Automatic database system conversion: schema revision, data translation, and source-to-source program transformation

by BEN SHNEIDERMAN
University of Maryland
College Park, MD
and
GLENN THOMAS
Kent State University
Kent, OH

ABSTRACT

Changing data requirements present database administrators with a difficult problem: the revision of the schema, the translation of the stored database, and the conversion of the numerous application programs. This paper describes an automatic database system conversion facility which provides one approach for coping with this problem. The Pure Definition Language and the Pure Manipulation Language have been designed to facilitate the conversions specified in the Pure Transformation Language. Two conversions and their effect on retrievals are demonstrated.

INTRODUCTION

Contemporary database management systems isolate the users from changing physical implementation strategies, but offer little assistance when logical structures must be modified. Several research projects have been directed at automating part or all of the database system conversion process. A complete strategy for coping with requirement changes would have to aid in the revision of the schema, translation of the stored database, and conversion of the application programs.

Data translation research at the University of Michigan has begun to include work on database conversion (Navathe and Fry, 1976,[19] Swarthwout, Deppe and Fry, 1977[28]), by classifying possible schema transformations for a network structured database and by specifying architectures for a conversion system.

The IBM Research group at San Jose, which developed the EXPRESS system (Shu et al., 1977[23]) for data translation, recognized that this powerful system was a natural basis for developing program conversion aids. Housel's paper (1977)[16] showed how CONVERT operations could be used as a query language as well as for describing schema transformations. He demonstrated a set of rules which enabled schema transformations described in CONVERT to be applied to CONVERT queries.

At the University of Florida, CONVERT transformations were applied to relational schemas and SEQUEL queries (Su and Liu, 1977,[26] Su and Reynolds, 1977,[27] and Su, 1976[24]). Dale and Dale (1976, 1977[13,14]) at the University of Texas studied program preserving transformations for the tree structured data model. Gerritsen and Morgan (1976)[15] dealt with a class of network schema transformations by dynamically translating program statements to match the revised schema. Navathe (1980)[18] examined transformations on schema diagrams which were related to the entity-relationship model and Sakai (1980)[20] suggested some transformations during logical design in the relational model. Shneiderman (1978)[22] provided a framework for research in network schema transformations and Taylor et al. (1979)[29] identified problem areas and offered several directions for research. Jacobs (1980)[17] described automatic conversion in the context of his database logic which provides a formal mathematical foundation for database systems.

AUTOMATIC CONVERSION IN THE PURE DATABASE SYSTEM

The Pure Database System was designed to facilitate schema changes which call for database translation and application program conversion. For example, changing a two-level schema structure, such as division records owning employee records, to a three-level structure, where division records own department records which in turn own employee records, generally requires special purpose translation programs to restructure the database and hand-written revisions to modify application programs. Using Pure Transformation Language (PTL) operators, database administrators can specify transformations which automatically generate a new schema, stored database, and application programs. The execution of the target application programs operating on the target database should produce output identical to that produced by the source application programs operating on the source database.

Our Pure Definition Language (PDL) and Pure Manipulation Language (PML) blended elegant high-level relational ideas with lower-level network concepts to produce a data model conducive to automatic transformation. Our goals were to ensure input/output equivalence where possible, minimize host language interactions, provide integrity assurance for the transformations, offer useful and effective definition and manipulation languages, and construct a convenient set of transformations. Although more efficient transformations are feasible, we felt that generality, modularity, simplicity, provability, and integrity assurance were more important criteria. The effectiveness of the PDL and PML and the utility of the PTL must be verified through field testing or controlled experiments with manual alternatives.

Refinements, extensions, and alternatives are easy to generate, but we felt the need to limit our focus and demonstrate a complete workable system. The PDL and the PML have been implemented using the XPL compiler-compiler to generate UNIVAC DMS-1100 code which is then executed through normal procedures. The PTL processor is more complex since it requires the preparation of a new schema, the generation of programs to restructure the database, and the revision of possibly hundreds of application programs embedded in host language code. Furthermore, before a transformation is carried out, the stored database must be examined to ensure that integrity constraints are satisfied and the application programs must be parsed to verify that the transformation is possible.

