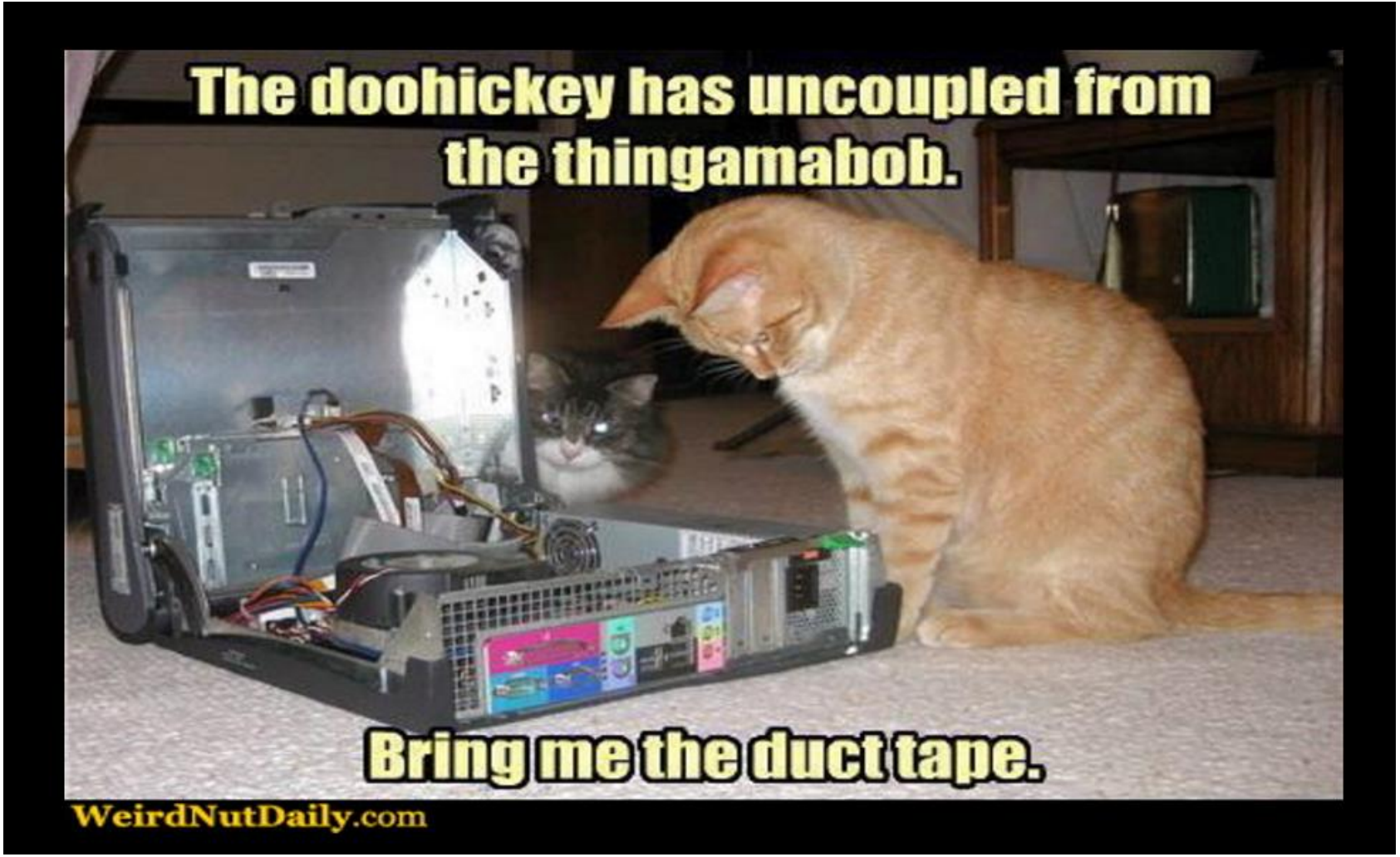

The doohickey has uncoupled from the thingamabob.
Bring me the duct tape.
WeirdNutDaily.com

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

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:

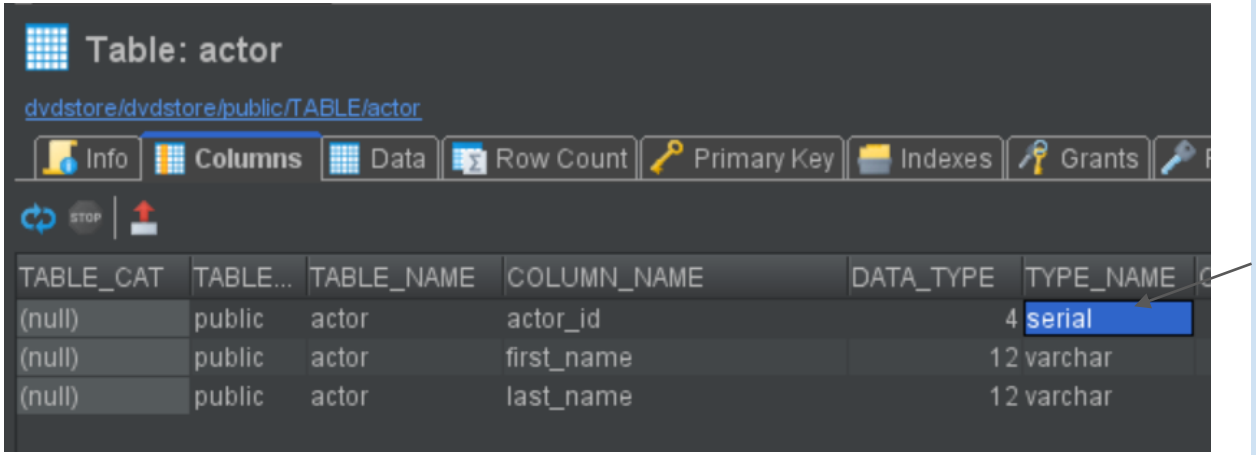
```
INSERT INTO actor (first_name, last_name) VALUES (‘SHIA',’LEBOUF');
```

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.

Let’s write some INSERT statements!

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 = ‘WAHLBERG'
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 = ‘WALBERG';
```

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

UPDATE statements example

Consider the following example:

```
UPDATE actor
SET
   first_name = ‘NICK’,
WHERE
   last_name = ‘WAHLBERG’ AND
   first_name = ‘NICHOLAS’;
```

UPDATE statements example

Consider the following example: A mistake was made for the film Beast Hunchback, it lists Jodie Degeneres as an actor, but it was actually Gina Degeneres who was the star.. Fix it!

~~
```
UPDATE actor
SET
   first_name = 'GINA',
WHERE
   last_name = 'DEGENERES' AND
   first_name = 'JODIE';
```
~~

```
UPDATE film
SET
   ??????
```

UPDATE statements example

Consider the following example: A mistake was made for the film Beast Hunchback, is lists Jodie Degeneres as an actor, but it was actually Gina Degeneres who stared in it.. Fix it!

```
SELECT a.actor_id,
       a.last_name, a.first_name
  FROM actor a
  WHERE a.last_name = 'DEGENERES';
```

Log | 1: actor [3]

*	actor_id	last_name	first_name
1	41	DEGENERES	JODIE
2	107	DEGENERES	GINA
3	166	DEGENERES	NICK

Want to be able to do this in 1 SELECT statement!

UPDATE statements example

Consider the following example: A mistake was made for the film Beast Hunchback, is lists Jodie Degeneres as an actor, but it was actually Gina Degeneres who stared in it.. Fix it!

