Database JDBC from Clojure

* Real work may involve a query
  to a database

Steps:

1. Create a project
2. Get the driver for your db
3. define your db
4. connect

```
]  lein new db-test
Created new project in: /Users/.../

]  cd db-test/
]  lein deps
Copying 1 file to /Users/.../db-test/lib

]  cp /Users/.../nzjdbc.jar lib/
]  lein repl
REPL started; server listening on localhost
port 51245
```

```
 Possible drivers...
 [org.xerial/sqlite-jdbc "3.7.2"] ; SQLite3
[mysql/mysql-connector-java "2.0.14"]   ; MySQL
[postgresql/postgresql "8.4-701.jdbc4"] ; PostgreSQL

 In our case, Netezza.
```

```
]  cp /Users/.../nzjdbc.jar lib/
]  lein repl
REPL started; server listening on localhost
port 51245

user=> (require '[clojure.contrib.sql :as sql])
nil
user=> (System/getenv "NZ_USER")
"aaelony"
```

```
(let [ db-host "10.18.99.120"
       db-port "5809"
       db-name "reporting_db"
       db-info {:classname "org.netezza.Driver"
                :subprotocol "netezza"
                :subname (str "//" db-host ":" db-port
                               "/" db-name)
                :user (System/getenv "NZ_USER")
                :password (System/getenv "NZ_PASSWORD" )}]
        (sql/with-connection db-info
           (sql/with-query-results rs
             ["select * from dm_locale"]
             (dorun (map #(println %) rs)))))
```

Example Output

```
user=> (let [ db-host "10.18.99.120"
       db-port "5809"
       db-name "reporting_db"
       db-info {:classname "org.netezza.Driver"
                :subprotocol "netezza"
                :subname (str "//" db-host ":" db-port "/" db-name)
                :user (System/getenv "NZ_USER")
                :password (System/getenv "NZ_PASSWORD" )}]
        (sql/with-connection db-info
           (sql/with-query-results rs ["select * from dm_locale"]
             (dorun (map #(println %) rs)))))
...elisions...
{:locale_id 5, :locale en_GB, :locale_descriptions Great Britain, :locale_site .UK, :created_at #<Date
2011-09-26>, :updated_at nil}
{:locale_id 13, :locale pt_BR, :locale_descriptions Brazil, :locale_site .BR, :created_at #<Date
2011-09-26>, :updated_at nil}
{:locale_id 2, :locale en_AU, :locale_descriptions Australia, :locale_site .AU, :created_at #<Date
2011-09-26>, :updated_at nil}
{:locale_id 4, :locale en_CA, :locale_descriptions Canada, :locale_site .CA, :created_at #<Date
2011-09-26>, :updated_at nil}
...elisions...
nil
user=>
```

```
user=> (use 'clojure.pprint)
nil
```

```
user=> (pprint (keys (ns-publics
'clojure.contrib.sql)))
(set-rollback-only
 update-values
 drop-table
 find-connection
 with-query-results
 insert-records
 connection
 transaction
 is-rollback-only
 with-connection
 insert-values
 do-commands
 create-table
 do-prepared
 insert-rows
 delete-rows
 update-or-insert-values)
nil
user=>
```

Lots of other functions in clojure.contrib.sql.

In Clojure 1.3, this is refactored into clojure.java.jdbc