Web Service Calls

Programming Assignment

TU Delft

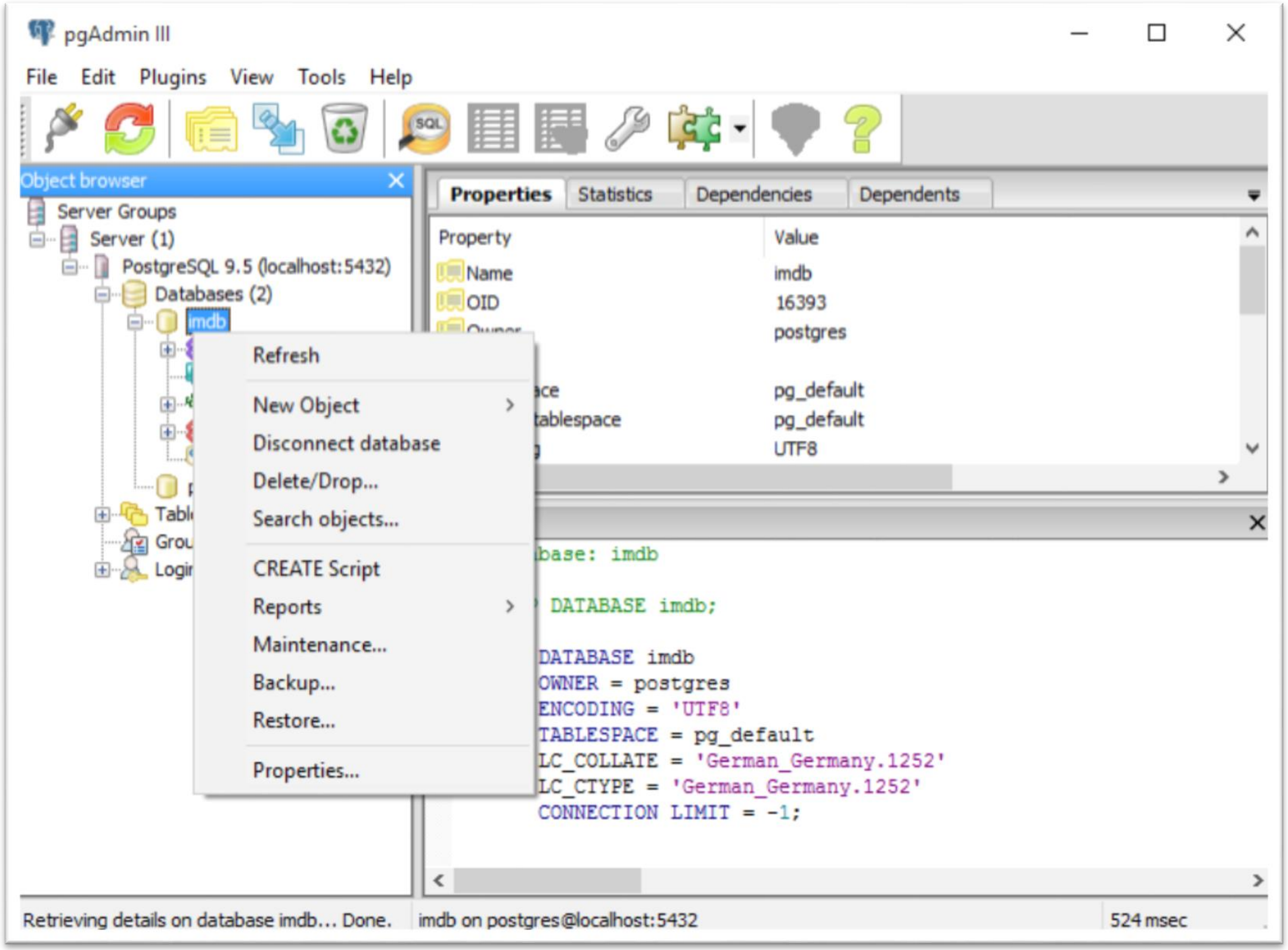
Development Project

- Core Idea: Develop a movie information web service
- Create multiple backends for the web service
  - Relational backend using PostgresSQL
  - Choose two from:
    - Key-value backend using either Redis community edition or Riak
    - Document-based using either MongoDB or CouchDB
    - Wide columnar-based using either Cassandra or Hbase
    - Graph-based using Neo4J or Openlink Virtuoso OS

