The Steam Engine Makers Database

This database contains information about the individual members of the Steam Engine Makers' Society, an early British trade union, between 1835 and 1876.

Contents

The Steam Engine Makers' Society: An Introduction

The records of the Steam Engine Maker's Society are a unique source, providing detailed information about many aspects of individual lives from 1835 to the First World War; details concerning unemployment, sickness, aging and migration which are available from no other source. However, they concern not some 'random sample' from the country's population but the membership of a particular trade union. If we wish to use the SEM data to draw conclusions which relate to some wider group, we must understand the particular place of SEM members within the union movement, the engineering industry, and the wider society they lived in.

A brief outline of the union's history

The Steam Engine Makers' Society (SEM) was a union of engineers founded in Liverpool in 1824 which expanded to limited national coverage in the late 1830s. No conventional history of the SEM has been published and what we know of it comes from its own reports and incidental mentions in studies of other unions. It took part in the discussions which led to the creation of the Amalgamated Society of Engineers (ASE) in 1851 but the majority of the membership rejected the advice of the Liverpool-based executive and chose to stay independent. Although it never achieved great size or prominence, it was always the ASE's greatest rival for its fundamental recruiting base, the skilled fitter and turner. It finally merged with the ASE in 1920 to form the Amalgamated Engineering Union, now the Amalgamated Engineering and Electrical Union and still one of Britain's largest unions.

Table 1: Growth of the SEM

Year	SEM Members	No. of branches	Year	SEM Members	No. of branches
1835	442	10	1880	4,134	85
1840	876	20	1890	5,822	94
1850	2,068	52	1900	8,566	107
1860	2,107	59	1910	13,401	144
1870	2,783	67	1919	29,199	225

Source: SEM *Annual Reports*

Known characteristics of the membership

Writing in the 1890s, Galton, research assistant to Sidney and Beatrice Webb and himself a former skilled tailor, described it as always having 'looked more to its organisation as a benefit society than for disputes in the workshop', as a result of which it:

exercised great care in the admission of new members .... It appears always to have consisted of a select body of steady workers, whose interests lay more in securing regular employment for themselves than in raising the minimum rate of wages throughout the trade. (Webb T.U.Coll. E.A. XV f.178)

The best way to characterise the SEM's membership is in relation to the larger and better known Amalgamated Society of Engineers (ASE). They grew at a similar rate, the mean annual rate of increase of the SEM between 1851 and 1911 being 4.38% (see table 1) while that of the ASE was 4.42%; by contrast, engineering employment grew by 2.04% and the population as a whole by 0.84%. Given that most admissions were of men between twenty and twenty-five, this meant that the age structures of both unions were markedly younger than that of the working population as a whole.

Table 2: Regional distribution of SEM membership c.1890

Region:	SEM (end 1890)	ASE (Jan 1891)	Male Mech. Engineers	Total Empl. (1891)
South East	20.45	17.72	16.09	27.36
East Anglia	1.96	0.48	1.13	3.08
South West	0.31	3.62	3.74	7.25
West Midlands	7.22	5.58	8.15	8.08
East Midlands	5.75	4.15	5.16	6.42
North West	40.85	28.09	19.97	15.33

Yorkshire	9.93	13.48	13.94	9.05
North	8.97	13.58	12.19	6.09
Wales	4.02	3.29	3.13	5.30
Scotland	0.54	9.99	16.49	12.05
(N)	5,721	60,487	262,643	14,499,732

Sources: SEM *Annual Report*; 1890; ASE *Monthly Report*, Jan. 1891; C.H. Lee, *British Regional Employment Statistics* (Cambridge, 1979)

Table 2 shows the percentage of the SEM's membership in each region and compares it with the ASE, all male mechanical engineers (Industrial Order 7), and the total employed population. Clearly, the SEM was particularly concentrated into the north-west, and barely existed in Scotland and south west England, but otherwise the two unions had broadly similar distributions. Table 3 shows the craft composition of the unions, again very similar. In general, turners would be mainly employed in firms manufacturing machinery while fitters might also be employed in maintenance work by purchasers of steam engines and other machines. These occupations were an elite within the working classes, union standard rates for fitters and turners in 1872 varying from 23 shillings per week in Edinburgh and 24.25 shillings in Greenock through 28 shillings in provincial towns such as Wolverhampton, Reading and Newport to 32 shillings in major northern manufacturing centres such as Sheffield and Manchester and 36 shillings per week in London (Board of Trade, 1908). In the same period, farm labourers were struggling to raise their wages from 10-12 shillings per week, and of course there were no taxes to offset this disparity. Actual wages paid varied around the union standard rate, and if SEM members were the better workmen, and more regularly employed, their earnings would have been somewhat higher.

Table 3: Craft composition (%) of SEM and ASE membership

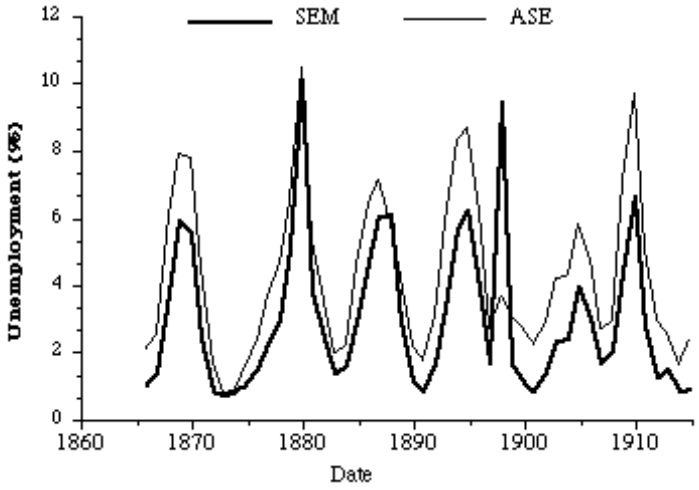
Craft	**SEM**	**ASE**
Fitters	59	59
Turners	30	26
Pattern Makers	5	2

Millwrights	1	1
Smiths	5	7
Other	-	4

Sources: SEM and ASE *Annual Reports*

Figure 1 presents crude unemployment rates in the SEM and ASE over a fifty year period; the SEM figures here are not adjusted for strikes, so that a lock-out in 1897-8 has a major impact. With this exception, the two unions had very similar experience, dominated by regular cyclic recessions. The SEM seems to have had slightly lower unemployment rates in the trough of most recessions, the exception being 1879. This may indicate that the SEM recruited an elite of workers who employers tried hardest to retain during lay-offs. Regional variation in unemployment has not been analysed systematically for the SEM, but crude rates for individual towns seem to have been broadly similar to those for the ASE: in general, members in northern industrial towns were much more likely to have been affected by cyclical distress (Southall, 1986).

Figure 1: Unemployment in the SEM and ASE, 1865-1914



The industrial policy of the SEM seems to have always been to follow the lead of the ASE. The clearest evidence for this is a series of interviews given to the Webbs by trade union organisers in the 1890s; for example, the ASE organiser for Wales and the south-west said he 'had no objection to the SEM, although he regards it as merely a friendly society. But he says its members here always do what he tells them and act in every way under his orders, and so it does not matter much. The two societies work well together having joint committees in cases of dispute' (Webb T.U.Coll., BLPES, E.A.XV, f.205). Although the SEM always followed an agreed line, in joint votes on industrial issues it was

generally more moderate; in five ballots between 1897 and 1908, the SEM membership were consistently more willing to accept employers' or arbitrators' proposals, often by a large margin (Weekes, 1970, pp.103, 106, 114, 224, 254).

Summing-up, the SEM's membership superficially closely resembled that of the ASE, but Galton's comments and other non-quantitative evidence suggests the union consisted of a still more selective group among skilled engineers, themselves an elite within the Victorian working class. However, the SEM Database was built not because of the union's significance for labour relations but because of the benefits it offered its members.

A description of the benefits provided by the union

The various welfare benefits are discussed here in the order they developed:

Travelling Relief

The only benefit provided under the SEM's initial 1827 rules was travelling relief, whereby a member seeking work could claim from each branch visited 'his supper, one pint of beer, one night's lodging, his breakfast in the morning, and one penny for every mile he may have travelled since last relieved' (article 14). The tramping system this benefit formed part of declined in importance after 1850, and the union supplemented it by the payment of members' train fares when travelling to known vacancies. This aspect of the union's work will not be further discussed here (see Southall, 1991 a and b).

Sick Pay

The SEM, like most artisan unions and also like thousands of small local friendly societies across the country, provided members with an income when sick. The SEM's 1846 rules made the following specific provision for sick members:

That any free Member (namely, who has been in the Society one year) when visited by sickness or lameness, (not occasioned by drunkenness or fighting, or any disease improperly contracted, ...) shall give notice of his disposition in writing ... the Secretary ... shall give him a note ... entitling him to the sum of One shilling and Eightpence per day, for each working day, for the space of Six Months ...; no Member shall receive benefit for less than three days. .... Should a Member's indisposition [exceed] six months, he shall receive the sum of Five Shillings per week for Six Months longer; and should his indisposition continue

longer than that period, and he be judged by a physician or surgeon to be incurable ... he shall receive the sum of Three Shillings and Sixpence per week, and be allowed to do what he can for further support. Should doubts arise in the minds of the Members of [his] branch, [they] shall appoint a Medical Adviser to investigate his case at the expense of the Society. (article 25)

Superannuation

The SEM also paid a lump sum benefit of ten pounds on the death of a member, and five pounds on the death of their wife; the same rates applied from 1846 through 1878. This was intended to pay for a funeral, a wake and other immediate expenses, not as long-term provision for dependents.

Fourthly, the 1846 rules provided:

That any person who has been a Member of this Society for the period of Twenty Years from the time he entered, **and shall be rendered incapable of following his employment, through old age or any infirmity**, shall receive a donation of Four Shillings per week for life, and be allowed to do what he can for further support. (article 34; emphasis added)

The 1865 rules labelled this provision 'superannuation' and varied the rate of benefit with the length of membership prior to it first being claimed (article 35, clause 12):

for a twenty years' membership, five shillings per week for life...
a twenty-five years' membership six shillings per week for life...
a thirty years' membership seven shillings per week for life...

The 1878 rules added further rates of eight, nine and ten shillings per week, corresponding to 35, 40 and 45 years of membership (SEM, *Rules*, article 31, clause 2). It will be clear that this provision differed significantly from a modern pension because it might be claimed by men in their forties but was conditional on the member's medical condition. Table 4 shows that numbers on superannuation were unsurprisingly low in the early years of the union but became significant in the 1890s.

