Design

ERIN KEITH

Goals

1. Background
2. Relations in SQL
3. Introduction to PostgreSQL

SQL

There is a current standard for SQL, called SQL-99.

Most commercial database management systems implement something similar, but not identical to, the standard.

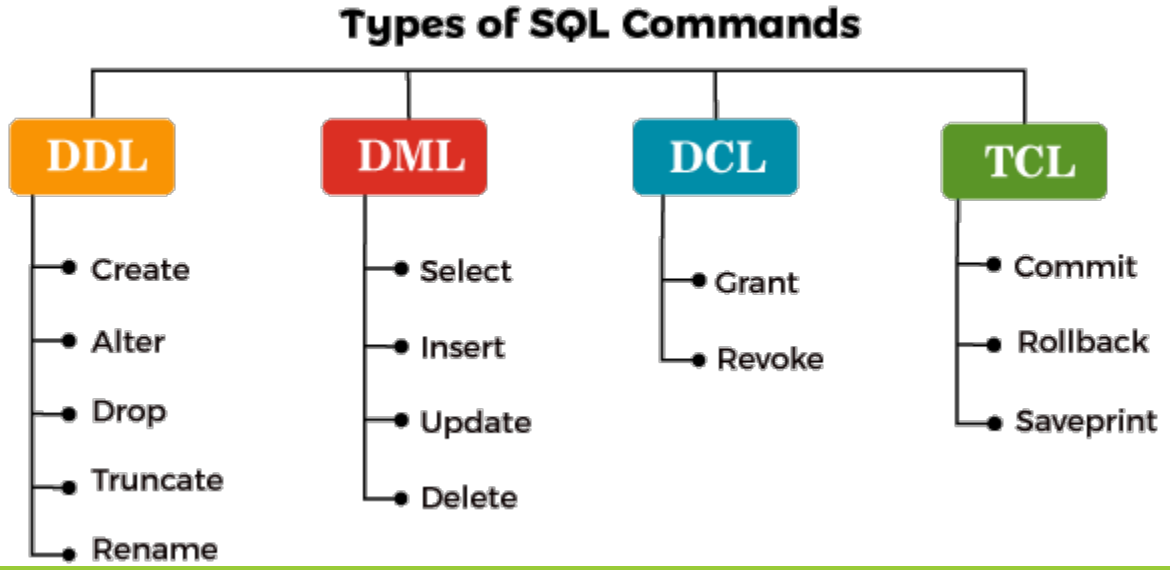
SQL Commands

DDL – Data Definition

DML – Data Manipulation

DCL – Data Control Language

TCL – Transaction Control Language



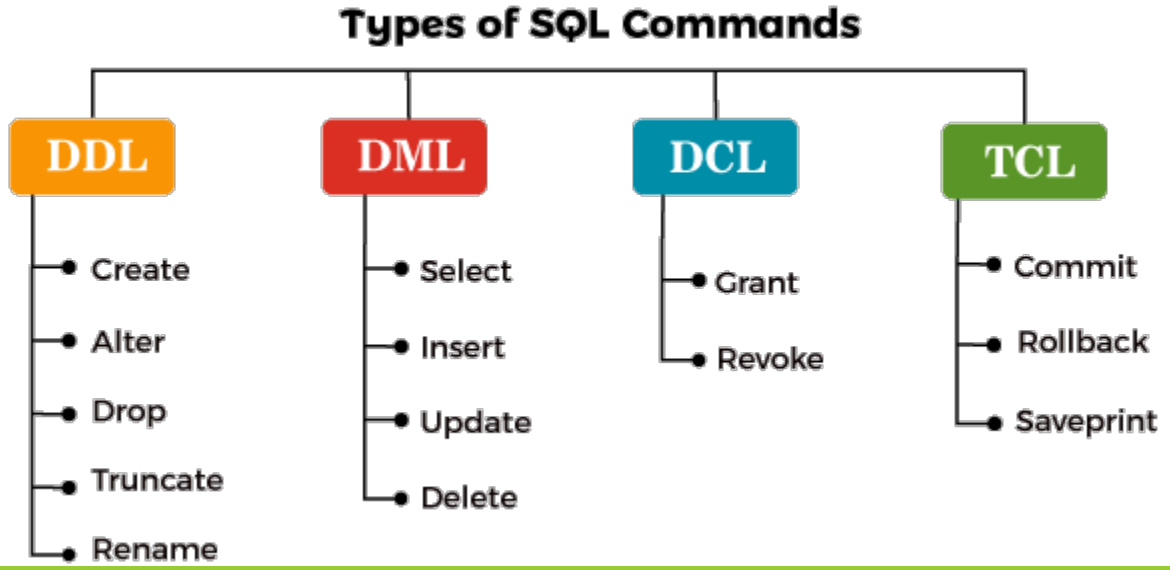
SQL Commands

DDL – tables

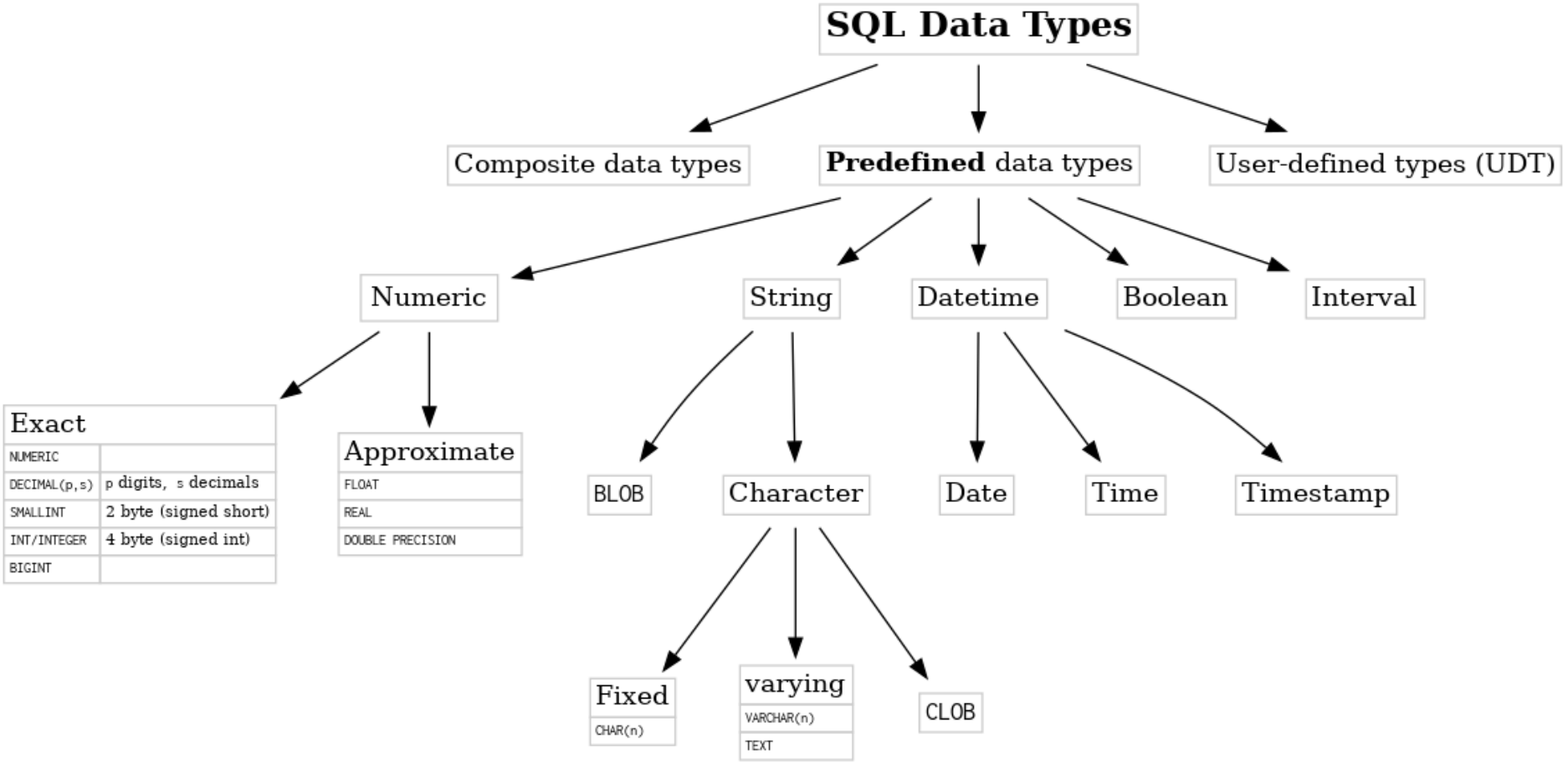
DML – queries

DCL – users

TCL – transactions



SQL Data Types



Create Table

SCHEMA

```
Movies(
     title:string,
     year:integer,
     length:integer,
     genre:string,
     studioName:string,
     producerC#:integer
)
```

SQL

```
CREATE TABLE Movies (
     title          CHAR(100),
     year           INT,
     length         INT,
     genre          CHAR(10),
     studioName     CHAR(30),
     producerC#     INT
);
```

QUESTION 2

SCHEMA

```
MovieStar(
     name:string,
     address:string,
     gender:char,
     birthdate:date
)
```

SQL

```
CREATE TABLE MovieStar (
    name       CHAR(30),
    address    VARCHAR(255),
    gender     CHAR(1),
    birthdate DATE
);
```

Keys

- cannot be **`NULL`**
- the set of keys must be *unique*
  - otherwise the insert operation will result in an error

```
CREATE TABLE MovieStar (
    name CHAR(30) PRIMARY KEY,
    address VARCHAR(255),
    gender CHAR(1),
    birthdate DATE
);
```

