University of Toronto
CSC343, Winter 2017

Assignment 1

Due: Wednesday 8 February at 8:00 pm sharp!

Learning Goals

By the end of this assignment you should have:

1. cleared up any gaps in your understanding of the relational model,
2. become comfortable reading and understanding a new schema,
3. mastered the individual techniques for writing relational algebra queries and integrity constraints that we learned in class,
4. practised applying them to complex problems, and
5. become well prepared to learn SQL.

Later in the course, you will learn about how to develop your own schema based on knowledge of the domain. Even though developing a schema is really the first step in building a database, it is a more advanced task than querying an existing database; this is why we will be learning about it and practising it later.

Schema

For this assignment, you will operate on the database for an online retailer. Here is its schema.

Relations

- Item($\underline{\text{IID}}$, type, description, manufacturer, price)
  A tuple in this relation represents an item that is for sale. *IID* is the item's identification number, *type* is the type of item it is, such as "book", *description* is a description of the item, *manufacturer* is the manufacturer of the item, and *price* is the price of the item.
- Manufacturer($\underline{\text{MID}}$, name, address, country, phone)
  A tuple in this relation represents a manufacturer of products. *MID* is the manufacturer's ID and *name* is the manufacturer's name. *address* is the street address, *country* is the country, and *phone* is the phone number of the manufacturer's head office.
- Subcategory($\underline{\text{a, b}}$)
  A tuple in this relation represents the fact that item type *a* is a subcategory of item type *b*. For example, we might record that 'denim' is a subcategory of 'apparel'. With this relation we can form one or more hierarchies of item types.
- Customer($\underline{\text{CID}}$, email, lastName, firstName, membership)
  A tuple in this relation represents a customer. *CID* is the customer ID, *email* is their email address, and *lastName* and *firstName* make up their name. Some customers buy a membership with the retailer. *membership* is the type of membership they have. Depending on membership type, a customer may have minimum requirements for their orders. (See Part 2.)

- Order(<u>OID</u>, CID, when, creditCard, number)
  A tuple in this relation represents an order by a customer. *OID* is the order ID, *CID* is the customer ID, *when* is date and time on which the order was made, *creditCard* is the name of the creditCard used for the order (such as 'MasterCard'), and *number* is the credit card number.

- LineItem(<u>OID, IID</u>, quantity)
  A tuple in this relation represents a line item that is part of a particular order. *OID* is the order ID, *IID* is the item ID, and *quantity* indicates how many of the item were ordered. **Note:** None of the queries make use of the *quantity* attribute; we only deal with the unit cost of an item.

- Review(<u>CID, IID</u>, when, rating, comment)
  A tuple in this relation represents a rating of an item by a customer. *IID* is ID of the item being reviewed, *CID* is the customer who gave the review, *when* is date and time on which the review was submitted, and *comment* is a comment that accompanied the review. You may assume that ratings can be compared using the normal comparison operators.

- Helpfulness(<u>reviewer, item, reader</u>, helpful)
  A tuple in this relation represents a reader's assessment of whether a review was helpful. *item* is ID of the item that was reviewed, *reviewer* is the customer who gave the review, *reader* is the customer who assessed the review, and *helpful* indicates whether or not the reader found the review helpful.

Integrity constraints

- Item[manufacturer] $\subseteq$ Manufacturer[MID]
- Order[CID] $\subseteq$ Customer[CID]
- LineItem[OID] $\subseteq$ Order[OID]
- LineItem[IID] $\subseteq$ Item[IID]
- Review[CID] $\subseteq$ Customer[CID]
- Review[IID] $\subseteq$ Item[IID]
- Helpfulness[reviewer, item] $\subseteq$ Review[CID, IID]
- Helpfulness[reader] $\subseteq$ Customer[CID]
- $\Pi_{helpful} Helpfulness \subseteq \{\text{“}yes\text{”}, \text{“}no\text{”}\}$
- $\Pi_{membership} Customer \subseteq \{\text{“}gold\text{”}, \text{“}silver\text{”}, \text{“}none\text{”}\}$
- $\sigma_{O1.CID=O2.CID \wedge O1.OID \neq O2.OID \wedge O1.when=O2.when}((\rho_{O1} Order \times \rho_{O2} Order)) = \emptyset$
- $\sigma_{R1.CID=R2.CID \wedge R1.IID \neq R2.IID \wedge R1.when=R2.when}((\rho_{R1} Review \times \rho_{R2} Review)) = \emptyset$

Part 1: Queries

Write the queries below in relational algebra. There are a number of variations on relational algebra, and different notations for the operations. You must use the same notation as we have used in class and on the slides. You may use the operators we have used in class ($\Pi, \sigma, \bowtie, \bowtie_{condition}, \times, \cap, \cup, -, \rho$) and assignment statements. Assume that all relations are sets (not bags), as we have done in class, and do not use any of the extended relational algebra operations from Chapter 5 of the textbook (for example, do not use the extended projection).

Some additional points to keep in mind:

- Do not make any assumptions about the data that are not enforced by the original constraints given above, including the ones written in English. Your queries should work for any database that satisfies those constraints.
- Assume that every tuple has a value for every attribute. For those of you who know some SQL, in other words, there are no null values.
- Remember that the condition on a select operation may only examine the values of the attributes in one tuple, not whole columns. In other words, to use a value (other than a literal value such as 100 or "books"), you must get that value into the tuples that your select will examine.
- The condition on a select operation can use comparison operators (such as $\leq$ and $\neq$) and boolean operators ($\vee$, $\wedge$ and $\neg$). Simple arithmetic is also okay, *e.g.*, attribute1 $\leq$ attribute2 + 5000.
- Some relations in our schema have a date attribute. You may use comparison operators on such values. You may refer to the year component of a date and time attribute d using the notation d.year.
- You are encouraged to use assignment statements to define intermediate results. Remember that, even before you write any algebra, you can outline your query as the LHSs for a series of well thought out assignment statements.
- The order of the columns in the result doesn't matter.
- When asked for a maximum or minimum, if there are ties, report all of them.

