CSC 4710/6710 - Midterm Project Description

- *Due date: March 26, 2021 – 11:59 PM*
- *Please upload all your deliverables to the designated iCollege folder.*

In the midterm project, you are assigned to implement a minimal relational model emulator using Java programming language that can provide basic relational model and algebra functionalities in-memory. This means that your emulator will perform the operations in the run time and do not have to actually store the data[1]. A successfully implemented emulator will be able to create relational model elements such as relations, tuples, and attributes, check the validity/constraints of the relation descriptions and relation instances, enforce those constraints, enable data modification and certain retrieval operations outlined below.

You will be given a base Relation implementation (See Relation.java, Tuple.java, Attribute.java along with Driver.java for a simple example) and you are required to modify them and enhance their functionalities when implementing the features in this project. You are expected to use standard Java libraries and not utilize additional libraries that implement data frames or SQL connectors/emulators. The purpose of this project is to gain understanding on how the relational model features can be implemented; it is not utilizing third party libraries in Java.

Below you will find project requirements specifics on data definition, manipulation, and querying features along with the demo case description. There is also a grading rubric in the end of this assignment description document that you can use as a checklist.

Data Definition Specifics

A relation is a set of tuples (rows). Each relation has a set of attributes (columns). All tuples in a relation have as many values as there are attributes in that relation. The domain of the attributes in this minimal implementation will be defined merely by data types. The allowed data types are Integer and String. The relations must conform to the implicit and explicit constraints of the relational model. Namely,

- A relation does not have duplicate tuples
- A relation is a set of attributes and a set of tuples
- The values in a relation are atomic
- The values of a particular attribute in a tuple comes from the domain of the attribute
- Every relation has a key attribute (for simplicity, assume that you will not have composite primary keys)
- The key of a relation cannot be null.
- Relations implement referential integrity constraints; i.e., the value in the foreign key column (or columns) FK of the referencing relation R1 can be either: (1) a value of an existing primary key value of a corresponding primary key PK in the referenced relation R2, or (2) null.
- A database schema is a set of relations and constraints on those relations.

[1] Although it is not required in the scope of this project, you can create reader and writer modules for interacting with the file system for persistence.

Your implementation is also required to include a relational schema class. Relational schema is a set of relations and a set of constraints defining the interrelationships between the relations in the schema.

Data Manipulation Specifics

There are three specific data manipulation operations that need to be covered in your implementation. For each relation the emulator is required to be capable of inserting new tuples, removing existing tuples and updating values in one or more tuples. You will implement

- Insert operation: this operation will insert a new tuple and check if constraints are violated.
- Delete operation: this operation will delete one or more tuples based on a condition (a Boolean). The boolean conditions that you can use are limited to 'equals', 'less than' or 'greater than'. The operands for boolean conditions can be either relation attributes or literals/values (a String or an Integer).
- Update operation: this operation modifies one attribute's value on a tuple. The updated tuples will be determined based on a condition similar to the delete operation. The boolean conditions that you can use are limited to 'equals', 'less than' or 'greater than'. The operands for boolean conditions can be either relation attributes or literals/values (a String or an Integer).

After each modification, you are required to check the integrity constraints. Namely,

- After insert operations, you are required to check if domain, key, referential integrity, and entity integrity constraints are violated for the relational schema.
- After update operations, you are again required to check if domain, key, referential integrity, and entity integrity constraints are violated.
- After delete operations, you are required to check if referential integrity constraints are violated

You will use default triggered actions for each different type of violation and operation. Below are those triggered actions:

- If an insert operation violates any of the constraints, the emulator is required to reject it. This means any insert that violates the constraints will not be executed.
- If an update operation violates domain and entity integrity constraints, it needs to be rejected (similar to insert). If it violates referential integrity constraints, the update should be cascaded to each attribute that references the modified value. This means if a referenced primary key value is changed, the referencing attributes who have the same value will also be updated.
- If a delete operation violates referential integrity constraints, it should be remedied by setting the values to null. This means that if the deleted tuples primary key attribute is referenced by another attribute, the referencing attribute's value needs to be set to null.

