Exercise 6:

The query plan selected for the query:

select d.fname, d.lname

from Actor a, Casts c, Movie\_Director m, Director d

where a.id=c.pid and c.mid=m.mid and m.did=d.id

and a.fname='John' and a.lname='Spicer';

Looks as such:

π(d.fname,d.lname),card:1

|

⨝(m.did=d.id),card:1

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

| |

⨝(c.mid=m.mid),card:2791 |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

| | |

⨝(a.id=c.pid),card:29729 | |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | |

| | | |

σ(a.lname=Spicer),card:379 | | |

| | | |

σ(a.fname=John),card:3019 | | |

| | scan(Movie\_Director m) |

scan(Actor a) scan(Casts c) scan(Director d)

Selecting based on name results in a large reduction factor. Joining with Casts is very efficient, because a.id is a primary key for Actor. Joining next with Movie\_Director allows the final join to be with Director, which utilizes its primary key on d.id.

Our result may be different for different size data sets. If a larger data means that the different tables scale unequally (i.e. Actor grows disproportionately to Casts), then this would result in a different decision made for join order.

Our other query is as follows:

select a.fname, a.lname from Actor a, Casts c, Movie m where a.id=c.pid and c.mid=m.id and m.name='A Darker Shade of Gray';

This returns 4 rows:

a.fname a.lname

-----------------------------

Jack Arndt

Nicodemus Hammil

Kenneth Wonderley

Kristin Barker

The query plan generated looks like this:

π(a.fname,a.lname),card:1

|

⨝(c.pid=a.id),card:1

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

| |

⨝(m.id=c.mid),card:29729 |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

| | |

σ(m.name=A Darker Shade of Gray),card:165 | |

| | scan(Actor a)

scan(Movie m) scan(Casts c)

This plan is generated because it first selects out the movie 'A Darker Shade of Gray.' This reduces the input size to the first join by a great deal. Then it joins with Casts, which will result in a small output cardinality, since there is only one cast per film (and there is probably only one film with that title). Then the final join will be very efficient because a.id is a primary key for Actor.