Techniques

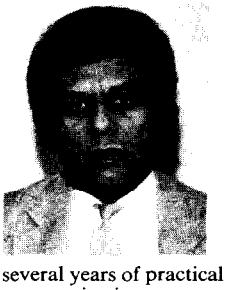
Pragmatic issues in conversions of database applications

Amjad Umar
School of Management, University of Michigan, Dearborn, MI 48128, USA

Daniel Teichroew
Program for Information Systems Engineering, Industrial and Operations Engineering Dept, The University of Michigan, Ann Arbor, MI 48109-2117, USA

The need to convert database applications from one DBMS to another arises frequently. This paper presents a cost/benefit evaluation model for the database conversion problem, applies this model to several situations involving convertion of large database applications and proposes a decision support system for database conversion decisions. It also points out that although most of the literature focuses on database models and structures, in practice organizational and human considerations play a key role in the conversion of database applications and may become a limit to the growth of new database applications.

Keywords: Database Conversions, Application Conversions, Database Applications, Database Management Systems, DBMS, Conversion Evaluation Model, Cost/Benefit Analysis, Performance Models, Conversion Case Studies.



Amjad Umar is an Associate Professor of Business Administration and Computer Information Systems at the University of Michigan-Dearborn and is a Research Scientist in the Program for Research in Information Systems Engineering (PRISE) at the University of Michigan. He has an M.S. in Computer, Information and Control Engineering and a Ph.D. in Industrial and Operations Engineering (major in Information Systems) from the University of Michigan. Dr. Umar has several years of practical industrial experience in database/data communication systems. He has consulted and taught industrial seminars in the U.S., England, Singapore and China.

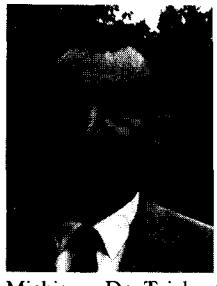
North-Holland
Information & Management 19 (1990) 149–166

1. Introduction

A database application (AP) uses a database management system (DBMS) to store, retrieve and manipulate data. Many such database applications are being developed at present and are an integral part of the future of computing technologies. In addition to the common business and management applications, databases are being used in CAD/CAM systems, software engineering environments, statistical applications, network' management, knowledge-based systems, real-time systems, and distributed systems.

Database applications utilize one or more databases, consist of one or more application programs and/or interactive queries, and use one or more procedures (run time commands) to execute various programs and perform adminsitrative and operational services (define security tables, backup and recovery procedures, normal and abnormal system startup procedures, specialized utility procedures, etc). A database conversion problem is frequently encountered when a database application implemented in one DBMS (D1) needs to be converted to run on another DBMS (D2). Common reasons for the need for conversion are

- The data model of D2 is more powerful and more suitable for AP.



Daniel Teichroew is a Professor of Industrial and Operations Engineering and Director of the Program for Research in Information Systems Engineering (PRISE), formerly ISDOS, project at the University of Michigan. He has been a Professor of Business Administration at Stanford University, Chairman of the Decision Sciences Department at Case Western Reserve University, and Chairman of the Industrial and Operations Engineering Department at the University of Michigan. Dr. Teichroew has been a Chairman of the Society of Management Information Systems. He has a Ph.D. in Statistics from the North Carolina State University.

- AP does not satisfy the stated performance requirements and D1 does not provide enough performance capability.
- D1 is not a distributed DBMS (DDBMS) and there is a need to distribute AP.
- AP has outgrown the capabailities of D1 in terms of the number of users supported, the number of active databases available to a user, and the maximum size of the databases etc.
- AP has stringent security, integrity and/or audit trail requirements that are not handled properly by D1.
- D1 does not support a programming language P which is (now) the programming standard of the organization.
- D1 is an old DBMS and the vendors are withdrawing support for D1.
- D2 is a new DBMS and the technical staff is anxious to learn about this new DBMS for their own professional growth.
- D2 is a key component of the architectural direction of a major vendor, e.g. SQL is a key component of IBM's System Application Architecture [18]. Adherence to such standards decouple the applications from specific computing platforms and thus increase the life of the application.
- The company wishes to market AP on a variety of equipment but D1 is not supported on all the available systems.

Database application conversion can be an expensive undertaking, because a large number of tangible and intangible factors may be involved. These factors must be considered carefully by management to evaluate the tradeoffs in converting database applications, because existing database applications reflect organizational investment. This issue is of importance at present, due to current interest in object oriented databases while most existing systems are still being converted from hierarchical/network DBMS to relational.

The goal of this paper is to study and analyze the tangible and intangible factors encountered in actual database application conversions and thus complement the current literature that places emphasis on the database structures and data models and does not address the technical, managerial, organizational, and economic tradeoffs in converting from one data model to another. For example, the current literature in object oriented database management systems (OODBMS) does not address the future problem of converting the growing number of application systems using the relational data model to object oriented data models.

2. Overview of Database Management System Capabilities

A database management system (DBMS) is a software package (in most cases, purchased commercially) that is responsible for storing, retrieving, and manipulating data. For the purpose of this paper, the capabilities of a DBMS are classified into the following major categories:

1. Capability to support one or more data models (DM) which allow a user to view and manipulate data. The following types of data models are in common at present ([24] gives a recent review of data models):

   - Hierarchical data models which view the data as hierarchical structures. This model is the oldest traditional data model and is supported by IBM's IMS.
   - Network data models which allow many to many relationships between data structures. A definition is given in the 1971 CODASYL Database Task Group Report [8], and has been expanded later. This model is supported by Cullane's IDMS, among others.
   - Relational data models which view the data as a collection of tables (relations) that are searched on values instead of pointers. Due to the performance limitations of value based searches, "pseudorelational" models have been suggested which allow direct pointers in relational tables [12].
   - Semantic and hypersemantic data models which allow a user to specify data in terms of objects and relationships between the objects in order to capture the meaning (semantics) as well as the structure. These models are the basis for the object oriented and framebased database management systems [1,4,19,33].

2. Data definition facilities which allow users to define data model for a specific application interactively or through batch processing by using a data definition language (DDL) and/or a graphic interface.

3. Data manipulation facilities which allow users to access and manipulate data by using a data manipulation language (DML) or a graphics interface. A DML itself may be a standalone interactive query language, a query language that can only be used when embedded in traditional procedural programming languages (e.g. IMS DLI), a specialized query language that may be executed as a procedure (e.g. DBASEIII DML) or a query language that may be executed interactively or embedded in a procedural programming language (e.g., SQL). In addition, some DML may support special facilities, like report writing and graphics.
4. Data administration facilities which allow a database administrator to enforce data security, to backup and recover selected data, the DO/REDO/UNDO logging facilities to isolate the effect of failing transactions, and the locking/unlocking facilities to provide concurrent access to shared data. An administrator may interact with a DBMS through a data control language (DCL). Basically, a DBMS must guarentee the integrity of data so that if data item d is modified, deliberately or inadvertantly, then all the related data items d' may have to be modified by the DBMS. It is also becoming increasing important for database administrators to provide encryption/decryption and comprehensive audit trails for forward and backward audit controls for sensitive data.
5. Database operation and support facilities/features which include the installation procedures, performance evaluation and tuning aids, startup/shutdown procedures, operator commands for status review and diagnostics, and specialized utilities for data conversion and reports. In addition, the storage/processing/operating systems requirements of DBMS and the vendor support and documentation may be included in this category.

