*Inform. Systems* Vol. 5, pp. 137–147
© Pergamon Press Ltd., 1980. Printed in Great Britain

0306-4379/80/0601-0137/$02.00/0

# AN EXPERIMENT IN LEARNING DBTG DATABASE ADMINISTRATION

W. EFFELSBERG, T. HÄRDER and A. REUTER

Fachbereich Informatik, Technische Hochschule Darmstadt, Hochschulstr. 1, D-6100 Darmstadt, Fedrep. Germany

(*Received* 26 *September* 1979; *revised* 23 *January* 1980)

**Abstract**—An experiment in optimizing the physical storage structures of a DBTG-like database w.r.t. a given transaction load is described. It was carried out during a practical course on database administration. Logical data structures and transaction programs were kept fixed while the underlying storage structures could be varied according to a rich set of options and parameters. The preferences for certain search key and set mode options leading to surprising time differences are discussed. Properties of good optimization solutions are presented. A number of recommendations for the use of the set modes POINTER-ARRAY and CHAIN are given. Finally, some general observations concerning the learning pattern of the students are summarized.

## 1. INTRODUCTION

The objective of a practical course on database administration is to give students the opportunity to become familiar with database programming, the design of logical and physical data structures and various tasks concerning the administration of databases. One of the severe problems facing database administrators is the problem of optimizing the storage structures and tailoring the access paths of a database to meet the demands made by the mix of transactions that the database system must service.

In order to effectively train graduate students in the database field, techniques have to be developed that will enable them to explore and understand the scope and impact of their decisions. The use of realistic experiments in the design and optimization of databases provides a way for future database administrators to learn the potential consequences of unsuitable decisions and to gain a feeling for the essential parameters of the problem.

This paper decribes an interesting experiment within a practical course on database administration. The successful attendance of this course required the solution of 6 different tasks performed in groups of 2–4 graduate students (fourth year) having already studied some theoretical courses on database systems. Among other things the following tasks related to the logical data structures, to the physical structures and access paths and the mapping of these structures to physical devices had to be achieved:

—COBOL–DML programs had to be written performing retrieval and update in a complex DBTG-like database consisting of 5 record types and 10 set types[1].

—A given database with a given transaction load had to be optimized w.r.t. the storage structures requiring the application of the full set of storage structure parameters. This experiment is to be described in this paper.

—A number of DML-calls had to be traced at the physical device level given a dump of the related data pages; by page we denote the unit of data transfer between database buffer and non-volatile storage.

For these tasks a DBTG-like database system[2] was used allowing the description of the logical data structures by the so called schema-DDL and subschema-DDL[3] and the description of the physical storage structures by the so called DSDL (data storage description language[4]). This concept of separating logical and physical issues by means of different description languages provides a good, but not complete isolation of the logical data from their underlying physical structures. A separate DSDL can also be considered as a powerful optimization tool while guaranteeing a reasonable degree of data independence.

Our experiment to be described in detail is based on the isolated design of storage structures obtained by means of a separate DSDL. As a vehicle to learn database administration, it was aimed to focus on the following objectives:

—to use the additional degrees of freedom gained by a separate DSDL to optimize a given database under a given transaction load.

—to realize the optimization tradeoffs of interrelated storage structures w.r.t. retrieval and update transactions.

—to demonstrate the potential gains by suitable design decisions w.r.t. storage structure parameters.

—to show the practical meaning of data independence in database systems and to reveal the restrictions of further optimization introduced by the lack of ideal data independence.

In addition, some hints were expected concerning the learning process of students and their preference of certain optimization parameters.

## 2. DESCRIPTION OF THE DATABASE AND THE TRANSACTIONS

In designing the database to be optimized by the students attending the practical course, a number of difficulties had to be overcome:

(a) The database had to be not too complex, because the manifold dependencies between a large number of record types and sets cannot be analyzed and understood within a few weeks.

(b) The database had to be not too large, because the time for generating and loading the database with data according to the frequency distributions, and running the transactions would have been too long to be repeated several times for more than 10 groups of students.

On the other hand it must be observed that:

(c) the database must not be too simple, because a simple structure leaves no possibility for optimization,

(d) the database must not be too small, because there might be no differences between the possible storage structures, if most of the record types and access path structures can be stored in one or a few pages.

We finally decided to use a database as shown in Fig. 1. It may be considered as a small part of a university information system, containing the professors and students of all departments (faculties) and the examinations passed by the students. The schema-DDL for this database is shown in Fig. 2. The occurrences of record type FACULTY are identified by FA-NR, the occurrences of PROF by PERS-NR, the occurrences of STUDENT by REG-NR. EXAMINER and EXAMINEE are information bearing sets. Hence, the occurrences of EXAMINATION cannot be identified by themselves. The other fields of the record types are of no interest or will be explained when describing the transactions.

## 2.1 *Definition of what is to be optimized*

If one is asked to optimize a database design, it must be defined precisely, how the success of such an attempt can be measured. It is impossible to optimize a database in an absolute way. Therefore, we need a sample of transactions representing the characteristic "workload" of the database. Then the optimization target is to reduce the total execution time of these transactions in a well defined environment. For our purpose we have defined six transactions, three of them performing retrieval and the others update operations on the database:

