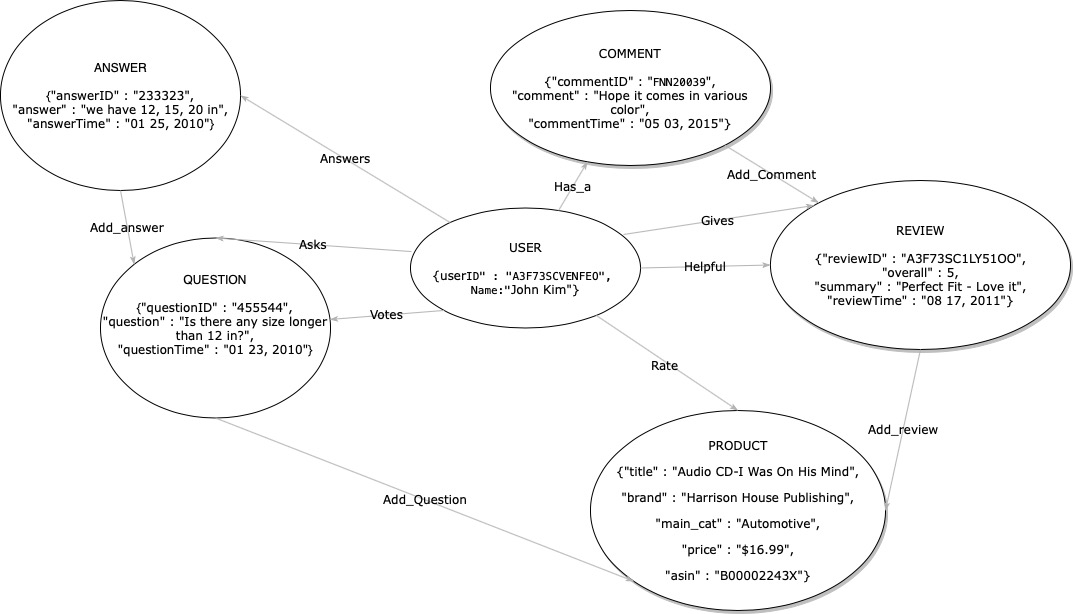
Mini Project 5

By: Anirudh Chaudhary, Quan Le, Bob LeBow, Richard More, Hao Nguyen

# Purpose 1

#### A - Requires that you make any needed changes to your data model from last week to identify the nodes (and their properties) and relationships (and their properties if needed). Report and justify your choice of data model [20pts].



Properties

|  |  |
| --- | --- |
| Nodes | Properties |
| Product | ASIN, title, brand, main\_cat, price |
| Review | reviewID, summary, reviewTime |
| Question | questionID, question, questionTime |
| Answer | answerID, answer, answerTime |
| Comment | commentID, comment, commentTime |
| User | userID, Name |

|  |  |
| --- | --- |
| Relationships | Properties |
| Add\_answer | Answer - relation - Question |
| Add\_question | Question - relation - Product |
| Add\_comment | Comment - relation - Review |
| Add\_review | Review - relation - Product |
| Votes | User - relation - Question |
| Has\_a | Commenter - relation - Comment |
| Gives | User - relation - Review |
| Rate | User - relation - Product |
| Asks | User - relation - Question |
| Answers | User - relation - Answer |
| Helpful | User - relation - Review |

Justification:

After carefully reviewing our data model based on our graph database knowledge, we decided to keep our database model the same.

We decided to keep Question and Answer in separate nodes because we can produce some query to aggregate based on Answer node such as “find the question that has the most answer”, “Find the user who provides the most answer” and “Find the average answer per question”.

We decided to keep Comment and Review in separate nodes because we can ask similar questions like separating Question and Answer.

We also kept the review in a separate node from Product as we want to aggregate Product based on the number of reviews.

Finally, as we stated in mini-project 4, we made the system center around the user as the main point, with the highest betweenness and the lowest closeness, believing that most queries would involve the user in one way or another.

#### Part B) requires that you prepare all your CREATE statements to create and populate the nodes and relationships. Report a representative sample of your CREATE statements and include any needed explanation[20pts].