Although great care and effort was devoted to constructing a set of transformations at an appropriate level, we recognize alternative approaches. Higher level transformations (Su and Lam, 1979[25]) would capture more "semantic" constructs, but a greater number of transformations might be needed to accommodate database administrator (DBA) needs. Lower level transformations might be easier to implement and prove correct, but would be complicated to use. Feedback from users and experience seems essential to help choose the most convenient approach.

PURE DEFINITIONS AND MANIPULATIONS

Like the CODASYL DBTG approach, the Pure schema has a collection of record types and set types. There is a singular record type, SYSTEM, which is the starting point for all searches. Each record type may be the owner and member of several set types but the schema graph must be acyclic. Within a set instance the record instances are in ascending order by the set keys. The simple schema shown in Figure 1a might be defined by the following Pure Definition statements:

```
SCHEMA NAME IS students.
RECORD SECTION.
    RECORD NAME IS stu.
    FIELDS ARE.
        sno         PIC 9(6).
        sname       PIC X(25).
    END RECORD.

    RECORD NAME IS crs.
    FIELDS ARE.
        cno         PIC 9(4).
        title       PIC X(60).
        grade       PIC X(3).
    END RECORD.

END RECORD SECTION.

SET SECTION.
    SET NAME IS sys-stu.
    OWNER ISE SYSTEM.
    MEMBER IS stu.
    SET KEY IS (sno).
    END SET.

    SET NAME IS stu-crs.
    OWNER IS stu.
    MEMBER IS crs.
    SET KEY IS (cno).
    END SET.

END SET SECTION.
END SCHEMA.
```

In this schema student records (stu) are in ascending order by student number (sno) and contain the student name (sname). Each student record owns a set of course records (crs) which are kept in ascending order by course number (cno) and contain a course title and grade.

The current design for the Pure Manipulation Language assumes embedding in a host language such as COBOL. The FIND statement specifies a search through the database and the creation of a train (an ordered collection of record identifiers, each of which uniquely specifies a database record) satisfying an access path expression. For example, to find the course records in which a student named 'JOE' received a grade of 'A' we might write:

```
FIND (crs: SYSTEM, sys-stu, stu(sname = 'JOE'), stu-crs,
        crs (grade = 'A')).
```

The target record type, crs, must be somewhere along the path expression which follows the colon. The path expression starts at the SYSTEM record and follows sets and records through the database. Records may have boolean expressions to qualify field names within a record and the path expression may traverse sets in forward and reverse order.

The GET statement retrieves a single record of a train specified by its numeric position in the train, and places the record in a buffer area associated with the record type. By embedding the GET statement in a loop all the records in a train can be retrieved. The STORE statement inserts a record in the database and ensures that the record will properly participate in all sets in which it is an owner or member. The DELETE statement can be used to delete a single record or a train of records from the database. Deletion can only be made if the database structure is preserved and if all records would be reachable after the deletion. Three forms of the MODIFY statement have been included: replacement of non-key field values in a record, alternation of key fields which effect set order only, and alteration of key fields which effect set membership. Improperly or incompletely specified modifications are not carried out.

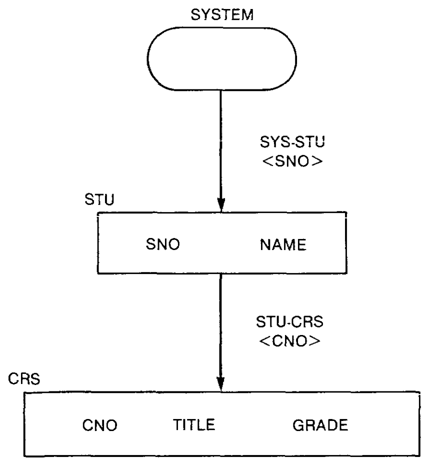
In summary, the Pure System blends the appeal of schema traversal using path expressions with the high level relational operations on collections of records. The network concepts of set ordering and explicit linkage have been combined with the relational notions of keys and tuple uniqueness.

