**code promotion:**

1.tag create:

air tag create project-only myproj-2006-12-02-release \

2.Creating a save file at the command line

Run air object save with the -exact-tag tag option to save objects listed in the specified tag from a source technical repository into a save file.

You can then use the air object

3. air object load command

4. that air promote load does

----> air sandbox diff

Displays the differences between two graphs, two plans, or two text files. For information about the arguments common to all air commands, see “Command-line syntax of air commands”.

air sandbox diff

----> air lock break

Breaks a user’s lock on a given repository object, on all the objects in a project, or on all locks. This command is branch-specific. For information about the arguments common to all air commands, see “Command-line syntax of air commands”.

air lock break {[-user username] ... | [-object object

----> air lock release

Unlocks a lock on a single object, on all objects in a project, or on all objects in all projects within a technical repository. This command is branch-specific. For information about the arguments common to all air commands, see “Command-line syntax of air commands”.

air lock release {[-object object

----> air lock show

Displays a listing of locks for a given user, object, project, or all locks in the technical repository. This command is branch-specific. For information about the arguments common to all air commands, see “Command-line syntax of air commands”.

air lock show {[-user username] ... | [-object object] ... | [-project project] ... | [-all]}

----> To see the revision history of the tagged versions of graph JoinCustomers.mp, type:

air object versions -quiet /Projects/lesson/mp/JoinCustomers.mp

The output might look like the following. Note that when ‑quiet is specified, the command displays only tagged versions of the graph. You see version 738 three times, because it has three different tags at that version.

Version Date User Tag Comment

738 2005-05-16 14:55:50 asmith tag1

738 2005-05-16 14:55:50 asmith tag2

738 2005-05-16 14:55:50 asmith tag3

748 2005-05-16 15:10:56 asmith tag4

---> air project export

Copies (or checks out) one or more projects or files from a technical repository to a sandbox. The sandbox directory (or directories) is created if necessary. This command is branch-specific. For information about the arguments common to all air commands, see “Command-line syntax of air commands”.

air project export project-path

air project export project-path

-basedir sandbox-root-dir

----> air project import

Copies sandbox files into an existing technical repository project. This process is called checkin. This command is branch-specific. For information about the arguments common to all air commands, see “Command-line syntax of air commands”.

air project import project-path

-basedir basedir

Examples

This section includes simple examples of how to run m\_eval from the command line. For examples of how to use this utility in shell scripts, see “Script examples”.

Evaluating a simple expression

In this example, m\_eval evaluates a simple arithmetic expression:

$ m\_eval '2+3'

5

Evaluating multiple expressions

In this example, m\_eval evaluates the result of a call to string\_replace:

$ m\_eval '1<<5' 'string\_replace("this is a test","test","success")'

32

"this is a success"

Evaluating a cast expression

In this example, m\_eval mimics converting values from one data type to another by evaluating a cast expression.

Example 1

In this example, m\_eval evaluates

Examples

Printing data from a file

In this example, m\_dump prints the data in xyz.dat as interpreted by the metadata in xyz.dml:

$ m\_dump xyz.dml xyz.dat

Printing particular records

In this example, m\_dump prints records with numbers 5, 6, 7, 8, and 9:

$ m\_dump xyz.dml xyz.dat -start 5 -end 9

Printing an expression result

In this example, m\_dump prints the result of evaluating the expression a+b for each record in the file abc.dat:

$ m\_dump abc.dml abc.dat -no-print-data -print "a+b"

Printing data from a remote file

In this example, m\_dump prints data from the file jan.dat, which resides on mktg.bigbiz.com:

$ m\_dump cust.dml file://mktg.bigbiz.com/custdata/jan.dat

Printing data from a multifile

In this example, m\_dump prints data from the multifile tempfile.dat:

$ m\_dump loader.dml mfile:mfs8/tempfile.dat

Printing data in a multifile partition

In this example, m\_dump prints the data in partition 3 of the multifile tempfile.dat:

$ m\_dump loader.dml mfile:mfs8/tempfile.dat -partition 3

Printing a table

-show-partial Prints corrupt data or incomplete record formats. For more information, see “Printing data that is corrupt or does not match the record format”.

-align Optional. Use for interval lookups that were created using Co>Operating System versions earlier than 2.15.3.r28, on platforms that align the data structure sizes to an 8-byte boundary

Data base:

\* The database-specific parameters in separate tables, one for each supported DBMS

For simple examples of how to use the database components, see “Database components tutorial”.