Querying Specifics

Your emulator implementation will support querying functionalities which are based on simple relational algebra operations. These include unary, set-based binary, join-related binary, and aggregate operations. Note that all of the relational algebra operations are closed. This means that the operations will return another relation object as a result. In the following subsections, you will find more information on each type of query operations.

Unary Operations

You are expected to implement two unary operations which are:

- Projection (vertical partitioning based on a set of attributes): This operation will take a list of attributes as input. The attributes must be a subset of relation or the query will return an error.
- Selection (condition based tuple selection from a relation): This operation returns a subset of tuples based on a boolean condition. The boolean conditions that you can use are limited to 'equals', 'less than' or 'greater than'. The operands for boolean conditions can be either relation attributes or literals/values (a String or an Integer literal).

Set-based Binary Operations

You are also expected to implement three set-based binary querying operations. Those are

- Union,
- Intersection, and
- Set difference.

These are relatively simpler operations to implement. For the set-based binary operations, you are required to make sure the operands are type-compatible.

Join-Related Binary Operations

For join-related operations, you will be implementing three binary operations which are:

- Cross product
- Equijoin
- Natural join

For natural join, you are expected to determine the attributes with the same name and domain to perform the join operation and drop the duplicate attributes in the resulting relation. If no attributes are shared, or if the attribute with the same name has a different domain, then the operation needs to be rejected. In the case of equijoin and cross product operations, use the name of the relation as a suffix to one of the column names if column names are repeated. For the equijoin, take the names of the attributes (one from each relation – which are required to be compared for performing the join operations) as the parameter.

Aggregate Operations

Lastly, you are expected to implement five aggregate functions. Those are

- Min
- Max
- Average (rounded)
- Sum
- Count

The result of simple aggregate function queries are still a relation, with one value (one attribute and one tuple). The attribute in the resulting relations should be named using the following format: [function_name]_[attribute_name]. Aggregate operations will take one relation, one (numerical) attribute, and one function as an attribute. If the column is not numerical, the operation will be rejected.

In addition to the aggregate operations, you are also required to implement a group-aggregate operation. In the group-aggregate operations, the result-set will be grouped by exactly one column and the aggregate operations will be applied to again one numerical column. The group-aggregate operation also needs to support these five above mentioned aggregate functions. The result of a group-aggregate operation is also a relation with exactly two attributes, where one is the grouping attribute and the other is the attribute the aggregate function is applied to. Please follow the same naming convention for the aggregated attribute; i.e., [function_name]_[attribute_name].

Demo Case and Report

In the second week of your project, you will be given a demo case (a set of text files as relations) and constraints. You will also be provided with queries. You will be asked to create a relational schema using the relations in this demo case, define the constraints and insert the tuples to those relations. Then, using the populated relations and implemented querying capabilities, you will implement the provided queries in your emulator. The report will include short descriptions of how you implemented the given relational schema and demonstrate the steps you have taken.

Grading Rubric

Category	Item/Feature	Percentage
Data Definition	Implement relational schema (set of relations and constraints)	25%
	Check tuple duplicates	
	Check domain constraints for an attribute on a tuple	
	Check entity integrity and primary key constraints (every relation has a P.K. and P.K. is unique and not null)	
	Implement referential integrity constraints	
	Check referential integrity constraints in a relational schema (referencing attribute exists in referred or it is null)	
Data Manipulation	Implement insert	25%
	Implement update	
	Implement delete	
	Check if modifications cause violations	
	Implement triggered actions (cascade on update, set null on delete	
Querying	Unary Operations	10% [5 each]
	Set-based Binary operations (with type compatibility)	9% [3 each]
	Join-related binary operations	15% [5 each]
	Aggregate operations	10% [2 each]
	Group aggregate operations	6%
Demo	Create report with demo case	20%