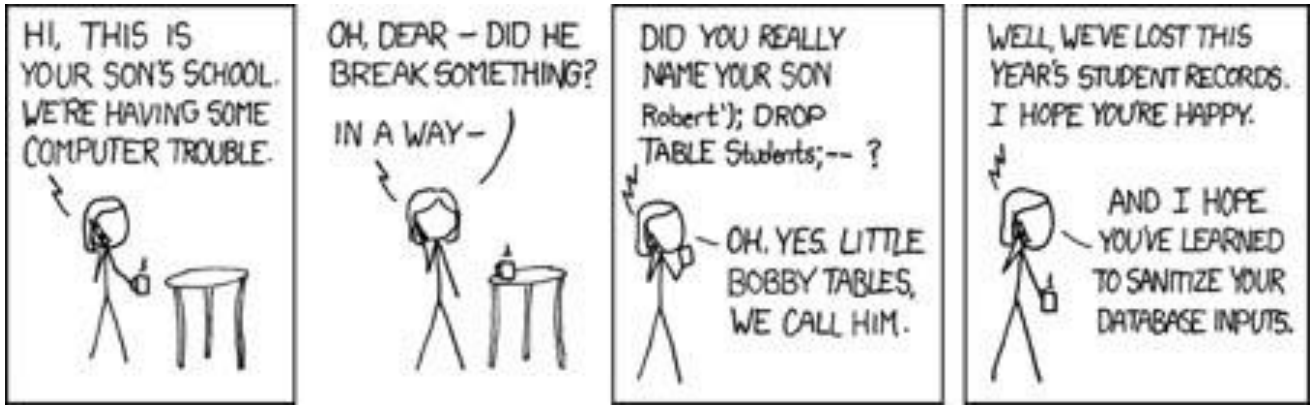

HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR - DID HE BREAK SOMETHING?
IN A WAY-
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

# Module 2-4

INSERT, UPDATE, DELETE

# Objectives

- INSERT
- DELETE
- UPDATE
- Constraints and referential integrity
- Transactions

# 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.

# 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]**)

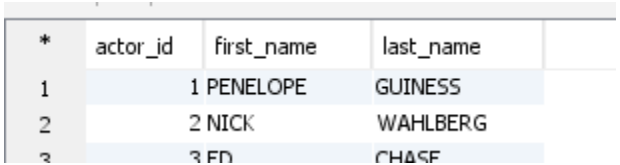
VALUES (**[value for col 1]**, **[value for col2]**);

# INSERT statements example

Consider the following example:

```
INSERT INTO actor (first_name, last_name) VALUES ('Shia','LeBouf');
```

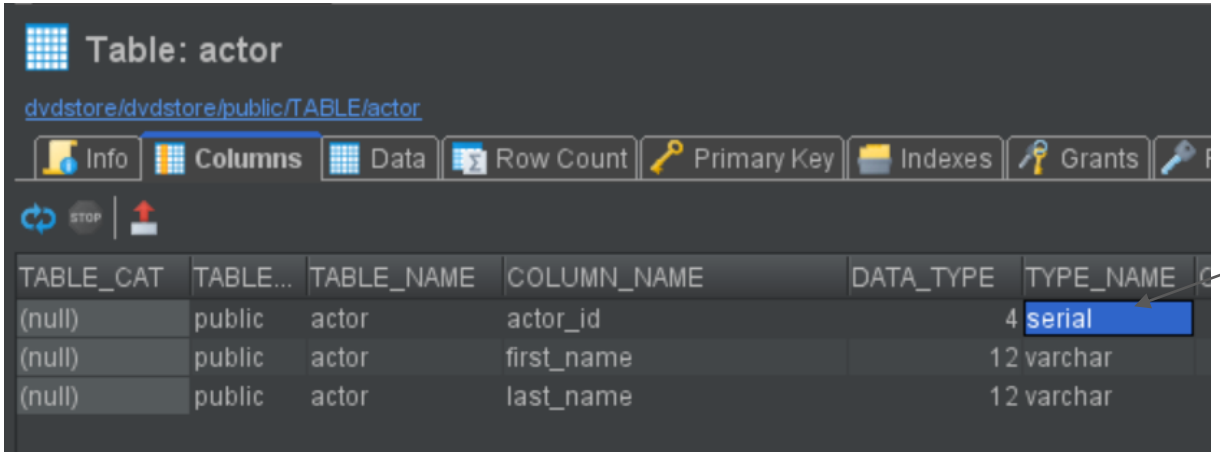
In English, this translates to insert a new row in the table actor, on this new row the value for first_name is going to be “Shia” and the value for the last_name is going to be “LeBouf”.



| * | actor_id | first_name | last_name |
|---|---|---|---|
| 1 | 1 | PENELOPE | GUINESS |
| 2 | 2 | NICK | WAHLBERG |
| 3 | 3 | ED | CHASE |

# INSERT statements example

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




- actor_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 actor
SET
first_name = 'Nicholas',
last_name = 'Wahberg'
WHERE
actor_id = 2;
```

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 actor
SET
first_name = 'Nicholas',
last_name = 'Wahberg';
```

We have just set every actors first name to Nicholas and their last name to Wahberg!!!