Table 4: SEM Members receiving superannuation, 1860-1915:

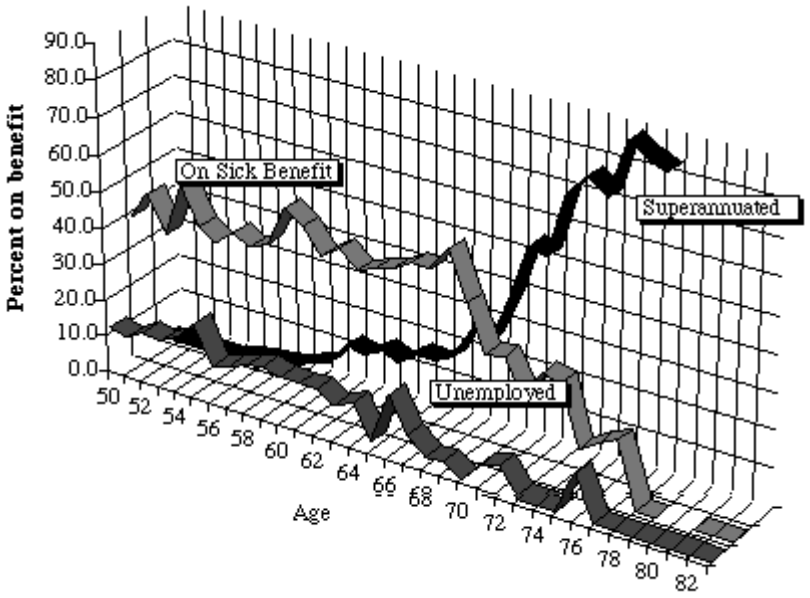
Year	Total Members	No. Superannuated	% on Super-annuation
1860	2050	5	0.2

1865	2521	15	0.6
1870	2783	21	0.8
1875	3871	32	0.8
1880	4134	59	1.4
1885	5062	80	1.6
1890	5822	102	1.8
1895	7085	167	2.4
1900	8566	203	2.4
1905	10645	281	2.6
1910	13401	410	3.1
1915	19664	481	2.5

Source: SEM *Annual Reports* for 1888 (p.34), 1905 (p.xxxvi) and 1915 (p. lxxxvi).

The following figure shows the relationship between superannuation and other benefits over the life course. In particular, as members moved into their late 60s claims on sickness benefit declined as the less healthy members claimed superannuation.

Figure 2: Take-up of benefits from age 50 (data from 1852-72)



Unemployment Pay

Unemployment benefit on modern lines, meaning a weekly payment rather than the aid to job-search provided by travelling benefit, was introduced by the SEM's London branch in 1838 without consulting the rest of the union (SEM, *Annual Report*, 1837-8 , p.42). This was formalised the following year, London members paying increased contributions to cover the cost (*ibid*., 1838-9, p.47). The first outside London listing unemployment benefit payments was Manchester, in the 1844-5 *Annual Report*, and by 1848-9 it was being paid by the branches in London, Manchester, Woolwich, Greenwich, Romford, Stratford (East London), Southampton, Patricroft, and Bristol. The SEM introduced unemployment pay comprehensively following the 1848 recession, when it was seeking to re-establish itself following the defection of large numbers of members to the newly-formed ASE; the ASE took most of its leadership and methods of operation from the *Journeymen* Steam Engine Makers' Society (JSEM), which had always relied on unemployment pay rather than tramping.

Other Benefits

Other benefits were an accident grant of fifty pounds for any member 'rendered incapable of following his employment, either by loss of limb or eyesight' (*Rules*, 1846, article 38), a separate 'Contingent Fund' created in the 1870s to provide strike pay, and a discretionary Benevolent Fund which would seem to have been the only provision for widows and orphans. In return for these benefits, members paid two shillings and three pence per month subscription in

1846, raised to three shillings per month in 1851; this was significantly less than the ASE, which charged a shilling per week. There was also an admission charge dependent on age, varying from 10 shillings and six pence for men under 23 to five pounds for men aged 45; men aged over 45 could not join.

The Records of the Steam Engine Makers' Society

The SEM is notable mainly because of the extremely detailed records which survive, mainly in printed form. This document describes the main sources and known holdings. Note that by far the most comprehensive set of *Monthly* and *Annual Reports* is held by the Bishopsgate Institute in the City of London.

Annual reports

These are the principal source used for the construction of the database, and their most remarkable feature was the quantity and variety of information about individual members which was included almost continuously from 1835 to 1919; a sample page is included as figure 2. The main section of each *Report* consisted of branch reports which begin by listing the members, by name and in descending order of seniority. Along with the member's name is various information about them including: an explanation of what had happened to men disappearing from the list (in general: died, gone abroad, excluded for non-payment, or transferred to another branch); payments for travel; unemployment payments; contingent benefit; superannuation expenses; sick expenses and funeral expenses. The branch records then end with a list of miscellaneous expenses, generally administrative but including payments to members on union business, and a financial summary for the branch. The Annual Reports also include financial tables for the union as a whole, names and addresses of branch secretaries, lists of superannuated members, including those granted superannuation during the year and those dying, and lists of recipients of *ad hoc* payments from the benevolent fund.

Known holdings are:

Amalgamated Engineering & Electrical Union, Peckham, London	1878-1897, 1899-1918
Bishopsgate Institute, London	1836/7-1849/50, 1852, 1853-69, 1871-1918
British Library of Political and Economic Science, London	1835/6-1837/8, 1839-40, 1881, 1884-87, 1890-94, 1909
Modern Records Centre, Warwick University	1878-1919
Trades' Union Congress Library	1914, 1918
Warwick University Library	1875-82,1885-1907, 1909-12, 1915
Working Class Movement Library, Salford	1911-1912, 1918

Our work uses the almost complete run held by the Bishopsgate Institute in the City of London, plus the 1835/6 report from the British Library of Political and Economic Science.

Monthly reports

The earliest monthly reports were single sheet newsletters, including short reports from the individual branches covering both the state of trade and industrial disputes. A report from the General Committee of Management, or later the Executive Council, was followed by a list of members dying, going abroad and excluded; once a quarter, the income and expenditure of the General Committee was listed, as were the results of union ballots when relevant. From 1875, the report also lists applicants for admission and generally includes their

age and some details of their previous history; for example, where they served their apprenticeship. From the 1870s onwards, the reports also include occasional tabulations listing wage rates and normal hours worked for each branch. However, although the monthly reports increased in length they never contained much systematic information and are mainly of interest for their detailed reports on local events and conditions. Known holdings are:

Bishopsgate Institute, London	1848-1903; 1861-1906; 1907-1919 (i.e. two separate runs cover 1861-1903)
British Library of Political and Economic Science, London	Dec 1913- Jun 1914, Dec 1916-Nov 1917
Modern Records Centre, Warwick	1907-18

Rulebooks

These document the union's workings, and in particular lay down rates of benefits, entitlement and so on. Known collections are as follows:

Bishopsgate Institute, London	1827, 1848, 1851, 1854, 1858, 1861, 1865, 1873, 1879, 1889, 1903, 1905, 1915, 1927
British Library of Political and Economic Science, London	1846, 1848, 1878; 1889
Trades' Union Congress Library	1826, 1878, 1889
Working Class Movement Library	1915

Manuscript records

The SEM Registration Books, 1853-1920, are held by the Modern Records Centre and list members joining by name, age , branch joined, trade (e.g. fitter), marital status at entry, date of admission, whether excluded and, if later re-admitted, the date and the name of the branch. The SEM Admission Schedules, 1900-1919 are held at the Peckham headquarters of the successor union, the Amalgamated Engineering and Electrical Union (AEEU) and consist of forms filled in for each recruit: age, trade, name of employer, and current wage. The MRC also holds the Bolton branch contributions book for 1827-37, listing members' names and their monthly contributions, and the Wigan branch minutes for 1844-90.

The SEM Database: Overall Structure

This document provides an overview of the SEM database.

Introduction

The information contained in the database includes:

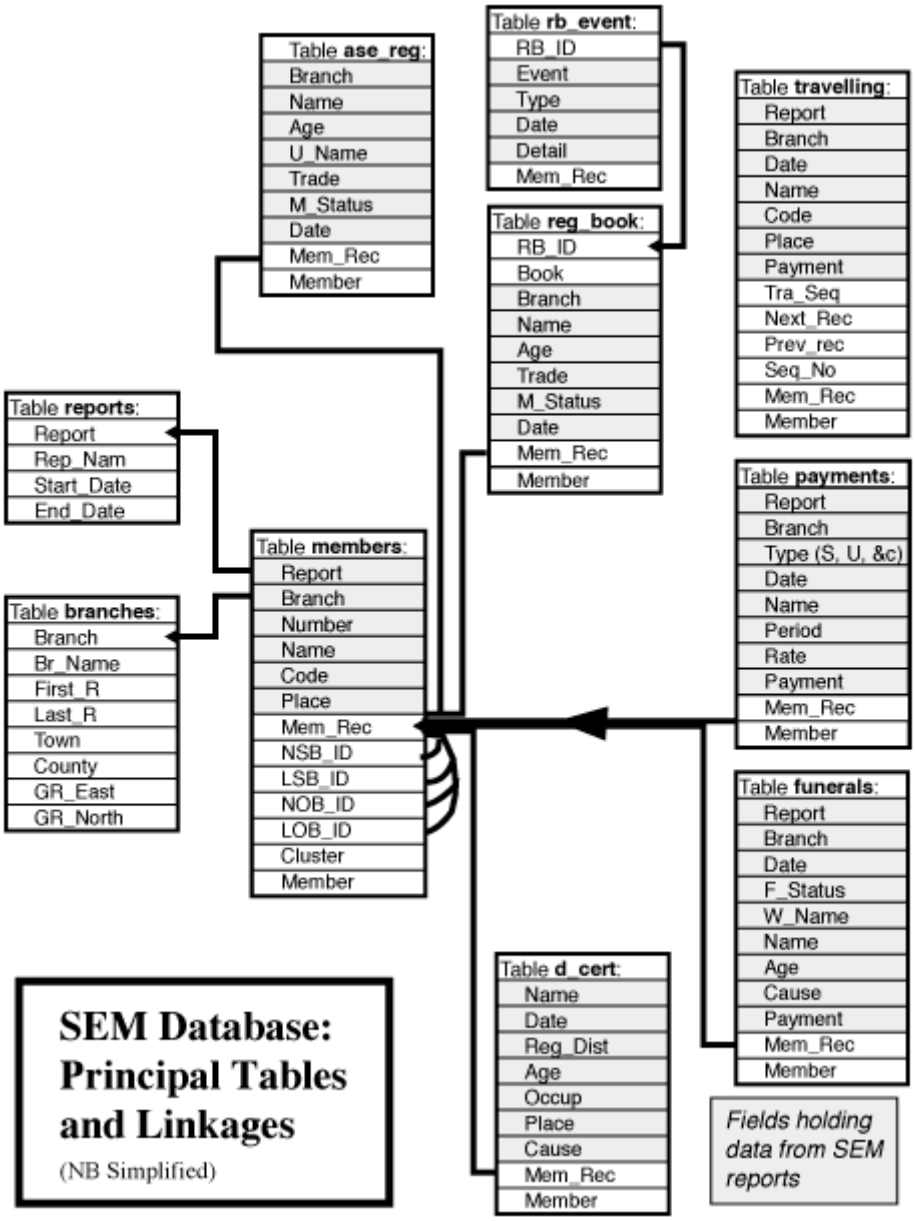
- **All** the information we have transcribed from the SEM *Annual Reports*, which amounts to an almost complete transcription of information concerning named individuals in all reports up to and including that for 1876.
- Selected information from other historical sources which extends our knowledge of the SEM membership: admissions registers, death certificates, etc.
- Additional fields added to the tables by us which links together this information to construct personal histories.
- Look-up tables which expand codes included in the data tables.

