A Methodology for Database Management of Time-Variant Encodings and/or Missing Information

by

William John Threlfall

B.Sc., The University of British Columbia, 1981

A Thesis Submitted in Partial Fulfillment

of the Requirements for the Degree of

Master of Science

in

THE FACULTY OF GRADUATE STUDIES

Department of Computer Science

We accept this thesis as conforming

to the required standard

THE UNIVERSITY OF BRITISH COLUMBIA

March 1988

In presenting this thesis in partial fulfilment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the head of my department or by his or her representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Department of Computer Science

The University of British Columbia
1956 Main Mall
Vancouver, B.C., Canada
V6T 1Y3

Date: Mar. 20, 1988

Abstract

A Methodology for Database Management of Time-Variant Encodings and/or Missing Information

The problem presented is how to handle encoded data for which the encodings or decodings change with respect to time, and which contains codes indicating that certain data is unknown, invalid, or not applicable with respect to certain entities during certain time periods.

It is desirable to build a database management system that is capable of knowing about and being able to handle the changes in encodings and the missing information codes by embedding such knowledge in the data definition structure, in order to remove the necessity of having applications programmers and users constantly worrying about how the data is encoded.

The experimental database management language DEFINE is utilized to achieve the desired result, and a database structure is created for a real-life example of data which contains many examples of time-variant encodings and missing information.

Contents

List of Tables

List of Figures

Acknowledgements

Most of all, I would like to express extreme gratitude to my supervisor, Professor Paul C. Gilmore for his support and for his extraordinary patience in dealing with a married part-time student with a full-time job, a newly-purchased older house and two small children.

Next I would like to profusely thank Dr. Pierre Band, head of the Division of Epidemiology, Biometry, and Occupational Oncology of the Cancer Control Agency of B.C., for his support and understanding, without which the completion of my degree would have been impossible.

I also thank Mr. Richard Gallagher, head of the Epidemiology Section of our Division, for providing the opportunity, and Mr. Harvey Hersom, Director of the Division of Vital Statistics of the B.C. Ministry of Health, for providing the data and partial descriptions of the encoding schemes and changes to such over the years.

Lastly, I thank my wife Laurie for her support and understanding, and for putting up with many lonely days and nights while I was working on this thesis.

Chapter 1

Introduction

Most recent research into database management has been focussed on entities, sets, relationships, associations (the entity-relationship model), or joins, tables, anomalies and relations (the relational model). There has been much work on model design and describing the overall structure of databases, continuing up to meta-level languages and kernels of database management. There has also been work on low-level concerns such as hashing, efficient data structures, and algorithms to perform various tasks.

Very little emphasis has been placed on the actual bytes of information (the *data*) that are stored inside the memory of the computer. The reason is presumably because the actual data is usually a trivial detail of application, to be input perfectly into the final computerized database management system (DBMS).

This thesis will explore some methods for handling "problem" data, where the data pre-exists before applying a database management structure for ease of manipulation.

1.1 "Problem" Data

The "problem" data presented consists of the following three cases:

1. A *time-variant encoding* is when a given field in the database contains two or more different encoded data values at different points in time, but the meaning intended by the

different data values (their *semantic content*) is identical.

2. A *time-variant decoding* is when a given single encoded data value in a given field in the database changes its semantic content at one or more points in time.

3. *Missing information* is when some encoded data values in a field of the database represent the fact that certain information is unknown, not applicable, or invalid (inconsistent with other information, or outside the domain of the attribute) with regard to certain entities during certain time periods.

Time-variant encodings or decodings usually arise because of some bureaucratic change or administrative reorganization. The changes are usually necessary and beyond the control of data management personnel.

Missing information of the type *value at present unknown* is a result of imperfect and incomplete knowledge of the real world, and is only partially controllable by a database administrator. Sometimes, despite the best efforts of data collection personnel, it is simply not possible to obtain all the information that one would like to know.

A database administrator is usually aware of attributes that might result in missing information of the type *attribute does not apply to entity*, but it is not always possible to avoid their occurrence, so it is desirable to develop methods of handling the data values representing this type of missing information.

Missing information of the type *value is inconsistent or invalid* is the most controllable type of problem data. With all-encompassing error-checking data entry software, this type of missing information could be prevented from entering the database. However, such software has not been popular because the time necessary to perform all the cross-checks slows down the speed of data entry to unacceptable levels. Rudimentary type, format or domain checks are sometimes used, but invalid or inconsistent data can still be accidentally entered.

Examples of all the various types of problem data will be drawn from an example database that really exists, a detailed description of which is given in Chapter 3. The example database

is called DVS-Ext.

1.2 An Unsatisfactory "Solution"

One might assume that the problem of having two different data values in the same field which contain the same information when decoded, and the reverse problem of having a single data value in a field with two different time-dependent meanings, can be solved by recoding the entire data set so that there is a one-to-one correspondence between data values in a particular field and the meanings of the decoded data values from that field. However, a detailed explanation in Chapter 3 of an example taken from DVS-Ext will show that such a recoding is not appropriate for DVS-Ext, and therefore would not be universally acceptable. Also, since DVS-Ext is a subset of data acquired from another source, recoding the data would make it incompatible and not comparable with the original full data set.

1.3 The DBMS Connection

The problems involved in responding to database queries, especially queries which ask for negative information such as "display all persons who have never been married", have not yet been adequately solved when missing information is involved. Database operations (such as relational joins) and computed functions (such as averages) have not as yet been able to adequately handle missing information. Also, the author is not aware of any proposals for database management of time-variant encodings/decodings.

It seems reasonable to attempt to solve the problems inherent in time-variant encodings and missing information using DBMS techniques. The goal is to have well-defined, detailed instructions on how to use the problem data built into the DBMS, so that neither the end users nor the applications programmers need to constantly refer to complicated interpretation specifications.

This thesis will show that when the changes over time and the mechanics of how missing

information is recorded are well-defined, database manipulation of problem data can occur with few problems and complications for both users and applications programmers.

1.4 Organization

The remainder of this thesis is organized as follows. Chapter 2 contains more detail on the reasons for encoding data, and the problems of time-variant encodings/decodings and missing information, as well as a review and discussion of relevant literature and a justification for the use of the language DEFINE to solve the problem presented. Chapter 3 contains a detailed description of the attributes (fields) of the example database DVS-Ext, including full explanation of all time-variant encodings/decodings, missing information, and other quirks that require special processing. Chapter 4 contains a brief syntactical and semantical explanation of the language DEFINE, and then proceeds to explain, for each field (attribute) of DVS-Ext, how to handle the various quirks of the data recorded in that field. Chapter 5 contains a discussion of what has or has not been accomplished in this thesis, what future work remains, and some conclusions derived from the experience of trying to deal with incredibly quirky data in a consistent, meaningful way using DBMS techniques.

This thesis is deliberately non-mathematical in nature. It is intended as an exploration of practical issues involved in database management of problem data rather than a detailed theoretical thesis on mathematical formalisms. Refer to Chapter 2 for more explanation.

Chapter 2

Description of Problem

2.1 Encoding

Most of the data stored for medical information or research purposes is *encoded*, meaning that a few numeric digits or a short character string are recorded in the database. The encoded data then has a meaning (its semantic content) when one refers to the appropriate source, which is usually a printed reference manual or book.

The meanings of the encoded data could be stored in the computer, but that would defeat one of the major reasons for encoding, which is to use less storage space inside the computer. Another reason for encoding is to standardize the data. If all data were recorded using character strings containing all relevant information, the problem would exist of different human beings recording the same information using quite different wordings. Encoding forces the data into a fixed format with a fixed decoding schema for relatively easy storage and retrieval of information.

For example, the fact that a person is female may by recorded in a certain field in the database, encoded as the letter F or as the digit 2.

The database administrators would keep records showing how the fields are encoded, and which fields contain which information. Despite careful management, problems which affect the possible use of a DBMS to manipulate the data can easily occur in a database for which

information is gathered over a long period of time.

2.2 Changing of Encoding Schemes

A reasonable question to ask is why the database administrators would suddenly change the encoding scheme for a certain piece of information. One explanation is given for the example database DVS-Ext in Section 4.3, where the changes are external, caused by other persons and factors over which the database administrators have no control.

There are other examples of changes which were totally under the control of DVS, and asking why those changes were made is a very good question. The answer is only speculation, but over many years and several changes in government, various political pressures, bureaucratic rules, and personal preferences can cause changes in encoding schemes. Perhaps someone simply decided that the previous encoding scheme was "poorly designed", so they created a new encoding scheme that "made more sense".

For example, a data manager may decide to encode the gender of persons using the digit 1 for males and the digit 2 for females. A few years later, a new data manager may be offended that males appear to have a "higher priority" than females, and may decide to change the encoding to the letter M for males and the letter F for females. This is an example of an actual change that occurred in the example database DVS-Ext, and a conjectured reasoning for the change.

Despite an intensive literature search, the author has been unable to locate any references relating to database management of time-variant encodings/decodings. It is possible that the concepts involved can be forced into the framework of so-called historical [Clifford 83] or temporal [Ariav 86] databases, but the notion of time-variance presented in this thesis is quite different in nature to the notion of time-variance in historical or temporal databases. Historical and temporal databases attempt to model the changes in reality over time, in order to "keep track" of attribute values that were recorded in the past and changed by updates, deletes, and insertions. For example, a temporal model would store the changes over time to an attribute

such as the salary of an employee. The type of database that this thesis is concerned with is a *static* database, where only the current value of each attribute is of interest. In fact, the example database DVS-Ext is such that the data does not change (except for possible corrections of errors) after it is entered.

There is another area of research which is referred to as either database *translation* or database *conversion*. See [Fry 78] for an overview. Database translation (conversion) is concerned with translating (converting) one database to another, or changing the logical or physical structure of the database. Translation of individual encoded data values is included, but with a different perspective than that presented in this thesis. The translation of data values is concerned with one-to-one mappings from one structure or format to another, not many-to-many mappings within one attribute domain of one database.

2.3 Missing Information

Aside from encoding changes, another major source of problems is missing information. Missing information can be one of three possibilities; either "unknown" (Type 1), "not applicable" (Type 2), or "invalid" (Type 3). An encoded data value whose semantic content is "value at present unknown" refers to the fact that some data value in the domain of the attribute must exist for a particular entity in a particular field, but that value is currently unknown. An encoded data value whose semantic content is "attribute does not apply to entity" refers to the fact that recording a value from the domain of the attribute in a particular field for a particular entity would not make sense. An encoded data value whose semantic content is "value inconsistent or invalid" refers to the fact that the code recorded is impossible in real life, or does not exist according to the encoding manual (is outside the domain of that attribute).

For example, every resident of British Columbia resides in one of the 75 administrative divisions of the province known as "School Districts". If it is known that a person *does* reside in the province of B.C., but not exactly which school district that person resides in, the value "unknown" must be recorded in the School District field in the database. For a person who does

not reside in the province of British Columbia, the value "not applicable" would be recorded in the School District field in the database, since none of the 75 possible values make sense when applied to a non-resident of B.C. The coding schema for School Districts does not specify the value 95 as having any meaning, so any person recorded as residing in School District 95 would have an "invalid" School District code.

This particular example also serves to illustrate another aspect of missing information. The School District field of the database as described herein contains "hidden" information as to whether or not a person is a resident of B.C. If such knowledge is not available, then another, different "unknown" value is needed to convey the information that it is unknown whether or not the person is a B.C. resident.

Several authors have presented theoretical research attempting to deal with missing information in databases. Most research has been restricted to the relational model [Codd 70] and Type 1 ("value at present unknown") missing information.

Codd introduced a three-valued logic (true, false, unknown) for his relational model (for Type 1 missing information) in [Codd 75] and follows up on some of the mechanics of database operations (joins, etc.) using the three-valued logic in [Codd 79]. Codd's approach is the simplest of the methods proposed for dealing with Type 1 missing information in the relational model, and could probably be implemented without too many adverse consequences.

In addition, Codd's work has been modified and expanded by several authors. Grant pointed out a case where Codd's original proposal would give an incorrect result to a query [Grant 77], and also gave a brief but sensible suggestion for handling Type 2 missing information within Codd's framework. In [Lien 79] rules are given for handling database operations and multivalued dependencies involving Type 1 missing information, also for the relational model. The work started in [Codd 79] was substantially expanded upon in [Biskup 83], in which nine questions left unanswered by Codd are answered by Biskup.

The main problem with the approach started by Codd and expanded by Biskup are (aside from being restricted to Type 1 missing information and the relational model) that so-called

"maybe-results" are introduced into the database. Maybe-results are tuples of a relation (attributes of entities) for which the answer to a query is "possibly satisfies", in the sense that, because of missing information, one cannot rule out the possibility that the entity described by the maybe-result tuple satisfies the query. Consider a query as defining a subset of entities. When using the approach of Codd/Biskup, two subsets could result from any given query, one subset of entities that definitely satisfy the query (the "true-result"), and another subset of entities that possibly satisfy the query (the "maybe-result"). The maybe-results clutter up the database unnecessarily, and then need to be dealt with in further queries or other processing.

The more general problem of *partial* information has been tackled by Lipski in [Lipski 79] and [Lipski 81]. The problem of Type 1 missing information is a subset of the problem of partial information. Rather than having a single encoded value whose semantic content is "value at present unknown", partial information substitutes a set (or range) of possible values that a particular attribute could take on. To take an example from DVS-Ext, a person may be known to have died at "over age 100", but the exact age at death is unknown. Thus the single Type 1 null value corresponds to the set of possible values being the entire domain of the attribute, when nothing is known that could restrict the set of possible values of which the value of the attribute could be a member.

Lipski's work is very elegant and attractive from a formal, theoretical point of view, but as pointed out in [Biskup 83], the complexity of processing required is too high for practical implementation. Lipski's "inner limit" corresponds to Codd's "true-result", and Lipski's "outer limit" corresponds to the union of Codd's "true-result" and "maybe-result", so there is some degree of overlap between Lipski's formalisms and Codd's work. In [Grant 79] an indication is given of how partial information might be handled within Codd's framework. Partial information is dealt with in some degree in this thesis, since aggregation of ranges of values of the *age at death* attribute is used to create a subset of persons whose age at death is specified only as "over age 100". However, this thesis is concerned with the case where the value of an attribute could be any value from the domain (rather than belonging to a specific subset of the domain), so nothing further will be said about the general problem of partial information until

the concluding remarks in Chapter 5.

Another approach is outlined in [Vassiliou 79], and followed up in [Vassiliou 80], which is similar to that taken by Codd and Biskup. Vassiliou's treatment uses denotational semantics, and handles Type 2 as well as Type 1 missing information, but also introduces maybe-results into the database. The idea is to treat the possible domain values as a partially ordered lattice, where the result of a query (subset of attributes for a subset of entities) contains either not enough information (Type 1 null), all the information (no nulls), or too much information (Type 2 null). The approach is interesting, but like Lipski's work has too high a complexity to be practically implemented.

Wong introduced statistics to database management of missing information in [Wong 80], and some of his results are tied to Lipski's. Wong's approach involves too much overhead with *a priori* knowledge, and not only includes maybe-results, but also probabilistic results in which every answer to a query may be accompanied by a probability of being correct. This would add too much uncertainty to database management and query processing.

