D. P. Gaver
S. S. Lavenberg
T. G. Price, Jr.

Exploratory Analysis of Access Path Length Data for a Data Base Management System

Abstract: An exploratory approach is taken to analyze a vast quantity of data recorded during the running of the data base management system IMS (Information Management System). The collection of data analyzed is a sequence of access path lengths for a day-long period. The number of segments accessed by IMS when searching a data base in order to retrieve a specified segment for a user is called an access path length. Part of the motivation for the analysis is to suggest reasonable stochastic models for the access path length sequence that can be conveniently utilized as input models for a stimulation model of an IMS installation. The exploratory approach taken to the data involves the use of graphical displays and simple numerical summaries to reveal characteristics of, and patterns in, the data. Some simple ways are presented in which the structure of the data revealed by the analysis can be incorporated into an input model for a system simulation.

Introduction

In this paper an *exploratory* or descriptive approach is taken to study a vast quantity of data recorded during a day-long period of operation of a computer system that is running a data base management system. The approach is to apply appropriate graphical methods to the time sequence of raw data, and to use simple numerical summaries to reveal characteristics of, and patterns in, the data. The data base management system is IMS (Information Management System), a processing program that facilitates the accessing of large data bases shared in common by several applications [1]. A description of IMS which provides the necessary background for this paper is given in [2]. The data analyzed are the sequences of access path lengths for a day-long period. The *access path length* is the number of segments accessed when searching a data base in response to a data base call issued by an application program to IMS (cf. [2]). The nature of the access path lengths has implications for the performance of an IMS installation (i.e., a computer system running IMS). It is reasonable to expect that if access path lengths are shortened, e.g., by modifying the set of pointers maintained under the hierarchical direct storage organization, then improved system performance will result. However, the precise way in which access path lengths affect system performance variables such as the response time for a transaction is not know.

The sequence of access path lengths (measured as described in the second section) is one of the input (workload) sequences for a simulation model of an IMS installation that is currently being developed. The model extends a previously developed analytic queueing model [2] of the data base management portion of IMS to incorporate representations of additional system activities (e.g., IMS data communication activities and operating system overhead). Part of the motivation for the exploratory data analysis of this paper is to suggest reasonable stochastic models for the access path length sequence that can be conveniently incorporated into the simulation model. In the analytic queueing model [2] it was assumed that successive access path lengths form a sequence of independent and identically distributed random variables that are geometrically distributed. This assumption, which facilitated the mathematical analysis of the model, is shown to be inconsistent with the available data.

Finally, a few disclaimers are in order. In general, no attempt has been made to explain the observed patterns in the data on physical grounds, although such explanations are certainly desirable. Neither is it claimed that the patterns noted will persist indefinitely at one installation or widely over many. Despite the fact that many path lengths were recorded, the duration of recording was one day, representing but one installation. Nevertheless, the data behavior is of interest because of the detailed structure revealed; no data base system model should ignore such structure in possible inputs. At the end of the paper some simple, tentative ways in which the observed structure can be incorporated into an input model for a system simulation are discussed.

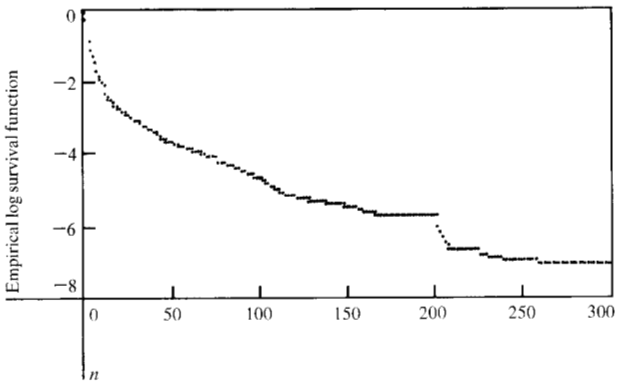


Figure 1 Empirical log survivor function for L.

Table 1 Numerical histogram for L.

Range	*Count*	*Range*	*Count*
0	9848	26-50	6488
1	55785	51-75	2020
2	62026	76-100	1256
3	22766	101-125	846
4	12318	126-150	218
5	5625	151-175	233
6	13893	176-200	51
7	4610	201-225	476
8	3869	226-250	38
9	2728	251-300	13
10	2185	301-1000	12
11	5176	1001-2000	35
12	2044	2001-3000	78
13-25	8478	3001-4005	71

Table 2 Summary statistics for L.

Sequence Size	223186
Mean	9.05
Variance	7.14×10^3
Coef. Var.	9.34
Minimum	0
Maximum	4005
Median	2
90th Percentile	12

Data measurement

• *Description of the source system*

Initially, data were collected during one day's normal operation of an on-line, manufacturing IMS installation. This installation will be referred to as the source system. The data bases contained information about production scheduling, production history, quality control, planning, etc. The installation was running IMS/360 on a 370/155. There were five data bases on-line containing 113.65, 16.91, 9.74, 34.7, and 0.77 megabytes, respectively. The structure and number of segment instances for each data base are given in the Appendix. The hierarchical direct storage organization was used for each data base and indices were maintained for root segment instances. The access method for this storage organization is called HIDAM [1]. Forward twin pointers, forward parent-to-first-child pointers, and backward child-to-parent pointers were maintained for each data base. The average number of terminals active was 34, and 48 types of transactions were processed during the day. Three message processing regions were active simultaneously.

• *Measurements on the source system*

To minimize data collection overhead on the source system, only a limited amount of data were collected. The collected information included, for each data base call, the name of the transaction that made the call, the data base referenced, the call type (there are nine types of data base calls in IMS, *get unique* and *get next* being two of the call types), the target segment type and instance, and the time that the call was completed. This information was recorded in the IMS log. The sequence of path segments accessed in order to reach the target segment was not recorded.

• *Measurements on the experimental system*

For the work described in this paper, access path length data were needed. These data were reconstructed on a system dedicated to measurement and experimental work. All of the data base calls for a particular instance of a transaction were grouped together into a transaction sequence. (In IMS, all data base calls arising from an instance of a transaction are processed consecutively in a message processing region with no intervening calls from other transactions processed in that region.) The transaction sequences were then ordered according to the completion times of the first call of each transaction. This was intended to approximate the order in which the transactions arrived and would have been processed on the source system if there were only one message processing region. The reordered sequence of calls was executed in a single region on the experimental system using reconstructions of the source data bases. The resulting single-region sequence of access path lengths was measured and is the one studied in this paper.

The single-region sequence of access path lengths is one of the input sequences for a simulation model, discussed in the Introduction, of an IMS installation with

Table 3 Sample statistics for sections of L (section size = 10000).

Section	*Mean*	*Coef. var.*	*Min.*	*Max.*	*Percentiles* 50th	*Percentiles* 90th	*Count of apls > 1000*
1	8.41	2.78	0	225	2	13	0
2	7.03	2.51	0	264	2	15	0
3	5.85	2.37	0	238	2	11	0
4	19.5	8.83	0	3936	2	12	41
5	6.96	10.4	0	3508	2	8	5
6	7.07	2.73	0	259	2	11	0
7	14.7	11.3	0	3917	2	8	35
8	9.8	12.1	0	3951	2	11	14
9	10.7	10.2	0	3632	2	12	16
10	5.46	2.54	0	212	2	11	0
11	6.66	2.28	0	230	2	13	0
12	8.12	5.54	0	3127	2	12	3
13	11.7	11.9	0	4005	2	11	18
14	13.9	12.3	0	4004	2	8	31
15	7.12	5.18	0	3273	2	13	1
16	13.1	1.94	0	207	2	46	0
17	8.91	2.57	0	230	2	15	0
18	11.2	10.9	0	3959	2	11	20
19	7.34	2.42	0	207	2	14	0
20	5.08	2.49	0	207	2	10	0
21	6.34	2.61	0	206	2	11	0
22	6.00	2.55	0	225	2	11	0
Mean	9.11	5.83					
Std. dev.	3.65	4.12					

multiple regions. The scheduling of transactions for processing in multiple regions is represented explicitly in the model, and the model transforms the single-region access path length sequence into a multiple-region sequence. The single-region sequence is independent of the scheduling algorithm used and, hence, is more system independent than the multiple-region sequence. One of the model outputs will be sequence of response times for successive transactions. Thus, the model transforms the single-region access path length sequence into a multiple-region sequence of transaction response times.

Exploratory analyses

• *Analysis of access path length sequence*

The access path length (apl) sequence L for a day-long period is investigated in this subsection. The sequence L consists of 223186 apls ordered as described in the preceding section [3]. Table 1 contains a numerical histogram for L [4]. Summary statistics for the entire sequence are given in Table 2. Notice that the median and 90th percentile are small, and that the data are highly positively skewed. The large coefficient of variation is accounted for by the substantial number of apls in the ranges 201-225 and 1001-4005.