OR

```
CREATE TABLE MovieStar (
    name CHAR(30),
    address VARCHAR(255),
    gender CHAR(1),
    birthdate DATE,
    PRIMARY KEY (name)
);
```

Keys

- cannot be **NULL**
- the set of keys must be *unique*
  - otherwise the insert operation will result in an error

```
CREATE TABLE Movies (
    title       CHAR(100),
    year        INT,
    length      INT,
    genre       CHAR(10),
    studioName  CHAR(30),
    producerC#  INT,
    PRIMARY KEY (title, year)
);
```

Delete Table

```
DROP TABLE <TableName>;
```

Alter Table

DROP or **ADD** an attribute on a table

ALTER TABLE <TableName> ADD <Attribute>

```
ALTER TABLE MovieStar ADD phone CHAR(16);
ALTER TABLE MovieStar DROP birthdate;
```

Default Values

```
birthdate DATE DEFAULT DATE '0000-00-00'
```

Database Servers

There are often many services involved in your system

We'll talk about how to put it all together soon, but for now we'll focus on

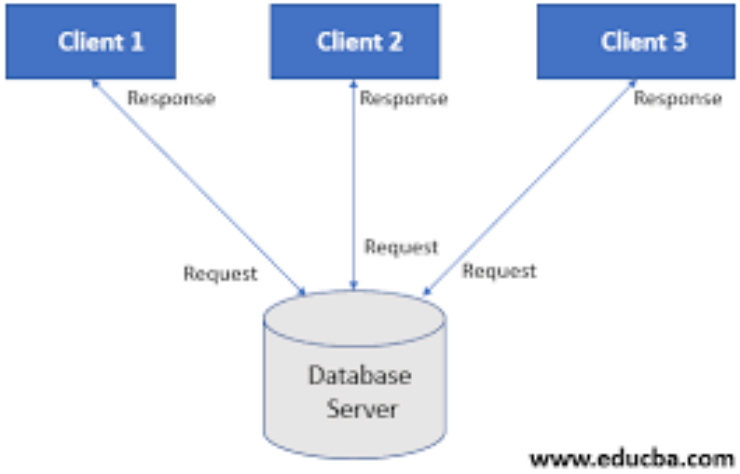
> *Database Servers.* These processes run the DBMS and perform queries and modifications at the request of the application servers.

The **D**atab**B**ase **M**anagement **S**ystem is a service that we can **start** or **stop** running.

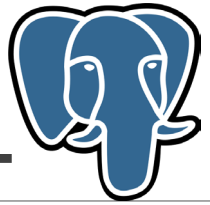
When the service is running, we can send it queries or commands and receive the results.

PostgreSQL

- https://www.postgresql.org/docs/current/tutorial-arch.html



- The database server can be running locally!
- PostgreSQL port: 5432

PostgreSQL 

- Clients

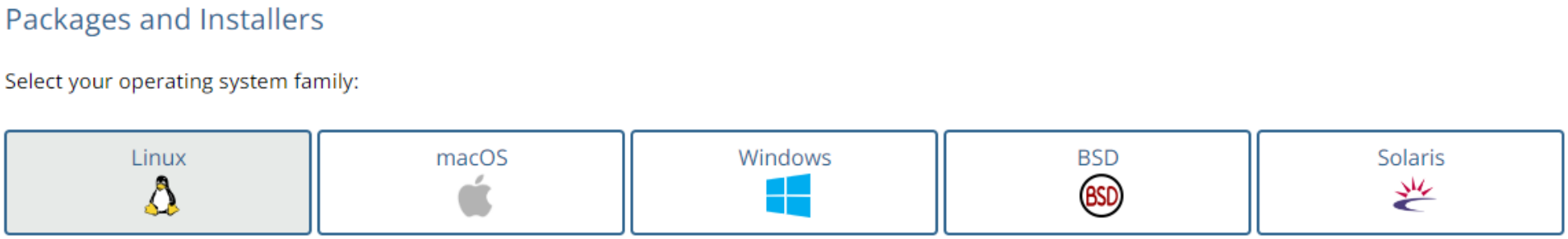
CLI – typing SQL commands in the terminal

GUI – like an IDE for the environment

Applications – code programmatically connects and interacts with the database

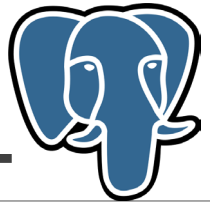
PostgreSQL

- Download and install: https://www.postgresql.org/download/



- Admin users: https://www.postgresql.org/docs/current/notation.html
- You may have to add the bin dir to the **`PATH`** env var
- You may have to add the absolute path to the binary folder to the **`PATH`** environment variable

PostgreSQL



- You can login as the admin user

```
> psql -U postgres
```

- You can create a user which matches your system username

```
C:\Users\Work\OneDrive\Documents\UNR\CS457> createuser -U postgres -P -s -e Work
```

- Then you can create a database and access it

```
> createdb Movies

> psql Movies
```

PostgreSQL Activity

PostgreSQL Activity

CS 457/657
Fall 2023 – PostgreSQL Activity

Student Names:

ALL QUESTIONS ARE REFERRING TO THE PostgreSQL DATABASE SYSTEM.