PURE TRANSFORMATIONS

The 18 Pure transformations presented in Table I permit conversion of a two-level schema to a three-level schema, factoring of common fields from member to owner records, distribution of fields from owner to member records, manipulation of set key fields, introduction (and elimination) of sets, records and fields, and name changes. We first offer a general three dimensional categorization for transformations before informally presenting the Pure Transformation Language.

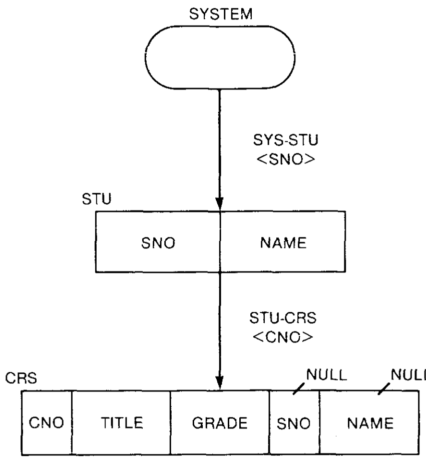
The schema of Figure 1a allows queries of the form: "What grade did student X receive in course Y?" A possible transformation would be to remove the field grade from the record type crs. This transformation is not information preserving because the new (target) schema does not contain all the information derivable from the old (source) schema. More generally a transformation is *information preserving* if all the information derivable from the source schema is derivable from the target schema.

The second categorization dimension of *data dependence* may be illustrated by a user requirements change. Assume, DEPT records may own EMPLOYEE records which contain a field MGR identifying the employee's manager. A corporate policy change may require that all employees within a department be managed by the same individual. In this case, it is reasonable to move the MGR field to the DEPT record type. The FACTOR and ERASE transformation copies a field value from the members of a set to their common owner. Before the transformation may be allowed, each occurrence of the set type linking DEPT and EMPLOYEE occurrences must be examined to determine whether or not the source



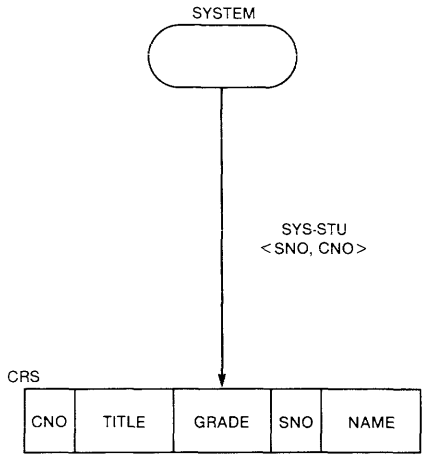
(a)

Add Field
  SNO PIC 9(6)
 To Record CRS.
Add Field
  NAME PIC X(20)
 To Record CRS.



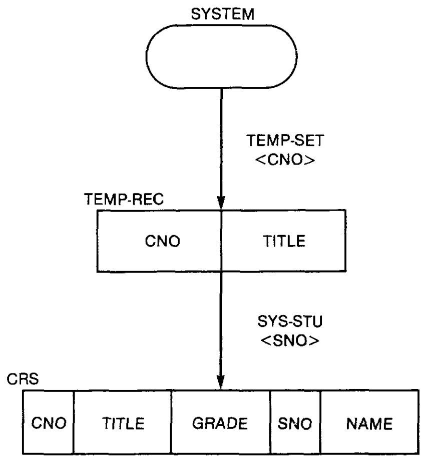
(b)

Distribute and Erase Fields NAME
 To Members of Set STU-CRS.
Separate From Between
 Record SYSTEM and Set STU-CRS
 Set SYS-STU
 And Record STU
 With Source SNO of CRS = SNO of STU.



