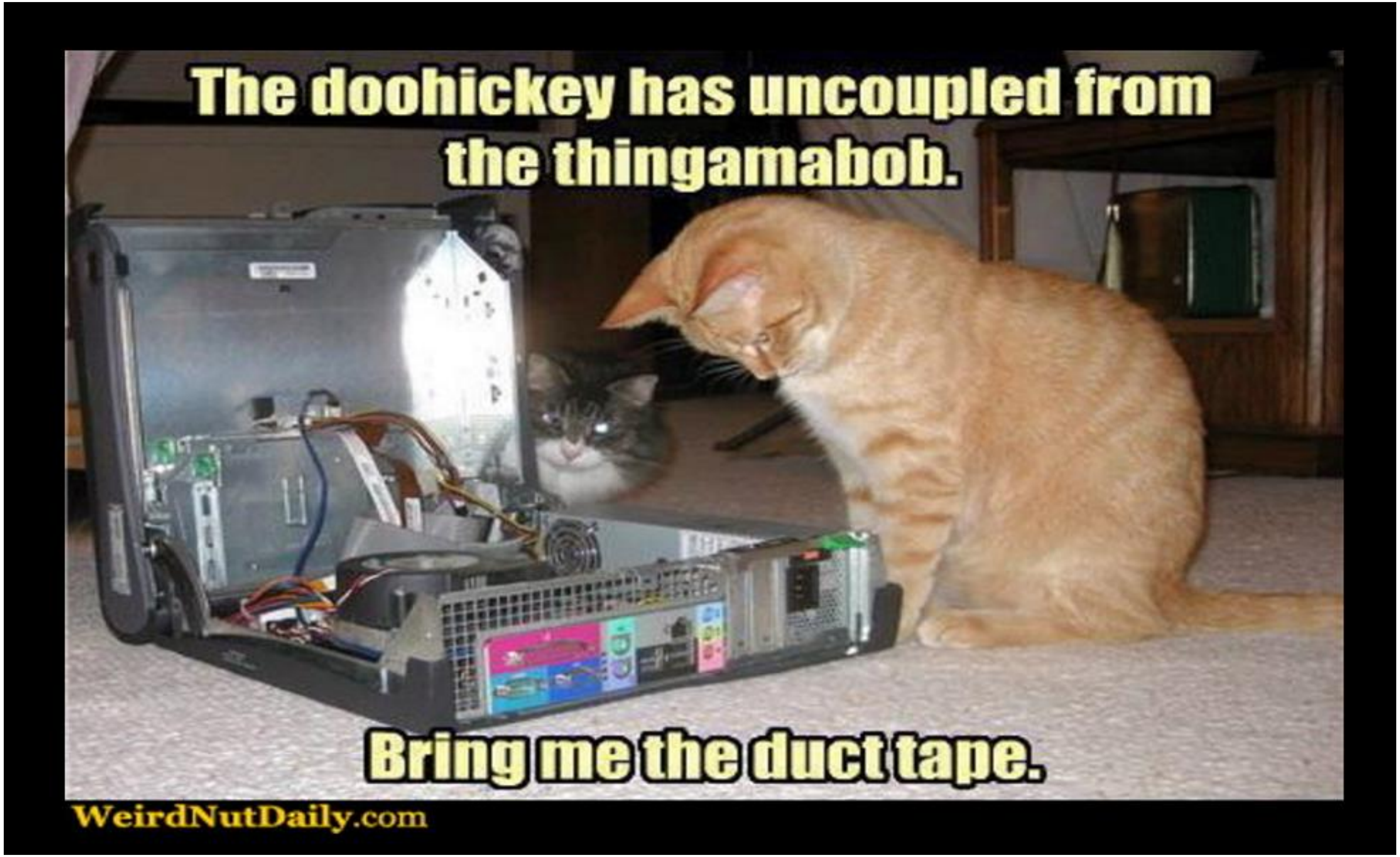

The doohickey has uncoupled from the thingamabob.
Bring me the duct tape.
WeirdNutDaily.com

