Adding and Deleting Foreign Keys

by Sophia

WHAT'S COVERED

This lesson describes the constraints involved in inserting or deleting data in tables with foreign key relationships and describes the correct order in which the statements must be executed, in two parts. Specifically, this lesson will cover:

1. Inserting With Foreign Keys

When it comes to foreign keys, it can be challenging to determine the order in which data must be inserted into tables, especially when we have multiple tables that relate to one another. In the example PostgreSQL database we have been working with so far in this course, we have the following foreign keys in place across the various tables:

constraint_name	table_name	column_name	foreign_table_name	foreign_column_name
album_artist_id_fkey	album	artist_id	artist	artist_id
track_album_id_fkey	track	album_id	album	album_id
employee_reports_to_fkey	employee	reports_to	employee	employee_id
customer_support_rep_id_fkey	customer	support_rep_id	employee	employee_id
invoice_customer_id_fkey	invoice	customer_id	customer	customer_id
track_genre_id_fkey	track	genre_id	genre	genre_id
invoice_line_invoice_id_fkey	invoice_line	invoice_id	invoice	invoice_id
track_media_type_id_fkey	track	media_type_id	media_type	media_type_id
invoice_line_track_id_fkey	invoice_line	track_id	track	track_id
playlist_track_track_id_fkey	playlist_track	track_id	track	track_id
playlist_track_playlist_id_fkey	playlist_track	playlist_id	playlist	playlist_id

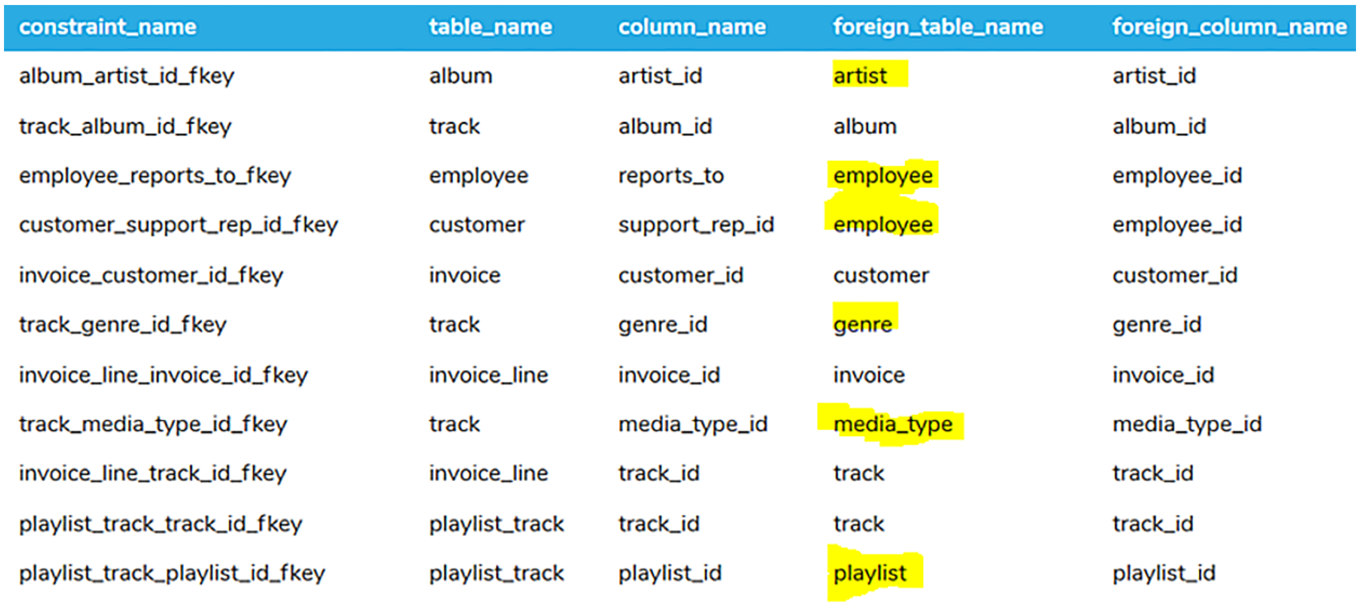
In the above list, the first column contains the constraint name. Notice that the default naming scheme for foreign key constraints is the following:

© 2024 SOPHIA Learning, LLC. SOPHIA is a registered trademark of SOPHIA Learning, LLC.

_<column></column>_fkey
The second and third columns contain the information about the child table with a foreign key. The fourth and fifth columns show information about the parent table and the primary key that the child is referencing.

At first glance, this seems backwards, given the column headings: You would think that the child table would be the one with "foreign" in the column headings, but it is the reverse of that. Think of it this way: The first column lists the foreign key, and the second and third columns list the table and attribute that contain that foreign key.

