Problems of Building a Hybrid Data Definition Facility

J. W. Dempsey
J. K. Mullin

RCA Corporation
Cinnaminson, New Jersey

Problems of Building a Hybrid Data Definition Facility

Abstract

The capability to interrogate COBOL describable files was added to an existing data base management system, RCA's UL/1. This paper discusses features of the implementation of UL/1 which tended to facilitate the COBOL additions and which features would have been desirable in order to simplify the extension. The authors believe that as the data base management field evolves, more extensions to the set of files handled by data management systems can be expected. The lessons learned in this implementation could well have broad applicability.

Introduction

As a result of the growing need for a flexible, easy-to-use tool for coping with the problem of storing, manipulating, and retrieving data, a large number of data management software packages have been developed. Such a system is UL/I, which has been developed by the RCA Corporation.

UL/I is a non-procedural language for interacting with a data base. The language consists of four divisions, each of which has several sections. The divisions are Establishment, Interrogat Update, and Revision.

Establishment is a process by which a file is added to a data base in a form standard to the system. The Establishment division processes a description of the file and reads the data to form a system standard file.

The Interrogation division is used to place criteria on items within a record and extract a set of data items from the records which satisfy the criteria. For example, to find and print the names of all employees who earn more than $10000 in a branch store in Boston:

```
INTERROGATE SALARIES   *
RETRIEVAL CRITERION
CITY EQ BOSTON AND SALARY GT 10000   *
PUBLISH REPORT HIGHSAL
    CITY ROW 1 COLUMN 3
    NAME ROW 1 COLUMN 12      *
```

The Update division is used to modify or delete existing records.

The Revision division is used to change the record structure of the file.

COBOL is also a language which is used to define operations on a file of data and it too has several divisions. The DATA DIVISION is used to describe the characteristics of the file and the PROCEDURE division is used to specify operations on the file. Unlike UL/I, however, COBOL is a procedural language.

A large number of users of existing COBOL files could benefit from the use of the non-procedural inquiry facilities of UL/I but were unwilling to convert their files to the UL/I format since this would require either scrapping existing COBOL programs which operated on the files or maintaining two versions of the files. It was decided, therefore, to add to UL/I the ability to accept a COBOL DATA DIVISION description of a file in lieu of the UL/I description - and to query the file directly without requiring conversion to the UL/I standard format. This paper describes some of the problems involved in forming such a hybrid data description facility and concludes with some suggestions for development of future data description facilities.

The Original Data Definition Facility

The data definition facility of UL/1 is contained in the Establishment division. This division may be viewed as a transducer which accepts as input a file and a description of the file and produces as output another file, in system standard format, and its description. It is thus a mapping of the data structure associated with the file into the storage structure of the UL/1 system.

UL/1 views a file as consisting of one or more similarly structured entries (called records) where a record may contain a hierarchy of groups and data items. The data definition facility is divided into several sections, each of which describes a particular characteristic of a record.

The relevant sections are

A) Data Identification

This section is used to assign to each item an identifier and a data type (numeric, alpha-numeric, coded or date)

B) Structure

This optional section is used to specify the grouping of the data items identified in the identification section and whether an item is single or multi-valued.

C) LAYOUT - Description of the Input Stream

This section is used to describe the format of the records being input to the file. Two methods of description are available. The positional form is a series of field length specifications describing where the data items are to be found in the input stream. A field length specification is in the general form:

item identifier [integer-1] X integer-2

where integer-2 specifies the length of the field containing the item and integer-1 specifies the number of repetitions of the field. The labeled form of input data requires that each input item be preceded by its item name. This label must be separated from the item value by at least one space.

An example of a data description in UL/1 is found below.

```
ESTABLISH SAMPLE                        the file name

IDENTIFICATION

#1 A NAME                               here types are associated
                                        with data names.  #1 and
#2 A SEX                                NAME are synonyms for an
                                        alphabetic (A) data field.
#3 N AGE

#4 N SALARY

#5 A JOB

#6 A SKILS


STRUCTURE                               the field #6 (SKILS) is
                                        multivalued.
#6 REPEATS


LAYOUT
                                          (1)
#1 X3 #2 X1 #3 X2 #4 X5 #5 X23 #6 3 X1


INPUT

JOEM3050000MANAGER OF OPERATIONS  ED


JIMM3050000 PROGRAMMER            EB   * *
```

(1) Specifies that item #1 is in the first 3 characters, #2 is in the next etc.

The file "SAMPLE" will be established with two records. The type of storage, fixed size or variable size selected for the data items and the lengths of fields will depend on the input data.

The UL/1 system accepts the data description and the input and produces the system standard file and its description.

The data definition produced by the Establishment division consists of a set of tables.

A) General File Information

This table contains information about the file. Included are maximum record size, blocking, record count, etc.

B) The Tree Directory

This table contains information about the data structure within the records.

C) A Name Directory

This table maps names associated with items into the table describing the attributes of the items.

D) The Item Information Table

Here is kept information about each data item including:

1) type (alpha, numeric, etc.)

