CS698F Advanced Data Management

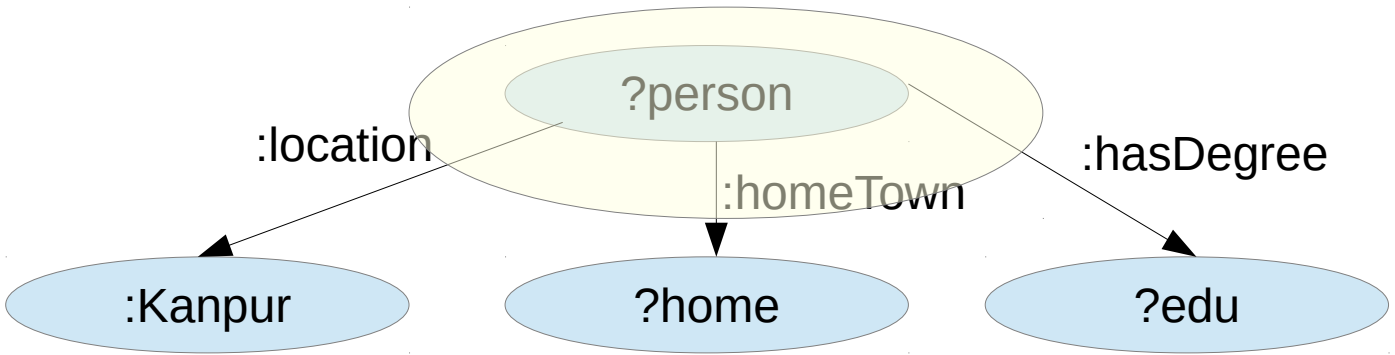
Instructor: Medha Atre

Options to distribute

- Consider graph as a list of edges ⟨S,P,O⟩
- Various options of distribution
  - S as the key, P as the key, O as the key
  - ⟨S,O⟩ as the key, ⟨S,P⟩ as the key… so on
  - What does this remind you of?
- Distributed join method changes slightly for each type of distribution.

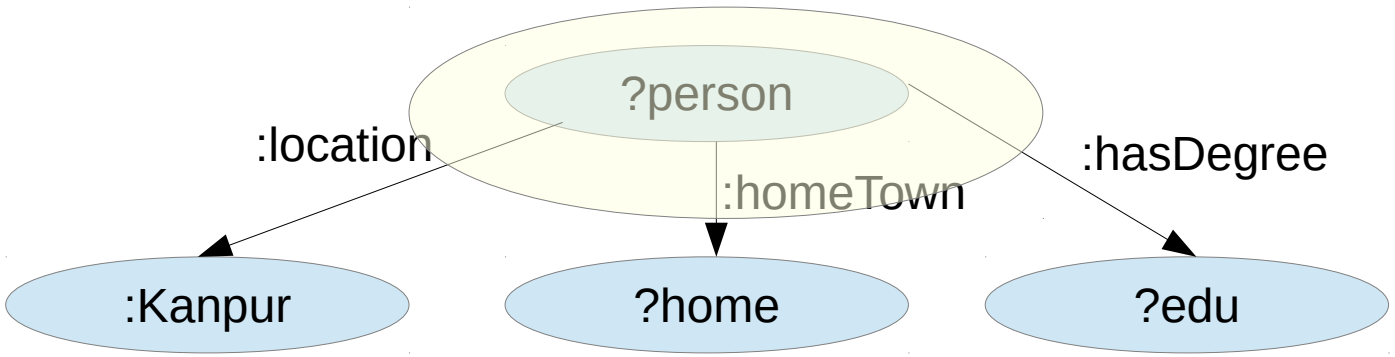
Options to distribute

- S as the key
  - All patterns that have join on S alone can be done without any "shipping" of tuples/edges.