It seems more practical to stick to a stricter interpretation of null values by default, while allowing the user or applications programmer to directly manipulate the null values if he or she wishes. By default, a strict *closed world assumption* [Reiter 78] would be in effect, and the so-called "maybe-results" would be ignored completely, since one is usually interested in the true-results. Therefore, the approach taken herein is to declare the database structure in such a way as to exclude missing information from any and all query results by default, while allowing specific reference to the encoded values representing the three types of missing information if required.

2.4 Justification for DEFINE

There are many different models for database management in the literature. The most widely known models are ones for which commercial applications have been developed. These include the hierarchical model [McGee 77], network model [CODASYL 71], and, of course, Codd's

relational model [Codd 70]. Quite a few models were proposed in the early 1980's, including an object-oriented model [Baroody 81], functional model [Shipman 81], and "semantic data" model [Hammer 81]. More recent developments have incorporated the concept of abstract data types into the relational model [Osborn 86], or used first-order logic [Rybinski 87] or logical deduction [Spyratos 87] as the formal basis for database processing.

The trend in the literature seems to be a move from models concerned with easy implementation and well-defined data structures towards models concerned with flexibility of design and manipulation of abstract data types. Traditional mathematical formalisms such as first-order logic, logical deduction, and set theory seem to be the formal basis for the new wave of database models.

The reason for mentioning the above models and formalisms is that the database declaration/manipulation language DEFINE [Gilmore 87b] incorporates first-order logic and set theory, and has its original formal basis in natural deductive logic [Gilmore 86]. Formal mathematics and proofs are presented in detail in the two references and so will not be duplicated in this thesis. DEFINE is extremely powerful and flexible, in that the specification of the database structure can mimic all of the models mentioned above. In fact, the example database DVS-Ext is declared in DEFINE as a relational database for simplicity and because such a design makes sense given the logical structure of DVS-Ext. See Chapter 4 for a brief description of the syntax and semantics of DEFINE.

Chapter 3

The Example Database DVS-Ext

3.1 General Description

The example database used in this thesis consists of subset of fields extracted from a data set collected, coded, and kept by the Division of Vital Statistics of the Ministry of Health of the Province of British Columbia (DVS). Nosologists employed by DVS encode the information contained on B.C. Death Certificates, and the encoded data is kept on tape for administrative and research purposes.

The DVS data contains all the information recorded on Death Registrations for all persons dying in the Province of British Columbia, beginning in 1950. The author is currently in possession of the extracted data for each year up to and including 1984. The example database is referred to as "DVS-Ext", an abbreviation of "Division of Vital Statistics - Extracted Death Information". Table 3.1 shows the format of each of the data fields of DVS-Ext.

Descriptions of the encoding schemes used for the various fields, as well as the changes in the encoding of the information over the years are detailed in the following sections.

INFORMATION	FORMAT	COMMENTS
Year of Death	STRING(2)	
Gender	STRING(1)	
Age Unit and Age Code	STRING(3)	
Place of Birth	STRING(2)	
Marital Status	STRING(1)	
Residence - Census Division	STRING(3)	N/A after 1971
Residence - Regional District	STRING(3)	N/A before 1972
Residence - Municipality	STRING(3)	
Residence - Census Tract	STRING(2)	
Residence - School District	STRING(3)	
Occupation - Job	STRING(3)	
Occupation - Industry	STRING(3)	
Cause of Death - Primary	STRING(4)	
Cause of Death - Secondary	STRING(4)	
Ethnic or Racial Origin	STRING(2)	N/A after 1973
Native Indian Status	STRING(1)	N/A before 1974
Year of Birth	STRING(2)	
Name of Deceased	STRING(25)	

Table 3.1: Data Format of DVS-Ext

3.2 The *Year of Death* Field

The *year of death* field is simply the last two digits of the actual calendar year in which a person died. The *year of death* field determines which encoding schema should be used to interpret the other fields of the death record for a particular deceased person, and is utilized in DEFINE to refer to machine-determined logical subsets of DVS-Ext for that purpose. It should be noted that if a data set does not include a field containing values via which it is possible to determine which encoding schema is used for other fields, then building a DBMS capable of handling time-variant encoding schemas would be much more difficult.

3.3 The *Gender* Field

Table 3.2 shows the encoding schema used to record the gender (sex) of decedents over the years.

According to DVS records, the nosologists were instructed to stop encoding females using

1950 – 1958	1959 – 1960	1961 – 1976	1977 – 1984
1 = male	1 = male	1 = male	M = male
K = female	K = female	2 = female	F = female
	2 = female		
0 = unknown	0 = unknown	0 = unknown	

Table 3.2: Time-Variant Encoding Scheme of *Gender*

the letter K as of January 1, 1960. However, it appears that in reality, some of the nosologists started encoding females using the digit 2 before January 1, 1960, and other nosologists kept encoding females using the letter K after January 1, 1960. Thus there are two different encodings for females during the registration years 1959 and 1960. Apparently, there are no death records with unknown gender after 1964. The precise reason is unknown, but is presumably due to improved forensic techniques and better investigation of deaths.

3.4 The *Age Unit* and *Age Code* Fields

The first character (***age unit***) indicates the units of the last 2 digits (***age code***). The meaning of the ***age unit*** and implied meanings for ***age code*** are as indicated below in Table 3.3.

Character	1950 – 1964	1965 – 1976	1977 – 1984
0	invalid code	invalid code	years
1	age not stated	years	years over 100
2	years	months	months
3	months	invalid code	invalid code
4	weeks	days	days
5	days	hours	hours
6	hours	minutes	minutes
7	minutes	invalid code	invalid code
–	invalid character	over 99 years	invalid character

Table 3.3: Time-Variant Encoding Scheme of *Age*

For 1950 – 1964, if the first digit is a one (1), then it means that the age at death was unknown, indeterminate or not stated by the physician filing the death registration. The last two digits in this case are always two zeroes (00). Also, an indeterminate number of months, weeks, etc. would be encoded as 300, 400, etc.

For 1965 – 1976, if the age at death was over 99, it was encoded as a dash (–) followed by the number of years over 100. (eg. –03 would mean 103). For 1965 only, any age at death over 99 was encoded as "one dash blank" (1–␣). For 1965 – 1976, if the last two digits are unknown, they are encoded as ampersands (&&) (eg. 1&& means an unknown number of years).

For 1977 – 1984, if the age of the decedent at death was unknown, it was estimated as closely as possible, or was set at 25 years if no reasonable estimate was possible. The *year of birth* field is encoded as "unknown" in such cases.

3.5 The *Place of Birth* Field

The *place of birth* field was encoded according to a standard schema of two-digit codes for various countries or regions of origin. The same schema was also used for recording ethnic or racial origin, with special codes for certain ethnic or racial groups. See Appendix B for a full description of the meanings of individual two-digit codes. The *place of birth* field did not involve any changes in encoding during the time span 1950 – 1984, except for the encoding of unknown place of birth and unknown ethnic or racial origin.

Unknown place of birth was encoded as the number 99, but only prior to 1965. After 1964, unknown birthplace was encoded as ampersands (&&).

Unknown ethnic or racial origin was encoded as two zeroes (00) or two ampersands (&&). DVS stopped recording ethnic or racial origin of deceased persons on December 31, 1973. For database management purposes, either all decedents after 1973 must be regarded as having "not applicable" as an ethnic or racial origin, or the database must somehow be aware that the ethnic or racial origin attribute applies only to records with a *year of death* attribute (when regarded as an integer) which is less than the number 74.

Pre-1957 subdivision	AND	School District Code is	IMPLIES	Post-1956 subdivision
041	.....	032,033	.....	041
041	.....	036,037,043	.....	044
041	.....	034,035,042,075,076	.....	045
042	.....	038,039,040,041	.....	043
042	.....	any	.....	042
056	.....	any	.....	056
055	.....	any	.....	054
054	.....	any	.....	055
053	.....	any	.....	053
052	.....	any	.....	051
051	.....	061,062,063,064	.....	051
051	.....	065,066	.....	052
051	.....	067,068	.....	053

Table 3.4: Correspondence of Pre-1957 and Post-1956 Census Subdivisions

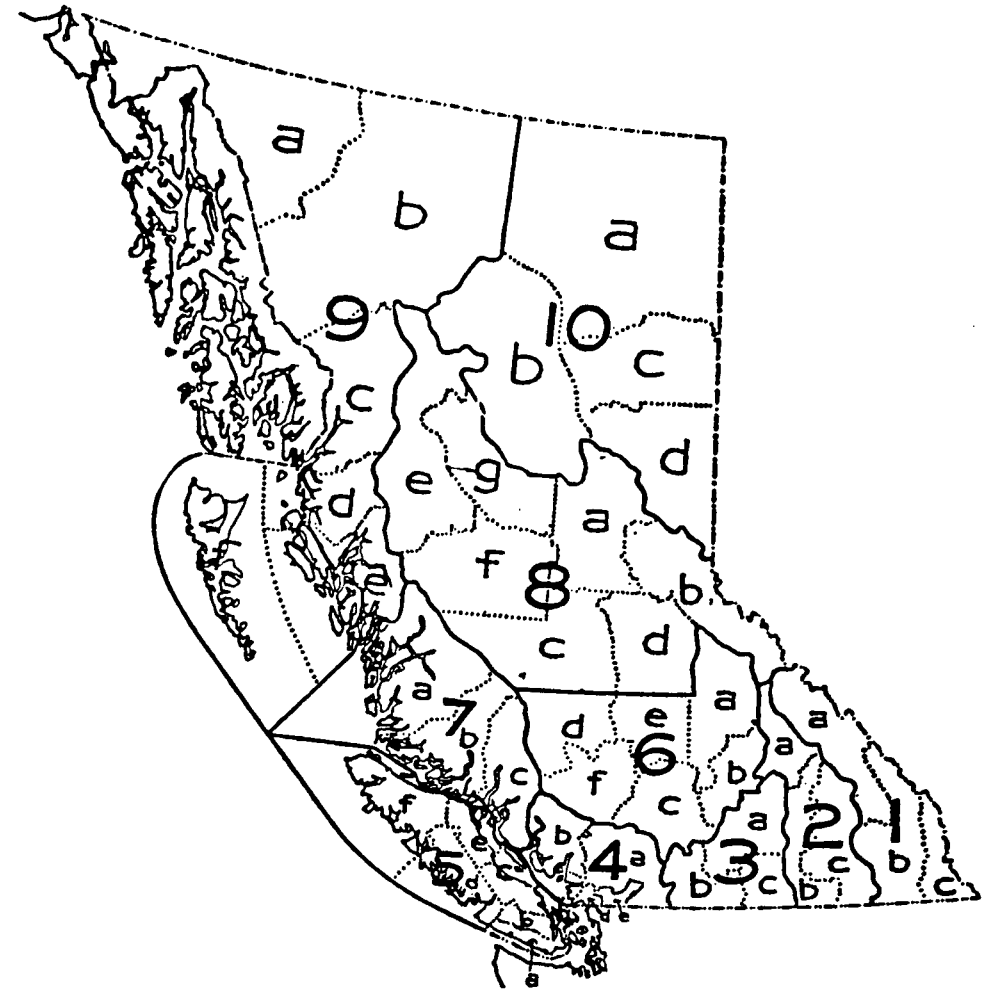
3.6 The *Marital Status* Field

The ***marital status*** field was encoded according to a very simple schema (1 = single, 2 = married, 3 = widowed, 4 = divorced, 5 = separated), which did not change during the years 1950 – 1984, except for the encoding of unknown marital status.

Unknown marital status was encoded as zero (0) prior to 1965 and ampersand (&) from 1965 – 1984.

3.7 The *Residence — Census Division* Field

The Census Divisions of British Columbia existed only prior to 1971. There were 10 major divisions, subdivided into 2 to 7 subdivisions labelled with the letters "a" through "g". For encoding purposes, the letters "a" through "g" were translated to the alphabetic digits 1 through 7 (eg. census subdivision 10c would be encoded as the character string 103). The census subdivisions of the Lower Mainland of British Columbia and Vancouver Island were reorganized after 1956. It is possible to determine which post-1956 census subdivision a person resided in from the pre-1957 census subdivision and the School District code, if it exists. See Table 3.4



CENSUS DIVISIONS 4 AND 5 (DETAIL)

Figure 3.1: Census Divisions — 1957 – 1971

for the implied correspondence between the pre-1957 and post-1956 census subdivisions.

Codes 000, 999, and blanks (␣␣␣) mean unknown census division. The Census Divisions of B.C. were phased out and replaced by Regional Districts after 1971. See Figure 3.1 for a map indicating the geographical distribution of Census Divisions in B.C. during the time period 1957 – 1971. Due to the totally different boundaries chosen for Regional Districts, it is impossible to find any direct or indirect correspondence between Census Divisions and Regional Districts. There is also no logical correspondence between Census Divisions and School Districts. Thus it is not possible to approximate pre-1971 Census Divisions by referring to Regional Districts or School Districts.

Since Census Divisions ceased to exist after 1971, the *Census Division* field is "not applicable" to decedents with a year of death greater than the integer 71.

3.8 The *Residence — Regional District* Field

The Regional Districts of B.C. were set up on January 1, 1971 for the gathering census population figures for Statistics Canada. The Regional Districts completely replaced the old Census Divisions. There is a good correspondence between Regional Districts and certain groups of School Districts, so that it is possible to approximate Regional Districts prior to 1971 by grouping together School Districts as indicated in Table 3.5.

Non-residents of B.C. are recorded as whatever code applied to their particular place of permanent residence at the time of death. The only way to tell which deaths are of non-residents of B.C. is by noticing that the Regional District code is not in the range 001 – 029, or alternatively by noticing that the School District code is 099.

Unknown Regional District is encoded as 000, 999, blanks (␣␣␣), or ampersands (&&&). See Figure 3.2 for the geographical distribution of Regional Districts in B.C.

Figure 3.2: Regional Districts — 1972 - 1984

Regional District IS COMPOSED OF	School Districts
001 = Alberni-Clayoquot	070
002 = Bulkley-Nechako	054,055,056
003 = Capital	061,062,063,064
004 = Cariboo	027,028
005 = Central Fraser Valley	034,035
006 = Central Kootenay	007,009,010,086
007 = Central Okanagan	023
008 = Columbia-Shuswap	018,019,089
009 = Comox-Strathcona	071,072,084
010 = Cowichan Valley	065,066
011 = Dewdney-Alouette	042,075
012 = East Kootenay	001,002,003,004
013 = Fraser-Cheam	032,033,076
014 = Fraser-Fort George	057
015 = Greater Vancouver	036,037,38,039,040,041,043,044,045
016 = Kitimat-Stikine	080,088
017 = Kootenay Boundary	011,012,013
018 = Mount Waddington	085
019 = Nanaimo	068,069
020 = North Okanagan	021,022
021 = Central Coast (Ocean Falls)	049
022 = Okanagan-Similkameen	014,015,016,017,077
023 = Peace River-Liard	059,060,081
024 = Powell River	047
025 = Skeena-Queen Charlotte	050,052
026 = Squamish-Lillooet	029,048
027 = Stikine	087
028 = Sunshine Coast	046
029 = Thompson-Nicola	024,026,030,031

Table 3.5: Regional Districts and their correspondence with School Districts

3.9 The *Residence — Municipality* Field

Municipality was encoded according to a standard schema, as detailed in Appendix C. There are many thousands of deceased persons who did not reside within any municipality, and are encoded as 000 or blanks (␣␣␣) prior to 1965, and to "dash blank blank" (¬␣␣) after 1964. These records have "not applicable" as their municipality code.