How do you create a new database?

Terminal: createdb dbName
OR
Invoke DBMS: CREATE DATABASE dbName
*name should be on same line

What are possible issues you might run into while attempting to create a new database?

Don't have permission
Incorrect PATH env var
Incorrect installation
Insufficient disk space
Database exists
Server is not running (or wrong port, etc.)

How do you delete a database?

dropdb dbName
OR
DROP DATABASE dbName

How do you access a database?
psql dbName

more on the back →

How do you create a table?

```
CREATE TABLE tblName(
        name VARCHAR(80), --a descriptive comment
        --
);
```

How do indicate a primary key?

```
CREATE TABLE tblName(
        name VARCHAR(80) PRIMARY KEY, --a descriptive comment
        --
);
```

How do you indicate a foreign key?

```
CREATE TABLE tblName(
        name VARCHAR(80) PRIMARY KEY, --a descriptive comment
        partNum INT REFERENCES tbl(col),
        --
        --FOREIGN KEY (partNum) REFERENCES tbl(col)
);
```

How do you add rows to a table?

```
INSERT INTO tblName VALUES("Erin", 134243);
*multiple rows???
```

How do you query a table?

```
SELECT * FROM tblName
```

How do you join two tables?

```
SELECT * FROM a b WHERE a.id = b.id;
SELECT * FROM t1 JOIN t2 on t1.partNum = t2.partNum
```

PostgreSQL Movies

CS 457/657
Fall 2023 – PostgreSQL Movies

Example 2.21: Consider the two relations from our running movie database:

```
Movies(title, year, length, genre, studioName, producerC#)
MovieExec(name, address, cert#, netWorth)
```

How do you create the new database?

createdb Movies

How do you access the database?

sql Movies

How do you create the MovieExec table?

```
CREATE TABLE MovieExec(
        name VARCHAR(80),
        address VARCHAR(255),
        cert_num INT PRIMARY KEY,
        net_worth INT
);
```

How do you create the Movie table?

```
CREATE TABLE Movie(
        title VARCHAR(80),
        year INT,
        length INT,
        genre VARCHAR(80),
        studio_name VARCHAR(80),
        producer_num INT REFERENCES MovieExec (cert_num),
        PRIMARY KEY (title, year)
);
```

How do you get the name and net worth for producers of movies made since the turn of the century?

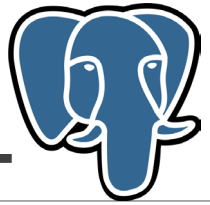
SQL Query Practice

Example 2.21: Consider the two relations from our running movie database:

```
Movies(title, year, length, genre, studioName, producerC#)
MovieExec(name, address, cert#, netWorth)
```

How do you get the name and net worth for producers of movies made since the turn of the century?

PostgreSQL



1.

```
Movies=# CREATE TABLE MovieExec(name VARCHAR(80), address VARCHAR(255), cert_num INT PRIMARY KEY, net_worth INT);
CREATE TABLE
```

2.

```
Movies=# CREATE TABLE Movie(title VARCHAR(80), year INT, length INT, genre VARCHAR(80), studio_name VARCHAR(80), produce
r_num INT REFERENCES MovieExec (cert_num), PRIMARY KEY (title, year));
CREATE TABLE
```

- You can show the tables!

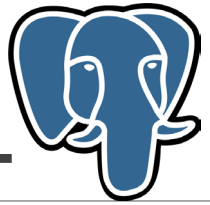
```
Movies=# \dt
         List of relations
 Schema |   Name    | Type  | Owner
--------+-----------+-------+-------
 public | movie     | table | Work
 public | movieexec | table | Work
(2 rows)
```

PostgreSQL

- You can show the columns!

```
Movies=# \d movie;
                        Table "public.movie"
    Column    |         Type          | Collation | Nullable | Default
--------------+-----------------------+-----------+----------+---------
 title        | character varying(80) |           | not null |
 year         | integer               |           | not null |
 length       | integer               |           |          |
 genre        | character varying(80) |           |          |
 studio_name  | character varying(80) |           |          |
 producer_num | integer               |           |          |
Indexes:
    "movie_pkey" PRIMARY KEY, btree (title, year)
Foreign-key constraints:
    "movie_producer_num_fkey" FOREIGN KEY (producer_num) REFERENCES movieexec(cert_num)
```

```
Movies=# \d movieexec
                   Table "public.movieexec"
  Column   |          Type          | Collation | Nullable | Default
-----------+------------------------+-----------+----------+---------
 name      | character varying(80)  |           |          |
 address   | character varying(255) |           |          |
 cert_num  | integer                |           | not null |
 net_worth | integer                |           |          |
Indexes:
    "movieexec_pkey" PRIMARY KEY, btree (cert_num)
Referenced by:
    TABLE "movie" CONSTRAINT "movie_producer_num_fkey" FOREIGN KEY (producer_num) REFERENCES movieexec(cert_num)
```