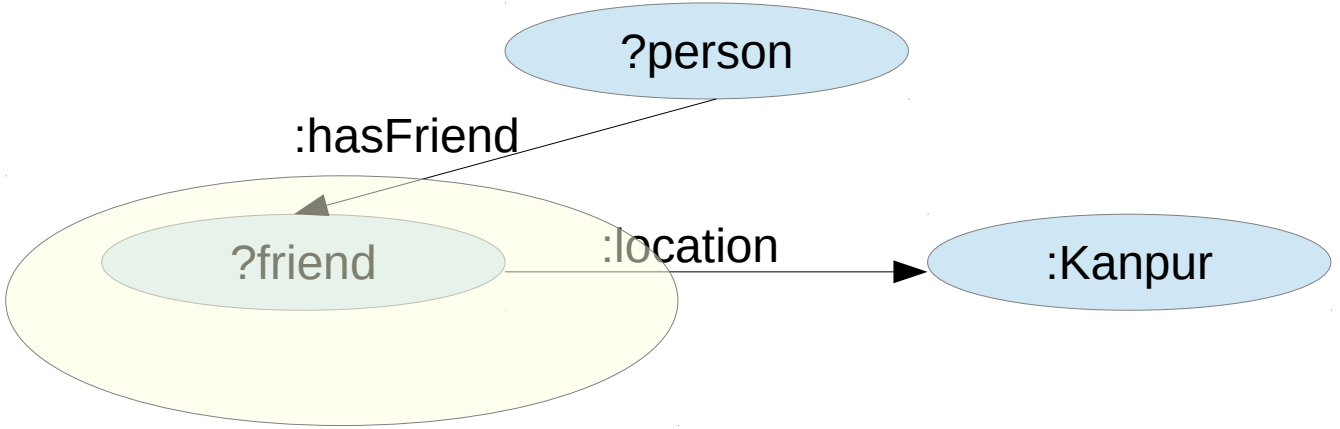
Options to distribute

Explanation shown on the board



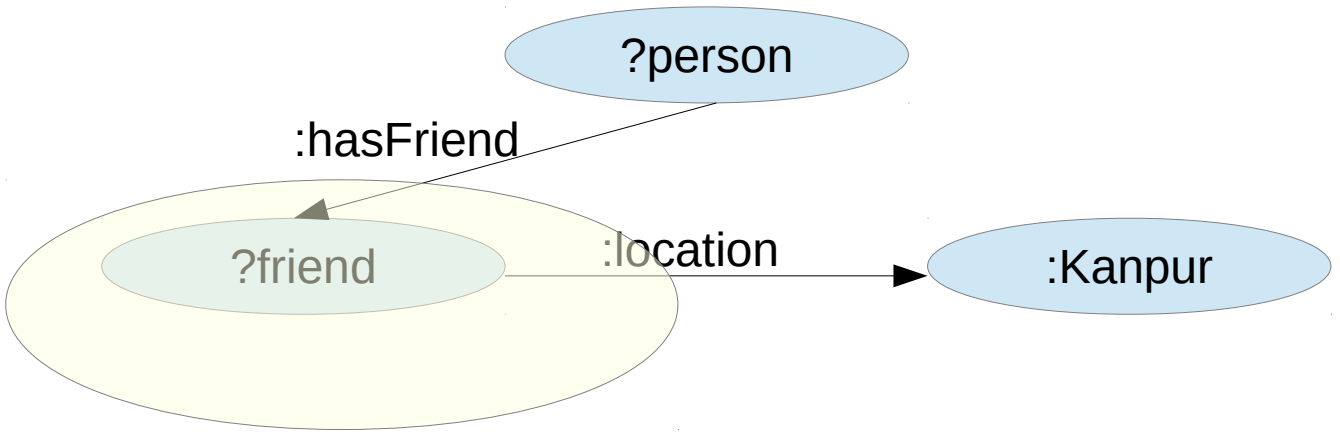
Options to distribute

- S as the key
  - Any combination of S-O join or a join on P requires "shipping".