3.10 The *Residence — Census Tract* Field

Unknown census tract is coded as 00 prior to 1965, and to ampersands (&&) after 1964. There are tens of thousands of records with *Residence — Census Tract* encoded as blanks (␣␣), which indicates that census tract is "not applicable" for those deaths. Census tracts are small divisions within large municipalities, and thus it is neither possible nor relevant to attach any specific meaning to the individual encoded numbers.

3.11 The *Residence — School District* Field

The school districts as they have existed since 1979 are as shown in Table 3.6. See Figure 3.3 for the geographical distribution of School Districts in B.C. as of 1979.

DVS continues to encode deaths as occuring in School District 05 (Creston) or 06 (Kaslo) even though those two districts were merged by the Ministry of Education to become School District 86 (Creston-Kaslo) in 1966. DVS also encodes deaths occuring in School District 92 (Nishga) to School District 88 (Terrace) even though Nishga was made a separate school district in 1978.

Unknowns are encoded as "dash blank blank" (¬␣␣), "dash dash blank" (—␣), or three blanks (␣␣␣). There have been several changes to school district boundaries during the years 1950 to 1975. Sometimes two districts were merged together, and sometimes a district was split into two. In order to reconcile historical school districts with those now in existance, the reclassifications shown in Table 3.7 have been made, sometimes using the municipality code, if it exists.

It is also possible to obtain a valid current School District code from records whose actual value in the *Residence — School District* field is "not applicable" (during 1950 – 1975 there were areas of B.C. that were not part of any official School District) by using the Municipality, Census Division, or Regional District codes, as shown in Table 3.8.

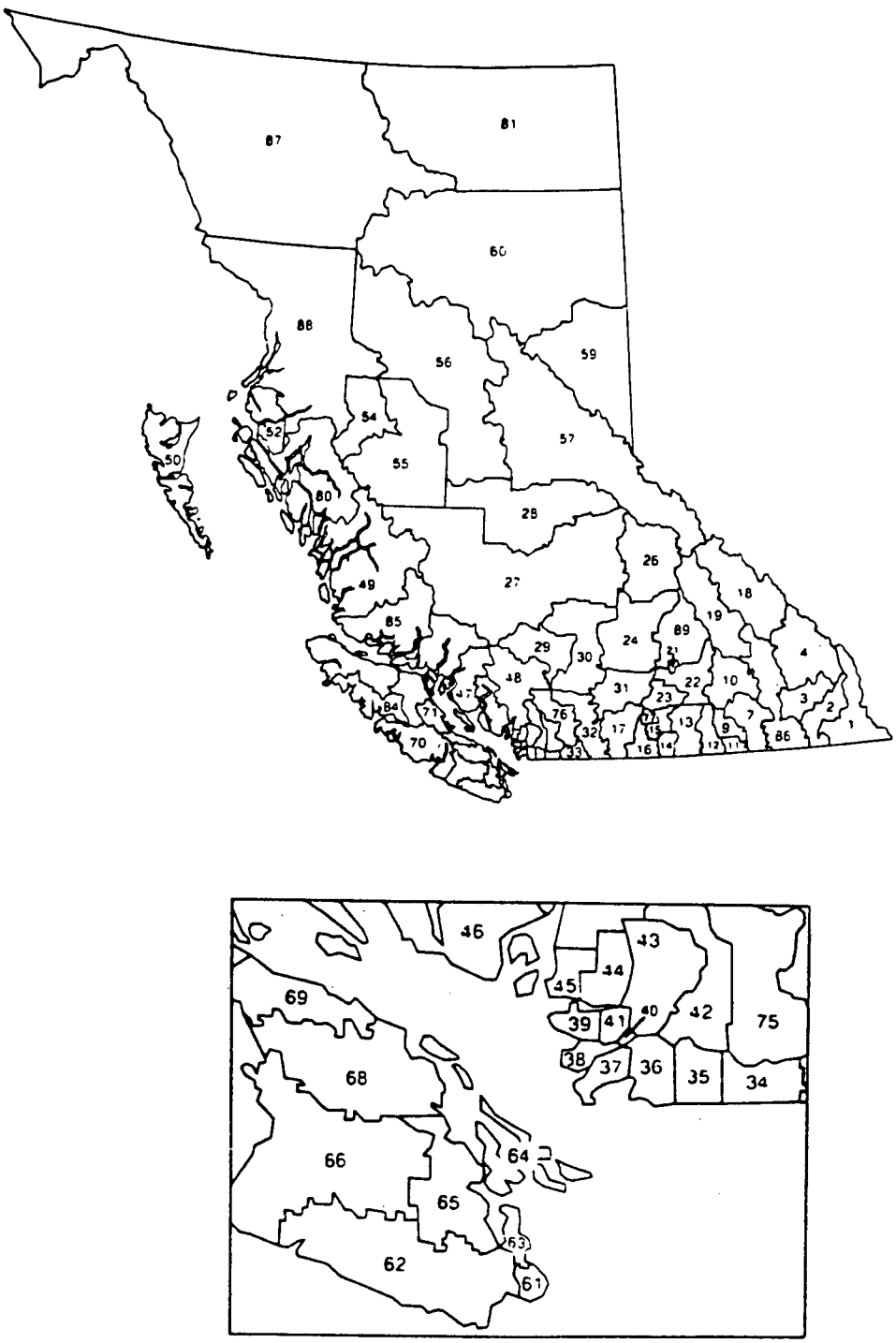


Figure 3.3: School Districts — 1979 – 1984

01 = Fernie	31 = Merritt	59 = Peace River South
02 = Cranbrook	32 = Hope	60 = Peace River North
03 = Kimberley	33 = Chilliwack	61 = Greater Victoria
04 = Windermere	34 = Abbotsford	62 = Sooke
07 = Nelson	35 = Langley	63 = Saanich
09 = Castlegar	36 = Surrey	64 = Gulf Islands
10 = Arrow Lakes	37 = Delta	65 = Cowichan
11 = Trail	38 = Richmond	66 = Lake Cowichan
12 = Grand Forks	39 = Vancouver	68 = Nanaimo
13 = Kettle Valley	40 = New Westminster	69 = Qualicum
14 = South Okanagan	41 = Burnaby	70 = Alberni
15 = Penticton	42 = Maple Ridge	71 = Courtenay
16 = Keremeos	43 = Coquitlam	72 = Campbell River
17 = Princeton	44 = North Vancouver	75 = Mission
18 = Golden	45 = West Vancouver	76 = Agassiz-Harrison
19 = Revelstoke	46 = Sunshine Coast	77 = Summerland
21 = Armstrong-Spallumcheen	47 = Powell River	80 = Kitimat
22 = Vernon	48 = Howe Sound	81 = Fort Nelson
23 = Central Okanagan	49 = Central Coast	84 = Vancouver Island West
24 = Kamloops	50 = Queen Charlotte	85 = Vancouver Island North
26 = North Thompson	52 = Prince Rupert	86 = Creston-Kaslo
27 = Cariboo-Chilcotin	54 = Smithers	87 = Stikine
28 = Quesnel	55 = Burns Lake	88 = Terrace
29 = Lillooet	56 = Nechako	89 = Shuswap
30 = South Cariboo	57 = Prince George	92 = Nishga

Table 3.6: Encoding of the *Residence — School District* Field

3.12 The *Occupation — Job* Field

The *Occupation — Job* field contains the occupational job code, which is a three-digit number from 000 to 999 with unknowns encoded as 999 or "dash blank blank" (¬␣␣) prior to 1965 and to 999 or three blanks (␣␣␣) after 1964. There are several hundred "invalid" job codes (codes that do not exist in the manual in use during the applicable time period) in the data.

Two different encoding schemas were used. The 1951 Canadian Occupational Manual (COM) [DBS 51] was used from 1950 - 1964, and the 1961 Canadian Occupational Manual [DBS 61] was used from 1965 - 1984.

In most cases it is possible to directly translate the 1951 encoding schema to the 1961 schema. However, there are a large number of occupations for which there was no separate

Historical School District Code	AND	Municipality Code is	IMPLIES	Present School District Code
005 (1950,1984), 006 (1950,1984)		any		086
020 (1950,1968), 078 (1950,1968)		any		089
051 (1950,1967), 053 (1950,1968)		any		088
073 (1950,1964), 074 (1950,1964)		any		085
025 (1950,1970)		any		024
058 (1950,1971)		any		057
079 (1950,1970)		any		070
082 (1956,1972)		any		027
083 (1963,1972)		any		060
067 (1950,1973)		015		068 (54%)
067 (1950,1973)		053		065 (46%)
008 (1950,1970)		029		007 (72%)
008 (1950,1970)		077,080		010 (28%)

The year ranges show the first and last years that DVS used each code.

Table 3.7: Historical Reconciliation of School Districts in B.C.

code in the 1951 manual. There are also a few occupations in the 1961 manual that did not exist in 1951. Thus it is not completely possible to directly translate from the earlier coding schema to the more recent schema. Translating from the 1961 codes to the 1951 codes would be easier, but less useful, meaningful, and current.

Therefore, it is more useful to group both sets of codes to another, less cumbersome set of encodings which comprise 206 of the most meaningful broad occupational groups in B.C. See Appendix D for the correspondence between the two official encoding schemas and the 206 groups, some of which are combinations of others.

3.13 The *Occupation — Industry* Field

The *Occupation — Industry* field contains the occupational industry code, which is a three-digit number from 000 to 999 with "unknown" encoded as 999 throughout the entire time span 1950 – 1984, and "not applicable" being encoded as "dash blank blank" (-␣␣) prior to 1965, and to three blanks (␣␣␣) after 1964. Two different encoding schemas were used, the 1948 Standard Industrial Classification Manual from 1950 – 1964 [DBS 48], and the 1960 version of

School District code = unknown AND	IMPLIES	Present School District
Municipality = 131		039
Municipality = 178		084
Municipality = 084		070
Municipality = 069		080
Census Division = 095		080
Census Division = 054,055		070
Census Division = 056		085
Census Division = 091,092		087
Census Division = 071		049
Census Division = 101		081
Census Division = 087,102		056
Census Division = 072,073		072
Census Division = 064		027
Census Division = 062		024
Census Division = 042,043		039
Census Division = 094		052
Census Division = 053		068
Census Division = 083		027
Census Division = 086 & Municipality = 071		055
Census Division = 086 & Municipality = 085		056
Regional District = 004		027

Table 3.8: Assignment of School District from other information

the same manual [DBS 60] from 1965 – 1984. The author has not yet attempted to find any correspondence between the two encoding schemas, nor has any attempt been made to translate both sets of codes to a third set of more relevant groupings.

3.14 The *Cause of Death — Primary* Field

The *Cause of Death — Primary* field has been encoded according to the International Classification of Diseases (ICD), versions 6, 7, 8 and 9. ICD-6 [WHO 48] and ICD-7 [WHO 57] are virtually identical, so it is possible to ignore the extremely few differences between them, and it is normal to consider ICD-6 and ICD-7 to be a single "version". ICD-6 and ICD-7 were used from 1950 – 1968, ICD-8 [NCHS 68] from 1969 – 1978, and ICD-9 [NCHS 78] from 1979 – 1984.

There are a small number of records with an "invalid" primary cause of death, in the sense that either the recorded code does not exist in the ICD manual used during the applicable time period, or the recorded cause of death contradicts the gender or age of the decedent (assuming that the gender and age information is correct). For example, a female person cannot die from cancer of the prostate gland, nor can an 85-year-old person die of complications of pregnancy or childbirth.

The ICD codes are three- or four-digit numbers recorded in an alpha-numeric format. The fourth digit is more precise than just three digits, but the physician completing the death certificate does not always specify the cause of death with enough precision to allow the nosologist to encode the fourth digit. In these cases, the fourth "digit" is recorded instead as a blank (␣), or a dash (–). The actual ICD codes as shown in the references have decimal points between the third and fourth digits, but for recording purposes DVS has simply ignored the decimal points.

Due to major revisions of the ICD stemming from increased knowledge of particular medical specialties, it is not possible to directly translate between the different versions of the ICD. A large percentage of the alpha-numeric codes from any one of the versions *do* have a direct correspondence with codes from the other two versions. However, some of the remaining codes are so vastly different in meaning that no correspondence is possible at the level of individual codes.

Since it is not possible to directly translate between the different versions of the ICD, and since dealing with eleven thousand different codes is unwieldy, it is useful to group the three different encoding schemas into another, less cumbersome set of codes. Using broad groupings of major causes of death of interest, it is possible to translate the three versions of the ICD into 165 of the most relevant groupings of causes of death. See Appendix E for the correspondence between the three versions of the ICD and the 165 groups, some of which are combinations of others.

3.15 The *Cause of Death — Secondary* Field

The *Cause of Death — Secondary* field has been coded similarly to the *Cause of Death — Primary* field. The difference between the encoding schemas for primary and secondary causes of death is an entirely different meaning to the codes having 8 and 9 as their first digit. These ranges of codes specify death from accidents, poisoning, and injuries. The primary cause of death indicates the external cause, and the secondary cause of death indicates the internal cause. (eg. the external cause may be "car accident" while the internal cause is "cerebral haemorrhage").

3.16 The *Ethnic or Racial Origin* Field

The *Ethnic or Racial Origin* field has been encoded according to the same schema as the encoding schema used to record the place of birth (see Appendix B), with differences for race as noted. The code for racial origin can be used to identify Registered Native Indians prior to 1974, who are encoded using the character string 15. Due to pressure from minority groups, DVS stopped collecting and encoding ethnic or racial origin on December 31, 1973. Therefore, records with *year of death* (when regarded as an integer) greater than 73 have "not applicable" as an ethnic or racial origin.

3.17 The *Native Indian Status* Field

The *Native Indian Status* field has been used to identify Registered Native Indians after 1973 (see Table 3.9).

There are several thousand "status not stated" records during the years 1974 – 1976. In consultation with DVS, the Native Indian status of these records is regarded as being "not Native Indian". The reason is that the nosologists were encoding strictly by what was recorded on the death certificate, but physicians were leaving the question about Native Indian status

Character	1974 – 1976	1975 – 1984
blank	invalid code	not Native Indian
0	not Native Indian	invalid code
1	Native Indian	invalid code
2	invalid code	Native Indian
&	status not stated	invalid character

Table 3.9: Time-Variant Encoding Schema of *Native Indian Status*

blank instead of filling in the word "NO" when the decedent was not a Registered Native Indian.

After ceasing to record ethnic or racial origin, the government was still interested in keeping statistics on the number and causes of deaths among Registered Native Indians for the federal government Department of Indian Affairs. Therefore, the new field of *Native Indian Status* began to be recorded only after 1973. Prior to 1974, the information supplied by the *Native Indian Status* field can be inferred from the *Ethnic or Racial Origin* field.

3.18 The *Year of Birth* Field

The *year of birth* field contains the last two digits of the year of birth of the decedent. Unknowns are coded 00 prior to 1958, and to two blanks (␣␣) after 1958. During 1958, both 00 and ␣␣ were used to mean "unknown". Sometimes 00 is a valid code prior to 1959, meaning the person was born in the year 1900. The only way to tell if a code 00 is valid or not is to cross-reference the year of death, the age at death, and the year of birth. Sometimes even the cross-reference does not yield a definite answer, in which case the year of birth must be regarded as "unknown".