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

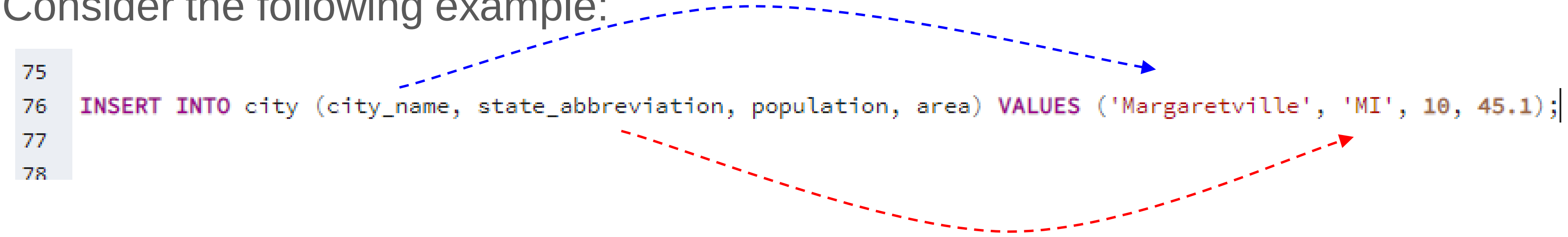
You can use the INSERT statement to insert 1 row into the database. The following pattern is used:

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

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

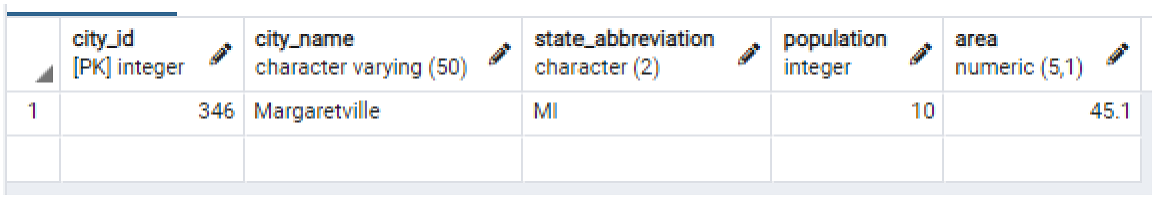
INSERT statements example

Consider the following example:



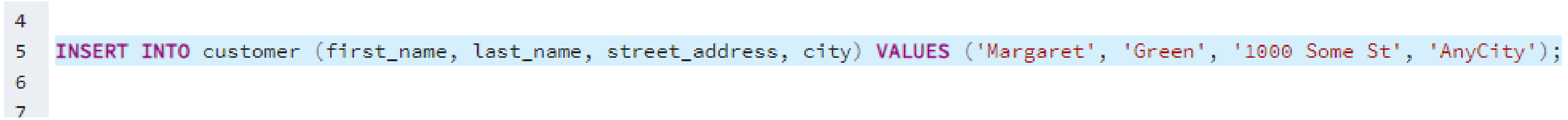
```sql
INSERT INTO city (city_name, state_abbreviation, population, area) VALUES ('Margaretville', 'MI', 10, 45.1);
```

In English, this translates to insert a new row in the table city, on this new row the values for city_name will be “Margaretville”, the state will be “MI”, the population will be 10 and the area will be 45.1



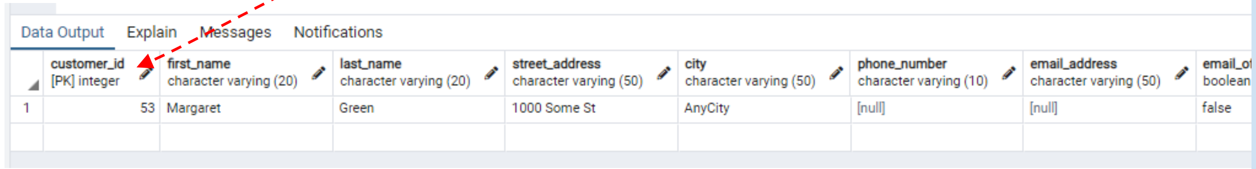
	city_id [PK] integer	city_name character varying (50)	state_abbreviation character (2)	population integer	area numeric (5,1)
1	346	Margaretville	MI	10	45.1