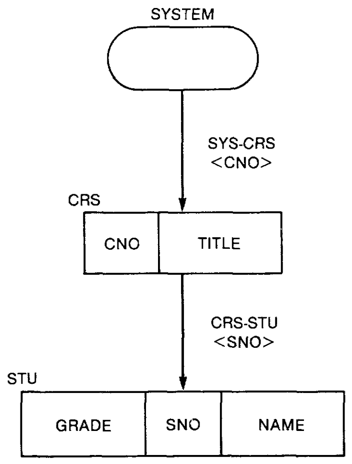
(c)

Permute Key of Set SYS-STU
 From (SNO, CNO) to (CNO, SNO).
Introduce Between Record SYSTEM and Set SYS-STU
Record
 Record NAME is TEMP-REC.
 Fields are.
  CNO PIC X(4).
  TITLE PIC X(30).
 End Record.
And Set
 Set NAME is TEMP-SET.
 Owner Record is SYSTEM.
 Member Record is TEMP-REC.
 Set Key is (CNO).
 End Set.
With Source CNO of TEMP-REC = CNO of CRS.
Factor and Erase Fields TITLE
 From Members of Set SYS-STU.



(d)

Remove Fields CNO, TITLE From Record CRS.
Change NAME of Record
 From CRS to STU.
Change NAME of Set
 From SYS-STU to CRS-STU
Change Name of Record
 From TEMP-REC to CRS.
Change NAME of Set
 From TEMP-SET to SYS-CRS.



(e)

Figure 1—Inversion of the student and course relationship

TABLE I—Categorization of pure transformations

	Information preserving		Not information preserving	
	Data independent	Data dependent	Data independent	Data dependent
Program independent	Change name Add field Permute Distribute Fields Distribute and erase fields Introduce set record Introduce between	Distribute set key Distribute and erase set key fields Introduce where		Detach Factor fields Factor and erase fields
Program dependent	Append		Remove fields Separate set Separate set record Separate from between	

stored database satisfies this corporate policy. Should any occurrence of the set type linking DEPT and EMPLOYEE have two or more member record occurrences with different values for the field MGR, then the transformation fails. Transformations are *data dependent* if the source stored database must be examined to determine whether or not current data values satisfy changing user requirements. Otherwise, a transformation is *data independent.*

The final categorization dimension is *program independence.* Should the field sname be removed from the Figure 1a record type stu, any source program referencing this field will have to be examined to determine whether it can be modified to run under the target schema or dropped from the set of programs accessing the stored database. While the conversion system can isolate such program dependencies, the database administrator is responsible for deciding on the correct action to be taken. Logical schema change. A transformation is *program dependent* if the source programs must be examined to determine whether or not they can be modified to run under the target database system. Otherwise a transformation is *program independent.*

CHANGE NAME, ADD FIELD, and REMOVE FIELDS allow the database administrator to change the name of any source set, record or field, add a new field to an existing record type, or remove a field from the definition of a record type. Of these, only REMOVE FIELDS requires program examination. The sequence:

```
ADD FIELD f TO RECORD r.
REMOVE FIELDS f FROM RECORD r.
```

yields a target database system that is identical to the source database system. Hence, REMOVE FIELDS is the inverse of ADD FIELD. However, the reverse is not true because REMOVE FIELDS destroys data values in the source database.

The transformations APPEND, PERMUTE and DETACH allow the database administrator to redefine the key of an existing set and logically reorder member record occurrences. APPEND adds a field as the least significant component of a set key. While information preserving and data independent, APPEND requires DBA interaction to modify storage paths involving the member record type. DETACH removes the least singificant set key field for some set. Each occurrence of the affected set must be examined to determine whether or not the resulting set key will uniquely identify the members. If it will not, DBA interaction is required to obtain the necessary uniqueness. APPEND and DETACH are inverses for each other when allowed. As illustrated by Figures 1c and 1d, PERMUTE redefines the left-to-right order of concatenation of set key fields. In addition to being information preserving, data independent, and program independent, PERMUTE is the only Pure transformation that is its own inverse.