Any two DBMS D1 and D2 will differ in these five categories. The changes can be represented by a 0-1 vector $CDBMS(i)$ for $i = 1$ to 5 where a 0 indicates no change and a 1 indicates a change in the following DBMS facilities:

- $i = 1$ indicates data model changes between D1 and D2
- $i = 2$ indicates data definition facility changes between D1 and D2
- $i = 3$ indicates data manipulationn facility changes between D1 and D2
- $i = 4$ indicates data administration facility changes between D1 and D2
- $i = 5$ indicates data operation and support facility changes between D1 and D2

For example, $CDBMS = [1\ 1\ 0\ 1\ 1]$ indicates that other than the data manipulation facility (e.g., SQL interface) all other facilities of DBMS D1 and D2 are different. The changes in the DBMS capabilities serve as the basic parameters in the conversion evaluation model.

3. A Conversion Evaluation Model

The database conversion decision from DBMS D1 to D2 can be viewed as a surrogate for the cost/benefit evaluation of information systems. Current techniques include accounting methods, such as present value and opportunity cost, quantitative methods, such as Bayesian analysis, and subjective estimation, such as Delphi techniques. Fig. 1 shows how the database conversion evaluation model may be subdivided into three: a one-time cost, a recurring benefit, and a performance submodel. These three submodels separate the mixture of tangible/intangible and the one-time recurring factors encountered in database application conversions.

3.1. Submodel for One-Time Costs (The Investment)

The total one-time tangible cost for converting a database application AP from DBMS D1 to D2 can be estimated by:

$$FC = \sum_{i=1}^{6}\sum_{j=1}^{5} TR(i, j).CDBMS(j) + \sum_{i=1}^{3}\sum_{j=1}^{5} RE(i, j).CDBMS(j) + \sum_{i=1}^{3}\sum_{j=1}^{5} UP(i, j).CDBMS(j) + \sum_{i=1}^{3}\sum_{j=1}^{5} AD(i, j).CDBMS(j),$$

where TR, RE, UP and AD represent the one-time cost matrices associated with the training,

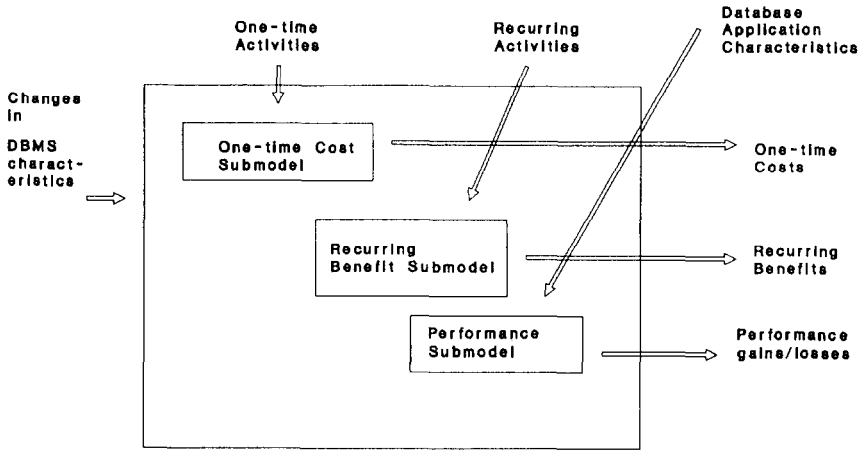


Fig. 1. A Database Conversion Evaluation Model.

Table 1
One-Time Cost Matrices

Changes in DBMS Characteristics from D1 to D2	Symbols	DM $j=1$	DDF $j=2$	DMF $j=3$	DAF $j=4$	DOSF $j=5$
One-time activities						
(a) Training	$TR(i, j)$					
– Database designer training	$(i=1)$	X	X			
– Programmer training	$(i=2)$		X	X		
– Technical support training	$(i=3)$	X	X	X	X	X
– Administrator training	$(i=4)$				X	
– Operator training	$(i=5)$					X
– End-user training	$(i=6)$		X	X		
(b) Redesign/reconstruction	$RE(i, j)$					
– Database redesign	$(i=1)$	X	X			
– Software redesign and reconstruction	$(i=2)$	X		X		
– Database repopulation	$(i=3)$					X
(c) Hardware/software upgrade	$UP(i, j)$					
– DBMS replacement cost	$(i=1)$					X
– Hardware upgrade cost	$(i=2)$					X
– Operating system upgrade	$(i=3)$					X
(d) Administrative conversions	$AD(i, j)$					
– Planning costs	$(i=1)$	X				
– Procedures conversion costs	$(i=2)$				X	X
– Documentation costs	$(i=3)$		X	X	X	X

Notes and Explanations:
1. This shows the qualitative dependencies between the costs/benefits and the changes in database facilities.
2. DM, DDF, DMF, DAF, DOSF indicate changes in the data model, data definition facilities, data manipulation facilities, data administration facilities and data operation and support facilties between D1 and D2.
3. An X indicates a dependence between the costs/benefits and changes in DBMS characteristics, a blank indicates there is no dependence.

redesign/reconstruction of database and software, hardware/software upgrade and administrative planning activities, respectively, and CDBMS represents the DBMS change vector. For each cost matrix, i indicates a cost category (e.g. $i = 1$ for TR matrix shows the database designer training cost) and j shows the impact of changes in the DBMS facilities (e.g. $j = 1$ indicates the data model changes). A qualitative view of the one-time cost matrices is shown in Table 1 (the dependence of the cost category on the DBMS capabilities changes is indicated by an X).

A. Training costs (TR):

- Database Designer Training ($i = 1$) – the database designers may need to be retrained if the data models and the data definition facilities of D1 and D2 are different. A representative training cost is about $5000 per designer, based on an estimate of two classes per designer, $2500 per three day class ($1000 tution, $1000 travel and room and board and $500 for in house orientation). This estimate is based on common industrial seminars, e.g. Digital Consulting, Inc [9] and Society of Manufacturing Engineering [29].
- Programmer Training ($i = 2$) – the application programmers will need to be trained to understand new data definition facilities, code new DML statements and to reorganize the programs. For example, Cobol programs written for IMS virtually need to be rewritten for SQL. This cost can also be estimated at about $5000 per programmer.
- Technical Support Training ($i = 3$) – the technical support personnel (systems programmers) responsible for generating, configuring and monitoring the performance of database systems. The technical support personnel usually need more training than the application programmers (at least three three-day classes).
- Administrator Training ($i = 4$) – the database administrator will need to be retrained to take advantage of the new data management facilities of D2. This cost can be estimated at about $2000 per administrator for a two-day class.
- Operator Training ($i = 5$) – the operators will need to be trained for new operational procedures (system startup/$-shutdown, backup and recovery, diagnostic and error handling, emergency and disaster handling procedures, etc) at an estimated cost of $2000 per operator.
- End-User Training ($i = 6$) – the end-users will need to be trained to utilize the new data definition and manipulation facilities of D2. This training may be minimized by presenting the same external user interfaces after the conversion. In this case, the end-users will only need to learn additional features (e.g. ad hoc query language, graphics, spreadsheets, etc).