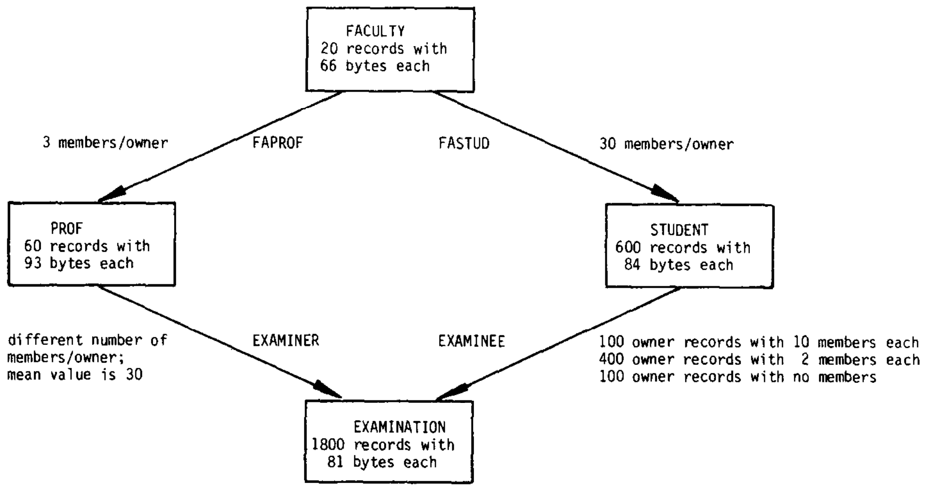

Fig. 1. Structure of the database to be optimized.

```
RECORD NAME IS FACULTY WITHIN DVP-FI1
       SEARCH KEY IS FA-NAME
       USING INDEX
       DUPLICATES ARE NOT ALLOWED
       FA-NR         PIC 9(2).
       FA-NAME       PIC X(30).
       FA-STREET     PIC X(30).
       FA-HOUSE      PIC 9(04).

RECORD NAME IS PROF    WITHIN DVP-FI1
       PERS-NR       PIC X(07).
       FA-MEMBER     PIC 9(02).
       FAM-NAME      PIC X(20).
       CHR-NAME      PIC X(20).
       INSTITUTE     PIC X(40).
       SAL-GROUP     PIC X(04).

RECORD NAME IS STUDENT WITHIN DVP-FI1
       REG-NR        PIC 9(06).
       FA-MEMBER     PIC 9(02).
       FAM-NAME      PIC X(20).
       CHR-NAME      PIC X(20).
       CURR-TERM     PIC 9(03).
       GRAD-EXP      PIC X(03).
       DOMICILE      PIC X(30).

RECORD NAME IS EXAMINATION WITHIN DVP-FI1
       SUBJECT       PIC 9(03).
       MARK          PIC 9(01).
       DATE          PIC 9(06).
       REPETITION    PIC X(01).
       REMARKS       PIC X(70).
```

```
SET NAME IS FAPROF
       ORDER IS SORTED BY DEFINED KEYS
       DUPLICATES ARE NOT ALLOWED
       OWNER IS FACULTY.
MEMBER IS PROF      MANDATORY AUTOMATIC
       ASCENDING KEY IS FAM-NAME
       SET OCCURRENCE SELECTION IS THRU CURRENT OF SET.
SET NAME IS FASTUD
    ORDER IS IMMATERIAL
    OWNER IS FACULTY.
MEMBER IS STUDENT MANDATORY AUTOMATIC
    SET OCCURRENCE SELECTION IS THRU CURRENT OF SET.
SET NAME IS EXAMINER
    ORDER IS SORTED BY DEFINED KEYS
    DUPLICATES ARE ALLOWED
    OWNER IS PROF.
MEMBER IS EXAMINATION OPTIONAL MANUAL
    ASCENDING KEY IS DATE;MARK
    SET OCCURRENCE SELECTION IS THRU CURRENT OF SET.

SET NAME IS EXAMINEE
    ORDER IS NEXT
    OWNER IS STUDENT.
MEMBER IS EXAMINATION OPTIONAL MANUAL
    SET OCCURRENCE SELECTION IS THRU CURRENT OF SET.
```

Fig. 2. DDL of the database to be optimized.

| | |
|---|---|
| Transaction 1: | Lists the FAM-NAMES of all professors examining subject 015. The instances of record type EXAMINATION contain 25 different subject codes with a uniform frequency distribution. |
| Transaction 2: | Lists the FAM-NAMES of all students having a mark 2 in a repeated examination. About 10% of the EXAMINATION records will be qualified by this predicate. |
| Transaction 3: | Lists the FAM-NAMES of all professors having examined 5 students, whose FAM-NAMES are obtained by random selection. The frequency distributions are given in Fig. 1. |
| Transaction 4: | Stores 25 new occurrences of record type STUDENT with a uniform distribution of values in field FA-MEMBER. |
| Transaction 5: | Erases 25 occurrences of record type STUDENT selected at random via REG-NR and all their members in set EXAMINEE. |
| Transaction 6: | Stores 100 new occurrences of record type EXAMINATION and connects them to their respective owners in the sets EXAMINER and EXAMINEE. |

In Fig. 3 a partial view of the database's logical structure is shown for each transaction. In each case only those paths within and between the different record types are drawn, which are possibly used or affected by the DML-statements constituting the respective transaction. As an example, let us consider transaction 5: To retrieve the STUDENT record to be erased a FIND STUDENT USING REG-NR (FIND-7) has to be executed taking advantage of an appropriate access path (SEARCH KEY) whenever possible. Erasing an occurrence of the STUDENT record type yields additionally an implicit disconnection of this occurrence from the FASTUD set and triggers the deletion of all occurrences of EXAMINATION which it is owner of, because an ERASE STUDENT ALL MEMBERS is issued by the transaction. Consequently, all these records have to be disconnected from their respective owners in the set EXAMINER.

These logical path graphes illustrating the access sequences of the different transactions are considered to be self-explanatory. We shall make use of this specific representation during the analysis of the various optimization approaches.

### 2.2 *Initial approaches to a "standard solution"*

The students could compare their results to an initial approach intended to give an example of a rather poor solution. It comprises the DDL of Fig. 2 and the DSDL as shown in Figs. 4(a and b). Figure 4(a) contains the complete DSDL according to the syntax of our database system; Fig. 4(b) shows a more illustrative graphic