Chapter 4

Solution of Problem using DEFINE

4.1 Brief Description of DEFINE

DEFINE is an experimental language for the declaration, manipulation and querying of databases (a DBMS), conceived and under development by Professor P.C. Gilmore of the University of British Columbia [Gilmore 87b].

The key to DEFINE is that it is based on set theory. One declares (defines) sets of entities with common properties, sets of constants to describe those properties, and sets of pairs (or tuples) of entities to describe an association between two (or more) entities from different sets (for example, to describe the fact that entity a from set A has a property described by constant b from set B).

The purpose in developing DEFINE is to provide a single, powerful database language with which it is possible to completely declare the structure of the database (as determined by an information needs analysis), and with which it is also possible to manipulate and query the database. DEFINE provides a precise and easily understandable syntax (based on first-order logic) to accomplish these goals.

The power of DEFINE is that by defining the sets in an appropriate manner, it is possible to handle time-variant encodings and missing information in a straightforward and even simple way. By embedding the interpretation specifications for the problem data into the DEFINE

declarations of the sets, we achieve the desired result of being able to remove the chore of interpretation from the minds of users and applications programmers.

A complete description of the theory, syntax, and semantics of DEFINE can be found in the reference. In order to understand the examples and specifications given in this thesis, it is only necessary to provide a brief summary description of the syntax, and some of the semantics of DEFINE.

4.2 Elementary Syntax and Semantics of DEFINE

The general syntax[1] of DEFINE is given by Figure 4.1 below, where the square brackets indicate optional syntax.

Declare: SETNAME
Select: x:DOMAIN(SETNAME)
Where: [*Define*: INTENSION(SETNAME)] [(comment)]
[*Degrees*: <LL, LU>, <RL, RU>]

Figure 4.1: General form of a DEFINE declaration

Semantically, the syntax of the formula given in Figure 4.1 defines the set SETNAME to be the set of all x which are members of the domain of SETNAME such that the INTENSION of SETNAME is satisfied.

The variable x can represent a single entity variable, or it can represent a tuple of the form $< x_1, x_2, \ldots x_n >$. DOMAIN(SETNAME) can be a single setname (if x is a tuple consisting of only one element) or a tuple of the form <SETNAME_1,SETNAME_2, ... SETNAME_n > specifying the names of the sets of which $x_1, x_2, \ldots x_n$ are members.

If SETNAME represents a set of fundamental entities of interest, DOMAIN(SETNAME) is simply SETNAME. These sets are referred to as *primitive* sets.

[1]Dr. Gilmore has recently altered the syntax of DEFINE. The semantics remain unchanged. See Appendix A for a brief description of the new, less cumbersome syntax.

The syntax INTENSION(SETNAME) specifies which members of domain are included as members of SETNAME. In the case of sets for which membership can only be determined by a human being, only the comment is necessary. These sets are referred to as ***base*** sets, which include primitive sets.

In all other cases, the intension is written in a modified first-order logic which is machine-interpretable. The first-order logic includes ***set terms*** like that in the *Select* clause in Figure 4.1, the usual Boolean connectives **and**, **or**, and **not**, and the universal and existential quantifiers [**For some** {*set term*}] and [**for all** {*set term*}]. The usual arithmetic comparison operators are allowed (">" (greater than), "<" (less than), "=" (equals), "¬" (not), etc.).

The degrees specify integrity constraints on associations, which are sets where x is a pair $< u, v >$ and DOMAIN(SETNAME) is a pair <SETNAME_1, SETNAME_2 >. The degrees are <left lower, left upper>, <right lower, right upper>, where "left" and "right" refer, respectively, to SETNAME_1 or SETNAME_2. The adjectives "lower" and "upper" specify, respectively, the minimum and maximum number of members from one set that can be associated with a member of the other set. It is only necessary to know whether the lower degrees are 0 (zero) or 1 (one), and whether the upper degrees are 1 (one) or * (asterisk, meaning one or more). If the left lower degree of the association is 0, then SETNAME_1 is ***partial*** on SETNAME_2, meaning that *not* every member of SETNAME_1 has a corresponding member of SETNAME_2 associated with it. If the left lower degree is 1, then SETNAME_1 is *total* on SETNAME_2, meaning that every member of SETNAME_1 is associated with at least one member of SETNAME_2. A left upper degree of 1 means that the association is ***single-valued***, and a left upper degree of * means that the association is ***multi-valued***. An association that is both total and single-valued is ***functional***. An ***attribute*** is an association whose left set is a value set.

4.3 The DEFINE Data Definition of DVS-Ext

If one were to start from before any data were collected, the data declarations for DVS-Ext would undoubtedly treat occupations, causes of death, school districts, etc. as separate entity

sets with attributes of their own (such as the name and code for each school district or cause of death). Then the database would contain (for example) the entire ICD-7 manual. This would mean the the causes of death would be explicitly defined by ICD-7, and the future change to ICD-8 would necessitate major changes to the database structure.

The changes to the encoding schemas over the years were largely beyond the control of DVS. The ICD codes (for example) are revised every ten years by an international committee of prominent and expert physicians. The definition of Census Divisions, and the change from Census Divisions to Regional Districts was the responsibility of the federal government. Even changes that were under the control of the provincial government, such as School District definitions, were done in a way that made sense bureaucratically and administratively to the Ministry of Education. The Ministry of Education could not worry about what effect their changes might have to the collecting of death registration data.

The exact format of death registration data was of very little importance in comparison to other government priorities. As long as the data were recorded and DVS was able to produce the required statistics in some way, the government was unconcerned about the difficulty in dealing with encoding changes. The mandate of DVS is to provide statistics about subsets of deceased persons (eg. persons employed as welders, dying of lung cancer, residing in the Prince George School District, etc.). DVS is not particularly interested in (for example) School Districts as entities, but only as a way of describing a subset of death registrations.

Since it was not the responsibility of DVS to keep track of ICD codes or School District definitions, but only to record death registrations, it makes sense to treat each of the fields of Table 3.1 as simple attributes of the single entity set DECEASED (referring to persons), recording only the codes. Then the School Districts, causes of death, etc. can be declared ***implicitly*** (as defined sets) in a logical manner from the codes. When future changes in encoding or physical definitions of geographic subdivisions of B.C. occur, only the implicit declarations need be changed, and not the actual structure of the database. Also, this method declares (for example) School Districts as subsets of DECEASED, which is exactly the way in which DVS is interested in School Districts.

DECEASED	DECEASED	People who died in B.C.
VYR	STRING(2)	*Define*: (value set for year codes)
DTHYR	DECEASED × VYR	A person had one year of death
VREGNO	STRING(6)	*Define*: (value set for death reg. number)
REGNO	DECEASED × VREGNO	A person had one death registration number
VGENDER	STRING(1)	*Define*: (value set for gender codes)
GENDER	DECEASED × VGENDER	A person had one gender
VAGUNIT	STRING(1)	*Define*: (value set for units of age code)
AGEUNIT	DECEASED × VAGUNIT	A person had one age unit
VAGEDTH	STRING(2)	*Define*: (value set for age at death code)
AGEDTH	DECEASED × VAGEDTH	A person had one age at death
VORIGIN	STRING(2)	*Define*: (value set for place of birth and/or ethnic/racial origin codes)
POB	DECEASED × VORIGIN	A person had one place of birth
VMARST	STRING(1)	*Define*: (value set for marital status)
MARST	DECEASED × VMARST	A person had one marital status
VCENDIV	STRING(3)	*Define*: (value set for Census Division codes)
CENDIV	DECEASED × VCENDIV	A person resided in one Census Division
VREGDIS	STRING(3)	*Define*: (value set for Regional District codes)
REGDIS	DECEASED × VREGDIS	A person resided in one Regional District
VMUNIC	STRING(3)	*Define*: (value set for Municipality codes)
MUNICIP	DECEASED × VMUNIC	A person resided in one Municipality
VCTRACT	STRING(2)	*Define*: (value set for Census Tract codes)
CTRACT	DECEASED × VCTRACT	A person resided in one Census Tract
VSCHDIS	STRING(3)	*Define*: (value set for School District codes)
SCHDIS	DECEASED × VSCHDIS	A person resided in one School District
VJOB	STRING(3)	*Define*: (value set for COM job codes)
JOB	DECEASED × VJOB	A person had one usual lifetime job
VINDUST	STRING(3)	*Define*: (value set for COM industry codes)
INDUST	DECEASED × VINDUST	A person had one usual lifetime industry
VCAUSE	STRING(4)	*Define*: (value set for ICD codes)
PRIMCOD	DECEASED × VCAUSE	A person had one primary cause of death
SECOCOD	DECEASED × VCAUSE	A person had one secondary cause of death
ETHRACE	DECEASED × VORIGIN	A person had one ethnic/racial origin
VNIS	STRING(1)	*Define*: (value set for Native Indian status)
NATIVE	DECEASED × VNIS	A person had one Native Indian status
BRTHYR	DECEASED × VYR	A person had one year of birth
VNAME	STRING(25)	*Define*: (value set for names of people)
NAME	DECEASED × VNAME	A person had one name

Table 4.1: Declared Set Table for DVS-Ext

The entity set DECEASED and all its attributes are given in Table 4.1. Normally a Declared Set Table would be accompanied by an Association Table showing the degrees of the associations, but in the case of DVS-Ext, all associations (attributes are associations between entity sets and value sets) have been made total, adding "not applicable" values if necessary. Note that by having a single entity set with only total attributes, the relational database model (with a single relation) has been mimicked by DEFINE.

Some of the attributes of DVS-Ext (eg. occupations, causes of death, school districts, etc.) could be declared as entity sets. Then it would be possible to restrict the associations between those sets and the set DECEASED to only those entities to which an association applies, thereby eliminating some of the "not applicable" values necessitated by the declaration of a single entity set with total attributes. One of the objectives of this thesis, however, is to process "not applicable" null values. Also, the declaration of a single entity set with all total attributes is sensible with respect to DVS-Ext, and shows how DEFINE can mimic the relational model. This thesis will therefore proceed with the declaration of DVS-Ext as a single entity set with all total attributes (a single relation), and other possible methods of declaring DVS-Ext will not be considered.

4.4 Structural Declarations

Only three types of sets are necessary for the structural declaration of DVS-Ext as shown in table 4.1. One example of each type of declaration will be given. The remaining declarations are quite simple alterations of the examples, and will be left as an exercise for the reader.

The only primitive set is the set DECEASED, which would be declared very simply, as shown in Figure 4.2.

All of the value sets are strings of various lengths, so a typical example would be the value set VNAME, declared as in Figure 4.3.

In the intension of VNAME, the construct $variable_1$:ASSOC:$variable_2$ is an alternative to

Declare: DECEASED
Select: x:DECEASED
Where: (All persons who died in British Columbia).

Figure 4.2: The primitive entity set DECEASED.

Declare: VNAME
Select: x:STRING
Where: *Define*: $\{x\text{:L:}\} \leq 25$
(A value set for names).

Figure 4.3: An example of a value set declaration.

the construct $< variable_1, variable_2 >$:ASSOC for the association ASSOC between the sets represented by $variable_1$ and $variable_2$. The construct $\{variable\text{:ASSOC:}\}$ is an example of a *parameterized* set name (see [Gilmore 87b] for full details). It is sufficient to state that the intension of VNAME is a restriction of the value set to strings of length (the L in the example is a built-in length function, an association between a string and an integer giving its length) less than or equal to 25 characters.

A typical attribute association is NAME, which associates an entity from the set DECEASED with a value from the set VNAME. The declaration as shown in Figure 4.4 has an alternative *Select* clause, namely x:DECEASED, y:VNAME.

Declare: NAME
Select: $< x, y >$:<DECEASED,VNAME>
Where: (A person x had one name y)
Degrees: $< 1,1 >, < 0,* >$.

Figure 4.4: A typical declaration of an attribute association.

In the declaration of NAME, the lower degrees indicate that the association is total and single-valued. The upper degrees indicate that the domain of names is not exhausted by the entity set, and any number of people may have had the same name.

4.5 Simple Subsets

Some subsets of DECEASED are very simply specified by direct reference to the actual code recorded in the field. For example, Figure 4.5 shows how the set consisting of all persons dying during the time span 1950 – 1960 can be specified.

Declare: DTHS50–60
Select: x:DECEASED
Where: *Define*: $< x, y >$:DTHYR **and** $y \geq$ '50' **and** $y \leq$ '60'
(All deaths between 1950 and 1960, inclusive).

Figure 4.5: A subset of DECEASED from a year range.

Any logical subset consisting of year ranges can be declared in a similar manner, by simply changing a few numbers in the declaration. Assume that declared subsets of DECEASED exist for any year range that might be desired, the name of the set being DTHS??-?? with the question marks replaced by the year range of interest. The logical declaration of subsets via the codes from the attributes of DECEASED simplifies the handling of time-variant encoding schemas. For example, Figure 4.6 shows how the subset of DECEASED consisting of all females would be specified in DEFINE, by referring to the appropriate code during the appropriate time period. By declaring logical sets using time-variant encodings, logical consistency is produced from physical inconsistency.

Declare: FEMALE
Select: x:DECEASED
Where: *Define*: $< x, y >$:GENDER **and** ((x:DTHS50-76 **and**
($y =$ 'K' **or** $y =$ '2')) **or** (x:DTHS77-84 **and** $y =$ 'F'))
(All female decedents).

Figure 4.6: The *logical* set of all female decedents.

The subset of DECEASED consisting of all males would be specified simply by replacing the fragment ($y =$ 'K' **or** $y =$ '2') with $y =$ '1' and the fragment $y =$ 'F' with $y =$ 'M' in Figure 4.6.

With regard to missing information in the GENDER field, there is no such thing as a person for whom gender is not applicable, but there are some decedents with unknown gender. Those decedents with unknown gender would count as both (**not** FEMALE) and (**not** MALE) by default, unless some specific reference is made to the "unknown" (Type 1) null code, which happens to be the digital character zero ('0') for the domain of the *gender* attribute. The next section indicates how such default or other special processing might occur.

4.6 Handling Missing Information with DEFINE

DEFINE is still in the design stage, and has not yet been implemented. No specific formal method of handling missing information has been included. However, because of the flexibility and power of DEFINE, it is capable of dealing with missing information in a logical manner as is. Let UNK(SETNAME) be the set of all entities with "unknown" values for SETNAME, where UNK is valid only for attributes. Similarly, let INV(SETNAME) be the set of all entities with "invalid" values for SETNAME, and N/A(SETNAME) be the set of all entities with "not applicable" values for SETNAME, where INV and N/A are valid only for attributes. When DEFINE is implemented, the intensions of all queries would be checked for value sets for which one or more of an UNK set, an INV set, or an N/A set are declared. The intension would then be expanded (by default) to include those of the fragments

... (**and not** *var*:UNK(SETNAME)) ... or

... (**and not** *var*:INV(SETNAME)) ... or

... (**and not** *var*:N/A(SETNAME)) ...

that are relevant (where *var* is the variable associated with the set SETNAME).

The processing of database operations (such as joins) would also exclude all entities belonging in any of the three missing information sets by default. Similarly, the computation of functions (such as averages) would exclude entities with declared missing information.