The shape of the empirical log survivor function $r(n)$ for L, where $r(n) \equiv \log[(\text{count of apls} \geq n)/\text{sequence size}]$, can help reveal whether the marginal distribution for the sequence is consistent with a specified, simple, functional form. For example, the theoretical log survivor function for a geometrically distributed random variable is linear in n. A plot of $r(n)$ versus n for $n \leq 300$ is shown in Fig. 1. Clearly, the plot is not consistent with the assumption that L is a realization of a stationary sequence of geometrically distributed random variables. Furthermore, the coefficient of variation of a geometrically distributed random variable with mean m is $[(1+m)/m]^{\frac{1}{2}}$ which is equal to 1.05 if $m = 9.05$; since the observed coefficient of variation for L is equal to 9.34, the simple geometric distribution is quite implausible. Also, the apparent preference for path lengths of 201-225 and the smaller, but noticeable, preference for path lengths between 2001 and 4005 are incompatible with a geometric distribution.

In order to investigate whether sample statistics for L vary over the day-long period, the first 220000 apls in L were divided into 22 non-overlapping sections of 10000 consecutive apls each. Sample statistics for the 22 sections are given in Table 3. The last two rows of the table contain, respectively, the mean and standard deviation of the numbers in the second and third columns of the table. Notice that the sample mean and coefficient of variation vary considerably from section to section while the sample median does not vary at all and the sample 90th percentile is relatively stable (except for Section 16). The sample coefficient of variation does not exceed

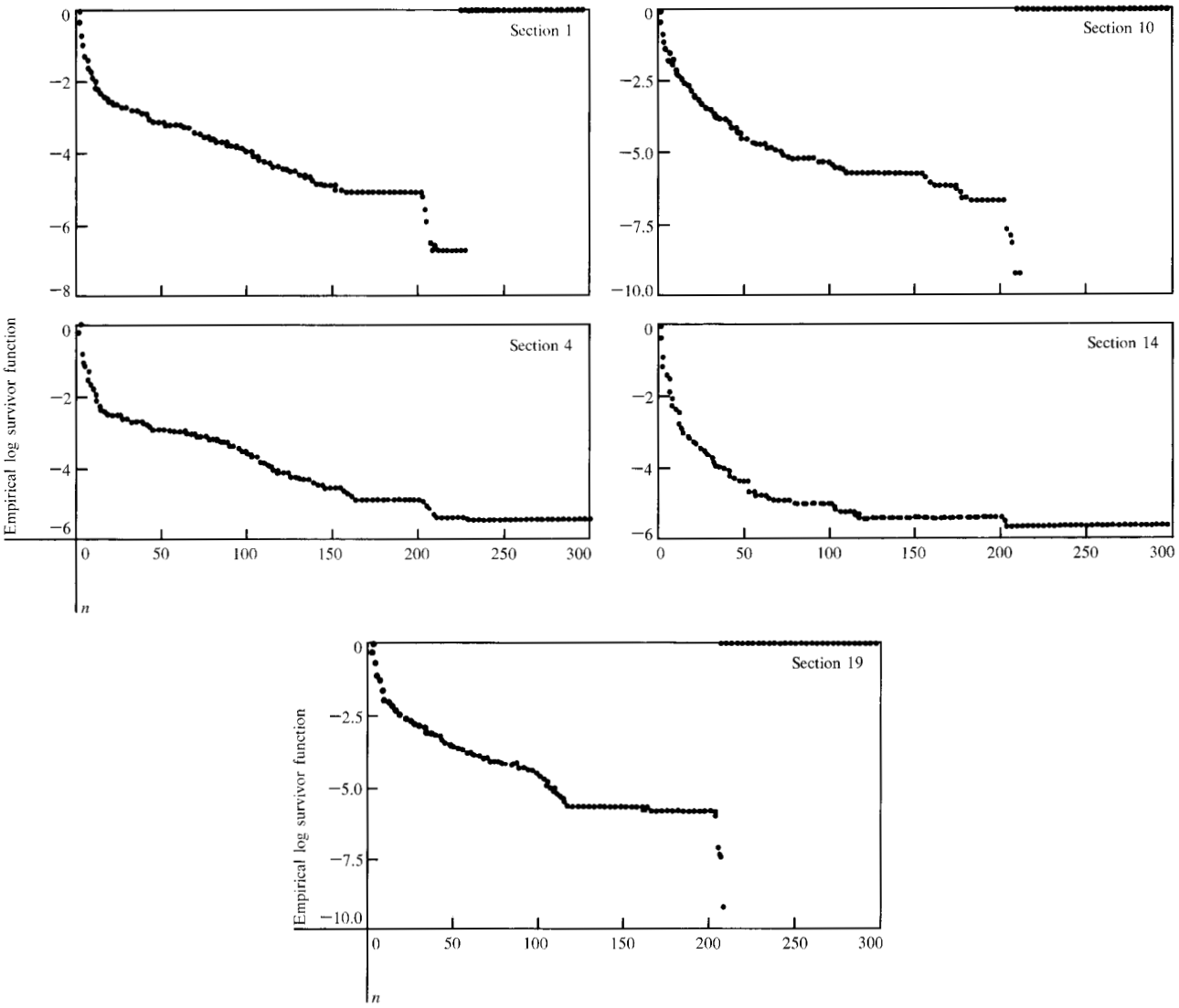


Figure 2 Empirical log survivor functions for sections of L (section size = 10000).

3 for those sections having no apls greater than 1000 and it exceeds 5 for all other sections. Plots of the empirical log survivor function for selected sections are given in Fig. 2, where minus infinity is plotted as zero. Notice that the shapes of the empirical log survivor functions for the different sections are somewhat similar for $n \leq 200$. However, the frequency of the occurrence of preferred values 201-225 and of values greater than 1000 varies considerably among sections.

The sequence L was also divided into 223 non-overlapping sections of 1000 consecutive apls each. The sample mean, coefficient of variation and serial correlation of lag 1 for each section are plotted versus the serial number of the section in Fig. 3. The plots reveal that while the sample statistics are highly variable from section to section, there is no apparent trend for these statistics. If the apls for a section were independent and identically distributed (i.i.d.) random variables, then the sample serial correlation of lag 1 for a section of size 1000 should (based on asymptotic theory) be approximately normally distributed with mean $1/1000 = 0.001$ and standard deviation $1/\sqrt{1000} = 0.032$ [5]. The probability of observing a correlation greater in magnitude than 0.10 would be less than 0.0025. The third plot in Fig. 3 indicates that apls are *not* i.i.d. for each section, since there are 24 serial correlations of lag 1 out of 223 that are greater in magnitude than 0.10. These large correlations are both positive and negative.

There are 741 apls greater than or equal to 200 in L. These "long" apls are plotted against their serial numbers of occurrence (index in the sequence) in Fig. 4. This plot reveals that the apls greater than 1000 occur in clusters. Clearly if L were a realization of a sequence of i.i.d. random variables, such clustering would not be anticipated. There are 184 apls greater than 1000, and these apls account for slightly more than 25 percent of the approximately 2×10^6 segment accesses for the day-long period being considered.

The gross characteristics of the access path length sequence as revealed in the above analysis are:

1. The empirical distribution function is highly positively skewed; certain long apls are preferred.
2. Sample statistics for the sequence vary greatly from section to section over the day.
3. There is no obvious trend or easily explained cyclic pattern in the sequence.
4. There is evidence of correlation between successive apls.
5. A small number (less than 0.1 percent of all apls) of very long apls (apls greater than 1000) occur in clusters in the sequence. (Later it will be seen that these very long apls are caused by calls to a single data base.)

• *Analysis of access path length sequences for different data bases*

It is plausible that the apl for a data base call depends on the particular data base referenced by the call, since the five data bases are quite different in structure and in size (see the Appendix). For the day-long period being considered, all but one of the data base calls were to one of four data bases, denoted as data bases 1, 2, 3, and 4. Four apl sequences L_1, L_2, L_3, and L_4 can be extracted from L, where L_i is the subsequence of L containing all apls resulting from calls to data base i.

The data base reference sequence D consists of 223 185 data base references (one for each data base call except for the call to the fifth data base) ordered in the same manner as the data base calls are ordered. D is a sequence on the integers 1, 2, 3, and 4. Given L_1, L_2, L_3, L_4, and D, the sequence L can be recreated in the obvious way (with the exception of one apl). In order to further explore the gross features of the access path length data, the apl sequence for each data base is examined next. The sequence D will be examined in the following subsection.

