefined type.

Furthermore the type of delivery is stored, which is one of two possible values; Caesarean or Vaginal.

There is much more details about the Birth, and this is described in the documents Database Implementation Description, [DID] and Demographic Surveillance System Database. [5]

As always, the document Database Emigration Specification [DES] gives some interesting and important information not noted elsewhere.

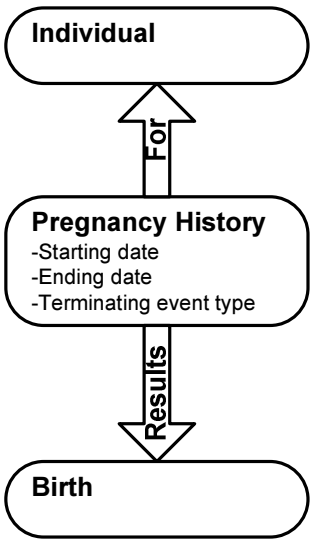


Figure 13

13.7 Health care centre

A health care centre covers one or more Districts, and they can overlap each others area of responsibility. (Figure 14)

It has its full name, a short code, and type of health care centre.

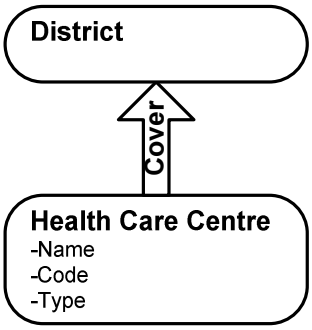


Figure 14

13.8 Head of house episodes

As seen in figure 15, an episode is a period of time when a certain person is head of a certain house.

The other persons living in the house at the same time has a relation to the head of the house. This relation can be a value chosen from a set of predefined types.

This data is used to determine the wealth of the family in the house.

Note: The design work for this is not complete; more information is needed from CIDS in order to fully and correctly implement this.

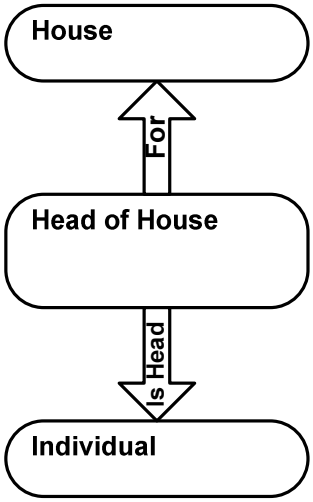


Figure 15

14 Design decisions

This chapter describes the design decisions that where made for the actual implementation of the database, which was made in Access 2000. A preliminary database was implemented in MySQL 5, but decision was taken to use Access. The reason for the transition was to make it easier for the people working at CIDS to start using and understanding the database, since they use Access in the current database, hence their expertise lies in Access. The choice of database engine has small impact on the conceptual design, but it can help tremendously to choose a good one, which have a lot of tools helping you with the design. Our choice was therefore MySQL, since the developers of MySQL also have a set of powerful tools, and a very good and detailed documentation. They also have an extensive discussion group with very good examples, design proposals and much, much more.

When it comes to the conceptual design, our primary source of inspiration has been the document Longitudinal Database Design. [1] This document made us throw away our own design proposal and start from scratch, and should be every database developer's must-read.

We have also strived to use a common set of naming conventions, as well as to define a common set of functionality, which design could be reused in the entire database. The main reason for this have been ease of understanding, our goal have been to create a database which to some extent is self explanatory.

We have also tried to create a modular design with as few interconnections as possible. This gives the possibility to extend the design, as well as to shut down parts of the database without affecting the other parts, which is great for testing and maintaining the database.

In some cases we have been forced to stick with the design of the current system, in order to not break or alter too much of the original functionality. The changes would require new data to be collected or, in some cases, it would mean very much work when transferring the data from the old database to the new. Also, since some design changes are not fully completed in the current database, we simply must import the old design so as to not destroy any data.

14.1 Events

Events happens once, for example the person dies, the baby is born, the person move in to the house, a new floor is built since last time of visit, etcetera.

Events always have a relation to the interview in which it was discovered. The reason for this is that it can be necessary to be able to return to the paper forms, or it might be necessary to see which interviewers performed the interview in order to consult them. Another reason is that it must be possible to sort events discovered in a certain period of time, and that is not possible with only the date of event. For example the event that a baby was born, date of birth almost never is the same as the date of interview.

Not every event has a date of when the event occurred. When a new floor is discovered no date for construction is stored, since it almost certainly is not made in a single day. The opposite is a birth, where the date of birth of course is stored.

An interview is called an Observation in this database design.

14.2 Episodes

An episode is a period in time, with a starting point and an ending point. These points are events, which together forms an episode. One example is a pregnancy, which have a date of

discovery, and one date for when the baby was born. Another example is the period of when a person lives in a specific house. An episode always has a starting date and an ending date. If the terminating date is not specified it means that the episode is currently active, for example the woman is still pregnant.

14.3 History tables

A history table is a list of events, where the newest replaces the older. For example, register a change of the floor in a house, or a person get a new job. The latest event is active until a new event occurs. It is possible to go back in the history, and see for example what type of floor the house had at a certain date.

14.4 Type tables

Type tables contain static values which are used by other tables.

They make it possible to let the database control the values the user enters, as well as presenting the user with a textual set of choices in dropdown menus or similar.

Type tables are used instead of enumerations, due to technical difficulties of adding new values in an enumeration set, and difficulties of extracting number of elements, as well as the elements themselves.

In certain cases the type table is represented by an enumeration instead. The reason for doing this can be that there exist only two or three values, and that it sometimes is not possible to add another value. An example is Male and Female. The reason to have an enumeration is simplified database design and fewer tables.

The name of a type table always ends with _types.

14.5 Other types tables

Other type tables contain free values, usually text or numbers, and always refer to a row in another table.

They always co-exist with a Type table, and give the ability to add a specific type when the set of predefined types does not apply. The main reason is however backwards compatibility with the current database design, where they chose to have free-text values instead of specific types.

They always end with _other_types.

An example from our implementation is the Type table Immigration_reason_types which contains a specific set of reasons for moving into a new house or area. The other type table Immigation_reason_other_types contains specific descriptions linked to the immigration event table. (Residence_immigration_events) The system knows that if the reason for immigration is set to the predefined value Otro (other), the reason can always be found in table Immigration_reason_other_types.

14.6 Naming conventions

These chapters describe the naming conventions which are used in the database.

6.14.1 Singular and plural

The name of a table is always in plural, since they can contain more than one line. Examples are emigration events, pregnancy episodes.

Singular is used when referencing to a single line in a table, for example in which house the new floor was built.

Example: The table House_floor_history has a member house_id, which contain the id of the house where the floor has changed. It also has a member observation_id, which contain the id to the observation where the discovery was made.

As you can see the singular form is used, and the same design idea has been used for both observation_id and house_id. This makes it easy to understand the intent of the designer.

6.14.2 Serial numbers

A serial number is the database's way of uniquely identify a row in a table. In this conceptual design, serial numbers are named id since it is short and descriptive.

The naming of references to other tables follows this scheme: the name of the table in singular + _id.

Examples of this are house_id, house_water_type_id, observation_id.

6.14.3 Description

The primary use for a description is in type tables, and contains the textual description of the value that the id stands for.

Example of this is description, a member of table House_wall_types. Its value is used in drop-down menus to present the user with a pre-defined set of choices.

6.14.4 Name

The name of an entity, in the cases they have a name. Examples are name of the health care centre, territory name and the name of the district.

6.14.5 Code

If an entity has a code of some sort, it is stored here. Examples of this are territories, individuals, health care centre and districts.

A code is not used by the database; it is for the users, usually because the paper forms have a certain structure with certain codes, which need to be mapped in the database. The exception to this rule is table Individuals, which id is actually of type Code.

More specific details for each table are found in the implementation description.

6.14.6 When no naming conventions apply

For these cases where no naming convention applies, a single word explains the meaning of the row. Examples are; address, sex, outcome etc.

6.14.7 Avoiding name collisions

In some cases there is a necessity to distinguish between two members of the same type, for example name. In these cases the name starts with the descriptive word that will distinguish the two. An example is first_name and last_name in table individuals.

6.14.8 History versus Event

There is an exception from the history rule, and these are the event tables, which name ends with _events. This is a question about descriptiveness; a table name ending with _history contains a history, while a table name ending with _events will contain events.

LUND
UNIVERSITY

Centre for Demographic and Health Research

Demographic Surveillance System Database

Database Implementation Specification

Author

Jon Lennryd

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

Contents

15 Introduction

This document describes the implementation of the demographic surveillance system database and how it is realized in MySql.

15.1 Intended Audience and Reading Suggestions

Intended audience is people who will implement the database, as well as people working with the database when it is implemented.

A good idea is to start with the document Database Conceptual Design [DSSDBCD], which will give a good overview of the way of thinking when designing a demographic surveillance database system.

15.2 Definitions, Acronyms, and Abbreviations

Word	Explanation
CHAR	A single byte characters, holding a numerical or an ASCII value.
DATE	A Date field, forcing the user to enter a correct date.
ENUM	A fixed set of textual values, used instead of numbers when comfortable. A good example is the set of days in a week.
INTEGER	A whole number.
VARCHAR	A text field, whose number of characters can vary, up to a given maximum.

Word	Explanation
Ascending	The rows are sorted in ascending order on this column's values.
Auto Increment	Mostly used for id-fields, and its value is automatically incremented each time a new row is inserted.
Indexed	By indexing the data of a given column, you can improve the time it takes for finding data in the table.
Foreign key	A value which identify a specific row in another table. For example, a table containing information about Persons has a reference to a specific row in another table, called Houses. Table Houses contains data about the Persons house. If several Persons live in the same house, they will all reference the same row in

Word	Explanation
	the table Houses.
Not null	The field's value must be defined, i.e. a row cannot be created if not all its values are given at the time of creation.
Primary key	A value that uniquely identifies a row in a table. The table is internally sorted by this value. It is a synonym to these keywords: Not null, Unique, Indexed.
Size (XX)	The field's size in bytes
Unique	The field's value is guaranteed to be unique. (Database or other logic prevent user from entering the same value twice in the table.)
Unsigned	A positive number. Negative numbers can not be stored in this field.

15.3 Implementation overview

The implementation of the database is pretty straightforward, since the conceptual design is very detailed. However, the conceptual design and the implementation goes hand in hand during the entire development, which is the best way to go, at least in my point of view. Each chapter in this document describe the tables that form a distinguishable unit of information. Each sub-chapter describes a single table.

The name of the sub-chapter is also the name of the table it describes. At the start of each sub-chapter, a short description is given. For a full description, see the document Database Conceptual Design. [DSSDBCD]

3.15.1 Table layout

This document has a lot of different tables, and each type has its own layout and purpose.

Fields

A field table defines each column in a database table.

The Fields table has the following rows:

- Name – The name of the column in the table.
- Data type – The data type stored in the column.
- Settings – Extra settings completing the Data type.
- Description – Rules for insertion and deletion of new rows.

Indexes

The indices of a table are the columns in the table which together forms a unique row. Sometimes this is the column Id, since this is a simple, compact and fast way of uniquely

address each row in a column. A good example is any Type table, where the id identifies each and every row.

In some cases the indices is made up of several rows, for example the table Health_care_centre_covers_districts, where no id column exists. It is only two columns in this table, and no single column can uniquely identify a row. Both columns together can uniquely identify a row, so they will be the indices of the table.

The Index table has the following rows:

- Field – The name of the column in the table.
- Settings – Specific settings for the column.

Foreign keys

A foreign key is a value pointing at a specific row in another table.

The Foreign keys table has the following rows:

- Field(s) – The name of the column in the table.
- Foreign table – The name of the foreign table.
- Foreign field(s) – The name of the row in the foreign table.

15.4 Unfinished business

The following table's design is not completely finished, and need some reviewing before actually using the design. See Final Report [FR], the chapter named Future Improvements for more information about this.

4.15.1 Pregnancy tables

The pregnancy tables suffer from a fragile design, due to a lot of exceptions, which must be handled in the code reading the database.

We started out early with the design of the pregnancy tables, before we had all the facts on the table. This resulted in a patchwork, where we tried to make it work with the new requirements.

As it is implemented now, it will work, but it is advisable to develop a better design which avoid these exceptions.

4.15.2 Death tables

The death event tables will probably change much, since we did not have time to research enough about how CIDS want to store the death events, neither what they want to store. This is a feature not much used in CIDS currently used database, so it might be a good idea to take the opportunity to make a good re-design here, and ask CIDS if it would be alright for them to re-enter this data instead of transferring from their database.

4.15.3 Head of house tables

The head of house tables will work, and it is possible to transfer this data from the database CIDS currently use. However, there is some improvements which can be done, and the value of some of the data can be questioned. The best way would be to ask around, and ask exactly what they want to be able to read out of the tables, and create a new design based on this.

4.15.4 Tables in draft status

The following tables are involved in these areas, and need a redesign.

- Residence_death_events
- Deaths_by_injury
- Death_cause_certificates
- Death_cause_no_certificates
- Death_around_pregnancy
- Head_of_house_episodes
- Relationship_to_head_episodes

16 Common tables

The conceptual design strives to minimize the number of different solutions, thereby preserving the simplicity and ease of understanding. The result of this is, among other things, a couple of designs which is repeated through the entire database. This chapter will describe them in detail. The rest of the document will refer back to these chapters.

16.1 Types tables

A Type table consist of a unique id and its textual description. Everywhere the description is referenced, the id is used to identify the description.

This is a one to one mapping, each id correspond to exactly one description. This means that both id and description are Unique.