B. Redesign and Reconstruction Costs (RE):

- Database Redesign Cost ($i = 1$) – the database will have to be redesigned if the data model of D1 is not the same as that of D2, or if D2 is a DDBMS and D1 is a centralized DBMS. In addition, the DDF influences the databases redesign and reconstruction cost. This cost can be roughly estimated by the following simple formula:

$$= EFF \times MCOST \times \sum_{d=1}^{Z} ENT(d),$$

where $ENT(d)$ is the number of entity types (or relations) per database d of AP, EFF is the average effort in manhours to convert an entity type from D1 to D2 and $MCOST$ is the dollar cost per manhour. If D2 is distributed, then EFF is the manhours used to allocate an entity type to a given network and is also a function of the number of nodes in the network. Our industrial experience of converting IMS applications to relational applications indicates that EFF is approximately 8 to 10 manhours for a trained database designer.
- Software Conversion Cost ($i = 2$) – the software of AP may need to be redesigned and converted if the data model of D2 is not the same as D1 and/or the data manipulation facilities are different. The data manipulation facilities have significant impact on software conversion because software will need to be converted if (i) the DML of D2 is different from the DML of D1, (ii) the programming language for D2 is not the same as D1 and/or (iii) the screen formatting statements of D1 are different from D2. It should be noted that the software conversion cost is zero if AP is being converted to a DDBMS because DDBMS provide location transparency to applications. In other cases, the software conversion cost can be estimated by using one of the software cost estimation meth-

ods [25,32]. For example, the following formula may be used to estimate the lines of code, in thousands, ($KLOC$) that need to be rewritten:

$$KLOC(\text{AP}) = KLOC(\text{dd}) + KLOC(\text{dml}) + KLOC(\text{scr}),$$

where $KLOC$(dd), $KLOC$(dml) and $KLOC$(scr) indicate the lines of code, in thousands, due to data definition, data manipulation and screen formatting, respectively. In most cases, a program has to be almost rewritten when it is converted from one DBMS to another. For example, conversion of an IMS Cobol program to an SQL Cobol program requires modification of 80% of the code because most of the data definitions, DML statements and screen formatting statements are dependent on the DBMS.

- Database Repopulation Cost ($i = 3$) – the data in D1 will have to be converted and stored in D2. Ignoring the computing costs, this cost is primarily dependent on availability of conversion utilities.

C. Hardware / Software Replacement / Upgrade Cost (UP):

- DBMS Replacement ($i = 1$) – the cost of purchasing and installing D2 to replace D1 needs to be considered. This may include new hardware needed to support D2. In most practical cases, D2 is purchased, installed and operated in parallel with D1 while the conversions are being performed, often for up to a year
- Hardware Upgrade ($i = 2$) – the existing hardware may need to be upgraded (memory, cpu) to operate D2
- Operating System Upgrade ($i = 3$) – the current operating system may need to be upgraded for D2

D. Administrative Costs (AD):

- Planning Cost ($i = 1$) – the database conversion needs to be planned, responsibilities for all tasks need to be assigned and arrangements need to be made to minimize disruption of current operations. An estimate of this cost is between 2 to 3 man weeks.
- Operational Procedures Conversion Cost ($i = 2$) – the operational and administrative procedures of AP need to be converted. For experienced and trained personnel, this conversion cost can be estimated at approximately one to two man weeks.
- Documentation Cost ($i = 3$) – several documents need to be modified for end-users, upper management, programmers, and designers.
- Machine Cost ($i = 4$) – machine time, especially in mainframe environments, need to be considered for development, testing, and parallel operation

3.2. Submodel for Recurring Intangible Benefits and Payoff

The recurring intangible benefits in relative units for converting a database application AP from DBMS D1 to D2 can be estimated by:

$$RC = \sum_{i=1}^{4}\sum_{j=1}^{5} PR(i, j).CDBMS(j) + \sum_{i=1}^{5}\sum_{j=1}^{5} DA(i, j).CDBMS(j) + \sum_{i=1}^{3}\sum_{j=1}^{3} OP(i, j).CDBMS(j),$$

where PR, DA and OP are the intangible payoff matrices (in dollars) due to the recurring activities such as (1) human productivity, (2) daily administration, and (3) daily operation. Table 2 shows these payoff matrices. A management/user assigned weight may be used in calculating the intangible benefits. Utility theory may be used to evaluate the tradeoff between the tangible costs and intangible benefits produced by the one-time and recurrent submodels [26].

A. Productivity Gains (PR)

- End-User Satisfaction and Productivity ($i = 1$) – D2 may support the DMF and DDF which may improve the productivity of end-users and contribute to the end-user satisfaction. This may involve studies of user satisfaction of DBMS by using regression analysis [28] or other techniques.
- Programmer Productivity ($i = 2$) – DMF of D2 may be easier to program and D2 may support special high level facilities to improve the programmer productivity.
- Database Designer Productivity ($i = 3$) – D2 data model may be easier for database design

Table 2
Submodel for Recurring Benefits

Changes in DBMS Characteristics from D1 to D2	Symbols	DM $j=1$	DDF $j=2$	DMF $j=3$	DAF $j=4$	DOSF $j=5$
Recurring activities						
(a) Human Productivity	$PR(i, j)$					
– End-user productivity	$(i=1)$		X	X		
– Application programmer productivity	$(i=2)$		X	X		
– Database designer productivity	$(i=3)$	X	X			
– Technical support productivity	(i = 4)				X	X
(b) Daily Administration	$DA(i, j)$					
– Ease of administration	$(i=1)$				X	
– Corporate information gains	$(i=2)$			X		
– Employee turnover gains	$(i=3)$			X		X
– Data security gains	$(i=4)$				X	
– Data integrity gains	$(i=6)$				X	X
(c) Daily Operations	$OP(i, j)$					
– Reliability/availability gains	$(i=1)$				X	X
– Backup/recovery gains	$(i=2)$				X	X
– Concurrency gains	$(i=3)$				X	

Notes and Explanations:
1. DM, DDF, DMF, DAF, DOSF indicate changes in the data model, data definition facilities, data manipulation facilities, data administration facilities and data operation and support facilties between D1 and D2.
2. An X indicates a dependence between the costs/benefits and changes in DBMS characteristics.

and D2 may provide special tools for improved database design and design evaluation.
- Technical Support Productivity ($i = 4$) – D2 may provide easier methods for installation, modification and diagnostics