NB while the database consists of a series of conventional data tables, which could each individually be loaded into a spreadsheet or statistical package (e.g. SAS or SPSS), this would be utterly pointless! Almost all questions of any real interest involve relational joins within the database. It is designed so that the appropriate queries will create either a very close approximation to the original reports or life histories for individual members. Appendices A and B illustrate usage.

The database is currently hosted on an Internet-accessible Sun SPARCstation 20 in the Department of Geography ('sun1.geog.qmw.ac.uk'). We are currently using the INGRES relational database package: the data entry system was written using the INGRES 4GL language and is highly specific to the package, so no documentation is provided here; however, the examples of SQL commands and the embedded C program in the appendices are reasonably generic. In the longer term, it is intended to move the database over to Oracle software.

The diagram below shows the structure of the database, arrows indicating how records are linked, and distinguishes fields containing data input from the reports from those holding linkage data. Two points should be obvious: the 'members' table, based on the branch membership lists which appear in every *Annual Report* is the key to the entire structure; and the database scarcely obeys the relational model despite being implemented using a relational database package. There are various detailed pragmatic reasons for this, but the more general reason is that we have no definitive list of the fundamental entities with which we are concerned: the individual members of the union. There was no master register - the union's Registration Book only covers part of the period and does not appear to list all members joining after it was started - and the union never used a system of membership numbers to identify individuals. We have therefore had to assemble career histories using members' names which were neither unique - there are many a 'Thomas Smith' - nor consistent - the same man might sometimes appear as 'Tom Smyth'.

While the reports provide many clues to help resolve these ambiguities, each membership history we have constructed must be seen as the result of a process of inference in which many of the individual decisions can be contested. The data tables consist partly of raw data which was directly and unproblematically transcribed from the original sources, and partly of additional fields we have added which link records together - in particular, the values of 'member' in the various tables in some senses serve as 'membership numbers', identifying all records for what we believe to be the same man. The 'lifeline' program listed in appendix A follows these linkage fields to assemble 'life histories' for individuals. However, as the next section explains, the linkage is currently incomplete, while users have the option of completely discarding our linkage and starting again with the raw data.



Current Limitations of the Database

Research on the SEM is on-going: we hope to add both more data, extending the system beyond 1876 as well as filling certain gaps, and also further linkage software which will both increase the data available for existing analyses and permit new types of analysis, particularly of mobility. The following limitations should therefore be noted:

1. Most obviously, the current database is limited to the SEM *Annual Reports* from 1835, as far as we know the first to survive, to 1876. However, a continuous run of SEM reports is held by the Bishopsgate Institute to 1920, when the SEM merged into the AEU, and our long-run

goal is of course to create a database covering the full 1835-1920 period. Extension to the early 1880s will cover the major recession of 1879 and permit cross-linkage to comprehensive 1881 individual census data.

2. No copy of the 1870/71 report has been located, while the reports covering 5/8/1849 to 5/8/1850 and 25/12/1857 to 1/6/1858 (referred to as the 1849 and 1857 reports) lack membership lists. The lists in the 'members' table were created by us by firstly including all members whose appearance in both the previous and subsequent lists showed they had remained continuously in the relevant branch over the missing period, and then adding those men listed in the union's *Monthly Reports* (also at Bishopsgate) as having died or been excluded in this period. This was very time-consuming, but still meant that we cannot trace many members, especially if they changed branch during the period of missing data.
3. The database does not currently include data for United States branches, which submitted financial statements in dollars and cents; the data system is being modified to cover multiple currencies, and once this is done this relatively small number of records will be added.
4. In general, men always moved up branch membership lists to maintain strict seniority order, and the linkage system relies on this. However, there are some unambiguous cases where members swapped positions; software could be written to cope with this.
5. Record linkage software has still to be written to trace men who were listed as changing branches. This is needed both to permit a more detailed study of geographical mobility and to extend the number of man/years of known age: at present, we only know a member's age while he was in either the branch he first joined (using ages in the Registration Book) or the branch he died in (using ages stated with Funeral Benefit payments or on death certificates). For such a mobile group, this is a major problem. Where footnotes to the membership lists state the branch a man moved to, construction of a linkage system is straightforward.
6. Similarly, linkage software has yet to be written to trace men moving from branch to branch claiming Travelling Benefit (hence the absence of any pointers involving this table in the structure diagram). The procedure for doing this is clear and the necessary fields for adding pointers have been included in the 'travelling' table, but the software required is quite different from any existing linkage tool so writing it is a substantial task. Once written, it will both permit detailed analysis of 'tramping' and permit many men to be traced who were listed in footnotes as having moved, but with no stated destination.
7. All existing linkage tools are designed to link one particular record to another specific record. However, there are limits to this approach: there is no systematic way of linking a sequence of travelling benefit payments

to the same man's membership records; some men changed branches without their destination being stated; others left or were expelled and then rejoined sometime later, or went abroad then returned. Under these circumstances we need to be able to view on screen the whole of two or more clusters of records, and then decide whether or not a pair of clusters in fact relate to the same man. This decision is complex, necessarily somewhat subjective, and must bear in mind whether the man's name is commonplace or unusual: if a 'John Smith' is expelled by Bolton branch and then another joins two years later, no link can sensibly be made but for 'Bigland Grundy' (a real example) we might decide differently. Such decisions need to be taken by the principal investigator and a quite different linkage tool needs to be written, using a windowing interface and permitting clusters of records to be examined, each cluster in its own window in various optional levels of detail. We have begun to specify this tool but it remains to be written. NB the separate 'member' and 'cluster' fields in the 'members' table are designed to support this.

8. At present, members who were 'excluded' (expelled, usually for arrears) from the union will not be included in analyses of the year in which they were excluded, as we do not know the date of the exclusion or, therefore, the number of days during the year for which they were members. Post 1850, excluded men were not listed in branch membership lists for the year in which they were excluded, so analyses will also ignore them for the previous year as the system assumes they left at some unknown point in that previous year. However, post-1850 such excluded members **do** appear in a separate list right at the back of each *Annual Report*. This seems to explain there being many unlinked benefit payments - they were to men who belonged to a branch but only appear in the list of exclusions. New software needs to be written to add these lists of excluded members to the membership lists, add the relevant footnote, and link them to the previous year's list. Ideally, this software should go further and permit the addition of lists of excluded members from the *Monthly Reports*. These indicate the month in which a man was excluded and means we can analyse the final year of membership, in the same way as is already possible for men who died. This matters because, just as men who died were particularly likely to receive sickness benefit, excluded members were particularly likely to have been unemployed.
9. In general, we have gone to great lengths to discover the ages of members, linking in data from several other sources. However, there are still some men for whom we know the date and place of the funeral, lack the age, but have yet to obtain Death Certificates - but then, they do cost £6 each.
10. As yet, funeral benefit payments for members' widows have still to be linked to the 'widows' table, and entries in that table need to be linked to

the relevant deceased member's funeral and membership records. Variants of existing tools will perform these tasks but are not a high priority.

Standard Definitions

The remainder of this document describes each of the individual tables in the database in turn. However, we have tried to standardise the organisation of similar information in different tables, and follow certain other general principles, so please note the following.

Where an entry in the 'contents' column of a table definition appears in bold, it is a Multi-Field Data Structure or Standard Integer Code as defined below.

Multi-Field Data Structures

Certain types of information are naturally organised into multi-field structures in which the individual fields always have the same names and types in whatever table the structure appears in. To save space, the table descriptions that follow refer simply to the structures, indicated by [field names in brackets], and their internal organisation is explained here:

Structure	Field	Type	Contents
Name	**cname**	Text (60 chars.)	Forename(s) of member. NB where a name has three elements, such as 'John Jenkin Jones', the division between forename and surname was made by the operator and may be arbitrary.
	sname	Text (40 chars.)	Surname of member.
	suffix	Text (20 chars.)	Suffix to name as given in report, e.g. 'Jun.', 'Sen.', '1st'. These usually only appear where there were two or more men within a branch with

			identical names, so if a man moved between branches he would not usually retain the suffix.
Date	**year**	Integer	Year part of date as listed in original report.
	month	Integer	Month part of date as listed in original report.
	day	Integer	Day part of date as listed in original report.
	date	Database internal date format	Date value computed from 'year', 'month' and 'day'. NB this will be null if (a) the original report did not give a complete date or (b) the date given was in error - e.g. 30th Feb.
Money	**currency**	Text (1 char.)	Currently always null, but handles foreign currencies e.g. US dollars.
	pounds	Integer	Pound part of the sum of money.
	shills	Integer	Shillings part of the sum of money.
	pence	Floating point	Pennies and fractions of pennies.
	value	Floating point	Computed as ('pounds'*240 + ('shills'*12) + 'pence'.

Standard Integer Codes

To simplify linkage and analysis, certain standard coding systems are used in a number of different tables in the database. These are all stored as integer fields,

but in the table definitions are described by the particular type of integer code they contain.

Report All records in the tables described in part 2 of the table definitions are drawn from one of the *Annual Reports* of the SEM, and the value of 'report', included with **every** record, indicates the volume in question. These values are integers which **broadly** indicate the year of the report, but the precise period covered by each report varied substantially over the union's history -- only from 1875 onwards do the reports precisely cover calendar years. For earlier periods, the value of 'report' is always the calendar year of the first day covered, but for example the report labeled '1850' covers 5/8/1850 to 25/12/1852, and '1852' covers 25/12/1852 to 25/12/1853. Precise information on the period covered, and a more accurate label for each report, is given in the 'reports' table.

Branch Similarly, almost every record drawn from the reports also comes from the report of a particular branch of the union, and this is identified by a numerical code. The 'br_names' table is used to supply these codes and contains alternative forms of each branch's name (e.g. 'St Helens', 'St.Helens', etc) while the 'branches' table contains conventional names and other information for each branch code. These codes were in fact allocated in order of each branch's first appearance in the reports; where a branch closed and later re-opened, it is allocated a new code.

Record_ID Fields whose type is **Record_ID** play a central role in linking together the database to construct life histories. Each value of this type is a 10-digit numeric code which identifies a particular entry in the 'members' table:

Branch from which record comes	Most significant 3 digits
Report from which record comes	Next 4 digits
Number of the record within branch list	Least significant 3 digits

This system both uniquely identifies each membership record and supplies useful information about it; for example, '21836004' indicates the fourth member listed in the 1836 report of the Bolton branch. Therefore, writing **Record_ID** values into records of benefit payments links a payment to the relevant membership record. However, these values are also used to link one membership record to another, and in particular the system links each record to the **first** record for a given man - so these pointers to the first record identify all records for a member, and therefore the relevant **Record_ID** value identifies not just the one record but the man's history as a whole, and serves as a pseudo-membership number in analytical work.