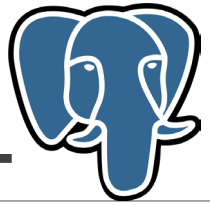
PostgreSQL 

1.

```
Movies=# INSERT INTO
Movies-# MovieExec(name, address, cert_num, net_worth)
Movies-# VALUES
Movies-#   ('Gary Kurtz', 'London, England', 12345, 7000000),
Movies-#   ('Mark Johnson', 'Washington DC', 67890, 17500000000),
Movies-#   ('Lorne Michaels', 'New York, NY', 99999, 500000000);
ERROR:  integer out of range
```

- Uh oh!

PostgreSQL

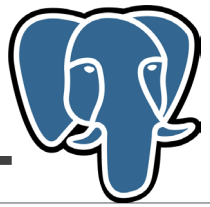


1.

```
Movies=# ALTER TABLE MovieExec ALTER COLUMN net_worth TYPE BIGINT;
ALTER TABLE
```

2.

```
Movies=# \d movieexec
                    Table "public.movieexec"
  Column   |          Type          | Collation | Nullable | Default
-----------+------------------------+-----------+----------+---------
 name      | character varying(80)  |           |          |
 address   | character varying(255) |           |          |
 cert_num  | integer                |           | not null |
 net_worth | bigint                 |           |          |
Indexes:
    "movieexec_pkey" PRIMARY KEY, btree (cert_num)
Referenced by:
    TABLE "movie" CONSTRAINT "movie_producer_num_fkey" FOREIGN KEY (producer_num) REFE
RENCES movieexec(cert_num)
```

PostgreSQL 

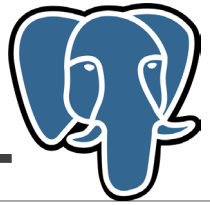
1.

```
Movies=# INSERT INTO
Movies-# MovieExec(name, address, cert_num, net_worth)
Movies-# VALUES
Movies-#   ('Gary Kurtz', 'London, England', 12345, 7000000),
Movies-#   ('Mark Johnson', 'Washington DC', 67890, 17500000000),
Movies-#   ('Lorne Michaels', 'New York, NY', 99999, 500000000);
INSERT 0 3
```

2.

```
Movies=# SELECT * FROM MovieExec;
      name       |     address      | cert_num |  net_worth
-----------------+------------------+----------+-------------
 Gary Kurtz      | London, England  |    12345 |     7000000
 Mark Johnson    | Washington DC    |    67890 | 17500000000
 Lorne Michaels  | New York, NY     |    99999 |   500000000
(3 rows)
```

PostgreSQL



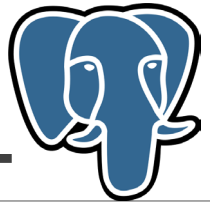
1.

```
Movies=# INSERT INTO
Movies-# Movie(title, year, length, genre, studio_name, producer_num)
Movies-# VALUES
Movies-#   ('Star Wars', 1977, 124, 'SciFi', 'Fox', 12345),
Movies-#   ('Galaxy Quest', 1999, 104, 'comedy', 'DreamWorks', 67890),
Movies-#   ('Wayne''s World', 1992, 95, 'comedy', 'Paramount', 99999);
INSERT 0 3
```

- If we had tried to insert into Movie first

```
Movies=# INSERT INTO
Movies-# Movie(title, year, length, genre, studio_name, producer_num)
Movies-# VALUES
Movies-#   ('Star Wars', 1977, 124, 'SciFi', 'Fox', 12345),
Movies-#   ('Galaxy Quest', 1999, 104, 'comedy', 'DreamWorks', 67890),
Movies-#   ('Wayne''s World', 1992, 95, 'comedy', 'Paramount', 99999);
ERROR:  insert or update on table "movie" violates foreign key constraint "movie_producer_num_fkey"
```

PostgreSQL



- How do you get the name and net worth for producers of movies made since the turn of the century?

1. Join

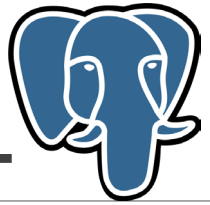
```
Movies=# SELECT * FROM Movie JOIN MovieExec on producer_num = cert_num;
    title     | year | length | genre  | studio_name | producer_num |      name      |     address     | cert_num |  net_worth
--------------+------+--------+--------+-------------+--------------+----------------+-----------------+----------+-------------
 Star Wars    | 1977 |    124 | SciFi  | Fox         |        12345 | Gary Kurtz     | London, England |    12345 |     7000000
 Galaxy Quest | 1999 |    104 | comedy | DreamWorks  |        67890 | Mark Johnson   | Washington DC   |    67890 | 17500000000
 Wayne's World| 1992 |     95 | comedy | Paramount   |        99999 | Lorne Michaels | New York, NY    |    99999 |   500000000
(3 rows)
```

- Add data so we can test our query

```
Movies=# INSERT INTO MovieExec VALUES('Paul Webster', 'United Kingdom', 25327, 5000000);
INSERT 0 1
```

```
Movies=# INSERT INTO Movie VALUES('Pride and Prejudice', 2005, 127, 'period', 'Universal', 25327);
INSERT 0 1
```

PostgreSQL



- How do you get the name and net worth for producers of movies made since the turn of the century?