CREATE commands:

*USING PERIODIC COMMIT 500*

*LOAD CSV WITH HEADERS FROM "file:///metas2.csv" AS row*

*CREATE (m:Meta {asin: row.asin, category: row.category, title: row.title, brand: row.brand, description: row.description, price: row.price})*

*USING PERIODIC COMMIT 500*

*LOAD CSV WITH HEADERS FROM "file:///reviews2.csv" AS row*

*CREATE (r:Review {asin: row.asin, category: row.category, reviewerID: row.reviewerID, unixReviewTime: row.unixReviewTime, reviewTime: row.reviewTime, overall: row.overall, reviewText: row.reviewText})*

*USING PERIODIC COMMIT 500*

*LOAD CSV WITH HEADERS FROM "file:///qa2.csv" AS row*

*CREATE (q:Qa {asin: row.asin, category: row.category, unixTime: toInteger(row.unixTime), answerType: row.answerType, questionType: row.questionType, answer: row.answer, question: row.question, answerTime: row.answerTime})*

*USING PERIODIC COMMIT 500*

*LOAD CSV WITH HEADERS FROM "file:///reviewers.csv" AS row*

*CREATE (q:Reviewer {reviewerID: row.reviewerID, reviewerName: row.reviewerName})*

*MATCH (a:Review), (b:Meta)*

*WHERE a.asin = b.asin*

*CREATE (a)-[r:REVIEWED]->(b)*

*MATCH (a:Qa), (b:Meta)*

*WHERE a.asin = b.asin*

*CREATE (a)-[r:ANSWERED]->(b)*

*MATCH (a:Reviewer), (b:Review)*

*WHERE a.reviewerID = b.reviewerID*

*CREATE (a)-[r:CREATED]->(b)*

Extra commands needed to get the reviewer dataset:

In MongoDB:

*use amazon2*

*db.createView("reviewers", "reviews", [{$group: {\_id: {reviewerID: "$reviewerID", reviewerName: "$reviewerName"}}}, {$project: {\_id: 0, reviewerID: "$\_id.reviewerID", reviewerName: "$\_id.reviewerName"}}])*

Data export command:

*mongoexport --db amazon2 --collection metas --type=csv --fields asin,category,title,brand,price,description --out /var/opt/metas.csv*

*mongoexport --db amazon2 --collection qa --type=csv --fields asin,category,unixTime,answerType,questionType,answerTime,answer,question --out /var/opt/qa.csv*

*mongoexport --db amazon2 --collection reviews --type=csv --fields asin,category,unixReviewTime,reviewerID,reviewTime,overall,reviewText --out /var/opt/reviews.csv*

*mongoexport --db amazon2 --collection reviewers --type=csv --fields reviewerID,reviewerName --out /var/opt/reviewers.csv*

IMPORTANT: Use commands below to clear up unnecessary backslashes:

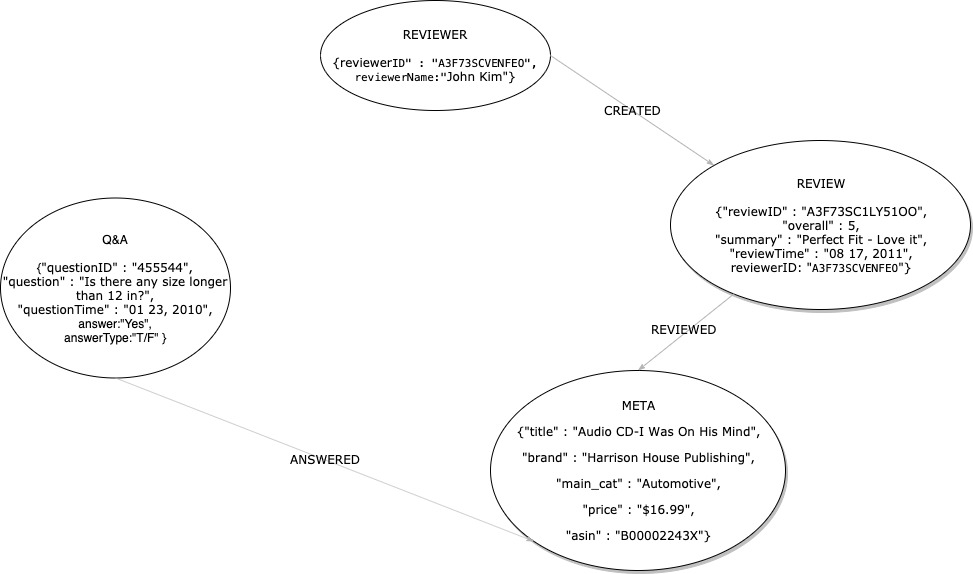
*sed -e "s/\\\//g" metas.csv > metas2.csv*

*sed -e "s/\\\//g" qa.csv > qa2.csv*

*sed -e "s/\\\//g" reviews.csv > reviews2.csv*

# Purpose 2

### Model



### Description:

Nodes: Meta, QA, Review, Reviewer

Relationships: REVIEWED, ANSWERED, CREATED

Nodes - Relationships: Reviewer -[REVIEWED]-> Meta, Reviewer -[CREATED]-> Review, QA -[ANSWERED]-> Meta

### Note:

The above model has been used while populating the database and while answering all the listed queries.