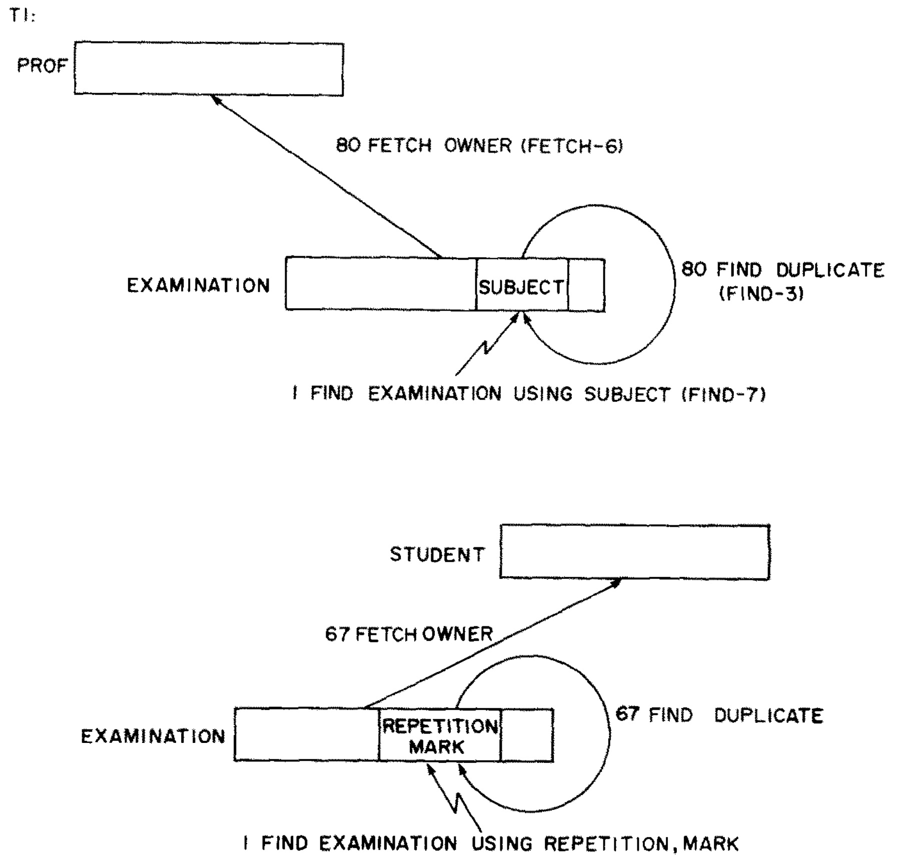

Fig. 3.

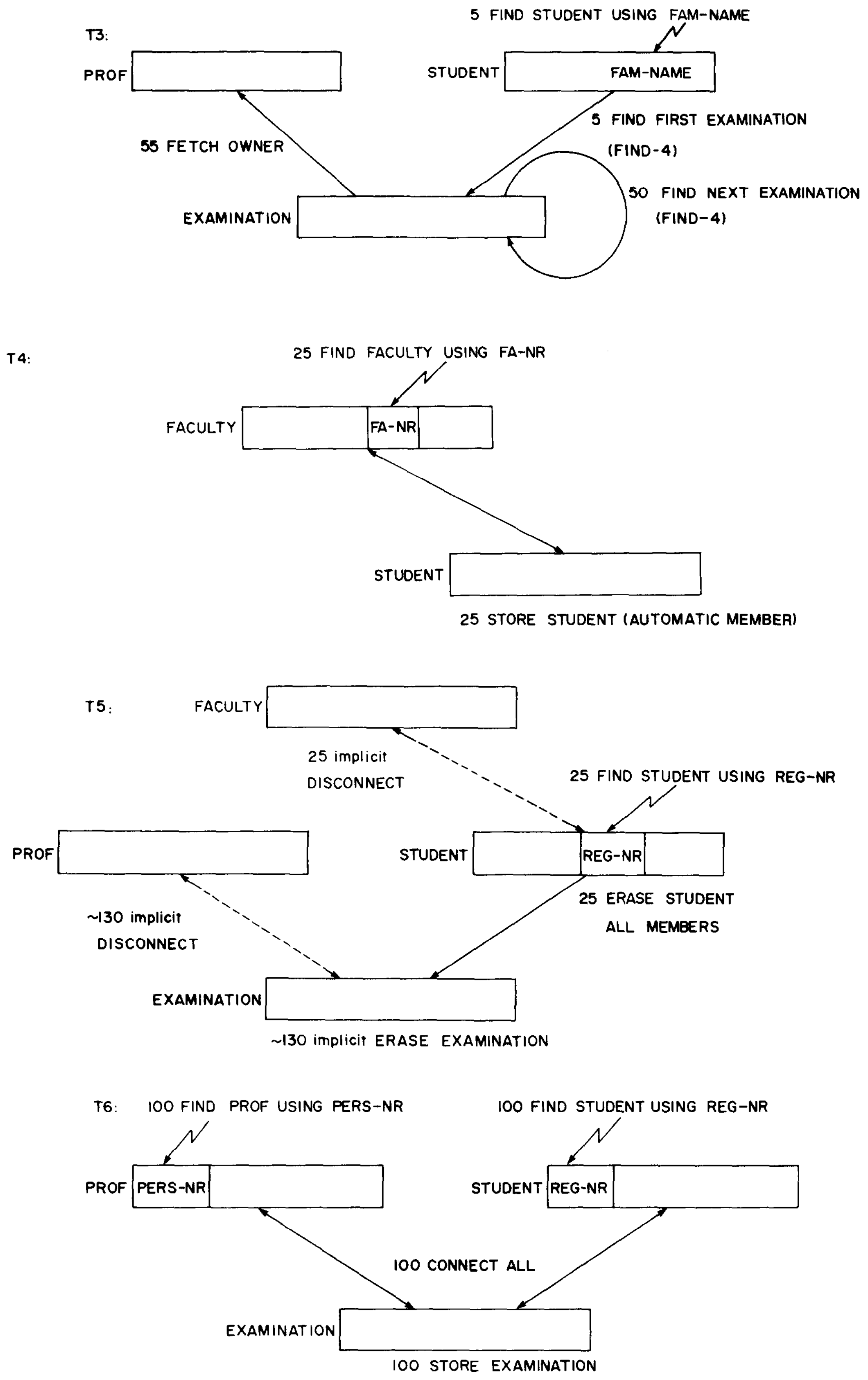
T3:
5 FIND STUDENT USING FAM-NAME
PROF
STUDENT
FAM-NAME
5 FIND FIRST EXAMINATION (FIND-4)
55 FETCH OWNER
50 FIND NEXT EXAMINATION (FIND-4)
EXAMINATION
T4:
25 FIND FACULTY USING FA-NR
FACULTY
FA-NR
STUDENT
25 STORE STUDENT (AUTOMATIC MEMBER)
T5:
FACULTY
25 implicit DISCONNECT
25 FIND STUDENT USING REG-NR
PROF
STUDENT
REG-NR
25 ERASE STUDENT ALL MEMBERS
~130 implicit DISCONNECT
EXAMINATION
~130 implicit ERASE EXAMINATION
T6:
100 FIND PROF USING PERS-NR
100 FIND STUDENT USING REG-NR
PROF
PERS-NR
STUDENT
REG-NR
100 CONNECT ALL
EXAMINATION
100 STORE EXAMINATION

Fig. 3. Logical access path graphs for the sample transactions.

```
RECORD NAME IS  FACULTY;
       DATABASE-KEY-TRANSLATION-TABLE IS 100.

RECORD NAME IS PROF ;
       DATABASE-KEY-TRANSLATION-TABLE IS 500.

RECORD NAME IS STUDENT;
       DATABASE-KEY-TRANSLATION-TABLE IS1000.

RECORD NAME IS  EXAMINATION;
       DATABASE-KEY-TRANSLATION-TABLE IS 3000.

   S E T S

SET NAME IS FAPROF;
    MODE IS CHAIN;
    POPULATION IS 10 INCREASE 5.

SET NAME IS FASTUD;
    MODE IS CHAIN;
    POPULATION 100 INCREASE 10.

SET NAME IS  EXAMINER;
    MODE IS CHAIN;
    POPULATION IS 250 INCREASE 25.

SET NAME IS  EXAMINEE;
    MODE IS CHAIN;
    POPULATION 10 INCREASE 5.
```

Fig. 4(a). Standard-DSDL of the database to be optimized.

