To: Distribution

From: Tommy James Ratliff

Date: December 20, 1978

Subject: Extended Data Submodel Capabilities

The function of a Multics Relational Data Store (MRDS) data submodel is to associate the user's view of a data base with the actual data base description (i.e., the data model). At the present time, this association is accomplished by equating the user's name for a relation or attribute to the data model name for the same item. This only allows relations to be defined in the data submodel which are subsets of relations in the data model. This MTB provides an overview of the changes which will be made to the create_mrds_dsm source which will enable the data submodel to be a relational derivative of the relations in the data model. It describes the new data submodel source segment and the implications of the new syntax.

The MRDS Reference Manual (Order No. AW53) can prove helpful if the reader is not familiar with relational data base terminology. The manual also describes the MRDS selection expression which is a very important part of the new source segment. It is assumed that the reader has read MTB-359 which describes the proposed enhancements to the Multics Data Base Manager (MDBM). Please forward all comments and suggestions to the author at:

Honeywell Information Systems
P. O. Box 6000 MS K-28
Phoenix, AZ 85005

-or-

Ratliff.Multics on system M

## Introduction

Currently, any relation defined in a MRDS data submodel must be a proper subset and/or a renaming of an existing relation defined in the data model. Hence, it is possible only to rename relations and attributes, omit relations and attributes, and reorder attributes within a relation. The data submodel capabilities will be extended to provide "virtual" relations, i.e., relations composed of attributes which may be distributed among several relations in the data model.

In order to construct data submodels composed of "virtual" relations, the capability to use a MRDS selection expression within the data submodel source segment will be provided. It is significant to note that these relations may not be in normal form since any tuple selectable via a MRDS selection expression may be used to form a relation. From a user standpoint, the incorporation of the MRDS selection expression into the submodel source segment will allow the use of a syntax similar to the one now used for regular update and retrieval.

## Benefits

The primary benefit of defining virtual relations in the data submodel is to simplify selection expressions for LINUS and MRDS users. Several limitations on the data submodel due to the old syntax will subsequently be eliminated. The new syntax will allow more relations to be defined in the data submodel than defined in the original data model. Likewise, the number of attributes present in the data submodel relation statement may be greater than the number of attributes defined in the existing data model relations. In addition, any tuple in the data model selectable by a MRDS selection expression may be used to form a relation in the data submodel. None of these features could be performed using the old data submodel source segment syntax. The addition of these features will make the data submodel more efficient. The new syntax will allow MRDS to optimize response time to a user's request and will allow the Data Base Administrator (DBA) to more precisely specify the portion of the database in which the user is allowed to manipulate.

## New Submodel Source Segment

The additions and changes to the capabilities which will be specified in the create_mrds_dsm source segment are discussed in

this section. Although the changes proposed will significantly alter the syntax of the create_mrds_dsm source segment, it will be possible to easily determine whether the source segment is of the current or new version, and to parse accordingly. Hence, the current style source segment may still be processed. The current and the new version are mutually exclusive and cannot be mixed within a segment. An example of the new source segment is given at the end of this MTB and may be used as a reference for understanding the new syntax described below.

A required clause specifying a MRDS selection expression will be introduced in the submodel source segment. This will eliminate the need for the equal convention used in the current source segment since the data model relations used to create the data submodel relation will be specified in the -range clause of the MRDS selection expression. The new relation statement is as follows:

relation: <dsm_relation_expression> [,...,<dsm_relation_expression>];

where <dsm_relation_expression> is a data submodel relation expression defined as:

<dsm_rel_name> (<dsm_attr_name> [...<dsm_attr_name>])
-mapping <selection_expression>

where <dsm_rel_name> is the data submodel relation name, <dsm_attr_name> is the name of the attribute in the data submodel relation, and <selection_expression> is a character string defined in Section 4 of the MRDS Reference Manual, Order No. AW53, under the heading "FORMAL DEFINITION OF THE SELECTION EXPRESSION".