Summary statistics for L_1-L_4 are given in Table 4, and plots of the empirical log survivor functions are shown in Fig. 5. Notice that L_2 contains all apls greater than 211 and that the mean, coefficient of variation, and 90th percentile for L_2 are much larger than for the

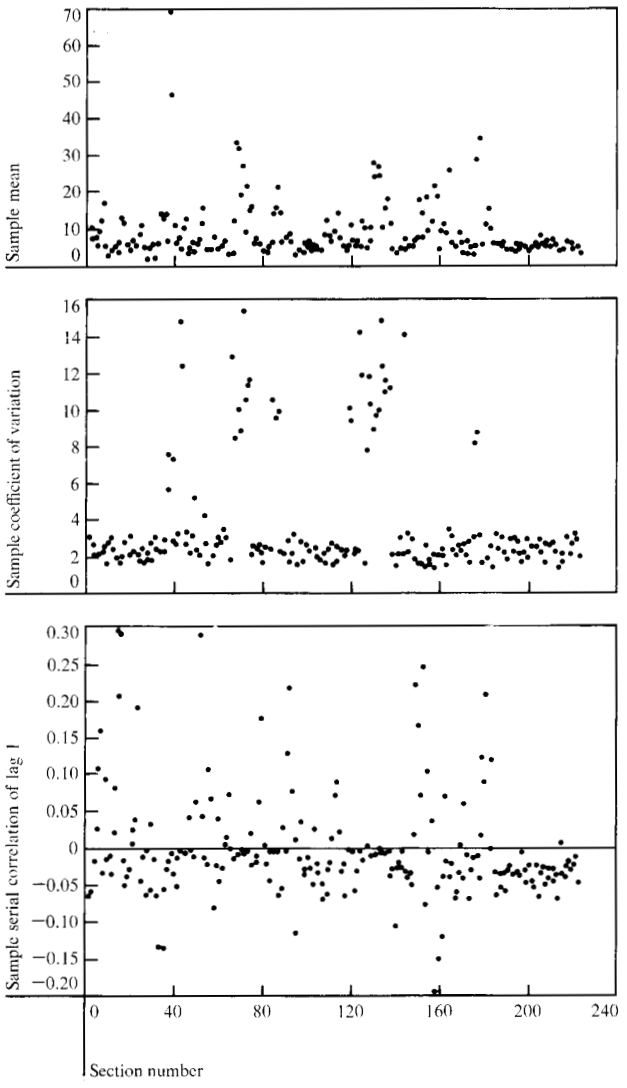
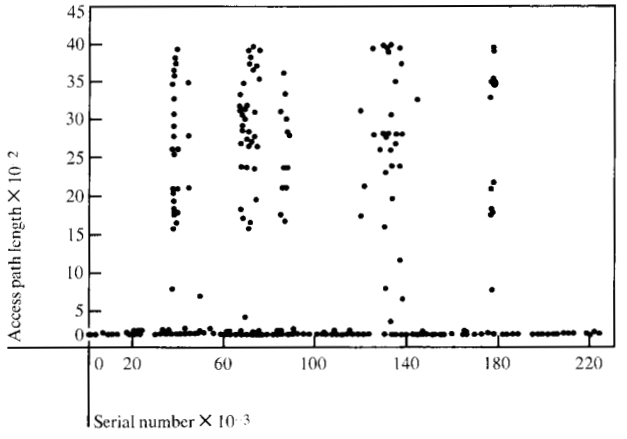


Figure 3 Sample statistics vs section number for L (section size = 1000).

Figure 4 Access path lengths ≥ 200 vs serial number for L.



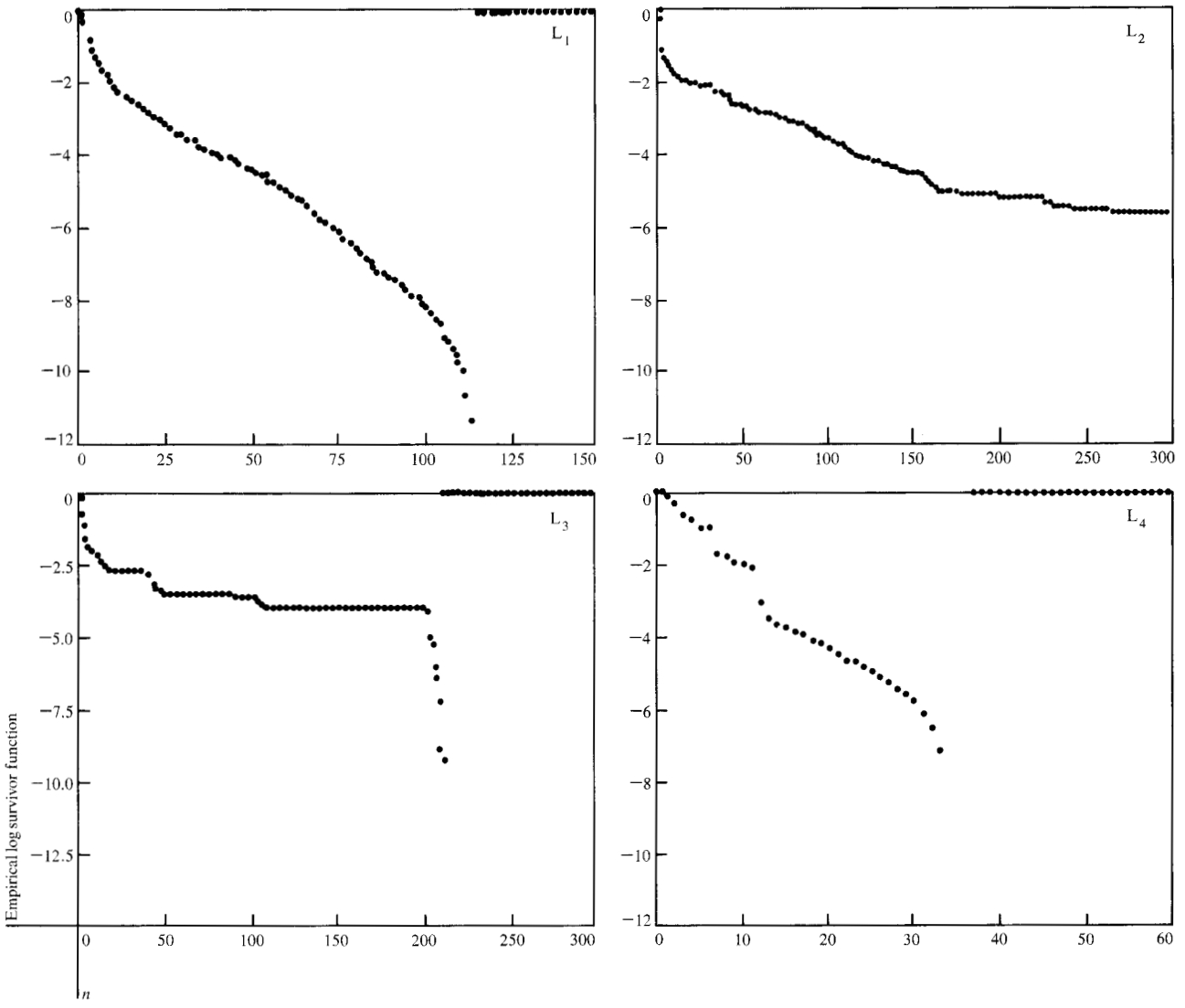


Figure 5 Empirical log survivor functions for L_1, L_2, L_3, L_4.

other three sequences. The empirical log survivor function for L_3 has a big jump at approximately 200, while the other three log survivor functions do not. (The sequence L_3 contains 430 apls between 201 and 211, while the other three sequences contain only six such apls.) Clearly, the empirical distribution functions for the four apl sequences differ greatly and are not geometric.

Each of the four sequences was divided into nonoverlapping sections of 1000 consecutive apls. For each sequence the sample mean, coefficient of variation, and serial correlation of lag 1 were computed for each section and plotted versus the serial number of the section. Only the plots of sample serial correlation are shown here, in Fig. 6. As was the case for the sequence L, the sample statistics are highly variable from section to section of each sequence. For each of the sequences L and L_1-L_4, the mean and standard deviation across sections of the sample statistics are given in Table 5, where the standard deviation appears below the mean. The standard deviation provides a measure of the variability from section to section. The variability from section to section of the sample statistics for L does not appear to be simply due to different data bases being referenced in different sections. There is evidence of trends in some of

Table 4 Summary statistics for L_1, L_2, L_3, L_4.

	L_1	L_2	L_3	L_4
Sequence size	99076	52413	25060	46636
Mean	5.22	20.5	8.67	4.48
Variance	83.8	2.97×10^4	865.0	19.2
Coef. var.	1.75	8.39	3.39	0.978
Minimum	0	0	0	0
Maximum	113	4005	211	36
Median	2	2	2	3
90th percentile	11	34	12	11

the sample statistics for L_1-L_4. For example, the sample serial correlation of lag 1 for L_1 decreases at the end of the sequence.