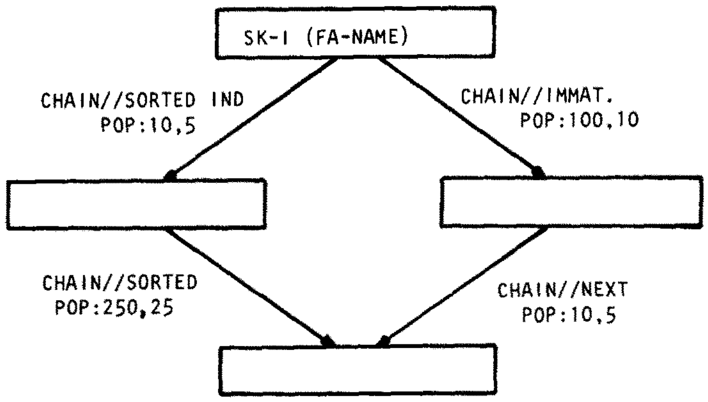

Fig. 4(b). Graphical representation of the standard solution.

representation containing the most important DDL- and DSDL-parameters influencing the execution time of the transaction load. We shall use this representation to discuss some sample solutions later in this paper. Its semantics should be obvious without further explanation. Apparently the standard solution can be optimized in many aspects: The ORDER IS SORTED-clauses in the DDL are not required in the DML-programs for transactions 1–6; there are no search keys speeding up the FIND-operations most frequently used; all the sets are implemented as single directed CHAINS, performing poorly w.r.t. STORE- and ERASE-operation etc.

The evaluation of this solution is shown in Fig. 5. The elapsed times of all DML-statements of the six transactions except READY and FINISH have been aggregated. Such a table serves to easily obtain information about the contribution of the statements to the total time.

| | | |
|---|---|---|
| T1 | FIND-7/3 | 50.1 |
| | FETCH-6 | 10.1 |
| T2 | FIND-7/3 | 55.1 |
| | FETCH-6 | 11.6 |
| T3 | FIND-7 | 70.2 |
| | FIND-4 | 10.8 |
| | FETCH-6 | 11.0 |
| T4 | FIND-7 | 68.2 |
| | STORE | 967.7 |
| T5 | FIND-7 | 437.o |
| | ERASE | 992.7 |
| T6 | FIND-7 | 333.6 |
| | CONNECT | 458.2 |
| | STORE | 131.7 |

Total elapsed time: 198,93 sec

Fig. 5. Elapsed times for the standard solution (average time per call in ms).

The elapsed times and the frequencies of use of the DML-statements are the primary criterion for the selection of optimization candidates to be supported by more appropriate storage structures. On the other hand, this table allows the analysis of each solution, i.e. the effects of the optimized clauses w.r.t. the access paths used by the different statements.

## 3. EVALUATION OF THE STUDENTS' OPTIMIZATIONS

During the experiment under consideration the logical data structure of the database and the transaction programs were kept fixed completely and had not to be altered by the students. Only the physical data structure could be changed in order to minimize the execution time of the given transactions defining the load of the database system.

A description of the various storage structures and access aids to be eligible for the optimization process is given in detail in the appendix. Unfortunately there is no strict separation of the logical and physical aspects of data structure in the CODASYL concept. As a consequence, some conceivable optimization steps were not possible without program modification while others were spread over the DDL and DSDL schemas. The restrictions w.r.t. the optimization features are listed in the appendix, too.

The evaluation of one optimization approach had to include the following steps: generating the database according to the respective DDL- and DSDL-description, loading the sample data, executing the sample transactions and recording the elapsed time for each DML-statement. To achieve comparability between the different solutions, all the evaluations had to take place in the very same environment. For our purpose we used the database system UDS[5], where the DSDL is called SSL (storage structure language) on a SIEMENS 7.748 with 1 MB of storage under a real storage operating system. The database system was given a buffer of 15 pages to reflect the small size of the total database. The database consisted of physical pages of 2048 bytes. The transactions were run in a fixed sequence in single user-mode with no other jobs executing concurrently. The database system performed UNDO-logging by recording physical before images of the changed pages. A measurement interface program linked between the user's program and the database connection module recorded the elapsed times for all DML-statements. These records were aggregated into statistical values by a separate program yielding tables as shown in Fig. 5.

### 3.1 *Results of our experiment*

The numerical results achieved by 12 groups competing in the optimization race are shown in Fig. 6. Many groups have started with a very poor initial approach being even worse than the standard solution in one case (group G02). But nearly all of them have considerably improved their solution in the 2. and 3. approach. The mean value drastically reduces from the 1. to the 2. approach and could have reached 58.46 sec in the 3. approach, if we had removed group G09 from the computation. This group had a disastrous third solution due to an essential misunderstanding of the differences between search keys on record types and sets, respectively. All the other groups had at least one solution sufficiently close to the optimum value of 50 sec. We don't want to consider, how many of the final improvements are due to inter-group communication, because the good solutions, whether they are found by chance or by analysis, necessarily look similar. This will be discussed in detail in Section 5. Figure 7 shows a graphical representation of the numerical results, but the groups are arranged in a different order to clarify the following aspect: We wanted to know, whether the students had tried different options in their approaches or whether they were primarily interested in refining a good solution to gain the minimum time, i.e. whether they looked upon the task as a game or as a competition. Analysing the DSDL's, we found two types of groups: type 1 had an initial solution showing a rather poor performance; as a consequence, they gave up this approach and tried a different one. Type 2 was obviously contented with the outcome of their initial solution; they didn't take any risk and tried only small variations of the approved DSDL. This shows that major changes were only applied to unfavourable solutions according to the rule never to change a winning team. The only group to take three different approaches was G00—a synonym for our own efforts to find an optimal solution.

### 3.2 *Use of the various optimization features*

The initial solutions showed a wide variety in the number of record search keys ranging from 0 to a maximum of 7 per group although the students have been warned of their extensive use.