Fields

Name	Data type	Settings	Description
id	INTEGER	Auto increment, Not null, Unsigned	Primary key.
description	VARCHAR	Size (50), Not null, Unique.	Textual description of the type. Must contain at least one character, and a maximum of 50.

Indices

Fields	Settings
id	Primary key, Ascending

Foreign keys

No foreign keys.

17 Health care centre tables

17.1 Health_care_centre_types

This table contain the type of health care centres.

[See chapter Types tables.]

17.2 Health_care_centres

This table contain all health care centres and its information.

Fields

Name	Data type	Settings	Description
id	INTEGER	Auto increment, Not null, Unsigned	Primary key.
name	VARCHAR	Size(50), Unique	Name of the health care.
health_care_centre_type_id	INTEGER	Not null, Unsigned	Foreign key to the table Health_care_centre_types.
code	CHAR	Size(3), Not null	Three character code identifying the health care centre. The 1st character must be in the range of A to Z. The 2nd character must be in the range of 0 to 9. The 3d character must be in the range of 0 to 9, or 1 to 9 if the 2nd character is 0.

Indices

Fields	Settings
id	Primary key, Unique, Ascending

Foreign keys

Field(s)	Foreign table	Foreign field(s)
health_care_centre_type_id	Health_care_centre_types	id

17.3 Health_care_centre_covers_districts

This table connect the table Health_care_centres to the table Districts.

A district can be covered by only one health care centre, so the district_id is Unique, meanwhile, one health care centre can cover several districts, so health_care_centre_id is not unique.

Fields

Name	Data type	Settings	Description
health_care_centre_id	INTEGER	Not null, Unsigned	Foreign key to the table Health_care_centres.
district_id	INTEGER	Not null, Unsigned, Unique	Foreign key to the table Districts.

Indices

Fields	Settings
health_care_centre_id	Ascending
district_id	

Foreign keys

Field(s)	Foreign table	Foreign field(s)
health_care_centre_ id	Health_care_centres	id
district_id	Districts	id

18 Area tables

18.1 Territories

A territory is a division of a municipal.

Fields

Name	Data type	Settings	Description
id	CHAR	Size (1)	Primary key. Must be in the range of A to Z.
name	VARCHAR	Size(50), Not null, Unique	Name of the territory. Must contain at least one character, and a maximum of 50.

Indices

Fields	Settings
id	Primary key, Ascending

Foreign keys

No foreign keys.

18.2 Districts

A District is a division of a Territory.

Fields

Name	Data type	Settings	Description
id	INTEGER	Unsigned, Auto increment.	Primary key.
name	VARCHAR	Size (50), Not null, Not unique.	Name of the district. Must contain at least one character, and a maximum of 50.
code	CHAR	Size (2), Not null, Unique.	The 2 character code identifying the district. The 1st character must be in the range of A to Z. The 2nd character must be in the range of 1 to 9.

Name	Data type	Settings	Description
zone_type	ENUM	Values (Rural, Urban), Not null.	Describing the type of the district.
territory_id	INTEGER	Unsigned, Not null.	Foreign key to table Territories.

Indices

Fields	Settings
id	Primary key, Ascending

Foreign keys

Field(s)	Foreign table	Foreign field(s)
territory_id	Territories	id

19 Observation tables

19.1 Censuses

A census is also called an interview cycle, which is a period in time where all houses in the surveillance area is visited, and all persons is interviewed. Start and end date is Unique since no two interview cycles can overlap.

Fields

Name	Data type	Settings	Description
id	INTEGER	Unsigned, Auto increment.	Primary key.
start_date	DATE	Not null, Unique.	Start date of the Census. There can be only one open census, which must be the latest. Can not be less than 1993-01-01. Can not be bigger than the date of today. Can not be less than the column end_date of the previous row.
end_date	DATE	Can be null, Unique.	End date of the Census. Can not be less than the rows start_date.

Indices

Fields	Settings
id	Primary key, Ascending

Foreign keys

No foreign keys.

19.2 Observers

This table contain the persons performing the censuses. (Interviews)

Fields

Name	Data type	Settings	Description
id	INTEGER	Unsigned, Auto increment.	Primary key.

Name	Data type	Settings	Description
first_name	VARCHAR	Size (50), Not null, Not unique.	First name of the observer. Must contain at least one character, and a maximum of 50.
last_name	VARCHAR	Size (50), Not null, Not unique.	Last name of the observer. Must contain at least one character, and a maximum of 50.

Indices

Fields	Settings
id	Primary key, Ascending

Foreign keys

No foreign keys.

19.3 Observations

This table contains data about the observations made during the interviews.

Fields

Name	Data type	Settings	Description
id	INTEGER	Unsigned, Auto increment.	Primary key.
house_id	CHAR	Size (7), Not null.	Foreign key to table Houses.
census_id	INTEGER	Unsigned, Not null.	Foreign key to table Censuses.
observer_id	INTEGER	Unsigned.	Foreign key to table Observers. Allows null values since current database has a lot of null values here. (Around 7500) However, new input should not allow null values.

Name	Data type	Settings	Description
date	DATE	Not null.	Must not be less than the start_date of the Census in which it belong. Must not be larger than the end_date of the Census in which it belong. If the Census end_date is null, the value can not be larger than the date of today.

Indices

Fields	Settings
id	Primary key, Ascending

Foreign keys

Field(s)	Foreign table	Foreign field(s)
house_id	Houses	id
census_id	Censuses	id
observer_id	Observers	id

20 House information tables

20.1 Houses

This table contain the houses; each house is positioned in a specific District.

Fields

Name	Data type	Settings	Description
id	CHAR	Size (7)	Primary key. Have to be 7 characters. Format: The 1st character must be in the range A to Z. The 2nd character must be in the range A to Z. The 3d character must be in the range 1 to 9. The 4th character must be in the range 0 to 9. The 5th character must be in the range 0 to 9, or 1 to 9 if the 4th character is 0. The 6th character must be in the range A to Z. The 7th character must be in the range 1 to 9.
house_code	CHAR	Size (2), Not null.	Number that uniquely identifies the house in the block. The 1st character must be in the range A to Z. The 2nd character must be in the range 1 to 9.

Name	Data type	Settings	Description
block_code	CHAR	Size (2), Not null.	The 2 character code identifying the block in which the house exist. The 1st character must be in the range 0 to 9. The 2nd character must be in the range of 0 to 9, or 1 to 9 if the 1st character is 0.
district_id	INTEGER	Unsigned, Not null.	Foreign key to table Districts.
address	VARCHAR	Size (200).	The house address, simply a textual description.

Indices

Fields	Settings
id	Primary key, Ascending

Foreign keys

Field(s)	Foreign table	Foreign field(s)
district_id	Districts	id

20.2 House_water_types

Define the types of water supplies that a house could have.

[See chapter Types tables.]

20.3 House_floor_types

Define the types of floors that a house could have.

[See chapter Types tables.]

20.4 House_wall_types

Define the type of walls that a house could have.

[See chapter Types tables.]

20.5 House_lavatory_types

Define the type of lavatories that a house could have.

[See chapter Types tables.]

20.6 House_bedrooms_history

This table describes the history of the bedroom's number in a house.

Fields

Name	Data type	Settings	Description
house_id	CHAR	Size (7), Not null.	Foreign key to table Houses.
observation_id	INTEGER	Unsigned, Not null.	Foreign key to table Observations.
bedrooms_count	INTEGER	Unsigned, Not null.	Number of bedrooms. Minimum value is 1, maximum is 50. A new value must differ from the previous for the same house.

Indices

Fields	Settings
house_id	Ascending

Foreign keys

Field(s)	Foreign table	Foreign field(s)
house_id	Houses	id
observation_id	Observations	id

20.7 House_electricity_history

This table describes the history of a house's electricity access.

Fields

Name	Data type	Settings	Description
house_id	CHAR	Size (7), Not null.	Foreign key to table Houses.
observation_id	INTEGER	Unsigned, Not null.	Foreign key to table Observations.
electricity_exists	ENUM	Values (Yes, No, NoInformation), Not null.	A new value must differ from the previous for the same house.

Indices

Fields	Settings

Fields	Settings
house_id	Ascending

Foreign keys

Field(s)	Foreign table	Foreign field(s)
house_id	Houses	id
observation_id	Observations	id

20.8 House_water_history

This table describes the history of a house's water supplies.

Fields

Name	Data type	Settings	Description
house_id	CHAR	Size (7), Not null.	Foreign key to table Houses.
observation_id	INTEGER	Unsigned, Not null.	Foreign key to table Observations.
house_water_type_id	INTEGER	Unsigned, Not null.	Foreign key to table House_water_types. A new value must differ from the previous for the same house.

Indices

Fields	Settings
house_id	Ascending

Foreign keys

Field(s)	Foreign table	Foreign field(s)
house_id	Houses	id
observation_id	Observations	id
house_water_type_id	House_water_types	id

20.9 House_floor_history

This table describes the history of a house's type of floors.

Fields

Name	Data type	Settings	Description

Name	Data type	Settings	Description
house_id	CHAR	Size (7), Not null.	Foreign key to table Houses.
observation_id	INTEGER	Unsigned, Not null.	Foreign key to table Observations.
house_floor_type_id	INTEGER	Unsigned, Not null.	Foreign key to table House_floor_types. A new value must differ from the previous for the same house.

Indices

Fields	Settings
house_id	Ascending

Foreign keys

Field(s)	Foreign table	Foreign field(s)
house_id	Houses	id
observation_id	Observations	id
house_floor_type_id	House_floor_types	id

20.10 House_wall_history

This table describes the history of a house's type of walls.

Fields

Name	Data type	Settings	Description
house_id	CHAR	Size (7), Not null.	Foreign key to table Houses.
observation_id	INTEGER	Unsigned, Not null.	Foreign key to table Observations.
house_wall_type_id	INTEGER	Unsigned, Not null.	Foreign key to table House_wall_types. A new value must differ from the previous for the same house.

Indices

Fields	Settings
house_id	Ascending

Foreign keys

Field(s)	Foreign table	Foreign field(s)
house_id	Houses	id
observation_id	Observations	id
house_wall_type_id	House_wall_types	id

20.11 House_lavatory_history

This table describes the history of a house's type of lavatories.

Fields

Name	Data type	Settings	Description
house_id	CHAR	Size (7), Not null.	Foreign key to table Houses.
observation_id	INTEGER	Unsigned, Not null.	Foreign key to table Observations.
house_lavatory_type_id	INTEGER	Unsigned, Not null.	Foreign key to table House_lavatory_types. A new value must differ from the previous for the same house.

Indices

Fields	Settings
house_id	Ascending

Foreign keys

Field(s)	Foreign table	Foreign field(s)
house_id	Houses	id
observation_id	Observations	id
house_lavatory_type_id	House_lavatory_types	id

21 Individuals tables

21.1 Individuals

This table contains the Individuals and their general information.

Fields

Name	Data type	Settings	Description
id	CHAR	Size (9).	Primary key. Have to be 9 characters. Format: The first 7 have the same rules as the id for Houses. The 8th character must be in the range 0 to 9. The 9th character must be in the range 0 to 9, or 1 to 9 if the 8th character is 0.
first_name	VARCHAR	Size (50), Not null, Not unique.	First name of the inhabitant. Must contain at least one character, and a maximum of 50.
last_name	VARCHAR	Size (50), Not null, Not unique.	Last name of the inhabitant. Must contain at least one character, and a maximum of 50.
sex	ENUM	Values (Male, Female), Not null.	Sex of the inhabitant.

Name	Data type	Settings	Description
date_of_birth	DATE	Not null.	The inhabitant's date of birth. Must be larger than 1880-01-01. Must be smaller than today's date. If relation to mother exists, this date must be bigger than her date of birth + 15 years. [according to an interview with Maria Mercedes]
mother_id	CHAR	Size (9), Can be null.	Foreign key to table Individuals. This is a relation to the mother if she lives or has lived in the test area, and can be identified and found in the database.

Indices

Fields	Settings
id	Primary key, Ascending

Foreign keys

Field(s)	Foreign table	Foreign field(s)
mother_id	Inhabitants	id

21.2 Individual_education_types

Define the education types that an individual could have.

[See chapter Types tables.]

21.3 Individual_occupation_types

Define the occupation types that an individual could have.

[See chapter Types tables.]

21.4 Individual_education_history

This table describes the history of an individual's education. The individual must be above five years age. (Source: Maria Teresa)

Fields

Name	Data type	Settings	Description
individual_id	CHAR	Size (9), Not null.	Foreign key to table Individuals.
observation_id	INTEGER	Unsigned, Not null.	Foreign key to table Observations.
individual_education_type_id	INTEGER	Unsigned, Not null.	Foreign key to table Individual_education_types. A new value must differ from the previous for the same house.

Indices

Fields	Settings
individual_id	Ascending

Foreign keys

Field(s)	Foreign table	Foreign field(s)
individual_id	Individuals	id
observation_id	Observations	id
individual_education_type_id	Individual_education_types	id

21.5 Individual_occupation_history

This table describes the history of an individual's occupation. The individual must be above five years age. (Source: Maria Teresa)

Fields

Name	Data type	Settings	Description
individual_id	CHAR	Size (9), Not null.	Foreign key to table Individuals.
observation_id	INTEGER	Unsigned, Not null.	Foreign key to table Observations.
individual_occupation_type_id	INTEGER	Unsigned, Not null.	Foreign key to table Individual_occupation_types. A new value must differ from the previous for the same house.