B. Daily Administrative Gains (DA)
- Ease of Administrator ($i = 1$) – DAF of D2 may provide special utilities and commands for improved administrator controls (security controls, integrity controls, and data dictioneries)
- Corporate Information Gains ($i = 2$) – D2 may provide better facilities for providing strategic planning (summaries, forecasting, and trend analysis).
- Employee 'Turnover ($i = 3$) – in some cases, good technical staff may leave if D2 is old and/or not very well known. This cost, if important, can be computed by using a staff replacement cost that takes into account the appropriate training costs.
- Data Security and Integrity ($i = 4$ and $i = 5$) – D2 may provide better control of data integrity by allowing automatic deletion and update of related data and may provide better security control

C. Operational Gains (OP)
- Reliability/Availability Gains ($i = 1$) – D2 may improve the reliability and availability of AP if D2 supports data duplication
- Backup/Recovery ($i = 2$) – D2 may have better facilities for backing up and recovering the entire databases or portions of the database
- Concurrency and Consistency ($i = 3$) – There may be differences between D1 and D2 in handling of concurrent transactions to update shared data

3.3. Submodel for Performance

AP may show different performance characteristics (resource utilization, average response time) when converted from D1 to D2. A performance and workload model is needed to predict if the converion of AP from DBMS D1 to D2 will exceed the resource capacity of existing computing systems. Detailed models can be found in literature [12,15,27,31]. The following simplified model has been found to be of most practical value. This model combines the overall effect of local database design with data allocation and queuing at local and network resources to estimate workload variations due to database conversions.

The workload W is divided into A applications, where each application processes T transactions

and each transaction consists of the following steps:

- Open Processing – the transaction execution is initiated by acquiring needed resources (cpu memory, disk blocks, files, etc). The lock requests are commonly issued in this step.
- Primary Processing – the data is read and written, the DO/UNDO/REDO logs are created, the computations are performed and the communication messages are read and written in this step.
- Secondary Processing – any duplicate data is updated in this step. If no duplicate copies of data exist, then this step is bypassed.
- Close Processing – the resources held by the transaction, including the locks owned by this transaction, are freed in this step.

In practice, a transaction may be subdivided into several subtransactions where each subtransaction may go through the above steps. The path length of a transaction t in terms of i/o and cpu instructions is given by the following formula:

$$\begin{aligned} PL(t) = {} & CPU_{op} + IO_{op} \\ & + NDB_{pp}.(CPU_{db} + IO_{db}.(1 + CPU_{io})) \\ & + NCIO_{pp}.(1 + CPU_{cio}) + CPU_{comp} \\ & + CPU_{cp} + IO_{cp} \\ & + NDB_{sp}.(CPU_{db} + IO_{db}.(1 + CPU_{io})), \end{aligned}$$

where CPU_{op} = cpu instructions due to open processing, IO_{op} = no. of i/o instructions in open processing, NDB_{pp} = no. of database calls issued during primary processing, CPU_{db} = no. of cpu instructions per database call, IO_{db} = no. of i/o per database call, CPU_{io} = cpu instructions per i/o, $NCIO_{pp}$ = no. of communication i/o during primary processing, CPU_{cio} = cpu instructions per communication i/o, CPU_{comp} = no. of cpu instructions issued for computations, CPU_{cp} = cpu instructions due to close processing, IO_{cp} = no. of i/o instructions in close processing and NDB_{sp} = no. of database calls issued during secondary processing. For most transactions, the path length due to open and close processing can be ignored and the cpu instructions due to computations can also be ignored for most business applications. The effect of locking/unlocking due to concurrency control algorithms can be included by assuming that each database call will require a lock/unlock [2,3,10,34]. This assumption works well when each database call accesses a different database tuple.

The main workload parameter is the number of database calls, measured in terms of logical i/o (LIO), where it indicates the number of references to a given database. The LIO is chosen as a basic performance variable because it can easily be estimated in terms of application characteristics (operation type and table sizes), it can be calibrated by using simple benchmarks to provide an estimate of time per LIO, and the path length can be estimated to compute the application path length.

The number of LIO for relational systems using indexed and non-indexed operations are shown in Table 3, and Table 4 shows the benchmarks con-

Table 3
Number OF LIO for Relational DBMS

SQL Operation	Logical IO (LIO)	
	No index	With Index
Selection T1	n	n
Projection T1	n	n
Join T1 with T2	$n \times m$	$n + m$
Union T1 with T2	$n \times m$	$n + m$
Insert k records	k	$k + k$
Delete k records	k	$k + k$

These estimates are based on the following observations:

- The indexed performance for Selection is somewhat exaggerated because in the absolute best-case a selection operation can be performed by just retrieving the tuples that satisfy the Where clause provided the Where clause only refers to the keyed attributes.
- The indexed and non-indexed estimates for a projection almost always will require access of all tuples.
- The estimation of LIO for Join operation is non-trivial. For example, the non-indexed estimate for Join is somewhat exaggerated and represents only an absolute worst-case scenario. In practice, most relational systems perform joins between non-keyed attributes by building a temporary index in the first scan and then using this index for future selection of records. Thus, the estimate for non-indexed join is $n + m + m$ (T2 is searched twice – once to build the index and then for actual record retrievals).
- The estimation of non-indexed LIO for Union is also exaggerated. In practice, an index can be built and used to determine the overlapping tuples.
- The estimates for Insert and Delete ignore the overhead due to creation of new record areas; the index adds overhead to the insertion/deletion I/O.

Table 4
LIO Benchmark on a One MIP Machines

Operation	Estimated LIO (using Table 3)	Total time	LIO per second
Join of 2 tables	20,000	174 secs	115
Join of 3 tables	40,000	386 secs	98
Join of 4 tables	50,000	603 secs	99
Join of 5 tables	60,000	583 secs	102

ducted to estimate $S(i)$, the service time per LIO at node i, for joins between tables with 10,000 tuples each on a 1 MIP machine. This table shows that an estimate of 100 LIO per second is adequate. Other benchmarks conducted with different systems yield $S(i) = 0.08$ sec on PS2 model 80 and $S(i) = 0.015$ for an IBM 3080. The benchmarks have shown that the LIO estimate of 10 to 100 per second is adequate for gross workload calculations.

This model assumes a typical computer communication environment where several micros, minis and mainframes are interconnected through a communication network. The input parameters of the model are: $R(t, d)$ = no. of read LIO issued from transaction t to dataset d; $U(t, d)$ = no. of update LIO issued from transaction t to dataset d – this includes the primary plus the secondary updates; $I(t)$ = no. of input (keyboard) messages read by transaction t during primary processing; $O(t)$ = no. of output (display) messages generated by transaction t during primary processing; $F(t, k)$ = arrivals of transaction t at node k per unit time; $A(d, i)$ = zero-one allocation matrix showing if dataset d is allocated to node i (= 1 if database d is allocated to node i, 0 otherwise); $L(t, k, i) = 1$ if t runs on node i when t arrives at node k, 0 otherwise; $M(d, k, i)$ – 1 if d is accesed at node i when any t arrives at k, 0 otherwise.