Development Project

- Data is provided as a Postgres backup file
  - IMDb dump file process with modified imdb2sql
    - http://github.com/ameerkat/imdb-to-sql
  - Getting started
    - Obtain current version of PostgreSQL
    - Import Backup into new DB in pgAdmin
    - After importing, do maintenance ops like index rebuilding and vacuuming!
    - Explore data, toy around with it

Development Project: pgAdmin



Property	Value
Name	imdb
OID	16393
Owner	postgres
ace	pg_default
tablespace	pg_default
	UTF8

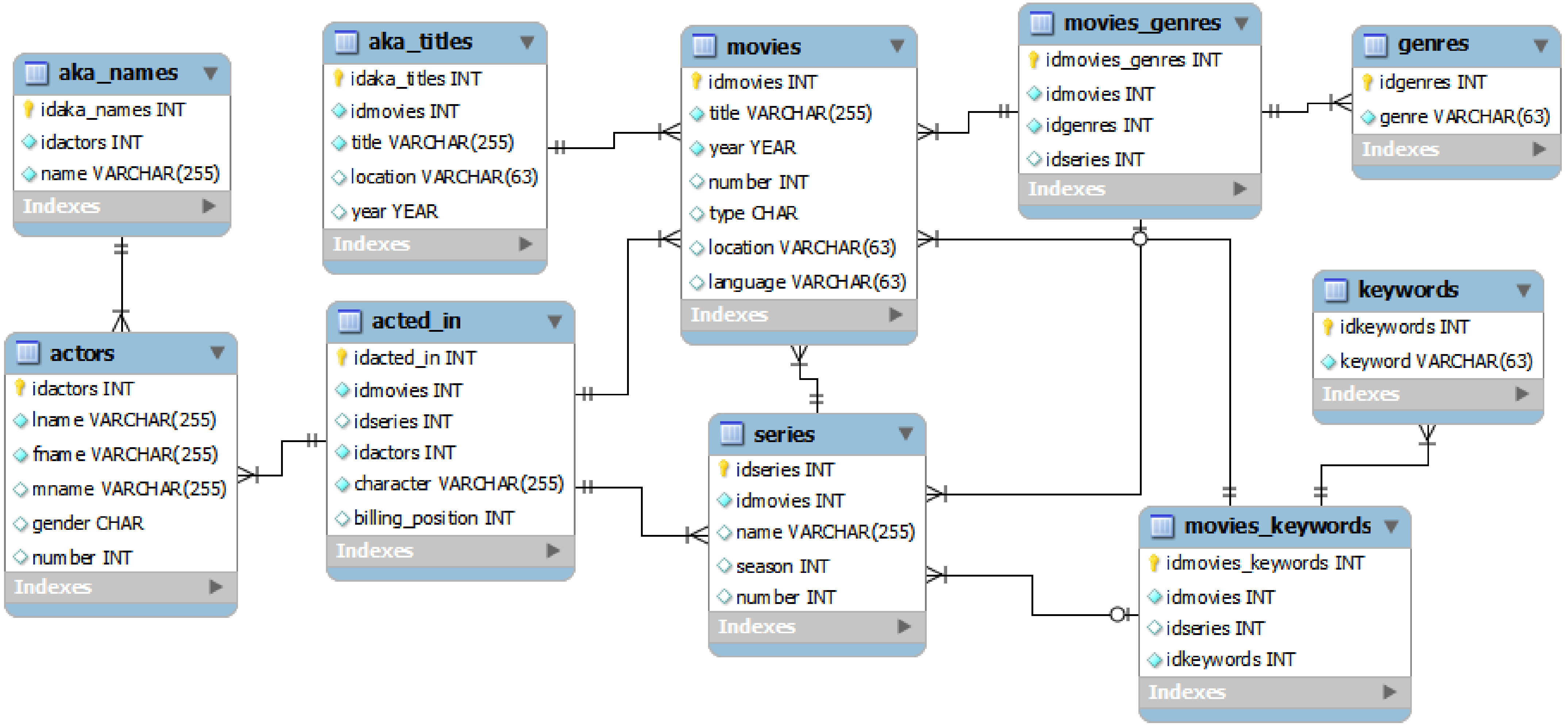
```
base: imdb

 DATABASE imdb;

DATABASE imdb
OWNER = postgres
ENCODING = 'UTF8'
TABLESPACE = pg_default
LC_COLLATE = 'German_Germany.1252'
LC_CTYPE = 'German_Germany.1252'
CONNECTION LIMIT = -1;
```