Methodological rules affecting all tables

A few other general points should be noted:

Codes: Although various codes are used in the tables to replace text appearing in the *Annual Reports*, to simplify analysis, the relevant wording in the reports was **almost always** highly standardised and where there were alternative phrasings we use multiple codes. For example, the 'code' field in the 'members' table summarises the footnotes to the membership lists: if a man was listed as 'Drawn clearance' the code is 'C' while if he 'Joined another branch' it is 'B', even though the two are almost synonymous. Where more

specific information was given, we have summarised it as a code but also included it verbatim in the 'comment' field.

Comments: **Every** data table includes a 'comment' field, which can be used to contain either explanatory information given in the reports or comments by the operator [the latter should always appear in square brackets like this]. Comments in the original sometimes include systematic information which needs to be available for analysis, such as the destination of a migrant or a cause of death, and wherever possible this information is placed in separate fields.

Date Values: Although the tables deposited were created by simply dumping all individual tables from INGRES and removing unnecessary spaces, etc., **almost all** fields contain either conventional text strings or integer numbers; the few floating point strings were created to hold fractional values such as half pennies or farthings in sums of money or ages expressed in years and months, so there are no problems with truncation. However, the tables do include various dates in a special format used by INGRES but fairly self-explanatory: the day of the month, the month expressed as a three-letter string, and the year, each separated by hyphens - for example, '02-jan-1837'. As the relevant tables always include, in addition, the year, month and day stored as three separate integer fields, dates in other proprietary formats can easily be created.

Indexing: **All** tables in the RDBMS are indexed by 'member', and many should also be indexed by 'branch' and/or 'report'.

Naming: Given that the SAS statistical analysis software is used for analytical work and directly accesses the data tables via the SAS Access module, we have tried to follow SAS naming conventions for **almost all** table and field names: maximum 8 characters, beginning with a letter. However, we have broken this rule when necessary to provide more meaningful names.

Project Management Tables

The following tables do not contain raw data but are required by the data entry system:

- reports: Details of the *Annual Reports*.
- br_names: Converts all variants of branch names to codes.
- branches: Converts branch codes to names and provides other information.

Table 'reports'

Function: Holds information about each SEM report, principally the period covered. NB most reports did not cover a calendar year, and several cover periods of more or less than twelve months.

Coverage: All *Annual Reports* published by the SEM between 1835 and 1880.

Size: 45 records.

Table 'reports'

Field Name	Contents	Usage
report	**Report**	The calendar year in which the first day covered by the report fell. Used to identify

		the report in other tables.
rep_name	Text (16 chars)	Standard name for report, for use in output (e.g. '1839/40' or '1853').
start_dy	Integer	Day within month of start of period covered by report.
start_mn	Integer	Month of the start of the report.
start_yr	Integer	Year of the start of the report.
start_date	**Date**	Date of first day covered by report, computed from above 3 integers and stored in RDBMS' internal date format.
end_dy	Integer	Day within month of end of period covered by report.
end_mn	Integer	Month of the end of the report.
end_yr	Integer	Year of the end of the report.
end_date	**Date**	Date of last day covered by report, computed from above 3 integers and stored in RDBMS' internal date format.

notes	Text (128 chars.)	This contains notes, e.g. the fact that no copy of the 1870/1 report has been found.

Table 'br_names'

Function: This table is used to look up branch names that appear in the report during data input to find the branch code. It provides alternative versions of branch names and handles any text string used to identify a branch. However, it does not handle all place names that may appear in the SEM data (e.g. destinations of emigrants).

Coverage: All text strings used to identify SEM branches at the start of the branch sections within the Annual Reports that appear in any report up to 1879.

Size: 285 records.

Notes:

1. The values of 'br_name' are not unique, as there are many examples of a branch closing and then another branch with the same name opening some years later, so to obtain a branch code a check must be made that the relevant report falls in the range first_r <= report <= last_r.
2. The precise format of some of the values of 'br_name' depended on how they were typed in by the data entry operator.

Table 'br_names':

Field Name	**Contents**	**Usage**
br_key	Integer	Unique string generated to index the table.
br_name	Text (70 characters)	The name as entered.
first_r	**Report**	The first report in which this branch appears.

last_r	**Report**	The last report in which this branch appears.
branch	**Branch**	The corresponding code for the branch which appears in the branches table.

Table 'branches'

Function: Holds information about each SEM branch identified by a unique code for each one. The values of branch are, effectively, accession numbers: Liverpool is '1' because it was the first branch, and subsequent branches have higher numbers; when a branch closed, those branches further down the list move up (but while their number in the printed report changes our code number of course stays the same).

Coverage: All branches ever existing in the SEM up to those listed in the 1878 *Annual Report*.

Size: 129 records.

Notes:

1. This table was constructed by systematically working through the SEM *Annual Reports* and includes ALL branches of the union that existed during the study period. This table is used when creating output to replace branch code numbers by the name of the branch, but other information, such as parent and successor branches, could not always be found.
2. In constructing the table there were a few problematic cases where we had to decide whether a branch which changed its name from one report to the next was in fact the same branch, and others where a branch closed and re-opened shortly afterwards.

Table 'branches':

Field Name	**Contents**	**Usage**
branch	**Branch**	Unique identifying code, used to identify the reporting branch in all the data tables. Liverpool = 1 and all subsequent branches get higher numbers in sequence.
br_name	Text (30 chars.)	This is a 'standard' version of the branch name, to appear in output.
first_r	**Report**	First report in which this branch appeared.
last_r	**Report**	Last report in which this branch appeared.
town	Text (20 chars.)	Used in the analysis and, where there was more than one branch in a town, to identify possible branches in linking tramping data, etc.
county	Text (20 chars.)	Registration county. Used to aggregate branches in the analysis.
gr_east	Integer (3-digits)	Ordnance Survey grid reference easting (in km); used to locate branches in the analysis.
gr_north	Integer (3-digits)	Ordnance Survey grid reference northing (in km); used to locate branches in the analysis.

o_day	Integer (2-digits)	Day when the branch was opened.
o_month	Integer (2-digits)	Month when the branch was opened.
o_year	Integer (4-digits)	Year when the branch was opened.
o_date	RDB date value	Date when the branch was opened; will be null unless day, month and year are all available.
c_day	Integer (2-digits)	Day when the branch was closed.
c_month	Integer (2-digits)	Month when the branch was closed.
c_year	Integer (4-digits)	Year when the branch was closed.
c_date	RDB date value	Date when the branch was closed; will be null unless day, month and year are all available.
parent	Text string (30 chars.)	Branch out of which this branch was created', inferred from report. Members transferring from the parent to the child branch in the year of creation will not necessarily have moved home or job.
p_code	Integer	Branch code of parent branch. Currently unset.
success	Text string (30 chars.)	Branch to which remaining members transferred on closure, inferred from

		report. Members transferring from the branch to its successor in the year of closure will not necessarily have moved home or job.
s_code	Integer	Branch code of successor branch. Currently unset.
notes	Text (512 chars.)	Comment

Annual Report Data Tables

The following tables hold the raw data from the *Annual Reports*:

- members: Information from membership lists and footnotes.
- widows: Information from membership lists and footnotes concerning members' widows.
- payments: Payments of all benefits other than travelling and funeral.
- travelling: Payments of travelling benefit.
- funerals: Payments of funeral benefit.

Table 'members'

Function: This holds all information taken from the membership lists and the footnotes, together with a number of fields needed by the linkage process. Note that many of the fields contain numerical codes pointing to other records in the same table.

Coverage: All branch membership lists appearing in the SEM *Annual Reports* between 1835 and 1876.

Size: 93,910 records.

Notes:

1. The table also includes synthesised membership lists covering the 1849, 1857 and 1870 reports, to replace missing data (see database overview).

Field Name	**Contents**	**Usage**
report	**Report**	The report from which the record comes.
branch	**Branch**	The branch from which the record comes.
sys_no	Integer	The member's actual ranking within the list, generated automatically by system.
list_no	Integer	The member's number as printed in the list; NB this is usually identical to sysgen_number', but not always.
[name]	**Name**	The member's name, as it appears in the list.
code	Text (1 char.): legal options are: A **A**broad B joined other **B**ranch C drew **C**learance D **D**ead E **E**xcluded L '**L**eft' (ambiguous) M **M**ember of other branch P **P**roposition	A code selected by the operator to summarise the footnote entry.

	S **S**uperannuated T drew **T**ravel certificate X **M**iscellaneous	
place	Text (50 chars.): a place name	Usually the destination given in a footnote.
notes	Text (512 chars.)	Other text from the original footnote; not needed where the footnote is of a very standard type.
returned	Text (1 char.): either blank or 'R'	Flag indicating that the member was footnoted as having returned.
r_number	3-digit integer (should be <null> unless RETURNED set)	New membership number for the same member on returning to the branch.
mem_seq	Integer	Unique accession number, generated by data entry system.

'members' table (cont.)

These additional fields hold values added during the linkage process:

Field Name	**Contents**	**Usage**
record_id	**Record_ID**	The ID of the record itself, constructed from BRANCH, REPORT and NUMBER.
nsb_id	**Record_ID**	(NEXT_SAME_ID) The ID (if any) of the membership record of the same man in the

		following year's report of the SAME branch.
lsb_id	**Record_ID**	(LAST_SAME_ID) The ID (if any) of the membership record of the same man in the previous year's report of the SAME branch.
nob_id	**Record_ID**	(NEXT_OTHER_ID) The ID (if any) of the membership record of the same man in another branch which he moved to from the present one (which might be in same year or the following year's report). [At present, always null]
lob_id	**Record_ID**	(LAST_OTHER_ID) The ID (if any) of the membership record of the same man in another branch which he moved from to the present one (which might be in same year or the previous year's report). [At present, always null]
clust_id	**Record_ID**	The ID of the first membership record in a clear sequence of membership records relating to the same man. This would be

		initially set as identical to mem_rec.
member	**Record_ID**	The ID of the first membership record in a set of clusters that are ultimately grouped together as referring to the same man. It would initially be set to be identical to mem_rec. [At present, this will always be identical to 'cluster'.]
l_status	Integer (single digit code)	Linkage status: used in linking footnote data. [At present, always null]
move	Text (1 char.): legal options are: A **A**broad B joined other **B**ranch I **I**nferred move to branch N **N**o evidence S **S**witch	Indicates type of move; generated later in the linkage and used for analysis. 'I' covers moves where nothing is footnoted but a move is inferred from the linkage; 'N' cases where a move is footnoted but there is no evidence it actually happened; 'S' changes of branch due to openings or closures. [At present, always null]

Table 'widows'

Function: From 1861 onwards, some branch membership lists ended with a list of members' widows who were entitled to funeral benefit, and this information is held here. NB the fields used are a sub-set of those in the 'members' table, but the data are held separately as separate linkage operations will be needed to use these data.

Coverage: All widows listed in branch membership lists in SEM *Annual Reports* between 1861 and 1876. It also includes synthesised lists covering the 1870 report, to replace missing data (see above), which make some use of information from the *Monthly Reports* for the relevant period.