INSERT statements example



```
INSERT INTO customer (first_name, last_name, street_address, city) VALUES ('Margaret', 'Green', '1000 Some St', 'AnyCity');
```

Note that in this example, we only specified four columns and did not specify that a value be inserted for person_id, phone_number or email_address.



	customer_id [PK] integer	first_name character varying (20)	last_name character varying (20)	street_address character varying (50)	city character varying (50)	phone_number character varying (10)	email_address character varying (50)	email_o boolean
1	53	Margaret	Green	1000 Some St	AnyCity	[null]	[null]	false

- customer_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**.

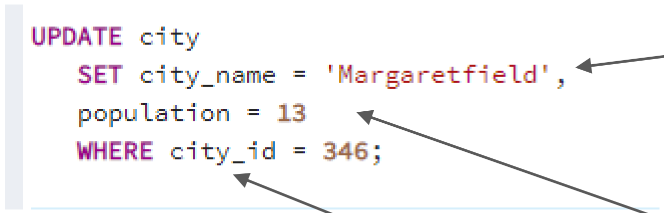
```
UPDATE [table name]

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

WHERE ...
```

UPDATE statements example

Consider the following example:

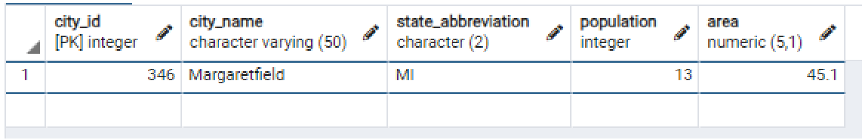


```
UPDATE city
   SET city_name = 'Margaretfield',
   population = 13
   WHERE city_id = 346;
```

In here, we have changed the value for 2 columns (city_name and population) but only for the row with an city_id of 346.

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

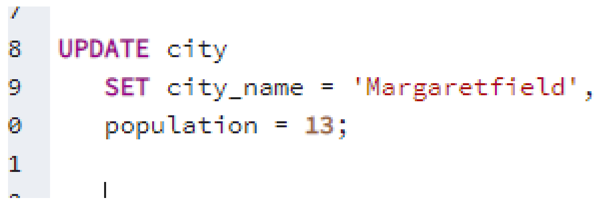
The syntax for structuring the WHERE statement remains unchanged.



	city_id [PK] integer	city_name character varying (50)	state_abbreviation character (2)	population integer	area numeric (5,1)
1	346	Margaretfield	MI	13	45.1

UPDATE statements example

Consider the following example:

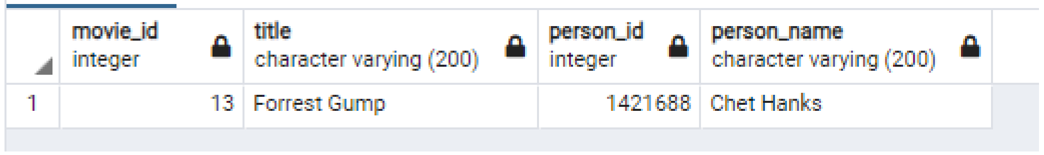


```
UPDATE city
   SET city_name = 'Margaretfield',
   population = 13;
```

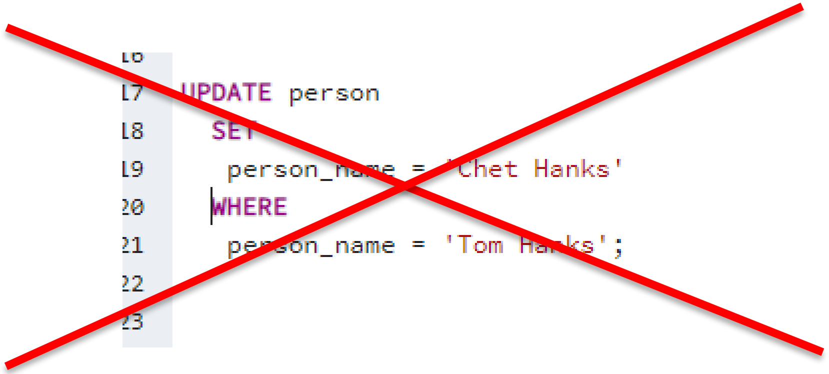
We have just set every city name to Margaretfield and their population to 13!!!