Only two search keys—SK(PERS-NR) on PROF and SK (REG-NR) on STUDENT—were essential for a good solution. A conceivable search key SK(FA-NR) on FACULTY didn't affect the elapsed time very much, because of the absence of update operations on that record type and the disregard of load time. Search keys on EXAMINATION were definite design errors provoking a high update overhead without any gain in retrieval. The use of search keys on sets being negative optimization features in our sample database was negligible during our experiment.

The following table shows the frequencies of use for search keys on record types irrespective of the INDEX

| | G00 | G01 | G02 | G03 | G04 | G05 | G06 | G07 | G08 | G09 | G10 | G11 | mean |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1. appr. | 57.60 | 177.69 | 307.40 | 118.32 | 65.49 | 172.53 | 145.99 | 90.82 | 101.59 | 152.09 | 54.59 | 62.09 | 125.51 |
| 2. appr. | 67.98 | 71.08 | 103.53 | 54.70 | 73.46 | 61.59 | 121.70 | 59.74 | 74.88 | 100.26 | 58.37 | 64.06 | 76.01 |
| 3. appr. | 49.75 | 63.63 | 55.59 | 57.63 | 56.84 | 55.55 | 58.25 | 55.96 | 64.65 | 241.49 | 55.55 | 69.57 | 73.72 |

Fig. 6. Numerical results for all groups in seconds of elapsed time.

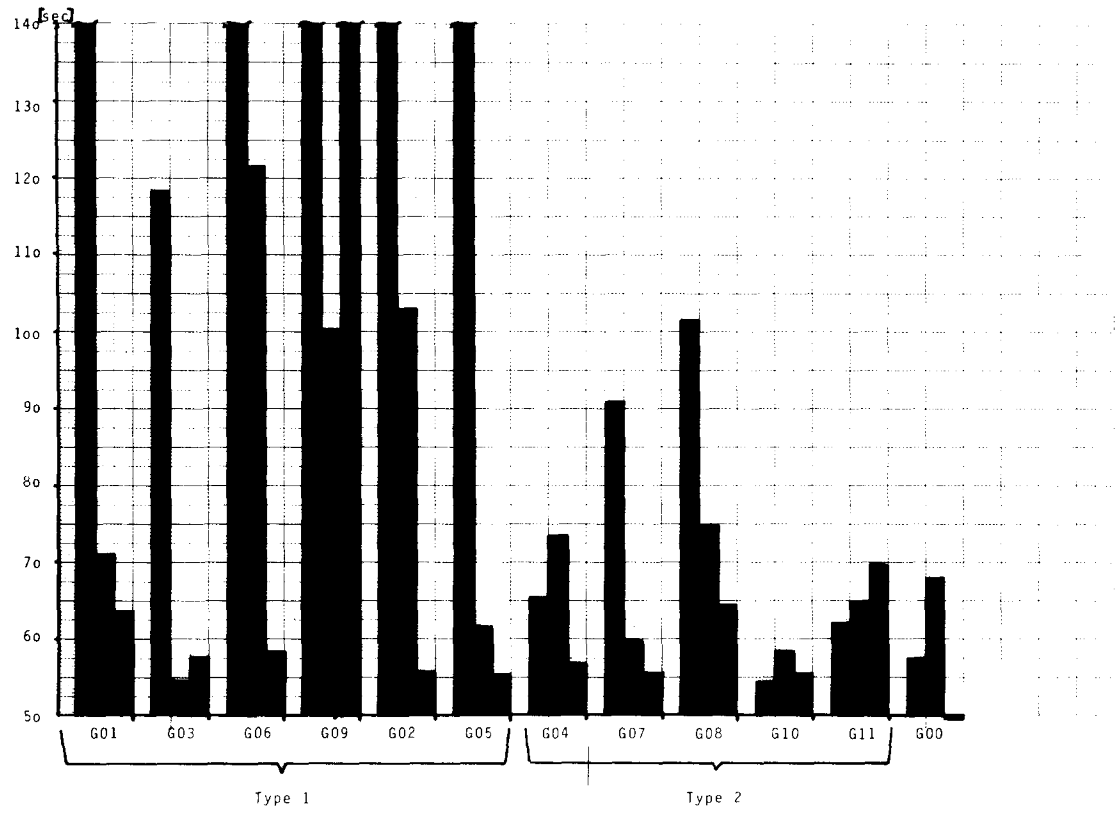

Fig. 7. Graphical distribution of the optimization approaches.

or CALC option. "Standard" indicates the standard approach having only one useless search key SK(FA-NAME) on FACULTY (12 times).

| | search keys used by 12 groups on | | | |
|---|---|---|---|---|
| | FACULTY | PROF | STUDENT | EXAMINATION |
| Standard | 12 | 0 | 0 | 0 |
| 1. approach | 5 | 10 | 12 | 6 |
| 2. approach | 6 | 12 | 14 | 5 |
| 3. approach | 8 | 11 | 11 | 0 |

A drastic change in the use of the 3 possible set modes was also observable. We consider the modes LIST, POINTER-ARRAY and CHAIN without further refinement due to ATTACHED/DETACHED-options. The use of LIST-mode was not possible in case of manual sets.

The following table aggregates the various design decisions concerning the set mode:

| | FAPROF | | | FASTUD | | | EXAMINER | | EXAMINEE | |
|---|---|---|---|---|---|---|---|---|---|---|
| | LIST | PA | CHAIN | LIST | PA | CHAIN | PA | CHAIN | PA | CHAIN |
| Standard | 0 | 0 | 12 | 0 | 0 | 12 | 0 | 12 | 0 | 12 |
| 1. approach | 4 | 0 | 8 | 1 | 7 | 4 | 7 | 5 | 8 | 4 |
| 2. approach | 4 | 0 | 8 | 2 | 9 | 1 | 12 | 0 | 11 | 1 |
| 3. approach | 4 | 0 | 8 | 2 | 10 | 0 | 12 | 0 | 12 | 0 |

The set mode of FAPROF was of minor importance due to the chosen transaction load and the small set occurrences. POINTER-ARRAY dominated absolutely in the remaining sets. About 90% of the PA-set modes chosen contained the ATTACHED-option. This frequent application of the ATTACHED-option destroying a possible cluster property of the owner record type is considered to be thoughtless from our point of view. The best solution (see Fig. 8a) gained about 10% by applying POINTER-ARRAY DETACHED twice. Initial approaches using CHAIN-mode without the LINKED TO

| | | |
|---|---|---|
| T1 | FIND-7/3 | 45.2 |
| | FETCH-6 | 10.2 |
| T2 | FIND-7/3 | 49.7 |
| | FETCH-6 | 12.8 |
| T3 | FIND-7 | 134.0 |
| | FIND-4 | 12.0 |
| | FETCH-6 | 14.3 |
| T4 | FIND-7 | 17.9 |
| | STORE | 131.6 |
| T5 | FIND-7 | 57.4 |
| | ERASE | 613.2 |
| T6 | FIND-7 | 38.0 |
| | CONNECT | 49.7 |
| | STORE | 58.1 |