Size: 441 records.

Field Name	Contents	Usage
report	**Report**	The report from which the record comes.
branch	**Branch**	The branch from which the record comes.
sys_no	Integer, generated automatically by system.	The widow's number within the list; NB not present in original but lists seem to be in seniority order.
cname	Text (60 chars.)	The widow's forename, usually not given.
sname	Text (40 chars.)	The widow's surname, as listed.
code	Text (1 char.): legal options are A, B, C, D, E, L, M, P, S, T and X (see	As for the 'members' table, but the only entries used are D, E, L and X.

	definitions in 'members')	
place	Text (50 chars.): place name	The destination given in a footnote, but currently always empty as no widows are listed as moving.
notes	Text (512 chars.)	Other text from the original footnote; e.g. re-marriage.
wid_seq	Integer	Unique accession number, generated by data entry system.
record_id	**Record_ID**	The ID of the record itself, constructed from BRANCH, REPORT and NUMBER.
nsb_id	**Record_ID**	(NEXT_SAME_ID) The ID (if any) of the membership record of the same widow in the following year's report of the branch.
lsb_id	**Record_ID**	(LAST_SAME_ID) The ID (if any) of the membership record of the same widow in the previous year's report of the branch.
clust_id	**Record_ID**	The ID of the first record in the sequence for this particular widow. This is currently set as identical to record_id

		but will be changed through linkage.
member	**Record_ID**	The ID of the first record in the sequence for this particular widow. This is currently set as identical to record_id but will be changed through linkage.

Table 'payments'

Function: This holds the bulk of the benefit payment data, the TYPE field indicating the particular type of benefit.

Coverage: All payments of sickness benefit to named members listed in the SEM *Annual Reports* from 1835 until 1876; all payments of unemployment benefit starting in 1837; all payments of superannuation from 1842; and all payments of strike pay from 1875. Items of miscellaneous expenditure are also included wherever there is a named beneficiary.

Size: 40,955 records.

Field Name	**Contents**	**Usage**
report	**Report**	The report from which the record comes.
branch	**Branch**	The branch making the payment.
type	Text (1 char.): legal options are: B **B**enevolent Fund (MR data)	The type of payment; NB each type of payment appears in a separate list in the original reports, and

	C **C**ontingent (strike pay) M **M**iscellaneous P **P**ension = Superannuation S **S**ickness U **U**nemployment	these codes are inserted by the data entry system. No benevolent fund data has yet been entered.
[date]	**Date**	The date of the payment.
[name]	**Name**	The name of the member receiving the payment.
weeks	Integer	The number of weeks covered by the payment.
days	Integer	The number of days covered by the payment (additional to weeks).
rate	Text (1 char.): legal options are: <null> Standard rate 3 At 3/6 4 At 4/0 5 At 5/0 F Full pay H Half pay L Lowest rate R Reduced rate	Code indicating the rate at which the benefit was paid (unemployment and sickness benefits dropped to a lower rate after a certain number of months; NB the codes provide for the variety of actual statements in the source and need to be translated to provide the actual weekly rate. Always null for post-1849 data but can be inferred from duration and value.

notes	Text (512 chars.)	Other information which appears; mainly necessary for miscellaneous payments.
[payment]	**Money**	The sum of money paid.
pay_seq	Integer	Unique accession number, generated by data entry system.
mem_rec	**Record_ID**	The RECORD_ID of the entry in 'members' to which the payment is linked.
member	**Record_ID**	Unique ID for member, added by linkage.

Table 'travelling'

Function: This table holds payments of travelling benefit, which differ markedly from the data in the 'payments' table firstly because this benefit usually involved an origin or destination, and secondly because most payments were to non-members of the paying branch, greatly complicating linkage.

Coverage: All payments of sickness benefit to named members listed in the SEM *Annual Reports* from 1835 until 1876; all payments of unemployment benefit starting in 1837; all payments of superannuation from 1842; and all payments of strike pay from 1875. Items of miscellaneous expenditure are also included wherever there is a named beneficiary.

Size: 36,039 records.

Notes:

1. This table in practice contains two rather different types of records. Most records coded as 'Accommodation', 'From' or 'Miscellaneous' are payments under the traditional 'tramping' system, whereby the union supported members as they moved around the country on foot seeking work. During the first twenty years of records, payments for members to go 'To' a destination were rare, and mainly concerned branch officials travelling on union business. However, from the mid-1850s the union paid a growing number of train fares for members being sent by their branch to a vacancy somewhere else. 'To' records will usually be straightforward to link, as they involve members of the paying branch, but even in 1876 were a minority of travelling benefit payments.
2. The table contains a large number of fields, currently unused, designed to support linkage; not all may prove necessary. The objective in linking tramping payments is to construct sequences of moves and this process must work backwards, using the value of 'origin' to determine the previous branch visited and then searching for a payment to the same member made by that branch: the two payment records will then be linked by the 'next_rec' and 'prev_rec' fields. The 'sequence' field identifies all records in the same sequence, and the 'seq_no' field provides the order (NB the date given for the payment is not a totally reliable guide to timing). The initial value of 'seq_no' for all entries will be 255, not 1 and as entries are joined on to the beginning of sequences they get **lower** numbers. The 'l_status' field is needed because there will need to be multiple passes through the data, the first finding obvious matches and later passes adding less obvious ones: 'l_status' will record whether a record has already been linked, but also what attempts have already been made to link it.

Table 'travelling':

Field Name	**Contents**	**Usage**
report	**Report**	The report from which the record comes.
branch	**Branch**	The branch making the payment.
[date]	**Date**	The date of the payment.

sys_no	Integer	Number added by data entry system indicating record's position within a particular branch's list of payments.
[name]	**Name**	The name of the member receiving the payment; NB this is normally **not** a member of the paying branch.
code	Text (1 char.): legal options are: A **A**ccommodation F **F**rom T **T**o X **M**iscellaneous	Type of payment; see notes.
place	Text (80 chars.)	The previous town visited (e.g. 'Newton')
stay	Text (1 char.): legal options = <null>, digits 1-9, S.	Length of stay in days ('S' indicates 'Sunday included')
notes	Text (512 chars.)	Any other text from report.
[payment]	**Money**	Amount of benefit paid.
tra_seq	Integer	Unique code identifying the sequence that this record is linked into; to be added.
record_id	Integer	Unique code identifying the record.

next_rec	Integer	Value of RECORD_ID for next payment in sequence; to be added.
prev_rec	Integer	Value of RECORD_ID for previous payment in sequence; to be added.
seq	Integer	Value of RECORD_ID for **final** record in sequence; to be added.
seq_no	Integer	Number of record within sequence; to be added.
mem_rec	**Record_ID**	Pointer to entry in membership list in branch where journey began; to be added.
member	**Record_ID**	Unique ID for member, to be added prior to analysis.
l_status	Integer (single-digit code)	Linkage status: used later in linkage.

Table 'funerals'

Function: Holds information on the deaths of members, their wives and widows taken mainly from the *Annual Reports*.

Coverage: All funeral benefit payments in the *Annual Reports* up to the end of 1876. Gaps in the *Annual Reports* were filled using data from the *Monthly Reports*.

Size: 1,423 records.

Notes:

1. Age and cause of death were generally only listed from c. 1860 onwards.

Field Name	**Contents**	**Usage**
report	**Report**	The report from which the record comes.
branch	**Branch**	The branch making the payment.
[date]	**Date**	The date of the payment.
f_status	Text (1 char.): legal options are: M **M**ember F Wi**f**e D Wi**d**ow	Category of person dying.
wname	Text (70 chars.)	Wife's christian name.
[name]	**Name**	The name of the member receiving the payment, or of the husband in case of wives and widows.
age	Integer	The age of the dead person.
cause	Text (70 chars.)	Cause of death.
notes	Text (512 chars.)	Any other text from report; the operator will need to use their judgment to distinguish this from 'cause'.
[payment]	**Money**	Sum paid to members

fun_seq	Integer	Unique accession number, generated by data entry system.
mem_rec	**Record_ID**	The RECORD_ID of the entry in 'members' or 'widows' to which the payment record is linked.
member	**Record_ID**	Unique ID for member, added later in linkage; NB in the case of widows and wives identifies the husband.

Other Data Tables

The following tables hold raw data from sources other than the *Annual Reports*: These are particularly important as sources of members' ages.

- reg_book: Information from the SEM's Registration Book for new members.
- rb_event: Information about later events taken from the Registration Book.
- ase_reg: Information about SEM members from the Amalgamated Engineers' Registration Book.
- d_cert: Information from deaths certificates of SEM members.

Table 'reg_book'

Function: Holds information taken from the SEM's manuscript *Registration Books*, containing information on age, trade and marital status for members on their admission to the unions.

Coverage: All pre-1900 entries in the first of the five volumes. The period covered is unclear, as most of the initial entries are for 1853 but the book

includes men joining as early as 1836; similarly, the main sequence of entries run up to 1893 but others from the 1900s were not input.

Size: 5,322 records.

Notes:

1. We thank the Modern Records Centre (Warwick U.) for loaning the volume.
2. These data were input into Filemaker Pro then bulk-loaded into the main database.

Field Name	**Contents**	**Usage**
rb_id	Integer	Unique identifier for each entry in the book. Indicates order of entries and permits related information in the 'rb_event' table to be linked.
book	Integer (all current entries are '1')	Volume from which entry was transcribed. All current data comes from MRC Accession number MSS.259/SE/2/1/1.
br_name	Text (70 chars.)	Name of branch where admitted.
branch	**Branch**	Branch where admitted (standard code).
[name]	**Name**	Member's name.
age	Integer	Age at admission.
trade	Text (70 chars.)	e.g. 'fitter', 'turner'.

m_status	Text (1 char.): legal entries are: S **S**ingle M **M**arried W **W**idowed	Marital status, as given in Registration Book (NB 'W' code is unused).
[date]	**Date**	Date of admission.
notes	Text (512 chars.)	Comments relating to initial entry (NB for info. re. subsequent history see 'rb_event'.
rb_seq	Integer	Unique key value, added while bulk-loading the data; runs in sequence, unlike rb_id, and used in checking data.
mem_rec	**Record_ID**	The RECORD_ID of the entry in 'members' for the year/branch first joined.
member	**Record_ID**	Unique ID for member, added later.

Table 'rb_event'

Function: Holds information concerning subsequent events listed in the SEM *Registration Book*. Most entries in the books contain no such information, but in some cases information about up to five subsequent events appears. This table can be used to make further linkages.

Coverage: All pre-1900 entries in the first of the five volumes.

Size: 2,125 records.

Notes:

1. These records add information to records in the 'reg_book' table to which they are linked by the 'rb_id' field. Some events, such as death or re-admission, can be used to extend the linkage of major clusters of records, most obviously where a man left and re-joined - without the *Registration Book*, there would be no certain way of knowing it was the same man, not two different men with the same name.
2. Much of this information is messy and incomplete, so use with care.

