

Solomiya Pobutska

Databases 3810

ASSN2

* Create a table called Roster with a name, street address, city, state, zip.

The screenshot shows a database management tool interface. On the left, a tree view displays the database structure: jdbc:h2:~/test, ROSTER, INFORMATION_SCHEMA, Users, SOLOMIYA POBUTSKA, and H2 1.4.200 (2019-10-14). The main area shows the SQL statement: create table roster (name varchar2(30), addr varchar2(30), city varchar2(30), state char(2), zip int); show columns from roster;. Below the statement, the results of the 'show columns' command are displayed as a table with 5 rows and 5 columns: FIELD, TYPE, NULL, KEY, and DEFAULT. The table shows the structure of the 'ROSTER' table: NAME (VARCHAR(30), YES, NULL), ADDR (VARCHAR(30), YES, NULL), CITY (VARCHAR(30), YES, NULL), STATE (CHAR(2), YES, NULL), and ZIP (INTEGER(10), YES, NULL). The results are summarized as (5 rows, 16 ms).

Auto Max 1000 Auto complete Auto select

Run Run Selected Auto complete Clear SQL statement:

```
create table roster (name varchar2(30), addr varchar2(30), city varchar2(30), state char(2), zip int);
show columns from roster;
```

show columns from roster;

FIELD	TYPE	NULL	KEY	DEFAULT
NAME	VARCHAR(30)	YES		NULL
ADDR	VARCHAR(30)	YES		NULL
CITY	VARCHAR(30)	YES		NULL
STATE	CHAR(2)	YES		NULL
ZIP	INTEGER(10)	YES		NULL

(5 rows, 16 ms)

* Create 6 entries. Show the table's layout.

The screenshot shows a SQL IDE interface. On the left, a file explorer displays the project structure: `jdbc:h2:~/test`, `ROSTER`, `INFORMATION_SCHEMA`, `Users`, and `SOLOMIYA POBUTSKA`. The main editor area contains the following SQL script:

```
//INSERT 6 ENTRIES
insert into roster values('Mia', 'Main St', 'Brooklyn', 'NY', 11214);
insert into roster values('Vincent', 'West St', 'Brooklyn', 'NY', 11209);
insert into roster values('Zack', 'Marks St', 'New York', 'NY', 11005);
insert into roster values('Lily', 'Carol St', 'Brooklyn', 'NY', 11313);
insert into roster values('John', 'East St', 'Brooklyn', 'NY', 11561);
insert into roster values('Mary', 'Happy St', 'New York', 'NY', 11000);
select * from roster;
```

Below the script, the execution results are displayed, showing the update count for each insert statement and the final query result:

```
insert into roster values('Mia', 'Main St', 'Brooklyn', 'NY', 11214);
Update count: 1
(0 ms)

insert into roster values('Vincent', 'West St', 'Brooklyn', 'NY', 11209);
Update count: 1
(0 ms)

insert into roster values('Zack', 'Marks St', 'New York', 'NY', 11005);
Update count: 1
(0 ms)

insert into roster values('Lily', 'Carol St', 'Brooklyn', 'NY', 11313);
Update count: 1
(0 ms)

insert into roster values('John', 'East St', 'Brooklyn', 'NY', 11561);
Update count: 1
(1 ms)

insert into roster values('Mary', 'Happy St', 'New York', 'NY', 11000);
Update count: 1
(0 ms)

select * from roster;
```

NAME	ADDR	CITY	STATE	ZIP
Mia	Main St	Brooklyn	NY	11214
Vincent	West St	Brooklyn	NY	11209
Zack	Marks St	New York	NY	11005
Lily	Carol St	Brooklyn	NY	11313
John	East St	Brooklyn	NY	11561
Mary	Happy St	New York	NY	11000

(6 rows, 1 ms)

* Add 2 new entries

The screenshot shows a database client window with a sidebar on the left containing a tree view of the database structure. The main area displays the SQL statement being executed and its results. The SQL statement is:

```
insert into roster values('John', 'East St', 'Brooklyn', 'NY', 11561);
insert into roster values('Mary', 'Happy St', 'New York', 'NY', 11000);
select * from roster;
```

The results show the execution of the SQL statement, including the insertion of two new entries and the retrieval of all data from the 'roster' table. The results are displayed in a table format with columns: NAME, ADDR, CITY, STATE, and ZIP.

NAME	ADDR	CITY	STATE	ZIP
Mia	Main St	Brooklyn	NY	11214
Vincent	West St	Brooklyn	NY	11209
Zack	Marks St	New York	NY	11005
Lily	Carol St	Brooklyn	NY	11313
John	East St	Brooklyn	NY	11561
Mary	Happy St	New York	NY	11000
Jordan	Neon St	New York	NY	11666
Clara	Mad St	New York	NY	11006

(8 rows, 0 ms)

* Modify the street address of 2 different people at the same time.