UPDATE statements example

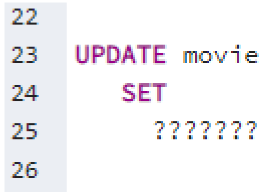
Consider the following example: A mistake was made for the movie Forrest Gump, it lists Chet Hanks as an actor, but it was actually Tom Hanks who was the star. Fix it!



	movie_id integer	title character varying (200)	person_id integer	person_name character varying (200)
1	13	Forrest Gump	1421688	Chet Hanks



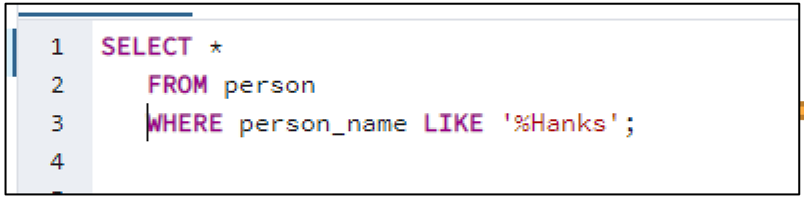
```
UPDATE person
  SET
   person_name = 'Chet Hanks'
  WHERE
   person_name = 'Tom Hanks';
```



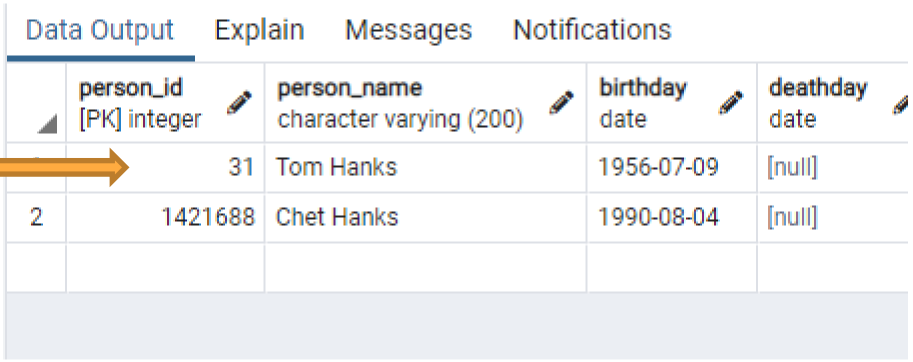
```
UPDATE movie
   SET
      ???????
```

UPDATE statements example

Consider the following example: A mistake was made for the movie Forrest Gump, it lists Chet Hanks as an actor, but it was actually Tom Hanks who was the star.



```sql
SELECT *
    FROM person
    WHERE person_name LIKE '%Hanks';
```

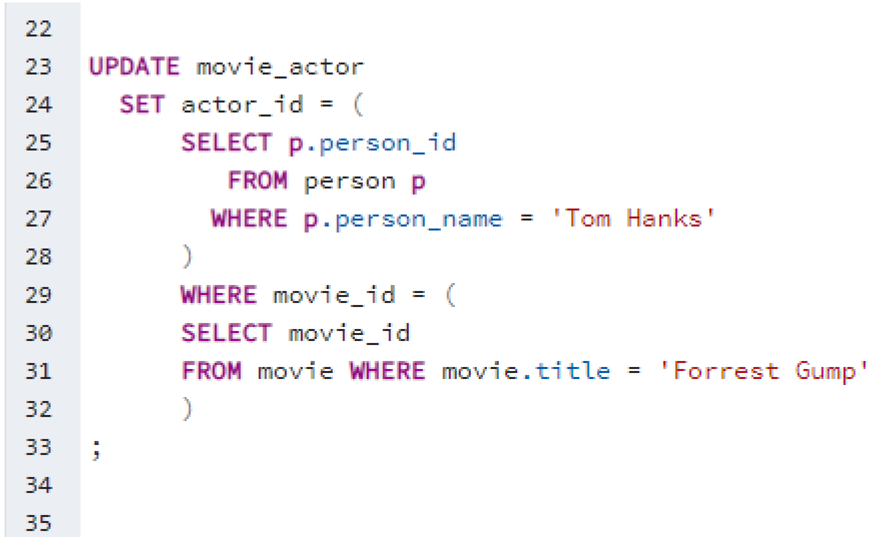


	person_id [PK] integer	person_name character varying (200)	birthday date	deathday date
	31	Tom Hanks	1956-07-09	[null]
2	1421688	Chet Hanks	1990-08-04	[null]