The first four parameters describe the transaction characteristics. $F(t, k)$ shows the throughput of the system and $A(d, i)$ shows where the databases are located in the system. The matrices $L(t, k, i)$ and $M(d, k, i)$ reflect the properties and options provided by various DDBMS and can be specified for given DDBMS. The basic outputs produced by this model are given by $TL(i)$ which shows the total number of LIO issued at node i per unit time and CIO(i, j) which shows the communication messages exchanged between nodes i and j. The following equations show these outputs:

$$TL(i) = \sum_t \sum_d \sum_k U(t, d).A(d, i).F(t, k) + \sum_t \sum_d \sum_k R(t, d).A(d, i) \times M(d, k, i).F(t, k). \quad (1)$$

$CIO(i, j) = 0$ if $i = j$ otherwise

$$CIO(i, j) = \sum_t \sum_d \sum_k L(t, k, i).U(t, d)A(d, j) \times F(t, k) + (* \textit{update} *) \sum_t \sum_d \sum_k L(t, k, i).R(t, d).A(d, j) \times M(d, k, j).F(t, k) + (* \textit{read} *) \sum_t \sum_d \sum_k L(t, k, i)(I(t) + O(t)).F(t, k). \quad (2)$$

$TL(i)$ and $CIO(i, j)$ reflect the node and communication workloads, respectively. Based on the standard class of open, analytically tractable Jackson queuing networks, the queuing delay at node i is given by:

$$QD(i) = S(i).QL(i) \quad (3)$$

where $S(i)$ = average service time per LIO at node i and $QL(i)$ = average queue length QL at node i, given by:

$$QL(i) = UT(i)/(1 - UT(i)) \quad \text{and} \quad (4)$$

$$UT(i) = \text{utilization of node } i = S(i).TL(i) \quad (5)$$

Equation (4) shows that the average queue length at i is less than 1 if the utilization of i is less than 0.5. This observation is used as a design rule of thumb for a given workload and can be used to calculate the desired $S(i)$ and to distribute programs and data. If the desired $S(i)$ exceeds the current system service time, then the capacity of current system will need to be expanded or the workload will need to be reduced by redistribution of programs and data to solve the following problem:

Determine $A(d, i)$ = allocation matrix showing if dataset d is allocated to node i
Subject to:
$TL(i) \leq TCL(i)$
$CIO(i, j) \leq TCIO(i, j)$

where *TCL* and *TCIO* are the technology constraints on workload.

4. Case Study: Conversion of a Business Application

This example shows the conversion parameters in a payroll/personnel application for a large regional educational computing center responsible for providing data processing services to approximately 30 goverment agencies. The center runs 1200 SNA terminals connected to 30 IBM 8100 distributed processors, each 8100 serving one agency. The 8100 systems are connected to a central IBM 3080 host under MVS and IMS.

4.1. Current Payroll / Personnel Application

The current system was implemented in the mid seventies and uses IMS, is written in Cobol and is operated on an IBM 3080 mainframe. The system conceptually consists of one large IMS logical database with 13 segments. There are about 20 large batch Cobol programs, 30 online programs and over 50 "ad hoc" queries. Datanalyzer from Decision Technology Inc. is being used to provide adhoc queries from IMS databases,

The logical database is partitioned into 30 identical physical databases due to backup/recovery and security/privacy issues. Several other IMS applications have been developed and the number of logical MS databases have exceeded 50. Since a logical database may itself consist of several physical databases due to primary and secondary indexes, the total number of physical databases at present exceeds 4500. This size presents unique challenges when conversion to other DBMS is considered.

There are serious backup and recovery considerations and disaster recovery plans are of utmost importance. The central site is legally responsible for the integrity of data. The payroll/personnel application at present is not distributed. The level of distribution is mainly front-end skeleton files which are created at the 8100 systems to perform front-end processing like range checks, etc.

The current system has been developed over the past 15 years and is quite reliable. The advantages of the current implementation are that IMS is at present a proven system with many backup/recovery and logging features, the data models for most applications are inherently hierarchical, the need for adhoc queries is minimal, and many performance measurement and evaluation tools are available.

4.2. Conversion Considerations: IMS to DB2

The management motivation for conversion from IMS to DB2/SQL is that SQL is a major ingredient of System Application Architecture (SAA) which, according to IBM, will provide transportable applications among IBM mainframes, micros and minis [16]. In addition, many DDBMS currently support SQL, thus it would be easy to distribute the applications at a future date. The center currently has installed IBM DB2 and is in the process of converting the applications.

Table 5 gives the values for the one-time and recurring cost/benefits of the decision to convert to DB2. The information shown in Table 5 is based on vendor provided data and rankings assigned by users, programmers and administrators.

The major observations about the IMS to DB2 conversion are:

- The total training cost is estimated at about $130,000 – a factor not previously considered.
- An initial relational database design for all of the applications in third normal form yields over 15,000 relational tables (conversion of one IMS payroll database requires approximately 13 tables).
- The one-time conversion costs from IMS to DB2 are high due to the differences between IMS and relational database design and the DML differences between IMS and SQL (80% of the code has to be rewritten).
- The recurring benefits of DB2 are mainly due to the availability of ad hoc SQL queries and the general ease of use of SQL over IMS DML. However, it has been found that, for many applications, the SQL code becomes extremely difficult to understand, especially when subqueries are used.
- A major benefit of conversion to DB2 is that the current staff is very excited about learning and working with a relational system.
- Backup/recovery and integrity of data is a major concern. For example, many relational DBMS do not support "referential integrity"

Table 5
Business Application Conversion: IMS to DB2

Activities	Cost/Benefit		Comments
One-Time Costs			
(a) Training			
- Designer training	$10K		2 designers
- Programmer training	$25K		5 programmers
- Technical support training	$18K		2 Tech. support
- Administrator training	$4K		2 Administrator
- Operator training	$6K		3 operators
- End-user training	$60K		30 users
(b) Redesign/reconstruction			
- Database redesign	$20K		50 databases
- Software redesign	$80K		120 programs (40 KLOC)
- Database repopulation	$18K		(10.000 tables)
(c) Hardware/software upgrade			
- DBMS replacement	$72K		(DB2 cost at $6K month)
- Hardware upgrade	0		(not needed)
- Operating system upgrade	0		(not needed)
(d) administrative costs			
- Planning costs	$10K		(1 man month)
- Procedures conversion costs	$10K		(1 man month)
- Documentation costs	$50K		(5 man months)
Total one time dollar estimated cost = $383,000			
Recurring Benefits			
	Relative Benefits (+3 to −3)	Mgmt Weights (0 to 1.0)	
(a) Productivity gains			
- End user satisfaction	3	1.0	(adhoc queries)
- Programmer productivity	2	1.0	(SQL easier to write/modify)
- Database designer productivity	1	0.5	(relational DB easier to design
- Tech. support productivity	0	0.5	
(b) Administrative gains			
- Ease of administration	−2	0.5	(lack of tools)
- Corporate information gains	0	0.5	
- Employee turnover gains	2	0.5	
- Data security gains	0	1.0	
- Data integrity − 3	1.0		(DB2 integrity is not sound)
(c) Operational gains			
- Reliability gains	0	1.0	(no gains)
- Backup/recovery	0	1.0	
- Concurrency	0	0.5	
- Performance gains/losses	−1	1.0	(DB2 slower)
Total recurring benefit = Σ relative benefit x management weights = 1.5 (net gain)			

(i.e. if information from one table is deleted, then the related information from other tables will need to be deleted manually). Another potential integrity problem is that of table update when multiple table joins are performed in one SELECT statement. The vendor implemen-

tations to handle this issue are not satisfactory and tools to manage over 15,000 tables in DB2 are not available.