There are a substantial number of sample serial correlations of lag 1 greater than 0.1 in magnitude in Fig. 6, suggesting significant serial correlation for some sections of L_1-L_4. The correlations greater than 0.1 in magnitude are both positive and negative for L_1 and L_4, but are negative only for L_2 and L_3. In order to further investigate whether there is significant serial correlation in L_1, an informal test was performed. The empirical distribution function was obtained for each section of L_1 of size 1000. Then 1000 independent samples were drawn from each of the above 99 empirical distribution functions to create a sequence $\hat{L}_1$. $\hat{L}_1$ is a realization of a sequence of independent random variables, where the random variables in a given section of size 1000 are identically distributed with distribution functional equal to the empirical distribution function for the corresponding section of L_1. The sample serial correlations of lag 1 for each of the 99 sections of $\hat{L}_1$ are plotted in Fig. 7. All sample serial correlations in Fig. 7 are less than 0.1 in magnitude, indicating that the large sample serial correlations for L_1 in Fig. 6 are unlikely to have arisen from a realization of a sequence of independent random variables. This graphical comparison of sample statistics obtained from the simulation of a model with sample statistics obtained from the data provides an informal test of the adequacy of the model. It was expected that similar results would be obtained if this test were repeated for L_2, L_3, or L_4 and the test was not repeated.

In Fig. 8 the 287 apls greater than 211 in L_2 are plotted against their serial numbers of occurrence in L_2. The apls greater than 1000 occur in clusters in the first three-fourths of the sequence.

The gross characteristics of the access path length sequences for the four data bases are:

1. The empirical distribution functions differ greatly for L_1-L_4 and are highly positively skewed for L_1-L_3.

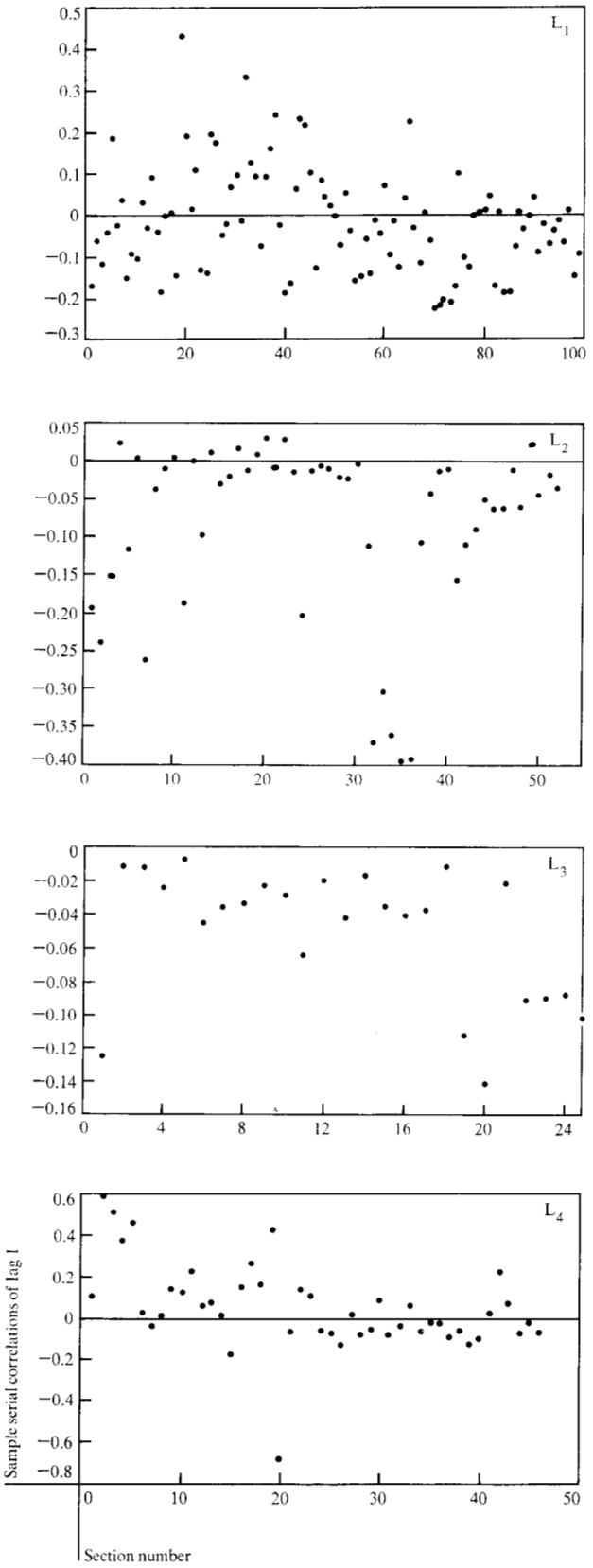


Figure 6 Sample serial correlations of lag 1 vs section number for L_1, L_2, L_3, L_4 (section size = 1000).

Table 5 Mean and standard deviation across sections of sample statistics for L and L_1-L_4 (section size = 1000).

	L	L_1	L_2	L_3	L_4	
Mean	9.06	5.23	20.07	8.66	4.49	Mean
	7.60	2.16	21.6	4.62	1.33	Std. dev.
Coef. var.	3.71	1.50	3.77	3.51	0.991	Mean
	3.20	0.336	2.64	0.686	0.342	Std. dev.
Ser. cor. 1	−0.003	−0.014	−0.083	−0.050	0.055	Mean
	0.059	0.124	0.116	0.039	0.208	Std. dev.

Table 6 One-step transition matrix for D

	1	*2*	*3*	*4*
1	0.747	0.240	0.012	3×10^{-5}
2	0.462	0.456	0.006	0.076
3	0.016	0.058	0.926	4×10^{-4}
4	0.010	0.070	0.007	0.914

Table 7 Sample statistics for run lengths for sections 1-3 of D (section size = 10000).

Section	*State*	*Sample size*	*Mean*	*Coef. var.*
1	1	1194	4.45	5.32
	2	1257	1.63	0.527
	3	105	17.7	1.80
	4	81	9.72	2.95
2	1	992	5.07	4.65
	2	1096	1.69	0.807
	3	87	20.3	1.72
	4	129	9.33	1.32
3	1	742	6.25	5.75
	2	814	2.39	4.27
	3	92	19.1	1.61
	4	96	15.9	1.35

2. Sample statistics for each sequence vary greatly from section to section over the day.
3. There are indications of trends in some of the sequences.
4. There is evidence of correlation between successive apls for each of the sequences, the significant correlations being either positive or negative for sections of L_1 and L_4, and negative only for sections of L_2 and L_3.
5. All apls greater than 1000 occur in clusters in the first three-fourths of L_2.

• *Analysis of data base reference sequence*

The data base reference (dbr) sequence D is a sequence on the integers 1-4. In Fig. 9, for each i, the number of occurrences of dbr i in the first n dbrs is plotted versus n, where the axes are labeled in units of 1000. These plots of the cumulative counts of references to each data base indicate that the sequence D is not stationary over the day. Data bases 1 and 3 are referenced more frequently in the first half of the sequence than in the last half, and data bases 2 and 4 are referenced more frequently in the last half of the sequence than in the first half. In addition, there are rapid short-term fluctuations in the frequency of reference to the four data bases. (The straight lines, plotted for comparison purposes, have slopes equal to the frequencies of reference to the four data bases for the entire day.)

The 4×4 matrix of one-step transition frequencies for D is given in Table 6 (the ijth entry in the matrix is the frequency of occurrence of j in the sequence, given j preceded by i). It is apparent that successive dbrs are dependent (e.g., a 1 is much more likely to follow a 1 than to follow a 3 or 4). Based on counts, the dominant one-step transitions in D are the 1-1, 1-2, 2-1, 2-2, 3-3, and 4-4 transitions. The counts of the dominant one-step transitions for non-overlapping sections of 10000 consecutive dbrs vary considerably from section to section. (The plots of these counts are not shown.)

A simple model for D is a first-order Markov chain with state space equal to the integers 1-4. The run length in each state of a first-order Markov chain has a geometric distribution starting at one. If m_i is the mean run length in state i, then the coefficient of variation of the run length in state i is $[(m_i - 1)/m_i]^{\frac{1}{2}}$, which is less than one. In order to informally test the adequacy of a first-order Markov chain model for D, sample statistics were computed for the run lengths of 1's, 2's, 3's, and 4's for each non-overlapping section of 10000 consecutive dbrs. The sample coefficients of variation of the run lengths in some of the states were too high to be consistent with the Markov model. Sample statistics for Sections 1-3 of D are given in Table 7.