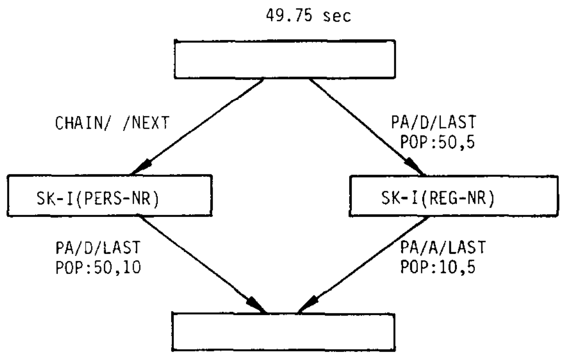

Fig. 8(a). The optimal solution.

| | | |
|---|---|---|
| T1 | FIND-7/3 | 57.3 |
| | FETCH-6 | 9.8 |
| T2 | FIND-7/3 | 63.7 |
| | FETCH-6 | 11.1 |
| T3 | FIND-7 | 72.4 |
| | FIND-4 | 11.4 |
| | FETCH-6 | 9.6 |
| T4 | FIND-7 | 17.7 |
| | STORE | 79.8 |
| T5 | FIND-7 | 54.0 |
| | ERASE | 564.8 |
| T6 | FIND-7 | 54.3 |
| | CONNECT | 103.3 |
| | STORE | 64.5 |

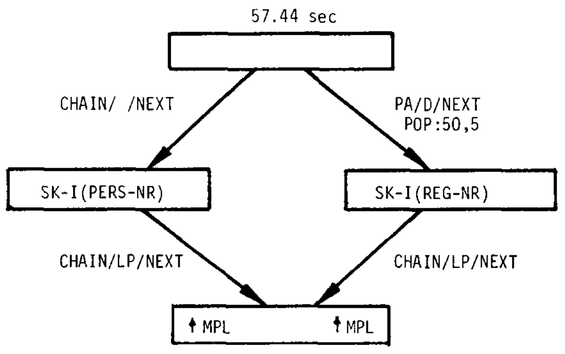

Fig. 8(b). An example of a pseudo optimization.

PRIOR-option (LP) were not very successful discouraging further optimization attempts. While a reasonable solution with CHAIN-mode and LP-option for the EXAMINER- and EXAMINEE-sets is conceivable, such a solution is not considered to be very stable for larger set occurrences and unclustered member records. The PA-solutions, however, promise a substantial robustness w.r.t. set occurrence growth and loss of cluster property of member records.

The evaluation of the set order chosen for the relevant sets FASTUD, EXAMINER and EXAMINEE can be condensed to the following statistics:

| | SORTED/SORTED IND. | NEXT/PRIOR | LAST/IMMAT. |
|---|---|---|---|
| Standard | 12 | 12 | 12 |
| 1. approach | 7 | 13 | 16 |
| 2. approach | 7 | 9 | 20 |
| 3. approach | 9 | 2 | 25 |

The set order has a major influence in connection with set mode LIST and CHAIN. IMMATERIAL has the effect of LAST when the DATABASE-KEYs are assigned sequentially. Therefore, a design of CHAIN together with LAST or IMMATERIAL set order had a disastrous influence on the optimization efforts. On the other hand, the POINTER-ARRAY-technique is supposed to be relatively insensitive to the choice of set order.

## 4. CHARACTERISTICS OF THE GOOD SOLUTIONS†

The major features of the best solutions (i.e. those below 60 sec) can be explained by means of the optimal DSDL shown in Fig. 8(a). For the set EXAMINER and EXAMINEE POINTER-ARRAY proved to be best choice w.r.t. the performance of transactions 4–6. The DETACHED-option for set EXAMINER was selected due to the following consideration: Transactions 5 and 6 do only affect the member records and the set connection data of set EXAMINER; in case of an ATTACHED-pointer array, each pointer array will be stored in physical proximity of the respective owner record occurrence. Consequently, the pointer array tables are spread over many pages. Hence, for each new ERASE- or CONNECT-statement another page has to be read, logged and written. The DETACHED-option achieves a clustering of the pointer arrays and thus reduces the physical I/O. This does not hold for set EXAMINEE, because transaction 5 deletes the STUDENT-records causing a log- and rewrite-operation for the respective page. Therefore, the update of an ATTACHED pointer array can be done without additional costs, whereas the DETACHED-option would trigger two more write operations. Hence, for set EXAMINEE ATTACHED should be better w.r.t. transaction 5, DETACHED w.r.t. transaction 6. This is confirmed by the results of different approaches showing no significant differences in either case.

For set FASTUD the set modes CHAIN and POINTER-ARRAY yield approximately equal results; LIST

†For a better understanding of following considerations we refer to the access path graphs shown in Fig. 3.

proved to be the optimal selection w.r.t. transaction 6 (this is due to a clustering of the updated records by FA-NR), but showed a comparatively poor performance for transactions 3 and 4. The search keys for PERS–NR and REG-NR on the record types PROF and STUDENT, respectively, are essentially necessary, irrespective of other features of the DSDL. The search key for PERS–NR is for free anyway, because no update is performed on record type PROF. That one for REG–NR causes a slight deterioration of 25 STORE-operations, but it considerably speeds up 100 FIND-7. Search keys on record type EXAMINATION are obviously disadvantageous since this record type is affected by more update- than retrieval-operations. The other clauses are of minor importance for the result.

Finally we want to demonstrate what can be called a pseudo optimization (Fig. 8b). The most frequently used sets are implemented as CHAIN LINKED TO PRIOR, and the elapsed time for this solution is close to the minimum. Here we have utilized the fact, that the member records of EXAMINER and EXAMINEE were stored in the same or adjacent pages. Hence, the overhead for updating the set connection data when storing or deleting an EXAMINATION-record became negligible. This solution would perform as poorly as the standard solution in case of distributed member records.