There will be two optional file definition keywords in the data submodel source segment. These keywords will allow the DBA to explicitly define the associations among the various relations in the data submodel and the associations between the files in the data model and the files in the data submodel. By providing two keywords to describe the data submodel file, these associations may be shown much more clearly then by using one keyword to describe both associations. Futhermore, the use of two keywords will allow the equal convention to be eliminated entirely throughout the data submodel source.

The optional file statement will be used to define the data submodel relations which reside in the data submodel file. The format of the file statement is as follows.

file: <dsm_file_name> (<dsm_rel_name> [,..., <dsm_rel_name>])
[,..., <dsm_file_name> (<dsm_rel_name> [,..., <dsm_rel_name>])];

where <dsm_file_name> is the data submodel file name being defined and <dsm_rel_name> is the name of the data submodel relation residing in the file.

The relationship between data model files and the data submodel file will be specified in an optional file_list statement in the data submodel source segment. The syntax of the file_list statement is:

file_list: <dsm_file_name> (<dm_file-name> [,..., <dm_file_name>])
[,..., <dsm_file_name> (<dm_file_name> [,..., <dm_file_name>])];

where <dsm_file_name> is the name of the data submodel file and <dm_file_name> is the name of the data model file containing relations which were used to compose relations in the data submodel file.

If the file statement is not present in the data submodel source segment, each data submodel relation is assumed to reside in an identically named file. When the file_list statement is not present in the the source segment, the data model files used to compose the data submodel are determined by the selection_expression in the relation statement. When neither the file statement nor the file_list statement are present in the submodel source, both of the above defaults occur.

The data submodel relation names, attributes, and file names do not have to be the same name as relations, attributes, or file names in the data model relation. However, the attributes in the data submodel must be qualified in the -select clause in the selection expression. A direct mapping between the attributes in the data submodel relation attribute list and the attributes in the corresponding data model relation will be performed.

## Restrictions

Some forms of the MRDS selection expression cannot be interpreted in the new data submodel source segment. The data submodel created from the use of a selection expression will contain all of the tuples defined by the selection expression. Therefore, the use of "-another" is not needed and, if used, will result in an error message. Similarly, the use of "-current" will not be allowed in the source segment.

Only one instance of duplicate selected items will be contained in the data submodel. If the -select clause contains the -dup option, it will be ignored and a warning message will be printed.

The ".V." control code cannot be used in selection expressions used to create the data submodel. If the ".V." control code is used, an error will result.

Notes

It would be advantageous to allow relations in the data submodel to be updatable. However, several problems which will not be resolved in this implementation do arise. To support the use of MRDS selection expression to define the data submodel, it is necessary to "map" the relations in the data submodel against the relations in the data model. Retrieval operations can be easily performed since retrieval does not affect the tuples in a data model relation. However, update operations will be allowed only when the data submodel relation is a subset of the data model relation. This restriction is due to the fact that update operations on the data submodel affect only the data visible in the data submodel relation. Since individual tuples can be omitted in the data submodel relation, a new tuple may be inserted into the data submodel with an attribute value which is a duplicate primary key in the data model. This can not be allowed. Futhermore, we could insert a tuple into the data model via the data submodel in which one or more of the attribute values are unspecified. The null attribute value may be a primary key and therefore, this will not be allowed. In general, a retrieve-only capability will apply in all other cases. However, other exceptions may be defined later.

It is important to note that if a tuple variable is specified in the -select clause by itself all the attribute values in the designated tuples will be selected. In this case, the data submodel relation must contain the same number of attributes as the original data model relation.

Data submodel attributes will be direct assignments from data model attributes. The capability for a submodel attribute to be the result of a function or expression of several data model attributes will not be allowed in the initial implementation.

Ideally, the data submodel would include definitions not only of the data submodel relations but also of the domains in which those relations are defined. Data submodel domains should be allowed to have different characteristics from the coresponding data model domains. Domains will not be redefined for attributes in the data submodel source. The capability to convert data types is provided whenever an attribute is redefined in a program which accesses the data submodel (also applies for redefinition of data types in the data model).