Retrieving details on database imdb... Done. imdb on postgres@localhost:5432 524 msec

Development Project: Schema



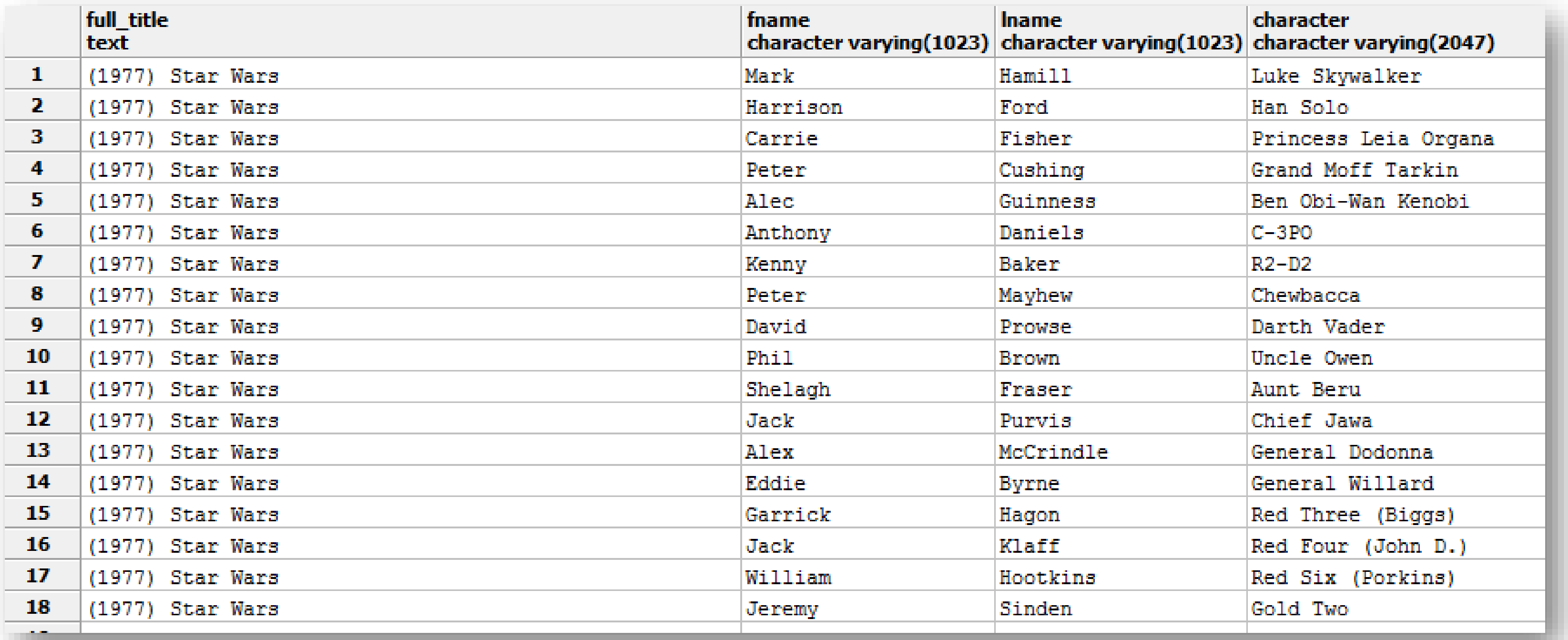
Development Project: Explore

- List the full cast of movies with “ Star Wars” in their title

```
SELECT '(' || m.year ||') ' || m.title AS full_title, a.fname, a.lname,
ai.character
 FROM movies m JOIN acted_in ai ON ai.idmovies=m.idmovies
        JOIN actors a ON a.idactors=ai.idactors
WHERE m.title LIKE '%Star Wars%' AND TYPE=3
ORDER BY full_title, ai.billing_position
```