The screenshot shows the H2 database console interface. On the left, a tree view displays the database structure: jdbc:h2:~/test, ROSTER, INFORMATION_SCHEMA, Users, and SOLOMIYA POBUTSKA. The main area shows the execution of SQL statements. The first statement is an insert, followed by a select. The second statement is a comment, followed by an update and a select. The results of the update and select are displayed below.

Auto Max 1000 Auto complete Auto select

Run Run Selected Auto complete Clear SQL statement:

```
insert into roster values('Clara', 'Mad St', 'New York' , 'NY', 11006);
select * from roster;
```

//MODIFY THE STREET ADDRESS OF THE TWO DIFFERENT PEOPLE

```
update roster set addr = 'NEW ST' where name IN ('Mia', 'Zack');
select * from roster;
```

//MODIFY THE STREET ADDRESS OF THE TWO DIFFERENT PEOPLE
update roster set addr = 'NEW ST' where name IN ('Mia', 'Zack');
Update count: 2
(8 ms)

select * from roster;

NAME	ADDR	CITY	STATE	ZIP
Mia	NEW ST	Brooklyn	NY	11214
Vincent	West St	Brooklyn	NY	11209
Zack	NEW ST	New York	NY	11005
Lily	Carol St	Brooklyn	NY	11313
John	East St	Brooklyn	NY	11561
Mary	Happy St	New York	NY	11000
Jordan	Neon St	New York	NY	11666
Clara	Mad St	New York	NY	11006

(8 rows, 0 ms)

* Delete 2 entries

The screenshot shows a database client window with a sidebar on the left displaying the database structure: jdbc:h2:~/test, ROSTER, INFORMATION_SCHEMA, Users, and SOLOMIYA POBUTSKA. The main area contains a SQL editor with the following code:

```
select * from roster;  
  
// DELETE TWO ENTRIES  
delete from roster where name = 'Clara';  
delete from roster where name = 'Jordan';  
select * from roster;
```

Below the editor, the execution results are displayed. The first two delete statements are shown with an update count of 1 and a duration of 1 ms. The final select statement is followed by a table of 6 rows:

NAME	ADDR	CITY	STATE	ZIP
Mia	NEW ST	Brooklyn	NY	11214
Vincent	West St	Brooklyn	NY	11209
Zack	NEW ST	New York	NY	11005
Lily	Carol St	Brooklyn	NY	11313
John	East St	Brooklyn	NY	11561
Mary	Happy St	New York	NY	11000

(6 rows, 0 ms)

* Create a second table with the same data as the previous table

The screenshot shows a database management tool interface. On the left is a sidebar with a tree view containing 'jdbc:h2:~/test', 'ROSTER', 'ROSTER2', 'INFORMATION_SCHEMA', 'Users', and 'H2 1.4.200 (2019-10-14)'. The main area has a top toolbar with 'Auto', 'Max', '1000', 'Auto complete', and 'Auto select'. Below the toolbar are buttons for 'Run', 'Run Selected', 'Auto complete', and 'Clear'. The 'SQL statement:' text area contains the following SQL code:

```
insert into roster values('Mary', 'Happy St', 'New York', 'NY', 11000);
select * from roster;

// ADD 2 ENTRIES
insert into roster values('Jordan', 'Neon St', 'New York', 'NY', 11666);
insert into roster values('Clara', 'Mad St', 'New York', 'NY', 11006);
select * from roster;

// MODIFY THE STREET ADDRESS OF THE TWO DIFFERENT PEOPLE
update roster set addr = 'NEW ST' where name IN ('Mia', 'Zack');
select * from roster;

// DELETE TWO ENTRIES
delete from roster where name = 'Clara';
delete from roster where name = 'Jordan';
select * from roster;

// CREATE SECOND TABLE WITH THE SAME DATA
create table roster2 as select * from roster;
select * from roster2;
```

Below the SQL statement area, the execution results are displayed on a light green background. It shows the command 'CREATE SECOND TABLE WITH THE SAME DATA', the statement 'create table roster2 as select * from roster;', the 'Update count: 0', and the execution time '(3 ms)'. Below this, the command 'select * from roster2;' is shown, followed by a table of results:

NAME	ADDR	CITY	STATE	ZIP
Mia	NEW ST	Brooklyn	NY	11214
Vincent	West St	Brooklyn	NY	11209
Zack	NEW ST	New York	NY	11005
Lily	Carol St	Brooklyn	NY	11313
John	East St	Brooklyn	NY	11561
Mary	Happy St	New York	NY	11000

At the bottom of the results area, it says '(6 rows, 0 ms)'.

* Remove a column from the second table.