Options to distribute

Explanation shown on the board



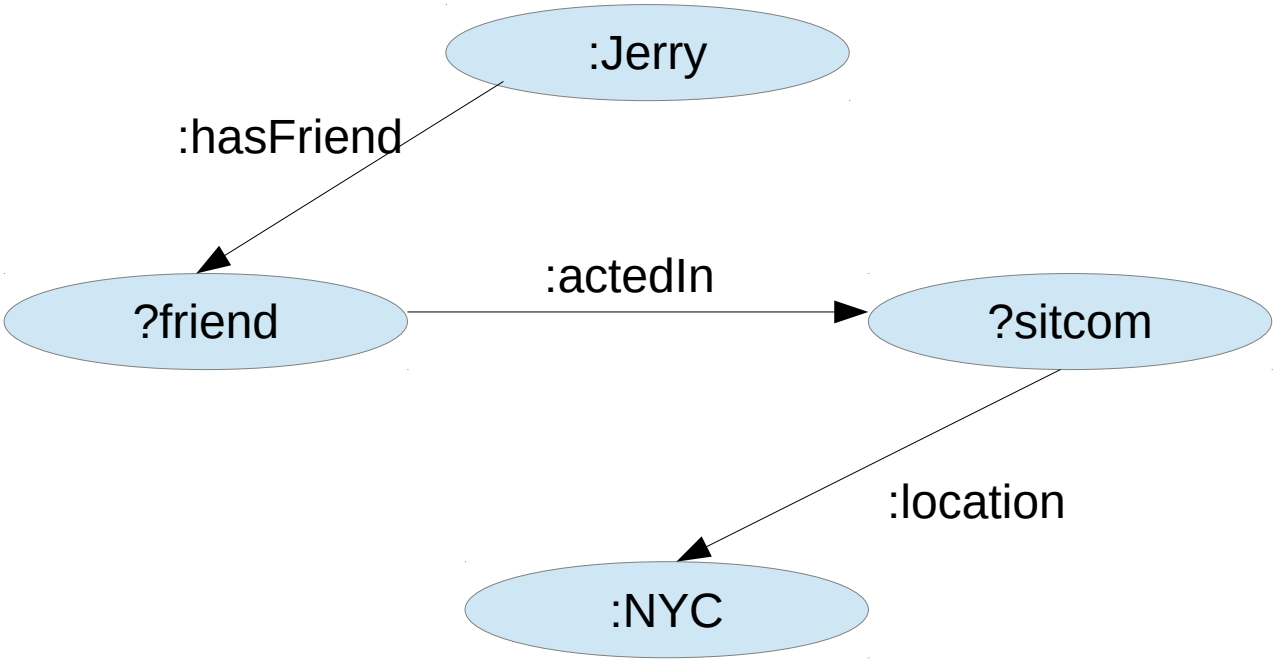
Options to distribute

- Then how to decide distribution "key"?
  - S, O, P?
  - Or ⟨S,O⟩, ⟨O,P⟩, ⟨S,P⟩?
- What do you join on always?
  - Vertices of the graph
- Do all joins happen only on S, or only on O, or S-O?
  - Most joins happen on S, O, and S-O too.