Development Project: Explore

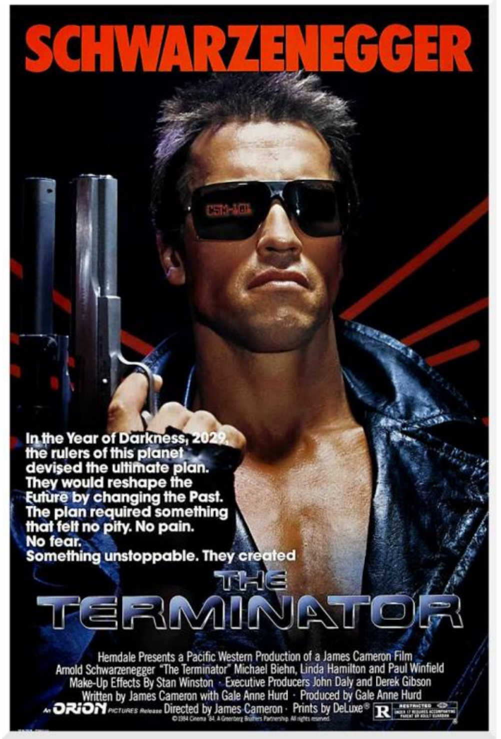
- List the full cast of movies with " Star Wars" in their title



	full_title text	fname character varying(1023)	lname character varying(1023)	character character varying(2047)
1	(1977) Star Wars	Mark	Hamill	Luke Skywalker
2	(1977) Star Wars	Harrison	Ford	Han Solo
3	(1977) Star Wars	Carrie	Fisher	Princess Leia Organa
4	(1977) Star Wars	Peter	Cushing	Grand Moff Tarkin
5	(1977) Star Wars	Alec	Guinness	Ben Obi-Wan Kenobi
6	(1977) Star Wars	Anthony	Daniels	C-3PO
7	(1977) Star Wars	Kenny	Baker	R2-D2
8	(1977) Star Wars	Peter	Mayhew	Chewbacca
9	(1977) Star Wars	David	Prowse	Darth Vader
10	(1977) Star Wars	Phil	Brown	Uncle Owen
11	(1977) Star Wars	Shelagh	Fraser	Aunt Beru
12	(1977) Star Wars	Jack	Purvis	Chief Jawa
13	(1977) Star Wars	Alex	McCrindle	General Dodonna
14	(1977) Star Wars	Eddie	Byrne	General Willard
15	(1977) Star Wars	Garrick	Hagon	Red Three (Biggs)
16	(1977) Star Wars	Jack	Klaff	Red Four (John D.)
17	(1977) Star Wars	William	Hootkins	Red Six (Porkins)
18	(1977) Star Wars	Jeremy	Sinden	Gold Two

Development Project: Explore

- Number of Named Characters for movies named something with “Terminator”