### Justification:

Compared to the data model presented in Purpose 1 Part A, in this data model we implemented, there are differences due to the following facts:

* We do not have Comments (label) in our data set; this way no ‘has\_a’ and ‘add\_comment’ edge were needed.
* The Questions and Answers are not separated; no separate ID for questions and separate ID for answers, so we can only have QA together; this way no ‘add\_answer’, ‘add\_question’ and ‘asks’ edge was needed
* We do not store votes for QAs, so no ‘Votes’ edge was needed
* The user is present only in the Review so no edge between QA and Reviewer as well as Meta/Product and Reviewer
* We do not store who gave the Helpful vote to the Review or the Ratings for the Products so no ‘Helpful’ or ‘Rating’ edge either

### A-Identify the products that a reviewer has reviewed (limit to 5 reviewers) [10pts].

MATCH (reviewer:Reviewer)-[:CREATED]->(:Review)-[:REVIEWED]->(product:Meta)

RETURN reviewer, collect(product) AS products

LIMIT 5



Summary: There are 5 Reviewers returned, out of which the first reviewer named Hartley G. Lesser has 6 products that she reviewed.

### B-Identify the reviewer with the most number of reviews[10pts].

MATCH (reviewer:Reviewer)-[created:CREATED]->()

RETURN reviewer, count(created) AS numReviews

ORDER BY numReviews DESC

LIMIT 1



Summary: Reviewer C. Hill CHF has created 181 reviews, with which he/she has the most amount of reviews.

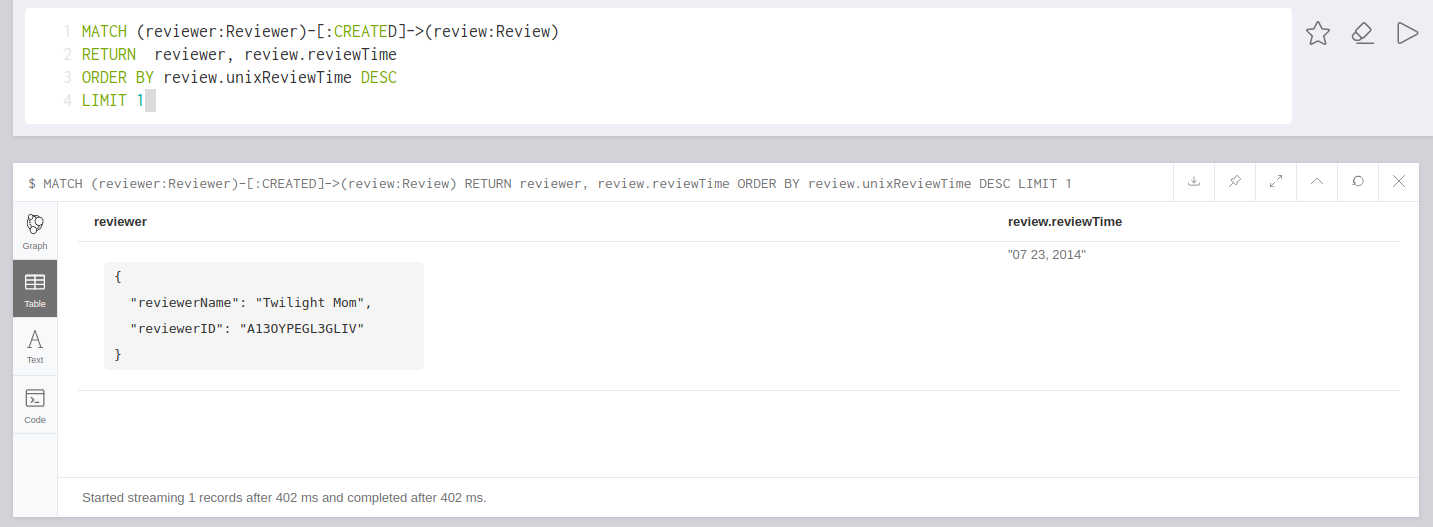
### C-Identify the reviewer with the newest review[10pts].