The database components are as follows:

\* BEGIN TRANSACTION marks the start of a transaction group in a transactional graph.

\* CALL STORED PROCEDURE calls a stored procedure that returns multiple result sets. The stored procedure can also take parameters.

\* END TRANSACTION marks the end of a transaction group in a transactional graph. It buffers records until the transaction commits or aborts.

\* INPUT TABLE unloads records from a database into a graph, allowing you to specify as the source either a database table, or an SQL statement that selects records from one or more tables.

\* JOIN WITH DB joins records from the flow or flows connected to its input port with records read directly from a database, and outputs new records containing data based on, or calculated from, the joined records.

\* MULTI UPDATE TABLE executes multiple SQL statements for each input record.

\* OUTPUT TABLE loads records from a graph into a database, letting you specify the records’ destination either directly as a single database table, or through an SQL statement that inserts records into one or more tables.

\* RUN SQL executes SQL statements in a database and writes confirmation messages to the log port.

\* TRUNCATE TABLE deletes all the rows in a database table, and writes confirmation messages to the log port.

\* UPDATE TABLE executes UPDATE, INSERT, or DELETE statements in embedded SQL format to modify a table in a database, and writes status information to the log port.

Transform ---->

Transform components

The transform components modify or manipulate records by using one or several transform functions. Modifications could include reformatting data, filtering data, or merging separate data streams:

\* COMBINE does any of the following: restores hierarchies of data flattened by the SPLIT component, creates a single output record by joining multiple input streams, or denormalizes vectors (including nested vectors).

\* DEDUP SORTED separates one specified record in each group of records from the rest of the records in the group. Dedup Sorted requires grouped input.

\* FILTER BY EXPRESSION filters records according to a specified DML expression.

\* FUSE applies a transform to corresponding records from each input flow. The transform is first applied to the first record on each flow, then to the second record on each flow, and so on. The result of the transform is sent to the out port.

\* JOIN performs inner, outer, and semi-joins on multiple flows of records. Join can process either sorted or unsorted input. When processing unsorted input, Join maximizes performance by loading input records into main memory.

\* MATCH SORTED combines and performs transform operations on multiple flows of records. Match Sorted requires grouped input.

\* MULTI REFORMAT changes the record format of records flowing between from one to 20 pairs of in and out ports by dropping fields, or by using DML expressions to add fields, combine fields, or transform the data in the records.

\* NORMALIZE generates multiple output records from each input record; you can specify the number of output records, or the number of output records can depend on a field or fields in each input record. Normalize can separate a record with a vector field into several individual records, each containing one element of the vector. It uses a multistage transform.

\* REFORMAT changes the record format of records by dropping fields, or by using DML expressions to add fields, combine fields, or transform the data in the records.

\* ROLLUP generates records that summarize groups of records. Rollup gives you more control over record selection, grouping, and aggregation than Aggregate. Rollup can process either grouped or ungrouped input. When processing ungrouped input, Rollup maximizes performance by keeping intermediate results in main memory. It uses a multistage transform.

\* SCAN generates a series of cumulative summary records — such as successive year-to-date totals — for groups of records. Scan can process either grouped or ungrouped input. When processing ungrouped input, Scan maximizes performance by keeping intermediate results in main memory. It uses a multistage transform.

\* SCAN WITH ROLLUP performs the same operations as Scan; it generates a summary record for each input group. It uses a multistage transform.

\* SPLIT does any of the following: flattens hierarchical data, selects a subset of fields from the data, normalizes vectors (including nested vectors), or retrieves multiple distinct outputs from a single pass through the data.

Related topics

\* About transform functions

\* Overview of components by category

Partition components

---------------------

Partitioning components

There are two types of partitioning components: partition components, which create data partitions, and departition components, which combine multiple flow partitions or multiple straight flows into a single flow:

\* BROADCAST arbitrarily combines all the records it receives into a single flow and writes a copy of that flow to each of its output flow partitions.

\* CONCATENATE appends multiple flow partitions of records one after another.

\* GATHER combines records from multiple flow partitions arbitrarily.

\* INTERLEAVE combines blocks of records from multiple flow partitions in round-robin fashion.

\* MERGE combines records from multiple flow partitions that have all been sorted according to the same key specifier, and maintains the sort order.

\* PARTITION BY EXPRESSION distributes records to its output flow partitions according to a specified DML expression.

\* PARTITION BY KEY distributes records to its output flow partitions according to key values.

\* PARTITION BY PERCENTAGE distributes a specified percent of the total number of input records to each output flow.