# 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!

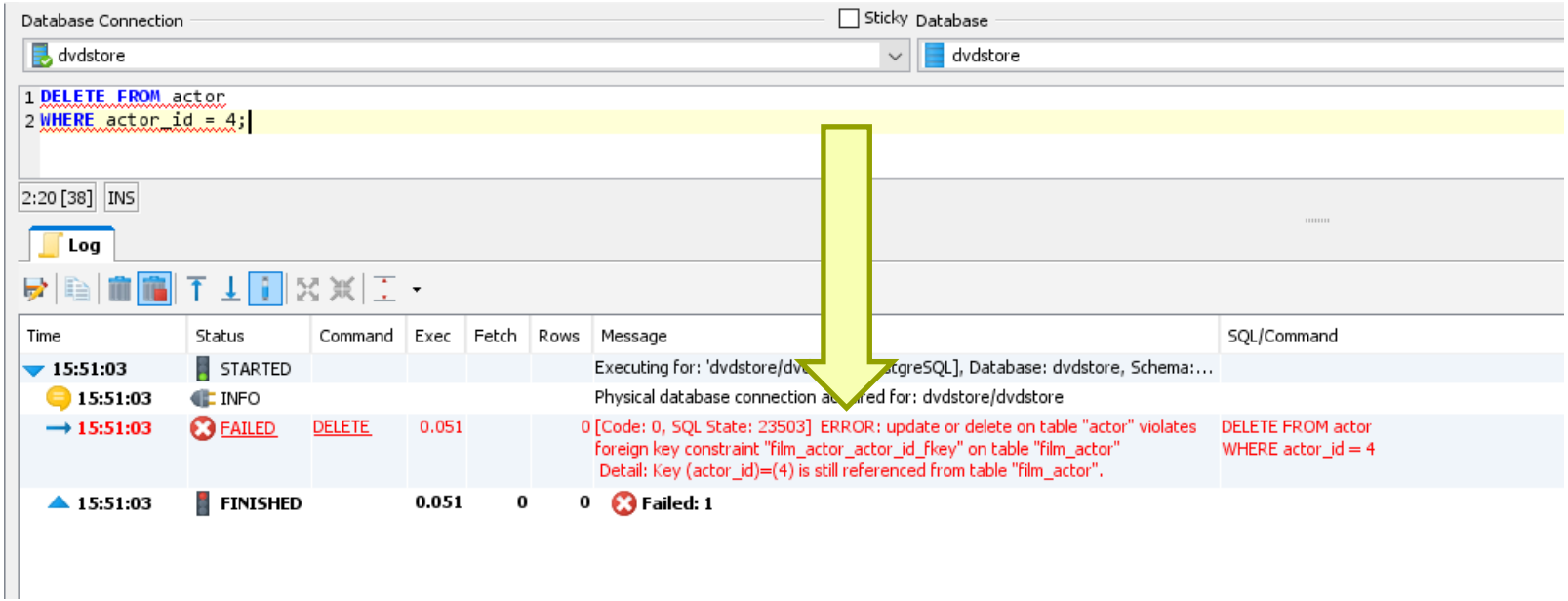
# DELETE statements example

Consider the following example.

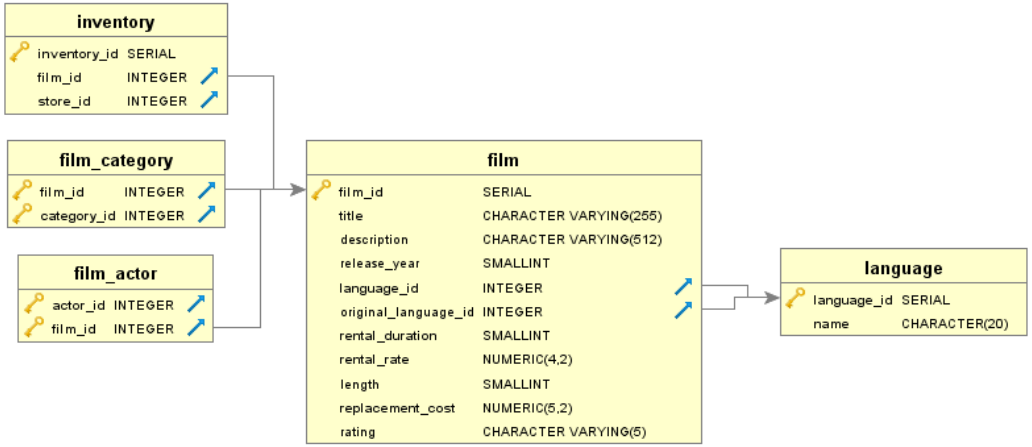
```
DELETE FROM film_actor
WHERE
actor_id = 2;
```

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

# Referential Integrity



# 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

# Constraints Demo

# Transactions

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

The following syntax is observed:

**BEGIN TRANSACTION;**

**// Lots of SQL statements.**

**COMMIT TRANSACTION;**

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