The six DISTRIBUTE and FACTOR transformations allow for the copying of field values from owner to member records (DISTRIBUTE) or vice versa (FACTOR). For FACTOR, this requires examining the source stored database to determine whether or not all members of each set occurrence share a common value for the field(s) being copied. When ERASE is specified, the field is set to the 'null' value after it has been copied. FACTOR and DISTRIBUTE are mutual inverses.

INTRODUCE SET RECORD adds a record type and a set type owning this record type to the schema. This transformation has no effect of the stored database or the set of programs as no instances of the new record type exist. SEPARATE SET RECORD is the inverse for INTRODUCE SET RECORD. When specified, all occurrences of the named record and set types are removed from the source database system. Because data values are lost, this is not an information preserving transformation. Thus, it has no inverse.

INTRODUCE WHERE allows the definition of a new set type between two existing record types. The WHERE clause specifies a selection criterion to associate every member record occurrence with exactly one owner record occurrence. These are then made members of a set occurrence of the new set type whose owner is selected by the WHERE clause criterion. SEPARATE SET is the inverse of INTRODUCE WHERE. Because this eliminates a path from the schema and destroys the information contained in this set type, this is not an information preserving transformation.

The final pair INTRODUCE BETWEEN and SEPARATE

FROM BETWEEN allow for transformations of the form illustrated by the schemata of Figures 1c and 1d where a new record and set type are introduced between an existing record and set type. A succession of INTRODUCE BETWEEN's may be employed to create a hierarchy within the schema while SEPARATE FROM BETWEEN may be employed to remove this hierarchy.

PURE TRANSFORMATION EXAMPLE

Figures 1 and 2 provide two examples of potential applications of an automatic database conversion system. The starting database in Figures 1a and 2a shows a collection of student records organized in ascending order by student number (sno). Each student record owns a collection of course records (crs) which are organized in ascending order by course number (cno). Figures 1b through 1e show successive transformation steps to convert the source schema into a target schema where courses own students. This conversion was called inversion by Navathe and Fry (1976).[19]

The example query shown earlier, Find the course records in which a student named 'Joe' received a grade of 'A', applies to the schema in Figure 1a:

```
FIND (crs: SYSTEM, sys-stu, stu(sname = 'JOE'), stu-crs,
           crs(grade = 'A')).
```

No program transformation is required for the schema in Figure 1b. The DISTRIBUTE AND ERASE in Figure 1c requires the introduction of a boolean path expression involving the EXISTS predicate:

```
FIND (crs: SYSTEM, sys-stu,
           stu(EXISTS (stu-crs, crs(sname = 'JOE'))),
           stu-crs, crs(grade = 'A')).
```

The SEPARATE FROM BETWEEN transformation in Figure 1c allows a more compact path expression:

```
FIND (crs: SYSTEM, sys-stu, crs (sname = 'JOE' and grade
           = 'A')).
```

The INTRODUCE BETWEEN transformation in Figure 1d forces a longer path expression:

```
FIND (crs: SYSTEM, temp-set, temp-rec, sys-stu,
           crs(sname = 'JOE' AND grade = 'A')).
```

Finally, the name changes in Figure 1e induce a simple transformation to the desired target query for the new schema:

```
FIND (crs: SYSTEM, sys-crs, crs, crs-stu,
           stu(grade = 'A' AND sname = 'JOE')).
```

Figures 2b through 2d show successive transformation steps to create a many to many relationship between student and course instances. Here again the paths expressions in FIND statements can be rewritten in an orderly way so that the same retrievals can be performed on the target database. Of course, transformations to STORE, DELETE, and MODIFY statements can present somewhat greater difficulty, but we feel that where a transformation is possible, our design supports it.