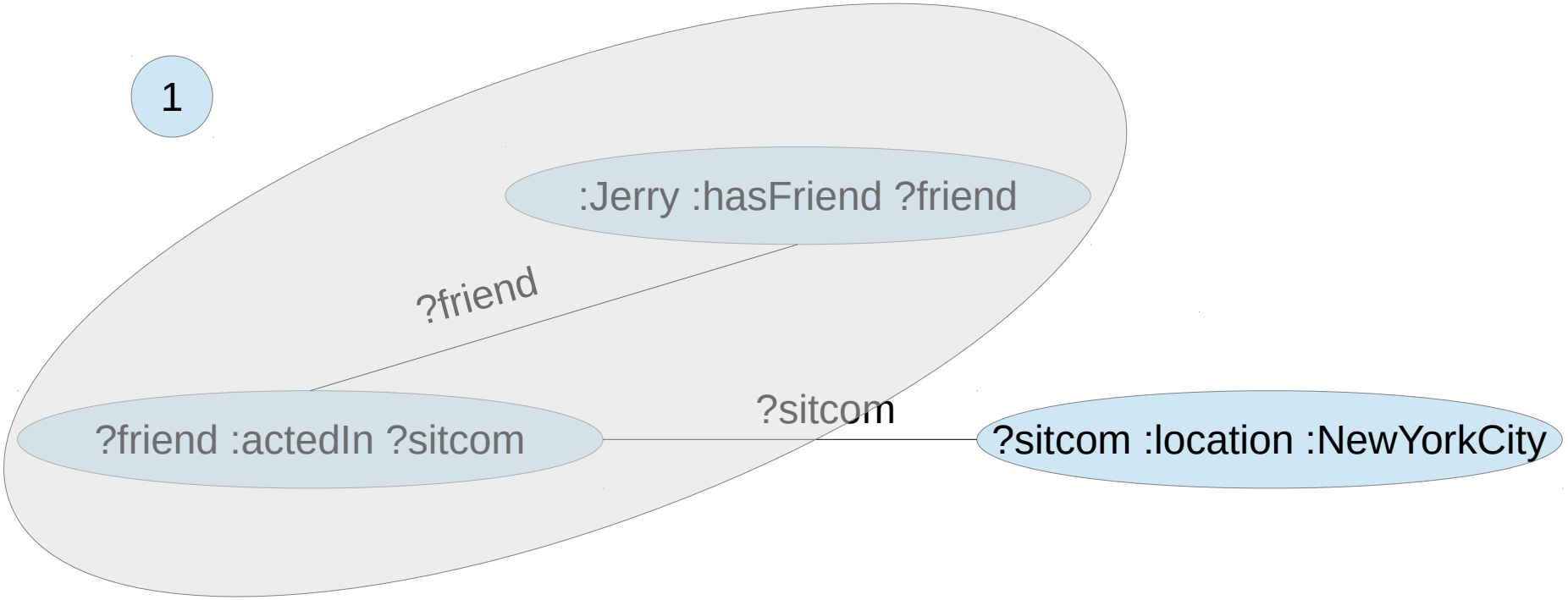
Options to distribute

- A join on S means you need "outgoing" edges of vertices.
- A join on O means you need "incoming" edges of vertices.
- A join on S-O means you need "incoming" AND "outgoing" edges of vertices.
- Then how to distribute?
  - For every edge of the graph, distribute ⟨hash(S), out-edge-list⟩, and ⟨hash(O), inc-edge-list⟩
- This ensures that first level join in any pattern can always be performed locally.

Back to our example



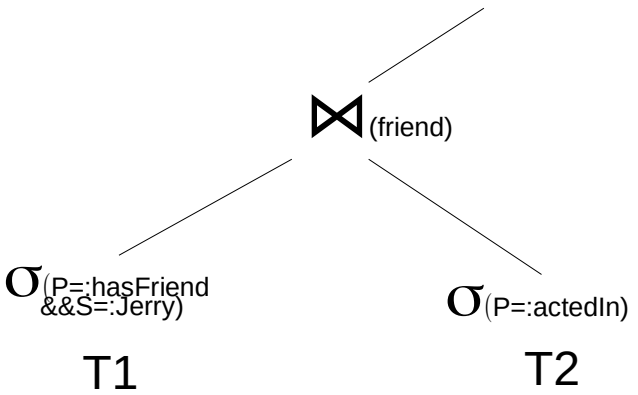
Join plans (std natural join)



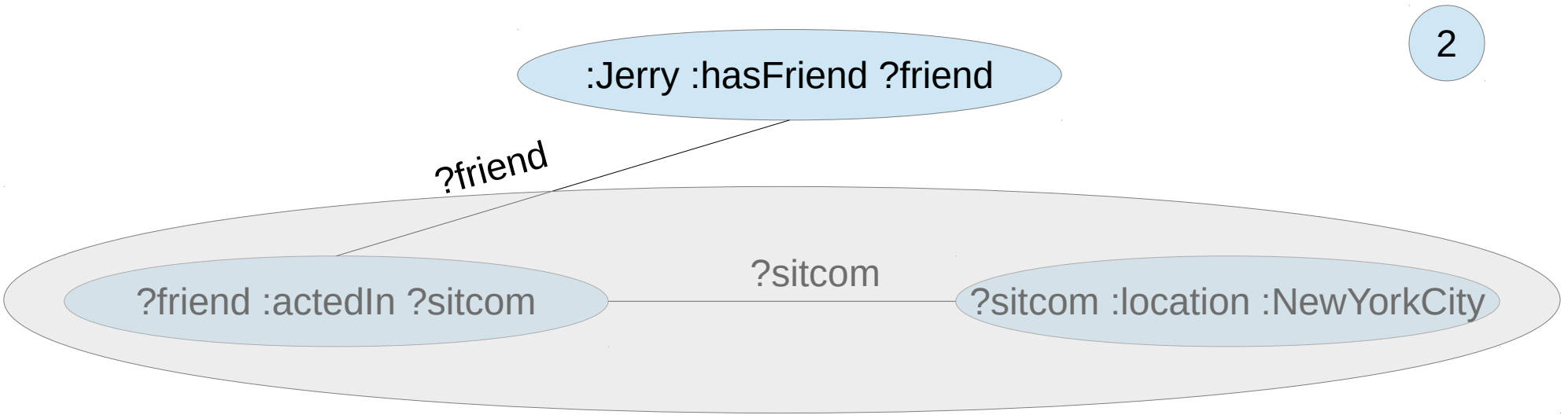
Join plans

- Can we perform this join without "shipping" of tuples/edges?
  - If we have used only ⟨*hash(S), out-edges*⟩ as the distribution policy?
  - If we have used only ⟨*hash(O), inc-edges*⟩ as the distribution policy?
  - If we have used **both** of the above?