The proposed data submodel source segment retains very little similiarity to the current submodel source segment syntax. However, when called for initialization, the scanner will do a string search of the data submodel source for either an equal sign, the mapping keyword, or quotes. Detecting the equal sign and/or not detecting the others will set a flag which will allow semantics to use the current source and disallow the new source. The opposite finding will cause the reverse to occur. This will allow the data submodel source segment to be parsed using an LRK grammar which will permit the use of both the current syntax and the new syntax, thereby providing complete upward compatability. The LRK grammar will utilize scanner/semantic communication to ensure proper use of the rules according to the source being parsed. The selection expression will be passed as a quoted string to its own parser.

## Implementation Method

Currently, the data submodel is represented as keyed-sequential files under vfile_. The relation definition record has as its key the data submodel relation name. The name of the corresponding data model relation and an entry for each data submodel attribute and the corresponding data model attribute are contained in each record. Similiarly, the file definition record is keyed on the character string "file:" concatenated with the file name. The name of the data model files and the data submodel relations appearing in the file are contained in each of these records.

The data submodel and the data submodel files will be implemented as segments using linked or threaded lists. This will simplify access to the data submodel and it will be consistent with the

way in which the data model is currently represented. All submodels will be validated when the submodel is opened, therefore, all existing submodels will be allowed to be implemented via vfile_.

Example

The following data model source segment defines a database.

```
domain:  bno    char (2)  unal,
         sno    char (2)  unal,
         bname  char (30) unal,
         sname  char (30) unal,
         status char (3)  unal,
         city   char (10) unal,
         pno    char (2)  unal,
         pname  char (15) unal,
         color  char (10) unal,
         wght   fixed bin (35) unal,
         qty    fixed bin (35) unal;

relation:  buyer    (bno* bname status city),
           supplier (sno* sname status city),
           part     (pno* pname color wght),
           sp       (sno* pno* qty),
           bp       (bno* pno* qty);


file:      person   (buyer, supplier) -blocked,
           order    (sp ,bp) -blocked 2 2/1,
           material (part) -unblocked;
```

The following data submodel source segments define valid data submodels corresponding to the above database.

```
relation:  buyer (bnum city status)
               -mapping "-range (a buyer)
                         -select a.bno a.city a.status",
           /*renamed, reordered, and omitted attributes*/


           vendor (num sname)
               -mapping "-range (a supplier)
                         -select a.sno a.sname",
           /*renamed relation, renamed attributes*/


           part (pno color wght)
               -mapping "-range (a part)
                        -select a.pno a.color a.wght
                        -where a.color=""red""",
           /*qualification*/
```

```
            inventory (pno qty)
                -mapping "(-range (a sp)
                          -select a.pno a.qty)
                          -union
                          (-range (b bp)
                          -select b.pno b.qty)",
            /*union of two relations*/


            supplier_buyer (sname city)
                -mapping "-range (a supplier) (b buyer)
                          -select a.sname a.city
                          -where a.city=b.city",
            /*more than one range definition in -range clause*/


            equal_locale (sname bname city)
                -mapping "-range (a supplier) (b buyer)
                          -select a.sname b.bname a.city
                          -where a.city=b.city",
            /*join of two relations*/


            join (bname bcity sname scity)
                -mapping "-range (a buyer) (b supplier)
                          -select a.bname a.city b.sname b.city",
            /*join of two relations with some of the same attributes*/


file:       person (buyer, vendor, supplier_buyer, equal_locale, join),
            /*  data submodel file name same as data model file name */
            /*  These files may or may not be associated */


            part   (part, inventory);
            /* data submodel file name same as a relation name */



file_list:  person (person),
            /* data submodel file associated with one data model file */


            part   (order, material);
            /* dsm file associated with more than one dm file */
```