Field Name	**Contents**	**Usage**
rb_id	Integer	Unique identifier for each entry in the book. Links 'rb_event' to 'reg_book'.
event	Integer (1-5)	Sequence number indicating placing of this event within those given for this entry in the Registration Book; to be used to present data in the original order.
type	Single character. Legal entries are: D Died E Excluded L Left the trade M Miscellaneous P Payment R Re-admitted S Superannuated W Wife died	Nature of event. NB 'Miscellaneous' is rare, as the more specific codes are comprehensive.
[date]	**Date**	Date of event. NB in many cases only the year, or the month and year are given.

place	Text (70 chars.)	Place where event occurred; usu. indicates moves to another branch.
detail	Text (512 chars.)	Other details given in register. NB many entries would be fully covered by the code in the type field and the date.
mem_rec	**Record_ID**	The RECORD_ID of the entry in 'members' to which the event can be connected; to be added by linkage
clust_id	**Record_ID**	Cluster ID, copied from the entry in 'members' pointed to by mem_rec.
member	**Record_ID**	Unique ID for member, added later in linkage.

Table 'ase_reg'

Function: Holds information taken from the Amalgamated Society of Engineers' *Registration Book*. Many members of the SEM transferred to the ASE on the creation of the latter union in 1851 (which explains the disruption to the union in that year and the break in the sequence of *Annual Reports*). In principle it includes **all** such ex-members of the SEM, plus a selection of men joining the ASE by other routes; these latter were included to permit research on other early unions.

Coverage: A complex sampling strategy was adopted, mixing a strict random sample to characterise the

book as a whole, an alphabetical sample of pre-1852 members and **all** members listed as having been initially admitted into the 'Engineers' or the 'SEM' (see the 'random_s', letter_s' and 'in_sem' fields). The last criteria was intended to include all men who are relevant to the SEM database, although some SEM members may have left the SEM and then joined another union and be listed there.

Size: 800 records.

Notes:

1. In 1978, I located a Registration Book covering the early years of the ASE in the AEU offices at Peckham and transcribed some of the entries, including those of former SEM members; the book lists members by their initial admission to either the ASE or any of its predecessors. These records provide ages, and appear to be the only source of admission dates for the SEM pre-1850.
2. The data were loaded into the SEM database and this description written in June 1995. At the same time, the *Registration Book* was found again at Peckham and some further data entry work carried out, covering all men who joined any component union before the end of 1841.
3. Strictly speaking, only records with a value of 'in_sem'=1 should be seen as part of the SEM database.

Table 'ase_reg':

Field Name	**Contents**	**Usage**
br_name	Text (40 chars.)	Name of branch where admitted.
branch	**Branch**	Branch where admitted (standard code).
[name]	**Name**	Member's name.
age_y	Integer	Age in years at time of admission

age_m	Integer	Any months stated as part of age at time of admission
age	Float	Age at time of admission; calculated from age_y and age_m.
u_name	Text (24 chars.)	Name of union originally joined. (NB 'union' is an INGRES reserved word)
trade	Text (16 chars.)	Branch of trade (skill) to which man belonged. NB where the record comes from the original sample gathered in 1978, this was originally entered as a single letter code and has been converted back to a full statement when the data were loaded into INGRES, so may not be an exact transcription.
m_status	Text (1 char.): legal entries are: M Married S Single W Widowed	Marital status.
[date]	**Date**	Date of event. NB in many cases only the year, or the month and year are given.
random_s	Integer (Flag: 0 or 1)	Value=1 if record was included as part of random sample (208 entries selected by page and line number).

letter_s	Integer (Flag: 0 or 1)	Value=1 if admission was before 1/1/1852 and surname begins with 'S' (all such entries were transcribed).
in_sem	Integer (Flag: 0 or 1)	Value=1 if record was included because man was apparently an SEM member (all entries where union='ENGINEERS' or 'SEM').
pre_1841	Integer (Flag: 0 or 1)	Value=1 if record was included because the man joined before the end of 1841; all such records are being transcribed.
notes	Text (128 chars.)	Other information from the original register; usu. indicating subsequent history, e.g. death or expulsion.
ar_seq	Integer	Unique identifier, added when data loaded into SEM database.
mem_rec	**Record_ID**	The RECORD_ID of the entry in 'members' to which the event can be connected; to be added by linkage
member	**Record_ID**	Unique ID for member, added later.

Table 'd_cert'

Function: Holds information taken from Death Certificates of SEM members, including age and cause of death.

Coverage: Most SEM members dying whose death was recorded in the union's reports but without age or cause of death information.

Size: 167 records.

Notes:

1. The SEM reports gave the name of the members, the date of their funeral and the branch they belonged to, permitting registered deaths with the right name, date and locality to be identified in the indexes to the registers of deaths available at St. Catherine's House. The only way to obtain further information (age and cause) was to buy the certificates. Where members had common names, multiple certificates had to be bought and the correct one identified by occupation (some form of engineer) and age (SEM members were adults while many duplicate certificates covered infants).
2. Each death certificate cost £5.00 in 1989, so this may be one of the most expensive data sets of its size in existence! Purchase of the certificates was funded by the Wellcome Trust.
3. The 'cause' field often ends with 'CTD.' or 'NOT CTD.', indicating whether or not the information was certified by a doctor.

Field Name	**Contents**	**Usage**
[name]	**name**	The name of the person dying.
[date]	**date**	Date of death (NB this may be different from the date of the funeral).
reg_dist	Text (40 chars.)	Name of Registration District where death occurred.
age	Integer	Usually a round number of years.

occup	Text (60 chars.)	e.g. 'Fitter', 'Turner'.
place	Text (140 chars.)	Place of death: may be precise street address.
cause	Text (140 chars.)	Cause of death
dc_seq	Integer	Unique identifier, added when data loaded into SEM database.
mem_rec	**Record_ID**	A pointer to the entry in the members table for the relevant member in the year and branch where he died; used to associate the data with the main database.
member	**Record_ID**	Unique ID for member, added in linkage.

Constructing a membership history

The organisation of the database makes re-creating a branch report trivial, but construction of personal histories for individual members of the union involves following pointers through the database. The following program was written to do this; it is written in c, but makes extensive use of embedded SQL statements to extract the data; the particular form of embedded SQL is that used by INGRES, but translation to other dialects should be straightforward.