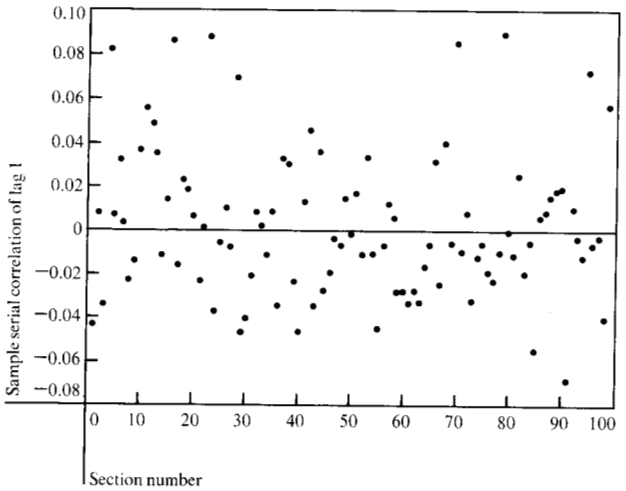


Figure 7 Sample serial correlations of lag 1 vs section number for L_1 to L_1 (section size = 1000).

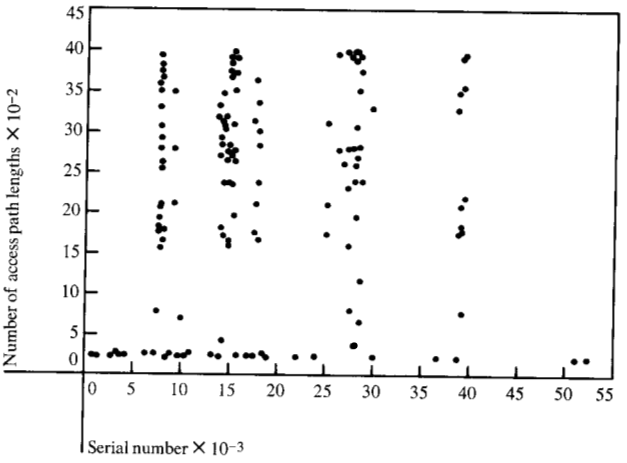


Figure 8 Access path lengths > 211 vs serial number for L_2.

Figure 9 Cumulative counts of references to data bases 1-4 (axes labeled in units of 1000).

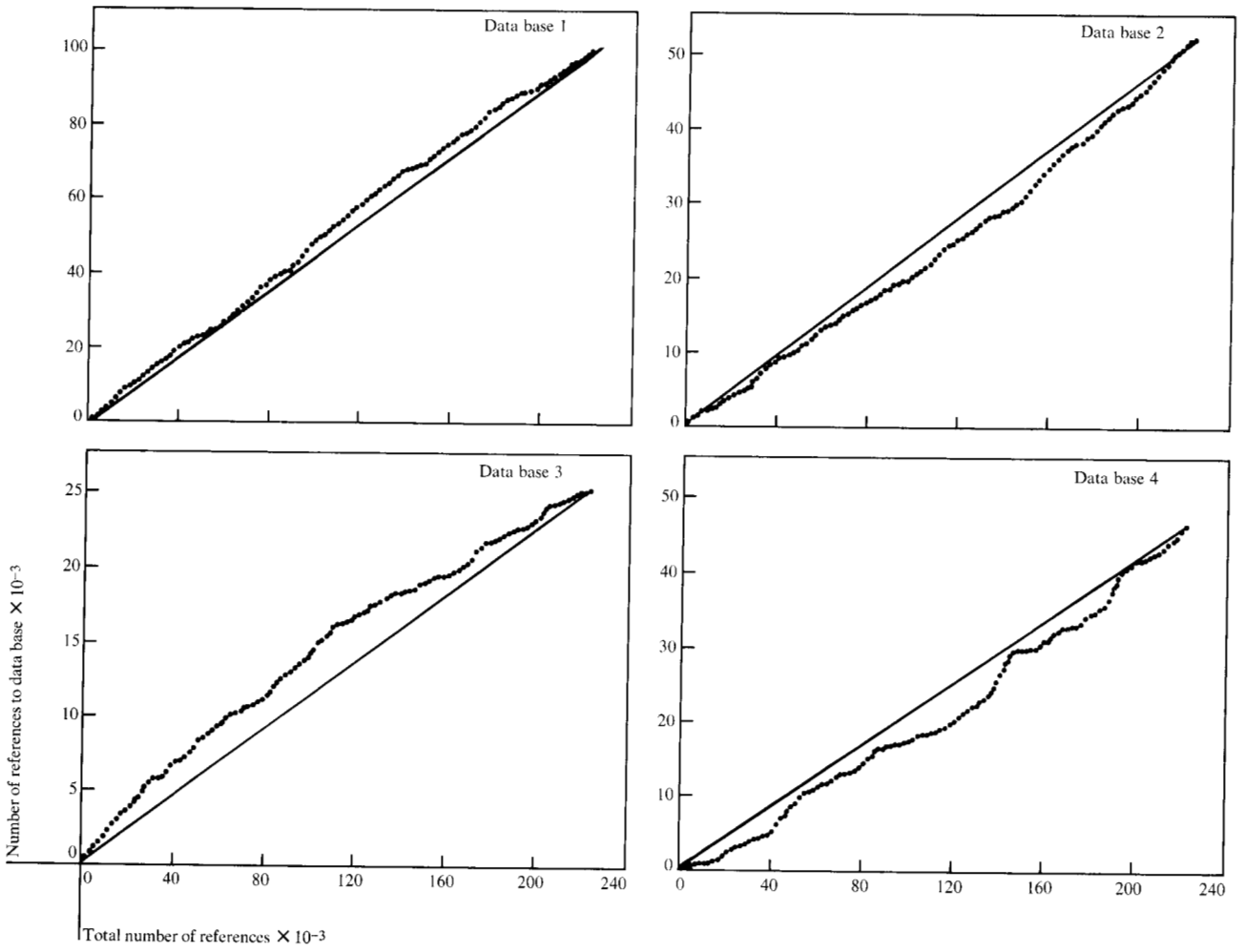


Table 8 Estimated transition matrices and powers of estimated transition matrices for D.

$(T_1)^2 =$

0.669	0.290	0.022	0.018
0.556	0.324	0.014	0.105
0.053	0.084	0.858	0.005
0.048	0.099	0.013	0.840

$T_2 =$

0.719	0.259	0.016	0.006
0.473	0.371	0.019	0.137
0.052	0.068	0.877	0.003
0.036	0.121	0.011	0.832

$(T_1)^4 =$

0.612	0.292	0.038	0.058
0.559	0.278	0.031	0.132
0.128	0.115	0.739	0.018
0.128	0.130	0.024	0.718

$T_4 =$

0.726	0.233	0.025	0.016
0.397	0.364	0.036	0.204
0.107	0.096	0.789	0.008
0.077	0.168	0.021	0.733

Table 9 Estimated transition matrices and powers of estimated transition matrices for D′.

$T_1' =$

0	0.930	0.069	7×10^{-4}
0.892	0	0.012	0.096
0.264	0.736	0	0
0.095	0.771	0.134	0

$(T_1')^2 =$

0.848	0.052	0.012	0.089
0.012	0.912	0.075	6×10^{-4}
0.656	0.246	0.027	0.071
0.723	0.187	0.016	0.074

$T_2' =$

0.867	0.051	0.012	0.070
0.012	0.917	0.070	6×10^{-4}
0.637	0.201	0.074	0.088
0.556	0.190	0.013	0.242

$(T_1')^4 =$

0.791	0.110	0.015	0.083
0.071	0.851	0.070	0.007
0.628	0.278	0.028	0.066
0.679	0.225	0.024	0.071

$T_4' =$

0.809	0.105	0.014	0.073
0.066	0.858	0.069	0.007
0.630	0.271	0.042	0.056
0.565	0.219	0.039	0.176

Another way to informally investigate the adequacy of a first-order Markov chain model for D is to compare the nth power of the estimated one-step transition matrix for D with the estimated n-step transition matrix for D. These should be approximately equal for a first-order Markov chain. (This method has been used to investigate Markov chain models of social and occupational mobility [6].) Denote by T_1 the estimated one-step transition matrix for D given in Table 6; $(T_1)^2$ and $(T_1)^4$ are given in the left half of Table 8. The estimated two-step and four-step transition matrices, denoted by T_2 and T_4, are given in the right half of the table. Arbitrarily define there to be a substantial difference between corresponding entries of $(T_1)^2$ and T_2 or between corresponding entries of $(T_1)^4$ and T_4 whenever the entries differ in magnitude by 0.05 or more; the corresponding entries which differ substantially are enclosed in boxes in Table 8. This table provides further evidence as to the inadequacy of a first-order Markov chain model for D.

Although D is apparently not adequately represented by a first-order Markov chain, it is possible that a higher order Markov chain would provide an adequate representation. This is not investigated here. Alternatively, a four-state semi-Markov process [7] with a state transition occurring at the beginning of every run might provide an adequate model. The run length in each state of a semi-Markov process has an arbitrary distribution and successive run lengths in a particular state are i.i.d. random variables. The embedded sequence of states that occur at the beginning of successive runs in a semi-Markov process is a first-order Markov chain. Note that successive states in the embedded sequence are not equal by definition.