The choice of set mode played the most important role in the design process of the physical structures. While set mode LIST is only eligible in special cases (Membership AUTOMATIC and only for one set type per member record type) usually the key decision for the database administrator is CHAIN vs POINTER-ARRAY. Therefore, some general observations facilitating the critical judgement and advantageous use of these storage structures are listed below:

—CHAIN is preferable only for very small set occurrences, if at all:

● clustering of member records in a page is especially favorable in case of update and logging. Only one page is affected in the average.

● LINKED TO PRIOR is indispensible except for very special cases

● data manipulation (retrieval) of the member record requires frequent change of the set type

● update is difficult and expensive depending on the set order clause, the cluster effect and the hidden logging costs

● the access time behavior is very sensitive to the set order and the growth of the structure

—the behavior of POINTER-ARRAY is very stable and does not rely on special set properties:

● it is an "average good" structure

● two pages are affected in general during update

● it is insensitive to the set order clause

● the access time behavior only minimally reflects the growth of the structure

● the ATTACHED and DETACHED options support the clustering of either owner records with the corresponding pointer-arrays or of all pointer-arrays belonging to the same set type depending on processing and update frequencies.

## 5. LEARNING DATABASE ADMINISTRATION

The following comments and observations concerning the learning pattern and the behavior of the students during their optimization efforts may be of general interest.

(a) At first, a very uncritical and frequent use of specific DSDL features and options could be observed, e.g. "attached", "search key" "placement optimization". This was probably due to the seductive name of these options promising an overall optimization. In fact, these clauses should be applied very cautiously because they are optimizing only special cases while destroying the cluster property of records or increasing the maintenance costs. The initial failure of the optimization efforts taught the students to be more critical in using these "obvious optimization features".

(b) Such a practical course in database administration requires a thorough and profound theoretical foundation in advance. Lectures about the various methods of physical storage structures (at a very low level), their detailed mapping to storage media and the dynamic interior behavior of the system w.r.t. buffer management, logging etc. have to be a necessary prerequisite to enable precise design decisions and approximate estimates of alternative solutions. Otherwise, blind guesses and trial and error approaches are the consequence as demonstrated by the less successful students. Without a good theoretical preparation such a practical course would only serve to simulate the behavior of a naive user.

(c) Even with experience in database administration it is very difficult to predict the precise outcome of a specific DSDL design. Simple "improvements or changes" on a particular clause have a number of subtle, but expensive reactions on other clauses. For example, the clustering of the records of a record type destroys the cluster of the corresponding records of a set type. The influence of such counter-effects is amplified by the replacement algorithm of the system buffer. Additional hidden costs implied by locking and logging strategies have to be taken into consideration. Wrong design decisions gave an incentive to the (ambitious) students to gain a deeper understanding of the relative influence of the various optimization clauses. Probably, the more important learn-effect is to consciously deal with the intricacies and the interrelationships of the physical structures and their manipulation than to find the optimum of a particular structure under a given load.

Good and poor solutions were sometimes very similar as far as the choice of parameters and storage structure is concerned. But the assignment of a useless search key, a wrong primary allocation factor or an insufficient increase parameter turned a good solution into an insufficient one.

(d) Various reasonable or promising optimization approaches failed due to the lack of data independence. Such a negative experience made the importance of clear separation of logical and physical data structuring more vivid to the students than theoretical arguments. The major restrictions were imposed by the following properties of our DBMS and the CODASYL concept, respectively:

—Set membership MANUAL is incompatible with set mode LIST; on the other hand the membership could not be changed to AUTOMATIC since this would have caused program modifications (STORE/CONNECT vs STORE).

—Access to a record type programmed via a FIND-7 statement cannot be supported by LOCATION MODE CALC because this option is bound to a FIND-2 statement. A unified approach to record access via FIND-7 wouldn't result in any loss of functionality and performance.

—Changing the set selection clause from CURRENT OF SET to LOCATION MODE OF OWNER or vice versa always requires program modification. Therefore the system should be responsible for the optimal set selection. An appropriate way to specify this feature is a value-based set selection clause e.g. the STRUCTURAL CONSTRAINT concept of [4].

## 6. CONCLUSION

A DSDL optimization experiment as discussed in this paper seems to be an appropriate way of learning database administration. It provoked a lot of competition and interest among the students and mediated considerable insight in the problems of storage structure optimization. It showed the substantial gain of good solutions to the standard solution by a factor of 4; time differences of the various optimization attempts spanned a factor of 6 and more. But above all, this experiment taught the students what data independence is good for.

## REFERENCES

[1] *CODASYL Database Task Group Rep.*, April 1971, available from IFIP Administration Data Processing Group, 40 Paulus Potterstraat, Amsterdam.

[2] H. Schenk: Implementational aspects of the CODASYL DBTG proposal. In: *Database Management* (Ed. by J. W. Klimble and K. L. Koffeman), pp. 399–411. North Holland, Amsterdam (1974).

[3] CODASYL DDL Journal of Development, *June 73 Rep.*, available from IFIP Administration Data Processing Group, 40 Paulus Potterstraat, Amsterdam.

[4] Report of the CODASYL Data Description Language Committee. *Inform. Syst.* 3, 247–320 (1978).

[5] Siemens-Manual: Software product UDS, Universal Database System, Schema-DDL and SSL, *Reference Manual*, No. D15/5169-01.

[6] R. W. Engles: An Analysis of the April 1971, *DBTG-Rep.* In: *Proc.* 1971 *ACM SIGFIDET Workshop on Data description*, Access and Control, San Diego, California, November 1971, pp. 69–91.

## APPENDIX

### A.1 *Possible variations of DDL-clauses*

Unfortunately the CODASYL–DDL does not only describe the logical data structure of the database, but has some influence on the physical representation of access paths to the data as well[6]. For instance, the LOCATION MODE IS CALC-clause supports the placement of the records by a defined user key and their retrieval by means of a hash algorithm. SEARCH KEY-clauses specified in the schema-DDL allow the creation and maintenance of the additional direct access paths to the data records.

Additionally some of these DDL-clauses govern the use of CODASYL–DML commands causing a high interdependence of DDL and DML. For example, the usage of FIND-2-statements (FIND ANY) in the DML requires the LOCATION MODE IS CALC-clause in the DDL for the record types concerned. A set membership OPTIONAL MANUAL altered to MANDATORY AUTOMATIC in the schema would provoke execution time errors for all CONNECT- and DISCONNECT-statements for that set type.