Alternate way (bushy)



Join plans (std natural join)



Join plans (std natural join)

- AFTER the first join, we have generated following partial results.
  - ⟨*:Jerry, :hasFriend, var-bind(?friend), :actedIn var-bind(?sitcom)*⟩
  - We need to join this partial result with the bindings for *(?sitcom :location :NYC)*.
  - Which join is this?
    - S-S, O-O, or S-O?
  - Can we perform this join without "shipping"?
    - If yes, how?
    - If not, why not?

Join plans (std natural join)

- At the first join
  - What was the join variable – *?friend*
  - So all the graph edges will be distributed according to the variable bindings (values) of *?friend*
  - All the same bindings/values of *?friend* will be on the same compute-node.
  - That will not help while joining on *?sitcom*
    - Why not?

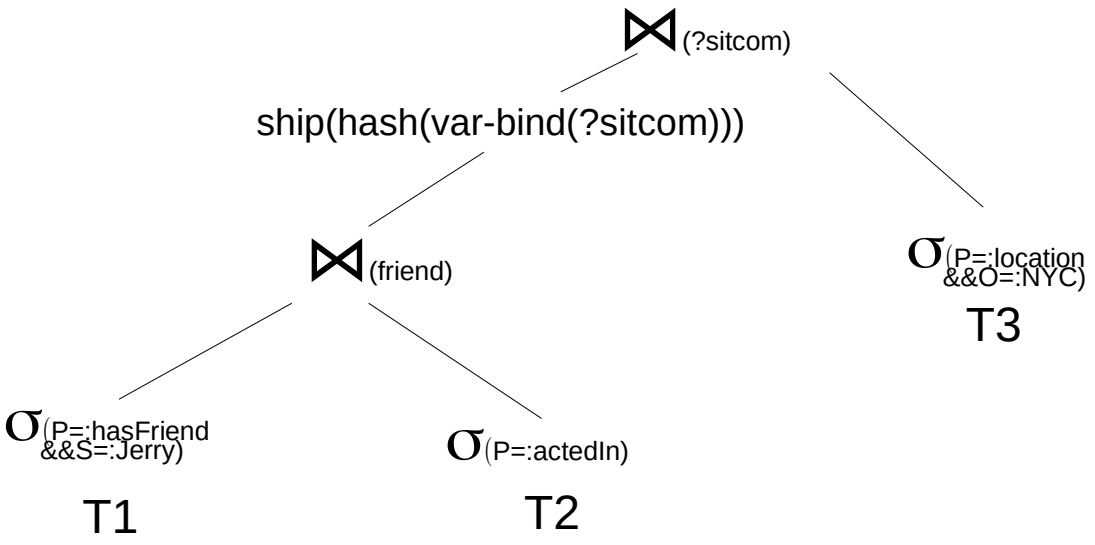
Join plans (std natural join)

- The values/bindings of *?sitcom* in ⟨*:Jerry, :hasFriend, var-bind(?friend), :actedIn var-bind(?sitcom)*⟩ are located on the compute-node according to *hash(var-bind(?friend))*
- For a join on *?sitcom*, we need to have graph edges located on compute node according to *hash(var-bind(?sitcom))*
  - This needs to be taken care of by shipping!
  - How do you decide what to ship, and where to ship?