Indices

Fields	Settings
individual_id	Ascending

Foreign keys

Field(s)	Foreign table	Foreign field(s)
individual_id	Individuals	id
observation_id	Observations	id
individual_occupation_type_id	Individual_occupation_types	id

22 Resident episode tables

22.1 Individual_residence_episodes

This table describe an individual's episode of residence in a house.

Fields

Name	Data type	Settings	Description
id	INTEGER	Unsigned, Auto increment.	Primary key.
individual_id	CHAR	Size (9), Not null.	Foreign key to table Individuals.
house_id	CHAR	Size (9), Not null.	Foreign key to table Houses.
start_date	DATE	Not null.	Start date of the residence episode. Must not be less than 1993-01-01. (Actually the date of initialization of the old system.)
initiating_event_type	ENUM	Values (DSS Entry, Immigration, Birth), Not null.	Can not register a new initiating residence period if the old is not closed.
start_observation_id	INTEGER	Unsigned, Not null.	Foreign key to table Observations.
end_date	DATE	Can be null.	The ending date of an episode. Can be null if terminating_event_type and end_observation_id are null as well, otherwise all three values must be entered. Cannot be smaller than start_date. Cannot be bigger than today's date.

Name	Data type	Settings	Description
terminating_event_type	ENUM	Values (Emigration, Death), Can be null.	Can be null if end_date and end_observation_id are null as well, otherwise all three values must be entered. Must follow after an initiating event, thereby closing an episode.
end_observation_id	INTEGER	Unsigned, Can be null.	Foreign key to table Observations. Can be null if end_date and terminating_event_type are null as well, otherwise all three values must be entered.

Indices

Fields	Settings
id	Primary key. Ascending

Foreign keys

Field(s)	Foreign table	Foreign field(s)
individual_id	Individuals	id
house_id	Houses	id
start_observation_id	Observations	id
end_observation_id	Observations	id

22.2 Residence_immigration_events

This table contains the information about when and why an individual move into a house.

Fields

Name	Data type	Settings	Description
id	INTEGER	Unsigned, Auto increment.	Primary key.
individual_residence_episode_id	INTEGER	Unsigned, Not null.	Foreign key to table Individual_residence_episodes.

Name	Data type	Settings	Description
immigration_origin_type_id	INTEGER	Unsigned, Not null.	Foreign key to table Immigration_origin_types.
immigration_reason_type_id	INTEGER	Unsigned, Not null.	Foreign key to table Immigration_reason_types.

Indices

Fields	Settings
id	Primary key. Ascending

Foreign keys

Field(s)	Foreign table	Foreign field(s)
individual_residence_episode_id	Individual_residence_episodes	id
immigration_origin_type_id	Immigration_origin_types	id
immigration_reason_type_id	Immigration_reason_types	id

22.3 Immigration_origin_types

Define the types of immigration origins that an individual could have.

[See chapter Types tables.]

22.4 Immigration_reason_types

It is a predefined set of reasons for immigration that an individual could have.

When the row immigration_reason_type_id in table Residence_immigration_events equals ‘Otros’, the table Immigration_reason_other_types will contain the alternative textual description.

[See chapter Types tables.]

22.5 Immigration_reason_other_types

When the row immigration_reason_type_id in table Residence_immigration_events equals ‘Otros’, this table must have a row giving the alternative description.

This table exist since the current database has a mixed set of values, and they have done some attempts to transform this into a predefined set of values. However, this transformation is not fully performed, and thus we need to import these values as well, in order not to loose them. See document Database Emigration Specification [DES] for more details about this.

Fields

Name	Data type	Settings	Description

Name	Data type	Settings	Description
immigration_event_id	INTEGER	Unsigned, Not null, Unique.	Foreign key to table Residence_immigration_events. Unique since you can only have one reason per immigration event.
description	VARCHAR	Size (50), Not null, Not unique.	Textual description of the type. Must contain at least one character, and a maximum of 50. Not unique since it is possible to have the same reason for several immigration events.

Indices

Fields	Settings
immigration_event_id	Indexed, Ascending

Foreign keys

Field(s)	Foreign table	Foreign field(s)
immigration_event_id	Residence_immigration_events	id

22.6 Residence_birth_events

When an individual is born in a house in the surveillance area, the baby's residence episode will start with its birth.

Fields

Name	Data type	Settings	Description
id	INTEGER	Unsigned, Auto increment.	Primary key.
individual_residence_episode_id	INTEGER	Unsigned, Not null.	Foreign key to table Residence_immigration_events.

Indices

Fields	Settings
id	Indexed, Ascending

Foreign keys

Field(s)	Foreign table	Foreign field(s)
individual_residence_episode_id	Residence_immigration_events	id

22.7 Residence_emigration_events

When an individual emigrates, move out of a house, his or hers residence episode will be closed with this event.

Fields

Name	Data type	Settings	Description
id	INTEGER	Unsigned, Auto increment.	Primary key.
individual_residence_episode_id	INTEGER	Unsigned, Not null.	Foreign key to table Individual_residence_episodes.
emigration_destination_type_id	INTEGER	Unsigned, Not null.	Foreign key to table Emigration_destination_types.
emigration_reason_type_id	INTEGER	Unsigned, Not null.	Foreign key to table Emigration_reason_types.

Indices

Fields	Settings
id	Primary key. Ascending

Foreign keys

Field(s)	Foreign table	Foreign field(s)
individual_residence_episode_id	Individual_residence_episodes	id
emigration_destination_type_id	Emigration_destination_types	id
emigration_reason_type_id	Emigration_reason_types	id

22.8 Emigration_destination_types

A set of emigration destination types an individual could give upon emigration.

[See chapter Types tables.]

22.9 Emigration_reason_types

Predefined set of reasons for emigration an individual can state.

When the row emigration_destination_type_id in the table Residence_emigration_events equals ´Otros´, the table Emigration_reason_other_types will contain an alternative textual description.

[See chapter Types tables.]

22.10 Emigration_reason_other_types

When the row emigration_destination_type_id in the table Residence_emigration_events equals ´Otros´, this table must have a row giving the alternative description.

Fields

Name	Data type	Settings	Description
emigration_event_id	INTEGER	Unsigned, Not null, Unique.	Foreign key to table Residence_emigration_events. Unique since you can only have one reason per emigration event.
description	VARCHAR	Size (50), Not null, Not unique.	Textual description of the type. Must contain at least one character, and a maximum of 50. Not unique since it is possible to have the same reason for several emigration events.

Indices

Fields	Settings
emigration_event_id	Indexed, Ascending

Foreign keys

Field(s)	Foreign table	Foreign field(s)
emigration_event_id	Residence_emigration_events	id

22.11 Residence_death_events

If an individual dies, his or hers residence episode will end with a death event.

Note that the design of the death tables and its support tables is not complete; it will need a review as well as more research on what data is important to store. These tables are only a draft. The reason for this is unclear information and documentation, and we never had the opportunity to meet any expert in the field. Also, no real implementation exists in the

current database at CIDS. There is an experimental implementation which is not very complete.

Instead most of this information is stored in paper documents. We had no time to examine these properly. It is also sensitive information, which needs a proper investigation if it is alright to store it in the database, and if so, decide about security levels for reading, adding and editing the data.

Fields

Name	Data type	Settings	Description
id	INTEGER	Unsigned, Auto increment.	Primary key.
individual_residence_episode_id	INTEGER	Unsigned, Not null.	Foreign key to table Individual_residence_episodes.
died_from_injuries	ENUM	Values (Yes, No), Not null.	If 'Yes', additional information exist in table Deaths_by_injury
death_certificate_exists	ENUM	Values (Yes, No), Not null.	If 'Yes', additional information exist in table Death_cause_certificates, if 'No', additional information exist in table Death_cause_no_certificates.
----NOTE			death_code should be covered here.

Indices

Fields	Settings
id	Primary key. Ascending

Foreign keys

Field(s)	Foreign table	Foreign field(s)
individual_residence_episode_id	Individual_residence_episodes	id

22.12 Deaths_by_injury

If row died_from_injuries in table Residence_death_events state 'Yes', this table must have a row with more specific details.

Fields

Name	Data type	Settings	Description

Name	Data type	Settings	Description
id	INTEGER	Unsigned, Auto increment.	Primary key.
residence_death_event_id	INTEGER	Unsigned, Not null, Unique.	Foreign key to table Residence_death_events. Unique because only one injury can be the cause of death. [according to interview with Maria Mercedes]
injury_type_id	INTEGER	Unsigned, Not null.	Foreign key to table Injury_types.

Indices

Fields	Settings
id	Primary key. Ascending

Foreign keys

Field(s)	Foreign table	Foreign field(s)
residence_death_event_id	Residence_death_events	id
injury_type_id	Injury_types	id

22.13 Injury_types

This table contain a predefined set of types of injuries an individual could die from.

When the row injury_type_id in table Deaths_by_injury equals ´Otro´, the table Injury_other_types will contain the alternative textual description.

[See chapter Types tables.]

22.14 Injury_other_types

When the row injury_type_id in table Deaths_by_injury equals ´Otro´, this table must have a row giving the alternative description.

Fields

Name	Data type	Settings	Description
death_by_injury_id	INTEGER	Unsigned, Not null, Unique.	Foreign key to table Deaths_by_injury. Unique since you can only have one injury per death cause.

Name	Data type	Settings	Description
description	VARCHAR	Size (50), Not null, Not unique.	Textual description of the injury. Must contain at least one character, and a maximum of 50. Not unique since it is possible to have the same injury for several death events.

Indices

Fields	Settings
death_by_injury_id	Indexed, Ascending

Foreign keys

Field(s)	Foreign table	Foreign field(s)
death_by_injury_id	Deaths_by_injury	id

22.15 Injury_instrument_types

This table give a textual description of the instrument which was the reason for the death of the inhabitant.

-Should be other name-ending than _types, since this is not a type table.

-Should probably consist of two tables in conjunction, like the tables Injury_types and Injury_other_types.

Fields

Name	Data type	Settings	Description
death_by_injury_id	INTEGER	Unsigned, Not null.	Foreign key to table Deaths_by_injury.
description	VARCHAR	Size (50), Not null, Not unique.	Textual description of the instrument. Must contain at least one character, and a maximum of 50. Not unique since it is possible to have the same instrument description for several death events.

Indices

Fields	Settings
death_by_injury_id	Indexed, Ascending

Foreign keys

Field(s)	Foreign table	Foreign field(s)
death_by_injury_id	Deaths_by_injury	id

22.16 Death_cause_certificates

Contains a detailed description of death causes, and if it's available a doctor's signature.

Fields

Name	Data type	Settings	Description
id	INTEGER	Unsigned, Auto increment.	Primary key.
residence_death_event_id	INTEGER	Unsigned, Not null, Unique.	Foreign key to table Residence_death_events. Unique since you can only have one certificate per death event.
certificate_issuer_type_id	INTEGER	Unsigned, Not null.	Foreign key to table Certificate_issuer_types.
direct_cause	VARCHAR	Size (100), Not null.	Textual description of the direct cause. Must contain at least one character, and a maximum of 50.
intermediate_cause	VARCHAR	Size (100), Not null.	Textual description of the intermediate cause. Must contain at least one character, and a maximum of 50.
basic_cause	VARCHAR	Size (100), Not null.	Textual description of the basic cause. Must contain at least one character, and a maximum of 50.
direct_cause_code	CHAR	Size(4), May be null,	Have to be 4 characters, or null.
intermediate_cause_code	CHAR	Size(4), May be null,	Have to be 4 characters, or null.

Name	Data type	Settings	Description
basic_cause_code	CHAR	Size(4), May be null,	Have to be 4 characters, or null.

Indices

Fields	Settings
id	Primary key, Ascending

Foreign keys

Field(s)	Foreign table	Foreign field(s)
residence_death_event_id	Residence_death_events	id

22.17 Certificate_issuer_types

This table contain the set of certificate issuer types.

When the row certificate_issuer_type_id in table Death_cause_certificates equals ´Otro´, the table Certificate_issuer_other_types must contain a textual description.

[See chapter Types tables.]

22.18 Certificate_issuer_other_types

When the row certificate_issuer_type_id in table Death_cause_certificates equals ´Otro´, this table must contain the alternative description.

Fields

Name	Data type	Settings	Description
death_cause_certificate_id	INTEGER	Unsigned, Not null, Unique.	Foreign key to table Death_cause_certificates. Unique since you can only have one reason per death cause certificate.
description	VARCHAR	Size (50), Not null, Not unique.	Textual description of the type. Must contain at least one character, and a maximum of 50. Not unique since it is possible to have the same reason for several death cause certificates.

Indices

Fields	Settings
death_cause_certificate_id	Indexed, Ascending

Foreign keys

Field(s)	Foreign table	Foreign field(s)
death_cause_certificate_id	Death_cause_certificates	id

22.19 Death_cause_no_certificates

When no certificate exists, for example when no doctor has examined the body, this table contain the description, usually given by the relatives, friends or neighbours.

Fields

Name	Data type	Settings	Description
residence_death_event_id	INTEGER	Unsigned, Not null, Unique.	Foreign key to table Residence_death_events. Unique since you can only have one reason per death event.
description	VARCHAR	Size (200), Not null, Not unique.	Textual description of the reason for death. Must contain at least one character, and a maximum of 200. Not unique since it is possible to have the same reason for several death events.

Indices

Fields	Settings
residence_death_event_id	Indexed, Ascending

Foreign keys

Field(s)	Foreign table	Foreign field(s)
residence_death_event_id	Residence_death_events	id

22.20 Death_around_pregnancy

When a woman dies in connection to a pregnancy, additional information is stored in this table.