Figures 1 and 2 show the effects of the transformation on a schema diagram, but the system is designed to take the actual code for the source schema and generate a target schema, to take the stored database and translate it to match the target schema, and to take the numerous application programs and convert them to run on the target schema. Of course, if information is deleted during a transformation, some of the application programs may not operate in the same way as they did before. The database administrator must decide if the results of such a conversion are acceptable. Whether the eighteen transformations we offer are convenient and provide enough power to be useful in commercial applications remains an open question.

CURRENT RESEARCH DIRECTIONS

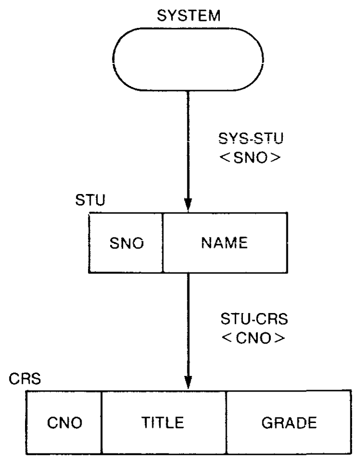
Our fundamental goal has always been to create a research system which demonstrates the feasibility of automatic database system conversion. We do not seek Pure Database System users, but rather hope that this work will inspire other designers to provide automatic database conversion facilities in their system architecture.

We are currently trying to apply the ideas in the Pure Database System to conversion in other data models and to conversion across data models. Shneiderman and Thomas (1982)[12] describe 15 transformations for the relational model of data and suggest an architecture for an automatic conversion system. Schema to sub-schema mappings can be defined with transformation operations (Thomas and Shneiderman, 1980).[4] We are also pursuing a formalization of these concepts so as to verify the correctness of a transformation, assess the range of our set of transformations, and uncover additional useful transformations.

This work is relevant to standardization efforts currently in progress because we beleive that the ease of conversion should be a consideration for all database definition and manipulation languages. Secondly, a standards planning effort would be useful to coordinate and unify the diverse proposals for transformation facilities.

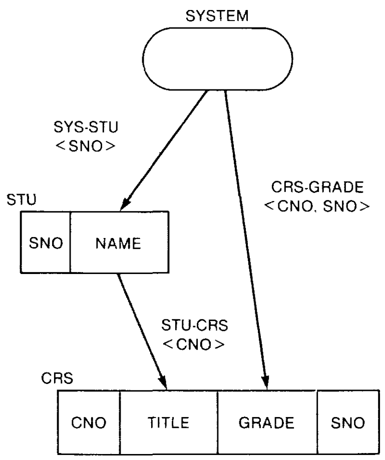
ACKNOWLEDGMENTS

Carl Fosler and Robyn Birckhead carried out the implementation of the PDL and PML. Nancy Sevitsky prepared a user's manual, Wayne Fuller assisted with the documentation, and Bonnie Zager provided administrative support. The Computer Science Center, of the University of Maryland provided some of the computer resources for this project. National Science Foundation grant MCS-77-22509 provided partial support. We appreciate the comments of our colleagues Michael L. Brodie, Barry Jacobs, Edgar H. Sibley and the members of the Database Program Conversion Task Group of the CODASYL Systems Committee, especially Stanley Su and Jim Fry.



(a)

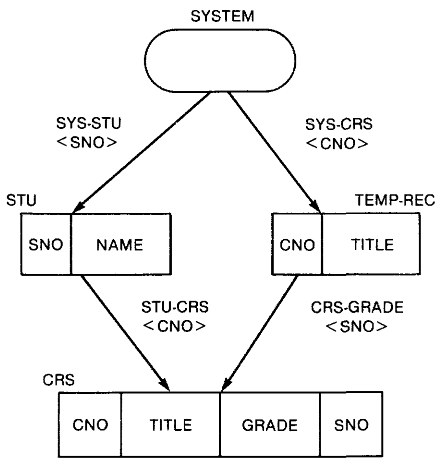
```
Add Field
  SNO PIC 9(6).
  To Record CRS.
Distribute Fields SNO
  To Members of Set STU-CRS.
Introduce Set
  Set NAME is CRS-GRADE.
  Owner Record is SYSTEM.
  Member Record is CRS.
  Set Key is (CNO, SNO).
  End Set.
  Storage Path is SYSTEM, CRS-GRADE,
    CRS (CNO = CNO-ID and SNO = SNO-ID).
```