\* PARTITION BY RANGE distributes records to its output flow partitions according to the ranges of key values specified for each partition.

\* PARTITION BY ROUND-ROBIN distributes records evenly to each output flow.

\* PARTITION WITH LOAD BALANCE distributes records to its output flow partitions, writing more records to the flow partitions that consume records faster.

Related topics

valdiate components

------------------------

Validate components

The validate components test, debug, and check records, and produce data for testing graphs:

\* CHECK ORDER tests whether records are sorted according to a key specifier.

\* COMPARE CHECKSUMS compares two checksums generated by Compute Checksums. Typically, you use Compare Checksums to compare checksums generated from two sets of records, each set computed from the same data by a different method, in order to check the correctness of the records.

\* COMPARE RECORDS compares records from two flows one by one.

\* COMPUTE CHECKSUM calculates a checksum for records.

\* CREATE DATA produces a series of one or more output records.

\* GENERATE RANDOM BYTES generates a specified number of records, each consisting of a specified number of random bytes. Typically, the output of Generate Random Bytes is used for testing a graph. For more control over the content of the records, use Generate Records.

\* GENERATE RECORDS generates a specified number of records with fields of specified lengths and types. You can let Generate Records generate random values within the specified length and type for each field, or you can control various aspects of the generated values. Typically, you use the output of Generate Records to test a graph.

\* REPAIR INPUT tries to correct malformed records.

\* VALIDATE RECORDS separates valid records from invalid records.

Related topics

string functions :

Previous Next

Co>Operating System Help > Developer reference > DML (Data Manipulation Language) > The DML core functions > DML core functions reference > String functions: Alphabetical list

--------------------------------------------------------------------------------

String functions: Alphabetical list

Function Description

char\_string Returns a one-character native string that corresponds to the specified character code.

decimal\_lpad Returns a decimal string of the specified length or longer, left-padded with a specified character as needed.

decimal\_lrepad Returns a decimal string of the specified length or longer, left-padded with a specified character as needed and trimmed of leading zeros.

decimal\_strip Returns a decimal from a string that has been trimmed of leading zeros and non-numeric characters.

edit\_distance Returns the minimum number of single-character insertions, deletions, and substitutions needed to turn a specified string into another.

ends\_with Returns 1 (true) if a string ends with the specified suffix; 0 (false) otherwise.

hamming\_distance Returns the minimum number of single character substitutions needed to turn a specified string into another string.

is\_blank Tests whether a string contains only blank characters.

is\_bzero Tests whether an object is composed of all binary zero bytes.

make\_byte\_flags Returns a vector of flags that indicates whether a character occurs in a specified string.

re\_get\_match Returns the first string that matches a regular expression.

re\_get\_matches Returns a vector of strings that match a regular expression containing up to 9 sets of capturing parentheses.

re\_get\_range\_matches Returns a vector that describes the position and length of a string that matches a specified regular expression containing up to 9 sets of capturing parentheses.

re\_index Returns the index of the first character of a string matching a regular expression.

re\_match\_replace Returns a new string after replacing any substrings matching the supplied pattern.

re\_replace Returns a string after replacing all substrings matching a regular expression.

re\_replace\_first Returns a string after replacing the first substring that matches a specified regular expression.

re\_split Splits a string into pieces using a specified regular expression.

re\_split\_no\_empty Behaves like re\_split, but excludes empty strings from its output.

starts\_with Returns true if the string starts with the supplied prefix.

string\_char Returns the character code of a specific character in a string.

string\_cleanse Replaces encoding errors with a specified string.

string\_cleanse\_euc\_jp Replaces encoding errors with a specified string and returns an output string in the euc\_jis string('\0') character set.

string\_cleanse\_shift\_jis Replaces encoding errors with a specified string and returns a string in the shift\_jis string('\0') character set.

string\_compare Returns a number representing the result of comparing two strings.

string\_concat Concatenates multiple string arguments and returns a NULL-delimited string.

string\_convert\_explicit Converts a string from one character set to another, replacing inconvertible characters with a specified string.

string\_downcase Returns a string with any uppercase letters converted to lowercase.

string\_filter Compares the contents of two strings and returns a string containing characters that appear in both of them.

string\_filter\_out Returns characters that appear in one string but not in another.

string\_from\_hex Returns an integer-prefixed native string from a series of hexadecimal characters that represent bytes.

string\_han\_to\_zen\_hiragana Converts Japanese half-width (hankaku) katakana to full-width (zenkaku) hiragana.