The screenshot shows the H2 database console interface. The left sidebar displays the database structure: jdbc:h2:~/test, ROSTER, ROSTER2, INFORMATION_SCHEMA, Users, and H2 1.4.200 (2019-10-14). The main area contains the SQL statement editor with the following commands:

```
insert into roster values('Jordan', 'Neon St', 'New York', 'NY', 11666);
insert into roster values('Clara', 'Mad St', 'New York', 'NY', 11006);
select * from roster;

// MODIFY THE STREET ADDRESS OF THE TWO DIFFERENT PEOPLE
update roster set addr = 'NEW ST' where name IN ('Mia', 'Zack');
select * from roster;

// DELETE TWO ENTRIES
delete from roster where name = 'Clara';
delete from roster where name = 'Jordan';
select * from roster;

// CREATE SECOND TABLE WITH THE SAME DATA
create table roster2 as select * from roster;
select * from roster2;

// REMOVE A COLUMN FROM THE SECOND TABLE
alter table roster2 drop column state, zip;
select * from roster2;
```

The results of the last command are shown below:

```
// REMOVE A COLUMN FROM THE SECOND TABLE
alter table roster2 drop column state, zip;
Update count: 0
(8 ms)

select * from roster2;
```

NAME	ADDR	CITY
Mia	NEW ST	Brooklyn
Vincent	West St	Brooklyn
Zack	NEW ST	New York
Lily	Carol St	Brooklyn
John	East St	Brooklyn
Mary	Happy St	New York

(6 rows, 7 ms)

* Clear out all entries from the second table.

The screenshot shows the H2 database console interface. On the left, a tree view displays the database structure: jdbc:h2:~/test, ROSTER, ROSTER2, INFORMATION_SCHEMA, Users, and H2 1.4.200 (2019-10-14). The main area contains a text editor with SQL statements and a results pane below it.

SQL Statements:

```
// MODIFY THE STREET ADDRESS OF THE TWO DIFFERENT PEOPLE
update roster set addr = 'NEW ST' where name IN ('Mia', 'Zack');
select * from roster;

// DELETE TWO ENTRIES
delete from roster where name = 'Clara';
delete from roster where name = 'Jordan';
select * from roster;

// CREATE SECOND TABLE WITH THE SAME DATA
create table roster2 as select * from roster;
select * from roster2;

// REMOVE A COLUMN FROM THE SECOND TABLE
alter table roster2 drop column state, zip;
select * from roster2;

// CLEAR OUT ALL ENTRIES FROM ROSTER2
truncate table roster2;
select * from roster2;
```

Execution Results:

```
// CLEAR OUT ALL ENTRIES FROM ROSTER2
truncate table roster2;
Update count: 0
(1 ms)

select * from roster2;
NAME ADDR CITY
(no rows, 0 ms)
```

NAME	ADDR	CITY
------	------	------

* Destroy the second table

The screenshot shows a SQL IDE window with a toolbar at the top containing icons for undo, redo, and a search icon, along with labels for 'Auto', 'Max', '1000', 'Auto complete', and 'Auto select'. On the left, a sidebar displays a file tree with 'jdbc:h2:~/test' selected, showing a database structure with 'ROSTER', 'INFORMATION_SCHEMA', 'Users', and 'H2 1.4.200 (2019-10-14)'. The main area has a 'SQL statement:' input field with buttons for 'Run', 'Run Selected', 'Auto complete', and 'Clear'. Below this, a text area contains the following SQL script:

```
//INSERT 6 ENTRIES
insert into roster values('Mia', 'Main St', 'Brooklyn', 'NY', 11214);
insert into roster values('Vincent', 'West St', 'Brooklyn', 'NY', 11209);
insert into roster values('Zack', 'Marks St', 'New York', 'NY', 11005);
insert into roster values('Lily', 'Carol St', 'Brooklyn', 'NY', 11313);
insert into roster values('John', 'East St', 'Brooklyn', 'NY', 11561);
insert into roster values('Mary', 'Happy St', 'New York', 'NY', 11000);
select * from roster;

// ADD 2 ENTRIES
insert into roster values('Jordan', 'Neon St', 'New York', 'NY', 11666);
insert into roster values('Clara', 'Mad St', 'New York', 'NY', 11006);
select * from roster;

// MODIFY THE STREET ADDRESS OF THE TWO DIFFERENT PEOPLE
update roster set addr = 'NEW ST' where name IN ('Mia', 'Zack');
select * from roster;

// DELETE TWO ENTRIES
delete from roster where name = 'Clara';
delete from roster where name = 'Jordan';
select * from roster;

// CREATE SECOND TABLE WITH THE SAME DATA
create table roster2 as select * from roster;
select * from roster2;

// REMOVE A COLUMN FROM THE SECOND TABLE
alter table roster2 drop column state, zip;
select * from roster2;

// CLEAR OUT ALL ENTRIES FROM ROSTER2
truncate table roster2;
select * from roster2;

// DESTROY ROSTER2
drop table roster2;
```

At the bottom of the window, a status bar displays the results of the last command:

```
// DESTROY ROSTER2
drop table roster2;
Update count: 0
(1 ms)
```