Program Listing:

```
/*****************************************************************
****/
/* lifeline: This program takes as its only command line parameter */
```

```
/* the ID of a member. It begins by finding and printing */
/* the first membership record for the member, to which */
/* it appends all linked benefit payments from the */
/* relevant report. It then uses values of lsbid and */
/* nobid to find the next linked membership record and */
/* similarly outputs this and its dependents. This */
/* process continues until no further linked records */
/* can be found. */
/* */
/* For now, the program ignores travelling benefit. */
/* */
/* (c) H.R.Southall. May 1995. */
/*****************************************************************
****/

#include <string.h>
#include <stdio.h>
#include <ctype.h>

EXEC SQL INCLUDE SQLCA;
EXEC SQL INCLUDE SQLDA;

#define SUCCESS 1
#define FAILURE 0
#define BUFLEN 1024
#define STRLEN 60

EXEC SQL BEGIN DECLARE SECTION;
/* Parameters needed for processing 'members' table: */
struct mem_ {
int report;
char br_name[STRLEN];
int number;
char cname[STRLEN];
char sname[STRLEN];
char suffix[STRLEN];
char code[2];
char place[STRLEN];
char comment[512];
int record_id;
int lsb_id;
int lob_id;
int nsb_id;
```

```
int nob_id;
} mem;
short mem_ind[14];

EXEC SQL END DECLARE SECTION;

EXEC SQL BEGIN DECLARE SECTION;
/* Parameters needed for processing 'payments' table: */
struct pay_ {
int report;
char br_name[STRLEN];
char type[2];
int year;
int month;
int day;
char cname[STRLEN];
char sname[STRLEN];
char suffix[STRLEN];
int weeks;
int days;
int pounds;
int shillings;
float pence;
char comment[512];
} pay;
short pay_ind[15];

EXEC SQL END DECLARE SECTION;

EXEC SQL BEGIN DECLARE SECTION;
/* Parameters needed for processing 'funerals' table: */
struct fun_ {
int report;
char br_name[STRLEN];
int year;
int month;
int day;
char f_status[2];
char wname[STRLEN];
char cname[STRLEN];
char sname[STRLEN];
char suffix[STRLEN];
int age;
```

```
char cause[STRLEN];
char comment[512];
int pounds;
int shillings;
float pence;
} fun;
short fun_ind[16];

EXEC SQL END DECLARE SECTION;

EXEC SQL BEGIN DECLARE SECTION;
/* Parameters needed for processing 'reg_book' table: */
struct rbk_ {
int book;
char br_name[STRLEN];
char cname[STRLEN];
char sname[STRLEN];
char suffix[STRLEN];
int age;
char trade[STRLEN];
char m_status[2];
int year;
int month;
int day;
char comment[512];
} rbk;
short rbk_ind[12];

EXEC SQL END DECLARE SECTION;

EXEC SQL BEGIN DECLARE SECTION;
/* Parameters needed for processing 'ase_reg' table: */
struct arb_ {
char br_name[STRLEN];
char cname[STRLEN];
char sname[STRLEN];
char suffix[STRLEN];
int age_y;
int age_m;
char u_name[STRLEN];
char trade[STRLEN];
char m_status[2];
int year;
```

```
int month;
int day;
char comment[512];
} arb;
short arb_ind[13];

EXEC SQL END DECLARE SECTION;

EXEC SQL BEGIN DECLARE SECTION;
/* Parameters needed for processing 'd_cert' table: */
struct dcert_ {
char cname[STRLEN];
char sname[STRLEN];
char suffix[STRLEN];
int day;
int month;
int year;
char reg_dist[STRLEN];
int age;
char occup[STRLEN];
char place[STRLEN];
char cause[140];
} dcert;
short dcert_ind[11];

EXEC SQL END DECLARE SECTION;

EXEC SQL BEGIN DECLARE SECTION;
int member; /* ID of member to be followed */
int target; /* ID of next Mem_Rec */
char sel_buf[BUFLEN]; /* Prepared SELECT */
int err; /* Error status */
int test_val; /* Test only */

EXEC SQL END DECLARE SECTION;

int open_flag, first_flag, char_buff;

main(argc, argv)
int argc;
char *argv[];
{
int i, len;
```

```
/* Check that membership ID has been supplied */
if (argc != 2) {
printf("\n%s%s%s\n\n",
"Usage: ", argv[0], " ID of member");
exit(1);
}

/* Place membership ID into parameter */
member = atoi(argv[1]);

printf("\nTracing lifeline for member %d:\n\n", member);

open_flag = first_flag = 0;

Init_Db();

/* Set initial value of target to membership ID */
target = member;

/* Start of main loop; continues to execute until no new */
/* value of target can be found. */
while(target != 0)
{
/* Find and print out membership record for target */
GetMem(target);
/* Find and output linked payments data */
GetPay(target);
/* Find and output linked funerals data */
GetFun(target);
if (mem_ind[12] == 0) target = mem.nsb_id;
else if (mem_ind[13] == 0) target = mem.nob_id;
else target = 0;
}

End_Db();
exit(0);
}

Init_Db()
{
if (open_flag == 0) {
EXEC SQL WHENEVER SQLERROR STOP;
```

```
EXEC SQL WHENEVER NOT FOUND CONTINUE;
EXEC SQL CONNECT sem;
open_flag=1;
}
}

End_Db()
{
if (open_flag != 0) {
EXEC SQL COMMIT;
EXEC SQL DISCONNECT;

}
}

/* Find and print out membership record for target */

GetMem(target)
{
EXEC SQL SELECT m.report, b.br_name, m.number,
m.cname, m.sname, m.suffix, m.code, m.place,
m.comment, m.record_id, m.lsb_id, m.lob_id,
m.nsb_id, m.nob_id
into :mem:mem_ind
from members m, branches b
where m.branch=b.branch and
m.record_id = :target;

if (mem_ind[5] == 0)
printf("%d: Member of %s (Name: %s %s %s)\n",
mem.report, mem.br_name,
mem.cname, mem.sname, mem.suffix);
else printf("%d: Member of %s (Name: %s %s)\n",
mem.report, mem.br_name,
mem.cname, mem.sname);

/* Output footnote information, if present: */
if (mem_ind[6] == 0) { GetFoot(); }
/* Output Reg.Book information, if possibly present: */
if (mem_ind[10] != 0) { GetRBook(target); }
/* Output ASE Reg.Book information, if possibly present: */
if (mem_ind[10] != 0) {
if(mem.report < 1852) {GetAseRB(target); }}
```

```
}

/* Format and output footnote information: */
GetFoot()
{
int f_type;

printf(" Footnote information: ");

f_type = mem.code[0];
switch (f_type) {
case 'A': if (mem_ind[7] == 0) {
printf("Went abroad to %s.\n", mem.place);
}
else {
printf("Went abroad.\n");
}
break;
case 'B': if (mem_ind[7] == 0) {
printf("Joined %s branch.\n", mem.place);
}
else {
printf("Joined another branch.\n");
}
break;
case 'C': if (mem_ind[7] == 0) {
printf("Drew clearance to %s.\n", mem.place);
}
else {
printf("Drew clearance.\n");
}
break;
case 'D': printf("Died.\n");
break;
case 'E': printf("Was excluded.\n");
break;
case 'L': printf("Left.\n");
break;
case 'M': if (mem_ind[7] == 0) {
printf("Member of %s branch.\n", mem.place);
}
else {
printf("Member of another branch.\n");
```

```
}
break;
case 'P': printf("Proposition.\n");
break;
case 'S': printf("Superannuated.\n");
break;
case 'T': if (mem_ind[7] == 0) {
printf("Drew trav. cert. and moved"
" to %s.\n", mem.place);
}
else {
printf("Drew travelling certificate.\n");
}
break;
case 'X': printf("Miscellaneous.\n");
break;
default:
printf("Code = %s", mem.code);
if (mem_ind[7] == 0) printf(" (Place: %s)", mem.place);
printf(".\n");
break;
}
}

GetPay(target)
{
int p_type;

EXEC SQL DECLARE c_1 cursor for
select p.report, b.br_name, p.type,
p.year, p.month, p.day,
p.cname, p.sname, p.suffix,
p.weeks, p.days,
p.pounds, p.shillings, p.pence, p.comment
from payments p, branches b
where b.branch = p.branch and
p.mem_rec = :target
order by p.year, p.month, p.day;

EXEC SQL WHENEVER SQLERROR STOP;
EXEC SQL WHENEVER NOT FOUND GOTO closec_1;
EXEC SQL OPEN c_1;
```

```
/* Check value of sqlcode to see if there are still rows to process */
while (sqlca.sqlcode == 0)
{
EXEC SQL FETCH c_1 INTO :pay:pay_ind;

p_type = pay.type[0];
switch (p_type) {
case 'M': printf(" Misc. payment at %s of Pnds%d/%d/%.1f"
" on %d/%d/%d:\n", pay.br_name,
pay.pounds, pay.shillings, pay.pence,
pay.day, pay.month, pay.year);
printf(" %s\n", pay.comment);
break;
case 'P': printf(" Pension: %s: %d/%d/%d: %d weeks %d days\n",
pay.br_name, pay.day, pay.month, pay.year,
pay.weeks, pay.days);
break;
case 'S': printf(" Sickness: %s: %d/%d/%d: %d weeks %d days\n",
pay.br_name, pay.day, pay.month, pay.year,
pay.weeks, pay.days);
break;
case 'U': printf(" Unempl: %s: %d/%d/%d: %d weeks %d days\n",
pay.br_name, pay.day, pay.month, pay.year,
pay.weeks, pay.days);
break;
default: printf(" Type %s: %s: %d/%d/%d: %d weeks %d days\n",
pay.type, pay.br_name, pay.day, pay.month, pay.year,
pay.weeks, pay.days);
break;
}
}

closec_1: EXEC SQL CLOSE c_1;
}

GetFun(target)
{
int f_type;

EXEC SQL DECLARE c_2 cursor for
select f.report, b.br_name,
f.year, f.month, f.day,
f.f_status, f.wname,
```

```
f.cname, f.sname, f.suffix,
f.age, f.cause, f.comment,
f.pounds, f.shillings, f.pence
from funerals f, branches b
where b.branch = f.branch and
f.mem_rec = :target
order by f.year, f.month, f.day;

EXEC SQL WHENEVER SQLERROR STOP;
EXEC SQL WHENEVER NOT FOUND GOTO closec_2;
EXEC SQL OPEN c_2;

/* Check value of sqlcode to see if there are still rows to process */
while (sqlca.sqlcode == 0)
{
EXEC SQL FETCH c_2 INTO :fun:fun_ind;

f_type = fun.f_status[0];
switch (f_type) {
case 'D': if (fun_ind[6] == 0) {
printf(" Funeral benefit for his widow, %s,"
" was paid at %s on %d/%d/%d:\n",
fun.wname, fun.br_name,
fun.day, fun.month, fun.year);
}
else {
printf(" His widow's funeral benefit was"
" paid at %s on %d/%d/%d:\n",
fun.br_name, fun.day, fun.month, fun.year);
}
break;
case 'F': if (fun_ind[6] == 0) {
printf(" Funeral benefit for his wife, %s,"
" was paid at %s on %d/%d/%d:\n",
fun.wname, fun.br_name,
fun.day, fun.month, fun.year);
}
else {
printf(" His wife's funeral benefit was"
" paid at %s on %d/%d/%d:\n",
fun.br_name, fun.day, fun.month, fun.year);
}
break;
```

```
case 'M': printf(" His funeral benefit was paid at %s"
" on %d/%d/%d:\n",
fun.br_name, fun.day, fun.month, fun.year);
break;
default: printf(" Unknown funeral type: %s.\n",
fun.f_status);
break;
}

if (fun_ind[10] == 0) printf(" Age: %d.\n", fun.age);
if (fun_ind[11] == 0) printf(" Cause: %s.\n", fun.cause);
if (fun_ind[12] == 0) printf(" Comment: %s\n",
fun.comment);

/* Get death certificate information */
GetDCert(target);
}

closec_2: EXEC SQL CLOSE c_2;
}

GetRBook(target)
{
int r_type;

EXEC SQL DECLARE c_3 cursor for
select r.book, r.br_name, r.cname, r.sname, r.suffix,
r.age, r.trade, r.m_status,
r.year, r.month, r.day, r.comment
from reg_book r
where r.mem_rec = :target
order by r.year, r.month, r.day;

EXEC SQL WHENEVER SQLERROR STOP;
EXEC SQL WHENEVER NOT FOUND GOTO closec_3;
EXEC SQL OPEN c_3;

/* Check value of sqlcode to see if rows remain to process */
while (sqlca.sqlcode == 0)
{
EXEC SQL FETCH c_3 INTO :rbk:rbk_ind;

printf(" Registration Book information:\n");
```

```
if(rbk_ind[8] == 0) printf(" Date of admission: "
"%d/%d/%d.\n",
rbk.day, rbk.month, rbk.year);
if(rbk_ind[1] == 0) printf(" Branch of admission: %s.\n",
rbk.br_name);
if(rbk_ind[6] == 0) printf(" Trade: %s.\n",
rbk.trade);
if(rbk_ind[5] == 0) printf(" Age: %d.\n",
rbk.age);
if(rbk_ind[7] == 0) {
r_type = rbk.m_status[0];
switch (r_type) {
case 'M': printf(" Marital status: Married.\n");
break;
case 'S': printf(" Marital status: Single.\n");
break;
case 'W': printf(" Marital status: Widowed.\n");
break;
default: printf(" [Invalid marital status code.]\n");
break;
}
}
}

closec_3: EXEC SQL CLOSE c_3;
}

GetAseRB(target)
{
int r_type;

EXEC SQL DECLARE c_4 cursor for
select r.br_name, r.cname, r.sname, r.suffix,
r.age_y, r.age_m, r.u_name, r.trade, r.m_status,
r.year, r.month, r.day, r.comment
from ase_reg r
where r.mem_rec = :target
order by r.year, r.month, r.day;

EXEC SQL WHENEVER SQLERROR STOP;
EXEC SQL WHENEVER NOT FOUND GOTO closec_4;
EXEC SQL OPEN c_4;
```

```
/* Check value of sqlcode to see if rows remain to process */
while (sqlca.sqlcode == 0)
{
EXEC SQL FETCH c_4 INTO :arb:arb_ind;

printf(" ASE Registration Book information:\n");
if(arb_ind[11] == 0) printf(" Date of admission: %d/%d/%d.\n",
arb.day, arb.month, arb.year);
if(arb_ind[6] == 0) printf(" Union of admission: %s.\n",
arb.u_name);
if(arb_ind[0] == 0) printf(" Branch of admission: %s.\n",
arb.br_name);
if(arb_ind[7] == 0) printf(" Trade: %s.\n",
arb.trade);
if(arb_ind[4] == 0) {
printf(" Age: %d Years", arb.age_y);
if(arb_ind[5] == 0) {
if(arb.age_m != 0) printf(" %d Months", arb.age_m);
}
printf(".\n");
}
if(arb_ind[8] == 0) {
r_type = arb.m_status[0];
switch (r_type) {
case 'M': printf(" Marital status: Married.\n");
break;
case 'S': printf(" Marital status: Single.\n");
break;
case 'W': printf(" Marital status: Widowed.\n");
break;
default: printf(" [Invalid marital status code.]\n");
break;
}
}
if(arb_ind[12] == 0) printf(" Other information:\n %s\n",
arb.comment);
}

closec_4: EXEC SQL CLOSE c_4;
}

GetDCert(target)
{
```

```
EXEC SQL DECLARE c_5 cursor for
select d.cname, d.sname, d.suffix,
d.day, d.month, d.year, d.reg_dist,
d.age, d.occup, d.place, d.cause
from d_cert d
where d.mem_rec = :target
order by d.year, d.month, d.day;

EXEC SQL WHENEVER SQLERROR STOP;
EXEC SQL WHENEVER NOT FOUND GOTO closec_5;
EXEC SQL OPEN c_5;

/* Check value of sqlcode to see if rows remain to process */
while (sqlca.sqlcode == 0)
{
EXEC SQL FETCH c_5 INTO :dcert:dcert_ind;

printf(" Death Certificate information:\n");
if(dcert_ind[5] == 0) printf(" Date of death: %d/%d/%d.\n",
dcert.day, dcert.month, dcert.year);
if(dcert_ind[6] == 0) printf(" Registration District: %s.\n",
dcert.reg_dist);
if(dcert_ind[9] == 0) printf(" Place: %s.\n",
dcert.place);
if(dcert_ind[7] == 0) printf(" Age: %d.\n",
dcert.age);
if(dcert_ind[8] == 0) printf(" Occupation: %s.\n",
dcert.occup);
if(dcert_ind[10] == 0) printf(" Cause: %s.\n",
dcert.cause);
}

closec_5: EXEC SQL CLOSE c_5;
}
```