The embedded sequence D′ and the sequences of run lengths in the four states were extracted from the first 30000 dbrs in D in order to investigate the adequacy of the semi-Markov model. D′ is a sequence of size 6686, and the run length sequences have sizes 2928, 3167,

Table 10 Sample statistics for fun lengths for first 30000 dbr's.

State	Sequence size	Mean	Coef. var.	r_1
1	2928	5.16	5.33	−0.090
2	3167	1.85	2.86	14,0
3	284	19.0	1.71	−0.211
4	306	11.5	1.81	0.063

Table 11 Number of calls due to five most active transactions

Transaction	Number of calls	Number of calls to data base			
		1	2	3	4
1	39657	28180	302	11175	0
2	32444	27560	4884	0	0
3	26659	0	181	0	26478
4	22215	10961	11254	0	0
5	20251	11757	8494	0	0

284, and 306, respectively. Since all run lengths for D′ are equal to one, the adequacy of a first-order Markov chain model for D′ cannot be assessed based on run lengths. The estimated one-step transition matrix for D′, denoted by T_1', and the second and fourth powers of this matrix are given in the left half on Table 9. The estimated two-step and four-step transition matrixes, denoted by T_2' and T_4', are given in the right half of the table. Corresponding entries which differ substantially (by at least 0.05 in magnitude) are enclosed in boxes in the table. All corresponding entries in the first two rows of $(T_1')^2$ and T_2', and in the first three rows of $(T_1')^4$ and T_4' differ in magnitude by less than 0.02. D′ is apparently more adequately represented by a first-order Markov chain than is D. Sample statistics for the run length sequences are given in Table 10. The quantity r_1 is the normalized sample serial correlation of lag 1, obtained by multiplying the sample serial correlation of lag 1 by the square root of the sample size. For a sequence of n i.i.d. random variables, where n is large, r_1 is approximately normally distributed with mean $1/\sqrt{n}$ and variance one [5], and a value of r_1 greater than 3 is very unlikely to occur. Thus, there is strong evidence that successive run lengths in state 2 are dependent. (The sequence of run lengths in state 2 contains 4 values equal to 147 and no other values greater than 18. Two of the 147's occur in succession and may considerably inflate the value of r_1. The value of r_1 computed with the 147's removed from the sequence is equal to 10.1, which is still significantly large.) Based on the values of r_1, successive run lengths in states 1, 3, and 4 appear to be independent.

The data base reference sequence has the following gross characteristics:

1. The sequence is not stationary over the day.
2. A first-order Markov chain is an inadequate model for the sequence; a semi-Markov process more adequately represents the sequence, although there are some large discrepancies between the model and the data.

• *Analysis of access path length sequences for different transactions and data bases*

It is plausible that the access path length for a data base call depends on the transaction that gives rise to the call in addition to the data base referenced by the call, since different transactions cause different application programs to be executed. Five transactions account for 63 percent of the data base calls for the day-long period being considered. The number of calls due to each of these transactions and the number of calls to each data base due to each of these transactions are given in Table 11. The cumulative count of data base calls due to each of these transactions is plotted versus the serial numbers of the calls in Fig. 10, where the axes are labeled in units of 1000 calls. Notice that the rate at which each transaction issues calls varies over the sequence of calls. It is possible that the variability of the sample statistics from section to section of each of the sequences L_1-L_4 is largely due to the variability from section to section of the number of calls arising from each transaction. Sample statistics for the apl sequence for a particular transaction and data base might be relatively stable from section to section.

To investigate this hypothesis, sequences L_{11}, L_{13}, and L_{21} were extracted from L where L_{ij} is the apl sequence for transaction i and data base j. (L_{11} and L_{21} contain over one-half of the apls in L_1, and L_{13} contains slightly less than one-half of the apls in L_3.) Each of these sequences was divided into non-overlapping sections of 1000 consecutive apls; the number of sections for L_{11}, L_{13}, and L_{21} is 28, 11, and 27, respectively. For each sequence the sample mean, coefficient of variation, and serial correlation of lag 1 were computed for each section. The mean and standard deviation across sections of each of the sample statistics are given in Table 12, where the standard deviation appears below the mean. Notice that the standard deviations across sections of the sample statistics for L_{11} are comparable in value to the standard deviations for L_1 given in Table 5. Thus, the sample statistics for L_{11} are as variable from section

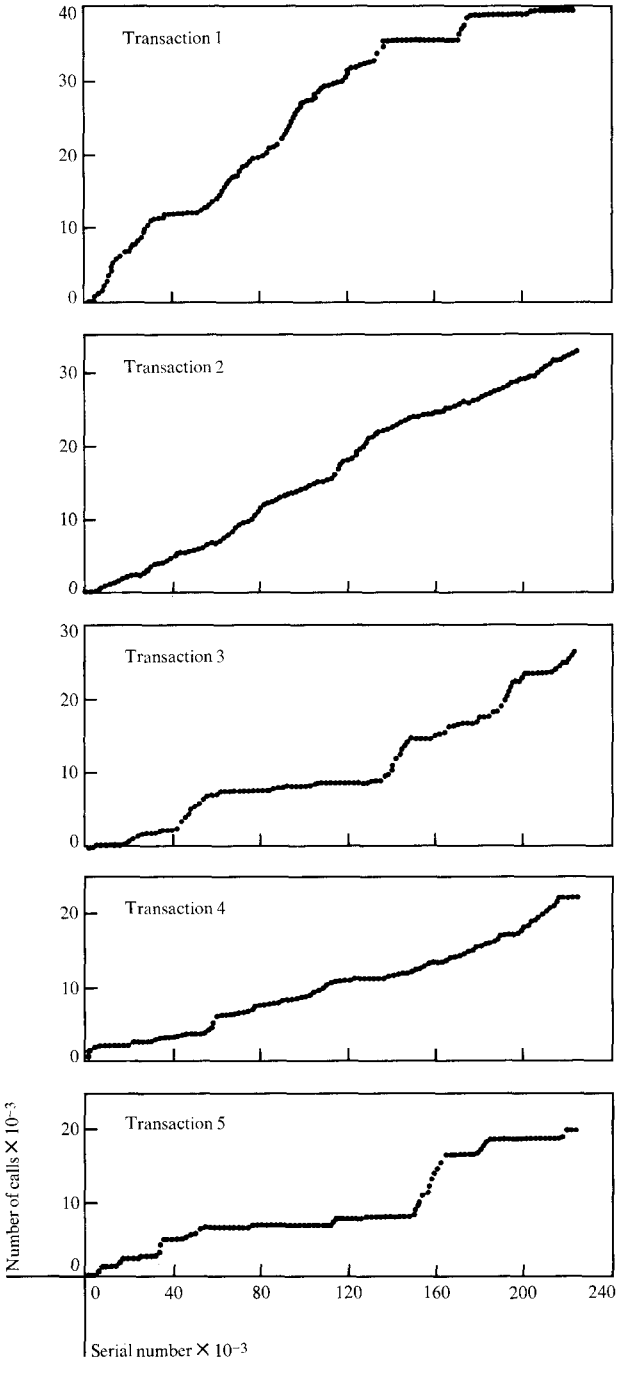


Figure 10 Cumulative counts of calls for five most active transactions.

to section as the sample statistics for L_1. The sample statistics for L_{21} are less variable than those for L_1, and the sample mean and serial correlation of lag 1 for L_{13} are less variable than for L_3. (The sample coefficient of variation is, however, more variable.) The sample serial correlation of lag 1 for each section of L_{11} is plotted versus the serial number of the section in Fig. 11. The values vary greatly from section to section, and there is evidence of significant serial correlation for several sections. It appears that it would be no easier to develop an adequate stochastic model for L_{11} than to develop one for L_1. The apl sequences for different transactions and data bases were not investigated further.

Models for the simulation of access path lengths

The analysis of the previous section shows that access path lengths have the following characteristics:

1. Access path length distributions, as represented by log survivor functions, appear to differ between data bases. Graphical presentations (see Fig. 5) provide the evidence.
2. The log survivor functions reveal that while the majority of the apls are short (≤ 12 about 90 percent of the time), the overall distributions are skewed to the right, showing a few preferred values, and eventually truncating or cutting off at a large finite value. The cutoff is relatively smooth for apls from data bases 1 and 4, appearing more abrupt for data base 3; the graph does not display a cutoff for data base 2. Thus, apl behavior cannot be realistically modeled as a geometric distribution with a linearly decreasing log survivor function. Modification of the geometric is evidently required.
3. Successive apls are somewhat correlated. The degree and sign of the correlation are peculiar to the data base being referenced.

