NAME
      rasqlinsert – write **argus(8)** data into mysql database tables.

SYNOPSIS
      rasqlinsert [**raoptions**] [**--** *filter-expression*]

DESCRIPTION
      Rasqlinsert writes **argus** data into a mysql database.

      The principal function of **rasqlinsert** is to insert and update flow data attributes, into a MySQL database table. Using the same syntax and strategies for all other ra* programs, **rasqlinsert** creates databases and database tables, based on the print specification on the either the command-line or the .rarc file.

      The concept is that where a ra* program would print fields to standard out in ascii, **rasqlinsert** will insert those fields into the database as attributes. The flow key, as defined by the "-m fields" option, provides the definition of any keys that would be used in the schema. A "-m none" option, will remove the use of any DBMS keys for inserted data, and is the method to use when inserting streaming, unprocessed, primitive argus data into a database table.

      The schema is important for database utility and performance. You can use MySQL querys against the attributes that you insert into the tables, such searching and sorting on IP addresses, time, packet counts, etc.... While **rasqlinsert** does not limit you to the number of attributes (columns) per record you provide, the RDBMS performance will quide you as to how many fields are useful.

      Rasqlinsert by default, includes the actual binary argus 'record' in the schema, and inserts and updates the binary record when needed. This enables a large number of fucnctions that extend beyond simple RDBMS schema's that are useful. Adding the 'record' is expensive, and some will elect to not use this feature. This can be controlled using the option '-s -record' as a print field option in the standard **ra.1** command line. When the 'record' attribute is present, **rasql.1** can read the records directly from the database, to provide additional processing on the database table contents.

      When keys are used, the database will enforce that any insertions meet the relaitional requirements, i.e. that the keys be unique. This requirement demands a sense of caching and key tracking, which **rasqlinsert** is specifically designed to provide.

      Rasqlinsert by default, will append data to existing tables, without checking the schema for consistency. If your schema has keys, and you attempt to append new records to an existing table, there is a high likely-hood for error, as **rasqlinsert** will attempt to insert a record that collides with an existing flow key. Use the "-M cache" option to cause **rasqlinsert** to reference the table contents prior to aggregation and insertion.

      The binary data **rasqlinsert** inserts by default, is read using **rasql**.

RASQLINSERT SPECIFIC OPTIONS
      Rasqlinsert, like all ra based clients, supports a number of **ra options** including filtering of input argus records through a terminating filter expression. **Rasqlinsert(1)** specific options are:

      –M cache
          This causes **rasqlinsert** to use the database table as its persistent cache store. This mechanism is used to control memory use when dealing with large amounts of data and flow keys.

      –M drop
          This causes **rasqlinsert** to drop any pre-existing database table that has the same name as the target table name, on startup.

INVOCATION

This invocation writes aggregated **argus(8)** data from the *file* into a database table. The standard 5-tuple fields, 'saddr daddr proto sport dport' are used as keys for each entry. **rasqlinsert** will aggregate all the data prior to inserting the data into the database:

    rasqlinsert -r file -w mysql://user@localhost/db/table

Because aggregation can require a lot of memory, **rasqlinsert** provides an option '-M cache' to have **rasqlinsert** use the database table as the persistent cache store for the aggregation. With this example, the standard 5-tuple fields, **rasqlinsert** will aggregate data over short spans of time as it reads the data from the file, and then commit the data to the database. If additional data arrives that matches that unique flow, **rasqlinsert** will fetch the entry from the database, aggregate, and then update the data entry in the database.

    rasqlinsert -M cache -r file -w mysql://user@localhost/db/table

rasqlinsert can provide the same function for streaming data read directly from an argus data source. This allows **rasqlinsert** to reassemble all status records for an individual flow, such that the resulting table has only a single entry for each communciation relationship seen.

    rasqlinsert -M cache -S argus -w mysql://user@localhost/db/table

This invocation writes **argus(8)** data from the *file* into a database table, without aggregation, by specifying no relational key in the data.

    rasqlinsert -m none -r file -w mysql://user@localhost/db/table

This invocation writes **argus(8)** data from the *stream* into a database table, without modification.

    rasqlinsert -m none -S argus -w mysql://user@localhost/db/table

This invocation writes **argus(8)** data from the *stream* into a daily database table, without modification. **rasqlinsert** will generate table names based on time and insert its data relative to the timestamps found in the flow records it processes. In this specific example, "-M time 1d" specifies daily tables.

    rasqlinsert -m none -S argus -w mysql://user@localhost/db/table_%Y_%m_%d -M time 1d

COPYRIGHT

Copyright (c) 2000-2014 QoSient. All rights reserved.

SEE ALSO

    rasql(1), ra(1), rarc(5), argus(8),

AUTHORS

Carter Bullard (carter@qosient.com).