Specimen Output:

```
bash$ lifeline 51847025
Tracing lifeline for member 51847025:

1847: Member of HAWARDEN (Name: JOHN PRICE COEDTALON)
```

Registration Book information:
Date of admission: 20/6/1848.
Branch of admission: HAWARDEN.
Trade: FITTER.
Age: 20.
Marital status: Married.
1848: Member of HAWARDEN (Name: JOHN PRICE)
1849: Member of HAWARDEN (Name: JOHN PRICE)
1850: Member of HAWARDEN (Name: JOHN PRICE)
Sickness: HAWARDEN: 10/9/1850: 1 weeks 0 days
1852: Member of HAWARDEN (Name: JOHN PRICE)
1853: Member of HAWARDEN (Name: JOHN PRICE)
1854: Member of HAWARDEN (Name: JOHN PRICE)
Sickness: HAWARDEN: 6/10/1855: 1 weeks 0 days
Sickness: HAWARDEN: 3/11/1855: 1 weeks 2 days
1855: Member of HAWARDEN (Name: JOHN PRICE)
Unempl: HAWARDEN: 22/3/1856: 1 weeks 0 days
Unempl: HAWARDEN: 6/9/1856: 1 weeks 4 days
1856: Member of HAWARDEN (Name: JOHN PRICE)
Sickness: HAWARDEN: 18/4/1857: 2 weeks 1 days
Unempl: HAWARDEN: 21/7/1857: 2 weeks 3 days
Sickness: HAWARDEN: 28/11/1857: 4 weeks 0 days
1857: Member of HAWARDEN (Name: JOHN PRICE)
1858: Member of HAWARDEN (Name: JOHN PRICE)
Sickness: HAWARDEN: 30/10/1858: 13 weeks 0 days
1859: Member of HAWARDEN (Name: JOHN PRICE)
1860: Member of HAWARDEN (Name: JOHN PRICE)
1861: Member of HAWARDEN (Name: JOHN PRICE)
Sickness: HAWARDEN: 15/2/1862: 0 weeks 3 days
1862: Member of HAWARDEN (Name: JOHN PRICE)
Sickness: HAWARDEN: 27/9/1862: 0 weeks 4 days
Unempl: HAWARDEN: 11/4/1863: 1 weeks 0 days
1863: Member of HAWARDEN (Name: JOHN PRICE)
1864: Member of HAWARDEN (Name: JOHN PRICE)
Sickness: HAWARDEN: 30/7/1864: 0 weeks 5 days
1865: Member of HAWARDEN (Name: JOHN PRICE)
Sickness: HAWARDEN: 3/6/1865: 3 weeks 0 days
1866: Member of HAWARDEN (Name: JOHN PRICE)
1867: Member of HAWARDEN (Name: JOHN PRICE)
Sickness: HAWARDEN: 27/7/1867: 7 weeks 0 days
1868: Member of HAWARDEN (Name: JOHN PRICE)
1869: Member of HAWARDEN (Name: JOHN PRICE)
1870: Member of HAWARDEN (Name: JOHN PRICE)

1871: Member of HAWARDEN (Name: JOHN PRICE)
Sickness: HAWARDEN: 16/9/1871: 2 weeks 0 days
Sickness: HAWARDEN: 30/3/1872: 0 weeks 4 days
1872: Member of HAWARDEN (Name: JOHN PRICE)
Sickness: HAWARDEN: 8/2/1873: 4 weeks 3 days
Sickness: HAWARDEN: 18/3/1873: 1 weeks 1 days
1873: Member of HAWARDEN (Name: JOHN PRICE)
1874: Member of HAWARDEN (Name: JOHN PRICE)
1875: Member of HAWARDEN (Name: JOHN PRICE)
1876: Member of HAWARDEN (Name: JOHN PRICE)
bash$

Creating a dataset for statistical analysis

Individual data tables are of limited value for analysis, while conventional statistical software cannot trace individual histories through various tables. The following sequences of SQL commands construct a new table with one record per man-year, with variables giving the man's age and the town he was in, the number of days during the year for which he was a member, and the number of days he was on each of the different benefits. A wide range of conventional analyses can be performed on this new table.

---

(1) Create a temporary look-up table of members' ages

These SQL commands consolidate information about members' ages into a single look-up table; four different tables containing ages are queried, in descending order of reliability. The table created also records the source of each age:

```
create table temp_age (
member integer,
b_year integer,
source vchar(1));

insert into temp_age (member, b_year, source)
select member, b_year=year-age, source='D'
from d_cert
where age is not null and member is not null
and date is not null;
```

```
insert into temp_age (member, b_year, source)
select member, b_year=year-age, source='F'
from funerals
where age is not null and member is not null
and date is not null
and member not in (select member from temp_age);

insert into temp_age (member, b_year, source)
select member, b_year=year-age, source='R'
from reg_book
where age is not null and member is not null
and date is not null
and age < 100
and member not in (select member from temp_age);

insert into temp_age (member, b_year, source)
select member, b_year=year-age_y, source='A'
from ase_reg
where age_y is not null and member is not null
and date is not null
and member not in (select member from temp_age);
```

---

(2) Create the basic dataset for analysis.

These commands create a new table containing one row for each SEM member of known age who was continuously a member during a particular year or died during it, with many values initially set to zero or left empty, and then classify the data into age groups:

```
create table ts_data as
select fin_year=m.report,
town=b.town,
member=m.member,
record_id=m.record_id,
event=m.code,
age=r.start_yr-a.b_year,
age_grp='Unset',
age_source=a.source,
year_len = int4(interval('days', r.end_date - r.start_date)),
u_days=0,
```

```
s_days=0,
p_days=0
from members m, branches b, temp_age a, reports r
where m.lsb_id is not null and
m.nsb_id is not null and
m.branch=b.branch and
m.member=a.member and
a.b_year is not null and
r.report=m.report;

/* Adds men who died to the table: */

insert into ts_data (fin_year, town, member, record_id, event, age,
age_grp, age_source, year_len, u_days, s_days, p_days)
select fin_year=m.report,
town=b.town,
member=m.member,
record_id=m.record_id,
event=m.code,
age=r.start_yr-a.b_year,
age_grp='Unset',
age_source=a.source,
year_len = int4(interval('days', f.date - r.start_date)),
u_days=0,
s_days=0,
p_days=0
from members m, funerals f, branches b, temp_age2 a, reports r
where m.lsb_id is not null and
m.nsb_id is null and
m.branch=b.branch and
m.record_id=f.mem_rec and
f.f_status='M' and
f.date is not null and
m.member=a.member and
a.b_year is not null and
r.report=m.report;

/* Assign each record to the relevant age group: */
update ts_data set age_grp='<30' where age < 30;
update ts_data set age_grp='30-39' where age >= 30 and age < 40;
update ts_data set age_grp='40-49' where age >= 40 and age < 50;
update ts_data set age_grp='50-59' where age >= 50 and age < 60;
update ts_data set age_grp='60+' where age >= 60;
```

(3) Add a count of the number of days unemployed

These commands first create a temporary summary table from the 'payments' table, and then uses this to insert the numbers of days a man was unemployed into the analytical data set. Almost identical sets of commands similarly insert the number of days sick and on superannuation:

```
create table temp_u_summary as
select mem_rec, u_days=(sum(weeks)*6)+sum(days)
from payments
where type='U'
group by mem_rec;

create unique index usum_idx on temp_u_summary (mem_rec)
with structure = btree,
compression = (nokey),
key = (mem_rec),
nonleaffill = 80,
leaffill = 70,
fillfactor = 80,
location = (ii_database)

update ts_data d from temp_u_summary u
set u_days=u.u_days
where d.record_id=u.mem_rec and
u.u_days is not null;
```

(4) Specimen Output

The output from the above sequence of commands is very extensive; this is just a small sample of records. The individual men covered are identified by the value in the 'Member' column, so this includes five men from two towns, while the first column identifies the year involved. The 'Event' column shows information taken from the footnotes to the membership list; among these examples, one of the Blackburn members was listed as having gone abroad, but he must have been back by the following year in order to have been included. 'Age' and 'Age Grp' are self-explanatory, while 'Age Src.' indicates the basis for our knowledge of the member's age. 'Year len.' is the number of days 'at risk' during the financial year, usually simply the number of days recorded in the

'reports' table, while the final three columns give the number of days during the year for which the member received each of the three types of benefit:

Fin. Record Age Age Age Year No. of days on:
Year Town Member ID Event Group Src. Len. Unem Sick Super

1873 BLACKBURN 71869036 71873026 . 35 30-39 F 364 0 0 0
1874 BLACKBURN 71869036 71874024 A 36 30-39 F 214 0 0 0
1875 BLACKBURN 71869036 71875023 . 36 30-39 F 365 0 0 0
1870 BLACKBURN 71869112 71870092 . 41 40-49 F 364 0 0 0
1871 BLACKBURN 71869112 71871083 . 42 40-49 F 365 36 0 0
1872 BLACKBURN 71869112 71872081 . 43 40-49 F 364 0 0 0
1872 BLACKBURN 71871100 71872094 . 35 30-39 F 364 0 24 0
1836 BIRMINGHAM 81835001 81836001 . 50 50-59 D 365 0 0 0
1837 BIRMINGHAM 81835001 81837001 . 51 50-59 D 365 0 0 0
1838 BIRMINGHAM 81835001 81838001 . 52 50-59 D 365 0 6 0
1836 BIRMINGHAM 81835020 81836019 . 32 30-39 F 365 0 6 0
1837 BIRMINGHAM 81835020 81837015 . 33 30-39 F 365 0 0 0
1838 BIRMINGHAM 81835020 81838014 . 34 30-39 F 365 0 0 0
1839 BIRMINGHAM 81835020 81839006 . 35 30-39 F 366 0 0 0
1840 BIRMINGHAM 81835020 81840005 . 36 30-39 F 365 0 0 0
1840 BIRMINGHAM 81835020 81840005 . 30 30-39 F 365 0 0 0
