SQL essentials

Mastering database queries: a comprehensive guide to SQL

7

Learning session 07

Adding, Changing, and Removing Data

Instructor
Péter Fülöp
peteristvan.fulop@hcl.com

Adding new data to your tables

Using the INSERT SQL command

- The **INSERT SQL** command is fundamental for adding new data to your tables.
- When using INSERT, it's important to **consider the structure of your table** and the requirements of each column
- **For columns that have no default values** (often marked as NOT NULL), it's essential to **specify** these columns **explicitly** in your INSERT statement.
-
```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```
is a typical format where each specified column is matched with a corresponding value.

Example 1: Using the INSERT SQL command

Insert "Data Analyst" and "Data Visualization Expert" into the job_titles table, assigning them the IDs 6 and 7, respectively.

Let's review the existing content and structure of the job_titles table before we proceed to add new data to it.

Show all rows in the job_titles table.

SQL

```
SELECT *
FROM job_titles;
```

OUTPUT

id	name
1	Data engineer
2	BI developer
3	Manager
4	Data Scientist
5	BI Support specialist

5 rows

Example 1: Using the INSERT SQL command

Display column information such as column names, data types, and nullability for the job_titles table in the practice_07 schema.

SQL

```
SELECT ordinal_position, column_name,
        data_type, is_nullable, column_default
FROM information_schema.columns
WHERE table_schema = 'practice_07' AND table_name = 'job_titles';
```

OUTPUT

ordinal_position	column_name	data_type	is_nullable	column_default
1	id	integer	NO	nextval('practic…
2	name	character v…	NO	*<null>*

1 rows

Example 1: Using the INSERT SQL command

Insert "Data Analyst" and "Data Visualization Expert into the job_titles table, assigning them the IDs 6 and 7, respectively.

SQL

```sql
INSERT INTO  job_titles(id, name) VALUES ( 6, 'Data Analyst');
INSERT INTO  job_titles(id, name) VALUES ( 7, 'Data Visualization Expert');
SELECT * FROM job_titles;
```

OUTPUT

id	name
5	BI Support specialist
6	Data Analyst
7	Data Visualization Expert

7 rows

Example 2: Using the INSERT SQL command

Add "Finance" and “Marketing” to the business_units table without specifying a particular id.

Let's review the existing content and structure of the business_units table before we proceed to add new data to it.

```sql
CREATE TABLE business_units
(
    id                  serial PRIMARY KEY,
    name                text NOT NULL,
    activity_start_date date NOT NULL
);
```

Show all rows in the business_units table.

```sql
SELECT *
FROM business_units;
```

OUTPUT

id	name	activity_start_date
1	Consulting 1	2010-01-15
2	Consulting 2	2011-02-20
3	Managed services	2012-03-10
4	Team augmentation 1	2011-04-05
5	Team Augmentation 2	2011-05-12
6	Generative AI	2023-09-09

6 rows

Example 2: Using the INSERT SQL command

Add "Finance" and “Marketing” to the business_units table without specifying a particular id.

BULK INSERT SQL

```sql
INSERT INTO business_units (name, activity_start_date)
VALUES ('Finance', TO_DATE('2013-01-01', 'YYYY-MM-DD')),
       ('Marketing', TO_DATE('2013-03-01', 'YYYY-MM-DD'));
```

OUTPUT

	id	name	activity_start_date
5	5	Team Augmentation 2	2011-05-12
6	6	Generative AI	2023-09-09
7	7	Finance	2013-01-01
8	8	Marketing	2013-03-01

8 rows

Inserting the results of a SELECT statement

Every year, the company recognizes a selection of its employees. In 2023, the management board chose to honor employees who have the IDs 1, 5, 7, and 10. The details such as id, first_name, middle_name, birth_date, and email address of these selected employees are to be duplicated into the annual_employee_awards table.

Here's the CREATE TABLE statement for the `annual_employee_awards table`. Let's review it together to gain a better understanding of the table's columns.

SQL