Models for apls that will provide the type of qualitative behavior noted above are now introduced. These models can readily be fitted to the observed log survivor function and are a convenient means for generating simulated apls having the observed characteristics 1)-3).

• *The representing function method*

First the sequence L_1 is considered. The log survivor function for L_1 appears similar to that of a geometric distribution for small variable values (say ≤ 10, but excluding zero), departs therefrom by decreasing more slowly (long-tailed behavior) for intermediate values (approximately in the range 11-75) and finally cuts off at large values (around 113). A random apl, Y, will be represented in terms of an underlying unit exponential random variable, X, as follows:

Let

$$\phi(X; \alpha, \beta, \varepsilon) = (\beta X e^{\alpha\beta X}) / (1 + \varepsilon\beta X e^{\alpha\beta X}),$$

and

$$Y = [\phi(X; \alpha, \beta, \varepsilon)], \qquad (1)$$

where [a] denotes the integer part of a and $\alpha, \beta, \varepsilon > 0$; ϕ is called a *representing function* [8]. In order to simulate apls Y, exponential realizations are obtained from X, where $P\{X > x\} = e^{-x}$, and converted to Y-realizations by use of Eq. (1). Examination of Eq. (1) reveals that Y behaves like a discretized exponential random variable (i.e., like a geometric random variable) for small X and cuts off at $1/\varepsilon$; at intermediate values it exhibits long-tailed behavior due to the influence of $e^{\alpha\beta X}$ on X. Methods for introducing correlations between successive Y-values will be considered shortly.

• *Fitting the representing function*

For simplicity, Y is not discretized in what follows; instead, $Y = \phi(X; \alpha, \beta, \varepsilon)$. Observe that if $F(y)$ is the distribution function of Y, then

$$1 - F(y_p) = e^{\Lambda(y_p)} = 1 - p = e^{-x_p},\ 0 \le p \le 1,$$

where y_p is the pth quantile of the Y-distribution, and x_p is the pth quantile of the unit exponential distribution. Hence, the log survivor function of Y, namely $\Lambda(y)$, satisfies

$$\Lambda(y_p) = \log(1 - p) = -x_p. \tag{2}$$

Furthermore, the monotonicity of ϕ guarantees that

$$y_p = \phi(x_p; \alpha, \beta, \varepsilon). \tag{3}$$

In order to fit ϕ to empirical data, α, β, and ε must be chosen. This may be done by matching suitable quantiles using Eq. (3). Most easily, $1/\varepsilon$ is the cutoff point for Y; from Fig. 5 this is 113 for L_1 so that one parameter value is established. Next, two other points on the empirical log survivor function are selected. Examination of the empirical log survivor function for L_1 reveals turning points in the neighborhood of $n = 12.5$, $r(n) = -2.37$ and $n = 75$, $r(n) = -6$; these two points are now selected.

1. Select $y_p = 12.5$, and set $\Lambda(y_p) = -2.37$; from Eq. (2) $x_p = 2.37$ and $p = 0.907$.
2. Select $y_p = 75$, and set $\Lambda(y_p) = -6$; from Eq. (2) $x_p = 6$ and $p = 0.998$.

This choice leads to the simultaneous equations

$$12.5 = (2.37\beta e^{2.37\alpha\beta}) / [1 + (2.37/113)\beta e^{2.37\alpha\beta}],$$

$$75 = (6\beta e^{6\alpha\beta}) / [1 + (6/113)\beta e^{6\alpha\beta}],$$

which are readily solved by a Newton-Raphson iteration procedure, as are other similarly derived equations for the log survivor functions for L_2, L_3, and L_4. The fitted values of α, β, and ε are given in Table 13. Plots of the fitted log survivor functions are superimposed on plots of the empirical log survivor functions in Fig. 12; the solid line curves are the fitted functions. For L_2 the cutoff value is so large that $1/\varepsilon$ was chosen equal to infinity (in which case $\phi = \beta X e^{\alpha\beta X}$). The empirical log survivor function for L_3 has an abrupt cutoff at 211. This log survivor function was fitted by choosing $1/\varepsilon$ equal to infinity and letting $\hat{Y} = \max(Y, 211)$; the log survivor function for $\hat{Y}$ is plotted in the figure.

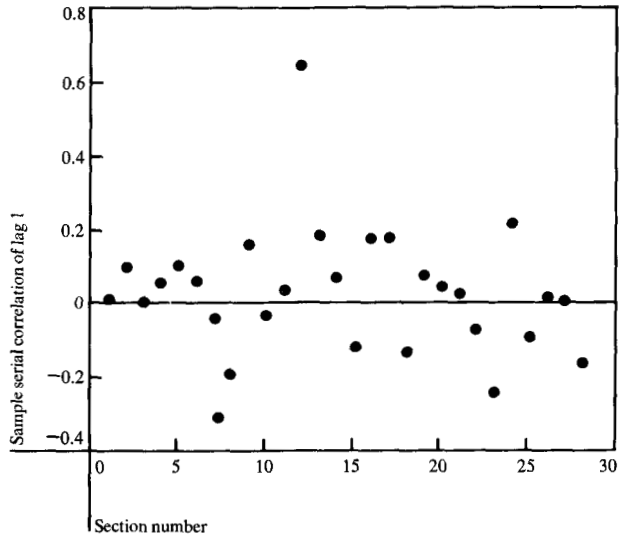


Figure 11 Sample serial correlation of lag 1 vs section number for L_{11} (section size = 1000).

Table 12 Mean and standard deviation across sections of sample statistics for L_{11}, L_{13}, L_{21} (section size = 1000).

	L_{11}	L_{13}	L_{21}	
Mean	5.14	2.09	2.90	Mean
	2.26	0.381	0.561	Std. dev.
Coef. var.	1.50	4.29	1.12	Mean
	0.291	0.867	0.172	Std. dev.
Ser. cor. 1	0.036	−0.019	−0.050	Mean
	0.172	0.007	0.057	Std. dev.

Table 13 Fitted parameter values for log survivor functions for L_1, L_2, L_3, L_4.

	α	β	$1/\epsilon$
L_1	0.283	1.79	113
L_2	0.126	3.84	∞
L_3	16.6	0.0984	∞
L_4	0.106	3.36	36

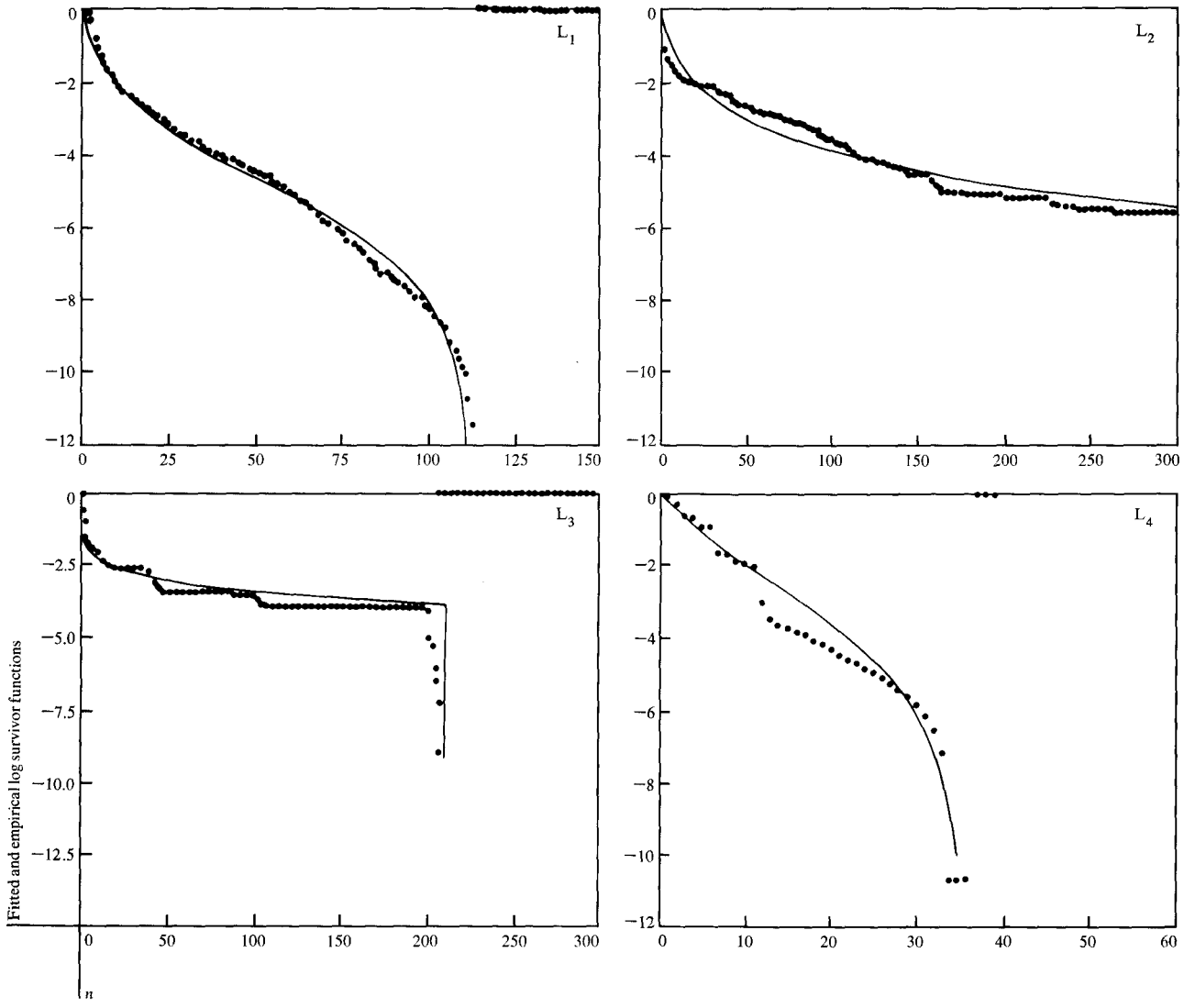


Figure 12 Fitted and empirical log survivor functions for L_1, L_2, L_3, L_4 .