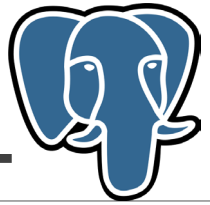
1. Join (combines the table)

```
Movies=# SELECT * FROM Movie JOIN MovieExec on producer_num = cert_num;
       title        | year | length | genre  | studio_name | producer_num |      name      |     address     | cert_num |  net_worth
--------------------+------+--------+--------+-------------+--------------+----------------+-----------------+----------+-------------
 Star Wars          | 1977 |    124 | SciFi  | Fox         |        12345 | Gary Kurtz     | London, England |    12345 |     7000000
 Galaxy Quest       | 1999 |    104 | comedy | DreamWorks  |        67890 | Mark Johnson   | Washington DC   |    67890 | 17500000000
 Wayne's World      | 1992 |     95 | comedy | Paramount   |        99999 | Lorne Michaels | New York, NY    |    99999 |   500000000
 Pride and Prejudice | 2005 |    127 | period | Universal   |        25327 | Paul Webster   | United Kingdom  |    25327 |     5000000
(4 rows)
```

2. Selection (refines the query)

```
Movies=# SELECT * FROM Movie JOIN MovieExec on producer_num = cert_num WHERE year >= 2000;
       title        | year | length | genre  | studio_name | producer_num |     name     |    address     | cert_num | net_worth
--------------------+------+--------+--------+-------------+--------------+--------------+----------------+----------+-----------
 Pride and Prejudice | 2005 |    127 | period | Universal   |        25327 | Paul Webster | United Kingdom |    25327 |   5000000
(1 row)
```

PostgreSQL



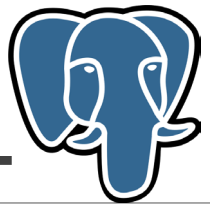
- How do you get the name and net worth for producers of movies made since the turn of the century?

2. Selection (refines the query)

```
Movies=# SELECT * FROM Movie JOIN MovieExec on producer_num = cert_num WHERE year >= 2000;
       title        | year | length | genre  | studio_name | producer_num |     name     |    address     | cert_num | net_worth
--------------------+------+--------+--------+-------------+--------------+--------------+----------------+----------+-----------
 Pride and Prejudice | 2005 |    127 | period | Universal   |        25327 | Paul Webster | United Kingdom |    25327 |   5000000
(1 row)
```

3. Projection (isolate the columns)…

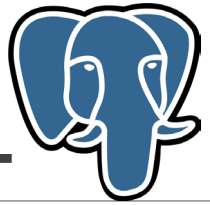
```
Movies=# SELECT name, net_worth FROM Movie JOIN MovieExec ON producer_num = cert_num WHERE year >= 2000;
     name     | net_worth
--------------+-----------
 Paul Webster |   5000000
(1 row)
```

PostgreSQL 

- How do you get the name and net worth for producers of movies made since the turn of the century?
- Join + Selection + Projection

```
Movies=# SELECT name, net_worth FROM Movie JOIN MovieExec ON producer_num = cert_num WHERE year >= 2000;
     name     | net_worth
--------------+-----------
 Paul Webster |   5000000
(1 row)
```

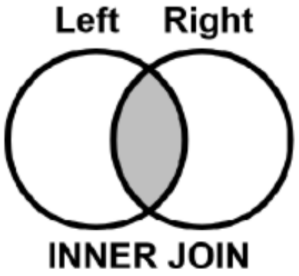
```
SELECT
        name,
        net_worth
FROM Movie JOIN MovieExec ON producer_num = cert_num
WHERE year >= 2000;
```

PostgreSQL 

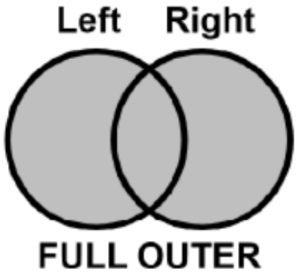
- How do you get the name and net worth for producers of movies made since the turn of the century?
- Join + Selection + Projection (Aliasing)

```
Movies=# SELECT me.name, me.net_worth FROM Movie AS m JOIN MovieExec AS me ON m.producer_num = me.cert_num WHERE m.year >= 2000;
     name     | net_worth
--------------+-----------
 Paul Webster |   5000000
(1 row)
```