```
CREATE TABLE practice_07.annual_employee_awards(
    employee_id   INTEGER,
    first_name    VARCHAR(25) NOT NULL,
    middle_name   VARCHAR(25),
    last_name     VARCHAR(25) NOT NULL,
    birth_date    DATE        NOT NULL,
    email_address VARCHAR(255) CHECK (email_address LIKE '%@hcl.com'),
    year          INTEGER     NOT NULL CHECK (year BETWEEN 2005 AND 2030),
    award         VARCHAR(255) DEFAULT NULL,
    PRIMARY KEY (employee_id, year));
```

Inserting the results of a SELECT statement

As a first step let us collect id, first_name, middle_name, birth_date, and email address of the employees with ID 1, 5, 7 and 10.

BULK INSERT — SQL

```sql
SELECT id AS employee_id, first_name, middle_name, last_name, birth_date,
       email_address, 2023 as year, ''   as award
FROM employees
WHERE id IN (1,5,7,10);
```

OUTPUT

id	first_name	middle_name	last_name	birth_date	email_address	year	award
1	Gabor	<null>	Nagy	1990-08-10	gabor@hcl.com	2023	
5	Katalin	Anna	Szabo	1998-07-05	katalin@hcl.com	2023	
7	Anita	<null>	Molnar	1997-09-12	anita@hcl.com	2023	
10	Laszlo	Gabor	Nagy	1993-06-28	laszlo@hcl.com		

4 rows

Inserting the results of a SELECT statement

Let us insert the results of the SELECT statement into the annual_employee_awards table.

SQL

```sql
INSERT INTO annual_employee_awards
SELECT id AS employee_id, first_name, middle_name, last_name, birth_date,
       email_address, 2023 as year, ''   as award
FROM employees
WHERE id IN (1,5,7,10);
```

OUTPUT

id	first_name	middle_name	last_name	birth_date	email_address	year	award
1	Gabor	<null>	Nagy	1990-08-10	gabor@hcl.com	2023	
5	Katalin	Anna	Szabo	1998-07-05	katalin@hcl.com	2023	
7	Anita	<null>	Molnar	1997-09-12	anita@hcl.com	2023	
10	Laszlo	Gabor	Nagy	1993-06-28	laszlo@hcl.com		

4 rows

The INSERT ... ON CONFLICT construct

Let's assume the email address for the employee with ID 10 in the employee table has been updated, and we need to re-insert this information into the `annual_employee_awards` table.

SQL

```
UPDATE employees
SET email_address='laszlo.nagy10@hcl.com'
WHERE id = 10;
```

Let us re-run out previous INSERT INTO command for employee with the id 10.

SQL

```
INSERT INTO annual_employee_awards
SELECT id as employee_id, first_name, middle_name, last_name, birth_date,
       email_address, 2023 as year, ''   as award
FROM employees
WHERE id = 10;
```

ERROR: duplicate key value violates unique constraint "annual_employee_awards_pkey Detail: Key (employee_id, year)=(10, 2023) already exists.

The INSERT ... ON CONFLICT construct

In the context of relational databases, the concept of '**UPSERT**' is often known as 'merge'.
This process involves inserting a new row into a table, but if the row already exists, PostgreSQL will update it instead. Hence, the term 'upsert' is a blend of 'update' and 'insert', reflecting this dual functionality.

SQL

```sql
INSERT INTO annual_employee_awards
SELECT id as employee_id, first_name, middle_name, last_name, birth_date,
       email_address, 2023 as year, ''   as award
FROM employees
WHERE id = 10
ON CONFLICT (employee_id, year)
    DO UPDATE
    SET email_address = excluded.email_address;
```

Updating the content of your tables

Using the UPDATE SQL command

- The **UPDATE** SQL command is used to modify existing records in a database table. It allows you to change data in one or more rows, based on a specified condition.

- The basic syntax of the UPDATE command is:

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

- **The WHERE clause is crucial.** Without it, all rows in the table will be updated. Test your UPDATE query with a SELECT statement first to ensure it affects the correct rows.

- The UPDATE command can be combined with JOIN to update values based on data in another table.

