Pulse Survey is open

Insert, Update, Delete, RI, and Constraints

Module 2: 04

Today’s Objectives

1. Inserts
2. Deletes
3. Updates
4. Constraints and referential integrity
5. Transactions

Data Operations - The CRUD

C - Create (INSERT)

R - Read (SELECT)

U - Update (UPDATE)

D - Delete (DELETE)

INSERT

Adds a new row of data to a table

```
INSERT INTO table_name (column1, column2, ..., column_n)
VALUES (value1, value2, ... value_n);
```

Can be shortened to insert every column:

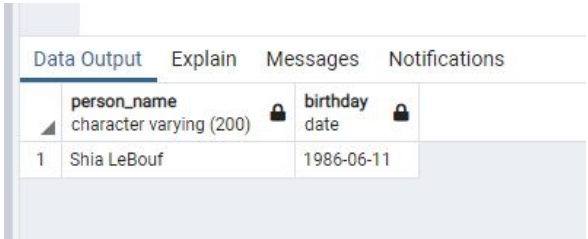
```
INSERT INTO table_name VALUES (value1, value2, ... value_n);
```

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”.



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

Updates the value of columns on an existing row of data for the specific rows.

```
UPDATE table_name
SET column = value
WHERE column = value;
```

Can update multiple columns in a since update statement.

```
UPDATE table_name
SET column1 = value1, column2 = value2
WHERE column = value;
```

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!!!

DELETE

Deletes row(s) of data from a table.

```
DELETE FROM table_name
WHERE column=value;
```

DELETE statements example

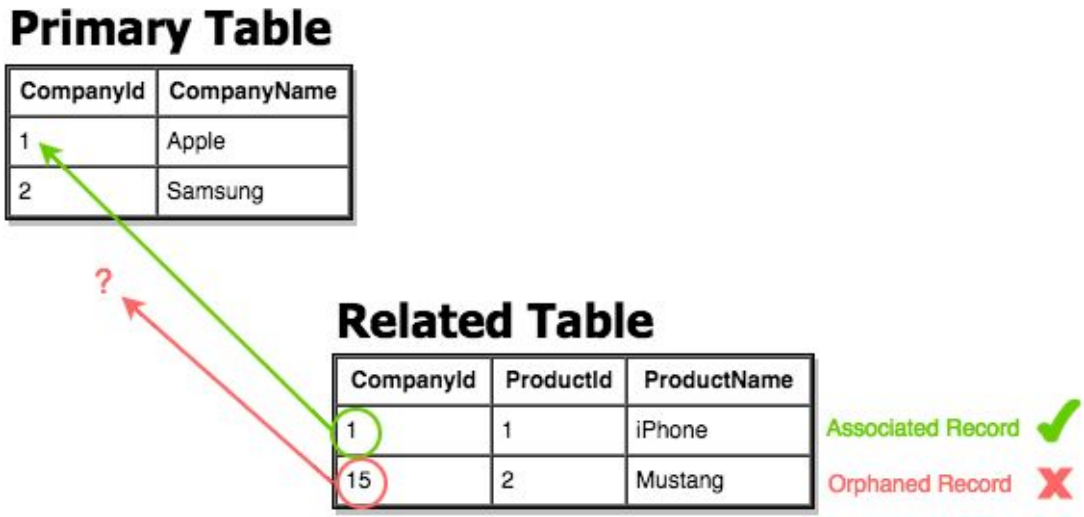
Consider the following example.

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

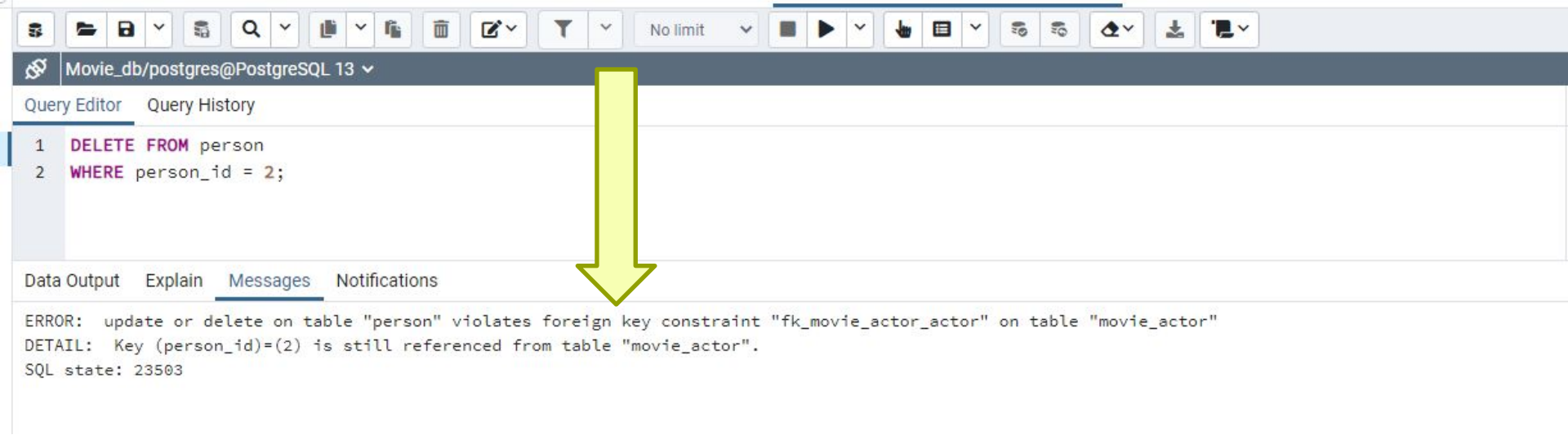
Here, we are deleting every row that has an actor_id of 2.

Referential Integrity

Referential Integrity is a property of the data stating whether or not references within it are valid. For example, to use a foreign key on a table, the value must exist on the primary table.



Referential Integrity



Constraints

A **constraint** on a table defines properties that the column data must comply with. It sets a rule that must be obeyed to maintain the integrity of the data.

NOT NULL	The column cannot contain a null value
UNIQUE	The column can only contain unique values
PRIMARY KEY	Enforces NOT NULL and UNIQUE. Allows Foreign Key relationships to be established.
FOREIGN KEY	Only allows values where the value exists on the related table. Does not allow the related value to be removed from the related table as long as the foreign key value is in use.
CHECK	Specifies a list of acceptable values that can be added into a column
DEFAULT	Provides a default value for a column, if no value is provided.

Transactions

A transaction is a single unit of work made up of multiple SQL statements that must all succeed or fail as one.

> When a transaction is successful it is ***committed*** and the data is saved in the new state.
>
> When a transaction fails it is ***rolled back*** and all the data is left in the original state.

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.

The ACID Test

The **ACID Test** is used to determine whether a series of actions should be a transaction.

1. **Atomicity (Atomic):** Must the actions occur as all or none
2. **Consistency (Consistent):** Once the series of actions is complete is the data left in a consistent state, meaning that saved data cannot violate the integrity of the database so any rules that pass before the transaction still pass after the transaction.
3. **Isolation:** Does the final result of the transaction leave the data in the same state as they would have if executed serially. Other transactions don't affect the outcome and must wait for this one to complete before they are applied.
4. **Durability:** Once the transaction has been committed, will it remain so even after an error, system crash, or power loss.