```
UPDATE film_actor
  SET actor_id = (
      SELECT a.actor_id
         FROM actor a
         WHERE a.first_name = ‘GINA’
         AND a.last_name = ‘DEGENERES’
      )
  WHERE film_id = (
       SELECT f.film_id
         FROM film f
         WHERE f.title = ‘BEAST HUNCHBACK’
   ) AND
  actor_id = (
       SELECT a.actor_id
         FROM actor a
         WHERE a.first_name = 'JODIE' AND a.last_name = 'DEGENERES’
   );
```

Let’s write some UPDATE statements!

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!

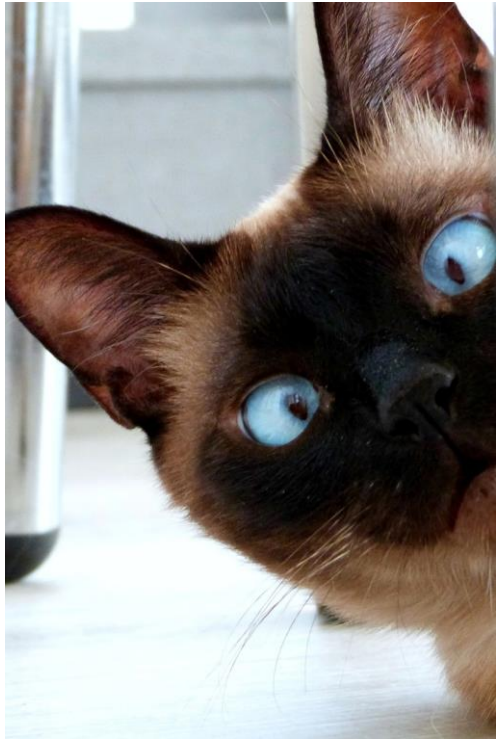
DELETE statements example

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.

Let’s write some DELETE statements!



Referential Integrity

Database Connection: dvdstore | Sticky | Database: dvdstore

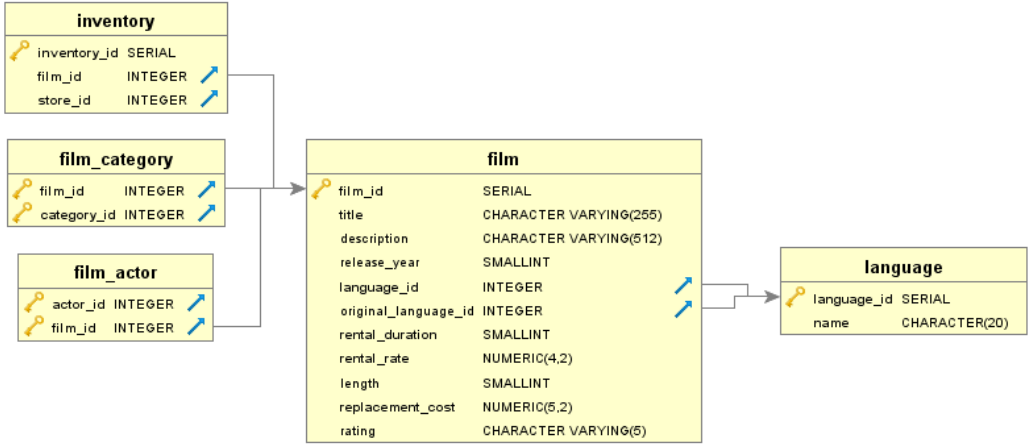
```
1 DELETE FROM actor
2 WHERE actor_id = 4;
```

2:20 [38] INS

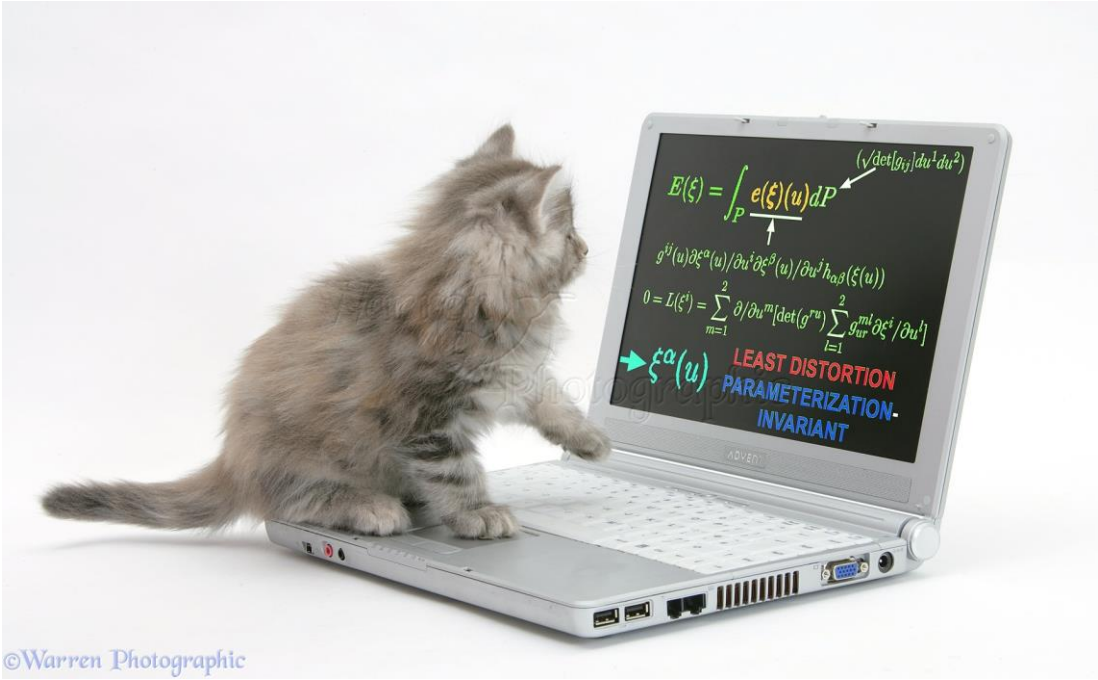
Log

Time	Status	Command	Exec	Fetch	Rows	Message	SQL/Command
15:51:03	STARTED					Executing for: 'dvdstore/dv[illegible]stgreSQL], Database: dvdstore, Schema:...	
15:51:03	INFO					Physical database connection ac[illegible]red for: dvdstore/dvdstore	
15:51:03	FAILED	DELETE	0.051		0	[Code: 0, SQL State: 23503] ERROR: update or delete on table "actor" violates foreign key constraint "film_actor_actor_id_fkey" on table "film_actor" Detail: Key (actor_id)=(4) is still referenced from table "film_actor".	DELETE FROM actor WHERE actor_id = 4
15:51:03	FINISHED		0.051	0	0	Failed: 1	

Referential Integrity



Let’s code!


$E(\xi) = \int_P e(\xi)(u) dP$
$(\sqrt{\det[g_{ij}]} du^1 du^2)$
$g^{ij}(u)\partial\xi^\alpha(u)/\partial u^i \partial\xi^\beta(u)/\partial u^j h_{\alpha\beta}(\xi(u))$
$0 = L(\xi^i) = \sum_{m=1}^{2} \partial/\partial u^m [\det(g^{ru}) \sum_{i=1}^{2} g^{mi}_{ur} \partial\xi^i/\partial u^l]$
$\xi^\alpha(u)$
LEAST DISTORTION
PARAMETERIZATION-
INVARIANT
