Module 2-4

INSERT, UPDATE, DELETE

Objectives

- INSERT
- DELETE
- UPDATE
- Understand benefits of referential integrity
- Understand how constraints limit changes that can be made
- Transactions

Data Operations - The CRUD

C - Create (INSERT)

R - Read (SELECT)

U - Update (UPDATE)

D - Delete (DELETE)

Changing data

The row data for each table in a database can be changed or deleted. New rows of data can also be added. There are 3 types of statements we will cover today:

- **INSERT**: Adds a new row to the table.
- **UPDATE**: Changes the column value for an existing row or rows.
- **DELETE**: Permanently removes a row from the table.

- DML, DDL, DCL – DB Manipulation Language

INSERT statements

You can use the INSERT statement to insert 1 row into the database. The following pattern is used:

INSERT INTO **[Name of Table]** (**[name of col 1]**, **[name of col 2]**)

VALUES (**[value for col 1]**, **[value for col2]**);

INSERT statements example

Consider the following example:

```
INSERT INTO person (person_name, birthday) VALUES ('Shia LeBouf',
'06/11/86');
```

In English, this translates to insert a new row in the table person, on this new row the value for person_name is going to be “Shia LeBouf” and the value for the birthday is going to be “06/11/86”.

Data Output | Explain | Messages | Notifications

	person_name character varying (200)	birthday date
1	Shia LeBouf	1986-06-11

INSERT statements example

Note that in the previous example, we only specified two columns and did not specify that a value be inserted for person_id.

Data Output | Explain | Messages | Notifications

	person_id [PK] integer	person_name character varying (200)	birthday date	deathday date	biography text
1	3984916	Shia LeBouf	1986-06-11	[null]	[null]

- person_id is of a special data type called **serial**.
- A column marked as serial will automatically increase in value with each new row.
- Columns marked as serial should not be included in the INSERT.

UPDATE statements

An update statement changes the column values for **one or more existing rows**.

UPDATE [table name]

SET [col 1 name] = **[col 1 value]**

WHERE ...

UPDATE statements example

Consider the following example:

```
UPDATE person
SET
person_name = 'Donald Wahlberg',
birthday = '08-16-1969'
WHERE person_id = 2680;
```

In here, we have changed the value for 2 columns (first_name and last_name) but only for the row with an actor_id of 2.

We can separate multiple columns that need updating with a comma.

The syntax for structuring the WHERE statement remains unchanged.

UPDATE statements example

Consider the following example:

```
UPDATE person
SET
person_name = 'Donald Wahlberg',
birthday = '08-16-1969'
```

We have just set every person's name to Donald Wahlberg and their birthday to 08/16/1969!!!

UPDATE statements example

Consider the following example:

```
UPDATE person
SET
person_name = 'Donnie Wahlberg'
WHERE
birthday = '08-16-1969'
AND deathday IS NULL;
```

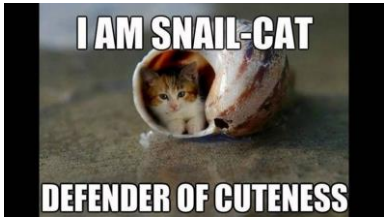
DELETE statements

A delete statement removes row or rows from the table. It follows this format:

DELETE FROM [table name]

WHERE …

In the absence of a WHERE statement, every row in the database will be deleted!



DELETE statements example

Consider the following example.

```
DELETE FROM movie_actor
WHERE
actor_id = 2;
```

Here, we are deleting every row that has an actor_id of 2.

Referential Integrity

No limit

Movie_db/postgres@PostgreSQL 13

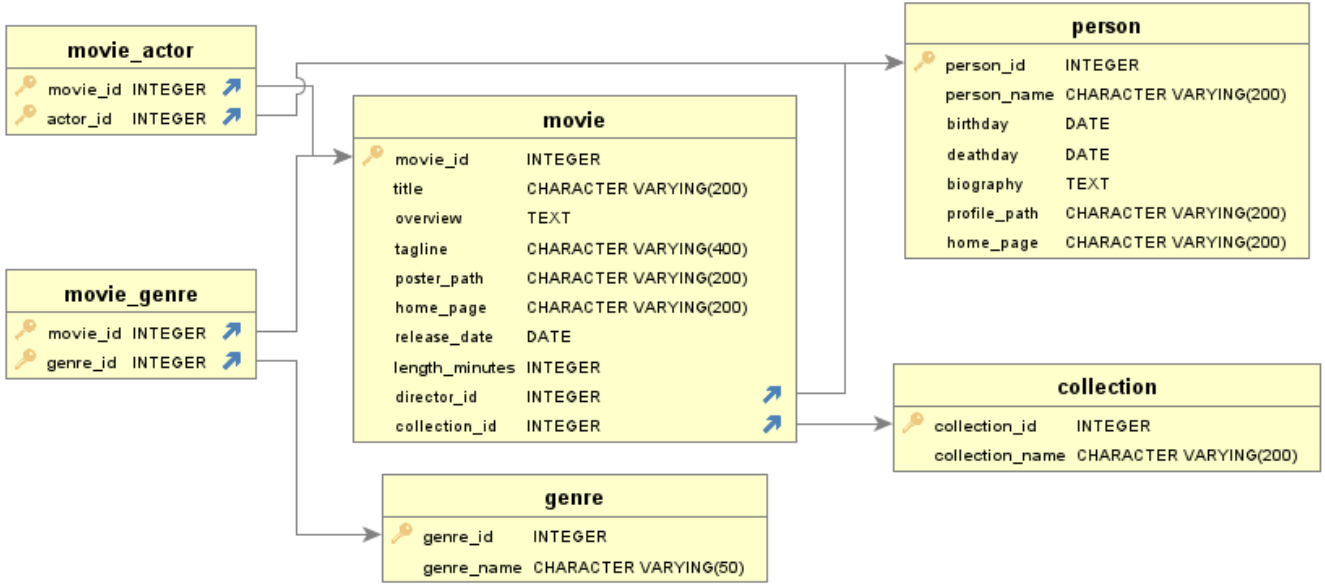
Query Editor Query History

```
1  DELETE FROM person
2  WHERE person_id = 2;
```

Data Output Explain Messages Notifications

```
ERROR:  update or delete on table "person" violates foreign key constraint "fk_movie_actor_actor" on table "movie_actor"
DETAIL:  Key (person_id)=(2) is still referenced from table "movie_actor".
SQL state: 23503
```

Referential Integrity



Constraints

Constraints are rules imposed on the table, upon creation, that limits the ability to change the data.

- **NOT NULL:** A value must be specified
- **PRIMARY KEY:** Define that certain column/columns are part of the key
  - **A primary key value cannot be NULL**.
- **FOREIGN KEY:** Defines a foreign key based on a primary key from a different table
- **CHECK:** Only certain values can be inserted or updated

Transactions

A large number of SQL statements can be rolled into a single transaction.

The following syntax is observed:

START TRANSACTION; -- or BEGIN TRANSACTION;

// Lots of SQL statements.

COMMIT TRANSACTION; -- or COMMIT;

Your INSERT or UPDATE SQL statements **will only commit (permanently save in the database) if all the SQL statements in the transaction end successfully**.

Transaction Syntax

`START TRANSACTION`

Do the UPDATE/INSERT/DELETE statements

`COMMIT` (ends the transaction and saves the changes)

OR

`ROLLBACK` (ends the transaction without saving the changes)

Transactions can be used to safely test a statement that changes the database during development/testing.