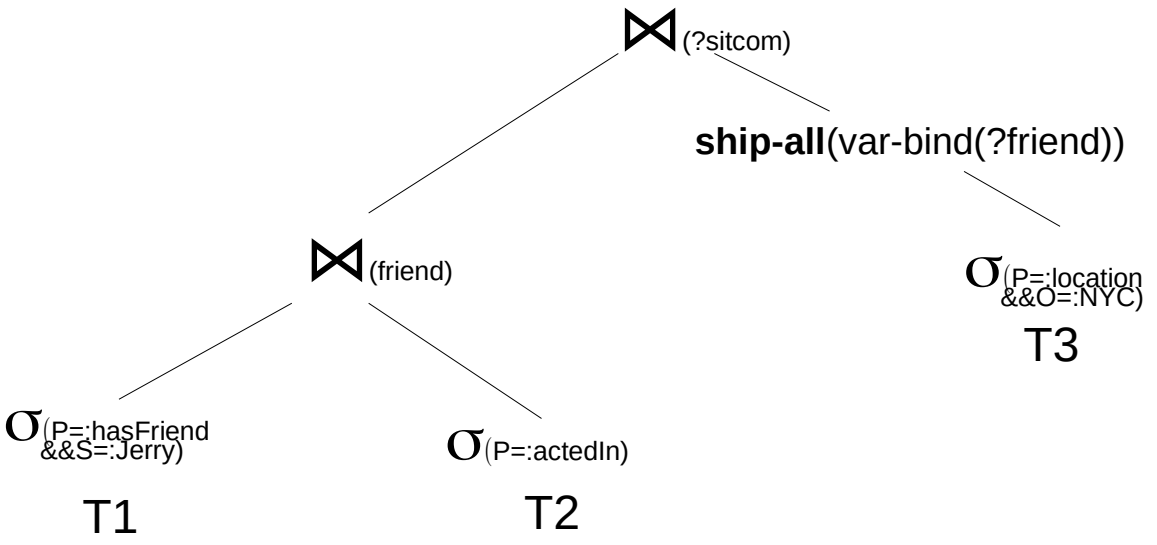
Options to ship

- Two options:
  - Ship ⟨*:Jerry, :hasFriend, var-bind(?friend), :actedIn var-bind(?sitcom)*⟩ by *hash(var-bind(?sitcom))* from each result.
  - Ship each graph edge matching *(?sitcom :location :NYC)* to **ALL** the compute-nodes?
    - Why? *Why not hash(var-bind(?sitcom))* from each edge matching *(?sitcom :location :NYC)*?
- Which of these would be cheaper?
  - Any intuition?

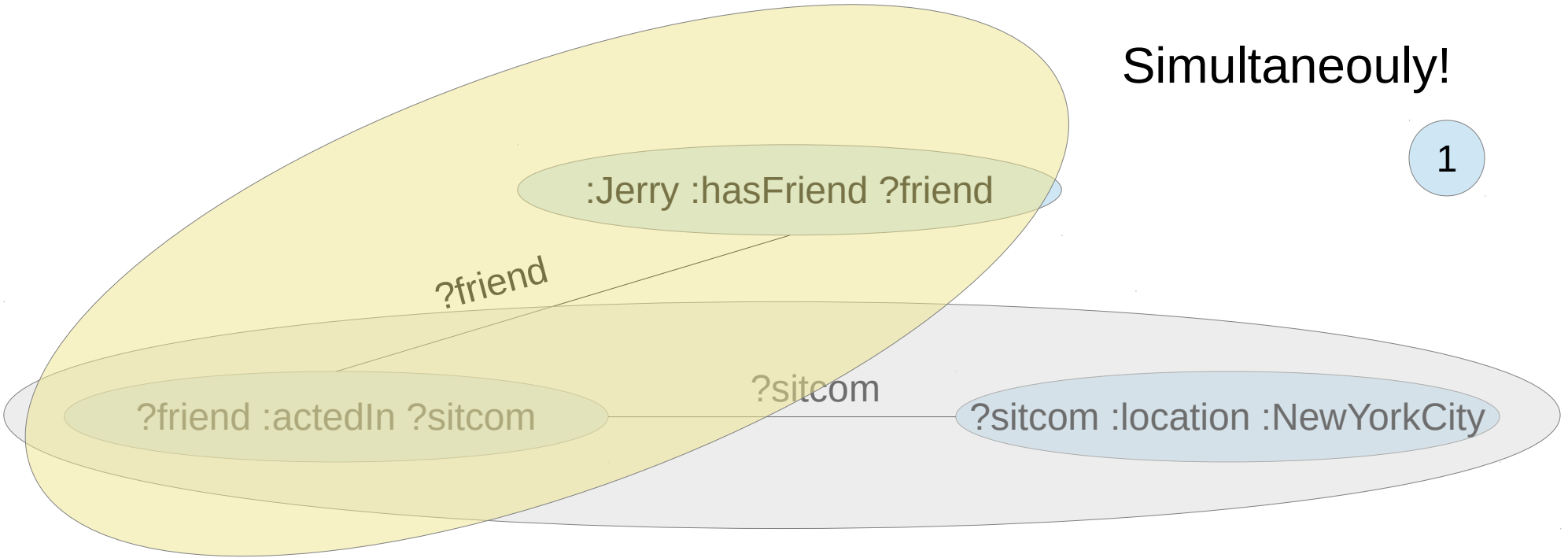
Alternate way (bushy)