At least one of the queries cannot be expressed in the language that you are using. In those cases, simply write "cannot be expressed". Note: The queries are not in order according to difficulty.

1. Find the manufacturers who make an item whose type is a descendant of "apparel" in the subcategory hierarchy/ies. Report the manufacturer ID, name, address, and phone number.
2. Let's say a "singleton order" is one that includes exactly one item. Find all gold customers who have made at least one singleton order in 2016. Report their CID, and the date and time when they made their first and their last singleton order that year.
3. Suppose we consider two orders to be "identical" if they contain exactly the same items (ignoring quantity). Find all pairs of customers who have made identical orders on the same day. Report each customer's CID and OID for the order that was identical. A pair could have multiple identical orders on the same day. If so, report them all.
4. Find all customers who have a silver membership, have placed at least two orders in 2014, fewer than 2 orders in 2015, and no orders at all in 2016. Report the CID.
5. Let's say the "top cost" on any order is the cost of the most expensive item. (There could be several items tied for that top cost.) Among all the orders a customer places in a year, let's say their "skimpiest" order is the one whose top cost is the lowest. (There could be several orders tied for skimpiest.) For each customer who has ever placed an order, find their skimpiest order. If several orders for that customer are tied for skimpiest, report them all. Report the customer ID, order ID, and the order's top cost.

6. Find every order that includes at least one item for which reviewers unanimously gave it a rating of 0[1] and at least one item for which reviewers unanimously gave it a rating of 5[2]. Report the customer ID, customer's last name and first name, order ID, and when the order was placed.

7. Find all pairs of customers c_1 and c_2 such that: c_2 has reviewed at least one item, and c_1 assessed every review of c_2 as helpful.

8. For every item that has been ordered, find the last customer to order it. Report the item ID and the customer ID of the customer who ordered it last. If several customers are tied to be last to order a particular item, report a tuple for each of these customers.

9. Find all the customers who have given a review that at most one reader assessed as helpful. For each of these customers, find every review that had more "yes" (helpful) assessments than "no" assessments. Report the customer ID, item ID, and item price. (A customer will appear multiple times if they have more than one qualifying review.)

10. Find all customers who have given at least three reviews, and for whom the rating they give has always gone down over time from review to review. (This customer has grown increasingly dissatisfied, so maybe we should reach out to him or her.) Report the customer ID, last name, and email address, and the item ID for the last item they reviewed.

11. A "top-level category" is one that is not a subcategory of anything else. Find all customers who have reviewed an item in each top-level category. Report just the customer ID.

    Note: An item type that has no subcategories and no parent category — it is not connected to any of the hierarchies — is considered a top-level category. We have to look in the Item relation to find these.

12. Find the orders with at least one item, and for which every item on the order had a type that was either "book" or a direct a subcategory of "book". Report the order ID.

13. Find the orders with more than three items, and for which at least half of the items have a category that is not "book". Report the order ID, customer ID, and the credit that they used.

Part 2: Additional Integrity Constraints

Express the following integrity constraints with the notation $R = \emptyset$, where R is an expression of relational algebra. You are welcome to define intermediate results with assignment and then use them in an integrity constraint.

1. A customer who reviews an item must have ordered that item.

2. Orders made by gold members have no limit on the items that can be included. However, orders made by silver members must include at least one item costing over \$50, and orders made by non-members cannot include any items costing under \$50.

When writing your queries for Part 1, don't assume that these additional integrity constraints hold.

[1] An item must have been reviewed at least once in order to pass this condition.
[2] Ditto!

Style and formatting requirements

In order to make your algebra more readable, and to minimize errors, we are including these style and formatting requirements:

- In your assignment statements, you must include names for all attributes in the intermediate relation you are defining. For example, write

$$HighestGrade(sID, oID, grade) := \quad \ldots$$

- Use meaningful names for intermediate relations and attributes, just as you would in a program.
- Above each assignment statement, you must include a comment explaining what tuples qualify to be in the relation. Make your comments stand out from the algebra, for example by using a different font. For example, this looks reasonable:

  – Students who had a very high grade in any offering of a csc course.
  $High(sID) := \Pi_{sID}\sigma_{dept='csc' \wedge grade>95}(Took \bowtie \mathit{Offering})$

A portion of your mark will be for good style and formatting.

Submission instructions

Your assignment must be typed; handwritten assignments will not be marked. You may use any word-processing software you like. Many academics use LaTeX. It produces beautifully typeset text and handles mathematical notation well. If you would like to learn LaTeX, there are helpful resources online. Whatever you choose to use, you need to produce a final document in pdf format.

You must declare your team (whether it is a team of one or two students) and hand in your work electronically using the MarkUs online system. Instructions for doing so are posted on the Assignments page of the course website. Well before the due date, you should declare your team and try submitting with MarkUs. You can submit an empty file as a placeholder, and then submit a new version of the file later (before the deadline, of course); look in the "Replace" column.

For this assignment, hand in just one file: A1.pdf. If you are working in a pair, only one of you should hand it in.

Check that you have submitted the correct version of your file by downloading it from MarkUs; new files will not be accepted after the due date.