Fields

Name	Data type	Settings	Description
residence_death_event_id	INTEGER	Unsigned, Not null, Unique.	Foreign key to table Residence_death_events. Unique since you can only have one note per death event. Can only be recorded here if it is a woman that died, in the age between 15 and 49 years. [according to an interview with Maria Mercedes]

Indices

Fields	Settings
residence_death_event_id	Indexed, Ascending

Foreign keys

Field(s)	Foreign table	Foreign field(s)
residence_death_event_id	Residence_death_events	id

23 Relation to head tables

23.1 Head_of_house_episodes

A house and its household always have someone who is the head of the house. This table store these episodes.

Fields

Name	Data type	Settings	Description
id	INTEGER	Unsigned, Auto increment.	Primary key.
individual_id	CHAR	Size (9), Not null.	Foreign key to table Individuals.
house_id	CHAR	Size (9), Not null.	Foreign key to table Houses.
start_observation_id	INTEGER	Unsigned, Not null.	Foreign key to table Observations.
end_observation_id	INTEGER	Unsigned, Can be null.	Foreign key to table Observations. Must differ from start_observation_id. Must not link to an observation which date is before start_observation_id.

Indices

Fields	Settings
id	Primary key. Ascending

Foreign keys

Field(s)	Foreign table	Foreign field(s)
individual_id	Individuals	id
house_id	Houses	id
start_observation_id	Observations	id
end_observation_id	Observations	id

23.2 Relationship_to_head_episodes

Every inhabitant in a house has a relation to the head of the house. These episodes are stored in this table.

Fields

Name	Data type	Settings	Description
id	INTEGER	Unsigned, Auto increment.	Primary key.
individual_id	CHAR	Size (9), Not null.	Foreign key to table Individuals.
house_id	CHAR	Size (9), Not null.	Foreign key to table Houses.
relationship_to_head_type_id	INTEGER	Unsigned, Not null.	Foreign key to table Relationship_to_head_types.
start_observation_id	INTEGER	Unsigned, Not null.	Foreign key to table Observations.
end_observation_id	INTEGER	Unsigned, Can be null.	Foreign key to table Observations. Must differ from start_observation_id. Must not link to an observation which date is before start_observation_id.

Indices

Fields	Settings
id	Primary key. Ascending

Foreign keys

Field(s)	Foreign table	Foreign field(s)
individual_id	Individuals	id
house_id	Houses	id
relationship_to_head_type_id	Relationship_to_head_types	id
start_observation_id	Observations	id
end_observation_id	Observations	id

23.3 Relationship_to_head_types

This table define the set of types of relation ships to head of household. Every individual has a relation.

[See chapter Types tables.]

24 Pregnancy information tables

24.1 Individual_pregnancy_episodes

This table contains all women's pregnancy episodes.

Unsolved design issues

- How to store data about a gravidity which has happened outside the surveillance area. The problem is start_observation_id and end_observation_id, which will be misleading since only one observation was made. Proposed solution is to set either start_observation_id or end_observation_id to null, but this would break the design rules. (All references should be valid; otherwise null value checking has to be done before using the value.)

Fields

Name	Data type	Settings	Description
id	INTEGER	Unsigned, Auto increment.	Primary key.
individual_id	CHAR	Size (9), Not null.	Foreign key to table Individuals.
start_observation_id	INTEGER	Unsigned, Can be null.	Foreign key to table Observations. If null, end_observation_id must not be null.
end_observation_id	INTEGER	Unsigned, Can be null.	Foreign key to table Observations. Must differ from start_observation_id. If null, start_observation_id must not be null. Must not link to an observation which date is before start_observation_id.
end_date	DATE	Can be null	End date of pregnancy. Must not be bigger than today's date. Must not be smaller than the mother's birth date +15 years. Can be null since the gravidity can be ongoing.

Name	Data type	Settings	Description
terminating_event_type	ENUM	Unsigned, Can be null, enumeration(Birth,Abortion)	This value must be set when recording the end_date.

Indices

Fields	Settings
id	Primary key. Ascending

Foreign keys

Field(s)	Foreign table	Foreign field(s)
individual_id	Individuals	id
start_observation_id	Observations	id
end_observation_id	Observations	id

24.2 Pregnancy_abortion_events

If gravidity ends in an abortion, the details about the abortion are stored in this table.

Fields

Name	Data type	Settings	Description
id	INTEGER	Unsigned, Auto increment.	Primary key.
individual_pregnancy_episode_id	INTEGER	Unsigned, Unique, Not null.	Foreign key to table Individual_pregnancy_episode.
abortion_type	ENUM	Not null, enumeration(Provoked, Spontaneous)	Type of abortion.

Indices

Fields	Settings
id	Primary key. Ascending

Foreign keys

Field(s)	Foreign table	Foreign field(s)
individual_pregnancy_episode_id	Individual_pregnancy_episode	id

24.3 Pregnancy_birth_events

If gravidity ends in a birth, the details are stored in this table. Note that several birth events can be connected to one pregnancy episode, in the case of twins.

Fields

Name	Data type	Settings	Description
id	INTEGER	Unsigned, Auto increment.	Primary key.
individual_pregnancy_episode_id	INTEGER	Unsigned, Unique, Not null.	Foreign key to table Individual_pregnancy_episode.
outcome	ENUM	Not null, enumeration(Live born, Still born)	Born alive or born dead.
sex	ENUM	Not null, enumeration(Male, Female)	

Indices

Fields	Settings
id	Primary key. Ascending

Foreign keys

Field(s)	Foreign table	Foreign field(s)
individual_pregnancy_episode_id	Individual_pregnancy_episode	id

24.4 Pregnancy_births_information

This table connect a birth event with birth information. This connector-table is necessary since twins share the same birth information.

Fields

Name	Data type	Settings	Description
pregnancy_birth_event_id	INTEGER	Unsigned, Unique, Not null.	Foreign key to table Pregnancy_birth_events. Unique since a birth can have only one reference to birth information id.
birth_information_id	INTEGER	Unsigned, Not null.	Foreign key to table Births_information. Not unique since several births can reference the same birth information id.

Indices

None.

Foreign keys

Field(s)	Foreign table	Foreign field(s)
pregnancy_birth_event_id	Pregnancy_birth_events	id
birth_information_id	Births_information	id

24.5 Births_information

This table keep information about a birth. The reason it is not connected to table Pregnancy_birth_events directly is that a child can be born outside the surveillance area, thus this information is not collected.

Fields

Name	Data type	Settings	Description
id	INTEGER	Unsigned, Auto increment.	Primary key.
delivery_place_type_id	INTEGER	Unsigned, Unique, Not null.	Foreign key to table Delivery_place_types.
birth_attendant_type_id	INTEGER	Unsigned, Unique, Not null.	Foreign key to table Birth_attendant_types.

Name	Data type	Settings	Description
delivery_type	ENUM	Not null, enumeration(Caesaren,Vaginal)	Whether the child(s) are born normal, or caesarean.

Indices

Fields	Settings
id	Primary key. Ascending

Foreign keys

Field(s)	Foreign table	Foreign field(s)
individual_pregnancy_episode_id	Individual_pregnancy_episode	id
delivery_place_type_id	Delivery_place_types	id
birth_attendant_type_id	Birth_attendant_types	id

24.6 Delivery_place_types

This table define the set of delivery place types.

[See chapter Types tables.]

24.7 Delivery_place_other_types

When the row delivery_place_type_id in table Births_information equals ´Otro´, this table must contain the alternative description.

Fields

Name	Data type	Settings	Description
birth_information_id	INTEGER	Unsigned, Not null, Unique.	Foreign key to table Births_information. Unique since you can only have one delivery place type per birth.
description	VARCHAR	Size (50), Not null, Not unique.	Textual description of the type. Must contain at least one character, and a maximum of 50. Not unique since it is possible to have the same description for several births.

Indices

Fields	Settings
birth_information_id	Indexed, Ascending

Foreign keys

Field(s)	Foreign table	Foreign field(s)
birth_information_id	Births_information	id

24.8 Birth_attendant_types

This table define the set of birth attendant types.

[See chapter Types tables.]

24.9 Birth_attendant_other_types

When the row birth_attendant_type_id in table Births_information equals ´Otro´, this table must contain the alternative description.

Fields

Name	Data type	Settings	Description
birth_information_id	INTEGER	Unsigned, Not null, Unique.	Foreign key to table Births_information. Unique since you can only have one birth attendant type per birth.
description	VARCHAR	Size (50), Not null, Not unique.	Textual description of the type. Must contain at least one character, and a maximum of 50. Not unique since it is possible to have the same birth attendant type for several births.

Indices

Fields	Settings
birth_information_id	Indexed, Ascending

Foreign keys

Field(s)	Foreign table	Foreign field(s)

Field(s)	Foreign table	Foreign field(s)
birth_information_id	Births_information	id

24.10 External_individuals_status

When a child is born outside the surveillance area, the information about the child is stored here.

Fields

Name	Data type	Settings	Description
id	INTEGER	Unsigned, Auto increment.	Primary key.
pregnancy_birth_event_id	INTEGER	Unsigned, Not null, Unique.	Foreign key to table Pregnancy_birth_events.
status	ENUM	Unsigned, enumeration(Alive, Dead, Unknown)	Whether the child is alive or deceased.

Indices

Fields	Settings
id	Primary key. Ascending

Foreign keys

Field(s)	Foreign table	Foreign field(s)
pregnancy_birth_event_id	Pregnancy_birth_events	id

24.11 External_individuals_deceased

In the case that a child is known to be dead, and is outside the surveillance area, information about the death cause and date is stored here.

Fields

Name	Data type	Settings	Description
external_individual_status_id	INTEGER	Unsigned, Not null, Unique.	Foreign key to table External_individual_status.
date_of_death	DATE	Not null, Not unique.	Must not be less than the child's date of birth. Must not be bigger than today's date.

Indices

None.

Foreign keys

Field(s)	Foreign table	Foreign field(s)
external_individual_status_id	External_individual_status	id

LUND
UNIVERSITY

Centre for Demographic and Health Research

Demographic Surveillance System Database

Emigration of Database Specification

Author

Jon Lennryd

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

Contents

25 Introduction

This document describes the process of transferring the data from the Access database LineaBase, into the DSSDB database.

Parts of this process is implemented and working in a transfer program, called Importer. This program has been the experimental base from which this document emerged, and is a fairly complete application designed for easy extending its functionality.

25.1 Purpose

The purpose of this document is to describe in semi code how every table in the DSSDB database is populated with data from the current Access database, Linea de base.

25.2 Definitions, Acronyms, and Abbreviations

Interview and Observation are used interchangeably throughout the document, but observation is the preferred.

25.3 Intended Audience and Reading Suggestions

Intended audience are people who wants to use or further develop the emigration program that where developed for transferring the data from the old Access database into the new database.

When reading this document, you should have an understanding of the LineaBase database, as well as the target database, the DSSDB database. The target database is described in detail in the document [Conceptual design of DSSDB database].

It is also recommended to check the actual implementation of the emigration program and its version history. Also read the document [Database implementation description], which describes this database's actual implementation.

When it comes to the current database, the LineaBase, this document will not give a very good understanding of the design. Instead read the documents referenced in Final Report, numbered [2], [3], [4] and [5].

25.4 Overview of this document

A good way of reading the document is to have an overview of the tables in front of you, for example the document Database Implementation Description. [DID]

Also have the old Access database open, so you can check the tables and their design. This is especially important if you don't understand Spanish.

If you are a programmer, look at the existing emigration program which presents our model of doing what this document describes. Run the program, and look at the process, or look at the results. It is always easier to grasp the idea if you can see it in action!

Every chapter describes how to make the emigration between the databases for a certain aspect of the database; health care centres, areas, houses, inhabitants, pregnancy information, and interview data.

It is described how to emigrate data into every single table in the new database. It is often necessary to collect the data from more than one table in the old Access database.

Depending on the complexity of the task of how to accomplish the emigration for a certain table, we have chosen different ways of presenting the task. The simplest example is when the source table only contains a few limited values. Then the emigration is performed simply by entering the values by hand in the target database.

A general presentation is the numbered list, which describes the task in a step by step model. If it is very simple, for example just copying the values between one table's columns into the other table's columns, we chose to have a table with the target table to the left, and the source table to the left.

For example:

Column name	**Source table & column**
id	EntFechVisit.CODLBCICLO.
start_date	EntFechVisit.FECHA. Should be one day bigger than the preceding census end_date.
end_date	Should be one day lesser than the following census start_date.

As you can see above, the column id in the target database take its value from the column CODLBCICLO inside the table EntFechVisit.

EntFechVisit.CODLBCICLO is a shorthand way of writing 'the column CODLBCICLO in table EntFechVisit.'

Also note the comment made on the second row to the right. This is a rule that checks for errors in the source database. If the date is not one day bigger than the preceding census end date, an error is raised, and some error handling occurs.

Sometimes the emigration procedure is further described with an example, in an attempt to remove any uncertainties still lingering.

Large parts of the transfer process have not been realized in the transfer application, and is only described here. These parts are experimental only, and should be seen as a guideline for continued work. It is noted in each chapter whether or not it has been implemented and tested or not.

4.25.1 Error controls, Conversion rules and Dependencies

It is almost always necessary to have error controls in the emigration process, which catch and removes errors, or in some cases correct them. There are a number of common error controls, as well as a set of specific error controls that is performed for almost every table.

Conversion rules are necessary in a number of cases, for example to convert the date into the preferred format used in the target database. Other conversions are made, for example to transform to upper case in a couple of cases, as well as to remove illegal characters in the first and last name.