©Warren Photographic

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;

// 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**.

Transactions and the ACID test

Atomicity – either all statements occur, or none occur.

Consistency – the transaction leaves the database in a consistent state at the end.

Isolation – Execution of transaction results as if operations were executed serially.

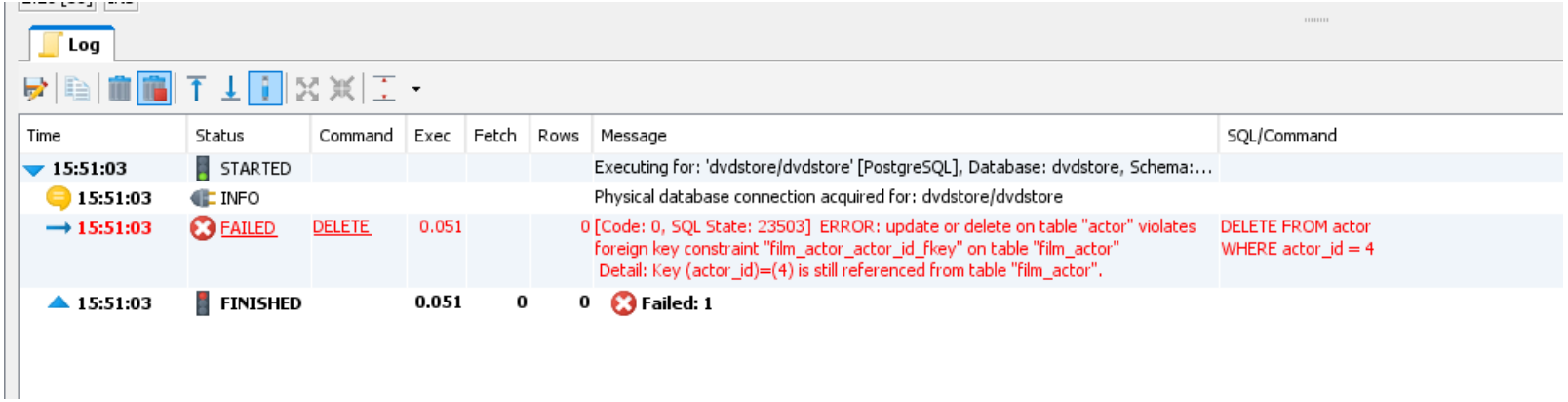
Durability – Once transaction is committed, it will remain so.

Objectives

- INSERT
- DELETE
- UPDATE

Objectives

- INSERT
- DELETE
- UPDATE
- Constraints and referential integrity



Log

Time	Status	Command	Exec	Fetch	Rows	Message	SQL/Command
15:51:03	STARTED					Executing for: 'dvdstore/dvdstore' [PostgreSQL], Database: dvdstore, Schema:...	
15:51:03	INFO					Physical database connection acquired for: dvdstore/dvdstore	
15:51:03	FAILED	DELETE	0.051		0	[Code: 0, SQL State: 23503] ERROR: update or delete on table "actor" violates foreign key constraint "film_actor_actor_id_fkey" on table "film_actor" Detail: Key (actor_id)=(4) is still referenced from table "film_actor".	DELETE FROM actor WHERE actor_id = 4
15:51:03	FINISHED		0.051	0	0	Failed: 1	

Objectives

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

```sql
BEGIN TRANSACTION;

CREATE TABLE country (
    code character(3) NOT NULL,
    name varchar(64) NOT NULL,
    continent varchar(64) NOT NULL,
    region varchar(64) NOT NULL,
    surfacearea real NOT NULL,
    indepyear smallint,
    population integer NOT NULL,
    lifeexpectancy real,
    gnp numeric(10,2),
    gnpold numeric(10,2),
    localname varchar(64) NOT NULL,
    governmentform varchar(64) NOT NULL,
    headofstate varchar(64),
    capital integer,
    code2 character(2) NOT NULL,
    CONSTRAINT pk_country_code PRIMARY KEY (code),
    CONSTRAINT country_continent_check CHECK ((continent = 'Asia') OR (continent :
);

```