The treatment of missing information described above has no higher complexity than any set declaration. For a given query, it is only necessary to decide whether each entity satisfies the first-order logic assertion. If an assertion contains a reference to an attribute for which one or more of the three missing information sets has been declared, only up to three more checks need be made. In contrast, the approaches given in [Lipski 81] and [Vassiliou 79] require complicated processing of the query into various special forms, and/or comparisons against or substitutions from the entire attribute domain. Proofs are given in [Gilmore 87b] to show that the first-order logic of DEFINE is decidable with a short sequence of expansions.

If a user or applications programmer wished to treat any of the three types of missing information in another way, it can be easily done by referring to the sets UNK, INV, or N/A. Direct reference to the individual codes whose semantic content are one of the three types of missing information is also possible.

For example, the DEFINE specification of DVS-Ext could include a declaration similar to that shown in Figure 4.7 to specify the subset of DECEASED with Type 1 ("unknown") null values in the *gender* field.

Declare: UNK(GENDER)
Select: x:DECEASED
Where: *Define*: $< x, y >$:GENDER **and** x:DTHS50-76 **and** $y =$ '0'
(Decedents with unknown gender).

Figure 4.7: A simple UNK set.

Then, if the intension of some set includes the fragment

... **and not** x:FEMALE ...

then the expanded intension would include the additional fragment

... (**and not** x:UNK(GENDER)) ... .

With this use of DEFINE, it is possible to decide, for each declared set, application, or query, whether to include or exclude missing information. If the set UNK(GENDER) were *not*

declared, then the set (**not** x:FEMALE) would include decedents with unknown gender, and if UNK(GENDER) *were* declared, then the set (**not** x:FEMALE) would *not* include decedents with unknown gender. Suppose UNK(GENDER) were *not* declared. Then it is possible to declare an attribute GENDER.DEFINITE, the set UNK(GENDER.DEFINITE) and the set FEMALE.DEFINITE so that decedents of unknown gender would be included in the set (**not** x:FEMALE) but *not* in the set (**not** x:FEMALE.DEFINITE).

Declaring DEFINITE sets for every attribute would mean declaring a substantial number of sets. Normally, though, one would always want to exclude UNK values, INV values and N/A values from negations. One obvious exception for DVS-Ext would be in the case of the NATIVE attribute, where those decedents with "unknown" Native Indian status during the time span 1974 – 1976 should be considered as not being Registered Native Indians. Other exceptional handling of missing information would include the cases of those decedents with "invalid" job codes, which have been grouped into a separate category among the 206 occupational groups, as descibed in Appendix D, and the assignment of a valid School District code from other information when the actual School District code is "unknown", as described in Section 3.11. These latter two anomalies along with other special handling of missing information, will be discussed in later sections.

The example above of treating decedents with "unknown" Native Indian status as not being Native Indians is an instance of ***attribution by default.*** Attribution by default occurs when an assumption is desired to be made to associate a certain attribute value to a certain entity under certain circumstances. In the case of all three types of missing information, attribution by default could occur if it is reasonable to assume a valid attribute value when there is a lack of information to the contrary. A formal presentation of attribution by default is available in [Gilmore 87a], where a method of declaration is shown that results in intuitively "proper" treatment of queries, most notably those asking for negative information. The method involves comparison with each valid value from the domain, so the complexity would be unacceptably high for large domains. In the following section, attribution by default will only be mentioned in cases where its use is reasonable in terms of size of domain and semantic interpretation.

For the declarations of DVS-Ext, assume that all attributes have UNK, INV and/or N/A sets declared in some form, using the "unknown", "invalid" and "not applicable" values and relevant year ranges as detailed in Chapter 3. The sets UNK, INV, and N/A would be declared analogously to the declaration of UNK(GENDER) above, so that missing information is by default *not* included in any negations. Any special or different treatment of the codes representing any of the three types of missing information will be specified in the appropriate places in the following section.

4.7 Examples of DEFINE Set Declarations for DVS-Ext

Logical subsets for values of GENDER and ranges of values of DTHYR were declared earlier in this chapter. Logical subsets for the values of MARST, or any range of values of BRTHYR would be simple to declare. One example of more complicated subsets would be to declare age group subsets. If the exact age at death were recorded prior to 1977 when the person was over age 99, it would be possible to compute a new attribute that would be the exact age at death, and would avoid the complicated encoding changes involved in the *age unit* and *age code* fields of DVS-Ext. Since such detail is unavailable, the best that can be done is to declare deaths with age at death 99 or over in separate group, as indicated in Figure 4.8.

Declare: 99&UP
Select: x:DECEASED
Where: *Define*: $< x, y >$:AGEUNIT and $< x, z >$:AGEDTH and
($(x$:DTHS50-64 and $y =$ '2' and $z =$ '99') or
(x:DTHS65-65 and $y =$ '1' and $z =$ '$\neg\sqcup$') or
(x:DTHS66-76 and ($y =$ '–' or ($y =$ '1' and $z =$ '99'))) or
(x:DTHS77-84 and ($y =$ '1' or ($y =$ '0' and $z =$ '99'))))
(Decedents who died at age 99 or over).

Figure 4.8: A logical subset of long-lived decedents.

Most other age groups can be easily declared as required, with the only other "special" age group being infant deaths (see Figure 4.9), ie. those whose age at death is less than one year.

Declare: INFANT
Select: x:DECEASED
Where: *Define*: $< x, y >$:AGEUNIT **and**
((x:DTHS50-64 and $y \geq$ '3' and $y \leq$ '7') or
(x:DTHS65-84 and $y \geq$ '2' and $y \leq$ '6'))
(Decedents with age at death less than one year).

Figure 4.9: A logical subset of short-lived decedents.

Those decedents with any of the various time-variant encodings whose semantic content is "age at death unknown" would be included in one or both of the sets UNK(AGEUNIT) or UNK(AGEDTH), so that those decedents would be excluded from any query-result set produced by a query mentioning age at death.

Similar treatment of unknown, not applicable, or invalid null values apply to most attributes, and so it is not necessary to repeat the information contained in the previous paragraph for each attribute. Assume that all three types of missing information are declared, if appropriate, according to the specifications in Chapter 3.

For the POB attribute, one could quite easily declare subsets for any group of codes. For example, the set of all decedents born in Europe (using the DVS expanded definition of Europe) would be as shown in Figure 4.10.

Declare: EUROPEAN
Select: x:DECEASED
Where: *Define*: $< x, y >$:POB **and (($y \geq$ '21' and $y \leq$ '27') or ($y \geq$ '51' and $y \leq$ '76'))**
(Decedents born in Europe).

Figure 4.10: An example of how to define ethnic groups.

Similar groups could be declared for the ETHRACE attribute, with a difference being that ETHRACE is "not applicable" to all decedents after 1973.

Declaring subsets of CENDIV would be a bit more complicated, since one would need to deal with the changes after 1956 (as detailed in Table 3.4), which also involves School Districts.

Therefore, it is necessary to declare logical School Districts first. Census Divisions have not existed since 1970, and cannot be logically approximated by other attributes, so they are not of much current interest.

With properly declared UNK(REGDIS), INV(REGDIS) and N/A(REGDIS) sets, one could very easily not bother declaring subsets of REGDIS. Since there are no encoding changes to deal with, the values 001 through 029 uniquely describe the 29 Regional Districts, and thus decedents residing in (eg.) Regional District 13 can be referred to with the fragment

$$\ldots \ (< x, y >\text{:REGDIS and } y = \text{'013'}) \ \ldots \ .$$

However, if one wanted to make use of the correspondence between Regional Districts and certain groups of School Districts as shown in Table 3.5, it is relatively simple to declare logical subsets of DECEASED for each of the 29 Regional Districts. The intension would simply use the actual value from VREGDIS for the years 1971 – 1984 together with the logical School District groups for the years 1950 – 1970. In this way it is possible to refer to decedents residing in Regional District 13 even in 1950. Of course, the logical School Districts must be previously declared. Assuming that the necessary logical School Districts have been declared, a "retroactive" declaration of Regional District 13 would look like that given in Figure 4.11.

Declare: RD13
Select: x:DECEASED
Where: *Define*: ($< x, y >$:REGDIS and y = '013') or x:SD32 or x:SD33 or x:SD76
(Decedents residing within the boundaries of Regional District 13).

Figure 4.11: A "retroactive" Regional District subset.

If there were some special reason to be interested in groups of Municipalities or Census Tracts, such declarations would be simple enough to make. The N/A sets for the MUNICIP and CTRACT sets would take care of all the decedents with "not applicable" values recorded for the *Residence — Municipality* and *Residence — Census Tract* fields of DVS-Ext.

There is no alternative to declaring logical School Districts, because of the several deletions (mergers) and additions (splits) done by the Ministry of Education in the past. In order to make

the best use of the information available, it would most likely be best to use both Table 3.7 and Table 3.8 to declare "retroactive" logical School District subsets of DECEASED. Such subsets would consist of all decedents residing within the current School District boundaries, regardless of whether or not some of the School Districts officially existed (according to the Ministry of Education) at the time the decedent died. For example, the "retroactive" School District 85 could be declared as in Figure 4.12.

Declare: SD85
Select: x:DECEASED
Where: *Define*: ($< x, y >$:SCHDIS **and** y = '085') **or**
(x:DTHS50-64 **and** (y = '073' **or** y = '074')) **or**
((y = '¬␣␣' **or** y = '—␣' **or** y = '␣␣␣') **and** x:CD56)
(Decedents residing within the boundaries of School District 85).

Figure 4.12: An inferred "retroactive" School District subset.

Notice that the fragment

(y = '¬␣␣' **or** y = '—␣' **or** y = '␣␣␣')

would normally be the major part of the intension of UNK(SCHDIS). Since these otherwise "invalid" codes are being used for the purpose of assigning decedents to valid School Districts (along with other information), the set UNK(SCHDIS) should be declared after all the logical "retroactive" School Districts have been declared. The UNK(SCHDIS) set would then consist of those decedents which still had not been included in any logical School District set even after using all the information from Tables 3.7 and 3.8. The intension of UNK(SCHDIS) would consist mainly of the fragment

not (x:SD01 **or** ... **or** x:SD89)

with the ellipsis expanded to include the rest of the logical School District sets.

Occupations would usually not be of much interest until a person has reached a reasonable "working age", so the set N/A(JOB) could be declared to include all decedents with an age at death less than age 20. There would be no UNK(JOB) or INV(JOB) sets, since all the

"unknown" and "invalid" job codes are grouped together in Occupational Groups 193 and 194 (see Appendix D) and treated as separate "valid" occupations. Group 194 would be declared last with the declaration as shown in Figure 4.13, with the ellipsis expanded appropriately.

Declare: OCC194
Select: x:DECEASED
Where: *Define*: ($< x, y >$:JOB **and** ((x:DTHS50-64 **and** y = '$\neg\sqcup\sqcup$') **or**
(x:DTHS65-84 **and** y = '$\sqcup\sqcup\sqcup$'))) **or**
not (x:OCC001 **or** ... **or** x:OCC193)
(Decedents with invalid or miscoded job)

Figure 4.13: Creating validity from missing information.

Notice that the set OCC194 would exclude members of N/A(JOB) according to the rules described in the previous section.

A reconciliation of the two different encoding schemas used for the *Occupation — Industry* field has not been done as yet. However, it would at least be necessary to deal with missing information. The fact that members of Occupational Groups 189 to 192 have a "not applicable" industry (by definition) must be treated differently from the N/A(INDUST) set (decedents under age 20). It would probably be best to declare separate Industry Groups to correspond to Occupational Groups 189 to 192. This could be done using attribution by default (as explained in the previous section), but the domain of VINDUST is rather large, so it would probably be more efficient to declare these new Industry Groups explicitly by directly referring to membership in Occupational Groups 189 to 192.

For the declaration of the Cause of Death Groups, let the notation x[1,3] be an arbitrary shorthand for the substring operation of considering the characters 1 through 3 of the string represented by the variable x. Then (choosing a relatively complicated example consisting of single 3-digit codes, code ranges, 4-digit codes and exclusions), the declaration of Cause of Death Group 75 would be as indicated in Figure 4.14.

Declare: COD75
Select: x:DECEASED
Where: *Define*: $< x, y >$:PRIMCOD **and** ((x:DTHS50-68 **and** y[1,3] = '325' **and**
$y \neq$ '3254') **or** (x:DTHS69-78 **and** (y[1,3] $\geq$ '310' **and**
y[1,3] $\leq$ '315' **or** y = '3330') **and** y[4,4] $\neq$ '5') **or**
(x:DTHS79-84 **and** (y[1,3] $\geq$ '317' **and** y[1,3] $\leq$ '319' **or**
y = '3301')))
(Primary Cause of Death category "Mental Retardation").

Figure 4.14: A complicated Cause of Death group.

The intension of INV(PRIMCOD) would be quite lengthy, in order to specify all the contradictions that could be contained in the data, but it could be done. Similar groups could be declared for the SECOCOD attribute, depending on need.

The last example is the declaration of a logical set consisting of all Registered Native Indians, in Figure 4.15.

Declare: INDIAN
Select: x:DECEASED
Where: *Define*: ($< x, y >$:ETHRACE **and** x:DTHS50-73 **and** y = '15') **or**
($< x, z >$:NATIVE **and** (x:DTHS74-76 **and** y = '1') **or**
(x:DTHS77-84 **and** y = '2'))
(Decedents with Registered Native Indian status).

Figure 4.15: Creating a logical set from two attributes.

The ***native indian status*** attribute is probably the best example of a good use of attribution by default. One could make the necessary declarations to ensure that decedents are by default considered to be not Registered Native Indians, as in the example given in [Gilmore 87a]. In this case, however, membership in the set INDIAN depends upon a time-dependent reference to the set ETHRACE, which has a fairly large domain of values from the VORIGIN set. It could be better to explicitly declare the set NONINDIAN whose intension would be simply x:DECEASED **and not** x:INDIAN.

4.8 Further Considerations

Declaring so many logical sets from the raw data may seem unwieldy, but once the task is complete, the sets are available to be referred to by applications programmers and users alike. The major advantage is that once the encoding changes and missing information codes have been properly specified internally, there would be little further need for every person utilizing the database to know all the details described in Chapter 3. Of course, every person utilizing the database would still need to know most of the pre-defined logical sets and of what their membership consists, but users must know *something* about the database they are using in order to accomplish anything. Some kind of user's reference manual would still be required, but it would be relatively simple and straightforward instead of incredibly complicated and confusing.

The majority of the logical sets which would be declared are because of the Occupational and Cause of Death Groups. Those groups are actually used for specific applications and are included in this thesis for the following reasons:

1. It is necessary to group the Occupations and Causes of Death in some way, since it is not possible to adequately handle the encoding changes at the level of individual codes.

2. Grouping reduces the 1000 different Occupation codes and the 11,000 different Cause of Death codes down to more manageable numbers of groups.

3. The groups chosen are representative and serve as good examples of how a data manager might use DEFINE to declare arbitrary sets, even when those sets involve complicated encoding changes and complicated missing information specifications.

A data manager creating a database from DVS-Ext could choose to create "bare minimum" groupings (grouping together only those codes that cannot be handled individually). Such groupings would give applications programmers and users more flexibility to declare sets as required, rather than being stuck with the 206 Occupational Groups and 165 Cause of Death Groups detailed in Appendices D and E. It would make sense to be as broad as possible in

declaring sets in the actual database specification, to avoid the need to refer to the encoding changes and missing information specifications for some unforseen application.