Dependencies often exist between tables in the target database, for example an interview is performed in a house, so there exists a relation between the interview table and the house table. This makes things a bit harder, because the emigration has to be performed in a certain order, in this particular example the houses need to be emigrated before the interviews, so the interviews can create the relations to the houses.

26 Health care centre tables

This chapter describes how to import data into the tables that forms the health care centres information.

26.1 Health_care_centre_types

See document Static Values [SV], the chapter with the same name.

26.2 Health_care_centres

Only 32 centres, imported by hand from table EntUnidadS.

See document Static Values [SV], the chapter with the same name for the actual values.

26.3 Health_care_centre_covers_districts

Keeps track of which hospital a certain district has, and is imported by hand from the tables EntUnidadS and EntComun. See document Static Values [SV], the chapter with the same name.

27 Area tables

This chapter describes how to import data into the tables that describes territories and districts.

27.1 Territories

There are three territories, see document Static Values [SV], the chapter with the same name.

27.2 Districts

There are 74 districts, see document Static Values [SV], the chapter with the same name. An outdated source can be found in table EntBarrio.

28 Observation tables

This chapter describes how to import data about observers, observations and observation cycles, called censuses.

28.1 Censuses

Censuses are a part of table EntFechVisit.

Imported by hand, in the following manner:

1. Sort out only the rows containing the value 0 in CODLBCICLO.
2. Sort by FECHA, oldest dates first.
3. The first real date in the row FECHA is the start date of the Census 0.
4. The last real date in the row FECHA is the end date of the Census 0.
5. Repeat for every value in column CODLBCICLO. (Currently 0, 1, 2, 3)

This is preferably done inside Access.

Dependencies

This table is not depending on other tables.

Column name	Source table & column
id	EntFechVisit.CODLBCICLO.
start_date	EntFechVisit.FECHA. Should be one day bigger than the preceding census end_date.
end_date	Should be one day lesser than the following census start_date.

28.2 Observers

This table is imported by hand from table ListEntrevistadora, which contain the name of the interviewers.

Dependencies

This table is not depending on other tables.

Column name	Source table & column
id	ListEntrevistadora.No
first_name	ListEntrevistadora.Entrevistadora, first name.
last_name	ListEntrevistadora. Entrevistadora, last name(s).

28.3 Observations

This table is imported from table EntFechVisit.

Error controls:

- The cycle number must exist in table Censuses.
- The date must be between 1993-01-01 and today's date.
- The date must not be the same for two observations in the same house.
- The observer id must exist in table Observers.
- The house id must exist in table Houses.

Conversion rules made in the import process:

- The date is converted to this format: yyyy-mm-dd.

Table EntFechVisit contains the data about the observations.

Dependencies

Dependant on table Houses, Observers and Censuses.

Column name	Source table & column
date_of_observation	EntFechVisit.FECHA
observer_id	EntFechVisit.ENTREV
house_id	EntFechVisit.CODVIV is compared to Houses.id. Then we have found the house, and the value Houses.id will be stored in Observations.house_id.
census_id	EntFechVisit.CODLBCICLO
id	Auto generated

29 House tables

This chapter describes how to import data into the tables that describes the houses and its history.

29.1 Houses

Error controls:

- Remove any occurrences of the character ‘. This is necessary since it is interpreted as a terminating character in some programming languages, which would create erroneous behaviour.
- The string EntiVivie.CODVIV must be exactly 7 characters.
- After extracting territory code from EntVivie.CODVIV, this code must exist in table Territories.
- After extracting district code from EntVivie.CODVIV, this code must exist in table Districts.

Conversion rules made in the import process:

- EntVivie.CODVIV is transformed to upper case, since some occasions of lower case exist.

The string EntVivie.CODVIV is divided into the following parts:

- The first letter in the string is the first letter in the name of the territory.
- The second and third letter from the left is the districts code. This is constructed of one letter followed by one digit. For example ‘A6’.
- The next two letters, letter number 4 and 5, are the block code, the code of the manzana. (A manzana is a block of houses.) This is two digits, for example 01.
- The last two letters are the code of the house. It is constructed of one letter followed by one digit. For example ‘A1’.

Table EntVivie contains information about all houses, and is mapped quite straightforward.

Column name	Source table & column
house_code	The last two letters in EntVivie.CODVIV, f ex. MA602A1->A1.
block_code	The fourth and fifth letter in EntVivie.CODVIV, f ex. MA602A1->02.
district_id	This is a relation to table Districts, so Districts.id is stored here. Where Districts.code is equal to the second and third letter in EntVivie.CODVIV, we also find Districts.id, which will be stored in Houses.district_id. Example: 'MA602A1' -> 'A6'.
address	EntVivie.DIRECCION. This value is sometimes null.
id	EntVivie.CODVIV, transformed to uppercase.

1.29.1 An example for importing one house

Table EntVivie in Access contains these two values that we are interested of:

- EntVivie.CODVIV = "MA602A1"
- EntVivie.DIRECCION = "Address description"

CODVIV is made of the following parts:

- 'M' is the code for territory.
- 'A6' is the code for the district.
- '02' are the code for the manzana, which is the block code.
- 'A1' is the house code.

Importing:

- Get id for territory "M" by comparing with Territories.code. Keep the Territories.id in memory.
- Get id for district "A6" by comparing with Districts.code, and compare the Territories.id from above with the Districts.territory_id.
  Keep the Districts.id in memory.

- Now, store the Districts.id from above in Houses.district_id.
- Store the block code “02” in Houses.block_code.
- Store the house code “A1” in Houses.house_code.
- Store the address from DIRECCION into Houses.address.
- Store the CODVIV into Houses.id.

29.2 House_water_types

See document Static Values [SV], the chapter with the same name.

29.3 House_floor_types

See document Static Values [SV], the chapter with the same name.

29.4 House_wall_types

See document Static Values [SV], the chapter with the same name.

29.5 House_lavatory_types

See document Static Values [SV], the chapter with the same name.

30 House history

The history of a house is divided into six different values, which is collected at the time of the interview of the members of the household.

These values are; number of bedrooms, whether or not they have electricity, the type of floor, the type of walls, the type of lavatory and the type of water supply. All these values are weighted together when deciding about the wealth of a house.

In the current database design these values are recorded at each visit, even if the value has not changed since the last visit. In our design, only the changes are recorded.

The values observation_id and house_id has the same function in each table.

30.1 General import procedure

Since all house history tables share the same design, they can have the same transfer procedure when transferring from the current database to the new. They also share the same source table, EntEstViv.

The only tricky thing is the big difference between the two databases, requiring some book keeping in order to keep everything right.

First, get all the records for a specific house. These records are collected from table EntEstViv, where every record is represented as one row. Sort the records so that the oldest comes first. This sorting is based on the date of interview, EntFechVisit.FECHA, not EntEstViv.FECHA.

Second, store all the data from the oldest record. (Fetch the columns NCUART, LUZ, PISO, EXCRETA, PARED and AGUA from table EntEstViv.) Each type of data should be stored in its own table, with a similar name.

After these initial steps all the remaining, now sorted, records, are scanned for changes. Only if there are changes in the value of interest, there will be a new record in the corresponding table. This will remove the unnecessary repetition of records.

An example of importing remaining records of the floor history:

Compare the current row's PISO value with the newest entered value of House_bedroom_history.bedroom_count. If these values differ, the value of PISO is stored in table House_bedroom_history as a new event.

Repeat this method for every value in the current EntEstViv row.

30.2 House_bedrooms_history

Error controls:

- Null values are not permitted; the row must be discarded as erroneous.
- The date of interview must be between 1993-01-01 and today's date.
- The observation id must exist.
- The observation event must point to the same house as this history event does.
- The number of bedrooms must not exceed 50, or drop below 0. This is only a sanity check; the high number 50 could be adjusted to a realistic number.

Conversion rules made in the import process:

- None.

Column name	Source table & column
bedroom_count	EntEstViv.NCUART
observation_id	EntEstViv.IDVISITA points to an observation, which contains date of interview; EntFechVisit.FECHA. This date, together with EntEstViv.CODVIV which points to a house, is enough to identify the interview in the new database: SELECT id FROM Observations WHERE date_of_observation = [the date] AND house_id = [the house];
house_id	Stored temporary in memory in order to be able to search for all records in table EntEstViv about that house.

30.3 House_electricity_history

Contain the history of changes of the houses electricity status.

Error controls made during the transfer:

- Null values are not permitted; the row must be discarded as erroneous.
- The date of interview must be between 1993-01-01 and today's date.
- The observation id must exist.
- The observation event must point to the same house as this history event does.
- Possible values of EntEstViv.LUZ;
    - 1 mean no electricity, and is converted to No,
    - 2 mean they have electricity, and is converted to Yes,
    - 0 or 99 mean the information is not collected at the time of interview. This can only be true if the time of interview is before year 2003; all interviews after 2003 must ask about electricity.
    - All other values should be interpreted as an error.

Conversion rules made in the import process:

- 2 is set to Yes, 1 to No.

Column name	Source table & column
has_eletricity	EntEstViv.LUZ is converted to Yes or No.
observation_id	Same as for table House_bedrooms_history.
house_id	Same as for table House_bedrooms_history.

30.4 House_water_history

Contain the history of changes of the houses water supply type.

Error controls made during the transfer:

- Null values are not permitted; the row must be discarded as erroneous.
- The date of interview must be between 1993-01-01 and today's date.
- The observation id must exist.
- The observation event must point to the same house as this history event does.
- The value EntEstViv.AGUA must be between 1 and the number of rows in table House_water_types, i.e. it must refer to a row in table House_water_types.

Conversion rules made in the import process:

- None.

Column name	Source table & column
house_water_type_id	EntEstViv.AGUA
observation_id	Same as for table House_bedrooms_history.
house_id	Same as for table House_bedrooms_history.

30.5 House_floor_history

Contain the history of changes of the houses floor type.

Error controls made during the transfer:

- Null values are not permitted; the row must be discarded as erroneous.
- The date of interview must be between 1993-01-01 and today's date.
- The observation id must exist.
- The observation event must point to the same house as this history event does.

- The value EntEstViv.PISO must be between 1 and the number of rows in table House_floor_types, i.e. it must refer to a row in table House_floor_types.

Conversion rules made in the import process:

- None.

Column name	Source table & column
house_floor_type_id	EntEstViv.PISO
observation_id	Same as for table House_bedrooms_history.
house_id	Same as for table House_bedrooms_history.

30.6 House_wall_history

Contain the history of changes of the houses wall type.

Error controls made during the transfer:

- Null values are not permitted; the row must be discarded as erroneous.
- The date of interview must be between 1993-01-01 and today's date.
- The observation id must exist.
- The observation event must point to the same house as this history event does.
- The value EntEstViv.PARED must be between 1 and the number of rows in table House_wall_types, i.e. it must refer to a row in table House_wall_types.

Conversion rules made in the import process:

- None.

Column name	Source table & column
house_wall_type_id	EntEstViv.PARED
observation_id	Same as for table House_bedrooms_history.
house_id	Same as for table House_bedrooms_history.

30.7 House_lavatory_history

Contain the history of changes of the houses lavatory type.

Error controls made during the transfer:

- Null values are not permitted; the row must be discarded as erroneous.
- The date of interview must be between 1993-01-01 and today's date.
- The observation id must exist.
- The observation event must point to the same house as this history event does.
- The value EntEstViv.EXCRETA must be between 1 and the number of rows in table House_lavatory_types, i.e. it must refer to a row in table House_lavatory_types.

Conversion rules made in the import process:

- None.

Column name	Source table & column
house_lavatory_type_id	EntEstViv. EXCRETA
observation_id	Same as for table House_bedrooms_history.
house_id	Same as for table House_bedrooms_history.

31 Individuals information tables

This chapter describes how to import data into the tables that forms the individual information, including occupation and education.

31.1 Individuals

Importing this is straightforward. For every row in table EntHabit, create a row in table Individuals, and copy the values.

The only exception is the mother relation, which must forego some transformation.

Error controls:

- Null values are not permitted, except EntHabit.CODMADRE.
- The string EntHabit.ID must be of length 9.
- EntHabit.ID must be constructed exactly as a house code is constructed, inclusive a serial number added to the end.
- EntHabit.ID must be unique.
- A person's birth can not occur before 1850.
- A person's birth can not occur after today's date.
- The EntHabit.SEXO must be either 'M' or 'F'. (Male of Female)

Conversions made in the import process:

- Every instance of EntHabit.ID must be converted to upper case.
- First and last names must be cleaned from the character '.
- Sex must be converted from 'F' and 'M' to 'Female and 'Male'.
- Date of birth shall have the format yyyy-mm-dd.

Column name	Source table & column
id	EntHabit.ID
first_name	EntHabit.NOMBRE
last_name	EntHabit.APELLIDO
sex	EntHabit.SEXO
date_of_birth	EntHabit.FECHNAC
mother_id	EntHabit.CODMADRE is the code for the mother. Translated into an id by searching the new database for this person, and get the id. This value sometimes is null.

31.2 Individual_education_types

These values are imported by hand from the [form A].

31.3 Individual_occupation_types

These values are imported by hand from the [form A].

31.4 Individual_education_history

Table EntSitHabit contains both occupation and education events, so much of the error control are equal.