```
UPDATE table1
SET column1 = table2.column2
FROM table2
WHERE table1.column_m = table2.column_n;
```

Example 1: Using the UPDATE SQL command

On December 1st, 2023, employees bearing IDs 5 and 9 underwent a salary revision. Each of these employees received a 10% increase in their salary. Accordingly, update the columns for salary and last_salary_review_date.

SQL

```sql
SELECT id, first_name, last_name, salary, last_salary_review_date
FROM employees
WHERE id=5 OR id=9;
```

OUTPUT

id	first_name	last_name	salary	last_salary_review_date
5	Katalin	Szabo	2800	2023-01-20
9	Lilla	Toth	4100	2022-08-10

2 rows

Example 1: Using the UPDATE SQL command

On December 1st, 2023, employees bearing IDs 5 and 9 underwent a salary revision. Each of these employees received a 10% increase in their salary. Accordingly, update the columns for salary and last_salary_review_date.

SQL

```sql
UPDATE employees
SET salary=ROUND(salary*1.1, 0),
    last_salary_review_date = TO_DATE('2023-12-01', 'YYYY-MM-DD')
WHERE id=5 OR id=9; -- re-run the previous SELECT statement to check the results
```

OUTPUT

id	first_name	last_name	salary	last_salary_review_dat
5	Katalin	Szabo	3080	2023-12-01
9	Lilla	Toth	4510	2023-12-01

2 rows

Example 2: Using the UPDATE SQL command

Modify all email addresses in the annual_employee_awards table for the employees receiving awards in 2023, ensuring they match the email addresses listed in the employees table.

SQL

```sql
UPDATE annual_employee_awards
SET email_address=employees.email_address
FROM employees
WHERE employees.id = annual_employee_awards.employee_id
      AND annual_employee_awards.year=2023;
```

OUTPUT

employee_id	first_name	middle_name	last_name	birth_date	email_address	year
1	Gabor	<null>	Nagy	1990-08-10	gabor@hcl.com	2023
5	Katalin	Anna	Szabo	1998-07-05	katalin@hcl.com	2023
7	Anita	<null>	Molnar	1997-09-12	anita@hcl.com	

4 rows

Removing data from your tables

Using the DELETE SQL command

- The **DELETE** command in SQL is used to remove one or more rows from a table.
- The basic syntax of the UPDATE command is:

```
DELETE FROM table_name
WHERE condition;
```

- **The WHERE clause is crucial.** If the condition is omitted, all rows in the table will be removed.
- Deleted data cannot be recovered unless you have a backup. It's good practice to run a SELECT statement with the same condition first to ensure you're deleting the intended rows.

Example 1: Using the DELETE SQL command

Upon identifying a mistake in the algorithm used for assigning awards, the company has made the decision to exclude the employee with employee_id 7 from the list of those awarded in the year 2023.

SQL

```sql
-- before executing the DELETEcommand, let us identify the row(s) to be deleted
SELECT * FROM annual_employee_awards WHERE id=7 AND year=2023;
-- delete the requested entry
DELETE FROM annual_employee_awards WHERE id=7 AND year=2023;
-- Run the SELECT * FROM annual_employee_awards; to test the results
```

OUTPUT

employee_id	first_name	middle_name	last_name	birth_date	email_address	year
1	Gabor	*null*	Nagy	1990-08-10	gabor@hcl.com	2023
5	Katalin	Anna	Szabo	1998-07-05	katalin@hcl.com	2023
10	Laszlo	Gabor	Nagy	1993-06-28	laszlo.nagy1	

3 rows