ITWS-4250/6250 Database Applications and Systems

Lab 1

Key:

Pi – Projection

Si – selection

X = natural join

This lab will focus on the use of relational algebra to query a relational schema, as well as concepts related to functional dependencies, normalization, and the relational model.

You will have roughly 90 minutes of time during the scheduled lecture period to complete it, but it’s not due on LMS until **Eod tonight**.

You may consult with your groupmates, but you must do and submit your own work.

1. Assume the existence of a database with the following relations:

*Ingredient*(*name, calories, cost, containsNuts*)

*Recipe*(*name, ingredientName, amount*)

*Menu*(*recipeName, season*)

which is used by a restaurant to manage its menu items. Assume that all amounts are in the same unit (e.g., grams) and that:

*πingredientNameRecipe ⊆ πnameIngredient πrecipeNameMenu ⊆ πnameRecipe*

Two facts that may be helpful to remember:

* + the natural join (|X| with no condition) will pair tuples that agree on attributes with the *same name*, regardless of the semantic meaning of that name.
  + recipes that contain nuts may also contain some ingredients without nuts

Write relational algebra expressions for the following:

1. (3 points) List the names of all recipes on the menu for either the Summer or Fall seasons

Pi recipeName ( Si menu.season = Summer OR menu.season = Fall (menu))

1. (3 points) List the names of the recipes on the Fall menu that contain nuts

Pi menu.recipeName ( (Si menu.season = Fall (Menu)) X menu.recipeName = recipe.name(recipe) X ingredient.name=recipe.ingredientName(ingredient) (Si ingredient.containsNuts = True (Ingredient)) )

1. (4 points) List the names of the recipes on the Winter menu that *do not* contain nuts

Pi menu.recipeName ( (Si menu.season = Winter (Menu)) X menu.recipeName = recipe.name (Recipe) X ingredient.name=recipe.ingredientName(ingredient) (Si ingredient.containsNuts = False(Ingredient)) )

1. (3 points) List the names of recipes that contain more than 4 units of salt

Pi name(Si ingredientName= salt and amount > 4 (recipe))

1. Assume the relation *R*(*a, b, c, d, e, f* ) with the following FDs:

*ab → c c → de f → ac be → f*

1. (4 points) Find the key(s) of *R*

*The key for the above mentioned relation will be {a,b,c,d,e,f}. no other subset of the attributes can be used to get all the attributes.*

1. (1 point) Why isn’t *{f, b, e}* a key?

{f,b,e} cannot be a key because there is a FD be -> f which makes part of {f,b,e} a trivial FD.

1. (2 points) Compute *{ac}*+. Show your work.

{ac} +=>

c gives de from c -> de which makes {ac}+ => {acde}

There are no other attributed that can be derived from the FDs given.

1. (2 points) Comput *{b, d}*+. Show your work

{bd} +=> {bd}

There are no other attributed that can be derived from the FDs given.

1. (3 points) Would *ac → d* form part of a minimal basis? Why or why not?

ac -> d is not a minimal basis because from the given FDs no further attributes can be derived for example c cannot derive d.

1. Answer the following questions:
   * (a) (2 points) What are the reasons behind normalization of relations in a database.

The reason behind normalizing relations in database is to reduce data redundancy and avoid anomalies like update anomaly, deletion anomaly.

* + (b) (3 points) What is the difference between BCNM and 3NF and why do we choose one over the other?

BCNF is stricter than 3NF, BCNF is stronger than 3NF. For a relation to be in 3NF it should satisfy 1NF and 2NF and a relation to be in BCNF it should satisfy 1NF, 2NF and 3NF. 3NF preserves all dependencies whereas in BCNF dependencies may or may not be preserved. In 3NF lossless decomposition can be achieved and in BCNF lossless decomposition is hard to achieve.

3NF is chosen over BCNF when we want the dependencies of the relation to be preserved. BCNF is chosen over 3NF when we want no anomalies to be guaranteed.

1. Answer the following questions:
   * (a) (2 points) What are the characteristics of a transaction in the context of a relational database management systems?

The characteristics of a transaction are defined by ACID properties which stands for atomicity, consistency, isolation, and durability. Atomicity means that a transaction should happen in full or not at all. Consistency means that the data should be good meaning that it should be according to the constraints defined in the schema. Isolation means that the transactions should happened one at a time, same data cannot be modified by multiple transactions at the same time. Durability stands for fault tolerance, meaning that the database should preserve the data in case of power failure etc.

* + (b) (2 points) What are the types of relations in SQL and what is a basic difference between them?

There are mainly three types of relationships in a database namely, one-to-one, one-to-many, and many-to-many. In one-to-one relationship each row in table 1 will have a corresponding row in table 2 for example husband and wife. In one-to-many relationship one record in table 1 is related to multiple records in table 2 for example teacher and student. In many to many relationship multiple records in one table correspond to multiple records in another table for example product and sales.

* + (c) (1 point) How is a key declared when multiple attributes are required to uniquely identify an instance in a relation?

The following SQL code displays how to declare primary key with multiple attributes:

CREATE TABLE table1 (  
id INT,  
name CHAR(IO),

**primary key (id, name)**

)