string\_han\_to\_zen\_katakana Converts Japanese half-width (hankaku) katakana to full-width (zenkaku) katakana.

string\_index Returns the index of the first character of the first occurrence of a string within another string.

string\_is\_alphabetic Returns 1 if a specified string contains all alphabetic characters, or 0 otherwise.

string\_is\_numeric Returns 1 if a specified string contains all numeric characters, or 0 otherwise.

string\_join Concatenates vector string elements into a single string.

string\_length Returns the number of characters in a string.

string\_like Tests whether a string matches a specified pattern.

string\_lpad Returns a string of a specified length, left-padded with a given character.

string\_lrepad Returns a string of a specified length, trimmed of leading and trailing blanks and left-padded with a given character.

string\_lrtrim Returns a string trimmed of leading and trailing blank characters.

string\_ltrim Returns a string trimmed of leading blank characters.

string\_pad Returns a right-padded string.

string\_prefix Returns a substring that starts at the beginning of the parent string and is of the specified length.

string\_repad Returns a string of a specified length trimmed of any leading and trailing blank characters, and then right-padded with a given character.

string\_replace Returns a string after replacing one substring with another.

string\_replace\_first Returns a string after replacing the first occurrence of one substring with another.

string\_rindex Returns the index of the first character of the last occurrence of a string within another string.

string\_split Returns a vector consisting of substrings of a specified string.

string\_split\_no\_empty Behaves like string\_split, but excludes empty strings from its output.

string\_substring Returns a substring of a string.

string\_split\_quoted Behaves like string\_split, but enables you to specify quoting and escaping options as in the READ SEPARATED VALUES component.

string\_suffix Returns a substring of a specified length that ends at the end of the parent string.

string\_to\_hex Returns a string of hexadecimal digits (0-9 a-f) in which each pair of characters represents the value of one byte of the input void (or string) object.

string\_trim Returns a string trimmed of trailing blank characters.

string\_truncate\_explicit Converts a string from one character set to another, truncating it if necessary and replacing inconvertible characters with a specified string.

string\_upcase Returns a string with any lowercase letters converted to uppercase.

test\_characters\_all Tests a string for the presence of ALL characters in another string.

test\_characters\_any Tests a string for the presence of ANY characters in another string.

to\_xml Returns an XML-formatted document from the value of a DML object.

unicode\_char\_string Returns the Unicode string represented by the specified character code.

url\_decode\_escapes Returns a string, replacing URL hexadecimal escape sequences with the corresponding ASCII characters.

url\_encode\_escapes Returns a string after introducing URL hexadecimal escape sequences.

--------------------------------------------------------------------------------

Previous Next

Copyright © 1997-2010, Ab Initio, Confidential and Proprietary. All rights reserved.

Popup

vectors:

------------

Previous Next

Co>Operating System Help > Developer reference > DML (Data Manipulation Language) > The DML core functions > DML core functions reference > Vector functions: Alphabetical list

--------------------------------------------------------------------------------

Vector functions: Alphabetical list

Function Description

make\_constant\_vector Returns a vector of a specified number of elements, each of which has the same value.

vector\_append Appends an object to a vector and returns the resulting vector.

vector\_avg Returns the average of all elements in a vector.

vector\_bsearch Returns the index of a vector element whose key field matches the key value of a specified element.

vector\_bsearch\_all Returns a pair of values, representing the indexes of the first and last vector elements whose key fields match the key field of a specified element.

vector\_concat Adds all elements of a vector to the end of another vector.

vector\_difference Given two vectors, returns all elements from the first vector that are not in the second vector.

vector\_intersection Returns a new vector that consists of all elements common to the first and second vectors, according to a specified key.

vector\_max Returns the maximum value of all the fields in a vector.

vector\_min Returns the minimum value of all the fields in a vector.

vector\_product Returns the product of all elements in a vector.

vector\_rank Returns a vector of integers showing the new positions the elements of the original vector would have if they were rearranged according to the sort order of the specified key field.

vector\_select Returns a new vector containing all elements of a vector that are equal to a specified element, according to a specified key.

vector\_slice Returns a new vector consisting of the subrange of elements in a specified vector.

vector\_sort Uses a stable sorting algorithm to return a new vector that consists of the input vector’s elements, sorted by a specified key.

vector\_sort\_dedup\_first Returns a new vector consisting of a sorted version of the input vector, which includes only the first element in each group of elements whose key matches the specified key.

vector\_sort\_dedup\_last Returns a new vector consisting of a sorted version of the input vector, which includes only the last element in each group of elements whose key matches the specified key.

vector\_stable\_sort Uses a stable sorting algorithm to return a new vector that consists of the input vector sorted by key.

vector\_stdev Returns the sample standard deviation of the elements in a vector.

vector\_sum Returns the sum of all elements in a vector.

vector\_union Returns a new vector containing all elements in the specified vector arguments, except for duplicates that match values for a specified key.