The performance of the application with DB2 is considerably degraded, mainly because almost twice as many LIO are needed in DB2, compared to IMS. For example, consider the payroll database. In order to list all the information for each employee in the database, IMS would require $1 + 9n + 2nm$ LIO where n is the average number of dependent segments per parent at level 2 and m is the average number of depenedent segments at level 3. DB2 will require a join between all the 13 tables and even in the best case of indexes for all 13 tables, the total number of LIO will be $2 \times (1 + 9n + 2nm)$, because, for each join, the index as well as the table will need to be accessed. In practice, it has also been found that DB2 almost doubles the CPU utilization for the same application.

The total tangible cost of conversion is estimated at about $330,000 and the intangible recurring gain is only 1.5 points. An attempt to convert this intangible gain into tangible gain has not been very successful although managers and users agreed that the expected benefits are low. Based on these initial analysis, the conversion from IMS to DB2 has been defered though a pilot project is continuing. From a management point of view, the problems of managing very large number of tables in DB2 without proper integrity controls and the difficulty in justifying a new processor for DB2 are the major hurdles.

4.3. Conversion Considerations: Distributed DBMS

Another group has studied the conversion of the centralized applications to distributed applications. The objective of the distribution effort is to offload the central site workload by distributing data at three levels: local workstations (IBM PCs), regional processors (8100 or equivalent) and central processor (IBM 3080).

Due to the commercial availability of DDBMS for relational data models [5,13,14,17], the decision of distribution is viewed as a squential decision; first IMS needs to be converted to DB2 and then DB2 needs to be distributed. Table 6 gives the values for the parameters in the evaluation model for conversion from centralized DB2 to a distributed DB2.

The following are the major observations:

- The one-time tangible conversion cost is much lower for DB2 to distributed DB2 than for MS to DB2, mainly because the software does not need to be converted and major retraining is not needed.
- Due to the regional nature of the computing services, the data distribution decisions are straightforward and there is no need for optimal data allocation. Basically, the databases for region x can be allocated to region x processor, and the mainframe can be used as a backup system. Thus the database redesign cost is also low.
- Due to the legal responsibility of the regional center, a current copy of data at the host must be kept and synchronized for updates and concurrency control [2,21].
- Several regional sites do not have trained staff to manage their own computing centers. Thus some regional centers resist this change. (It has been reported that management of a local area network (LAN) is much harder than management of a collection of terminals connected to remotely located hosts [11].
- Several central site staff members are not in favour of distribution because they are apprehensive that their jobs may diminish or even disappear with distribution. For example, the Operations Department at the central site runs three shifts with three operators per shift. Movement of data from the central site to regional processors means loss of processing requirement at host and thus a reduction of personnel in the Operations Department.

From this analysis, the primary advantage of distributed applications is that the performance can be significantly improved, thus the performance step becomes an important tool.

Based on a workload of 20,000 transactions per day, the following processing requirements can be estimated by using equitions (1) and (2) for the exsiting centrally controlled system where indicates the central site: TL(i) is 10,000,000 per day at i = c, 0 at all other nodes and is 300 per second (approximately), assuming 10 hour per day; $C(i, j)$ is 400,000 messages per day between i = c

Table 6
Business Application Conversion: Centralized to Distributed

Activities	Cost Benefit		Comments
One-Time Costs			
(a) Training costs			
– Designer training	$10K		2 designers
– Programmer training	$0K		5 programmers no training
– Technical support training	$18k		2 tech. support
– Administrator training	$4K		2 administrator
– Operator training	$6K		3 operators
– End-user training	$0K		no training
(b) Redesign / reconstruction			
– Database redesign	$20K		50 databases
– Software redesign	$0K		120 programs (no reprogramming
– Database repopulation	$5K		(10.000 tables)
(c) Hardware / software upgrade			
DBMS replacement	$60K		(DDBMS cost at $5K per month)
– Hardware cost	$50K		(need more PCs)
– Operating system upgrade	0		(not needed)
(d) Administrative costs			
– Planning costs	$20K		(2 man month)
– Procedures conversion costs	$10K		(1 man month)
– Documentation costs	$50K		(5 man months)
Total one time dollar cost = $223.000			
Recurring Benefits	Relative Benefits (+3 to −3)	Mgmt Weights (0 to 1)	
(a) Productivity gains			
– End user satisfaction	3	1.0	(better access)
– Programmer productivity	0	1.0	
– Database designer productivity	−2	0.5	(task more complex)
– Tech. support productivity	−1	0.5	(task more complex)
(b) Adminstrative gains			
– Ease of administration	−2	0.5	(lack of tools)
– Corporate information gains	1	0.5	(more information)
– Employee turnover gains	0	0.5	(employees anxious to learn DDBMS)
– Data security gains	−1	1.0	
– Data integrity	−1	1.0	(DDBMS integrity control is weak)
(c) Operational gains			
– Reliability/availability	2	1.0	(more sites)
– Backup/recovery	1	1.0	(more sites)
– Concurrency	1	0.5	(improved)
– Performance gains/losses	3	1.0	(DDBMS faster)
Total recurring benefit = Σ relative cost x management weights = 5.5 (net gain)			

and j is any other node, 0 for all other values of i and j. $C(i, j) = 400{,}000 \times 80 \times 10/10 \times 3600 =$ 8000 bits per sec assuming 80 byte messages and 10 bits per second.

The node service time needed to handle this workload yields about 600 LIO per second at the central site for $U(i) = 0.6$ from equations (4) and (5). The management guideline was not to exceed

1 million LIO at the central node. Thus 8 million LIO were partitioned and distributed to the 30 regional minis which was within the allowed increased mini workload.

4.4. Management Decision

At present, the management is committed to distributing applications in the next two years for workload distribution, despite some objections raised by internal staff and regional centers. The overall plan of the management is (i) choose an application for conversion (ii) convert the chosen application from IMS to DB2, (iii) run the application at the host for one year and at the same time train staff for distributed operations, (iv) convert the remaining applications to distributed operations, repeating steps (i), (ii), and (iii).