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

UPDATE statements example

Consider the following example: A mistake was made for the movie Forrest Gump, it lists Chet Hanks as an actor, but it was actually Tom Hanks who was the star… Fix it!



```

UPDATE movie_actor
  SET actor_id = (
      SELECT p.person_id
         FROM person p
        WHERE p.person_name = 'Tom Hanks'
      )
      WHERE movie_id = (
      SELECT movie_id
      FROM movie WHERE movie.title = 'Forrest Gump'
      )
;


```

Let’s write some UPDATE statements!

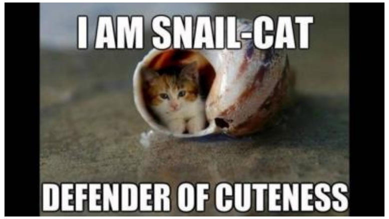
DELETE statements

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

DELETE FROM [table name]

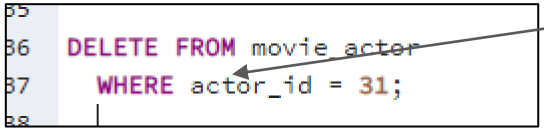
WHERE …

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



DELETE statements example

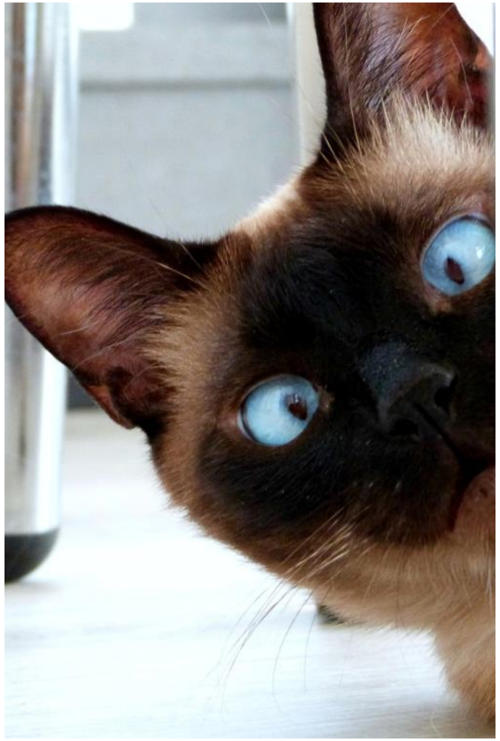
Consider the following example.