MATCH (reviewer:Reviewer)-[:CREATED]->(review:Review)

RETURN reviewer, review.reviewTime

ORDER BY review.unixReviewTime DESC

LIMIT 1



Summary: Reviewer Twilight Mon is the reviewer who created the newest review on 07/23/2014

### D-Identify the reviewer who has been reviewing for the longest period of time[15pts].

MATCH (reviewer:Reviewer)-[:CREATED]->(review:Review)

RETURN reviewer, max(review.unixReviewTime) - min(review.unixReviewTime) AS present

ORDER BY present DESC

LIMIT 1



Summary: S. Jentsch is the reviewer who has the longest reviewing history with 447638400 seconds between his/her oldest and newest review

### E-Identify the reviewers who reviewed a common set of products [15pts].

MATCH (reviewer:Reviewer)-[:CREATED]->(review:Review)-[:REVIEWED]->(product:Meta)

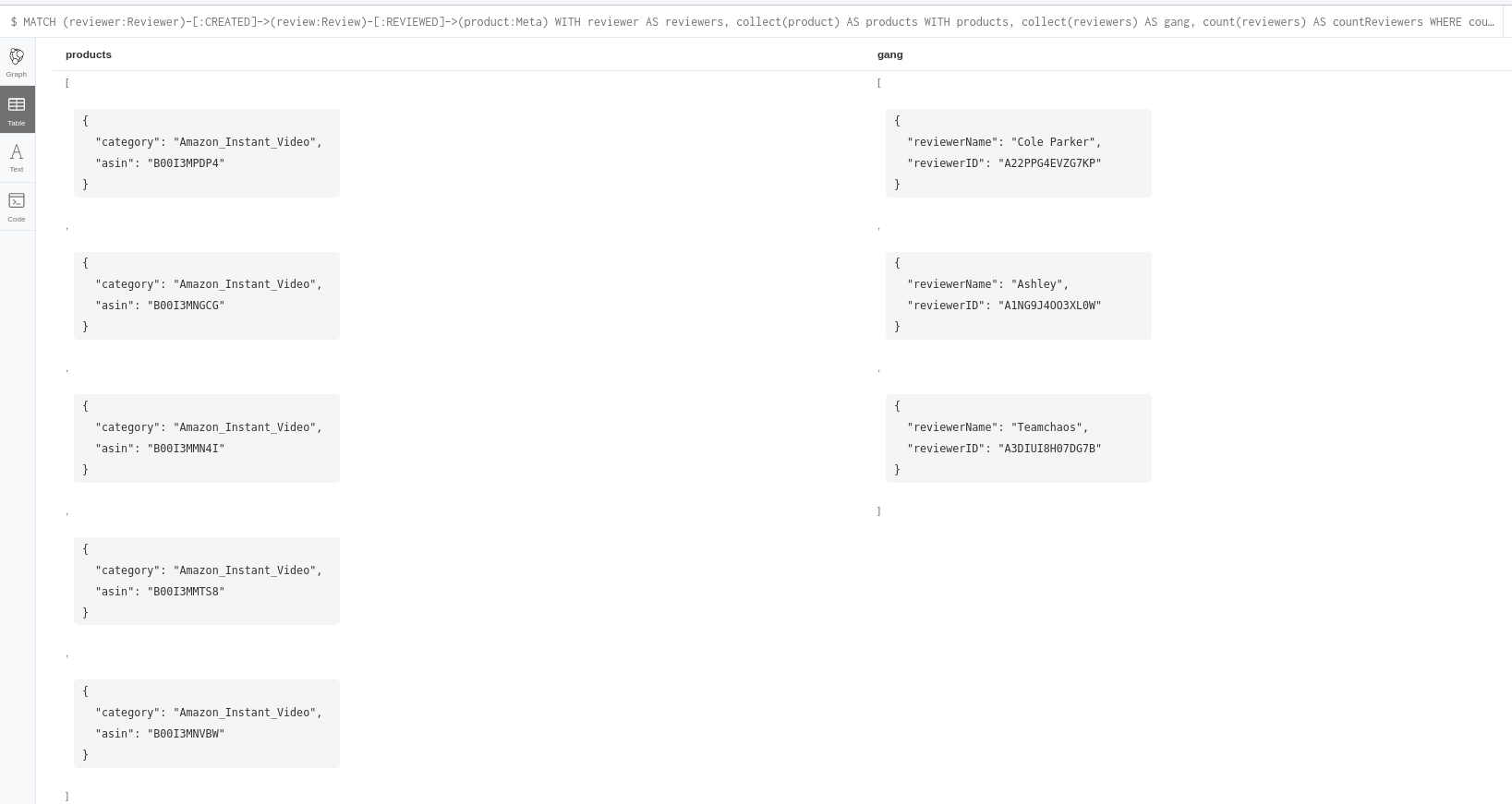
WITH reviewer AS reviewers, collect(product) AS products

WITH products, collect(reviewers) AS gang, count(reviewers) AS countReviewers

WHERE countReviewers > 2

RETURN products, gang

ORDER BY countReviewers DESC



Summary: There are two groups returned, the first has the products (ASIN) B00I3MPDP4, B00I3MNGCG, B00I3MMN4I, B00I3MMTS8, and B00I3MNVBW; and these were all reviewed by the gang of Cole Parker, Ashley, and Teamchaos.