--------------------------------------------------------------------------------

Previous Next

Copyright © 1997-2010, Ab Initio, Confidential and Proprietary. All rights reserved.

Popup

**UNIX INTERVIEW**

sed -n 1,4p test ---> to print range of records in file

100,vvvv,3000,hyd,tnl

200,babu,400,chni,gnt

653,chavali,399.vijj,chvl

900,deep,bplt.4000,vvvv

sed -n '1!p' test ---> to print except first record using SED command

sed -n '$p' test ---> to print last record only using SED command

sed -n '-e1p' '-e4p' test ---> to print specifi record in file

sed -n '1!d' test --->to delete except first record.

sed '/^$/d' test ---> to delete empty lines in file.

sed '1s/vvvv/jjjjj/' test ---> to replace string only first occurence in file.

sed 's/vvvv/jjjjj/g' test ---> to replace string all occurence in file.

sed '1,4s/vvvv/jjjjj/' test ---> to replace string specific records in file.

grep -c "chavali" test ---> total word count in file.

1

grep -w "vvvv" test ----> search the matching word in file.

100,vvvv,3000,hyd,tnl

900,deep,bplt.4000,vvvv

200,vvvv,40000,hyd,tnl

grep -o "vvvv" test ---> to display only matching patern

vvvv

vvvv

vvvv

grep -n "vvvv" test ----> to display line number matching pattern in file.

1:100,vvvv,3000,hyd,tnl

4:900,deep,bplt.4000,vvvv

5:200,vvvv,40000,hyd,tnl

grep -v "vvvv" test -----> inverse (Except words) of the pattern matching in file.

200,babu,400,chni,gnt

653,chavali,399.vijj,chvl

500,kumar,989,bapatla,9303

grep "^100" test ----> matching the line start with word.

100,vvvv,3000,hyd,tnl

grep -e "chavali" test ---> grep the finding words in file.

653,chavali,399.vijj,chvl

100,vvvv,3000,hyd,tnl

900,deep,bplt.4000,vvvv

200,vvvv,40000,hyd,tnl

cut -d "," -f1,2 test --> to print only specific fileds in file.

100,vvvv

200,babu

653,chavali

900,deep

200,vvvv

500,kumar

cut -c 2-4 test ---> to print specific character to specific character in file.(2-4)

00,

00,

53,

00,

00,

cut -b 1,3 test ---> to print the specific character in file(1c and 3character)

10

20

63

90

20

50

cut -b -3 test or cut -c -3 test ---> to print the specified lend of character in file

100

200

653

900

200

awk '/hyd/{print}' test ---> to print pattern match in file.

100,vvvv,3000,hyd,tnl

200,vvvv,40000,hyd,tnl

awk '{print $1,$3}' test ----> to print columns in file.

100,vvvv,

200,babu,

653,chava,

900,deep,

awk '{print NR,$0}' test ---display the line numbers.

1 100,vvvv,3000,hyd,tnl

2 200,babu,400,chni,gnt

3 653,chavali,399.vijj,chvl

awk '{print $1, $NF}' test ---> to display first and last column in file.

awk 'NR==1, NR==3 {print NR, $0}' test ---> to display the line number with specified range records in file.

1 100,vvvv,3000,hyd,tnl

2 200,babu,400,chni,gnt

3 653,chavali,399.vijj,chvl

awk 'END {print NR }' test ---> count of records in file.

awk '{print "START"$0}' test ----> to add START for all line for all

START100,vvvv,3000,hyd,tnl

START200,babu,400,chni,gnt

awk '{print $0"END"}' test to add END for line

100,vvvv,3000,hyd,tnlEND

200,babu,400,chni,gntEND

653,chavali,399.vijj,chvlEND

900,deep,bplt.4000,vvvvEND

find . /abinitio -name test\_bkp ---> to find file in particular directory.

./abinitio/test\_bkp

find -type f -name "t\*" ---> to find the files NAME

./abinitio/test\_bkp

./test

./test1

find the duplicate records in table

select cust\_name,order\_name,count(\*) from orders group by cust\_name,order\_name having count(\*)>1