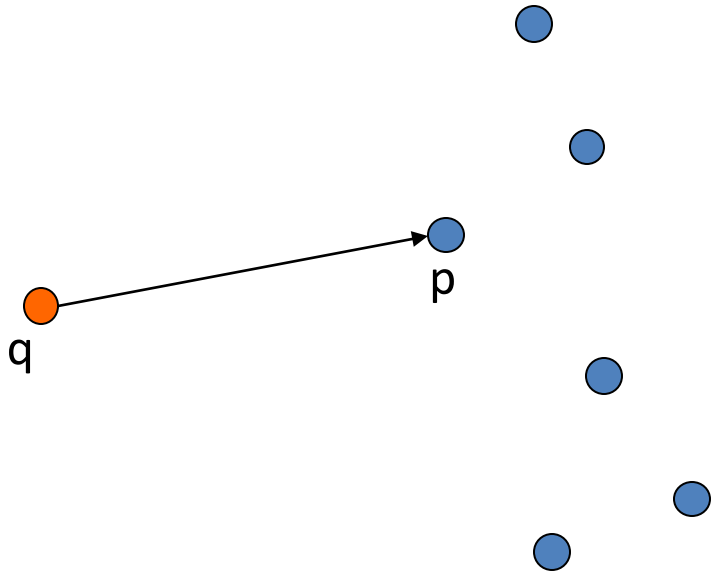
Multidimensional queries

Nearest Neighbor (NN) Search

Data is often modeled as a set of n points in a multidimensional space (R^d)

Given a query point q, a nearest neighbor query will find the nearest neighbor p of q, i.e. the closest point given a distance metric

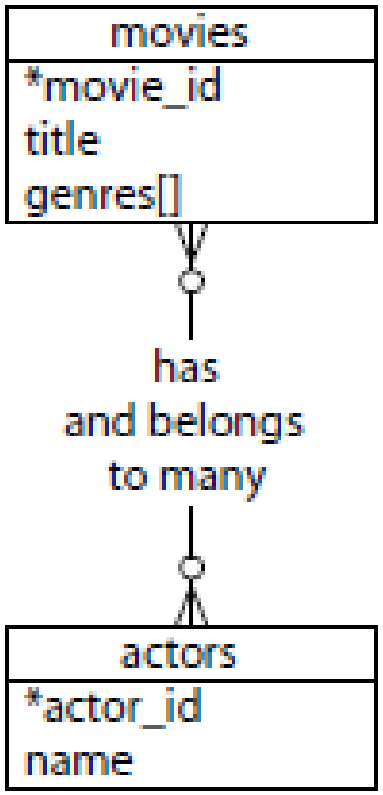


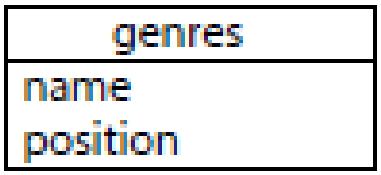
Nearest neighbor applications

(Longer list available in Wikipedia)

- Pattern recognition
- Statistical classification
- Computer vision
- Content-based image retrieval
- Recommendation systems
- Internet marketing
- Spell checking
- Plagiarism detection
- Similarity scores
- Cluster analysis
- Chemical similarity

Our movie system



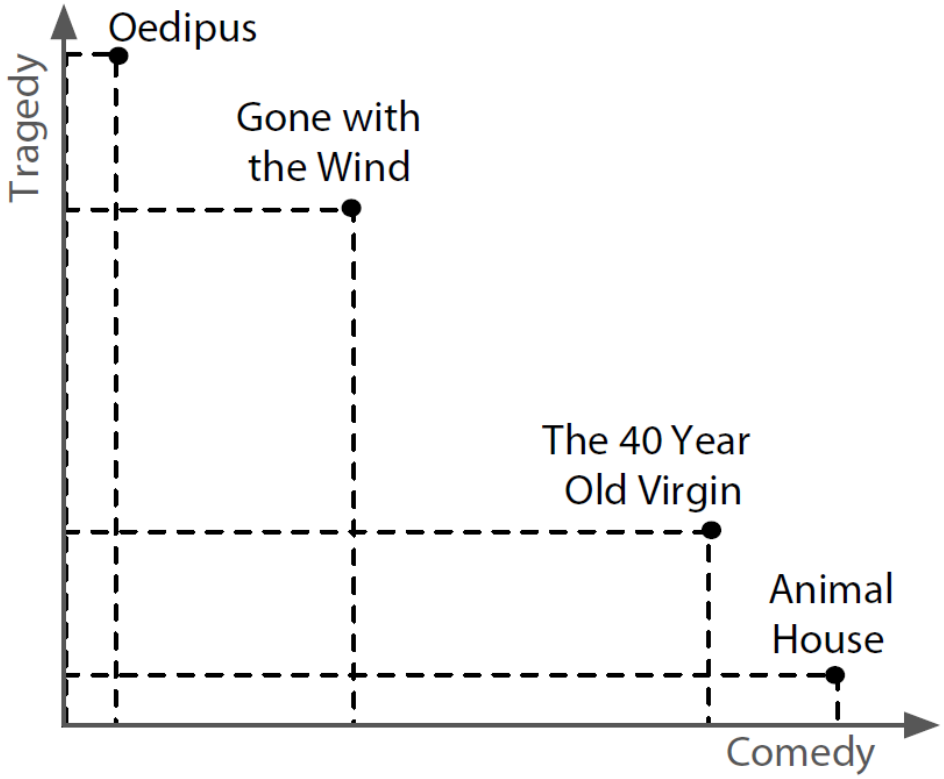


- When we created the movies table, we defined genres as a `cube` datatype, i.e. a multidimensional vector
- **Cube** is a contributed package