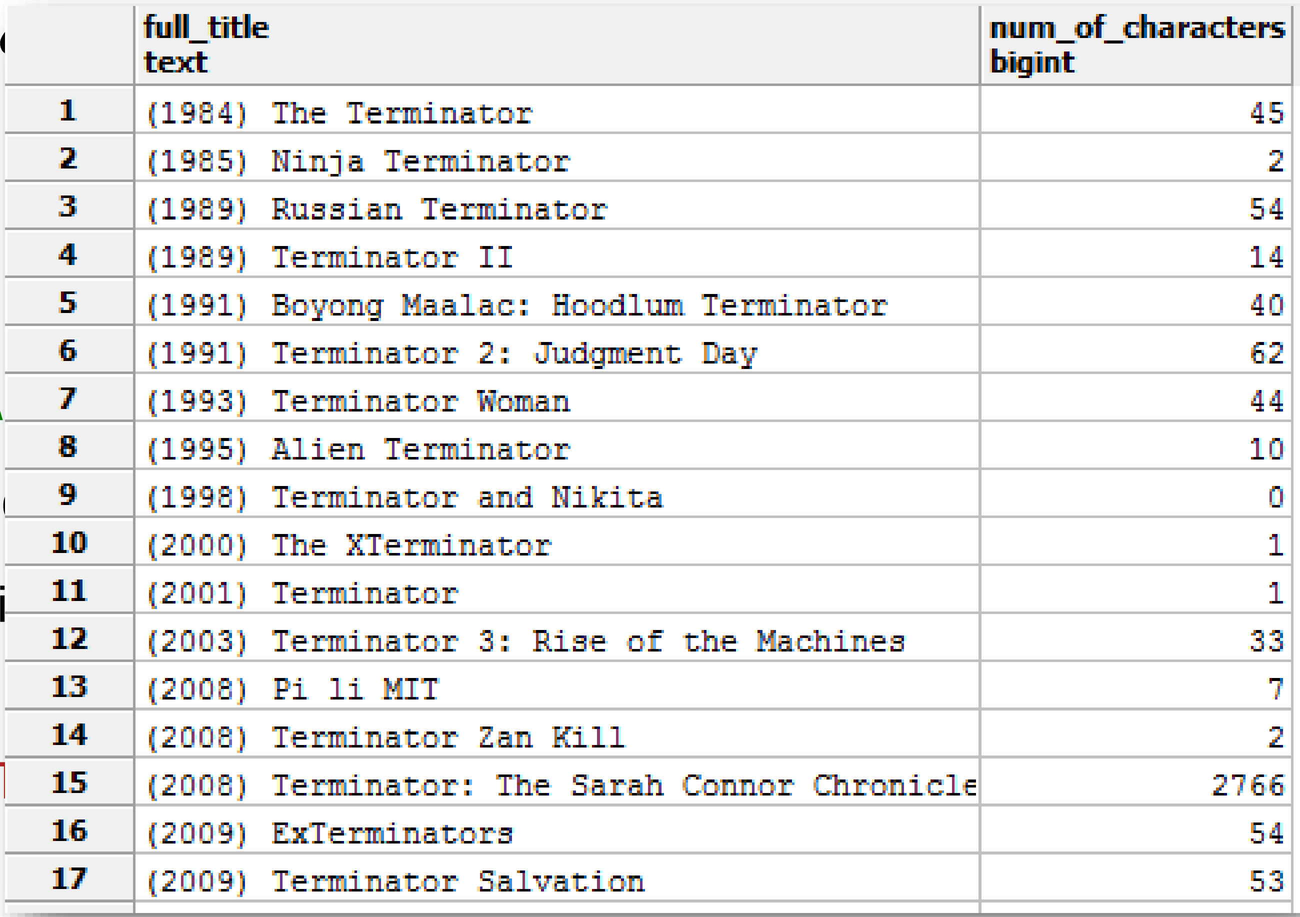

SCHWARZENEGGER
In the Year of Darkness, 2029, the rulers of this planet devised the ultimate plan. They would reshape the Future by changing the Past. The plan required something that felt no pity. No pain. No fear. Something unstoppable. They created
THE TERMINATOR

```
SELECT '(' || m.year ||') ' || m.title AS full_title,
    count(ai.character) AS num_of_character
FROM movies m JOIN acted_in ai ON ai.idmovies=m.idmovies
    JOIN actors a ON a.idactors=ai.idactors
WHERE m.title LIKE '%Terminator%'
    AND TYPE=3 GROUP BY m.year, m.title ORDER BY m.year, m.title
```

