On this page

Insert

SQL

```
-- A :result value of :n below will return affected row count:
-- :name insert-character :! :n
-- :doc Insert a single character
insert into characters (name, specialty)
values (:name, :specialty)

-- :name insert-characters :! :n
-- :doc Insert multiple characters with :tuple* parameter type
insert into characters (name, specialty)
values :tuple*:characters
```

Clojure

```
(characters/insert-character-sqlvec
 {:name "Westley", :specialty "love"})  ;;=>
["insert into characters (name, specialty)
  values (?, ?)"
,"Westley"
,"love"]

(characters/insert-character db
  {:name "Westley", :specialty "love"})  ;;=>
1

(characters/insert-character db
  {:name "Buttercup", :specialty "beauty"})  ;;=>
```

```
1

(characters/insert-characters-sqlvec
 {:characters
  [["Vizzini" "intelligence"]
   ["Fezzik" "strength"]
   ["Inigo Montoya" "swordmanship"]]})  ;;=>
["insert into characters (name, specialty)
  values (?,?),(?,?),(?,?)"
,"Vizzini"
,"intelligence"
,"Fezzik"
,"strength"
,"Inigo Montoya"
,"swordmanship"]

(characters/insert-characters
 db
 {:characters
  [["Vizzini" "intelligence"]
   ["Fezzik" "strength"]
   ["Inigo Montoya" "swordmanship"]]})  ;;=>
3
```

Retrieving Last Inserted ID or Record

It is often the case that you want to return the record just inserted or at least the auto-generated ID. This functionality varies greatly across databases and JDBC drivers. HugSQL attempts to help where it can. You will need to choose an option that fits your database.

Option #1: INSERT ... RETURNING

If your database supports the RETURNING clause of INSERT (e.g.,Postgresql supports this), you can specify your SQL insert statement command type to be `:returning-execute`, or `:<!` for short:

SQL

```
-- :name insert-into-test-table-returning :<!
-- :doc insert with an sql returning clause
insert into test (id, name) values (:id, :name) returning id
```

Option #2: Get Generated Keys / Last Insert ID / Inserted Record

HugSQL's `:insert`, or `:i!` command type indicates to the underlying adapter that the insert should be performed, and then `.getGeneratedKeys` called in the jdbc driver. The return value of `.getGeneratedKeys` varies greatly across different databases and jdbc drivers. For example, see the following code from the HugSQL test suite:

SQL

```
-- :name insert-into-test-table-return-keys :insert :raw
insert into test (id, name) values (:id, :name)
```

Clojure

```
(testing "insert w/ return of .getGeneratedKeys"
;; return generated keys, which has varying support and return values
;; clojure.java.jdbc returns a hashmap, clojure.jdbc returns a vector of hashmaps
  (when (= adapter-name :clojure.java.jdbc)
    (condp = db-name
      :postgresql
      (is (= {:id 8 :name "H"}
          (insert-into-test-table-return-keys db {:id 8 :name "H"} {})))

      :mysql
      (is (= {:generated_key 9}
             (insert-into-test-table-return-keys db {:id 9 :name "I"})))
```

```
      :sqlite
      (is (= {(keyword "last_insert_rowid()") 10}
             (insert-into-test-table-return-keys db {:id 10 :name "J"} {})))

      :h2
      (is (= {(keyword "scope_identity()") 11}
             (insert-into-test-table-return-keys db {:id 11 :name "J"} {})))

      ;; hsql and derby don't seem to support .getGeneratedKeys
      nil))

  (when (= adapter-name :clojure.jdbc)
    (condp = db-name
      :postgresql
      (is (= [{:id 8 :name "H"}]
             (insert-into-test-table-return-keys db {:id 8 :name "H"} {})))

      :mysql
      (is (= [{:generated_key 9}]
             (insert-into-test-table-return-keys db {:id 9 :name "I"})))

      :sqlite
      (is (= [{(keyword "last_insert_rowid()") 10}]
             (insert-into-test-table-return-keys db {:id 10 :name "J"} {})))

      :h2
      (is (= [{(keyword "scope_identity()") 11}]
             (insert-into-test-table-return-keys db {:id 11 :name "J"} {})))

      ;; hsql and derby don't seem to support .getGeneratedKeys
      nil)))
```

Edit this page