Error controls:

- EntSitHabit.OCUPACION and EntSitHabit.EDUCACION must not be null at the same time. One or the other can be null, but not both.
- The inhabitant that EntSitHabit.ID points at must exist.
- The number EntSitHabit.OCUPACION must be one of the values specified in the table Inhabitant_occupation_types.
- The number EntSitHabit.EDUCACION must be one of the values specified in the table Inhabitant_education_types.
- Check that the date EntSitHabit.FECHA_SIT is larger than the person's birth date + 5 years. The person must be older than five years.
- Check that the date EntSitHabit.FECHA_SIT is smaller or equal to the person's date of death.
- The education event must not be a duplicate. The date of event and the person's id together forms a unique key.
- Check that the connected EntEvento.CODEVENTO is 0, 2 or 7, which is immigration event. The only time they collect the education and occupation data is when someone is immigrating. This might change in the future, so they collect this data every cycle.

Conversion rules made in the import process:

- The string EntSitHabit.ID must be converted to upper case.
- The date EntSitHabit.FECHA_SIT must be converted to this format; yyyy-mm-dd.

Column name	**Source table & column**
individual_id	EntSitHabit.ID gives the code for the Individual. if Individuals.code = EntSitHabit.ID then individual_id = Individuals.id
education_id	Simply the value of EntSitHabit.EDUCACION, because of the similar way of mapping.

observation_id	EntSitHabit.IDEVENTO points at the EntEvento table. EntEvento.IDVISITA points to EntFechVisit, which contains the date of the interview. EntEvento.CODVIV points to a house. Date of interview and house code uniquely identifies an observation. SELECT id FROM Observations WHERE date_of_observation = [the date] AND house_id = [the house];

31.5 Individual_occupation_history

Much of the error controls and conversion rules that apply to education history also apply to occupation history.

Error controls:

- Identical to these in Individual_occupation_history.

Conversion rules made in the import process:

- Identical to these in Individual_occupation_history.

Column name	**Source table & column**
individual_id	See Individual_occupation_history for the same field.
occupation_id	Simply the value of EntSitHabit.OCCUPACION, because of the similar way of mapping.
observation_id	See Individual_occupation_history for the same field.

32 Resident episode tables

These tables describe an individual's residence episodes over time.

32.1 Source tables

The tables we are going to populate in this chapter is collected from the following tables in the current database, LineaBase.

1.32.1 Table EntEvento in current database LineaBase

Table EntEvento in the current database contains the events in the individual's life, birth, death, moving out and in of houses or areas. This is the table where we will extract all events and put them in one of our tables Residence_Immigration_events, Residence_Birth_events, Residence_emigration_events or Residence_death_events.

The column CODEVENTO contains the type of event. These values tell us in which table we should collect the rest of the data about the events. See the table in chapter Individual_residence_episodes below for the actual values.

The columns FECHAEVENT, ID and CODEVENTO give the date of the interview discovering the event, the id of the individual, and the type of event.

Note that the row FECHAEVENT should be the date of the interview, but some misunderstandings have been done, and are sometimes the date of birth or date of immigration etc. Our solution was to ignore this, since it was less than a hundred dates set before 1993, when the survey began.

A much bigger problem is the large amount of FECHAEVENT's set to null, it is more than 1600 rows. A solution to this would be to use the date of interview, found in table EntFechVisit. The interview is identified in column IDVISITA.

1.32.2 Table EntMigracion

Table EntMigracion contain around 20.000 rows of data about immigration and emigration events. This table have the following columns of interest; CausaEvento, Otros and Comentario.

The column TipoMigrac was an attempt to give a type of migration, but not very complete, and therefore ignored.

The column DondeMig means 'where did you move', and was an attempt to keep track of where people moved. In reality this was a very unsure value, since people do not know why someone moved out, and sometimes they just lie because the real reason could be private problems of any kind. This value is therefore ignored.

The column CausaEvento means 'reason of moving', and contains a mix of text and numbers. The numbers is a set of predefined reasons, and can be transferred directly into our database, since we chose to have the same system. The rows containing a text value is inserted in our database as 'Otro', and the actual text is stored in a _other_types table, either Immigration_reason_other_types or Emigration_reason_other_types, depending whether it is an immigration or emigration.

The column Otros can almost always be transferred directly to one of the _other_types tables, but some errors exist here as well. The value of column

CausaEvento must be 12 or 13 (Other or Unknown) in order to be able to use the value of column Otros.

The column Comentario has been used in some cases (around 1000) to give extra information. A suggestion is to just concatenate this text with the CausaEvento, but the best is to ask again.

1.32.3 Table EntHistEmb

Contain the details about pregnancies.

1.32.4 Table E_Nacim

Contain the details about births.

1.32.5 Table E_Fallecido

Contain the details about death causes.

32.2 Individual_residence_episodes

The current database design does not implement episodes in the same way as we have done in our design, so the transfer process is a bit complicated.

The table EntEvento contains the events for every individual. These events describe the individual's birth, death, emigration and emigration, and can be sorted by date. By sorting out the events for one individual and sort them by date, it is easier to extract them into episodes.

An episode is always a starting event and an ending event, where the ending event's date is bigger or equal to the starting event's date.

Now take the sorted list from above, and start with the oldest event. This is the starting event of the individual's first episode. The next event is then the ending event of that episode. This process is repeated until the list is traversed in total.

If the last and thereby newest event is a starting event, the episode created will be open. It will be closed the next time an ending event is recorded.

Now, this is the theory. The reality is quite a bit more complicated, and with a lot of exceptions to the rules. See below for a detailed description of the transfer process.

Data sources:

- Table EntEvento contains all necessary data about the resident episodes. The following values are collected: FECHAEVENT, ID, CODVIV, CODEVENTO and IDVISITA.
- Table EntMigracion is added later as an attempt to improve the current database, but never came into use. This table is therefore ignored.
- Table ListEventos contains an outdated list of the values of EntEvento.CODEVENTO. The most up to date list of values is below.

Mapping of the values of EntEvento.CODEVENTO:

Value	Type	Description
0	DSS Entry	Immigration in 1993, start with this one, in order to migrate all the people to the houses they lived in 1993. Starts a new resident episode.
1	Ignore	People that lives in the same house as 1993. This data is still collected. Ignore.
2	Immigration	Immigration between 1993 and 2002. This is new person's entering the control area from external areas. Starts a new resident episode.
3	Emigration	Emigration between 1993 and 2002. This is person's leaving the control area. Terminates a resident episode.
4	Ignore	Internal migration between 1993 and 2002. Should be ignored, since the methods for keeping track of this were not satisfactory.
5	Death	Death event. Terminates a resident episode.
6	Birth	Birth event. Starts a new resident episode.
7	Immigration	Immigration from external areas after 2002. Starts a new resident episode.
8	Emigration	Emigration from internal to external areas after 2002. Terminates a resident episode.

Something about the storage in the current database:

All episodes must start with one of the values DSS Entry (0), Immigration (2), Birth (6) or Immigration (7).

All episodes can only end with one of the values Emigration (3), Death (5) or Emigration (8). The last (newest) episode does not have to have an ending episode.

The Emigration event with the value 8 have a slightly special meaning; when an episode is ended because the person moved out of the house, the value 8 is written in the database.
The next time the interviewers come back to the house, they ask again if the person does or does not live in the house. If not, the value 8 is once again written.
The third time the interviewer's returns, they ask, and if the person still doesn't live in the house, the value 3 (or 8) is written in the database.

This method ensures that a person is not registered as emigrated just because he visited her parents or similar occasions. Because of the three visits rule, some time passes before it is realized that he has moved.

This means that three emigration events (3 or 8) must follow after each other before the individual can be considered an emigrant, i.e. left the surveillance area.

Note that if the person is encountered living in a new house, before or while these three emigration events has been recorded, an immigration event is recorded, with no emigration event from the old house!

As a result, it is perfectly legal to enter an immigration event after another immigration event in the current database.

Also, it is perfectly legal to enter up to three emigration events and then any type of event after that.

Import procedure:

- Select all events from table EntEvento with only the CODEVENTO values of interest. (0, 2, 3, 5, 6, 7, 8), sort them by ID (The individual's id) and FECHAEVENT (Date of interview). This will create a list where every person's every event is sorted first by person and then by date. In other words, a list of all events for one person, sorted by date, and then the next persons all events, and so on. This will allow us to traverse the list from start to finish as described in the beginning of this chapter.
- For every individual's list of events:
    a. The first event must be of type Birth or Immigration. This event will initialize the episode. It is also the oldest record about that person.
    b. If more events exist for that individual, it must be of type Emigration or Death. This event will terminate the episode.
        - If it was a Death, no more events are allowed for that person.
        - If it was an Emigration, there can follow 1 or 2 more Emigration events. If more than 2 follow, it is an error. It is, however, legal to have a death event after two emigration events.
    c. If more events follow for the same individual, it must be of type Immigration, which initializes a new episode. Go to step a.

Error controls:

- None of the values are allowed to be null.
- The individual must exist in the database.
- The house must exist in the database.
- Dates must be bigger or equal than the individual's date of birth.
- Dates must be smaller or equal than the individual's date of death.
- The first event must be DSS Entry, Birth or Immigration. (0, 6, 2 or 7)
- All following events can be of any type except Birth (6).
- Not more than three Emigration (3 or 8) events can follow each other.
- No more events are allowed after a Death (5) event.
- No event's date can be smaller than the previous event.

Conversion rules made in the import process:

- The strings ID and CODVIV are converted to uppercase.
- The date of event is converted to this format: yyyy-mm-dd.
- The value EntEvento.CODEVENTO is transformed to one of the values DSS Entry, Immigration, Emigration, Birth or Death. (See conversion table above).
- If an Immigration event follows a DSS Entry, Immigration or Birth event, an Emigration event must be created in between them. The date of the created Emigration event should be the same as the following event's date.
- If an Emigration event follows another Emigration event, it should be ignored.
- If a Death event follows an Emigration event, the Emigration event should be replaced with the Death event.

- When all the rules above has been applied on all of the individual's events, the episodes can be created without problems, by connecting the first event to the second event, the third with the fourth, and so on.

Specific precautions:

- The date of all DSS Entry events (where EntEvento.CODEVENTO is 0) is set to 1993-01-01, since this is the date of the start of the system.
- The exception to this rule is if the event occurred in a house which is inside one of the new areas that was added in 2002. The date of event is then set to 2002-01-01.
    - The list called NUEVAS AREAS 2002 contains the houses which where added to the surveillance area in 2002.

Column name	Source table & column
id	Auto generated
individual_id	EntEvento.ID
house_id	EntEvento.CODVIV
start_date	EntEvento.FECHAEVENT when EntEvento.CODEVENTO = {0,2,6,7}
initiating_event_type	DSS Entry, Immigration or Birth. Converted from EntEvento.CODEVENTO when the value are one of the following: {0,2,6,7}
start_observation_id	Search the EntFechVisit which IDVISITA equals EntEvento.IDVISITA, store the EntFechVisit.FECHA. Search the single Observation which date and house_id equals EntFechVisit.FECHA and

	EntEvento.CODVIV. Store the Observation.id from the matching observation in start_observation_id.
end_date	EntEvento.FECHAEVENT when EntEvento.CODEVENTO = {3,5,8} Can be null.
terminating_event_type	Emigration or Death. Converted from EntEvento.CODEVENTO when the value are one of the following: {3,5,8} Can be null.
last_observation_id	Exactly the same procedure as for start_observation_id. Can be null.

33 Event tables

The event tables are connected to the Resident episode table. The events start or end an episode.

The current database contains some attempts to split out the immigrations, emigrations, death events and birth events into its own tables, and link them to the table EntEvento, from where we extracted the content of table Individual_residence_episodes above. These attempts are done mid-way, and therefore are not complete. This is a complicating factor when transferring the data to our new database, and requires careful analysis.

33.1 Residence_immigration_events

An immigration event starts an episode. When the value of the column CODEVENTO in table EntEvento equals Immigration (2 or 7), and a new episode is successfully opened in table Individual_residence_episodes, we must also store a row in Residence_immigration_events. The information to store is found in table EntMigracion.

According to the discussion in the chapter Table EntMigracion above, the columns CausaEvento and Otros contain the actual data.

In order to find the correct row in table EntMigracion, the column EntEvento.IDEVENTO must match EntMigracion.IdEvent.

Column name	Source table & column
individual_residence_episode_id	Points to the newly created row in table Individual_residence_episode, which this immigration event is starting.
immigration_origin_type_id	Taken from column EntMigracion.DondeMig. (WhereMigrated - To or From, since the row is used both for immigration and emigration.) Our design implementation assumes we can transform the textual description into a limited set of immigration origin types. This means the text-value of DondeMig must be searched for in table Immigration_origin_types, and if found, replaced with its row-number. If not found, we can either ignore the value, or transform it to the closest value during the import procedure.
immigration_reason_type_id	If EntMigracion.CausaEvento is a number, just use the number to point to a row in the Immigration_reason_types. This number should always be the same as in EntEvento.CODEVENTO, otherwise it is an error in the database. If instead, the EntMigracion.CausaEvento is a text, it must be stored in table

	Immigration_reasons_other, and point back to this row. The immigration_reason_type_id is set to Otros.
id	Auto generated

1.33.1 Immigration_origin_types

See document Static Values [SV], the chapter with the same name.

1.33.2 Immigration_reason_types

See document Static Values [SV], the chapter with the same name.

1.33.3 Immigration_reason_other_types

When EntMigracion.CausaEvento and EntEvento.CODEVENTO equals, this table is not concerned at all. In the case of when EntMigracion.CausaEvento is a textual value, describing the cause of the immigration, we must store this text in table Immigration_reason_other_types, and connect it back to the active row in Residence_immigration_events. The active row is the row currently being imported.

Column name	**Source table & column**
immigration_event_id	Points to the active row in table Residence_immigration_events.
description	The value in EntMigracion.CausaEvento if it is a textual value. If the column EntMigracion.Otros has a value in the active row, and the value of column CausaEvento is 12 or 13 (Other or Unknown), the value can be concatenated to the end of the description.