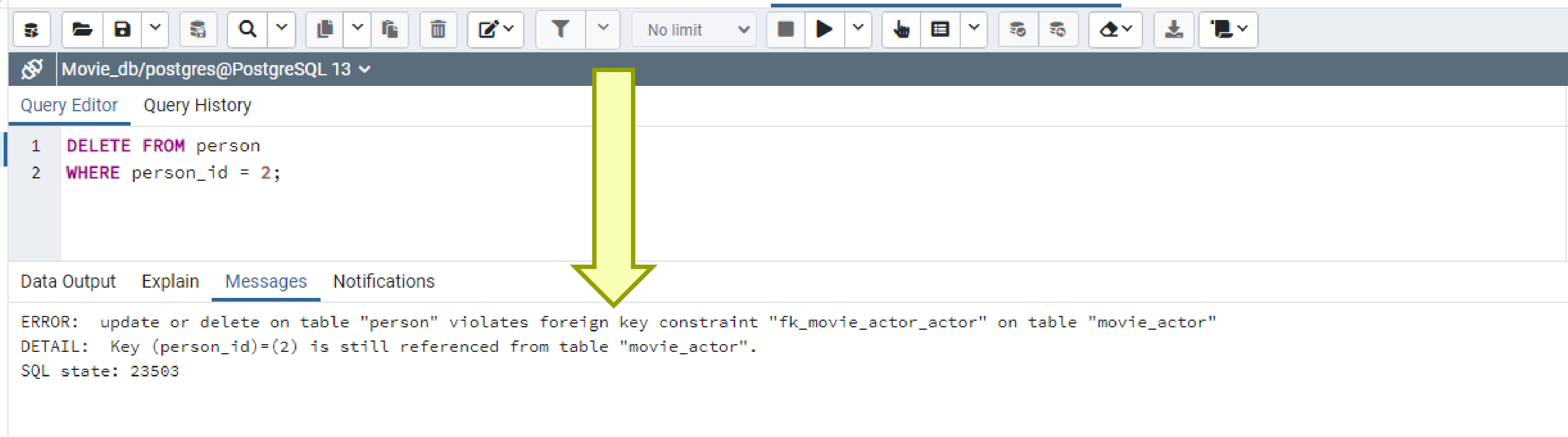
```
DELETE FROM movie_actor
  WHERE actor_id = 31;
```

Here, we are deleting every row that has an actor_id of 31.

Let’s write some DELETE statements!