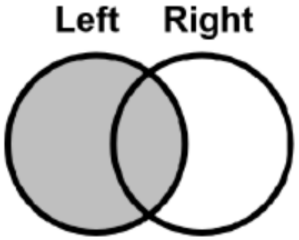
```
SELECT
        me.name,
        me.net_worth
FROM Movie AS m JOIN MovieExec AS me ON m.producer_num = me.cert_num
WHERE m.year >= 2000;
```



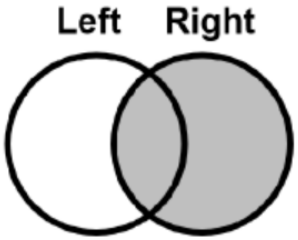
INNER JOIN



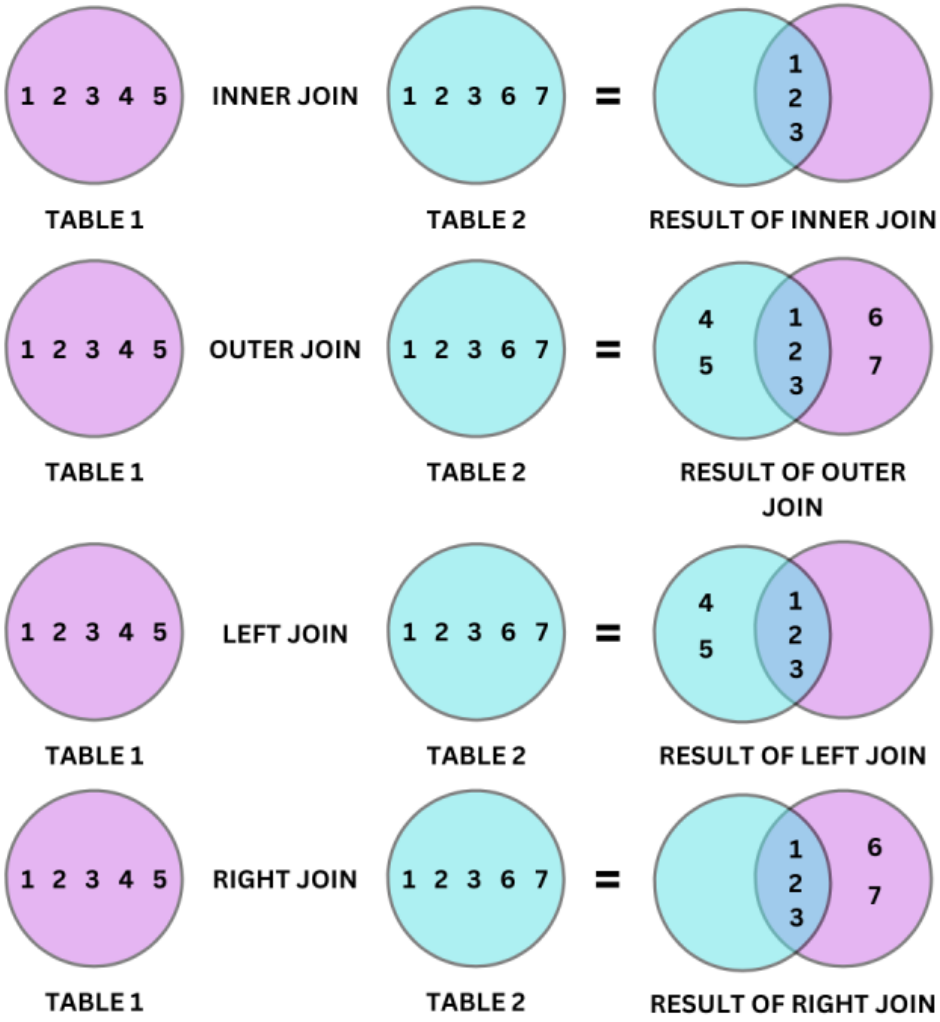
FULL OUTER JOIN



LEFT OUTER JOIN



RIGHT OUTER JOIN

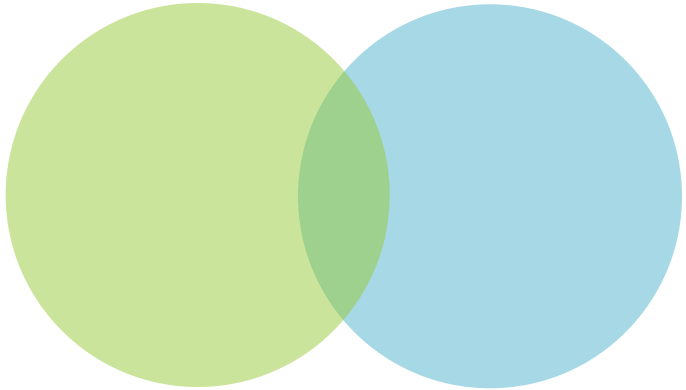

1 2 3 4 5
INNER JOIN
1 2 3 6 7
=
1
2
3
TABLE 1
TABLE 2
RESULT OF INNER JOIN
1 2 3 4 5
OUTER JOIN
1 2 3 6 7
=
4
5
1
2
3
6
7
TABLE 1
TABLE 2
RESULT OF OUTER JOIN
1 2 3 4 5
LEFT JOIN
1 2 3 6 7
=
4
5
1
2
3
TABLE 1
TABLE 2
RESULT OF LEFT JOIN
1 2 3 4 5
RIGHT JOIN
1 2 3 6 7
=
1
2
3
6
7
TABLE 1
TABLE 2
RESULT OF RIGHT JOIN