33.2 Residence_birth_events

A birth event starts an episode, and must be the first event for an individual.

When the value of the column CODEVENTO in table EntEvento equals Birth (6), and a new episode is successfully opened in table Individual_residence_episodes, we must also store a row in Residence_birth_events. The information to store is found in table E_Nacim.

E_Nacim.IDEVENT must equal EntEvento.IDEVENTO.

No code for importing this data exist in the import program, so the exact procedure for doing this is yet to be found out.

Column name	Source table & column
id	Auto generated
individual_residence_episode_id	Points to the newly created row in table Individual_residence_episode, which this birth event is starting.
pregnancy_outcome_birth_event_id	Points to the row containing the pregnancy result in table Pregnancy_outcome_birth_events.

33.3 Residence_emigration_events

An emigration event ends an episode. When the value of the column CODEVENTO in table EntEvento equals Emigration (3 or 8), and an open episode exists in table Individual_residence_episodes, we must also store a row in Residence_emigration_events. The information to store is found in table EntMigracion.

According to the discussion in the chapter Table EntMigracion above, the columns CausaEvento and Otros contain the actual data.

In order to find the correct row in table EntMigracion, the column EntEvento.IDEVENTO must match EntMigracion.IdEvent.

Column name	Source table & column
individual_residence_episode_id	Points to the active row in table Individual_residence_episode, which this emigration event is ending.
emigration_destination_type_id	Taken from column EntMigracion.DondeMig. (WhereMigrated - To or From, since the row is used both for immigration and emigration.) Our design implementation assumes we can transform the textual description into a limited set of emigration origin types. This means the text-value of DondeMig must be searched for in table Emigration_destination_types, and if found, replaced with its row-number. If not found, we can either ignore the value, or transform it to the closest value during the import procedure.
emigration_reason_type_id	If EntMigracion.CausaEvento is a number, just use the number to point to a row in the Emigration_reason_types. This number should always be the same as in EntEvento.CODEVENTO, otherwise it is an

	error in the database. If instead, the EntMigracion.CausaEvento is a text, it must be stored in table Emigration_reasons_other, and point back to this row. The emigration_reason_type_id is set to Otros.
id	Auto generated

3.33.1 Emigration_destination_types

See document Static Values [SV], the chapter with the same name.

3.33.2 Emigration_reason_types

See document Static Values [SV], the chapter with the same name.

3.33.3 Emigration_reason_other_types

When EntMigracion.CausaEvento and EntEvento.CODEVENTO equals, this table is not concerned at all. In the case of when EntMigracion.CausaEvento is a textual value, describing the cause of the emigration, we must store this text in table Emigration_reason_other_types, and connect it back to the active row in Residence_emigration_events. The active row is the row currently being imported.

Column name	Source table & column
emigration_event_id	Points to the active row in table Residence_emigration_events.
description	The value in EntMigracion.CausaEvento if it is a textual value. If the column EntMigracion.Otros has a value in the active row, and the value of column CausaEvento is 12 or 13 (Other or Unknown), the value can be concatenated to the end of the description.

33.4 Residence_death_events

A death event ends an episode, and must be the last event for an individual.

When the value of the column CODEVENTO in table EntEvento equals Death (5), and an open episode exists in table Individual_residence_episodes, we must also store a row in Residence_death_events.

Table E_Fallecido contains the details about the death event. Not every death event has a corresponding row in table E_Fallecido; less than 50 percent has been recorded this way. The reason for this is that table E_Fallecido is a new design, and when they recorded death events before this new design, they did not collect this information. There is more than 1600 death events in total, but only around 500 is recorded in

table E_Fallecido. This means our database must cope with the case of missing information.

Transfer process

- E_Fallecido.IDEVENT must equal EntEvento.IDEVENTO. If no corresponding row exists in E_Fallecido for a death cause, the missing values for died_from_injuries and death_certificate_exists must default to 'No', and death_code defaults to 0.
- Column LLENCERT tells if there is a death certificate or not. Value 1 means no, 2 means yes.

Column name	Source table & column
id	Auto generated
individual_residence_episode_id	Points to the active row in table Individual_residence_episode, which this emigration event is ending.
died_from_injuries	
death_certificate_exists	Taken from column LLENCERT in table E_Fallecido, where 1 is transformed into 'Yes' and 2 into 'No'.
death_code	

4.33.1 Death_around_pregnancy

If a women has died when pregnant or in the period of 3 months after giving birth, the death is stored in this table. Note that this is not finished in our database implementation, why there is no transfer process defined.

Table E_Fallecido contains this information. The column EMBMRT (Embarazo Muerto, died when pregnant) is set to the value 2 when the woman has died due to the pregnancy, or if strong suspicions exist.

4.33.2 Death_cause_certificates

If a doctor can examine the body, he will give a death cause certificate. It is however usual that no doctor examine the body and then only the statements of the relatives or neighbours will exist.

If a doctor is present, a death cause certificate will exist, and will be stored in this table. If only the statements of the relatives exist, these will be stored in table Death_cause_no_certificates, see next chapter.

Since the design of these tables is not finished in our conceptual design, and no data is stored in the current database, no transfer process is described here.

Table E_Fallecido contains this information. If the column LLENCERT is set to 1, the certificate details is filled in as well, otherwise these fields are empty.

4.33.3 Death_cause_no_certificates

When no doctor has been available to examine the body, it is just the statements of the relatives or neighbours which will be stored. These are stored as a textual description. As for Death_cause_certificates, no data exist in the current database, and consequently no transfer process is defined.

4.33.4 Certificate_issuer_types

See document Static Values [SV], the chapter with the same name.

4.33.5 Certificate_issuer_other_types

Since the standard with a defined set of Certificate_issuer_types was introduced later in the current database, several different descriptions and words exist. Since the work with translating these to the predefined set is not finished, we must transfer all these words and sentences as well.

When inserting a new row in this table, the value certificate_issuer_type in table Death_cause_certificates must be set to ‘Otro’.

No transfer process is described for this table, see Death_cause_certificates above.

4.33.6 Deaths_by_injury

4.33.7 Injury_instrument_types

4.33.8 Injury_types

See document Static Values [SV], the chapter with the same name.

4.33.9 Injury_other_types

If an injury type is not listed in table Injury_types, it must be stored here as a textual value. The value injury_type in table Deaths_by_injury must be set to ‘Otro’ as always with _other_types tables.

34 Relation to head tables

The relation to head tables is a way of keeping track of how people are related economically to each other, as well as relational in the normal sense.

Each individual has a relation to the head of the house. This information is found in table EntRelCF.

Columns of interest in table EntRelCF:

- IDEVENTO is a reference back to the row in table EntEvento, where the value IDVISITA is referring to a row in table EntFechVisit, which contains data about the interview.
- ID is the individual's id,
- ID_HEAD is the id of the head of house,
- RELHEAD is the relation the individual has to the head of house.
- FECHA_REL is the date of when the relation was active, and is usually the same date as the date of the interview.

The RELHEAD values span from null, 0 to 99. The values null and 99 should be interpreted as value missing. Values 1 to 7 can be directly mapped into our implementation and these values stand for about 85% of the content.

The rest of the values, 0 and 8 to 98 are currently unknown, but a simple questionnaire to the computer people at CIDS should give a good answer.

Note that no implementation of a transfer progress for these tables has been done, so the process described below is clearly experimental and not tested.

34.1 Head_of_house_episodes

Contain episodes for a house where an individual has been head of the household.

When the value in EntRelCF.RELHEAD is equal to CF (1), it means that the individual is the head of the household.

This means it is fairly straightforward to extract the episodes for the head of the household. We must take into account the house, which is defined in table EntEvento.CODVIV. The connected row is found where EntEvento.IDEVENTO equals EntRelCF.IDEVENTO.

1. Sort out the rows where RELHEAD is equal to CF (1).
2. For each row, find the connected row in EntEvento, extract the CODVIV value.
3. Now sort the rows by CODVIV and then by FECHA_REL.

This will give us a list sorted by house, and in sequence based on time.

Now it is easy to traverse this list, taking the first individual (EntRelCF.ID) and open a new episode. Each

1. The first record creates a new episode, which start_date will be set to the FECHA_REL of the event. Individual_id is set to EntRelCF.ID, house_id is set to CODVIV.
2. The successive records value EntRelCF.ID is compared with the open episode's individual_id, and only when they differ, the episode is ended. The end_date is then set to the current records FECHA_REL.
3. At the same time a new episode is created, go to step 1.

This procedure is repeated until no more events exist for a house.

Column name	Source table & column
id	Auto generated
individual_id	Directly transferred from EntRelCF.ID
house_id	The row in EntEvento where IDEVENTO match EntRelCF.IDEVENTO contain the value CODVIV, which is set as the house_id.
start_date	EntRelCF.FECHA_REL
start_observation_id	Trough EntEvento.IDVISITA the row in table EntFechVisit can be found.
end_date	EntRelCF.FECHA_REL
end_observation_id	Trough EntEvento.IDVISITA the row in table EntFechVisit can be found.

34.2 Relationship_to_head_episodes

Contains episodes where an individual have had a certain relationship to the head of the household during the same period of time.

The easiest way to extract the data from table EntRelCF is to sort the table first by ID, then by FECHA_REL, which is the date of the registration of the relation.

Now it will be possible to traverse the generated list from beginning to end, extracting each individual's events in order oldest to newest.

4. The first record for an individual creates a new episode, which start_date will be set to the FECHA_REL of the event and the RELHEAD is stored in relationship_type_id.
5. The successive records value RELHEAD is compared with the open episode's relationship_type_id, and only when they differ, the episode is ended. The end_date is then set to the current records FECHA_REL.
6. At the same time a new episode is created, go to step 1.

This procedure is repeated until no more events exist for an individual.

Column name	Source table & column
id	Auto generated
individual_id	Directly transferred from EntRelCF.ID
relationship_type_id	The value EntRelCF.RELHEAD.
start_date	Value EntRelCF.FECHA_REL for the first event with a value RELHEAD differing from the previous episode. If no previous episode exists, a new episode is created.
start_observation_id	Trough EntEvento.IDVISITA the row in table EntFechVisit can be found.
end_date	When the value of EntRelCF.FECHA_REL differs from the open episode's relationship_type_id, the episode is closed by setting the end_date to EntRelCF.FECHA_REL.
end_observation_id	Trough EntEvento.IDVISITA the row in table EntFechVisit can be found.

34.3 Relationship_to_head_types

See document Static Values [SV], the chapter with the same name.

35 Pregnancy information tables

The pregnancy information tables are in many ways the most important part of the database, since it keep track of the health of the mothers and children, in other words the health of the population.

The table EntHistEmb contains the history of the pregnancies and births among the women in the surveillance area. A woman's all pregnancies are recorded here, even historical ones.

The table E_Nacim contains the information about a child's birth when it is born inside the surveillance area. Children born outside of the surveillance area is not recorded here.

The following fields are of interest in table EntHistEmb:

- ID – The mother's id.
- NUMEMB – The number of pregnancies including this pregnancy the woman has had. There are some values 04A and 04B, which I assume can be transformed into 4. The maximum value is 20, which means there is some effective couples.
- FINEMB – The date of the birth. There are about 1200 null values, and some invalid values, but most values are ok.
- RESEMB – The result, live born, still born, spontaneous abortion or provoked abortion. The values span from null, 1 to 4, where 1 mean yes, 2 means no. 3 means spontaneous abortion and 4 means provoked abortion. Note: This might be incorrect, and need a new analysis.
- LLORNAC – Cry on birth, this is a polite question they ask when the child has died, to find out whether or not the baby was alive on birth, or already dead when born. RESEMB is effectively the same value. This table is empty and therefore not used in the transfer process.
- SEXON – Boy or girl. 1 is boy, 2 is girl. There are 3 occurrences of 'F' as in female, 3 of 'H' as in hombre and 2 of 'M' as in male.
  Also there is around 5000 which does not have a value. The absence of a value almost always mean the value of RESEMB is set to abortion, since the sex is unknown in these cases.
- ESTACT – (Estado Actual) Whether or not the child is alive today. Possible values are 1 (Yes) and 2 (No). When the value is No, the FCMUE below is set to the date of death.
  There are about 5500 rows with no value, which should be interpreted as 'Don't know', or possibly the child was born inside the surveillance area, and therefore the information is available elsewhere in the system.
  There are also about 1000 rows with the value 4.
- FCHMUE – Date of death. This value is only entered if the child lived outside the surveillance area, or died before the start of the surveillance system. If born inside the surveillance area after the start, the death is registered in other parts of the system.

The following fields are of interest in table E_Nacim:

- EMBNUM – The same as NUMEMB in table EntHistEmb; the number in the family of children. This value adds no more data, and is ignored in the transfer process.
- TIPOEMB – The same as RESEMB in table EntHistEmb. This value adds no more data, so it is ignored in the transfer process.
- ATENPARTO – Spans from 1 to 4. Chapter Birth_attendant_types in document [Static values] specify these. If the value is set 4 (‘Otro’) the column OTROENPA will contain more information.
- OTROTENPA – When ATENPARTO is set to 4 (‘Otro’), this column contain a string with information of who else was present at the birth.
- LUGATEN – Where the birth happened. This value span from 1 to 5. Chapter Delivery_place_types in document [Static values] specify these. If the value is set to 5 (‘Otro’), the column OTROLUGATE will contain more information.
- OTROLUGATE – When LUGATEN is set to 5 (‘Otro’), this column contain a string with the place of birth.
- TERMPARTO – Type of birth. Values can be 1 or 2, and means Vaginal and Caesarean.