Now, let's say that we want to insert records into both the parent and child tables. The order in which these insert operations occur is important. We must begin by inserting records into the tables that do NOT have foreign keys to other tables. If we look at the set of table constraints, we can see that those tables are artist, genre, media_type, playlist, and employee, as shown below. These are the tables that are not referenced in any of the constraints in the first column.

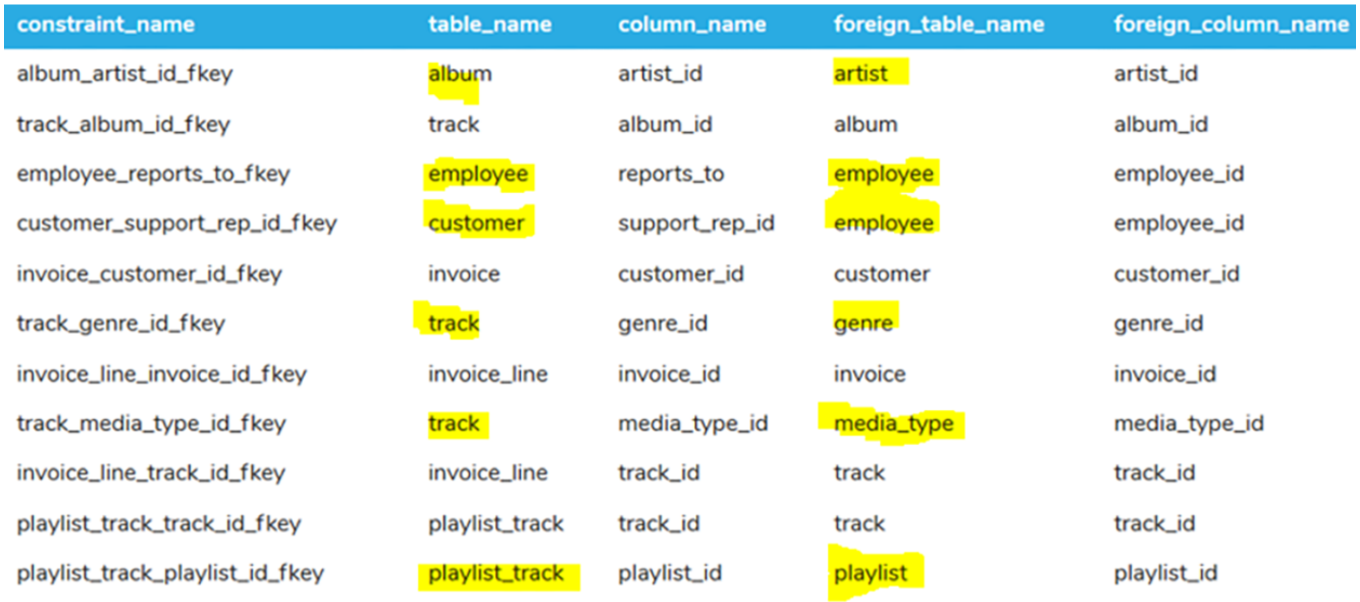


constraint_name	table_name	column_name	foreign_table_name	foreign_column_name
album_artist_id_fkey	album	artist_id	artist	artist_id
track_album_id_fkey	track	album_id	album	album_id
employee_reports_to_fkey	employee	reports_to	employee	employee_id
customer_support_rep_id_fkey	customer	support_rep_id	employee	employee_id
invoice_customer_id_fkey	invoice	customer_id	customer	customer_id
track_genre_id_fkey	track	genre_id	genre	genre_id
invoice_line_invoice_id_fkey	invoice_line	invoice_id	invoice	invoice_id
track_media_type_id_fkey	track	media_type_id	media_type	media_type_id
invoice_line_track_id_fkey	invoice_line	track_id	track	track_id
playlist_track_track_id_fkey	playlist_track	track_id	track	track_id
playlist_track_playlist_id_fkey	playlist_track	playlist_id	playlist	playlist_id

Note that the employee table links to itself, as the manager is linked using the reports_to column in the same table. So, the first five tables that we would insert records into are:

- artist
- employee
- genre
- media_type
- playlist

The order among the five tables does not matter, as they are all on the first level of tables. Next, we look at the tables that reference those five tables.

