Collections of lists

Michael Butler

28 April 2020

Collections of lists

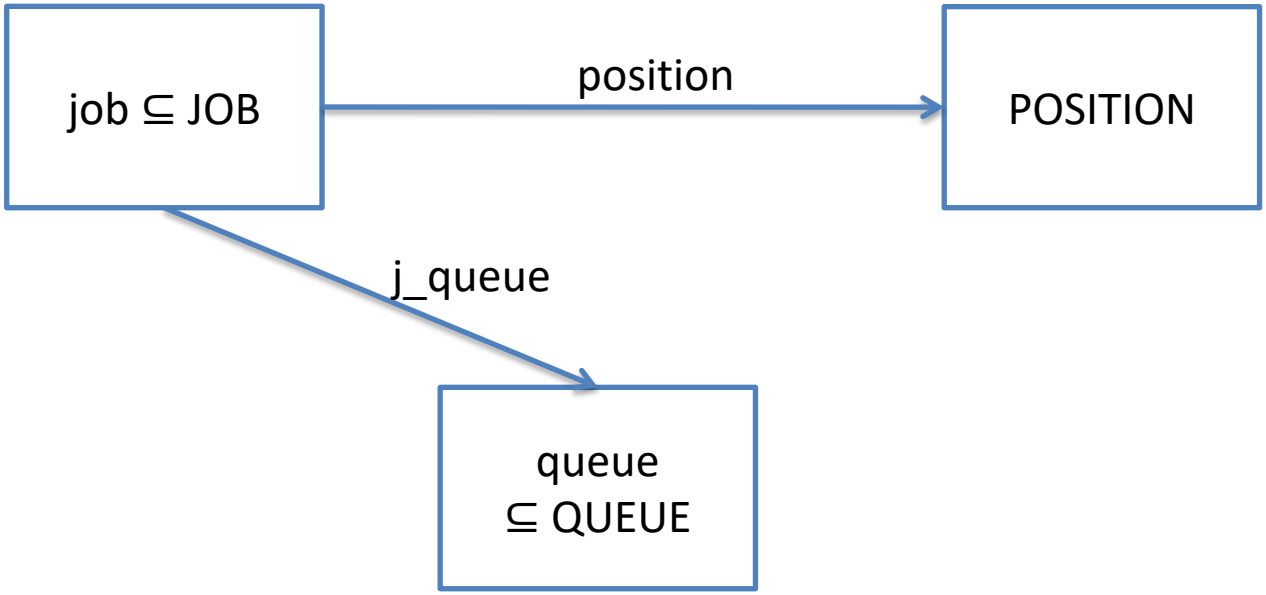
- Already seen:
  - Set: unordered collection
  - List: ordered collection

- In this lesson we look at modelling collections of lists
  - E.g., collection of printer queues, collection of quizes

Managing multiple queues

- Rather than managing a single print queue, we want to model a system that manages a collection of queues
- Introduce carrier set QUEUE to distinguish queues
- Associate each job with a queue as well as a position

Collection of queues



@inv j_queue ∈ job → queue

Ordering within a queue

- For single queue we had injectivity:

  @inv position ∈ job ⤔ POSITION

- This is too strong as ordering is only required within each queue

- Reformulation ordering invariant:

  @inv ∀j, k · j∈job ∧ k∈job∧

  j ≠ k ∧

  j_queue(j) = j_queue (k)

  ⇒ position(j) ≠ position(k)