By and large, the general shapes of the empirical log survivor functions are imitated by the simple parametric models with the parameter values given in the table. Simulation of independent apls via the models is easily accomplished. One noticeable difference between the simulated apls and those observed in the data will be the smoothness and lack of preferred values in the former. No physical explanation has been obtained for these preferred values; they may not remain a significant feature as more data are analyzed.

• *Correlations*

Analysis of the apls in successive sections of sequences L_1-L_4 revealed that mild but definite (statistically significant) correlations, of both signs, exist between two successive apls. Very likely some kind of dependency structure exists between apls of even greater separation. Although no physical explanation for their behavior can be advanced at this date, it is of interest to suggest simple schemes for simulating such correlated apls. The following are a few tentative suggestions. None has been, as yet, fitted to the available data.

A) A method based on a first-order autoregressive sequence of exponentials

Gaver and Lewis [9] showed that a sequence $\{X_n\}$ of random variables having unit exponential marginals may be constructed as follows:

$$X_{n+1} = \rho X_n + \varepsilon_n,\ n = 0,\ 1,\ 2, \cdots,$$

V. Y. Lum
N. C. Shu
B. C. Housel

A General Methodology for Data Conversion and Restructuring

Abstract: This paper presents a methodology and a model for data conversion or translation. The model assumes that both source and target systems are available and that conversion interfaces may be required to interact between these systems and the conversion system. To achieve data conversion or translation using this approach, two languages are needed: 1) a language to describe the data structures, and 2) a language to specify the mapping between source and target data. This paper describes these two languages, DEFINE and CONVERT and gives numerous examples to show the capabilities of these languages and how they can be used in data conversion and restructuring. Both languages are high level and nonprocedural and have the power to deal with most situations encountered in data conversion processes. In addition, the paper also describes some of the facilities in the languages specifically designed for data checking in a data conversion process.

Introduction

In recent years applications of data base systems have grown very rapidly. While the use of data base systems relieves users of the task of having to know much of the implementation details, it has at the same time made data conversion a necessity because of various reasons. In general, data conversion is a complex problem requiring more of our attention than it has received in the past. This paper proposes a solution applicable to a broad class of logical data conversion problems.

Relatively little work has been done to find a solution making data conversion easier [1-11]. All investigations so far are preliminary. Only few individuals are actively involved. The most comprehensive work is done by members of the Stored Data Definition and Translation Task Group under CODASYL's System Committee, which attempts to develop a general method for defining data structures, storage structures, their relationship, and translation from one structure to another. Similar work goes on at the University of Michigan and to a lesser extent elsewhere (see references). The paper of Sibley and Taylor [11] gives a good account of some of these related works.

As reported in reference [12], the authors initiated a similar project at IBM. This project was established to investigate and develop a methodology for application conversion and migration. Application conversion is defined to include the movement of both data and programs from one system (or one form) to another. After studying the problem for some time, it became clear that current technology is inadequate in solving the general problem. Our initial attack is to solve first the problem of data conversion. This approach not only provides us with a more fundamental understanding of the problem but it actually is a necessary first step since we must understand what is needed for data conversion before we know what is to be done in the programs. Attention is paid, however, to the larger problem so that the results obtained can be used as a foundation in the solution of total application migration.

At present data conversion is done infrequently because of its complexity. In spite of changes in requirements, users are reluctant to change their data structures. It is believed that conversions will take place more frequently when better techniques are known, when automatic or semi-automatic aids are available, and when greater data independence is achieved.

Problem environment

A study of current works revealed that current approaches to data conversion are either too broad and general, as in the case of CODASYL Task Group or Smith's and Taylor's work [5, 6], or too narrow in application as in Lin ahd Heller [13]. In the first case an economically feasible solution requires much more research and, therefore, appears distant. In the second case, a narrow approach is not really solving the main problem and, therefore, will provide benefits to only a small subset of computer users. The approach we have adopted is a compromise which will provide help to a broad class of users in the near future.

The approach assumes that the conversion system will run under the operating system of either the source or the target system. We also assume that an interface on the source system is available to transform the source data into an intermediate form acceptable to the translator, and an interface on the target system is available to take the output of the translator and transform it into the target data. In this way the translator is shielded from many of the physical incompatabilities of the source and target data such as parity schemes, etc. Specific details of our model are discussed later.

A basic assumption in our approach is that it is generally impossible to perform data conversion without the users' help. It is therefore visualized that it is the users' responsibility to describe the data structures for both the source and target data and to define the mappings between them. It is possible, however, to have an advanced system which may provide some prompting through interaction.

Two languages have been defined for this purpose: 1) DEFINE a language to define data structures, and 2) CONVERT, a language to specify mappings between source and target data, each of which may contain multiple logical record types and logical views. This paper discusses at some length these two languages. For a complete discussion, readers should refer to [14, 15].

In designing these languages we assumed that the users are skilled programmers. The programmers are familiar with their data's content, not in the sense of how many screws and nuts are in a parts' file, but in the sense of knowing that there exists a field for describing a part and that this field may contain blanks if no description exists. They know the semantics of their data and its structure at a logical level and what they want to be done in the mapping process. These aspects are quite different from the assumptions of the designers of data base systems who frequently consider their users to be casual users with little knowledge of the underlying data structure.

Assuming that the users are sophisticated and know their data, they do not know, however, the implementation details of their data structure, nor do they want to be burdened with the details of how to accomplish the whole conversion process. Another assumption is that the users are willing to follow some simple syntactic rules of the languages, but are unwilling to learn another complex language comparable to, say, COBOL or PL/1. We have also assumed that these users are not mathematically oriented and they do not appreciate semantics in mathematical terms. As a result we set out at the beginning to make our languages high level, nonprocedural, easy to learn, and simple to use.

The above aspects cannot be achieved without some expense. As opposed to a general language like PL/1, our languages are simple only because we tailored them to a specific purpose, namely, data conversion and in certain cases we traded capabilities for simplicity. Our philosophy is to provide a language to handle a great majority of the cases encountered frequently in data conversion and let the remaining small number of cases be handled by the computer's procedural languages. In any case, the languages have been so structured that additional capabilities can be included without much difficulty.

The conversion model

Figure 1 illustrates the overall conversion process in our model. The source systems which originally process the source data is used to access it and interacts with the conversion interface module to produce a nearly system independent source data called linearized source files. As the name implies, linearized files are sequential files. (More is said about them in a subsequent section). These files become the input to the converter/translator. The output from the converter/translator is another set of linearized files called linearized target files, which are changed into physical target files with the use of the conversion interface and the target system.

Generally speaking, data conversion can be divided into two basic categories: 1. from files to data base, and 2. from data base to data base. These two categories have some basic differences. Several points are salient in the first case. 1) Data is generally not well organized. It contains much redundancy and much of the data description is carried implicitly in the procedures. In fact, frequently additional information is contained there. For example, a census file may be separated into two parts such that the first part contains information about males and the second part about females, but this separation is not stated explicitly when the data structure for this file is defined. In our system all this descriptive information is made explicit. 2) These source files are sequential files. Since the real world at this time has a preponderance of sequential files to be converted to data bases, we have attempted to define in our data definition language a capability that can describe most of these files instead of imposing severe limitations on the formats of linearized files. 3) The COBOL files deserve further attention because a great majority of commercial users are COBOL oriented. Hence, our data definition language has been designed to have a strong COBOL flavor and the capability to describe the common COBOL files. Thus, we define a linearized file to be a file belonging to that subset of sequential files describable by our data definition language. It may have a flat or hierarchical record structure. It may contain self-defining data, terminators of different kinds, multiple record types within the same