Chapter 5

Conclusions

Chapter 1 gave an introduction to the topic of handling "dirty" data with a database management system, and a motivation for why one would want to do so. Chapter 2 provided definitions and examples of the types of problem data considered, namely time-variant encodings and decodings, "unknown" null values, "not applicable" null values, and "invalid" null values. Chapter 2 also gave a review of previous attempts to include processing of problem data in database management systems, and indications of why the approaches were not completely satisfactory. Chapter 3 gave detailed descriptions of the intricacies that would be involved in declaring and processing an example data set called DVS-Ext. The example data set is one that really exists, and has incredibly complicated specifications for interpretation of the encoded data values. Chapter 4 introduced the database declaration/manipulation/query language called DEFINE. Chapter 4 then showed (briefly) how one might declare the database version of DVS-Ext in order to include the interpretation specifications for each field (attribute) in the structure of the database itself. The result is to remove most of the worry and fuss about the specifications from the minds of users and applications programmers.

5.1 Accomplishments

The set theory and first-order logic of DEFINE allow quite straightforward handling of time-variant encodings or decodings. Simply declare logical sets from the actual codes, specifying

(for example) that males are defined by code "1" during the time period 1950 – 1976 and code "M" during the time period 1977 – 1984. With this method, the actual code stored in the memory of the computer remains unchanged, while at the same time a "logical recoding" has taken place. Different codes can be treated as identical, or the same code can be treated differently, depending on the how the encodings have been altered over time. Since the actual data remains unchanged, it can still be referred to if necessary, and in the case of DVS-Ext, it can be easily merged with or compared to the original full death registration file. Once the logical sets have been declared, there should be very little need to refer back to the original time-variant encoding specifications. In that sense, the problem of handling time-variant encodings has been solved.

Assuming none of the three types of missing information can be totally avoided, dealing with such values is more complicated than dealing with time-variant encodings. If different applications need to treat each of the three types of missing information in two different ways, it could be necessary to declare as many as six different versions of the same logical set. It could be quite confusing when one wants to know which interpretation of missing information to use. It seems reasonable that most of the time only one interpretation of missing values is necessary, usually the default (exclude entities associated with those null values from all query results, database operations, and function calculations). When special treatment is required, it is normally only one type of special treatment. The method proposed is to declare UNK, N/A, and INV sets for each attribute for which each of those types of missing information exist. Then the intension of any query would be expanded (by default, except where those sets are not declared) to exclude entities with attribute values belonging to any of the three sets UNK, N/A, or INV. For cases where it is reasonable to assume a certain value for an attribute when there is a lack of contradictory data, the method of attribution by default can be used when the domain is relatively small. The problem of handling missing information in a database management system is solved in the sense that with careful declaration of logical sets, it is possible to treat each type of missing information in any way one wishes. When the declarations are complete, even queries asking for negative information are answered in a way

that makes intuitive sense. The user or applications programmer need not worry or fuss about all the different missing information specifications unless some treatment is required that has not already been provided for.

5.2 Remaining Work

The major difficulty that has yet to be overcome is actual implementation of DEFINE. Some success has been accomplished, but there are several implementation problems that have defied satisfactory solution. Only more research and hard work can resolve the problems and bring implementation of DEFINE closer to realization. Encouragingly, the obstacles do not seem to be insurmountable.

It is not possible to forsee all desired future user views and different applications in advance, so it is likely that someone at some time will need to review the details of the time-variant encodings and missing information for some task. However, with a proper survey of user needs, it should be possible to declare enough logical sets of different types (for example, treating missing information in different ways) to satisfy the vast majority of future applications. The need to refer to all the quirks of the DVS-Ext database (for example) cannot be completely eliminated, but most users and applications programmers would not rarely require such reference. Each instance of problem data that can be handled by the database management system itself results in freeing users and applications programmers from dealing with that particular difficulty, most of the time.

One possible objection to the methods described in this thesis is that it could be necessary to declare many different logical sets. As indicated in the previous section, it could even be necessary to declare several different versions of the "same" set, each with a different treatment of one of the types of missing information. Most of the logical sets are *defined* sets, though, meaning that membership in those sets is determined by the system rather than being physically stored in memory. Except for possible confusion over the semantics of each of the defined logical sets, the number of such sets is therefore unimportant.

The approach to missing information in this thesis could be considered a curse as well as a blessing. Flexibility in the way missing information is handled has been substituted for rigid formal treatment of missing information. This methodology allows almost any treatment of null values that a user or applications programmer can think of, but also leaves open the possibility of accidentally declaring an inappropriate treatment of nulls. Perhaps more research could be done on how to prevent users or applications programmers from declaring unsensible treatments of each of the three types of missing information.

One area of work that has not been included is a treatment of the general problem of partial information, as described in Chapter 2. Since DEFINE is based on set theory, and partial information replaces a single value for an attribute with a *set* of possible values, it seems possible to include some kind of processing of partial information in DEFINE. Such methods for handling partial information is left to future research.

Just as the previous work on null values has been restricted to the relational model, so the methods described herein are currently only supported within the set model. Perhaps the treatment of time-variant encodings can be included in some of the more recent research involving use of abstract data types within the relational model context. As described in Chapter 2, there has been some success in handling both Type 1 and Type 2 missing information within the relational framework. For Type 3 missing information, recoding the "invalid" values to the "unknown" null value seems to be a reasonable approach (except for compatibility with the original data). A value that is not valid may give some clue to the correct value (i.e. a keying error where one digit is wrong), but the correct value is still unknown. With this recoding, Type 3 missing information can be treated identically to Type 1 missing information. The proposed formalisms for dealing with missing information within the relational model are still too inflexible. However, a combination of using abstract data types and first-order logic in the relational model may permit enough flexibility to use the approach detailed in this thesis.

5.3 Final Comments

The topic of handling problem data within a database management system has been advanced in the following ways:

1. A method of dealing with time-variant encodings and decodings has been provided, whereas no mention of time-variant encodings or decodings was found during an intensive literature search. Therefore, this thesis has introduced, discussed, and solved the problem of handling a data inconsistency which has not been dealt with previously.

2. Three distinct types of missing information have been defined, and a method of processing all three types has been given. The focus of most of the literature has been on Type 1 missing information only, with some reference to Type 2 missing information. The literature has seemed to ignore the possible existence of Type 3 missing information. Most importantly, the proposed method of handling missing information is extremely flexible, in contrast to the rigid treatments described in the literature.

3. The original desired outcome of having the database management system itself handle all of the quirks of "dirty" data (such as that of DVS-Ext) has been largely realized. Given that it is impossible to forsee all future needs, a data declaration that encompasses most future needs is quite acceptable.

Bibliography

[Ariav 86] Ariav G., A Temporally Oriented Data Model, ACM Trans. Database Syst., Vol. 11, #4, Dec. 1986, pp 499–527.

[Baroody 81] Baroody A.J.Jr., DeWitt D.J., An Object-Oriented Approach to Database System Implementation, ACM Trans. Database Syst., Vol. 6, #4, Dec. 1981, pp 576–601.

[Biskup 81] Biskup J., A Formal Approach to Null Values in Database Relations, in *Advances in Data Base Theory* (eds. Gallaire H., Minker J., Nicolas J.M.), Plenum NY, 1981.

[Biskup 83] Biskup J., A Foundation of Codd's Relational Maybe-Operations, ACM Trans. Database Syst., Vol. 8, #4, Dec. 1983, pp 608–636.

[Clifford 83] Clifford J., Warren D.S., Formal Semantics for Time in Databases, ACM Trans. Database Syst., Vol. 8, #2, Jun. 1983, pp 214–254.

[CODASYL 71] *Data Base Task Group Report* (ed. CODASYL), ACM NY, 1971.

[Codd 70] Codd E.F., A Relational Model of Data for Large Shared Data Banks, Commun. ACM, Vol. 13, #6, Jun. 1970, pp 377–387.

[Codd 75] Codd E.F., *Understanding Relations* (Installment 7), FDT Bulletin of ACM/SIGMOD 7, Vol. 3–4 (1975), pp 23–28.

[Codd 79] Codd E.F., Extending the Database Relational Model to Capture More Meaning, ACM Trans. Database Syst., Vol. 4, #4, Dec. 1979, pp 397–434.

[DBS 48] ***Standard Industrial Classification Manual***, Catalogue 12-501B, Dominion Bureau of Statistics, Ottawa, 1948.

[DBS 51] ***Classification of Occupations***, Ninth Census of Canada, Dominion Bureau of Statistics, Ottawa, 1951.

[DBS 60] ***Standard Industrial Classification Manual***, Catalogue 12-501, Dominion Bureau of Statistics, Ottawa, 1960.

[DBS 61] ***Occupational Classification Manual***, 1961 Census of Canada, Dominion Bureau of Statistics, Ottawa, 1961.

[Fry 78] Fry J.P., Birss E., et. al., An Assessment of the Technology for Data- and Program-Related Conversion, Proc. AFIPS National Computer Conf., 1978, pp 887–907.

[Gilmore 86] Gilmore P.C., Natural Deduction Based Set Theories: A New Resolution of the Old Paradoxes, J. Symbolic Logic, Vol. 51, #4, May 1986, pp 393–411.

[Gilmore 87a] Gilmore P.C., ***Formalizing Attribution by Default***, Tech. Rep. 87-26, Univ. Brit. Col., Jul. 1987.

[Gilmore 87b] Gilmore P.C., ***Concepts and Methods for Database Design***, Tech. Rep. 87-31, Univ. Brit. Col., Aug. 1987.

[Grant 77] Grant J., Null Values in a Relational Database, Inform. Processing Letters, Vol. 6, #5, Oct. 1977, pp 156–157.

[Grant 79] Grant J., Partial Values in a Tabular Database Model, Inform. Processing Letters, Vol. 9, #2, Aug. 1979, pp 97–99.

[Hammer 81] Hammer M., McLeod D., Database Description with SDM: A Semantic Database Model, ACM Trans. Database Syst., Vol. 6, #3, Sept. 1981, pp 351–386.

[Lien 79] Lien Y.E., Multivalued Dependencies with Null Values in Relational Databases, Proc. Fifth Internat. Conf. on Very Large Databases, Oct. 1979, pp 61–66.

[Lipski 79] Lipski W.Jr., On Semantic Issues Connected with Incomplete Information Databases, ACM Trans. Database Syst., Vol. 4, #3, Sept. 1979, pp 262–296.

[Lipski 81] Lipski W., On Databases with Incomplete Information, J. ACM, Vol. 28, #1, Jan. 1981, pp 41–70.

[McGee 77] McGee W.C., The Information Management System IMS/VS, IBM Syst. J., Vol. 16, #2, Jun. 1977, pp 84–168.

[NCHS 68] *International Classification of Diseases, Injuries and Causes of Death*, Adapted 8th Revision, U.S. Dept. H.E.W., Public Health Service, National Center for Health Statistics, Washington D.C., 1968.

[NCHS 78] *International Classification of Diseases*, 9th Revision, Clinical Modification, U.S. Dept. H.E.W., Public Health Service, National Center for Health Statistics, Washington D.C., 1978.

[Osborn 86] Osborn S.L., Heaven T.E., The Design of a Relational Database System with Abstract Data Types for Domains, ACM Trans. Database Syst., Vol. 11, #3, Sept. 1986, pp 357–374.

[Reiter 78] Reiter R., On Closed World Data Bases, in *Logic & Data Bases* (eds. Gallaire H., Minker J.), Plenum NY, 1978.

[Rybinski 87] Rybinski H., On First-Order Logic Databases, ACM Trans. Database Syst., Vol. 12, #3, Sept. 1987, pp 325–349.

[Shipman 81] Shipman D.W., The Functional Data Model and the Data Language DAPLEX, ACM Trans. Database Syst., Vol. 6, #1, Mar. 1981, pp 140–173.

[Spyratos 87] Spyratos N., The Partition Model: A Deductive Database Model, ACM Trans. Database Syst., Vol. 12, #1, Mar. 1987, pp 1–37.

[WHO 48] *Manual of the International Statistical Classification of Diseases, Injuries and Causes of Death*, 6th Revision, World Health Organization, Geneva, 1948.

[WHO 57] *International Classification of Diseases*, 7th Revision, World Health Organization, Geneva, 1957.

[Wong 80] Wong E., A Statistical Approach to Incomplete Information in Database Systems, ACM Trans. Database Syst., Vol. 7, #3, Sept. 1982, pp 470–488.

[Vassiliou 79] Vassiliou Y., Null Values in Database Management: A Denotational Semantics Approach, Proc. ACM/SIGMOD Internat. Symposium on Management of Data, May 1979, pp 162–169.

[Vassiliou 80] Vassiliou Y., Functional Dependencies and Incomplete Information, Proc. Sixth Internat. Conf. on Very Large Databases, Oct. 1980, pp 260–269.

Appendix A

New Syntax of DEFINE

Dr. Gilmore has recently improved the syntax of DEFINE declarations, to make them more compact and less wordy. Refer to Figure 4.1 and the remainder of Section 4.2 in order to relate the new syntax to the original syntax.

The new syntax for declaration of base sets is

SETNAME for {DOMAIN(SETNAME) | degrees | comment}.

DOMAIN(SETNAME) is as previously described, and if left blank, the set is primitive. The degrees are <LL,LU>, <RL,RU> as described in Section 4.2. The degrees are not specified (left blank) for primitive sets. The comment is unchanged from the old syntax.

The new syntax for declaration of defined sets is

SETNAME for {domain + variable declaration | INTENSION(SETNAME) | comment}.

The "domain + variable declaration" part takes the form

variable:DOMAIN(SETNAME)

with the same semantics as the *Select* clause from the old syntax. INTENSION(SETNAME) is the same as described in Section 4.2. The comment is unchanged from the old syntax.

Appendix B

Encoding Schema for the VORIGIN Set

Codes for *racial or ethnic origin* are shown in parentheses, if different from the code for *place of birth* (country of origin).

