proSQLite: Exploring file based databases via an SQLite interface

Sander Canisius, Nicos Angelopoulos and Lodewyk Wessels

overview

proSQLite a prolog interface to SQLite.

. . . modelled after SWI's ODBC interface
. . . plus database tables as prolog facts

why SQLite

- bioinformatics
  - applications need to access large datasets
- application back-end
  - Firefox : 13 .sqlite files
  - OSX . . .
- characteristics
  - server-less
  - zero configuration
  - well suited to web-deployment
  - OS agnostic

overall architecture

low-level: core functionality

mid-level: introspection + formatted queries

top-level: importing tables as predicates

low-level

name/arity	moded arguments
sqlite_connect/2	+File, ?Conn
sqlite_connect/3	+File, ?Conn, +Opts
sqlite_disconnect/1	+Conn
sqlite_current_connection/1	-Conn
sqlite_default_connection/1	-Conn
sqlite_query/2	+SQL, -Row
sqlite_query/3	+Conn, +SQL, -Row

connection options

`alias(Alias)`
register a connection alias.

`as_predicates(AsPred)`
map each sqlite table to a prolog predicate

`at_module(AtMod)`
defines the module for table predicates

`exists(Exists)`
for `false` do not throw an error for missing file

example

table name	population	columns
secondary_accessions	286525	secondary_accession
		primary_accession
identifier_mapping	3044651	uniprot_accession
		identifier_type
		target_identifier

```
sqlite_connect('uniprot.sqlite', uniprot)
```

formated queries & introspection

name/arity	moded arguments
sqlite_format_query/3	+Conn, +SQL, -Row
sqlite_current_table/2	+Conn, -Row
sqlite_table_column/3	+Conn, ?Table, -Column
sqlite_table_count/3	+Conn, +Table, -Count

tables as predicates

```
H =..[Name|Args],
B = sqlite_holds(C,Name,Arity,Clmns,Args),
<Mod>:assert((H :- sqlite:(B) )).


?-
  Opt = as_predicates(true),
  sqlite_connect('uniprot.sqlite',uniprot,Opt),
  findall(A,secondary_accessions(A,_),As),
  length(As, Len).

  As = ['A0A111', 'A0A112', 'A0A113', 'A0A131'|...
  Len = 286525.
```

multi-arity predicates

```
?-

  Opts = [as_predicates(true),arity(pallete)]
  sqlite_connect('uniprot.sqlite',uniprot,Opt),
  findall(A,secondary_accessions(id=A),As),
  length(As, Len).

  As = ['A0A111', 'A0A112', 'A0A113', 'A0A131'|..
  Len = 286525.
```

db_facts

```
db_create( simple,
     cited_by(  pubmed_id+bigint,
                ret_date-date,citer+bigint) ) )

db_assert( cited_by(123, Date, 321)

db_holds(
 cited_by(pubmed_id=Pid,ret_date=D1,citer=Cit)
)

db_retractall( cited_by(pubmed_id=120) )
```

piece-meal prolog bioinformatics

r..eal	Swi/Yap <-> R interface
pubmed	access pumed citation records
proSQLite	Swi/Yap <-> SQLite interface
rcy	graph visualisation
depth search	depth limited reachability

versus the more holistic

blip : `http://www.blipkit.org/`

availability

for Swi (/Yap) as open source from:

`http://bioinformatics.nki.nl/~nicos/sware/prosqlite`

... also via git

... and

?- pack_install(prosqlite) .

bottom-line

simple, yet effective
common design, makes porting trivial
no need to re-invent the wheel

future work
- integrate with ODBC
- allow feature selection notation

Sander Canisius core C-code
Jan Wielemaker fixes and packaging help (+ODBC !)
Nicos extensions and fixes to core, doc and Prolog code

demo

. . . time