5. Case Study: Conversion of Network Management Application

The second case study is for a network management application that is being developed to manage the installation, administration and operation of large networks. Network management is currently receiving great attention (see for example, the IEEE Network Special Issue, March 1988). One definition, based on the 150 standards community, is that network management is concerned with managing the following activities:

- Fault Management – detecting, diagnosing and recovering from network faults.
- Configuration Management – defining, changing, monitoring, and controlling network resources and data.
- Accounting – recording usage of network resources and generating billing information.
- Performance Analysis – monitoring current and long term performance of the network.
- Security – ensuring only authorized access to network resources.
- Resource Management/User Directory – supporting directories for managing network assests and user instructions.

Network management is a challenging application that, by definition, requires implementation on a distributed network, high performance, ability to change rapidly, has real-time requirements and depends on a distributed database. Examples of commercially available network management systems are IBM's Netview [23] and Digital's Network Management System [30].

5.1. Current Implementation of Network Management

Fig. 2 shows a conceptual architecture of a Network Manager. The case study system currently utilizes an SQL relational database system to provide the basic database capabilities. Approximately 200 tables are accessed from almost 100 different procedures, and each procedure issues between 5 to 20 different SQL calls. There is a permanent and temporary copy of each table in the system; the permanent table keeps the committed changes to the system while the temporary table contains uncommitted changes. The current implementation uses a central node exclusively.

A main concern with the current implementation is that the application was developed by geographically distributed groups and is being continually modified to produce improved releases and versions. Communications between various managers and development groups involves questions such as the following:

- which tables are referenced by which procedures?
- how many and which procedures will need to be modified if a table is modified?
- if one table is modified what other tables will need to be modified?
- how many multi-table operations (joins, unions) need to be performed?
- how many and which tables are never referenced alone?
- what are the potential performance bottlenecks?
- what are the best case/worst case bounds on performance?

It has been difficult to answer these questions without manually cataloging all of the procedures and the database calls issued by the procedures because the current source is not under adequate configuration control.

Another major concern is the performance of the current implementation when the number of users increases and the size of the tables grows.

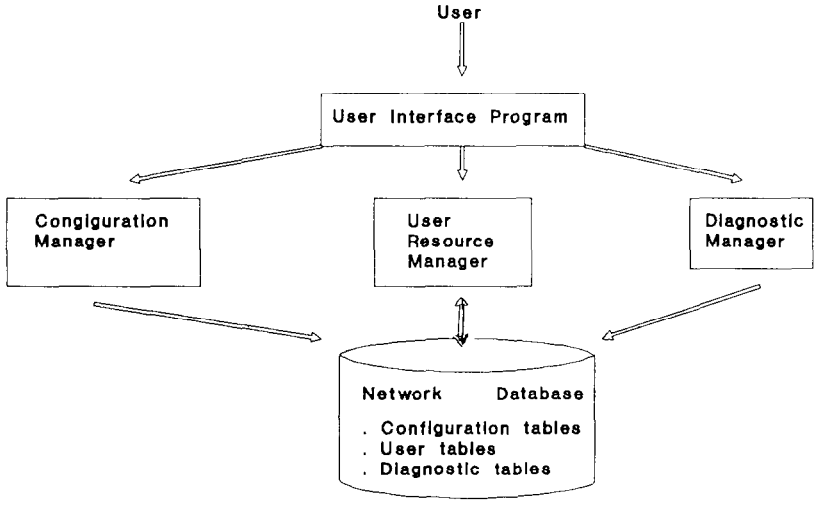


Fig. 2. A Network Manager.

For example, it was found that many SQL queries were very complex; several queries requiring joins between 7 to 8 tables. Assuming an average table size of 10,000 tuples, more than 100 million LIO would need to be executed in one working day. If the working day is 10 hours, more than 3000 LIO per second will need to be executed; this is well beyond the capacity of current hardware capability. The calibration performed earlier indicated capabilities of 10 to 100 LIO per second.

Other concerns about the existing implementation include data integrity, i.e. the referential integrity and the fact that the data retrieval by value only of relational systems may cause performance problems. Due to the above difficulties with the current relational implementation, the application is being evaluated for conversion to other DBMS.

5.2. Conversions to other DBMS

A short feasibility study has been conducted to consider the convertion of this application to a network database management system rather than a relational one. The main advantage of network database management systems is that the relationships between entities can be easily modeled and the physical pointers provide better performance than relational joins. In addition, the concerns of integrity can be addressed because in network DBMS the referential integrity can be enforced. However, a decision was made not to pursue this option, mainly because the network DBMS is considered a thing of the past and technical staff does not show any interest in it. Instead, the object oriented databases were considered next. The main expected benefit of object oriented DBMS is that a network can be easily defined by using class hierarchies and the subnet definition can especially benefit from these hierarchies. However, it was decided that the application cannot be converted to object oriented databases because proven commercial database systems are not available.

The principle emphasis at present is on distributing this relational application to several nodes. Table 7 shows an analysis of the conversion decision. The results shown in this table are very similar to those in Table 6 because both tables show conversions of relational databases from centralized to distributed. The main difference is in the performance analysis.

Based on a 30,000 transactions per day, the following workload is estimated. $TL(i)$ = 80,000,000 per day at i = c, 0 at all other sites (from equation 1) = 2000 per second (assumimg 10 hour work day). $C(i, j)$ = 700,000 messages per day between i = c and j is any workstation (from equation 2) = 700,000 × 80 × 10/10 × 3600 = 16,000 bits per sec.

Table 7
Network Application Conversion: Centralized to Distributed

Activities	Cost Benefit		Comments
One-Time Costs			
(a) Training costs			
- Designer training	$15K		3 designers
- Programmer training	$0K		10 programmers no training
- Technical support training	$18k		2 tech. support
- Administrator training	$2K		1 Administrator
- Operator training	0		0 operators
- end-user training	0		no training
(b) Redesign/reconstruction			
- Database redesign	$10K		200 databases
- Software redesign	$0K		100 programs (no reprogramming
- Database repopulation	$5K		(200 tables)
(c) Hardware/software upgrade			
- DBMS replacement	$36K		(DDBMS cost © $3K
- Hardware upgrade	0		
- Operating system upgrade	0		(not needed)
(d) administrative costs			
- Planning costs	0		
- Procedures conversion costs	0		
- Documentation costs	0		
Total one time dollar cost = $86,000			
Recurring Benefits			
	Relative Benefits (+3 to −3)	Mgmt Weights (0 to 1)	
(a) Productivity gains			
- End user satisfaction	3	1.0	(better access)
- Programmer productivity	0	1.0	
- Database designer productivity	−2	0.5	(task more complex)
- Tech. support productivity	−1	0.5	(task more complex)
(b) adminstrative gains			
- Ease of administration	−2	0.5	(lack of tools)
- Corporate information gains	1	0.5	(more information)
- Employee turnover gains	0	0.5	(employees anxious to learn DDBMS)
- Data security gains	−1	1.0	
- Data integrity	−1	1.0	(DDBMS integrity control is weak) control is weak)
(c) Operational gains			
- Reliability/availability	2	1.0	(more sites)
- Backup/recovery	1	1.0	(more sites)
- Concurrency	1	0.5	(improved)
- Performance gains/losses	3	1.0	(DDBMS faster)
Total recurring benefit = Σ relative cost x management weights = 5.5 (net gain)			

5.3. Management Decision

Workload analysis shows that the host workload will have to be reduced by a factor of almost 40. This is difficult to do because the central site in this implementation maintains data elements essential for network management. To achieve the desired workload reduction, the following multi-

level workload distribution is being planned:

- divide the network into subnets where each subnet has its own centralized manager,
- allocate the proper files to the subnet managers so that the number of LIO at each node do not exceed 10 per second,

This method is similar to the one described by [6]. The solution for an 80 million LIO per day apears to be to equally subdivide this workload into 40 subnets where each subnet consists of about 10 nodes. As mentioned earlier, the subnet workload can now be subdivided among the 10 nodes of each subnet.

