

IS 1
IS 2
IS 3
IS 4
IS 5
IS 6
IS 7

Interactive / short / 1

query	Interactive / short / 1				
title	Profile of a person				
pattern	<pre>graph LR Person["person: Person id = \$personId firstName lastName birthday locationIP browserUsed gender createDate"] -- isLocatedIn --> City["city: City id"]</pre>				
desc.	Given a start Person, retrieve their first name, last name, birthday, IP address, browser, and city of residence.				
params	1	personId	ID		
result	1	person.firstName	String	R	
	2	person.lastName	String	R	
	3	person.birthday	Date	R	
	4	person.locationIP	String	R	
	5	person.browserUsed	String	R	
	6	city.id	ID	R	
	7	person.gender	String	R	
	8	person.createDate	DateTime	R	

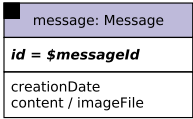
Interactive / short / 2

IS 1	query	Interactive / short / 2			
IS 2	title	Recent messages of a person			
IS 3	pattern	 <pre> graph TD person[person: Person] -- hasCreator --> message[message: Message] message -- replyOf*0.. --> post[post: Post] post -- hasCreator --> originalPoster[originalPoster: Person] </pre> <p>The diagram illustrates the query pattern. It starts with a person: Person entity (orange box) with a variable <code>id = \$personId</code>. This person is the creator of a message: Message entity (purple box). The message entity has attributes <code>id</code>, <code>content / imageFile</code>, and <code>creationDate</code>. This message is a reply to a post: Post entity (red box) via the <code>replyOf*0..</code> relationship. The post entity has an <code>id</code> attribute. The post is created by an originalPoster: Person entity (orange box) with attributes <code>id</code>, <code>firstName</code>, and <code>lastName</code>.</p>			
IS 4					
IS 5					
IS 6					
IS 7					
	desc.	<p>Given a start Person, retrieve the last 10 Messages created by that user. For each Message, return that Message, the original Post in its conversation (post), and the author of that Post (originalPoster). If any of the Messages is a Post, then the original Post (post) will be the same Message, i.e. that Message will appear twice in that result.</p>			
	params	1	personId	ID	
	result	1	message.id	ID	R
		2	message.content or message.imageFile (for photos)	Text	R
		3	message.creationDate	DateTime	R
		4	post.id	ID	R
		5	originalPoster.id	ID	R
		6	originalPoster.firstName	String	R
		7	originalPoster.lastName	String	R
	sort	1	message.creationDate	↓	
		2	message.id	↓	

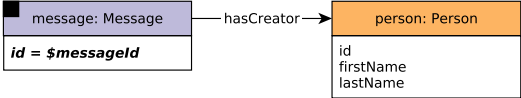
Interactive / short / 3

IS 1	query	Interactive / short / 3			
IS 2	title	Friends of a person			
IS 3	pattern				
IS 4					
IS 5					
IS 6					
IS 7	desc.	Given a start Person, retrieve all of their friends, and the date at which they became friends.			
	params	1	personId	ID	
	result	1	friend.id	ID	R
		2	friend.firstName	String	R
		3	friend.lastName	String	R
		4	knows.creationDate	DateTime	R
	sort	1	knows.creationDate	↓	
		2	friend.id	↑	

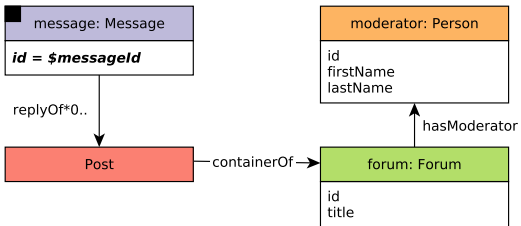
Interactive / short / 4

IS 1	query	Interactive / short / 4			
IS 2	title	Content of a message			
IS 3	pattern				
IS 4					
IS 5					
IS 6					
IS 7	desc.	Given a Message, retrieve its content and creation date.			
	params	1	messageId	ID	
	result	1	message.creationDate	DateTime	R
		2	message.content or message.imageFile (for photos)	Text	R

Interactive / short / 5

IS 1	query	Interactive / short / 5			
IS 2	title	Creator of a message			
IS 3	pattern	 <pre> graph LR message["message: Message
id = \$messageId"] -- hasCreator --> person["person: Person
id
firstName
lastName"] </pre>			
IS 4					
IS 5					
IS 6					
IS 7	desc.	Given a Message, retrieve its author.			
	params	1	messageId	ID	
	result	1	person.id	ID	R
		2	person.firstName	String	R
		3	person.lastName	String	R

Interactive / short / 6

IS 1	query	Interactive / short / 6			
IS 2	title	Forum of a message			
IS 3	pattern	 <pre> graph TD message["message: Message
id = \$messageId"] -- replyOf*0.. --> Post["Post"] Post -- containerOf --> forum["forum: Forum
id
title"] forum -- hasModerator --> moderator["moderator: Person
id
firstName
lastName"] </pre>			
IS 4					
IS 5					
IS 6					
IS 7	desc.	Given a Message, retrieve the Forum that contains it and the Person that moderates that Forum. Since Comments are not directly contained in Forums, for Comments, return the Forum containing the original Post in the thread which the Comment is replying to.			
	params	1	messageId	ID	
	result	1	forum.id	ID	R
		2	forum.title	Long String	R
		3	moderator.id	ID	R
		4	moderator.firstName	String	R
		5	moderator.lastName	String	R

Interactive / short / 7

IS 1
IS 2
IS 3
IS 4
IS 5
IS 6
IS 7

query	Interactive / short / 7			
title	Replies of a message			
pattern	<pre> graph TD M["message: Message id = \$messageId"] C["comment: Comment id content creationDate"] MA["messageAuthor: Person"] RA["replyAuthor: Person id firstName lastName"] M -- hasCreator --> MA C -- hasCreator --> RA C -- replyOf --> M MA -.-> «opt» knows RA </pre>			
desc.	<p>Given a Message, retrieve the (1-hop) Comments that reply to it.</p> <p>In addition, return a boolean flag <code>knows</code> indicating if the author of the reply (<code>replyAuthor</code>) knows the author of the original message (<code>messageAuthor</code>). If author is same as original author, return <code>False</code> for <code>knows</code> flag.</p>			
params	1	messageId	ID	
result	1	comment.id	ID	R
	2	comment.content	Text	R
	3	comment.creationDate	DateTime	R
	4	replyAuthor.id	ID	R
	5	replyAuthor.firstName	String	R
	6	replyAuthor.lastName	String	R
	7	knows	Boolean	C True if the <code>knows</code> edge exists between the <code>replyAuthor</code> and the <code>messageAuthor</code> nodes, False otherwise (including the case when the two nodes are the same)
sort	1	comment.creationDate	↓	
	2	replyAuthor.id	↑	