Table 5.2.3.1. *Permitted constructions for a Star_Base conditional request*

<data request>
<data request> <operator> <text string>
<conditional request> & <conditional request>
<conditional request> | <conditional request>
!<conditional request>

of characters and a question mark (?) represents *any single* character.

(vi) A request for looped data returns the items in the order requested, making any necessary adjustments to the structuring of nested loops to preserve the original context.

(vii) A request for data within a save frame returns those items plus the associated context.

(viii) If a requested data item includes a save-frame pointer as a value, the referenced save frame is returned intact. All other pointers contained within the returned data are resolved.

(ix) A request for a data item in a global data block will also return the data-block headers within the scope of the global block.

(x) The scope of a data request is *the entire input file*. Control of the search scope is only possible within branching requests.

5.2.3.3. The *Star_Base* conditional request

While a data request allows retrieval of data items according to *name*, conditional requests allow retrieval of data items by *value*. The general form of a conditional request may be characterized as <data request><operator><text string>, where <data request> is any data request as defined in the preceding section, <operator> is any of the test operators defined below, and <text string> is a string pattern against which values of data items retrieved by the data request are matched according to the operator specified.

Conditional requests may be combined by set operators &, | and ! to provide logical AND, OR and NOT tests. Table 5.2.3.1 lists the allowed constructions for a conditional request. A bare data request is considered a degenerate case of a conditional request.

The construction <conditional request> & <conditional request> allows for the *conjunction* of conditionals. All data are returned (including context) from the intersection of sets of data that individually satisfy the conditions to be a non-empty set.

It is important to note that the conjunction of conditionals based on different data names is the empty set.

The construction <conditional request> | <conditional request> allows for the *disjunction* of conditionals. All data are returned (including context) from the union of sets of data that individually satisfy the conditions to be a non-empty set.

The construction !<conditional request> allows for the *negation* or *complement* of conditionals. All data are returned (including context) from the universal set of data that do not satisfy the conditions of the conditional request. The universal set is defined as the input file.

Table 5.2.3.2 lists the permitted value-matching operators when a retrieved data value is compared with a target text string in the basic test <data request><operator><text string> described above. (If the <text string> contains white-space characters, it must be quoted with matching single or double quotes. The test is performed on the value of the text string, *i.e.* the complete text string including white-space characters but omitting the surrounding quote characters.)

Two classes of operators are defined. *Text operators* may be used to test for string equality, substring containment or greater and lesser values (where the 'greater' and 'lesser' values for text strings are based on the ASCII character set ordering sequence).

Table 5.2.3.2. *Value-matching operators in Star_Base conditional requests*

Requests are of the form <data request> <operator> <text string>. The second column describes the relationship that data identified by the <data request> must satisfy against the <text string> in order to be returned as part of the result set.

Operator	Relationship
Text comparison operators:	
∼=	Is identically equal to
?=	Includes as a substring
∼<	Is less than (in ASCII order)
∼>	Is greater than (in ASCII order)
∼!=	Is not identically equal to
?!=	Does not include as a substring
∼<=	Is not greater than (in ASCII order)
∼>=	Is not less than (in ASCII order)
Numerical comparison operators:	
=	Is equal to
<	Is less than
>	Is greater than
!=	Is not equal to
<=	Is not greater than
>=	Is not less than

These tests are valid for any STAR application. *Numerical operators* permit comparison of the numerical values implied by the returned data-value strings. Recall from Chapter 2.1 that data values in STAR are specified only as character strings. Casting to different types may be performed by specific applications, but is *not* defined for arbitrary STAR applications. Nevertheless, *Star_Base* recognizes that a majority of STAR applications will in fact specify numeric types, and therefore allows for numerical comparisons based on interpretations of certain value strings according to the conventions adopted by CIF for the **numb** data type (Section 2.2.7.4.7.1). Such values may be given as integers, real numbers or in scientific notation.

5.2.3.4. The *Star_Base* branching request

Both conditional and data requests will retrieve matching data items wherever they may be found in the input file; the scope of the query in both cases is the entire file.

The top-level query type supported by *Star_Base*, the *branching request*, allows selection of sub-requests based on the results of prior tests, and also allows the narrowing or expansion of the scope of a request. The effect is to permit extensive control over the selection of data matching complex conditions. It is this which gives *Star_Base* the power of a database query language.

Note again that the user will in general need prior knowledge of the arrangement of data items within a STAR File in order to compose meaningful requests; *Star_Base* is agnostic about the organization and structure of the contents of a data file and will simply return exactly those data items and their context that match the specified conditions.

A branching request takes the following form:

if_ <condition> <branch_request>
  [**else_** <branch_request>]
  [**unknown_** <branch_request>]
endif_

The <condition> has exactly the same form as a conditional request, but does not return data to the calling process. It returns only a logical value that is used to determine which branch to evaluate. This logical return value may be TRUE if the condition is satisfied, UNKNOWN if the condition is not satisfied because there was no occurrence of a requested data name within the current