Example 2.21: Consider the two relations from our running movie database:

```
Movies(title, year, length, genre, studioName, producerC#)
MovieExec(name, address, cert#, netWorth)
```

QUESTION 1

- Which joins produce the same results for these tables?



```
Movies(title, year, length, genre, studioName, producerC#)
StarsIn(movieTitle, movieYear, starName)
MovieExec(name, address, cert#, netWorth)
```

Joins

ON StarsIn.starName = MovieExec.name

- **JOIN (INNER JOIN)**
  - **Only results where they match**
  - **Values from each table in each tuple**
- **OUTER JOIN**
  - **Every tuple from each table**
    - **If actor doesn't produce, no values in MovieExec cols**
    - **If producer doesn't act, no values in StarsIn cols**
- **LEFT OUTER JOIN**
  - **Every tuple from StarsIn table**
    - **If actor doesn't produce, no values in MovieExec cols**
    - **If producer doesn't act, no values in StarsIn cols**
- **RIGHT OUTER JOIN**
  - **Every tuple from MovieExec table**
    - **If producer doesn't act, no values in StarsIn cols**

Query Activity

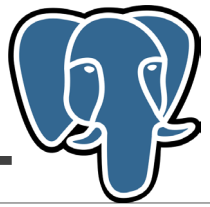
SQL Query Practice

Example 2.21: Consider the two relations from our running movie database:

```
Movies(title, year, length, genre, studioName, producerC#)
MovieExec(name, address, cert#, netWorth)
```

How do you get the title and year for the movie produced by the producer with the highest net worth?

PostgreSQL



1. `SELECT * FROM Movie JOIN MovieExec ON producer_num = cert_num;`
2. `SELECT * FROM Movie JOIN MovieExec ON producer_num = cert_num ORDER BY net_worth;`
3. `SELECT * FROM Movie JOIN MovieExec ON producer_num = cert_num ORDER BY net_worth DESC;`
4. `SELECT * FROM Movie JOIN MovieExec ON producer_num = cert_num ORDER BY net_worth DESC LIMIT 1;`
5. **`SELECT title, year FROM Movie JOIN MovieExec ON producer_num = cert_num ORDER BY net_worth DESC LIMIT 1;`**

Subqueries

1. can return a single constant to be compared with another value in a **`WHERE`** clause
2. can return relations that can be used in various ways in **`WHERE`** clauses
3. can appear in **`FROM`** clauses, followed by a tuple variable that represents the tuples in the result of the subquery

Subqueries

1. can return a single constant to be compared with another value in a **WHERE** clause

```
1)  SELECT name
2)  FROM MovieExec
3)  WHERE cert# =
4)      (SELECT producerC#
5)       FROM Movies
6)       WHERE title = 'Star Wars'
        );
```

```
SELECT name
FROM Movie JOIN MovieExec ON producer_num = cert_num
WHERE title = 'Star Wars';
```

Subqueries

2. can return relations that can be used in various ways in **`WHERE`** clauses

Finding the producers of Harrison Ford's movies

```
Movies(title, year, length, genre, studioName, producerC#)
StarsIn(movieTitle, movieYear, starName)
MovieExec(name, address, cert#, netWorth)
```

Subqueries

2. can return relations that can be used in various ways in **`WHERE`** clauses

```
1)  SELECT name
2)  FROM MovieExec
3)  WHERE cert# IN
4)      (SELECT producerC#
5)       FROM Movies
6)       WHERE (title, year) IN
7)           (SELECT movieTitle, movieYear
8)            FROM StarsIn
9)            WHERE starName = 'Harrison Ford'
             )
        );
```

Subqueries

2. can return relations that can be used in various ways in **`WHERE`** clauses

```
SELECT name
FROM MovieExec JOIN Movie ON cert_num = producer_num
JOIN StarsIn ON title = movie_title AND year =
movie_year
WHERE star_name = 'Harrison Ford';
```

Subqueries

3. can appear in **FROM** clauses, followed by a tuple variable that represents the tuples in the result of the subquery

```
1)  SELECT name
2)  FROM MovieExec, (SELECT producerC#
3)                   FROM Movies, StarsIn
4)                   WHERE title = movieTitle AND
5)                         year = movieYear AND
6)                         starName = 'Harrison Ford'
7)                  ) Prod
8)  WHERE cert# = Prod.producerC#;
```

Figure 6.11: Finding the producers of Ford's movies using a subquery in the FROM clause

Practice

Let's revisit our schema for a College Ranking System.

Next Class

Module:

Week 8: Midterm

Topic:

Midterm Review