

ShadowDB API

See [Term notation](#) if you are unfamiliar with nuprl term textual representation.

API for ShadowDB/Nuprl :

```
Database Commands: <cmd>

create{<table-name>:s}() -> ()

delete{<table-name>:s}() -> ()
  - delete table

put{<table-name>:s}(<key>;<value>) -> ()

get{<table-name>:s}(<key>) -> <value>

remove{<table-name>}(<key>) -> ()
  - remove key from table.

iterate{<table-name>:s,<handle>:s}
  -> ([some(<key>,<value>) | some()])
  <handle> : if null string then start new iteration,
            otherwise continue with iteration indicated by <handle>
  some() return indicates end
  ? arbitrary behavior if table updated during iteration.

filter{<table-name>:s,<new-table-name>:s,<regex>:s}(<pattern>) -> {<count>:n}
  * create new table with by matching table values against <pattern>

<pattern>
  : AND(<pattern>,<pattern>)
  | OR(<pattern>,<pattern>)
  | NOT(<pattern>)
  | EXISTS(<term>;<pattern>)
  | MATCH(<patternterm>)
  | SLOT()
  | EQUALS(<term>)
  * use operator of <term> of EXISTS to extract embedded list
  to check <pattern> against.
  * might want ALL(<term>;<pattern>)

TODO: <patternterm> is a term to be matched, <pattern> is an expression to be
interpreted. A PATTERN(<pattern>) in a term switches back to interpret mode.
Currently java implementation is using MATCH for PATTERN at the moment

A <patternterm> is a term with embedded SLOT() or MATCH(<pattern>) terms.
Anything matches a slot. A MATCH(<pattern>) causes recursion of match operation
with new <pattern>. Otherwise, the MATCH proceeds as long as the <patternterm>
is identical to the instance.

There is a pitfall here in that the <patternterm> can not contain a MATCH opid
intended to match a MATCH opid in the instance as it will be interpreted instead
of matched. Could be fixed with quoting mechanism, but it is unlikely to be a problem.

Future addition:
Enhance patterns with variables:

<patternterm>
  : <term>
  | MATCHVAR{<var>:s}          // should not occur in filter commands.
  | PATTERN(<pattern>)
  | <operator>(<patternterm> list)

<pattern>
  : AND(<pattern>,<pattern>)
  | OR(<pattern>,<pattern>)
  | NOT(<pattern>)
  | EXISTS(<term>;<pattern>)
```

```

| MATCH(<patternterm>)
| EQUALS(<term>)
| PARAMETERPATTERN{<i>:n, <regex>:s, <type>:s}

<ipatternterm>      : <term>
| MATCHVAR{<var>:s}
| <operator>(<ipatternterm> list)

```

to allow map:

```

mapfilter{<name>:s,<regex>:s}(<pattern>; <vpatternterm>; <vpatternterm>)
  -> {<count>:n}
  * first ipatternterm is to instantiate key
  * second instantiates value
  * INDEX() as key pattern indicates using ints as indices.

```

Protocol

```

<req>      : !req{<seq>:n}(<cmd>)
<rsp>      : !rsp{<seq>:n}(<result>)
<notice>   : !notice{<seq>:n}(<message>)
  * no response

<result>   : !value()( <term>)
| !ack()
| !fail(<message>)

```

FTTB : expect straightforward one-to-one sequential Nuprl sends <req> and gets <rsp>.

Future :

- nuprl sends sequence of reqs without waiting for response.
 - * possibly add form of requests that don't expect response.
- add call backs from shadowdb to nuprl.
- maybe add partial response to return sequence of result terms