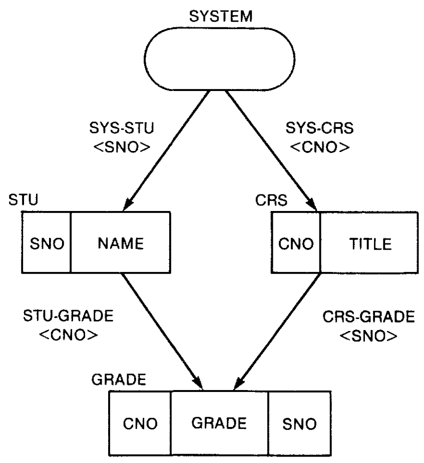
(b)

```
Introduce Between Record SYSTEM and Set
  CRS-GRADE Record
    Record NAME is TEMP-REC.
    Fields Are.
      CNO  PIC X(4).
      TITLE PIC X(30).
    End Record.
And Set
      Set NAME is SYS-CRS.
      Owner Record is SYSTEM.
      Member Record is TEMP-REC.
      Set Key is (CNO).
      End Set.
With Source CNO of TEMP-REC = CNO of CRS.
```

```
Factor and Erase Fields TITLE
  From Members of Set CRS-GRADE.
Remove Fields TITLE From Record CRS.
Change NAME of Record
  From CRS to GRADE.
Change NAME of Record
  From TEMP-REC to CRS.
Change NAME of Set
  From STU-CRS to STU-GRADE.
```



(c)



(d)

Figure 2—Transformation from a one-to-many to a many-to-many relationship

PURE SYSTEM REPORTS

1. Shneiderman, B., and G. Thomas. *Automatic Database System Conversion I: Data Definition and Manipulation Facilities.* Computer Science Technical Report Series TR-82, University of Maryland, College Park, Md., 20742 (1979), 39 pages. (Submitted for publication.)
2. Thomas, G., and B. Shneiderman. *Automatic Database System Conversion II: A Transformation Language.* Computer Science Technical Report Series TR-281, University of Maryland, College Park, Md. 20742 (1979), 46 pages. (Submitted for publication.)
3. Shneiderman, B., and G. Thomas. "Path Expressions for Complex Queries and Automatic Database Program Conversion." *Proceedings of the 6th Very Large Data Bases Conference.* Montreal (1980), pp. 33–44.
4. Thomas, G., and B. Shneiderman. "Automatic Database System Conversion: A Transformation Language Approach to Sub-Schema Implementation." *Proceedings of the IEEE COMPSAC '80 Conference,* Chicago, (1980).
5. Shneiderman, B., and G. Thomas, *Pure Database System Report: A Transformation Language Approach to Automatic Schema, Stored Data and Program Conversion.* Computer Science Technical Report Series TR-880, University of Maryland, College Park, Md. 20742 (1980), 91 pages. (Submitted for publication.)

The Pure Language components defined by this reports are separately described in:

6. Shneiderman, B., and G. Thomas. *Pure Definition Language Manual.* University of Maryland, College Park, Md. 20742 (1980), 10 pages.
7. Shneiderman, B., and G. Thomas. *Pure Manipulation Language Manual.* University of Maryland, College Park, Md. 20742 (1980), 30 pages.
8. Shneiderman, B., and G. Thomas. *Pure Transformation Language Manual,* University of Maryland, College Park, Md. 20742 (1980) 50 pages.