In order to avoid these problems some DDL options were not eligible for the optimization of the access behavior. The two most important DDL-clauses used as optimization candidates were the ORDER-clause for sets, and the SEARCH KEY-clause for sets and record types. The ORDER-clause has the following form:

$$\text{ORDER IS}\left\{\begin{array}{l}\text{SORTED [INDEXED]}\\ \text{FIRST}\\ \text{LAST}\\ \text{NEXT}\\ \text{PRIOR}\\ \text{IMMATERIAL}\end{array}\right\}$$

The sequence of the set members in our database had no logical meaning; thus the DDL-ORDER-clause was of no importance for the results of our six transactions and could be used for optimization of the set processing.

The SEARCH KEY-clause specifies, that certain fields of a record type can be used for the fast direct retrieval of records of that type from the database. Such search keys are implemented and maintained by sorted tables according to the dynamic-levelled-index-table concept[2] (B*-tree concept) in case of the INDEX option or by scatter tables (indirect hash areas) in case of the CALC-option. In our database system search keys can be either specified for the member records of each set in the schema, or they can be specified for all records of a certain record type. In the latter case an implicit set with OWNER IS SYSTEM is automatically generated, and all records of the corresponding record type are members of that set. The search keys for record types can thus be treated like set search keys.

### A.2 *Optimization features of the DSDL*

Apart from the schema-DDL-clauses described above other access path optimizations are possible by means of the data storage description language (DSDL). If the SSL-parameters are not explicitly specified, our database system assumes a default storage structure with predefined values which may yield a poor performance for many applications. The DSDL is supposed to be the most important tool for the database administrator tailoring the physical data structure of this database to the special characteristics of the transaction load. The different DSDL-clauses are totally invariant to the semantics of database transactions guaranteeing their isolated variations for optimization purposes. The most important DSDL-clauses are discussed briefly.

A.2.1 *The RECORD-clauses.* The record concept of the DBTG-proposal requires a unique identifier of a record (DATABASE-KEY) within the database remaining unchanged during the time the record exists in the database. An appropriate implementation technique giving some kind of indirection for reorganization purposes is the use of "transformation tables" associating the DATABASE-KEY of a record to its physical address. The so called DATABASE-KEY-TRANSLATION-TABLE (DBTT)-clause specifies the maximum number of entries in such a table defined for every record type. The DBTT of a record type is located in consecutive pages of an area separated from pages containing the corresponding record occurrences. Hence, the location of a record via its DATABASE-KEY requires two logical accesses to the database. The PLACEMENT OPTIMIZATION FOR SET-clause is used to obtain physical proximity of records within set occurrences independent of the set mode specified. This clause causes the database system to reserve free space for sets and to place owner record, member records and set connection data in one or more consecutive disk pages.

The INDEX-clause for records allows the placement control for search key tables or hash areas (PLACING). Furthermore a TYPE- and a DYNAMIC REORGANIZATION-parameter are valuable options to tailor large B*-trees to specific requirements. The TYPE-option defines the format of entries in a search key table (repeated key or database-key-list in case of duplicate keys). The DYNAMIC REORGANIZATION-clause is used to specify the number of pages involved in the reorganization process of index tables. This parameter controls a typical time–space tradeoff. A high number of pages to be reorganized causes a higher overhead when page splitting occurs. On the other hand, a better filling rate of the index pages is gained saving storage space and sequential access time. Because of the size of our database these parameters are supposed to be of minor influence.

*A.2.2 The SET-clauses.* The MODE-clause specifies the SET mode, that is, the way how the owner and members of a set are connected physically. Irrespective of the actual set mode chosen every member record contains the logical address (DATABASE-KEY) of his owner. This measure greatly facilitates the member-owner access in case of information bearing sets. The insertion order is given by the set ORDER-clause in the DDL. As a peculiarity, ORDER IS IMMATERIAL is implemented in our system as SORTED BY DATABASE-KEY resulting often in an insertion sequence LAST. The complete form of the mode clause[5] is:

```
         ⎧ CHAIN [LINKED TO PRIOR]
         ⎪
MODE IS  ⎨ ⎧ POINTER-ARRAY ⎫ ⎧ ATTACHED  TO OWNER                   ⎫ ⎫
         ⎪ ⎩ LIST          ⎭ ⎨ DETACHED [WITHIN  realm-name]        ⎬ ⎬
         ⎩                   ⎩           [WITH PHYSICAL LINK]       ⎭ ⎭
```

MODE IS CHAIN causes the set owner and all his members to be connected by forward pointers (NEXT) embedded in the record occurrences. When LINKED TO PRIOR is specified all pointers of the chain are bidirectional.

MODE IS POINTER-ARRAY leads to the creation of a table for each set occurrence containing pointers to all member records of the set occurrence. The page address of the pointer-array is stored in the owner's entry in the DBTT.

MODE IS LIST specifies that all members of a set occurrence are to be stored in physical contiguity within a page and, if necessary, in pointer-connected pages. As a consequence, the same record type cannot be member of two sets with MODE IS LIST. Because of their cluster property lists are preferable in case of frequent sequential processing. The set mode LIST requires the set membership MANDATORY AUTOMATIC; thus, the DML-statements CONNECT and DISCONNECT are not allowed for LIST-sets.

In case of an order clause SORTED INDEXED pointer-arrays and lists are maintained as B*-trees. SORTED INDEXED in connection with CHAIN-mode generates a sorted chain and an additional index table.

If the option ATTACHED TO OWNER has been specified for a LIST or POINTER-ARRAY, it will preferably be located in the same page as the related owner record occurrence reducing a possible cluster effect for the occurrences of the owner record type. The access from owner to member is supposed to be accelerated by this option.

DETACHED WITHIN realm-name places the POINTER-ARRAY or LIST in the owner's realm or the specified realm respectively; it will not be located in the owner's page preserving a possible cluster property of the owner record type. The option WITH PHYSICAL LINK causes the owner record to contain the physical address (page address) of his related POINTER-ARRAY or LIST.

The SET-clause POPULATION IS integer-1 [INCREASE IS integer-2] is used to reserve space for tables, lists and/or pointer-arrays by specifying the number of member record occurrences to be taken into account per set occurrence. Integer-1 serves for the initial assignment, integer-2 describes the increase for the secondary allocation in case of overflow. Whatever the values of integer-1 and integer-2 are, the initial assignment will never exceed one page.

The INDEX-clause for sets corresponds to the INDEX clause for record types and has the same meaning (see chapter A.2.1).

The clause MEMBER IS PHYSICALLY LINKED TO OWNER causes the member records of the set concerned to contain a physical owner pointer accelerating the member-owner access path.