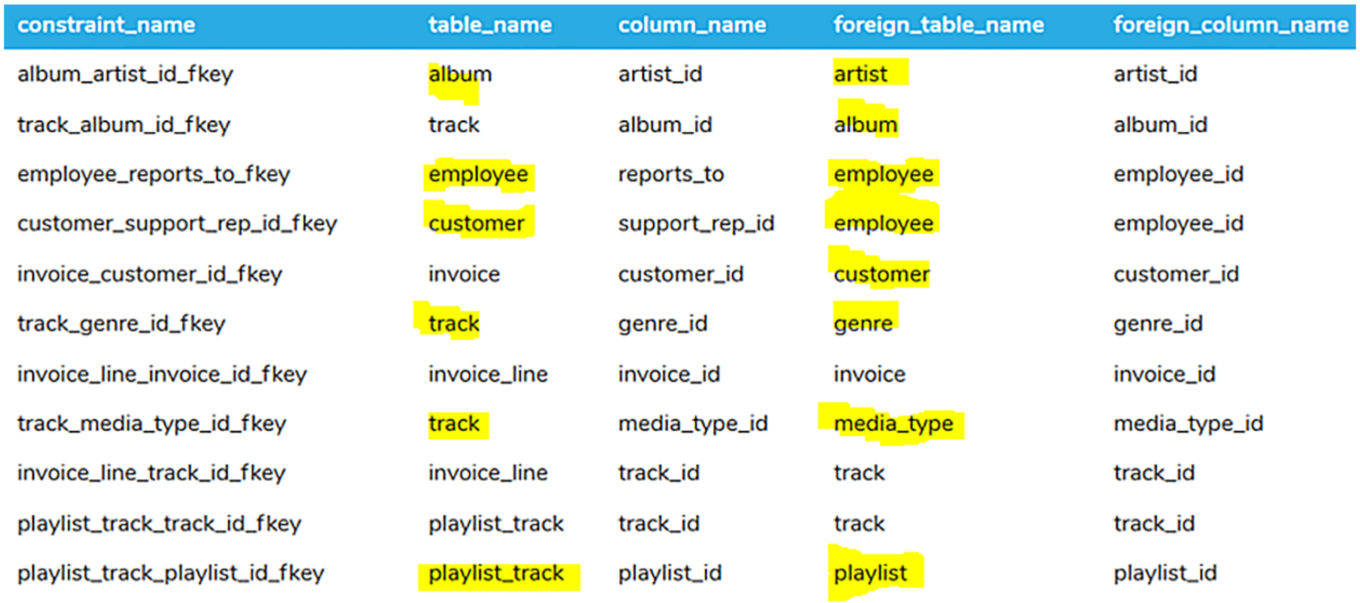


constraint_name	table_name	column_name	foreign_table_name	foreign_column_name
album_artist_id_fkey	album	artist_id	artist	artist_id
track_album_id_fkey	track	album_id	album	album_id
employee_reports_to_fkey	employee	reports_to	employee	employee_id
customer_support_rep_id_fkey	customer	support_rep_id	employee	employee_id
invoice_customer_id_fkey	invoice	customer_id	customer	customer_id
track_genre_id_fkey	track	genre_id	genre	genre_id
invoice_line_invoice_id_fkey	invoice_line	invoice_id	invoice	invoice_id
track_media_type_id_fkey	track	media_type_id	media_type	media_type_id
invoice_line_track_id_fkey	invoice_line	track_id	track	track_id
playlist_track_track_id_fkey	playlist_track	track_id	track	track_id
playlist_track_playlist_id_fkey	playlist_track	playlist_id	playlist	playlist_id

Here we have the album, employee, customer, playlist_track, and track table. However, if we look at the second row, we see that the track table also has a foreign key to the album_id in the album table. The album table was not in the first set of tables, so the track table has yet to be added. Likewise, in the last row, the playlist_track has a link to the track table, which isn't available yet. So, the next set of tables that can be inserted into is:

- album
- customer

Next, we can identify that the following tables are linked to the ones at the prior level and do not have other dependencies:



constraint_name	table_name	column_name	foreign_table_name	foreign_column_name
album_artist_id_fkey	album	artist_id	artist	artist_id
track_album_id_fkey	track	album_id	album	album_id
employee_reports_to_fkey	employee	reports_to	employee	employee_id
customer_support_rep_id_fkey	customer	support_rep_id	employee	employee_id
invoice_customer_id_fkey	invoice	customer_id	customer	customer_id
track_genre_id_fkey	track	genre_id	genre	genre_id
invoice_line_invoice_id_fkey	invoice_line	invoice_id	invoice	invoice_id
track_media_type_id_fkey	track	media_type_id	media_type	media_type_id
invoice_line_track_id_fkey	invoice_line	track_id	track	track_id
playlist_track_track_id_fkey	playlist_track	track_id	track	track_id
playlist_track_playlist_id_fkey	playlist_track	playlist_id	playlist	playlist_id

Now that we have added the records to the album table, we can add each album's tracks to the track table. The invoice table depends only on the customer table, so invoices can also be entered at this point. So, on the third level, we can insert records into the track and invoice tables, in any order.