Two *different* jobs on the same queue cannot have the same position

Adding a job to a queue

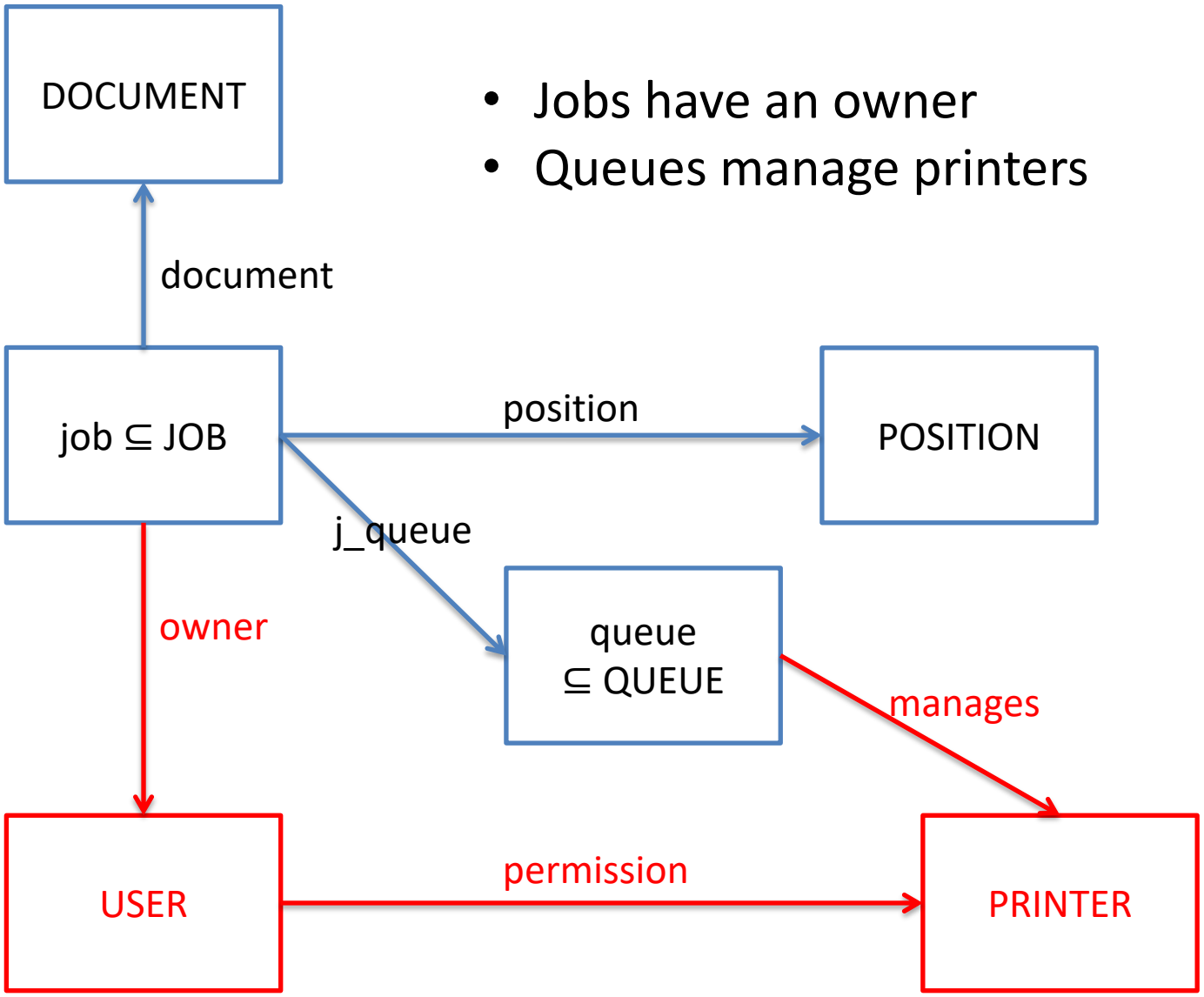
event QueueJob
any j d p q
where
@grd1 $j \in JOB \setminus job$
@grd2 $d \in DOCUMENT$
@grd3 $q \in queue$
@grd4 $p \in POSITION$
@grd5 $\forall k \cdot k \in job \wedge j\_queue(k)=q \Rightarrow p > position(k)$
then
@act1 $job \coloneqq job \cup \{ j \}$
@act2 $document(j) \coloneqq d$
@act3 $position(j) \coloneqq p$
@act4 $j\_queue(j) \coloneqq q$
end

Remove a job from a queue

event FifoRemoveJob
  any j, q
  where
    @grd1 $j \in job$
    @grd2 $q = j\_queue(j)$
    @grd3 $\forall k \cdot k \in job \land j\_queue(k)=q \Rightarrow position(j) \leq position(k)$
  then
    @act1 $job := job \setminus \{ j \}$
    @act2 $document := \{ j \} \lhd document$
    @act3 $position := \{ j \} \lhd position$
    @act4 $j\_queue := \{ j \} \lhd j\_queue$
  end

Extensions:

- Jobs have an owner
- Queues manage printers



Recap

- Collection of queues modelled by introducing explicit queue identifier
  - position is injective within each queue

Pattern for collections of lists

- Pattern for collections of lists:
  - Elements are associated to a group
  - Element order is injective within each group
- Queuing system:
  - Jobs are associated to a queue
  - Job position is injective within each queue
- Quiz system:
  - Questions are associated to a quiz
  - Question number is injective within each quiz
- Auction system:
  - Bids are associated to an auction
  - Bid value is injective within each auction