2) addressing information

3) length

4) multiplicity (single or multivalued)

5) group membership

6) maximum number of subitems

7) external numeric form

8) a security level number

The COBOL Data Description Facility

The COBOL data file is described through the FILE SECTION of the DATA DIVISION. This section is used to describe the structure of the record and the type, size and names of the individual items.

This section has the following features:

A) A level numbering scheme to assign a hierarchical structure to the data.

B) A USAGE clause to assign each item a data type.

C) A PICTURE clause to give information about the length and editing features of the data. The PICTURE clause also gives additional information about the data characteristics such as scaling factor for numeric data.

D) The OCCURS clause which specifies the number of repetitions of a multi-valued item.

The file described on page 5 would be described by COBOL as follows:

```
          DATA DIVISION
          FILE SECTION
          FD SAMPLE; BLOCK CONTAINS 2 RECORDS;
          RECORDING MODE IS F; LABEL RECORDS ARE
          STANDARD; DATA RECORD IS MASTER RECORD.
     01   MASTER-RECORD.
          02 NAME-PICTURE IS X(3).
          02 SEX PICTURE IS X.
          02 AGE PICTURE IS 99.
          02 SALARY PICTURE IS 9(5).
          02 JOB PICTURE IS X(23).
          02 SKILS OCCURS 3 TIMES PICTURE
             IS X.
```

The Hybrid Data Definition Facility

Two factors made it impossible to map the COBOL DATA DIVISION definition of a file directly into the data definition produced by the ESTABLISHMENT DIVISION of UL/1:

1. Whereas UL/1 assumes that numeric data was either integer or floating point, COBOL allows five different types of numeric data and allows specification of a scaling factor as well.
2. UL/1 uses a combination of information in the structure information table and pointers stored with the data records to access data for an item which is a member of a group. To access data from the COBOL files we had to rely solely on the structure information derived from the DATA DIVISION.

It was necessary, therefore, to modify the form of the data definition produced for the COBOL files.

We were constrained in our choice of implementation strategy by the requirement that the changes in the existing system's subroutines be kept to a minimum. Since there are over three hundred modules in the system and most of these referen the data definition tables either directly or indirectly this would have been an extremely difficult task.

Fortunately, however, the system was designed so that all access to the data records was channeled through a single routine. This meant that by modifying this routine we could reinterpret those fields in the item information table which were used to locate data. These fields we used to point to auxiliary tables which contained the additional typi and structure information which was required as well as the information needed to locate the data. We also modified this routine so that in the case of those types of numeric data in COBOL which do not exist in UL/1 it converted the data to a standard UL/1 type before passing the value to the calling routine.

We were forced to use auxiliary tables rather than revise the format of the item information table because there were many modules which accessed information in that table directly. The publication translator, for example, used information in the table to format the output of an interrogation. As a result we could not change the format of this table without making corresponding changes in the system.

Thus it was the existence of a common data access routine which made our task less difficult and the lack of a common definition table access routine which made it more complicated. The conclusion is obvious: a more flexible system requires that the data definition tables be built and accessed through a small set of functional subroutines. In this way the semantics of the definition can be freed from a rigid syntax. (A beneficial side-effect is that more readable and more easily debugged code should result.)

Conclusions

The approach we took in adding the COBOL data definition facility to UL/1 was feasible because:

1. The COBOL data types could be converted into UL/1 data types.
2. The logical data structures were much the same in COBOL and UL/1 (although the physical storage structures were different).

This is not the general case, however, and we would not want to follow this approach to add other data definition facilities to the UL/1 system. Even with the existence of central data access and data attribute access routines, the labor involved in building the translators and interfacing with these routines is extensive and must be done for each new language. We feel that a generalized data definition facility must be developed to eliminate this problem.

Such a data definition facility must provide a flexible means of specifying the location within a record of data values associated with items. It should not provide a standard record format but a standard way of describing record formats. Both COBOL and UL/1 specify record formats in terms of length of data items. UL/1 also takes a step away from fixed format by allowing each data item to be preceded by a delimiter of the form "#n," where n is an integer. We believe that a generalized data definition facility must allow the use of a much broader class of delimiters to free us from fixed record formats.

It must also provide a flexible means of specifying the semantics of the data. One of the components of semantics is the structure of the data, i.e., the specification of relations among data items. Hence a powerful means of specifying the mapping from data structure to storage structure must be provided. The semantics is also controlled by the operators on the data. These operators are independent of the definition facility. The specification of data types, however, provides a selector function which controls the semantics of the operators within a system. For example, "+" operating on an item of data of type "numeric" would specify addition; operating on an item of type "string" it could mean concatenation. We do not believe it wise to limit the number of data types. Hence we believe that a means must be provided to define new data types by specifying the effect of these data types on existing operators.

In summary, we think that we need a language which is capable of:

1) defining record formats in a flexible way
2) specifying relations among data items
3) defining new data types

The creation of such a facility would go a long way toward making data management facilities more broadly applicable and it would also facilitate the transfer of data between systems.