6. Summary and Conclusions

This paper has focussed on the pragmatic aspects of converting database applications. The basic premise is that datbase application conversion is a far more complex process than program conversions; it involves a large number of technical, management, human and staffing factors and is not motivated by the appeal of database models. Conclusions reached from study of several large scale database conversions are:

- Organizations are motivated to convert applications from one DBMS to another, mainly for perceived advanatges in long range growth; the appeal of particular database models appears to play a very minor role.
- The attractiveness of technical advantages are less significant when adminstrative factors are considered. For example, conversions from hierarchical/ network DBMS to relational DBMS increases the number of databases by a factor of ten or more; thus creating serious data management problems.
- Many human problems in conversion from centralized to distributed DBMS can lead to staff reassignments and reclassifications and cause personnel problems.
- The notion of "if it is'nt broke, don't fix it" prevails commonly among technical staff especially while converting from centralized to distributed applications.
- Many programmers and managers are under the false impression that in a relational DBMS a database design can be changed easily without affecting applications.
- The impression that relational systems queries are very simple and that experienced programmers are not needed to write and modify queries is misleading, because the SQL queries in practical cases tend to be very difficult to understand.
- A great many management and operational tools have been built around mature DBMS like IMS. Tools of this nature are not yet availble for newer DBMS such as DB2.
- Research is needed in building support environments which would partially automate the conversion process and thus reduce the cost and effort in conversions. Considerable work is also needed in multidatabase interoperability issues [22].

References

[1] T. Andrews and C. Harris, "Combining Language and Database Advances in an Object-Oriented Development Environment", OOPSLA'87, Oct. 1987.

[2] P.A. Bernstein and N. Goodman, "Concurrency Control in Distributed Database Systems", ACM Computing Surveys, Vol. 13, No. 2, June 1981, pp.185–222.

[3] P.A. Bernstein, V. Hadzilacos, and N. Goodman, "Concurrency Control and Recovery in Database Systems", Addison Wesley, 1987.

[4] L. Bic and J. Gilbert, "Learning From AI: New Trends in Database Technology", IEEE Computer, March 1986, pp. 44–55.

[5] A.F. Cardenas, "Heterogeneous Distributed Database Management: the HD-DBMS", Proceedings of the IEEE, May 1987, pp. 588–600.

[6] S. Ceri and G. Pellagitti, "Distributed Databases: Principles and Systems", McGraw Hill, 1984.

[7] D. Cohen, J.E. Holcomb, and M.B. Suryanarayana, "Distributed Database Management to Support Network Services", IEEE Conference on Communications, 1982.

[8] DBTG Task Group, Technical Report, 1971.

[9] Digital Consulting, Inc. Seminars, May, 1988.

[10] A.A. Farrag and M.T. Ozsu, "Towards a General Concurrency Control Algorithm for Database Systems", IEEE Trans. on Software Engineering, Vol. SE-13, No.10, Oct. 1987, pp. 1073–1073.

[11] H. Fersko-Weiss, "Who Manages the Network", Personnel Computing, March 1987, pp. 107–115.

[12] M. Gillenson, "The Duality of Database Structures and and Design Techniques", CACM, Dec. 1987, pp. 1056–1065.

[13] G. Graham, "Real-World Distributed Databases", Unix Review, May 1987.

[14] N. Goodman, "Interview with Phil Bernstein", Unix Review, May 1987.

[15] P. Heidelberger and M. Lakshmi, "A Performance Comparison of Multimicro and Mainframe Database Archi-

tecures", IEEE Trans. on Software Engineering, April 1988, pp. 522–532.
[16] IBM DB2 Announcement, April, 1988.
[17] W. Rauch-Hinden, "True Distributed DBMSes Presage Big Dividends", Mini-Micro Systems, May and June, 1987.
[18] IBM Announcement, "System Application Architecture", April, 1987.
[19] C. Kellog, "From Data Management to Knowledge Management", IEEE Computer, January 198, pp. 75–84.
[20] M.A. Ketabchi V. Berzins, "Modeling and Managinging CAD Databases", IEEE Computer, Feb. 1987, 93–100.
[21] W.H. Kohler, "A Survey of Techniques for Synchronization and Recovery in Decentralized Computer Systems", ACM Computing Surveys, Vol. 13, No. 2, June 1981, pp. 149–184.
[22] W. Litwin and A. Abdellatif, "Multidatabase Interoperability", IEEE Computer, Dec. 1986, pp. 10–18.
[23] Netview: Concepts and Facilities, IBM Document, 1988.
[24] W.D. Potter and R.P. Trueblood, "Traditional, Semantic, and Hyper-Semantic Approaches to Data Modeling", IEEE Computer, June 1988, pp. 53–64.
[25] R.S. Pressman, "Software Engineering – A Practitioner's Approach", McGraw Hill, 1987.
[26] H. Raiffa, "Decision Analysis – Introductory Lrctures on Choices Under Uncertainty", Addison Wesley, 1970.
[27] J. Rumbaugh, "Relational Database Design Using an Object-Oriented Methodology" CACM, April, 1988, pp. 414–427.
[28] A. Rushinek and S. Rushinek, "End-User Satisfaction of Data Base Management Systems", Data Base, Winter, 1986, pp. 17–26.
[29] Society of Manufacturing Engineering, Professional Development Seminars, January 1989.
[30] M. Sylor, "Managing Phase V DECnet Networks: The Entity Model" IEEE Network, March 1988, pp. 30–36.
[31] T.J. Teorey and J.P. Fry, "Design of Database Structures", Prentice Hall, 1982.
[32] J. Verner and T. Tate, "Estimating Size and Effort in Fourth Generation Development", IEEE Software, July 1988.
[33] G. Wiederhold, "Views, Objects and Databases", IEEE Computer, Dec. 1986, pp. 37–44.
[34] O. Wolfson, "The Overhead of Locking (and Commit) Protocols in Distributed Databases", ACM Trans. on Database Systems, Vol. 12, No. 3, Sep. 1987, pp. 453–471.