Development Project: Explore

- Number of Named Characters for movies named something with “

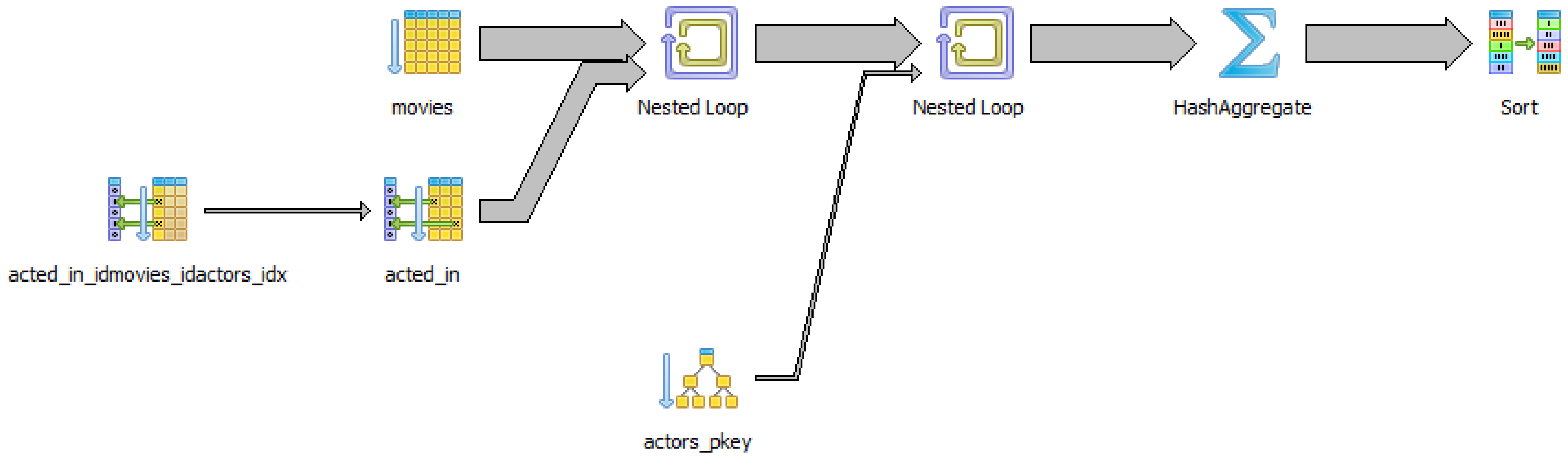
```
SELECT '(' || m.year |
    count(ai.character) A
FROM movies m JOIN acte
    JOIN actors a ON a.i
    LEFT JOIN aka_titles
WHERE (aka.title LIKE '%T
    AND TYPE=3 GROUP
```



	full_title text	num_of_characters bigint
1	(1984) The Terminator	45
2	(1985) Ninja Terminator	2
3	(1989) Russian Terminator	54
4	(1989) Terminator II	14
5	(1991) Boyong Maalac: Hoodlum Terminator	40
6	(1991) Terminator 2: Judgment Day	62
7	(1993) Terminator Woman	44
8	(1995) Alien Terminator	10
9	(1998) Terminator and Nikita	0
10	(2000) The XTerminator	1
11	(2001) Terminator	1
12	(2003) Terminator 3: Rise of the Machines	33
13	(2008) Pi li MIT	7
14	(2008) Terminator Zan Kill	2
15	(2008) Terminator: The Sarah Connor Chronicle	2766
16	(2009) ExTerminators	54
17	(2009) Terminator Salvation	53

Development Project: Explore

- Enjoy the magic of SQL!



Development Project: What to do next?

- Decide on you favorite programming language
- Look into how to implement RESTful Web Services in the language of your choice
  - Web service calls via **JSON**
  - Define web service endpoints for the required functionality
  - Connect your service to the backend data storage

Development Project: What to do next?

- Decide on two additional back ends
  - Convert your data to the respective formats, import into new database
    - This will be tricky!
    - Connect your web service to the new backend
  - Why are we doing this?
    - See how different technology behaves in the same usage scenario!
  - Must be running at least on 3 machines at the same time!
    - Use your own laptops, or consider using virtual machines

Coding Assignment: Service Interfaces

- Try to implement the following services interfaces
  - Should be trivial using Postgres
  - Can get quite hard with other databases
    - (to be fair: the assignment is skewed because we started with a relational schema, and the service interfaces contain a lot of queries where relation DBs are good at)
    - If you **cannot implement** something, or can only **implement** it **partially**, EXPLAIN WHY!