Other Pure System reports are:

9. Fosler, C. *Pure System XPL—DMS/1100 Implementation Documentation.* Computer Science Technical Report Series TR-872, University of Maryland, College Park, Md. 20742 (1980), 38 pages.
10. Fosler, C. *Pure PDL and PML Runstream and Examples.* University of Maryland, College Park, Md. 20742 (1980), 32 pages.
11. Sevitsky, N. *Pure User's Manual.* University of Maryland, College Park, Md. 20742 (1980), 52 pages.
12. Shneiderman, B., and G. Thomas. "An Architecture for Automatic Relational Database System Conversion." *ACM Transactions on Database Systems* (June 1982.)

REFERENCES

13. Dale, A. and N. Dale. "Main Schema—External Schema Interaction in Hierarchically Organized Data Bases." *Proc. ACM SIGMOD Conference, 1977* pp. 102–110.
14. Dale, A., and N. Dale. "Schema and Occurrence Structure Transformations in Hierarchical Systems." *Proc. ACM SIGMOD conference* (1978).
15. Gerritsen, R. and H. L. Morgan. "Dynamic Restructuring of Databases with Generation Data Structures." *ACM National Conference 1976,* pp. 281-286.
16. Housel, B. "A Unified Approach to Program and Data Conversion." *Proc. 3rd Very Large Data Bases Conference,* Tokyo (1977).
17. Jacobs, B. "Applications of Database Logic to Automatic Program Conversion." Submitted for publication.
18. Navathe, S. B. "Schema Analysis for Database Restructuring." *ACM Transactions on Database Systems, 5* (1980), pp. 157–184.
19. Navathe, S. B., and J. P. Fry. "Restructuring for Large Databases: Three Levels of Abstraction. *ACM Transactions on Database Systems* 1 (1976), pp. 138–156.
20. Sakai, H. "Entity-Relationship Approach to the Conceptual Schema Design." *Proceedings of the ACM SIGMOD Conference, 1980,* pp. 1–8.
21. Shu, N., B. Housel, R. W. Taylor, S. Ghosh, and V. Lum "EXPRESS: A Data Extraction, Processing, and Restructuring System." *ACM Transactions on Database Systems* 2, (1977) pp. 134–174.
22. Shneiderman, B. "A Framework for Automatic Conversion of Network Database Programs Under Schema Transformations." *Third Jerusalem Conference on Information Technology* (J. Moneta, ed.) Amsterdam: North-Holland, 1978.
23. Shu, N. C., B. C. Housel, and V. Y. Lum. "CONVERT: A High Level Translation Definition Language for Data Conversion." *Communications of the ACM,* 18 (1975), pp. 557–567.
24. Su, S. Y. W. "Application Program Conversion Due to Database Changes." *Proc. 2nd International Conference Very Large Data Bases,* Brussels, Belgium (September 1976). Amsterdam: North-Holland, 1976, pp. 143–158.
25. Su, S. Y. W., and H. Lam. "Transformation of Data Traversals and Operation in Application Programs to Account for Semantic Changes in Databases." Department of Computer and Information Sciences, University of Florida, Gainesville, Florida, 1979.
26. Su, S. Y. W., and B. J. Liu. "A Methodology of Application Program Analysis and Conversion Based on Database Semantics." *Proceedings of the ACM SIGMOD Conference, 1977,* pp. 75–87.
27. Su., S.Y. W., and M. J. Reynolds, "Conversion of High-Level Sublanguage Queries to Account for Database Changes." *AFIPS, Proceedings of the National Computer Conference* (Vol. 47), 1978, pp. 857–875.
28. Swarthwout, D. E., M. E. Deppe, and J. P. Fry. "Operational Software for Restructuring Network Databases." *AFIPS, Proceedings of the National Computer Conference* (Vol. 46), 1977, pp. 499–508.
29. Taylor, R. W., J. P. Fry, B. Shneiderman, D. C. P., Smith, and S. Y. W. Su, "Database Program Conversion: A Framework for Research." *Proceedings of the 5th Very Large Database Conference.* Available from ACM, New York, 1979.