00 = Unknown
01 = Prince Edward Island
02 = Nova Scotia
03 = New Brunswick
04 = Quebec
05 = Ontario
06 = Manitoba
07 = Saskatchewan
08 = Alberta
09 = British Columbia
10 = Yukon Territory
11 = Northwest Territories
12 = Newfoundland
17 = Canada, province not stated
13 = Barbados or West Indies
18 = Other British Possessions in America
(British West Indian race = 37)
(Eskimo or Inuit race = 14)
(Non-ward Indian race = 16)
(Ward Indian race = 15)
(Halfbreed = 17)

21 = England
22 = Northern Ireland (Irish race = 23)
23 = Irish Free State
24 = Scotland
25 = Wales
26 = Lesser Isles
27 = British NOS

31 = Australia and Mandates (race = 37)
32 = New Zealand and Mandates (race = 37)
33 = South and South West Africa (race = 37)
34 = Other British Possessions in Africa (race = 37)
35 = India or Pakistan (race = 86)

36 = Other British Possessions in Asia (race = 37)
37 = Other British Possessions

41 = United States of America
42 = Mexico (race = 72)
43 = Other North American Countries (race = 72)
(Cuban, Dominican, Puerto Rican race = 72)
44 = Central American Countries (race = 72)
45 = South American Countries (race = 72)
(Brazilian race = 70, Haitian race = 92)

51 = Albania	61 = Greece	71 = Romania
52 = Austria	62 = Holland/Netherlands	72 = Spain
53 = Belgium	63 = Hungary	73 = Sweden
54 = Bulgaria	64 = Iceland	74 = Switzerland
55 = Czechoslovakia	65 = Italy	75 = Yugoslavia
56 = Denmark	66 = Latvia	76 = Other European
57 = Estonia	67 = Lithuania	77 = U.S.S.R.
58 = Finland	68 = Norway	(Russian race = 79)
59 = France	69 = Poland	(Ukranian race = 97)
60 = German	70 = Portugal	(Mennonite race = 78)

82 = China	86 = Indian, Pakistan, Burma
83 = Japan or Korea	87 = Laos, Thailand, Cambodia, Vietnam
84 = Syria	88 = Indonesia, Malaysia, Philippines
85 = Turkey	89 = Other Asiatic, Fiji

91 = African Countries, not British (Black NOS = 92)
93 = Other Countries (Caucasian NOS = 94) (Oriental NOS = 95)
96 = Palestine, Israel (Jewish or Hebrew race = 96)
98 = At Sea 99 = Unkown

Appendix C

Encoding Schema for the VMUNICIP Set

001 = Alberni
002 = Armstrong
003 = Chilliwack City
004 = Courtenay
005 = Cranbrook
006 = Cumberland
007 = Duncan
008 = Enderby
009 = Fernie
010 = Grand Forks
011 = Greenwood
012 = Kamloops
013 = Kaslo
014 = Kelowna
015 = Ladysmith
016 = Merritt
017 = Nanaimo
018 = Nelson
019 = New Westminster
020 = North Vancouver City
021 = Port Alberni
022 = Port Coquitlam
023 = Port Moody
024 = Prince George
025 = Prince Rupert
026 = Revelstoke
051 = Matsqui
052 = Mission Mun.
053 = North Cowichan
054 = N. Vancouver Mun.
055 = Oak Bay
056 = Peachland
057 = Penticton
058 = Pitt Meadows
059 = Richmond
060 = Saanich
061 = Salmon Arm Dist.
062 = Spallumcheen
063 = Sumas
064 = Summerland
065 = Surrey
066 = Tadanac
067 = West Vancouver
068 = Central Saanich
069 = Kitimat
070 = Abbotsford
071 = Burns Lake
072 = Creston
073 = Gibsons Landing
074 = Hope
075 = McBride
076 = Mission City
126 = Cache Creek
131 = UBC Endow. Lands
140 = Powell River
141 = Port Alice
142 = North Saanich
143 = Gold River
144 = Hudson Hope
145 = Sparwood
146 = Port Hardy
147 = Mackenzie
170 = Fort St. John
171 = Squamish
172 = Kinnaird
173 = Marysville
174 = Harrison Hot Springs
175 = Invermere
176 = Ucluelet
177 = Ashcroft
178 = Zeballos
179 = Sidney
180 = Telkwa
181 = Warfield
182 = Princeton
183 = Fruitvale
184 = Fort St. James
185 = Lumby

027 = Rossland
028 = Salmon Arm Village
029 = Slocan
030 = Trail
031 = Vancouver
032 = Vernon
033 = Victoria
034 = Kimberley
035 = Lytton
036 = North Kamloops
037 = Castlegar
038 = Salmo
039 = Lillooet
040 = Burnaby
041 = Chilliwhack Municip.
042 = Coldstream
043 = Coquitlam
044 = Delta (Ladner)
045 = Esquimalt
046 = Fraser Mills
048 = Kent
049 = Langley Municip.
050 = Maple Ridge
077 = New Denver
078 = Pouce Coupe
079 = Quesnel
080 = Silverton
081 = Smithers
082 = Stewart
083 = Terrace
084 = Tofino
085 = Vanderhoof
086 = Williams Lake
087 = Dawson Creek
088 = Chapman Camp
090 = Qualicum Beach
092 = Lake Cowichan
093 = Parksville
094 = Oliver
095 = Alert Bay
096 = Comox
097 = Osoyoos
099 = Campbell River
101 = Langley City
102 = White Rock
125 = Guisachan
186 = Hazelton
187 = Sechelt
188 = Montrose
189 = Pemberton
190 = Keremeos
191 = Houston
192 = Golden
193 = Taylor
194 = Masset
196 = Aennofield
197 = Chetwynd
198 = Valemount
199 = Clinton
201 = Nakusp
202 = 100 Mile House
203 = Port McNeill
204 = Port Edward
205 = Fraser Lake
206 = Midway
208 = Sayward
209 = Chase

Appendix D

Groupings of Occupation Codes

Group	1951 Codes	1961 Codes	Occupational Title
001	056	001,002	Advertising & Credit Managers
002	041,049	004,005	Sales Managers
003	333	008	Purchasing Agents & Buyers
004	001,011,020,040, 046,047,050,052, 053,054,058,059, 369	006,007,010	Owners & Managers N.E.S., & Government Officials
005	073	101	Civil Engineers
006	076	102,104	Mechanical Engineers
007	075	105	Electrical Engineers
008	072	108	Chemical Engineers
009	078	107,109	Other Engineers
010	066	111,147	Chemists (including Pharmacists)
011	099	112,114,119,121, 144,145,149,186, 198,199	Scientists (Geologists, Physicists, Biologists, Optometrists & Other)
012	098	124	Veterinarians
013	061	129	Agricultural Professionals
014	090	131	Professors & College Principals
015	095	135	School Teachers
016	096	139	Other Teachers & Instructors
017	089	140	Physicians & Surgeons
018	069	141	Dentists

Group	1951 Codes	1961 Codes	Occupational Title
019	086,087	142,143	Nurses
020	088	146	Osteopaths & Chiropractors
021	080,880	148	Medical & Dental Technicians
020	088	146	Osteopaths & Chiropractors
021	080,880	148	Medical & Dental Technicians
022	079	151	Judges & Magistrates
023	081	153	Lawyers
024	068,085,091	161,163,169	Religious
025	063,064	171,172	Artists
026	065	174	Journalists
027	084	176	Musicians
028	062	181	Architects
029	070	182	Draughtsmen/women
030	074	183	Surveyors
031	092,097	184	Statisticians & Actuaries
032	N/A	187	Computer Analysts & Programmers
033	060	188	Accountants
034	067	191	Dieticians
035	094	192	Social Workers
036	082	194	Librarians
037	343	195	Interior Decorators
038	093,886	196,915	Photographers & Photo Processing Workers
039	110	201	Bookkeepers & Cashiers
040	113	203	Office Machine Operators
041	117	214	Shipping & Receiving Clerks
042	205,209	221,223	Baggage Checkers & Ticket Agents
043	119	232,234	Secretaries
044	115,312,321	212,249	Other Clerical Workers
045	111	241	Office Assistants to Doctor or Dentist
046	301,336	301,325	Retail Sales Clerks
047	304	303	Auctioneers
048	309,317	307,312,339	Door-to-Door Salesmen/women & Pedlars
049	315	314	Commercial Travellers
050	324	316	News Vendors

Group	1951 Codes	1961 Codes	Occupational Title
051	339	323	Service Station Attendants
052	302	327	Advertising Salesmen/women & Agents
053	362	331	Insurance Agents
054	364	334	Real Estate Agents
055	306,366	336,338	Brokers & Financial Salesmen/women
056	473	401	Firefighters
057	477	403	Police
058	474,479	405	Guards & Watchmen/women
059	475,476	407,408	Members of Armed Forces
060	425	411	Boarding House & Hotel Keepers
061	458	412	Stewards/Stewardesses
062	452	413	Cooks
063	443	414,415	Bartenders & Waiters/Waitresses
064	429	416	Nursing Aides
065	432	417	Porters
066	456	418,419	Domestics
067	492	431,433	Entertainers & Athletes
068	402	451	Barbers & Hairdressers
069	414	452	Launderers & Dry Cleaners
070	416	453	Elevator Operators
071	409,422	454	Janitors
072	435	455	Funeral Directors
073	418	456	Guides
074	403,449,497,499	457,459	Recreation Attendants
075	201,203,223	510	Transport Inspectors & Supervisors
076	207	520	Aircraft Pilots
077	231,233	531,532	Locomotive Engineers & Fire-Stokers
078	211,219,245	534,535,537	Locomotive Conductors, Brake Workers & Switchers
079	215	541	Deck Officers on Ship
080	225	543	Engineering Officers on Ship
081	241	545	Deck Ratings
082	227	547	Engine Room Ratings

Group	1951 Codes	1961 Codes	Occupational Title
083	213,239	551,561	Bus Drivers
084	217	552	Taxi Drivers
085	247,249	554,556,563	Truck Drivers
086	229,251	569	Other Transport Workers N.E.S.
087	261,265	570	Communications Inspectors & Supervisors
088	274,278	581,582	Radio & Television Announcers & Technicians
089	281,288	584,585	Telephone Operators
090	271	587	Postal Workers
091	237	588	Messengers
092	500,501	601,603	Farmers & Farm Managers
093	504	605	Farm Labourers
094	506,509	607,609	Gardeners & Nursery Workers
095	561,564,568	611,613,615	Loggers & Forest Rangers
096	550	631	Fishermen/women
097	554	633	Trappers & Hunters
098	601	651	Mine & Quarry Supervisors
099	609	652	Prospectors
100	603,607,615	653,654,657	Miners & Mine Labourers
101	605	655	Mine Mill Workers
102	610	656	Gas & Oil Well Drillers
103	619	659	Quarry & Related Workers
104	707	701	Millers of Flour & Grain
105	701	702	Bakers
106	703	703	Meat Cutters
107	706	704	Meat Canners, Curers & Packers
108	705	705	Fish Canners & Packers
109	704,709,710	706,707,708,709	Other Food Processing Workers
110	731,733,737,739	711,713,719,768	Tire Builders, Vulcanizers & Other Rubber Workers
111	752	721	Leather Cutters
112	755,756	722,724	Shoemakers
113	757	916	Tanners

Group	1951 Codes	1961 Codes	Occupational Title
114	753,754,759	729	Other Leather Workers
115	760,761,762,763, 764,765,767,768, 769	731,732,733,734, 735,736,737,738, 739	Textile Workers
116	772,774,775,782, 785,789	741,742,745,746, 749	Tailors, Dressmakers & Other Fabric Workers
117	743	743	Furriers
118	777,780	744	Milliners
119	796	747	Upholsterers
120	915	751	Carpenters
121	792	752	Cabinet & Furniture Makers
122	795	754	Sawyers
123	797	756	Wookworking Machine Operators
124	790	758	Graders & Scalers
125	791,793,794,799	759	Woodworking Occupations N.E.S.
126	872	761	Oil Refinery Operators
127	805,809,877	763,765,766	Pulp Preparers
128	876,879	769	Chemical & Related Workers
129	814	771	Typesetters
130	817	772	Printing Press Operators
131	819	779	Printing Occupations N.E.S.
132	815	773,775	Photo-Engravers & Lithographers
133	812,813	776,778	Bookbinders
134	831	781	Metal Furnace Workers
135	832,854	782,783	Metal Heat Treaters & Rolling Mill Operators
136	822	784	Blacksmiths
137	845,846	786,787	Moulders & Coremakers
138	847,851,858,859	788,789,819,912	Other Metal Mill Workers
139	826,834	791,793	Jewellers, Watchmakers & Metal Engravers
140	856	801	Tool & Die Makers
141	836	802	Machinists
142	827	803	Filers & Grinders
143	844	805	Millwrights

Group	1951 Codes	1961 Codes	Occupational Title
144	828,855	806,811	Sheet Metal Workers
145	835	808	Metalworking Machine Operators
146	934	810	Plumbers
147	853	812	Riveters
148	824,935	813	Boilermakers & Structural Iron Workers
149	825	815	Electroplaters & Dip Platers
150	857	817	Welders
151	850	818	Polishers & Buffers
152	820	917	Metal Inspectors & Supervisors
153	837	821	Aircraft Mechanics
154	838	822	Auto Mechanics
155	840	825	Railroad Mechanics
156	841	824,829	Office Machine & Other Machine Repairers
157	924	831	Electricians
158	895	833	Power Station Operators
159	267,299	838	Telephone Servicemen/women & Line Workers
160	821,852	832,835,839	Electric & Electronic Assemblers & Repairers
161	494	836	Projectionists
162	927	841,843	Painters & Glaziers
163	910	851	Construction Supervisors
164	911	852	Construction Inspectors
165	913	854	Bricklayers & Tilesetters
166	917	855	Cement Finishers
167	932	856	Plasterers
168	939	857,859	Other Construction Workers (including Insulators)
169	885	861	Lens Grinders & Opticians
170	864	762,862	Glass & Ceramic Furnace Workers
171	867	864	Stone Cutters
172	860,869	869	Clay, Glass & Stone Workers
173	890,897	871,872	Stationary Engineers
174	892	873	Motormen/women

Group	1951 Codes	1961 Codes	Occupational Title
175	894	874	Crane & Derrick Operators
176	899	875	Riggers & Cable Splicers
177	922	876,877	Heavy Equipment Operators
178	896	878	Oilers & Greasers
179	235	881	Longshoremen/women
180	243	890	Railway Track Workers
181	700	900	Supervisors & Foremen/women N.E.S.
182	720	911	Tobacco Workers
183	328,883	913	Bottlers, Wrappers & Labelers
184	800	914	Paper Product Makers
185	920	883	Warehouse Workers
186	322,349	918	Inspectors & Graders N.E.S.
187	798,889	919	Production Process Workers N.E.S.
188	950	920	Labourers N.E.S.
189	960	960	Permanently Disabled
190	961	961	Occupation coded as "Retired"
191	962	962	Students & Mentally Handicapped
192	963	963	Homemakers
193	999	980,999	Prisoners, Unemployed, or Occupation Not Stated
194	Blank, Dash, or Invalid	Blank, Dash, or Invalid	Invalid or Miscoded Occupations
195	001,011,020,040, 041,046,047,049, 050,052,053,054, 056,058,059,333, 369	001,002,004,005, 006,007,008,010	All Owners & Managers
196	072,073,075,076, 078	101,102,104,105, 107,108,109	All Professional Engineers
197	086,087,429	142,143,416	All Nurses
198	215,225,227,241	541,543,545,547	All Sailors
199	601,603,605,607, 609,610,615,619	651,652,653,654, 655,656,657,659	All Mine, Quarry & Related Workers

Group	1951 Codes	1961 Codes	Occupational Title
200	752,753,754,755, 756,757,759	721,722,724,729, 916	All Leather Workers
201	790,791,792,793, 794,795,797,799, 915	751,752,754,756, 758,759	All Woodworkers
202	814,817,819	771,772,779	All Printers
203	820,822,831,832, 845,846,847,851, 854,858,859	781,782,783,784, 786,787,788,789, 819,912,917	All Metal Mill Workers
204	961,Blank,Dash, Invalid	961,Blank,Dash, Invalid	Occupation Unknown
205	960,962,963,999	960,962,963,980, 999	Disabled, Students, Homemakers & Unemployed
206	001 to 999	001 to 999	All Occupations

Appendix E

Groupings of Cause of Death Codes

Note: 3-digit codes include any 4th digit (or character), except those 4-digit codes which are specifically excluded (to be included elsewhere). A dash indicates a range of consecutive codes.