© 2024 SOPHIA Learning, LLC. SOPHIA is a registered trademark of SOPHIA Learning, LLC.

constraint_name	table_name	column_name	foreign_table_name	foreign_column_name
album_artist_id_fkey	album	artist_id	artist	artist_id
track_album_id_fkey	track	album_id	album	album_id
employee_reports_to_fkey	employee	reports_to	employee	employee_id
customer_support_rep_id_fkey	customer	support_rep_id	employee	employee_id
invoice_customer_id_fkey	invoice	customer_id	customer	customer_id
track_genre_id_fkey	track	genre_id	genre	genre_id
invoice_line_invoice_id_fkey	invoice_line	invoice_id	invoice	invoice_id
track_media_type_id_fkey	track	media_type_id	media_type	media_type_id
invoice_line_track_id_fkey	invoice_line	track_id	track	track_id
playlist_track_track_id_fkey	playlist_track	track_id	track	track_id
playlist_track_playlist_id_fkey	playlist_track	playlist_id	playlist	playlist_id

Then, on the final level, we can now insert into invoice_line and playlist_track, as all dependencies have been accounted for.

In review, we would have to insert into any grouping of the following levels of tables:

Level 1

- artist
- employee
- genre
- media_type
- playlist

Level 2

- album
- customer

Level 3

- track
- invoice

Level 4

- invoice_line
- playlist_track

Looking at the above structure, suppose that we need to create an invoice for an existing customer. We start out at Level 3, where the invoice table is located, and then after creating the invoice, we move to the child table at Level 4, invoice_line, and enter the records that detail the invoice's contents.

2. Deleting With Foreign Keys

When deleting records, it's also important to do so in the right order. It's the same order as when inserting—but backwards. We want to delete the items at the lowest level first, and work upward.

As you learned in the previous lesson, there are options you can set for foreign keys that can ensure that deleted data does not lead to a data integrity problem.

A foreign key constraint that specifies the ON DELETE CASCADE option will automatically delete all rows that depend on the referenced table when a row is deleted in the referenced table. The purpose of this is to prevent orphaned records from being left behind.

The ON DELETE SET NULL option sets NULL the foreign key values in the referencing table when a row is deleted in the referenced table. When the deletion of data shouldn't lead to data loss but rather nullify the relationship, it might be appropriate to do this.

Most database systems set ON DELETE NO ACTION as the default, which prevents the referenced row from being deleted if it has dependent rows in the referencing table. As a result, the relationships between the data are maintained. The ON DELETE RESTRICT option is very similar to ON DELETE NO ACTION, and in most cases, it does the same thing—it restricts the referenced row from being deleted.

In our sample database, the tables do not have ON DELETE CASCADE options set up for the foreign keys, so NO ACTION is the default. That means our delete operations will fail unless we delete the records in the correct order: the opposite direction from when we inserted. In other words, we must delete the child rows before the parent rows.

Records from the tables would have to be deleted in the following order by level:

Level 4

- Invoice_line
- playlist_track

Level 3
- track
- invoice

Level 2
- album
- customer

Level 1
- artist
- employee

© 2024 SOPHIA Learning, LLC. SOPHIA is a registered trademark of SOPHIA Learning, LLC.

- genre
- media_type
- playlist

For example, if we wanted to delete a customer, we would have to delete from the invoice_line of the invoices that belong to that customer, then delete the invoices in the invoice table that belong to the customer. Then, finally, delete the customer.

TRY IT

Your turn! Open the SQL tool by clicking on the LAUNCH DATABASE button below. Then, enter one of the examples above and see how it works. Next, try your own choices for which columns you want the query to provide.

SUMMARY

In this lesson, you learned that SQL databases rely heavily on foreign key constraints to maintain data integrity and enforce referential integrity between tables. When **inserting new records in tables with foreign keys**, we must insert starting with the parent tables and work our way down to the child tables. When **deleting records in tables with foreign keys**, we go in the opposite direction: first the records from the child tables and then the corresponding records from the parent tables. You were also reminded of the CASCADE options you learned about earlier, which can be used to specify what should happen when someone tries to delete records in an order different from that. Some of these options include SET NULL and NO ACTION.

Source: THIS TUTORIAL WAS AUTHORED BY DR. VINCENT TRAN, PHD (2020) AND FAITHE WEMPEN (2024) FOR SOPHIA LEARNING. PLEASE SEE OUR **TERMS OF USE**.

© 2024 SOPHIA Learning, LLC. SOPHIA is a registered trademark of SOPHIA Learning, LLC.