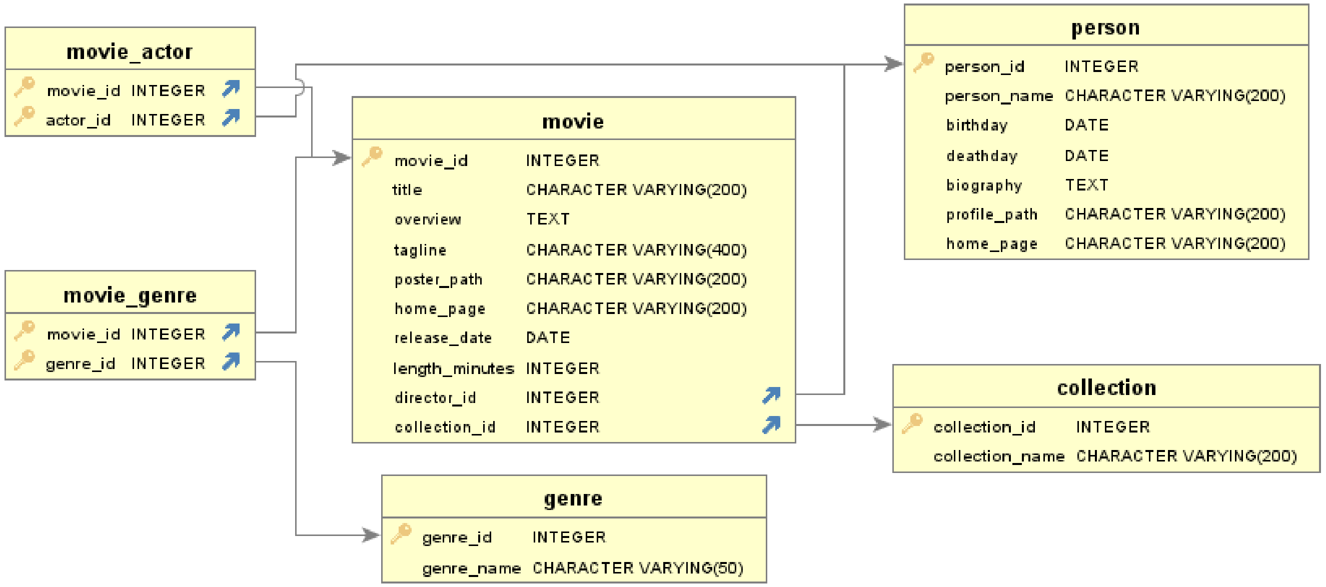
Referential Integrity



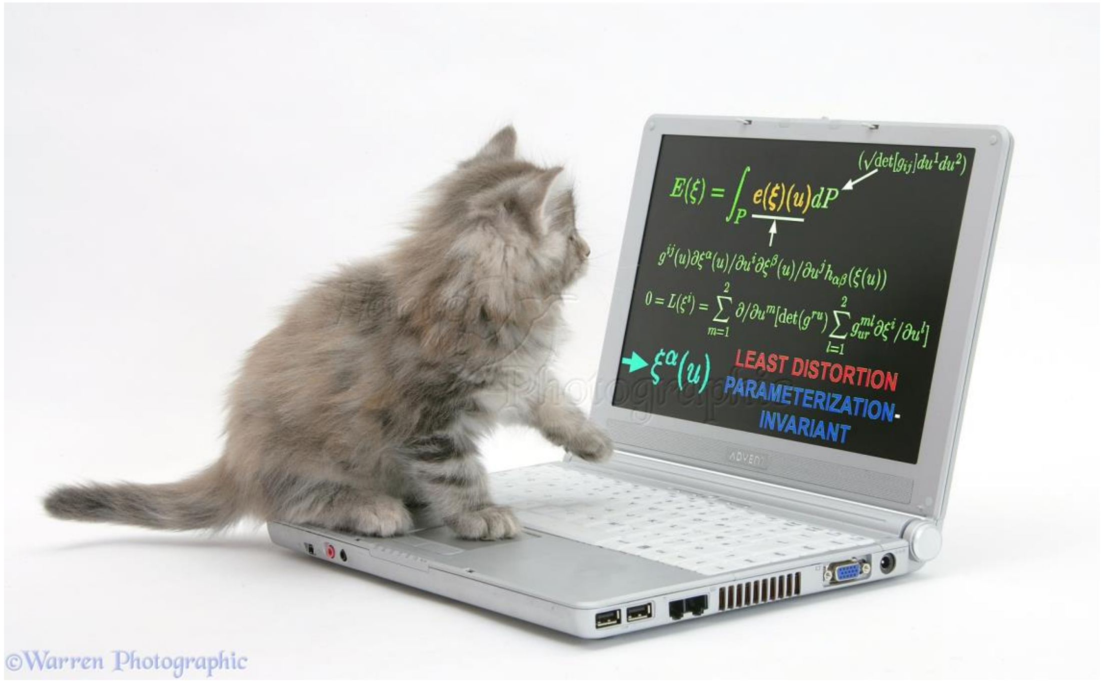
```sql
DELETE FROM person
WHERE person_id = 2;
```

```
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



Let’s code!


$E(\xi) = \int_P e(\xi)(u)dP$
$(\sqrt{\det[g_{ij}]}du^1du^2)$
$g^{ij}(u)\partial\xi^\alpha(u)/\partial u^i\partial\xi^\beta(u)/\partial u^j h_{\alpha\beta}(\xi(u))$
$0 = L(\xi^i) = \sum_{m=1}^{2} \partial/\partial u^m[\det(g^{ru}) \sum_{l=1}^{2} g^{ml}_{ur}\partial\xi^i/\partial u^l]$
$\xi^\alpha(u)$
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; -- 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**.

Transactions and the ACID test

Atomicity – either all statements occur, or none occur.