- Each value for genres in movies is a point in 18-dimensional space with each dimension representing a genre
- Why does it make sense? Movies can span multiple genres, many movies are not 100% comedy or 100% tragedy – they are in between

Movies recommendation

- We'll find similar movies by finding NN
- Figure shows four movies in a 2-dimensional space
- If your favorite movie is *Animal House* , you'd probably would like to see *The 40-year-old virgin* more than *Oedipus*
- We'll use 18 genres, but the principle will be the same



Genres in our movie system

- Each genre is scored from 0-10 (totally arbitrary)
  - 0 means nonexistent
  - 10 being the strongest

- What is the genre vector of *Star wars*?
  - `SELECT title, genre FROM movies WHERE lower(title)='star wars'`
  - (0,7,0,0,0,0,0,0,0,7,0,0,0,0,10,0,0,0)

- Each position maps to an entry in the genres table:
  - `SELECT * FROM genres;`

The cube package

- https://www.postgresql.org/docs/10/cube.html

- We can decrypt the genre values by using `cube_ur_coord`

cube_ur_coord(cube, integer)	Returns the ***n***-th coordinate value for the upper right corner of the cube.

- SELECT name,
  cube_ur_coord('(0,7,0,0,0,0,0,0,0,7,0,0,0,0,10,0,0,0)', position)
  as score FROM genres g
  WHERE cube_ur_coord('(0,7,0,0,0,0,0,0,0,7,0,0,0,0,10,0,0,0)',
  position) > 0;

Select movies with a particular genre

- *SELECT title, genre FROM movies WHERE cube_ur_coord(genre, 15)>5*

- *SELECT title, genre FROM movies WHERE cube_ur_coord(genre, (select position from genres where name='Comedy'))>5;*

name	position
Action	1
Adventure	2
Animation	3
Comedy	4
Crime	5
Disaster	6
Documentary	7
Drama	8
Eastern	9
Fantasy	10
History	11
Horror	12
Musical	13
Romance	14
SciFi	15
Sport	16
Thriller	17
Western	18

Nearest neighbor search

- We'll use function
  `cube_distance(point1, point2)`

- *Find the distance of all movies to the Star Wars genre vector, nearest first:*

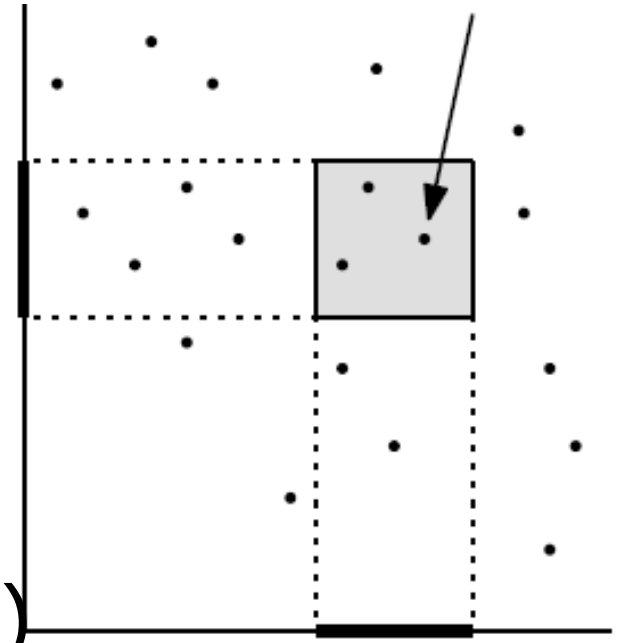
```
SELECT *,
cube_distance(genre,
'(0,7,0,0,0,0,0,0,0,7,0,0,0,0,10,0,0,0)')
dist
FROM movies
ORDER BY dist;
```

Nearest neighbor performance

- What is the cost of a NN query?
- What about the movies_genres_cube cube index we created last lecture?

- We can rewrite the NN query as a range query:
- *Near neighbor: find one/all points within distance r from q*
- The index can be used to reduce the number of points we need to look at
- Any downside?

Near neighbor query



- *Define a bounding cube containing the query*
- *All points inside the cube are potential NN*
- Use `cube_enlarge(cube,r,n)` to increase the size of the cube by the specified radius ***r*** in at least ***n*** dimensions.
- Create a two-dimensional square around point (1,1) of one unit: the lower-left point would be at (0,0), and the upper-right point at (2,2):

```
SELECT cube_enlarge('(1,1)',1,2);
```

Recommending movies

Find all movies within a distance of 5-unit cube from the *Star Wars* genre point

Contains operator @>: a @> b returns whether cube a contains cube b

```
SELECT title,
cube_distance(genre,
'(0,7,0,0,0,0,0,0,0,7,0,0,0,0,10,0,0,0)') dist FROM
movies WHERE
cube_enlarge('(0,7,0,0,0,0,0,0,0,7,0,0,0,0,10,0,0,0
)'::cube, 5, 18)
@> genre ORDER BY dist;
```

Recommending movies

Using a subselect, we can get the genre by movie name and perform our calculations against that genre using a table alias.

```
SELECT m.movie_id, m.title
FROM movies m,
(SELECT genre, title FROM movies WHERE title =
'Mad Max') s
WHERE cube_enlarge(s.genre, 5, 18) @> m.genre
AND s.title <> m.title
ORDER BY cube_distance(m.genre, s.genre)
LIMIT 10;
```

For today…

Create a stored procedure that enables you to input a movie title or an actor's name and then receive the top five suggestions based on either movies the actor has starred in or films with similar genres.

Include two calls to your function.