The exact transfer procedure for the pregnancy tables has never been defined and implemented, I had no time to do it. Although, the rough analysis below can help to understand how it can be done.

Transfer procedure for EntHistEmb:

1. When table EntEvento’s column CODEVENTO equals Birth (6), we can find the connected row in EntHistEmb with the key IDEVENTO.
2. By sorting the table by the ID of the mother, and then by NUMEMB, we get a list where we can find a woman’s all pregnancies sorted in time order. We could also sort by ID and then FINEMB and get the same result. It would be fun to do both, and then compare the lists, and maybe that way find some mistakes.
3. For each woman and pregnancy row, extract the values IDEVENTO, FINEMB, RESEMB, SEXON, ESTACT and FCHMUE. These are (in the same order) Id of event, Date of birth, Result of birth, Sex, Alive today, Date of death.
4. With the Id of the event, we can find the corresponding row in the table E_Nacim, and get the rest of the child’s information. Note: This might be incorrect, and need a new analysis.

35.1 Individual_pregnancy_episodes

35.2 Pregnancy_ abortion_events

35.3 Pregnancy_ birth_events

35.4 Pregnancy_births_information

35.5 Births_information

35.6 Delivery_place_types

See document Static Values [SV], the chapter with the same name.

35.7 Delivery_place_other_types

35.8 Birth_attendant_types

See document Static Values [SV], the chapter with the same name.

35.9 Birth_attendant_other_types

35.10 External_individuals_status

35.11 External_individuals_deceased

36 References

Document	Description
[SDP]	The document ‘Software Development Specification’
[Form A]	Form ‘Formulario de linea de base’*
[Form Pregnancies]	Form ‘Formulario historia de embarazos’*
[Form Death]	Form ’Formulario Defunciones’*
[Form Birth]	Form ’Formulario Nacimientos’*
[Form Immigrations]	Form ’Formulario Inmigracion’*
[Form Emigrations]	Form ’Formulario Emigracion’*

*Note that these ‘formularios’ is inaccessible and need to be requested from CIDS.

LUND
UNIVERSITY

Centre for Demographic and Health Research

Demographic Surveillance System Database

Static Values

Author

Jon Lennryd

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

Contents

37 Introduction

This document contains the static tables of the database. This data is collected from the current database, and is considered static in the way that the data is not expected to change during the lifetime of the database. If a change is necessary it is not a problem, since the design of the database is not dependant on the content of these tables.

38 Content of static database tables

Each field in a database table is defined on the first (grey) row in the tables below. The following (white) rows contain the static values of the table.

38.1 Health_care_centre_types

id	description
1	CS
2	PS

38.2 Health_care_centres

The values in row health_care_centre_type_id are missing, yet to extract from the original database.

id	name	code	health_care_centre_type_id
1	Mantica	M01	
2	1ro de Mayo	M02	
3	Oscar P.C	M03	
4	Bejamín Z	M04	
5	William F	M05	
6	William R	M06	
7	Recolecion	M07	
8	Lechecuago	M08	
9	Tololar	M09	
10	Recolección	M10	
11	Platanal	M11	
12	Perla M	P01	
13	Villa 23 julio	P02	
14	Recreo	P03	
15	Arrocera	P04	
16	Fundesi	P05	
17	Calvarito	P06	
18	Santana	P07	
19	Salinas Grande	P08	
20	Miramar	P09	
21	La Leóna	P10	
22	La Ceiba	P11	
23	Chacaras	P12	
24	Rubén D	P13	
25	Sutiava	S01	
26	Providencia	S02	
27	Poneloya	S03	
28	Walter Ferrety	S04	

id	name	code	health_care_centre_type_id
29	Goyena	S05	
30	Troilo	S06	
31	Abangasca	S07	
32	Movil	S08	

38.3 Health_care_centre_covers_districts

This table is populated with the id of the health care centre, and the id of the district the health care centre is responsible for.

For compactness, the column district_id contain several values. When created in a real database, each district_id will of course have its own row, so the health_care_centre_id will be repeated several times.

health_care_centre_id	district_id
1	65,66,71,72,63
2	62,61
3	None
4	73,64
5	70
6	69,67
7	None
8	74,72
9	None
10	None
11	None
12	41,32,34
13	31
14	37,42,38,36,39,40
15	None
16	28,29,30
17	33
18	56,55,72,35
19	43,57
20	None
21	None
22	47,49,50,51,59,48,53,46,52,60
23	44,45
24	None
25	14,19,15,2,8,1,9,11,17,13,12,20,16,10,18
26	7,6,5,3,4
27	None

health_care_centre_id	district_id
28	None
29	24
30	27,25,23,26
31	21,22
32	None

38.4 Territories

id	name
M	Mantica
P	Perla Maria
S	Subtiava

38.5 Districts

id	name	code	zone_type	territory_id
1	Consejo No 3	A2	Urban	S
2	Consejo No 1	A3	Urban	S
3	La Providencia	A4	Urban	S
4	Rpto. Roger Deshon	A5	Urban	S
5	Asent. Justo E. Centeno	A7	Urban	S
6	Adiac II Etapa	A8	Urban	S
7	Adiac I Etapa	B1	Urban	S
8	Consejo No 2	B2	Urban	S
9	Consejo No 4	B3	Urban	S
10	Rpto. Ronald Sandino	B4	Urban	S
11	Consejo No 5	B5	Urban	S
12	Rpto. Fanor Urroz	B7	Urban	S
13	Rpto. san Mateo	B8	Urban	S
14	Asent. Hipolito Sanchez	B9	Urban	S
15	Colonia Paulino Guevara	C1	Urban	S
16	Rpto. Reynaldo Hernandez	C2	Urban	S
17	Rpto. Juan Jose Alvarez	C3	Urban	S
18	Rpto. Santa Lucia	C4	Urban	S
19	Asent. Saul Alvarez	D2	Urban	S
20	Rpto. Felipe Santana	D4	Urban	S
21	Abangasca Central	E1	Rural	S
22	Abangasca Norte	E2	Rural	S
23	Troilo	E3	Rural	S
24	Monte Oscuro	E5	Rural	S
25	Goyena	E6	Rural	S
26	Villa Jerusalen	E7	Rural	S
27	Abangasca Sur	E8	Rural	S
28	Fundeci I	A2	Urban	P
29	Fundeci II	A4	Urban	P
30	Fundeci III	A5	Urban	P
31	Villa 23 de Julio	B6	Urban	P

id	name	code	zone_type	territory_id
32	El Calvario	B9	Urban	P
33	El Calvarito	C1	Urban	P
34	San Sebastian	C3	Urban	P
35	Rpto. Rigoberto Lopez P.	C5	Urban	P
36	Rpto. Che Guevara	C8	Urban	P
37	Rpto. 18 de Agosto	C9	Urban	P
38	Rpto. Carlos Fonseca	D1	Urban	P
39	Rpto. Emir Cabezas	D2	Urban	P
40	Rpto. Juan R. Sampson	D4	Urban	P
41	Bo.El Laborio	D6	Urban	P
42	Rpto. Brisas de Acosasco	D8	Urban	P
43	Salinas Grandes	E1	Rural	P
44	Chacraseca	E2	Rural	P
45	Mojon Sur	E3	Rural	P
46	La Ceiba	E4	Rural	P
47	Amatitan	E5	Rural	P
48	Hato Grande No 3	E6	Rural	P
49	Boca de Cantaro	E7	Rural	P
50	El Convento	E8	Rural	P
51	Hato Grande No 3 No2	F1	Rural	P
52	Mjon Sur No 2	F2	Rural	P
53	La Arenera	F3	Rural	P
54	Paso de Tabla	F7	Rural	P
55	Divino Ni?o	G1	Rural	P
56	Ciudadela	G2	Rural	P
57	Villa Esperanza	G3	Rural	P
58	Ato Grande No1, Reparto Divino Niño	J1	Rural	P
59	Hato Grande No 1	J5	Rural	P
60	Villa Asuncion	J7	Rural	P
61	Rpto. Jose B. Escobar	A6	Urban	M
62	Rpto. 1ro de Mayo	A7	Urban	M
63	Rpto. Aracely Perez	B5	Urban	M
64	Rpto. Benjamin Zeledon	B9	Urban	M

id	name	code	zone_type	territory_id
65	Bo. Ermita de Dolores	C3	Urban	M
66	Bo San Juan - San Jose	C4	Urban	M
67	Rpto. Heroes y M. de Zaragoza	C7	Urban	M
68	Poso Hondo	G8	Rural	M
69	Bo. Zaragoza	C8	Urban	M
70	Lechecuagos	C9	Rural	M
71	Monte Redondo	D1	Rural	M
72	Palo de Lapa	D2	Rural	M
73	Anexo Maritza Lopez	D8	Urban	M

38.6 Censuses

This table contain start_date and end_date for several censuses. One period describe the time for one survey, when all houses are visited and every person is interviewed.

This table is not yet extracted from the original database, due to unclear information.

38.7 Observers

This table contain the first_name and last_name of the interviewers. It is not extracted from the original database since the values has not been updated for several years.

38.8 House_water_types

Table ListCondViv is an old outdated version of all values for a house, but this and the other House tables below is the most up to date.

id	description
1	Tubería adentro
2	Tubería puesto comunal
3	Pozo propio
4	Pozo comunal
5	Río/Quebrada
6	Comprada en Barril/Bidones
7	Otros

38.9 House_floor_types

id	description
1	Ladrillo de Cerámica
2	Ladrillo de Cemento
3	Ladrillo de Barro
4	Embaldosado
5	Suelo

38.10 House_wall_types

id	description
1	Ladrillo / Cemento
2	Adobe / Taquezal
3	Madera
4	Palma
5	Cartón / Plástico / Metal / Ripios
6	Otros

38.11 House_lavatory_types

id	description
1	Inodoro
2	Excusado
3	No tiene

38.12 Individual_education_types

id	description
1	1er Nivel
2	2do Nivel
3	3er Nivel
4	1er Grado
5	2do Grado
6	3er Grado
7	4to Grado
8	5to Grado
9	6to Grado
10	1er Año
11	2do Año
12	3er Año
13	4to Año
14	5to Año
15	6to Año
16	1er Año Universitario
17	2do Año Universitario
18	3er Año Universitario
19	4to Año Universitario
20	5to Año Universitario
21	Profesional
22	No Aplicable
23	Analfabeto
24	Alfabetizado
25	Habilitación Técnica (4 meses)
26	Técnico Básico
27	Técnico Medio (Diploma)

38.13 Individual_occupation_types

Table ListOcupation contains this list as well.

id	description
1	Desempleado
2	Vendedor ambulante
3	Ama de casa
4	Empleada domestica
5	Trabajador industrial
6	Trabajador agricula
7	Estudiante
8	Ejecicio profesional
9	Artesano
10	Empresario
11	Comerciante
12	Pensionado
13	Discapacitado
14	Ofic/Maest/Salud
15	No escuela
16	No trabaja
17	Seguridad
18	Otros
19	Conductor

38.14 Immigration_origin_types

id	description
1	Dentro del municipio
2	De otro municipio
3	De otro departamento
4	De otro país

38.15 Immigration_reason_types

id	description
1	Por empleo
2	Pleito / Problemas
3	Unión / Matrimonio
4	Divorcio / Separación
5	Problemas de salud
6	Independencia / Hacinamiento
7	Vino a estudiar
8	Posante / Inquilino
9	Muerte del tutor
10	Poco ingreso
11	Desconocido
12	Otros

38.16 Emigration_destination_types

The translation is not part of the table.

id	description	*translation*
1	Dentro del municipio	*Inside the jurisdictional district*
2	A otro municipio	*To another jurisdictional district*
3	Fuera del departamento	*Left the region*
4	Fuera del país	*Left the country*
5	No sabe	*Dont know*

38.17 Emigration_reason_types

The translation is not part of the table.

id	description	*translation*
1	Desempleo	*Unemployed*
2	Cambio de trabajo	*Change of work*
3	Pleito / Problemas	*Fighting/Problems*
4	Unión / Matrimonio	*Marriage*
5	Divorcio / Separación	*Divorce*
6	Problemas de salud	*Problem with health*
7	Independencia / Hacinamiento	*Independence/Community*
8	Fue a estudiar	*For studies*
9	Posante / Inquilino	*Lodge/Rent*
10	Muerte del tutor	*Responsible died*
11	Poco ingreso	*Low income*
12	Desconcido	*Unknown*
13	Otros	*Other*

38.18 Certificate_issuer_types

id	description
1	Medico
2	Personal de enfermeria
3	Otro

38.19 Injury_types

id	Description
1	Accidente de trafico
2	Homicidio
3	Suicidio
4	Desconocido
5	Otro

38.20 Relationship_to_head_types

Table ListRelacion contains a list of these values as well.

id	description
1	CF (Cabeza de familia)
2	Esposo(a)
3	Hijo(a)
4	Otro familiar
5	Empleado(a)
6	No familiar
7	Otro

38.21 Delivery_place_types

Table ListLugarAtenc is a source table with similar values as these. The table below is most up to date.

Id	Description	*Translation*
1	Hospital	*Hospital*
2	Centro de Salud	*Health care centre*
3	Clinica	*Private clinic*
4	Domicilio	*Home*
5	Otros	*Other*

38.22 Birth_attendant_types

Table ListAtencEmb contains an outdated list with similar values. However, the list below is the most accurate.

id	Description	*Translation*
1	Medico	*Doctor*
2	Enfermera	*Nurse*
3	Partera	*Midwife*
4	Otros	*Other*