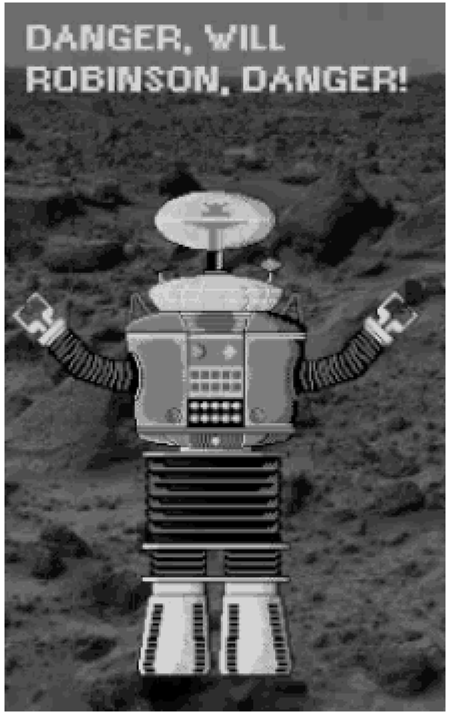
Consistency – the transaction leaves the database in a consistent state at the end.

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

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

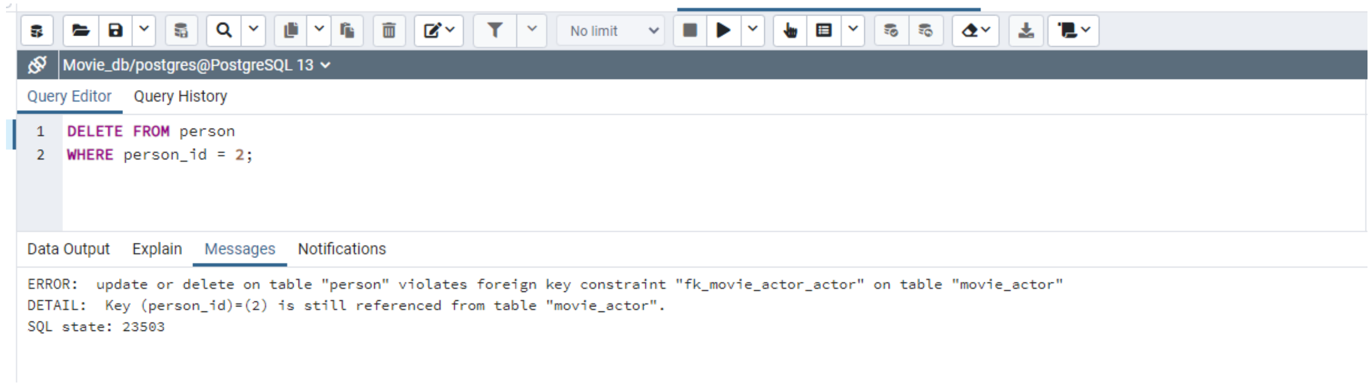
Objectives

- INSERT
- DELETE
- UPDATE



Objectives

- INSERT
- DELETE
- UPDATE
- Constraints and referential integrity

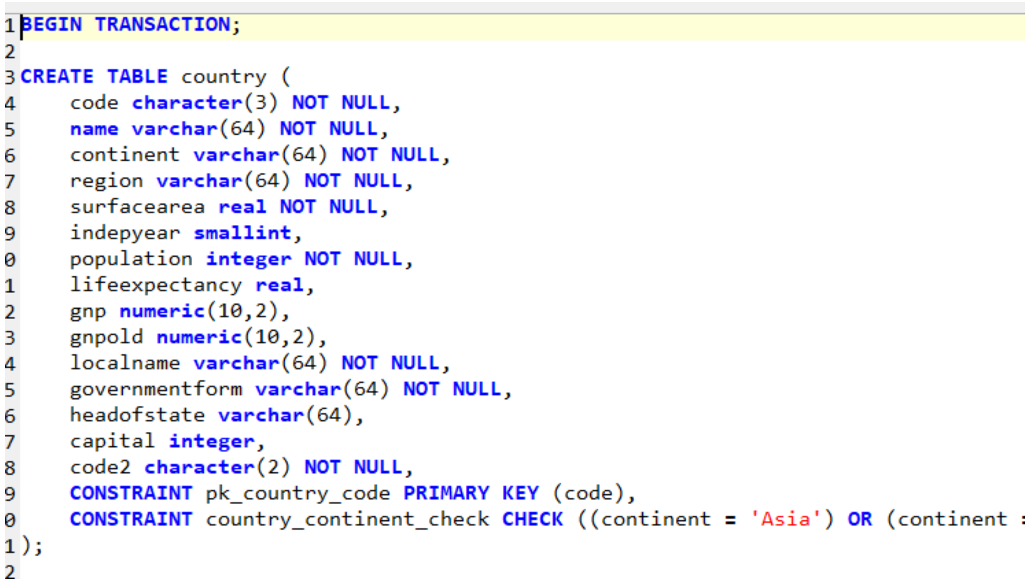


```sql
DELETE FROM person
WHERE person_id = 2;
```

```
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
```

Objectives

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



```
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 :
);
```