Coding Assignment: Service Interfaces

- SC1: Detailed movie information
  - Get info about movies based on their id or titles
  - **Input:** One of the following
    - movieId
    - title
      - If possible, also allow for partial title matches ("Star Wars" also finds "Star Wars – Return of Jedi"
      - For partial matches, try to restrict to movies only
        - » Movies have a type, and IMDB stores all kind of different things in the movie table like games, advertisements, TV spots, etc.
        - » This is reflected by the "type" attribute. I think the correct type for movies was 3.
        - » Keep this in mind also for the other services interfaces
      - Optional: filter by year
  - **Output:** List of <Full Movie Info>

Coding Assignment: Service Interfaces

- SC2: Detailed actor information
  - Get info on actors and their movies based on either actorID or actor names
  - **Input:** One of the following
    - actorID
    - firstname and/or lastname
  - **Output:**
    - List of actor <actor>
      - Also list all movies by name and year
        - » If possible, order by year

Coding Assignment: Service Interfaces

- SC3: Short actor statistics
  - Get short info on actors and their movie statistics
  - **Input:** One of the following
    - actorID
    - firstname and/or lastname
  - **Output:**
    - Full name of the actor(s), number of movies she played in

Coding Assignment: Service Interfaces

- SC4: Genre exploration
  - Given a genre and a year, return all movies with that genre and year
    - Optional: Provide end year, e.g., comedies from 2014-2016
  - **Input:**
    - genre label
    - year
    - Optional: end year
  - **Output:**
    - List of actor <actor>
      - List of all movies
        - » If possible, order by year and then by name

Coding Assignment: Service Interfaces

- SC5: Genre statistics
  - For a certain year (year range), provide statistics on the number of movies in each genre
  - **Input:**
    - Year
    - Optional: end year
  - **Output:**
    - List of genre labels, number of movies for each genre for that year range

Coding Assignment: Service Interfaces

- Types
  - <movie info>
    - Id, Title, Year of movie
  - <full movie info>
    - <movie info>
    - Name of the series (if any)
    - All genre labels
    - All keywords
    - All <actor> and their respective role (if known)
      - If possible ordered by billing position
  - <actor>
    - First name, last name, gender

Code Submission

- Package your **code** with a **description**, and send it to me via slack before the 18th
  - Or send me a link to a repository / cloud storage
- The **description** should outline the **data model** you used for each data store
  - e.g., how did you import/convert the Postgres data into the data store of your choice? How did you change, aggregate, replicate, or transform the data for your chosen data stores?
- The **description** should also briefly describe how you implemented each of the service interfaces from a conceptual point of view
  - e.g., briefly describe the involved queries / map-reduce-statements / manual data manipulations you used for each data store
    - I am not interested in code here, I am interested in how you interact with the data
  - If you could not fully implement a service interface, discuss why that is
  - **Especially focus on how you think your system would behave when scaled to more nodes and many users**
- The **description** should also discuss your **experience** with the chosen data store
  - What worked well? What did not? Is your chosen data store suitable for the task? If you had to do the assignment again, what would you do differently? etc.

Presentation

- You already got a time-slot for your final presentation
  - Also send me the slides together with the other delibverables
  - 20 minutes plus discussions afterward
  - Do a very short “proof” that your code is working
    - Max 5 minutes
    - Discuss what is missing and why
  - See the announcement on blackboard for more info. Focus on the things already mentioned on the previous slides
    - **Data Models and Queries in your chosen data store**
      - **This should be the main part of your presentation**
      - How did you transform, change, aggregate data?
      - How did you design your queries to fulfill the requirements?
    - How did you deal with the limitations of your data store?
    - Which advantages did you data store provide?
    - What are your lessons learned?

Summary

- Three deliverables for programming assignment
  - Code
    - July 18th
  - Description
    - July 18th
  - Presentation
    - As agreed upon

TU Delft