Group	ICD–7	ICD–8	ICD–9	Cause of Death Title
001	001	010	0114	Silico-Tuberculosis
002	002–008, 010–019	011–019	010–018, 137 (ex. 0114)	Tuberculosis
003	040–064, 764	000–009, 020–025, 0261,027, 030–039	001–009, 020–025, 0261,027, 030–038, 040,041	Bacterial Diseases
004	080,081, 0821, 084–091, 093, 095,096, 696,697	040–046, 050–057, 060,061, 067,068, 072–079	045–057, 060,061, 065,066, 072–079, 138,1391, 7711 (ex. 0498,0499)	Viral Diseases
005	082,083 (ex. 0821)	062–066	0498,0499, 062–064, 1390	Encephalitis
006	092	070	070	Hepatitis

Group	ICD–7	ICD–8	ICD–9	Cause of Death Title
007	094	071	071	Rabies
008	071, 100–108, 110–117, 120,121	080–088	080–087	Rickettsial Diseases
009	020–029	090–097	090–097	Syphilis
010	030–039	098,099	098,099	Other Venereal Diseases
011	070, 072–074	026, 100–104 (ex. 0261)	026, 100–104 (ex. 0261)	Other Spirochaetal Diseases
012	131–134	110–117	039, 110–112, 114–118	Mycoses
013	122–130, 135–138	089, 120–136	088, 120–136, 139 (ex. 1390,1391)	Parasitic Diseases
014	140	140	140	Cancer: Lip
015	141–144	141–145	141–145	Cancer: Mouth
016	145	146	146	Cancer: Oropharynx
017	146	147	147	Cancer: Nasopharynx
018	147	148	148	Cancer: Hypopharynx
019	148	149	149	Cancer: Pharynx Unspecified
020	150	150	150	Cancer: Esophagus
021	151	151	151	Cancer: Stomach
022	152	152	152	Cancer: Small Intestine
023	153	153	153	Cancer: Colon
024	154	154	154	Cancer: Rectum
025	155 (ex. 1551)	155	155 (ex. 1552)	Cancer: Liver
026	1551	156	156	Cancer: Gallbladder
027	157	157	157	Cancer: Pancreas
028	158,159	158,159	158,159	Cancer: Other Digestive Organs
029	160	160	160	Cancer: Nose & Nasal Sinus

Group	ICD–7	ICD–8	ICD–9	Cause of Death Title
030	161	161	161	Cancer: Larynx
031	162,163 (ex. 1622)	162	162	Cancer: Lung
032	1622	1630	163	Cancer: Pleural Cavity
033	164	163 (ex. 1630)	164,165 (ex. 1640, 1641)	Cancer: Mediastinum
034	196	170	170	Cancer: Bone
035	197	171	1641,171	Cancer: Soft Tissue
036	190	172 (ex. 1725)	172	Cancer: Melanoma
037	191	173 (ex. 1735)	173,1877	Cancer: Other (Non–Melanoma) Skin
038	170	174	174,175	Cancer: Breast
039	171	180	180	Cancer: Cervix
040	173	181	181	Cancer: Chorionepithelioma
041	172,174	182	179,182	Cancer: Endometrium
042	175	183	183	Cancer: Ovary
043	176	184	184	Cancer: Other Female Genital
044	177	185	185	Cancer: Prostate
045	178	186	186	Cancer: Testis
046	179	1725,1735, 187	187 (ex. 1877)	Cancer: Other Male Genital
047	181 (ex. 1817)	188	188	Cancer: Bladder
048	180	189 (ex. 1899)	189 (ex. 1893,1894, 1898,1899)	Cancer: Kidney
049	1817	1899	1893,1894, 1898,1899	Cancer: Other Urinary Organs
050	192	190	190	Cancer: Eye
051	193	191,192	191,192	Cancer: Brain & Central Nervous System
052	194	193	193	Cancer: Thyroid
053	195	194	1640,194	Cancer: Other Endocrine Glands

Group	ICD–7	ICD–8	ICD–9	Cause of Death Title
054	156,165, 198,199 206,207	195–199	1552, 195–199	Cancer: Other & Unspecified
055	200 (ex. 2002)	200	200 (ex. 2002,2008)	Cancer: Lymphosarcoma
056	201	201	201	Cancer: Hodgkin's Disease
057	2002,202	202 (ex. 2021)	2002,2008, 202,2053, 2290 (ex. 2021)	Cancer: Other Lymphoma
058	205	2021	2021	Cancer: Mycosis Fungoides
059	203	203	203	Cancer: (Multiple) Myeloma
060	204	204–207	204–208 (ex. 2053, 2071)	Cancer: Leukemia
061	294	208,2890	2071,2384, 2890	Cancer: Polycythaemia
062	2923	209,2858	2858,2898	Myelofibrosis
063	223	224,225	224,225	Benign Neoplasm of Brain & Central Nervous System
064	210–220, 222–229	210–228, 7571	210–229 (ex. 2117, 2290)	All Benign Neoplasms
065	237	238,7434	2340, 2375,2376, 2377,2379, 2396,2397	Unspecified Neoplasm of Brain & Central Nervous System
066	230–239	230–239, 7434	230–239 (ex. 2384)	All Unspecified Neoplasms
067	250–254	240–246	240–246, 3762	Diseases of Thyroid
068	260	250	250	Diabetes

Group	ICD–7	ICD–8	ICD–9	Cause of Death Title
069	270–277	251–258	2117, 251–259	Diseases of Other Endocrine Glands
070	280–289	260–274, 276–279	260–272, 274,275, 277–279, 579	Metabolic Deficiency States
071	2924	284	284	Aplastic Anaemia
072	290–293 (ex. 2923)	280–285 (ex. 2858)	280–285 (ex. 2858)	All Anaemias
073	295–299	275, 286–289 (ex. 2890)	273, 286–289 (ex. 2890, 2898)	Other Diseases of Blood & Blood Forming Organs
074	300–324, 326,6881	290–309	290–316	Psychoses & Neuroses
075	325 (ex. 3254)	310–315, 3330 (ex. 3105,3115, 3125,3135, 3145,3155)	317–319, 3301	Mental Retardation
076	340–344	320–324	320–326	Meningitis & Encephalitis
077	345	340	340	Multiple Sclerosis
078	350–352, 355–357, 7440,7818	330–333, 341–344, 347–349, 7330,7817 (ex. 3330)	330–336, 341–344, 347–349, 3561,358, 359 (ex. 3301)	Other Central Nervous System Diseases
079	353	345	345	Epilepsy
080	354	346	346	Migraine
081	360–369	350–358	337, 350–357, 7292 (ex. 3561)	Diseases of Facial Nerves

Group	ICD–7	ICD–8	ICD–9	Cause of Death Title
082	370–398, 7813	360–389, 7813	360–367, 3680,3685, 3686, 369–389 (ex. 3762, 3763,3765, 3795)	Eye & Ear Diseases
083	400–402, 410–416	390–398 (ex. 3949, 3959,3969)	390–398	Rheumatic Heart Disease
084	420	410–414	410–414	Acute Heart Disease
085	420–422	3949,3959, 3969, 410–414, 424,428	410–414, 424,4290, 4291	Arteriosclerotic Heart Disease
086	440–447	400–404	401–405	Hypertension with or without Other Heart Disease
087	430–434, 7824	420–423, 425–427, 429,7824	415–417, 420–423, 425–429 (ex. 4151, 4253,4290, 4291)	Other Diseases of Heart
088	330–334	430–438	430–438	Cerebral Haemorrhage
089	450–456, 460–468	440–448, 450,451, 453–458	4151, 440–444, 446–448, 451, 453–459, 7854	Diseases of Circulatory System
090	470–475, 500	460–466	460–466	Acute Upper Respiratory Infections
091	480–483	470–474	487	Influenza
092	490–493	480–486	480–486	Pneumonia
093	501,502, 5271	490–492	490–492	Bronchitis & Emphysema

Group	ICD–7	ICD–8	ICD–9	Cause of Death Title
094	241–243, 245	493,507, 708	477,493, 708	Asthma & Allergies
095	240, 510–517	500–506, 508	470–476, 478	Other Infectious Respiratory Diseases
096	518	510	510	Empyema
097	519	511	511	Pleurisy
098	520	512	512	Pneumothorax
099	521	513	513	Abscess on Lung
100	522	514	514	Hypostasis of Lung
101	5230	5150	502	Silicosis
102	5232	5152	501	Asbestosis
103	001,523, 524	010,515, 516	0114,495, 500–508	All Occupational Lung Disease
104	525–527 (ex. 5271)	517–519	494,496, 515–519	Other Chronic Lung Disease
105	530–538	520–529	520–529	Diseases of Mouth & Dentition
106	539	530	530	Diseases of Esophagus
107	540–542	531–534	531–534	Intestinal Ulcer
108	543–545, 7842	535–537, 7842	535–537	Other Disorders of Stomach & Duodenum
109	550–553	540–543	540–543	Appendicitis
110	560–561	550–553	550–553	Hernia
111	570,571	560,561	558,560	Intestinal Obstruction
112	572	562,563	555,556, 562	Ulcerative Colitis
113	573–578, 7845,7858	564–569, 7845,7857	557, 564–569, 578	Peritonitis & Other Intestinal Diseases
114	580,581	570,571	570,571	Cirrhosis of Liver
115	582,583	452,572, 573	452,572, 573	Other Diseases of Liver
116	584–586	574–576	574–576	Diseases of Gallbladder
117	587	577	577	Diseases of Pancreas

Group	ICD–7	ICD–8	ICD–9	Cause of Death Title
118	590–594, 600–604, 792	580–584, 590–594, 792	580–594	Nephritis & Other Kidney Diseases
119	605–609, 7892,7894	595–599, 7891,7893	595–599	Infections of Bladder & Urethra
120	610–612	600–602	600–602	Prostatitis
121	613–617, 7866	603–607, 7866	603–608	Orchitis & Other Male Genital Diseases
122	620,621	610,611	610,611	Cystic Breast Disease
123	622–626, 630–637	612–616, 620–629	614–629 (ex. 6250, 6251,6256)	Diseases of Ovaries & Uterus
124	640–652, 660, 670–678, 680–689 (ex. 6881)	630–645, 650–662, 670–678	630–648, 650–676	Pregnancy & Complications of Pregnancy
125	221,244, 690–695, 698, 700–716 (ex. 7054, 7100,7102)	680–686, 690–698, 700–707, 709	680–686, 690–698, 700–707, 709,7293	Skin Diseases
126	720–727, 7875	710–715, 717,718, 7287	711–716, 7193,720, 7217,7219, 7235,7242, 7245,7260, 7262,7269, 7282,7290, 7291	Arthritis & Rheumatism
127	730	720	730	Osteomyelitis
128	7054,7100, 731–738, 740–747,	716, 721–738, 787	710, 717–719, 721–729,	Diseases of Bones, Joints, Tendons & Fascia

Group	ICD–7	ICD–8	ICD–9	Cause of Death Title
	749,787 (ex. 7440, 7875)	(ex. 7287, 7330,7876)	731–739 (ex. 7193, 7217,7219, 7235,7242, 7245,7260, 7262,7269, 7282,7290, 7291,7292, 7293)	
129	3254,7102, 748, 750–759	3105,3115, 3125,3135, 3145,3155, 740–759 (ex. 7434, 7571)	4253, 740–759, 7786	Congenital Anomalies
130	760–763, 765–776	760–779	760–779 (ex. 7711, 7786)	Perinatal Conditions
131	780–786, 7877, 788–791, 793–795 (ex. 7813, 7818,7824, 7842,7845, 7858,7866, 7892,7894)	780–786, 7876, 788–791, 793–796 (ex. 7813, 7817,7824, 7842,7845, 7857,7866, 7891,7893)	276,368, 3763,3765, 3795,6250, 6251,6256, 780–799 (ex. 3680, 3685,3686, 7854)	Symptoms & Ill–Defined Conditions (including Senility)
132	800–802	800–807	800–807	Accident: Railway
133	810–825, 830–835, 840–845	810–823, 825–827	810–829	Accident: Motor Vehicle or Road
134	850–858	830–838	830–838	Accident: Water Transport
135	860–866	840–845	840–845	Accident: Aircraft
136	870–880	850–860, 868	850–858, 860,865	Accident: Poison: Foods, Drugs, or Alcohol

Group	ICD–7	ICD–8	ICD–9	Cause of Death Title
137	881–888	861–867, 869	861–864, 866	Accident: Poison: Chemicals, Metals, or Solvents
138	890–895	870–877	867–869	Accident: Poison: Gases or Vapours
139	900–904	880–887	880–888	Accident: Falls
140	916	890–899, 9230,9232, 9239	890–899, 9230,9232, 9239	Accident: Burns
141	926–928, 930–935	900–902, 904–909	900–902, 904–909	Accident: Due to Environmental Factors
142	929	910	910	Accident: Drowning
143	921,922, 924,925	911–913	911–913	Accident: Suffocation
144	920,923	914,915	914,915	Accident: Caused by Foreign Body
145	910	916	916	Accident: Struck by Falling Object
146	913	920	920	Accident: Caused by Cutting or Piercing Object
147	911,912	927,928	846–848, 919	Accident: Other Industrial
148	915,919	921–923 (ex. 9230, 9232,9239)	921–923 (ex. 9230, 9232,9239)	Accident: Explosion or Shot with Projectile
149	917	924	924	Accident: Caused by Hot or Corrosive Substance
150	914	925	925	Accident: Electrocution
151	918	926	926	Accident: Due to (Ionizing) Radiation
152	940–946, 950–962	930–936, 940–949	870–876, 878,879, 929–949	Deaths due to Medical Treatment or Late Effects
153	963, 970–979	950–959	950–959	Suicide
154	964, 980–983	960–969	960–969	Homicide
155	984,985	970–978	970–978	Legal Intervention

Group	ICD–7	ICD–8	ICD–9	Cause of Death Title
156	936	903, 917–919, 929, 980–989	903,917, 918,927, 928, 980–989	Accident: Other & Unspecified
157	965, 990–999	990–999	990–999	War Deaths
158	800–802, 810–825, 830–835, 840–845, 850–858, 860–866, 870–888, 890–895, 900–904, 910–936	800–807, 810–823, 825–827, 830–838, 840–845, 850–877, 880–887, 890–929, 980–989	800–807, 810–838, 840–848, 850–858, 860–869, 880–888, 890–928, 980–989	Accident: All Accidents
159	145–148	146–149	146–149	Cancer: All Pharynx
160	140–148	140–149	140–149	Cancer: All Pharynx & Buccal Cavity
161	150–155, 157–159	150–159	150–159	Cancer: All Digestive Organs
162	200,202, 205	200,202	200,202, 2053,2290	Cancer: All Non–Hodgkin's Lymphomas
163	140–148, 150–165, 170–181, 190–207, 294	140–163, 170–174, 180–208, 2890	140–165, 170–175, 179–208, 2290,2384, 2890	Cancer: All Cancers
164	193,223, 237	191,192, 224,225, 238,7434	191,192, 224,225, 2340,2375, 2376,2377, 2379,2396, 2397	All Tumours of Brain & Central Nervous System
165	001–999	000–999	001–999	All Causes of Death