Alternate way (bushy)



Alternate way (bushy)



Alternate way (bushy)

- When can we do these joins simultaneously?
  - If we distribute using only S as the key?
  - If we distribute using only O as the key?
  - If we distribute using both?
- How do we combine the independent join results?

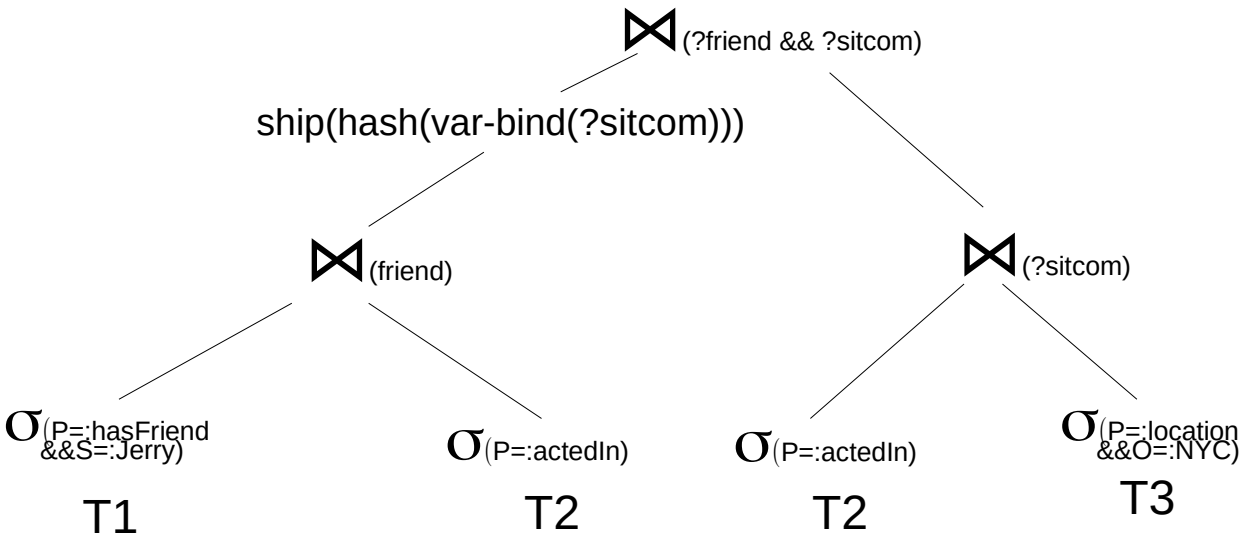
Alternate way (bushy)

- After the first two simultaneous joins partial results of
  - *<:Jerry, :hasFriend, var-bind(?friend), :actedIn, var-bind(?sitcom)>*
    - Each result located as per *hash(var-bind(?friend))*
  - *<var-bind(?friend), :actedIn, var-bind(?sitcom), :location, :NYC>*
    - Each result located as per *hash(var-bind(?sitcom))*
- Now what to ship where?
  - Hint: what would you join these patial results on?

Two options to ship

- Two options:
  - Ship <*:Jerry, :hasFriend, var-bind(?friend), :actedIn var-bind(?sitcom)*> by *hash(var-bind(?sitcom))* from each result, leaving other results at their location.
  - Ship <*var-bind(?friend), :actedIn, var-bind(?sitcom), :location, :NYC*> by *hash(var-bind(?friend))* from each result, leaving other results at their location.
- Which of these would be cheaper?
  - How do you decide?
    - Hint: recall our join result size estimation!

Alternate way (bushy)



Alternate way (bushy)